Preprints of the 19th World Congress
The International Federation of Automatic Control
Cape Town, South Africa. August 24-29, 2014

State dependent parametrizations for nonlinear MPC

Gregor Goebel, **Frank Allgöwer**

Institute for Systems Theory and Automatic Control, University of Stuttgart, Pfaffenwaldring 9, 70550 Stuttgart, Germany.

Abstract: This paper aims at reducing the computational load caused by the online optimization in nonlinear model predictive control (MPC) by introducing a new type of parametrizations for the predicted input trajectory. In order to determine offline the state dependent parametrizations, a tailored data mining algorithm is introduced. Refinements to achieve feasibility of the parametrized constrained optimization problem are presented. Theoretical guarantees on constraint satisfaction and feasibility employing the parametrizations are provided for Lipschitz continuous systems. In a numerical example the benefits of the new method are illustrated.

1. INTRODUCTION

Model predictive control (MPC) is a modern optimization based control strategy. It consists of online repeatedly computing an optimal predicted input trajectory over a finite horizon and applying the first part of it to the plant to be controlled. Its two most important advantages are the possibility to take constraints on states and controlled inputs directly into account and that control goals can be incorporated in form of a performance criterion. On the downside of MPC is the high computational load caused by the optimization that has to be carried out repeatedly online. This is even more significant if the system to be controlled is nonlinear. One way of reducing this computational load is to reduce the number of optimization variables. As stated by Qin and Badgwell [2003], so called move blocking is common in practical applications of MPC. Move blocking follows the strategy to reduce the number of free variables by introducing a simple parametrization for the predicted input trajectory keeping it constant over some time steps. Yet, this way generally only suboptimal solutions can be obtained on a shrinked feasible set. A different strategy to reduce the online compuational load is to move some of the computational effort offline. State dependent solutions of the optimization problem are determined offline globally and approximations thereof are made available online. For example, Schulze Darup and Mönnigmann [2012] present a method to partition the state space into a number of polytopes such that for each of the polytopes the optimal predicted input trajectory can be appxroximated by an affine function. Online only the polytope containing the current system state has to be identified and the corresponding affine function has to be evaluated but no optimization is necessary. This method is called explicit nonlinear MPC. It requires to store and process large amounts of data.

In this paper a result combining ideas from both paradigms is presented. Piecewise affine state dependent parametrizations of the predicted input trajectory are determined offline such that they can be used online in an MPC scheme. In comparison to explicit MPC, this way the storage requirements are reduced. On the other hand, in comparison to move blocking MPC, conservatism is reduced and the feasible region is enlarged. This is made possible by storing and exploiting some state dependent knowledge on the optimal solutions via the parametrizations. The type of parametrizations to be used has been introduced in our previous papers for the linear case, Goebel and Allgöwer [2013, 2014]. Here the results are extended to nonlinear systems. As the suggested MPC scheme combines characteristics of explicit MPC and conventional online optimization based MPC, we call it *semi-explicit nonlinear MPC.*

In order to determine the parametrizations, first, optimal input trajectories for an appropriately chosen set of system states are computed. These trajectories are then fed into a tailored data mining algorithm which returns preliminary parametrizations. In an iterative procedure, the parametrizations can be refined to satisfy the constraints of the MPC setup. Included in this paper are theoretic guarantees in terms of constraint satisfaction and feasibility applying the parametrizations for Lipschitz continuous systems and a numerical example to illustrate the virtue of the results.

The remainder of this paper is organized as follows. First, the MPC scheme considered and the parametrizations to be used are introduced and a problem formulation is derived in Section 2. Section 3 addresses the parametrizations including determination via a data mining algorithm, refinements for feasibility and theoretical guarantees. Online application of the parametrizations is briefly discussed in Section 4. Finally, a numerical example is considered in Section 5 and conclusions are given in Section 6.

2. PRELIMINARIES AND PROBLEM FORMULATION

In this section, first the considered MPC scheme and the corresponding optimization problem are introduced. Then the proposed parametrizations are presented and a problem formulation for this paper is derived.

2.1 The considered MPC scheme

Throughout this paper, a nonlinear, time-invariant, discrete-time system with constrained state $x_k \in \mathcal{X} \subset \mathbb{R}^n$ and constrained input $u_k \in \mathcal{U} \subset \mathbb{R}^m$, given by the equation

$$x_{k+1} = f(x_k, u_k),\ k = 0, 1, 2 \dots \tag{1}$$

is considered. Without loss of generality, let $f(0,0) = 0$, i.e., the origin is a steady state of system (1). The problem of driving the system state from an initial condition x_0 in a set $\mathcal{X}_f \subseteq \mathcal{X}$ to the origin subject to a certain performance criterion is addressed. For this purpose an asymptotically stabilizing MPC scheme is to be used. At the core of the considered MPC scheme is the optimization problem

$$\mathbf{P1}(x_0): \quad \min_{U=(u_0^T,\dots,u_{N-1}^T)^T} J(x_0, U)$$

with

$$J(x_0, U) = \sum_{k=0}^{N-1} l(x_k, u_k) + F(x_N),$$

$$\text{s.t. } x_{k+1} = f(x_k, u_k), \tag{2a}$$

$$x_k \in \mathcal{X},\ u_k \in \mathcal{U}\ \forall k = 1, ..., N-1, \tag{2b}$$

$$u_0 \in \mathcal{U},\ x_N \in \mathcal{X}_N. \tag{2c}$$

Therein, $x_k,\ k = 1, \dots, N$ are predicted states, $u_k,\ k = 0, \dots, N-1$ are predicted inputs and N is the prediction horizon. $l(\cdot,\cdot)$ is a positive definite function with $l(0,0) = 0$ which makes up the stage cost. $F(\cdot)$ is called terminal cost and it accounts for the cost-to-go from the terminal state x_N on. The terminal state is restricted to lie in the terminal set $\mathcal{X}_N$. As is well known, a typical MPC algorithm consists of solving $\mathbf{P1}(x_0)$ for the current state x_0, applying the first part u_0 of the optimal predicted input trajectory U^* to the system and then repeating the procedure for the new state over a shifted horizon.

Throughout this paper, the following assumption is used.

Assumption 1. (1) $\mathcal{X}_N \subset \mathcal{X}$, $\mathcal{X}_N$ is closed, $0 \in \mathcal{X}_N$.
(2) There is a terminal control law $\kappa(x)$ defined on $\mathcal{X}_N$ such that $\kappa(x) \in \mathcal{U}$ for $x \in \mathcal{X}_N$ and
(3) $f(x, \kappa(x)) \in \mathcal{X}_N$ for all $x \in \mathcal{X}_N$.
(4) $F(f(x, \kappa(x))) - F(x) + l(x, \kappa(x)) \leq 0$ for all $x \in \mathcal{X}_N$.

These assumptions are typically used to guarantee asymptotic stability of the MPC scheme, see the paper by Mayne et al. [2000]. A specific choice of the parameters has been suggested by Chen and Allgöwer [1998] for example.

2.2 Parametrizations of the input trajectory

The computational load of the described MPC scheme is mainly caused by solving the nonlinear optimization problem **P1** at each time step. On the one hand, the complexity of this optimization is determined by its structure imposed by fixed quantities like the cost function, the system dynamics and the constraints. On the other hand, the computational demand increases with the number of optimization variables. Optimization problem **P1** contains mN optimization variables. This paper aims at reducing the computational load of the optimization by reducing the number of optimization variables. To this end, the predicted input trajectory U in $\mathbf{P1}(x_0)$ is to be parametrized using a lower number $q < mN$ of parameters and optimization is to be carried out over these parameters.

We suggest to use for each system state one out of a predefined number K of different parametrizations. Each of the parametrizations consists of a linear map of the parameters and the system state to the input trajectories. In more detail, the parametrizations are given by

$$p(x, \tilde{U}) = M_i \tilde{U} + \mathcal{K}_i x, \text{ for } x \in \mathcal{S}_i,\ i \in \{1, ..., K\}. \tag{3}$$

Therein, we call $M_i \in \mathbb{R}^{mN \times q}$ the parametrization matrix and $\mathcal{K}_i \in \mathbb{R}^{mN \times n}$ the feedback matrix as it creates a feedback of the state to the predicted input trajectory. $\mathcal{S}_i \subset \mathbb{R}^n,\ i = 1, \dots, K$ are segments in the state space determining which one of the matrices $M_i, \mathcal{K}_i$ to apply. $\tilde{U}$ will be used as new optimization variable. Using these parametrizations, the following simplified optimization problem can be formulated and used in the MPC scheme

$$\mathbf{P2}(x_0): \quad \min_{\tilde{U}} J\left(x_0, p(x_0, \tilde{U})\right)$$
$$\text{s.t. (2).}$$

2.3 Problem formulation

By applying parametrizations in the optimization **P2**, input trajectories can be searched for only in a subset of $\mathbb{R}^{mN}$ which is available for the original optimization **P1**. Given a desired feasible set $\mathcal{X}_f \subset \mathcal{X}$, **P1** feasible on $\mathcal{X}_f$, the subset of available input trajectories has to be chosen such that also **P2** is feasible on $\mathcal{X}_f$. Beyond that it should be chosen such that the optimum attained by **P2** is close to the one attained by **P1**. As the optimizers of **P1** are state dependent, it makes sense to establish a relation of the sets $\mathcal{S}_i$ and the parameters M_i and $\mathcal{K}_i$. Hence, ideally, these parameters are determined within one step depending on each other. Yet, this is generally a challenging task. Summarizing, the remainder of this paper addresses the following goals:

- State an algorithm to determine a segmentation $\mathcal{S}_i,\ i = 1, \dots, K$ of $\mathcal{X}_f$ and preliminary M_i, $\mathcal{K}_i$ jointly.
- The parameters should result in low suboptimality of **P2** w.r.t. **P1**.
- Refinements of the parametrizations to improve feasibility of **P2** on a given set $\mathcal{X}_f \subset \mathcal{X}$ are to be presented.
- For given **P1** and parametrization $p(\cdot,\cdot)$, state results on maximal constraint violation and guarantees on constraint satisfaction of **P2**.

3. OBTAINING THE PARAMETRIZATIONS

In order to find the parametrizations offline, a tailored data mining algorithm is applied to a set of pairs of states and corresponding optimal predicted input trajectories. In this section, first the data mining algorithm to be applied is presented. Secondly, application of the algorithm to obtain preliminary parametrizations is explained and refinements of the parametrizations are introduced. Finally, guarantees on feasibility and constraint satisfaction are given.

3.1 Clustering Algorithm

In this section an algorithm is presented extending the K-q-flats clustering regression procedure introduced by Goebel and Allgöwer [2014]. The latter procedure takes a set of data point pairs $(x_r, U_r) \in \mathbb{R}^n \times \mathbb{R}^{mN}$, $r = 1, \dots, p$ and parameters $K, q \in \mathbb{N}$ as input. The data points are grouped into K clusters such that the residual after establishing for each cluster a linear correlation of the x parts and the U parts lies near a q-dimensional subspace. In other words, a piecewise linear function approximation is done and the residual is required to lie close to a q-dimensional subspace for each piece. The current algorithm, in addition, allows to influence the shape of the clusters in x space, i.e., compact clusters in x space can be favored. The benefits of this feature will be shown in the sequel of this paper. Summarizing, the clustering procedure aims at finding an approximation of the solution of the following optimization problem:

$$\begin{aligned}\min_{M_i,\mathcal{K}_i,\tilde{U}_r,\mu_i^r} \quad & V(M_i,\mathcal{K}_i,\tilde{U}_r,\mu_i^r)\\ \text{s.t. } \mu_i^r \in \{0,1\} \text{ and } & \sum_{i=1}^{K}\mu_i^r = 1,\end{aligned} \tag{4}$$

$$V(M_i,\mathcal{K}_i,\tilde{U}_r,c_i,\mu_i^r) =$$
$$\sum_{i=1}^{K}\sum_{r=1}^{p}\mu_i^r\left(\gamma||U_r - M_i\tilde{U}_r - \mathcal{K}_i x_r||_2 + (1-\gamma)||x_r - c_i||_2\right).$$

Therein, μ_i^r determines cluster membership of the data. The matrices $\mathcal{K}_i \in \mathbb{R}^{mN\times n}$ establish the linear correlation of the x and the U parts and the matrices $M_i \in \mathbb{R}^{mN\times q}$ determine the subspaces the residuals lie close to. $c_i \in \mathbb{R}^n$ is the center of the ith cluster in x space. $\gamma \in [0,1]$ weights approximation accuracy for the U values against compactness of the clusters in x space. $\tilde{U}_r \in \mathbb{R}^q$ is only needed for compact problem formulation. Note that for the case $\gamma = 0$ this algorithm reduces to the well known K-means clustering algorithm applied to the x parts whereas for $\gamma = 1$ the K-q-flats clustering regression algorithm applied to the U parts is obtained. Due to its complexity and non-convexity, obtaining a good solution of (4) directly is impossible. Hence, following the spirit of the K-means algorithm, the current algorithm applies the strategy to iterate in between updating the clusters in one step and updating cluster membership in the second step. Both subproblems are simple and can be solved efficiently.

Cluster update step In the cluster update step, new matrices M_i, $\mathcal{K}_i$, $i = 1,\dots,K$ are determined via solving

$$\min_{M_i,\mathcal{K}_i,\tilde{U}_r}\sum_r \mu_i^r||U_r - M_i\tilde{U}_r - \mathcal{K}_i x_r||_2 \tag{5}$$

for given cluster membership μ_i^r. As this optimization has an explicit solution it can be solved efficiently numerically, see the paper by Gabriel [1978]. New centers c_i, $i = 1,\dots,K$ for the clusters in x space are determined via

$$c_i = \frac{1}{\#\{r|\mu_i^r = 1\}}\sum_{\{r|\mu_i^r=1\}} x_r. \tag{6}$$

Assignment step In the assignment step, cluster membership μ_i^r is to be optimized based on fixed cluster specific M_i, $\mathcal{K}_i$, c_i. Each data point is assigned to the cluster which provides the best approximation. Hence, μ_i^r is taken as the argument of the solution of the following optimization problem

$$\begin{aligned}\min_{\mu_i^r,\tilde{U}_r}\sum_{i=1}^{K}\sum_{r=1}^{p}\mu_i^r\Big(&\gamma||U_r - M_i\tilde{U}_r - \mathcal{K}_i x_r||_2\\ &+ (1-\gamma)||x_r - c_i||_2\Big)\end{aligned} \tag{7}$$
$$\text{s.t. } \mu_i^r \in \{0,1\} \text{ and } \sum_{i=1}^{K}\mu_i^r = 1,$$

The whole clustering technique is summarized as follows.

Algorithm 1. (Compactifying K-q-flats clustering regression)

Input: Pairs $(x_r, U_r) \in \mathbb{R}^n \times \mathbb{R}^{mN}$, $r = 1,\dots,p$, parameters $K, q \in \mathbb{N}$, $\gamma \in [0,1]$.

(1) Initialize cluster membership randomly, i.e., choose μ_i^r, $i = 1,\dots,K$, $r = 1,\dots,p$ randomly subject to the constraints in (4).
(2) Update M_i, $\mathcal{K}_i$, $i = 1,\dots,K$ according to (5) and $c_i, i = 1,\dots,K$ according to (6).
(3) Determine new assignment μ_i^r, $i = 1,\dots,K$, $r = 1,\dots,p$ via (7). If the assignment μ_i^r has changed w.r.t. previous iteration and it results in lower cost V than previous one, go back to step 2). Else: done.

Output: $M_i, \mathcal{K}_i, c_i$, $i = 1,\dots,K$ and cluster memberships μ_i^r, $i = 1,\dots,K$, $r = 1,\dots,p$.

The presented algorithm aims at finding a simple approximation of the U values as far as possible exploiting structure of the data. Yet, revealing structure of the data is not the main goal as it is, for example, in typical applications of K-means clustering. Hence, the values K, q, γ are not determined a priori by the data considered but they are design parameters: Increasing K and q, the approximation gets more accurate but, at the same time, the approximation gets more complex consisting of more segments and higher dimensional subspaces. γ in addition can be used to enforce clusters of compact shape in x space. The following convergence result for the algorithm can be stated.

Theorem 1. (Convergence of clustering). Algorithm 1 converges within a finite number of iterations. The solution obtained can not be improved by changing cluster membership alone nor by changing any of the clusters alone.

The proof of this result follows exactly the same lines as in the K-q-flats clustering regression case given by Goebel and Allgöwer [2014], and, hence, is omitted for brevity. In the sequel of this paper, the algorithm will be applied and also the effect of different γ values will be considered.

3.2 Obtaining preliminary parametrizations

The strategy in order to obtain preliminary parametrizations is as follows. Take suitable sample points x_r, $r = 1,\dots,p$ from the desired feasible region $\mathcal{X}_f$ of the state space and compute the corresponding optimal predicted input trajectories $U_r = U(x_r)$, $r = 1,\dots,p$ based on **P1**. Feed these samples into the above clustering algorithm and take its output $\mathcal{K}_i$ and M_i for the preliminary parametrizations. The segments $\mathcal{S}_i$ can then be determined based on cluster membership of the sample points in state space. The following algorithm summarizes this procedure.

Algorithm 2. (Preliminary parametrizations).
Input: Optimization problem **P1**, parameters $K, q \in \mathbb{N}$, $\gamma \in [0,1]$, sample points $x_r \in \mathcal{X}_f$, $r = 1,\dots,p$.

(1) Using **P1**, compute optimal predicted input trajectories $U_r = U(x_r)$, $r = 1,\dots,p$.
(2) Run Algorithm 1 using $(x_r, U_r), r = 1,\dots,p$ and K, q, γ as input.
(3) Take the outputs M_i as preliminary parametrization matrices $\hat{M}_i$, the outputs $\mathcal{K}_i$ as preliminary feedback matrices $\hat{\mathcal{K}}_i$ and use cluster membership μ_i^r.

Output: Preliminary parametrization matrices $\hat{M}_i$, preliminary feedback matrices $\hat{\mathcal{K}}_i$, clustering μ_i^r of points x_r.

Let $g : \mathcal{X}_f \to \{x_r | r = 1, \dots, p\}$ be a function which maps each point in $\mathcal{X}_f$ to the closest grid point, choosing the one with the lowest cluster index in non-unique cases. The segments $\mathcal{S}_i$ of the state space are then defined by

$$\mathcal{S}_i := \{x \in \mathcal{X}_f | \text{cluster}(g(x)) = i\}. \tag{8}$$

Loosely speaking, each point is assigned to the same cluster as the closest grid point and $\mathcal{S}_i$ consists of all points assigned to cluster i. By this definition, $\mathcal{S}_i \cap \mathcal{S}_j = \varnothing,\ i \neq j$ and $\cup_{i=1}^{K} \mathcal{S}_i = \mathcal{X}_f$ is guaranteed which is necessary for the parametrizations to be well defined. In the latter algorithm, in principle any collection of points x_r taken from $\mathcal{X}_f$ could be used. Noting that knowledge about optimal input trajectories is sampled at these points, it is reasonable to use points from an equally spaced grid over $\mathcal{X}_f$. By virtue of optimization (4), the preliminary parametrizations approximate the optimal predicted input trajectories optimally w.r.t. the 2-norm. Thus, suboptimality of the parametrized optimization can be expected to be small with respect to the unparametrized version. Nevertheless, so far the constraints (2) of **P1** have not been taken directly into account. To address this, corresponding refinements of the parametrizations are discussed next.

3.3 Refinement of the parametrizations

In this subsection, a parametrization refinement procedure improving feasibility of **P2** for fixed segments $\mathcal{S}_i$ is presented. Refinement of the parametrization used on one exemplary $\mathcal{S}_i$ will be described and the procedure applies for all segments $\mathcal{S}_i,\ i = 1, \dots, K$ likewise. We follow the strategy to consider a number of test points $x_r^i \in \mathcal{S}_i, r = 1, \dots, N_t^i$ and change $\mathcal{K}_i$ and M_i such that feasibility of **P2** for these points is achieved. More in detail, the procedure weights the test points against each other such that an update of $\mathcal{K}_i$ and M_i using the weighted test points renders **P2** feasible for all of them. For the refinement procedure, an extended form of the cluster update step (5) is needed including weights of the single data point pairs. Let $(x_r, U_r),\ r = 1 \dots N_t^i$ be test points with their corresponding input trajectory determined via **P1** and let $w_r > 0,\ r = 1 \dots N_t^i$ be corresponding weights. The weighted cluster update step is then given by

$$\min_{M_i, \mathcal{K}_i, \tilde{U}_r} \sum_r w_r ||U_r - M_i \tilde{U}_r - \mathcal{K}_i x_r||_2. \tag{9}$$

The following algorithm describes the procedure in detail:

Algorithm 3. (Refinements).
Input: Test points $(x_r, U_r),\ r = 1 \dots N_t^i$, maximum number of iterations $I_{\max}$, optimization problem **P1**.

(1) Initialize state weights $w_r = 1,\ r = 1 \dots N_t^i$ and iteration index $I = 0$.
(2) Determine $\mathcal{K}_i, M_i$ as arguments of the solution of (9).
(3) Set $I^+ = I + 1$.
(4) For each r, check feasibility of $\mathbf{P2}(x_r)$. If $\mathbf{P2}(x_r)$ infeasible, update $(w_r^i)^+ = 5(w_r^i)$.
(5) If any w_r^i has changed in (4) and $I < I_{\max}$, go to (2). Else: done.

Output: Parameters $\mathcal{K}_i, M_i$.

A main advantage of the procedure is its numerical simplicity even for large problems. Feasibility is checked only for one state at a time and the updating step (9) considering all points at once is numerically uncritical. Note that the weight boosting factor in step (4) is a design parameter which can be set to any number larger than 1.

The procedure improves approximation of optimal input trajectories for infeasible states by increasing the weight of these states. As the optimal input trajectory is feasible, approximating it sufficiently well renders **P2** feasible for the corresponding state. This way the procedure generally improves feasibility in the test points. Nevertheless, it is not guaranteed to achieve feasibility even for the test points. Intuitively, one could say that the algorithm helps to distribute the degrees of freedom provided by $\mathcal{K}_i, M_i$ such that feasibility is achieved for all points. Yet, the total degrees of freedom of the parameters might not be sufficient to achieve feasibility for all points. Thus, increasing either of the values K, q will increase the available degrees of freedom and thereby improve feasibility of the refinement step. As q is the number of online opimization variables and K the number different parts of the parametrization to be stored, the online computational effort and the storage requirements can be traded off against each other via q and K. $q = mN$ is a theoretic upper bound guaranteeing feasibility of the parametrizations.

3.4 Guarantees on constraint satisfaction and feasibility

In order to state guarantees on constraint satisfaction and feasibility applying the parametrizations, Lipschitz continuity of the system is assumed.

Assumption 2. Assume that the system dynamic function $f(\cdot, \cdot)$ is Lipschitz continuous on $\mathcal{X} \times \mathcal{U}$ with Lipschitz constant L, i.e., $||f(x_1, u_1) - f(x_2, u_2)|| \leq L(||x_1 - x_2|| + ||u_1 - u_2||)$ for all $x_1, x_2 \in \mathcal{X}$ and $u_1, u_2 \in \mathcal{U}$.

Furthermore, an assumption on boundedness of the feedback matrix of the parametrizations is needed.

Assumption 3. Defining $((\mathcal{K}_i^1)^T \dots (\mathcal{K}_i^N)^T) := \mathcal{K}_i^T,\ \mathcal{K}_i^j \in \mathbb{R}^{m \times n}$, assume that $||\mathcal{K}_i^j||_{\text{F}} \leq \alpha,\ i = 1, \dots, K,\ j = 1, \dots, N$.

The following result on maximum constraint violation using parametrizations which are feasible for a number of test points can be stated.

Theorem 2. (Maximal constraint violation). Let Assumptions 2 and 3 hold and let optimization $\mathbf{P2}(x)$ be feasible for a number of test points $x_r,\ r = 1, \dots, N_t$. Let

$$\beta \geq \max_i \max_{x \in \mathcal{S}_i} \min_{\{x_r \in \mathcal{S}_i | r = 1, \dots, N_t^i\}} ||x_r - x||$$

be an upper bound on the distance of any point in $\mathcal{X}_f$ to the closest test point in the same segment $\mathcal{S}_i$. Then, relaxing the constraints $\mathcal{X}$ and $\mathcal{X}_N$ in (2) of **P2** by

$$\tau := \beta \max_{j \in \{1, \dots, N\}} \left(L^j + \alpha \left(\sum_{l=1}^{j} L^l \right) \right), \tag{10}$$

and relaxing $\mathcal{U}$ by $\alpha\beta$ renders $\mathbf{P2}(x)$ feasible for all $x \in \mathcal{X}_f$.

Proof. Consider a state x in $\mathcal{S}_i \subset \mathcal{X}_f$ and its closest test point x^t. By assumption, parameters $\tilde{U}^t$ exist such that the predicted states $x_1^t, \dots, x_N^t$ in **P2** originating from x^t fulfill the original constraints. Let $x_1, \dots, x_N$ be the predicted states applying $\tilde{U}^t$ for initial condition x. We show the following relation which implies the theorem

$$||x_j - x_j^t|| \leq \beta \left(L^j + \alpha \left(\sum_{l=1}^{j} L^l \right) \right), \ j = 1, \ldots, N. \quad (11)$$

Let u_j^t and u_j, $j = 0, \ldots, N-1$ be the predicted inputs using $p(x^t, \tilde{U}^t)$ and $p(x, \tilde{U}^t)$, respectively. We have $||u_j^t - u_j|| = ||\mathcal{K}_i^{(j+1)}(x^t - x)|| \leq \alpha\beta$. Inequality (11) can now be shown for $j = 1$:

$$\begin{aligned} ||x_1^t - x_1|| &= ||f(x^t, u_0^t) - f(x, u_0)|| \\ &\leq L||x^t - x|| + L||u_0^t - u_0|| \leq L\beta + L\alpha\beta. \end{aligned}$$

Using induction, (11) can be shown for $j = 2, \ldots, N$ completely analogously. For brevity this is omitted here. $\square$

The latter result can be utilized to obtain parametrizations such that optimization $\mathbf{P2}(x)$ is feasible for all $x \in \mathcal{X}_f$.

Corollary 3. (Guaranteed feasibility). Let the same assumptions as in Theorem 2 hold. Assume in addition that all test points are feasible using tightened constraint sets

$$\begin{aligned} \underline{\mathcal{X}} &= \{x \in \mathbb{R}^n | x + \Delta \in \mathcal{X} \text{ for all } ||\Delta|| < \tau\}, \\ \underline{\mathcal{X}_N} &= \{x \in \mathbb{R}^n | x + \Delta \in \mathcal{X}_N \text{ for all } ||\Delta|| < \tau\}, \\ \underline{\mathcal{U}} &= \{u \in \mathbb{R}^m | u + \Delta \in \mathcal{U} \text{ for all } ||\Delta|| < \alpha\beta\} \end{aligned}$$

in (2) of $\mathbf{P2}$. Then feasibility of $\mathbf{P2}(x)$ is guaranteed for all x in $\mathcal{X}_f$ using the original constraints.

This result follows directly from the latter theorem. It provides a way of determining parametrizations which are guaranteed to result in a feasible simplified optimization problem on a predefined subset $\mathcal{X}_f$ of the state space. For this purpose, the pre-tightening levels τ and $\alpha\beta$ of the constraints can be traded off against the number N_t of test points used. Using more test points allows to reduce β and thereby τ which results in less strict tightened constraints. Vice versa, using less test points will enlarge β and thereby τ which requires stricter tightening of the constraints.

4. APPLYING THE PARAMETRIZATIONS ONLINE

In this section, online application of the parametrization is briefly discussed. A main challenge when applying the suggested type of parametrizations is storing the segments $\mathcal{S}_i$ and evaluating membership of the current system state to one of them. If the segments introduced in (8) are used, storage requirements are high and identifying the right segment would cause large computational effort. Hence, using approximations $\tilde{\mathcal{S}}_i$ of the original segments $\mathcal{S}_i$ is advisable. For example, the sample points introduced above could be used to train a multiclass support vector machine (mSVM), see, e.g., the overview paper by Hsu and Lin [2002]. A mSVM can approximate the original segments up to any desired accuracy at relatively low storage requirements and it allows to evaluate membership of a point to the segments efficiently. Note that for approximated segments $\tilde{\mathcal{S}}_i$ guarantees analogously to Theorem 2 and Corollary 3 can be established simply by changing β to a possibly slightly larger value.

On the other hand, stability of the closed loop has to be taken care of when applying parametrizations. By Assumption 1, each predicted input trajectory can be continued appending the input obtained by the terminal control law. Hence, initial feasibility guarantees that for all future time steps an admissible predicted input trajectory is available. In order to guarantee stability, such continued input trajectory has to be considered in each time step and a newly optimized trajectory can be used only if it results in improved performance, i.e., a lower value of $J(x_0, \cdot)$. Taking the J obtained this way as a Lyapunov function, asymptotic stability of the closed loop results.

5. EVALUATION VIA A NUMERICAL EXAMPLE

In this section, the presented results are illustrated and evaluated via numerical studies of an example MPC problem. For this purpose, all algorithms presented in this paper were implemented in MATLAB. The general setup is as follows: A nominal optimization problem $\hat{\mathbf{P}}\mathbf{1}$ is taken as reference for the feasible set and for performance. Algorithms 2 and 3 are then used to determine a simplified optimization problem $\mathbf{P2}$ which contains a lower number of optimization variables but approximates the feasible set and the performance of the nominal optimization problem $\hat{\mathbf{P}}\mathbf{1}$. The system described by the discrete time dynamics

$$\begin{aligned} x_1^+ &= x_1 + 0.1\,(x_2 + (0.5 + 0.5x_1)u) \\ x_2^+ &= x_2 + 0.1\,(x_1 + (0.5 - 2x_2)u) \end{aligned}$$

is considered with state constraints $(x_1, x_2) \in \mathcal{X} = [-4, 4] \times [-4, 4]$ and input constraints $u \in \mathcal{U} = [-2, 2]$. In the MPC setup, $l(x_k, u_k) = x_k^T Q x_k + u_k R u_k$ is used with the weighting matrices $Q = \left(\begin{smallmatrix} 0.05 & 0 \\ 0 & 0.05 \end{smallmatrix}\right)$ and $R = 0.1$ and $F(x_N) = x_N^T P x_N$ with $P = \left(\begin{smallmatrix} 5.9353 & 5.2774 \\ 5.2774 & 5.9353 \end{smallmatrix}\right)$. For the nominal optimization problem, a prediction horizon of $\hat{N} = 15$ is used. The example and most of the parameters are taken from the paper of Schulze Darup and Mönnigmann [2012]. The only difference here is that the admissible set of states is chosen larger which basically means to aim for a larger feasible set of the optimization problem.

Preparing application of the presented algorithms, a regular grid of 41×41 points covering $\mathcal{X}$ was taken and for each of them $\hat{\mathbf{P}}\mathbf{1}$ was checked for feasibility using the nominal horizon $\hat{N}$. This way, an estimate of the feasible region $\mathcal{X}_f$ of $\hat{\mathbf{P}}\mathbf{1}$ was obtained. The resulting 727 feasible points were taken as sample points x_r, $r = 1, \ldots, p = 727$ for Algorithm 2 and likewise as test points for Algorithm 3. Applying parametrizations in the optimization problem, generally the set of feasible states shrinks and performance deteriorates. To counteract this effect, we started using an optimization $\mathbf{P1}$ with extended horizon $N = 18 > \hat{N}$ to derive the parametrized version $\mathbf{P2}$ from.

Algorithm 2 was run for different parameters K, q and γ. Therein, in each case the clustering part was initialized using 20 different random values and the one resulting in the lowest objective function V was used. Subsequently, refinements via Algorithm 3 were executed doing at most $I_{\max} = 20$ iterations. Whereas only in some cases the preliminary parametrizations were feasible for all test points, the refinement procedure turned out to be very effective in rendering such parametrizations feasible for all test points. As it is to be expected, choosing higher values for K and q generally resulted in better feasibility of the parametrizations. Despite that, feasibility for all test points could even be achieved for very low numbers of parameters, i.e., for example $q = 2$ for large enough K.

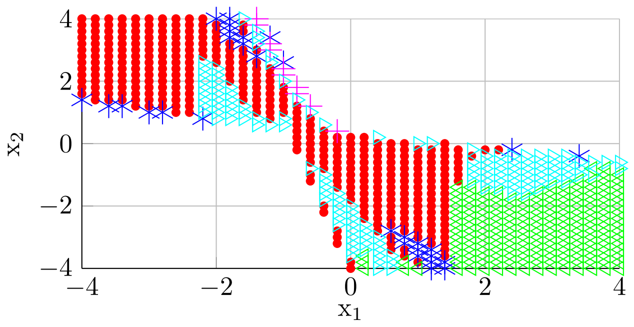

Fig. 1. Clusters in the state space for $K = 5$, $q = 3$, $\gamma = 1$.

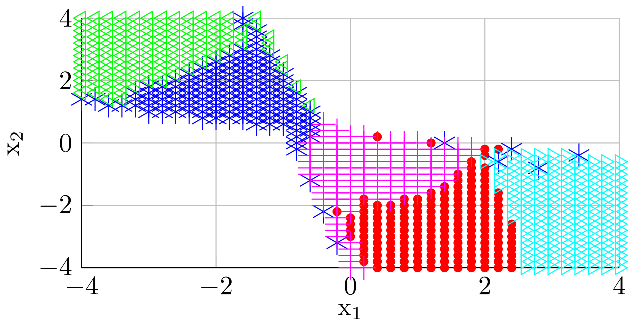

Fig. 2. Clusters in the state space for $K = 5, q = 3, \gamma = 0.5$.

The parameter γ turned out to be very effective in influencing the shape of the sets $\mathcal{S}_i$ in state space. For higher values of γ the sets $\mathcal{S}_i$ tended to be scattered over $\mathcal{X}$ whereas for lower values of γ the sets had a more compact shape. Figures 1 and 2 illustrate these findings for parameters $K = 5$, $q = 3$ and $\gamma_1 = 1$ and $\gamma_2 = 0.5$, respectively. Thus, γ can be used to simplify the shape of the sets $\mathcal{S}_i$ and thereby simplify online identification of the set the current state lies in. Furthermore, it turned out that compact sets generally had better feasibility properties. For example, for parameters $K = 10$, $q = 2$, $\gamma_1 = 1$, the refinement step could not render the parametrized optimization problem feasible for all test points whereas for $K = 10$, $q = 2$, $\gamma_2 = 0.5$ this was possible.

Finally, for two test cases feasibility and performance was evaluated on a finer grid of points in the state space. For this purpose the number of points in the estimated feasible set was doubled along both coordinate axes yielding a total of 2752 points. To determine membership of the new points to one of the sets $\mathcal{S}_i$, a multiclass support vector machine was first trained using the original points and corresponding set membership and then evaluated for the new points. Then, for each element of the enlarged set of grid points optimization problem **P2** was considered and feasibility and performance J was evaluated. This was done numerically using the MATLAB function `fmincon()` applying default settings and 0 as initial condition. The parameter sets used for this test were $K = 5$, $q = 3$, $\gamma = 0.5$ and $K = 10$, $q = 2$, $\gamma = 0.5$, respectively. In both cases, the parametrized optimization problems were feasible for almost all of the tested points and suboptimality w.r.t $\hat{\mathbf{P}}\mathbf{1}$ was very low. This means, using only two or three instead of 15 optimization variables, almost the full feasible set and the full performance were recovered. Distinct advantages could also be observed in comparison to different move blocking schemes which were tested in the same setting. Table 1 gives an overview of the results. For a fixed number of optimization variables, the best move blocking scheme we could find resulted in at least 8 times as many infeasible states as the proposed method and at the same time suboptimality was at least 2.5 times higher for the move blocking scheme. In the trade off of suboptimality vs. size of the feasible set vs. number of optimization variables, the proposed method showed to be significantly superior to move blocking.

Table 1. Comparison of feasibility and suboptimality with respect to the nominal optimization problem for different parametrizations.

Method	inf. points	avg. subopt.
move blocking, 2 parameters	12.5%	7.39 %
new method, $K = 10, q = 2, \gamma = 0.5$	1.3%	2.85 %
move blocking, 3 parameters	1.6%	2.52 %
new method, $K = 5, q = 3, \gamma = 0.5$	0.2%	0.52 %

6. CONCLUSIONS

A method to obtain parametrizations which allow to reduce the number of optimization variables in nonlinear MPC has been presented. For this purpose, a tailored data mining algorithm has been introduced and applied. The resulting parametrizations are state-dependent and they allow to exploit a linear correlation of states and predicted input trajectories. Guarantees on maximum constraint violation and feasibility of the parametrized optimization problem have been presented for Lipschitz continuous system dynamics. Finally, via a numerical example it was shown that the proposed method has the potential to provide low suboptimality and a large feasible set using a drastically reduced number of optimization variables.

REFERENCES

H. Chen and F. Allgöwer. A quasi-infinite horizon nonlinear model predictive control scheme with guaranteed stability. *Automatica*, 34(10):1205–1217, 1998.

K. R. Gabriel. Least squares approximation of matrices by additive and multiplicative models. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 186–196, 1978.

G. Goebel and F. Allgöwer. Obtaining and employing state dependent parametrizations of prespecified complexity in constrained MPC. In *52nd IEEE Conf. on Decision and Control, 2013*, pages 7077–7082. IEEE, 2013.

G. Goebel and F. Allgöwer. Improved state dependent parametrizations including a piecewise linear feedback for constrained linear MPC. In *American Control Conference, 2014*. IEEE, 2014. Accepted for publication.

C.-W. Hsu and C.-J. Lin. A comparison of methods for multiclass support vector machines. *Neural Networks, IEEE Transactions on*, 13(2):415–425, 2002.

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

S. J. Qin and T. A. Badgwell. A survey of industrial model predictive control technology. *Control engineering practice*, 11(7):733–764, 2003.

M. Schulze Darup and M. Mönnigmann. Low complexity suboptimal explicit NMPC. In *Nonlinear Model Predictive Control*, volume 4, pages 406–411, 2012.