Preprints: 8th IFAC Conference on Nonlinear Model Predictive Control
August 21-24, 2024 | Kyoto, Japan

Physics-Informed Online Learning by Moving Horizon Estimation: Learning Recurrent Neural Networks in Gray-box Models [★]

Kristoffer Fink Løwenstein [*,**] **Daniele Bernardini** [**]
Alberto Bemporad [***] **Lorenzo Fagiano** [*]

[*] *Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, Via Ponzio 34/5, Milano, 20133, Italy, (e-mail: kristofferfink.loewenstein@polimi.it, lorenzo.fagiano@polimi.it).*
[**] *ODYS S.r.l, Via Pastrengo 14, Milano, 20159, Italy, (e-mail: kristoffer.lowenstein@odys.it, daniele.bernardini@odys.it)*
[***] *IMT School for Advanced Studies, Piazza San Francesco 19, Lucca, 55100, Italy, (e-mail: alberto.bemporad@imtlucca.it)*

Abstract: In Model Predictive Control (MPC) closed-loop performance heavily depends on the quality of the underlying prediction model, where such a model must be accurate and yet simple. A key feature in modern MPC applications is the potential for online model adaptation to cope with time-varying changes, part-to-part variations, and complex features of the system dynamics not caught by models derived from first principles. In this paper, we propose to use a *physics-informed*, or *gray-box*, model that extends the physics-based model with a data-driven component, namely a Recurrent Neural Network (RNN). Relying on *physics-informed* models allows for a rather limited size of the RNN, thereby enhancing online applicability compared to pure black-box models. This work presents a method based on Moving Horizon Estimation (MHE) for simultaneous state estimation and learning of the RNN sub-model, a potentially challenging issue due to limited information available in noisy input-output data and lack of knowledge of the internal state of the RNN. We provide a case study on a quadruple tank benchmark showing how the method can cope with part-to-part variations.

Keywords: Learning-based MPC, Nonlinear MPC, Moving Horizon Estimation, Physics-informed learning, Adaptive MPC, Recurrent Neural Network, Gated Recurrent Unit

1. INTRODUCTION

A fundamental requirement for model-based control and estimation techniques, such as MPC and Extended Kalman Filtering (EKF), is a dynamical model that is accurate and yet as simple as possible for reduced online computations. Modeling the dynamics of real-world processes in industrial systems using solely physical principles is challenging and time-consuming due to the complex underlying physical phenomena. Model mismatches occurring in industrial applications can be categorized into three types: *(i)* modeling simplifications, *(ii)* part-to-part variations, and *(iii)* time-varying changes of the system dynamics due to the environment or wear and tear of components. Recently, learning-based control has become a favored solution method due to the success of machine learning and increasing computational resources, see (Hewing et al., 2020; Hou and Wang, 2013; Brunke et al., 2022). Learning-based control incorporates principles from machine learning in the control strategy, and the use of machine learning ideas in control-oriented modeling is far from new (Draeger et al., 1995; Pan and Wang, 2008; Masti and Bemporad, 2021). Offline learning can be used to address issues *(i)* and *(ii)*, and ideas have been proposed to cope with issue *(iii)*, see (Bemporad, 2023; Maiworm et al., 2021; Nguyen-Tuong and Peters, 2011) for what is often called *lifelong learning*. For example, lifelong learning in robotics has been a topic of research for decades (Thrun and Mitchell, 1995) and is still active today (Taylor et al., 2021). One promising research direction is *physics-informed* learning, e.g., gray-box modeling where available physical knowledge is embedded in the model or training process (Karniadakis et al., 2021).

The main contribution of this paper is to extend the Moving Horizon Estimation (MHE) based framework proposed in (Løwenstein et al., 2023) from learning static model mismatches to learning dynamical subsystems, therefore accounting for the role of the past history of the subsystem on its current and future (predicted) evolution. A well-suited choice of model structure is the RNN. Contrary to standard Feedforward Neural Networks (FNNs), RNNs incorporate memory through their hidden states, making them superior for long-term prediction. The MHE-based method fits very well with RNNs because the hid-

[★] This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No. 953348 (ELO-X).

den states can readily be incorporated as states to be estimated. Further, constraints can be imposed to embed additional physical knowledge, which is not possible by using classical estimation methods such as EKF, while still being real-time feasible (Kühl et al., 2011). Lastly, MHE provides a meaningful way of preserving existing knowledge through the so-called *arrival cost*, which makes it particularly suitable for model adaptation. Specifically for system identification Long-Short Term Memory (LSTM) networks, Gated Recurrent Units (GRUs), and Echo State Networks (ESNs) have shown particularly good performance (Hochreiter and Schmidhuber, 1997; Cho et al., 2014; Jaeger, 2001). Recent developments range from the investigation of theoretical properties (Bonassi et al., 2022, 2021) to real-world applications (Lanzetti et al., 2019). Interestingly, (Bonassi et al., 2021) provides conditions for guaranteeing Input-to-State-Stability (ISS) and Incremental-Input-to-State-Stability (δ-ISS) for these types of RNNs. The stability conditions can be formulated as inequality constraints and from a theoretical viewpoint fit very well with optimization-based estimation techniques such as the proposed MHE framework for learning and adapting RNNs. However, one finding of the work presented here is that the implementation of such constraints significantly increases the complexity of the optimization problem to be solved online, due to the nonlinear and non-smooth nature of the stability conditions. Therefore imposing such constraints may not be suitable for fast real-time applications. The impact of neglecting the stability conditions during learning is discussed further in the numerical example reported in Section 4.

The paper is organized as follows. Section 2 develops the model structure and introduces the learning problem. Section 3 presents the MHE-based framework and training algorithm. Simulation results are shown in Section 4 and conclusions are drawn in Section 5.

2. BACKGROUND AND PROBLEM FORMULATION

We consider a dynamical system governed by the discrete-time dynamics

$$x_{k+1} = f\left(x_k, u_k, p_k, z_k\right) + v_{x,k} \tag{1a}$$

where $k \in \mathbb{Z}$ is the discrete time variable, $x_k \in \mathbb{R}^{n_x}$ is the state vector, $u_k \in \mathbb{R}^{n_u}$ the control input, $p_k \in \mathbb{R}^{n_p}$ is an output of an unknown dynamical subsystem defined below in (1d), and $v_{x,k}$ is a state-noise term that we assume normally distributed with zero-mean and covariance Q_x. Vector $z_k \in \mathbb{R}^{n_z}$ collects measured exogenous signals which impact the system's behavior, e.g., temperature or pressure. The output equation of the system is defined as

$$y_k = g\left(x_k, u_k, p_k, z_k\right) + v_{y,k} \tag{1b}$$

where $y_k \in \mathbb{R}^{n_y}$ is the output vector and $v_{y,k}$ the measurement noise, that we assume normally distributed with zero-mean and covariance R_y. We represent the unknown part of the model as a dynamical subsystem governed by its own dynamics

$$s_{k+1} = r\left(s_k, q_k, k\right) + v_{s,k} \tag{1c}$$

where $s_k \in \mathbb{R}^{n_s}$ is the hidden internal state and $v_{s,k}$ is a state disturbance term that we assume normally distributed with zero-mean and covariance Q_s. Vector $q_k \in$

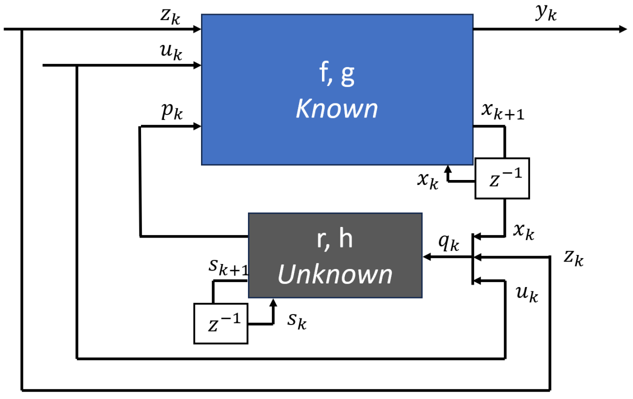


Fig. 1. Structure of the dynamical system. It consist of a known part and a unknown dynamical subsystem. The aim is to learn a model of the unknown system and adapt it as the system evolves.

$\mathbb{R}^{n_q}$ defines the inputs to the subsystem and q_k may collect x_k, u_k, and z_k fully or partially depending on the structure of the overall system. Further, we define the output of the subsystem as

$$p_k = h\left(s_k, q_k, k\right) \tag{1d}$$

which describes the interaction between the subsystem (1c)-(1d) and the overall dynamical system (1a)-(1b). The structure of the dynamical system is depicted in Fig. 1.

In this paper, we model the unknown subsystem in (1c)-(1d) using the RNN model

$$\hat{s}_{k+1} = \hat{r}_{RNN}\left(\hat{s}_k, \hat{q}_k; \Phi_k\right) \tag{2a}$$

with the associated output equation

$$\hat{p}_k = \hat{h}_{RNN}\left(\hat{s}_k, \hat{q}_k; \Phi_k\right) \tag{2b}$$

where $\Phi_k \in \mathbb{R}^{n_\Phi}$ is the vector of parameters defining the RNN at time k, assumed to be time-varying in accordance with the possibly time-varying nature of the system at hand. Without loss of generality, any of the aforementioned RNN structures can be adopted in (2). In this work we focus on GRUs due their beneficial trade-off between model-capabilities and complexity, which is favorable for online applications (Chung et al., 2014). We remark that, besides modeling physical subsystems, $\hat{h}_{RNN}$ and $\hat{r}_{RNN}$ could represent complex disturbances or model mismatches between the true dynamics and their model which cannot be captured by static approximations. Equation (2) defines the model to be identified and the fundamental challenge is to learn and adapt the RNN parameter vector Φ_k in (2) such that it is consistent with the true subsystem in (1c)-(1d). Since p_k, x_k, and s_k are not available, but only the signals u_k, z_k, and the noise corrupted measurements y_k, we cannot directly learn the model in (2), but must rely on a prediction model of (1) defined as

$$\hat{x}_{k+1} = f\left(\hat{x}_k, u_k, z_k, \hat{p}_k\right) \tag{3a}$$

$$\hat{y}_k = g\left(\hat{x}_k, u_k, z_k, \hat{p}_k\right) \tag{3b}$$

$$\hat{s}_{k+1} = \hat{r}_{RNN}\left(\hat{s}_k, \hat{q}_k; \Phi_k\right) \tag{3c}$$

$$\hat{p}_k = \hat{h}_{RNN}\left(\hat{s}_k, \hat{q}_k; \Phi_k\right). \tag{3d}$$

The goal of this paper is to solve the learning problem of matching the gray-box model (3) to the true system in (1), formally stated as

$$\min_{\hat{x}_0,\hat{s}_0,\{\Phi_j\}} \sum_{j=0}^{k} \ell_y\left(y_j,\hat{y}_j\right) \tag{4a}$$
$$\text{s.t. } \hat{x}_{j+1} = f(\hat{x}_j,u_j,z_j,\hat{p}_j) \tag{4b}$$
$$\hat{y}_j = g\left(\hat{x}_j,u_j,z_j,\hat{p}_j\right) \tag{4c}$$
$$\hat{s}_{j+1} = \hat{r}_{RNN}\left(\hat{s}_j,q_j;\Phi_j\right) \tag{4d}$$
$$\hat{p}_j = \hat{h}_{RNN}(\hat{s}_j,q_j;\Phi_j) \tag{4e}$$
$$x_{j,\min} \leq \hat{x}_j \leq x_{j,\max} \tag{4f}$$
$$q(\hat{s}_j,\hat{x}_j,u_j,z_j,\hat{p}_j,\Phi_j) \leq 0 \tag{4g}$$

where the loss $\ell_y\left(y_j,\hat{y}_j\right)$ penalizes the dissimilarity between the output measurement y_j and its estimate $\hat{y}_j$, $\ell_y : \mathbf{R}^{n_y} \times \mathbf{R}^{n_y} \to \mathbf{R}$. Equation (4f) introduces possible constraints on states derived from physics, whereas (4g) models general constraints combining states, inputs, and parameters (e.g, structural knowledge or other physical properties of the system such as conservation of energy). Solving (4) aims at maximizing the goodness of fit between (3) and (1) as measured by ℓ_y based on input-output data. Solving (4) at each time step corresponds to solving the full information problem (Rawlings et al., 2017). It is a well-known fact that as k grows, solving (4) becomes intractable; however, a meaningful way of reducing the complexity while preserving all the past information cumulated from time 0 to k is MHE. In the next section we show how MHE can be adopted to learn RNN sub-models.

3. MOVING HORIZON ESTIMATION FOR LEARNING RECURRENT NEURAL NETWORKS

We consider the following adaptation of the MHE scheme presented in (Løwenstein et al., 2023) to the combined state-estimation and network parameter learning problem posed in (4):

$$\min_{\hat{x}_{k-L},\hat{s}_{k-L},w_{k-L},\ldots,w_{k-1},\Phi} r(\mathbf{\Phi}) + \sum_{j=k-L}^{k} \left(\|V_{y,j}(y_j-\hat{y}_j)\|_2^2\right)$$
$$+\sum_{j=k-L}^{k-1}\left(\|W_{x,j}w_j\|_2^2\right) + \left\| P_{k-L}\begin{bmatrix}\hat{x}_{k-L}-\bar{x}_{k-L}\\ \hat{s}_{k-L}-\bar{s}_{k-L}\\ \Phi-\bar{\Phi}_{k-L}\end{bmatrix}\right\|_2^2 \tag{5a}$$
$$\text{s.t. } \hat{x}_{j+1} = f(\hat{x}_j,u_j,z_j,\hat{p}_j)+w_j \tag{5b}$$
$$\hat{y}_j = g\left(\hat{x}_j,u_j,z_j,\hat{p}_j\right) \tag{5c}$$
$$\hat{s}_{j+1} = \hat{r}_{RNN}\left(\hat{s}_j,\hat{x}_j,u_j,z_j;\Phi\right) \tag{5d}$$
$$\hat{p}_j = \hat{h}_{RNN}(\hat{s}_j,\hat{x}_j,u_j,z_j;\Phi) \tag{5e}$$
$$x_{j,\min} \leq \hat{x}_j \leq x_{j,\max} \tag{5f}$$
$$q(\hat{s}_j,\hat{x}_j,u_j,z_j,\hat{p}_j,\Phi) \leq 0 \tag{5g}$$

where $r(\mathbf{\Phi})$ is a term introduced to regularize the learning problem, e.g., to prevent overfitting by introducing ℓ_1 or ℓ_2-regularization, and L is the size of the estimation window. The weight matrices $V_{y,j}^TV_{y,j}$, $W_{x,j}^TW_{x,j}$ and $P_{k-L}^TP_{k-L}$ in the cost function (5a) can be interpreted as inverses of covariance matrices (Kühl et al., 2011). In particular, V_j could be seen as the inverse Cholesky factor $R_{y,j}^{-\frac{1}{2}}$ of a given covariance matrix $R_{y,j}$, and similarly $W_{x,j}$ the inverse Cholesky factor $Q_{x,j}^{-\frac{1}{2}}$ related to (1a). The term in (5a) related to P_{k-L} is the so-called *arrival cost* and has the role of carrying on the knowledge obtained by previous measurements that are not considered in the current estimation window, summarized in vectors $\bar{x}_{k-L}$, $\bar{s}_{k-L}$, $\bar{\Phi}_{k-L}$, and P_{k-L} itself. In particular, matrix $P_{k-L}^TP_{k-L}$ quantifies the trust in current estimates $\bar{x}_{k-L}$, $\bar{s}_{k-L}$ and $\bar{\Phi}_{k-L}$. A good approximation of the full-information estimation can be achieved by a suitable selection of the arrival cost (Rawlings et al., 2017). We use the efficient method proposed in (Kühl et al., 2011) (Section 2.1, Equation 7). An artificial noise variable with covariance Q_Φ related to the RNN parameters must be introduced in the arrival cost update despite the model assumption of constant RNN parameters over the estimation window to allow adaptation. The covariance Q_Φ can be considered as a tuning parameter as it impacts the values of $\bar{x}_{k-L}$, $\bar{s}_{k-L}$, $\bar{\Phi}_{k-L}$ and P_{k-L} in (5a). Contrary to hyper-parameter tuning in classical training algorithms (Kingma and Ba, 2017), the impact of Q_Φ on the learning process can be intuitively understood due to the similarity with the tuning of EKF. This stems from the fact that setting the horizon $L = 1$ in (5) and exploiting the arrival cost update from (Kühl et al., 2011) yields an MHE which is equivalent to the square-root formulation of the EKF in (Bellantoni and Dodge, 1967).

The main difference between the MHE scheme presented in (Løwenstein et al., 2023) and (5) is the inclusion of the hidden states s_j of the RNN sub-model as additional states to be estimated by the MHE, and the addition of the state disturbance variables w_j and their corresponding weighting matrices $W_{x,j}$. The addition of w_j in (5b) allows one to learn a model where (1a) is not perfectly known or subject to state noise. Formulating the learning problem in terms of an MHE scheme allows us to keep the state variables as free optimization variables and solve the problem in the so-called *non-condensed* version (a.k.a. "direct multiple shooting"), where the equality constraints of (5b) and (5d) are enforced by the solver, rather than the *condensed* problem (a.k.a. "single shooting"), where the states are explicitly accounted for in the cost and constraint functions, and thus possibly improve the numerical performance (Hicks and Ray, 1971). Further, the constraint in (5g) allows to embed additional physical knowledge in the training process, guiding the RNN parameters towards a solution that is feasible from a physical perspective.

MHE is inherently developed for online state and parameter estimation and can therefore directly be applied in an online setting to continuously adapt Φ as the system in (1) evolves over time. However, training a model prior to deployment is preferable if offline training data are available. Based on the MHE problem in (5) we propose a modified version of the training algorithm provided in (Løwenstein et al., 2023) (Section 3, Algorithm 1) tailored for learning RNNs and capable of processing multiple trajectories of training data. Given a training data set $D = \{\{(u_0,y_0),\ldots,(u_N,y_N)\}\}_{j=0}^{N_{traj}}$ where N_{traj} denotes the number of system trajectories available for offline training, we randomly split the data into a training data set $D_{train} = \{\{(u_0,y_0),\ldots,(u_N,y_N)\}\}_{j=1}^{N_T}$, a validation data set $D_{val} = \{\{(u_0,y_0),\ldots,(u_N,y_N)\}\}_{j=1}^{N_V}$, and a testing data set $D_{test} = \{\{(u_0,y_0),\ldots,(u_N,y_N)\}\}_{j=1}^{N_{test}}$, where N_T, N_V, and N_{test} denote the number of system trajectories used for training, validation, and testing respectively. N_T, N_V and N_{test} are selected such that $N_T + N_V + N_{test} = N_{traj}$. Consider again the learning problem in (4) for $k = N-1$ for all system trajectories in the training data set D_{train} and a constant parameter vector Φ (i.e., assuming that (1) does not change during data collection)

$$\min_{\hat{x}_{0,\dots,N_T},\hat{s}_{0,\dots,N_T},\{\Phi\}} \sum_{i=0}^{N_T}\sum_{j=0}^{N-1} \ell_y\left(y_j,\hat{y}_j\right) \tag{6a}$$

$$\text{s.t. } \hat{x}_{j+1} = f(\hat{x}_j,u_j,z_j,\hat{p}_j) \tag{6b}$$

$$\hat{y}_j = g\left(\hat{x}_j,u_j,z_j,\hat{p}_j\right) \tag{6c}$$

$$\hat{s}_{j+1} = \hat{r}_{RNN}\left(\hat{s}_j,\hat{x}_j,\bar{u}_j,z_j;\Phi\right) \tag{6d}$$

$$\hat{p}_j = \hat{h}_{RNN}(\hat{s}_j,\hat{x}_j,u_j,z_j;\Phi) \tag{6e}$$

$$x_{j,\min} \leq \hat{x}_j \leq x_{j,\max} \tag{6f}$$

$$q(\hat{s}_j,\hat{x}_j,u_j,z_j,\hat{p}_j,\Phi) \leq 0 \tag{6g}$$

Due to the dynamical constraints in (6b), each system trajectory must be processed in one shot if the problem is solved using different nonlinear programming solvers (Nocedal and Wright, 2006) and despite the gradient of the cost function being available via computational efficient methods, i.e., backpropagation through time (Werbos, 1990), this could lead to the vanishing or exploding gradient problem (Hochreiter, 1998).

To mitigate the aforementioned problems we propose to solve the training problem in (6) by processing the training set D_T for a number N_e of epochs by randomly selecting N_b mini-batches of length L_b which are processed sequentially using the MHE in (5). Notice that the batch length L_b must be selected such that $L_b \gg L$. More specifically, for N_b mini-batches in each epoch, a random system trajectory is selected from D_T such that $i_{T_j} \sim U[1,N_T]$, where i_{T_j} denotes the index of the randomly selected training trajectory, and the MHE (5) is used to process the data sequentially starting from $k_0 \sim U[0,N-L_b]$ to k_0+L_b where k_0 is a randomly selected starting point of the current mini-batch. To construct an initial guess of x_{k_0} and s_{k_0}, an MHE with fixed RNN parameters (i.e., the learned parameters from the previous epoch) is used to process a fixed number of data points from $k_0 - L_{init}$ to k_0 for some random initialization of x and s and thus reducing potential impact of bad initialization on the learning procedure by replacing the so-called washout period typically used when training RNNs (Jaeger, 2001). After each epoch, the part of the arrival cost related to the state estimates is reset while the knowledge obtained on the neural network parameters is kept for the next training epoch. Before starting a new epoch the covariance matrix related to the parameters of the RNN is possibly scaled as $Q_\Phi \leftarrow \alpha Q_\Phi$ with $0<\alpha<1$ to promote rapid learning in the initial epochs while enhancing fine-tuning in later epochs. This stems from the fact that the estimates can only be determined within the limits given by Q_x and Q_Φ (Kühl et al., 2011). Further, to monitor the progress of the training, the RNN parameters obtained in each epoch is used to evaluate the gray-box model on the validation set D_V. The validation is carried out by means of open-loop simulation from $k_{0,val}$ to N where $x_{k_{0,val}}$ and $s_{k_{0,val}}$ is constructed using an MHE with fixed RNN parameters. To better avoid biases towards the last batches of each epoch, we average the RNN parameters over the last epoch before carrying out the validation. We define the performance measure as the Mean Squared Error (MSE) of the measurable outputs and define the early termination criterion by $\epsilon > 0$

$$\text{MSE}_{val} = \frac{1}{N_{val}}\frac{1}{(N-k_{0,val})}\sum_{i=1}^{N_{val}}\sum_{j=k_{0,val}}^{N} \ell_y(y_{i,j},\hat{y}_{i,j}) \leq \epsilon \tag{7}$$

where the loss ℓ_y is the squared error of the output prediction. After the offline training algorithm terminates, the model is evaluated on the unseen data set D_{test} using the approach described for the validation sets. The proposed offline training method and the online adaptation are summarized in Algorithm 1, where M denotes the number of measurements processed in the online phase. To further reduce the complexity in the online phase one could adapt only a subset of the RNN parameters e.g., the bias terms or the parameters in the output layer. However, for the sake of completeness we adapt all RNN parameters in the example provided.

4. NUMERICAL EXAMPLE

To evaluate the proposed algorithm we investigate the performance on the well-known quadruple tank benchmark problem (Alvarado et al., 2011). The system consists of four tanks with water levels h_1, h_2, h_3 and h_4. The control variables are the flow rates of two pumps q_a and q_b. The flow to each tank is controlled using triple valves such that $q_1 = \gamma_a q_a$, $q_2 = \gamma_b q_b$, $q_3 = (1-\gamma_b)q_b$, and $q_4 = (1-\gamma_a)q_a$. The continuous time dynamics are governed by

$$\dot{h}_1 = -\frac{a_1}{S}\sqrt{2gh_1} + \frac{a_2}{S}\sqrt{2gh_3} + \frac{\gamma_a}{S}q_a \tag{8a}$$

$$\dot{h}_2 = -\frac{a_2}{S}\sqrt{2gh_2} + \frac{a_4}{S}\sqrt{2gh_4} + \frac{\gamma_b}{S}q_b \tag{8b}$$

$$\dot{h}_3 = -\frac{a_3}{S}\sqrt{2gh_3} + \frac{1-\gamma_b}{S}q_b \tag{8c}$$

$$\dot{h}_4 = -\frac{a_4}{S}\sqrt{2gh_4} + \frac{1-\gamma_a}{S}q_a \tag{8d}$$

where $a_1 = 1.31\cdot 10^{-4}\,m^2$, $a_2 = 1.51\cdot 10^{-4}\,m^2$, $a_3 = 9.27\cdot 10^{-5}\,m^2$, $a_4 = 8.82\cdot 10^{-5}\,m^2$, $S = 0.06\,m^2$, $\gamma_a = 0.3$, and $\gamma_b = 0.4$. To obtain a discrete-time model, we deploy an Euler scheme with $T_s = 15\,s$ such that $x_{k+1} = f(x_k,u_k) + v_x$, where v_x is a state noise with zero mean and covariance matrix $Q_x = (1\cdot 10^{-4})^2 \cdot I \in \mathbb{R}^4$. The associated output equation is

$$y_k = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}\left[h_1\ h_2\ h_3\ h_4\right]^T + v_y \tag{9}$$

where v_y is an output noise with zero mean and covariance matrix $R_y = (5\cdot 10^{-3})^2 \cdot I \in \mathbb{R}^2$. The control input, $u = \left[q_a\ q_b\right]^T$ is subject to the actuator limits

$$q_a \in \left[0, 0.9\cdot 10^{-3}\right]\ \frac{m^3}{s} \text{ and } q_b \in \left[0, 1.1\cdot 10^{-3}\right]\ \frac{m^3}{s}. \tag{10}$$

Further, the water level of each tank cannot exceed $h_{1,2}^{max} = 1.36\,m$ and $h_{3,4}^{max} = 1.3\,m$ to avoid overflow. In our example we consider tank 4 to be unknown and therefore the submodel defined by the RNN in (2a) must capture the underlying dynamics of (8d) while the output $\hat{p} \in \mathbb{R}^2$ of the RNN must recover $p_1 = \frac{a_4}{S}\sqrt{2gh_4}$ to capture the interaction with the overall system and $p_2 = h_4$ to readily apply the model in MHE and MPC for tracking of set

Algorithm 1 Physics-informed learning of NN submodel

```
1: Inputs: $Q_{\Phi 0}$, $Q_{\Phi}$, $Q_{\Phi_{online}}$, $L$, $N_e$, $N_b$, $L_b$, $L_{init}$, $\alpha$, $\epsilon$,
   $Q_x$, $Q_{x0}$, $Q_s$, $Q_{s0}$, $R$, $D = \{\{(u_k, y_k)\}_{k=0}^{N}\}_{j=0}^{N_{traj}}$;
2: Initialize:
       $\bar{\mathbf{\Phi}}_0 \leftarrow$ Xavier intialization (Glorot and Bengio, 2010)
       Randomly split $D$ into $D_T$, $D_V$ and $D_{test}$
       Covariance initialization:
       $P_0 \leftarrow \begin{bmatrix} Q_{x0}^{-1/2} & 0 & 0 \\ 0 & Q_{s0}^{-1/2} & 0 \\ 0 & 0 & Q_{\Phi 0}^{-1/2} \end{bmatrix}$;
       $V_{y,j} \leftarrow R^{-(1/2)}$, $\forall j$;
       $W_{x,j} \leftarrow Q_x^{-(1/2)}$, $\forall j$;
Offline MHE learning - $D_N \neq \emptyset$
3: for $h = 1, \dots, N_e$ do:
4:     for $j = 1, \dots, N_b$ do
5:         $i_{T_j} \sim U[1, N_T]$
6:         $k_0 \sim U[0, N - L_b]$
7:         $\bar{x}_L \leftarrow \hat{x}_{k0}$ and $\bar{s}_L \leftarrow \hat{s}_{k0}$ obtained using MHE with fixed $\Phi$;
8:         for $k = k_0, \dots, k_0 + L_b$ do:
9:             Update $\bar{x}_{k-L}$, $\bar{s}_{k-L}$, $\bar{\mathbf{\Phi}}_{k-L}$, $P_{k-L}$ according to (Kühl et al., 2011);
10:            Solve MHE problem in (5);
11:        end for
12:        Reset the MHE:
13:        $P_0 \leftarrow \begin{bmatrix} Q_{x0}^{-1/2} & 0 & 0 \\ 0 & & \\ 0 & Q_{s0}^{-1/2} & 0 \\ 0 & 0 & [P_{k-L}]_{n_x+n_s:,n_x+n_s:} \end{bmatrix}$;
14:    end for
15:    $\Phi = \frac{1}{N_b L_b} \sum_{i=1}^{N_b} \sum_{j=1}^{L_b} \Phi_{i,j}$
16:    Evaluate loss on $D_V$
17:    if MSE$_{val} \leq \epsilon$ then
18:        break;
19:    end if
20:    $Q_{\Phi} \leftarrow \alpha Q_{\Phi}$;
21: end for
22: Evaluate loss on $D_{test}$
Online MHE learning
23: $Q_{\Phi} \leftarrow Q_{\Phi_{online}}$
24: for $k = 0, \dots, M-1$ do:
25:     Update $\bar{x}_{k-L}$, $\bar{s}_{k-L}$ $\bar{\mathbf{\Phi}}_{k-L}$ and $P_{k-L}$ according to (Kühl et al., 2011);
26:     Solve MHE problem in (5);
27: end for
```

points on h_4. Therefore, we can define the gray-box model to be learned as

$$\hat{x}_{k+1} = [h_1 \;\; h_2 \;\; h_3]^T = f(\hat{x}_k, u_k, \hat{p}_k) \tag{11a}$$

where $f(\hat{x}_k, u_k, \hat{p}_k)$ is obtained by integration of (8a)-(8c), with $p_{1,k}$ replacing $\frac{a_4}{S}\sqrt{2gh_4}$ in (8b). The output equation becomes

$$\hat{y}_k = [h_2 \;\; h_4]^T = g(\hat{x}_k, u_k, \hat{p}_k) = [h_2 \;\; \hat{p}_{2,k}]^T . \tag{11b}$$

Defining $q = [q_a]$, the subsystem is governed by

$$\hat{s}_{k+1} = \hat{r}_{RNN}\left(\hat{s}_k, q_k; \Phi_k\right) \tag{11c}$$

and

$$\hat{p}_k = \hat{h}_{RNN}\left(\hat{s}_k, q_k; \Phi_k\right) . \tag{11d}$$

For the GRU we use $\hat{s} \in \mathbb{R}^2$ which results in $\Phi \in \mathbb{R}^{30}$.

Table 1. Offline training results.

Method	MSE$_y$	MSE$_p$	MSE$_{y,NLMPC}$	cost$_{\text{NLMPC}}$
MHE	$6.43 \cdot 10^{-5}$	$3.83 \cdot 10^{-5}$	0.072	0.072
EKF	$8.83 \cdot 10^{-5}$	$1.02 \cdot 10^{-5}$	0.072	0.072
Pytorch	–	$1.14 \cdot 10^{-5}$	–	–
True model	–	–	0.068	0.068

4.1 Offline training training using MHE

30 open-loop trajectories of 1500 input-output pairs are collected from random initial conditions using Multilevel-Pseudo-Random-Signals (MPRS) as proposed in (Bonassi et al., 2021). To safely collect open-loop trajectories, i.e., without violating the maximum water level, only a subset of input range given by (10) is used during data collection. The flow rates are limited to $q_a \in \left[0, 0.65 \cdot 10^{-3}\right] \frac{m^3}{s}$ and $q_b \in \left[0, 0.75 \cdot 10^{-3}\right] \frac{m^3}{s}$. Alternatively, a simple controller could be introduced in the data collection to reduce the flow rate if any of the four tanks come close to the maximum level, but instead we consider this to be learned during online adaptation with the MHE scheme and show how it is done safely without violating the physical constraints. To train the gray-box model in (11) with Algorithm 1 we use $Q_{\Phi 0} = (1 \cdot 10^{-1})^2 \cdot I \in \mathbb{R}^{30 \times 30}$, $Q_{\Phi} = (1 \cdot 10^{-3})^2 \cdot I \in \mathbb{R}^{30 \times 30}$, $L = 10$, $N_e = 25$, $N_b = 25$, $L_b = 500$, $L_{init} = 100$, $\alpha = 0.95$, $Q_x = (1 \cdot 10^{-4})^2 \cdot I \in \mathbb{R}^{3 \times 3}$, $Q_{x0} = (1 \cdot 10^{-3})^2 \cdot I \in \mathbb{R}^{3 \times 3}$, $Q_s = (1 \cdot 10^{-3})^2 \cdot I \in \mathbb{R}^{2 \times 2}$, $Q_{s0} = (1 \cdot 10^{-3})^2 \cdot I \in \mathbb{R}^{2 \times 2}$, and $R_y = (5 \cdot 10^{-3})^2 \cdot I \in \mathbb{R}^{2 \times 2}$. Further, we use $N_T = 20$, $N_V = N_{test} = 5$ and $k_{0,val} = 300$. We do not terminate the algorithm early using ϵ. For the remainder of this paper we use horizon $L = 10$ for both MHE and MPC. For comparison a GRU is trained using EKF for simultaneous state and parameter estimation by replacing the MHE in Alg. 1. The implementation is carried out in Casadi (Andersson et al., 2018) and the MHE problem in (5) is solved using Ipopt (Wächter and Biegler, 2006). Further, we train a GRU using Pytorch (Paszke et al., 2019) using Adam (Kingma and Ba, 2017) with learning rate 0.001 on the true input-output sequence subsystem, which in our setup is considered unknown, but in a simulated environment can serve as "best-case" achievable GRU and thus serve as a valuable comparison.

In Table 1 the offline training methods are compared in terms of performance on the unseen test trajectories in open-loop simulation from $k_0 = 300$ where the initial state has been reconstructed using MHE and EKF respectively. We compare the ability to reconstruct the true outputs of the subsystem, p, and the output of the full system, y. Further, we compare performance of the learned models in closed-loop MPC where the performance of MPC with the true model is reported for comparison. For the MPC we impose state constraints corresponding to the maximum water levels and inputs constraints according to (10). We penalize the deviation from the MPRS generated set point trajectory for the outputs h_2 and h_4 by $Q_{MPC} = 1 \cdot I \in \mathbb{R}^{2 \times 2}$ and the input increments $\Delta u_k = u_k - u_{k-1}$ by

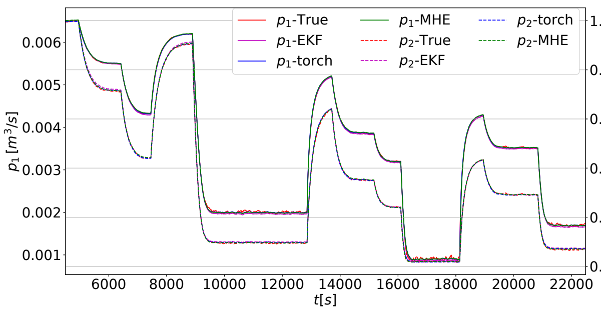


Fig. 2. The RNNs ability to reconstruct the signal p from the unknown submodel. Both the GRU learned from EKF and MHE are capable of reconstructing p. The plot shows the evaluation on a single test trajectory.

$W_{MPC} = 10^4 \cdot I \in \mathbb{R}^{2\times 2}$. The set points is generated such that the state constraints on h_1 and h_3 are active and thereby prevent perfect set-point tracking. Fig. 2 shows that both EKF and MHE are capable of learning the unknown sub-model with very high accuracy. Therefore for this specific example, the main advantage of the MHE is the possibility to impose extra constraints through (5f) and (5g). To truly understand the trade-offs between EKF and MHE for training RNNs, more complex examples should be investigated. However, we remark that we found MHE to be more robust than the EKF to different training setups. This could be explained by the fact that MHE is more robust to bad initialization (Kühl et al., 2011). Interestingly, we find that Algorithm 1 tends to generalize better and obtain satisfactory performance with less data when the trajectories are processed in random mini-batches as proposed rather than processing the full training trajectories sequentially. Despite numerous attempts we did not succeed in imposing the stability constraint proposed in (Bonassi et al., 2021), e.g., Ipopt either found the problem infeasible, failed in the restoration phase, or kept iterating without satisfying the termination criteria. In a real-time application for online adaptation such solver failures might have catastrophic consequences and imposing such a constraint may not be worth the risk. During training the proposed MHE algorithm did not encounter any issues related to instability. Further, (Mohajerin and Waslander, 2019) claim that the stability of the learned RNN is inherently chained to the stability of the system being learned, but an unstable instance of the RNN can occur during training leading to a diverging learning process. Such blow ups are more evident when relying on classical training algorithms as open-loop simulation over longer trajectories are required to evaluate the loss. In such cases the δ-ISS constraint can provide a valuable measure in the training process (Bonassi et al., 2021). On the other hand, the proposed MHE framework tries to mitigate these issues by *(i)* using a short estimation window, i.e., the dynamics will only be propagated L steps which is much shorter than the batch sizes used in conventional training algorithms; *(ii)* solving the non-condensed problem which is known to be suitable for unstable systems and serves to avoid gradient "blow up"; *(iii)* embedding physical knowledge to guide the training, i.e., by letting the gray-box model inherit the stability properties of the known part of the system.

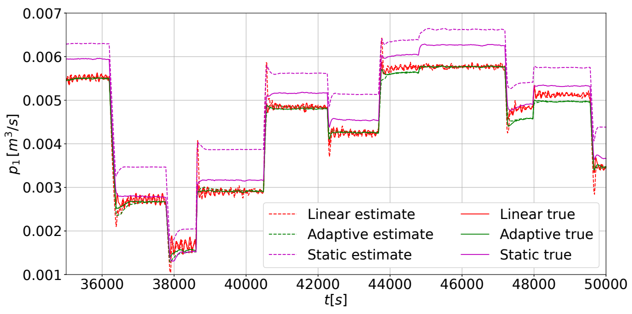
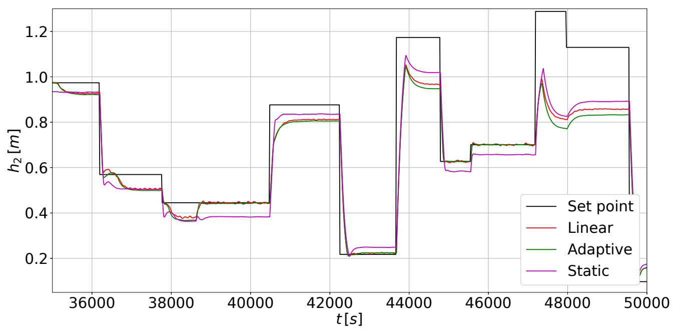
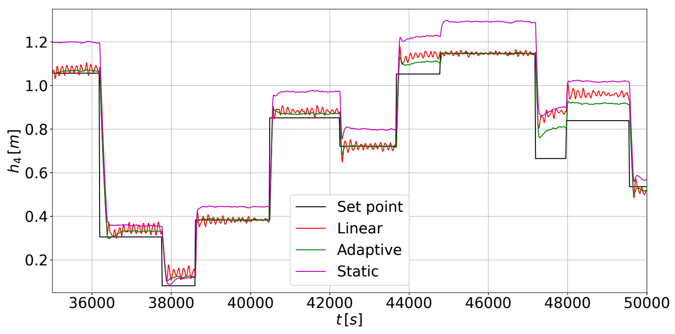


Fig. 3. Estimation of p_1 online. Dashed lines show the estimate while the solid lines are the true values. The figure shows how the adaptive MHE scheme is able to cope with part-to-part variations.



(a) Closed-loop trajectory of h_2.



(b) Closed-loop trajectory of h_4.

Fig. 4. Trajectories of the outputs tracked by the NMPC.

4.2 Online adaptation and nonlinear MPC

In this section we show how the model learned in Section 4.1 is adapted online using the MHE scheme in (5). We consider the scenario of part-to-part variations, i.e., the system that the MHE scheme is applied to differs from the system in Section 4.1: $a_4 = 5.82 \cdot 10^{-5}$ and $p_1 = \frac{a_4}{S}\sqrt{2g(h_4 + dh_4^2)}$ where $d = 0.5$. We use $Q_{\Phi online} = (1 \cdot 10^{-4})^2 \cdot I \in \mathbb{R}^{30\times 30}$. For comparison we also show the performance of the model learned in Section 4.1 without online adaptation. As a final comparison, we use a linear model where $\hat{p}_1 = c_1 h_4$ and $\dot{h}_4 = c_2 h_4 + c_3 q_a$ where the parameters, c_1, c_2 and c_3, are estimated online using a standard MHE-formulation (Kühl et al., 2011).

Fig. 3 and Fig. 4 show how the adaptive MHE (5) is capable of adjusting to the unknown changes of the system. The MSE_p and $\text{MSE}_{y,NLMPC}$ for the adaptive, static and linear model are shown in Table 2. Namely, the adaptive

Table 2. Results in online adaptation.

Method	MSE_p	$\mathrm{MSE}_{y,NLMPC}$
MHE adaptive	$5.1 \cdot 10^{-5}$	0.068
Static	$1.1 \cdot 10^{-2}$	0.078
Linear model	$1.0 \cdot 10^{-2}$	0.068

models (RNN and linear model) performs 14% better than the static model in closed-loop MPC. At first glance, the performance of the linear model looks promising, but from Fig. 4b it is evident that the learning of such a simplified linear model leads to oscillatory behavior which might be critical depending on the use-case. The adaptive gray-box model is superior at estimating p and the MSE_p is three orders of magnitude lower than both the static and linear model. In this work we have considered part-to-part variations which could also imitate rapid changing system parameters, however, it is expected that the proposed method will perform equally well with slowly varying system parameters.

5. CONCLUSION AND FUTURE WORK

This paper proposed an MHE for learning and adapting RNN structures exemplified with a GRU. By using RNNs the proposed method can learn state-full sub-models and thereby embed memory of the past history of the system, contrary to classical feedforward neural networks. Our method is inherently suitable for online adaptation and enhanced further by the physics-informed model embedding the RNN. The MHE can also be used offline to reconstruct a model from a given set of noisy input/output data available prior to deployment in an online setting. One key feature is the ability to process the data in short batches due to the preservation of knowledge through the arrival cost, thereby avoiding long forward simulations of the RNN which could lead to problems during training. Further, the learning process can be enhanced through the natural ability of MHE to handle constraints on the estimated quantities. The work could readily be extended to more complex or slightly unstable systems to see if the proposed method is capable of learning when classical training methods tend to fail. A more theoretical line of research is the derivation of less conservative stability conditions that are well-suited for online implementation. Further, a theoretical sound scaling of the parameter covariance correlated with the model fit would improve the proposed algorithm. Lastly, a tailored implementation based on a real-time-iteration scheme would enhance the online applicability of the proposed method.

ACKNOWLEDGEMENTS

This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No. 953348 (ELO-X).

REFERENCES

Alvarado, I., Limon, D., Muñoz de la Peña, D., Maestre, J., Ridao, M., Scheu, H., Marquardt, W., Negenborn, R., De Schutter, B., Valencia, F., and Espinosa, J. (2011). A comparative analysis of distributed MPC techniques applied to the HD–MPC four-tank benchmark. *Journal of Process Control*, 21(5), 800–815. Special Issue on Hierarchical and Distributed Model Predictive Control.

Andersson, J.A.E., Gillis, J., Horn, G., Rawlings, J.B., and Diehl, M. (2018). CasADi – A software framework for nonlinear optimization and optimal control. *Mathematical Programming Computation*.

Bellantoni, J.F. and Dodge, K.W. (1967). A square root formulation of the Kalman-Schmidt filter. *AIAA Journal*, 5(7), 1309–1314.

Bemporad, A. (2023). Recurrent neural network training with convex loss and regularization functions by extended Kalman filtering. *IEEE Transactions on Automatic Control*, 68(9), 5661–5668.

Bonassi, F., Farina, M., and Scattolini, R. (2021). On the stability properties of gated recurrent units neural networks. *Systems and Control Letters*, 157, 105049.

Bonassi, F., Farina, M., Xie, J., and Scattolini, R. (2022). On recurrent neural networks for learning-based control: Recent results and ideas for future developments. *Journal of Process Control*, 114, 92–104.

Brunke, L., Greeff, M., Hall, A.W., Yuan, Z., Zhou, S., Panerati, J., and Schoellig, A.P. (2022). Safe learning in robotics: From learning-based control to safe reinforcement learning. *Annual Review of Control, Robotics, and Autonomous Systems*, 5(1), 411–444.

Cho, K., van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representations using RNN encoder–decoder for statistical machine translation. In A. Moschitti, B. Pang, and W. Daelemans (eds.), *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 1724–1734. Association for Computational Linguistics, Doha, Qatar.

Chung, J., Gulcehre, C., Cho, K., and Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling.

Draeger, A., Engell, S., and Ranke, H. (1995). Model predictive control using neural networks. *IEEE Control Systems Magazine*, 15(5), 61–66.

Glorot, X. and Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In *International Conference on Artificial Intelligence and Statistics*.

Hewing, L., Wabersich, K.P., Menner, M., and Zeilinger, M.N. (2020). Learning-based model predictive control: Toward safe learning in control. *Annual Review of Control, Robotics, and Autonomous Systems*, 3(1), 269–296.

Hicks, G.A. and Ray, W.H. (1971). Approximation methods for optimal control synthesis. *The Canadian Journal of Chemical Engineering*, 49(4), 522–528.

Hochreiter, S. (1998). The vanishing gradient problem during learning recurrent neural nets and problem solutions. *Int. J. Uncertain. Fuzziness Knowl.-Based Syst.*, 6(2), 107–116.

Hochreiter, S. and Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*, 9(8), 1735–1780.

Hou, Z.S. and Wang, Z. (2013). From model-based control to data-driven control: Survey, classification and perspective. *Information Sciences*, 235, 3–35. Data-based Control, Decision, Scheduling and Fault Diagnostics.

Jaeger, H. (2001). The "echo state" approach to analysing and training recurrent neural networks. GMD Report 148, GMD - German National Research Institute for Computer Science.

Karniadakis, G.E., Kevrekidis, I.G., Lu, L., Perdikaris, P., Wang, S., and Yang, L. (2021). Physics-informed machine learning. *Nature Reviews Physics*, 3(6), 422–440.

Kingma, D.P. and Ba, J. (2017). Adam: A method for stochastic optimization.

Kühl, P., Diehl, M., Kraus, T., Schlöder, J.P., and Bock, H.G. (2011). A real-time algorithm for moving horizon state and parameter estimation. *Comput. Chem. Eng.*, 35, 71–83.

Lanzetti, N., Lian, Y.Z., Cortinovis, A., Dominguez, L., Mercangöz, M., and Jones, C. (2019). Recurrent neural network based MPC for process industries. In *2019 18th European Control Conference (ECC)*, 1005–1010.

Løwenstein, K.F., Bernardini, D., Fagiano, L., and Bemporad, A. (2023). Physics-informed online learning of gray-box models by moving horizon estimation. *European Journal of Control*, 74, 100861.

Maiworm, M., Limon, D., and Findeisen, R. (2021). Online learning-based model predictive control with gaussian process models and stability guarantees. *International Journal of Robust and Nonlinear Control*, 31(18), 8785–8812.

Masti, D. and Bemporad, A. (2021). Learning nonlinear state-space models using autoencoders. *Automatica*, 129, 109666.

Mohajerin, N. and Waslander, S.L. (2019). Multistep prediction of dynamic systems with recurrent neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 30(11), 3370–3383.

Nguyen-Tuong, D. and Peters, J. (2011). Model learning for robot control: A survey. *Cognitive processing*, 12, 319–40.

Nocedal, J. and Wright, S.J. (2006). *Numerical Optimization*. Springer, New York, NY, USA, 2e edition.

Pan, Y. and Wang, J. (2008). Nonlinear model predictive control using a recurrent neural network. In *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*, 2296–2301.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, 8024–8035. Curran Associates, Inc.

Rawlings, J., Mayne, D., and Diehl, M. (2017). *Model Predictive Control: Theory, Computation, and Design*. Nob Hill Publishing.

Taylor, A.T., Berrueta, T.A., and Murphey, T.D. (2021). Active learning in robotics: A review of control principles. *Mechatronics*, 77, 102576.

Thrun, S. and Mitchell, T.M. (1995). Lifelong robot learning. *Robotics and Autonomous Systems*, 15(1), 25–46. The Biology and Technology of Intelligent Autonomous Agents.

Wächter, A. and Biegler, L.T. (2006). On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*.

Werbos, P. (1990). Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10), 1550–1560.