European Journal of Control 74 (2023) 100861

Contents lists available at ScienceDirect

European Journal of Control

journal homepage: www.elsevier.com/locate/ejcon

Physics-informed online learning of gray-box models by moving horizon estimation☆

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

[a] Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, Via Ponzio 34/5, Milano, 20133, Italy
[b] ODYS S.r.l, Via Pastrengo 14, Milano, 20159, Italy
[c] IMT School for Advanced Studies, Piazza San Francesco 19, Lucca, 55100, Italy

ARTICLE INFO

Article history:
Received 15 May 2023
Accepted 8 June 2023
Available online 3 July 2023

Recommended by Prof. T Parisini

Keywords:
Gray-box modeling
Physics-informed learning
Learning-based MPC
Moving horizon estimation
Online neural network adaptation

ABSTRACT

A simple yet expressive prediction model is an essential ingredient in model-based control and estimation. Models derived from fundamental physical principles may fail to capture the complexity of the actual system dynamics. A potential solution is the use of a *physics-informed*, or *gray-box* model that extends a physics-based model with a data-driven part. Learning the latter might be challenging, due to noisy measurements and lack of full state information. This work presents a method based on Moving Horizon Estimation (MHE) for simultaneous state estimation and training of a black-box submodel, such as a neural network. The method can be used in offline training or applied online for adaptation without any prior knowledge than the white-box submodel. We analyze the capabilities of the method in a two degree of freedom robotic manipulator case study, also showing how it can be used for online adaptation to cope with a time-varying model mismatch.

1. Introduction

In model-based control and estimation, such as Model Predictive Control (MPC) and Extended Kalman Filtering (EKF), a fundamental component is a model of the system dynamics that is accurate, yet as simple as possible. In industrial systems it is often hard to capture the real-world dynamics exactly by using physical principles only, due to three types of issues: (*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. Due to increasing closed-loop performance and estimation quality requirements, addressing these issues becomes even more pressing. One potential solution that has attracted great attention, due to the success of machine learning and increasing computational resources, is learning-based control, see [5,11,14]. Learning-based control incorporates principles from machine learning in the control strategy, and the use of machine learning ideas in control-oriented modeling has been investigated in the past [8,18,22]. Offline learning can be used to address issue (*i*) and (*ii*), and ideas have been proposed to cope with issue (*iii*), see [3,17,19] and is what is often called *lifelong learning*. For example, lifelong learning in robotics has been a topic of research for decades [27] and is still active today [26]. Many ideas rely on the use of black-box models, that is, disregard all physical knowledge and rely only on models derived from data [1,9,19]. However, a more appealing approach is the use of physics-informed learning, e.g. gray-box modeling where available knowledge from physical principles is preserved and extended with a data-driven component [15,25]. Recently, non-parametric structures such as Gaussian processes have gained attention and have been used widely in gray-box models [6,17,19]. Despite their many advantages, their applicability is often restricted by their limited scalability.

In this paper, we propose a framework in which parametric function approximations of the data-driven component are used. The proposed state and parameter estimation approach consists of an optimization-based training algorithm based on a Moving Horizon Estimation (MHE) concept. In MHE, an optimization problem is solved over a window covering a limited number of past measurements, which moves over time in a receding horizon fashion [23,24]. Based on such a window of past measurements, MHE can be used to estimate both the current state and parameter

☆ 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).

* Corresponding author.
E-mail addresses: kristofferfink.loewenstein@polimi.it (K.F. Løwenstein), daniele.bernardini@odys.it (D. Bernardini), lorenzo.fagiano@polimi.it (L. Fagiano), alberto.bemporad@imtlucca.it (A. Bemporad).

https://doi.org/10.1016/j.ejcon.2023.100861

values. A core advantage of MHE is the possibility to add constraints in a natural way, which is not possible in classical estimation methods based on recursive formulations such as EKF. Furthermore, MHE is more robust to inaccurate initial conditions than EKF and, by suitable numerical implementations, MHE is often feasible for embedded real-time applications [16]. A key feature of MHE is its ability to summarize and preserve existing knowledge through the so-called *arrival cost*, which makes it quite suitable for model adaptation.

A MHE-based approach for online adaptation of learning-based models was proposed in [4]. However, a fixed learning-rate was used to adapt the model, thus not relying on prior information through the arrival cost. Moreover, the authors assume that all the states are measurable and only consider the noise-free case. Since they rely on black-box modeling, the online optimization problem becomes hard to solve in a real-time setting, due to the non-negligible size of the neural network that is needed to capture the entire model for systems with complex nonlinearities.

The main contribution of this paper is to introduce the described framework for concurrent state estimation and learning of the black-box submodel. The method can directly deal with noisy data and problems where not all the states are measured and is inherently suited for safe online learning due MHE-framework used. Furthermore, preserving the physical knowledge allows one to use function approximations of rather limited size, thereby enhancing the potentials for real-time use.

The paper is organized as follows. Section 2 introduces the learning problem and formulates the MHE-based learning framework based on neural networks to model the "black-box" part of the system. Section 3 presents the algorithm for learning gray-box models by MHE. 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 (partially unknown) discrete-time dynamics

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

where $x \in \mathbb{R}^{n_x}$ is the state vector, $u \in \mathbb{R}^{n_u}$ the control input, $p \in \mathbb{R}^{n_p}$ is a vector of parameters that are a function of z_k as well as the time k where $z_k \in \mathbb{R}^{n_z}$ may collect states, inputs, and other exogenous signals to model degradation and environmental effects that can change over time, and v_x is a state-noise term that we assume normally distributed with zero-mean and covariance Q_x. The index k denotes the sampling instant. The (partially unknown) output equation of the system is defined as

$$y_k = g(x_k, p(z_k, k)) + v_{y,k} \tag{1b}$$

where $y \in \mathbb{R}^{n_y}$ is the output vector and v_y the measurement noise, that we assume normally distributed with zero-mean and covariance R. We represent the unknown part of the model as $p : \mathbb{R}^{n_z} \times \mathbb{Z} \to \mathbb{R}^{n_p}$ in (1).

The fundamental challenge is to derive a model of the unknown $p(z_k, k)$ and adapt it as the system evolves. Such a setup fits well to many physics-informed models combining ideas from physics-based modeling and data-driven approaches, for example in the (quite common) case of known integrators combined with static nonlinearities. Accordingly, the prediction model used in this work is defined as

$$\hat{x}_{k+1} = \hat{f}\big(\hat{x}_k, u_k, f_{NN}(\mathbf{w}_k, z_k)\big) \tag{2a}$$

with the associated output equation

$$\hat{y}_k = \hat{g}\big(\hat{x}_k, f_{NN}(\mathbf{w}_k, z_k)\big) \tag{2b}$$

where $f_{NN}(\mathbf{w}_k, z_k)$ is a feedforward neural network (NN) with n_L layers described by

$$\begin{aligned} v_0 &= z \\ v_1 &= A_1 v_0 + b_1 \\ v_2 &= A_2 f_1(v_1) + b_2 \\ &\vdots \\ v_{n_L-1} &= A_{n_L-2} f_{n_L-2}(v_{n_L-2}) + b_{n_L-2} \\ p &= v_{n_L-1} \end{aligned} \tag{3}$$

In (2), vector $\mathbf{w}_k$ collects all the weights A_i and bias terms b_i appearing in (3), $A_i \in \mathbb{R}^{n_i \times n_{i-1}}$, with n_i indicating the number of neurons in ith layer of the network, defining the weights, $b_i \in \mathbb{R}^{n_i}$ defining the bias of the ith layer. v_i is the vector containing the outputs of neurons of the ith layer, and f_i the activation function used in the ith layer. The hyper-parameters describing the NN model structure, once the inputs and outputs of the neural network have been selected, are $\{n_i\}$, n_L, and $\{f_i\}$. The subscript on $\mathbf{w}_k$ indicates that the NN parameters might change over time to capture the time dependence in $p(z_k, k)$.

The NN f_{NN} shall capture the function of $p(z_k, k)$ that, besides physical parameters, could also represent disturbances and model mismatches between the true dynamics f, g and their model $\hat{f}, \hat{g}$.

Neural networks are well known to be universal function approximators, and indeed a NN with two nonlinear layers can approximate any arbitrary nonlinear function [28]. Note that the choice of a feedforward NN is without loss of generality in our approach: other black-box models for p could be adopted, such as polynomial functions and other universal function approximators, without substantial changes.

The main challenge of this paper is to develop a method to make the gray-box model in (2) matching (1) as much as possible, where the goodness of such a matching is measured by the difference between y_k and $\hat{y}_k$ based on input and output data. More formally, considering the formulation of the gray-box model defined by (2) and the fact that only noisy input/output data are available, the learning problem can be stated as

$$\min_{\hat{x}_0, \{\mathbf{w}_j\}} \sum_{j=0}^{k} \ell\big(y_j, \hat{y}_j\big) \tag{4a}$$

$$\text{s.t. } \hat{x}_{j+1} = \hat{f}\big(\hat{x}_j, u_j, \hat{p}_j\big) \tag{4b}$$

$$\hat{y}_j = \hat{g}\big(\hat{x}_j, \hat{p}_j\big) \tag{4c}$$

$$\hat{p}_j = f_{NN}\big(\mathbf{w}_j, \hat{z}_j\big) \tag{4d}$$

$$x_{j,\min} \le \hat{x}_j \le x_{j,\max} \tag{4e}$$

$$p_{j,\min} \le f_{NN}(\mathbf{w}_j, \hat{z}_j) \le p_{j,\max} \tag{4f}$$

where the loss $\ell(y_j, \hat{y}_j)$ penalizes the dissimilarity between the output measurement y_j and its estimate $\hat{y}_j$, $\ell : \mathbf{R}^{n_y} \times \mathbf{R}^{n_y} \to \mathbf{R}$. Eqs. (4e) and (4f) introduce possible constraints on states and parameters derived from physics. The time dependence in $\mathbf{w}_j$ is introduced to model the fact that the relation between p_j and z_j may vary over time as well. Solving (4) corresponds to solving the full information estimation problem at each time step, which becomes intractable as k grows. As we show in the next section, MHE provides a meaningful way of reducing such a complexity while preserving all the past information cumulated between time 0 and k.

3. Moving horizon estimation for neural network learning

We consider the following adaptation of the MHE scheme [16] to the combined state-estimation and network parameter learning problem

$$\min_{\hat{x}_{k-L},\mathbf{w}} r(\mathbf{w}) + \sum_{j=k-L}^{k} \|V_j(y_j - \hat{y}_j)\|_2^2 + \left\| P_{k-L} \begin{bmatrix} \hat{x}_{k-L} - \bar{x}_{k-L} \\ \mathbf{w} - \bar{\mathbf{w}}_{k-L} \end{bmatrix} \right\|_2^2 \tag{5a}$$

$$\text{s.t. } \hat{x}_{j+1} = \hat{f}(\hat{x}_j, u_j, \hat{p}_j),\ j = k-L, \dots, k-1 \tag{5b}$$

$$\text{s.t. } \hat{y}_j = g(\hat{x}_j, \hat{p}_j),\ j = k-L, \dots, k \tag{5c}$$

$$\text{s.t. } \hat{p}_j = f_{NN}(\mathbf{w}, \hat{z}_j),\ j = k-L, \dots, k \tag{5d}$$

$$\text{s.t. } x_{j,\min} \le \hat{x}_j \le x_{j,\max},\ j = k-L, \dots, k \tag{5e}$$

$$\text{s.t. } p_{j,\min} \le f_{NN}(\mathbf{w}, \hat{z}_j) \le p_{j,\max},\ j = k-L, \dots, k \tag{5f}$$

where $r(\mathbf{w})$ is a term introduced to regularize the learning problem, e.g. to prevent overfitting by introducing ℓ_1 or ℓ_2-regularization, $\hat{z}_j$ may collect $\hat{x}_j$, u_j, and other measured exogenous signals, and L is the size of the estimation window.

The weight matrices $V_j^T V_j$ and $P_{k-L}^T P_{k-L}$ in the cost function (5a) can be interpreted as inverses of covariance matrices [16], in particular V_j could be seen as the inverse Cholesky factor $R_j^{-\frac{1}{2}}$ of a given covariance matrix R_j. The term in (5a) related to P_{k-L} is the so-called *arrival cost* and has the role of preserving the knowledge obtained by previous measurements that are not considered in the current estimation window, that are summarized in vectors $\bar{x}_{k-L}$, $\bar{p}_{k-L}$, and P_{k-L} itself. In particular, matrix $P_{k-L}^T P_{k-L}$ quantifies the trust in current estimates $\bar{x}_{k-L}$, $\bar{p}_{k-L}$. By a good selection of the arrival cost, one can get a good approximation of the full-information estimation [24]. We use the efficient method proposed in [16] (Section 2.1 Equation (7)) to update the arrival cost. To enable adaptation of NN parameters, a related noise variable is employed in the arrival cost update despite the model assumption of constant NN parameters over the estimation window in (5) and the selected parameter covariance, Q_w can be considered as a tuning parameter. By setting the horizon $L = 1$ in (5) and using the arrival cost update from [16], the MHE is equivalent to the square-root formulation of the EKF [2].

For comparison, we will also consider the following standard MHE formulation from [16]

$$\min_{\hat{x}_{k-L},\hat{p}} \sum_{j=k-L}^{k} \|V_j(y_j - \hat{y}_j)\|_2^2 + \left\| P_{k-L} \begin{bmatrix} \hat{x}_{k-L} - \bar{x}_{k-L} \\ \hat{p} - \bar{p}_{k-L} \end{bmatrix} \right\|_2^2 \tag{6a}$$

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

$$\hat{y}_j = \hat{g}(\hat{x}_j),\ j = k-L, \dots, k \tag{6c}$$

$$x_{j,\min} \le \hat{x}_j \le x_{j,\max},\ j = k-L, \dots, k \tag{6d}$$

$$p_{\min} \le \hat{p} \le p_{\max} \tag{6e}$$

that attempts to directly estimate vector p without associating an explicit model to it.

To possibly increase the performance of the numerical solution to (5) and (6), the MHE problems are solved in the so-called *non-condensed* version (a.k.a. "direct multiple shooting") where all state variables are free and the equality constraints (5b) and (6b) are enforced by the solver rather than explicitly taken into account in the cost and constraint functions i.e., the *condensed* the problem formulation (a.k.a. "direct single shooting") [12]. The formulation presented in (5) is only considering neural network parameters, but could easily be extended to estimate slowly-varying parameters.

In our setup (5), the part of the arrival cost related to the neural network parameters $\mathbf{w}$ can be seen as an adaptive learning-rate. It can be tuned by setting the covariance on the neural network parameters, Q_w, in the arrival cost update. Note also that with constraint (5f) one can attempt to explicitly constrain the learned model in a way that is meaningful from the point of view of the physics of the system and the formulation in (5) could easily be extended to explicitly constrain the neural network parameters.

Since MHE is a method developed for online state and parameter estimation, the formulation in (5) can be directly applied in an online setting to continuously adapt $\mathbf{w}$ in (2) to changes in the way p depends on z_k. The method can even be cold started without a pre-trained neural network and rely on the knowledge contained in the white-box model while adapting to the unknown model mismatch online. In case of online learning, using a pre-trained neural network is preferable if training data are available prior to deployment. In the following, we present how the MHE scheme can be used as an offline training algorithm to obtain such a network. Given a training dataset $D_N = \{(u_0, y_0) \dots (u_{N-1}, y_{N-1})\}$ of N input/output pairs, consider again Problem (4) for $k = N-1$ and constant parameter vector $\mathbf{w}$

$$\min_{\hat{x}_0,\mathbf{w}} \sum_{k=0}^{N-1} \ell\big(y_k, \hat{y}_k\big) \tag{7a}$$

$$\text{s.t. } \hat{x}_{k+1} = \hat{f}(\hat{x}_k, u_k, \hat{p}_k) \tag{7b}$$

$$\hat{y}_k = \hat{g}\big(\hat{x}_k, \hat{p}_k\big) \tag{7c}$$

$$\hat{p}_k = f_{NN}(\mathbf{w}, z_k) \tag{7d}$$

$$x_{k,\min} \le \hat{x}_k \le x_{k,\max} \tag{7e}$$

$$p_{k,\min} \le f_{NN}(\mathbf{w}, z_k) \le p_{k,\max} \tag{7f}$$

Such a reduced problem can be solved using different nonlinear programming solvers [20], but requires processing the entire data set in one shot, as the cost function of the problem is not separable due to the dynamic constraints in (7b). The gradient of the cost function can be evaluated efficiently by backpropagation through time [29], although this could lead to the problem of vanishing or exploding gradients [13].

We propose to solve the training problem (7) by processing the training dataset D_N for a number N_e of epochs using the MHE formulation (5). In each epoch, the data is processed sequentially by the MHE (5); 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, that is

$$P_0 \leftarrow \begin{bmatrix} Q_{x0}^{-1/2} & 0 \\ 0 & [P_{k-L}]_{nx:,nx:} \end{bmatrix} \tag{8}$$

The approach can be interpreted as a mini-batch algorithm with batches of length L for learning a recurrent model with hidden

states, where the loss function used in each batch is updated recursively.

Before starting a new epoch, the covariance matrix associated with the neural network parameters is possibly scaled as $Q_w \leftarrow \alpha Q_w$, with $0 \leq \alpha \leq 1$. After the N_e epochs have been processed, the average of $\{\mathbf{w}_k\}$ obtained over the last epoch is taken as initial parameters for f_{NN}, which avoids possible biases of the network towards better explaining the last part of the training data set D_N. The proposed method is summarized in Algorithm 1 where M denotes the number of measurements processed in the online phase.

Algorithm 1 Physics-informed learning of NN submodel.

Inputs: Q_{w0}, Q_w, $Q_{w_{online}}$, L, N_e, ϵ, Q_x, Q_{x0}, R, $\bar{x}_0$, $D_N = \{(u_k, y_k)\}_{k=0}^{N-1}$;

Initialize:

Xavier initialization [10]:

$A_i \sim U[-\frac{\sqrt{6}}{\sqrt{n_i+n_{i+1}}}, \frac{\sqrt{6}}{\sqrt{n_i+n_{i+1}}}]$, $b_i = 0$;

$\bar{\mathbf{w}}_0 \leftarrow A_i, b_i,\ i = 1, \ldots, n_L$;

Covariance initialization:

$$P_0 \leftarrow \begin{bmatrix} Q_{x0}^{-1/2} & 0 \\ 0 & Q_{w0}^{-1/2} \end{bmatrix};$$

$V_j \leftarrow R^{-(1/2)}, \forall j$;

Offline MHE learning - $D_N \neq \emptyset$

for $h = 1, \ldots, N_e$ **do**:
  for $k = L, \ldots, N-1$ **do**:
    Update $\bar{x}_{k-L}$, $\bar{\mathbf{w}}_{k-L}$, P_{k-L}according to [19];
    Solve MHE problem in (5);
  end for
  $\text{MSE}_y = \frac{1}{N}\|y - \hat{y}\|_2^2$;
  if $\text{MSE}_y \leq \epsilon$ **then**
    Break;
  end if
  Reset the MHE:
  $\bar{x}_L \leftarrow x_0$;
  $P_0 \leftarrow \begin{bmatrix} Q_{x0}^{-1/2} & 0 \\ 0 & [P_{k-L}]_{nx:,nx:} \end{bmatrix}$;
  $Q_w \leftarrow \alpha Q_w$;
end for
$\mathbf{w} \leftarrow \frac{1}{N}\sum_{k=0}^{N-1} \mathbf{w}_k$.

Online MHE learning

$Q_w \leftarrow Q_{w_{online}}$
for $k = 0, \ldots, M-1$ **do**:
  Update $\bar{x}_{k-L}$, $\bar{\mathbf{w}}_{k-L}$ andP_{k-L} according to [19];
  Solve MHE problem in (5);
end for

To monitor the progress during training, the Mean Square Error (MSE) on the output estimates obtained at each stage during training using $\mathbf{w}_k$ is evaluated. This is used to possibly stop the algorithm early, before the maximum number of epochs has been reached in case $\text{MSE}_y \leq \epsilon$, where $\epsilon > 0$ is a prescribed tolerance. The initial state can be obtained from prior knowledge or estimated using the white-box model.

4. Numerical example

We apply the proposed training algorithm to learn a model of a two degree of freedom robotic manipulator on simulated data. The robotic manipulator can be seen in Fig. 1. Algorithm 1 is first used to train a gray-box model, showing its estimation and prediction capabilities. Using the pre-trained neural network, we also show how the MHE in (5) can be used for online adaptation.

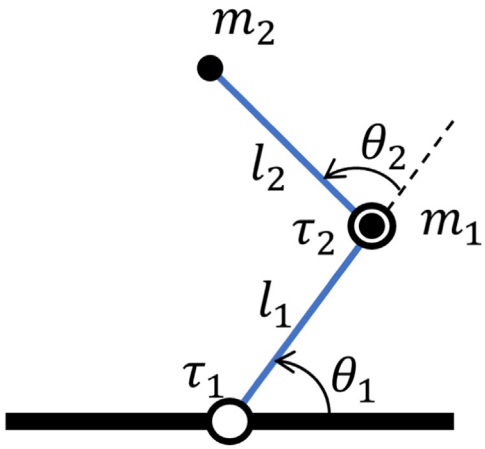


Fig. 1. Two degree of freedom robotic manipulator used in numerical example.

4.1. Dynamics

The equation of motion is derived based on the recursive Newton-Euler equations [7, Chapter 6.5]

$$\ddot{\theta} = M^{-1}(\theta)\left(\tau - \mathbf{V}(\theta, \dot{\theta}) - \mathbf{G}(\theta) - \left(\bar{c} + c_d(\theta, \dot{\theta})\right)\dot{\theta}\right) \tag{9}$$

where $M(\theta)$ is the mass matrix

$$M(\theta) = \begin{bmatrix} l_2^2 m_2 + 2 l_1 l_2 m_2 \cos(\theta_2) & l_2^2 m_2 \\ +l_1^2(m_1 + m_2) & +l_1 l_2 m_2 \cos(\theta_2) \\ l_2^2 m_2 + l_1 l_2 m_2 \cos(\theta_2) & l_2^2 m_2 \end{bmatrix} \tag{10}$$

$\mathbf{V}(\theta, \dot{\theta})$ is the vector collecting the velocity terms

$$\mathbf{V}(\theta, \dot{\theta}) = \begin{bmatrix} -m_2 l_1 l_2 \sin(\theta_2)\dot{\theta}_2^2 - 2 m_2 l_1 l_2 \sin(\theta_2)\dot{\theta}_1\dot{\theta}_2 \\ m_2 l_1 l_2 \sin(\theta_2)\dot{\theta}_1^2 \end{bmatrix} \tag{11}$$

$\mathbf{G}(\theta)$ is the vector collecting all terms containing the gravitational constant, g

$$\mathbf{G}(\theta) = \begin{bmatrix} m_2 l_2 g \cos(\theta_1 + \theta_2) + (m_1 + m_2) l_1 g \cos(\theta_1) \\ m_2 l_2 g \cos(\theta_1 + \theta_2) \end{bmatrix} \tag{12}$$

$\bar{c}$ is the nominal damping, c_d is a nonlinear damping acting on the joints, m_1 and m_2 are the joint masses, and l_1 and l_2 denotes the length of the joints (see Fig. 1). The state vector consists of the vectors θ of joint angles and $\dot{\theta}$ of joint velocities

$$x = \begin{bmatrix} \theta_1 & \theta_2 & \dot{\theta}_1 & \dot{\theta}_2 \end{bmatrix}^T \tag{13}$$

The dynamics of the system can be expressed by the system of ODE's

$$\dot{x} = f(x, u) + v_x = \begin{bmatrix} \dot{\theta} \\ \ddot{\theta} \end{bmatrix} + v_x \tag{14}$$

where v_x is a state noise term with zero mean and covariance matrix Q_x, while $\ddot{\theta}$ is computed from (9). The measured output of the system is the vector of joint angles

$$y = g(x) + v_y = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} x + v_y \tag{15}$$

where v_y is an output noise term with zero mean and covariance R. The dynamics are discretized using a fourth-order Runge-Kutta scheme with sampling time $T_s = 0.05s$. In our numerical experiments we set the covariance matrices $Q_x = 0.001^2 I \in \mathbf{R}^{4\times4}$ and $R = 0.005^2 I \in \mathbf{R}^{2\times2}$. We also set

$$c_d(x) = c_1 \sin(\theta_1) + c_2\theta_2 + c_3\dot{\theta}_1 + c_4\dot{\theta}_2 \tag{16}$$

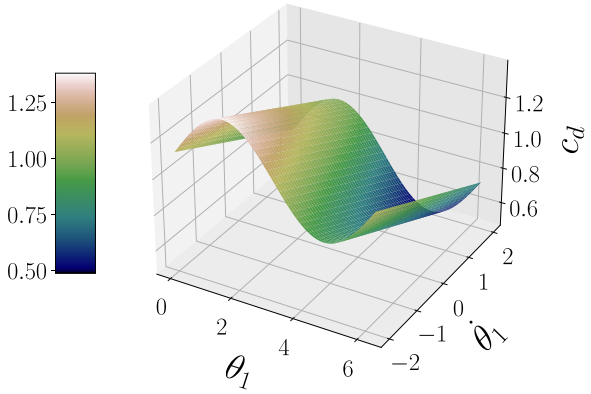


Fig. 2. Illustration of the (unknown) nonlinear damping, c_d, for fixed values of $\theta_2 = \pi$ and $\dot{\theta}_2 = -1.5$.

where $c_1 = 0.25\frac{Nm \cdot s}{rad}$, $c_2 = 0.25\frac{Nm \cdot s}{rad^2}$, $c_3 = -0.1\frac{Nm \cdot s^2}{rad^2}$, and $c_4 = -0.1\frac{Nm \cdot s^2}{rad^2}$. Eq. (16) is unknown during training and c_d acts as p in (1). The nonlinear damping is illustrated in Fig. 2 for a fixed values of θ and $\dot{\theta}_2$. Furthermore, we use $\bar{c} = 5\frac{Nm \cdot s}{rad}$, the joint masses $m_1 = m_2 = 2.5kg$, and joint lengths $l_1 = 0.5m$ and $l_2 = 0.25m$. For a more detailed understanding of the dynamics in (9), see [7, Chapter 6.5–8].

4.2. White-box model for data collection

We consider a white-box model of the dynamical equations in (9), in which the nominal damping $\bar{c}$ is used while the unknown damping caused by c_d is neglected, for controlling the system while generating the data. Hence, the dynamics of the white-box model are

$$\ddot{\theta} = M^{-1}(\theta)\left(\tau - \mathbf{V}(\theta, \dot{\theta}) - \mathbf{G}(\theta) - \bar{c}\dot{\theta}\right) \tag{17}$$

4.3. Gray-box model for estimation and control

Regarding the gray-box model, we introduce a neural network which takes the state vector as the input and gives an estimate of c_d as the output

$$\hat{c}_d(\hat{x}) = f_{NN}(\mathbf{w}, \hat{x}) \tag{18}$$

Therefore, the dynamics of the gray-box model used in training and for estimation and control becomes

$$\ddot{\theta} = M^{-1}(\theta)\left(\tau - \mathbf{V}(\theta, \dot{\theta}) - \mathbf{G}(\theta) - \left(\bar{c} + \hat{c}_d(\theta, \dot{\theta})\right)\dot{\theta}\right) \tag{19}$$

We consider a neural network with $v_0 \in R^4$, $v_{out} \in R^1$, $n_L = 3$, and $v_1 \in R^6$, which results in a parameter vector $\mathbf{w} \in R^{37}$ that must be learned. Furthermore, we use sigmoid activation functions $f_i = \frac{1}{1+e^{-v}}$. This structure was found to be sufficient for capturing the dynamics of c_d, but for more complex systems it might be worth to analyze the NN structure and dimensions considering the trade-off between performance and computational complexity. However, one could use a larger network during offline training and use ℓ_1-regularization to prune the network by removing NN parameters below a certain threshold.

4.4. Model for standard MHE

For the MHE in (6) used for comparison we consider the following model which assumes c_d to be constant

$$\ddot{\theta} = M^{-1}(\theta)\left(\tau - \mathbf{V}(\theta, \dot{\theta}) - \mathbf{G}(\theta) - (\bar{c} + \bar{c}_d)\dot{\theta}\right) \tag{20}$$

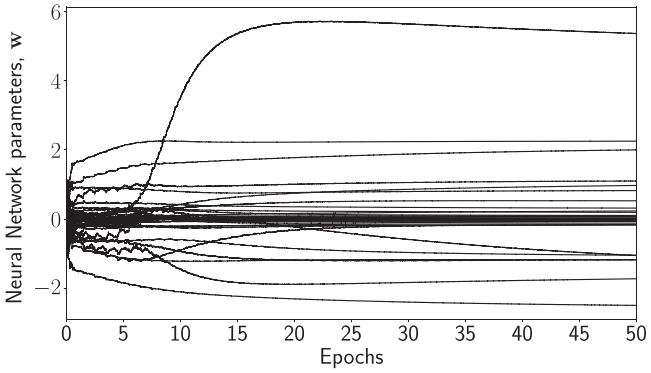


Fig. 3. Convergence of the neural network parameter vector **w** during training.

4.5. Offline training using MHE

The gray-box model can be pre-trained offline with the algorithm provided in Section 3. To this end, training data are collected by tracking random set points for both joints using an MPC controller based on the white-box model (17). Each constant set-point value is kept for 12.5 s. To fully explore the state space, $N = 20000$ samples are collected for training while 5000 samples are kept for testing the model. Offline training is carried out over $N_e = 50$ epochs and the parameter covariance $Q_w = (5 \cdot 10^{-5})^2 I \in R^{37\times37}$ is used, with $\alpha = 0.9$. We use $Q_{x0} = (1 \cdot 10^{-3})^2 I \in R^{4\times4}$, $Q_{w0} = (1 \cdot 10^{-1})^2 I \in R^{37\times37}$ and the initial state $x_0 = \begin{bmatrix}0 & 0 & 0 & 0\end{bmatrix}^T$. Moreover, the ℓ_1-regularization term $w_{l1}\|\theta\|_1$ is used with $w_{l1} = 1 \cdot 10^{-5}$. Unless otherwise stated, the horizon length $L = 10$ is used for MHE throughout the rest of the paper. No bound on $\hat{x}_j$ and $\hat{p}_j$ as in (5) is imposed. Fig. 3 shows how the neural network parameters evolve during training and tend to converge. One interesting finding is that a relatively good prediction performance is obtained very quickly, so that the learned parameters could be used already after a few epochs.

Two neural networks with the same dimensions are trained for comparison by using the ODYS Deep Learning Toolset [21]. A neural network is trained with full knowledge of the states and c_d i.e. the true states and values of c_d obtained in simulation are used as inputs and outputs of the NN respectively in training. This can be seen as the best possible achievable performance and serves as a baseline for comparison. As a second term of comparison, another neural network is trained by means of a two-step procedure. Firstly, the training data are processed using standard MHE in (6) to simultaneously estimate the states and c_d. Secondly, the neural network is trained using the estimated states as input and the estimated values of c_d as output. We remark that obtaining reasonable results required a very carefully tuning. Finally, a third term of comparison is obtained by training a NN with an EKF [3] instead of MHE in Algorithm 1. As performance index, the MSE between the estimate by the neural network, $\hat{c}_d$, and the true value c_d is used

$$\text{MSE}_{c_d} = \frac{1}{N}\sum_{k=0}^{N-1}\left(c_{d_k} - \hat{c}_{d_k}\right)^2 \tag{21}$$

For a fair comparison, we use the true states contained in the test dataset and feed them directly to the NN's. From Fig. 4, it is apparent that the neural network trained using MHE is comparable in performance with the neural network trained with full-state information. Both neural networks are able to reconstruct c_d, which is not possible for the neural network trained with the aforementioned two-step procedure. The MSE for the neural network trained using the proposed method is $6.29 \cdot 10^{-4}$ and the MSE for the neural network trained with full information is $1.61 \cdot 10^{-4}$. The neural network trained using EKF has an MSE of $1.08 \cdot 10^{-2}$ and

Table 1
MSE_{c_d} on test data for offline training methods. Obtained by evaluating the neural networks on the true states from simuation.

Method	True	"2-step"	EKF	MHE
MSE_{c_d}	$1.61 \cdot 10^{-4}$	$1.07 \cdot 10^{-1}$	$1.08 \cdot 10^{-2}$	$6.29 \cdot 10^{-4}$

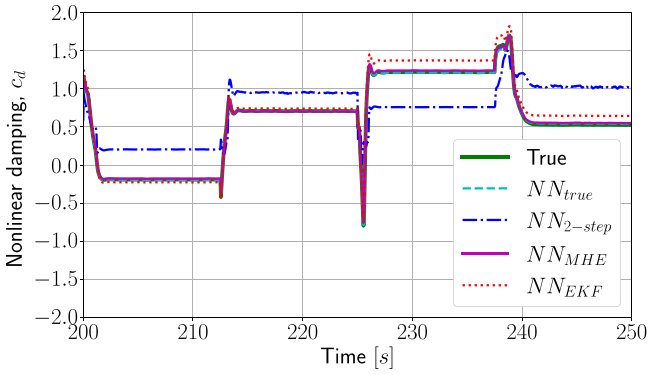


Fig. 4. Comparison of estimation capabilities of nonlinear damping, c_d, for neural networks obtained using different training methods.

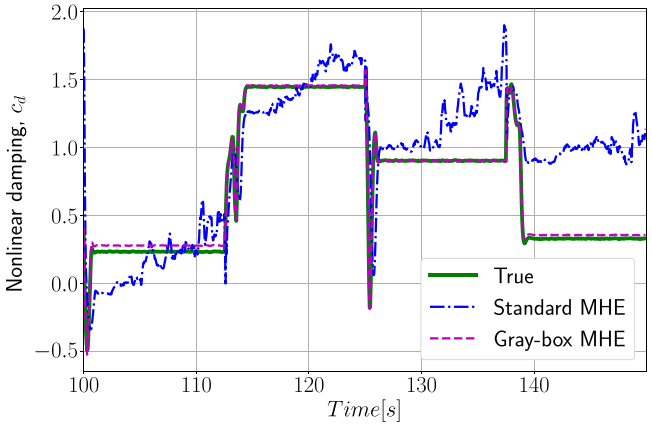


Fig. 5. Comparison of parameter estimation between gray-box model MHE and standard MHE.

the neural network trained using state estimates and c_d has an MSE of $1.01 \cdot 10^{-1}$. The results are summarized in Table 1.

4.6. Comparison in terms of estimation and prediction

As estimation and predictive capabilities are crucial for using the model in control applications such as MPC, we evaluate the ability of the gray-box model to perform such tasks. The gray-box model obtained in Section 4.5 is used for state estimation without online adaptation of the neural network. For comparison, we consider a MHE based on the formulation in (6) to estimate the state and c_d, and compare the MSE of state and c_d estimates. For the MHE using the gray-box model, the MSE on the states is $MSE_x = 2.37 \cdot 10^{-5}$ while standard MHE gives $MSE_x = 1.03 \cdot 10^{-3}$. The MSE on c_d is $MSE_{c_d} = 1.67 \cdot 10^{-3}$ for the gray-box model and $MSE_{c_d} = 0.26$ for the standard MHE. In Fig. 5 it can be seen how using the gray-box model improves estimation of c_d significantly. Further, the average and maximum computational time was decreased when using the gray-box model which is very relevant for real applications. In terms of one-step-ahead prediction quality, the gray-box model provides an $MSE_x = 2.37 \cdot 10^{-5}$, while using a constant $\bar{c}_d$ gives $MSE_x = 1.52 \cdot 10^{-3}$. Moreover, doing prediction on a longer horizon, which is the case in MPC, leads to even greater prediction errors by using a static estimate of c_d. In a 10-step ahead prediction, the gray-box model provides an $MSE_x = 8.70 \cdot 10^{-5}$, while using a constant $\bar{c}_d$ gives $MSE_x = 1.10 \cdot 10^{-2}$ which is comparable to what is obtained with the white-box model, see Table 2. The results obtained in this section underline the potential of gray-box models trained by the proposed framework to achieve high accuracy, that can improve the performance in estimation and control obtained for example with MHE and MPC. We highlight this by comparing the results obtained in closed-loop simulation with MHE and nonlinear MPC (NMPC) based on the white-box model, a model using a constant $\bar{c}_d$ over the estimation and prediction horizon obtained by (6), and the gray-box model respectively. Random set points changing every 12.5s are tracked. Further, we impose state constraints on the joint velocities, that is $-2\frac{rad}{s} \leq \dot{\theta}_1 \leq 2\frac{rad}{s}$ and $-1.5\frac{rad}{s} \leq \dot{\theta}_2 \leq 1.5\frac{rad}{s}$. The MSE obtained on the set points is $MSE_{sp} = 0.40$ for the white-box model, $MSE_{sp} = 0.39$ using a constant estimate of $\bar{c}_d$ over the horizon in MHE and MPC, and $MSE_{sp} = 0.37$ using the gray-box model i.e., the gray-box model reduces the MSE on the set point tracking by 5.6% with respect to using the standard formulation in (6). For control applications with high precision requirements such as robotic manipulators, an improvement of 5.6% is significant especially if the impact on end-effector position is considered. Further, the improvement in closed-loop MPC performance is mainly seen during transients (i.e. both models performs well in steady state) which is not reflected properly in the MSE, see Fig. 6b. As seen in Fig. 6d the NMPC based on the gray-box model is operating closer to the constraints, thanks to the capabilities of the model to better predict the impact of the unknown damping on the acceleration of the joints, which explains the improved performance during transients.

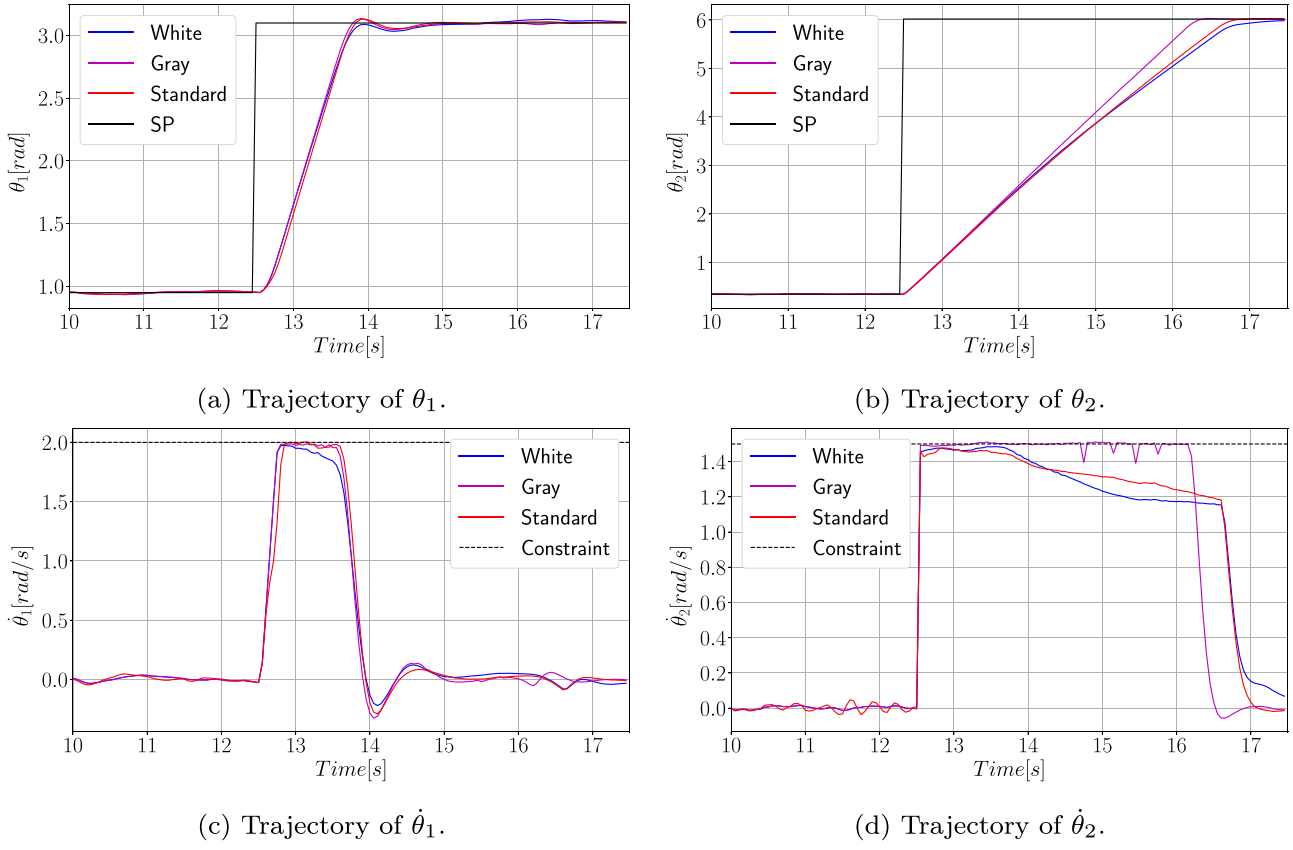
Table 2
Results in estimation, prediction and closed-loop nonlinear MPC.

Scenario	White-box	Standard MHE	Gray-box
MSE_{c_d}	-	0.26	$1.67 \cdot 10^{-3}$
MSE_x -Estimation	$1.03 \cdot 10^{-2}$	$1.03 \cdot 10^{-3}$	$2.37 \cdot 10^{-5}$
MSE_x -1-step prediction	$1.19 \cdot 10^{-2}$	$1.52 \cdot 10^{-3}$	$2.37 \cdot 10^{-5}$
MSE_x -10-step prediction	$4.41 \cdot 10^{-2}$	$1.10 \cdot 10^{-2}$	$8.70 \cdot 10^{-5}$
MSE_{sp} -Nonlinear MPC	0.40	0.39	0.37

4.7. Online adaptation using MHE

In this section we show how the MHE scheme in (5) performs in an online setting for adaptation. We use the pre-trained gray-box model from Section 4.5 but change the coefficients in (16). The covariance used for adaptation are $Q_w = (5 \cdot 10^{-5})^2 I \in R^{37\times37}$. After 125$s$, the nonlinear damping coefficients are slowly changed linearly over the following 250s and kept constant for the rest of the simulation to mimic wear and tear on the system. The new values are $c_1 = 0.15\,\frac{Nm \cdot s}{rad}$, $c_2 = 0.45\,\frac{Nm \cdot s}{rad^2}$, $c_3 = -0.2\,\frac{Nm \cdot s^2}{rad^2}$ and $c_4 = 0.0\,\frac{Nm \cdot s^2}{rad^2}$.

For comparison, $\hat{c}_d$ is also evaluated with the pre-trained neural network. From Fig. 7 it is seen how the MHE is capable of coping with the changes and provide a better estimate of $\hat{c}_d$ than the neural network relying on the parameters obtained in offline training. Evaluating the neural network with the pre-trained parameters yields $MSE_{c_d} = 4.34 \cdot 10^{-1}$, while the adaptive neural network gives $MSE_{c_d} = 9.12 \cdot 10^{-3}$ The results in Fig. 7 and the MSE confirm the applicability of (5) for online adaptation of gray-box dynamical models containing neural networks. In this work only slowly changing coefficients in (16) have been considered, however it expected that the method will perform similarly well under rapid



(a) Trajectory of θ_1.

(b) Trajectory of θ_2.

(c) Trajectory of $\dot{\theta}_1$.

(d) Trajectory of $\dot{\theta}_2$.

Fig. 6. Comparison of NMPC results based on different model types: white box (17), gray box (19), and standard MHE model (20).

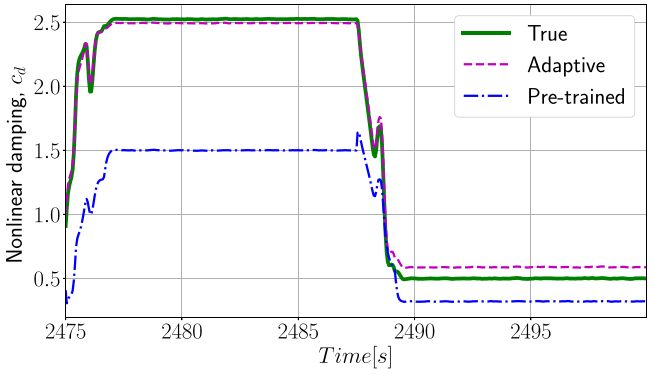


Fig. 7. Values of $\hat{c}_d$ calculated by the pre-trained neural network and the neural network adapted online by MHE.

changes, e.g. step changes, since the standard MHE in (6) has been seen to handle step changes during parameter estimation well.

5. Conclusions and future work

In this paper we have proposed a moving horizon estimation scheme to learn and adapt gray-box models from noisy input-output data in which neural networks are used to model hidden dynamics. The approach can be used offline to reconstruct a model by processing a given input/output dataset over multiple epochs and is also inherently suited for online adaptation. The key ingredient to be able to process the dataset in short batches is the use of a properly defined arrival cost. A further feature offered by MHE is its natural ability to handle constraints on estimated quantities throughout the learning process, possibly enhancing estimation quality by enforcing known physical limitations. While promising results have been achieved, many interesting research challenges lie ahead. One promising line of research is to exploit the knowledge contained in the arrival cost. With an uncertainty estimate on all neural network parameters, the optimization problem to be solved online could be reduced to a subset of all the neural network parameters. This would allow to decrease the MHE computation time or to handle larger neural networks. Another interesting topic to investigate is the adaptive tuning of the artificial covariance matrix of the neural network parameters; discovering a suitable adaptation mechanism is expected to be useful for both offline training and online adaptation.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

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

[1] L.B. Armenio, E. Terzi, M. Farina, R. Scattolini, Model predictive control design for dynamical systems learned by echo state networks, IEEE Control Syst. Lett. 3 (4) (2019) 1044–1049, doi:10.1109/LCSYS.2019.2920720.
[2] J.F. Bellantoni, K.W. Dodge, A square root formulation of the Kalman- schmidt filter, AIAA J. 5 (7) (1967) 1309–1314, doi:10.2514/3.4189.
[3] A. Bemporad, Recurrent neural network training with convex loss and regularization functions by extended Kalman filtering(2021). Submitted for publication. Available on https://arxiv.org/abs/2111.02673.

[4] F. Bonassi, J. Xie, M. Farina, R. Scattolini, Towards lifelong learning of recurrent neural networks for control design, in: 2022 European Control Conference (ECC), 2022, pp. 2018–2023, doi:10.23919/ECC55457.2022.9838393.
[5] L. Brunke, M. Greeff, A.W. Hall, Z. Yuan, S. Zhou, J. Panerati, A.P. Schoellig, Safe learning in robotics: from learning-based control to safe reinforcement learning, Annu. Rev. Control Rob. Auton. Syst. 5 (1) (2022) 411–444, doi:10.1146/annurev-control-042920-020211.
[6] A. Carron, E. Arcari, M. Wermelinger, L. Hewing, M. Hutter, M.N. Zeilinger, Data-driven model predictive control for trajectory tracking with a robotic arm, IEEE Rob. Autom. Lett. 4 (4) (2019) 3758–3765, doi:10.1109/LRA.2019.2929987.
[7] J. Craig, Introduction to Robotics: Mechanics & Control, Addison-Wesley Pub. Co., 1986.
[8] A. Draeger, S. Engell, H. Ranke, Model predictive control using neural networks, IEEE Control Syst. Mag. 15 (5) (1995) 61–66, doi:10.1109/37.466261.
[9] L. Fagiano, C. Novara, A combined moving horizon and direct virtual sensor approach for constrained nonlinear estimation, Automatica 49 (1) (2013) 193–199, doi:10.1016/j.automatica.2012.09.009.
[10] X. Glorot, Y. Bengio, Understanding the difficulty of training deep feedforward neural networks, in: International Conference on Artificial Intelligence and Statistics, 2010.
[11] L. Hewing, K.P. Wabersich, M. Menner, M.N. Zeilinger, Learning-based model predictive control: toward safe learning in control, Annu. Rev. Control Robot. Auton. Syst. 3 (1) (2020) 269–296, doi:10.1146/annurev-control-090419-075625.
[12] G.A. Hicks, W.H. Ray, Approximation methods for optimal control synthesis, Canadian J. Chem. Eng. 49 (4) (1971) 522–528, doi:10.1002/cjce.5450490416.
[13] S. Hochreiter, The vanishing gradient problem during learning recurrent neural nets and problem solutions, Int. J. Uncertain. Fuzziness Knowl.-Based Syst. 6 (2) (1998) 107–116, doi:10.1142/S0218488598000094.
[14] Z.-S. Hou, Z. Wang, From model-based control to data-driven control: survey, classification and perspective, Inf. Sci. 235 (2013) 3–35, doi:10.1016/j.ins.2012.07.014. Data-based Control, Decision, Scheduling and Fault Diagnostics
[15] G.E. Karniadakis, I.G. Kevrekidis, L. Lu, P. Perdikaris, S. Wang, L. Yang, Physics-informed machine learning, Nat. Rev. Phys. 3 (6) (2021) 422–440, doi:10.1038/s42254-021-00314-5.
[16] P. Kühl, M. Diehl, T. Kraus, J.P. Schlöder, H.G. Bock, A real-time algorithm for moving horizon state and parameter estimation, Comput. Chem. Eng. 35 (2011) 71–83.
[17] M. Maiworm, D. Limon, R. Findeisen, Online learning-based model predictive control with gaussian process models and stability guarantees, Int. J. Robust Nonlinear Control 31 (18) (2021) 8785–8812.
[18] D. Masti, A. Bemporad, Learning nonlinear state-space models using autoencoders, Automatica 129 (2021) 109666.
[19] D. Nguyen-Tuong, J. Peters, Model learning for robot control: a survey, Cognit. Process. 12 (2011) 319–340, doi:10.1007/s10339-011-0404-1.
[20] J. Nocedal, S.J. Wright, Numerical Optimization, 2nd ed., Springer, New York, NY, USA, 2006.
[21] ODYS s.r.l, ODYS deep learning (version 0.1.0, 2020), https://www.odys.it/deep-learning/.
[22] Y. Pan, J. Wang, Nonlinear model predictive control using a recurrent neural network, in: 2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence), 2008, pp. 2296–2301, doi:10.1109/IJCNN.2008.4634115.
[23] C.V. Rao, J.B. Rawlings, D.Q. Mayne, Constrained state estimation for nonlinear discrete-time systems: stability and moving horizon approximations, IEEE Trans. Autom. Control 48 (2) (2003) 246–258, doi:10.1109/TAC.2002.808470.
[24] J.B. Rawlings, D.Q. Mayne, M. Diehl, Model Predictive Control: Theory, Computation, and Design, Nob Hill Publishing, 2017.
[25] B. Sohlberg, E.W. Jacobsen, Grey box modelling – branches and experiences, IFAC Proc. Vol. 41 (2) (2008) 11415–11420, doi:10.3182/20080706-5-KR-1001.01934. 17th IFAC World Congress
[26] A.T. Taylor, T.A. Berrueta, T.D. Murphey, Active learning in robotics: a review of control principles, Mechatronics 77 (2021) 102576, doi:10.1016/j.mechatronics.2021.102576.
[27] S. Thrun, T.M. Mitchell, Lifelong robot learning, Rob. Auton. Syst. 15 (1) (1995) 25–46, doi:10.1016/0921-8890(95)00004-Y. The Biology and Technology of Intelligent Autonomous Agents
[28] A.S. Weigend, B.A. Huberman, D.E. Rumelhart, Predicting the future: a connectionist approach, Int. J. Neural Syst. 01 (03) (1990) 193–209, doi:10.1142/S0129065790000102.
[29] P.J. Werbos, Backpropagation through time: what it does and how to do it, Proc. IEEE 78 (10) (1990) 1550–1560, doi:10.1109/5.58337.