Neural Network-Based Model Predictive Control with Input-to-State Stability

Katrine Seel[*,**], Esten I. Grøtli[**], Signe Moe[**], Jan T. Gravdahl[*] and Kristin Y. Pettersen[*]

***Abstract*— Learning-based controllers, and especially learning-based model predictive controllers, have been used for a number of different applications with great success. In spite of good performance, a lot of these cases lack stability guarantees. In this paper we consider a scenario where the dynamics of a nonlinear system are unknown, but where input and output data are available. A prediction model is learned from data using a neural network, which in turn is used in a nonlinear model predictive control scheme. The closed-loop system is shown to be input-to-state stable with respect to the prediction error of the learned model. The approach is tested and verified in simulations, by employing the controller to a benchmark system, namely a continuous stirred tank reactor plant. Simulations show that the proposed controller successfully drives the system from random initial conditions, to a reference equilibrium point, even in the presence of noise. The results also verify the theoretical stability result.**

I. Introduction

In the last decades, model predictive control (MPC) has established itself as the most important method for constrained control. MPC is a powerful control design because it enables the user to add constraints on a system's states and control inputs, in addition to optimize the different states and control objectives. Recent successes in the field of machine learning have contributed to an increased interest in the combination of the two, namely learning-based MPC.

MPC uses a prediction model to optimize the predicted behavior of a system over a limited time horizon. It is well known that the performance of an MPC scheme is closely related to the accuracy of its prediction model. In some cases it may be hard or impossible to create an accurate prediction model from first principles. An example is when either the entire or parts of the system dynamics are unknown. However, if data is available, machine learning methods can be used to remedy this.

The literature on general learning-based MPC is vast. For a more thorough overview, the reader is referred to [1]. In the following we focus on a scenario where either the entire or parts of the system dynamics are unknown, but with data available, so that a prediction model can be built or improved offline. In order to be used in MPC, the resulting prediction model should be accurate, while at the same time not too computationally complex.

[*] Department of Engineering Cybernetics, Norwegian University of Science and Technology, Trondheim, Norway, {katrine.seel, jan.tommy.gravdahl, kristin.y.pettersen}@ntnu.no

[**] Department of Mathematics and Cybernetics, SINTEF Digital, Trondheim/Oslo, Norway, {esteningar.grotli, signe.moe}@sintef.no

Different learning methods have been used for building or improving prediction models from data, suited for MPC. In [2], Gaussian process (GP) regression is used to improve a nominal model by learning a disturbance model, used to design an MPC algorithm for a mobile robot. Similarly, in [3], GPs and local linear regression methods are compared in order to improve the prediction model of a robot. Because GPs are known to scale poorly with the number of data points used in training, learning-based MPC has been tested using sparse GPs, such as in [4]. Neural networks (NNs) have also proven to be highly flexible function approximators, and have become state-of-the art for a variety of challenging tasks, ranging from image analysis and classification to language processing [5], [6]. This learning method has also proved effective for learning the dynamics of nonlinear systems [7], [8], [9]. Consequently, NNs have been used to build prediction models suited for MPC, such as in [10], [11], [12], [13].

For safety-critical applications, a necessary aspect of learning-based MPC, is being able to prove stability of the closed-loop system. For many references on learning-based MPC, this is currently missing. In [14], stability is analyzed for a general class of learning-based MPC, where the dynamics are divided into a nominal, linear model and a function to be learned. Using that the amplitude of the learned function is bounded, robust stability is proved for the linear MPC. Because the bound must hold for all unmodelled nonlinearities in addition to model errors, the assumption is rather conservative. In [15] and [16] the entire system dynamics are learned, using GPs and parameter optimized kinky inference (POKI) respectively, and stability is proved assuming boundedness of the modeling uncertainty. Stability is analyzed for learning-based MPC using neural networks, such as in [17] and [18]. However, both of these references consider more complex MPC designs than the design we consider in this paper.

This work is inspired by the stability analyses done in [15] and [16], but is adapted to a case where the system dynamics are learned using a NN. The main contributions of this paper are:

- Design of an output MPC scheme using a nonlinear autoregressive model with exogenous input (NARX) prediction model learned using a NN
- Conditions for which the closed-loop system is input-to-state stable (ISS) with respect to the prediction error
- Implementation and testing of the control design for a benchmark system

This is the author accepted version of an article published in
2021 American Control Conference (ACC)
https://doi.org/10.23919/ACC50511.2021.9483190

The rest of the paper is organized as follows. In Section II the problem statement of this paper is outlined. A detailed description of the prediction model is given in Section III, and in Section IV the NN used to learn the prediction model is presented. Subsequently, the stability of the resulting controller is analyzed in Section V. Section VI describes the continuous stirred tank reactor (CSTR) for which the MPC design was tested. Simulation results are presented in Section VII. Finally, conclusions are given in Section VIII.

II. Problem statement

We study nonlinear discrete-time systems, where $y_k \in \mathbb{R}$ is the measured output and $u_k \in \mathbb{R}$ is the control input. It is assumed that the dynamics can be described using a NARX model, given by

$$y_{k+1} = f(\mathbf{x}_k, u_k) + w_k, \tag{1}$$

where the output is corrupted by noise, w_k, which is assumed to lie in a compact set $\mathcal{W}$, that is $w_k \in \mathcal{W} \subset \mathbb{R}$. The input is subject to hard constraints, $u_k \in \mathcal{U}$, and the output is subject to soft constraints, $y_k \in \mathcal{Y}$. The NARX state vector is given by

$$\mathbf{x}_k = [y_k, \ldots, y_{k-m_y}, u_{k-1}, \ldots, u_{k-m_u}], \tag{2}$$

where m_y and m_u denotes the number of previous outputs and inputs used to describe the next output, so that $\mathbf{x}_k \in \mathbb{R}^{m_y+m_u+1}$.

It is assumed that the system dynamics of the plant are unknown, and that $f(\cdot,\cdot)$ from (1) is an unknown function, but that input and output data are available. The data is used to build a prediction model

$$\hat{y}_{k+1} = \hat{f}(\mathbf{x}_k, u_k). \tag{3}$$

Throughout this paper, $\hat{(\cdot)}$ will be used to denote either variables that are, or functions that are used to make, predictions. We want to steer the process from random initial conditions, within the given input and output sets, to a desired reference equilibrium point, given by $(\mathbf{x}_{\text{ref}}, u_{\text{ref}})$, where $\mathbf{x}_{\text{ref}} \in \mathbb{R}^{m_y+m_u+1}$ and $u_{\text{ref}} \in \mathbb{R}$. To this end, we design an MPC scheme, using the the prediction model from (3).

Remark 1. *Here, we have considered a case which is single-input single-output. However, the control design and the stability results in this paper can easily be extended to the multi-input multi-output case.*

III. Data-based prediction model

A. NARX prediction model

In the MPC algorithm we consider prediction of the state. We therefore reformulate the NARX model into a state space representation on the form

$$\begin{aligned} \mathbf{x}_{k+1} &= \mathbf{F}(\mathbf{x}_k, u_k, e_k) \\ y_k &= \mathbf{c}^\intercal \mathbf{x}_k + w_k, \end{aligned} \tag{4}$$

where $\mathbf{c}^\intercal = [1 \quad 0 \quad \ldots \quad 0]$ and e_k represents modeling uncertainty, in the form of a prediction error, introduced because we will use the prediction model from (3). The prediction error denotes the difference between the true model (1) and the learned prediction model (3)

$$\begin{aligned} e_k &:= y_{k+1} - \hat{y}_{k+1} \\ &= f(\boldsymbol{x}_k, u_k) + w_k - \hat{f}(\boldsymbol{x}_k, u_k). \end{aligned} \tag{5}$$

Using the definition of the NARX state vector from (2), the vector-valued function from (4) is given by

$$\begin{aligned} \mathbf{F}(\mathbf{x}_k, u_k, e_k) = [\hat{f}(\mathbf{x}_k, u_k) + e_k, y_k, ..., y_{k-m_y+1}, \\ u_k, ..., u_{k-m_u+1}]. \end{aligned} \tag{6}$$

We now define the nominal state space model, for which $\hat{\mathbf{F}}(\mathbf{x}_k, u_k) \triangleq \mathbf{F}(\mathbf{x}_k, u_k, 0)$. This is written as

$$\hat{\mathbf{F}}(\mathbf{x}_k, u_k) = [\hat{f}(\mathbf{x}_k, u_k), y_k, ..., y_{k-m_y+1}, u_k, ..., u_{k-m_u+1}], \tag{7}$$

where the first argument can be either the previous predicted state, $\hat{\mathbf{x}}_k$, or the true state known from measurements, $\mathbf{x}_k$, and is used to obtain the next state prediction $\hat{\mathbf{x}}_{k+1} = \hat{\mathbf{F}}(\mathbf{x}_k, u_k)$.

IV. NARX Neural Network

When a multi-layer perceptron (MLP) network is used to approximate the function $f(\cdot)$ in a NARX model, the resulting structure is known as a NARX network [19]. Here, a single hidden layer is proposed for approximation. It has been shown that a single hidden layer feedforward NN with a sufficient number of neurons, is capable of approximating any continuous function to an arbitrary degree of accuracy [20], [21].

Training of a NARX network can be done in two different modes [19]. If the network is in so-called series-parallel mode, the previous outputs are actual values of the system's output. This can be written as

$$\hat{y}_{k+1} = \hat{f}_{\text{NN}}(y_k, ..., y_{k-m_y}, u_k, ..., u_{k-m_u}). \tag{8}$$

If the network is in parallel mode, predicted past outputs are used to predict the next output, according to

$$\hat{y}_{k+1} = \hat{f}_{\text{NN}}(\hat{y}_k, ..., \hat{y}_{k-m_y}, u_k, ..., u_{k-m_u}). \tag{9}$$

In parallel mode, the NARX network is a recurrent neural network (RNN), with feedback connections enclosing the layers of the network.

Series-parallel mode is generally preferred in training if the actual values of the system's outputs are available. This has two benefits, the first being that using the actual output values gives better accuracy. In addition, the series-parallel architecture results in a purely feedforward network, so that static backpropagation methods can be used during training.

In order to make predictions, the NARX network should be converted to parallel mode after training. When the NARX network is making multi-step ahead predictions, the true output values will no longer be available, and in parallel mode the network will instead use its own output predictions. A NARX network in parallel mode is visualized in Figure 1.

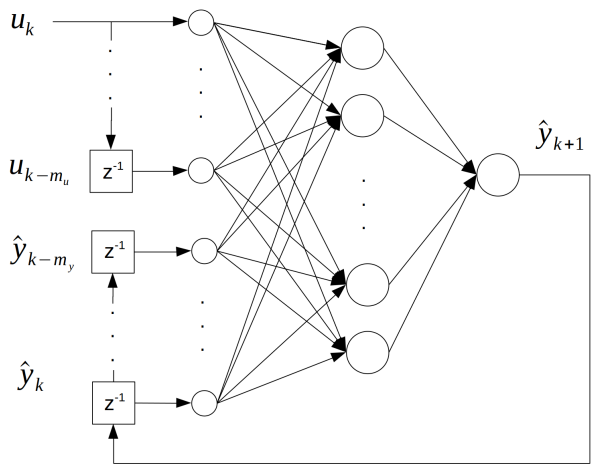


Fig. 1: Visualization of a NARX network in parallel mode, reproduced from [22].

V. Stabilizing data-based MPC

In the following section, we define a nonlinear MPC scheme that uses the nominal model (7) learned from data using a NARX network. The following notation is used. A continuous function $\alpha : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ is a class $\mathcal{K}$-function if $\alpha(0) = 0$ and it is strictly increasing. A class $\mathcal{K}$-function that is also unbounded, that is $\alpha(s) \to \infty$ as $s \to \infty$, is a class $\mathcal{K}_\infty$-function. A function $\beta : \mathbb{R}_{\geq 0} \times \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ is a class $\mathcal{KL}$-function if $\beta(s,t)$ is of class $\mathcal{K}$ with respect to s, $\beta(s,t)$ is decreasing with respect to t and $\beta(s,t) \to 0$ as $t \to \infty$. We will use $\|\cdot\|$ to denote the 2-norm.

A. Optimal control problem

MPC is based on solving an open-loop optimization problem at every time step. In each iteration the optimization routine will minimize a given cost function.

Because we want the MPC scheme to drive the process to a reference state, we use a positive definite stage cost that penalizes deviations from the reference, given by $\ell_s(\cdot,\cdot)$. A barrier function, $\ell_b(\cdot)$, is added to the total stage cost, in order to fulfill soft output constraints, that is to ensure that the output stays in a certain set, $\mathcal{Y} \subseteq \mathbb{R}$, if possible. Based on the stability analysis in Section V-B, $\ell_b(\cdot)$ is designed such that

$$\ell_b(\hat{y}_k) \geq \alpha_b(d(\hat{y}_k, \mathcal{Y})) \tag{10}$$

and $\ell_b(\hat{y}_k) = 0 \; \forall \; \hat{y}_k \in \mathcal{Y}$, where $\alpha_b(\cdot)$ is a class $\mathcal{K}$-function and $d(\cdot)$ is the shortest distance from a point $\hat{y}_k \in \mathbb{R}$ to the set $\mathcal{Y} \subseteq \mathbb{R}$.

The total stage cost is then

$$\ell(\hat{\mathbf{x}}_k, u_k) = \ell_s(\hat{\mathbf{x}}_k - \mathbf{x}_{\text{ref}}, u_k - u_{\text{ref}}) + \ell_b(\hat{y}_k), \tag{11}$$

where $(\mathbf{x}_{\text{ref}}, u_{\text{ref}})$ denotes the reference equilibrium point. Using the stage cost (11) and the nominal model (7), the final optimization problem is formulated. Note that this is done without defining a terminal constraint [23], resulting in

$$\begin{aligned} \min_{\mathbf{u}_k} \sum_{i=0}^{N-1} \ell(\hat{\mathbf{x}}_{k+i|k}, u_{k+i|k}) + \lambda V_f(\hat{\mathbf{x}}_{k+N|k} - \mathbf{x}_{\text{ref}}) \\ \text{s.t} \quad \forall i \in [0, N-1]: \\ \hat{\mathbf{x}}_{k+i+1|k} = \hat{\mathbf{F}}_{\text{NN}}(\hat{\mathbf{x}}_{k+i|k}, u_{k+i|k}) \\ \hat{\mathbf{x}}_{k|k} = \mathbf{x}_k \\ \hat{y}_{k+i|k} = c^\intercal \hat{\mathbf{x}}_{k+i|k} \\ u_{k+i|k} \in \mathcal{U}, \end{aligned} \tag{12}$$

where $\hat{(\cdot)}$ denotes predicted variables. We let $\hat{\mathbf{F}}_{\text{NN}}(\cdot,\cdot)$ denote the nominal state space model (7), using the neural-based prediction model (9). The control sequence applied over the prediction horizon N is given by $\mathbf{u}_k = \{u_{k|k}, ..., u_{k+N-1|k}\}$. The optimization problem is subject to hard input constraints, where $\mathcal{U}$ is the set of feasible inputs. In order to guarantee stability, we make use of a terminal cost $V_f(\cdot)$ and an associated terminal control law $\kappa_f(\cdot)$. The terminal cost is weighted by a design parameter $\lambda \geq 1$ and $\mathbf{x}_k$ is the initial condition of the NARX state.

B. Stability

We want to consider ISS of the closed-loop system

$$\mathbf{x}_{k+1} = \mathbf{F}(\mathbf{x}_k, \kappa_{\text{MPC}}(\mathbf{x}_k), e_k), \tag{13}$$

where $\kappa_{\text{MPC}}(\mathbf{x}_k)$ is the nominal controller resulting from the optimization problem (12), and $\mathbf{F}(\mathbf{x}_k, \kappa_{\text{MPC}}, e_k)$ is the true model given by (6). The stability proof for the learning-based MPC in [15] and [16] is here adapted for an MPC scheme where the prediction model is approximated by an NN.

Definition 1. *A system $\mathbf{x}_{k+1} = \mathbf{F}(\mathbf{x}_k, \kappa_{\text{MPC}}(\mathbf{x}_k), e_i)$ is ISS if there exists a class $\mathcal{KL}$-function β and a class $\mathcal{K}$-function γ such that*

$$\|\mathbf{x}_k\| \leq \beta(\|\mathbf{x}_0\|, k) + sup_{i=0,\dots,k} \gamma(|e_i|) \tag{14}$$

for all initial conditions $\mathbf{x}_0$, modeling uncertainties e_i and for all $k \geq 0$.

We first establish stability in the nominal case, for which the prediction error is assumed to be zero, so that the true and predicted state is the same. We then consider robust stability, when the prediction error is non-zero, but bounded. For the nominal case, we want to establish that the control error, $\tilde{\mathbf{x}}_k = \mathbf{x}_k - \mathbf{x}_{\text{ref}}$, converges asymptotically to zero.

For now, we consider the following closed-loop system using the nominal model (7), with the neural prediction model (9)

$$\begin{aligned} \tilde{\mathbf{x}}_{k+1} &= \hat{\mathbf{F}}_{\text{NN}}(\mathbf{x}_k, \kappa_{\text{MPC}}(\mathbf{x}_k)) - \mathbf{x}_{\text{ref}} \\ &= \tilde{\mathbf{F}}_{\text{NN}}(\tilde{\mathbf{x}}_k, \kappa_{\text{MPC}}(\tilde{\mathbf{x}}_k)). \end{aligned} \tag{15}$$

We make use of the following assumptions regarding the total stage cost, $\ell(\mathbf{x}_k, u_k)$, the terminal cost, $V_f(\cdot)$, and the terminal controller, $\kappa_f(\cdot)$.

Assumption 1. *There exists a terminal control law $\kappa_f(\tilde{\mathbf{x}}_k)$, a control Lyapunov function $V_f(\tilde{\mathbf{x}}_k)$ and a region defined by*

$\Omega = \{\tilde{\mathbf{x}}_k \in \mathbb{R}^{m_y+m_u+1} : V_f(\tilde{\mathbf{x}}_k) \leq \alpha\}$, *where* $\alpha > 0$, *such that* $\forall \tilde{\mathbf{x}}_k \in \Omega$ *the following holds*

1)

$$\begin{aligned} \alpha_1(\|\tilde{\mathbf{x}}_k\|) \leq V_f(\tilde{\mathbf{x}}_k) \leq \alpha_2(\|\tilde{\mathbf{x}}_k\|), \\ V_f(\tilde{\mathbf{F}}_{\mathrm{NN}}(\tilde{\mathbf{x}}_k, \kappa_f(\tilde{\mathbf{x}}_k)) - V_f(\tilde{\mathbf{x}}_k) \\ \leq -\ell(\tilde{\mathbf{x}}_k + \mathbf{x}_{\mathrm{ref}}, \kappa_f(\tilde{\mathbf{x}}_k)), \end{aligned} \tag{16}$$

2)

$$\kappa_f(\tilde{\mathbf{x}}_k) \in \mathcal{U}, c^{\intercal}(\tilde{\mathbf{x}}_k + \mathbf{x}_{\mathrm{ref}}) \in \mathcal{Y}, \tag{17}$$

where $\tilde{\mathbf{F}}_{\mathrm{NN}}(\cdot,\cdot)$ *is the nominal model (15),* $\ell(\cdot,\cdot)$ *is the stage cost (11),* $\mathbf{x}_{\mathrm{ref}}$ *is the reference state,* $\alpha_1(\cdot)$ *and* $\alpha_2(\cdot)$ *are class* $\mathcal{K}_\infty$*-functions and* $\mathcal{U}$*,* $\mathcal{Y}$ *are compact input and output sets, respectively.*

Assumption 2. *There exists a class* $\mathcal{K}_\infty$*-function,* $\alpha_y(\cdot)$*, such that the stage cost (11), satisfies* $\ell(\mathbf{x}_{\mathrm{ref}}, u_{\mathrm{ref}}) = 0$ *and* $\ell(\tilde{\mathbf{x}}_k + \mathbf{x}_{\mathrm{ref}}, u_k) \geq \alpha_y(\|\tilde{\mathbf{x}}_k\|)$ *for all* $u_k \in \mathcal{U}$*, where* $(\mathbf{x}_{\mathrm{ref}}, u_{\mathrm{ref}})$ *denotes the reference equilibrium point.*

We now propose the following theorem, which is an adaption of Theorem 4 in [23].

Theorem 1. *Let Assumptions 1-2 hold, and let* $\kappa_{\mathrm{MPC}}(\tilde{\mathbf{x}}_k)$ *be the resulting control law of the optimization problem in (12). Then* $\forall \lambda \geq 1$*, where* λ *is a weighting factor, there exists a domain of attraction,* $\mathcal{X}_N(\lambda)$*, defined without terminal constraint, such that for all initial conditions* $\mathbf{x}_0 \in \mathcal{X}_N(\lambda)$*, the closed-loop system (15) is asymptotically stable at the origin.*

Proof: Having satisfied Assumption 1-2, Theorem 4 in [23] states that the controller resulting from the solution of the optimization problem (12) will stabilize the system (15) asymptotically in the set $\mathbf{x}_k \in \mathcal{X}_N(\lambda)$. For the complete proof the reader is referred to [23].

The weight of the terminal cost, λ, can be used to adjust the size of the domain of attraction. The greater λ, the larger is $\mathcal{X}_N(\lambda)$, but the worse is the approximation, $\lambda V_f(\cdot)$, of the optimal cost.

Having established that the nominal system is asymptotically stable, the robust case will now be considered. To that end, we make use of the following assumptions:

Assumption 3. *There exists a constant* $\mu < \infty$*, such that the prediction error (5) is bounded* $\forall k \ |e_k| \leq \mu$*.*

Remark 2. *The prediction error encompasses the approximation error as well as the measurement noise. The measurement noise is assumed to lie in a compact set and is therefore bounded. In general it is not possible to prove that the approximation error of NNs is bounded. However, this is still a commonly used assumption [24], [25], [26]. In this case we can argue that if the prediction error is bounded on the training set, and the training data is representative of data seen in closed-loop operation, then Assumption 3 is reasonable.*

Assumption 4. *The nominal model* $\hat{\mathbf{F}}_{\mathrm{NN}}(\mathbf{x}_k, \kappa_{\mathrm{MPC}}(\mathbf{x}_k))$ *from (7) is uniformly continuous in* $\mathbf{x}_k$ *for all* $\mathbf{x}_k \in \mathcal{X}_{\mathcal{N}}(\lambda)$*.*

Remark 3. *The nominal model* $\hat{\mathbf{F}}_{\mathrm{NN}}(\mathbf{x}_k, \kappa_{\mathrm{MPC}}(\mathbf{x}_k))$ *will be uniformly continuous, if the prediction model* $\hat{f}_{\mathrm{NN}}(\mathbf{x}_k, \kappa_{\mathrm{MPC}}(\mathbf{x}_k))$ *is uniformly continuous. This is ensured by choosing uniformly continuous activation functions for the neurons in the layers of the NARX network. All the commonly used activation functions in RNNs, such as the sigmoid, tanh and ReLU function, are uniformly continuous as their derivatives are uniformly bounded. Considering the interval [0,*∞*), all the above-mentioned activation functions are in addition continuously differentiable, which is a stronger continuity property than uniform continuity.*

The following theorem is an adaption of Theorem 4 in [27].

Theorem 2. *Let* $\kappa_{\mathrm{MPC}}(\mathbf{x}_k)$ *be the predictive controller derived from the optimal control problem (12) and let Assumptions 1-4 hold. Then,* $\Omega_r \subseteq \mathcal{X}_N(\lambda)$ *is a robust invariant set for a sufficiently small bound on the uncertainty. The system (13) fulfills the ISS property within the robust invariant set* Ω_r*.*

Proof: Satisfaction of Assumptions 1-2 guarantees asymptotic stability of the nominal system (15) as stated by Theorem 1. If Assumption 3 and 4 also hold, then Theorem 4 of [27] holds directly. Satisfaction of Theorem 4 guarantees ISS for the closed-loop system (13).

Remark 4. *Because the continuity requirement to the neural-based prediction model relates to the choice of activation functions, the user has freedom to select a different number of hidden layers as well as other network architectures, such as other standard RNNs. Nonetheless, MPC is a computational heavy control design, because an optimization problem is solved at every iteration. This provides a valid argument for keeping the architecture as simple as possible in a neural-based MPC.*

Remark 5. *ISS can also be shown by establishing uniform continuity of the optimal cost, by adding some additional assumptions regarding the terminal cost and the stage cost used in the objective function. For details, the reader is referred to C1 in Proposition 1 in [27].*

VI. Case study

To test the performance and robustness of the proposed MPC design, the CSTR process is considered. This is often used as a benchmark process for learning-based MPC.

A. The continuous stirred tank reactor

A CSTR process describes the reaction where a reactant is converted from $A \rightarrow B$ [28]. The following differential equations are used to model this process

$$\dot{C}_A(t) = \frac{q_0}{V}(C_{Af} - C_A(t)) - k_0 e^{\frac{-E}{RT(t)}} C_A(t) \tag{18a}$$

$$\dot{T}(t) = \frac{q_0}{V}(T_f - T(t)) - \frac{\Delta H_r k_0}{\rho C_p} e^{\frac{-E}{RT(t)}} C_A(t) \tag{18b}$$

$$+ \frac{UA}{V\rho C_p}(T_c(t) - T(t)) \tag{18c}$$

$$\dot{T}_c(t) = \frac{T_r(t) - T_c(t)}{\tau}, \tag{18d}$$

TABLE I: CSTR process parameters

Param.	Definition	Value
q_0	Reactive input flow	10 l/min
V	Liquid volume in the tank	150 l
k_0	Frequency constant	$6{\cdot}10^{10}$ l/min
E/R	Arrhenius constant	9750 K
$-\Delta H_r$	Reaction enthalpy	10000 J/mol
UA	Heat transfer coefficient	70000 J/(min K)
ρ	Density	1100 g/l
C_p	Specific heat	0.3 J/(g K)
τ	Time constant	1.5 min
C_f	C_A in input flow	1 mol/l
T_f	Input flow temperature	370 K

where C_A [mol/l] is the concentration of the reactant, T [K] is the temperature in the tank, T_c [K] is the temperature of the coolant, and T_r [K] is the coolant temperature reference. For this control problem we have input and output according to $u = T_r$ and $y = C_A$. The model parameters are given in Table I, and are similar to those used in [15] and [16]. We define the following input constraints, $\mathcal{U} = \{335 \text{ K} \leq \text{T}_\text{r} \leq 372 \text{ K}\}$, and output constraints, $\mathcal{Y} = \{0.35 \text{ mol/l} \leq \text{C}_\text{A} \leq 0.65 \text{ mol/l}\}$.

B. Obtaining the dataset

The model equations in (18) were used in simulation and for generating training data, but otherwise assumed unknown. The equations were implemented in MATLAB, discretized using Euler's method and sampled at $T_s = 0.5$ min. An input signal was designed to do open-loop simulations. The requirement for input design for system identification is that the data needs to be persistently exciting, i.e the data must contain sufficiently many distinct frequencies [29].

The input signal was designed to cover the relevant area of input-output space, considering the input and output constraints. To this end, the system was excited by a total of 7 sweeping chirp signals, with different amplitudes and length. A total of 17500 data points were used to train the NN. As the resulting training data covers a large part of the input-output space which is considered in closed-loop operation, we assume that a bounded prediction error for the training data implies a bounded prediction error in closed-loop operation.

C. Training the NARX network

Training the NARX network was done in MATLAB, using a series-parallel architecture, as described in Section IV. All data was normalized ahead of training, so that all values fall within a range $[-1, 1]$. This was done to compensate for the input data having different scales, as both past inputs and past outputs are fed to the NN. For a NARX network, the size of the input layer is dictated by the number of input and output delays in the NARX model, given by m_u and m_y. Different architectures were tested, until we obtained satisfactory performance, using a single hidden layer, with 10 neurons, and $m_u = 1$ and $m_y = 2$ as the number of input and output delays, respectively. The sigmoid function was used in the neurons in the hidden layer, and a linear activation function was used in the output neuron. The data was split randomly into a train, test and validation set, that corresponded to 70/15/15 % of the data, respectively. A Levenberg-Marquardt algorithm was used for optimization during training. Early stopping was used to prevent the model from overfitting. For multi-step ahead prediction, the NARX network was converted to parallel mode (9).

The resulting architecture of the NN represents a trade-off between modeling flexibility, needed to capture the system dynamics, and computational complexity for the resulting control design. An independent dataset with noise, was set aside for validation, not previously seen in training. To quantify the prediction accuracy on the separate validation dataset, the mean squared error (MSE) is introduced

$$\text{MSE} = \frac{1}{N_\text{pred}} \sum_{k=1}^{N_\text{pred}} \|y_k - \hat{y}_k\|^2. \tag{19}$$

The validation results are seen in Figure 2, where the NN is tested for $N_\text{pred} = 30$, initialized using noisy measurements. For the given prediction horizon we get MSE $= 2.84 \cdot 10^{-6}$. Here the prediction horizon is relatively long, and the multi-step ahead prediction is fairly accurate for the unseen validation data. In closed-loop model-based control the prediction horizon will be often be shorter, and we conclude that the prediction model should be suited.

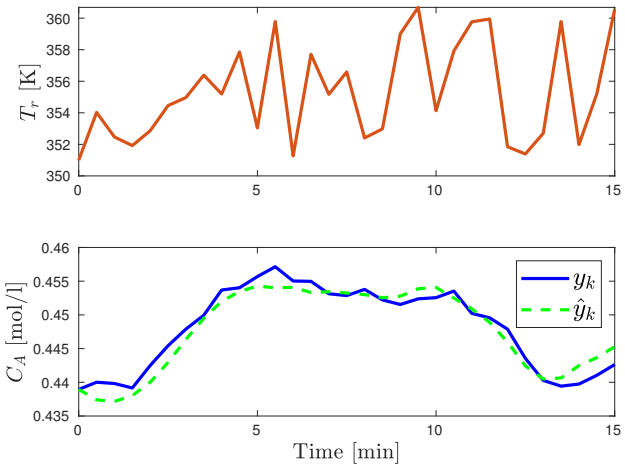


Fig. 2: Top: random steps in the coolant temperature reference. Bottom: true and multi-step ahead predictions of the concentration, with the network in parallel mode, using a separate validation set not previously seen by the network. The output is shown in original scale. The multi-step ahead prediction seems to be fairly accurate for a random input sequence, and for that reason more complex and larger architectures are avoided in order to keep the prediction model simple.

D. Control of the reactor

The control objective is to drive the process from a random initial condition, to a reference equilibrium point.

The following quadratic stage-cost was used

$$\ell_s(\mathbf{x}_k - \mathbf{x}_{\text{ref}}, u_k - u_{\text{ref}}) = (\mathbf{x}_k - \mathbf{x}_{\text{ref}})^\intercal \mathbf{Q}(\mathbf{x}_k - \mathbf{x}_{\text{ref}}) + (u_k - u_{\text{ref}})^\intercal \mathbf{R}(u_k - u_{\text{ref}}), \tag{20}$$

where $\mathbf{x}_{\text{ref}} = [y_{\text{ref}}, y_{\text{ref}}, y_{\text{ref}}, u_{\text{ref}}]$, for the selected choice of m_u and m_y, as addressed in Section VI-C. The optimization problem defined in (12) was solved using `fmincon` in MATLAB. The prediction horizon was set to $N = 5$, and we used $\mathbf{Q} = \text{diag}(50, 0, 0)$, $\mathbf{R} = 0.2$ and $\lambda = 1$.

E. Finding the reference equilibrium point

Because we consider a system with unknown dynamics, the input reference corresponding to the output reference, is not necessarily known. Here it is assumed that the input and output reference point is known a priori, namely $u_{\text{ref}} = 356$ K and $y_{\text{ref}} = 0.439$ mol/l. If this is not the case, the learned prediction model can be used to obtain an estimate of the corresponding input reference, by solving an optimization problem. See for example [12].

F. Soft output constraints

A barrier function was added to the stage cost to account for soft output constraints. The following barrier function was adopted from [15], [16]

$$\ell_b(y_k) = \zeta\left(1 - e^{\frac{-d(y_k, \mathcal{Y})}{\delta}}\right), \tag{21}$$

where $\delta = 1$, $\zeta = 100$ and $d(\cdot)$ is given by

$$d(y, \mathcal{Y}) = \inf_{z \in \mathcal{Y}} \|z - y\|_\infty , \tag{22}$$

where $\|\cdot\|_\infty$ denotes the infinity norm. For the benchmark system, this means that the coolant temperature reference will not exceed its limit, since input constraints are hard, while the concentration limit is "desirable", given by soft output constraints.

G. Finding the terminal cost

The selected terminal controller and the terminal cost are defined as $\kappa_f(\tilde{\mathbf{x}}_k) = \mathbf{K}^\intercal \tilde{\mathbf{x}}_k + u_{\text{ref}}$ and $V_f(\tilde{\mathbf{x}}_k) = \tilde{\mathbf{x}}_k^\intercal \mathbf{P} \tilde{\mathbf{x}}_k$ [30], where $\tilde{\mathbf{x}}_k = \mathbf{x}_k - \mathbf{x}_{\text{ref}}$. The terminal cost is found by solving the LQR problem for the linearized model around the reference point, where $\mathbf{P}$ is the solution to the discrete Ricatti equation.

The linearized model is found using the learned prediction model. For the choice of m_y and m_y, the NARX state vector is given by $\mathbf{x}_k = [y_k, y_{k-1}, y_{k-2}, u_{k-1}]$, and consequently the linearized model should be on the form

$$\begin{bmatrix} y_{k+1} \\ y_k \\ y_{k-1} \\ u_k \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} y_k \\ y_{k-1} \\ y_{k-2} \\ u_{k-1} \end{bmatrix} + \begin{bmatrix} b_{11} \\ 0 \\ 0 \\ 1 \end{bmatrix} u_k. \tag{23}$$

Because the learned model is not defined explicitly, the linearized model is obtained numerically [15], [16]. The coefficients of the linearized system can be approximated accoording to

$$a_{11} = \frac{\partial \hat{f}_{\text{NN}}}{\partial y_k},\ a_{12} = \frac{\partial \hat{f}_{\text{NN}}}{\partial y_{k-1}},\ a_{13} = \frac{\partial \hat{f}_{\text{NN}}}{\partial y_{k-2}} \quad a_{14} = \frac{\partial \hat{f}_{\text{NN}}}{\partial u_{k-1}}, b_{11} = \frac{\partial \hat{f}_{\text{NN}}}{\partial u_k}, \tag{24}$$

where $\hat{f}_{\text{NN}}(\cdot, \cdot)$ is the prediction model (9). The coefficients in (24) were evaluated at $(u_{\text{ref}}, y_{\text{ref}})$ and $(y_{\text{ref}} + \Delta, u_{\text{ref}} + \Delta)$, where Δ represents a small perturbation. We used $\Delta = 0.015$, found by trial and error.

The coefficients for the linearized system evaluated at the reference point, were found to be $a_{11} = 2.1742$, $a_{12} = -1.5402$, $a_{13} = 0.3958$, $a_{14} = -0.0003$ and $b_{11} = 0.0$. Using `dlqr` in MATLAB, the following gain and solution to the Ricatti equation were found:

$$\mathbf{K}^\intercal = \begin{bmatrix} -838.08 & 897.23 & -300.55 & 0.20 \end{bmatrix}, \tag{25}$$

$$\mathbf{P} = 10^6 \begin{bmatrix} 1.5672 & -1.6776 & 0.5619 & -0.0004 \\ -1.6776 & 1.7959 & -0.6015 & 0.0004 \\ 0.5619 & -0.6015 & 0.2015 & -0.0001 \\ -0.0004 & 0.0004 & -0.0001 & 0.0000 \end{bmatrix}. \tag{26}$$

Assumptions 1-4 from Section V-B are fulfilled given the choice of the terminal cost and terminal controller, the selected total stage cost, composed of (20) and (21), the considerations regarding the prediction error in closed-loop operation and the selected NN architecture and activation functions. By Theorem 2, the considered closed-loop system is ISS.

VII. Simulation results

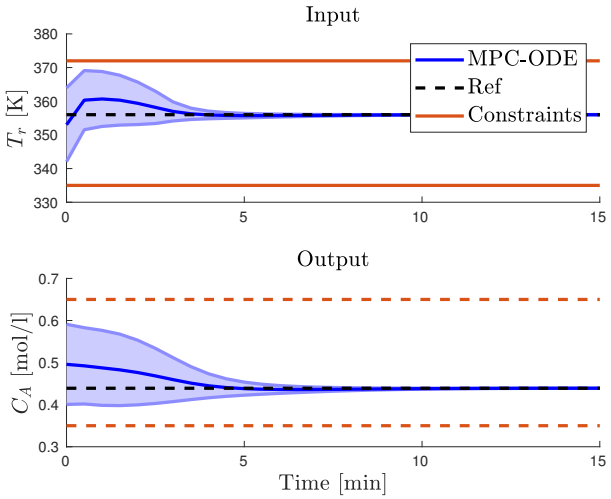
In order to evaluate the performance of the MPC-NN, using the NARX network as the prediction model, an MPC scheme using the ordinary differential equations (ODEs) (18), was implemented. To determine the terminal ingredients for this controller, the linearization of the system was also done numerically, as described by (24), but using the ODEs to determine the coefficients. The same MPC parameters were used for both the MPC-NN and the MPC-ODE design.

For each controller, simulations were run for $t = 15$ min. For each simulation random initial conditions were selected from the valid input and output sets. The measurements were corrupted by 2.5% sensor noise. A total of 100 simulations were run to test for different realizations of noise and initial conditions. The following performance index was used to evaluate the controller performance

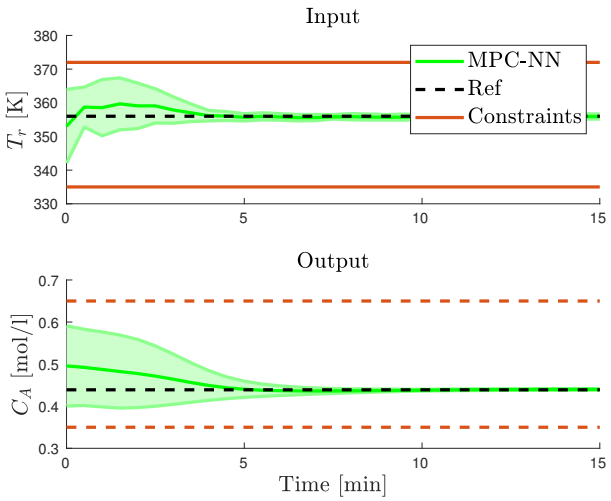
$$\phi = \frac{1}{N_{\text{sim}}} \sum_{j=1}^{N_{\text{sim}}} \sum_{k=1}^{t_{\text{sim}}} \ell(\mathbf{x}_k^j, u_k^j), \tag{27}$$

where $\mathbf{x}_k$ and u_k are the resulting state and control input for each iteration, t_{sim} is the number of time steps in one simulation and N_{sim} is the number of simulations.

In simulations the MPC-ODE design was expected to be superior, because the true model was used as the prediction



(a) MPC-ODE



(b) MPC-NN

Fig. 3: Results from 100 simulations. The thick line represents the emperical mean, and the shaded area represents $\pm\sigma$. In each figure, the upper subplot shows the coolant temperature reference and the bottom subplot shows the concentration. Note that the input constraints are hard, while the output constraints are soft.

model. Comparing the closed-loop results seen in Figure 3(a) and 3(b), we see that the MPC-NN performs almost as good as the ideal controller. This was also verified by the evaluated performance index (27), which was found to be $\phi_{\text{ODE}} = 273.86$ and $\phi_{\text{NN}} = 281.44$. For the tested scenario where the system dynamics are unknown, but input-output data is available, the proposed learning-based MPC design obtained close to ideal behavior.

Simulation results show that the MPC-NN successfully steered the system from all tested random initial conditions, to the reference equilibrium point, even in the presence of noise. In Figure 4 we also see that the one-step prediction error stays small and bounded for all realizations of noise. This is important with respect to Assumption 3 in the stability proof. The initial peak in the one-step prediction error is

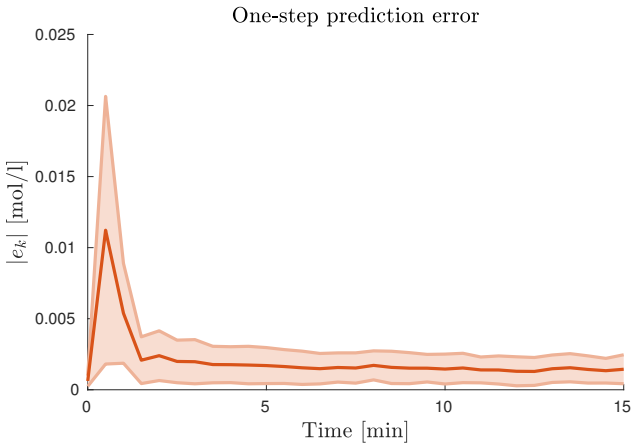


Fig. 4: The one-step prediction error for the prediction model used by the MPC-NN. The thick line represents the empirical mean, and the shaded area represents $\pm\sigma$ from 100 simulations.

due to the fact that at the beginning of each simulation, the network is initialized using only one noisy, random initial condition.

VIII. Conclusion

We have presented a nonlinear model predictive control scheme that uses a learned prediction model. The prediction model was learned based on past input and output data, using a neural network. The proposed controller is proved to be input-to-state stable with respect to the prediction error, assuming that the latter is sufficiently bounded. Because this is an output feedback MPC algorithm, full state information is not necessary, so the control design may be applied to systems where the state might not be measured. The control design was implemented and tested for a continuous stirred tank reactor. Compared to an ideal MPC implementation, the learning-based MPC design performed almost as good. Simulations also show stability of the controlled system for all investigated cases, and supports the assumption that the prediction error is bounded.

In future work, we will investigate the effect of training the neural network using noisy data. Also, it could be interesting to consider how to implement hard output constraints for the investigated MPC design.

Acknowledgment

The authors thank Kristian Gaustad Hanssen and Mark Haring at SINTEF Digital for valuable discussions. This work was supported by the industry partners Borregaard, Elkem, Yara, Hydro and the Research Council of Norway through the project TAPI: Towards Autonomy in Process Industries, project number: 294544.

References

[1] L. Hewing, K. P. Wabersich, M. Menner, and M. N. Zeilinger. Learning-Based Model Predictive Control: Toward Safe Learning in Control. *Annual Review of Control, Robotics, and Autonomous Systems*, 3(1):269–296, 2020.

[2] C. J. Ostafew, A. P. Schoellig, and T. D. Barfoot. Learning-based nonlinear model predictive control to improve vision-based mobile robot path-tracking in challenging outdoor environments. *IEEE International Conference on Robotics and Automation*, pages 4029–4036, 2014.

[3] C. D. McKinnon and A. P Schoellig. Learning probabilistic models for safe predictive control in unknown environments. *European Control Conference*, pages 2472–2479, 2019.

[4] L. Hewing, A. Liniger, and M. N. Zeilinger. Cautious nmpc with gaussian process dynamics for autonomous miniature race cars. *European Control Conference*, pages 1341–1348, 2018.

[5] I. Goodfellow, Y. Bengio, and A. Courtville. *Deep Learning*. MIT Press, 2016.

[6] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105, 2012.

[7] A. Punjani and P. Abbeel. Deep learning helicopter dynamics models. *IEEE International Conference on Robotics and Automation*, pages 3223–3230, 2015.

[8] N. Mohajerin and S. L. Waslander. Multistep Prediction of Dynamic Systems with Recurrent Neural Networks. *IEEE Transactions on Neural Networks and Learning Systems*, 30(11):3370–3383, 2019.

[9] S. Bansal, A. K. Akametalu, F. J. Jiang, F. Laine, and C. J. Tomlin. Learning quadrotor dynamics using neural network for flight control. *IEEE 55th Conference on Decision and Control*, pages 4653–4660, 2016.

[10] D. L. Yu and J. B. Gomm. Implementation of neural network predictive control to a multivariable chemical reactor. *Control Engineering Practice*, 11(11):1315–1323, 2003.

[11] A. Wurzinger, H. Leibinger, S. Jakubek, and M. Kozek. Data driven modeling and nonlinear model predictive control design for a rotary cement kiln. *IFAC-PapersOnLine (Proc. 11th Symposium on Nonlinear Control Systems)*, 52(16):759–764, 2019.

[12] N. Lanzetti, Y. Z. Lian, A. Cortinovis, L. Dominguez, M. Mercangoz, and C. Jones. Recurrent neural network based MPC for process industries. *European Control Conference*, pages 1005–1010, 2019.

[13] P. Kittisupakorn, P. Thitiyasook, M. A. Hussain, and W. Daosud. Neural network based model predictive control for a steel pickling process. *Journal of Process Control*, 19(4):579–590, 2009.

[14] A. Aswani, H. Gonzalez, S. S. Sastry, and C. Tomlin. Provably safe and robust learning-based model predictive control. *Automatica*, 49(5):1216–1226, 2013.

[15] M. Maiworm, D. Limon, J. M. Manzano, and R. Findeisen. Stability of Gaussian Process Learning Based Output Feedback Model Predictive Control. *IFAC-PapersOnLine (Proc. 6th IFAC Conference on Nonlinear Model Predictive Control)*, 51(20):455–451, 2018.

[16] J. M. Manzano, D. Limon, D. M. De la Peña, and J. P. Calliess. Output feedback MPC based on smoothed projected kinky inference. *IET Control Theory and Applications*, 13(6):795–805, 2019.

[17] H. G. Han, X. L. Wu, and J. F. Qiao. Real-time model predictive control using a self-organizing neural network. *IEEE Transactions on Neural Networks and Learning Systems*, 24(9):1425–1436, 2013.

[18] Z. Wu, A. Tran, D. Rincon, and P. D. Christofides. Machine learning-based predictive control of nonlinear processes. Part I: Theory. *AIChE Journal*, 65(11), 2019.

[19] K. S. Narendra and K. Parthasarathy. Identification and control of dynamical systems using neural networks. *IEEE Transactions on Neural Networks*, 1(1):4–27, 1990.

[20] G. Cybenko. Approximation by superposition of a sigmoidal function. *Mathmatics of control, signals and systems*, 2(4):303–314, 1989.

[21] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989.

[22] J. M. P. Menezes and G. A. Barreto. Long-term time series prediction with the NARX network: An empirical evaluation. *Neurocomputing*, 71(16-18):3335–3343, 2008.

[23] D. Limon, T. Alamo, F. Salas, and Eduardo F. Camacho. On the stability of constrained MPC without terminal constraint. *IEEE Transactions on Automatic Control*, 51(5):832–836, 2006.

[24] G. Shi, X. Shi, M. O. Connell, R. Yu, K. Azizzadenesheli, A. Anandkumar, Y. Yue, and S. Chung. Neural Lander : Stable Drone Landing Control Using Learned Dynamics. *IEEE International Conference on Robotics and Automation*, pages 9784–9790, 2019.

[25] L. Wang, T. Chai, and L. Zhai. Neural-network-based terminal sliding-mode control of robotic manipulators including actuator dynamics. *IEEE Transactions on Industrial Electronics*, 56(9):3296–3304, 2009.

[26] T. Yang, N. Sun, H. Chen, and Y. Fang. Neural Network-Based Adaptive Antiswing Control of an Underactuated Ship-Mounted Crane with Roll Motions and Input Dead Zones. *IEEE Transactions on Neural Networks and Learning Systems*, 31(3):901–914, 2020.

[27] D. Limon, T. Alamo, D. M. Raimondo, D. M. De La Peña, J. M. Bravo, A. Ferramosca, and E. F. Camacho. *Input-to-state stability: a unifying framework for robust model predictive control*, volume 384. Springer, 2009.

[28] D. E. Seborg, T. F. Edgar, and D. A. Mellichamp. *Process dynamics and control*. Wiley, New York, 1989.

[29] L. Ljung. *System Identification (2nd Ed.): Theory for the User*. Prentice Hall PTR, USA, 1999.

[30] J. B. Rawlings and D. Q. Mayne. *Model predictive control: Theory and design*. Nob Hill Pub., 2009.