Pergamon

PII: S0005-1098(96)00050-7

Automatica, Vol. 32, No. 8, pp. 1217–1221, 1996

0005-1098/96 $15.00 + 0.00

Brief Paper

Guaranteed Tuning, with Application to Robust Control and Motion Planning*

LUC JAULIN† and ÉRIC WALTER‡

Key Words—Bounding methods; discrete-time systems; filtering; interval analysis; motion planning; nonlinear control; PID controllers; robust control; set values; structured uncertainty.

Abstract—Many design problems, e.g. in control theory, amount to tuning a parameter vector $\mathbf{c}$ so as to guarantee that specifications are met for all feasible values of some unknown perturbation vector $\mathbf{p}$. A new prototype algorithm for solving this guaranteed-tuning problem is proposed, and its convergence properties are established. It applies when the design specifications translate into a finite number of (possibly nonlinear) inequalities. Three test cases taken from the field of control are considered, namely the design of a PID controller robust to structured uncertainty, the control of a nonlinear discrete-time model with uncertain parameters and initial state, and a problem of motion planning, with obstacles to be avoided. Copyright © 1996 Elsevier Science Ltd.

1. *Introduction*

The problem to be considered is the choice of a value of some tuning parameter vector $\mathbf{c}$ in $\mathbb{C}$ that guarantees the satisfaction of a list of design specifications for all values of some unknown vector $\mathbf{p}$ in $\mathbb{P}$, where $\mathbb{P}$ and $\mathbb{C}$ are prior feasible sets for $\mathbf{p}$ and $\mathbf{c}$. This list of specifications is assumed to translate into a finite set of (possibly nonlinear) inequalities to be satisfied by $\mathbf{c}$ and $\mathbf{p}$. The problem can then be reformulated as

$$\text{Find } \textit{one } \mathbf{c} \in \mathbb{S}_c = \{\mathbf{c} \in \mathbb{C} \mid \forall \mathbf{p} \in \mathbb{P}, \mathbf{f}(\mathbf{c}, \mathbf{p}) > \mathbf{0}\}, \qquad (1)$$

where $\mathbf{f}$ is a vector function that can be evaluated via an algorithm, and the inequality is to be taken componentwise. Algorithms based on interval analysis for characterizing sets defined by inequalities can be found in Moore (1992) and Jaulin and Walter (1993). Problem (1) is at the same time more complex, because it involves a quantifier, and simpler, for the aim is only to find one feasible vector. This makes it possible to consider a larger number of tuning parameters.

Many design problems can be cast in the form (1). As examples from control theory or signal processing, one may mention the following.

Robust linear control. $\mathbf{c}$ may be the vector of the parameters of a controller, while $\mathbf{p}$ is that of the uncertain parameters of the process, only known to belong to some feasible set $\mathbb{P}$. The inequalities $\mathbf{f}(\mathbf{c}, \mathbf{p}) > \mathbf{0}$ to be satisfied may be associated with necessary and sufficient conditions for asymptotic stability, as provided for instance by the Routh criterion in the continuous-time case (Walter and Jaulin, 1994). Any collateral requirements on the properties of the controlled system (such as feasible domains for damping coefficients, static gain and stability margins) that can be expressed as a set of inequalities to be satisfied can be appended to $\mathbf{f}$. This situation will be illustrated by a test case in Section 5.1. The algorithm to be presented can be used for *design*, i.e. to choose the value of $\mathbf{c}$, contrary to other branch-and-bound algorithms, which can only be used for *analysis* (see e.g. Malan *et al.*, 1992).

Filtering. If $\mathbf{H}(\mathbf{c}, j\omega)$ is the transfer matrix of a filter with tuning parameters $\mathbf{c}$, the problem of computing a value of $\mathbf{c}$ that insures that $\mathbf{H}$ belongs to a given gabarit for all ω in a given range (see e.g. Rabiner and Gold, 1975) can be written in the form (1), where $p = \omega$. Often, if $\mathbf{c}$ corresponds to the physical parameters of the filter, $\mathbf{H}$ turns out to be nonlinear in $\mathbf{c}$, and no guaranteed method seems to be available in the literature to solve this problem.

Experiment design. When estimating the parameters $\mathbf{p}$ of a model from experimental data, the quality of the estimates depends on the procedure used for data collection. Assume that the experiment can be described by a vector $\mathbf{c}$ (which may, for instance, consist of measurement times). The most commonly used method (see e.g. Fedorov, 1972) for quantifying the quality of an experiment is to compute the determinant of the Fisher information matrix $\mathbf{F}(\mathbf{c}, \mathbf{p})$, which usually depends on the unknown parameters $\mathbf{p}$. One may be interested in finding $\mathbf{c}$ such that the determinant of the Fisher information matrix be larger than a given value δ for any possible value of $\mathbf{p}$, i.e. in solving (1) with $f(\mathbf{c}, \mathbf{p}) = \det \mathbf{F}(\mathbf{c}, \mathbf{p}) - \delta$.

Nonlinear discrete-time control of uncertain systems. Driving the state of a system in m steps to a feasible set (characterized by inequalities) for all feasible values of some uncertain vector $\mathbf{p}$ (parameters and/or initial conditions) can be written in the form (1), where $\mathbf{c}$ is the vector of the first m inputs. The technique to be presented in this paper thus makes it possible to combine nonlinearity, structured uncertainty and guaranteed results, as evidenced by the example treated in Section 5.2.

Motion planning. Let M_0 and M_1 be some given initial and final points. Let $M(\mathbf{c}, t)$ be a family of motions (in position space) such that for any $\mathbf{c} \in \mathbb{C}$, $M(\mathbf{c}, 0) = M_0$ and $M(\mathbf{c}, 1) = M_1$. Assume that avoidance of the obstacles is characterized by $\mathbf{g}(\cdot) > \mathbf{0}$. A feasible motion satisfies $\mathbf{g}(M(\mathbf{c}, t)) > \mathbf{0}$ for any $t \in [0, 1]$. Finding such a feasible motion amounts to solving (1) with $p = t$, $\mathbb{P} = [0, 1]$ and $\mathbf{f}(\mathbf{c}, p) = \mathbf{g}(M(\mathbf{c}, p))$. An illustrative example will be considered in Section 5.3.

In what follows, $\mathbf{f}$ is assumed to be continuous. For the sake of simplicity of exposition, $\mathbb{C}$ and $\mathbb{P}$ will be taken as axis-aligned boxes $\mathbb{c}_0$ and $\mathbb{p}_0$, but sets described by unions of such boxes could be considered as well. Our purpose is to find *one* value of $\mathbf{c}$ in the posterior feasible set for $\mathbf{c}$, defined by

$$\mathbb{S}_c = \{\mathbf{c} \in \mathbb{c}_0 \mid \forall \mathbf{p} \in \mathbb{p}_0, \mathbf{f}(\mathbf{c}, \mathbf{p}) > \mathbf{0}\} = \{\mathbf{c} \in \mathbb{c}_0 \mid \mathbf{f}(\mathbf{c}, \mathbb{p}_0) > \mathbf{0}\}.$$

Figure 1 evidences the fact that $\mathbb{S}_c$ may be disconnected.

Solving (1) can be seen as a problem of elimination of quantifiers, for which several formal approaches are available

* Received 9 November 1994; revised 26 September 1995; received in final form 1 March 1996. This paper was not presented at any IFAC meeting. This paper was recommended for publication in revised form by Associate Editor Kenko Uchida under the direction of Editor Tamer Başar. Corresponding author Professor Éric Walter. Tel. +33 1 69 85 17 21; Fax +33 1 69 41 30 60; E-mail walter@lss.supelec.fr.

† University of Angers, Faculté des Sciences, 2, Boulevard Lavoisier, 49045 Angers Cedex 01, France.

‡ Laboratoire des Signaux et Systèmes, CNRS-SUPÉLEC, Plateau de Moulon, 91192 Gif-sur-Yvette Cedex, France.

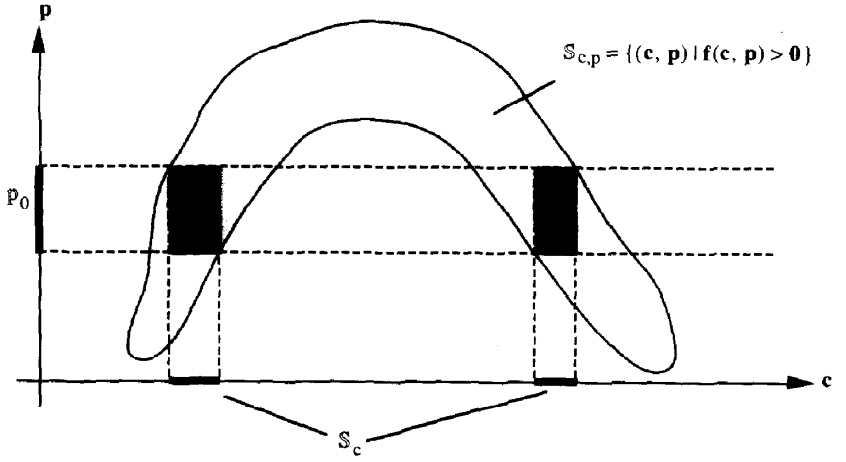

Fig. 1. The posterior feasible set $\mathbb{S}_c$ for $\mathbf{c}$ may be disconnected.

(see e.g. (Collins, 1975; Davenport *et al.*, 1987), most of which are based on Tarski's algorithm (Tarski, 1951). In practice, only the simplest problems can be considered. The present paper is a first attempt towards solving (1) in a numerical but guaranteed way.

This paper is organized as follows. Section 2 recalls the very few basic tools of interval analysis (see e.g. Moore, 1979) needed to understand the new algorithm to be proposed, and introduces the notation used. Section 3 presents the algorithm. Section 4 analyses its convergence, and test cases are treated in Section 5.

2. *Interval analysis*

A *box* or *vector interval* $\mathbb{x}$ of $\mathbb{R}^n$ is the Cartesian product of n real intervals $\mathbb{x} = [x_1^-, x_1^+] \times \ldots \times [x_n^-, x_n^+] = \mathbb{x}_1 \times \ldots \times \mathbb{x}_n$. The set of all boxes of $\mathbb{R}^n$ will be denoted by $\mathbb{IR}^n$. A *principal plane* of $\mathbb{x}$ is a symmetry plane of $\mathbb{x}$ normal to a side of maximum length. The notation $\mathbb{x} > \mathbf{0}$ will mean that any component of any vector $\mathbf{x}$ in the box $\mathbb{x}$ is positive. Let $\mathbf{f}: \mathbb{R}^n \to \mathbb{R}^p$ be a vector function, the set-valued function $\mathbb{f}: \mathbb{IR}^n \to \mathbb{IR}^p$ will be an *inclusion function* of $\mathbf{f}$ if and only if it satisfies $\mathbf{f}(\mathbb{x}) \subset \mathbb{f}(\mathbb{x})$ for any $\mathbb{x}$ of $\mathbb{IR}^n$. It will be *inclusion-monotonic* if $\mathbb{x} \subset \mathbb{y} \Rightarrow \mathbb{f}(\mathbb{x}) \subset \mathbb{f}(\mathbb{y})$ and *convergent* if $w(\mathbb{x}) \to 0 \Rightarrow w(\mathbb{f}(\mathbb{x})) \to 0$, where $w(\mathbb{x})$ is the *width* of the box $\mathbb{x}$, i.e. the length of its largest side(s). The derivation of an inclusion-monotonic and convergent inclusion function associated with any continuous function defined by an explicit formal expression is usually very simple. It is routinely performed by commercially available languages such as PASCAL XSC (see e.g. Klatte *et al.*, 1992). Consider, for instance, the function $f(x_1, x_2) = x_1^2 + \sin(x_1 * x_2)$. A natural inclusion function for f is $\mathbb{f}(\mathbb{x}_1, \mathbb{x}_2) = \mathbb{x}_1^2 + \sin(\mathbb{x}_1 * \mathbb{x}_2)$. If $\mathbb{x}_1 = [-1, 2]$ and $\mathbb{x}_2 = [0, \pi/4]$ then

$$\begin{aligned}\mathbb{f}(\mathbb{x}_1, \mathbb{x}_2) &= [-1, 2]^2 + \sin([-1, 2] * [0, \pi/4]) \\ &= [0, 4] + \sin([-\pi/4, \pi/2]) \\ &= [0, 4] + [-\sqrt{2}/2, 1] \\ &= [-\sqrt{2}/2, 5].\end{aligned}$$

In practice, numerical rounding must be taken into account, so that inclusion functions can no longer be strictly convergent.

3. *Algorithm*

The algorithm to be presented for solving (1) consists of two procedures. Its main procedure (FPS, for 'feasible point searcher') attempts to find a vector $\mathbf{c}$ belonging to a set $\mathbb{S}_c$ included in some prior box $\mathbb{c}_0$. It relies on the procedure CSC (for 'computable sufficient conditions') for proving that a vector $\mathbf{c}$ belongs to $\mathbb{S}_c$ or that a box $\mathbb{c}$ has an empty intersection with $\mathbb{S}_c$. In what follows, $\mathbb{f}(\mathbb{c}, \mathbb{p})$ is a convergent and inclusion-monotonic inclusion function of $\mathbf{f}(\mathbf{c}, \mathbf{p})$.

3.1. *Procedure CSC.* The procedure CSC attempts to prove either that the center of a given box $\mathbb{c}$ belongs to $\mathbb{S}_c$ (in which case, the problem of finding a feasible $\mathbf{c}$ is solved) or that the box $\mathbb{c}$ does not intersect $\mathbb{S}_c$ (in which case, this box can be dropped from further consideration). It is based on two easy-to-compute sufficient conditions. The first one is $\mathbb{f}(\mathbf{c}, \mathbb{p}_0) > \mathbf{0}$, which implies that $\mathbf{f}(\mathbf{c}, \mathbb{p}_0) > \mathbf{0}$, and therefore that $\mathbf{c} \in \mathbb{S}_c$. The vector $\mathbf{c}$ is then a solution of the guaranteed-tuning problem. In practice, it will usually be necessary to split the box $\mathbb{p}_0$ into several boxes, so as to improve the accuracy of the inclusion function by taking advantage of its convergence. The second sufficient condition is $\exists i \mid \mathbb{f}_i(\mathbb{c}, \mathbf{p}) \leq 0$ for some $\mathbf{p}$ in $\mathbb{p}_0$, which implies that $\exists i \mid f_i(\mathbb{c}, \mathbf{p}) \leq 0$, and therefore that $\mathbb{c} \cap \mathbb{S}_c = \emptyset$. The box $\mathbb{c}$ then contains no solution of the guaranteed-tuning problem. These two easy-to-compute conditions are illustrated by Fig. 2.

CSC uses a *stack* (i.e. a first-in-last-out list, think to a stack of plates) in which subboxes of $\mathbb{p}_0$ are stored. The Boolean variable *go* is true only if CSC is still allowed to attempt to prove that center $(\mathbb{c}) \in \mathbb{S}_c$. The inputs for CSC are the feasible box $\mathbb{p}_0$ for $\mathbf{p}$ and the current box $\mathbb{c}$ of interest for $\mathbf{c}$. CSC is initialized as follows:

$$\text{stack} := \{\mathbb{p}_0\}, \quad go := \text{true},$$

and its iteration is given by

Step 1: Unstack into $\mathbb{p}$.
Step 2: If $\exists i \mid \mathbb{f}_i(\mathbb{c}, \text{center } (\mathbb{p})) \leq 0$ then return '$\mathbb{c} \cap \mathbb{S}_c = \emptyset$'. End.
Step 3: If $\mathbb{f}(\text{center } (\mathbb{c}), \mathbb{p}) > \mathbf{0}$ then go to Step 6.
Step 4: If $w(\mathbb{p}) < w(\mathbb{c})$ then *go* := false. Go to Step 6.
Step 5: Bisect $\mathbb{p}$ along a principal plane and stack the two resulting boxes.
Step 6: If the stack is not empty then go to Step 1.
Step 7: If *go* = true then return 'center $(\mathbb{c}) \in \mathbb{S}_c$'. End
Step 8: Return 'No conclusion has been reached, $\mathbb{c}$ is indeterminate'. End.

If CSC terminates on Step 2 then there exists $\mathbf{p} \in \mathbb{p}_0$ such that $\mathbf{f}(\mathbf{c}, \mathbf{p}) > \mathbf{0}$ is not satisfied for any $\mathbf{c}$ in $\mathbb{c}$. This implies that $\mathbb{c} \cap \mathbb{S}_c = \emptyset$. If CSC terminates on Step 7 then *go* = true and the stack is empty. It means that $\mathbb{p}_0$ has been partitioned into boxes $\mathbb{p}$ that all satisfy $\mathbb{f}(\text{center } (\mathbb{c}), \mathbb{p}) > \mathbf{0}$, so that center $(\mathbb{c}) \in \mathbb{S}_c$. If CSC terminates on Step 8, then *go* = false and the stack is empty; CSC has failed to reach any conclusion. The purpose of Step 4 is to avoid uselessly splitting $\mathbb{p}_0$ *ad infinitum*. Since a box that satisfies $w(\mathbb{p}) < w(\mathbb{c})$ will never be bisected, all the subboxes $\mathbb{p}$ of $\mathbb{p}_0$ generated by CSC will satisfy $w(\mathbb{p}) \geq w(\mathbb{c})/2$. Only a finite number of nonoverlapping subboxes can thus be generated. CSC is therefore a finite procedure.

3.2. *Procedure FPS.* The main procedure FPS organizes a systematic examination of the prior feasible set $\mathbb{c}_0$ for $\mathbf{c}$ by CSC. If CSC has proved that the center of the current box $\mathbb{c}$ is feasible then FPS terminates. If the current box $\mathbb{c}$ has been proved by CSC to contain no feasible point then it is eliminated. If no conclusion has been reached for the box $\mathbb{c}$

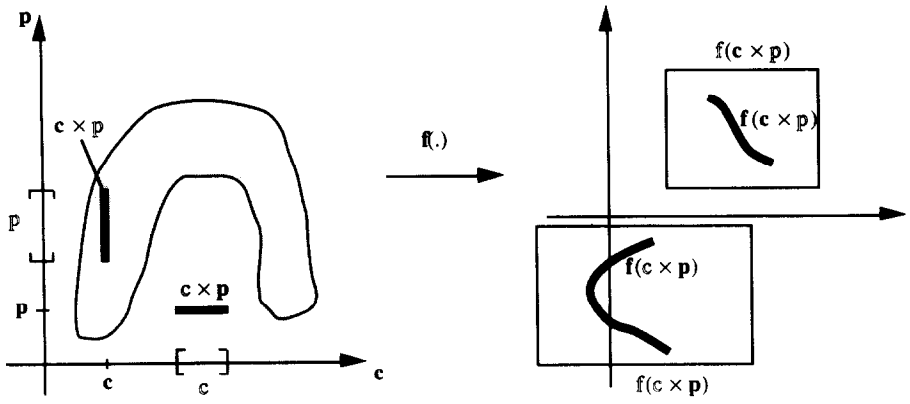

Fig. 2. The two sufficient conditions used by CSC to establish properties of the box c.

then c will be split into two boxes, to be stored in a *queue* (i.e. a first-in-first-out list) for further consideration. Because of the convergence property of inclusion functions, such as splitting will increase the probability of reaching a conclusion at a later stage. FPS is initialized as

$$\text{queue} := \emptyset, \quad \mathbb{c} := \mathbb{c}_0,$$

and its iteration is given by

Step 1: Call CSC. If it returns 'center ($\mathbb{c}$) $\in \mathbb{S}_c$' then return center ($\mathbb{c}$). End.
Step 2: If CSC returns '$\mathbb{c} \cap \mathbb{S}_c = \emptyset$' then go to Step 4.
Step 3: Bisect $\mathbb{c}$ along a principal plane and push the two resulting boxes into the queue.
Step 4: If queue $\neq \emptyset$ then pull its first box into $\mathbb{c}$ and go to Step 1.
Step 5: Return 'No feasible vector exists, $\mathbb{S}_c$ is empty'.

Convergence results of Section 4 indicate that FPS will almost always terminate. If it terminates on Step 5 then the initial box $\mathbb{c}_0$ has been partitioned into a finite union of boxes, none of which contains any feasible vector. If it terminates on Step 1 then a feasible point **c** has been found at the center of the current box. Since only unfeasible boxes have been discarded, at each iteration the union of all boxes in the queue contains $\mathbb{S}_c$. Since a queue strategy is employed, it is always one of the largest boxes that is bisected, so that the size of all boxes in the queue tends to zero. This property will be very useful in proving the convergence of the algorithm in the next section.

4. *Convergence analysis*

The approach followed to find a feasible **c** is to use the algorithm FPS + CSC. In this section, two theorems about the convergence properties of this algorithm will be given. In their proofs, k denotes the iteration counter of FPS.

Theorem 1. If $\mathbb{S}_c \neq \emptyset$ then FPS + CSC will find a feasible **c** in a finite number of iterations.

Proof. This is by contradiction. If $\mathbb{S}_c \neq \emptyset$ then there exists a vector $\mathbf{c}_{in} \in \mathbb{S}_c$, i.e. such that $\mathbf{f}(\mathbf{c}_{in}, \mathbb{p}_0) > \mathbf{0}$. The set $\mathbb{S}_{c,p} = \{(\mathbf{c}, \mathbf{p}) \mid \mathbf{f}(\mathbf{c}, \mathbf{p}) > \mathbf{0}\} = \mathbf{f}^{-1}(\mathbb{O})$, where $\mathbb{O}$ is the strictly positive orthant in the image space of **f**, is open since it is the reciprocal image (in a set-theoretical sense) of an open set by a continuous function. Then there exists a box $\mathbb{c}_{in}$, with center $\mathbf{c}_{in}$, such that $\mathbf{f}(\mathbb{c}_{in}, \mathbb{p}_0) > \mathbf{0}$, i.e. $\mathbb{c}_{in}$ is included in $\mathbb{S}_c$. Assume that FPS + CSC never stops. Since all the boxes in the queue have a width tending to zero and $\mathbb{c}_{in}$ is a subset of the union of all boxes stored in the queue, there exists a nested subsequence $\{\mathbb{c}(m)\}$ of the sequence of current boxes $\{\mathbb{c}(k)\}$, such that all boxes of $\{\mathbb{c}(m)\}$ belong to $\mathbb{c}_{in}$. Since $\mathbb{c}(m)$ is indeterminate, there exists a box $\mathbb{p}(m)$ associated to $\mathbb{c}(m)$ that switched *go* to false during Step 4 of CSC. Let $\mathbf{c}(m)$ and $\mathbf{p}(m)$ be the centers of the boxes $\mathbb{c}(m)$ and $\mathbb{p}(m)$. From Step 3 of CSC, and since $\mathbf{c}(m) \in \mathbb{c}_{in}$, the box $\mathbf{f}(\mathbf{c}(m), \mathbb{p}(m))$ has at least one component that contains zero. Since $\mathbf{f}$ is a converging inclusion function and $w(\mathbb{p}(m))$ tends to zero as m tends to infinity, $\mathbf{f}(\mathbf{c}(m), \mathbf{p}(m))$ has at least one component that tends to zero. This is impossible, since $\mathbf{f}(\mathbb{c}_{in}, \mathbb{p}_0) > \mathbf{0}$, $\mathbf{c}(m) \in \mathbb{c}_{in}$ and $\mathbf{p}(m) \in \mathbb{p}_0$. □

Theorem 2. If there exists a vector $\boldsymbol{\varepsilon} > \mathbf{0}$ such that $\mathbb{S}_{c,\varepsilon} = \{\mathbf{c} \mid \mathbf{f}(\mathbf{c}, \mathbb{p}_0) > -\boldsymbol{\varepsilon}\}$ is empty then FPS + CSC will prove that $\mathbb{S}_c$ is empty in a finite number of iterations.

Proof. This is also by contradiction. Assume that FPS + CSC never stops. Then there exists an infinite nested subsequence of boxes $\{\mathbb{c}(m)\}$ in the queue that accumulates on some vector **c**. Assume that

$$\exists (\mathbf{p} \in \mathbb{p}_0, i \in 1, \ldots, \dim \mathbf{f}) \mid \mathbf{f}_i(\mathbf{c}, \mathbf{p}) < 0. \tag{2}$$

Since the conditions in Steps 2 and 3 of CSC cannot be satisfied for any $\mathbb{c}$ containing **c** and $\mathbb{p}$ containing **p**, when CSC is called with the argument $\mathbb{c}(m)$, it generates a box $\mathbb{p}(m)$ such that $\mathbf{p} \in \mathbb{p}(m)$ and $w(\mathbb{p}(m)) < w(\mathbb{c}(m))$. Since $\mathbf{f}$ is convergent, there exists an m such that $\mathbf{f}_i(\mathbb{c}(m), \text{center}(\mathbb{p}(m))) < 0$, and then $\mathbb{c}(m)$ would be eliminated during Step 2 of CSC. This is impossible, since **c** is an accumulation vector. Condition (2) is therefore false, i.e. **c** satisfies $\mathbf{f}(\mathbf{c}, \mathbb{p}_0) \geq \mathbf{0}$. Then $\forall \boldsymbol{\varepsilon} > \mathbf{0}, \mathbf{f}(\mathbf{c}, \mathbb{p}_0) > -\boldsymbol{\varepsilon}$, so that $\mathbf{c} \in \mathbb{S}_{c,\varepsilon}$, which is inconsistent with the assumption of Theorem 2. □

Remark. If FPS + CSC never stops, then, from Theorem 1, $\mathbb{S}_c = \mathbb{S}_{c,0} = \emptyset$, whereas, from Theorem 2, for any $\boldsymbol{\varepsilon} > \mathbf{0}$, $\mathbb{S}_{c,\varepsilon} \neq \emptyset$. There is therefore a nongeneric discontinuity of the solution set with respect to an infinitesimal enlargement of $\mathbb{S}_c$ into $\mathbb{S}_{c,\varepsilon}$. This only happens in atypical situations where $\exists \mathbf{c} \mid \mathbf{f}(\mathbf{c}, \mathbb{p}_0) \geq \mathbf{0}$ but $\nexists \mathbf{c} \mid \mathbf{f}(\mathbf{c}, \mathbb{p}_0) > \mathbf{0}$, as illustrated by Fig. 3.

5. *Test cases*

5.1. *Robust linear control.* Consider the uncertain system described by the transfer function

$$H(s, \mathbf{p}) = \frac{K\omega_0^2}{(1 + Ts)(s^2 + 2z\omega_0 s + \omega_0^2)},$$

where $\mathbf{p} = (z, T, \omega_0, K)^T \in \mathbb{p}_0 = [0.95, 1.05] \times [-1.05, -0.95] \times [0.95, 1.05] \times [0.95, 1.05]$. This system is to be asymptotically stabilized by a PID controller $C(s, \mathbf{c}) = (c_1 + c_2 s + c_3 s^2)/s$ inserted in the forward path, with a negative unity feedback. The problem is therefore to find a vector $\mathbf{c} = (c_1, c_2, c_3)^T$ that ensures the asymptotic stability of the controlled model for any vector **p** in the box $\mathbb{p}_0$. The closed-loop characteristic polynomial can be written as

$$\begin{aligned} P_{\mathbf{p},\mathbf{c}}(s) &= s^4 + (2z\omega_0 + T^{-1})s^3 + (2z\omega_0 T^{-1} + \omega_0^2(1 + c_3 K T^{-1}))s^2 \\ &\quad + \omega_0^2(1 + c_2 K)T^{-1}s + \omega_0^2 K c_1 T^{-1}. \end{aligned}$$

Using, for instance, the Routh criterion to obtain necessary and sufficient conditions for asymptotic stability under the form of inequalities, one gets the formal expression for $\mathbf{f}(\mathbf{c}, \mathbf{p})$

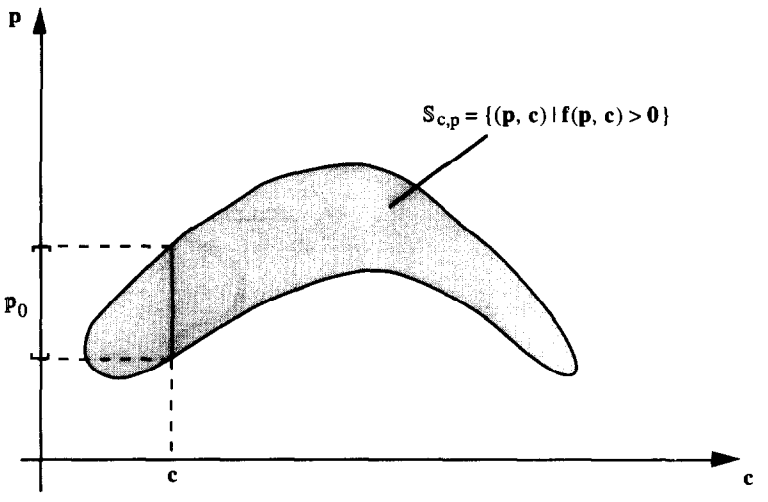

Fig. 3. Atypical situation where FPS + CSC would reach no conclusion.

(Walter and Jaulin, 1994). In the prior box $\mathbb{c}_0 = [-10, 10]^3$, in 10 s on a Compaq 386/33, the algorithm FPS + CSC finds the robustly stabilizing controller

$$C(s) = -\frac{0.625 + 3.75s + 8.75s^2}{s}.$$

This controller is *guaranteed* to stabilize all systems $H(s, \mathbf{p})$ such that $\mathbf{p} \subset \mathbb{p}_0$.

5.2. *Nonlinear discrete-time control.* Consider the discrete-time state-space model

$$x_1(k+1) = x_1(k) * \cos(a * x_1(k) * x_2(k)) + b(k) + u(k),$$
$$x_2(k+1) = 3x_1^2(k) - \sin((b(k) + u(k)) * x_2(k)),$$
$$\mathbf{x}(0) = \begin{pmatrix} x_1(0) \\ x_2(0) \end{pmatrix}.$$

where the parameter a, input noise $b(k)$ and initial state $\mathbf{x}(0)$ are only assumed to satisfy

$$a \in \mathbb{a} = [0.95, 1.05], \quad b(k) \in \mathbb{b}(k) = [-0.02, 0.02] \quad \forall k,$$
$$x_1(0) \in \mathbb{x}_1(0) = [0.98, 1.02], \quad x_2(0) \in \mathbb{x}_2(0) = [1.98, 2.02].$$

Driving the state into the (open) box $\mathbb{x}_f =]x_f^-, x_f^+[^2 =]-0.2, 0.2[^2$ in two steps amounts to solving (1) where $\mathbf{c} = (u(0), u(1))^T$ and $\mathbf{p} = (a, b(0), b(1), x_1(0), x_2(0))^T$. The function $\mathbf{f}(\mathbf{c}, \mathbf{p})$ is computed by the pseudocode

```
For k := 0 to 1 do
  begin
    x1(k+1) := x1(k) * cos(a * x1(k) * x2(k)) + b(k) + u(k);
    x2(k+1) := 3x1^2(k) - sin((b(k) + u(k)) * x2(k));
  end;
f(c, p) := (x1(2) - xf^-, x2(2) - xf^-, xf^+ - x1(2), xf^+ - x2(2))^T;
```

For a prior feasible domain for the controls $\mathbb{c}_0 = [-1, 1]^2$, in less than 11 s, the algorithm produces the set of boxes presented in Fig. 4. All grey boxes have been eliminated. The union of all white boxes and the black box is guaranteed to contain $\mathbb{S}_c$. The center $\mathbf{c} = (u(0), u(1))^T = (0.65625, -0.21875)^T$ of the black box has been proved to solve (1).

5.3. *Motion planning.* A point M in the plane is to be moved from $M(0) = M_0$ to $M(1) = M_1$, where $M_0 = (-1, -0.6)^T$ and $M_1 = (6, 0)^T$. For any $t \in [0, 1]$, $M(t) = (x, y)^T$ must satisfy

$$(x - 4.8)^2 + (y - 1)^2 - 1 > 0 \quad \text{and} \quad y - \sin(x) > 0.$$

$M(t)$ is tentatively chosen as a polynomial of degree 3. It can

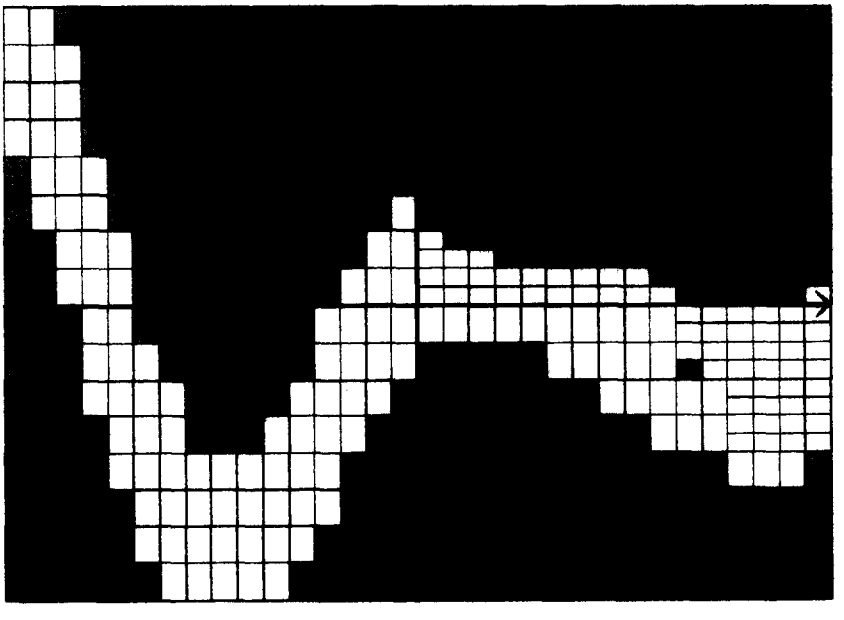

Fig. 4. Set of boxes generated by the algorithm in the control space $(u(0), u(1))$ for the example of Section 5.2. The frame corresponds to the search domain $[-1, 1]^2$.

therefore be written as a linear combination of the four Bernstein polynomials of degree 3 (see e.g., Lorentz, 1953)

$$B_0^3 = (1-t)^3, \quad B_1^3 = 3t(1-t)^2, \quad B_2^3 = 3t^2(1-t), \quad B_3^3 = t^3.$$

Bernstein polynomials have been chosen here because they are known to lead to a better conditioning of interpolation problems than the canonical basis for polynomials. This has been confirmed by numerical experimentation. Taking the initial and final positions into account, one gets

$$M(t) = M_0 B_0^3(t) + AB_1^3(t) + BB_2^3(t) + M_1 B_3^3(t),$$

where $A = (a_x, a_y)^T$ and $B = (b_x, b_y)^T$ are two control points to be found. This problem has the form (1), with $\mathbb{P} = [0, 1]$ and $\mathbf{f}(\mathbf{c}, p)$ computed by the followng pseudocode, with $p = t$ and $\mathbf{c} = (a_x, a_y, b_x, b_y)^T$:

$$x := x_0 B_0^3(t) + a_x B_1^3(t) + b_x B_2^3(t) + x_1 B_3^3(t);$$

$$y := y_0 B_0^3(t) + a_y B_1^3(t) + b_y B_2^3(t) + y_1 B_3^3(t);$$

$$\mathbf{f}(\mathbf{c}, p) := ((x - 4.8)^2 + (y - 1)^2 - 1, y - \sin(x))^T.$$

For $\mathbf{c}_0 = [-10, 10]^4$, the algorithm finds, in 101 s, $A = (2.5, 7.5)^T$ and $B = (2.5, -2.5)^T$, which correspond to the trajectory M_0M_1 indicated in Fig. 5, together with the control points A and B. In Step 4 of CSC, $w(\mathbb{P}) < w(\mathbf{c})$ was replaced by $w(\mathbb{P}) < 0.005\, w(\mathbf{c})$, to favor the analysis of the one-dimensional p space over that of the four-dimensional $\mathbf{c}$ space. This reduced the computing time by an order of magnitude. Note that if no feasible third-degree polynomial had been found, one might have increased the order so as to get more degrees of freedom.

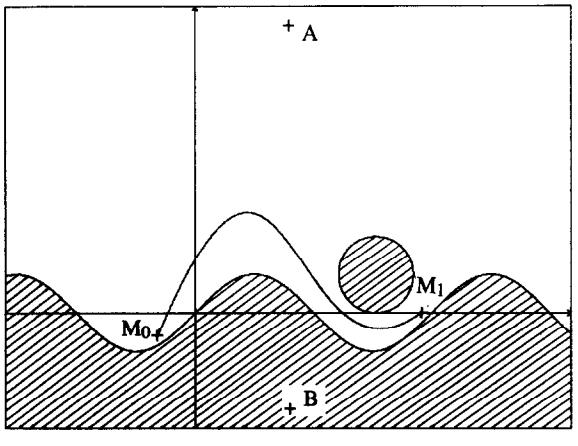

Fig. 5. Solution trajectory for the motion-planning problem of Section 5.3. The frame corresponds to $[-5, 10] \times [-3, 8]$.

6. *Conclusions*

Many design problems, including control and signal processing problems, can be formulated in the framework of guaranteed tuning, a special class of problems combining inequalities and quantifiers. Contrary to the formal approaches based on computer algebra usually used in this context, a prototype numerical algorithm based on interval analysis has been proposed to solve such problems in a guaranteed way. Under quite general conditions, it has been shown either to find a feasible tuning or to prove that none existed in a finite number of steps. The worst-case complexity of this algorithm should be exponential in $\dim \mathbf{c}$ and $\dim \mathbf{p}$, but a detailed analysis remains to be carried out. More efficient algorithms based on the same principles are presently under study.

Guaranteed tuning is but one example of a general class of problems of set characterization involving optimization, nonlinear inequalities and quantifiers, for which interval analysis should be much helpful because of its ability to produce guaranteed results even in a nonlinear context.

Acknowledgements—The authors wish to thank Michel Petitot and Olivier Didrit for helpful discussions.

References

Collins, G. E. (1975). Quantifier elimination for real closed fields by cylindrical algebraic decomposition. In H. Brakhage (Ed.), *Proc. 2nd GI Conf. on Automata Theory and Formal Languages*, Kaiserslautern, pp. 134–183. Lecture Notes in Computer Science, Vol. 33, Springer-Verlag, Berlin.

Davenport, J., Y. Siret and E. Tournier (1987). *Calcul formel*. Masson, Paris.

Fedorov, V. V. (1972). *Theory of Optimal Experiments*. Academic Press, New York.

Jaulin, L. and E. Walter (1993). Set inversion via interval analysis for nonlinear bounded-error estimation, *Automatica*, **29**, 1053–1064.

Malan, S., M. Milanese, M. Taragna and J. Garloff (1992). B^3 algorithm for robust performances analysis in presence of mixed parametric and dynamic perturbations. In *Proc. 31st IEEE Conf. on Decision and Control*. Tucson, AZ, pp. 128–133.

Klate, R., U. W. Kulisch, M. Neaga, D. Ratz and C. Ullrich (1992), *PASCAL-XSC: Language Reference with Examples*. Springer-Verlag, Heidelberg.

Lorentz, G. G. (1953). *Bernstein Polynomials*, 1st ed. University of Toronto Press, Toronto (2nd ed., Chelsea, New York, 1986).

Moore, R. E. (1979). *Methods and Applications of Interval Analysis*. SIAM, Philadelphia.

Moore, R. E. (1992). Parameter sets for bounded-error data. *Math. Comput. Simul.*, **34**, 113–119.

Rabiner, L. R. and B. Gold (1975). *Theory and Application of Digital Signal Processing*. Prentice-Hall, Englewood Cliffs, NJ.

Tarski, A. (1951). *A Decision Method for Elementary Algebra and Geometry*, 2nd ed. University of California Press, Berkeley.

Walter, E. and L. Jaulin (1994). Guaranteed characterization of stability domains via set inversion. *IEEE Trans. Autom. Control*, **AC-49**, 886–889.