Proceeding of the 2004 American Control Conference
Boston, Massachusetts June 30 - July 2, 2004

Two Level Model Predictive Control for the Maximum Control Invariant Set

Pascal Grieder* , Zhaoyang Wan†, Mayuresh Kothare‡ and Manfred Morari*

Abstract— **This paper presents an extension of the Two-Level Model Predictive Control (MPC) scheme presented in [13]. The procedure in [13] allows for computationally efficient MPC over a subset of the controllable state-space. The first controller level provides a stability guarantee and the second level optimizes performance. This two-level control scheme allows for a transparent tradeoff between the necessary on-line computation power and performance. However, the scheme also suffers from two drawbacks. The two-level controller [13] does not cover all controllable states and in order to guarantee constraint satisfaction in closed-loop, it is necessary to resort to open-loop control for certain initial states. These issues are dealt with in this paper. We will extend the procedure such that the controller covers the infinite time controllable set $\mathcal{K}_\infty(\mathcal{X}_I)$ with closed-loop stability and feasibility guarantee. The set $\mathcal{K}_\infty(\mathcal{X}_I)$ denotes all states which may be driven to the set $\mathcal{X}_I$ by an admissible control law.**

I. INTRODUCTION

In Model Predictive Control (MPC) an optimization problem is solved on-line over a prediction horizon N and subsequently only the first element of the obtained input sequence is applied to the plant. This procedure is repeated at each time step, thus leading to closed loop control. In general, a stability proof for the closed-loop system is obtained by imposing a set constraint on the terminal state x_N as well as an associated terminal cost $x'Px$ [11]. However, the constraint on x_N may require the prediction horizon N to be very large for the optimization problem to be feasible. Large prediction horizons inherently result in prohibitive computation times, thus creating a clear need for fast and efficient alternatives to the classic MPC approach.

The authors in [3] proposed to use multi-parametric programming to solve the quadratic optimization problem associated with MPC off-line. However, the computational complexity is exponential in the number of inputs m and prediction horizon N, thus making it unsuitable for large problems. The authors in [10] proposed an interpolation of feasible input sequences which were computed off-line beforehand to obtain robust stabilizing control. This idea was applied and extended in [13], [1], where the authors presented an efficient way of partitioning the state-space and interpolating input sequences by using a two level controller which provides close to optimal closed-loop performance. Level 1 is computationally efficient and generates a stabilizing feedback law whereas Level 2 is used to further optimize for performance.

The concept of [13] is extended in this paper to cover the infinite time controllable set $\mathcal{K}_\infty(\mathcal{X}_I)$. The resulting controller guarantees that the state will enter an arbitrarily small neighborhood of the origin in finite time and does not rely on the intermediate use of open-loop control to provide these properties. The method presented in this paper replaces Level 1 of [13] whereas Level 2 is identical to the scheme presented in [13]. The paper is structured as follows. A general problem definition and a recap of the results in [13] is given in Section II. Section III and IV describe the proposed control scheme while Section V illustrates the advantages of the proposed scheme on numerical examples.

II. PROBLEM STATEMENT AND PROPERTIES

In this section, we will first introduce the formulation of the MPC problem considered here, before we restate some of the results of two-level MPC published in [13].

Definition 1: A polytope is a bounded and closed set defined by a finite intersection of hyperplanes. A hyperplane bounding a polytope is referred to as a facet of that polytope. A full dimensional polytope in $\mathbb{R}^n$ with $n+1$ vertices is referred to as simplex.

A. Formulation of MPC

Assume a linear, time-invariant, discrete-time system

$$x(k+1) = Ax(k) + Bu(k), \qquad (1)$$

with $A \in \mathbb{R}^{n\times n}$ and $B \in \mathbb{R}^{n\times m}$. Let $x(k)$ denote the measured state at time k and x_k denote the predicted state at time k given the state $x(0)$. Let u_k be the computed input for time k, given $x(0)$. Assume now that the states and the inputs of the system in (1) are subject to the following constraints

$$x(k) \in \mathbb{X} \subset \mathbb{R}^n, \qquad u(k) \in \mathbb{U} \subset \mathbb{R}^m, \quad \forall k > 0, \qquad (2)$$

where $\mathbb{X}$ and $\mathbb{U}$ are bounded and closed polytopic sets containing the origin in their interior, and consider the finite-

Corresponding Author: E-mail: `grieder@control.ee.ethz.ch`, Tel. +41 01 632 7313
*Institut für Automatik, ETH - Swiss Federal Institute of Technology, CH-8092 Zürich
† GE Betz, Inc., Trevose, PA 19053, U.S.A.
‡ Department of Chemical Engineering, Lehigh University, Bethlehem, PA 18015, U.S.A.

time constrained optimal control problem

$$J_N^*(x(0)) = \min_{u_0,\ldots,u_{N-1}} \sum_{k=0}^{N-1} \left(u_k' \mathcal{R} u_k + x_k' \mathcal{Q} x_k\right) + x_N' \mathcal{Q}_f x_N \tag{3a}$$

$$\text{subj. to } x_k \in \mathbb{X},\ \forall k \in \{1,\ldots,N\}, \tag{3b}$$

$$u_{k-1} \in \mathbb{U},\ \forall k \in \{1,\ldots,N\}, \tag{3c}$$

$$x_N \in T_{\text{set}}, \tag{3d}$$

$$x_{k+1} = Ax_k + Bu_k,\quad x_0 = x(0), \tag{3e}$$

$$\mathcal{Q} \succeq 0,\quad \mathcal{Q}_f \succeq 0,\quad \mathcal{R} \succ 0. \tag{3f}$$

The optimizer $U_N^*(x(0)) = [u_0', \ldots, u_{N-1}']'$ to problem (3) is a function of the initial condition $x(0)$.

Definition 2: [5] The set $\mathcal{X}_I$ will denote the maximum admissible set for linear systems (1) subject to the optimal unconstrained LQR feedback law F_{LQR} derived from the cost objective in (3):

$$\mathcal{X}_I = \{x(0) \in \mathbb{R}^n |\ x(k) \in \mathbb{X},\ F_{LQR}x(k) \in \mathbb{U},\ x(k+1) = (A + BF_{LQR})x(k),\ \forall k \geq 0\}$$

Assume P_{ARE} to be the solution of the Algebraic Riccati Equation (ARE). If the terminal set constraint $T_{\text{set}} = \mathcal{X}_I$ and $\mathcal{Q}_f = P_{ARE}$ in (3), then stability and feasibility are guaranteed if (3) is implemented as Receding Horizon Control (RHC) [11]. RHC is a control policy where the optimization problem (3) is solved at each time-step but only the first input u_0 of the resulting input sequence is applied. We will henceforth assume that $T_{\text{set}} = \mathcal{X}_I$.

Definition 3: [8] The finite time controllable set $\mathcal{K}_s(\mathcal{X}_I)$ is the largest set of states in $\mathbb{R}^n$ for which there exists an admissible (for (2)) time-varying state feedback control law such that the set $\mathcal{X}_I$ is reached in s steps. Here,

$$\mathcal{K}_s(\mathcal{X}_I) = \{x(0) \in \mathbb{X} |\ \exists u(0) \in \mathbb{U},\ Ax(0) + Bu(0) \in \mathcal{K}_{s-1}(\mathcal{X}_I)\},$$

with $\mathcal{K}_0(\mathcal{X}_I) = \mathcal{X}_I$. The set $\mathcal{K}_s(\mathcal{X}_I)$ can easily be computed by applying projection methods (e.g., [7]).
The infinite time controllable set $\mathcal{K}_\infty(\mathcal{X}_I)$ is defined accordingly for $s \to \infty$. If $\mathcal{K}_s(\mathcal{X}_I) = \mathcal{K}_{s-1}(\mathcal{X}_I)$, this implies $\mathcal{K}_\infty(\mathcal{X}_I) = \mathcal{K}_s(\mathcal{X}_I)$ [8]. Since the constraints in (2) are compact the set $\mathcal{K}_\infty$ is also bounded. Note that the set $\mathcal{K}_\infty(\mathcal{X}_I)$ may not be finitely determined, even if it is bounded (e.g., if $\mathcal{K}_\infty(\mathcal{X}_I)$ has open boundaries).

B. Two Level MPC

The concept of two-level MPC is that at level 1, a stabilizing MPC solution is constructed by linear interpolation of off-line solutions, while at level 2, a suboptimal solution tailored to the computation resources is computed by solving an optimization problem of arbitrary size. Equation (3) can be formulated as an optimization problem with the free variables $U_N^*(x(0))$ and $\alpha(x(0))$ which corresponds to an upper bound on the value function, i.e., $J_N^*(x(0)) \leq \alpha(x(0))$.

Theorem 1: [13] Consider an LTI system (1) subject to the input and output constraints (2). At each sampling time k, a sequence of N-step control moves $U_N^*(x(k))$, which minimizes the upper bound $\alpha(x(k))$ on the MPC objective function $J_\infty^*(x(k))$, are obtained from the solution (if it exists) of the following linear objective minimization problem:

$$\min_{\alpha(x(k)),\, U(x(k))} \alpha(x(k)),\quad J_N^*(x(k)) \leq \alpha(x(k)), \tag{4}$$

subject to (3b) - (3e). The first control move of the sequence $U_N^*(x(k))$ is applied to the system. Suppose that (4) is feasible for $k = 0$, then the proposed controller makes the closed-loop system asymptotically stable.
Equation (4) can be solved by using LMI or quadratically constrained quadratic program (QCQP) solvers.

Corollary 1: [13] Consider a polytopic set of states $\mathcal{X} = Co\{x^{(1)}, ..., x^{(L)}\}$ in the state space $\mathbb{R}^n$, where Co denotes the convex hull, and $x^{(j)},\ j = 1, .., L$, are vertices of the convex hull. Suppose for each vertex $x^{(j)}$, the solution of the minimization in Theorem 1 is $\alpha^{(j)}$ and $U^{(j)}$. On line, at time k, if $x(k) = \sum_{j=1}^{L} \theta_j x^{(j)} \in \mathcal{X}$ with $\sum_{j=1}^{L} \theta_j = 1$ and $0 \leq \theta_j \leq 1$, then $\tilde{\alpha}(x(k)) = \sum_{j=1}^{L} \theta_j \alpha^{(j)}$ and $\tilde{U}(x(k)) = \sum_{j=1}^{L} \theta_j U^{(j)}$ are a feasible solution for the minimization in (4).
Given a Cartesian coordinate system $\mathbb{R}^n$ with a set of unit base vectors $e^{(1)}, ..., e^{(n)}$, the authors in [13] consider a polytope of the form $\mathcal{X} = Co\{\pm a_1 e^{(1)}, ..., \pm a_n e^{(n)}\}$ with $2n$ vertices and subsequently apply an interpolated input sequence according to Corollary 1 in Level 1 of the algorithm. In Level 2, an LMI (or QCQP) as in (3) is solved for $m_{free} < N$ free inputs, i.e., $U_{m_{free}}^* = [u_0^*, \ldots, u_{m_{free}-1}^*, u_{m_{free}}, \ldots, u_{N-1}]$, where the input sequence $[u_{m_{free}}, \ldots, u_{N-1}]$ is taken from Level 1. If the new cost functional improves upon that of Level 1 (i.e., $J_{m_{free}}^*(x(k)) < \tilde{\alpha}(x(k))$) the input from Level 2 is applied, otherwise the solution of Level 1 is used.

The 2-Level approach boasts significant computational advantages over standard solution methods, since the degrees of freedom m_{free} in the on-line optimization problem may be arbitrarily chosen. However, only considering diamond shaped sets $\mathcal{X}$ is restrictive since only a subset of the controllable states can be controlled with the 2-Level controller. Furthermore, even if the initial state is contained in $\mathcal{X}$, there is no guarantee the state will be able to remain within $\mathcal{X}$ for the future time steps, thus making it necessary to rely on open-loop control if the state exits the controllable set.

III. Level 1 Control for $\mathcal{K}_s(\mathcal{X}_I)$

As previously stated, the contribution of this paper is an efficient way of computing an interpolated input sequence according to Corollary 1 for all controllable states $x \in$

$\mathcal{K}_s(\mathcal{X}_I)$. In this section we will first describe the off-line computation procedure before presenting an efficient way of applying the results on-line.

A. Off-Line Computation

In this section, three methods of pre-computing certain elements of the optimal control solution will be presented. The three methods offer a trade-off between the necessary on-line and off-line computation time. Note that the off-line computation time may not be negligible for the approaches presented here, which makes the distinction between the three approaches necessary. First, the infinite time controllable set $\mathcal{K}_\infty(\mathcal{X}_I)$ will be computed in both half-plane

$$\mathcal{K}_\infty(\mathcal{X}_I) = \{x \in \mathbb{R}^n | Hx \leq K\}$$

and vertex

$$\mathcal{K}_\infty(\mathcal{X}_I) = \{x \in \mathbb{R}^n | x = \sum_{i=1}^{L} \Phi_i x^{(i)},\ \Phi_i \geq 0,\ \sum_{i=1}^{L} \Phi_i = 1\}$$

representation, where $x^{(i)}$ denotes the $i-th$ vertex of $\mathcal{K}_\infty(\mathcal{X}_I)$. Since the iterative computation of $\mathcal{K}_\infty(\mathcal{X}_I)$ may not terminate in finite time it is advisable to set an upper bound s on the number of iterations which are computed, such that we will henceforth consider only the set $\mathcal{K}_s(\mathcal{X}_I)$. The half-plane and vertex representation of polytopes will be referred to as $\mathcal{H}$ and $\mathcal{V}$ representation respectively. Though the move from $\mathcal{H}$ to $\mathcal{V}$ representation is computationally taxing [7], [4], [14], the procedure is performed off-line where runtime is not of primary importance. It should be noted however, that the requirement to move from $\mathcal{H}$ to $\mathcal{V}$ puts a limit on the size of problems which are tractable with the proposed approach.

In the second step of the off-line procedure, the optimal input sequence $U_N(x^{(i)})$ and the upper bound on the value function $\alpha(x^{(i)})$ are computed for each of the vertices $x^{(i)}$ of the reach-set $\mathcal{K}_s(\mathcal{X}_I)$ by solving (3). Note that a terminal set constraint (3d) needs to be imposed to guarantee feasibility (invariance) and that we also require an appropriate terminal weight $\mathcal{Q}_f$ to be used in order to guarantee stability [11].

Note that unlike [13], the state space $\mathcal{K}_s(\mathcal{X}_I)$ is not automatically divided into simplices. In order to deal with this issue, a lookup table is computed which associates the $f-th$ facet $h_f x \leq k_f$ to the vertices of that facet, i.e., $X^{(f)} = \mathcal{T}(f)$ where $X^{(f)}$ is a set of vertices which lie on the $f-th$ facet and $\mathcal{T}(f)$ denotes the lookup table function. After this initialization, three approaches are feasible:

(a) **Simplex-Based 1:** For each facet f, subdivide the polytope which is defined by the vertices $X^{(f)}$ and the origin into simplices (see Figure 1(b)). It is necessary to add the origin to each set of vertices in order to obtain a full dimensional simplex. A polytope with the vertices $X^{(f)}$ and the origin will henceforth be referred to as segment f. The simplex division of segment f can be achieved by applying Delaunay Triangulation [14] which is included in commercial software [2], [4]. Subsequently create a second lookup table $S(f,c)$ which associates all the simplices in $\mathcal{V}-$representation with the respective facet. In the table $X^{(f,c)} = S(f,c)$, f denotes the facet number, $c \in \mathbb{N}^+$ is the index for the associated simplex number and $X^{(f,c)}$ denotes the set of vertices defining simplex c.

(b) **Simplex-Based 2:** Directly divide the full polytope $\mathcal{K}_s(\mathcal{X}_I)$ into simplices via Delaunay Triangulation, i.e., do not deal with each segment f separately (see Figure 1(c)). This will result in a smaller number of simplices, but the on-line identification of the active simplex may take more time, i.e., there is a tradeoff between the necessary storage space and on-line computational speed. Subsequently store all simplices in $\mathcal{V}-$representation in a data structure which we will denote as $X^{(c)} = S(c)$, where c is an index counter used to access each of the simplices. As before $X^{(c)}$, is a set containing all vertices which define simplex c.

(c) **Facet-Based:** Add the origin to each vertex list stored in $\mathcal{T}(f)$.

Note that there is no difference in the off-line computation between methods (a) and (c) in two dimensions. While method (a) always divides the entire set $\mathcal{K}_s(\mathcal{X}_I)$ into simplices, method (c) will subdivide $\mathcal{K}_s(\mathcal{X}_I)$ into full dimensional polytopes. Once the vertices of each simplex (a,b) or of the set $\mathcal{K}_s(\mathcal{X}_I)$ (c) have been computed, solve (3) for each of the vertices and store the associated input sequence $U_N(x^{(i)})$ and $\alpha(x^{(i)}) = J_N^*(x^{(i)})$ in an appropriate data structure.

B. On-Line Computation

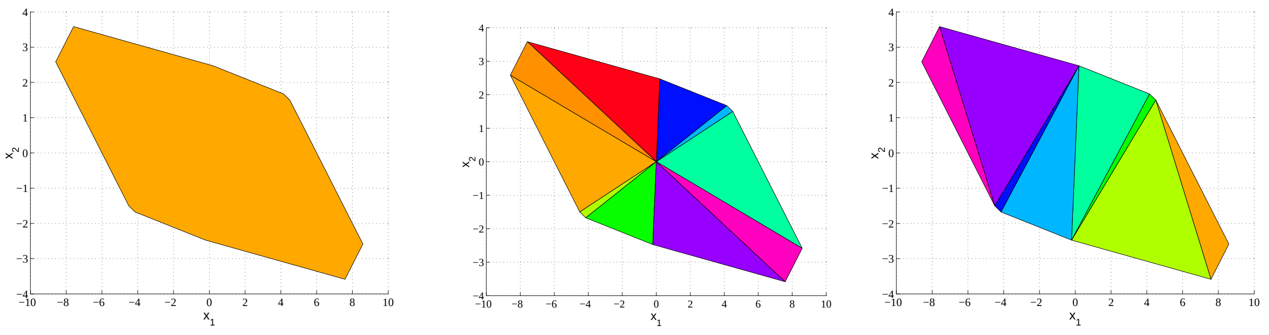
The following Lemmas will be used in the on-line procedure.

Lemma 1: [12] (i) Define the polytopes P_f as segment f of $\mathcal{K}_s(\mathcal{X}_I)$, i.e., P_f is the polytope defined by all vertices on facet f of $\mathcal{K}_s(\mathcal{X}_I)$ and the origin. (ii) Normalize the inequalities defining $\mathcal{K}_s(\mathcal{X}_I)$ according to

$$Hx - K \leq 0; \quad K^T = \begin{bmatrix} 1 \\ 1 \\ \vdots \end{bmatrix}; \quad H = \begin{bmatrix} h_1^T \\ h_2^T \\ \vdots \end{bmatrix}$$

Note that $0 \in \mathcal{K}_s(\mathcal{X}_I)$ follows from (2). (iii) Compute the values $\gamma_f = h_f^T x$ and compute f for which γ_f is a maximum. Then if $x \in \mathcal{K}_s(\mathcal{X}_I)$, it follows that x lies in polytope P_f.

Lemma 2: If a state is contained in a polytope $x \in \mathcal{P}$, then there exists a vector Φ, such that $x = \sum_{i=1}^{L} \Phi_i x^{(i)}$, $0 \leq \Phi_i \leq 1$, $\sum_{i=1}^{L} \Phi_i = 1$, where $x^{(i)}$ denotes the i-th vertex of $\mathcal{P}$. If $\mathcal{P}$ is a simplex and all $x^{(i)}$ are known, the computation of Φ can be realized with a simple function evaluation. By substituting $\Phi_1 = 1 - \sum_{i=2}^{L} \Phi_i$ an equation system with n equalities and $L = n$ unknowns is



(a) Reach-set $\mathcal{K}_6$.

(b) Triangulation method 1 of reach-set $\mathcal{K}_s$ (10 simplices).

(c) Triangulation method 2 of reach-set $\mathcal{K}_s$ (8 simplices).

Fig. 1. Reach-set $\mathcal{K}_s$ with the corresponding triangulation.

obtained which can be solved by simple matrix inversion. If the polytope $\mathcal{P}$ is a simplex, the inverse of the matrix will exist.

For efficient on-line implementation we propose the following algorithm,

Algorithm 1:

1) Obtain a state measurement $x(k) \in \mathbb{R}^n$. If $x(k) \notin \mathcal{K}_\infty(\mathcal{X}_I)$ then no feasible input sequence exists for the given state.
2) Initialize $c = 0$ and then proceed depending on which off-line approach was taken:
   a) **Simplex-Based 1:** Obtain the facet which is closest to the state $x(k)$ according to Lemma 1. Obtain the simplices in $\mathcal{V}$ representation associated with facet f through the lookup table $X^{(f,c)} = S(f,c)$. Apply the procedure of Lemma 2 to obtain Φ. If $||\Phi||_1 \neq 1$ or any $\Phi_i(x) < 0$, set $c = c + 1$ and repeat (a).
   b) **Simplex-Based 2:** Compute Φ according to Lemma 2. If $||\Phi||_1 \neq 1$ or any $\Phi_i(x) < 0$, set $c = c + 1$ and repeat (b).
   c) **Facet-Based:** Obtain the facet which is closest to the state $x(k)$ according to Lemma 1. Obtain the vertices associated with facet f through the lookup table $X^{(f)} = S(f)$ and solve the following LP

$$\begin{aligned}&\min_{\Phi} \quad [\alpha^{(1)}, \ldots, \alpha^{(I)}]\Phi, \text{ subj. to,}\\&0 \leq \Phi_i \leq 1, \quad i = \{1, \ldots, I\},\\&\sum_{i=1}^{I} \Phi_i = 1, \quad x(k) = \sum_{i=1}^{I} X^{(f)}(i)\Phi_i,\end{aligned}$$

   where I denotes the number of vertices associated with facet f and $X^{(f)}(i)$ denotes the $i-th$ vertex of facet f.
3) Obtain the input sequence $U(x(k)) = \sum_{i=1}^{I} \Phi_i U^{(i)}$, the cost bound $\alpha(x(k)) = \sum_{i=1}^{I} \Phi_i \alpha^{(i)}$.
4) If $\alpha(x(k)) < \alpha(x(k|k-1))$, apply the first input of the sequence $U(x(k))$. The term $(k|k-1)$ denotes the predicted state at time k given a state measurement at time $k - 1$. If $\alpha(x(k)) > \alpha(x(k|k-1))$, apply the second input of $U(x(k-1))$. For $U(x(k-1)) = [u_0, u_1, \ldots, u_{N-1}]$, now set $U(x(k)) = [u_0, \ldots, u_{N-1}, F_{LQR}x_N]$ and $\alpha(x(k)) = \alpha(x(k|k-1))$. The shifted bounds $\alpha(x(k|k-1))$ are easily computed as shown in [13] (Note that $\alpha(x(k|k-1)) < \alpha(x(k-1))$). Goto 1.

Method (a) and (b) have the advantage that no LPs will need to be solved on-line. Method (a) is faster on-line because of the efficient detection of the active simplex. However, (a) requires more storage space than (b), since more simplices are needed to cover the full polytope $\mathcal{K}_s(\mathcal{X}_I)$. Method (c) relies on solving LPs on-line. Though this is computationally more taxing than method (a) or (b), it is possible to include an objective when formulating the LP. The objective $[\alpha^{(1)}, \ldots, \alpha^{(I)}]\Phi$ will yield the interpolation resulting in the lowest bound on the value function and may thus produce a superior control law compared to Method (a) or (b). It should also be noted that although an LP needs to be solved at each time step, the size of this LP is given by the number of vertices of each segment f whereas the size of the control problem (3) is mainly influenced by the prediction horizon times the number of inputs. Therefore, the proposed approach may be significantly faster than direct computation of a feasible input sequence.

Theorem 2: Algorithm 1 ascertains asymptotic stability and constraint satisfaction (2) for all time.

Proof: **Feasibility:** Since the reach set $\mathcal{K}_s(\mathcal{X}_I)$ of an optimization problem as in (3) is convex and $Ax^{(i)} + Bu^{(i)} \in \mathcal{K}_s(\mathcal{X}_I)$ this directly implies $\sum_{i=1}^{I} \Phi_i(Ax^{(i)} + Bu^{(i)}) \in \mathcal{K}_s(\mathcal{X}_I)$. Since (3) will be feasible for all $x(k) \in \mathcal{K}_s(\mathcal{X}_I)$, constraint satisfaction is guaranteed for all time.

Stability: As is clear from Algorithm 1, Step 4, the function $\alpha(x)$ is strictly decreasing. Since $\mathcal{K}_s(\mathcal{X}_I)$ is bounded and because of Theorem 1, $\alpha(x)$ is also bounded from above and below. Therefore, the function $\alpha(x)$ serves as a Lyapunov function which guarantees asymptotic stability of the closed-loop system. ■

IV. Level 2 Control for $\mathcal{K}_s(\mathcal{X}_I)$

We will refer the reader to [13] for a detailed description of the second level and simply restate the basics in this section. Once the input sequence $U_N^*(x) = [u_0, \dots, u_{N-1}]$ and the bound on the cost function $\alpha(x)$ are obtained with the methods in Section III-B for an initial state x, an optimization problem is solved over an arbitrary number of free inputs m_{free}, i.e.,

$$L^*(x, U_{m_{free}}) = \min_{u_0^*,\dots,u_{m_{free}}^*} L(x, U_{m_{free}}) \tag{5a}$$

$$\text{subj. to } x_k \in \mathbb{X},\ \forall k \in \{1,\dots,N\}, \tag{5b}$$

$$u_{k-1} \in \mathbb{U},\ \forall k \in \{1,\dots,N\}, \tag{5c}$$

$$x_N \in T_{\text{set}}, \tag{5d}$$

$$J_N^*(x, U_{m_{free}}) < \alpha(x) \tag{5e}$$

$$x_{k+1} = Ax_k + Bu_k, \quad x_0 = x(0), \tag{5f}$$

where $U_{m_{free}} = [u_0^*, \dots, u_{m_{free}}^*, u_{m_{free}+1}, \dots, u_{N-1}]$. In general we will choose $m_{free} \ll N$, such that the on-line optimization problem is easily solved. This second level allows us to make use of the long prediction horizon computed at level 1 while optimizing for performance over only a small number of inputs. If the cost of Level 1 can be improved upon, i.e. (5e) is satisfied, the input sequence obtained at the second level will be implemented, otherwise the solution obtained at Level 1 is applied. Note that the Level 2 controller cannot destabilize the closed-loop system since (5e) is satisfied.

Remark 1: The presentation up to this point has been restricted to LTI systems and the standard open loop MPC formulation. However, as shown in [13], the extension to LTV or polytopic LDI systems is straightforward. It is furthermore possible to consider systems subject to persistent additive disturbances. Moreover, since MPC algorithms for LTV and LDI systems provide a computationally efficient alternative to nonlinear MPC algorithms for constrained nonlinear systems, the proposed algorithm can also be implemented for the control of constrained nonlinear systems.

V. Examples

Example 1: Consider the discrete-time double integrator with the state-space representation:

$$\begin{aligned} x(k+1) &= \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} x(k) + \begin{pmatrix} 1 \\ 0.5 \end{pmatrix} u(k) \\ y(k) &= [1 \ \ 0]\, x(k) \end{aligned}$$

The task is to regulate the system to the origin while fulfilling the input and output constraints

$$-1 \le u(k) \le 1, \quad -50 \le y(k) \le 50, \quad \forall k \ge 0$$

Example 1	Simplex 1	Simplex 2	Facet-Based
$\mathcal{K}_1$	8	6	3
$\mathcal{K}_5$	16	14	3
$\mathcal{K}_{10}$	26	24	3
$\mathcal{K}_{20}$	26	24	3
$\mathcal{K}_{30}$	36	34	3
$\mathcal{K}_{40}$	44	42	3

TABLE I
THE NUMBERS FOR THE SIMPLEX-BASED METHODS 1 & 2 DENOTE THE NUMBER OF SIMPLICES IN EACH REACH SET, WHILE THE NUMBER FOR THE FACET BASED METHODS DENOTES THE MAXIMUM NUMBER OF VERTICES FOR EACH SEGMENT OF $\mathcal{K}_s(\mathcal{X}_I)$.

The cost on the state is set to $Q = I$ and the input-cost is $R = 1$.
For example 1, the reach set computation terminates after 40 steps, i.e., $\mathcal{K}_\infty(\mathcal{X}_I) = \mathcal{K}_{40}(\mathcal{X}_I)$. The solution complexity of the result is shown in Table I. Note that the on-line implementation complexity for the simplex methods grows linearly with the number of obtained simplices, since a set-membership test needs to be performed for each simplex, in the worst case. The complexity of the facet-based approach grows polynomially, since interior-point LP solvers have polynomial complexity. For the Level 1 controller, problem (3) was solved for each vertex for a prediction horizon of $N = 40$. The resulting runtime and performance is given in Tables II and III. The runtime was compared with the MATLAB QP solver. The performance was measured by gridding the state space and computing the closed loop-trajectory cost to the origin for each initial state. A total of 115 initial states were considered. We did not consider the additional impact of the Level 2 controller, in order to highlight the contribution of this paper, which is the modification to the Level 1 controller. Note that even without the additional improvement of the level 2 controller, the performance degradation incurred by the Level 1 controller is marginal (see Table III).

Before concluding, a brief comparison of feedback controllers based on triangulation and multi-parametric feedback controllers [6] will be given. Note that unlike the method in [6], the triangulation based controller considered here does not guarantee optimal performance. Specifically, 20 random stable systems with $n = 4$ states and $m = 2$ inputs were generated. Subsequently, both the infinite horizon optimal solution with the multi-parametric algorithm in [6] and the triangulation controller (b) from Section III-A were computed. Figure 2 depicts the difference in the number of controller regions for two different cost objectives. As can be deduced, the number of controller regions obtained with multi-parametric programming can vary widely depending on the cost function in (3), whereas the complexity of the triangulation controller is independent of the performance objective. Therefore triangulation based 2-Level MPC may be preferable to multi-parametric controllers for certain problems.

Example 1	Simplex 1	Simplex 2	Facet-Based	QP solver
Average Time	1.24 msec	1.38 msec	1.62 msec	140 msec
Worst-Case Time	1.42 msec	1.58 msec	1.88 msec	258 msec

TABLE II

THE COMPUTATION TIME TO OBTAIN THE INPUT SEQUENCE FOR A GIVEN INITIAL STATE IS GIVEN FOR EXAMPLE 1 FOR THE DIFFERENT METHODS PRESENTED IN SECTION III-A. 'QP' DENOTES THE TIME NECESSARY TO SOLVE THE ASSOCIATED QUADRATIC PROGRAM FOR A PREDICTION HORIZON OF $N = 40$. THE RUN TIMES WERE OBTAINED ON A PENTIUM IV 2.25GHZ MACHINE, 1GB RAM AND THE MATLAB QP AND LP SOLVERS WERE USED.

Example 1	Simplex 1	Simplex 2	Facet-Based
Mean Performance Decrease	3.2%	4.3%	3.2%
Worst-Case Performance Decrease	8%	8.1%	8%

TABLE III

THE AVERAGE AND WORST CASE PERFORMANCE DECREASE INCURRED FOR EXAMPLE 1 FOR THE DIFFERENT METHODS PRESENTED IN SECTION III-A VERSUS AN QP SOLUTION OBTAINED BY OPTIMIZING FOR A PREDICTION HORIZON OF $N = 40$.

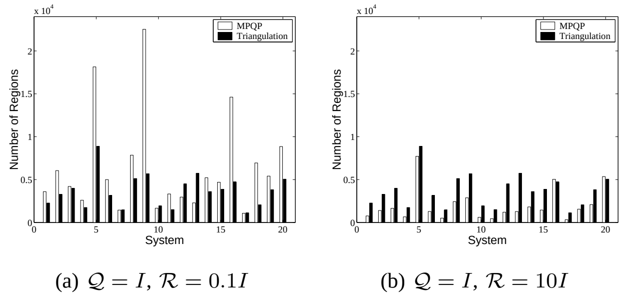


Fig. 2. Number of controller partitions mp-QP vs. Triangulation for different cost objectives in (3). The same systems were used for the two cost objectives.

VI. CONCLUSION

It was shown in this paper, how the 2-level MPC approach in [13] can be extended to cover the maximum controllable set. In the proposed method, input sequences which can be computed off-line are used in the on-line implementation to obtain feasible and stabilizing input sequences. As was shown, the proposed procedure may significantly reduce the on-line runtime necessary to obtain input sequences. Two fundamentally different procedures were presented. First, sets may be split into simplices by applying the Delaunay triangulation methods. For this method, on-line computation of the input sequence boils down to a simple set-membership test with subsequent function evaluation. Alternatively, polytopes may be defined for which the input sequence can be interpolated on-line by solving an LP. The main drawback of the proposed schemes is the computational complexity of the off-line procedure, which makes the approach intractable for large problems.

The presented algorithms are contained in the MPT toolbox [9] which is available for download.

REFERENCES

[1] M. Bacic, M. Cannon, and B. Kouvaritakis. Constrained nmpc via state-space partitioning for input-affine non-linear systems. *Int. J. Control*, 76(15), October 2003.
[2] C. B. Barber, D. P. Dobkin, and H. T. Huhdanpaa. The quickhull algorithm for convex hulls. *ACM Trans. on Mathematical Software*, 22(4):469–483, December 1996.
[3] A. Bemporad, M. Morari, V. Dua, and E.N. Pistikopoulos. The explicit linear quadratic regulator for constrained systems. *Automatica*, 38(1):3–20, January 2002.
[4] K. Fukuda. *Cdd/cdd+ Reference Manual*, December 1997. `www.cs.mcgill.ca/~fukuda/soft/cdd_home/cdd.html`.
[5] E. G. Gilbert and K. T. Tan. Linear systems with state and control constraints: the theory and applications of maximal output admissible sets. *IEEE Trans. Automatic Control*, 36(9):1008–1020, 1991.
[6] P. Grieder, F. Borrelli, F.D. Torrisi, and M. Morari. Computation of the constrained infinite time linear quadratic regulator. In *Proc. 2003 American Control Conference*, Denver, Colorado, USA, June 2003.
[7] S. S. Keerthi and K. Sridharan. Solution of parametrized linear inequalities by fourier elimination and its applications. *J. Opt. Theory and Applications*, 65(1):161–169, 1990.
[8] E. C. Kerrigan. *Robust Constraints Satisfaction: Invariant Sets and Predictive Control*. PhD thesis, Department of Engineering, The University of Cambridge, Cambridge, England, 2000.
[9] M. Kvasnica, P. Grieder, M. Baotić, and M. Morari. Multi Parametric Toolbox (MPT). In *Hybrid Systems: Computation and Control*, Lecture Notes in Computer Science. Springer Verlag, 2003. `http://control.ee.ethz.ch/~mpt`.
[10] Y. I. Lee and B. Kouvaritakis. Superposition in efficient robust constrained predictive control. *Automatica*, 38(5):875–878, May 2002.
[11] 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, June 2000.
[12] J. A. Rossiter and P. Grieder. Using interpolation to simplify explicit model predictive control. In *American Control Conference, Boston, USA*, June 2004.
[13] Z. Wan and M. V. Kothare. A two-level model predictive control formulation for stabilization and optimization. In *Proceedings of the American Control Conference*, pages 5294–5299, Denver, Colorado, June 2003.
[14] G. M. Ziegler. *Lectures on Polytopes*. Springer, 1994.