Control-Oriented Neural State-Space Models for State-Feedback Linearization and Pole Placement

Alexandre Hache, Maxime Thieffry, Mohamed Yagoubi and Philippe Chevrel

***Abstract*— Starting from a data set consisting of input-output measurements of a dynamical process, this paper presents a training procedure for a specifically control-oriented model. The considered dynamic model adopts a particular neural state-space representation: its structure guarantees its linearizability by state feedback. Moreover, the linearizing control law follows trivially from the parameters of the learned model. The method relies on a parameterized continuous-time neural state-space model whose structure is inspired from well-known exact linearization. The feasibility and efficiency of the approach is illustrated on a nonlinear identification benchmark, namely the Silverbox one. The quality of learning and linearizing feature of the control design are validated on two nonlinear models by comparing the input-output behavior of each closed-loop and its best linear approximation.**

I. INTRODUCTION

In control community it is generally accepted that the design of linear systems control can be done in a more systematic way than in the nonlinear case. In this framework, feedback linearization techniques have proven to be an appealling challenge. The latter are positioned, in particular, in a framework where the nonlinear system can be transformed exactly into a linear system proceeding by state feedback and coordinate change. The dynamic requirements on the controlled system are then expressed on the resulting linear model.

Feedback linearization theory usually considers input affine nonlinear models of the form:

$$P : \begin{cases} \dot{x} = f(x) + g(x)u \\ y = h(x) \end{cases} \tag{1}$$

where $u \in \mathbb{R}^{n_u}$ is the system input, $y \in \mathbb{R}^{n_y}$ the output, $x \in \mathbb{R}^{n_x}$ is the state and where functions f, g and h are nonlinear mappings.

Founding works on exact linearization problems for this class of systems were published in the late 1980s [1] and necessary and sufficient conditions for an exact linearization of input-affine systems have been established. A lock, however, consists in working from experimental input-output data, in the absence of a relevant and accurate physical model.

During the last decades, neural networks have aroused the interest of the scientific community due to their universal approximation properties [2] and the development of dedicated and powerful optimization algorithms [3]. In nonlinear identification, they have proven their ability to provide accurate dynamic models and the recent rise of physics-informed neural networks [4], [5] makes them attractive for modeling nonlinear dynamical systems.

Using machine learning techniques and especially neural networks to learn linearizing controllers is not new, see e.g. [6] or [7] for a recent review. However, recent works take advantage of the development of computational resources to tackle the problem of learning linearizing controllers from data, using reinforcement learning in [8] or Gaussian processes to improve robustness of feedback linearization in [9].

Furthermore, recent advances in nonlinear system identification of neural (state-space) models are numerous and open the way to promising simulation and analysis approaches. A neural state-space model (NSSM) is a model where the functions f, g and h from (1) are approximated using neural networks. This kind of models has first been introduced in discrete time in [10] and recent works propose to add an integration scheme on top of the standard forward pass to approximate ODEs [11]. Both the state and the output equations of (1) can be approximated by a one-hidden layer feedforward neural network:

$$\begin{aligned} f(x) &= W_f \sigma(W_{fx} x + b_{fx}) + b_f \\ g(x) &= W_g \sigma(W_{gx} x + b_{gx}) + b_g \\ h(x) &= W_h \sigma(W_{hx} x + b_{hx}) + b_h \end{aligned} \tag{2}$$

where W_i and b_i are weight matrices and biases respectively and $\sigma(\cdot)$ an appropriate nonlinear activation function, usually sigmoid or hyperbolic tangent.

It has been shown that the inclusion of an explicit linear part in the nonlinear model (2) improves the learning stage by providing a good initialization, see e.g. [12]. Thus, recent works introduce an explicit linear part in NSSM via the matrices A, B, C, D in such a way:

$$\begin{aligned} \dot{x} &= Ax + Bu + f(x) + g(x)u \\ y &= Cx + Du + h(x) \end{aligned} \tag{3}$$

The purpose of this article is to proceed by constrained learning of neural state-space models, which are by design feedback linearizable and where the control law and the targeted linear dynamics appear explicitly in the structure of the network.

II. RELATED WORK AND STATE OF THE ART

This section provides a reminder of the so-called exact and approximate feedback linearization techniques.

Authors are with IMT Atlantique, LS2N, UMR CNRS 6004, Nantes, France. Contact : alexandre.hache@imt-atlantique.fr
This work was supported by ANR project AI@IMT - Responsible AI for Industry and Society (ANR-20-THIA-0019).

A. Exact feedback linearization

Considering an input-affine nonlinear system of the form (1), the objective of exact feedback linearization is to find a control law of the form:

$$u = \alpha(x) + \beta(x)v \tag{4}$$

where $\alpha(x)$, $\beta(x)$ are nonlinear functions of the state, and a diffeomorphism $z = \Phi(x)$ such that the closed-loop system according to (1) and (4) is an input-output linear one, from v to z and y:

$$\begin{aligned} \dot{z} &= Az + Bv \\ y &= Cz \end{aligned} \tag{5}$$

Several conditions on the state equation (1) must be fulfilled in order to perform exact linearization. In the case where these conditions (which will not be recalled, we refer the reader to [1] for more details) are satisfied, there exists an output function $\lambda(x)$, called flat output, such that:

$$\begin{aligned} \dot{x} &= f(x) + g(x)u \\ \tilde{y} &= \lambda(x) \end{aligned} \tag{6}$$

has a relative degree equals to the size of the state-space [13]. Given a well-suited $\lambda(x)$, a valid change of coordinates is then given by:

$$z = \begin{pmatrix} \lambda(x) \\ \dot{\lambda}(x) \\ \dots \\ \lambda^{(n_x-1)}(x) \end{pmatrix} = \Phi(x) \tag{7}$$

This transformation leads to a linear system which is a chain of integrators and a state-space model expressed in Brunovsky canonical form that possesses no physical meaning of the system being linearized. In addition to being ill-conditioned, it changes completely the dynamic of the system and thus may not be robust to modelling uncertainties. To cope with these limitations, one can do another change of coordinates in order to linearize the system around an equilibrium point leading to a model better suited to a robust controller design [14]. Moreover, this exact linearization method requires a precise knowledge of the plant P and the equilibrium point around which the linearization is performed.

B. Approximate feedback linearization

Approximate feedback linearization comes at hand when the system is not exactly linearizable or when a model is not available. Different techniques exist for approximate linearization depending on the objective: some of them will be discussed in this section and a comprehensive review is available in [15] for interested readers.

The classic linearization of a nonlinear system is its Taylor approximation around an equilibrium point x_0 induced by u_0:

$$\begin{aligned} \dot{x} &= Ax + Bu \\ y &= Cx + Du \end{aligned} \tag{8}$$

where:

$$\begin{aligned} A &= \left.\frac{\partial f}{\partial x}\right|_{x=x_0} & B &= \left.\frac{\partial f}{\partial u}\right|_{x=x_0} \\ C &= \left.\frac{\partial g}{\partial x}\right|_{x=x_0} & D &= \left.\frac{\partial g}{\partial u}\right|_{x=x_0} \end{aligned} \tag{9}$$

There are systems where this first order Taylor approximation fails or is only valid in a restricted region of the state space. In [16] a higher order approximation is proposed by ignoring the third order (and higher) terms in the Taylor approximation around the equilibrium point (u_0, x_0) in order to increase its domain of validity.

In [17], the concept of a pseudolinear system is introduced that is a system of the form:

$$\dot{z} = Az + Bv + q(z, v) \tag{10}$$

where the term $q(z, v)$ has desired properties and where the A, B matrices comes from Taylor approximation and are independant from the operating point. Pseudo-linearization then seeks a feedback control input such that the state equation of (1) is equivalent to (10).

Finally, in [18], an approximate linearization approach is proposed for affine nonlinear systems that do not satisfy all the conditions required for input-to-state linearization but are, instead, linearly controllable in a neighborhood of a set of operating points.

III. PRESENT CONTRIBUTIONS

Given a dataset described by input-output measurements (u^D, y^D) our objective is to identify a feedback linearizable model using neural networks. A first approach could be to enforce the conditions for exact linearization described in [13], but the computation of the iterated Lie Brackets and the rank condition that must be imposed on the resulting matrix make the implementation of this solution too complicated practically.

This paper proposes a data-driven control design solution to overcome these difficulties. Neural networks are involved, first to identify a neural state-space model of the process, and then to trawe make the assumption that there exists a diffeomorphismin a structured one with the same input-output behavior, parameterized and with the property to be feedback linearizable.

The remainder of the paper is organized as follows : Section IV presents the main results, introducing the proposed linearizable network and giving the key elements for its implementation. Then this approach is tested in section V on a popular benchmark used for nonlinear identification. Finally, open challenges and further works are discussed in section VI.

IV. MODEL IDENTIFICATION WITH LINEARIZABILITY PROPERTY

A. Preliminar results

Given a process P described by an unknown input-affine nonlinear model of the form (1) and considering that the state x is available either directly or from the use of an observer,

we make the assumption that there exists a diffeomorphism $z = \Phi(x)$ such that (1) is equivalent to the following model:

$$\tilde{P} : \begin{cases} \dot{z} = Az + B\beta(x)(u - \alpha(x)) \\ y = Cz \end{cases} \tag{11}$$

Proposition 1. The control law defined by $u = \beta(x)^{-1}v + \alpha(x)$ is a feedback linearizing control input for model $\tilde{P}$ and therefore for P. ∎

However, functions $\alpha(x)$ and $\beta(x)$ are in practice not known. We then consider the following parameterized model:

$$M : \begin{cases} \dot{z} = \hat{A}z + \hat{B}\hat{\beta}(x)(u - \hat{\alpha}(x)) \\ \hat{y} = \hat{C}z \end{cases} \tag{12}$$

where $\hat{A}, \hat{B}, \hat{C}, \hat{\alpha}(x)$ and $\hat{\beta}(x)$ are unknown matrices and functions to be identified.

Definition 1. Two models Σ_1 and Σ_2 are said similar in terms of input-output behavior if, whatever input u applied to both systems and time T, the output y_1 of Σ_1 and y_2 of Σ_2 are sufficiently close:

$$\forall\ u, \forall\ T \in \mathbb{R}^+, \int_0^T (y_1(t) - y_2(t))^2 dt < \epsilon \tag{13}$$

with ϵ sufficiently small. ∎

These notations and definitions lead to the main contribution of this paper. The following result holds:

Proposition 2. If models M and $\tilde{P}$ are similar in terms of their input-output behavior, then the control law defined as:

$$\hat{u} = \hat{\beta}(x)^{-1}v + \hat{\alpha}(x) \tag{14}$$

is a feedback-linearizing control input for model M, $\tilde{P}$ and therefore for P. ∎

B. Methodological procedure

The neural-network training procedure ensures that models M and $\tilde{P}$ are similar in terms of input-output behavior. The loss function of the training is defined as the mean squared error between the data measurements y^D and the simulated output y^{sim} of model M:

$$J = \frac{1}{T}\sum_{i=0}^{T-1}(y_i^D(t) - y_i^{sim}(t))^2 \tag{15}$$

The structure of model M makes it feedback linearizable by design and the control law (14) yields a linear closed-loop system of the form:

$$\begin{aligned} \dot{z} &= \hat{A}z + \hat{B}v \\ y &= \hat{C}z \end{aligned} \tag{16}$$

where v is the new input of the closed-loop system.

The linear structure of the corresponding closed-loop system eases the design of efficient control laws. In addition, if the matrix $\hat{A}$ has desired dynamics and is fixed during training, then the control law given by (14) is a feedback linearizing control input and the resulting linear model has desired closed-loop dynamics.

C. Implementation

1) Learning algorithm: In this paper, continuous-time neural networks are considered. In order to learn the underlying ODEs from data points, we follow the framework described in [5]: an integration scheme is added on top of the forward pass of the neural network and back-propagation is performed overall. In addition, the initial state for the hidden layer, namely z_0, is considered as a variable for optimization. A tensor Z containing the initial states of the size of the training data is created and is optimized along the way. The implementation[1] is described in algorithm 1, θ denoting the network parameters including the matrices $\hat{A}, \hat{B}, \hat{C}$ and one hidden-layer networks :

$$\begin{aligned} \alpha(x) &= W_\alpha \sigma(W_{\alpha_x} x + b_{\alpha_x}) + b_\alpha \\ \beta(x) &= W_\beta \sigma(W_{\beta_x} x + b_{\beta_x}) + b_\beta \end{aligned} \tag{17}$$

The training is implemented using PyTorch framework [19] using a Runge-Kutta 4 step method in the forward pass as integration scheme and ADAM optimization algorithm [3].

Algorithm 1: Training procedure

Inputs : (u^D, x^D, y^D) dataset, n^x state order, n^h number of hidden neurons, N_B batch size, T sequence length, N number of optimization steps; t_s integration step, α learning rate

Initialize the network :
- Determine linear initialisation for A_0, B_0, C_0 ;
- Set desired closed-loop poles p_i via state-feedback pole-placement $\hat{A} = (A_0 - B_0 K(p_i))$;
- Freeze $\hat{A}$ matrix;
- Instantiate hidden states tensor $Z = 0$;

for $i \leftarrow 0$ **to** N **do**
- Select randomly z_0 tensors among Z;
- **for** $j \leftarrow 0$ **to** $T - 1$ **do**
  - $y_j^{sim} = \hat{C}z_j$;
  - $\dot{z}_j = \hat{A}z_j + \hat{B}\hat{\beta}(x_j)(u_j - \hat{\alpha}(x_j))$;
  - $z_j^{sim} = ODEINT(t_s, \dot{z}_j)$;
- **end**
- Compute the loss :

$$J(\theta, Z) = \frac{1}{N_B T}\sum_{k=0}^{N_B-1}\sum_{j=0}^{T-1}(Y_{k,j}^D - Y_{k,j}^{sim}(\theta, Z))^2$$

- Compute gradients : $\nabla_\theta J = \frac{\partial J}{\partial \theta}$, $\nabla_Z J = \frac{\partial J}{\partial Z}$;
- Update parameters :

$$\theta \leftarrow \theta - \alpha \nabla_\theta J$$
$$Z \leftarrow Z - \alpha \nabla_Z J$$

end

Outputs: Network parameters θ

[1]Code is available upon request.

2) Network initialization: Inspired by the efficient initialization approach as in the general state-space case [20] and one-hidden-layer neural networks, we propose the following initialization. The network (12) is initialized with linear part only, i.e. before first iteration $\hat{\beta}(x) = 1$ and $\hat{\alpha}(x) = 0$. The network is thus initialized with the following linear model:

$$\begin{aligned} \dot{z} &= \hat{A}z + \hat{B}_0 v \\ \hat{y} &= \hat{C}_0 z \end{aligned} \tag{18}$$

where matrices $\hat{B}_0$ and $\hat{C}_0$ are defined using classic linear model estimation techniques such as best linear approximation (BLA, see e.g. [21]).

The matrix $\hat{A}$ is computed before training and fixed during the learning stage in such a way that the resulting closed-loop tends to be as close as possible to the dynamics imposed in the linear case through the pole-placement:

$$\hat{A} = \hat{A}_0 - \hat{B}_0 K_0 \tag{19}$$

V. SIMULATION RESULTS

The proposed control-oriented training approach is illustrated on a popular benchmark for nonlinear identification : the SilverBox benchmark[2]. It is an electrical oscillator analogous to a second-order mechanical system with nonlinear polynomial spring:

$$m\ddot{y}(t) + dy(t) + k(y(t))y(t) = u(t) \tag{20}$$

where $k(y(t)) = a + by^2$. This system is affine with respect to the control input and fits the class of systems for feedback linearization. Moreover one can see this system is feedback linearizable with $u(t) = by^3$.

The input-output signals are made of 131 072 samples and only the first 40 000 points of the dataset (the arrowhead) are given as a training set.

A. Training results

In order to validate our approach, we first identify an input-affine model (P, see equation (1)) of the plant. It will be our simulator to access the state x for the training procedure of a feedback linearizable model (M, equation (12)).

As the Silverbox example is a single-input single-output system, $\beta(x)$ is invertible if we impose $\beta(x) \neq 0$ during training. To do so, the following activation function is added at the final layer of the network β:

$$\eta(x) = ELU + 1 = \begin{cases} x + 1, & x \geq 0 \\ e^x, & x < 0 \end{cases} \tag{21}$$

where ELU stands for exponential linear unit. This choice may not be optimal and the choice of η is left to the user but it has the advantage over ReLu to have a non-zero gradients for all x.

To validate both models we simulate them and compute the mean squared simulation error on the whole dataset. The training procedure is performed[3] using algorithm 1 with parameters given in table I. The learning results are gathered in table II and the simulation errors for both models are displayed on figure 1, which confirms that both model P and M represents faithfully the Silverbox benchmark.

[2]The training data are publicly available for download, see [22] or visit `https://sites.google.com/view/nonlinear-benchmark/benchmarks/silverbox`

[3]Training is performed on a *Intel(R) Core(TM) i7-8665U CPU@1.90GHz, 4 cores* and both models are trained for 30min.

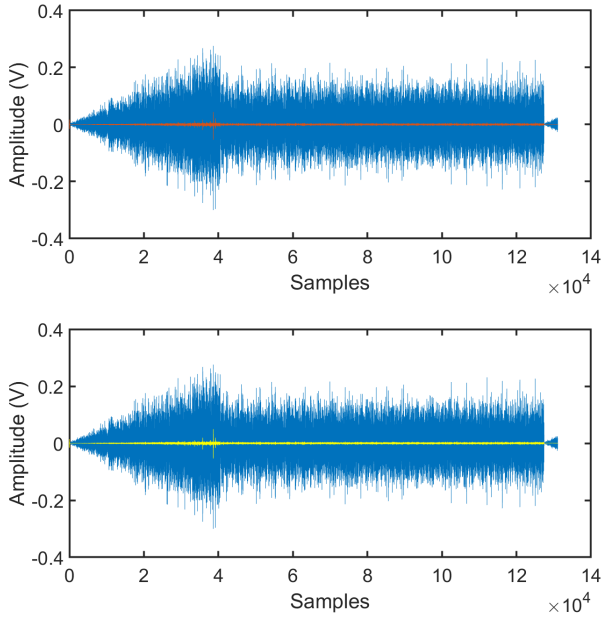

Fig. 1. Illustrations of training results. Blue : data measurements y^D. Red (top) : error between data and simulation model P. Yellow (bottom) : error between data and trained model M.

Model	N_B	N	T	t_s	α
P	512	30 000	20	$1.63e-3$	$5e-3$
M	512	20 000	30	$1.63e-3$	$1e-3$

TABLE I

HYPERPARAMETERS USED FOR TRAINING MODEL P AND M FOR THE SILVERBOX BENCHMARK.

B. Closed-loop results

The whole modelling structure for the closed-loop is depicted in figure 2. It summarizes equation (1), (12) and (14) : the dynamics of plant P from v to y in closed-loop is equivalent to the one of a linear system.

In order to estimate the capability of our approach to impose specific dynamics for the closed-loop, we set the poles of $\hat{A}$ in (12) in order to remove oscillations. From BLA estimate $\hat{A}_0$ with eigenvalue p_0, the system matrix

Model	MSE J (15)
P	$2.12\ 10^{-6}$
M	$2.20\ 10^{-6}$

TABLE II

TRAINING RESULTS FOR SILVERBOX BENCHMARK.

MEAN-SQUARED ERROR (15) BETWEEN REAL DATA AND SIMULATED OUTPUT FOR BOTH MODELS P AND M.

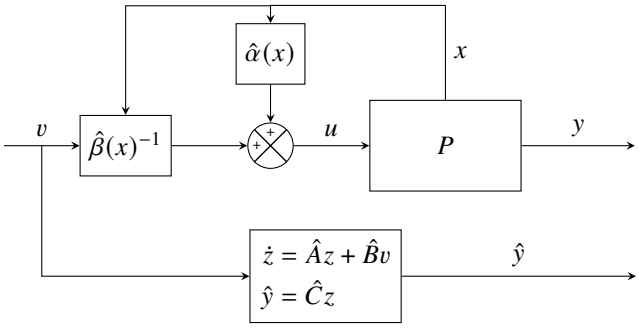

Fig. 2. Architecture for the closed-loop: the plant P in closed-loop (top) from v to y is equivalent to the linear system from v to $\hat{y}$ (bottom).

Model	Fit % Open Loop	Fit % Closed loop
Data	80.01%	/
P	83.66%	98.36 %
M	84.32%	99.99%

TABLE III

OPEN-LOOP AND CLOSED-LOOP COMPARISON OF FITTING PERCENTAGE OF A LINEAR MODEL BOTH MODELS P AND M SIMULATED WITH SAME DATA u^D.

$\hat{A}$ is computed via pole placement so that it has desired eigenvalues $-|p_0|$.

1) Linearity validation via Best Linear Approximation: After training, to validate our approach and verify that the closed-loop is linear we simulate the obtained model with the training data u^D and use BLA algorithm from MATLAB, i.e `ssest` function, to estimate how accurately a linear model can describe the input-output behavior of the closed-loop.

The fitting criterion for linearity estimation is the Fit percent:

$$Fit = 100\left(1 - \frac{\|y^{sim}(t) - y^{lin}(t)\|}{\|y^{sim}(t) - \bar{y}^{sim}(t)\|}\right) \tag{22}$$

where y^{sim}, y^{lin} are the simulated output and the output of the estimated resulting linear model respectively and where $\|\cdot\|$ denotes the 2−norm of the corresponding vector and $\bar{y}$ its mean value.

Table III presents the open-loop and closed-loop result: model M in closed-loop presents a linear fitting criterion of nearly 100% and model P in closed-loop corresponds to 98.36% to a linear model. This validates the theoretical results presented above : the control law (14) is an exact feedback linearizing control input for model M and an approximate feedback linearizing control input for model P.

2) Validation of closed-loop control design: As a second test in order to validate our approach, we compare the time-response of the closed-loop models. Figure 3 shows the comparison between the open-loop and closed-loop response to initial conditions $y(t = 0) = 0.05V$, $\dot{y}(t = 0) = 0$. The top figure shows that both models P (output in blue) and M (output in red) have same open-loop behavior and the bottom figure shows that model M is perfectly damped, in closed-loop with the linearizing control law, while oscillations of model P are highly removed. Even if the model P in

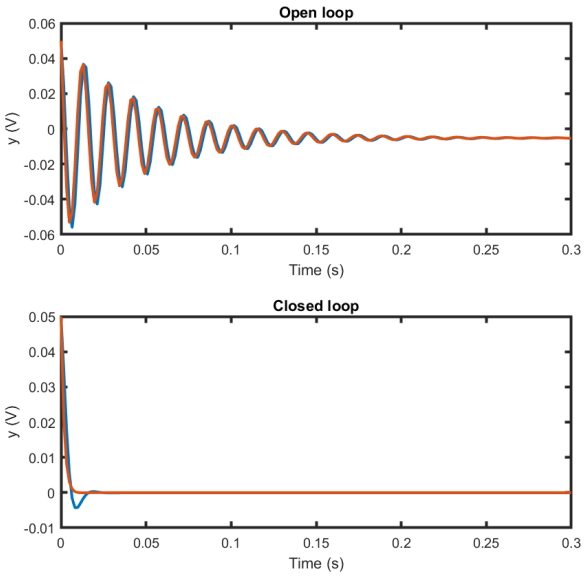

Fig. 3. Comparison of open and closed-loop behavior for model P and M.
Blue : output of model P, Red : output of model M.
Top : open-loop response, Bottom : closed-loop response.

closed-loop is not exactly the same as M, it shows that the linearizing control law designed for model M yields promising results when applied to system P. Indeed, both models approach to a seemingly linear behavior.

VI. CONCLUSION

In this paper a data-based approach has been presented to linearize a system using neural networks. During the initialisation phase, pole placement techniques are used to enforce specific dynamics to the resulting linearized closed-loop. The presented method is generic as it is suitable for any input-affine nonlinear process for which input-output data measurements are available. The linearizing control law and the resulting linear model are directly obtained from the learning stage. A nonlinear benchmark, the Silverbox one, is used as a demonstrator to show the effectiveness of the proposed control-oriented model structure and identification method. Future work will focus on the validation of the

presented control design on real-world processes and to the extension of this work to a broader class of systems.

References

[1] A. Isidori, *Nonlinear control systems: an introduction*. Springer, 1985.

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

[3] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

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

[5] M. Forgione and D. Piga, "Continuous-time system identification with neural networks: Model structures and fitting criteria," *European Journal of Control*, vol. 59, pp. 69–81, May 2021.

[6] A. Yeşildirek and F. L. Lewis, "Feedback linearization using neural networks," *Automatica*, vol. 31, no. 11, pp. 1659–1664, Nov. 1995.

[7] M. Alamir, "Learning against uncertainty in control engineering," *Annual Reviews in Control*, 2022.

[8] T. Westenbroek, D. Fridovich-Keil, E. Mazumdar, S. Arora, V. Prabhu, S. S. Sastry, and C. J. Tomlin, "Feedback Linearization for Uncertain Systems via Reinforcement Learning," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, May 2020, pp. 1364–1371.

[9] M. Greeff and A. P. Schoellig, "Exploiting Differential Flatness for Robust Learning-Based Tracking Control Using Gaussian Processes," *IEEE Control Systems Letters*, vol. 5, no. 4, pp. 1121–1126, Oct. 2021.

[10] J. Suykens, B. De Moor, and J. Vandewalle, "Nonlinear system identification using neural state space models, applicable to robust control design," vol. 62, no. 1. Taylor & Francis, 1995, pp. 129–152.

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

[12] J. Sjoberg, "On estimation of nonlinear black-box models: how to obtain a good initialization," in *Neural Networks for Signal Processing VII. Proceedings of the 1997 IEEE Signal Processing Society Workshop*, Sep. 1997, pp. 72–81.

[13] A. Isidori, "Elementary theory of nonlinear feedback for multi-input multi-output systems," in *Nonlinear Control Systems*. Springer, 1995, pp. 219–291.

[14] A. Franco, H. Bourles, E. De Pieri, and H. Guillard, "Robust Nonlinear Control Associating Robust Feedback Linearization and H_∞ Control," *IEEE Transactions on Automatic Control*, vol. 51, no. 7, pp. 1200–1207, Jul. 2006.

[15] G. O. Guardabassi and S. M. Savaresi, "Approximate linearization via feedback — an overview," *Automatica*, vol. 37, no. 1, pp. 1–15, Jan. 2001.

[16] A. J. Krener, "Approximate linearization by state feedback and coordinate change," *Systems & Control Letters*, vol. 5, no. 3, pp. 181–185, 1984.

[17] C. Champetier, P. Mouyon, and C. Reboulet, ""Pseudolinearization of multi-input nonlinear systems"," in *The 23rd IEEE Conference on Decision and Control*, Dec. 1984, pp. 96–97.

[18] J. Hauser, "Nonlinear control via uniform system approximation," *Systems & Control Letters*, vol. 17, no. 2, pp. 145–154, Aug. 1991.

[19] A. Paszke *et al.*, "PyTorch: An Imperative Style, High-Performance Deep Learning Library," in *Advances in Neural Information Processing Systems*, vol. 32. Curran Associates, Inc., 2019.

[20] M. Schoukens, "Improved initialization of state-space artificial neural networks," in *2021 European Control Conference (ECC)*. IEEE, 2021, pp. 1913–1918.

[21] R. Pintelon, M. Schoukens, and J. Lataire, "Best Linear Approximation of Nonlinear Continuous-Time Systems Subject to Process Noise and Operating in Feedback," *IEEE Transactions on Instrumentation and Measurement*, vol. 69, no. 10, pp. 8600–8612, Oct. 2020.

[22] T. Wigren and J. Schoukens, "Three free data sets for development and benchmarking in nonlinear system identification," in *2013 European Control Conference (ECC)*, Jul. 2013, pp. 2933–2938.