Proceedings of the 42nd IEEE
Conference on Decision and Control
Maui, Hawaii USA, December 2003

Complexity Reduction of Receding Horizon Control

Pascal Grieder and Manfred Morari
Automatic Control Laboratory, ETH Zentrum, ETL K 13, CH-8092 Zürich, Switzerland
grieder | morari @control.ee.ethz.ch

Abstract— **The off-line solution of constrained optimal control problems has garnered much attention because implementation can be realized with a simple look-up table. In this paper, a detailed complexity analysis of algorithms used to compute the explicit optimal control solution is given. Based on this analysis, two novel algorithms to compute controllers of significantly lower complexity are presented. Stability and feasibility of the closed-loop system are guaranteed by both algorithms. Extensive simulation results suggest that, on average, the controller complexity can be reduced by orders of magnitude at the cost of a performance decrease below** 0.5%**.**

Index Terms— **Optimal Control, Receding Horizon Control, Model Predictive Control, Stability, Feasibility**

I. Introduction

Receding Horizon Control (RHC) has become standard practice in modern control application. RHC is a control policy which computes an input sequence for a given system at each time step. After applying the first element in the input sequence, the input is recomputed at the next time step and the procedure is repeated. Stability of the closed-loop system can be guaranteed by solving an infinite-horizon optimization problem [11] or by imposing terminal set constraints [9].

The input sequence can be computed by solving optimization problems (e.g. quadratic or linear programs) on-line (*Model Predictive Control* - MPC)or by evaluating a look-up table which provides an optimal feedback law. The feedback solution of constrained optimal control problems has received increasing attention in literature [2]. The focus of this paper will be on the feedback solution, although the results may be used to speed up on-line optimization as well. The objective considered in this paper is the regulation of the state to the origin.

One of the key problems with the feedback solution to optimal control problems is the fact that the computational complexity grows very quickly. In this paper, we present a detailed complexity analysis which extends that of [2] by identifying the key drivers of complexity. Based on the insights gained, two new algorithms are proposed whose complexity is significantly lower than any other comparable algorithm to date [2], [5], [12]. The procedures can be applied to both linear and quadratic performance objectives. This observation is supported by extensive numerical examples.

II. Problem Formulation

In this paper we will consider optimal control problems for discrete-time linear time-invariant systems

$$x(t+1) = Ax(t) + Bu(t) \tag{1}$$

with $A \in \mathbb{R}^{n\times n}$ and $B \in \mathbb{R}^{n\times m}$. Let $x(t)$ denote the measured state at time t and $x_{t+k|t}$ denote the predicted state at time $t+k$ given the state at time t. For brevity we will denote $x_{k|0}$ as x_k. 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

$$x \in \mathbb{X} \subset \mathbb{R}^n, \qquad u \in \mathbb{U} \subset \mathbb{R}^m \tag{2}$$

where $\mathbb{X}$ and $\mathbb{U}$ are compact polyhedral sets containing the origin in their interior, and consider the finite-time constrained optimal control problem

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

$$\begin{aligned} \text{subj. to}\quad & x_k \in \mathbb{X}, \quad \forall k \in \{1,\dots,N\}, & (3b)\\ & u_k \in \mathbb{U}, \quad \forall k \in \{0,\dots,N-1\}, & (3c)\\ & x_{k+1} = Ax_k + Bu_k,\ x_0 = x(0), & (3d)\\ & \mathcal{Q} = \mathcal{Q}' \succeq 0,\ \ \mathcal{Q}_f = \mathcal{Q}_f' \succeq 0,\ \ \mathcal{R} = \mathcal{R}' \succ 0. & (3e)\end{aligned}$$

The solution to problem (3) has been studied in [2]. We will briefly summarize the main results. By substituting $x_i = A^i x(0) + \sum_{j=0}^{i-1} A^j B u_{i-1-j}$, problem (3) can be reformulated as

$$J_N^*(x(0)) = x(0)'\mathcal{Y}x(0) + \min_{U_N}\left\{U_N'\mathcal{H}U_N + x(0)'\mathcal{F}U_N\right\} \tag{4a}$$

$$s.t. \qquad GU_N \le W + Ex(0) \tag{4b}$$

where the column vector $U_N \triangleq [u_0', \dots, u_{N-1}']' \in \mathbb{R}^{Nm}$ is the optimization vector and $\mathcal{H}$, $\mathcal{F}$, $\mathcal{Y}$, G, W, E are easily obtained from $\mathcal{Q}$, $\mathcal{R}$, $\mathcal{Q}_f$, (1) and (2) (see [2] for details). Before going further, we will introduce the following definitions:

Definition 1. *We define the N-step feasible set $\mathcal{X}_f^N \subseteq \mathbb{R}^n$ as the set of initial states $x(0)$ for which the optimal control problem (3) is feasible, i.e.*

$$\begin{aligned}\mathcal{X}_f^N \;=\; & \{x(0) \in \mathbb{R}^n \,|\, \exists U_N \in \mathbb{R}^{Nm}, s.t.\\ & x_k \in \mathbb{X},\ u_{k-1} \in \mathbb{U},\ \forall k \in \{1,\dots,N\}\}.\end{aligned}$$

Definition 2. *The set of active constraints $\mathcal{A}^N(x(0))$ at point $x(0)$ of problem (4) is defined as*

$$\begin{aligned}\mathcal{A}^N(x) &= \{i \in I \mid G_i U_N^*(x) - W_i - E_i x = 0\},\\ I &= \{1, 2, \dots, m_G\},\end{aligned}$$

where G_i, W_i and E_i denote the i-th row of the matrices G, W and E, respectively and $G \in \mathbb{R}^{m_G \times s}$. N is the horizon length used to solve (4).

The solution to (3) can be found either by solving a quadratic program (QP) for a given $x(0)$, or as shown in [2], by solving problem (3) for all $x(0)$ within a polyhedral set of values, i.e. by considering (3) as a *multi-parametric Quadratic Program* (mp-QP).

Theorem 1. *[2] Consider the finite time constrained LQR problem (3). Then, the set of feasible parameters $\mathcal{X}_f^N$ is convex, the optimizer $U_N^* : \mathcal{X}_f^N \mapsto \mathbb{R}^{Nm}$ is continuous and piecewise affine (PWA), i.e.*

$$\begin{aligned} U_N^*(x(0)) &= \tilde{F}_r x(0) + \hat{G}_r \quad \text{if} \quad x(0) \in \mathcal{P}_r, \\ \mathcal{P}_r &= \{x \in \mathbb{R}^n | H_r x \leq K_r\}, \ r = 1, \dots, R, \end{aligned}$$

and the optimal solution $J_N^* : \mathcal{X}_f^N \mapsto \mathbb{R}$ is continuous, convex and piecewise quadratic.

Henceforth, we will denote the feedback law which provides the first input as $u_0 = F_r x + G_r$. Analysis of the mp-QP solution [2] leads to the following results on the solution of (3): $\{\mathcal{P}_r\}_{r=1}^R$ is a polyhedral partition of $\mathcal{X}_f^N$, where each $\mathcal{P}_r$ is defined by a constant set of active constraints, i.e. $\mathcal{A}^N(x) = \mathcal{A}_r^N$ if and only if $x \in \mathcal{P}_r$ ($\mathcal{A}_r^N \subseteq I, \forall r \in \{1, \dots, R\}$). Henceforth, we will denote the polyhedron $\mathcal{P}_r$ as *region* r. Once a certain set of active constraints $\mathcal{A}_r^N$ is identified, the authors in [12], [2], [1], [5] use the KKT conditions to compute the polyhedron $\mathcal{P}_r$ and the feedback law F_r, G_r as recalled next. First, new matrices $G_{\mathcal{A}_r^N}, W_{\mathcal{A}_r^N}$ and $S_{\mathcal{A}_r^N}$ are formed by extracting the rows indexed by $\mathcal{A}_r^N$ from G, W and S, where $S = E + GH^{-1}F'$. For region r, the feedback law is then given by $u = F_r x + G_r$:

$$F_r = \mathcal{H}^{-1} G'_{\mathcal{A}_r^N} (G_{\mathcal{A}_r^N} H^{-1} G'_{\mathcal{A}_r^N})^{-1} S_{\mathcal{A}_r^N} - \mathcal{H}^{-1}\mathcal{F}', \tag{5a}$$

$$G_r = \mathcal{H}^{-1} G'_{\mathcal{A}_r^N} (G_{\mathcal{A}_r^N} H^{-1} G'_{\mathcal{A}_r^N})^{-1} W_{\mathcal{A}_r^N}, \tag{5b}$$

and the polyhedron $\mathcal{P}_r = \{x \in \mathbb{R}^n | H_r x \leq K_r\}$ is computed as

$$H_r = \begin{bmatrix} G(F_r + \mathcal{H}^{-1}\mathcal{F}') - S \\ (G_{\mathcal{A}_r^N} \mathcal{H}^{-1} G'_{\mathcal{A}_r^N})^{-1} S_{\mathcal{A}_r^N} \end{bmatrix}, \tag{6a}$$

$$K_r = \begin{bmatrix} W - GG_r \\ -(G_{\mathcal{A}_r^N} \mathcal{H}^{-1} G'_{\mathcal{A}_r^N})^{-1} W_{\mathcal{A}_r^N} \end{bmatrix}. \tag{6b}$$

Various methods to identify the sets of active constraints $\mathcal{A}_k^N$, have been proposed in the literature [1], [2], [12].

III. Complexity Analysis

The following notation will be used in this section.

Number of states	n
Average number of constraints on one state	$\bar{n}_c$
Number of inputs	m
Average number of constraints on one input	$\bar{m}_c$
Prediction horizon	N
Number of terminal state constraints	$F(T_{set})$

Here, $\bar{n}_c = \frac{c_x}{n}$ and $\bar{m}_c = \frac{c_u}{m}$ with $\mathbb{X} \in \mathbb{R}^{c_x \times n}$ and $\mathbb{U} \in \mathbb{R}^{c_u \times m}$ in (2). The substitution was made in order to simplify the equations in the following subsections. Note that $T_{set} = \{x \in \mathbb{R}^n | H_{Tset} x \leq K_{Tset}\}$ represents a set of states and $f = F(T_{set})$ the number of facets of the set, i.e., $H_{Tset} \in \mathbb{R}^{f \times n}$.

In the following, we will assume that the algorithm in [1] is applied. Other algorithms, such as [12], [2], [5], vary in some aspects but the key issues remain the same. First, the algorithm run time per region is analyzed in Section III-A. Subsequently, an upper bound on the number of regions is given in Section III-B. The discussion is concluded with an analysis of the overall complexity in Section III-C. In the following analysis, only the number of quadratic programs (QPs) and linear programs (LPs) are considered, since the computation of QPs and LPs make up the bulk of the algorithm's runtime.

A. Run Time per Region

Starting from an initial region, a small step is taken over a facet of the region to obtain a point 'on the other side'. This requires the computation of one LP to obtain a point in the center of the facet. Subsequently a quadratic program is solved to compute the active constraints for the given initial state. Based on the active constraints, the region where the computed feedback law is optimal can be obtained [2]. Since the polyhedral description (6) contains redundant constraints, the polyhedral set has to be reduced to a minimal representation. To identify whether a given constraint is redundant, one LP is solved. Linear Programs are only solved for these two operations. This implies the following complexity for computing R regions:

$$\begin{aligned} \#LPs &= R(\bar{f} + p_{\text{reduce}}) \\ \#QPs &= R \end{aligned}$$

where R denotes the total number of regions, p_{reduce} denotes the number of LPs required to obtain the minimal realization of a region and $\bar{f}$ denotes the average number of facets of the minimal description of (6). The number of rows of H_r and K_r in (6) is $N(\bar{n}_c n + \bar{m}_c m)$, i.e. the total number of constraints on all inputs and states. This is because the constraints which are active do not need to be restated as inequalities in the first row of (6). Therefore, the following holds:

$$\begin{aligned} n + 1 \leq \ \bar{f} \ &\leq N(\bar{n}_c n + \bar{m}_c m), \\ p_{\text{reduce}} &= N(\bar{n}_c n + \bar{m}_c m), \end{aligned}$$

which gives an upper bound on the number of LPs

$$\#LPs \leq 2RN(\bar{n}_c n + \bar{m}_c m), \tag{7}$$

The issue of determining an upper bound on the number of regions R is addressed in the next section.

B. Number of Regions

From Section II it is clear that the number of possible regions is equal to the number of possible combinations of active constraints $\mathcal{A}$. This observation is now used to derive an upper bound on the number of regions. For sake of simplicity, we will first make some restrictive assumptions which will be dropped later. Assume a system with one input ($m = 1$) and n states. If the optimization problem (4) is solved for horizon $N = 1$, the upper bound on the number of regions can then be written as

$$R_{\text{bound}} = 1 + \bar{m}_c + \bar{n}_c n + F(T_{\text{set}}),$$

since only one constraint can be active at a time, such that the upper bound on the number of regions is equal to the number of constraints (plus 1 for no active constraints). Allowing arbitrary prediction horizons N and multiple inputs m, the upper bound on the number of possible combinations of active constraints is given by

$$R_{\text{bound}} = \sum_{i=0}^{Nm} \binom{Nm}{i} (\bar{m}_c + \bar{n}_c n + F(T_{\text{set}}))^i \qquad (8)$$

The binomial coefficient indicates that the number of active constraints may vary from region to region.

C. Overall Complexity

In practice, the computation time to obtain the explicit solution as well as the complexity of the resulting controller, i.e., the number of regions, is of interest. The complexity of the controller computation (off-line time) can be expressed for (7) and (8),

$$\begin{aligned}\#LPs &\leq \sum_{i=0}^{Nm} \binom{Nm}{i} (\bar{m}_c + \bar{n}_c n + F(T_{\text{set}}))^i \\ &\qquad \cdot 2N(\bar{n}_c n + \bar{m}_c m), \qquad (9a)\\ \#QPs &\leq \sum_{i=0}^{Nm} \binom{Nm}{i} (\bar{m}_c + \bar{n}_c n + F(T_{\text{set}}))^i. \quad (9b)\end{aligned}$$

The complexity of the on-line computation is proportional to the number of regions whose upper bound is given in (8). Therefore, it can be concluded that the key factors in reducing computation and controller complexity are the prediction horizon N and the number of inputs m, since they are the only variables in the exponent. Furthermore, the dominant aspect in the region exploration is the polytope reduction, which accounts for the bulk of the computation time.

IV. Iterative Algorithm

A. Off-Line Computation

As described in Section III-B, the key to reducing complexity of controllers lies in small prediction horizons and few inputs. Reducing the number of input variables is usually not an option because inputs are often defined by the system to be controlled. Most RHC algorithms to date [9], [5] rely on long prediction horizons to obtain stability and feasibility guarantees.

The procedure proposed here consists of iteratively solving 1-step optimization problems (10) with varying terminal set constraints X_{set}:

$$\begin{aligned} J_1^*(x(0)) &= \min_{u_0} u_0'\mathcal{R}u_0 + x_1'\mathcal{Q}_f x_1, && (10a)\\ \text{subj. to} \quad & x_1 \in \mathbb{X},\ x_1 \in T_{\text{set}},\ u_0 \in \mathbb{U}, && (10b)\\ & x_1 = Ax_0 + Bu_0,\ x_0 = x(0), && (10c)\\ & \mathcal{Q}_f = \mathcal{Q}_f' \succeq 0,\quad \mathcal{R} = \mathcal{R}' \succ 0. && (10d)\end{aligned}$$

The approach is similar to [10]. Due to the on-line implementation of the feedback controllers obtained when (10) is solved as an mp-QP, stability and feasibility guarantees can be given, as Section IV-B will show. Before moving on, the following definition is introduced:

Definition 3. *Let the control invariant set $\mathcal{X}_{\text{LQR}}$ be the biggest invariant set containing the origin, if the system in (1) is subject to the optimal feedback law F_{LQR} (Riccati LQR) for the unconstrained problem (3a). Specifically,*

$$\begin{aligned}\mathcal{X}_{\text{LQR}} = \{x(0) \in \mathbb{R}^n |\ & x(k) \in \mathcal{X}_{\text{LQR}},\ x(k) \in \mathbb{X},\\ & F_{\text{LQR}}x(k) \in \mathbb{U},\ x(k+1) = (A + BF_{\text{LQR}})x(k),\ \forall k \geq 0\}.\end{aligned}$$

Definition 4. *[8] We define the infinite time controllable set $\mathcal{K}_\infty(\mathcal{X}_{\text{LQR}})$ as the largest set of states for which there exists a control sequence satisfying (2), such that the state $x(k)$ will enter $\mathcal{X}_{\text{LQR}}$ as $k \to \infty$ for the system in (1).*

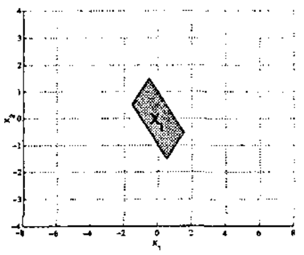
The set $\mathcal{X}_{\text{LQR}}$ can easily be computed with the methods in [4]. The algorithm proposed in this paper works as follows:

Algorithm 1.

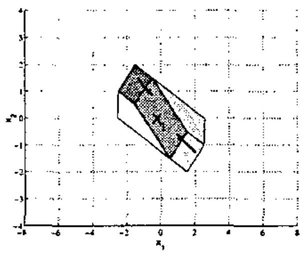
1) *Compute the invariant set $\mathcal{X}_{\text{LQR}}$ around the origin and set $T_{set} = \mathcal{X}_{\text{LQR}}$, $(\mathcal{X}_f^1)^0 = \mathcal{P}_0 = \mathcal{X}_{\text{LQR}}$, $F_0^0 = F_{LQR}$ and $G_0^0 = 0$. Initialize the iteration counter $iter = 1$.*
2) *Solve (10) as an mp-QP to obtain the region partition $\mathcal{P}_r^{iter} = \{x \in \mathbb{R}^n | H_r^{iter}x \leq K_r^{iter}\}$, $\forall r \in \{1, \ldots, R\}$ with the convex union [2] $(\mathcal{X}_f^1)^{iter} \equiv \bigcup_1^R \mathcal{P}_r^{iter}$ and the associated feedback laws $u = F_r^{iter}x + G_r^{iter}$. Store the region partition and the associated feedback laws in a separate data structure.*
3) *If $T_{set} \neq (\mathcal{X}_f^1)^{iter}$, set $T_{set} = (\mathcal{X}_f^1)^{iter}$, $iter = iter + 1$ and goto 2.*

Figure 1 illustrates the computation procedure of Algorithm 1 on a generic example. Note how the partition obtained at each iteration overlaps the previous controllers. The complexity of Algorithm 1 can be written as

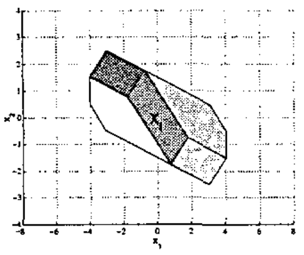
$$R_{\text{bound}} = \sum_{l=0}^{m} \binom{m}{l} N(\bar{m}_c + \bar{n}_c n + \bar{T}_{\text{set}}(n, m, N))^l,$$



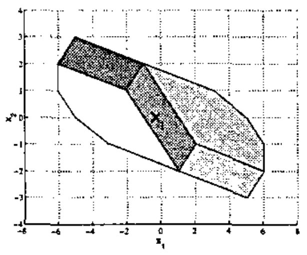
(a) Iteration $iter = 0$, $(\mathcal{X}_f^1)^0 = \mathcal{X}_{\text{LQR}}$.



(b) Iteration $iter = 1$, $T_{\text{set}} = (\mathcal{X}_f^1)^0 = \mathcal{X}_{\text{LQR}}$, $R = 5$ regions.



(c) Iteration $iter = 2$, $T_{\text{set}} = (\mathcal{X}_f^1)^1$, $R = 5$ regions.



(d) Iteration $iter = 3$, $T_{\text{set}} = (\mathcal{X}_f^1)^2$, $R = 5$ regions.

Fig. 1. Region partitions obtained with Algorithm 1 for a generic double-integrator system. Since $\mathcal{K}_\infty(\mathcal{X}_{\text{LQR}}) = (\mathcal{X}_f^1)^3$, the controller obtained with algorithm 1 consists of $R = 16$ regions.

where $\bar{T}_{\text{set}}(n, m, N)$ denotes the average number of facets of the terminal set over all iterations, which is a function of the number of states, inputs and prediction horizon. Note that the prediction horizon has moved from the exponent to the base, thus reducing the influence of the horizon on complexity. It is difficult to gauge the number of facets of $\bar{T}_{\text{set}}(n, m, N)$. In theory, $\bar{T}_{\text{set}}(n, m, N)$ may be exponentially dependent on the prediction horizon N. However, as Section VI will show, in practice the complexity of this quantity remains limited due to the constraints on state and input.

B. On-Line Application

Due to the nature of the algorithm, several regions may overlap such that a given state x is contained in multiple regions. In order to guarantee stability and feasibility, the feedback law associated to the region computed at the smallest iteration number is selected, i.e.,

$$iter_{\min} = \min_{iter} iter, \text{ s.t. } x \in (\mathcal{X}_f^1)^{iter}, \tag{11a}$$

$$u = F_r^{iter_{\min}} x + G_r^{iter_{\min}}, \text{ s.t.} x \in \mathcal{P}_r^{iter_{\min}}. \tag{11b}$$

Note that this two-level procedure is far more efficient than checking set-membership (11b) for each region as in [2].

Theorem 2. *The controller computed according to Algorithm 1 and applied as in (11) covers the set $\mathcal{K}_\infty(\mathcal{X}_{\text{LQR}})$, stabilizes the system (1) and guarantees satisfaction of the constraints in (2) for all time.*

Proof. The controller covers $\mathcal{K}_\infty(\mathcal{X}_{\text{LQR}})$ since at the final iteration $(\mathcal{X}_f^1)^{iter} = T_{\text{set}}$ for (10)[8]. Equations (10) and (11) guarantee that any states will enter a control invariant region $\mathcal{X}_{\text{LQR}}$ around the origin in $iter_{min}$ steps. For the set $\mathcal{X}_{\text{LQR}}$ the LQR feedback controller stabilizes the system (Definition 3). Thus, stability and feasibility of the closed-loop system is guaranteed. ∎

V. Stability Analysis with LMIs

Even though the procedure in section IV provides a simple controller with stability and feasibility guarantees, an even simpler controller may be obtained by additionally applying the analysis procedure in [3] which is based on [7]. The procedure aims at identifying a piecewise quadratic Lyapunov function which proves stability for a PWA controller partition. The partition at the final iteration of Algorithm 1 is control invariant and therefore feasibility is guaranteed for all time. If stability for this partition can be shown as well, it is no longer necessary to store the partitions at each iteration, since the final controller guarantees stability and feasibility. The complexity of the resulting controller is significantly lower. However, there is no a priori guarantee that the LMI methods will provide the necessary control Lyapunov function, since the LMI formulation is merely sufficient. Therefore we propose the following algorithm:

Algorithm 2.

1) *Compute a controller partition according to Algorithm 1 and extract the controller partition $\{\mathcal{P}_r\}_1^R$ corresponding to the last iteration.*
2) *Solve an LMI as in [3] to find a piecewise quadratic Lyapunov function for the closed-loop system.*
3) *If a Lyapunov function was found, apply the controller $\{\mathcal{P}_r\}_1^R$ obtained in step 1. Otherwise resort to the results of Algorithm 1 and apply them as described in Section IV-B.*

Theorem 3. *The controller obtained with Algorithm 2 stabilizes the system (1), guarantees constraint satisfaction (2) for all time and guarantees that any state $x \in \mathcal{K}_\infty(\mathcal{X}_{\text{LQR}})$ will enter $\mathcal{X}_{\text{LQR}}$ in finite time.*

Proof. The origin is contained in $\mathcal{X}_{\text{LQR}}$ and exponential stability is guaranteed by the Lyapunov function in [3]. Therefore the Lyapuonv function decrease is strictly smaller zero and finitely determined (i.e., not arbitrarily small) for all states $x \in (\mathcal{K}_\infty(\mathcal{X}_{\text{LQR}}) \setminus \mathcal{X}_{\text{LQR}})$, which guarantees that the state will enter $\mathcal{X}_{\text{LQR}}$ in finite time. The rest of the proof follows from Theorem 2. ∎

VI. Numerical Results

In this section we will compare the complexity decrease and degradation in performance incurred by Algorithms 1 and 2 with the infinite horizon optimal controller in [5] based on 20 random stable systems for $n = 2,3,4$ states and $m = 2$ inputs (total of 60 systems). The inputs for all systems were constrained to $-1 \leq u_{1,2} \leq 1$ and the states were limited to $-10 \leq x_i \leq 10$ ($i = 1,2,3,4$). The system dynamics are given by (1). Two different performance objectives (3) were considered: small and large weights on the input, i.e., $\mathcal{R}_1 = 0.1I$ and $\mathcal{R}_2 = 10I$. $\mathcal{Q} = I$ was used throughout. Performance was measured by gridding the state space and computing the closed-loop trajectory cost to the origin. As can be gathered from Figures 2 and 3, the decrease in controller complexity is enormous (e.g. Figure 2(c), system 9: $R = 22529$ for [5] vs. $R = 2021$ for Alg. 1 vs. $R = 85$ for Alg. 2). Despite the conservative formulation, the LMI analysis in Algorithm 2 always succeeded in finding a Lyapunov function. On average, Algorithm 1 decreases complexity relative to [5] by a factor of about 10 and Algorithm 2 by a factor of about 100. As can be concluded from Figure 2, the improvement is more significant for larger problems. At the same time, the average decrease in closed-loop performance is below 1%. It may seem surprising that the performance degradation is greater if the weight on the input is small (see Figure 3). Small weights on the inputs would seem to indicate that the minimum time controller in Algorithm 1 is optimal. However, small weights on the input inherently result in smaller invariant sets $\mathcal{X}_{\text{LQR}}$, where the input sequence obtained with Algorithm 1 is optimal, thus the degradation in performance. For large weights on the input, there is essentially no degradation incurred with Algorithm 2 because expensive control action implies that the trajectory is almost completely defined by constraint satisfaction. It is also remarkable to note that a change in the cost matrix $\mathcal{R}$ may reduce the number of controller regions by a factor of up to 10 (see Fig. 2(c) and 2(f), System 9).

VII. Conclusion

A complexity analysis of explicit optimal control solvers was presented, emphasizing the strong influence of horizon length and the number of inputs on complexity. Based on this analysis, two novel algorithms whose complexity is orders of magnitude lower than all comparable algorithms [2], [12], [5] is presented. The extensive set of examples clearly indicates that the proposed algorithm is significantly faster than all other algorithms to date. The price for this reduced complexity is an average decrease in closed-loop performance by less than 1%. Furthermore, stability and feasibility guarantees are retained by the algorithms. The results can be extended to uncertain systems with the methods in [6].

The presented algorithms can be downloaded from: `http://control.ee.ethz.ch/~grieder/pg_downloads.msql`

VIII. References

[1] M. Baotic. An efficient algorithm for multi-parametric quadratic programming. Technical Report AUT02-04, Automatic Control Laboratory, ETH Zurich, Switzerland, February 2002.

[2] 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.

[3] G. Ferrari-Trecate, F.A. Cuzzola, D. Mignone, and M. Morari. Analysis of discrete-time piecewise affine and hybrid systems. *Automatica*, 38:2139–2146, 2002.

[4] E.G. Gilbert and K. Tin 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.

[5] 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.

[6] P. Grieder, P. Parillo, and M. Morari. Robustness of receding horizon control. In *Proc. 42th IEEE Conf. on Decision and Control*, Maui, Hawaii, USA, December 2003.

[7] M. Johannson and A. Rantzer. Computation of piecewise quadratic Lyapunov functions for hybrid systems. *IEEE Trans. Automatic Control*, 43(4):555–559, 1998.

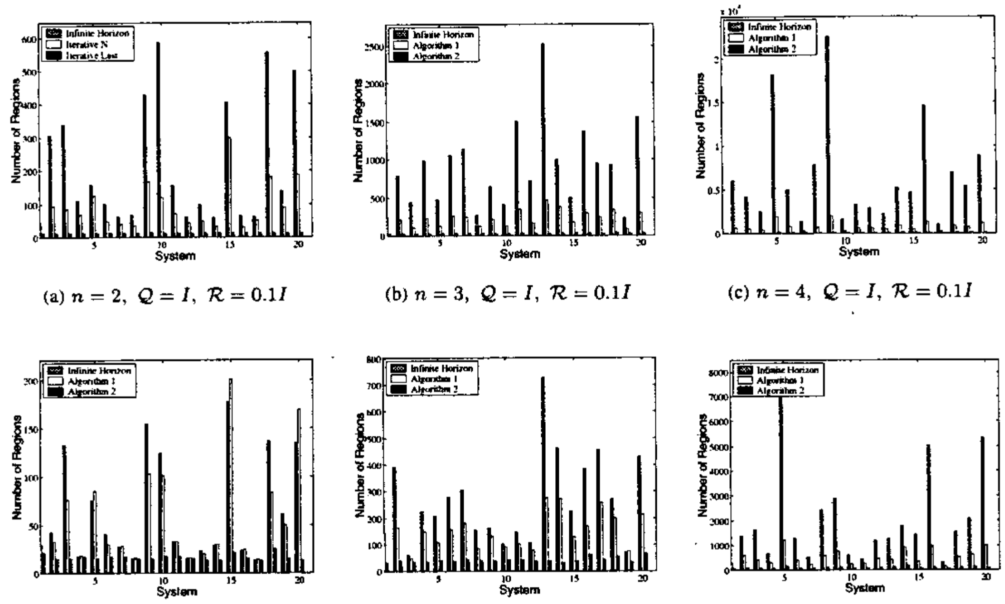
[8] E. C. Kerrigan. *Robust Constraints Satisfaction: Invariant Sets and Predictive Control.* PhD thesis, Department of Engineering, The University of Cambridge, Cambridge, England, 2000.

[9] 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.

[10] D.Q. Mayne and W. R. Schroeder. Robust time-optimal control of constrained linear systems. *aut*, 33(12):2103–2118, 1997.

[11] M. Sznaier and M.J. Damborg. Suboptimal control of linear systems with state and control inequality constraints. In *Proc. 26th IEEE Conf. on Decision and Control*, volume 1, pages 761–762, December 1987.

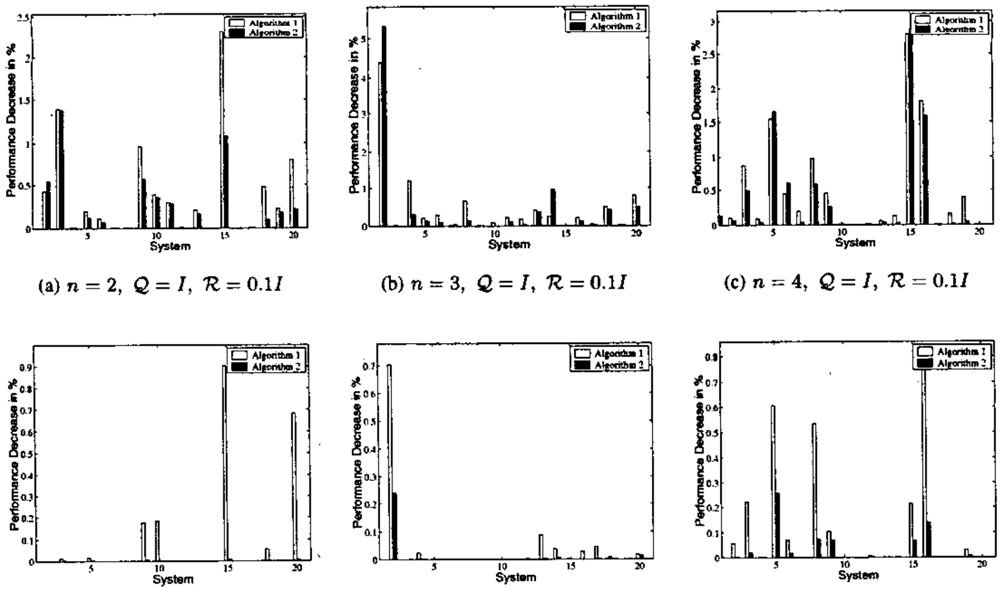
[12] P. Tøndel, T.A. Johansen, and A. Bemporad. An algorithm for multi-parametric quadratic programming and explicit MPC solutions. In *Proc. 40th IEEE Conf. on Decision and Control*, December 2001.



(a) $n = 2,\ \mathcal{Q} = I,\ \mathcal{R} = 0.1I$ (b) $n = 3,\ \mathcal{Q} = I,\ \mathcal{R} = 0.1I$ (c) $n = 4,\ \mathcal{Q} = I,\ \mathcal{R} = 0.1I$

(d) $n = 2,\ \mathcal{Q} = I,\ \mathcal{R} = 10I$ (e) $n = 3,\ \mathcal{Q} = I,\ \mathcal{R} = 10I$ (f) $n = 4,\ \mathcal{Q} = I,\ \mathcal{R} = 10I$

Fig. 2. Complexity reduction: Results for 20 random systems with $n = 2, 3, 4$ states and $m = 2$ inputs with varying the cost objective in (3).



(a) $n = 2,\ \mathcal{Q} = I,\ \mathcal{R} = 0.1I$ (b) $n = 3,\ \mathcal{Q} = I,\ \mathcal{R} = 0.1I$ (c) $n = 4,\ \mathcal{Q} = I,\ \mathcal{R} = 0.1I$

(d) $n = 2,\ \mathcal{Q} = I,\ \mathcal{R} = 10I$ (e) $n = 3,\ \mathcal{Q} = I,\ \mathcal{R} = 10I$ (f) $n = 4,\ \mathcal{Q} = I,\ \mathcal{R} = 10I$

Fig. 3. Performance decrease vs. [5]: Results for 20 random systems with $n = 2, 3, 4$ states and $m = 2$ inputs with varying the cost objective in (3).

to x and θ. Furthermore, $A\widehat{x}(\theta) + B\theta + d = \sum_{k=0}^{m} \mu_k(Ax^k + B\theta^k + d) = 0$. To prove (5), we note that $\overline{V}(\theta) = \sum_{k=0}^{m} \mu_k V^*(\theta^k) = \sum_{k=0}^{m} \mu_k f(x^k, \theta^k) \geq f(\sum_{k=0}^{m} \mu_k x^k, \sum_{k=0}^{m} \mu_k \theta^k) = f(\widehat{x}(\theta), \theta) = \widehat{V}(\theta)$, where the inequality follows from the joint convexity of f, and $\widehat{V}(\theta) = f(\widehat{x}(\theta), \theta) \geq f(x^*(\theta), \theta) = V^*(\theta)$, where $x^*(\theta)$ denote any optimizer of (CP_θ). □

Thus, $\overline{V}$ and $\widehat{V}$ are both upper bounds of V^* on S, tight on every vertex on S, where $\overline{V}$ is affine in θ and $\widehat{V}$ is tighter than $\overline{V}$.

3.2 Lower Bounds on the Value Function

Assuming that a subgradient of V^* is available at every vertex of S, we can construct a piecewise affine lower bound of V^*. More precisely, let s^k be a subgradient of V^* at θ^k $(k = 0, 1, \ldots, m)$. Since V^* is convex, we have $V^*(\theta) \geq V^*(\theta^k) + (s^k)'(\theta - \theta^k)$. As a consequence, we define $\underline{V}(\theta) \triangleq \max_{k=0,1,\ldots,m}\{V^*(\theta^k) + (s^k)'(\theta - \theta^k)\}$. By construction,

$$\underline{V}(\theta) \leq V^*(\theta) \quad \text{for all } \theta \in S, \tag{6}$$

so that $\underline{V}$ is a piecewise affine lower bound on V^* inside S, tight at every vertex of the simplex.

We assume the following:

Assumption 1 *Functions f and g_i $(i = 1, \ldots, p)$ are differentiable with respect to both x and θ inside their domain.*

For convenience, let $g(x, \theta) \triangleq [g_1(x, \theta), \ldots, g_p(x, \theta)]'$. The Karush-Kuhn-Tucker optimality conditions for problem (CP_θ) are (see, e.g., [6, Chapter 5]):

$$\begin{array}{l} g(x,\theta) \leq 0, \quad Ax + B\theta + d = 0, \\ \lambda \geq 0, \quad \lambda' g(x,\theta) = 0, \\ \nabla_x f(x,\theta) + J_x g(x,\theta)'\lambda + A'\nu = 0, \end{array} \tag{7}$$

where $\lambda \in \mathbb{R}^p$ and $\nu \in \mathbb{R}^q$ are the vectors of Lagrange multipliers, $\nabla_x f(x, \theta) \in \mathbb{R}^n$ denotes the gradient of f with respect to x, and $J_x g(x, \theta)$ denotes the $p \times n$ Jacobian matrix of the partial derivatives of g with respect to x.

Proposition 3 *Let (x^k, λ^k, ν^k) be a solution of (7) for $\theta = \theta^k$, for any $k = 0, 1, \ldots, m$. Then*

$$s^k \triangleq \nabla_\theta f(x^k, \theta^k) + J_\theta g(x^k, \theta^k)'\lambda^k + B'\nu^k$$

is a subgradient of V^* of (CP_θ) at θ^k, where $\nabla_\theta f(x, \theta) \in \mathbb{R}^m$ denotes the gradient of f with respect to θ and $J_\theta g(x, \theta)$ denotes the $p \times m$ Jacobian matrix of partial derivatives of g with respect to θ.

Proof: See [2]. □

A similar result was shown by Fiacco [8, Chapter 9] using an auxiliary lower-bounding multiparametric linear programming problem.

In case a primal-dual method is used for computing $V^*(\theta^k)$, both optimal primal variables x^k and Lagrange multipliers λ^k, ν^k are available. If also the derivatives of f and g_i are available, then a subgradient s^k valid at θ^k, and therefore a linear lower bound on V^*, can be immediately constructed according to Proposition 3.

3.3 Error Estimates Inside a Simplex

We wish to approximate V^* by using $\widehat{V}$ inside the simplex S, with vertices θ^k, $k = 0, 1, \ldots, m$. In this way, the maximum absolute error we introduce is

$$\epsilon^{MAX}(S) \triangleq \max_\theta\{\widehat{V}(\theta) - V^*(\theta) : \theta \in S\}.$$

Unfortunately, the above optimization problem is a nonconvex DC programming problem, and thus the exact evaluation of $\epsilon^{MAX}(S)$ is, in general, hard [12]. For this reason, we analyze two practically computable upper bounds on $\epsilon^{MAX}(S)$.

Proposition 4 *Let $s^k \in \mathbb{R}^m$ be a subgradient of V^* at θ^k, and let $w_k \triangleq -V^*(\theta^k) - (s^k)'\theta^k$, for all $k = 0, 1, \ldots, m$. Define*

$$\epsilon^{LP}(S) \triangleq \left\{ \begin{array}{ll} \max_x & \overline{V}(\theta) - t \\ \text{s. t.} & (s^k)'\theta - t \leq w_k \quad (k = 0, \ldots, m) \\ & -M^{-1}\binom{1}{\theta} \leq 0, \end{array} \right. \tag{8}$$

where $M = \left[\begin{smallmatrix} 1 & 1 & \cdots & 1 \\ \theta^0 & \theta^1 & \cdots & \theta^m \end{smallmatrix}\right]$, and $\theta^0, \ldots, \theta^m$ are the vertices of S. Then, $V^*(\theta) \geq \widehat{V}(\theta) - \epsilon^{LP}(S)$, $\forall \theta \in S$.

Proof: We have $\max_\theta\{\widehat{V}(\theta) - V^*(\theta) : \theta \in S\} \leq \max_\theta\{\overline{V}(\theta) - \underline{V}(\theta) : \theta \in S\} = \max_\theta\{\overline{V}(\theta) - \max_k\{V^*(\theta^k) + (\delta^k)'(\theta - \theta^k) : k = 0, 1, \ldots, m\} : \theta \in S\} = \max_{\theta,t}\{\overline{V}(\theta) - t : t \geq V^*(\theta^k) + (\delta^k)'(\theta - \theta^k)\ (k = 0, 1, \ldots, m), \theta \in S\} = \epsilon^{LP}(S)$. □

Proposition 5 *Let*

$$\epsilon^{CP}(S) \triangleq \left\{ \begin{array}{ll} \max_{x,\theta} & \overline{V}(\theta) - f(x,\theta) \\ \text{s. t.} & g(x,\theta) \leq 0 \\ & Ax + B\theta + d = 0 \\ & -M^{-1}\binom{1}{\theta} \leq 0, \end{array} \right. \tag{9}$$

where M is defined as in Proposition 4. Then, $V^*(\theta) \geq \widehat{V}(\theta) - \epsilon^{CP}(S)$, $\forall \theta \in S$.

Proof: Let $F(\theta) \triangleq \{x \in \mathbb{R}^n : g(x, \theta) \leq 0, Ax + B\theta + d = 0\}$ (i.e., $F(\theta)$ is the feasible set of (CP_θ)). We have: $\epsilon^{MAX}(S) = \max_\theta\{\widehat{V}(\theta) - V^*(\theta) : \theta \in S\} \leq \max_\theta\{\overline{V}(\theta) - V^*(\theta) : \theta \in S\} = \max_\theta\{\overline{V}(\theta) - \min_x\{f(x, \theta) : x \in F(\theta)\} : \theta \in S\} = \max_\theta\{\overline{V}(\theta) + \max_x\{-f(x, \theta) : x \in F(\theta)\} : \theta \in S\} = \max_{x,\theta}\{\overline{V}(\theta) - f(x, \theta) : x \in F(\theta), \theta \in S\} = \epsilon^{CP}(S)$. □

Proposition 6 *For all simplices $S \subseteq \Theta_f$, $\epsilon^{LP}(S) \geq \epsilon^{CP}(S) \geq \epsilon^{MAX}(S)$.*

Proof: From the proofs of Proposition 4 and Proposition 5, and from (6), we have: $\epsilon^{LP}(S) = \max_\theta\{\overline{V}(\theta) - \underline{V}(\theta) : \theta \in S\} \geq \max_\theta\{\overline{V}(\theta) - V^*(\theta) : \theta \in S\} = \epsilon^{CP}(S)$, $\forall S \subseteq \Theta_f$. □

Proposition 6 shows that both ϵ^{LP} and ϵ^{CP} are upper bounds on ϵ^{MAX}. Computing ϵ^{LP} involves solving of a linear program with $m + 1$ variables, whereas computing ϵ^{CP} involves solving a convex program with $m + n$ variables. However, obtaining the subgradients used to compute ϵ^{LP} may require an additional effort, unless the parametric program takes some special form. In the next section we provide a recursive approximation algorithm for problem (CP_θ) that only makes use of ϵ^{CP}.

multiparametric quadratic programs). The algorithm applies to the general framework of convex multiparametric programming, and may conveniently fitted for special cases of relevant interest. In particular, the case of multiparametric semidefinite programming is briefly examined and exemplified on a test example.

The goal of our approach is to open up the application of explicit receding horizon techniques to several robust model predictive control schemes based on convex optimization.

2 Convex Multiparametric Programming

Consider the convex multiparametric program

$$(CP_\theta) \qquad \begin{array}{ll} \min_x & f(x,\theta) \\ \text{s. t.} & g_i(x,\theta) \leq 0 \quad (i=1,\dots,p) \\ & Ax + B\theta + d = 0 \end{array}$$

where $x \in \mathbb{R}^n$ are the decision variables, $\theta \in \mathbb{R}^m$ are the parameters, $f : \mathbb{R}^n \times \mathbb{R}^m \mapsto \mathbb{R}$ is the objective function, $g_i : \mathbb{R}^n \times \mathbb{R}^m \mapsto \mathbb{R}$, for all $i = 1,\dots,p$, A is a $q \times n$ real matrix, B is a $q \times m$ real matrix, and $d \in \mathbb{R}^q$. We assume that f and g_i $(i = 1,\dots,p)$ are jointly convex in both the variables and the parameters. Clearly, the left-hand side of each equality constraint is jointly affine in both the variables and the parameters. We are interested in characterizing the solution of problem (CP_θ) for a given polytopic set of parameters

$$\Theta = \{\theta \in \mathbb{R}^m : \ Q\theta \leq R\} \subset \mathbb{R}^m.$$

The solution of problem (CP_θ) is defined as follows. The *feasible parameter set* Θ_f is the set of all $\theta \in \Theta$ for which the corresponding problem (CP_θ) admits a solution, i.e., there exists a vector x satisfying the constraints of (CP_θ). The *value function* $V^\star : \Theta_f \mapsto \mathbb{R}$ is the function that associates with every $\theta \in \Theta_f$ the corresponding unique optimal value of (CP_θ). The *optimizer set function* $X^\star : \Theta_f \mapsto 2^{\mathbb{R}^n}$ is the function that associates to a parameter vector $\theta \in \Theta_f$ the corresponding set of optimizers $X^\star(\theta) = \{x \in \mathbb{R}^n : f(x,\theta) = V^\star(\theta)\}$ of problem (CP_θ). An *optimizer function* $x^\star : \Theta_f \mapsto \mathbb{R}^n$ is a function that associates to a parameter vector $\theta \in \Theta_f$ (one of) the optimizer(s) $x^\star(\theta) \in X^\star(\theta)$.

The following basic result for convex multiparametric programming was proved in [16, Lemma 1] in the absence of equality constraints:

Lemma 1 *Consider the multiparametric problem (CP_θ) and let f, g_i be jointly convex functions of (x,θ), for all $i = 1,\dots,p$. Then, Θ_f is a convex set and $V^\star$ is a convex function of θ.*

The result can be easily generalized to the presence of linear equality constraints.

Hereafter we assume that Θ_f and Θ are full-dimensional sets. A numerical test for verifying such an assumption will be provided in Section 4.

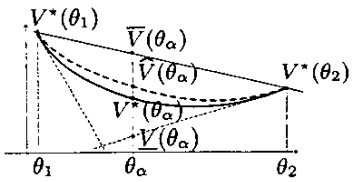


Figure 1: Approximation of the value function in convex parametric programming: the scalar case

3 Approximate Multiparametric Programming

3.1 Upper Bounds on the Value Function

Let $\theta^0, \theta^1, \dots, \theta^m \in \mathbb{R}^m$ be affinely independent points in Θ_f, and define S as the following m-dimensional simplex:

$$S \triangleq \{\theta \in \mathbb{R}^m : \theta = \textstyle\sum_{k=0}^m \mu_k \theta^k, \ \sum_{k=0}^m \mu_k = 1, \ \mu_k \geq 0 \ (k = 0,1,\dots,m)\}. \tag{1}$$

Let x^k be an optimizer of (CP_{θ^k}), for all $k = 0,1,\dots,m$; define the matrices

$$M \triangleq \begin{bmatrix} 1 & 1 & \cdots & 1 \\ \theta^0 & \theta^1 & \cdots & \theta^m \end{bmatrix}, \ X \triangleq \begin{bmatrix} x^0 & x^1 & \cdots & x^m \end{bmatrix}, \tag{2}$$

and note that by construction M is nonsingular. A proof of the following simple result can be found in [9].

Proposition 1 *The system of linear inequalities $M^{-1}\binom{1}{\theta} \geq 0$ is a minimal representation of S.*

In the following, we introduce upper and lower bounds on $V^\star$ inside S. Such bounds generalize to the multidimensional case the concepts introduced by Fiacco [8, Chapter 9] to bound the value function of a parametric convex program inside a line segment (cf. also [13]).

Define the vector $v \triangleq [V^\star(\theta^0) \ V^\star(\theta^1) \ \cdots \ V^\star(\theta^m)]'$, and the function

$$\widehat{x}(\theta) \triangleq XM^{-1}\binom{1}{\theta}. \tag{3}$$

Furthermore, define

$$\widehat{V}(\theta) \triangleq f(\widehat{x}(\theta),\theta), \tag{4}$$

and $\overline{V}(\theta) \triangleq v'M^{-1}\binom{1}{\theta}$. Note that both $\widehat{x}$ and $\overline{V}$ are affine functions of θ.

Proposition 2 *For all $\theta \in S$ the vector $\widehat{x}(\theta)$ is a feasible solution of (CP_θ), and*

$$\overline{V}(\theta) \geq \widehat{V}(\theta) \geq V^\star(\theta). \tag{5}$$

Proof: We first prove that $\widehat{x}(\theta)$ is feasible. The vector $\mu = M^{-1}\binom{1}{\theta}$ is the unique solution of $M\mu = \binom{1}{\theta}$. Thus, if $\theta \in S$ then $\mu \geq 0$, $\sum_{k=0}^m \mu_k = 1$, $\theta = \sum_{k=0}^m \mu_k \theta^k$, and $\widehat{x}(\theta) = \sum_{k=0}^m \mu_k x^k$. As a consequence, for all $i = 1,\dots,p$, $g_i(\widehat{x}(\theta),\theta) = g_i(\sum_{k=0}^m \mu_k x^k, \sum_{k=0}^m \mu_k \theta^k) \leq \sum_{k=0}^m \mu_k g_i(x^k,\theta^k) \leq 0$, where the first inequality follows from the joint convexity of g_i with respect to x and θ. Furthermore, $A\widehat{x}(\theta) + B\theta + d = \sum_{k=0}^{m} \mu_k(Ax^k + B\theta^k + d) = 0$. To prove (5), we note that $\overline{V}(\theta) = \sum_{k=0}^{m} \mu_k V^\star(\theta^k) = \sum_{k=0}^{m} \mu_k f(x^k, \theta^k) \geq f(\sum_{k=0}^{m} \mu_k x^k, \sum_{k=0}^{m} \mu_k \theta^k) = f(\widehat{x}(\theta), \theta) = \widehat{V}(\theta)$, where the inequality follows from the joint convexity of f, and $\widehat{V}(\theta) = f(\widehat{x}(\theta), \theta) \geq f(x^\star(\theta), \theta) = V^\star(\theta)$, where $x^\star(\theta)$ denote any optimizer of (CP_θ). □

Thus, $\overline{V}$ and $\widehat{V}$ are both upper bounds of $V^\star$ on S, tight on every vertex on S, where $\overline{V}$ is affine in θ and $\widehat{V}$ is tighter than $\overline{V}$.

3.2 Lower Bounds on the Value Function

Assuming that a subgradient of $V^\star$ is available at every vertex of S, we can construct a piecewise affine lower bound of $V^\star$. More precisely, let s^k be a subgradient of $V^\star$ at θ^k $(k = 0, 1, \dots, m)$. Since $V^\star$ is convex, we have $V^\star(\theta) \geq V^\star(\theta^k) + (s^k)'(\theta - \theta^k)$. As a consequence, we define $\underline{V}(\theta) \triangleq \max_{k=0,1,\dots,m}\{V^\star(\theta^k) + (s^k)'(\theta - \theta^k)\}$. By construction,

$$\underline{V}(\theta) \leq V^\star(\theta) \quad \text{for all } \theta \in S, \tag{6}$$

so that $\underline{V}$ is a piecewise affine lower bound on $V^\star$ inside S, tight at every vertex of the simplex.

We assume the following:

Assumption 1 *Functions f and g_i $(i = 1, \dots, p)$ are differentiable with respect to both x and θ inside their domain.*

For convenience, let $g(x, \theta) \triangleq [g_1(x, \theta), \dots, g_p(x, \theta)]'$. The Karush-Kuhn-Tucker optimality conditions for problem (CP_θ) are (see, e.g., [6, Chapter 5]):

$$\begin{array}{l} g(x,\theta) \leq 0, \quad Ax + B\theta + d = 0, \\ \lambda \geq 0, \quad \lambda' g(x,\theta) = 0, \\ \nabla_x f(x,\theta) + J_x g(x,\theta)'\lambda + A'\nu = 0, \end{array} \tag{7}$$

where $\lambda \in \mathbb{R}^p$ and $\nu \in \mathbb{R}^q$ are the vectors of Lagrange multipliers, $\nabla_x f(x, \theta) \in \mathbb{R}^n$ denotes the gradient of f with respect to x, and $J_x g(x, \theta)$ denotes the $p \times n$ Jacobian matrix of the partial derivatives of g with respect to x.

Proposition 3 *Let (x^k, λ^k, ν^k) be a solution of (7) for $\theta = \theta^k$, for any $k = 0, 1, \dots, m$. Then*

$$s^k \triangleq \nabla_\theta f(x^k, \theta^k) + J_\theta g(x^k, \theta^k)'\lambda^k + B'\nu^k$$

is a subgradient of $V^\star$ of (CP_θ) at θ^k, where $\nabla_\theta f(x, \theta) \in \mathbb{R}^m$ denotes the gradient of f with respect to θ and $J_\theta g(x, \theta)$ denotes the $p \times m$ Jacobian matrix of partial derivatives of g with respect to θ.

Proof: See [2]. □

A similar result was shown by Fiacco [8, Chapter 9] using an auxiliary lower-bounding multiparametric linear programming problem.

In case a primal-dual method is used for computing $V^\star(\theta^k)$, both optimal primal variables x^k and Lagrange multipliers λ^k, ν^k are available. If also the derivatives of f and g_i are available, then a subgradient s^k valid at θ^k, and therefore a linear lower bound on $V^\star$, can be immediately constructed according to Proposition 3.

3.3 Error Estimates Inside a Simplex

We wish to approximate $V^\star$ by using $\widehat{V}$ inside the simplex S, with vertices θ^k, $k = 0, 1, \dots, m$. In this way, the maximum absolute error we introduce is

$$\epsilon^{MAX}(S) \triangleq \max_\theta\{\widehat{V}(\theta) - V^\star(\theta) : \theta \in S\}.$$

Unfortunately, the above optimization problem is a nonconvex DC programming problem, and thus the exact evaluation of $\epsilon^{MAX}(S)$ is, in general, hard [12]. For this reason, we analyze two practically computable upper bounds on $\epsilon^{MAX}(S)$.

Proposition 4 *Let $s^k \in \mathbb{R}^m$ be a subgradient of $V^\star$ at θ^k, and let $w_k \triangleq -V^\star(\theta^k) - (s^k)'\theta^k$, for all $k = 0, 1, \dots, m$. Define*

$$\epsilon^{LP}(S) \triangleq \left\{ \begin{array}{ll} \max_x & \overline{V}(\theta) - t \\ \text{s. t.} & (s^k)'\theta - t \leq w_k \quad (k = 0, \dots, m) \\ & -M^{-1}\binom{1}{\theta} \leq 0, \end{array} \right. \tag{8}$$

where $M = \left[\begin{smallmatrix} 1 & 1 & \cdots & 1 \\ \theta^0 & \theta^1 & \cdots & \theta^m \end{smallmatrix}\right]$, and $\theta^0, \dots, \theta^m$ are the vertices of S. Then, $V^\star(\theta) \geq \widehat{V}(\theta) - \epsilon^{LP}(S)$, $\forall \theta \in S$.

Proof: We have $\max_\theta\{\widehat{V}(\theta) - V^\star(\theta) : \theta \in S\} \leq \max_\theta\{\overline{V}(\theta) - \underline{V}(\theta) : \theta \in S\} = \max_\theta\{\overline{V}(\theta) - \max_k\{V^\star(\theta^k) + (\delta^k)'(\theta - \theta^k) : k = 0, 1, \dots, m\} : \theta \in S\} = \max_{\theta,t}\{\overline{V}(\theta) - t : t \geq V^\star(\theta^k) + (\delta^k)'(\theta - \theta^k)\ (k = 0, 1, \dots, m), \theta \in S\} = \epsilon^{LP}(S)$. □

Proposition 5 *Let*

$$\epsilon^{CP}(S) \triangleq \left\{ \begin{array}{ll} \max_{x,\theta} & \overline{V}(\theta) - f(x,\theta) \\ \text{s. t.} & g(x,\theta) \leq 0 \\ & Ax + B\theta + d = 0 \\ & -M^{-1}\binom{1}{\theta} \leq 0, \end{array} \right. \tag{9}$$

where M is defined as in Proposition 4. Then, $V^\star(\theta) \geq \widehat{V}(\theta) - \epsilon^{CP}(S)$, $\forall \theta \in S$.

Proof: Let $F(\theta) \triangleq \{x \in \mathbb{R}^n : g(x, \theta) \leq 0, Ax + B\theta + d = 0\}$ (i.e., $F(\theta)$ is the feasible set of (CP_θ)). We have: $\epsilon^{MAX}(S) = \max_\theta\{\widehat{V}(\theta) - V^\star(\theta) : \theta \in S\} \leq \max_\theta\{\overline{V}(\theta) - V^\star(\theta) : \theta \in S\} = \max_\theta\{\overline{V}(\theta) - \min_x\{f(x, \theta) : x \in F(\theta)\} : \theta \in S\} = \max_\theta\{\overline{V}(\theta) + \max_x\{-f(x, \theta) : x \in F(\theta)\} : \theta \in S\} = \max_{x,\theta}\{\overline{V}(\theta) - f(x, \theta) : x \in F(\theta), \theta \in S\} = \epsilon^{CP}(S)$. □

Proposition 6 *For all simplices $S \subseteq \Theta_f$, $\epsilon^{LP}(S) \geq \epsilon^{CP}(S) \geq \epsilon^{MAX}(S)$.*

Proof: From the proofs of Proposition 4 and Proposition 5, and from (6), we have: $\epsilon^{LP}(S) = \max_\theta\{\overline{V}(\theta) - \underline{V}(\theta) : \theta \in S\} \geq \max_\theta\{\overline{V}(\theta) - V^\star(\theta) : \theta \in S\} = \epsilon^{CP}(S)$, $\forall S \subseteq \Theta_f$. □

Proposition 6 shows that both ϵ^{LP} and ϵ^{CP} are upper bounds on ϵ^{MAX}. Computing ϵ^{LP} involves solving of a linear program with $m + 1$ variables, whereas computing ϵ^{CP} involves solving a convex program with $m + n$ variables. However, obtaining the subgradients used to compute ϵ^{LP} may require an additional effort, unless the parametric program takes some special form. In the next section we provide a recursive approximation algorithm for problem (CP_θ) that only makes use of ϵ^{CP}.

4 An Approximate Multiparametric Solver

We are in a position to state a basic approximation algorithm for (CP_θ). The algorithm consists of two phases. In the initialization phase, we test if the set Θ_f of feasible parameters is full dimensional. If this is the case, a polyhedral inner approximation of the set Θ_f of feasible parameters is obtained and subdivided into a collection of simplices. In the second phase, each simplex is recursively subdivided into smaller simplices until the desired degree of accuracy in approximating the value function is reached in each simplex.

We start by describing the recursive phase. Let S be a full-dimensional simplex contained in Θ_f, defined as in (1), and let $\widehat{x}$ and $\widehat{V}$ be defined as in (3) and (4) respectively. Let $\epsilon > 0$ be a given maximum error. The following Algorithm 4.1 summarizes the recursive steps.

```
1. Build $M$ and $X$ as defined in (2);
2. if $M$ is nonsingular then
    2.1. Solve problem (9), getting $\epsilon^{CP}(S)$ and $(\overline{x},\overline{\theta})$;
    2.2. if $\epsilon^{CP}(S) > \epsilon$ then
        2.2.1. for $k = 0,1,\ldots,m$ do
            2.2.1.1 Replace the $k$th vertex of $S$ by $\overline{\theta}$ and let $S_k$
                    be the new simplex;
            2.2.1.2 Call this algorithm on $S_k$;
    2.3. else Return $M^{-1}$ and $X$;
```

Algorithm 4.1: Recursive splitting of an initial simplex and approximation of the value function and of the optimizer. Note that $\widehat{x}$ and $\widehat{V}$ are readily evaluated by using the returned matrices.

In Algorithm 4.1, we used $\epsilon^{CP}(S)$ for convenience. However, we remark that any of the upper bounds on the error described in the previous section may be used in the same way. Note that, at each recursive iteration, the current simplex is split into at most $m+1$ full-dimensional simplices with nonoverlapping interiors.

Algorithm 4.1 builds up a piecewise affine function $\widehat{x} : S \mapsto \mathbb{R}^n$ and a piecewise analytical function $\widehat{V} : S \mapsto \mathbb{R}$ such that: (i) $\widehat{x}(\theta)$ is a feasible solution of (CP_θ) for all $\theta \in S$, (ii) $\widehat{V}(\theta) = f(\widehat{x}(\theta),\theta)$ for all $\theta \in S$, and (iii) $0 \leq \widehat{V}(\theta) - V^\star(\theta) \leq \epsilon$ for all $\theta \in S$. Note that $\widehat{x}$ and $\widehat{V}$ may not be continuous on the boundary of the returned simplices. As a consequence, the approximate optimizer and value function may be defined more than once for some $\theta \in S$, though this fact can happen only on a subset of S with null measure.

By using $\epsilon^{CP}(S)$ (or even $\epsilon^{LP}(S)$), the proposed method controls the absolute error on the value function with respect to $\overline{V}$, which constitutes an approximation of $V^\star$ worse than the actually returned $\widehat{V}$. As a consequence, there may be cases where a simplex is split because $\epsilon^{CP}(S) > \epsilon$ though the maximum difference between $\widehat{V}$ and $V^\star$ is less than the prescribed ϵ. In order to possibly avoid unnecessary splits, consider the error quantity $\underline{\epsilon}(S) \triangleq \widehat{V}(\overline{\theta}) - V^\star(\overline{\theta}) = f(\widehat{x}(\overline{\theta}),\overline{\theta}) - V^\star(\overline{\theta}) \leq \epsilon^{MAX}(S)$, where $\epsilon^{MAX}(S)$ is the maximum absolute error on S. If $\underline{\epsilon}(S) > \epsilon$ then clearly $\epsilon^{MAX}(S) > \epsilon$ and hence the simplex S must be split. On the other hand, when $\underline{\epsilon}(S) \leq \epsilon$ there is the possibility that the actual error $\epsilon^{MAX}(S)$ is smaller than ϵ. A technique based on a piecewise linear approximation of $\widehat{V}$ over S for estimating $\epsilon^{MAX}(S)$ with an arbitrary precision is described in [2].

Remark 4.1 If in every recursive call vector $\overline{\theta}$ lies in the interior of its simplex, then $\widehat{x}$ and $\widehat{V}$ are both continuous functions of the parameter θ. If the continuity property is required, we may force the above condition by imposing in (9) the tighter constraint $M^{-1}\binom{1}{\theta} \geq \sigma e$, where σ is a comparatively small positive scalar and $e \in \mathbb{R}^{m+1}$ is a vector of ones. This is equivalent to letting $\mu_k \geq \sigma > 0$ for all $k = 0,1,\ldots,m$, where μ_k are the coefficients of the convex combination of the vertices of the simplex. As an alternative, in order to enforce continuity and obtain a geometric balance, one may always decide to split S in its center $\frac{1}{m+1}\sum_{k=0}^{m}\theta^k$.

4.1 Initialization

So far, we have assumed that Θ_f is a full-dimensional set. This assumption can be verified as follows. First of all, a necessary condition for Θ_f to be full dimensional is that the equality constraints $Ax + B\theta = d$ do not restrict θ to lie on a lower-dimensional affine subspace of $\mathbb{R}^m$ (i.e., the set $\{\theta \in \mathbb{R}^m : \exists x \in \mathbb{R}^n : Ax + B\theta + d = 0\}$ has dimension m). This can be easily verified by computing a Gauss reduction of $[A\ B\ d]$ and then checking if equality constraints of the form $a'\theta = \alpha$ appear with $a \neq 0 \in \mathbb{R}^m$. Assuming that the linear constraints $Ax+B\theta+d = 0$ do not reduce the dimension of Θ_f, let $S(\theta,\rho) = \text{conv}(\theta+\rho e^0, \theta+\rho e^1, \ldots, \theta+\rho e^m)$, where "conv" denotes the convex hull, e^j is the jth column of the m-by-m identity matrix, $j = 1,\ldots,m$, and $e^0 = 0 \in \mathbb{R}^m$. We determine the largest simplex $S(\theta,\rho)$ contained in Θ_f, by solving

$$\begin{array}{rl}\max\limits_{\theta,\rho,y^0,\ldots,y^m} & \rho \\ \text{s. t.} & g(y^k,\theta+\rho e^k) \leq 0, \\ & Ay^k + B(\theta+\rho e^k) = d, \ (k = 0,\ldots,m) \\ & Q(\theta+\rho e^k) \leq R \end{array} \tag{10}$$

which is a convex program in $(m+1)(n+1)$ variables. Then Θ_f is full-dimensional if and only if the optimal value $\rho^\star$ is strictly positive (the volume of the largest simplex being $(\rho^\star)^m/m! > 0$).

Once the full-dimensionality of Θ_f is tested, we determine an inner polyhedral approximation $\widehat{\Theta}_f$ through a "ray-shooting" procedure, described as follows. Let $r^0, r^1 \ldots, r^{t+m}$ be $m+t+1$ directions in $\mathbb{R}^m$, $t \geq 0$, such that the convex positive cone $C = \{\theta \in \mathbb{R}^m : \theta = \sum_{i=0}^{m+t}\mu_i r^i,\ \mu_i \geq 0\} = \mathbb{R}^m$. For instance, r^i may be obtained by collecting uniformly distributed samples of the unit hyper-sphere. For each $i = 0,\ 1,\ \ldots,\ m+t$, solve the convex problem $\max\limits_{x,\theta}\{(r^i)'\theta : g(x,\theta) \leq 0, Ax + B\theta + d = 0, Q\theta \leq R\}$, and let (x^i,θ^i) be the obtained optimal solution. Set $\widehat{\Theta}_f \triangleq \text{conv}(\theta^0, \theta^1, \dots, \theta^{m+t})$, and discard redundant[1] vectors θ^i. For simplicity of notation, we assume that $\theta^{m+h+1}, \dots, \theta^{m+t}$ are the redundant vertices, where $h \le t$ and $h \ge 0$ because Θ_f is full dimensional. So, let $\theta^0, \theta^1, \dots, \theta^{m+h}$ be the vertices of $\widehat{\Theta}_f$, and assume that they are lexicographically ordered, i.e., $\theta^0 \le \theta^1 \le \dots \le \theta^{m+h}$ (componentwise inequalities). Rather than computing a hyperplane representation of $\widehat{\Theta}_f$, we compute a set of simplices $S_1, \dots, S_h$ such that *(i)* $\cup_{i=1}^h S_i = \widehat{\Theta}_f$, and *(ii)* S_i, S_j have mutually disjoint interiors for $i \neq j$. The simplices S_i are defined recursively as follows: (1) Let $L_0 \triangleq \{\theta^0, \theta^1, \dots, \theta^m\}$ be the set of the first $m+1$ vertices of $\widehat{\Theta}_f$, according to the lexicographic order; set $S_0 \triangleq \text{conv}(L_0)$; (2) for all $j = 1, \dots, h$: Let $\widetilde{\theta}$ be the (unique) element in L_{j-1} such that $\widetilde{\theta} + \beta(\theta^{m+j} - \widetilde{\theta}) \in S_{j-1}$ for some $\beta > 0$. Set $L_j \triangleq L_{j-1} \setminus \{\widetilde{\theta}\} \cup \{\theta^{m+j}\}$; set $S_j \triangleq \text{conv}(L_j)$. More efficient ways of obtaining the triangularization $S_0, S_1, \dots, S_h$ of $\widehat{\Theta}_f$ may be devised, although this is beyond the scope of this paper.

Note that the full-dimensionality test (10) may be substituted by the condition rank $\left[\begin{smallmatrix} 1 & 1 & \cdots & 1 \\ \theta^0 & \theta^1 & \cdots & \theta^{m+h} \end{smallmatrix}\right] = m$, i.e., by testing that $\widehat{\Theta}_f$ is a full-dimensional polyhedron. On the other hand, test (10) is independent on the choice of the directions r^i, which provides more numerical robustness.

4.2 Evaluation of the Solution

Algorithm 4.1 provides the solution of (CP_θ) organized on a tree structure T. The root node of T corresponds to the given set of parameters of interest $\Theta = \{\theta : Q\theta \le R\}$. At the first level, the nodes correspond to the initial simplices $S_0, S_1, \dots, S_h$ obtained by the ray-shooting procedure. Each node at the first level is the root of a subtree corresponding to the simplicial partition produced by the recursive procedure.

The multiparametric solution is defined over the simplices associated with the leaf nodes, and in principle the internal nodes do not provide any information. However, by keeping such an information, the tree can be exploited to evaluate the multiparametric solution in a very efficient manner. In fact, it is easy to check that for a given $\theta \in \mathbb{R}^m$, determining the simplex which contains θ requires at most $m^2(h+(N-1)(m+1))$ basic arithmetic operations, where N is the depth of T, and h is the number of simplices S_i obtained by the ray-shooting procedure. Note that this way of evaluating the solution requires not only the storage of (M^{-1}, X) in the leaf nodes, where M, X are defined in (2), but also the storage of M^{-1} in all the internal nodes.

[1]A vector θ^i is redundant if $\text{conv}(\theta^0, \dots, \theta^{m+t}) = \text{conv}(\theta^0, \dots, \theta^{i-1}, \theta^{i+1}, \dots, \theta^{m+t})$. Redundancy can be easily tested via linear programming.

5 Approximate Multiparametric Semidefinite Programming

Parametric semidefinite programming (SDP) was addressed in [11] for the case of scalar perturbations of the cost function. In order to deal with SDP problems with multiparametric perturbations, the analysis and the algorithm developed in the previous sections for the convex multiparametric program (CP_θ) can be extended to generalized inequalities and generalized convexity. Here we focus on a parametric semidefinite program where all functions are affine and the inequalities are defined with respect to the proper cone $\mathbb{S}_+^p$ of symmetric positive semidefinite $p \times p$ real matrices; we denote the condition $P \in \mathbb{S}_+^p$ by $P \succcurlyeq 0$.

More precisely, we formulate a multiparametric semidefinite programming problem as follows:

$$\begin{array}{ll} \min & c'x + f'\theta \\ \text{s. t.} & \sum_{i=1}^n x_i F_i + G_0 + \sum_{j=1}^m \theta_j G_j \succcurlyeq 0 \\ & Ax + B\theta + d = 0 \end{array} \qquad (11)$$

where $c \in \mathbb{R}^n$, $f \in \mathbb{R}^m$, F_i are real symmetric $p \times p$ matrices for all $i = 1, \dots, n$, G_j are real symmetric $p \times p$ matrices for all $j = 0, 1, \dots, m$, $A \in \mathbb{R}^{q \times n}$, $B \in \mathbb{R}^{q \times m}$, and $d \in \mathbb{R}^q$. Note that the term $f'\theta$ in the objective function is irrelevant for the optimization.

Lemma 2 *Let Θ_f be the feasible parameter set and let V^* be the value function of problem (11). Then, Θ_f is a convex set and V^* is a convex function.*

Proof: See [2]. □

As the convexity of V^* is the key hypothesis behind our development, Lemma 2 implies that the analysis of Section 3 and the solver of Section 4 can be extended to a problem of the form (11) in a straightforward manner.

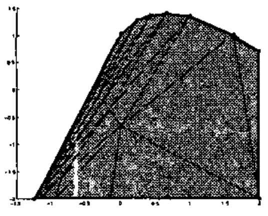
5.1 A Numerical Example

Consider the multiparametric semidefinite program

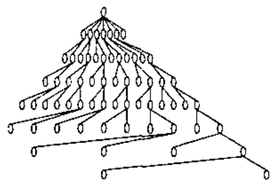
$$\begin{array}{l} \min\limits_{x \in \mathbb{R}^3} \; x_1 - 2x_2 + x_3 \\ \text{s. t.} \begin{bmatrix} 1 & 2 & -3 \\ 2 & 4 & -1 \\ -3 & -1 & 3 \end{bmatrix} + \begin{bmatrix} 1 & -1 & 2 \\ -1 & 1 & 3 \\ 2 & 3 & 2 \end{bmatrix} \theta_1 + \begin{bmatrix} -1 & 1 & 0 \\ 1 & 1 & 2 \\ 0 & 2 & -2 \end{bmatrix} \theta_2 + \\ \begin{bmatrix} 3 & -2 & 4 \\ -2 & 1 & -2 \\ 4 & -2 & -2 \end{bmatrix} x_1 + \begin{bmatrix} -3 & 1 & 1 \\ 1 & -2 & -1 \\ 1 & -1 & 1 \end{bmatrix} x_2 + \begin{bmatrix} 5 & 4 & 2 \\ 4 & 1 & 1 \\ 2 & 1 & -1 \end{bmatrix} x_3 \succcurlyeq 0. \end{array} \qquad (12)$$

We are interested in approximating the multiparamentric solution within the box $\Theta = \{\theta \in \mathbb{R}^2 : -2 \le \theta_1, \theta_2 \le 2\}$ with a precision $\epsilon = 0.5$. To this end, we run Algorithm 4.1, which returns the solution after 2.05 s[2]. In Figure 2(a) we depict the simplicial partition determined by the algorithm, while in Figure 2(b) the associated tree structure for evaluation of the approximate solution, which consists of eight levels. The polyhedral partition in Figure 2(a) contains 35 regions, corresponding to the leaf nodes in Figure 2(b). In Figure 3 we show the value function $V^*(\theta)$ and the error

[2]The results were obtained on a PC Pentium III mobile 850 Mhz running Matlab 5.3 and the SDP solver [19].

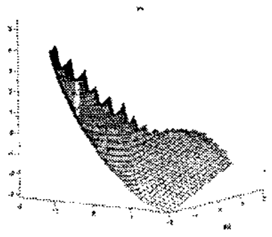


(a) Partition in θ-space. Ray-shoots for estimating the set of feasible parameters are represented by circles

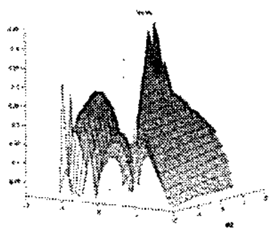


(b) Tree structure for evaluation of the approximate solution (total number of nodes: 64)

Figure 2: Approximate multiparametric solution of problem (12)



(a) Value function $V^*(\theta)$



(b) Error $\widehat{V}(\theta) - V^*(\theta)$

Figure 3: Multiparametric solution associated with problem (12)

$\widehat{V}(\theta) - V^*(\theta)$, where $V^*(\theta)$ was computed numerically by gridding. Note that the error is always smaller than the prescribed precision $\epsilon = 0.5$, is zero at the vertices of the simplices, and is always below about 10% of the range of values of the optimal value function.

Note that in the present multiparametric SDP context only an approximate description of an optimal solution may be obtained, as an exact analytical characterization of the value function V^* is not yet known. This is a topic currently under investigation.

6 Conclusions

In this paper we have provided a recursive algorithm for determining approximate multiparametric solutions of convex nonlinear programming problems, where the value function is approximated within a given suboptimality threshold. The approximate solution is expressed as a piecewise affine function over a simplicial partition of a given set of feasible parameters.

We envision several applications of the technique, especially for the practical implementation of robust model predictive control schemes based on convex optimization, of which several formulations are already available in the literature. It is a topic for further research to analyze which schemes lead to multiparametric programs that are convex both in the variables and in the parameters, and how to maintain robust stability properties in spite of the approximation error.

The results of this paper were also extended for approximating solutions to multiparametric geometric programs within a given relative error.

References

[1] A. Bemporad, F. Borrelli, and M. Morari. Model predictive control based on linear programming — The explicit solution. *IEEE Trans. Automatic Control*, 47(12):1974–1985, December 2002.

[2] A. Bemporad and C. Filippi. Approximate multiparametric convex programming. Technical Report 07/03, Dip. Ingegneria dell'Informazione, University of Siena, 2003. `http://www.dii.unisi.it/~bemporad`.

[3] A. Bemporad and C. Filippi. Suboptimal explicit RHC via approximate multiparametric quadratic programming. *Journal of Optimization Theory and Applications*, 117(1):9–38, April 2003.

[4] A. Bemporad, M. Morari, V. Dua, and E.N. Pistikopoulos. The explicit linear quadratic regulator for constrained systems. *Automatica*, 38(1):3–20, 2002.

[5] F. Borrelli, A. Bemporad, and M. Morari. A geometric algorithm for multi-parametric linear programming. *Journal of Optimization Theory and Applications*, 2002. Accepted for publication as a regular paper.

[6] S. Boyd and L. Vandenberghe. Convex optimization. `http://www.stanford.edu/~boyd/cvxbook.html`, December 2002.

[7] Model Predictive Control. *E.F. Camacho and C. Bordons.* Advanced Textbooks in Control and Signal Processing. Springer-Verlag, London, 1999.

[8] A.V. Fiacco. *Introduction to sensitivity and stability analysis in nonlinear programming.* Academic Press, London, U.K., 1983.

[9] C. Filippi. An algorithm for approximate multiparametric linear programming. *Journal of Optimization Theory and Applications.* To appear.

[10] T. Gal. *Postoptimal Analyses, Parametric Programming, and Related Topics.* de Gruyter, Berlin, 2nd edition, 1995.

[11] D. Goldfarb and K. Scheinberg. On parametric semidefinite programming. *Applied Numerical Mathematics*, 29:361–377, 1999.

[12] R. Horst and N.V. Thoai. Dc programming: Overview. *Journal of Optimization Theory and Applications*, 103(1):1–43, October 1999.

[13] T.A. Johansen. On multi-parametric nonlinear programming and explicit nonlinear model predictive control. In *Proc. 41th IEEE Conf. on Decision and Control*, pages 2768–2773, Las Vegas, Nevada, USA, December 2002.

[14] T.A. Johansen and A. Grancharova. Approximate explicit constrained linear model predictive control via orthogonal search tree. *IEEE Trans. Automatic Control*, 58(5):810–815, May 2003.

[15] J.M. Maciejowski. *Predictive Control with Constraints.* Prentice Hall, 2002.

[16] O.L. Mangasarian and J.B. Rosen. Inequalities for stochastic nonlinear programming problems. *Operations Research*, 12:143–154, 1964.

[17] M.M. Seron, J.A. DeDoná, and G.C. Goodwin. Global analytical model predictive control with input constraints. In *Proc. 39th IEEE Conf. on Decision and Control*, pages 154–159, 2000.

[18] P. Tøndel, T.A. Johansen, and A. Bemporad. An algorithm for multi-parametric quadratic programming and explicit MPC solutions. *Automatica*, 2003. To appear.

[19] L. Vandenberghe, S. Boyd, and B. Alkire. SP — Software for semidefinite programming (Version 1.1). `http://www.ee.ucla.edu/~vandenbe/sp.html`, May 1999.