Computational Optimization and Applications, 35, 87–108, 2006

DOI: 10.1007/s10589-006-6447-z

An Algorithm for Approximate Multiparametric Convex Programming

ALBERTO BEMPORAD bemporad@unisi.it
Dip. Ingegneria dell'Informazione, Università di Siena, Italy

CARLO FILIPPI carlo@math.unipd.it
Dip. Matematica Pura e Applicata, Università di Padova, Italy

Received October 5, 2004; Revised July 25, 2005

Published online: 23 March 2006

Abstract. For multiparametric convex nonlinear programming problems we propose a recursive algorithm for approximating, within a given suboptimality tolerance, the value function and an optimizer as functions of the parameters. The approximate solution is expressed as a piecewise affine function over a simplicial partition of a subset of the feasible parameters, and it is organized over a tree structure for efficiency of evaluation. Adaptations of the algorithm to deal with multiparametric semidefinite programming and multiparametric geometric programming are provided and exemplified. The approach is relevant for real-time implementation of several optimization-based feedback control strategies.

Keywords: multiparametric programming, convex programming, sensitivity analysis

1. Introduction

Parametric programming considers optimization problems where the data depend on one or more parameters. Parametric programming techniques systematically subdivide the parameter space into characteristic regions where the optimal value and an optimizer are given as explicit functions of the parameters.

In recent years, a new interest in parametric programming arose in the model predictive control (MPC) community. MPC is a well-known technique widely used in the process industry for the automatic regulation of plants under operating constraints [9, 23]. In model predictive control, the next command action is obtained by solving an optimization problem where the cost function and the constraints depend on the current sensor measurements. In the classic setting, the optimization problem is solved on-line at each time step. However, most of the optimization effort may be moved off-line by solving a multiparametric program where variables correspond to command inputs, and parameters correspond to sensor measurements [2, 5, 30].

A vast literature is concerned with parametric programming, but it is almost always restricted to a single parameter and/or to very well-known problems, like linear programs [7, 15] or convex quadratic programs [5, 30, 31]. We may distinguish two main issues explaining these limitations of the research efforts: (i) contrarily to the case of one scalar parameter, where the parametric solution consists of a subdivision of the real

axis into segments, parametric solutions with more than one parameter are difficult to analyze by a human decision maker; (ii) for more general convex optimization problems the exact characterization of the optimal value function may not be expressible in analytical form.

When MPC applications are assumed, the above issue (i) vanishes, as the output analysis competes to an electronic device. On the other hand, designing methods to get an approximate description of the optimal value function and of a sub-optimal solution is a promising direction for coping with the above issue (ii). A seminal contribution in this direction was given by ([12], Chapter 9). In the context of general parametric convex nonlinear programming, he sketched a strategy for approximating optimal value functions along a mono-dimensional cut of the parameter space. Essentially, Fiacco noted that optimal primal solutions associated with two fixed parameter vectors may be used to compute an affine upper bound along the line segment joining the same parameter vectors; furthermore, optimal dual solutions associated with the two parameter vectors may be used to compute a piecewise affine lower bound along the same line segment. By following similar observations, Filippi [14] developed an algorithm for approximate multiparametric linear programming. A completely different approach was used by [4] to get an approximate solution to a multiparametric strictly convex quadratic programming problem. They proposed to enlarge the exact characteristic region corresponding to a fixed active constraint set by relaxing the first-order optimality conditions, while preserving primal feasibility. Another approach was taken by [20] for obtaining piecewise affine approximate solutions of multiparametric nonlinear programming problems using local quadratic approximations. By extending a previous work of [21, 22] proposed a further approach to multiparametric nonlinear programming, where the parameter space is partitioned by boxes organized in a tree structure. Inside each box, an affine function describing a feasible suboptimal solution is obtained by solving a nonlinear program having one constraint for each vertex of the box.

The problem of multiparametric mixed-integer semidefinite programming was tackled in [28], where the authors find approximate solutions by solving sequences of multiparametric linear programs.

In this paper we consider a quite general class of multiparametric convex programs, and propose a recursive algorithm for approximating, within a given prescribed tolerance, the optimal value and an optimizer as explicit functions of the parameters. Our approach is inspired by the lines suggested in ([12], Chapter 9) and [14], and its main ideas are the following: (i) given a full-dimensional simplex in the parameter space and an optimizer for each simplex vertex, the linear interpolation of the given solutions gives a primal feasible approximation of an optimizer inside the simplex; (ii) if the resulting absolute error in the objective exceeds a prescribed tolerance then the simplex is split into smaller simplices where it applies recursively; (iii) initial simplices are obtained by a triangulation of a polyhedral estimate of the set of feasible parameters. The resulting approximate solution is expressed as a piecewise affine function over a simplicial partition of a subset of the set of feasible parameters, and organized over a tree structure for efficiency of evaluation (a similar tree structure based on boxes rather than simplices was used in [21] and [22]).

The algorithm described in this paper applies to multiparametric convex programming, but may be conveniently adapted to other cases of relevant interest. In particular, the case of multiparametric semidefinite programming is briefly examined and exemplified on a test example. Our algorithm also applies to multiparametric nonconvex problems that can be equivalently reformulated as convex ones. In particular, the case of geometric programming is considered in this paper.

One of the goals of our approach is to open up the application of explicit receding horizon techniques [5] to several robust model predictive control schemes based on convex optimization. A first attempt in this direction was done in [25], where the authors use the approximate multiparametic programming algorithm of this paper to compute robust controllers for uncertain constrained linear dynamical systems.

2. Multiparametric convex programming

Consider the multiparametric convex program

$$(CP_\theta) \qquad \begin{array}{ll} \min_x & f(x,\theta) \\ \text{subject to} & g_i(x,\theta) \le 0 \quad (i=1,\ldots,p) \\ & Ax + B\theta + d = 0, \end{array}$$

where $x \in I\!R^n$ are the decision variables, $\theta \in I\!R^m$ are the parameters, $f : I\!R^n \times I\!R^m \mapsto I\!R$ is the objective function, $g_i : I\!R^n \times I\!R^m \mapsto I\!R$, for all $i = 1, \ldots, p$, A is a $q \times n$ real matrix, B is a $q \times m$ real matrix, and $d \in I\!R^q$. We assume that f and g_i $(i = 1, \ldots, p)$ are jointly convex in the variables and the parameters. We are interested in characterizing the solution of problem (CP_θ) for a given full-dimensional, convex, and bounded set Θ of parameters. In order to describe more precisely this task, we give some definitions.

Definition 2.1. The *feasible parameter set* Θ^* is the set of all $\theta \in \Theta$ for which the corresponding problem (CP_θ) admits an optimal solution.

Definition 2.2. The *value function* $V^* : \Theta^* \mapsto I\!R$ is the function that associates with every $\theta \in \Theta^*$ the corresponding unique optimal value of (CP_θ).

Definition 2.3. The *optimizer set function* $X^* : \Theta^* \mapsto 2^{I\!R^n}$ is the function that associates to a parameter vector $\theta \in \Theta^*$ the corresponding set of optimizers $X^*(\theta) = \{x \in I\!R^n : f(x,\theta) = V^*(\theta)\}$ of problem (CP_θ).

Definition 2.4. An *optimizer function* $X^* : \Theta^* \mapsto I\!R^n$ is a function that associates to a parameter vector $\theta \in \Theta^*$ (one of) the optimizer(s) $x^*(\theta) \in X^*(\theta)$.

Solving problem (CP_θ) amounts to determining the feasible parameter set Θ^*, an optimizer function x^*, and the value function V^* as explicit functions of θ, for all $\theta \in \Theta^*$.

The following basic result for multiparametric convex programming was proved in ([24], Lemma 1) in the absence of equality constraints; it can be easily generalized to the presence of linear equality constraints.

Lemma 2.1. *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^* is a convex set and V^* is a convex function of θ.*

Hereafter we assume that Θ^* is a full-dimensional set. A numerical test for verifying such an assumption will be provided in Section 4.

2.1. Exact multiparametric solution

In the multiparametric linear and quadratic cases, the exact characterization of Θ^*, x^*, and V^* can be obtained from the Karush Kuhn Tucker (KKT) conditions. In fact, one can fix different combinations of active constraints that correspond to an optimal solution for at least one value of the parameter vector, and determine *linear* equality and inequality relations from the KKT conditions. Such relations define the *polyhedral* subset of Θ^* of all parameters θ for which the fixed combination of constraints is the optimal one (see e.g. [5, 7, 15] for details).

In general, applying the same approach to problem (CP_θ) leads to *nonlinear* equalities defining *nonconvex* subsets of Θ^*. In fact, assuming that f, g_i are differentiable, the KKT optimality conditions for problem (CP_θ) are (see, e.g., [8, Chapter 5]):

$$g_i(x, \theta) \le 0, \quad (i = 1, \dots, p) \tag{1a}$$

$$Ax + B\theta + d = 0, \tag{1b}$$

$$\lambda_i \ge 0, \quad (i = 1, \dots, p) \tag{1c}$$

$$\lambda_i g_i(x, \theta) = 0, \quad (i = 1, \dots, p) \tag{1d}$$

$$\nabla_x f(x, \theta) + \sum_{i=1}^{p} \lambda_i \nabla_x g_i(x, \theta) + A'v = 0, \tag{1e}$$

where $\nabla_x f(x, \theta)$ and $\nabla_x g_i(x, \theta)$ $(i = 1, \dots, p)$ denote the gradients of the respective functions computed in (x, θ), and $\lambda \in \mathbb{R}^p$ and $v \in \mathbb{R}^q$ are the vectors of dual variables (or Lagrange multipliers).

Denoting by $I \subseteq \{1, \dots, p\}$ the set of indices corresponding to a selected combination of active constraints, the KKT conditions lead to the relations

$$\begin{cases} g_i(x, \theta) = 0, \quad (i \in I) \\ Ax + B\theta + d = 0, \\ \lambda_i = 0, \quad (i \notin I) \\ \nabla_x f(x, \theta) + \sum_{i \in I} \lambda_i \nabla_x g_i(x, \theta) + A'v = 0, \end{cases} \tag{2a}$$

$$\begin{cases} g_i(x, \theta) \le 0, \quad (i \notin I) \\ \lambda_i \ge 0, \quad (i \in I). \end{cases} \tag{2b}$$

For each given θ, conditions (2a) represent $p + q + n$ (possibly nonlinear) equality relations in the $p + q + n$ unknowns x, λ, v. In general, relations $x(\theta)$, $\lambda(\theta)$,

$\nu(\theta)$ satisfying (2a) may not be expressible in analytical form. By substituting $x(\theta)$ and $\lambda(\theta)$ in (2b) one would obtain the characteristic (in general nonconvex) region of parameters θ for which the selected combination of active constraints is the optimal one.

From the above considerations, it is apparent that obtaining the *exact* characterization of the feasible parameter set, of the value function, and of an optimizer function may be impractical, if not impossible. For this reason, in the rest of the paper we describe a multiparametric programming algorithm for determining an *approximate* characterization within an arbitrary given prescribed tolerance.

3. Error bounds

Let $\theta^0, \theta^1, \ldots, \theta^m \in IR^m$ be affinely independent points in Θ^*, and define S as the following m-dimensional simplex:

$$S \triangleq \left\{ \theta \in IR^m : \theta = \sum_{k=0}^{m} \mu_k \theta^k,\ \sum_{k=0}^{m} \mu_k = 1,\ \mu_k \geq 0,\ k = 0, 1, \ldots, m \right\}. \tag{3}$$

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

$$M \triangleq \begin{bmatrix} 1 & 1 & \cdots & 1 \\ \theta^0 & \theta^1 & \cdots & \theta^m \end{bmatrix}, \qquad X \triangleq [x^0\ x^1\ \cdots\ x^m], \tag{4}$$

and note that by construction M is nonsingular. As shown in [14], the system of linear inequalities $M^{-1}\left[\begin{smallmatrix} 1 \\ \theta \end{smallmatrix}\right] \geq 0$ represents simplex S by using the minimum number of constraints.

In the following, we introduce upper and lower bounds on V^* inside S; all of them are visualized for convenience in Figure 1. Such bounds generalize to the multidimensional case the concepts introduced by ([12], Chapter 9) to bound the value function of a parametric convex program inside a line segment (cf. also [20]).

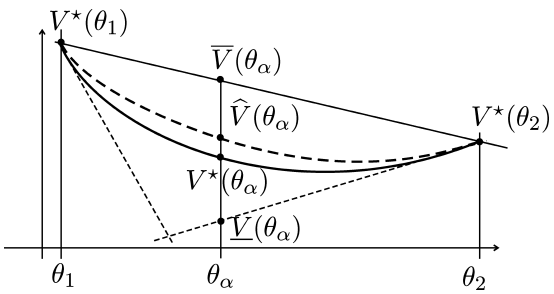

Figure 1. Approximation of the value function in convex parametric programming: the scalar case.

3.1. Upper bounds on the value function

Define the vector

$$v \triangleq [V^*(\theta^0)\ V^*(\theta^1)\ \dots\ V^*(\theta^m)]', \tag{5}$$

and, for a generic $\theta \in I\!R^m$,

$$\hat{x}(\theta) \triangleq X M^{-1} \begin{bmatrix} 1 \\ \theta \end{bmatrix}. \tag{6}$$

Furthermore, define

$$\hat{V}(\theta) \triangleq f(\hat{x}(\theta), \theta), \tag{7}$$

and

$$\bar{V}(\theta) \triangleq v' M^{-1} \begin{bmatrix} 1 \\ \theta \end{bmatrix}. \tag{8}$$

Note that both $\hat{x}$ and $\bar{V}$ depend affinely on θ.

Proposition 3.1. *For all $\theta \in S$, vector $\hat{x}(\theta)$ is a feasible solution of (CP_θ) and*

$$\bar{V}(\theta) \geq \hat{V}(\theta) \geq V^*(\theta). \tag{9}$$

Proof: We first prove that $\hat{x}(\theta)$ is feasible. Vector $\mu = M^{-1}\left[\begin{smallmatrix} 1 \\ \theta \end{smallmatrix}\right]$ is the unique solution of $M\mu = [\begin{smallmatrix} 1 \\ \theta \end{smallmatrix}]$. 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 $\hat{x}(\theta) = \sum_{k=0}^m \mu_k x^k$. As a consequence, for all $i = 1, \dots, p$,

$$g_i(\hat{x}(\theta), \theta) = g_i\left(\sum_{k=0}^m \mu_k x^k, \sum_{k=0}^m \mu_k \theta^k \right) \leq \sum_{k=0}^m \mu_k g_i(x^k, \theta^k) \leq 0,$$

where the first inequality follows from the joint convexity of g_i with respect to x and θ. Furthermore,

$$A\hat{x}(\theta) + B\theta + d = \sum_{k=0}^m \mu_k (Ax^k + B\theta^k + d) = 0.$$

To prove (9), we note that

$$\begin{aligned} \bar{V}(\theta) = \sum_{k=0}^m \mu_k V^*(\theta^k) &= \sum_{k=0}^m \mu_k f(x^k, \theta^k) \\ &\geq f\left(\sum_{k=0}^m \mu_k x^k, \sum_{k=0}^m \mu_k \theta^k \right) = f(\hat{x}(\theta), \theta) = \hat{V}(\theta), \end{aligned}$$

where the inequality follows from the joint convexity of f, and

$$\hat{V}(\theta) = f(\hat{x}(\theta), \theta) \geq f(x^*(\theta), \theta) = V^*(\theta),$$

where $x^*(\theta)$ denotes an optimizer of (CP_θ). □

In summary, $\bar{V}$ and $\hat{V}$ are both upper bounds of V^* on S, exact on every vertex on S, $\bar{V}$ is affine in θ, and $\hat{V}$ is tighter than $\bar{V}$.

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:

$$\underline{V}(\theta) \triangleq \max_{k=0,1,\ldots,m} \{V^*(\theta^k) + (s^k)'(\theta - \theta^k)\}. \tag{10}$$

As first noted by [12],

$$\underline{V}(\theta) \leq V^*(\theta) \quad \text{for all } \theta \in S, \tag{11}$$

and hence $\underline{V}$ is a piecewise affine lower bound on V^* inside S, exact at every vertex of S.

Proposition 3.2. *Assume f and g_i $(i = 1, \ldots, p)$ are differentiable with respect to both x and θ inside their domain, and let (x^k, λ^k, ν^k) be a solution of the KKT conditions* (1) *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: For convenience, let $g(x, \theta) \triangleq [g_1(x, \theta) \ \ldots \ g_p(x, \theta)]'$, and let $J_x g(x, \theta)$ denote the $p \times n$ Jacobian matrix of the partial derivatives of g with respect to x. Let $x^*(\theta)$ be an optimizer of (CP_θ). By using (1) and the convexity and differentiability of f and g we obtain

$$\begin{aligned}
V^*(\theta) &\geq f(x^*(\theta), \theta) + (\lambda^k)'g(x^*(\theta), \theta) + (\nu^k)'(Ax^*(\theta) + B\theta + d) \\
&\geq f(x^k, \theta^k) + \nabla_x f(x^k, \theta^k)'(x^*(\theta) - x^k) + \nabla_\theta f(x^k, \theta^k)'(\theta - \theta^k) \\
&\quad + (\lambda^k)'[g(x^k, \theta^k) + J_x(x^k, \theta^k)(x^*(\theta) - x^k) + J_\theta(x^k, \theta^k)(\theta - \theta^k)] \\
&\quad + (\nu^k)'[A(x^*(\theta) - x^k) + B(\theta - \theta^k)] \\
&= f(x^k, \theta^k) + (\lambda^k)'g(x^k, \theta^k) \\
&\quad + [\nabla_x f(x^k, \theta^k) + J_x(x^k, \theta^k)'\lambda^k + A'\nu^k]'(x^*(\theta) - x^k) \\
&\quad + [\nabla_\theta f(x^k, \theta^k) + J_\theta(x^k, \theta^k)'\lambda^k + B'\nu^k]'(\theta - \theta^k) \\
&= V^*(\theta^k) + (s^k)'(\theta - \theta^k).
\end{aligned}$$

□

A similar result was shown by ([12], 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 3.2.

3.3. Error estimates inside a simplex

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

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

Unfortunately, the above optimization problem is a DC (Difference of Convex functions) programming problem, and thus the exact evaluation of $\epsilon^{MAX}(S)$ is, in general, hard [19]. For this reason, we analyze the two practically computable bounds

$$\epsilon^{LP}(S) \triangleq \max_{\theta \in S} \{\bar{V}(\theta) - \underline{V}(\theta)\},$$
$$\epsilon^{CP}(S) \triangleq \max_{\theta \in S} \{\bar{V}(\theta) - V^*(\theta)\},$$

that are related to $\epsilon^{MAX}(S)$ as shown in the following proposition.

Proposition 3.3. *For all simplices $S \subseteq \Theta^*$ and for all $\theta \in S$, the following inequalities hold*

$$0 \leq \hat{V}(\theta) - V^*(\theta) \leq \epsilon^{MAX}(S) \leq \epsilon^{CP}(S) \leq \epsilon^{LP}(S).$$

Proof: The condition $0 \leq \hat{V}(\theta) - V^*(\theta)$ immediately follows from Proposition 3.1. Moreover, we have:

$$\begin{aligned}\hat{V}(\theta) - V^*(\theta) &\leq \max_{\theta \in S} \{\hat{V}(\theta) - V^*(\theta)\} = \epsilon^{MAX}(S) \\ &\leq \max_{\theta \in S} \{\bar{V}(\theta) - V^*(\theta)\} = \epsilon^{CP}(S) \\ &\leq \max_{\theta \in S} \{\bar{V}(\theta) - \underline{V}(\theta)\} = \epsilon^{LP}(S),\end{aligned}$$

where the second inequality follows from (9) and the third inequality follows from (11). □

Proposition 3.4. *Consider a given set of subgradients $s^k \in IR^m$ of V^* at θ^k, $k = 0, 1, \ldots, m$, and let $w_k \triangleq -V^*(\theta^k) + (s^k)'\theta^k$. Then the corresponding error bound*

$\epsilon^{LP}(S)$ is the optimal value of the following linear program:

$$\begin{aligned} \max_{\theta,t} \quad & \bar{V}(\theta) - t \\ \text{subject to} \quad & (s^k)'\theta - t \le w_k \quad (k = 0, 1, \ldots, m) \\ & M^{-1}\begin{bmatrix} 1 \\ \theta \end{bmatrix} \ge 0, \end{aligned} \tag{12}$$

where M is defined in (4) and $\bar{V}(\theta)$ is defined in (5), (8).

Proof: We have:

$$\begin{aligned} \epsilon^{LP}(S) &= \max_{\theta \in S}\{\bar{V}(\theta) - \underline{V}(\theta)\} \\ &= \max_{\theta \in S}\{\bar{V}(\theta) - \max_k\{V^*(\theta^k) + (s^k)'(\theta - \theta^k) : k = 0, 1, \ldots, m\} : \theta \in S\} \\ &= \max_{\theta,t}\{\bar{V}(\theta) - t : t \ge V^*(\theta^k) + (s^k)'(\theta - \theta^k)\,(k = 0, 1, \ldots, m), \theta \in S\}. \end{aligned}$$

□

Proposition 3.5. *The error bound $\epsilon^{CP}(S)$ is the optimal value of the following convex program:*

$$\begin{aligned} \max_{x,\theta} \quad & \bar{V}(\theta) - f(x, \theta) \\ \text{subject to} \quad & g_i(x, \theta) \le 0 \quad (i = 1, \ldots, p) \\ & Ax + B\theta + d = 0 \\ & M^{-1}\begin{bmatrix} 1 \\ \theta \end{bmatrix} \ge 0. \end{aligned} \tag{13}$$

Moreover, if $(\bar{x}, \bar{\theta})$ is an optimal solution of (13) then $\overline{x}$ is an optimal solution of $(CP_{\overline{\theta}})$, i.e., $f(\overline{x}, \overline{\theta}) = V^*(\overline{\theta})$.

Proof: Let $F(\theta) \triangleq \{x \in IR^n : g_i(x, \theta) \le 0\ (i = 1, \ldots, p), Ax + B\theta + d = 0\}$ be the feasible set of (CP_θ). Then,

$$\begin{aligned} \epsilon^{CP}(S) &= \max_{\theta \in S}\{\bar{V}(\theta) - V^*(\theta)\} \\ &= \max_{\theta \in S}\{\bar{V}(\theta) - \min_x\{f(x, \theta) : x \in F(\theta)\}\} \\ &= \max_{\theta \in S}\{\bar{V}(\theta) + \max_x\{-f(x, \theta) : x \in F(\theta)\}\} \\ &= \max_{x,\theta}\{\bar{V}(\theta) - f(x, \theta) : x \in F(\theta), \theta \in S\}. \end{aligned}$$

The last statement can be trivially proved by contradiction. □

In conclusion, both ϵ^{LP} and ϵ^{CP} are computable upper bounds to ϵ^{MAX}, with ϵ^{LP} raising from a multidimensional extension of the ideas suggested in [12]. As a consequence, both ϵ^{LP} and ϵ^{CP} can be embedded in an approximate multiparametric convex solver.

Computing ϵ^{LP} involves solving 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. Furthermore, the computation of $\epsilon^{LP}(S)$ yields a parameter vector $\bar{\theta} \in S$ such that $\bar{V}(\bar{\theta}) - \underline{V}(\bar{\theta}) = \epsilon^{LP}(S)$, whereas the computation of $\epsilon^{CP}(S)$ yields a parameter vector $\bar{\theta} \in S$ such that $\bar{V}(\bar{\theta}) - V^*(\bar{\theta}) = \epsilon^{CP}(S)$ *and* an optimal solution of $(CP_{\bar{\theta}})$. Since this latter information seems crucial to obtain an efficient multiparametric solver, in the sequel we shall focus on the use of ϵ^{CP}.

3.4. Error bounds on the optimizer

In some applications the focus may be on approximating the optimizer rather than the value function. In principle, this is possible, but hardly practicable. Some computable error bounds on the optimizer have been proposed in the literature for nonlinear programs, but usually they are very hard to obtain (see, e.g., [13, 18, 27]).

The most promising error bound has been proposed by Fiacco and Kyparisis [13] in connection with the approximation method of ([12], Chapter 9). As mentioned in Section 1, Fiacco suggested to approximate the optimal solution of a parametric convex program along a line segment in the parametric space by using the linear interpolation of the optimal solutions computed at the extremes of the same segment. Fiacco and Kyparisis [13] showed how to bound the distance of such a linear interpolation from a genuine optimizer by means of uniform quadratic underestimation of the objective function value. Their approach is easily extendable to the case where the line segment is replaced by a full-dimensional simplex. However, computing Fiacco-Kyparisis' bound requires, in general, solving a multiparametric nonconvex problem, and thus leads to an unpracticable method.

We simply mention that Fiacco-Kyparisis' bound may be computable in some special, though important, cases, i.e., separable objective function, convex quadratic programming, and nonparametric objective function with bounded feasible set. In the latter case, however, the obtained bound may be very conservative.

4. An approximate multiparametric convex solver

We are in a position to state a basic approximation algorithm for (CP_θ). We first analyze in detail the case when an initial full-dimensional simplex $S \subseteq \Theta^*$ is given, then we embed the resulting algorithm in a more general solver for convex, bounded, and full-dimensional sets Θ. The general solver also handles the case of lower dimensional Θ^*.

4.1. A recursive algorithm

The following algorithm takes as input:

(*a*) $m+1$ parameter vectors $\theta^0, \theta^1, \ldots, \theta^m \in \Theta^*$;
(*b*) the corresponding optimal values $V^*(\theta^0), V^*(\theta^1), \ldots, V^*(\theta^m)$;

(*c*) $m+1$ vectors $x^0, x^1, \ldots, x^m \in IR^n$ such that x^k is an optimal solution of (CP_{θ^k}) for all $k = 0, 1, \ldots, m$.

The input either comes from the algorithm itself because of a recursive call, or from the general solver described in Section 4.2. In the latter case, vectors $\theta^0, \theta^1, \ldots, \theta^m$ are guaranteed to be affinely independent, so that their convex hull is a full-dimensional simplex contained in Θ^*.

Algorithm 4.1

Build M and X as defined in (4);
if *M is nonsingular* **then**
  compute the optimum $\epsilon^{CP}(S)$ *and an optimizer* $(\bar{x}, \bar{\theta})$ *of (13);*
  if $\epsilon^{CP}(S) > \epsilon$ **then**
    for $k = 0, 1, \ldots, m$ **do**
      replace θ^k *by* $\bar{\theta}$, $V^*(\theta^k)$ *by* $V^*(\bar{\theta}) = f(\bar{x}, \bar{\theta})$, *and* x^k *by* $\bar{x}$;
      call this algorithm on the modified data;
  else *return* (M^{-1}, X)

Note that, at each recursive iteration, the current simplex is split into at most $m+1$ full-dimensional simplices with nonoverlapping interiors. The output of Algorithm 4.1 is a collection $\{(M_h^{-1}, X_h) : h = 1, \ldots, L\}$ from which we can obtain:

(A) a simplicial partition $\{S_h : h = 1, \ldots, L\}$ of the initial simplex S, where $S_h \triangleq \{\theta \in IR^m : M_h^{-1}\begin{bmatrix} 1 \\ \theta \end{bmatrix} \geq 0\}$;
(B) a piecewise affine function $\hat{x} : S \mapsto IR^n$ defined as:

$$\hat{x}(\theta) \triangleq X_h M_h^{-1} \begin{bmatrix} 1 \\ \theta \end{bmatrix} \quad \text{if } \theta \in S_h \qquad (h = 1, \ldots, L);$$

(C) a piecewise analytical function $\hat{V} : S \mapsto IR$ defined as

$$\hat{V}(\theta) \triangleq f(\hat{x}(\theta), \theta) \quad \text{for all } \theta \in S.$$

In particular, the above functions enjoy the following properties:

(i) $\hat{x}(\theta)$ is a feasible solution of (CP_θ) for all $\theta \in S$;
(ii) $0 \leq \hat{V}(\theta) - V^*(\theta) \leq \epsilon$ for all $\theta \in S$;
(iii) $\hat{V}(\theta^\#) = V^*(\theta^\#)$ for any vector $\theta^\#$ that is a vertex of a simplex in the obtained partition.

Remark 4.1. The values of $\hat{x}$ and $\hat{V}$ might not be uniquely defined on overlapping boundaries of the returned simplices (a subset of S of null measure) although any single-valued function one can extract would still enjoy the above properties. However, if in

every recursive call vector $\bar{\theta}$ lies in the interior of its simplex, then $\hat{x}$ and $\hat{V}$ are both continuous functions of the parameter θ. If the continuity property is required, we may force the above condition by imposing in (13) the tighter constraint $M^{-1}\left[\begin{smallmatrix}1\\ \theta\end{smallmatrix}\right] \geq \sigma e$, where σ is a comparatively small positive scalar and $e \in I\!R^{m+1}$ 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 $\frac{1}{m+1}\sum_{k=0}^{m}\theta^k$.

Remark 4.2. By using $\epsilon^{CP}(S)$, the proposed method controls the absolute error on the value function with respect to $\bar{V}$, which constitutes an approximation of V^* worse than the actually returned $\hat{V}$. As a consequence, there may be cases where a simplex is split because $\epsilon^{CP}(S) > \epsilon$ though the maximum difference $\epsilon^{MAX}(S)$ between $\hat{V}$ and V^* is less than the prescribed ϵ. In order to possibly avoid unnecessary splits, consider the error quantity $\underline{\epsilon}(S) \triangleq \hat{V}(\bar{\theta}) - V^*(\bar{\theta}) = f(\hat{x}(\bar{\theta}), \bar{\theta}) - V^*(\bar{\theta}) \leq \epsilon^{MAX}(S)$, 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 $\hat{V}$ over S for estimating $\epsilon^{MAX}(S)$ with an arbitrary precision before deciding to split the simplex is described in [3].

4.1.1. Complexity analysis. We now discuss how to bound the complexity of Algorithm 4.1. Such a complexity depends on the solution of a number of convex programs. Since the time complexity of convex programming depends on the properties of the model and the algorithm implemented, we consider a convex programming solver as an oracle, and we evaluate the complexity by the maximum number of convex programs that have to be solved. Accordingly, we denote by $\mathbf{cp}[\alpha, \beta, \gamma]$ a time complexity of solving a convex program with α variables, β nonlinear convex constraints, and γ affine constraints.

Consider Algorithm 4.1. The complexity of each recursive call is clearly dominated by the solution of problem (13). Then, the time complexity of each call of Algorithm 4.1 is simply

$$O\,(\mathbf{cp}[n+m,\, p, q+m+1]).$$

The total number of calls depends on the number of simplices generated to build an approximation of V^* satisfying the required tolerance. We may guess that such a number is exponential in the input size (cf. [26]). On the other hand, it is reasonable to express the overall complexity of Algorithm 4.1 as a function of the output size. To this end, we need a definition and a technical lemma.

A rooted tree T is *full* if every node of T either is a leaf or has at least two children (cf. Chapter 5 of [10]). The smallest full tree is composed by three nodes: the root and its two children. Let $\rho(\lambda)$ denote the maximum number of nodes of a full tree with λ leaves. It is easy to prove by contradiction that if T is a full tree with λ leaves and $\rho(\lambda)$ nodes, then T must be binary. Moreover, it is easy to prove by induction on λ that

if T is a full binary tree with λ leaves then T has exactly $2\lambda - 1$ nodes. We deduce the technical lemma.

Lemma 4.1. $\rho(\lambda) = 2\lambda - 1$.

We are thus able to state the following.

Lemma 4.2. *Algorithm 4.1 builds up an approximate description of S using L simplices in time*

$$O\,(L\mathbf{cp}[n+m,\, p,\, q+m+1]). \tag{14}$$

Proof: Each time Algorithm 4.1 is called, a simplex in the parameter space is explored. If the upper bound on the error inside the simplex is less than the prescribed tolerance, the current simplex is returned; otherwise the simplex is subdivided in two or more smaller simplices and a recursive call is performed for each of them. Thus, the exploration of the parameter space performed by Algorithm 4.1 may be visualized by a search tree, where nodes correspond to explored simplices and arcs correspond to recursive calls. When Algorithm 4.1 stops, a problem (13) has been solved for every node of the search tree. If $L = 1$ then the search tree is a single node, and the stated time complexity trivially holds. If $L > 1$, the assumed L simplices returned by the algorithm are the leaves of the search tree and Algorithm 4.1 has solved $I + L$ problems (13), where I denote the number of the tree nodes with at least one child. By construction, each tree node with at least one child has in fact a number of children ranging from 2 to $m + 1$. Thus the search tree is full, and we may write $I + L \leq \rho(L)$. As from Lemma 4.1.1 we have that $\rho(L) = O(L)$, the stated time complexity follows. □

The following result is immediate.

Theorem 4.1. If **cp** is a polynomial function, then the time complexity of Algorithm 4.1 is polynomial in the output size.

4.2. The general solver

So far, we have assumed that Θ^* is a full-dimensional set, and our analysis has been restricted to a full-dimensional simplex contained in Θ^*. In order to obtain a general approximate multiparametric convex solver, we need first to verify the full-dimensionality assumption, and then to approximate Θ^* by an initial collection of nonoverlapping simplices.

A necessary condition for Θ^* to be full-dimensional is that the equality constraints $Ax + B\theta + d = 0$ do not restrict θ to lie on a lower-dimensional affine subspace of $I\!R^m$ (i.e., the set $\{\theta \in I\!R^m : \exists x \in I\!R^n : Ax + B\theta + d = 0\}$ 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 $a \neq 0 \in I\!R^m$.

Assuming that the linear constraints $Ax + B\theta + d = 0$ do not reduce the dimension of Θ^*, let $S(\theta, \rho)$ be the convex hull of $\theta + \rho e^0, \theta + \rho e^1, \ldots, \theta + \rho e^m$, where e^j is the jth column of the $m \times m$ identity matrix, $j = 1, \ldots, m$, and $e^0 = 0 \in I\!R^m$. We determine the largest simplex $S(\theta, \rho)$ contained in Θ^* by solving

$$\begin{aligned} \rho^\star = \max_{\theta, \rho, y^0, \ldots, y^m} \quad & \rho \\ \text{subject to} \quad & g_i(y^j, \theta + \rho e^j) \le 0, \quad (i = 1, \ldots, p; j = 0, \ldots, m) \\ & Ay^j + B(\theta + \rho e^j) = d, \quad (j = 0, \ldots, m) \\ & \theta + \rho e^j \in \Theta, \quad (j = 0, \ldots, m) \end{aligned} \tag{15}$$

which is a convex program in $(m + 1)(n + 1)$ variables. Then Θ^* is full-dimensional if and only if $\rho^\star > 0$, as the volume of the largest simplex is $(\rho^\star)^m/m! > 0$.

Once the full-dimensionality of Θ^* is tested, we determine an inner polyhedral approximation $\hat{\Theta}$ through a "ray-shooting" procedure, described as follows. Let $r^0, r^1 \ldots, r^{t+m}$ be $m + 1 + t$ directions in $I\!R^m$, $t \ge 0$, such that the convex positive cone $C = \{\theta \in I\!R^m : \theta = \sum_{k=0}^{m+t} \mu_k r^k,\ \mu_k \ge 0\}$ is equal to $I\!R^m$. For instance, r^k may be obtained by collecting uniformly distributed samples of the unit hyper-sphere. For each $k = 0, 1, \ldots, m + t$ solve the convex problem

$$\begin{aligned} \max_{x, \theta} \quad & (r^k)'\theta \\ \text{subject to} \quad & g_i(x, \theta) \le 0, \quad (i = 1, \ldots, p) \\ & Ax + B\theta + d = 0, \\ & \theta \in \Theta, \end{aligned}$$

denoting by (x^k, θ^k) the obtained optimal solution. Define $\hat{\Theta}$ as the convex hull of $\theta^0, \theta^1, \ldots, \theta^{m+t}$. It is convenient to discard all vectors θ^k which are not vertices of $\hat{\Theta}$. To this aim, note that a vector $\theta^{\bar{k}}$ is a vertex of $\hat{\Theta}$ if and only if the linear system

$$\sum_{k \ne \bar{k}} \theta^k \mu_k = \theta^{\bar{k}}, \quad \sum_{k \ne \bar{k}} \mu_k = 1, \quad \mu_k \ge 0, \quad (k \ne \bar{k})$$

has no solution. This can be checked via linear programming.

For convenience, we thus assume that $\theta^0, \theta^1, \ldots, \theta^{m+H}$ are the vertices of $\hat{\Theta}$, where $H \le t$ and $H \ge 0$ because Θ^* is full dimensional. There is no need to compute the hyperplane representation of $\hat{\Theta}$. Instead, through the Delaunay triangulation [33] of $\theta^0, \theta^1, \ldots, \theta^{m+H}$, one computes a set of simplices $S_1, \ldots, S_N$ such that:

(i) $\cup_{i=1}^N S_i = \hat{\Theta}$;
(ii) S_i, S_j have disjoint interiors for $i \ne j$;
(iii) $N = O\left((m + H + 1)^{\lceil m/2 \rceil}\right)$, as reported in [29].

Clearly, the full-dimensionality test (15) may be substituted by the condition

$$\operatorname{rank}\begin{bmatrix} 1 & 1 & \dots & 1 \\ \theta^0 & \theta^1 & \dots & \theta^{m+H} \end{bmatrix} = m, \tag{16}$$

i.e., by testing that $\hat{\Theta}$ is a full-dimensional polyhedron. On the other hand, test (15) is independent on the choice of the directions r^k, and therefore it is more robust from a numerical viewpoint.

4.2.1. Evaluation of the solution. The proposed method provides the solution of (CP_θ) organized on a tree structure T. The root node of T corresponds to the whole IR^m. At the first level, the nodes correspond to the initial simplices $S_1, \dots, S_N$ obtained by the ray-shooting and triangulation 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 $\theta \in IR^m$, determining the simplex which contains θ requires at most $m^2(N + (D-1)(m+1))$ basic arithmetic operations, where D is the depth of T. 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 (4), but also the storage of M^{-1} in all the internal nodes.

4.2.2. Complexity analysis. We conclude this section with some remarks on the time complexity of the proposed general solver for approximate multiparametric convex programming.

Testing the full-dimensionality of Θ^* by problem (15) requires

$$O\left(\mathbf{cp}[(m+1)(n+1), p(m+1), (q+u)(m+1)]\right) \tag{17}$$

time, where u is the number of inequalities representing Θ, and thus the time complexity is polynomial in the input size provided **cp** is a polynomial function.

The ray-shooting procedure requires

$$O\left((m+t)\mathbf{cp}[m+n, p, q+u]\right) \tag{18}$$

time, where $m + t + 1$ is the total number of shot rays. It is easy to recognize that identifying the vertices of $\hat{\Theta}$ requires at most

$$O\left((m+t)\mathbf{cp}[m+t, 0, 2m+t]\right) \tag{19}$$

time. Under the weak hypothesis that t is polynomial in m, both the above complexities are polynomial in the input size provided **cp** is a polynomial function. The complexity of the Delaunay triangulation depends on the chosen algorithm, see e.g. p.381 of [17]

where the worst-case complexity of

$$O\left((m+H+1)^{\lceil m/2 \rceil}\right) \tag{20}$$

is reported and $m+H+1 \leq m+t+1$ is the number of nonredundant shot vertices. Clearly, such a complexity is exponential in the input size. However, by virtue of property (iii) of Section 4.2, the Delaunay triangulation has a linear complexity $O(N)$ in the output size N.

Let $\bar{L}$ be the total number of simplices returned by the application of Algorithm 4.1 to each of the simplices obtained by the initial triangulation. By summing up the time requirements (14) we obtain the the total running time required by the N calls of Algorithm 4.1 is

$$O\left(\bar{L}\ \mathbf{cp}[n+m,\ p,\ q+m+1]\right). \tag{21}$$

The overall time complexity of the proposed approach is obtained by summing up (17)–(20) and (21). As that the total output size is $O(mn\bar{L})$, we may conclude that if the number of shot rays is at most polynomial in the number $\bar{L}$ of output simplices and if **cp** is a polynomial function, then the overall complexity of the proposed approach is polynomial in the output size. Note that assuming that t grows polynomially with $\bar{L}$ is a rather weak hypothesis in the case Θ^* is bounded by nonlinear manifolds, as the number t of shot rays is usually not much larger than the number $H = O(\log N)$ of vertices of $\hat{\Theta}$, where $N \leq L$.

The general solver was implemented in Matlab 6.5, considering the convex solver as a library function to be chosen according to the type of problem. The initial set of simplices $S_1, \ldots, S_N$ is obtained via Delaunay triangulation using function `delaunayn`, that is based on the Qhull package [1]. An application example is reported in the next section. Further numerical results are reported in [25].

5. Adaptation to other classes of multiparametric problems

In this section we show how one can easily adapt the approximate multiparametric approach developed above to two general problem classes: multiparametric semidefinite programming and multiparametric geometric programming.

5.1. Approximate multiparametric semidefinite programming

The structure of parametric semidefinite programming (SDP) was analyzed in [16] for the case of scalar perturbations of the cost function. To the best of the authors' knowledge, the only way to obtain an exact analytical characterization of the value function of a (multi)parametric SDP problem consists in two steps. In the first step the problem is reformulated as a multiparametric convex one; in the second step the value function of the equivalent problem is characterized. Unfortunately, the first step produces complex analytical expressions (see, e.g., the approaches proposed in [6]), while the second

step encounters the difficulties pointed out in Section 2.1. Thus, at present, only an approximate solution can be given to a multiparametric SDP.

In order to deal with SDP problems with multiparametric perturbations, the analysis and the algorithm developed in the previous sections for the multiparametric convex program (CP_θ) must 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:

$$\begin{array}{ll} \min_x & c'x + f'\theta \\ \text{subject to} & \displaystyle\sum_{i=1}^{n} x_i F_i + G_0 + \sum_{j=1}^{m} \theta_j G_j \succcurlyeq 0 \\ & Ax + B\theta + d = 0 \end{array} \tag{22}$$

where $c \in I\!R^n$, $f \in I\!R^m$, F_i are all real symmetric $p \times p$ matrices, G_j are all real symmetric $p \times p$ matrices, A is a real $q \times n$ matrix, B is a real $q \times m$ matrix, and $d \in I\!R^q$. We add the term $f'\theta$ in the objective function for consistency with respect to the formulation of (CP_θ), though such a term is irrelevant for the optimization.

Lemma 5.1. Let Θ^* be the feasible parameter set and let V^* be the value function of problem (22). Then, Θ^* is a convex set and V^* is a convex function.

Proof: We first show that Θ^* is a convex set. Let $\theta^h \in \Theta^*$, and let $x^h = [x_1^h \ \dots \ x_n^h]'$ be a corresponding optimal solution of problem (22), with $h = 1, 2$; let $\alpha \in [0, 1]$. Let $M^h \triangleq \sum_{i=1}^n x_i^h F_i + G_0 + \sum_{j=1}^m \theta_j^h G_j$, with $h = 1, 2$. We have:

$$\sum_{i=1}^{n} \left(\alpha x_i^1 + (1-\alpha)x_i^2\right) F_i + G_0 + \sum_{j=1}^{m} \left(\alpha\theta_j^1 + (1-\alpha)\theta_j^2\right) G_j = \alpha M^1 + (1-\alpha)M^2.$$

As x^1 and x^2 are feasible with respect to θ^1 and θ^2, respectively, the right-hand side of the above equation is a nonnegative combination of positive semidefinite matrices, and thus it is a positive semidefinite matrix. Furthermore,

$$\begin{aligned} & A(\alpha x^1 + (1-\alpha)x^2) + B(\alpha\theta^1 + (1-\alpha)\,\theta^2) + d \\ & \quad = \alpha(Ax^1 + B\theta^1 + d) + (1-\alpha)\,(Ax^2 + B\theta^2 + d) \\ & \quad = 0 \end{aligned}$$

Hence, $\alpha x^1 + (1-\alpha)x^2$ is feasible with respect to $\alpha\theta^1 + (1-\alpha)\theta^2$, proving that Θ^* is a convex set. Since

$$\begin{aligned} V^*(\alpha\theta^1 + (1-\alpha)\theta^2) &\le c'(\alpha x^1 + (1-\alpha)x^2) + f'(\alpha\theta^1 + (1-\alpha)\theta^2) \\ &= \alpha(c'x^1 + f'\theta^1) + (1-\alpha)(c'x^2 + f'\theta^2) \\ &= \alpha V^*(\theta^1) + (1-\alpha)V^*(\theta^2). \end{aligned}$$

it follows that V^* is a convex function. □

As the convexity of V^* is the key hypothesis behind our development, Lemma 5.1 implies that the analysis of Section 3 and the solver of Section 4 can be extended to a problem of the form (22) in a straightforward manner.

5.1.1. A Numerical Example. Consider the multiparametric semidefinite program

$$\begin{aligned}\min_{x\in\mathbb{R}^3} \quad & x_1 - 2x_2 + x_3 \\ \text{subject to } & \begin{bmatrix} 1 & 2 & -3 \\ 2 & 4 & -1 \\ -3 & -1 & 3 \end{bmatrix} + \begin{bmatrix} 1 & -1 & 2 \\ -1 & 1 & 3 \\ 2 & 3 & 2 \end{bmatrix}\theta_1 + \begin{bmatrix} -1 & 1 & 0 \\ 1 & 1 & 2 \\ 0 & 2 & -2 \end{bmatrix}\theta_2 \\ & + \begin{bmatrix} 3 & -2 & 4 \\ -2 & 1 & -2 \\ 4 & -2 & -2 \end{bmatrix}x_1 + \begin{bmatrix} -3 & 1 & 1 \\ 1 & -2 & -1 \\ 1 & -1 & 1 \end{bmatrix}x_2 + \begin{bmatrix} 5 & 4 & 2 \\ 4 & 1 & 1 \\ 2 & 1 & -1 \end{bmatrix}x_3 \\ & \succcurlyeq 0. \end{aligned} \tag{23}$$

We are interested in approximating the multiparamentric solution within the box $\Theta = \{\theta \in IR^2 : -2 \le \theta_1, \theta_2 \le 2\}$ with a precision $\epsilon = 0.5$. To this end, we run our general solver, which returns the solution after 0.85 s (the results were obtained on a laptop PC 1.4 Ghz running the Matlab 6.5 code of our solver and the SDP solver of [32]). In Figure 2(a) we depict the simplicial partition determined by the algorithm, while in Figure (2b) the associated tree structure for evaluation of the approximate solution, which consists of eight levels. The polyhedral partition in Figure (2a) contains 20 regions, corresponding to the leaf nodes in Figure (2b). In Figure 3 we show the value function $V^*(\theta)$ and the error $\hat{V}(\theta) - V^*(\theta)$ on a grid of $\theta \in \hat{\Theta}$. 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.

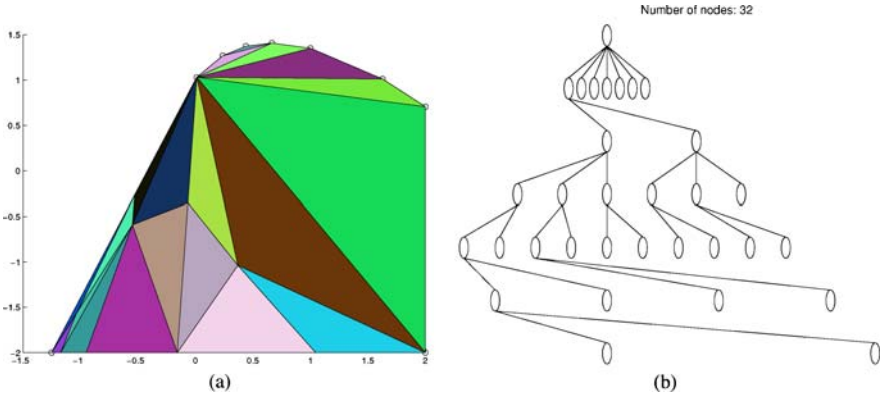

Figure 2. Approximate multiparametric solution of problem (23). (a) Partition in θ-space. The vertices of $\hat{\Theta}$ are represented by circles and (b) Tree structure for evaluation of the approximate solution.

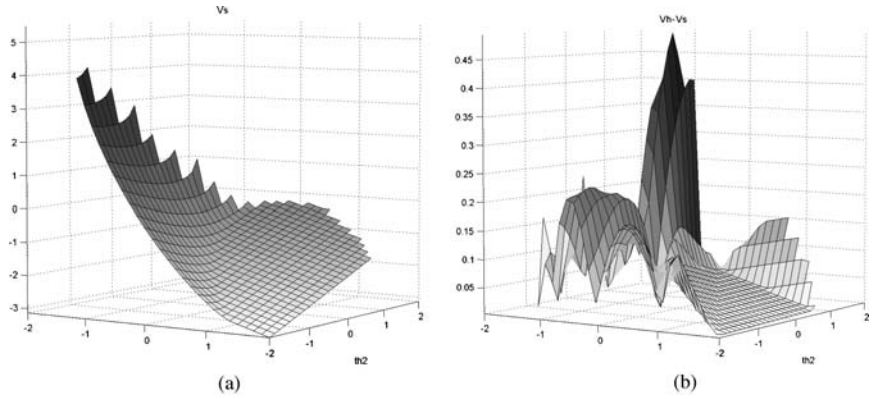

Figure 3. Multiparametric solution associated with problem (23). (a) Value function $V^*(\theta)$ and (b) Error $\hat{V}(\theta) - V^*(\theta)$.

5.2. *Approximate multiparametric geometric programming*

The approximate multiparametric programming approach developed earlier can be applied in principle to any multiparametric nonconvex programming problem that admits an equivalent convex reformulation. In particular, here we consider the class of multiparametric geometric programming problems, defined as follows.

A *posynomial* function $f : IR^n \times IR^m \mapsto IR$ has the form

$$f(x,\theta) = \sum_{k=1}^{K} c_k x_1^{a_{1k}} x_2^{a_{2k}} \dots x_n^{a_{nk}} \theta_1^{b_{1k}} \theta_2^{b_{2k}} \dots \theta_m^{b_{mk}},$$

where all a_{ik}, b_{ik}, and c_k coefficients are real numbers, and $c_k \geq 0$ for all i and k. If $K = 1$, function f is called *monomial*. A multiparametric geometric programming (GP) problem is defined as

$$\begin{aligned} V^*(\theta) = \quad & \min_x \quad f(x,\theta) \\ & \text{subject to} \quad g_i(x,\theta) \leq 1, \quad (i = 1,\dots,p) \\ & \qquad\qquad h_j(x,\theta) = 1, \quad (j = 1,\dots,q) \end{aligned} \tag{24}$$

where f and g_i are all posynomial functions, and h_j are all monomial functions [11].

By a change of variables and a transformation of the objective and constraint functions, the GP problem (24) can be transformed in the convex form

$$\begin{aligned} W^*(\phi) = \quad & \min_y \ \log\left(\sum_{k=1}^{K_0} e^{a'_{0k} y + b'_{0k}\phi + c_{0k}}\right) \\ & \text{subject to} \quad \log\left(\sum_{k=1}^{K_i} e^{a'_{ik} y + b'_{ik}\phi + c_{ik}}\right) \leq 0, \qquad (i = 1,\dots,p) \\ & \qquad\qquad Ay + B\phi + D = 0, \end{aligned} \tag{25}$$

where $\phi = \log\theta$ is the vector of parameters,[1] $y = \log x$ is the optimization vector, the real coefficients a_{ik}, b_{ik}, c_{ik} and matrices $A \in I\!R^{q\times n}$, $B \in I\!R^{q\times m}$, $D \in I\!R^{q}$ are obtained from f, g_i, h_j as detailed for instance in Chapter 4 of [8], and

$$\log V^*(\theta) = W^*(\log\theta).$$

As the objective function and the constraints in problem (25) are jointly convex in y and ϕ, we can apply the general solver developed in the previous sections. For a given $\epsilon > 1$, let $S_h = \{\phi \in \mathbb{R}^m : M_h^{-1}\left[\begin{smallmatrix}1\\ \phi\end{smallmatrix}\right] \geq 0\}$, with $h = 1, \ldots, L$, be the simplices in the ϕ-space generated by the recursive algorithm with maximum error $\log\epsilon$, and let $\hat{\Phi}$ be the union of all such simplices. The solver finds an approximate piecewise affine optimizer function $\hat{y} : \hat{\Phi} \mapsto \mathbb{R}^n$ and an approximate value function $\hat{W} : \hat{\Phi} \mapsto \mathbb{R}$ such that

$$\hat{W}(\phi) = \log\left(\sum_{k=1}^{K_0} e^{a'_{0k}\hat{y}(\phi)+b'_{0k}\phi+c_{0k}}\right)$$

with $0 \leq \hat{W}(\phi) - W^*(\phi) \leq \epsilon$ for all $\phi \in \hat{\Phi}$.

Coming back to the original multiparametric GP problem (24), let $\hat{\Theta} = \{\theta \in \mathbb{R}^m : e^\theta \in \hat{\Phi}\}$, and define the functions $\hat{x} : \hat{\Theta} \mapsto \mathbb{R}^n$ and $\hat{V} : \hat{\Theta} \mapsto \mathbb{R}$ such that $\hat{x}(\theta) = e^{\hat{y}(\log\theta)}$ and $\hat{V}(\theta) = e^{\hat{W}(\log\theta)}$. It is straightforward to prove that

$$\frac{\hat{V}(\theta)}{\epsilon} \leq V^*(\theta) \leq \hat{V}(\theta) \quad \text{for all } \theta \in \hat{\Theta}. \tag{26}$$

Equation (26) provides a *relative* approximation error, rather than an *absolute* one, for any choice of $\epsilon > 1$. Furthermore, the approximate feasible parameter set $\hat{\Theta}$ is such that $\hat{\Theta} = \cup_{h=1}^{L} Z_h$, where

$$Z_h = \left\{\theta \in \mathbb{R}^m : M_h^{-1}\begin{bmatrix}1\\ \log\theta\end{bmatrix} \geq 0\right\}, \quad (h = 1, \ldots, L)$$

and Z_h, Z_k have disjoint interiors for all $h \neq k$.

We remark that from the results in [24] it follows that the exact optimizer function $y^*(\phi)$ of problem (25) is continuous in ϕ, and therefore $x^*(\theta) = e^{y^*(\log\theta)}$ is also continuous. Thus, in the (nonconvex) multiparametric GP case the exact optimizer function is continuous.

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.

Note

1. For a given vector $x \in I\!R^n$, we denote by $\log x$ the vector $[\log x_1 \ldots \log x_n]'$ and by e^x the vector $[e^{x_1} \ldots e^{x_n}]'$.

References

1. C.B. Barber, D.P. Dobkin, and H. Huhdanpaa, "Qhull homepage," The Geometry Center, University of Minnesota, 1993. http://www.geom.umn.edu/software/qhull/.
2. A. Bemporad, F. Borrelli, and M. Morari, "Model predictive control based on linear programming—The explicit solution," IEEE Trans. Automatic Control, vol. 47, no. 12, pp. 1974–1985, 2002a.
3. A. Bemporad and C. Filippi, "Approximate multiparametric convex programming," In Proc. 42th IEEE Conf. on Decision and Control, Maui, Hawaii, USA, pp. 3185–3190, 2003a.
4. A. Bemporad and C. Filippi, "Suboptimal explicit RHC via approximate multiparametric quadratic programming," Journal of Optimization Theory and Applications, vol. 117, no. 1, pp. 9–38, 2003b.
5. A. Bemporad, M. Morari, V. Dua, and E.N. Pistikopoulos, "The explicit linear quadratic regulator for constrained systems," Automatica, vol. 38, no. 1, pp. 3–20, 2002b.
6. H.Y. Benson and R.J. Vanderbei, "Solving problems with semidefinite and related constraints using interior-point methods for nonlinear programming," Mathematical Programming, vol. 95, no. 2, pp. 279–302, 2003.
7. F. Borrelli, A. Bemporad, and M. Morari, "A geometric algorithm for multi-parametric linear programming," Journal of Optimization Theory and Applications, vol. 118, no. 3, pp. 515–540, 2003.
8. S. Boyd and L. Vandenberghe, "Convex optimization," Cambridge, MA: Cambridge University Press. http://www.stanford.edu/ boyd/cvxbook.html, 2004.
9. E.F. Camacho and C. Bordons, "Model predictive control," Advanced Textbooks in Control and Signal Processing, London: Springer-Verlag, 2nd edition, 2004.
10. T.H. Cormen, C.E. Leiserson, and R.L. Rivest, *Introduction to Algorithms*, Chapt. 5. New York: McGraw-Hill, 1990.
11. R.J. Duffin, E.L. Peterson, and C. Zener, "Geometric programming – theory and applications," New York: Wiley, 1967.
12. A.V. Fiacco, "Introduction to sensitivity and stability analysis in nonlinear programming," London, U.K., Academic Press, 1983.
13. A.V. Fiacco and J. Kyparisis, "Computable bounds on parametric solutions of convex problems," Mathematical Programming, vol. 40, pp. 213–221, 1988.
14. C. Filippi, "An algorithm for approximate multiparametric linear programming," Journal of Optimization Theory and Applications, vol. 120, no. 1, pp. 73–95, 2004.
15. T. Gal, *Postoptimal Analyses, Parametric Programming, and Related Topics*, Berlin: de Gruyter, 2nd edition, 1995.
16. D. Goldfarb and K. Scheinberg, "On parametric semidefinite programming," Applied Numerical Mathematics, vol. 29, pp. 361–377, 1999.
17. J.E. Goodman and J. O'Rourke (Eds.), "Handbook of discrete and computational geometry," Discrete Mathematics and Its Applications, New York: CRC Press, 1997.
18. E. Hansen, "Global optimization with data perturbation," Computers and Operations Research, vol. 11, pp. 97–104, 1984.
19. R. Horst and N.V. Thoai, "DC Programming: Overview," Journal of Optimization Theory and Applications, vol. 103, no. 1, pp. 1–43, 1999.
20. T.A. Johansen, "On multi-parametric nonlinear programming and explicit nonlinear model predictive control," In Proc. 41th IEEE Conf. on Decision and Control, Las Vegas, Nevada, USA, pp. 2768–2773, 2002.

21. T.A. Johansen, "Approximate explicit receding horizon control of constrained nonlinear systems," Automatica, vol. 40, no. 2, pp. 293–300, 2004.
22. T.A. Johansen and A. Grancharova, "Approximate explicit constrained linear model predictive control via orthogonal search tree," IEEE Trans. Automatic Control, vol. 48, no. 5, pp. 810–815, 2003.
23. J. Maciejowski, Predictive Control with Constraints, Harlow, UK: Prentice Hall, 2002.
24. O.L. Mangasarian and J.B. Rosen, "Inequalities for stochastic nonlinear programming problems," Operations Research, vol. 12, pp. 143–154, 1964.
25. D. Muñoz de la Peña, A. Bemporad, and C. Filippi, "Robust explicit MPC based on approximate multiparametric convex programming," In Proc. 43th IEEE Conf. on Decision and Control, Paradise Island, Bahamas, pp. 2491–2496, 2004.
26. K.G. Murty, "Computational complexity of parametric linear programming," Mathematical Programming, vol. 19, pp. 213–219, 1980.
27. S.M. Robinson, "Computable error bounds for nonlinear programming," Mathematical Programming, vol. 5, 235–242, 1973.
28. C. Rowe and J.M. Maciejowski, "An algorithm for multi-parametric mixed integer semidefinite optimisation," In Proc. 42th IEEE Conf. on Decision and Control, Maui, Hawaii, USA, pp. 3197–3202, 2003.
29. R. Seidel, "Exact upper bounds for the number of faces in d-dimensional Voronoi diagram," In P. Gritzmann and B. Sturmfels (Eds.): Applied Geometry and Discrete Mathematics—The Victor Klee Festschrift, DIMACS Series in Discrete Mathematics and Theoretical Computer Science. Providence, RI: American Mathematical Society, pp. 517–529, 1991.
30. M. Seron, J. DeDoná, and G. Goodwin, "Global analytical model predictive control with input constraints," In Proc. 39th IEEE Conf. on Decision and Control, Sydney, Australia, pp. 154–159, 2000.
31. P. Tøndel, T.A. Johansen, and A. Bemporad, "An algorithm for multi-parametric quadratic programming and explicit MPC solutions," Automatica, vol. 39, no. 3, 489–497, 2003.
32. L. Vandenberghe, S. Boyd, and B. Alkire, "SP — Software for semidefinite programming (version 1.1)," http://www.ee.ucla.edu/ vandenbe/sp.html, 1999.
33. L. Yepremyan and J. Falk, "Delaunay partitions in $\mathbb{R}^n$ applied to non-convex programs and vertex/facet enumeration problems," Computers and Operations Research, vol. 32, 793–812, 2005.