Computers and Chemical Engineering 150 (2021) 107291

Contents lists available at ScienceDirect

Computers and Chemical Engineering

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

Industrial, large-scale model predictive control with structured neural networks

Pratyush Kumar[a,*], James B. Rawlings[a], Stephen J. Wright[b]

[a] Department of Chemical Engineering, University of California, Santa Barbara, CA 93106, United States
[b] Computer Sciences Department, University of Wisconsin-Madison, Madison, WI 53706, United States

ARTICLE INFO

Article history:
Received 25 November 2020
Revised 15 March 2021
Accepted 16 March 2021
Available online 30 March 2021

Keywords:
Model predictive control
Machine learning
Deep neural networks
Large-scale systems
quadratic programming
Inherent robustness

ABSTRACT

The design of neural networks (NNs) is presented for treating large, linear model predictive control (MPC) applications that are out of reach with available quadratic programming (QP) solvers. First, we introduce a new feedforward network architecture that enables practitioners to obtain offset-free closed-loop performance with NNs. Second, we discuss the data generation procedure to sample the state space relevant to training the NNs based on anticipated online setpoint changes and plant disturbances. Third, we use the input-to-state stability results available in the MPC literature and establish robustness properties of NN controllers. Finally, we present illustrative simulation studies on process control examples. We apply the NN design approach and compare the performance with online QP based MPC on an industrial crude distillation unit model with 252 states, 32 control inputs, and a control-sample horizon length of 140. Parallel computing is used for data generation and graphical processing units are used for network training. Anticipated plant operational scenarios with setpoints and disturbances that may change during operation must be sampled for NN training. After the offline design phase, NNs execute MPC three to five orders of magnitude faster than an available QP solver with less than 1% loss in the closed-loop performance.

1. Introduction

Model predictive control (MPC) is an online optimization based feedback control technology. A dynamic model of the plant is used to make forecasts of the plant measurements in response to the actuator movements, and an optimization problem is solved in real time to determine the optimal actuator move to be applied to the plant. For linear plant models, the optimization problem is a quadratic program (QP). The development of efficient QP algorithms (Kouzoupis et al., 2018; Wright, 2019) allowed practitioners to apply MPC in the process industries (Qin and Badgwell, 2003; Lahiri, 2017).

An alternative strategy to real time optimization for the deployment of MPC is to characterize the MPC feedback law as a piecewise affine function defined on polyhedral partitions of the state space, and use the feedback law online (Bemporad et al., 2002; Seron et al., 2003). The real time computation in this approach is reduced to a table look-up in the state space based on the current value of parameters in the QP such as the state estimate and some steady-state targets that are computed for offset-free control. The number of polyhedral partitions of the state space grows exponentially with model dimensions, however, which prohibits the deployment of this strategy on large-scale systems.

The MPC feedback law can be approximated using parametric functions such as polynomials (Kvasnica et al., 2011), various types of piecewise affine functions (Bemporad et al., 2011; Wen et al., 2009), and neural networks (NNs) (Cavagnari et al., 1999). Among the function approximators, NNs that use the rectified linear unit (ReLU) as the activation function have gained attention recently (Chen et al., 2018; Karg and Lucia, 2020; Lovelett et al., 2020; Paulson and Mesbah, 2020) to approximate the MPC feedback law due to their ability to represent complex piecewise affine functions (Montufar et al., 2014) and execute MPC faster in real time than QP solvers. The proposed NN design approach in these works is to generate training data for the NN by solving QPs offline for a set of feasible states of the QP, use the collected data to train a standard feedforward network, then use the trained NN as the feedback controller online.

To make progress in the size of control applications achievable with MPC, large problems in which QP solvers fail to deliver the

* Corresponding author.
E-mail addresses: pratyushkumar@ucsb.edu (P. Kumar), jbraw@ucsb.edu (J.B. Rawlings), swright@cs.wisc.edu (S.J. Wright).

https://doi.org/10.1016/j.compchemeng.2021.107291
0098-1354/

control in the available sample time should be addressed. The scalability of the NN approach to such problems with large model dimensions and control horizon length in the QP has not been demonstrated to date. Two issues arise in these problems: (i) In a large dimensional state space, the entire set of feasible states of the QP cannot be sampled densely for the NN training. This issue is a form of the well-known curse of dimensionality. (ii) One may sample the state space partially to avoid this problem, but the time required for the data generation can still be impractical due to the long time required to solve a QP for each sampled state. We demonstrate in this article by case studies that the issue (ii) can be resolved for many size of problems with parallel computing, and the issue (i) by sampling the state space for typical plant operational scenarios. The following feature must be present in the large-scale application of interest for the partial sampling scheme proposed in this article to be applicable: For a particular operating mode of the plant, the number of frequently changing setpoints and large magnitude disturbances must be small enough such that only a small fraction of the entire large dimensional state space is visited during that operation mode. The MPC feedback law can then be reliably approximated using NNs for the different modes of plant operations, each driven by their respective set of setpoint changes and disturbances.

From the above viewpoint, this article discusses the design of NNs as an alternative to real time optimization in MPC, with an emphasis on large applications that are challenging for QP solvers. We focus on setpoint tracking MPC problems relevant to the process industries. We present a new feedforward network architecture to obtain offset-free closed-loop performance with the NNs. Additionally, we use the input-to-state stability results available in the MPC literature, and establish conditions under which NN controllers are robust to state estimation errors and process disturbances. We present case studies on process control examples that demonstrate the scalability of the proposed NN design approach.

The related work of Chen et al. (2019) demonstrates the scalability of NNs to approximate the MPC feedback law on systems with state dimensions up to 36, on the control problem of regulation to the origin. By contrast, our paper considers setpoint tracking offset-free MPC problem and state dimensions up to 250. Drgoňa et al. (2018) take a different approach for the scalability of NN controllers, in which the dimensionality of the parameters in the MPC QP is first reduced using principal component analysis, and the NN controller is developed in a low dimensional state space.

The rest of this article is organized as follows. In the next section, we briefly review offset-free linear MPC. In Section 3, we discuss the NN controller design procedure. We present the robustness property of NNs in Section 4. Section 5 presents simulation studies on two large process control examples and some concluding thoughts appear in Section 6.

Notation. The symbols $\mathbb{I}$ and $\mathbb{R}$ are used to denote integers and reals respectively. Subscripts denote restrictions (e.g., $\mathbb{R}_{\geq 0}$ for nonnegative reals and $\mathbb{I}_{a:b}$ for integers in the closed interval $[a, b]$). The Euclidean norm is denoted by $|\cdot|$. A function $\alpha : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ is of class $\mathcal{K}$ if it is continuous, zero at zero, and strictly increasing. This function is of class $\mathcal{K}_\infty$ if it is of class $\mathcal{K}$ and unbounded ($\alpha(s) \to \infty$ as $s \to \infty$). A function $\beta : \mathbb{R}_{\geq 0} \times \mathbb{I}_{\geq 0} \to \mathbb{R}_{\geq 0}$ is of class $\mathcal{KL}$ if it is continuous, for each $k \in \mathbb{I}_{\geq 0}$, $\beta(\cdot, k)$ is of class $\mathcal{K}$, and for each $s \geq 0$, $\beta(s, \cdot)$ is nonincreasing and satisfies $\lim_{k\to\infty} \beta(s, k) = 0$. Given $V : X \to \mathbb{R}_{\geq 0}$ and $\tau > 0$, define $\text{lev}_\tau V = \{x \in X \mid V(x) \leq \tau\}$. Bold symbols, e.g., $\mathbf{d}$ denote sequences, $d(k)$ denotes an element of $\mathbf{d}$ at time $k \in \mathbb{I}_{\geq 0}$, and $\mathbf{d}_i$ denotes the collection of elements of $\mathbf{d}$ for $k \in \mathbb{I}_{0:i-1}$. Define $||\mathbf{d}_i|| = \max_{k\in\mathbb{I}_{0:i-1}} |d(k)|$. For a given vector a, lower and upper bounds $(\underline{a}, \overline{a})$, define the saturation function as $\text{sat}(a, \underline{a}, \overline{a}) = \{a \text{ if } \underline{a} \leq a \leq \overline{a}; \underline{a} \text{ if } a < \underline{a}; \overline{a} \text{ if } \overline{a} < a\}$.

2. Linear model predictive control

2.1. Model

We consider a linear discrete time model augmented with an integrating disturbance model (Pannocchia and Rawlings, 2003):

$$x^+ = Ax + Bu + B_d d, \quad d^+ = d, \quad y = Cx + C_d d, \tag{1}$$

in which $x \in \mathbb{R}^n$ is the state, $u \in \mathbb{R}^m$ is the control input, and $d \in \mathbb{R}^d$ is the state of the disturbance model. The matrices $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times m}$, $C \in \mathbb{R}^{p\times n}$ denote the actuator to measurement model; and $B_d \in \mathbb{R}^{n\times d}$, $C_d \in \mathbb{R}^{p\times d}$ is the disturbance model. The objective of the disturbance model is to remove offset in the controlled plant measurements at steady state operations and maintain the controlled variables at their setpoints. We assume in the rest of this article that a Kalman filter can be constructed to estimate the model state $(\hat{x})$ and disturbance $(\hat{d})$ from measurements.

2.2. Target selector

Given the disturbance estimate, the input and controlled measurement setpoints, we consider the following target selector QP:

$$\min_{x_s,\ u_s} \frac{1}{2}|u_{sp} - u_s|^2_{R_s} \tag{2}$$

subject to

$$\begin{bmatrix} I - A & -B \\ HC & 0 \end{bmatrix} \begin{bmatrix} x_s \\ u_s \end{bmatrix} = \begin{bmatrix} B_d\hat{d} \\ r_{sp} - HC_d\hat{d} \end{bmatrix}, \tag{3}$$

$$\underline{u} \leq u_s \leq \overline{u}, \tag{4}$$

in which (x_s, u_s) is the target steady-state pair, $r = Hy$ are the controlled measurements chosen as some subset or a linear combination of all the measurements, $(\underline{u}, \overline{u})$ are the actuator bounds, and (u_{sp}, r_{sp}) are the input and controlled measurement setpoints. The equality constraint $r_{sp} = H(Cx_s + C_d\hat{d})$ can be difficult to satisfy exactly in real time depending on the controlled measurement setpoint and disturbance estimate. In this case, the hard setpoint equality constraint can be relaxed and moved to the stage cost, such that the target selector computes a steady state to minimize the offset in the controlled measurements. We assume for the simulation studies in this article that the input setpoint (u_{sp}) is fixed at some chosen steady state and only the controlled measurement setpoint (r_{sp}) changes in real time.

2.3. Regulator

Based on the state estimate and target steady-state pair, the following QP is solved to determine the actuator move to be applied to the plant:

$$V_N(\tilde{x}(0), \tilde{\mathbf{u}}) = \frac{1}{2}\sum_{k=0}^{N-1}\left(|\tilde{x}(k)|^2_Q + |\tilde{u}(k)|^2_R\right) + \frac{1}{2}|\tilde{x}(N)|^2_P \tag{5}$$

subject to

$$\tilde{x}^+ = A\tilde{x} + B\tilde{u}, \quad \tilde{x}(0) = \hat{x} - x_s, \tag{6}$$

$$\underline{u} \leq \tilde{u} + u_s \leq \overline{u}, \tag{7}$$

in which $\hat{x}$ is the state estimate after the current measurement, N is the control horizon length, and $(\tilde{x}, \tilde{u})$ are the state and control in deviation from the current steady-state targets (x_s, u_s). The penalty matrix P is chosen as the optimal cost-to-go matrix of the unconstrained infinite horizon linear quadratic problem. The decision variable in this QP is the control sequence $\tilde{\mathbf{u}} =$

$[\tilde{u}(0)', \tilde{u}(1)', \ldots, \tilde{u}(N-1)']'$, and the state sequence $\tilde{\mathbf{x}}$ is fully determined by the model equality constraints (6). The first move of the optimal solution is denoted as the MPC feedback law, $\kappa_N(\hat{x}, x_s, u_s) = \tilde{\mathbf{u}}^0(0; \hat{x}, x_s, u_s) + u_s$, which is applied to the plant. This feedback law is a function of the state estimate and target steady-state pair. A rate-of-change penalty on the control can be imposed in the QP objective by augmenting the dynamic model with a surrogate state for the control as discussed in Rao and Rawlings (1999). In this case, the control injected at the previous timestep (u_{-1}) is supplied to the QP, and the feedback law becomes a function of the previous control, in addition to the state estimate and target steady-state pair.

Several algorithms have been proposed in the literature over the years to solve the above MPC regulator QP. Kouzoupis et al. (2018) and Wright (2019) give recent reviews on the development of convex optimization algorithms to solve this linear MPC QP. For the simulation studies in this article, we eliminate the state trajectory from the set of decision variables and formulate a dense QP. The decision variable in the dense QP is the future control trajectory $\tilde{\mathbf{u}}$, and the QP solver CVXOPT (Vandenberghe, 2010), which is tailored for dense problems is used to generate data for NN training and for timing comparisons with online QP based MPC. Any improvement in the QP formulation and solution algorithm is advantageous for both the online QP based MPC and NN approaches, as fast QP solvers reduce the offline data generation time required for the design of NNs.

3. Neural network design

In the offset-free linear MPC algorithm discussed in the previous section, the target selector QP is usually small compared to the regulator QP, which consumes most of the online computation time. Therefore we focus on designing a NN that approximates the MPC feedback law ($\kappa_N(x, x_s, u_s)$) for an operationally relevant set of states and steady-state targets, such that the NN can be used online as the feedback controller for the plant in place of solving the MPC regulator QP.

3.1. Structured neural network

An intuitive approach is to build a feedforward NN that takes the triple (x, x_s, u_s) as its input and outputs a control that is close to the optimal control. This strategy has been proposed in Karg and Lucia, 2020 and Chen et al. (2018) for a single fixed steady state. The approach has not been demonstrated to scale to the size of model dimensions and the setpoint tracking offset-free MPC problem considered in this article. An issue with this strategy is that the NN has no knowledge about the MPC feedback law, and can require large amounts of training data in large state dimensions to obtain a reasonable controller. At a minimum, the MPC feedback law is such that at the origin, the control action taken by the MPC controller is zero. For the setpoint tracking MPC problem considered in this article, the NN control law must be such that at steady-state operations, the NN maintains the plant at the desired steady state such that the controlled measurements are at their setpoints. This structure implies $u_s = \kappa_{NN}(x = x_s, x_s, u_s)$, in which we use $\kappa_{NN}(\cdot)$ to denote the NN control law. To incorporate this information in a NN, we introduce the following architecture:

$$
\begin{aligned}
z_0 &= \begin{bmatrix} x', & x_s', & u_s', & x_s', & x_s', & u_s' \end{bmatrix}', \\
f_i(z_{i-1}) &= \begin{bmatrix} W_i & 0 \\ 0 & W_i \end{bmatrix} z_{i-1} + \begin{bmatrix} b_i \\ b_i \end{bmatrix}, \\
z_i &= g(f_i(z_{i-1})), \text{ for } i \in \mathbb{I}_{1:h}, \\
u &= u_s + \begin{bmatrix} W_{h+1}, & -W_{h+1} \end{bmatrix} z_h,
\end{aligned} \tag{8}
$$

in which $g(a) = \max(0, a)$, is the ReLU operation applied elementwise on its vector input, z_i is the output of each hidden layer of the network, z_0 is the input to the network, h is the number of hidden layers, and $i \in \mathbb{I}_{1:h}$. The parameters to be optimized using training data are (W_i, b_i) and W_{h+1}, which are matrices of appropriate dimensions. At steady state, when $x = x_s$ in z_0, (8) outputs u_s regardless of the choice of weights and the activation function, as the output of all hidden layers (z_i) have same subvector repeated in the upper and lower half and the term $\begin{bmatrix} W_{h+1}, & -W_{h+1} \end{bmatrix} z_h$ equals zero. The NN structure (8) is equivalent to the difference in the outputs of a standard feedforward NN with inputs (x, x_s, u_s) and (x_s, x_s, u_s) respectively, with u_s added to the final output. The software tensorflow (Abadi et al., 2015) is used for NN training; the structured architecture can be built with symbolic differentiation tools available in tensorflow.

When a rate-of-change penalty on the control is included in the regulator QP, the input to the structured NN is modified to:

$$
z_0 = \begin{bmatrix} x', & u_{-1}', & x_s', & u_s', & x_s', & u_s', & x_s', & u_s' \end{bmatrix}',
$$

in which u_{-1} is the previous control applied to the plant and the dimensions of the matrices (W_i, b_i) and W_{h+1} are appropriately adjusted. Even after training, the control produced by the NN may not satisfy the hard actuator constraints, so the saturation function $(\text{sat}(\cdot, \underline{u}, \overline{u}))$ is applied to the output of the NN.

3.2. Data generation

For one fixed steady state, the set of feasible states for the MPC QP defined by (5)–(7) is the entire n-dimensional state space $\mathbb{R}^n$. Additionally, the target steady state may change during plant operation, depending on the setpoint and disturbance estimate. The domain of the MPC control law ($\kappa_N(x, x_s, u_s)$) is therefore the state space $\mathbb{R}^n$ and the possible set of steady-state target pairs. Sampling this entire domain of the MPC control law for NN training is difficult in large dimensions; we show in Section 5 that it is not required for practical applications.

A majority of large-scale petrochemical plants operate in a relatively small number of operating regimes or scenarios. Each operational scenario is driven by a selected few controlled measurement setpoints that depend on product demands and some large magnitude disturbances that may change frequently, while the setpoints for other measurements remain constant for long periods of time. The states and steady-state targets spanned in a closed-loop operation per scenario is thereby considerably less than the entire domain of the MPC control law. These characteristics of typical operation of large chemical plants can be exploited, and we propose the following data generation procedure to perform offline simulations using the model to collect the operationally relevant set of $(x, x_s, u_s, \kappa_N(\cdot))$ for NN training.

1. Determine the anticipated range of controlled measurement setpoint changes (r_{sp}) to be made during the plant operational scenario. We denote this range of setpoints as $(\underline{r}_{sp}, \overline{r}_{sp})$.
2. Identify the set of physical disturbances (d) that may affect the plant and their range of values $(\underline{d}, \overline{d})$. We use physical disturbances in this article and assume that a disturbance model identification is performed to obtain the matrices B_d and C_d.
3. Create pseudo random binary signals (PRBS) of r_{sp} and d in their respective range of values.
4. Initialize the model at some chosen steady state. Perform an offline simulation with the generated PRBS signals by solving the target selector and regulator QPs for the transient states and steady-state targets encountered in the simulation.

During plant operation, it is expected that the NN will encounter states and steady-state target pairs not used in training.

But similar values are expected, since an adequate model and appropriate range of anticipated setpoints and plant disturbances are used for the data generation. During the closed-loop operation, the NN produces an interpolated actuator move that is applied to the plant.

As we consider large MPC problem sizes in this article for the NN controller, the data generation step can take impractical amounts of time if performed serially, due to the long time (minutes) required to solve each QP. For these large problems, several PRBS signals of the setpoints and disturbances are generated and multiple simulations are performed in parallel over several cores in a CPU and across multiple CPUs as well. The data generated from all the simulations are collected and used for the NN training.

3.3. Training

The following mean squared error (MSE) is minimized to determine the weights for the NN controller:

$$J(\theta) = \frac{1}{2}\sum_{j=1}^{N_{tr}}[\kappa_{NN}(x_j, x_{sj}, u_{sj}; \theta) - \kappa_N(x_j, x_{sj}, u_{sj})]^2 \tag{9}$$

in which θ is the set of parameters, (W_i, b_i, W_{h+1}) for $i \in \mathbb{I}_{1:h}$, subscript j is used to denote the training sample index, and N_{tr} is the number of training samples. The stochastic optimization algorithm Adam (Kingma and Ba, 2014) is used for all the case studies in this article. We do not consider any regularization penalty in the training objective. The recent empirical and theoretical works in the machine learning literature have shown that NNs with large number of parameters have good interpolation capabilities (Belkin et al., 2019; Zhang et al., 2017; Arora et al., 2019; Allen-Zhu et al., 2019) without any form of explicit regularization. The generalization abilities of NNs for the architectures considered in this article is examined in the simulations presented in the Section 5.

4. Robustness of neural networks

We now present the robustness properties of NN controllers. The optimal MPC controller designed for a nominal linear system ignoring disturbances is known to be inherently robust to bounded state estimation errors and disturbances (Heath and Wills, 2005; Pannocchia et al., 2011). The approximation error of the MPC feedback law by a NN can be viewed as an additional disturbance, and existing input-to-state stability results (Sontag and Wang, 1995) can be used to establish the allowable approximation error in the NN for the NN controller to be robust to disturbances. Theorem 1 below states this allowable approximation error and specifies the size of disturbances for which the NN controller is robust. This result also holds for other approximate MPC controllers, which use different parametric functions to approximate the optimal control law, such as polynomials and other piecewise affine functions.

The related work by Hertneck et al. (2018) examines the robustness of nonlinear systems in feedback with approximate MPC controllers. The MPC control law approximation error is treated as an input disturbance, and a robust MPC formulation based on constraint tightening procedures is used to establish the allowable approximation error. The robustness of NN controllers trained with samples of nominal MPC control laws in presence of process disturbances and state estimation errors has not been established to date in the literature.

The problem of steering the state of the linear system (1) to one fixed steady state (x_s, u_s) is considered. For the analysis in this section, the state and control are transformed in deviation variables as $x := x - x_s$ and $u := u - u_s$, and we define the control problem as regulation to the origin. The optimal MPC control law is denoted as $\kappa_N(x)$, and the NN control law as $\kappa_{NN}(x) = \kappa_N(x) + e_{NN}(x)$, in which $e_{NN}(x)$ is the approximation error of the optimal control law by the network. Assume in the analysis that the target steady state is such that the unconstrained linear quadratic regulator (LQR) satisfies the actuator constraints in a small neighborhood near the origin.

As reviewed in Mayne et al. (2000), the nominal stability of MPC can be established using the optimal cost function $(V_N^0(x))$ of the QP as a Lyapunov function. The region of attraction of the closed-loop system $x^+ = Ax + B\kappa_N(x)$ when no hard terminal region constraint is included in the QP can be characterized as $\mathcal{X}_N = \text{lev}_{Nd+\tau}\, V_N^0(x)$ (Limon et al., 2006). The parameter τ is the level set parameter of the terminal region chosen as $\mathbb{X}_f = \text{lev}_\tau\, x'Px$, $d > 0$ is a constant such that $\ell(x, u) \geq d$ for all $x \in \mathbb{R}^n \setminus \mathbb{X}_f$, and $\underline{u} \leq u + u_s \leq \overline{u}$. The parameter τ is chosen small enough such that the unconstrained LQR satisfies the actuator constraints for all $x \in \mathbb{X}_f$. The optimal cost function is continuous and satisfies $c_1|x|^2 \leq V_N^0(x) \leq c_2|x|^2$ and $V_N^0(x^+) - V_N^0(x) \leq -c_1|x|^2$ for some $c_2 \geq c_1 > 0$.

Assume that a state estimate $(\hat{x})$ is used by the NN to compute the control. To analyze robustness, the following perturbed linear system is considered:

$$\hat{x}^+ = A\hat{x} + B\kappa_N(\hat{x}) + Be_{NN}(\hat{x}) + w - Ae + e^+, \tag{10}$$

in which $\hat{x} = x + e$ is the state estimate, w is a process disturbance, and e, e^+ are the state estimation errors at the current and next timestep, respectively. We use $\phi_d(k; \hat{x})$ to denote a solution of the perturbed system at time k starting from an initial state $\hat{x}$; and $\phi(k; x)$ to denote a solution of the nominal system starting from an initial state x when the state estimation errors and process disturbances are zero.

Theorem 1. *For all $0 < \rho \leq Nd + \tau$, there exist constants $\delta_1, \delta_2, \delta_3 > 0$, functions $\beta(\cdot) \in \mathcal{KL}$ and $\alpha_e(\cdot), \alpha_w(\cdot), \alpha_n(\cdot) \in \mathcal{K}$, such that for all disturbance sequences satisfying $||\mathbf{e}_{k+1}|| \leq \delta_1$, $||\mathbf{w}_k|| \leq \delta_2$, and $\left|e_{NN}(\hat{x})\right| \leq \delta_3$, and for all $\hat{x}$ in the set $S := \text{lev}_\rho\, V_N^0(\hat{x})$, we have the bound $\left|\phi_d(k; \hat{x})\right| \leq \beta(\left|\hat{x}\right|, k) + \alpha_e(||\mathbf{e}_{k+1}||) + \alpha_w(||\mathbf{w}_k||) + \alpha_n(\bar{e}_{NN})$, in which $\bar{e}_{NN} = \max_{\hat{x}\in S}\left|e_{NN}(\hat{x})\right|$.*

The proof of this theorem and the required input-to-state stability definitions are provided in Appendix A. In the nominal case, the disturbance sequences $\mathbf{e}$ and $\mathbf{w}$ are zero, and the closed-loop states have the bound $|\phi(k; x)| \leq \beta(|x|, k) + \alpha_n(\bar{e}_{NN})$. An additional value of the NN training is to reduce the worst approximation error $(\bar{e}_{NN})$ in the state space of interest, thus reducing the bound on the closed-loop states.

5. Simulation studies

In this section, two case studies are presented that demonstrate the scalability of the proposed NN design approach. The closed-loop performance of NNs and computational benefits over online QP based MPC are analyzed in these case studies. After the offline design of NNs, two types of validation simulations are performed: (i) with setpoints and disturbances not present in the training data but within the same range of values used to generate the training data, (ii) with extra setpoint signals assumed to be unknown during the offline network design phase. Both these types of simulations shed light on the interpolation and extrapolation abilities of the NNs, and are referred subsequently as simulations with *expected* and *unexpected* plant behaviors, respectively.

The validation simulations are performed directly with the plant model for our case studies. For industrial deployment, the NN training should be followed by an offline validation step of the trained network to verify the quality of the NN controller prior to online deployment. This validation can be performed by: a) quantifying the MSE of the trained NN controller on test data generated

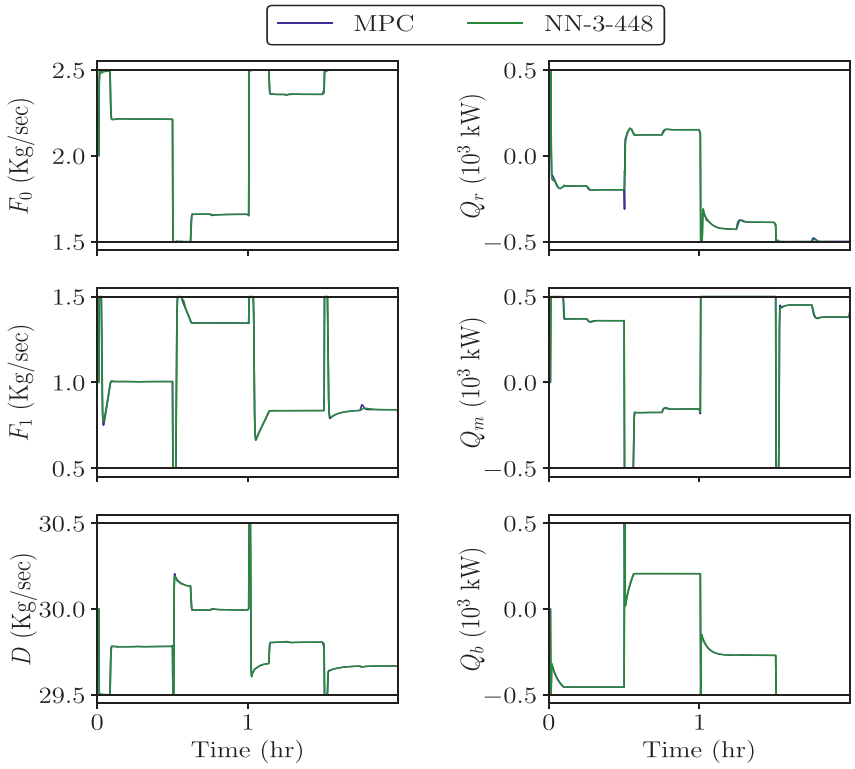

Fig. 1. Transient closed-loop actuator trajectories of the CSTRs with flash separator example with the NN and optimal MPC controllers. The performance of the NN and optimal MPC controllers are almost indistinguishable.

using the optimal MPC controller, b) examining closed-loop performance metrics in simulations carried out with the linear model used in MPC and the optimal MPC and NN controllers, similar to the simulations performed in this section with the plant model. Additional probabilistic validation techniques for NNs trained using robust MPC solutions for nonlinear models are proposed in Karg et al. (2019) and Hertneck et al. (2018).

The closed-loop performance of the following alternative and equivalently fast controllers as NNs are also compared in the validation simulations: (a) steady state controller (SS): the solution of the target selector is directly applied to the plant $u = u_s$; (b) saturated linear quadratic regulator (satK): the unconstrained LQR gain K is computed and used in the control law $u = \text{sat}(K(x - x_s) + u_s, \underline{u}, \overline{u})$; (c) short horizon controller (SH): an MPC problem with a short control horizon is solved online; (d) Unstructured NN (NN-UNS): a standard feedforward network is trained with the triple (x, x_s, u_s) as the input and then used as the feedback controller online.

To gauge the performance of NNs in the validation simulations, the offline computational effort required to build the NNs, and the memory required for the deployment of NNs, the following metrics are examined:

- Controller performance index

$$\Lambda_k = \frac{1}{k}\sum_{t=1}^{k}\left(|x(t) - x_s(t)|_Q^2 + |u(t) - u_s(t)|_R^2 + |\Delta u(t)|_S^2\right);$$

- % Performance loss $= 100(\Lambda_{N_t}^{\text{F}} - \Lambda_{N_t}^{\text{MPC}})/\Lambda_{N_t}^{\text{MPC}}$;
- Average and worst-case speedups;
- Data generation and NN training times;
- Memory required to store the weights of the NNs.

Here, N_t is the number of simulation time steps, and $\Lambda_{N_t}^{\text{F}}$ and $\Lambda_{N_t}^{\text{MPC}}$ are the average stage costs obtained at the end of the simulation period by the fast controllers and the MPC controller. The data generation and online timing comparisons are performed on a computing cluster that has several multi-core CPUs of clock speed 2.4 GHz. The NN trainings are performed on a Tesla V100-SXM2 GPU that has 32 GB memory. Code for the simulation studies is available at: https://github.com/pratyushkumar211/industrial_nnmpc_2021.

5.1. CSTRs with a flash separator

For the first example, the plant consists of two stirred tank reactors in series with a flash separator, depicted in Fig. 3. In both the CSTRs, a desired product B is produced with the reaction $A \to B$, and an undesired product C is produced with the reaction $B \to C$. The reactant A is the major component in the feed streams supplied to the reactors. After the second CSTR, the reaction mixture is sent to the non-adiabatic flash that separates the reactant A from the product B. The A rich vapor phase is recycled back to the first CSTR. The plant is simulated using the set of nonlinear ordinary differential equations (ODEs) available in Venkat (2006), Appendix. The model has 12 states $(H_r, x_{Ar}, x_{Br}, T_r, H_m, x_{Am}, x_{Bm}, T_m, H_b, x_{Ab}, x_{Bb}, T_b)$, 6 control inputs $(F_0, F_1, D, Q_r, Q_m, Q_b)$, and 5 disturbances $(x_{A0}, x_{A1}, x_{B0}, x_{B1}, T_0)$. The controlled measurements with setpoints are the heights and temperatures of the three units, and we assume that all the states are measured. The sample time for the measurements is chosen as 10 seconds. The parameter values used to simulate the ODEs for the plant, the actuator constraints, and the bounds of the controlled measurements and disturbances used for the data generation and validation simulations are shown in Table 3, Appendix B.

A linear discrete time model with the chosen sample time is obtained at the steady state shown in the Table 3. This model, inputs and outputs are scaled for the MPC controller such that the input constraints satisfy $\overline{u} - \underline{u} = \mathbf{2}$. The tuning parameters

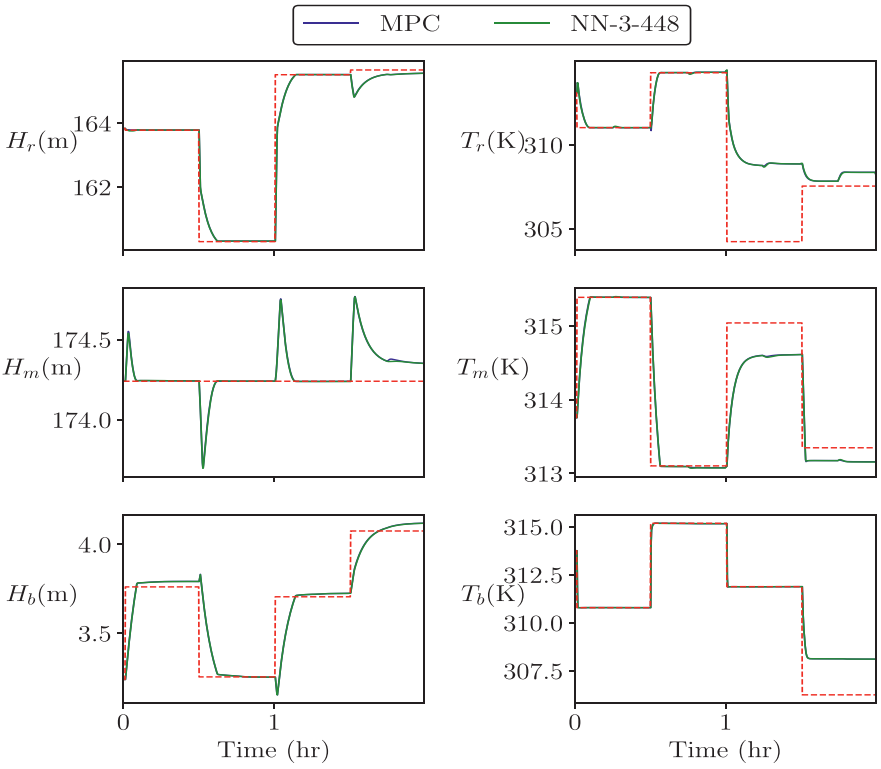

Fig. 2. Transient closed-loop controlled measurement trajectories of the CSTRs with flash separator example with the NN and optimal MPC controllers. The performance of the NN and optimal MPC controllers are almost indistinguishable.

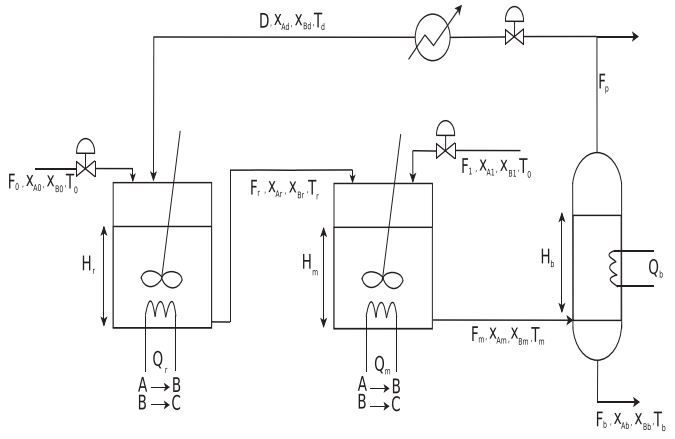

Fig. 3. Schematic of the CSTRs with a flash separator model.

and control horizon length for the regulator are chosen as $Q = 10^3 C'C$, $R = 0.1I$, $S = 0.1I$, and $N = 450$. A total 1.5×10^5 samples of $(x, u_{-1}, x_s, u_s, \kappa_N(\cdot))$ is generated for NN training, parallelized in 100 separate offline simulations. The time consumed for this data generation step was 4.2 hours. The setpoint of the height of the second CSTR (H_m) is assumed to be fixed during this data generation process, and is included as an additional setpoint in the validation simulations for the unexpected plant behavior case.

Three structured NN architectures are considered. Each NN has three hidden layers with 448 (NN-3-448), 480 (NN-3-480), and 512 (NN-3-512) nodes in each hidden layer respectively. For validation simulations, two sets of setpoint and disturbance signals with 24 setpoint changes and 48 disturbance changes are generated for 4320 timesteps (12 hours), one set each for the expected and unexpected plant behavior cases. The short horizon MPC controller is constructed with a horizon length of $N = 10$, and the architecture of the unstructured NN consists of three hidden layers with 224 nodes in each hidden layer. An additional scaling of x and x_s is performed for the NN training as $x := 2x/(x_{\max} - x_{\min})$ and $x_s := 2x_s/(x_{\max} - x_{\min})$, in which $x_{\max}$ and $x_{\min}$ are the maximum and minimum values of the state observed in the training data. This scaling is also performed in real time when the NN is used to compute the actuator move.

First, we study the amount of data required to obtain an accurate NN controller is studied. To this aim, NNs with the three chosen architectures are trained with incremental increases of 10^4 training samples, ranging from 4×10^4 to 1.5×10^5 samples. A batch size of 1024 samples is used in Adam, and all the NNs are trained for 2000 epochs. For all the NN trainings performed, 10% of the training dataset is kept as a buffer dataset, which is monitored during training, and the network weights are updated only when the MSE decreases on this buffer dataset after every training epoch. The NNs after training are used in validation simulations with the setpoint and disturbance signals generated for the expected plant behavior case. The closed-loop performance is compared with the optimal MPC controller and the % performance loss obtained is plotted in Fig. 4 (top left) as a function of the number of training samples. All the trained NNs provide less than 1% loss in this performance metric after training with about 9×10^4 samples. The best % performance loss obtained by the NNs is summarized in Table 1.

Second, we use the above trained NNs with the best performance losses in validation simulations, with the setpoint and disturbance signals generated for the unexpected plant behavior case. The % performance losses obtained in these simulations are also summarized in the Table 1, and the performance degrades from less than 1% to 5–8%. This deterioration in the performance is due to some unseen (x, u_{-1}, x_s, u_s) encountered in real-time by the NN resulting from the setpoint changes of the height of the second

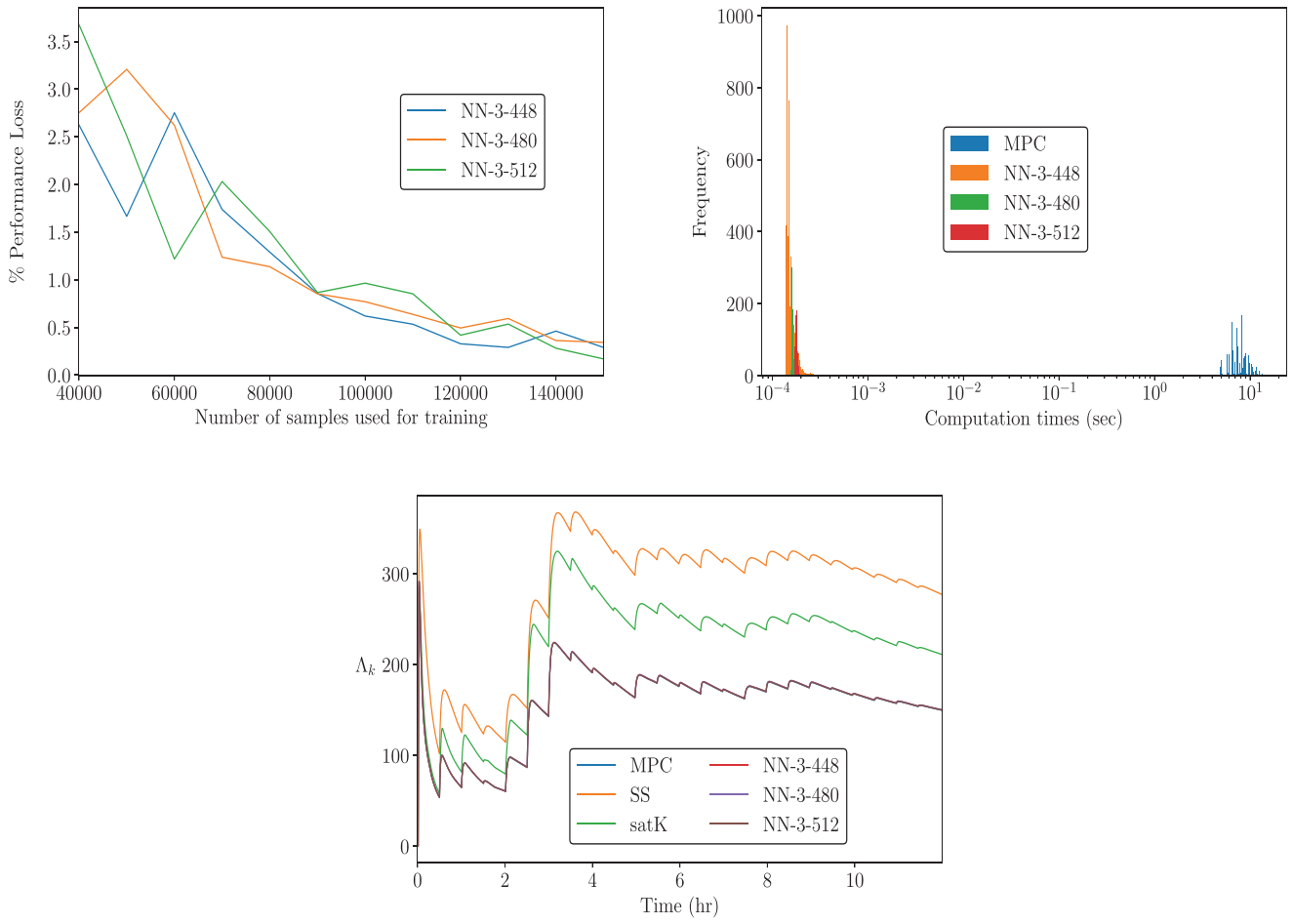

Fig. 4. Simulation statistics for the CSTRs with a flash separator example. Data requirement to obtain accurate NN controllers (top left), online computation times of the NNs and MPC controller (top right), and the controller performance index obtained with the NNs, SS, satK and MPC controllers (center bottom).The performance of the three NN and optimal MPC controllers are indistinguishable.

Table 1
Summary of the simulation study for the CSTRs with a flash separator example.

Controller	Neural network architecture	Number of parameters	Memory footprint (MB)	Training time (min)	% Performance losses (Expected, Unexpected plant behaviors)	Average speedup	Worst case speedup
SS					85.18%, 106.39 %		
satK					41.03%, 27.61 %	2.47×10^5	3.07×10^5
SH					1.61%, 2.46 %	3.75×10^3	8.4×10^1
NN-UNS	[36, 224, 224, 224, 6]	1.10×10^5			80.49%, 73.29 %	9.31×10^4	4.99×10^3
NN-3-448	[72, 448, 448, 448, 6]	1.10×10^5	0.84	13.02	0.28 %, 5.57 %	5.26×10^4	1.45×10^3
NN-3-480	[72, 480, 480, 480, 6]	1.26×10^5	0.96	12.51	0.34 %, 7.58%	5.09×10^4	1.53×10^4
NN-3-512	[72, 512, 512, 512, 6]	1.42×10^5	1.08	12.77	0.16 %, 5.90 %	4.62×10^4	1.58×10^4

CSTR, which was assumed to be fixed during the data generation. These simulations illustrate that one must not expect the NNs to extrapolate outside the training data. The practical implication of this study is that all possible plant scenarios with the setpoints and disturbances must be identified, prior to deployment of NNs, and should be used to generate the training data.

Third, the transient closed-loop performance of the trained NNs is examined for one particular validation simulation for the expected plant behavior case. Figs. 1 and 2 show the transient actuator and controlled measurements obtained with the NN-3-448 architecture, and optimal MPC controllers for a simulation period of 2 hours. Fig. 4 (center bottom) shows the transient values of the controller performance index obtained in this validation simulation with the three structured NNs, SS, satK and optimal MPC controllers for the full simulation period of 12 hours. Closed-loop performance obtained using NNs is almost the same as the optimal MPC controller and the degradation in the performance is quantified with the % performance loss metric.

Fourth, the online computational benefits of NNs over the QP based MPC is analyzed. Fig. 4 (top right) shows a histogram of online computation times of the NNs and QP solver in one validation simulation. The average and worst-case speedups obtained using NNs are summarized in the Table 1. The QP solver takes about 8 to 13 seconds in this example and all the NNs compute the control in just 0.1 to 4 milliseconds, and are easily deployable in the sample time of 10 seconds.

The % performance loss obtained using the four heuristic controllers, their corresponding speedups, maximum time required to train each NN, number of parameters in each NN, and the memory required to store the weights of the NNs is summarized in

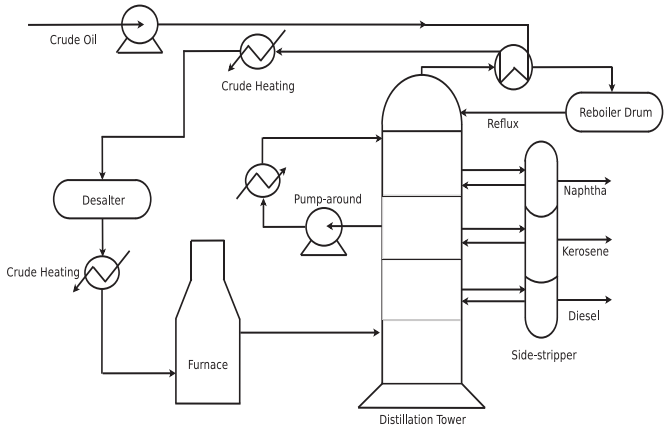

Fig. 5. Schematic of the industrial crude distillation unit model.

the Table 1. The memory requirement shows that NNs are deployable on resource constrained hardware, which can be crucial for industrial deployment at regulatory control layers. The 85% performance loss reported for the unstructured NN corresponds to training with 1.5×10^5 samples. Such performance is unacceptable for an industrial application. The noticeable performance improvement with the structured NNs is because the control law is exactly the same as the optimal feedback law at steady-states, which simplifies the optimal feedback law approximation problem since the training process only has to improve the NN control law for the transient conditions.

5.2. Large-scale crude distillation unit

The scalability of the proposed NN design approach is next demonstrated on an industrial crude distillation unit model (Pannocchia et al., 2007) with 252 states, 32 controls, and 90 measurements. The schematic of this model is shown in Fig. 5, which is a typical crude unit found in petrochemical refineries. The plant model is linear, and input and outputs are scaled for the MPC regulator as in the previous example. Only four measurements that represent the quality of the crude side products have setpoints, and five disturbances are used on the crude composition, fuel gas quality, and steam header pressure. The sample time for the measurements is 1 min. The tuning parameters and control horizon length for the MPC regulator are chosen as $Q = 2C'C$, $R = 0.1I$, and $N = 140$. A total 3.6×10^5 samples of $(x, x_s, u_s, \kappa_N(\cdot))$ are generated for the NN training, parallelized over 149 offline simulations. The time consumed for this data generation step was 27.8 hours.

Three structured NN architectures are considered for this example. Each NN has three hidden layers, with 1664 (NN-3-1664), 1792 (NN-3-1792), and 1920 (NN-3-1920) nodes in each hidden layer respectively. For the validation simulations, one set of setpoint and disturbance signals for the expected plant behavior case is generated with 24 setpoint changes and 48 disturbance changes for 2880 timesteps (2 days). The short horizon MPC controller is

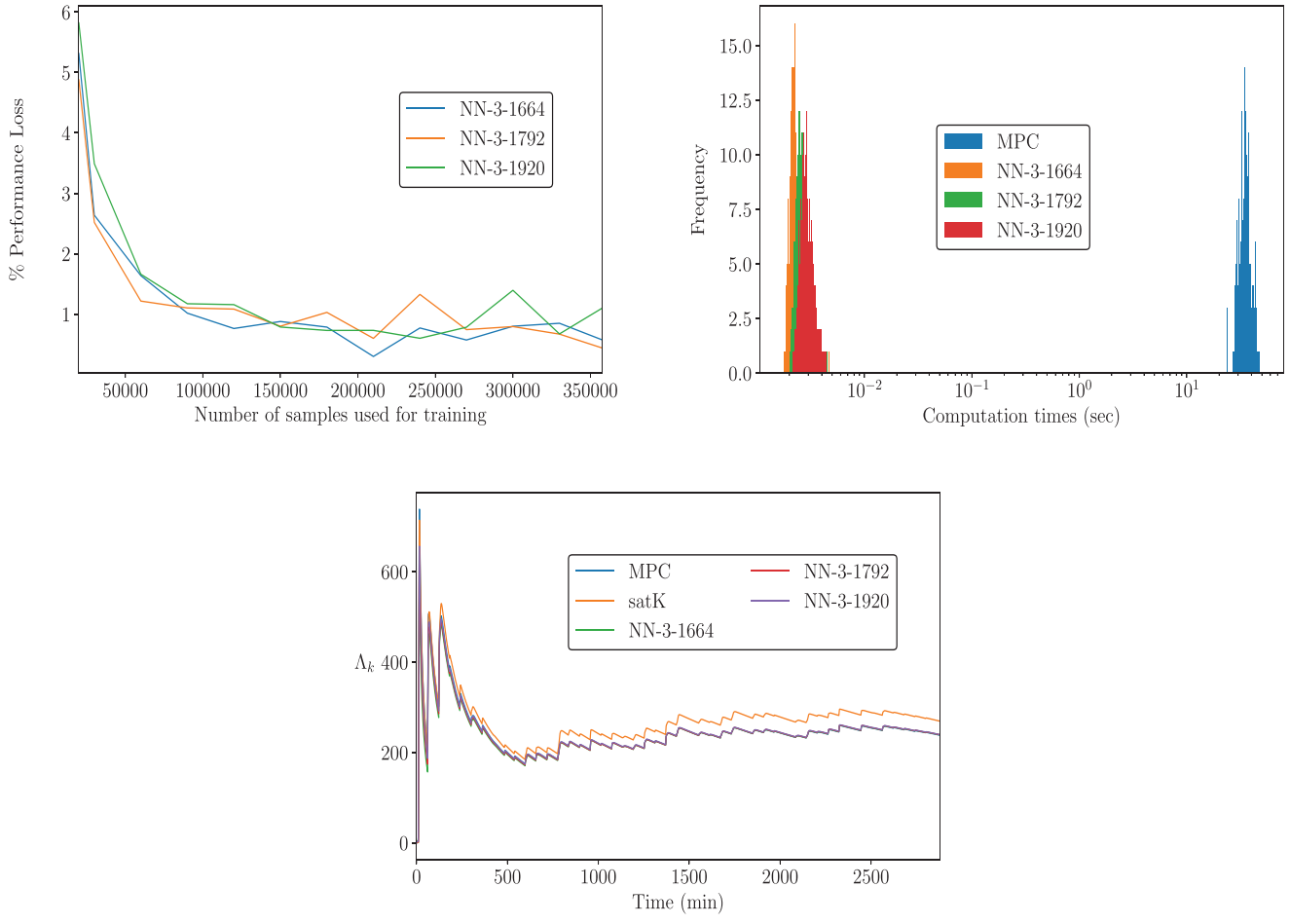

Fig. 6. Simulation statistics for the industrial crude distillation unit example. Data requirement to obtain accurate NN controllers (top left), online computation times of the NNs and MPC controller (top right), and the controller performance index obtained with the NNs, satK and MPC controllers (center bottom). The performance of the three NN and optimal MPC controllers are indistinguishable.

Table 2
Summary of the simulation study for the industrial crude distillation unit example.

Controller	Neural network architecture	Number of parameters	Memory footprint (MB)	Training time (min)	% Performance loss	Average speedup	Worst case speedup
SS					120.59 %		
satK					13.07 %	4.83×10^5	1.51×10^5
SH					1.56 %	6.66×10^3	2.51×10^3
NN-3-1664	[1072, 1664, 1664, 1664, 32]	1.85×10^6	14.18	52.66	0.29 %	1.52×10^4	6.57×10^3
NN-3-1792	[1072, 1792, 1792, 1792, 32]	2.11×10^6	16.15	55.27	0.43 %	1.33×10^4	5.25×10^3
NN-3-1920	[1072, 1920, 1920, 1920, 32]	2.39×10^6	18.24	59.24	0.59 %	1.18×10^4	4.48×10^3

constructed with a horizon length of $N = 3$. The states and target steady-states (x, x_s) are scaled for the NN training as in the previous example.

Fig. 6 shows the statistics for the simulation study performed for this example. The data requirement to obtain an accurate NN controller is shown in the top left. The batch size in Adam was chosen as 2048 samples, and all the NNs are trained for 1500 epochs. The size of the buffer dataset was chosen as 5% of the training dataset. The transient controller performance index for one validation simulation obtained with the three structured NNs, satK, and MPC controllers is shown in the center bottom. The top right plot shows the histogram of online computation times of the NNs and QP solver. Table 2 gives a summary of the maximum time required to train the NNs, number of parameters in each NN, memory footprint of the NNs, and the % performance loss metric obtained by the structured NNs, SS, satK, and SH controllers in the validation simulations.

The offline data generation and training times for the NNs shows that this large-scale example is tractable with the proposed NN approach. This example is challenging, since on average the QP solver CVXOPT requires 35 seconds, and 47 seconds in the worst-case. By comparison, all the NNs execute MPC in about 2 to 7 milliseconds. As observed from the plot of the transient controller performance index, the performance degradation by the NNs relative to the optimal MPC controller is negligible. The sampling of the state space with the setpoint and disturbance signals for one particular operational scenario, combined with the offset-free NN structure are the key properties that enable scalability in this example.

6. Conclusion

Neural networks have been demonstrated to approximate the MPC feedback law for treating large-scale, linear MPC applications that may be out of reach with online QP solvers. The proposed methods avoiding online optimization to date in the literature, such as storing the entire MPC feedback law and sampling the large dimensional state space for the NN training, are not tractable for the problems considered in this article.

The key features for the scalability of the proposed approach are the structure of the NN and data generation for only typical plant operational scenarios. The NN design approach was demonstrated to scale on an industrial crude distillation unit model with state and control dimensions of 252 and 32 respectively, and a control-sample horizon length of 140. After the offline design phase, NNs execute MPC about three to five orders of magnitude faster than an available QP solver with less than 1% performance degradation in the closed-loop. The deployment decision of the NNs must be based on the ability to reliably identify and sample the anticipated plant operational scenarios with setpoints and large magnitude disturbances that may change for a reasonable duration of operation. Parallel computing can be used for the offline data generation step, if solving QPs for each sampled state starts to take excessive amounts of time. After the training process, the NNs are easily deployable on memory constrained hardware. We established conditions under which the NN controllers are robust to state estimation errors and process disturbances, providing some theoretical support to practitioners interested in the deployment of NNs.

The online QP based MPC was adopted by practitioners due to its ability to systematically handle multivariable systems and process constraints, and the reliability of QP solution algorithms. The approach has guaranteed stability properties. The need for sampling the plant operational scenarios and additional computing hardware for training large applications are additional complexities that must be considered in the design of NN controllers. The advantage of NNs is their speed of MPC controller execution and convenience for deployment on memory constrained hardware. Future research may be focused to establish conditions for guaranteed closed-loop stability with NN controllers, systematically handling state constraints, approximating the target selector QP, and extensions to large, nonlinear MPC applications.

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

Pratyush Kumar: Conceptualization, Simulations, Writing - original draft. **James B. Rawlings:** Conceptualization, Writing - review & editing, Funding acquisition, Supervision. **Stephen J. Wright:** Conceptualization, Writing - review & editing, Supervision.

Acknowledgments

The funding for this work was provided by the industrial members of the Texas-Wisconsin-California Control Consortium (TWCCC). The computational facilities purchased with funds from the National Science Foundation (CNS-1725797, OAC-1925717) and administered by the Center for Scientific Computing (CSC) was used for the simulation studies presented in this article. The CSC is supported by the California NanoSystems Institute and the Materials Research Science and Engineering Center (MRSEC; NSF DMR 1720256) at UC Santa Barbara.

Appendix A. Proof of Theorem 1

We use the following definitions of robust positive invariance of a set with respect to disturbances, input-to-state stability (ISS), ISS-Lyapunov function, and the available result on the ISS property of a system if it admits an ISS-Lyapunov function (Jiang and Wang, 2001; Rawlings et al., 2017, Appendix B).

Definition 1 (Robust positive invariance)**.** A closed set X is robustly positive invariant for the system $x^+ = f(x, w),\ w \in \mathbb{W}$ if $x \in X$ implies $f(x, w) \in X$ for all $w \in \mathbb{W}$.

Definition 2 (Input-to-state stability (ISS)). $\mathbb{W}$ is a compact set containing the origin and that X is a closed robustly positive invariant set for the system $x^+ = f(x,w),\ w \in \mathbb{W}$. This system is ISS in X if there exist functions $\beta(\cdot) \in \mathcal{KL}$ and $\sigma(\cdot) \in \mathcal{K}$, such that for all $x \in X$ and $w(i) \in \mathbb{W},\ i \in \mathbb{I}_{0:k-1},\ |\psi(k;x,\mathbf{w}_k)| \le \beta(|x|,k) + \sigma(||\mathbf{w}_k||)$, in which $\psi(k;x,\mathbf{w}_k)$ is the solution of the system at time k, starting from an initial state x, and with the disturbance sequence $\mathbf{w}_k$ affecting the system.

Definition 3 (ISS-Lyapunov function). A function $V : X \to \mathbb{R}_{\ge 0}$ is an ISS-Lyapunov function in X for the system $x^+ = f(x,w),\ w \in \mathbb{W}$ if there exist functions $\alpha_1(\cdot), \alpha_2(\cdot), \alpha_3(\cdot) \in \mathcal{K}_\infty$ and $\sigma(\cdot) \in \mathcal{K}$ such that for all $x \in X$ and $w \in \mathbb{W}$, we have

$$\alpha_1(|x|) \le V(x) \le \alpha_2(|x|),$$
$$V(f(x,w)) - V(x) \le -\alpha_3(|x|) + \sigma(|w|).$$

Proposition 1 (ISS-Lyapunov function implies ISS). *$\mathbb{W}$ is a compact set containing the origin and that X is a closed robustly positive invariant set for $x^+ = f(x,w),\ w \in \mathbb{W}$. If $f(\cdot)$ is continuous and there exists a continuous ISS-Lyapunov function in X for the system $x^+ = f(x,w),\ w \in \mathbb{W}$, then the system is ISS in X.*

Additionally, we require the following two useful propositions from Allan et al. (2017) and Rawlings and Ji (2012), to bound values of continuous functions and an inequality for $\mathcal{K}$ functions respectively.

Proposition 2. *Let $C \subseteq D \subseteq \mathbb{R}^n$, C compact, D closed, and $f : D \to \mathbb{R}^n$ continuous. Then there exists a $\sigma(\cdot) \in \mathcal{K}_\infty$, such that for all $x \in C$ and $y \in D$, we have that $|f(x) - f(y)| \le \sigma(|x-y|)$.*

Proposition 3. *Let $\sigma(\cdot) \in \mathcal{K}$, the following holds for all $a_i \in \mathbb{R}_{\ge 0},\ i \in \mathbb{I}_{1:n}$:*

$$\sigma(a_1 + a_2 + \cdots + a_n) \le \sigma(na_1) + \sigma(na_2) + \cdots + \sigma(na_n)$$

Proof of Theorem 1: We proceed in two steps. First, we show that there exist constants $\delta_1, \delta_2, \delta_3 > 0$ such that the set S is robustly positive invariant. Then, we show that $V_N^0(\hat{x})$ is an ISS Lyapunov function for the perturbed system (10) on this set S. Denote $\tilde{x}^+ = A\hat{x} + B\kappa_N(\hat{x})$ as the nominal state evolution using the optimal MPC controller, and $d(k) = [e_{NN}(\hat{x}(k))'B', w(k)', e(k)', e(k+1)']'$ as the combination of all the disturbances at a given time k.

To show robust positive invariance of S, we establish that if $V_N^0(\hat{x}) \le \rho$, then $V_N^0(\hat{x}^+) \le \rho$. Since the optimal cost function $(V_N^0(\hat{x}))$ is continuous, using $\mathbb{R}^n$ as the closed set $(\hat{x}^+ \in \mathbb{R}^n)$ and S as the compact set $(\tilde{x} \in S)$, from Proposition 2 we have for some $\sigma_1(\cdot) \in \mathcal{K}_\infty$ that

$$\begin{aligned}
&\left|V_N^0(\hat{x}^+) - V_N^0(\tilde{x}^+)\right| \le \sigma_1\left(\left|\hat{x}^+ - \tilde{x}^+\right|\right)\\
&\le \sigma_1\left(\left|Be_{NN}(\hat{x})\right| + |w| + |A||e| + \left|e^+\right|\right)\\
&\le \sigma_1(|d| + |d| + |A||d| + |d|) := \sigma_2(|d|) \qquad (11)
\end{aligned}$$

in which $\sigma_2(s) := \sigma_1(|A|s + 3s)$, and the last inequality follows because $|a| \le \left|[a', b']'\right|$, in which a and b are both vectors, and $\sigma_2(\cdot) \in \mathcal{K}_\infty$. We partition S in outer and inner parts based on the value of the optimal cost $V_N^0(\hat{x})$, and bound the combination of disturbances d in each part such that $V_N^0(\hat{x}^+) \le \rho$.

Case 1: $\rho/2 \le V_N^0(\hat{x}) \le \rho$. Using the upper bound on the optimal cost, we have $\rho/(2c_2) \le \left|\hat{x}\right|^2$ for all $\hat{x}$ in this part of S. To analyze the worst-case disturbance, we have from the equation (11) that $V_N^0(\hat{x}^+) \le V_N^0(\tilde{x}^+) + \sigma_2(|d|)$. From nominal MPC cost decrease, we obtain $V_N^0(\hat{x}^+) \le V_N^0(\hat{x}) - c_1\left|\hat{x}\right|^2 + \sigma_2(|d|)$. Thus, if $\sigma_2(|d|) \le \rho c_1/(2c_2)$, at all times in this outer part of S, we have $V_N^0(\hat{x}^+) \le \rho$.

Case 2: $V_N^0(\hat{x}) \le \rho/2$. From Eq. (11) and nominal MPC cost decrease, we have for this part of S that $V_N^0(\hat{x}^+) \le \rho/2 + \sigma_2(|d|)$. Thus, if $\sigma_2(|d|) \le \rho/2$, at all times in this inner part of S, we have $V_N^0(\hat{x}^+) \le \rho$.

Therefore, if $|d| \le \min\{\sigma_2^{-1}(\rho c_1/(2c_2)), \sigma_2^{-1}(\rho/2)\} := \delta_{\max}$ at all times, then the set S is robustly positive invariant. Since $c_1 \le c_2$, $\delta_{\max} = \sigma_2^{-1}(\rho c_1/(2c_2))$. The constants δ_1 and δ_2 are $\delta_{\max}/4$ each, and δ_3 is $\delta_{\max}/(4|B|)$. The optimal MPC cost function satisfies

$$V_N^0(\hat{x}^+) - V_N^0(\hat{x}) \le -c_1\left|\hat{x}\right|^2 + \sigma_2(|d|)$$

in the robustly positive invariant set S and the disturbance set $|d| \le \delta_{\max}$. Since the optimal cost function also satisfies the upper and lower bounds $c_1|x|^2 \le V_N^0(x) \le c_2|x|^2$, from the Definition 3, it satisfies the requirements of an ISS-Lyapunov function. Hence, using the Proposition 1, the perturbed linear system (10) is ISS (Definition 2), and there exist functions $\beta(\cdot) \in \mathcal{KL}$ and $\sigma(\cdot) \in \mathcal{K}$, such that $\left|\phi_d(k;\hat{x})\right| \le \beta\left(\left|\hat{x}\right|, k\right) + \sigma(||\mathbf{d}_k||)$. Using Proposition 3 and the definition of $d(k)$, we have

$$\begin{aligned}
\left|\phi_d(k;\hat{x})\right| &\le \beta\left(\left|\hat{x}\right|, k\right) + \sigma(|B|\bar{e}_{NN} + ||\mathbf{w}_k|| + 2||\mathbf{e}_{k+1}||)\\
&\le \beta\left(\left|\hat{x}\right|, k\right)\\
&\quad + \sigma(3|B|\bar{e}_{NN}) + \sigma(3||\mathbf{w}_k||) + \sigma(6||\mathbf{e}_{k+1}||)
\end{aligned}$$

Therefore, the bound in the Theorem 1 is established with $\alpha_e(s) := \sigma(6s)$, $\alpha_w(s) := \sigma(3s)$, $\alpha_n(s) := \sigma(3|B|s)$, and $\alpha_e(\cdot), \alpha_w(\cdot), \alpha_n(\cdot) \in \mathcal{K}$.

Appendix B. Parameters used in the CSTRs example

Table 3
Parameters used in the ODEs to simulate the plant in the CSTRs in series with a flash separator example, actuator constraints, setpoint and disturbance bounds, and the steady state used to obtain the linear model for the MPC controller.

Parameter	Value	Unit	Parameter	Value	Unit
Parameters used to simulate the ODEs					
α_A	3.5		k_m	2.5	m^2
α_B	1.1		k_b	1.5	m^2
α_C	0.5		ΔH_1	40	kJ/kg
ρ	50	kg/m^3	ΔH_2	50	kJ/kg
C_p	3	kJ/kg-K	E/R	150	K
A_r	0.3	m^2	k_1^*	4×10^{-4}	sec^{-1}
A_m	2	m^2	k_2^*	1.8×10^{-6}	sec^{-1}
A_b	4	m^2	T_d	313	K
k_r	2.5	m^2			
Actuator constraints $(\underline{u}, \overline{u})$					
F_0	(1.5, 2.5)	kg/sec	Q_r	(500, 500)	kW
F_1	(0.5, 1.5)	kg/sec	Q_m	(500, 500)	kW
D	(29.5, 30.5)	kg/sec	Q_b	(500, 500)	kW
Setpoint bounds $(\underline{r}_{sp}, \overline{r}_{sp})$					
H_r	(158.8, 168.8)	m	T_r	(303, 323)	K
H_m	(169.2, 179.2)	m	T_m	(310, 316)	K
H_b	(2.2, 4.2)	m	T_b	(303, 323)	K
Disturbance bounds $(\underline{d}, \overline{d})$					
x_{A0}	(0.7, 0.85)		x_{B1}	(0, 0.15)	
x_{B0}	(0, 0.15)		T_0	(305, 321)	K
x_{A1}	(0.7, 0.85)				
Steady state used for linearization (x_s, u_s, d_s)					
H_r	163.8	m	F_0	2	kg/sec
x_{Ar}	0.40		F_1	1	kg/sec
x_{Br}	0.54		D	30	kg/sec
T_r	313.1	K	Q_r	0	kW
H_m	174.2	m	Q_m	0	KW
x_{Am}	0.37		Q_b	0	kW
x_{Bm}	0.58		x_{A0}	0.8	
T_m	313.7	K	x_{B0}	0.1	
H_b	3.24	m	x_{A1}	0.8	
x_{Ab}	0.15		x_{B1}	0.1	
x_{Bb}	0.73		T_0	313	K
T_b	313.7	K			

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., Zheng, X., 2015. TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

Allan, D.A., Bates, C.N., Risbeck, M.J., Rawlings, J.B., 2017. On the inherent robustness of optimal and suboptimal nonlinear MPC. Syst. Control Lett. 106, 68–78. doi:10.1016/j.sysconle.2017.03.005.

Allen-Zhu, Z., Li, Y., Liang, Y., 2019. Learning and generalization in overparameterized neural networks, going beyond two layers. In: Advances in Neural Information Processing Systems, pp. 6158–6169.

Arora, S., Du, S.S., Hu, W., Li, Z., Wang, R., 2019. Fine-grained analysis of optimization and generalization for overparameterized two-layer neural networks. In: Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9–15 June 2019, Long Beach, California, USA.

Belkin, M., Hsu, D., Ma, S., Mandal, S., 2019. Reconciling modern machine-learning practice and the classical bias–variance trade-off. Proc. Natl. Acad. Sci. 116 (32), 15849–15854.

Bemporad, A., Morari, M., Dua, V., Pistikopoulos, E.N., 2002. The explicit linear quadratic regulator for constrained systems. Automatica 38 (1), 3–20.

Bemporad, A., Oliveri, A., Poggi, T., Storace, M., 2011. Ultra-fast stabilizing model predictive control via canonical piecewise affine approximations. IEEE Trans. Autom. Control 56 (12), 2883–2897.

Cavagnari, L., Magni, L., Scattolini, R., 1999. Neural network implementation of nonlinear receding-horizon control. Neural Comput. Appl. 8 (1), 86–92.

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.

Chen, S. W., Wang, T., Atanasov, N., Kumar, V., Morari, M., 2019. Large scale model predictive control with neural networks and primal active sets. arXiv preprint arXiv:1910.10835

Drgoňa, J., Picard, D., Kvasnica, M., Helsen, L., 2018. Approximate model predictive building control via machine learning. Appl. Energy 218, 199–216.

Heath, W., Wills, A., 2005. The inherent robustness of constrained linear model predictive control. IFAC Proc. Vol. 38 (1), 71–76.

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.

Jiang, Z.-P., Wang, Y., 2001. Input-to-state stability for discrete-time nonlinear systems. Automatica 37, 857–869.

Karg, B., Alamo, T., Lucia, S., 2019. Probabilistic performance validation of deep learning-based robust NMPC controllers. arXiv preprint arXiv:1910.13906

Kingma, D. P., Ba, J., 2014. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980

Karg, B., Lucia, S., 2020. Efficient representation and approximation of model predictive control laws via deep learning. IEEE Trans. Cybern. 50 (9), 3866–3878. doi:10.1109/TCYB.2020.2999556.

Kouzoupis, D., Frison, G., Zanelli, A., Diehl, M., 2018. Recent advances in quadratic programming algorithms for nonlinear model predictive control. Vietnam J. Math. 46 (4), 863–882.

Kvasnica, M., Löfberg, J., Fikar, M., 2011. Stabilizing polynomial approximation of explicit MPC. Automatica 47 (10), 2292–2297.

Lahiri, S.K., 2017. Multivariable Predictive Control: Applications in Industry. John Wiley & Sons, Inc., Hoboken, NJ.

Limon, D., Alamo, T., Salas, F., Camacho, E.F., 2006. On the stability of MPC without terminal constraint. IEEE Trans. Autom. Control 51 (5), 832–836.

Lovelett, R.J., Dietrich, F., Lee, S., Kevrekidis, I.G., 2020. Some manifold learning considerations toward explicit model predictive control. AIChE J. 66 (5), e16881.

Mayne, D.Q., Rawlings, J.B., Rao, C.V., Scokaert, P.O.M., 2000. Constrained model predictive control: stability and optimality. Automatica 36 (6), 789–814.

Montufar, G.F., Pascanu, R., Cho, K., Bengio, Y., 2014. On the number of linear regions of deep neural networks. In: Advances in Neural Information Processing Systems. Curran Associates, Inc., Red Hook, NY, pp. 2924–2932.

Pannocchia, G., Rawlings, J.B., 2003. Disturbance models for offset-free MPC control. AIChE J. 49 (2), 426–437.

Pannocchia, G., Rawlings, J.B., Wright, S.J., 2007. Fast, large-scale model predictive control by partial enumeration. Automatica 43, 852–860.

Pannocchia, G., Rawlings, J.B., Wright, S.J., 2011. Conditions under which suboptimal nonlinear MPC is inherently robust. Syst. Control Lett. 60, 747–755.

Paulson, J.A., Mesbah, A., 2020. Approximate closed-loop robust model predictive control with guaranteed stability and constraint satisfaction. IEEE Control Syst. Lett. 4 (3), 719–724.

Qin, S.J., Badgwell, T.A., 2003. A survey of industrial model predictive control technology. Control Eng. Pract. 11 (7), 733–764.

Rao, C.V., Rawlings, J.B., 1999. Steady states and constraints in model predictive control. AIChE J. 45 (6), 1266–1278.

Rawlings, J.B., Ji, L., 2012. Optimization-based state estimation: current status and some new results. J. Proc. Control 22, 1439–1444.

Rawlings, J.B., Mayne, D.Q., Diehl, M.M., 2017. Model Predictive Control: Theory, Design, and Computation, second ed. Nob Hill Publishing, Madison, WI 770 pages, ISBN 978-0-9759377-3-0.

Seron, M.M., Goodwin, G.C., De Doná, J.A., 2003. Characterisation of receding horizon control for constrained linear systems. Asian J. Control 5 (2), 271–286.

Sontag, E.D., Wang, Y., 1995. On the characterization of the input to state stability property. Syst. Control Lett. 24, 351–359.

Vandenberghe, L., 2010. The CVXOPT linear and quadratic cone program solvers. Online: http://cvxopt.org/documentation/coneprog.pdf.

Venkat, A.N., 2006. Distributed Model Predictive Control: Theory and Applications. University of Wisconsin–Madison.

Wen, C., Ma, X., Ydstie, B.E., 2009. Analytical expression of explicit MPC solution via lattice piecewise-affine function. Automatica 45 (4), 910–917.

Wright, S.J., 2019. Efficient convex optimization for linear MPC. In: Handbook of Model Predictive Control. Springer, pp. 287–303.

Zhang, C., Bengio, S., Hardt, M., Recht, B., Vinyals, O., 2017. Understanding deep learning requires rethinking generalization. In: 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24–26, 2017, Conference Track Proceedings.