# The application of learning techniques to improve the control of surface vehicles

Victor Luis Gomes Marchesini Fonseca
victor.fonseca@tecnico.ulisboa.pt

Instituto Superior Técnico, Lisboa, Portugal

October 2023

**Abstract**

This thesis delves into the application of artificial intelligence (AI) techniques for real-time fine-tuning of dynamic models in surface vehicles for control purposes. In the initial stage of this investigation, two baseline nonlinear models were conceived to capture the characteristics of a conventional boat and a hydrofoil, respectively. The first model was designed to be simple, with small nonlinearities, while the second was thought to be challenging with sharp nonlinearities. From those base models, the ground-truth data were generated and used in the learning process of three AI methods for identification. Firstly, the Sparse Identification of Nonlinear Dynamics (SINDy) technique was adopted to identify both the simple and the challenging model. Secondly, a feed-forward Neural Networks (NN) was trained to predict a sequence of states of the hydrofoil model, revealing its potential for application in a reinforcement learning structure. Finally, the hydrofoil model was identified offline and adjusted in real-time using a Long Short Term Memory (LSTM), a Gate Recurrent Unit (GRU), and Vanilla Recurrent Neural Networks (RNN) combined with Extended Kalman Filter (EKF). The models were trained offline by applying Backpropagation Through Time (BPTT), and later they were refined on-the-fly by the EKF. This novel approach allowed adaptation and optimization of control systems during operation. Overall, this research offers insights into the use of AI techniques, including SINDy, feed-forward NN, and RNN, to improve control and model identification capabilities in surface vehicles, particularly hydrofoils.
**Keywords:** RNN; Real-Time Learning; Control; Identification; Marine Vehicle Dynamics.

## 1. Introduction

Autonomous or unmanned vehicles have been playing an important role in research and production fields. Such devices can be used to perform tedious or dangerous tasks for people, reducing the risk of accidents, human errors, and the cost of processes. Autonomy has been applied to self-driven street cars, flying drones, manufacturing automation, and also to the marine and maritime industry.

The progress of autonomous marine vessels is directly related to the development of Control Theory and Artificial Intelligence (AI). This master's research is based on the combination of both. Traditional Control Theory proved to be reliable and consolidated, but it requires dynamic systems to be identified correctly. This can be difficult in some cases. Engineers have to extract from physical properties, and sometimes to assume, the relations between actuators and states. AI can be used to identify, parameterize, and assist in the design of models.

Artificial Intelligence is a broad concept, and in this work, it will be referred to as the most usual learning-based frameworks, which are Machine Learning Learning (ML), Deep Learning (DL), and Reinforcement Learning (RL). Machine Learning, feed-forward, and recurrent neural networks applied to identification and real-time fine-tuning are the objects of investigation presented here. The tests are carried out with the case study of a simple marine surface vehicle and a case study of a hydrofoil. There are difficulties in modeling a hydrofoil, because its dissipative terms, added mass, and Coriolis change drastically to the extent that velocity varies, as seen in Figure 1.

As a summary of this document, Section 2 introduces the most important references for this thesis. Section 3 details the methods used to assess SINDy, the feed-forward NN, and the RNN's. Section 4 depicts the results, and Section 5 presents the conclusions and future work.

## 2. Background

To harness Control with AI, many different approaches can be adopted under the scope of ML, DL, or RL. In what concerns RL, [11] is one of the most emblematic applications. In this work, the

**Figure 1:** Two different modes of a hydrofoil. The normal mode is characterized by low-speed/high-drag while the flying is high-speed/low-drag.

authors showed the power of model-based RL to learn from scratch how to outperform humans in complex games like go, chess, shogi, and even Atari games. The authors continued their work with [12], introducing a module called "Reanalyse" capable of learning new experiences online. The success of [11] reinvigorated the application of RL in many contexts, including marine robotics.

[15] developed a method that learns to choose among three different guidance modes to keep a safe but effective trajectory. [18] uses the RL algorithm Soft actor critic deep reinforcement learning algorithm (SAC) to aim two targets simultaneously: tracking control and collision avoidance. [6] used neural networks and a RL framework to control an Autonomous Underwater Vehicle (AUV). [5] presented a method with deep reinforcement learning, with a quality table Q, where an Unmanned Surface Vehicle (USV) learns path following and collision avoidance by controlling its position and velocity. [17] proposed a method that uses a Deep Reinforcement Learning (DRL) architecture derived from Deep Deterministic Policy Gradient (DDPG) for path following of an USV.

Some authors used RL as a supervisor that fine-tune traditional control loops. This is the case for [4] and [8]. Instead of learning from scratch, this approach takes advantage of the already available information of the model and only learns to correct small deviations.

Despite the expressive results, RL has limitations that must be addressed whenever it is used for physical systems. The sample inefficiency, issues with the formal consensus of stability, risks, and limitations associated with real hardware defy RL applications for robotics. Those aspects led many authors to conduct their studies of RL for robotics with simulations, and not with real experiments. [9] described in their survey how convergence to policy optimality can be inaccurate, slow, and how it can require a massive number of simulations or real-world training steps. These difficulties motivated the application of simpler alternatives in this thesis, such as ML and neural nets.

Regarding ML, [14] demonstrated how Sequential Monte Carlo (SMC) could be a powerful tool to work with complex models such as the progression of a disease. To some extent, their approach could be classified as ML. Other methods are a more direct application of ML, one example is the methodology presented by [3]. [3] created an identification method with machine learning, so-called Sparse Identification of Nonlinear Dynamics (SINDy). The procedure presented by [3] tackles the problem of identifying sparse Ordinary Differential Equations (ODE)'s and Partial Differential Equations (PDE)'s. Sparse means that each state variable derivative is a function of a few state variables. [3] claim that physical problems with high dimensions are commonly sparse. This can be true for the lumped model representation of a marine vessel presented by [7], particularly, if a surface or underwater vehicle is symmetrical, slender, and designed to be stable.

[1] and [10] apply deep neural networks for identification of dynamical systems. [16] presented a methodology that might be considered simultaneously in the domains of deep neural nets, Model Predictive Control (MPC), and reinforcement learning. The authors built a data-driven module to control the gait of a four-legged robot. They trained a feedforward NN to output the next 20 state variations $\delta s_t, .., \delta s_{t+H}$, given a sequence of actions $(a_1, a_2, ..., a_H)$ and the state $s_t$. In other words, it learns how to predict the behavior of the physical model, while the Data-Efficient Cross Entropy Method (CEM) takes care of planning the next optimal sequence of actions to attain the control target.

Additionally, in the matter of using NN as a surrogate for dynamic systems, [13] demonstrated that LSTM's can be used to identify discrete nonlinear dynamics. [2] uses EKF as an alternative to Stochastic Gradient Descend (SGD) method to train Recurrent Neural Networks (RNN)'s with loss functions that are convex. The proposed solution considers regularization terms like L1 and L2. The authors show that the Kalman Filter method has a performance comparable to SGD in a nonlinear system identification benchmark and in training a linear system for classification. [2] mentions that this approach has the potential of training RNN's "on-the-fly".

[2], [16], and [3] are the main references for the work developed in this thesis. They were the foundation for the methods explained in section 3

### 3. Methodology

In this research, three learning approaches were carried out to identify nonlinear USV models, they are: SINDy, a feed-forward NN predictor, and RNN's with real-time learning. The research consisted of testing the simplest of the three, SINDy, and verifying where it fails. Then, the feed-forward NN, and the RNN methods, which are more complex, were applied to the condition where SINDy did not perform well. In the end, a comparison was established among all learning structures.

To apply the learning methods, datasets were generated from a 3degree-of-freedom surface vehicle model based on [7]. [7] presents a lumped model derived from

$$\tau = M\dot{\nu} + C(\nu)\nu + D(\nu)\nu, \tag{1}$$

with $\tau$ being the external forces and torques applied to the vessel; C, M, and D being Coriolis, mass, and drag matrices. $\nu$ is the surge, sway, and yaw velocities of the vessel in the body frame, given by

$$\nu = [u, v, r]^T. \tag{2}$$

The train and test datasets were extracted from 2 different base models:

1. a simple surface vehicle model;
2. a complex hydrofoil model composed by a combination of two nonlinear drag models.

Both the simple and the complex dynamic systems have two actuators, which are a central propeller, and a rudder. Sections 3.1, 3.2 describe those two models, while Sections 3.4, 3.3, and 3.5 detail the learning strategies for identification explored in this thesis.

#### 3.1. Simple Baseline Model

In the simple linear drag model, C, M, and D from (1) are

$$M = \begin{bmatrix} m_{11} & 0 & 0 \\ 0 & m_{22} & m_{23} \\ 0 & m_{32} & m_{33} \end{bmatrix}, D = \begin{bmatrix} d_{11} & 0 & 0 \\ 0 & d_{22} & d_{23} \\ 0 & d_{32} & d_{33} \end{bmatrix}, C = \begin{bmatrix} 0 & 0 & c_{13} \\ 0 & 0 & c_{23} \\ c_{31} & c_{32} & 0 \end{bmatrix}. \tag{3}$$

Combining (1), (2), (3), and including the propeller dynamics, given by

$$P_f = PP_{cmd} - \dot{P}_f, \tag{4}$$

the resulting dynamic system becomes:

$$\begin{aligned} X = y &= \begin{bmatrix} \nu \\ P_f \end{bmatrix}, \\ \dot{X} &= \begin{bmatrix} M^{-1}(\tau - C\nu - D\nu) \\ PP_{cmd} - P_f \end{bmatrix}, \\ \tau &= [P_f,\ 0,\ u\, RD_{cmd}]^T. \end{aligned} \tag{5}$$

In (4), the propeller dynamics are considered, and $P_f$ is the actual force produced given a reference $PP_{cmd}$. $RD_{cmd}$ is the proportional relation between the torque generated by the rudder and the surge velocity u. Under these conditions, the actuators $\mu$ are defined by

$$\mu = [PP_{cmd},\ RD_{cmd}]^T. \tag{6}$$

#### 3.2. Challenging Baseline Model

The second base model was created to be more realistic and challenging for identification. It is formed by a combination of two lumped models with quadratic drag. The models represent the flying and normal modes of operation exhibited in Figure 1. One of the two lumped models corresponds to a hydrofoil operating in high velocity and low drag (flying mode), and the other corresponds to a hydrofoil operating in low velocity and high drag (normal mode). The transition between the modes is carried out by a sigmoid function. As a result, the dynamic system becomes

$$\begin{aligned} y &= \begin{bmatrix} \nu \end{bmatrix}, \quad X = \begin{bmatrix} \nu \\ P_f \end{bmatrix}, \\ \dot{X} &= \sigma(u - u_0)\,[f_{fly}(X, \mu)] \\ &+ \sigma(-u + u_0)\,[f_{normal}(X, \mu)], \end{aligned} \tag{7}$$

with $u_0$ being the surge velocity where the hydrofoil transits from normal to flying mode. $f_{fly}$ and $f_{normal}$ are, respectively, the dynamic systems in high speed/low drag and low speed/high drag. The sigmoid function is expressed by

$$\sigma(x) = \frac{1}{\exp\left(-12\frac{s}{m}\,x\right) + 1}. \tag{8}$$

The dynamic relation expressed by (4) is preserved for the challenging model. However, it is assumed that the force produced by the propeller $P_f$ is not measured, and the relation between the rudder torque $RD_{cmd}$ and the rudder angle $RD_{ang}$ is not known. Only $PP_{cmd}$ and the rudder angle $RD_{ang}$ are available. In this case, the control $\mu_a$ and the force exerted by the actuators $\tau_h$ become

$$\begin{aligned} \mu_a &= [PP_{cmd},\ RD_{ang}]^T, \\ \tau_h &= [P_f,\ 0,\ u\, sin(RD_{ang})K]^T, \end{aligned} \tag{9}$$

with K being a constant that depends on the physical features of the rudder. The hydrofoil drag used here is given by

$$D_h = \begin{bmatrix} d_{11} + d_{a11}|u| & 0 & 0 \\ 0 & d_{22} + d_{a22}|v| & d_{23} + d_{a23}|r| \\ 0 & d_{32} + d_{a23}|v| & d_{33} + d_{a33}|r| \end{bmatrix}. \quad (10)$$

The reduction of drag and added mass coefficients for flying and normal modes is 50 %. This leads to

$$f_{normal} = M^{-1}(\tau_h - C\nu - D_h\nu), \quad (11)$$

$$f_{fly} = M_{fly}^{-1}(\tau_h - C\nu - 0.5D_h\nu), \quad (12)$$

with $M_{fly}$ being the mass matrix with the flying (smaller) added mass. This way, the second model becomes more challenging to identify due to:

- the quadratic drag;
- the composition of two models multiplied by sigmoids;
- the forces applied by the actuators are not directly measured and instantiated;
- a Gaussian noise with standard deviation $[1\,cm/s,\ 1\,cm/s,\ 1\,deg/s]$ was added to the generated data.

### 3.3. SINDy Structure and Test Arrangement

The first learning method tested was SINDy. SINDy was applied to identify the basic model (5), and the complex model (7) with the Python package SINDyPy. The process of identification with SINDy comprised cleaning the input data and choosing properly the hyperparameters, particularly the sparsity parameter $\lambda$. The library used was the Polynomial Library, and the optimizer adopted was the least square. The identified system was tested with Feedback Linearization. Anticipating the results shown in Section 4, SINDy results were very good for the simple system presented. However, it was not able to identify correctly the complex model. This limitation motivated the investigation of other methods; the Offline estimator, and RNN's variations (LSTM, GRU and Vanilla RNN).

### 3.4. Offline NN Structure and Test Arrangement

A feedforward NN was applied to estimate the 20 future states of the hydrofoil (complex model). The neural network and its state predictions are defined as

$$\hat{s}_{t+1} = f_\theta(s_t, a_t). \quad (13)$$

In this work, the inputs of $f_\theta$ were specified as the 4 latest states, expressed by

$$s_t(k) = \{y(k-3),\ y(k-2),\ y(k-1),\ y(k)\}, \quad (14)$$

alongside the latest 4 command steps concatenated with the future 19 command steps, expressed by

$$a_t(k) = \{\mu(k-3),\ \mu(k-2),\ ...\ \mu(k+19)\}. \quad (15)$$

The output of $f_\theta(s_t, a_t)$ are the estimated states of the future 20 steps, given by

$$\hat{s}_{t+1}(k) = \{\hat{y}(k+1),\ \hat{y}(k+2),\ ...\ \hat{y}(k+20)\}. \quad (16)$$

After Z score normalization, before forwarding the data to the neural net, the inputs are flattened and concatenated. The NN is formed by linear layers of sizes $(58 \times 256)$, $(256 \times 256)$, $(256 \times 256)$, $(256 \times 60)$. The activation function used was the hyperbolic tangent. Figure 2 shows the arrangement adopted. For training, the loss function chosen was the mean square error, given by

$$\mathcal{L} = \sum_{n=1}^{20} ||y_{norm}(k+n) - \hat{y}(k+n)_{norm}||_2^2, \quad (17)$$

and Adam Optimizer was selected for the Stochastic Gradient Descent process. The dataset

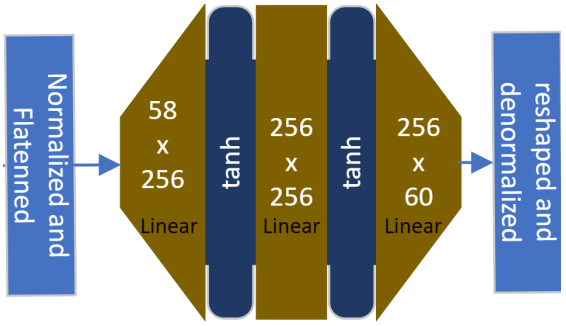


**Figure 2:** Neural Network design for system identification and MPC.

summed up approximately 2 hours of simulation with a step of 10 ms (around 72000 steps). The training data was generated with Matlab Simulink simulations. The control inputs for training were carefully chosen to excite the system dynamics in a variety of ways. To fulfill this objective, the commands sent to the actuators were a composition of: random variables; constant 0.0 values (to train the natural decaying effect, and independent impact of each actuator); chirp functions; triangular functions, etc. The system excitation was also designed in a way that the hydrofoil would operate in low and high-speed modes. In this case, the transition between low and high-speed modes of the hydrofoil occurred when u crossed 2.5 m/s.

### 3.5. RNN's Structures and Test Arrangements

The process to identify the complex model in (7) with the RNN's (LSTM, GRU, and Vanilla RNN)

also included a real-time adjusting mechanism. Initially, the RNN hydrofoil model was trained offline with BPTT, similarly to what has been done with the feed-forward network with SGD. Afterward, the model learned offline was put into operation and improved or fine-tuned by the EKF technique proposed by [2]. The RNN transition function presented by [2] is

$$\begin{aligned} h(k+1) &= f_h\left(h(k), \mu(k), \theta_h(k)\right) + \xi(k) \\ y(k) &= f_y\left(h(k), \mu(k), \theta_y(k)\right) + \zeta(k) \\ \theta(k+1) &= \theta(k) + \eta(k), \quad \theta(k) \triangleq \begin{bmatrix} \theta_h(k) \\ \theta_y(k) \end{bmatrix}, \end{aligned} \tag{18}$$

where $f_h$ and $f_y$ are recurrent neural networks, and $\theta_h$ and $\theta_y$ are, respectively, their parameters. $\mu(k)$ are the input commands and $h(k)$ a hidden state. In this work, the hidden layer for the LSTM, GRU is 18, and the Vanilla has 2 layers. $\zeta(k)$, $\xi(k)$, $\eta(k)$ are Gaussian noises. y(k) is the estimated output state. For The hydrofoil dynamic system with 2 modes defined by (7), y(k) are the surge, sway, and yaw velocities,

$$y(k) = [u(k),\ v(k),\ ,r(k)]^T. \tag{19}$$

To assess the performance of the EKF method to adapt to unknown conditions in real time, the RNN models were initially trained offline through backpropagation with Adam Optimizer. The offline training dataset had only trajectories in low speed. This means that the models are expected to perform poorly when exposed to high-speed hydrofoil mode. For real-time learning, the dataset used contained both low and high-speed modes. Figure 3 has a visual description of this process.

## 4. Results & discussion

### 4.1. SINDy results

The results of SINDy for identifying the simple model described in 3.1 are shown in Figure 4. It can be seen that the controlled trajectory follows the circular reference with a small error. This result showed that SINDy is suitable for identifying simple nonlinear systems.

Despite the good results exhibited in Figure 4, SINDy did not have a good performance in identifying the complex system from (7). Depending on the hyperparameters chosen for SINDy, the controlled trajectory either diverged quickly to infinite values or remained zero. The poor performance of the identified models precluded their use for Feedback Linearization. It could still have been possible to improve results by using equality constraints or increasing the complexity of the nonlinear terms, for example. However, setting up those suggestions could be cumbersome, would require more in-depth knowledge of the problem, and would undermine the flexibility and freedom offered by data-driven methods.

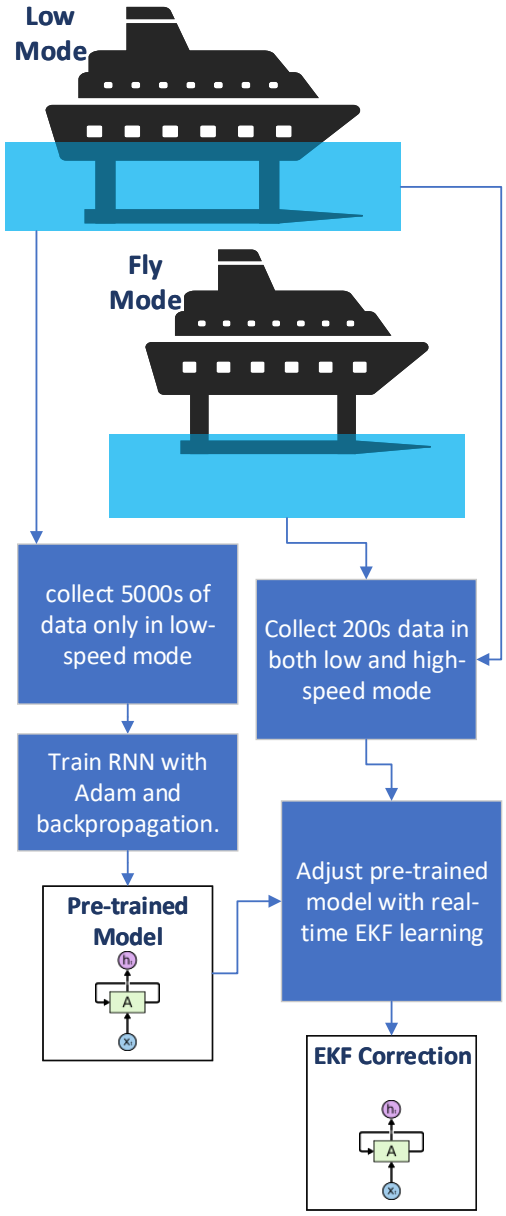


**Figure 3:** Process to verify the performance of the EKF real-time learning in comparison with offline learning.

### 4.2. Feed-forward NN results

To verify the accuracy of the NN, its results were compared with an ideal model using only the low-speed mode of the hydrofoil ("C0" in Figure 5). With this, the error between the ground-truth and "C0" is expected to increase every time the surge speed exceeds 2.5 m/s (from where the hydrofoil enters the flying mode). Conversely, the error between the ground-truth and the trained NN should remain stable and smaller than "C0" in the flying mode.

Figure 5 shows the error after training the NN. As expected, the errors of the reference "C0" grow as the speed goes above the transition point of 2.5 m/s, while for the neural networks, the error re-

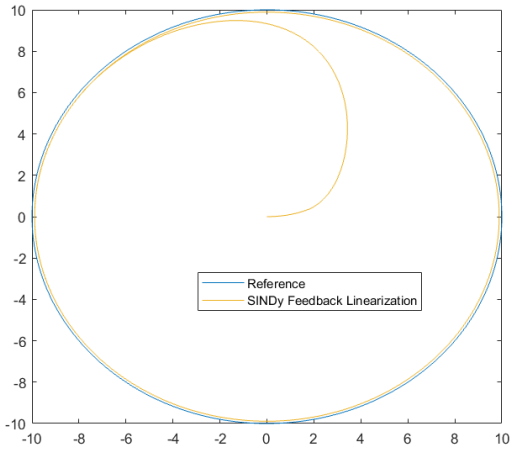


**Figure 4:** Tracking a circular trajectory using Feedback Linearization with the model identified with SINDy, with data extracted from (5).

mains small, regardless if the surge velocity is below or above the threshold of 2.5 m/s. This result shows that the NN effectively learned to predict the next 20 states of the hydrofoil. The error for each velocity seen in Figure 5 is calculated using

$$Error(k) = \frac{1}{200}\sum_{m=0}^{9}\sum_{n=1}^{20}||\nu(k+20m+n) - \hat{\nu}(k+20m+n|s_t(k+20m), a_t(k+20m))||_1,$$

with $\hat{\nu}(k+n|s_t(k), a_t(k))$ being the velocities predicted by the neural network or the reference model "C0", given the inputs $s_t(k)$ and $a_t(k)$.

### 4.3. LSTM, GRU, and Vanilla RNN results

The results of the LSTM are shown in Figures 6 and 7. Those plots were obtained from a lawnmower trajectory. The transition velocity from low-speed mode to high-speed mode was 2.5 m/s. With the EKF real-time correction, the average error was reduced in all the cases, for all velocities.

When compared with the pre-trained, the maximum error in the whole trajectory was also reduced in all predictions tested for u and v velocities, except from step 97 to 99 for u. For yaw velocity, the maximum prediction error for the real-time was below the pre-trained up to 42 steps ahead, after 40 steps the model presented a greater maximum error than the pre-trained. The explanation for this comes from the fact that, during more dramatic velocity changes, peaks of error pop out, indicating that it takes some time to adapt to the new conditions. This behavior is observed in the chart of yaw velocity (r) from Figure 8. Despite the errors, the LSTM trained in real time can be used to predict the next 20 steps and seems to be reasonable to be applied as a predictor in an MPC framework.

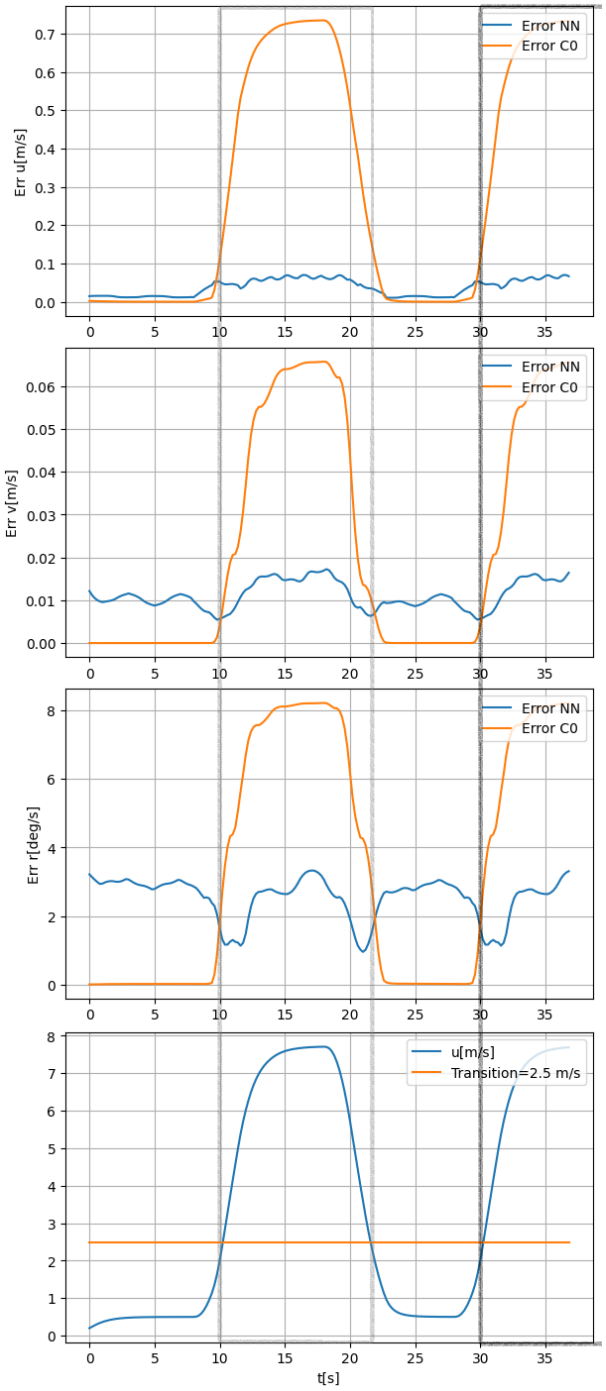


**Figure 5:** "Error NN" is the error between the NN prediction for either u, v, or r and the hydrofoil. Similarly, "Error C0" is the error between a low-speed model and the hydrofoil. "u[m/s]" is the surge velocity of the boat for the test dataset, and "Transition" indicates the borderline between low and high-speed mode.

The results of the GRU and the Vanilla RNN were also improved by the EKF, as seen in Figure 9. Although the Vanilla RNN was slightly enhanced by the real-time adjustment, the performance of LSTM and GRU were far superior for both the pre-trained and real-time corrected.

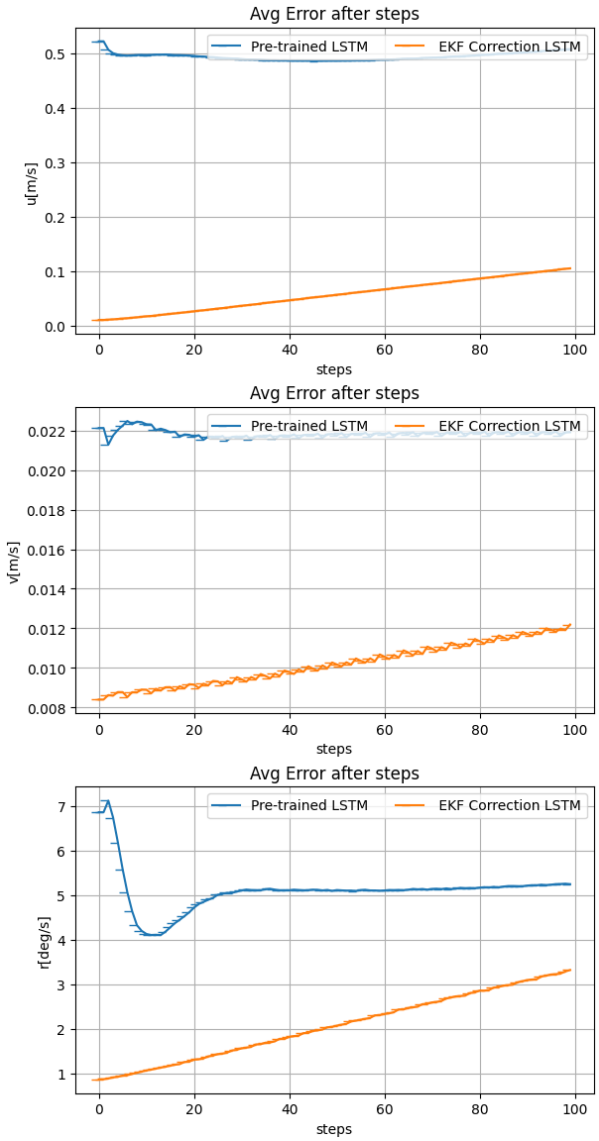


**Figure 6:** Average error of an LSTM as a function of the number of steps predicted ahead.

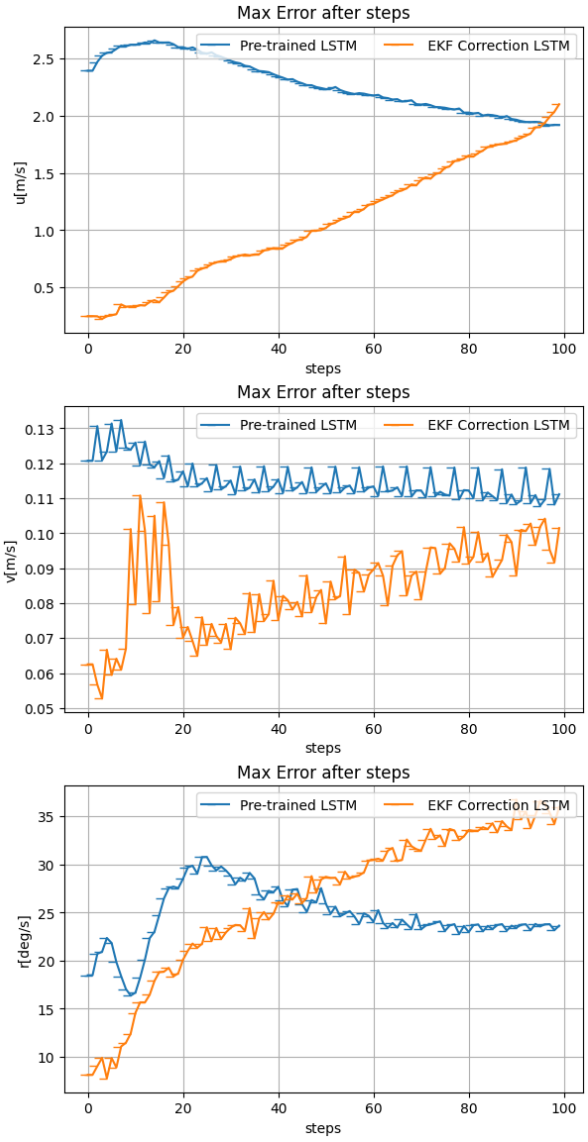


**Figure 7:** Maximum error of an LSTM as a function of the number of steps predicted ahead, for the test set.

## 5. Conclusions

The learning methods studied in this research can facilitate or contribute to control identification for marine dynamic systems. Nevertheless, it is necessary to comprehend under which conditions their maximum potential can be exploited. The results shown in Section 4 provide the basis to understand how those approaches can be harnessed. Table 1 sums up a comparison among SINDy, Offline NN training, Vanilla RNN, GRU and LSTM.

Section 4.1 demonstrates that SINDy is suitable for identifying nonlinear marine vehicle dynamics. This method is the simplest learning method, when compared with Offline NN training, Vanilla RNN, GRU, and LSTM. It learns quickly, does not require a large amount of memory, and consumes less data for training. Because the identified model is in the canonical form, Feedback Linearization or Backstepping can be applied directly with this method. The Python library PySINDy allows a design with polynomial terms or other nonlinear terms that can be easily customized. The more it is known about the nonlinear system, the more SINDy can be shaped with constraints and proper nonlinear terms to fit better the marine craft dynamics. In opposition to this, if the system to be modeled has sharp nonlinearities, like the hydrofoil model (7), or if not much information is available beforehand, the performance can be poor. For the case where SINDy fails, Sections 4.2 and 4.3 evidenced that neural networks can be trained to predict the dynamic states of marine vessels.

Section 4.2 showed that with enough data, the identification with NN's do not require much knowledge from the control designers beforehand and the predictions are very accurate when it is prop-

**Table 1:** Relative comparison between methods presented in Section 3.

| | | | | | | Accuracy | |
|---|---|---|---|---|---|---|---|
| | **Training Time** | **Data Required** | **Memory Required** | **Real-time Learning Capabilities** | **Flexibility with different models** | **Simple systems** | **Complex systems** |
| **SINDy** | 0.1~0.5 h | Low | Low | High | Low | High | Low |
| **Offline NN** | 5~12 h | High | High | Low | High | High | High |
| **Vanilla RNN** | 5~12 h | Medium | Medium | Medium | Medium | Medium | Low |
| **GRU** | 1~5 h | Medium | Medium | High | High | High | Medium |
| **LSTM** | 1~5 h | Medium | Medium | High | High | High | Medium |

erly trained. This is the most complex learning method, when compared with SINDy, Vanilla RNN, GRU and LSTM, though. It learns slowly, requires a large amount of memory, and consumes more data for training. It is possible to use EKF real-time learning with this method with some adaptations, but the arrangements for that were left as future work.

Section 4.3 showed that RNN's are apt to be trained in real-time with EKF. This learning structure is simpler, and learns much quicker, when compared with the offline NN predictor. The predictions can be very accurate when it is properly designed and trained, which makes it appropriate to identify complex systems, such as the hydrofoil in (7). With enough data, the identification does not require much knowledge from the control designers beforehand. Feedback Linearization or Backstepping can be applied if the RNN structure is designed to be in the canonical form $f(X) + g(X)\mu$.

Despite being effective, the EKF real-time learning has some limitations. Firstly, [2] clarifies that the EKF method does not guarantee a convergence to global optima. The EKF can lead the model to be "greedy" with respect to the current conditions. It can "forget" what was learned previously, especially if the NN does not have enough degrees of freedom to represent the actual system. Due to the greediness of EKF in real time, the model is subjected to become degraded. Thorough tests carried out with Vanilla RNN, GRU and LSTM showed that if the EKF training is repeated for several epochs, the model can diverge. Thus, it is desirable to have a module that checks model integrity along the EKF. Finally, the Extended Kalman Filter requires extra attention to memory usage, especially when graphics processing unit (GPU) is employed. This occurs because EKF uses data structures that grow quadratically with the number of parameters $\theta$.

**6. Future work**

This work corroborated with the hyposthesis that SINDy or NN's model-based can be used for identification. It also showed that real time adjusting can be applied for marine surface vehicles with EKF. Nevertheless, this research can be improved, extended, or even completely reinvented. Considering this, five possible future works are suggested:

1. designing and testing RNN structures that support Backstepping or Feedback Linearization;
2. applying regularization terms to improve performance for real-time learning;
3. applying RNN real-time learning under the RL framework (it learns to control, instead of the state transition);
4. designing and testing a structure with real-time learning for the NN predictor from Section 3.4;
5. applying RNN real-time learning to predict the $n^{th}$ step ahead, given a sequence of control commands, instead of the first.

**References**


[1] A. Ayyad, M. Chehadeh, M. I. Awad, and Y. Zweiri. Real-time system identification using deep learning for linear processes with application to unmanned aerial vehicles. *IEEE Access*, 8:122539–122553, 2020.

[2] A. Bemporad. Recurrent neural network training with convex loss and regularization functions by extended kalman filtering. *IEEE Transactions on Automatic Control*, 2022.

[3] S. L. Brunton, J. L. Proctor, and J. N. Kutz. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the national academy of sciences*, 113(15):3932–3937, 2016.

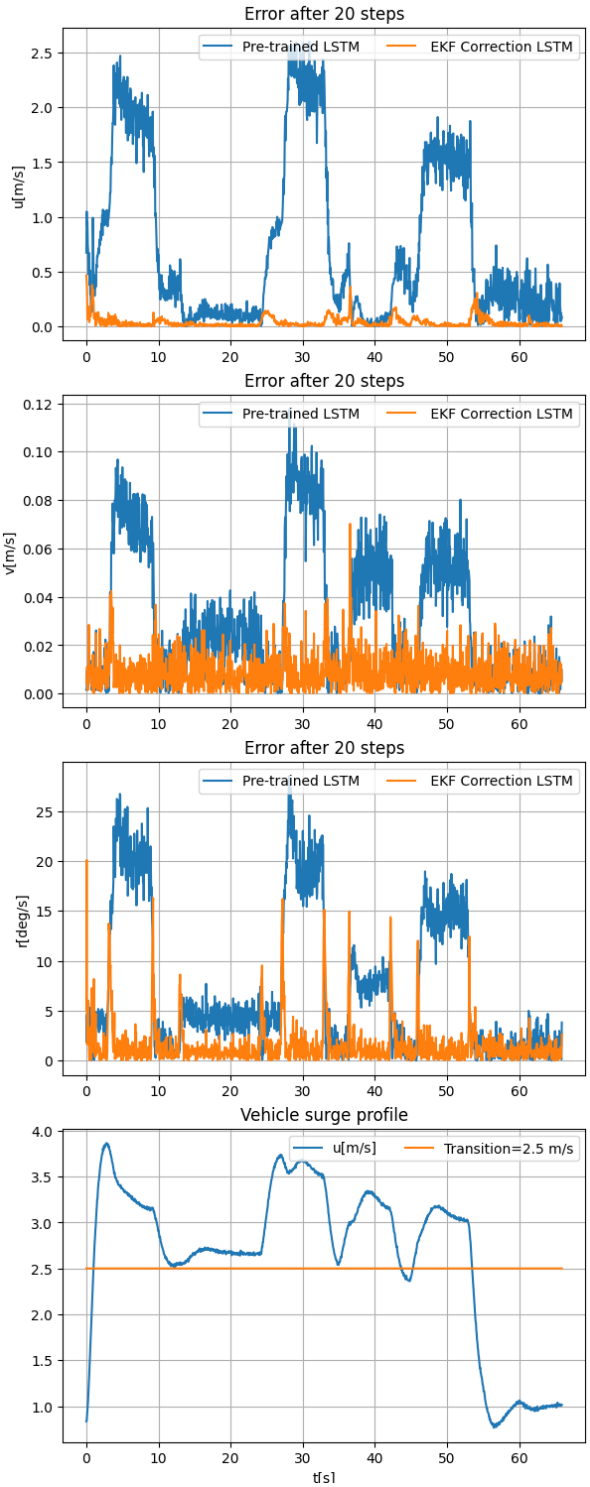


**Figure 8:** Comparison of the errors for a prediction of 20 steps ahead, between the pre-trained LSTM and the EKF real-time adjusted LSTM. The model had a cell size of 18 and a hidden size of 3.

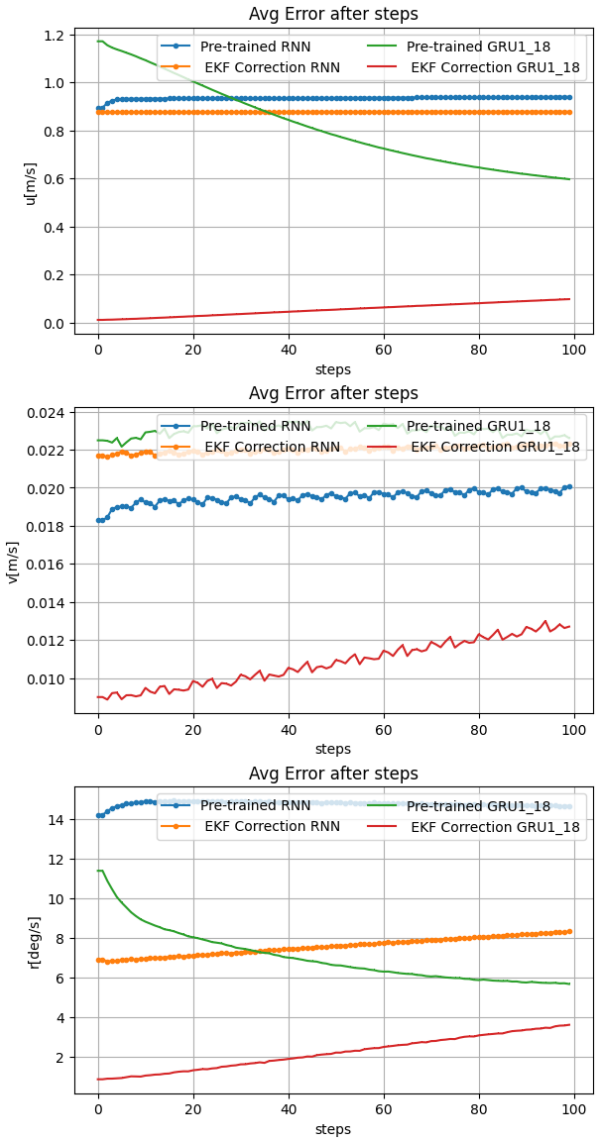


**Figure 9:** Average error of an GRU and Vanilla RNN as a function of the number of steps predicted ahead.


[4] I. Carlucho, M. De Paula, and G. G. Acosta. An adaptive deep reinforcement learning approach for mimo pid control of mobile robots. *ISA transactions*, 102:280–294, 2020.

[5] Y. Cheng and W. Zhang. Concise deep reinforcement learning obstacle avoidance for underactuated unmanned marine vessels. *Neurocomputing*, 272:63–73, 2018.

[6] R. Cui, C. Yang, Y. Li, and S. Sharma. Adaptive neural network control of auvs with control input nonlinearities using reinforcement learning. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 47(6):1019–1029, 2017.

[7] T. I. Fossen. *Handbook of marine craft hydrodynamics and motion control*. John Wiley & Sons, 2011.

[8] T. Johannink, S. Bahl, A. Nair, J. Luo, A. Kumar, M. Loskyll, J. A. Ojea, E. Solowjow, and S. Levine. Residual reinforcement learning for robot control. In *2019 International Con-*

*ference on Robotics and Automation (ICRA)*, pages 6023–6029. IEEE, 2019.

[9] S. Kuutti, R. Bowden, Y. Jin, P. Barber, and S. Fallah. A survey of deep learning applications to autonomous vehicle control. *IEEE Transactions on Intelligent Transportation Systems*, 22(2):712–733, 2020.

[10] L. Ljung, C. Andersson, K. Tiels, and T. B. Schön. Deep learning and system identification. *IFAC-PapersOnLine*, 53(2):1175–1181, 2020.

[11] J. Schrittwieser, I. Antonoglou, T. Hubert, K. Simonyan, L. Sifre, S. Schmitt, A. Guez, E. Lockhart, D. Hassabis, T. Graepel, et al. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839):604–609, 2020.

[12] J. Schrittwieser, T. Hubert, A. Mandhane, M. Barekatain, I. Antonoglou, and D. Silver. Online and offline reinforcement learning by planning with a learned model. *Advances in Neural Information Processing Systems*, 34:27580–27591, 2021.

[13] Y. Wang. A new concept using lstm neural networks for dynamic system identification. In *2017 American control conference (ACC)*, pages 5324–5329. IEEE, 2017.

[14] A. Wigren, J. Wågberg, F. Lindsten, A. G. Wills, and T. B. Schön. Nonlinear system identification: Learning while respecting physical models using a sequential monte carlo method. *IEEE Control Systems Magazine*, 42(1):75–102, 2022.

[15] J. Woo and N. Kim. Collision avoidance for an unmanned surface vehicle using deep reinforcement learning. *Ocean Engineering*, 199:107001, 2020.

[16] Y. Yang, K. Caluwaerts, A. Iscen, T. Zhang, J. Tan, and V. Sindhwani. Data efficient reinforcement learning for legged robots. 100:1–10, 30 Oct–01 Nov 2020.

[17] Q. Zhang, J. Lin, Q. Sha, B. He, and G. Li. Deep interactive reinforcement learning for path following of autonomous underwater vehicle. *IEEE Access*, 8:24258–24268, 2020.

[18] Q. Zhang, W. Pan, and V. Reppa. Model-reference reinforcement learning for collision-free tracking control of autonomous surface vehicles. *IEEE Transactions on Intelligent Transportation Systems*, 23(7):8770–8781, 2022.