Obtaining and employing state dependent parametrizations of prespecified complexity in constrained MPC

Gregor Goebel and Frank Allgöwer

Abstract—In this paper we propose a method of obtaining and employing state dependent parametrizations in order to reduce the on-line computational load in linear model predictive control (MPC). At the core of our results is the application of a data mining algorithm off-line to obtain a number of suitable parametrizations to approximate solutions of the MPC optimization problem. We show how to refine the parametrizations to achieve constraint satisfaction and employ them in an overall MPC scheme which provides guaranteed stability and constraint satisfaction at considerably reduced computational load. We apply the results in an illustrative example which shows the benefits of the proposed method.

I. INTRODUCTION

Model predictive control (MPC) is a well established optimization based control strategy, see e.g. [1]. Its ability to explicitly handle state and input constraints and the possibility to take control goals directly into account in form of a performance criterion are important reasons for the success of MPC. In its classical form, MPC requires on-line solution of an optimization problem at each time step, see e.g. [2, 3]. Yet, the computational load imposed by this optimization problem renders such MPC algorithms unsuitable for too large or fast systems. An attractive alternative for smaller linear systems is explicit MPC where the computational effort of the optimization is moved off-line completely [4]. A tesselation of the state space into polytopes is determined off-line such that for each polytope a linear affine state feedback law replaces the input obtained conventionally via optimization. Nevertheless, in this case the number of polytopes needed grows rapidly with the problem size. For applications this means large storage requirements and high computational effort in order to identify the right polytope. Consequently, for both types of MPC, simplifications have been proposed. In the case of explicit MPC this includes exploitation of obvious structure in the solution [5] and unification of polytopes [6, 7]. Computations in classical MPC have been simplified mainly by introducing different kinds of parametrizations for the optimization problem, however at the price of slight suboptimality. Most common in practical applications is simple move blocking, e.g. [8–11], newer results consider singular value decomposition based parametrizations [12, 13]. In [14] a combination of conventional and explicit MPC has been reported. Simplified explicit MPC solutions are used to warm start the on-line optimization.

In the current paper we also aim at combining ideas from both paradigms, however in a significantly different way. A strength in explicit MPC is that no solution of an optimization problem needs to be determined on-line since enough state dependent information about the solution is stored. A strength of conventional MPC, on the other hand, is that no state dependent data is needed since via the optimization problem all necessary knowledge can be computed on-line. So combining strengths of both approaches, namely keeping an optimization problem on-line and using some state dependent knowledge to simplify it, is promising. This way the storage requirements are considerably lower than for an explicit MPC scheme and the computational load is considerably reduced compared against a conventional scheme. In order to follow this road a method is needed to condense state dependent knowledge about optimal input trajectories such that it can be stored and evaluated efficiently. A major contribution of this paper is to employ a suitable data mining algorithm for this purpose. The condensed knowledge which will be used are state dependent parametrizations of the input trajectories, conceptually similar to the global parametrizations used in [12].

More in detail, we introduce an MPC algorithm as follows: In a first off-line stage, knowledge on typical solutions of the MPC optimization problem is generated. Namely, solutions to the optimization problem are considered on the desired feasible region of the state space. A specific data mining algorithm is then applied to these solution. The algorithm returns a number of linear affine subspaces which are suitable to approximate these optimal solutions. Both the number of subspaces used and the dimension of each subspace can be prespecified. Furthermore, an assignment of regions in the state space to one of the parametrizations is made. In a second step, the affine subspaces are refined in order to guarantee feasibility of the parametrized constrained MPC optimization problems on these regions. In the third stage of the proposed algorithm, these subspaces are exploited on-line for the MPC optimization problem. For each point of a prespecified feasible region within the state space, they allow to solve one out of a number of simplified optimization problems. By construction of the whole algorithm, stability and constraint satisfaction of the closed loop system is guaranteed.

The remainder of the paper is organized as follows. We start by giving a more detailed problem formulation and basics on MPC in Section II. Section III provides preliminaries on the clustering algorithm used. Our main results are presented in Section IV. An illustrative example and a conclusion are given in Sections V and VI, respectively.

G. Goebel and F. Allgöwer are with the Institute for Systems Theory and Automatic Control, University of Stuttgart, Pfaffenwaldring 9, 70550 Stuttgart, Germany, {goebel, allgower}@ist.uni-stuttgart.de
II. PROBLEM FORMULATION AND PRELIMINARIES ON MPC

We consider a linear, time-invariant, discrete-time system of the form
\[ x_{k+1} = Ax_k + Bu_k, \quad k = 0, 1, 2 \ldots \]  
with constrained state \( x_k \in \mathcal{X} \subset \mathbb{R}^n \) and constrained input \( u_k \in \mathcal{U} \subset \mathbb{R}^m \), wherein the constraint sets \( \mathcal{X} \) and \( \mathcal{U} \) are polytopic sets.

The system is to be controlled towards the origin from an initial condition \( x \) within a polytopic set \( \mathcal{X}_f \subset \mathcal{X} \). This can be achieved using a standard MPC setup. The following optimization problem is at the core of the MPC scheme.

Optimization Problem \( P_1(\hat{x}) \):
\[
\begin{aligned}
& \min_{\hat{u}(\hat{x})} \sum_{k=0}^{N-1} x_k^T Q x_k + \hat{u}_k^T R \hat{u}_k + x_N^T P x_N \\
& \text{s.t. } x_{k+1} = Ax_k + Bu_k, \\
& \quad x_k \in \mathcal{X}_f, \quad \hat{u}_k \in \mathcal{U} \quad \forall k = 1, \ldots, N-1, \\
& \quad x_0 = x, \quad \hat{u}_0 \in \mathcal{U}, \quad x_N \in \mathcal{X}_N
\end{aligned}
\]

where \( N \) is the prediction horizon and \( \hat{U} = (\hat{u}_0^T, \ldots, \hat{u}_{N-1}^T)^T \in \mathbb{R}^{mN} \) is the predicted control input. The positive definite symmetric matrices \( Q \in \mathbb{R}^{n \times n} \) and \( R \in \mathbb{R}^{m \times m} \) are state and input weights that make up the stage cost. The set \( \mathcal{X}_N \) is the terminal region and the positive definite symmetric matrix \( P \) is the terminal weight which accounts for the cost-to-go from the terminal state \( x_N \) on. We will call the optimal value of the optimization problem value function and denote it by \( \hat{J}(x) \). The corresponding argument is denoted by \( \hat{U}(x) \).

In an MPC scheme, the optimization problem is solved for the current state \( x \), the first part \( \hat{u}_0 \) of the obtained optimal input \( \hat{U} \) is applied to the plant and the process is repeated over a shifted horizon for the new system state.

Choosing the terminal region \( \mathcal{X}_N \subset \mathcal{X} \) and the terminal weight \( P \in \mathbb{R}^{n \times n} \) properly, this scheme results in an asymptotically stable closed loop.

Theorem 1 ([15]). Consider system (1) and the MPC scheme described above. Let the terminal region \( \mathcal{X}_N \) be the maximal invariant and constraint-admissible set for the LQR state feedback and let \( P \) be the positive definite solution of the algebraic Riccati equation associated with \( A, B, Q \) and \( R \). Then the closed loop is asymptotically stable.

The proof of the latter theorem is based on the strategy to use the value function \( \hat{J}(x) \) as Lyapunov function candidate. By choice of the parameters, it is guaranteed to decrease along closed loop trajectories.

At each time step, this MPC scheme requires solving an optimization problem over \( mN \) optimization variables. In order to reduce this number, and thereby reduce the computational load, we introduce a number of parametrizations for the predicted input trajectories. Let \( K \) be the number of different parametrizations used and let \( q < mN \) be the number of parameters. The parametrized predicted inputs are
\[ U = M_i \hat{U} + a_i, \quad i \in \{ 1, \ldots, K \} \]
with \( \hat{U} \in \mathbb{R}^q \) the parameters, \( M_i \in \mathbb{R}^{mN \times q} \) the parametrizations and \( a_i \in \mathbb{R}^{mN} \) an additional constant part. Seen from a general perspective, each parametrization can be considered as a \( q \)-dimensional linear affine subspace of \( \mathbb{R}^{mN} \). The columns of \( M_i \) span this subspace. From a more problem specific point of view, the columns of \( M_i \) are basis functions and the parameter \( a_i \) is a constant offset for the predicted input.

We will now compactly reformulate optimization problem \( P_1(x) \) and replace the predicted input \( \hat{U} \) by the parametrized version (2). To this end, substitute \( x_k = A^k x + \sum_{j=1}^{k-1} A^j Bu_{k-1-j} \). Then for suitable matrices \( H, F, Y, G, W, E \), and replacing \( \hat{U} \) by \( U \) we obtain
\[ P_2(x) : \min_{U} \langle M_i U + a_i \rangle^T H (M_i U + a_i) \]
\[ + x^T F (M_i U + a_i) + x^T Y x \]
s.t. \( G (M_i U + a_i) \leq W + Ex \).

We denote the optimal value obtained by this optimization problem by \( J_i^*(x) \) and the corresponding argument by \( U_i^*(x) \). \( P_2(x) \) is the simplified optimization problem that will be solved on-line in our MPC scheme. Since \( \mathcal{X}_f \) is our desired feasible region, for each \( x \in \mathcal{X}_f \) at least one \( i \in \{ 1, \ldots, K \} \) is needed such that the optimization problem \( P_2(x) \) is feasible. Based upon this introduction, we will give a proper problem formulation next.

Problem formulation

The overall goal of this paper is to present a simplification of a given MPC scheme by replacing the optimization problem \( P_1(x) \) by a number of parametrized versions \( P_2(x) \). For the sake of simplicity, we make the following assumption.

Assumption 1. For a given system (1) with desired feasible region \( \mathcal{X}_f \), the parameters \( N, Q, R, P, \mathcal{X}_N \) in optimization problem \( P_1(x) \) are such that \( \mathcal{X}_f \) is contained in the feasible region, \( \mathcal{X}_N \) is the maximal invariant and constraint-admissible set for the LQR state feedback, and \( P \) is the positive definite solution of the algebraic Riccati equation associated with \( A, B, Q \) and \( R \).

The task is then for a given system (1) with desired feasible region \( \mathcal{X}_f \) and given parameters \( K \) and \( q \) and a given MPC optimization problem \( P_1(x) \),

1) determine polytopes \( S_1, \ldots, S_K \subset \mathbb{R}^n \) such that \( \mathcal{X}_f \subset \bigcup_{j=1}^{K} S_j \) and
2) determine parameters \( M_i \in \mathbb{R}^{mN \times q}, a_i \in \mathbb{R}^{mN}, \quad i = 1, \ldots, K \) such that for \( x \in S_i \), \( P_2(x) \) is feasible and \( J_i^*(x) \) is finite.

Obviously the two tasks are related and thus should ideally be solved jointly. They will make up the off-line part of our algorithm. Furthermore, the on-line part of our algorithm should consist of a numerically efficient asymptotically stabilizing MPC scheme which employs these parametrizations.

III. PRELIMINARIES ON SUBSPACE CLUSTERING

In the following, some preliminaries on the data mining algorithm used in our results are given. Data mining denotes the application of methods and algorithms to extract meaningful patterns and knowledge out of large data sets,
see e.g. [16]. As a subclass, clustering algorithms group data points into a finite number of clusters based upon a particular criterion. If each one of the clusters to be found lies in or near a lower dimensional subspace of a higher dimensional surrounding space, subspace clustering methods are suitable to find the subspaces and determine membership of data points to the subspaces [17]. In our results, we make use of the “K-q-flats” clustering technique [17–19], a subspace clustering algorithm to find linear affine subspaces.

The input to the algorithm is a set of data points \( U_r \in \mathbb{R}^{mN}, r = 1, \ldots, p \), a number \( K \) of subspaces to be found and a parameter \( q \) which determines the dimension of the single subspaces. The algorithm then determines an approximation of the solution of the following optimization problem

\[
\min_{M_i, a_i, \tilde{U}_i, \mu_i} \sum_{i=1}^{K} p \sum_{r=1}^{p} \mu_i^r ||U_r - a_i - M_i\tilde{U}_i||_2
\]

s.t. \( \mu_i^r \in \{0, 1\} \) and \( \sum_{i=1}^{K} \mu_i^r = 1. \)

Therein, \( M_i \in \mathbb{R}^{mN \times q} \) and \( a_i \in \mathbb{R}^{mN}, i = 1, \ldots, K \) are the bases and the affine parts of the subspaces, respectively. The parameter \( \mu_i^r \) determines class membership of each data point \( U_r \). The parameters \( \tilde{U}_i \) are not used any further.

The K-q-flats algorithm then works as follows. Start with a random initialization of the subspaces \( M_i, a_i \), and iterate in between updating the class membership \( \mu_i^r \) according to the subspaces \( M_i, a_i \), and updating the subspaces \( M_i, a_i \) according to the data points in cluster \( i \). If the class membership \( \mu_i^r \) does no longer change, the algorithm has terminated. The algorithm is guaranteed to converge to a local minimum in finite time.

In our application the algorithm is primarily used to approximate data points instead of discovering underlying structure of the data. Hence in this case the fact that the K-q-flats clustering technique allows to prespecify the parameters \( K \) and \( q \) is a big advantage which allows to trade off approximation accuracy versus complexity.

IV. THE ALGORITHM

Within this section, we present the main results of this paper. We start by introducing the off-line part of our algorithm. It determines a segmentation of the state space into possibly overlapping polytopes. Assigned to each polytope, a parametrization of the MPC optimization problem is found. Following, the on-line part of the algorithm is introduced which results in an asymptotically stabilizing MPC scheme.

A. Off-line part of the algorithm

The core of the off-line part of our algorithm is the application of the K-q-flats clustering method to a characteristic collection of optimal predicted input trajectories. It returns preliminary versions of the searched for state space segmentation and of the parametrizations. Subsequently only slight refinements are necessary to ensure feasibility on the whole desired region of the state space.

1) Obtaining preliminary segmentation and parametrizations: The idea of this approach is the following: Take a number of points from the desired feasible region of the state space and determine the optimal predicted input trajectory for each of these points. Identify each of the obtained input trajectories with a single point in the input trajectory space \( \mathcal{Y}_f^N \subset \mathbb{R}^{mN} \). In the resulting collection of data points in \( \mathbb{R}^{mN} \) beneficial knowledge about typical solutions of the MPC optimization problem is hidden. Data mining algorithms can be used to extract this knowledge and make it accessible. In particular the K-q-flats clustering method is suitable since a prespecified number of linear affine subspaces of prespecified dimension are used to approximate the points. As shown above, such linear affine subspaces can directly be applied in the optimization problem to approximate the predicted inputs.

In detail, the first stage of our algorithm is as follows.

1) Take the desired feasible region of the state space \( \mathcal{X}_f \) and choose a regular grid of points \( x^k, k = 1, \ldots, N_f \) in \( \mathcal{X}_f \). Take in addition the vertices \( \hat{x}_f^k, k = 1, \ldots, N_f \) of the polytope \( \mathcal{X}_f \). Define the set of points

\[
\mathbf{X} = \{x^k : k = 1, \ldots, N_f\} \cup \{\hat{x}_f^k : k = 1, \ldots, N_f\}.
\]

2) For each \( x \in \mathbf{X} \), compute the associated optimal predicted input trajectory \( \hat{U}^*(x) \), i.e. the argument of the solution of the optimization problem \( P1(x) \). Define the set of all predicted input trajectories as

\[
\mathbf{U} = \{\hat{U}^*(x), x \in \mathbf{X}\}.
\]

3) Consider \( \mathbf{U} \) as a cloud of points in \( \mathbb{R}^{mN} \) and apply the K-q-flats clustering algorithm to the points.

4) The algorithm returns a separation of the predicted inputs into \( K \) classes: \( \mathbf{U} = \{\hat{U}_i^k : k = 1, \ldots, N_i, i = 1, \ldots, K\} \).

5) Furthermore, the parameters which describe the subspaces of the single clusters are obtained from the algorithm. We consider them as preliminary parametrizations \( \tilde{M}_i, \tilde{a}_i, i = 1, \ldots, K \).

6) Next, translate the clusters in the predicted input space into clusters in the state space: For each predicted input \( \hat{U}_i^k \) assign its associated point \( x \in \mathbf{X} \) to the same class as the predicted input, i.e. such that \( \hat{U}_i^k = \hat{U}^*(x^k) \). We obtain \( K \) clusters of points in the state space: \( \mathbf{X} = \{x_i^k : k = 1, \ldots, N_i, i = 1, \ldots, K\} \).

7) Take the convex hull of each cluster in the state space to obtain a preliminary segmentation, i.e. \( \hat{S}_i = \text{convhull} \{x_i^k : k = 1, \ldots, N_i\} \) for \( i = 1, \ldots, K \). At this point, locally optimal approximations of the predicted inputs \( \hat{U}^* \in \mathbf{U} \) with respect to the 2-norm have been determined for the chosen numbers \( K \) and \( q \). Nevertheless, since, so far constraints of the original problem have not been taken into account directly, the results are not applicable yet. Hence, refinements fixing these shortcomings have to be done.

2) Refining the segmentation to obtain the polytopes \( \hat{S}_i \): By construction, the union of all preliminary segments \( \hat{S}_i \) does not necessarily cover the whole desired feasible region \( \mathcal{X}_f \). Areas in between two neighboring polytopes \( \hat{S}_i, \hat{S}_j \) and
areas at the edge of the set $\mathcal{X}_f$ might not be covered by any of the segments. Yet, due to the construction of the regular grid, this gap does not exceed the size of a hypercube defined by $2^n$ neighboring grid points for a reasonable clustering result. Let $S_0$ be such hypercube defined by $2^n$ neighboring grid points which is centered at the origin. We define the final segments $S_i$ as $S_i = S_j \oplus 2S_0$, wherein “$\oplus$” denotes the Minkowski sum, and we make the following assumption.

**Assumption 2.** Assume that for the polytopes $S_i$, $i = 1, \ldots, K$ constructed as described above the following holds $\mathcal{X}_f \subset \bigcup_{i=1}^K S_i$.

3) Refining the parametrizations: In order to render the optimization problem $P_2(x)$ feasible for all $x \in S_i$, it suffices to render the optimization problem feasible at the vertices of the set $S_i$. This becomes clear from the fact that the optimization problem $P_2(x)$ is convex in $x$ and that the sets $S_i$ are convex polytopes. Hence, we next state an optimization problem which, if feasible, guarantees feasibility of $P_2(x)$ at the vertices of $S_i$. Let $\bar{x}_i^k$, $k = 1, \ldots, \bar{N}_i$ denote the vertices of $S_i$. Define the parameters

$$ U_i^k = M_i^+(U^* \bar{x}_i^k - \hat{a}_i) $$

where “$+$” denotes the Moore-Penrose pseudoinverse. Consider the optimization problem

$$
\min_{M_i, a_i} \sum_{k=1}^{\bar{N}_i} (M_i U_i^k + a_i)^T H (M_i U_i^k + a_i) \\
+ (\bar{x}_i^k)^T F (M_i U_i^k + a_i)
\quad \text{s.t. } G(M_i U_i^k + a_i) \leq W + E \bar{x}_i^k, \quad k = 1, \ldots, \bar{N}_i.
$$

We can state the following theorem.

**Theorem 2** (Feasibility on segments $S_i$). Let $S_i$ be determined as described above. Assume optimization problem (6) is feasible and $M_i$ and $a_i$ are arguments of the solution. Then the optimization problem $P_2(x)$ is feasible for all $x \in S_i$.

**Proof.** i) Feasibility of $P_2(x)$ for vertices $\bar{x}_i^k$: Since (6) is feasible, the inequality $G(M_i U_i^k + a_i) \leq W + E \bar{x}_i^k$ holds for all $k = 1, \ldots, \bar{N}_i$. This implies that $U_i^k$ satisfies the constraints of $P_2(x)$ at the vertex $\bar{x}_i^k$; ii) Feasibility on $S_i$: By construction, $S_i$ is a polytope. The constraints in $P_2(x)$ are convex in $x$. Hence, feasibility at the vertices of $S_i$ implies feasibility on $S_i$. □

Note that Theorem 2 also holds if a general value for $U_i^k$ is used. In order to further improve feasibility, an iterative procedure of determining $M_i, a_i$ from (6) and $U_i^k$ from (5) could be applied.

Feasibility of (6) highly depends on a proper choice of the design parameters $K$ and $q$ in relation to the desired feasible region $\mathcal{X}_f$. If $K$ and $q$ are chosen too small, only very inexact approximations of the predicted input trajectories are possible and constraint satisfaction can in general not be achieved. Furthermore, by parametrizing the predicted input trajectory, the feasible region of an MPC scheme will shrink. This means that the union of the feasible regions of $P_2(x)$, $i = 1, \ldots, K$ is in general smaller than that of $P_1$. Conversely, in order to reach feasibility of (6) the prediction horizon $N$ has to be chosen large enough such that the feasible region of the original problem $P_1(x)$ is larger than the desired feasible region $\mathcal{X}_f$ of the parametrized version.

The next corollary concludes the off-line part of the algorithm.

**Corollary 1** (Feasibility on $\mathcal{X}_f$). Let the sets $S_i$, $i = 1, \ldots, K$ be determined as described above and let Assumption 2 hold. Let the optimization problem (6) be feasible for all $i = 1, \ldots, K$ with arguments $M_i$, $a_i$ of the feasible solutions. If these parameters $M_i$, $a_i$ are used in the optimization problems $P_2(x)$, it holds that for all $x \in \mathcal{X}_f$ there is at least one $i$ such that $P_2(x)$ is feasible.

**Proof.** Assumption 2 guarantees that for each $x \in \mathcal{X}_f$ there is at least one $i$ such that $x \in S_i$. Theorem 2 guarantees that $P_2(x)$ for this particular $i$ is feasible. □

Summarizing, this part of our algorithm yields a prespecified number of optimized parametrizations of prespecified complexity for the MPC optimization problem and an assignment of points in the state space to the best parametrization. Each parametrization is feasible on a particular polytopic section of the state space and the union of all sections covers the whole desired feasible region.

**B. On-line part of the algorithm**

The goal for the on-line part of our algorithm is to employ the parametrization introduced above in order to obtain an efficient asymptotically stabilizing MPC scheme. Thereby, we have to cope with the following issue, inherent to employing parametrizations in MPC. Meanwhile in each time step the prediction horizon is shifted forward in time, the parametrization is kept constant. As a result, the continuation of a specific input trajectory determined in one time step might not be available in the next time step, see also [10]. In addition, in our case the section $S_i$ might be changed from one time step to the next one, which changes the available parametrization completely. As a consequence, a Lyapunov argument to show stability of the closed loop can not be established directly as in the non parametrized case. Thus, a remedy is to explicitly keep a continuation of the predicted input trajectory in each time step as a backup solution for the next time step. Note that by Assumption 1, the input $u = K_{LQR}x_N$ is always admissible and it can be appended to the shifted previous input trajectory to obtain an admissible new trajectory. The on-line part of our algorithm is as follows.

1) Let the current state be $x$. Find an $i$ such that $x \in S_i$.
2) Compute $J_i^*(x)$ from $P_2(x)$ and obtain the argument $U_i^*(x)$.
3) If the backup solution $U_b$ has been determined before, evaluate the value function using $U_b$,

$$ J_b(x) = x^T F x + U_b^T H U_b + x^T F U_b. $$

Else, set $J_b(x) = \infty$.
4) Set $J(x) = \min\{J_i^*(x), J_b(x)\}$ and set $U$ the predicted input associated with the lower value function.
5) Prepare the next backup predicted input trajectory as $U_b = (u_1, \ldots, u_{N-1}, (K_{LQR}x_N)^T)^T$. 

7080
6) Apply the first predicted input $u_0$ to the system and go back to 1).

Loosely speaking, this scheme guarantees that the predicted input trajectory never becomes worse over time. Note that by construction constraints of the original MPC scheme are guaranteed to be satisfied at all times. Applying the scheme, the theorem below holds.

**Theorem 3 (Stability).** Consider System (1), the original MPC scheme based upon optimization problem $P1(x)$ and let Assumption 1 hold. Let furthermore all parameters of optimization problems $P2(i)$, $i = 1, \ldots, K$ be determined as stated in the off-line part of the presented algorithm and let Assumption 2 hold. Then, applying the on-line-part of the presented algorithm results in an asymptotically stable closed loop system in the set $\mathcal{X}_f$.

**Proof.** The proof follows the same lines as in the unparametrized case of Theorem 1. Take the value function $J(x)$ which is obtained in the on-line part of the algorithm as Lyapunov function candidate. Since $J(x)$ is obtained using parametrizations and the terminal cost resembles the optimal cost-to-go, it is bounded from below by the value function of the unparametrized case $\hat{J}(x)$. This is a positive definite function, so we have $0 < \hat{J}(x) \leq J(x)$, $x \neq 0$. On the other hand, according to Corollary 1, $J(x)$ is defined and finite for all $x \in \mathcal{X}_f$. In the algorithm, a new solution $U^*_i(x)$ is used only if it results in a lower value function than the backup solution. So in order to show that $J(x)$ decreases in each time step, it is sufficient to show that it decreases using only backup solutions. Because the terminal cost equals the actual cost-to-go applying the LQR state feedback, it also equals the actual cost-to-go applying only backup solutions. Hence, from one time instance to the next one, the value function $J(x)$ decreases exactly by the stage cost of the step taken. The stage cost is positive definite in $x$, which lets $J(x)$ decrease along closed loop trajectories. Thus, $J(x)$ is a Lyapunov function and the closed loop is stable.

Note that if a particular state $x$ lies in several sets $S_i$, the algorithm does not depend on which one of the admissible parametrizations is used.

This on-line part completes the presentation of our algorithm. It renders the parametrizations obtained in the previous subsection a valuable tool to simplify the MPC optimization problem of an existing MPC scheme while maintaining all guarantees on stability and constraint satisfaction of the closed loop. We will illustrate application and properties of the algorithm by an example in the next section.

**V. ILLUSTRATIVE EXAMPLE**

In this section, we apply the above introduced algorithm to a simple example system to provide insight into the results and benefits of the method. We consider an unstable oscillating system which is given in continuous time by the following equations

$$
\dot{x}(t) = \begin{pmatrix} 0.15 & 1 \\ -1 & 0.15 \end{pmatrix} x(t) + \begin{pmatrix} 0 \\ 1 \end{pmatrix} u(t).
$$

![Fig. 1. Clusters of grid points in the state space.](image)

The system is discretized at a sampling rate of 0.25s and we impose the input constraints $u \in [-0.1, 0.1]$ on the discretized system. For the MPC setup, state weight $Q = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}$ and input weight $R = 1$ are used. The terminal region $\mathcal{X}_N$ is computed using MATLAB and the Multi Parametric Toolbox (MPT) [20]. As a desired feasible region $\mathcal{X}_f$ exactly the feasible region of the non-parametrized MPC scheme with a prediction horizon of $N_{\text{feas}} = 20$ is desired.

We choose an initial grid of $35 \times 35$ equally spaced points in the state space which covers the whole desired feasible region $\mathcal{X}_f$. Points not contained in $\mathcal{X}_f$ and points contained in the unconstrained LQR region $\mathcal{X}_N$ are discarded. We determine the optimal predicted input trajectory under the given constraints for each of the points using a prediction horizon of $N = 24 > N_{\text{feas}}$. We then apply the K-q-flats clustering algorithm to the resulting data points in $\mathbb{R}^4$. The K-q-flats clustering result is obtained by starting and running the algorithm from 20 sets of initial conditions and choosing the best solution in a 2-norm sense. Running this clustering part takes less than 60 seconds using MATLAB on a standard PC (Intel core 2 quad 2.83GHz with 8GB of RAM). It turns out that for the clustering parameters $K = 5$ and $q = 4$ we obtain a feasible parameter refinement optimization problem (6).

The result of the clustering process in the state space is depicted in Figure 1, where the grid points used are colored and shaped according to their cluster membership. We see that the clusters do not just spread randomly over the set $\mathcal{X}_f$ but cluster membership of the points actually is state dependent. This underscores usefulness of the algorithm since the algorithm implicitly is founded on the assumption that there is an exploitable state dependency of good parametrizations for the input trajectories. Furthermore, the clusters are shaped almost point symmetrically around the origin, as is reasonable for the considered kind of symmetric problem. On the other hand, the clusters are not shaped in a completely predictable or trivial way which would make application of the algorithm unnecessary.

Figure 2 visualizes some of the obtained parameters. Columns of the matrices $M_i$ and the affine parameter $a_i$ are depicted as trajectories over the prediction horizon, to suggest their interpretation as basis functions. The upper
sub-figure belongs to the central cluster in Figure 1. We see that the constant parameter $a_1$ is almost kept constantly at zero. Since the corresponding cluster in the state space is located symmetrically around the origin, this could have been expected. In the lower sub-figure, parameters which belong to the lower left cluster in state space in Figure 1 are depicted. In this case it becomes obvious how the structure imposed by active input constraints is found and exploited by the algorithm. During step 1-3 the input constraints are kept active by the affine parameter alone meanwhile the remaining parameters are kept constantly at zero during this period.

Finally, we want to have a look at the complexity of the algorithm in comparison to alternative MPC implementations. Table I gives an overview on different alternatives considered. Besides conventional MPC, explicit MPC and our method, we included a move blocking scheme subject to the same stability constraints as the conventional method. For that purpose, we implemented different heuristic blocking schemes and tested for feasibility on a dense grid in $\mathcal{F}_f$. For a version using 8 parameters there were still single infeasible initial conditions even though overall feasibility seemed to be good. Note that in comparison to conventional and explicit MPC, move blocking and our algorithm provide suboptimal approximated results. From the table, the advantages of our method become obvious. Our scheme reduces the computational effort considerably in comparison not only to a conventional implementation but also to the move blocking scheme. It requires optimization over only 4 instead of 8 or even 20 variables. At the same time the on-line storage requirements and the effort to locate the proper region in the state space have been reduced largely. Only 5 instead of 945 regions are needed.

VI. SUMMARY AND OUTLOOK

We presented a method of simplifying a given MPC scheme by introducing state dependent parametrizations of prespecified complexity for the solution of the MPC optimization problem. The parametrizations were obtained off-line by applying a data mining technique called K-q-flats clustering to a set of predicted input trajectories. A refinement step was incorporated to guarantee feasibility on a prespecified set. Asymptotic stability of the resulting overall MPC scheme was shown. Finally, a numerical example was presented to illustrate the algorithm and its benefits.

Performance considerations and complexity studies of the algorithm will be subject of further investigations.

REFERENCES