Learning an Approximate Model Predictive Controller with Guarantees

Michael Hertneck[1], Johannes Köhler[2], Sebastian Trimpe[3], Frank Allgöwer[2]

***Abstract*—A supervised learning framework is proposed to approximate a model predictive controller (MPC) with reduced computational complexity and guarantees on stability and constraint satisfaction. The framework can be used for a wide class of nonlinear systems. Any standard supervised learning technique (e.g. neural networks) can be employed to approximate the MPC from samples. In order to obtain closed-loop guarantees for the learned MPC, a robust MPC design is combined with statistical learning bounds. The MPC design ensures robustness to inaccurate inputs within given bounds, and Hoeffding's Inequality is used to validate that the learned MPC satisfies these bounds with high confidence. The result is a closed-loop statistical guarantee on stability and constraint satisfaction for the learned MPC. The proposed learning-based MPC framework is illustrated on a nonlinear benchmark problem, for which we learn a neural network controller with guarantees.**

***Index Terms*—Predictive control for nonlinear systems; Machine learning; Constrained control**

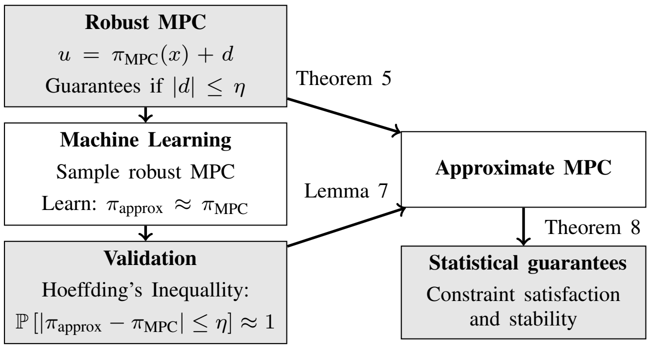

Robust MPC
$u = \pi_{\text{MPC}}(x) + d$
Guarantees if $|d| \leq \eta$
Theorem 5
Machine Learning
Sample robust MPC
Learn: $\pi_{\text{approx}} \approx \pi_{\text{MPC}}$
Lemma 7
Approximate MPC
Theorem 8
Validation
Hoeffding's Inequallity:
$\mathbb{P}\left[|\pi_{\text{approx}} - \pi_{\text{MPC}}| \leq \eta\right] \approx 1$
Statistical guarantees
Constraint satisfaction and stability

Fig. 1. Diagram of the proposed framework. We design an MPC π_{MPC} with robustness to input disturbance d. The resulting feedback law is sampled offline and approximated (π_{approx}) via supervised learning. Hoeffding's Inequality is used for validation and yields a bound on the error between approximate and original MPC in order to guarantee stability and constraint satisfaction. The result is an approximate MPC with statistical guarantees.

I. Introduction

Model predictive control (MPC) [1] is a modern control method based on repeatedly solving an optimization problem online. It can handle general nonlinear dynamics, hard state and input constraints, and general objective functions. One major drawback of MPC is the computational effort of solving optimization problems online under real-time requirements. Especially for settings with a large number of optimization variables or if a high sampling rate is required, the online optimization may get computationally intractable. Hence, it is often desirable to find an explicit formulation of the MPC that can be evaluated online in a short deterministic execution time, also on relatively inexpensive hardware.

For linear systems, the MPC optimization problem can be formulated as a multi-parametric quadratic program, which can be solved offline to obtain an explicit control law [2]. The extension of [2] to nonlinear systems is not straightforward. Hence, the goal of this paper is to develop a framework for approximating a nonlinear MPC through supervised learning with statistical guarantees on stability and constraint satisfaction.

[1]Michael Hertneck is an M.Sc. student at the University of Stuttgart, 70550 Stuttgart, Germany (email: michaelhertneck@yahoo.de).

[2]Johannes Köhler and Frank Allgöwer are with the Institute for Systems Theory and Automatic Control, University of Stuttgart, 70550 Stuttgart, Germany (email: {johannes.koehler, frank.allgower}@ist.uni-stuttgart.de).

[3]Sebastian Trimpe is with the Intelligent Control Systems Group at the Max Planck Institute for Intelligent Systems, 70569 Stuttgart, Germany (email: trimpe@is.mpg.de).

This work was supported in part by the German Research Foundation (DFG) grant GRK 2198/1, the Max Planck Society, and the Cyber Valley Initiative.

A sketch of the main ideas is given in Figure 1. First, a *robust* MPC (RMPC) design is carried out that ensures robustness to bounded input disturbances d with a user defined bound ($|d| \leq \eta$). The resulting RMPC feedback law $\pi_{\text{MPC}}(x)$ is sampled offline for suitable states x_i and approximated via any function approximation (regression) technique such as neural networks (NNs). If the error of the approximate control law $\pi_{\text{approx}}(x)$ is below the admissible bound of the RMPC, recursive feasibility and closed-loop stability for the learned RMPC can be guaranteed. In order to guarantee a sufficiently small approximation error, we use Hoeffding's Inequality on a suitable validation data set. The overall result is an approximate MPC (AMPC) with lower computational requirements and statistical guarantees on closed-loop stability and constraint satisfaction. The proposed approach is applicable to a wide class of nonlinear control problems with state and input constraints.

Contributions: This paper makes the following contributions: We present an MPC design which is robust for a *chosen* bound on the input error. Furthermore, we propose a validation method based on Hoeffding's Inequality to provide a statistical bound on the approximation error of the approximated MPC. By combining these approaches, we obtain a complete framework to learn an AMPC from offline samples in an automatic fashion. The framework is suited for nonlinear systems with constraints, can incorporate a user defined cost function, and supports high sampling rates on cheap hardware. The framework is demonstrated on a numerical benchmark example, where the AMPC is represented by an NN.

Related work: In the literature, there exist several approaches to obtain offline an approximate solution for an MPC. In [3], a semi-explicit MPC for linear systems is presented that ensures stability and constraint satisfaction with a decreased online computational demand, by combining a subspace clustering with a feasibility refinement strategy. In [4], a learning algorithm is presented with additional constraints to guarantee stability and constraint satisfaction of the approximate MPC for linear systems. Both approaches stand in contrast to our approach, which can be used with any learning method and supports nonlinear systems.

One approach to approximate a nonlinear MPC is convex multi-parametric nonlinear programming [5], [6]. Approximating an MPC by NNs, as also done herein, has for example been proposed in [7], [8]. In contrast to the proposed framework, these approaches cannot *guarantee* stability and constraint satisfaction for the resulting AMPC. In [9], [10], it is shown, that guarantees on stability and constraint satisfaction are preserved for arbitrary small approximation errors (due to inherent robustness properties). In [11], an MPC with Lipschitz based constraint tightening is learned that provides guarantees for a non vanishing approximation error. The admissible approximation error deduced in [9], [10], [11] is typically not achievable (compare example) and is thus not suited within proposed framework.

Neural networks have recently been increasingly popular as control policies for complex tasks, [12]. One powerful framework for training (deep) NN policies is guided policy search (GPS) [13], which uses trajectory-centric control to generate samples for training the NN policy via supervised learning. In [14], an MPC is used with GPS for generating the training samples. While that work is conceptually similar, it does not provide closed-loop guarantees for the learned controller as we do herein. The main novelty of the proposed framework lies in the approximation of a *robust* control technique and its combination with a suitable validation approach. The validation method has resemblances to [15], where statistical guarantees for robustness are derived based on the Chernoff bound.

Outline: This paper is structured as follows: We formulate the problem and present our main idea in Section II. In Section III, the RMPC design is presented. In Section IV, we present a validation method and propose a procedure to learn an MPC with guarantees. A numerical example to show the applicability of the proposed framework is given in Section V. Section VI concludes the paper.

Notation: The euclidean norm with respect to a positive definite matrix $Q = Q^\top$ is denoted by $\|x\|_Q^2 = x^\top Q x$. The Pontryagin set difference is defined by $X \ominus Y := \{z \in \mathbb{R}^n : z + y \in X, \forall y \in Y\}$. For X a random variable, $\mathbb{E}(X)$ denotes the expectation of X, and $\mathbb{P}[X \geq x]$ denotes the probability for $X \geq x$. We denote $\mathbf{1}_p = [1,..,1]^\top \in \mathbb{R}^p$. .

II. Main approach

In this section, we pose the control problem and describe the proposed approach.

A. Problem formulation

We consider the following nonlinear discrete-time system

$$x(t+1) = f(x(t), u(t)) \tag{1}$$

with the state $x(t) \in \mathbb{R}^n$, the control input $u(t) \in \mathbb{R}^m$, the time step $t \in \mathbb{N}$, and f continuous with $f(0,0) = 0$. We consider compact polytopic constraints

$$\mathcal{X} = \{x \in \mathbb{R}^n | Hx \leq \mathbf{1}_p\}, \; \mathcal{U} = \{u \in \mathbb{R}^m | Lu \leq \mathbf{1}_q\}.$$

The control objective is to ensure stability of $x = 0$, constraint satisfaction, i.e. $(x(t), u(t)) \in \mathcal{X} \times \mathcal{U} \; \forall t \geq 0$, and optimize some cost function. We shall consider these objectives under initial conditions $x(0) \in \mathcal{X}_{\text{feas}}$, where $\mathcal{X}_{\text{feas}}$ is a feasible set to be made precise later. The resulting controller should be implementable on cheap hardware for systems with high sampling rates.

B. General approach

The controller synthesis with the proposed framework works as follows: An MPC is designed for (1) that is robust to inaccurate inputs u within chosen bounds. The RMPC is sampled offline over the set of feasible states $\mathcal{X}_{\text{feas}}$ and approximated using supervised learning techniques based from these samples. In this paper, we will use NNs to approximate the RMPC, but any other supervised learning technique or regression method can be used likewise. The learning yields an AMPC $\pi_{\text{approx}} : \mathcal{X}_{\text{feas}} \to \mathcal{U} \; | u = \pi_{\text{approx}}(x)$. With this controller, the closed-loop system is given by

$$x(t+1) = f(x(t), \pi_{\text{approx}}(x(t))). \tag{2}$$

Stability of the closed loop (2) is guaranteed if the approximation error is below the admissible bound on the input disturbance from the RMPC. We use a validation method based on Hoeffding's Inequality to guarantee this bound.

III. Input Robust MPC

In this section, we present an input robust MPC design with robust guarantees on stability and constraint satisfaction for bounded additive input disturbances. To achieve the robustness, we combine a standard MPC formulation [16] with a robust constraint tightening [17]. The RMPC optimization problem can be formulated as

$$\min_{u(\cdot|t)} \sum_{k=0}^{N-1} \|x(k|t)\|_Q^2 + \|u(k|t)\|_R^2 + \|x(N|t)\|_P^2 \tag{3a}$$

$$s.t. \; x(0|t) = x(t), \quad x(N|t) \in \mathcal{X}_f, \tag{3b}$$

$$x(k+1|t) = f(x(k|t), u(k|t)), \tag{3c}$$

$$x(k|t) \in \bar{\mathcal{X}}_k, \quad u(k|t) \in \bar{\mathcal{U}}_k, \; k = 0, ..., N-1 \tag{3d}$$

We denote the set of states x where (3) is feasible by $\mathcal{X}_{\text{feas}}$. The solution to the RMPC optimization problem (3) is denoted by $u^*(\cdot|t)$. The MPC feedback law is $\pi_{\text{MPC}}(x(t)) := u^*(0|t)$. The state and input constraints from standard MPC formulations are replaced by tightened constraints $\bar{\mathcal{X}}_k$ and $\bar{\mathcal{U}}_k$. This tightening guarantees stability and constraint satisfaction despite bounded input errors and will be discussed in more detail in the following. The positive definite matrices Q and R are design parameters. The design of the terminal ingredients $\mathcal{X}_f$ and P will be made precise later. The closed-loop of the RMPC under input disturbances is given by

$$x(t+1) = f(x(t), \pi_{\text{MPC}}(x(t)) + d(t)) \tag{4}$$

with $d(t) \in \mathcal{W} = \{d \in \mathbb{R}^m : \|d\|_\infty \leq \eta\}$, $\forall t \geq 0$ for some η. The following assumption will be used in order to design the RMPC:

Assumption 1: (Local incremental stabilizability [17], [18]) There exists a control law $\kappa : \mathcal{X} \times \mathcal{X} \times \mathcal{U} \to \mathbb{R}^m$, a δ-Lyapunov function $V_\delta : \mathcal{X} \times \mathcal{X} \times \mathcal{U} \to \mathbb{R}_{\geq 0}$, that is continuous in the first argument and satisfies $V_\delta(x, x, v) = 0$ $\forall x \in \mathcal{X}$, $\forall v \in \mathcal{U}$, and parameters $c_{\delta,l}$, $c_{\delta,u}$ δ_{loc}, $k_{\max} \in \mathbb{R}_{>0}$, $\rho \in (0, 1)$, such that the following properties hold for all $(x, z, v) \in \mathcal{X} \times \mathcal{X} \times \mathcal{U}$, $(z^+, v^+) \in \mathcal{X} \times \mathcal{U}$ with $V_\delta(x, z, v) \leq \delta_{\text{loc}}$:

$$c_{\delta,l}\|x - z\|^2 \leq V_\delta(x, z, v) \leq c_{\delta,u}\|x - z\|^2, \tag{5}$$

$$\|\kappa(x, z, v) - v\| \leq k_{\max}\|x - z\|, \tag{6}$$

$$V_\delta(x^+, z^+, v^+) \leq \rho V_\delta(x, z, v) \tag{7}$$

with $x^+ = f(x, \kappa(x, z, v))$, $z^+ = f(z, v)$.
This assumption is quite general. Sufficient conditions for this property can be formulated based on the linearization, provided that f is locally Lipschitz, compare [18]. The concept of incremental stability [17] describes an incremental robustness property, and is thus suited for the constraint tightening along the prediction horizon. To overestimate the influence from the system input on the system state, we use the following assumption:

Assumption 2: (Local Lipschitz continuity) There exists a $\lambda \in \mathbb{R}$, such that $\forall x \in \mathcal{X}, \forall u \in \mathcal{U}, \forall u + d \in \mathcal{U}$

$$\|f(x, u + d) - f(x, u)\| \leq \lambda\|d\|_\infty. \tag{8}$$

With this assumption, we can introduce a bound on the admissible input disturbance which will be used in the proof of robust stability and recursive feasibility for the RMPC:

Assumption 3: (Bound on the input disturbance) The input disturbance bound satisfies $\eta \leq \frac{1}{\lambda}\sqrt{\frac{\delta_{\text{loc}}}{c_{\delta,u}}}$.
To guarantee robust constraint satisfaction we use a growing tube inspired constraint tightening, based on the incremental stabilizability property in Assumption 1 as in [17]. Consider the polytopic tightened set $\mathcal{U}_t = \mathcal{U} \ominus \mathcal{W} = \{u \in \mathbb{R}^m : L_t u \leq \mathbf{1}_p\}$. This set ensures that $u + d \in \mathcal{U}$, $\forall u \in \mathcal{U}_t$, $\forall d \in \mathcal{W}$. We set

$$\epsilon := \eta\lambda\sqrt{\frac{c_{\delta,u}}{c_{\delta,l}}} \max\{\|H\|_\infty, \|L_t\|_\infty k_{\max}\}. \tag{9}$$

The constraint tightening is achieved with a scalar tightening parameter $\epsilon_k := \epsilon\frac{1-\sqrt{\rho}^k}{1-\sqrt{\rho}}$, $k \in \{0, ..., N\}$ based on the asymptotic decay rate ρ and ϵ. The tightened constraint sets are given by

$$\bar{\mathcal{X}}_k := (1 - \epsilon_k)\mathcal{X} = \{x \in \mathbb{R}^n : Hx \leq (1 - \epsilon_k)\mathbf{1}_p\},$$

$$\bar{\mathcal{U}}_k := (1 - \epsilon_k)\mathcal{U}_t = \{u \in \mathbb{R}^m : L_t u \leq (1 - \epsilon_k)\mathbf{1}_q\}.$$

This constraint tightening can be thought of as an over approximation of the constraint tightening used in [19], with the difference that it can be easily applied to nonlinear systems, compare [17]. Clearly, the size of the tightened constraints depends on ϵ and thus on the size of the bound on the input disturbance η. The maximum influence of the input disturbance d on the predicted state $x(N|t)$ is bounded by $\mathcal{W}_N = \left\{x \in \mathbb{R}^n : \|x\| \leq \lambda\eta\sqrt{\rho^N \frac{c_{\delta,u}}{c_{\delta,l}}}\right\}$. We use the following Assumption on the terminal set to guarantee recursive feasibility and closed-loop stability similar to [16]:

Assumption 4: (Terminal set) There exists a local control Lyapunov function $V_f(x) = \|x\|_P^2$, a terminal set $\mathcal{X}_f = \{x : V_f(x) \leq \alpha_f\}$ and a control law $k_f(x)$, such that $\forall x \in \mathcal{X}_f$:

$$f(x, k_f(x)) + w \in \mathcal{X}_f, \ \forall w \in \mathcal{W}_N, \tag{10}$$

$$V_f(f(x, k_f(x))) \leq V_f(x) - (\|x\|_Q^2 + \|k_f(x)\|_R^2), \tag{11}$$

$$(x, k_f(x)) \subseteq (\bar{\mathcal{X}}_N \times \bar{\mathcal{U}}_N). \tag{12}$$

It is always possible to design P, $\mathcal{X}_f$ and k_f to satisfy Assumption 4, if the linearization of the system is stabilizable, $(0, 0)$ lies in the interior of $\mathcal{X} \times \mathcal{U}$ and η is small enough. Now, we are ready to state a theorem that guarantees stability and constraint satisfaction for the RMPC despite bounded input disturbances:

Theorem 5: Let Assumption 1, 2, 3 and 4 hold. Then, for all initial conditions $x(0) \in \mathcal{X}_{\text{feas}}$, the RMPC closed-loop (4) satisfies $(x(t), u(t)) \in \mathcal{X}_{\text{feas}} \times \mathcal{U}$ $\forall t \geq 0$. Furthermore, the closed-loop system (4) converges to a robust positive invariant set $\mathcal{Z}_{\text{RPI}}$ around the origin.

Proof: The proof is a straightforward adaption of [17], for details see [20]. ∎

Remark 6: The size of $\mathcal{Z}_{\text{RPI}}$ depends on the input error bound η. If η is chosen small enough, $\mathcal{Z}_{\text{RPI}} \subseteq \mathcal{X}_f$ and asymptotic stability of the origin can be guaranteed by applying the terminal controller in the terminal set.

IV. Learning the RMPC

In this section, we discuss how the RMPC can be approximated with supervised learning methods. We also present a method for the validation of π_{approx} based on Hoeffding's Inequality to obtain guarantees on stability and constraint satisfaction.

A. Supervised learning

To learn the RMPC, we generate an arbitrary number of samples $(x, \pi_{\text{MPC}}(x)) \in \mathcal{X}_{\text{feas}} \times \mathcal{U}$. Supervised learning methods can be used to obtain the approximation π_{approx} of π_{MPC}. To preserve guarantees from the RMPC under the approximation, we will show that

$$\|\pi_{\text{approx}}(x) - \pi_{\text{MPC}}(x)\|_\infty \leq \eta \tag{13}$$

holds for all relevant states with the chosen η from Assumption 3. An approximation with a sufficient small approximation error is possible with state-of-the-art machine learning techniques; a comprehensive overview of possible methods is given in e.g. [21]. Equation (13) implies $\pi_{\text{approx}}(x) =$

$\pi_{\text{MPC}}(x) + d$ with $\|d\|_\infty \leq \eta$. Thus, Theorem 5 guarantees stability and constraint satisfaction for π_{approx} if (13) holds.

Neural networks (NNs) are a popular and powerful method for supervised learning; see standard literature such as [21], [22]. Specifically, NNs have successfully been employed as control policies (see 'Related work'). While we also consider NNs in the example in Section V, the proposed approach equally applies to any regression or function approximation technique such as [9], [10].

It is well known that relevant classes of NNs are universal approximators; that is, they can *in principle*[1] approximate any sufficiently regular function to arbitrary accuracy provided that the network has sufficiently many hidden units, [22]. However, it is in general difficult to provide an *a-priori* guarantee that a learned NN satisfies the desired bound on the approximation error η. To overcome this problem, we propose a validation method based on statistical learning bounds in the next subsection.

B. Probabilistic guarantees

In this subsection, we propose a probabilistic method to validate an approximator π_{approx}. For the validation, we will consider trajectories of the system (2), which is controlled by the approximate MPC. We introduce

$$X_i := \{x(t), t \in \{0, \dots, T_i\} \,:\, x(0) = x_i \in \mathcal{X}_{\text{feas}}, \qquad (14)$$
$$x(T_i) \in \mathcal{X}_f \text{ and } x(t+1) = f(x(t), \pi_{\text{approx}}(x(t)))\}$$

to denote a trajectory of (2) starting at $x(0) = x_i \in \mathcal{X}_{\text{feas}}$ and ending in $\mathcal{X}_f$, where we can guarantee stability and constraint satisfaction with the terminal controller. Then, let

$$I(X_i) := \begin{cases} 1 & \text{if } \|\pi_{\text{MPC}}(x) - \pi_{\text{approx}}(x)\|_\infty \leq \eta,\ \forall x \in X_i \\ 0 & \text{otherwise} \end{cases}$$

be an indicator function, which indicates whether a learned control law π_{approx} satisfies the posed accuracy η along a trajectory until the terminal set is reached.

For the validation, we consider p trajectories X_j, $j = 1, ..., p$ with initial conditions $x(0)$ independently sampled from some distribution Ω over $\mathcal{X}_{\text{feas}}$. Because the initial conditions are independent, identically distributed (iid), also X_j and thus $I(X_j)$ are iid. Next, we state a statistical bound for the approximation accuracy of π_{approx} along iid trajectories (14).

Define the empirical risk as

$$\tilde{\mu} := \frac{1}{p}\sum_{j=1}^{p} I(X_j). \qquad (15)$$

The RMPC guarantees stability and constraint satisfaction if

$$I(X_i) = 1,\ \forall X_i \text{ with } x(0) = x_i \in \mathcal{X}_{\text{feas}} \qquad (16)$$

holds. The probability for $I(X_i) = 1$ is $\mu := \mathbb{P}[I(X_i) = 1]$ for X_i with iid initial condition $x_j(0) \in \mathcal{X}_{\text{feas}}$ from the distribution Ω. Thus, μ is a lower bound for the probability of stability and constraint satisfaction. We can use Hoeffding's Inequality to estimate μ from the empirical risk $\tilde{\mu}$:

Lemma 7: (Hoeffding's Inequality [23, pp. 667-669]) Let $I(X_j)$ $j = 1, ..., p$ be p iid random variables with $0 \leq I(x_j) \leq 1$. Then,

$$\mathbb{P}\left[|\tilde{\mu} - \mu| \geq \varepsilon_h\right] \leq 2\exp\left(-2p\varepsilon_h^2\right). \qquad (17)$$

Denote $\delta_h := 2\exp\left(-2p\varepsilon_h^2\right)$ as the confidence level. Then (17) implies that with confidence of at least $1 - \delta_h$,

$$\mathbb{P}[I(X_i) = 1] = \mu \geq \tilde{\mu} - \varepsilon_h. \qquad (18)$$

Hence, with confidence $1-\delta_h$, the probability that the approximation error is below the chosen bound η along a trajectory with a random initial condition from Ω is larger than $\tilde{\mu} - \varepsilon_h$. We can use this to establish a validation method to guarantee a chosen bound $\mu_{\text{crit}} \leq \mathbb{P}\left[I(X_i) = 1\right]$ and a chosen confidence δ_h. If, for a number of samples p, the empirical risk $\tilde{\mu}$ satisfies

$$\mu_{\text{crit}} \leq \tilde{\mu} - \varepsilon_h = \tilde{\mu} - \sqrt{-\frac{\ln\left(\frac{\delta_h}{2}\right)}{2p}}, \qquad (19)$$

we can rewrite (18) as $\mathbb{P}\left[I(X_i) = 1\right] \geq \mu_{\text{crit}}$, which holds at least with confidence level $1 - \delta_h$. We use this for validation as follows: for chosen desired confidence δ_h and μ_{crit}, we compute $\tilde{\mu}$ and ε_h for a given number p of samples. If (19) holds for this p, the validation is successful. If (19) does not hold for this p, the number of samples for the validation p is increased, which decreases ε_h. The validation is then repeated iteratively while increasing p. We say the validation is failed, if p exceeds a maximum number of samples $p_{\max}$ and stop the validation. Then the learning has to be repeated and improved (to increase $\tilde{\mu}$). This validation method is independent of the chosen learning method.

Given the proposed validation method, the overall procedure for the computation of an AMPC is summarized in Algorithm 1. The following theorem ensures stability and constraint satisfaction of the resulting learned AMPC.

Algorithm 1 Learn approximate control law

```
1) Show incremental stabilizability. Compute $\lambda$
   (Assumption 1, 2).
2) Choose an accuracy $\eta$ (Assumption 3).
3) Design the RMPC:
   a) Set $\epsilon$ according to (9).
   b) Compute terminal ingredients (Assumption 4).
   c) Check whether (10) from Assumption 4 holds.
      If not, decrease $\eta$.
4) Learn $\pi_{\text{approx}} \approx \pi_{\text{MPC}}$, e.g. with a NN.
5) Validate $\pi_{\text{approx}}$ according to Lemma 7.
6) If the validation fails, repeat the learning from
   step 4).
```

Theorem 8: Let Assumptions 1, 2, 3 and 4 hold. Suppose that Algorithm 1 is used with suitable chosen $\eta, \mu_{\text{crit}}, \delta_h$ and $p_{\max}$ and with an approximation method that can achieve an

[1]It can still be challenging to actually train an NN to desired accuracy in practice, e.g., because the number of required hidden units is unknown, and typically only local optima are found during training.

approximation error smaller than the required bound η. Then Algorithm 1 terminates. With a confidence level of $1-\delta_h$, the resulting approximate MPC ensures closed-loop stability and constraint satisfaction for a fraction of μ_{crit} of the random initial conditions distributed according to Ω.

Proof: Because Algorithm 1 terminates, (19) holds. Lemma 7 implies (18). We thus have $\mathbb{P}\left[I(X_i)=1\right] \geq \mu_{\text{crit}}$ with confidence at least $1-\delta_h$. That is, with confidence $1-\delta_h$, for at least a fraction of μ_{crit} trajectories X_i, we have $I(X_i)=1$, which implies stability and constraint satisfaction by Theorem 5. ■

V. Example

In this section, we demonstrate the AMPC scheme with a numerical example. We go step by step through Algorithm 1 and compare the resulting controller to a discrete-time Linear Quadratic Regulator (LQR). We consider a discrete-time version of a continuous stirred tank reactor with

$$f(\tilde{x},\tilde{u}) = \begin{pmatrix} \tilde{x}_1 + h\left(\frac{(1-\tilde{x}_1)}{\theta} - k\tilde{x}_1 e^{-\frac{M}{\tilde{x}_2}}\right) \\ \tilde{x}_2 + h\left(\frac{x_f-\tilde{x}_2}{\theta} + k\tilde{x}_1 e^{-\frac{M}{\tilde{x}_2}} - \alpha\tilde{u}(\tilde{x}_2 - x_c)\right)\end{pmatrix},$$

where $\tilde{x}_1$ is the temperature, $\tilde{x}_2$ is the concentration, and $\tilde{u}$ is the coolant flow. This system is a common benchmark taken from[2] [24]. We consider stabilization of the unstable steady state $x_e = (0.2632, 0.6519)$ with steady-state input $u_e = 0.7853$. To achieve $f(0,0) = 0$, we use the transformed coordinates $x = \tilde{x} - x_e$ and $u = \tilde{u} - u_e$. We consider the constraint sets $\mathcal{X} = [-0.2, 0.2] \times [-0.2, 0.2]$, $\mathcal{U} = [0 - u_e, 2 - u_e]$, the stage cost $Q = I, R = 10^{-4}$ and use $N = 180$ for the prediction horizon.

Step 1: Incremental stabilizability and Lipschitz constant: The verification of the local incremental stability assumption can be done according to [18] based on a griding of the constraint set. According to this, Assumption 1 holds with[3] $c_{\delta_l} = 12.33$, $c_d = 199.03$, $k_{\max} = 45.72$, $\rho = 0.9913$, $\delta_{\text{loc}} = 0.01$. The Lipschitz constant is $\lambda = 5.5 \cdot 10^{-3}$.

Step 2: Set Accuracy: For the approximation accuracy η, we choose $\eta = 5.1 \cdot 10^{-3}$, which satisfies Assumption 3.

Step 3: RMPC design: Instead of (9), we use the following less conservative bound

$$\epsilon = \eta\lambda\sqrt{c_{\delta,u}} \max\left\{\tilde{k}_{\max}\|L_t\|_\infty, \frac{\|H\|_\infty}{\sqrt{c_{\delta,l}}}\right\} = 2.2 \cdot 10^{-3}$$

with $\tilde{k}_{\max} = \|P_r^{-1/2}K_r^\top\|$ (for details see [20]). Given the stage cost Q and R, we compute the terminal cost P, the terminal controller $k_f(x)$ and the terminal region $\mathcal{X}_f$ based on the LQR[4], compare [16].

[2]The parameters are $\theta = 20$, $k = 300$, $M = 5$, $x_f = 0.3947$, $x_c = 0.3816$, $\alpha = 0.117$, $h = 0.1$. This corresponds to an Euler discretization with the sampling time $h = 0.1s$ for the system from [24] .

[3]We use $V_\delta(x,z) = \|x-z\|^2_{P_r}$ and $\kappa(x,z,v) = v + K_r(x-z)$ with matrices P_r and K_r continuous valued in $r = (x,u) \in \mathcal{X} \times \mathcal{U}$. For details on the computation and for numerical values of P_r and K_r see [18], [20].

[4]The terminal ingredients are $k_f(x) = -46.01x_1 + 101.74x_2$,

$$P = \begin{pmatrix} 33.21 & -3.61 \\ -3.61 & 6.65\end{pmatrix}, \ \alpha_f = 9.2 \cdot 10^{-5}.$$

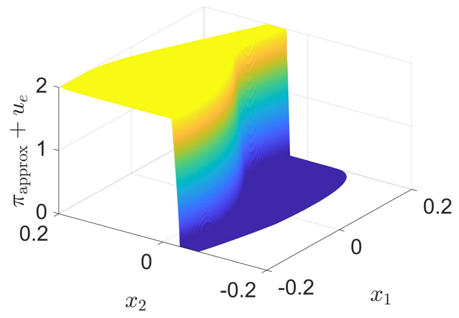


Fig. 2. Approximate MPC π_{approx} over the feasible set $\mathcal{X}_{\text{feas}}$.

Step 4: Learning: A fully connected feedforward NN with two neurons in the first hidden layer and two further hidden layers with 50 neurons each is trained as the AMPC. The activations in the hidden layers are hyperbolic tangent sigmoid functions. In the output layer, linear activations are used. To create the learning samples, the RMPC is sampled over $\mathcal{X}$ with a uniform grid using Casadi 2.4.2 [25] with Python 2.7.2 with grid size of $2.5 \cdot 10^{-4}$. Therewith, $1.6 \cdot 10^6$ feasible samples of data points $(x, \pi_{\text{MPC}}(x))$ are generated for learning. The NN is initialized randomly and trained with the Levenberg Marquardt algorithm using Matlab's neural network toolbox on R2017a. The resulting AMPC over $\mathcal{X}_{\text{feas}}$ is shown in Figure 2.

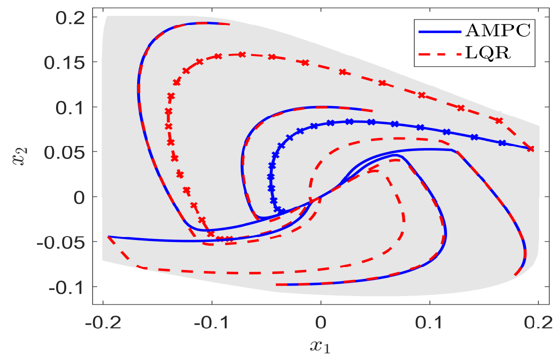
Step 5: Validation: We choose $\delta_h = 0.01$ and $\mu_{\text{crit}} = 0.99$. As validation data, we sample initial conditions uniformly from $\mathcal{X}_{\text{feas}}$. Algorithm 1 terminates, with $\tilde{\mu} = 0.9987$ for $p = 34980$ trajectories. This implies $\mu_{\text{crit}} < \tilde{\mu} - \varepsilon_h$ and hence, we achieve the desired guarantees. We thus conclude from Theorem 8 that, with a confidence of 99%, the closed-loop system (2) is stable and satisfies constraints with a probability of at least 99%.

The overall time to execute Algorithm 1 was roughly 500 hours on a Quad-Core PC. It is possible to significantly reduce this time, e.g., by parallelizing the sampling and the validation.

Simulation results: We compare the performance of the AMPC to a standard LQR based on the linearization around the setpoint, the weights Q, R, and in addition with u saturated in $\mathcal{U}$.[5] Contrary to the AMPC, the LQR does not guarantee stability and constraint satisfaction for general nonlinear constrained systems. The convergence of both controllers is illustrated in Figure 3 (top) for some exemplary initial conditions.

While for some initial conditions, the AMPC and LQR trajectories are close (when nonlinearities play a subordinate role), there are significant differences for others. For example, in Figure 3 (bottom), the input flow is shown for the marked trajectory (top). Initially, the opposite input constraint is active for the LQR in comparison to the AMPC. This causes an initial divergence of the LQR trajectory leading to over three times higher costs. Due to the small approximation error, the

[5]We have chosen Q and R such that the LQR shows good performance for a fair comparison. However, for some other choices (e.g. Q = I, R = 5), the LQR does not stabilize the system, while the AMPC does.



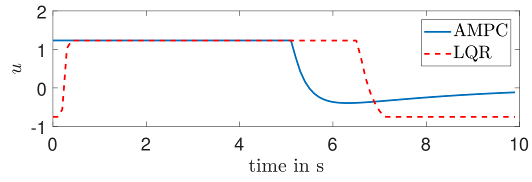


Fig. 3. Top: concentration (x_1) vs. temperature (x_2): trajectories of AMPC (solid blue) and LQR (dashed red) and $\mathcal{X}_{\text{feas}}$ (grey area). One exemplary pair of trajectories is marked with crosses. Bottom: Coolant flow for the marked trajectory pair.

trajectories of the AMPC are virtually indistinguishable from the original RMPC.

Remark 9: In principle, the RMPC method from [11] could also be used within the framework. However, the admissible approximation error for this method would be at most $\eta = 10^{-41}$, which makes the learning intractable.

To investigate the online computational demand of the AMPC, we evaluated the AMPC and the online solution of the RMPC optimization problem for 100 random points over $\mathcal{X}_{\text{feas}}$. The evaluation of the RMPC with Casadi took 0.71 s on average, while the NN could be evaluated in Matlab in 3 ms. This is already over 200 times faster using a straightforward NN implementation in Matlab, and further speed-up may be obtained by alternative NN implementations.

Overall, the AMPC delivers a cost optimized, stabilizing feedback law for a nonlinear constrained system, which can be evaluated online on a cheap hardware with a high sampling rate.

VI. Conclusion

We proposed an RMPC scheme that guarantees stability and constraint satisfaction despite approximation errors. Stability and constraint satisfaction can thus be ensured if the approximate input is within user-defined bounds. Furthermore, we proposed a probabilistic method to validate the approximated MPC. By using statistical methods, we avoid conservatism, analyze complex controller structures and automate the controller synthesis with few design variables.

Tailored learning algorithms and more general verification with different indicator functions, e.g., for the application to higher dimensional systems, are part of future work.

Acknowledgment

The authors thank M. Wuethrich for insightful discussions.

References

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

[2] A. Bemporad, M. Morari, V. Dua, and E. N. Pistikopoulos, “The explicit linear quadratic regulator for constrained systems,” *Automatica*, vol. 38, no. 1, pp. 3–20, 2002.

[3] G. Goebel and F. Allgöwer, “Semi-explicit MPC based on subspace clustering,” *Automatica*, vol. 83, pp. 309–316, 2017.

[4] A. Domahidi, M. N. Zeilinger, M. Morari, and C. N. Jones, “Learning a feasible and stabilizing explicit model predictive control law by robust optimization,” in *50th IEEE Conference on Decision and Control and European Control Conference (CDC-ECC)*, 2011, pp. 513–519.

[5] T. A. Johansen, “Approximate explicit receding horizon control of constrained nonlinear systems,” *Automatica*, vol. 40, no. 2, pp. 293–300, 2004.

[6] A. Grancharova and T. A. Johansen, “Computation, approximation and stability of explicit feedback min–max nonlinear model predictive control,” *Automatica*, vol. 45, no. 5, pp. 1134–1143, 2009.

[7] T. Parisini, M. Sanguineti, and R. Zoppoli, “Nonlinear stabilization by receding-horizon neural regulators,” *International Journal of Control*, vol. 70, no. 3, pp. 341–362, 1998.

[8] B. M. Åkesson and H. T. Toivonen, “A neural network model predictive controller,” *Journal of Process Control*, vol. 16, no. 9, pp. 937–946, 2006.

[9] A. Chakrabarty, V. Dinh, M. J. Corless, A. E. Rundell, S. H. Żak, and G. T. Buzzard, “Support vector machine informed explicit nonlinear model predictive control using low-discrepancy sequences,” *IEEE Transactions on Automatic Control*, vol. 62, no. 1, pp. 135–148, 2017.

[10] M. Canale, L. Fagiano, and M. Milanese, “Fast nonlinear model predictive control via set membership approximation: an overview,” in *Nonlinear Model Predictive Control*. Springer, 2009, pp. 461–470.

[11] G. Pin, M. Filippo, F. A. Pellegrino, G. Fenu, and T. Parisini, “Approximate model predictive control laws for constrained nonlinear discrete-time systems: analysis and offline design,” *International Journal of Control*, vol. 86, no. 5, pp. 804–820, 2013.

[12] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, “A brief survey of deep reinforcement learning,” *arXiv preprint arXiv:1708.05866*, 2017.

[13] S. Levine, C. Finn, T. Darrell, and P. Abbeel, “End-to-end training of deep visuomotor policies,” *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 1334–1373, 2016.

[14] T. Zhang, G. Kahn, S. Levine, and P. Abbeel, “Learning deep control policies for autonomous aerial vehicles with MPC-guided policy search,” in *IEEE International Conference on Robotics and Automation (ICRA)*, 2016, pp. 528–535.

[15] R. Tempo, E.-W. Bai, and F. Dabbene, “Probabilistic robustness analysis: Explicit bounds for the minimum number of samples,” *Systems & Control Letters*, vol. 30, no. 5, pp. 237–242, 1997.

[16] H. Chen and F. Allgöwer, “A quasi-infinite horizon nonlinear model predictive control scheme with guaranteed stability,” *Automatica*, vol. 34, no. 10, pp. 1205–1217, 1998.

[17] J. Köhler, M. A. Müller, and F. Allgöwer, “A novel constraint tightening approach for nonlinear robust model predictive control,” *American Control Conference (ACC)*, 2018, to appear.

[18] ——, “Nonlinear reference tracking: An economic model predictive control perspective,” *IEEE Transaction on Automatic Control*, 2018, to appear.

[19] L. Chisci, J. A. Rossiter, and G. Zappa, “Systems with persistent disturbances: predictive control with restricted constraints,” *Automatica*, vol. 37, no. 7, pp. 1019–1028, 2001.

[20] M. Hertneck, “Learning an Approximate Model Predictive Controller with Guarantees,” Studentthesis, University of Stuttgart, 2018. [Online]. Available: http://dx.doi.org/10.18419/opus-9722

[21] S. Shalev-Shwartz and S. Ben-David, *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.

[22] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT Press Cambridge, 2016.

[23] U. Von Luxburg and B. Schölkopf, “Statistical learning theory: Models, concepts, and results,” in *Handbook of the History of Logic*. Elsevier, 2011, vol. 10, pp. 651–706.

[24] D. Q. Mayne, E. C. Kerrigan, E. Van Wyk, and P. Falugi, “Tube-based robust nonlinear model predictive control,” *International Journal of Robust and Nonlinear Control*, vol. 21, no. 11, pp. 1341–1353, 2011.

[25] J. Andersson, “A General-Purpose Software Framework for Dynamic Optimization,” PhD thesis, Arenberg Doctoral School, KU Leuven, Belgium, 2013.