Computers and Chemical Engineering 148 (2021) 107266

Contents lists available at ScienceDirect

Computers and Chemical Engineering

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

Approximate moving horizon estimation and robust nonlinear model predictive control via deep learning

Benjamin Karg, Sergio Lucia*

Process Automation Systems, TU Dortmund University, Emil-Figge-Str. 70, 44227 Dortmund, Germany

ARTICLE INFO

Article history:
Received 2 September 2020
Revised 25 November 2020
Accepted 24 February 2021
Available online 2 March 2021

Keywords:
Robust control
Nonlinear model predictive control
Learning-based control
Moving horizon estimation

ABSTRACT

Optimization-based methods for output-feedback control enable dealing with multiple-input and multiple-output nonlinear systems in the presence of uncertainties and constraints. The combination of moving horizon estimation (MHE) and nonlinear model predictive control (NMPC) can be especially powerful because of its general formulation but its implementation requires solving two optimization problems at every sampling instant, which can be challenging due to hardware or time constraints. We propose to take advantage of the expressive capabilities of deep neural networks to approximate the solution of the MHE and NMPC problems. By substituting the MHE and NMPC with their learning-based counterparts, the required online computations are significantly reduced. We also propose to use sensitivity analysis to compute an approximate upper-bound of the maximum one-step divergence from the optimal performance caused by the approximation error. The efficacy of the proposed learning-based approach is illustrated with simulation results of a semi-batch reactor for industrial polymerization.

1. Introduction

The combination of moving horizon estimation (MHE) and nonlinear model predictive control (NMPC) is a powerful choice to achieve high-performance output-feedback control that can systematically deal with nonlinear systems and constraints. The main drawback of those two methods is the requirement to solve one computationally complex optimization problem to derive a state estimate and one to find the optimal control input at each control instant, which might be prohibitive when time and computational power are a scarce resource.

To mitigate this drawback, methods like explicit model predictive control (Bemporad et al., 2002; Johansen, 2004) have been proposed to remove the necessity to solve the optimization problem online. This can be done by parametrically pre-computing the solution of the optimization problem for all possible initial conditions of the system. However, the computation of the explicit functions suffers from the curse of dimensionality and limits its usage to small-scale systems (Bayat et al., 2011). An alternative is to use neural networks to approximate the solution of the MPC optimization problems, which was was proposed as early as 1995 (Parisini and Zoppoli, 1995) and has recently gained momentum due to the fast progress in deep learning (Chen et al., 2018; Karg and Lucia, 2018). For linear systems it was shown that deep neural networks can exactly represent explicit MPC laws (Karg and Lucia, 2020a) and that stability can be enforced (Karg and Lucia, 2020b) while for nonlinear systems strong probabilistic statements about sub-optimality and safety can be made (Chen et al., 2018; Hertneck et al., 2018; Karg et al., 2019; Nubert et al., 2020).

In this work, we extend the results from Lucia and Karg (2018) where neural networks were used to approximate a robust nonlinear model predictive control law. Multi-stage NMPC (Lucia et al., 2013) is a robust NMPC scheme, which reduces conservatism by taking future feedback and adaptions of the corresponding control inputs into account as opposed to min-max NMPC (Scokaert and Mayne, 1998). The extension in this work includes the development of a learning-based estimator, which, to the authors knowledge, has not been previously proposed. In addition, we also propose the simultaneous learning of the full estimator-controller algorithm. Finally, we present how sensitivity analysis can be used to quantify the influence that the approximation errors of the different learned components can have on the closed-loop control performance.

The main advantage of the proposed method is that it renders the application of advanced estimation and control methods possible where real-time optimization would fail due to fast sampling times, the complexity of the optimization problem or limited computational power. We highlight the advantages of the proposed method by applying the method for an industrial-scale case study

* Corresponding author.
E-mail address: sergio.lucia@tu-dortmund.de (S. Lucia).

https://doi.org/10.1016/j.compchemeng.2021.107266
0098-1354/

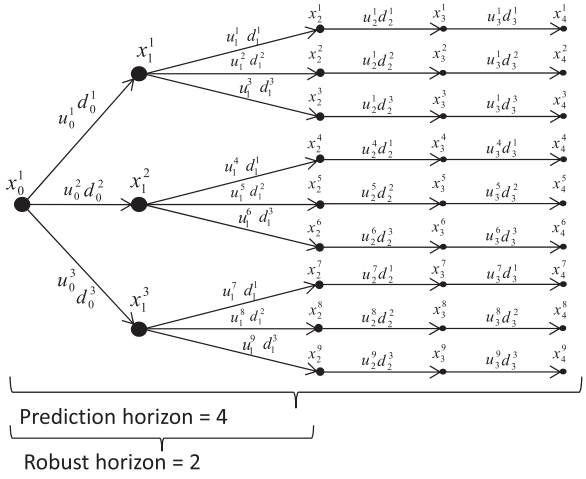

Fig. 1. Scenario tree representation of the evolution of the uncertainty for multi-stage NMPC.

and by deploying the approximate estimator and controller on a low-cost microcontroller.

The remainder of the paper is organized as follows. Section 2 presents the mathematical framework, namely multi-stage NMPC, moving horizon estimation and deep neural networks. In Section 3, we present our approach to learn an approximation of an NMPC controller, an MHE estimator and a simultaneous learning of both controller and estimator. Section 4 presents the industrial batch polymerization reactor which is used for the analysis of the presented methods in Section 5. The paper is then concluded in Section 6.

Notation

The (non-negative) natural numbers are denoted by $\mathbb{N}^+$ and $\mathbb{R}$, $\mathbb{R}^n$, $\mathbb{R}^{n\times m}$ denote real numbers, real vectors of size n and real matrices of shape $n \times m$. The norm x^TPx is abbreviated via $||x||_P^2$. The composition of two functions α and β is denoted by $\alpha \circ \beta(\cdot)$, which is equal to $\alpha(\beta(\cdot))$. Solutions of an optimization problem are marked with a star, e.g. x^*. The average of the singular values and the maximum singular value of a matrix are denoted by $\sigma_{\text{av}}(\cdot)$ and $\sigma_{\max}(\cdot)$, respectively.

2. Background

In this work, we consider general discrete-time nonlinear systems:

$$\begin{aligned} x_{k+1} &= f(x_k, u_k, d_k), \\ y_k &= h(x_k, u_k, d_k), \end{aligned} \tag{1}$$

where x_k are the states, u_k the control inputs, d_k the uncertain parameters or disturbances and y_k the measurements. This section presents the necessary background on three key ideas used throughout the paper: multi-stage robust nonlinear model predictive control, moving horizon estimation and deep neural networks.

2.1. Multi-stage nonlinear model predictive control

This section reviews the main concepts of the multi-stage NMPC approach. In multi-stage NMPC, the uncertainty is modeled by a tree of discrete scenarios that branch at each prediction step for each possible value of the uncertainty (see Fig. 1). By formulating the evolution of the uncertainty as a tree of possible scenarios, it is explicitly taken into account that future control inputs can be adapted when the future measurements become available. This makes the approach less conservative than traditional robust open-loop approaches (Lee and Yu, 1997). Formulating the uncertain decision making problem as a scenario tree is a well-known approach in the field of multi-stage stochastic programming (Shapiro, 2009). In the context of model predictive control, these ideas have been used for the linear case in Scokaert and Mayne (1998); Bernardini and Bemporad (2009) and in the nonlinear case e.g. in Lucia et al. (2013, 2014). Recently, it has been shown that the multi-stage approach can obtain a good performance in many different applications (Yu and Biegler, 2018; Haßkerl et al., 2018; Krishnamoorthy et al., 2016; 2018; Lucia, 2015).

The main challenge of multi-stage NMPC is that the size of the optimization problem grows exponentially with the number of uncertainties and the prediction horizon. To avoid the exponential growth with the prediction horizon, it is possible to branch the tree only up to a certain stage (called the robust horizon N_r) and consider that the uncertainty remains constant afterwards, as shown in Fig. 1. The stability properties of the approach, which are out of the scope of this work, have been studied in Lucia et al. (2020b). In addition, multi-stage NMPC can be combined with tube-based MPC approaches (Mayne et al., 2005), as shown in Subramanian et al. (2018), to obtain the desired tradeoff between performance and computational complexity.

The optimization problem to be solved at each sampling instant is:

$$\min_{x_{k+1}^j, u_k^j, \forall (j,k)\in I} \quad \sum_{i=1}^{N} \omega_i J_i(X_i, U_i) \tag{2a}$$

subject to:

$$x_{k+1}^j = f\left(x_k^{p(j)}, u_k^j, d_k^{r(j)}\right), \qquad \forall (j,k) \in I_{[0,N-1]}, \tag{2b}$$

$$0 \geq g\left(x_{k+1}^j, u_k^j, d_k^{r(j)}\right), \qquad \forall (j,k) \in I_{[0,N-1]}, \tag{2c}$$

$$u_k^j = u_k^l \text{ if } x_k^{p(j)} = x_k^{p(l)}, \qquad \forall (j,k),(l,k) \in I_{[0,N-1]}, \tag{2d}$$

where X_i, U_i are the set of states and control inputs that belong to the scenario S_i and N is the prediction horizon. The set $I_{[0,N-1]}$ includes all the indices (j,k) in the scenario tree from stage 0 until stage $N-1$. The importance in the cost of each scenario is weighted via ω_i. The constraints on inputs and states are denoted by $g(\cdot)$. Terminal constraints could be added if necessary. The cost of each scenario is denoted by $J_i(\cdot)$ and can be written as:

$$J_i(X_i, U_i) := \sum_{k=0}^{N_p-1} L\left(x_{k+1}^j, u_k^j\right), \quad \forall x_{k+1}^j, u_k^j \in S_i. \tag{3}$$

The constraints (2d) are called non-anticipativity constraints which enforce causality of the control policy. That is, the control inputs cannot anticipate the realization of the uncertainty.

2.2. Moving horizon estimation

The goal of moving horizon estimation is to provide an accurate state estimate x_{est} based on N_{est} past measurements y_k and N_{est} past inputs u_k by solving an optimization problem. Similar to MPC, the system model is considered via equality constraints in the following least-squares problem:

$$\begin{aligned} \min_{x_0,\ldots,x_{N_{\text{est}}},d_{\text{est}}} \quad & \tfrac{1}{2}||x_0 - x_a||_{P_x}^2 + \tfrac{1}{2}||d_{\text{est}} - d_{\text{prev}}||_{P_d}^2 \\ & + \sum_{k=1}^{N_{\text{est}}} \tfrac{1}{2}||y_k - h(x_k, u_k, d_{\text{est}})||_{P_y}^2 \end{aligned} \tag{4a}$$

subject to:

$$x_{k+1} = f(x_k, u_k, d_{\text{est}}), \qquad k = 0, \ldots, N_{\text{est}} - 1, \tag{4b}$$

$$0 \geq g_{\text{est}}(x_k, u_k, d_{\text{est}}), \qquad k = 0, \ldots, N_{\text{est}} - 1, \tag{4c}$$

$$d_{\text{lb}} \leq d_{\text{est}} \leq d_{\text{ub}}, \tag{4d}$$

where P_x, P_d and P_y are the weights for the arrival cost, the parameter estimation and the distance between measurements and computed measurements. The parameter estimate from the previous step is given by d_{prev}. To counteract the information loss due to the finite horizon, the arrival cost (the first term in (4a)) includes information from previous estimations via x_a, which is the x_1^* from the previous estimation. Since this work deals with uncertain systems, it is necessary to estimate the values of the uncertain parameters to provide accurate state estimates. If the uncertainty interval is known, it can be incorporated via (4d). In the case of output-feedback, the state estimate x_{est} computed by the MHE, which is the state at the last time in the considered time-window ($x_{\text{est}} = x_{N_{\text{est}}}$), is used as the root node of the scenario tree in (2) via $x_0^1 = x_{\text{est}}$.

Remark 1. The constraints (4c) and (2c) can be chosen differently. In some cases, relaxing the constraints for the MHE to enable accurate state estimation even in the presence of minor constraint violations in the control task can be beneficial.

2.3. Deep neural networks

A neural network is a mapping $\mathcal{N} : \mathbb{R}^{n_{in}} \to \mathbb{R}^{n_{out}}$ that is composed of affine functions λ_i with $i = 1, \ldots, L+1$ and nonlinear activation functions γ_i with $i = 1, \ldots, L$:

$$\mathcal{N}(x_{\text{in}}; \theta, M, L) = \lambda_{L+1} \circ \gamma_L \circ \ldots \circ \gamma_1 \circ \lambda_1(x), \tag{5}$$

where $L \in \mathbb{N}^+$ is the number of hidden layers (also called depth) of the neural network which should be chosen as $L \geq 2$ to be considered a deep neural network. The number of neurons in each hidden layer is denoted by $M \in \mathbb{N}^+$. Each affine function is given by:

$$\lambda_i = W_i \xi_{i-1} + b_i, \tag{6}$$

where W_i is the weight matrix and b_i the bias vector of the ith layer and ξ_{i-1} is the output of the previous layer. For the first layer $i = 1$ the input of the previous layer coincides with the input to the function: $\xi_0 = x_{\text{in}}$. In this work we consider networks which have n_{in} inputs, n_{out} outputs and we always use the hyperbolic tangent as activation function:

$$\tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}, \tag{7}$$

which is evaluated element-wise on vectors. The resulting sizes of the weights are:

$$W_i \in \begin{cases} \mathbb{R}^{M \times n_{\text{in}}} & \text{if } i = 1, \\ \mathbb{R}^{M \times M} & \text{if } i = 2, \ldots, L, \\ \mathbb{R}^{n_{\text{out}} \times M} & \text{if } i = L+1, \end{cases} \tag{8}$$

and of the biases:

$$b_i \in \begin{cases} \mathbb{R}^{M} & \text{if } i = 1, \ldots, L, \\ \mathbb{R}^{n_{\text{out}}} & \text{if } i = L+1. \end{cases} \tag{9}$$

All the weights and biases, which completely parametrize a neural network, are summarized in $\theta = \{W_1, b_1, \ldots, W_{L+1}, b_{L+1}\}$.

3. Deep learning-based robust NMPC and MHE

Multi-stage NMPC and MHE are optimization-based control and estimation methods, which may require a significant computational effort. The goal of this work is to use deep neural networks to ease the computational load of deploying advanced control and estimation techniques by learning the mapping defined by the MPC and MHE optimization problems. The main idea is to exploit the fact that the optimization problems described in (2) and in (4) are parametric. The parameter that determines the solution of the MPC problem is the initial state x_0^1. The parameters that determine the solution of the MHE problem are the measurement and input trajectory as well as the previously estimated state.

3.1. Learning an approximate explicit representation

We consider three different learning-based functions which approximate the functions implicitly described by the optimization problems (2) and (4). The first two networks approximate directly either the MPC, leading to the neural network controller (NNC), or the MHE, leading to the neural network estimator (NNE). The third network simultaneously learns a controller and an estimator, leading to the neural network estimator and controller (NNEC). The crucial part for learning a good approximation of the MPC and MHE are the data sets, which need to contain enough samples to enable the learned networks to achieve the desired approximation accuracy, and more importantly the desired closed-loop performance, while having a tractable size.

For the data generation, we use closed-loop simulations, as opposed to purely random sampling, quasi-random sampling (James et al., 1997) or gridding, to ensure that only samples that represent situations occurring in the operation of the system are included in the data set. Every simulation step out of the N_{s} considered steps is fully described by a tuple $(s_p, x_{\text{est},p}, x_p, u_p^*, d_p)$, where $s_p = [x_a, y_{-N_{\text{est}}+1}, \ldots, y_0, u_{-N_{\text{est}}}, \ldots, u_{-1}]$ contains the arrival state, measurements and control input trajectories used by the MHE to generate the state estimate $x_{\text{est},p}$. The current state of the system is x_p, u_p^* is the optimal control input computed by the MPC based on the state estimate x_{est} and d_p is the realization of the uncertain parameters. These tuples enable the learning of an NNE and an NNC either separately or simultaneously.

The behavior of the MPC is approximated by the network $\mathcal{N}_{\text{C}} : \mathbb{R}^{n_x} \to \mathbb{R}^{n_u}$ and the behavior of the MHE by $\mathcal{N}_{\text{E}} : \mathbb{R}^{n_s} \to \mathbb{R}^{n_x}$ with $n_s = n_x + N_{\text{est}} \cdot (n_y + n_u)$. That is, the approximate MHE takes as input the trajectories of inputs and measurements of length N_{est} as well as the previously estimated state as in the original MHE formulation. The output of the approximate MHE is the current state estimate.

The third learning-based approach has a similar architecture as the network for state estimation, but also directly gives an optimal input which results in a mapping $\mathcal{N}_{\text{EC}} : \mathbb{R}^{n_s} \to \mathbb{R}^{n_x+n_u}$. The state estimates of NNE and NNEC will be used after $N_{\text{est}} - 1$ steps as the arrival state x_{a} in the closed-loop operation. The optimal weights and biases for each network are determined during training by minimizing the following loss functions:

$$\text{NNC}: \quad \min_{\theta} \frac{1}{N_{\text{s}}} \sum_{p=1}^{N_{\text{s}}} (u_p^* - \mathcal{N}_{\text{C}}(x_{\text{est},p}))^2, \tag{10a}$$

$$\text{NNE}: \quad \min_{\theta} \frac{1}{N_{\text{s}}} \sum_{p=1}^{N_{\text{s}}} (x_p - \mathcal{N}_{\text{E}}(s_p))^2, \tag{10b}$$

$$\text{NNEC}: \quad \min_{\theta} \frac{1}{N_{\text{s}}} \sum_{p=1}^{N_{\text{s}}} \left(\begin{bmatrix} x_p \\ u_p^* \end{bmatrix} - \mathcal{N}_{\text{EC}}(s_p) \right)^2. \tag{10c}$$

It is interesting to note some differences between the training of the approximations of the control and estimation problems. While in (10a) the neural network sees exactly the same data as the MPC in the closed-loop, in (10b) and (10c) we exploit that the simulation-based data generation grants access to the real state.

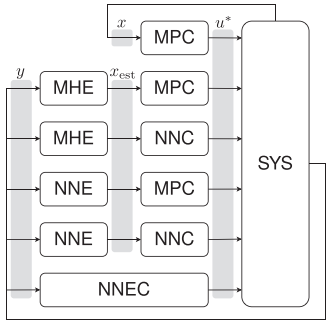

Fig. 2. Overview of the different configurations which can be built from MHE, MPC and their neural network based approximations NNE and NNC. MPC via state-feedback is also included which would represent the perfect estimator and NNEC is using past measurements and control inputs to provide an optimal control input.

Because x_p would be the result of a perfect estimation, we use it as the target instead of the state $x_{\text{est},p}$ estimated by the MHE. The possible configurations for the output-feedback control task, combining optimization-based and learning-based methods, are visualized in Fig. 2 where also MPC with state-feedback is presented, as it shows the performance in the ideal case, i.e. when the estimator provides perfect state estimates ($x_{\text{est}} = x$) and the MPC optimization problem is exactly solved.

3.2. Sensitivity of the closed-loop system

Analyzing an output-feedback controlled system with uncertainty and measurement noise is a difficult task. The complexity is increased when the impact of the learning process on the performance should be investigated because MPC and / or MHE are substituted by their neural network approximations. Because the evolution of the states can only be affected via the control inputs, the influence of the approximation errors on the control input is crucial. The approximation errors for a sample at time k are defined as:

$$\text{NNC}: \qquad r_k^{\text{C}} = ||u_k^* - \mathcal{N}_{\text{C}}(x_{\text{est},k})||_2, \tag{11}$$

$$\text{NNE}: \qquad r_k^{\text{E}} = ||x_k - \mathcal{N}_{\text{E}}(s_k)||_2. \tag{12}$$

The impact that the approximation error of the approximate controller NNC has on the state trajectory can be analyzed by computing the following sensitivity:

$$J_{u,k} = \frac{\partial f(x,u,d)}{\partial u}\Big|_{x_k,u_k^*,d_k}, \tag{13}$$

based on the system Eq. (1). The resulting changes in the successor state with respect to deviations to the optimal input can then be approximated, using a first order Taylor approximation, by:

$$x_{k+1}^* - \hat{x}_{k+1} \approx J_{u,k}(u_k^* - \hat{u}_k), \tag{14}$$

where $\hat{u}_k$ is the approximate control input and the successor states are $x_{k+1}^* = f(x_k, u_k^*, d_k)$ when the optimal input is applied and $\hat{x}_{k+1} = f(x_k, \hat{u}_k, d_k)$ when the approximate input is applied.

Quantifying the effects of the estimation quality on deviations from the optimal control input cannot be handled as straightforwardly. Because the state estimate $x_{\text{est},k}$ is used in the MPC scheme (2) via $x_0^1 = x_{\text{est},k}$, it would be necessary to compute the sensitivity of the optimal control problem w.r.t. to the estimated state. Changes in the active set can render computed sensitivities invalid when relying on NLP sensitivities (Fiacco, 1983). To circumvent this process, we exploit that the neural network controller is an explicit function which approximates the solution of the optimal control problem. When assuming state-feedback, the closed-loop system with NNC derived from (1) is:

$$x_{k+1} = f_{\text{c}}(x_k, d_k), \tag{15}$$

with $f_{\text{c}}(x_k, d_k) = f(x_k, \mathcal{N}_{\text{c}}(x_k), d_k)$. We use this formulation to approximate the sensitivity of the successor state with respect to the current estimated state by:

$$J_{x,k} = \frac{\partial f_{\text{c}}(x,d)}{\partial x}\Big|_{x_k,d_k}. \tag{16}$$

Please note that (16) is only a close approximation of the sensitivity of the real successor state w.r.t. to the estimated state, if $||u^* - \mathcal{N}_{\text{C}}(x)||_2$ is sufficiently small for all states x along closed-loop trajectories. The local first-order approximation of the deviation from the optimal successor state due to estimation errors can then be written as:

$$x_{k+1}^* - \tilde{x}_{k+1} \approx J_{x,k}(x_k - x_{\text{est}}), \tag{17}$$

where $\tilde{x}_{k+1} = f(x_k, \tilde{u}_k, d_k)$ and $\tilde{u}_k$ is the input computed by MPC for the estimated state x_{est}.

The deviations from the optimal successor state due to the approximation error in the learned controller and learned estimator can then be locally upper-bounded based on the maximum singular value of the sensitivities and the local approximation error:

$$\|x_{k+1}^* - x_{k+1}^{\text{C}}\|_2 \lessapprox \sigma_{\max}\left(J_{u,k}\right) \cdot r_k^{\text{C}}, \tag{18}$$

$$||x_{k+1}^* - x_{k+1}^{\text{E}}||_2 \lessapprox \sigma_{\max}(J_{x,k}) \cdot r_k^{\text{E}}, \tag{19}$$

where $x_{k+1}^{\text{C}} = f_{\text{c}}(x_k, d_k)$ is the successor state when NNC is used and $x_{k+1}^{\text{E}} = f(x_k, u_{\text{E}}^*, d_k)$ is the successor state when the optimal input u_k^{E} is computed via MPC based on the NNE state estimate $x_{\text{est},k} = \mathcal{N}_{\text{E}}(s_k)$.

The main idea of this analysis is to illustrate that for the same approximation error of the control (r_k^{C}) and estimation problems (r_k^{E}), the effect on the closed-loop trajectories can be significantly different depending on the described sensitivities. This effect can be clearly observed in the results presented in Section 5.

If the maximum sensitivities and the maximum approximation errors out of all samples are considered, a global estimate for the maximum deviation in the successor states can be computed. Probabilistic validation methods like those presented in Alamo et al. (2015) can be applied to achieve rigorous probabilistic guarantees of the approximation errors.

4. Case study

For the analysis of the proposed approach we consider an industrial batch polymerization reactor, which was first presented in Lucia et al. (2014).

The system can be described by the following differential equations:

$$\begin{aligned}
\dot{m}_{\text{W}} &= \dot{m}_{\text{F}}\,\omega_{\text{W,F}} \\
\dot{m}_{\text{A}} &= \dot{m}_{\text{F}}\omega_{\text{A,F}} - k_{\text{R1}}\,m_{\text{A,R}} - k_{\text{R2}}\,m_{\text{AWT}}\,m_{\text{A}}/m_{\text{ges}}, \\
\dot{m}_{\text{P}} &= k_{\text{R1}}\,m_{\text{A,R}} + p_1\,k_{\text{R2}}\,m_{\text{AWT}}\,m_{\text{A}}/m_{\text{ges}}, \\
\dot{m}_{\text{A}}^{\text{acc}} &= \dot{m}_{\text{F}}, \\
\dot{T}_{\text{R}} &= 1/(c_{\text{p,R}}m_{\text{ges}})\,[\dot{m}_{\text{F}}\,c_{\text{p,F}}(T_{\text{F}} - T_{\text{R}}) + \Delta H_{\text{R}}k_{\text{R1}}m_{\text{A,R}} \\
&\quad - k_{\text{K}}A(T_{\text{R}} - T_{\text{S}}) - \dot{m}_{\text{AWT}}\,c_{\text{p,R}}(T_{\text{R}} - T_{\text{EK}})], \\
\dot{T}_{\text{S}} &= 1/(c_{\text{p,S}}m_{\text{S}})\,[k_{\text{K}}A(T_{\text{R}} - T_{\text{S}}) - k_{\text{K}}A(T_{\text{S}} - T_{\text{M}})], \\
\dot{T}_{\text{M}} &= 1/(c_{\text{p,W}}m_{\text{M,KW}})\,[\dot{m}_{\text{M,KW}}\,c_{\text{p,W}}\left(T_{\text{M}}^{\text{IN}} - T_{\text{M}}\right) \\
&\quad + k_{\text{K}}A(T_{\text{S}} - T_{\text{M}})],
\end{aligned}$$

$$
\begin{aligned}
\dot{T}_{\mathrm{EK}} &= 1/(c_{\mathrm{p,R}} m_{\mathrm{AWT}})\,[\dot{m}_{\mathrm{AWT}} c_{\mathrm{p,W}} (T_{\mathrm{R}} - T_{\mathrm{EK}}) \\
&\quad - \alpha (T_{\mathrm{EK}} - T_{\mathrm{AWT}}) + k_{\mathrm{R2}}\, m_{\mathrm{A}}\, m_{\mathrm{AWT}} \Delta H_{\mathrm{R}} / m_{\mathrm{ges}}], \\
\dot{T}_{\mathrm{AWT}} &= [\dot{m}_{\mathrm{AWT,KW}}\, c_{\mathrm{p,W}} \left(T^{\mathrm{IN}}_{\mathrm{AWT}} - T_{\mathrm{AWT}}\right) \\
&\quad - \alpha (T_{\mathrm{AWT}} - T_{\mathrm{EK}})]/(c_{\mathrm{p,W}} m_{\mathrm{AWT,KW}}), \\
\dot{T}_{\mathrm{adiab}} &= \frac{\Delta H_{\mathrm{R}}}{m_{\mathrm{ges}} c_{\mathrm{p,R}}} \dot{m}_{\mathrm{A}} - (\dot{m}_{\mathrm{W}} + \dot{m}_{\mathrm{A}} + \dot{m}_{\mathrm{P}}) \left(\frac{m_{\mathrm{A}} \Delta H_{\mathrm{R}}}{m^2_{\mathrm{ges}} c_{\mathrm{p,R}}} \right) + \dot{T}_{\mathrm{R}},
\end{aligned}
\tag{20}
$$

where:

$$
\begin{aligned}
U &= m_{\mathrm{P}}/(m_{\mathrm{A}} + m_{\mathrm{P}}), \\
m_{\mathrm{ges}} &= m_{\mathrm{W}} + m_{\mathrm{A}} + m_{\mathrm{P}}, \\
k_{\mathrm{R1}} &= k_0 e^{\frac{-E_a}{R(T_{\mathrm{R}}+273.15)}} (k_{\mathrm{U1}}(1-U) + k_{\mathrm{U2}} U), \\
k_{\mathrm{R2}} &= k_0 e^{\frac{-E_a}{R(T_{\mathrm{EK}}+273.15)}} (k_{\mathrm{U1}}(1-U) + k_{\mathrm{U2}} U), \\
k_{\mathrm{K}} &= (m_{\mathrm{W}} k_{\mathrm{WS}} + m_{\mathrm{A}} k_{\mathrm{AS}} + m_{\mathrm{P}} k_{\mathrm{PS}})/m_{\mathrm{ges}}, \\
m_{\mathrm{A,R}} &= m_{\mathrm{A}} - m_{\mathrm{A}} m_{\mathrm{AWT}}/m_{\mathrm{ges}}.
\end{aligned}
$$

The states describing the system are mass balances and product hold-ups for water m_{W}, monomer m_{A} and polymer m_{P} in the reactor and the total mass of monomer that has been fed to the reactor $m^{\mathrm{acc}}_{\mathrm{A}}$. Further states include the temperatures of the reactor T_{R}, the vessel T_{S}, the jacket T_{M}, the fluid mix in the external heat exchanger T_{EK} and the coolant flowing out of the external heat exchanger T_{AWT}, which are used to describe the energy balances of the system. The details including all the parameter values can be found in Lucia et al. (2014) and the code is also publicly available[1]. The available control inputs are the feed flow $\dot{m}_{\mathrm{F}}$, the coolant temperature at the inlet of the jacket $T^{\mathrm{IN}}_{\mathrm{M}}$ and the coolant temperature at the inlet of the external heat exchanger $T^{\mathrm{IN}}_{\mathrm{AWT}}$. An auxiliary variable which plays a major role for the safe control of the reactor is T_{adiab}, which describes the temperature that would be reached if the cooling fails. As an important safety constraint, T_{adiab} always needs to be lower than a certain threshold.

In each batch, a fixed amount of polymer needs to be produced as fast possible, which coincides with minimizing the batch time. At the same time, the constraints for the safe operation of the reactor ($T_{\mathrm{adiab}} \leq T_{\mathrm{adiab,max}}$), a high-quality product ($T_{\mathrm{R,min}} \leq T_{\mathrm{R}} \leq T_{\mathrm{R,max}}$) and the physical limitations of the actuators need to be satisfied. The two uncertain parameters are k_{U1} and k_{U2} and are considered constant throughout one batch.

5. Results

This section presents how the learning-based approximations were obtained and the performance that is achieved for each one of the closed-loop configurations depicted in Fig. 2. A common and reasonable concern for the use of machine learning in a closed-loop is related to the unexpected performance that can be obtained when applied to situations that have not been *seen* previously during training. To analyze the robustness of both, the optimization-based and the learning-based approaches, we present the results for the case when the intervals of parametric uncertainties and possible initial conditions are larger than those used for training, or assumed in the optimization-based formulations. We also show that the impact of the approximation quality can be analyzed using the sensitivity considerations described in Section 3. Additionally, the efficacy of the proposed approach is highlighted by deploying the estimation and control networks on a low-cost microcontroller.

[1] www.do-mpc.com.

5.1. Data generation

For the generation of the training data we ran 200 simulations of the batch process with MHE in connection with MPC, which corresponds to the second configuration in Fig. 2. For the multi-stage NMPC scheme we used a prediction horizon $N = 20$ and a robust horizon $N_{\mathrm{r}} = 1$ with a sampling time of $t_{\mathrm{s}} = 50s$. The MHE implementation used the same sampling time as the controller and the horizon was chosen to $N_{\mathrm{est}} = 10$ which was sufficient for the proper estimation of the uncertain parameters k_{U1} and k_{U2} and consequently provided good state estimates. For each simulation of a batch, the initial state, the estimated initial state and the value of the uncertain parameters were varied according to the values given in the *Training intervals* column of Table 2 following a uniform distribution. We used do-mpc (Lucia et al., 2017) to generate the simulation results, as it supports multi-stage NMPC and MHE natively. It relies on IPOPT (Wächter and Biegler, 2006) for the resulting nonlinear programming problems and CasADi (Andersson et al., 2019) to obtain exact derivatives based on automatic differentiation. The simulation was carried out via the integrators provided by the SUNDIALS toolbox (Hindmarsh et al., 2005) ensuring a high accuracy.

Remark 2. In general no statements can be made about how many samples are necessary and how they need to be generated such that the neural networks obtain an approximation quality which leads to the satisfaction of desired performance criteria. Additionally, the resulting learned networks are obtained from stochastic optimization algorithms which renders deriving analytical guarantees impossible. If performance guarantees are required, one can resort to a posteriori probabilistic validation on closed-loop trajectories (Karg et al., 2019; Karg and Lucia, 2019).

5.2. Training of neural networks

The 200 generated trajectories contain 23,299 data tuples of which 90% were used for training the different neural network approaches and 10% were reserved for computing the validation error. While for learning the NNC all tuples can be considered, the number of available samples is reduced to 21,299 for NNE and NNEC. Since in every batch the first $N_{\mathrm{est}} = 10$ tuples are used to initialize the input to the networks, there are $200 \times 10 = 2000$ less data samples. The data was scaled such that the input and output samples for the networks belonged to the unit hypercube. The training of the neural networks was carried out via Keras (Chollet et al., 2015) relying on the Tensorflow (Abadi, 2015) backend using Adam (Kingma and Ba, 2014) as the optimization algorithm, which is a variant of stochastic gradient descent.

The validation error throughout the training progress averaged over 20 training runs, computed via (10), is visualized in Fig. 3. It can be seen that the validation error decreases steadily for NNC while the curves for NNE and NNEC are characterized by various peaks. The minor peaks are due to the stochastic nature of the optimization algorithm and can also be seen for NNC. The major peaks show that small changes in the network parameters might lead to a large change in the approximation quality, which can be seen as an indicator of the fact that that learning an estimator is a more difficult problem than learning a controller. To take into account the complexity of the learning task, the size of the network chosen for the NNC was the smallest and for the NNEC approach the largest, because it incorporates the estimator and the controller. Further increasing the size of the estimator networks smoothened the curves of the validation error, but led to larger validation errors and the obtained larger networks performed worse in closed-loop simulations than the ones considered in the rest of this work. Investigations on the impact of

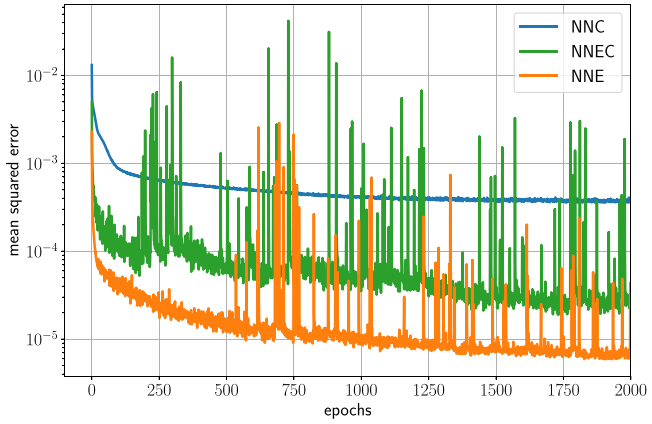

Fig. 3. Average validation mean squared loss of 20 training runs over 2000 epochs for the learned controller, estimator and the simultaneous approach. The samples were scaled such that the inputs and outputs belong to the unit hypercube with respective dimensionality.

Table 1
Architectural details of the networks including number of inputs n_{in}, neurons per hidden layer M, number of hidden layers L and number of outputs n_{out}.

	n_{in}	M	L	n_{out}
NNC	9	25	6	3
NNE	109	80	6	9
NNEC	109	120	8	12

the network depth and width on validation error and performance is out of the scope for this work and a potential topic for future work. The architectural details of the three considered networks are listed in Table 1. For the subsequent investigations the network parameters providing the lowest validation error throughout the 2000 training epochs were selected.

5.3. Optimization-based vs. learning-based approaches

The six configurations summarized in Fig. 2 were evaluated for 100 randomly generated settings within the training intervals. Each setting is defined by the real initial value of the system x_{init}, the corresponding initial guess x_{est} used for the estimator and the controller, and the realization of the uncertain parameters. Note that the parameters remain uncertain but constant for the whole batch irrespective of the controller used in the closed-loop. The possible variations of all these parameters and the measurement noise are given in Table 2 with $e_{\text{init}} = x_{\text{init}} - x_{\text{est}}$.

The quantitative results for the 100 tests that use the same intervals (but not the same realizations within the intervals) as those used for training are summarized in Table 3. Due to the economic objective, it is optimal to operate the reactor at the bounds of T_{R}. This leads to constraint violations even for the baseline, where MPC is used in connection with state-feedback as it can be seen in the first row of Table 3. These minor violations are caused by discretization errors. The 100 evaluated trajectories for the considered configurations are presented in Fig. 4. All configurations provide a comparable performance with exception of NNEC. The reasons for the inferior performance of NNEC will be explained in detail in the following sections. The performance of the NNEC could not be enhanced by changing its architecture, i.e. simply choosing a larger network for NNEC did not improve its closed-loop performance.

5.4. Robustness of deep-learning-based approximations

From the previous results it is clear that the learned estimators and controllers provide a solid performance within the domain they were trained for. But during operation, cases might occur where the system leaves the domain which was covered by the training procedure. To analyze the performance for unseen scenarios, we analyzed the behavior of the different configurations, when both uncertainties vary between ±30% instead of ±20%. Additionally, the interval in which the initial values of the system may lie and the deviation of the estimated initial state from the real state is enlarged too. For the exact values please consider Table 2.

In the same manner as for the investigation with the training intervals, we tested each closed-loop configuration for 100 randomly generated scenarios with the extended intervals. For the MHE formulation, we adapted the bounds of the parameters (4d) to match the ±30% uncertainty, while the considered uncertainty in the scenario tree of the MPC was kept unchanged. To avoid infeasible optimization problems when solving the exact MPC problem (2) due to the enlarged uncertainty intervals that are not considered in the scenario tree, the critical constraints were implemented as soft constraints. The results are summarized in Table 4 and the 100 trajectories of the reactor temperature for all six configurations are shown in Fig. 5.

The results show that, as expected, the performance for all approaches degrades due to the enlarged uncertainty and initial condition intervals. This can be clearly seen in larger constraint violations. However, the degradation of all six configurations is of the same order and only the NNEC approach has significant average constraint violations for the reactor temperature, as it can be seen in Fig. 5. What we would like to point out is that the learning-based approaches, when exposed to unseen scenarios, do not *necessarily* have a worse (average or worst-case) performance in comparison to the optimization-based approaches MHE and MPC when wrong assumptions are made.

5.5. Performance deterioration due to learning-based approximations

In the previous sections it was shown that the learning-based methods can achieve a performance level comparable to the online optimization-based methods. This section illustrates the fact that the same approximation error of a neural network estimator or a neural network controller can have a very different influence on the closed-loop performance.

To analyze this effect, we make use of the sensitivities described in Section 3 and compute them for a given trajectory. For every step along a trajectory (except for the first N_{est}) the tuple (s_p, x_p, u_p^*, d_p) is available, where x_p is the current state of the system, u_p^* is the optimal input computed based on the current state, d_p is the parameter realization and s_p carries the information necessary for the estimator.

The local sensitivities w.r.t. to errors in the computed input and errors in the estimated state can be directly computed via (13) and (16). As shown in (18) and (19), the maximum singular values of the sensitivities can be used to obtain an approximate upper bound of the successor state error. The maximum singular value occurring throughout all n_{traj} considered trajectories can be computed via:

$$\sigma_{\max,x} = \max_{i,j} \left(\sigma_{\max}(J_{x,i,j})\right), \tag{21a}$$

$$\sigma_{\max,u} = \max_{i,j} \left(\sigma_{\max}(J_{u,i,j})\right), \tag{21b}$$

where $i = 1, \ldots, n_{\text{traj}}$ denotes the considered trajectory and $j = 1, \ldots, n_{\text{steps},i}$ denotes the time step of the ith trajectory. The average singular value along a trajectory can provide further insights

Table 2
Intervals of initial states and uncertainties as considered in the training process and when the different methods are exposed to extended scenarios. The deviation of the initial estimated state from real state is $e_{\text{init}} = x_{\text{init}} - x_{\text{est}}$ The measurement noise is zero-mean gaussian and fully described by the standard deviation. States without measurement noise cannot be measured.

Parameter	Unit	Training intervals		Extended intervals		Meas. noise
		x_{init}	e_{init}	x_{init}	e_{init}	std. dev.
m_{W}	kg	[9900.0, 10100.0]	[−10.0, 10.0]	[9800.0, 10200.0]	[−15.0, 15.0]	1.0
m_{A}	kg	[851.0, 855.0]	[−2.0, 2.0]	[848.0, 858.0]	[−4.0, 4.0]	–
m_{P}	kg	[26.0, 27.0]	[−0.5, 0.5]	[22.0, 31.0]	[−0.5, 0.5]	–
T_{R}	K	[362.15, 364.15]	[−0.1, 0.1]	[361.65, 364.65]	[−0.1, 0.1]	0.1
T_{S}	K	[362.15, 364.15]	[−0.1, 0.1]	[361.15, 365.15]	[−0.2, 0.2]	0.1
T_{M}	K	[362.15, 364.15]	[−0.1, 0.1]	[361.15, 365.15]	[−0.2, 0.2]	0.1
T_{EK}	K	[306.15, 310.15]	[−0.1, 0.1]	[304.15, 312.15]	[−0.2, 0.2]	0.1
T_{AWT}	K	[306.15, 310.15]	[−0.1, 0.1]	[304.15, 312.15]	[−0.2, 0.2]	0.1
m_{monom}	kg	[290.0, 210.0]	[−2.0, 2.0]	[280.0, 320.0]	[−4.0, 4.0]	1.0
k_{U2}	–	[25.6, 38.4]		[22.4, 41.6]		–
k_{U1}	–	[3.2, 4.8]		[2.8, 5.2]		–

Table 3
Performance comparison of the five considered estimation and control configurations. All configurations were simulated for the same 100 randomly generated uncertainty realizations.

Configuration	Batch time			Violation $\mathbf{T}_{\text{R}}$		Violation $\mathbf{T}_{\text{adiab}}$	
	Average	Minimum	Maximum	Average	Maximum	Average	Maximum
SF + MPC	1.6065	1.5000	1.7639	0.0020	0.1712	0.0000	0.0000
MHE + MPC	1.6157	1.5000	1.7917	0.0032	0.9191	0.0013	0.1751
MHE + NNC	1.6518	1.4861	1.9028	0.0030	0.3013	0.0001	0.1606
NNE + MPC	1.6385	1.4444	1.7083	0.0111	1.0810	0.0071	1.0261
NNE + NNC	1.5243	1.4306	1.6528	0.0185	0.6795	0.0001	0.0013
NNEC	1.5635	1.4167	1.7778	0.6378	3.4836	0.0497	0.5810

on the importance of the approximation errors, and can be computed as:

$$\sigma_{\text{av},x} = \frac{1}{n_{\text{steps}}} \sum_{i=1}^{n_{\text{traj}}} \sum_{j=1}^{n_{\text{steps},i}} \sigma_{\text{av}}(J_{x,i,j}), \tag{22a}$$

$$\sigma_{\text{av},u} = \frac{1}{n_{\text{steps}}} \sum_{i=1}^{n_{\text{traj}}} \sum_{j=1}^{n_{\text{steps},i}} \sigma_{\text{av}}(J_{u,i,j}), \tag{22b}$$

where $n_{\text{steps}} = \sum_{i=1}^{n_{\text{traj}}} n_{\text{steps},i}$. Table 5 shows that the average singular values of the sensitivity w.r.t. the state estimate $\sigma_{\text{av},x}$ are higher than for the optimal input $\sigma_{\text{av},u}$. This means that a given error in the approximation of NNE leads in general to larger deviations from the optimal trajectory than for NNC.

The approximation errors for NNC and NNE are computed via (11) and (12). When NNEC is considered, which has the largest network structure since it needs to learn both the estimator and the controller, the approximation error for a sample k is given as:

$$r_k^{\text{EC}} = \left|\left| \begin{bmatrix} x_k \\ u_k^* \end{bmatrix} - \mathcal{N}_{\text{EC}}(s_k) \right|\right|_2. \tag{23}$$

The maximum and average approximation errors for each network $r_{i,j} \in \{r_{i,j}^{\text{C}}, r_{i,j}^{\text{E}}, r_{i,j}^{\text{EC}}\}$ can be analyzed in an equivalent fashion as the singular values. By replacing $\sigma_{\max}(J_{\cdot,i,j})$ in (21) and $\sigma_{\text{av}}(J_{\cdot,i,j})$ in (22) with $r_{i,j}$, the maximum approximation errors ($r_{\max} \in \{r_{\max}^{\text{C}}, r_{\max}^{\text{E}}, r_{\max}^{\text{EC}}\}$) and average approximation errors ($r_{\text{av}} \in \{r_{\text{av}}^{\text{C}}, r_{\text{av}}^{\text{E}}, r_{\text{av}}^{\text{EC}}\}$) are obtained. The values for the resulting approximated errors are summarized in the upper half of Table 5. It can be seen that the approximation error of NNEC exceeds both the

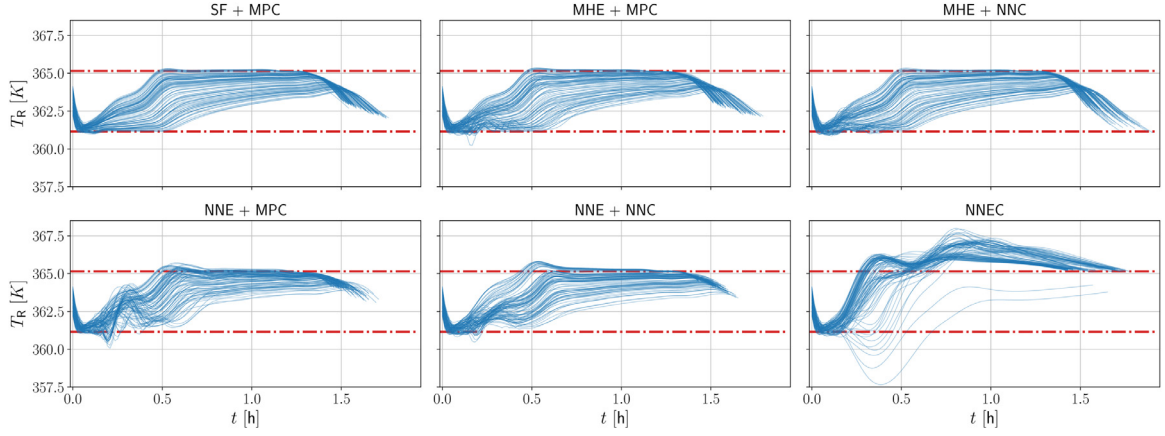

Fig. 4. Reactor temperature for 100 scenarios randomly generated with the training intervals from Table 2 for all six considered configurations (Fig. 2).

Table 4
Performance comparison of the five considered estimation and control configurations. All configurations were simulated for the same 100 randomly generated uncertainty realizations.

Configuration	Batch time			Violation $\mathbf{T}_\mathrm{R}$		Violation $\mathbf{T}_\mathrm{adiab}$	
	Average	Minimum	Maximum	Average	Maximum	Average	Maximum
SF + MPC	1.6199	1.4583	1.9167	0.0357	1.1692	0.0001	0.0124
MHE + MPC	1.6282	1.4583	1.9028	0.0401	1.2872	0.0015	0.2865
MHE + NNC	1.6722	1.4583	2.0417	0.0418	0.9633	0.0003	0.2078
NNE + MPC	1.5478	1.4028	1.7778	0.0371	1.7051	0.0202	2.8845
NNE + NNC	1.5329	1.4028	1.7639	0.0582	2.4689	0.0003	0.2281
NNEC	1.5565	1.3889	1.8333	0.6279	3.1819	0.0001	0.1138

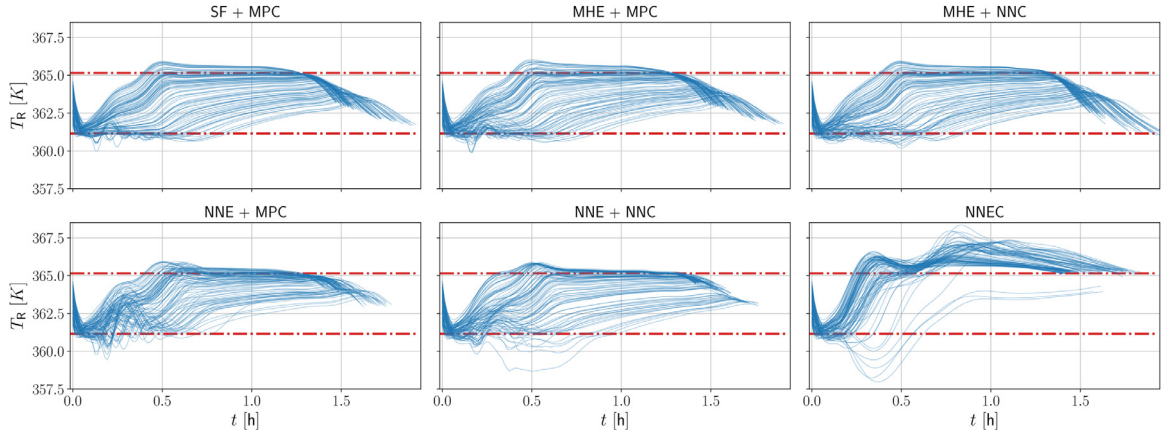

Fig. 5. Reactor temperature for 100 scenarios randomly generated with intervals larger than in the training process (see extended intervals in Table 2) for all six considered configurations (Fig. 2).

Table 5
The maximum and average approximation errors are computed based on (12), (11) and (23). The maximum and average singular values are computed via (21) and (22). The sub-optimality of the resulting control inputs for the three considered configurations are given in (24).

	NNE + MPC	SF + NNC	NNEC
r_max	1873.09	7054.83	15935.78
r_av	450.29	185.36	7500.17
σ_max	51887.967	100.00	–
σ_av	1075.38	53.75	–
e_u	1300.34	185.36	4116.96

largest maximum and average approximation error of NNE and NNC. The maximum and the average approximation errors for NNC and NNE are of the same magnitude, but the significantly higher sensitivity w.r.t. to the state estimate $\sigma_{\mathrm{av},x}$ means that replacing MHE with NNE will lead to a larger performance deterioration than replacing MPC with NNC.

To directly quantify the effect of the approximation error in the closed loop, we can also compare the optimal input u_p^* to the sub-optimal inputs provided by: (i) NNC ($\mathcal{N}_\mathrm{C}(x_p) = u_p^\mathrm{C}$); (ii) when the optimal input is computed by solving (2) based on the estimate $x_{\mathrm{est},p} = \mathcal{N}_\mathrm{E}(s_p)$ resulting in u_p^E; (iii) when NNEC is used with the estimated state ($\mathcal{N}_\mathrm{EC}(s_p) = [x_{\mathrm{est},p}, u_p^\mathrm{EC}]^T$). The average root mean squared error of the euclidean distance of the approximate control inputs from the optimal control inputs along all trajectories, denoted as e_u, is then defined by:

$$\text{SF + NNC: } e_u = \frac{1}{n_\mathrm{steps}} \sum_{i=1}^{n_\mathrm{traj}} \sum_{j=1}^{n_{\mathrm{steps},i}} \|u_{i,j}^\mathrm{C} - u_{i,j}^*\|_2, \tag{24a}$$

$$\text{NNE + MPC: } e_u = \frac{1}{n_\mathrm{steps}} \sum_{i=1}^{n_\mathrm{traj}} \sum_{j=1}^{n_{\mathrm{steps},i}} \|u_{i,j}^\mathrm{E} - u_{i,j}^*\|_2, \tag{24b}$$

$$\text{NNEC: } e_u = \frac{1}{n_\mathrm{steps}} \sum_{i=1}^{n_\mathrm{traj}} \sum_{j=1}^{n_{\mathrm{steps},i}} \|u_{i,j}^\mathrm{EC} - u_{i,j}^*\|_2. \tag{24c}$$

The values for the 100 trajectories are summarized in the last row of Table 5. The difference between the optimal input and the control input provided by NNEC is the largest which is consistent with its inferior performance observed in Fig. 4. The deviation caused when the learning-based estimator is used is significantly larger than the deviation caused by the learning-based controller, which confirms the importance of the larger sensitivities despite similar average and maximum approximation errors (r_max, r_{av}). This is also consistent with the better performance (smaller constraint violations) observed by the MHE + NNC configuration when compared to the NNE + MPC configuration.

5.6. Embedded implementation

We believe that the presented technique enables the deployment of advanced control and decision making algorithms on hardware with limited computational resources, as the major work load, consisting of generating data and training the neural networks, is shifted offline. The online part then only requires simple matrix-vector multiplications and the evaluation of nonlinearities, which for this work is the element-wise *tanh* function (7). This allows to apply said methods for extremely fast systems such as power electronics (Lucia et al., 2020a) and plasma jets (Bonzanini et al., 2020), when the computational power of the hardware is very limited (e.g. microcontrollers) or when large-scale

systems are considered, including the consideration of various uncertainties (Lucia and Karg, 2018).

Since neural networks are a collection of simple mathematical operations, implementing them on embedded devices is straightforward. We deployed the two networks for control and estimation on a low-power 32-bit microcontroller ARM Cortex-M0+ with 32 KB SRAM and 256 KB Flash memory running at 48 MHz. The necessary memory capacity was 35.3 KB and 166.7 KB for NNC and NNE, respectively. The evaluation times were 60.0 ms for NNC and 191.0 ms for NNE.

6. Conclusions

Optimization-based methods such as robust nonlinear model predictive control and moving horizon estimation provide means for controlling complex systems in an optimal manner, but are often computationally challenging. When very fast control sampling times are required or the uncertainty needs to be taken into account explicitly, solving the resulting optimization problems online might be intractable due to time or hardware constraints. By using deep neural networks to approximate the solution of the optimal control and estimation problem, these drawbacks can be mitigated, since neural networks are explicit functions composed of simple arithmetic operations which can be easily evaluated.

The three main messages of our paper can be summarized as: (i) both the control problem (MPC) as well as the estimation problem (MHE) can be properly approximated using deep neural networks even for non-trivial case studies; (ii) the performance degradation obtained when exposed to previously unseen scenarios does not have to be larger than the degradation observed when solving the MPC and MHE problems exactly under the same wrong assumptions about the possible scenarios. The latter strongly depends on how the chosen solver handles infeasible optimization problems or on the constraint relaxation that is employed; (iii) the training error is not a good indicator for the closed-loop performance of an approximate MHE or MPC as it is strongly affected by the sensitivities of the problem as well as by the scaling factors used during training.

The learning-based approaches can be easily deployed on any computing hardware. We illustrate this fact by performing MPC and MHE of an industrial batch reactor on a low-cost microcontroller where a state estimate was provided within 191 ms and the corresponding control input within 60 ms.

Future work will include the rigorous statistical validation of the approximation quality and adaptive sampling to reduce the effort for the generation of the data sets.

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.

CRediT authorship contribution statement

Benjamin Karg: Conceptualization, Methodology, Software, Writing - original draft. **Sergio Lucia:** Conceptualization, Methodology, Supervision, Writing - review & editing.

Acknowledgments

The authors acknowledge the funding by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) grant number 423857295.

References

Abadi, M., 2015. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. Software available from tensorflow.org.
Alamo, T., Tempo, R., Luque, A., Ramirez, D.R., 2015. Randomized methods for design of uncertain systems: sample complexity and sequential algorithms. Automatica 52, 160–172.
Andersson, J.A.E., Gillis, J., Horn, G., Rawlings, J.B., Diehl, M., 2019. CasADi—A software framework for nonlinear optimization and optimal control. Math. Program. Comput. 11 (1), 1–36. doi:10.1007/s12532-018-0139-4.
Bayat, F., Johansen, T.A., Jalali, A.A., 2011. Flexible piecewise function evaluation methods based on truncated binary search trees and lattice representation in explicit MPC. IEEE Trans. Control Syst. Technol. 20 (3), 632–640.
Bemporad, A., Morari, M., Dua, V., Pistikopoulos, E.N., 2002. The explicit linear quadratic regulator for constrained systems. Automatica 38 (1), 3–20.
Bernardini, D., Bemporad, A., 2009. Scenario-based model predictive control of stochastic constrained linear systems. In: Proc. of the 48th IEEE Conference on Decision and Control, 2009., pp. 6333–6338.
Bonzanini, A.D., Paulson, J.A., Graves, D.B., Mesbah, A., 2020. Safe dose delivery in fast sampling atmospheric plasmas using projection-based approximate economic MPC. In: Proc. IFAC World Congr..
Chen, S., Saulnier, K., Atanasov, N., Lee, D.D., Kumar, V., Pappas, G.J., Morari, M., 2018. Approximating explicit model predictive control using constrained neural networks. In: 2018 Annual American control conference (ACC). IEEE, pp. 1520–1527.
Chollet, F., et al., 2015. Keras. https://github.com/fchollet/keras.
Fiacco, A.V., 1983. Introduction to Sensitivity and Stability Analysis in Nonlinear Programming. Elsevier.
Haßkerl, D., Lindscheid, C., Subramanian, S., Diewald, P., Tatulea-Codrean, A., Engell, S., 2018. Economics optimizing control of a multi-product reactive distillation process under model uncertainty. Comput. Chem. Eng. 118, 25–48.
Hertneck, M., Köhler, J., Trimpe, S., Allgöwer, F., 2018. Learning an approximate model predictive controller with guarantees. IEEE Control Syst. Lett. 2 (3), 543–548.
Hindmarsh, A.C., Brown, P.N., Grant, K.E., Lee, S.L., Serban, R., Shumaker, D.E., Woodward, C.S., 2005. SUNDIALS: suite of nonlinear and differential/algebraic equation solvers. ACM Trans. Math. Softw. (TOMS) 31 (3), 363–396.
James, F., Hoogland, J., Kleiss, R., 1997. Multidimensional sampling for simulation and integration: measures, discrepancies, and quasi-random numbers. Comput. Phys. Commun. 99 (2-3), 180–220.
Johansen, T.A., 2004. Approximate explicit receding horizon control of constrained nonlinear systems. Automatica 40 (2), 293–300.
Karg, B., Alamo, T., Lucia, S., 2019. Probabilistic performance validation of deep learning-based robust NMPC controllers. arXiv:1910.13906.
Karg, B., Lucia, S., 2018. Deep learning-based embedded mixed-integer model predictive control. In: 2018 European Control Conference (ECC). IEEE, pp. 2075–2080.
Karg, B., Lucia, S., 2019. Learning-based approximation of robust nonlinear predictive control with state estimation applied to a towing kite. In: 2019 18th European Control Conference (ECC). IEEE, pp. 16–22.
Karg, B., Lucia, S., 2020. Efficient representation and approximation of model predictive control laws via deep learning. IEEE Trans. Cybern. 50, 3866–3878.
Karg, B., Lucia, S., 2020. Stability and feasibility of neural network-based controllers via output range analysis. In: 2020 59th IEEE Conference on Decision and Control (CDC). IEEE, pp. 4947–4954.
Kingma, D. P., Ba, J., 2014. Adam: A method for stochastic optimization. arXiv:1412.6980.
Krishnamoorthy, D., Foss, B., Skogestad, S., 2016. Real-time optimization under uncertainty applied to a gas lifted well network. Processes 4 (4), 52.
Krishnamoorthy, D., Suwartadi, E., Foss, B., Skogestad, S., Jäschke, J., 2018. Improving scenario decomposition for multistage MPC using a sensitivity-based path-following algorithm. IEEE Control Syst. Lett. 2 (4), 581–586.
Lee, J.H., Yu, Z.H., 1997. Worst-case formulations of model predictive control for systems with bounded parameters. Automatica 33 (5), 763–781.
Lucia, S., 2015. Robust Multi-Stage Nonlinear Model Predictive Control. TU Dortmund.
Lucia, S., Andersson, J., Brandt, H., Diehl, M., Engell, S., 2014. Handling uncertainty in economic nonlinear model predictive control: a comparative case-study. J. Process Control 24, 1247–1259.
Lucia, S., Finkler, T., Engell, S., 2013. Multi-stage nonlinear model predictive control applied to a semi-batch polymerization reactor under uncertainty. J. Process Control 23, 1306–1319.
Lucia, S., Karg, B., 2018. A deep learning-based approach to robust nonlinear model predictive control. IFAC-PapersOnLine 51 (20), 511–516.
Lucia, S., Navarro, D., Karg, B., Sarnago, H., Lucia, Ó., 2020a. Deep learning-based model predictive control for resonant power converters. IEEE Trans. Ind. Inform. 17 (1), 409–420.
Lucia, S., Subramanian, S., Limon, D., Engell, S., 2020b. Stability properties of multi-stage nonlinear model predictive control. Syst. Control Lett. 143, 104743.
Lucia, S., Tatulea-Codrean, A., Schoppmeyer, C., Engell, S., 2017. Rapid development of modular and sustainable nonlinear model predictive control solutions. Control Eng. Practice 60, 51–62.
Mayne, D., Seron, M., Rakovic, S., 2005. Robust model predictive control of constrained linear systems with bounded disturbances. Automatica 41, 219–224.
Nubert, J., Köhler, J., Berenz, V., Allgöwer, F., Trimpe, S., 2020. Safe and fast tracking on a robot manipulator: Robust MPC and neural network control. IEEE Robot. Autom. Lett. 5 (2), 3050–3057.

Parisini, T., Zoppoli, R., 1995. A receding-horizon regulator for nonlinear systems and a neural approximation. Automatica 31 (10), 1443–1451.
Scokaert, P., Mayne, D., 1998. Min-max feedback model predictive control for constrained linear systems. IEEE Trans. Autom. Control 43 (8), 1136–1142.
Shapiro, A., 2009. Lectures on Stochastic Programming: Modeling and Theory. SIAM.
Subramanian, S., Lucia, S., Engell, S., 2018. A synergistic approach to robust output feedback control: tube-based multi-stage NMPC. IFAC-PapersOnLine 51 (18), 500–505.
Wächter, A., Biegler, L., 2006. On the implementation of a primal-dual interior point filter line search algorithm for large-scale nonlinear programming. Math. Program. 106, 25–57.
Yu, Z.J., Biegler, L.T., 2018. Advanced-step multistage nonlinear model predictive control. IFAC-PapersOnLine 51 (20), 122–127. 6th IFAC Conference on Nonlinear Model Predictive Control NMPC 2018.