

Active Set Prediction using Machine Learning Methods for Complexity Reduction in Nonlinear MPC

Raphael Dyrska^{*a}, Karol Kiš^b, Martin Klaučo^b, and Martin
Mönnigmann^a

^a*Automatic Control and Systems Theory, Ruhr-Universität Bochum, Germany*

^b*Institute of Information Engineering, Automation, and Mathematics, Slovak
University of Technology in Bratislava, Slovakia*

November 2025

Preprint

Abstract

We use Machine Learning (ML) methods to simplify the Nonlinear Program (NLP) arising in Nonlinear Model Predictive Control (nonlinear MPC, NMPC). Since every solution to an NMPC problem defines a set of active and a set of inactive constraints, we propose to use predictions about these sets to reduce the complexity of the NLP to be solved. Specifically, we use ML methods to predict active sets for NMPC problems. The required classification networks are simple enough to be evaluated online, i.e., during the runtime of the controller. They can be trained to a high accuracy, qualifying as suitable candidates for an application to NMPC. The results are evaluated using numerical simulations for a model of a continuous stirred-tank reactor.

Keywords: nonlinear model predictive control, machine learning, neural networks, classification, active sets

^{*}Corresponding author. Email address: raphael.dyrska@rub.de

1 Introduction

In automatic control, the concept of Nonlinear Model Predictive Control (Nonlinear MPC, NMPC) was one of the main research topics addressed during the last decades, due to the ability of NMPC to handle both constraints and systems with multiple inputs and outputs (see, e.g., [1, 2, 3, 4, 5]). In NMPC, the control signal applied to the system is determined by solving a constrained nonlinear program (NLP) in every time step. Solving the NLP in every time step can be computationally demanding and might be restrictive for certain applications. Thus, intense research focused on approaches reducing the computational effort for solving the underlying NLP. In the classic NMPC literature, these approaches can be divided into contributions addressing the algorithmic performance to speed-up the solution process itself (see, e.g., [6, 7]) and approaches exploiting information about the structure of the solution to either simplify the underlying NLP before solving it (see, e.g., [8, 9]) or to even avoid its solution (see, e.g., [10, 11, 12]).

The combination of linear and nonlinear MPC with Machine Learning (ML) methods is another well-studied field. Many contributions use neural networks to replace the controller completely (see, e.g., [13, 14, 15, 16, 17]). However, ML methods also have the potential to assist NMPC and vice-versa. ML methods are, for example, often used to form the prediction model (see, e.g., [18]) or to tune the cost function of the optimization problem (see, e.g., [19]). Conversely, MPC methods can be used to provide safety guarantees to learning-based controllers (see, e.g., [20]). More recent approaches use ML methods to cope with uncertainties, by predicting plant-model mismatch, for example, but also other sources of uncertainty (see, e.g., [21, 22, 23]).

Another branch of approaches uses ML methods to accelerate the solution of the MPC problem (see, e.g., [24, 25]). In MPC, a solution to the underlying optimization problem determines the active set, i.e., the set of inequality constraints that actually hold with equality at the optimal solution. The authors in [26] propose to use classification algorithms to predict the set of constraints that are expected to be active for the optimal solution of a linear MPC problem and to use this prediction for warm-starting an active-set algorithm used for solving the underlying optimization problem.

In this article, we extend the idea of predicting active sets using ML

methods as presented by [26] from linear to nonlinear MPC problems. However, instead of using the predicted active sets on an algorithmic level as proposed by [26], we use a straightforward approach for simplifying the NLP before solving it. In previous works (see [8, 27], see also [28, 29]) we used *a priori* information about active and thus inactive constraints to simplify the underlying NLP by rearranging and removing constraints. While the prediction of the active set was based on the principle of optimality for NMPC on a shrinking horizon in [27], we used predictions based on the solution of the previous time step to simplify the NLP also for the classic, receding horizon case in [8]. Following these works, we now propose to use ML methods to predict active constraints and to apply the same simplification approach to the underlying NLP. The non-convexity of the underlying NLP poses challenges for the generation of training data since, for example, several local minima with the same active set and several local minima with different active sets may exist for the same initial state (see [30] for properties of NMPC solutions resulting from non-convexity).

In Sections 2.1 and 2.2, we state the class of NMPC problems investigated here and introduce a model of a continuous stirred-tank reactor (CSTR) that serves as an example throughout the paper, respectively. Section 3 states the methods applied in this article, specifically the process of generating training data (Sec. 3.1), the type of classification networks and their training for active constraint prediction (Sec. 3.2), and the simplification approach for a known set of active constraints (Sec. 3.3). In Section 4, we provide closed-loop simulation results for the CSTR introduced in Section 2.2. Conclusions and an outlook to future work are given in Section 5.

2 Problem Statement

2.1 NMPC Problem Class

We consider the class of nonlinear dynamical systems in discrete time given by difference equations

$$x(k+1) = f(x(k), u(k)), \quad k = 0, 1, \dots, \quad (1)$$

with system states $x(k) \in \mathbb{R}^n$ and input signals $u(k) \in \mathbb{R}^m$. We assume the nonlinear mapping $f : \mathbb{R}^n \times \mathbb{R}^m \rightarrow \mathbb{R}^n$ to be twice continuously differentiable

and we assume $f(0, 0) = 0$.

To regulate system (1) to the origin we solve, in every time step, the constrained nonlinear optimization problem

$$\min_{X, U} \|x(N)\|_P^2 + \sum_{k=0}^{N-1} (\|x(k)\|_Q^2 + \|u(k)\|_R^2) \quad (2a)$$

$$\text{s.t. } x(k+1) = f(x(k), u(k)), \quad k \in \mathbb{N}_0^{N-1}, \quad (2b)$$

$$x(k) \in \mathcal{X}, \quad k \in \mathbb{N}_0^{N-1}, \quad (2c)$$

$$u(k) \in \mathcal{U}, \quad k \in \mathbb{N}_0^{N-1}, \quad (2d)$$

$$x(N) \in \mathcal{T}, \quad (2e)$$

$$x(0) = x(k), \quad (2f)$$

for horizon length N , with optimization variables $X = (x(1)^\top, \dots, x(N)^\top)^\top$ and $U = (u(0)^\top, \dots, u(N-1)^\top)^\top$ describing the predicted states and inputs, respectively. The matrices P , Q and R have obvious dimensions and are assumed to be positive definite. The sets \mathcal{X} and \mathcal{U} are assumed to contain the origin in their interiors and to be defined by finite numbers of linear inequalities, which implies they are convex polyhedra. The terminal set \mathcal{T} in (2e) is an ellipsoid here (see, e.g., [31] for this problem class)

$$\mathcal{T} = \{x | x^\top P x \leq \gamma\}, \quad (3)$$

which implies it is represented by one additional inequality. We assume there exists a stabilizing control law $\kappa(x)$ in \mathcal{T} such that the terminal cost function $\|x(N)\|_P^2$, the terminal set \mathcal{T} , and $\kappa(x)$ fulfill the stability conditions summarized as in [1], A1–A4.

To simplify the notation, we introduce $z = (X^\top, U^\top)^\top$ and rewrite the optimization problem (2) in the compact form

$$\min_z \|z\|_H^2 \quad (4a)$$

$$\text{s.t. } F(x(0), z) = 0, \quad (4b)$$

$$G(x(0), z) \leq 0, \quad (4c)$$

with $H \in \mathbb{R}^{N(n+m) \times N(n+m)}$, $H \succ 0$. In (4), the equality constraints are

abbreviated as

$$F(x(0), z) := \begin{pmatrix} x(1) - f(x(0), u(0)) \\ \vdots \\ x(N) - f(x(N-1), u(N-1)) \end{pmatrix},$$

and the inequality constraints described by (2c)–(2e) are summarized in $G(x(0), z)$.

Let the optimal solution to (4) for an initial state $x(0)$ be denoted by $z^*(x(0))$, which we often abbreviate by z^* . If not stated otherwise, the optimal solution to (4) is assumed to be a local minimum. Let $\mathcal{Q} = \{1, \dots, q\}$ denote the set of indices of all q constraints. A constraint $i \in \mathcal{Q}$ is called active for the optimizer z^* if row i of G is zero, i.e. $G_i(x(0), z^*) = 0$. We recall that inactive constraints are fulfilled with strict inequality, i.e., $G_i(x(0), z^*) < 0$. We introduce the active and inactive set \mathcal{A} and \mathcal{I} collecting the indices of active and inactive constraints, respectively. We write $\mathcal{A}(x)$ for an active set that results for the initial state x .

2.2 Benchmark Process: Continuous Stirred-Tank Reactor

We use the following CSTR model [32] as an illustrating example throughout the paper:

$$\frac{dc_A}{dt} = \frac{q}{V}(c_{A,f} - c_A) - k_0 e^{-\frac{E}{RT}} c_A \quad (5a)$$

$$\frac{dT}{dt} = \frac{q}{V}(T_{A,f} - T) - \frac{\Delta H k_0}{\rho c_p} e^{-\frac{E}{RT}} c_A + \frac{UA}{V \rho c_p}(T_c - T). \quad (5b)$$

State variables c_A and T describe the concentration of substance A and the reactor temperature, respectively, and the manipulated variable T_c is the temperature of the coolant stream. Parameter values are collected in Table 1.

The control task is to regulate the system to an unstable steady state. Following [32], we define the state and input vectors

$$x_1 = \frac{c_A - 0.5}{0.5}, \quad x_2 = \frac{T - 350}{20}, \quad u = \frac{T_c - 300}{20}$$

by shifting the variables to the unstable equilibrium. The system is discretized with a sampling time of $T_s = 0.03$ min, and the weighting matrices

Table 1: Parameter values of the CSTR.

parameter	value	parameter	value
ρ	1000 g L^{-1}	c_p	$0.239 \text{ J g}^{-1} \text{ K}^{-1}$
ΔH	$-5 \times 10^4 \text{ J mol}^{-1}$	$\frac{E}{R}$	8750 K
k_0	$7.2 \times 10^{10} \text{ min}^{-1}$	UA	$5 \times 10^4 \text{ J min}^{-1} \text{ K}^{-1}$
q	100 L min^{-1}	$T_{A,f}$	350 K
V	100 L	$c_{A,f}$	1.0 mol L^{-1}

in the NMPC cost function (2a) and the prediction horizon are chosen to be

$$P = \begin{pmatrix} 265.88 & 49.39 \\ 49.39 & 125.99 \end{pmatrix}, \quad Q = \begin{pmatrix} 10 & 0 \\ 0 & 10 \end{pmatrix}, \quad R = 1, \quad N = 10. \quad (6)$$

Furthermore, we introduce input and terminal constraints

$$-1 \leq u(k) \leq 3.5, \quad (7)$$

$$x(N) \in \mathcal{T}, \quad \mathcal{T} = \{x \in \mathcal{X} \mid x^\top P x \leq \gamma\}, \quad \gamma = 9.3353, \quad (8)$$

and the terminal controller as $\kappa(x) = K_{\mathcal{T}}x$, with $K_{\mathcal{T}} = (-1.6154, -3.6644)$, such that the terminal weighting matrix, the terminal set, and the terminal controller fulfill the stability conditions referred to in Section 2.1.

3 Active Set Prediction using Classification

We first introduce the generation of suitable training data, followed by a description of the training process of a classification network $\Phi(x)$ for predicting $\mathcal{A}(x)$. In the last subsection, we state a simplification approach for the NLP (4) as one possibility to benefit from the prediction of active sets.

3.1 Generation of Training Data

For a reliable prediction of an optimal active set, the training data should be representative for the expected initial states that occur during closed-loop control and consider features of the system at hand. As shown in [30], the non-convexity of the NLP can lead to the existence of several local minima with different active sets for the same initial state. This may lead to inconclusive training data for parts of the state space such as, e.g., areas

of states with mixed active sets, which will in general impede the training process in terms of convergence and resulting accuracy.

We showed in [30] that the resulting minimum strongly depends on the initial guess forwarded to the optimizer. During control, we often apply a warm-start that results from extending the solution of the previous time step using the terminal controller $\kappa(x)$ together with the prediction model. To generate the training data, we follow the strategy described in [26] for the linear case and regulate a well-distributed set of initial states closed-loop into the terminal set. The use of the same warm-start here is essential to minimize the influence of varying local minima corrupting the training data. Further, by generating the training data in a closed-loop fashion, the highest concentration of data will result for the area around the terminal set, which is also expected to be the area containing most of the states for which problem (4) needs to be solved when controlling the corresponding system. Formally, we summarize pairs of a specific x^i and \mathcal{A}^i as a data tuple \mathcal{D}^i for the training. Initial conditions resulting in the same active set will be summarized, and we call the resulting set a class for brevity. All optimization problems solved for generating training and testing data and during the computational study in Section 4 were implemented in Matlab using the nonlinear optimizer `fmincon` and the interior-point algorithm.

For the CSTR, we generated a grid of feasible initial states with a step size of $\Delta x_1 = \Delta x_2 = 0.01$. Regulating all states of the grid into the terminal set, in total 134 736 data tuples of states and corresponding optimal active set were determined. Tuples \mathcal{D}^i with equal active sets \mathcal{A}^i were summarized into one class, resulting in 185 different classes. Figure 1(a) shows the resulting state space, where initial states that yield the same optimal active set and thus belong to the same class are shown with the same color. Since the training data was generated by controlling all states of the initial grid into the terminal set \mathcal{T} , the highest concentration of training data results for the area around \mathcal{T} .

The training process can be expected to benefit from a balanced training set. Classes with only few members can be a sign for outliers or numerically unstable results. We opt for a heuristic reduction of the number of classes while simultaneously maintaining a high resolution of the training data. Since the data tuples of each class will be split into a training and a verification set for the internal training process in the ratio 3 : 1 (see Sec. 3.2

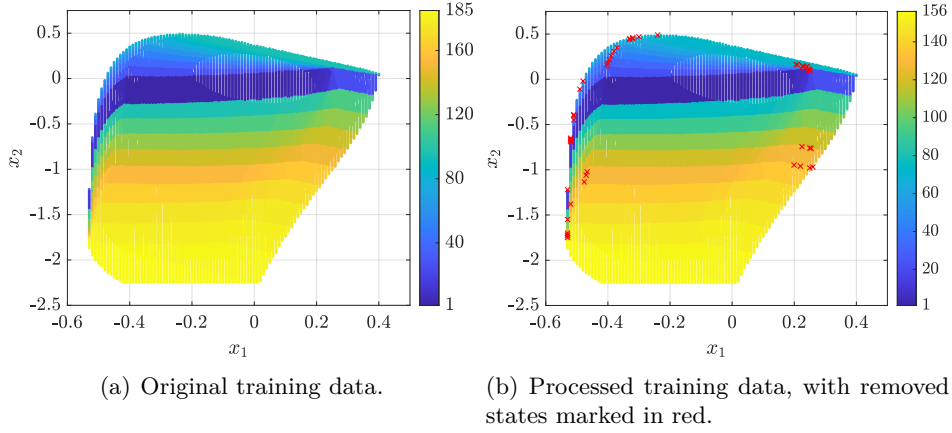


Figure 1: Training data for the classification network. The color of the states indicates the corresponding class, omitted states are shown in red.

below), we propose to remove classes with 3 or less pairs.

The histogram in Figure 2 visualizes the distribution of the 134 736 states into the 185 different classes. Obviously, the majority of states belongs to the class on the very left. At the same time, classes on the very right form only a fraction of the overall simulation data. Removing classes with 3 or less data tuples, the number of classes could be reduced by around 15.68%, from 185 to 156, while only 0.04% of all data tuples were removed. The states of the training set that have been removed are marked in red in Figure 1(b). The

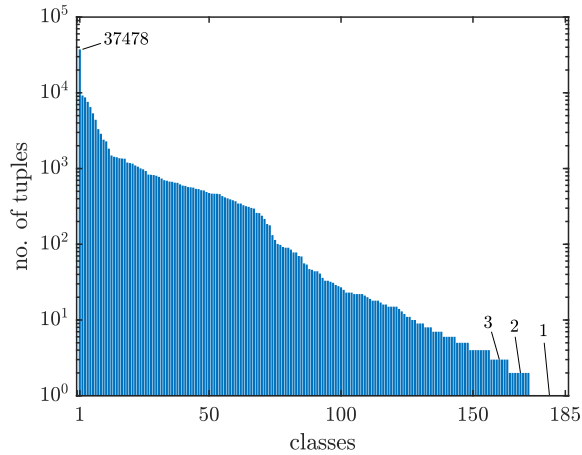


Figure 2: Number of data tuple per class. With 37 478 tuples, the class labeled as '1' is the highest represented class. Note that classes are sorted for cardinality and do not represent the enumeration depicted in Fig. 1.

position near the boundary of the feasible set indicates that (i) numerical issues might be a reason for outlier results and (ii) removing these states will not severely influence the results since in general most initial states will be located in the interior of the feasible set rather than close to its boundary.

3.2 Active Set Classification Network

A feed-forward neural network (NN) consisting of n_L fully connected layers is used as classification network $\Phi(x)$, mapping states x to a corresponding active set \mathcal{A} by classifying them into a class representing one unique combination of active constraints. The transformation for each hidden layer l with $1 \leq l < n_L$ follows

$$v^{(l)} = \omega^{(l)} a^{(l-1)} + \xi^{(l)}, \quad (9)$$

with v the vectorized input to each node, weights ω , biases ξ , and activation function a targeting the previous layer. For the activation function of the hidden layers, the Rectified Linear Unit (ReLU) function

$$a^{(l)} = \text{ReLU}(v^{(l)}), \quad \text{with} \quad \text{ReLU}(v) = \max(0, v),$$

is chosen to capture nonlinear behavior in the relation $x \rightarrow \mathcal{A}$. For the multi-class classification between n_a different classes, the activation function for each input i of the output layer n_L is chosen as the softmax function

$$a_i^{(n_L)} = \sigma(v_i^{(n_L)}) = \frac{\exp(v_i^{(n_L)})}{\sum_{c=1}^{n_a} \exp(v_c^{(n_L)})}, \quad (10)$$

to generate a probability for each of the n_a different classes. An illustration of the overall prediction process is shown in Figure 3.

Weights $\Omega = (\omega^{(1)}, \dots, \omega^{(n_L)})$ and biases $\Xi = (\xi^{(1)}, \dots, \xi^{(n_L)})$ for all n_L layers are the training parameters of the classification network $\Phi(x)$. They are optimized by solving the cross-entropy loss function

$$\min_{\Omega, \Xi} - \sum_{c=1}^{n_a} y_c \log \left(\sigma(v_c^{(n_L)}) \right), \quad (11)$$

which is a standard choice for multi-class classification problems. In (11), y_c indicates the presence of class c as for the input $x^i \in \mathcal{D}^i$, and the softmax

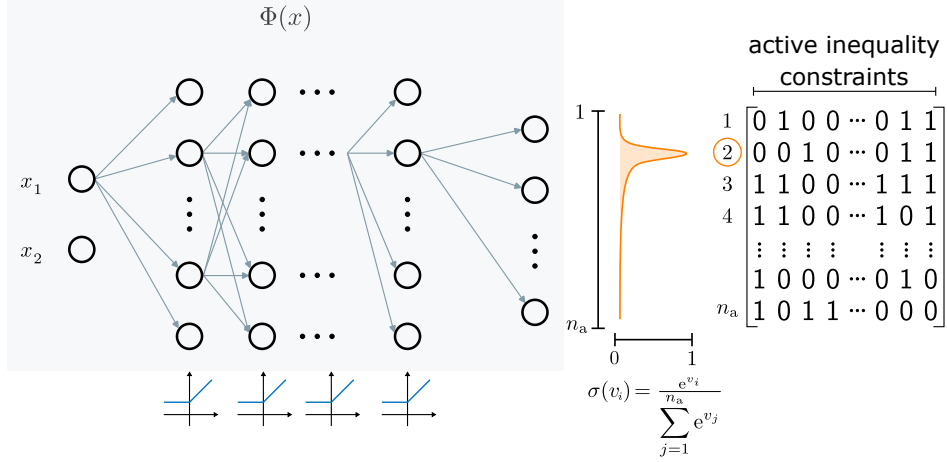


Figure 3: Illustration of the prediction of a class related to the active set $\mathcal{A}(x)$ with the classification network $\Phi(x)$.

function σ (see (10)) denotes the predicted probability that an input x^i belongs to a class c . The class labels are encoded using one-hot encoding (see [33]).

The resulting NN configuration for the CSTR consists of an input layer with two neurons for state x , four hidden layers containing 30 neurons each, and an output layer with 156 neurons, each corresponding to one of the $n_a = 156$ different classes. This results in a total of 7716 trainable parameters. The input features x of the NN are normalized to a range $[0, 1]$. The final configuration was determined by experiments evaluating the loss function (11) for different numbers of hidden layers.

The training set was split into a training and testing set with ratio 3 : 1. The training process was conducted in Python using the TensorFlow ML framework (see [34]). The Adam optimizer (see [35]) was used for model compilation. The initial learning rate was set to 0.001 and adjusted using the ReduceLROnPlateau scheduler (see [36]), which decreases the learning rate by a factor of 0.1 once learning stagnates, to a minimum of 10^{-6} . The training process was designed for 1000 epochs, with the actual number of epochs being adjusted using early stopping criteria to prevent overfitting.

We further generated another test set by regulating 1000 randomly chosen, feasible initial states into the terminal set, resulting in 7604 pairs of states and active set. For the pair-wise comparison, in total 7156 out of the 7604 active sets were predicted correctly by the classification network

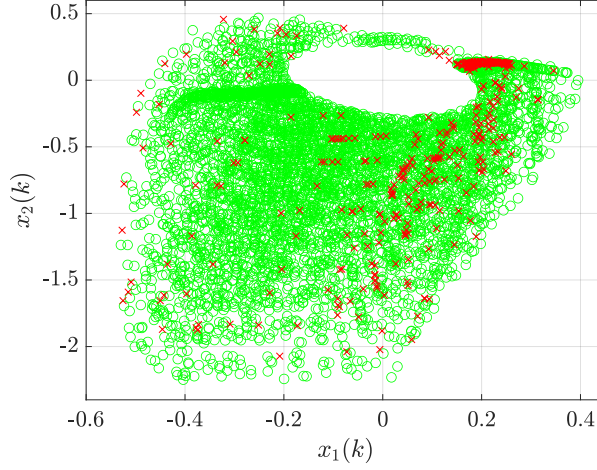


Figure 4: Test data for the CSTR indicating correct (green circles) and incorrect (red crosses) active set predictions. Note that the density of initial states is highest close to the ellipsoidal terminal set that can be anticipated by the white space.

$\Phi(x)$, resulting in an accuracy of around 94.11 %. Figure 4 shows all 7 604 states, with correct and incorrect predictions shown as green circles and red crosses, respectively. The same 1 000 initial states will also be used within the computational study to evaluate the reduction of the computational effort.

3.3 Exploiting information about active and inactive constraints

If the active set $\mathcal{A}(x)$ for the current state x is known in advance, we can simplify NLP (4) by changing constraints corresponding to \mathcal{A} to equality constraints, and removing constraints corresponding to $\mathcal{I} = \mathcal{Q} \setminus \mathcal{A}$. This simplification describes the core of active-set methods (see, e.g., [37, Chpt. 15.2]) and was already applied in previous works (see [8], [27, Prop. 3], see also [28, 29]).

For an active set \mathcal{A} predicted by the classification network $\Phi(x)$ for the

current system state x , we solve the simplified NLP

$$\begin{aligned}
& \min_z \|z\|_H^2 \\
& \text{s.t. } F(x, z) = 0 \\
& \quad G_i(x, z) = 0, \quad i \in \mathcal{A}.
\end{aligned} \tag{12}$$

If a solution z^* to (12) exists that does not violate the original constraints, i.e., $G(x, z^*) \leq 0$ holds, we apply the resulting control input to the system. If no solution exists or some of the constraints are violated by z^* , we solve the regular NLP as a fallback solution.

We note that, since the prediction of the active set $\mathcal{A}(x)$ may be inaccurate, a solution to the simplified NLP (12) that does not violate the original constraints can in general only be guaranteed to be feasible (see [28, 29]). We will see in Section 4, however, that the results for the regular and the simplified NLP are in general quite similar, if not identical. For readability, we thus keep the notation u^* for a resulting input also if the solution is suboptimal (i.e., not optimal, but feasible) for (4), but optimal for (12).

The proposed simplification approach is implemented in Algorithm 1. After evaluating $\Phi(x)$, we rearrange and remove the constraints of problem (4) and solve the simplified NLP (12) (lines 2–3). If a solution to (12) exists and none of the original constraints are violated by z^* , the resulting input signal u^* is applied to the system (lines 4–5). Solving the original NLP (4) as fallback strategy is only necessary if one of the aforementioned conditions are violated (lines 6–7). The procedure repeats for the next time step and updated initial state $x(0)$. A visual representation of the algorithm

Algorithm 1 NMPC using ML based active set prediction

- 1: **Input:** Current state $x(k)$, classification network $\Phi(x)$
 - 2: Determine active set $\mathcal{A}(x) = \Phi(x)$
 - 3: Solve (12) for $\mathcal{A}(x)$ and $x(0) = x(k)$
 - 4: **if** Solution z^* exists and $G(x(0), z^*) \leq 0$ **then**
 - 5: Apply u^* from z^*
 - 6: **else**
 - 7: Solve (4) for $x(0) = x(k)$
 - 8: Apply u^* from z^*
 - 9: **end if**
 - 10: **Output:** Measured state $x(k + 1)$ for next sampling instant
-

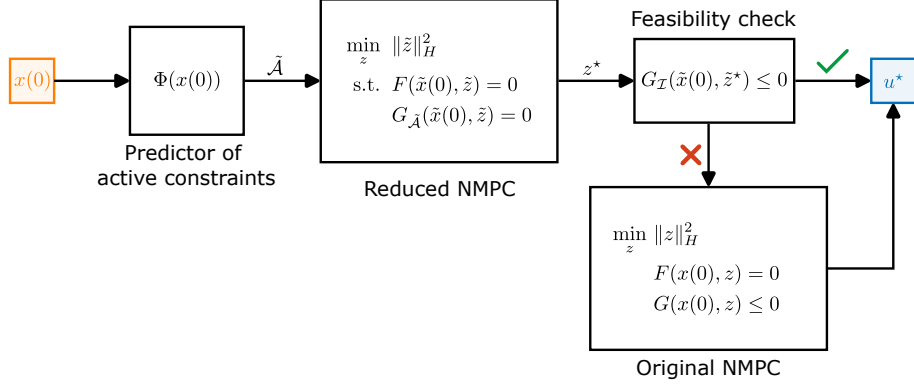


Figure 5: Block diagram of the proposed NMPC algorithm using ML based active set prediction.

is given in Figure 5.

Note that the approach summarized by Algorithm 1 does not decrease the worst-case computational effort. The results in Section 4 will prove, however, that we can reduce the average computational effort required for determining control inputs. The reduction of average computation times can be useful for, e.g., reducing the overall energy consumption when implemented on embedded devices (see, e.g., [38], Sec. 5.2 for an evaluation of energy saving potentials for embedded hardware).

4 Application to the CSTR

The accuracy of the classification network for a point-wise comparison was already examined along the training procedure (see column 1 in Tab. 2). To quantify the closed-loop benefits of the proposed approach, we used the same test set to evaluate the number of time steps in which the predicted active set resulted in a feasible input and compared the number of iterations necessary to find a solution using the regular (4) and the simplified (12) NLP. We distinguish between the accuracy of $\Phi(x)$ in a point-wise comparison and the number of simplified NLPs resulting in a feasible $u^*(0)$ since we expect, for a small number of initial states, to find feasible control signals also for active sets that are predicted differently than determined for the test set. Furthermore, due to the non-convexity of problems (4) and (12), different local minima with equal, but also varying active sets may exist for the same

initial state (see [30]).

For two states of the test set, Figure 6 shows the state (top) and input (middle) sequences and the number of iterations performed by `fmincon` to find the input signal for both the regular NMPC scheme (color) and the proposed approach exploiting active set predictions (black). In Figure 6(a), the closed-loop results for initial state $x(0) = (0.3515, -0.4340)^\top$ show no difference in states and inputs, while the number of iterations to solve the NLP using Algorithm 1 is smaller for all time steps. For initial state $x(0) = (0.2335, -0.9187)^\top$, Figure 6(b) depicts the case that the regular NMPC problem had to be solved as a backup in one time step. Thus, the number of iterations for the simplified NLP (12) and the regular NLP (4) are added up in time step $k = 5$.

The results for regulating all 1 000 initial states of the test set into the terminal set \mathcal{T} are summarized in Table 2. In 96.3 % of all cases, the classification network $\Phi(x)$ was able to predict an active set such that the simplified NLP resulted in a feasible solution (column 2). The difference of around 2 % between the accuracy of $\Phi(x)$ and the number of cases with feasible control inputs indicates that in the worst case only a small fraction resulted in a feasible, but no longer optimal control input. If one of these cases resulted in

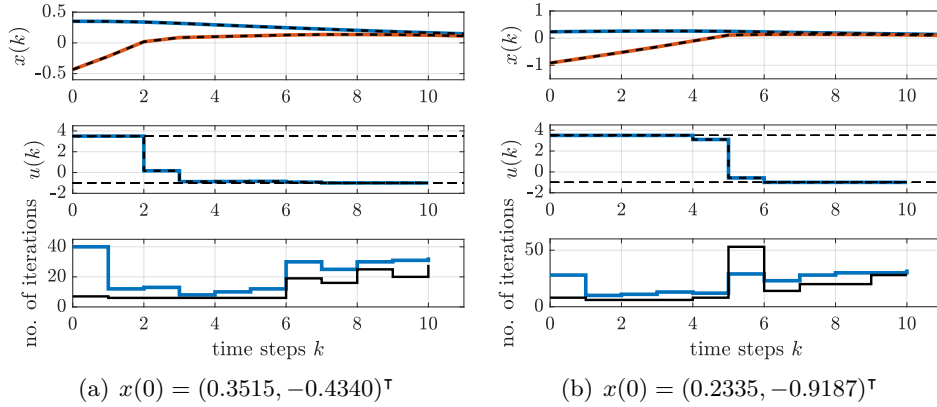


Figure 6: States (top), inputs (middle), and number of iterations (bottom) along time steps k resulting for two different initial states by applying regular NMPC (color) and the proposed simplification approach (black). In (a), all predicted active sets coincided with the actual ones. In (b), the regular NLP had to be solved for $k = 5$ as backup, such that the number of iterations for both solution attempts are added up in this time step.

Table 2: Accuracy of the classification network in a point-wise comparison, amount of feasible simplified NLPs, and number of iterations to determine an input signal for both the regular NMPC scheme and the proposed method.

Accuracy of $\Phi(x)$	Feasible simplified NLPs	Number of iterations		
		Regular NMPC	Algorithm 1	Red. in %
94.11	96.3%	27.33	24.6	9.99

a different local minimum than for the original problem, the number of feasible, but suboptimal solutions further decreases. The mean squared error (MSE) between the cost function values (including $x(0)$) and the applied input signals that result from solving the regular and the simplified NLP read $\text{MSE}_{\hat{v}^*} = 5.72 \cdot 10^{-5}$ and $\text{MSE}_{u^*} = 7.23 \cdot 10^{-6}$, respectively. The values indicate that the suboptimality induced into the closed-loop control is negligible and can hardly be distinguished from numerical sources.

As stated in Section 3.1, all NLPs were solved in Matlab using `fmincon` and the interior-point algorithm, which in general only converges to a bound, i.e., an active constraint. A constraint was declared active if the deviation between the optimization parameter and the corresponding bound fell below a tolerance value of $\epsilon = 10^{-4}$. Thus, the solution to the simplified NLP (12) with equality constraints based on predictions resulting from interior-point solutions may deviate from the solution to the regular NLP (4) within numerical magnitudes, while still describing the same optimal solution.

Columns 3 – 5 show the reduction of computational effort in terms of the number of iterations performed to determine a control input. While on average around 27.33 iterations were necessary for the regular NMPC approach to find an optimal input value $u^*(0)$, Algorithm 1 was able to reduce the number of iterations to determine a control signal by 9.99%, to on average around 24.6 iterations. For time steps in which the regular NLP had to be solved as a fallback (line 7 in Alg. 1), the number of iterations of both solution attempts were added up.

5 Conclusion

We applied machine learning methods for the prediction of active sets in nonlinear model predictive control. We generated training data coping with challenges arising from the non-convexity of the underlying optimization

problem to train classification networks with a precision of more than 94%. The predicted active sets were used to simplify the nonlinear programs solved in every time step. With a CSTR model as example, we could present a reduction of the average number of iterations necessary to find a control input of almost 10%.

Future work will investigate different forms of classification networks, and investigate techniques such as transfer learning to deal with, e.g., changing parameters of the optimization problem.

Acknowledgment

This paper is funded by the Alexander von Humboldt Foundation research group linkage cooperation program and by the European Commission under the grant no. 101079342 (Fostering Opportunities Towards Slovak Excellence in Advanced Control for Smart Industries). KK and MK also gratefully acknowledge the contribution of the Scientific Grant Agency of the Slovak Republic under the grants VEGA 1/0545/20 and the Slovak Research and Development Agency under the project APVV-21-0019.

References

- [1] D. Q. Mayne, J. B. Rawlings, C. V. Rao, P. O. M. Scokaert, Constrained model predictive control: Stability and optimality, *Automatica* 36 (2000) 789–814. doi:10.1016/S0005-1098(99)00214-9.
- [2] S. J. Qin, T. A. Badgwell, An overview of nonlinear model predictive control applications, *Nonlinear model predictive control* (2000) 369–392. doi:10.1007/978-3-0348-8407-5_21.
- [3] T. Faulwasser, L. Grüne, M. A. Müller, Economic nonlinear model predictive control, *Foundations and Trends® in Systems and Control* 5 (1) (2018) 1–98. doi:10.1561/26000000014.
- [4] L. T. Biegler, A perspective on nonlinear model predictive control, *Korean Journal of Chemical Engineering* 38 (7) (2021) 1317–1332. doi:10.1007/s11814-021-0791-7.

- [5] J. B. Rawlings, D. Q. Mayne, M. Diehl, *Model Predictive Control: Theory and Design*, Nob Hill Publishing, 2017.
- [6] M. Diehl, R. Findeisen, F. Allgöwer, H. Bock, J. Schlöder, Nominal stability of real-time iteration scheme for nonlinear model predictive control, *IEE Proceedings-Control Theory and Applications* 152 (3) (2005) 296–308. doi:10.1049/ip-cta:20040008.
- [7] R. Quirynen, K. Berntorp, S. Di Cairano, Embedded optimization algorithms for steering in autonomous vehicles based on nonlinear model predictive control, in: *2018 Annual American Control Conference (ACC)*, 2018, pp. 3251–3256. doi:10.23919/ACC.2018.8431260.
- [8] R. Dyrcka, M. Mönnigmann, Simplified nonlinear programs for NMPC based on active set construction, in: *2022 IEEE 61st Conference on Decision and Control (CDC)*, 2022, pp. 3699–3704. doi:10.1109/CDC51059.2022.9992520.
- [9] M. de Freitas Virgilio Pereira, I. V. Kolmanovsky, C. E. Cesnik, Nonlinear model predictive control with aggregated constraints, *Automatica* 146 (2022) 110649. doi:10.1016/j.automatica.2022.110649.
- [10] T. A. Johansen, Approximate explicit receding horizon control of constrained nonlinear systems, *Automatica* 40 (2004) 293–300. doi:10.1016/j.automatica.2003.09.021.
- [11] M. Mönnigmann, J. Otten, M. Jost, Nonlinear MPC defines implicit regional optimal control laws, *IFAC-PapersOnLine* 48 (23) (2015) 142–147, 5th IFAC Conference on Nonlinear Model Predictive Control NMPC 2015. doi:10.1016/j.ifacol.2015.11.274.
- [12] S. Mate, S. Bhartiya, P. S. V. Nataraj, Multiparametric nonlinear MPC: A region free approach, *IFAC-PapersOnLine* 53 (2) (2020) 11374–11379, 21st IFAC World Congress. doi:10.1016/j.ifacol.2020.12.548.
- [13] A. Domahidi, M. N. Zeilinger, M. Morari, C. N. Jones, Learning a feasible and stabilizing explicit model predictive control law by robust optimization, in: *2011 50th IEEE Conference on Decision and Control and European Control Conference*, 2011, pp. 513–519. doi:10.1109/CDC.2011.6161258.

- [14] J. Drgoňa, D. Picard, M. Kvasnica, L. Helsen, Approximate model predictive building control via machine learning, *Applied Energy* 218 (2018) 199–216. doi:10.1016/j.apenergy.2018.02.156.
- [15] S. Lucia, D. Navarro, B. Karg, H. Sarnago, O. Lucía, Deep learning-based model predictive control for resonant power converters, *IEEE Transactions on Industrial Informatics* 17 (1) (2021) 409–420. doi:10.1109/TII.2020.2969729.
- [16] J. Drgoňa, K. Kiš, A. Tuor, D. Vrabie, M. Klaučo, Differentiable predictive control: Deep learning alternative to explicit model predictive control for unknown nonlinear systems, *Journal of Process Control* 116 (2022) 80–92. doi:10.1016/j.jprocont.2022.06.001.
- [17] S. Leonow, R. Dyrška, M. Mönnigmann, Embedded implementation of a neural network emulating nonlinear MPC in a process control application, in: *2023 European Control Conference (ECC)*, 2023, pp. 1–6. doi:10.23919/ECC57647.2023.10178137.
- [18] Z. Hu, J. Fang, R. Zheng, M. Li, B. Gao, L. Zhang, Efficient model predictive control of boiler coal combustion based on NARX neural network, *Journal of Process Control* 134 (2024) 103158. doi:10.1016/j.jprocont.2023.103158.
- [19] A. S. Ira, I. Shames, C. Manzie, R. Chin, D. Nešić, H. Nakada, T. Sano, A machine learning approach for tuning model predictive controllers, in: *2018 15th International Conference on Control, Automation, Robotics and Vision (ICARCV)*, 2018, pp. 2003–2008. doi:10.1109/ICARCV.2018.8581227.
- [20] L. Hewing, K. P. Wabersich, M. Menner, M. N. Zeilinger, Learning-based model predictive control: Toward safe learning in control, *Annual Review of Control, Robotics, and Autonomous Systems* 3 (2020) 269–296. doi:10.1146/annurev-control-090419-075625.
- [21] C. Shang, F. You, A data-driven robust optimization approach to scenario-based stochastic model predictive control, *Journal of Process Control* 75 (2019) 24–39. doi:10.1016/j.jprocont.2018.12.013.

- [22] S. H. Son, J. W. Kim, T. H. Oh, D. H. Jeong, J. M. Lee, Learning of model-plant mismatch map via neural network modeling and its application to offset-free model predictive control, *Journal of Process Control* 115 (2022) 112–122. doi:10.1016/j.jprocont.2022.04.014.
- [23] R. He, K. Ju, L. Zhao, J. Long, M. Yang, Data-driven worst case model predictive control algorithm for propylene distillation column under uncertainty of top composition, *Journal of Process Control* 124 (2023) 199–213. doi:10.1016/j.jprocont.2023.03.001.
- [24] Y. Vaupel, N. C. Hamacher, A. Caspari, A. Mhamdi, I. G. Kevrekidis, A. Mitsos, Accelerating nonlinear model predictive control through machine learning, *Journal of Process Control* 92 (2020) 261–270. doi:10.1016/j.jprocont.2020.06.012.
- [25] R. R. Hossain, R. Kumar, Machine learning accelerated real-time model predictive control for power systems, *IEEE/CAA Journal of Automatica Sinica* 10 (4) (2023) 916–930. doi:10.1109/JAS.2023.123135.
- [26] M. Klaučo, M. Kalúz, M. Kvasnica, Machine learning-based warm starting of active set methods in embedded model predictive control, *Engineering Applications of Artificial Intelligence* 77 (2019) 1–8. doi:10.1016/j.engappai.2018.09.014.
- [27] R. Dyrska, M. Mönnigmann, Active set prediction for nonlinear model predictive control on a shrinking horizon based on the principle of optimality, *International Journal of Robust and Nonlinear Control* 34 (4) (2024) 2768–2780. doi:10.1002/rnc.7110.
- [28] R. Dyrska, M. Mönnigmann, Corrigendum to “Simplified Nonlinear Programs for NMPC Based on Active Set Construction” [2022 IEEE 61st Conference on Decision and Control (CDC), 2022, pp. 3699–3704], Zenodo. doi:10.5281/zenodo.17347362.
- [29] R. Dyrska, M. Mönnigmann, Correction to “Active Set Prediction for Nonlinear Model Predictive Control on a Shrinking Horizon Based on the Principle of Optimality”, *International Journal of Robust and Nonlinear Control* (2025). doi:10.1002/rnc.70281.

- [30] R. Dyrška, M. Mönnigmann, Properties of nonlinear MPC solutions illustrated with a simple example, IFAC-PapersOnLine 58 (18) (2024) 41–46, 8th IFAC Conference on Nonlinear Model Predictive Control NMPC 2024. doi:10.1016/j.ifacol.2024.09.007.
- [31] G. Pannocchia, J. B. Rawlings, S. T. Wright, Inherently robust sub-optimal nonlinear MPC: Theory and application, in: 2011 50th IEEE Conference on Decision and Control and European Control Conference, 2011, pp. 3398–3403. doi:10.1109/CDC.2011.6161240.
- [32] D. Limon, T. Alamo, E. F. Camacho, Stable constrained MPC without terminal constraint, in: Proc. of American Control Conference, 2003, pp. 4893–4898. doi:10.1109/ACC.2003.1242498.
- [33] S. Okada, M. Ohzeki, S. Taguchi, Efficient partition of integer optimization problems with one-hot encoding, Scientific reports 9 (1) (2019) 13036. doi:10.1038/s41598-019-49539-6.
- [34] J. V. Dillon, I. Langmore, D. Tran, E. Brevdo, S. Vasudevan, D. Moore, B. Patton, A. Alemi, M. Hoffman, R. A. Saurous, Tensorflow distributions, arXiv (2017). doi:10.48550/arXiv.1711.10604.
- [35] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, arXiv (2014). doi:10.48550/arXiv.1412.6980.
- [36] A. Al-Kababji, F. Bensaali, S. P. Dakua, Scheduling techniques for liver segmentation: ReduceLRonPlateau vs OneCycleLR, in: International Conference on Intelligent Systems and Pattern Recognition, Springer, 2022, pp. 204–212. doi:10.1007/978-3-031-08277-1_17.
- [37] J. Nocedal, S. J. Wright, Numerical Optimization, 2nd Edition, Springer New York, 2006. doi:10.1007/978-0-387-40065-5.
- [38] R. Dyrška, M. Horváthová, P. Bakaráč, M. Mönnigmann, J. Oravec, Heat exchanger control using model predictive control with constraint removal, Applied Thermal Engineering 227 (2023) 120366. doi:10.1016/j.applthermaleng.2023.120366.