Proceedings of the 41st IEEE
Conference on Decision and Control
Las Vegas, Nevada USA, December 2002

Interval methods for nonlinear identification and robust control

Luc Jaulin[1] Isabelle Braems[2] Eric Walter[2]

Abstract

Key ideas of interval analysis and constraint propagation are presented and applied to two problems frequently encountered in control. The first one is the guaranteed characterization of the set of all parameter vectors that are consistent with experimental data up to bounds on the acceptable errors. The second one is the guaranteed characterization of the set of all PI controllers robustly stabilizing a set of models that may have been obtained as the solution to the first problem.

1 Introduction

The aim of this paper is to explain how *interval constraint propagation* (ICP) can be used to solve globally and reliably some difficult nonlinear problems of automatic control. ICP, first presented independently by Cleary [1] and Davis [2], combines *interval computation* [3] and *constraint propagation* [4]. The resulting algorithms are very easy to understand and implement, and lead to guaranteed results even when the functions involved are nonlinear, although only numerical computations are used. Moreover, these algorithms may be very efficient in high dimensions as bisection is avoided whenever possible. Interval constraint propagation will be briefly recalled in Section 2. Its application to bounded-error estimation will be illustrated on a simple example in Section 3. A branch-and-bound algorithm to compute the projection of a set defined by inequalities is given in Section 4. This algorithm will then be used in Section 5, to characterize the set of all PI controllers that robustly stabilize a system, the uncertainty of which may have been characterized by bounded-error estimation as suggested in Section 3.

2 Interval constraint propagation

To understand the basic idea of interval constraint propagation, consider three variables x, y and z and assume that they belong to some prior feasible domains $x \in [x^-, x^+] = [1,5]$, $y \in [y^-, y^+] = [3,8]$, $z \in [z^-, z^+] = [-1,6]$. Assume also that these three variables are linked by the constraint

$$\text{ADD} : z = x + y. \tag{1}$$

Remark 1 *The constraint* ADD *is usually defined in a more formal way as a subset of* $\mathbb{R}^3$*:* ADD $\triangleq \{(x,y,z) \in \mathbb{R}^3 \mid z = x+y\}$*, but generally written as in (1) for short. Since* ADD *is a subset of* $\mathbb{R}^3$ *(or equivalently, since the addition involves three variables),* ADD *is said to be a ternary constraint.*

ADD can be used to contract the *prior* feasible domains for x, y and z by removing inconsistent values. For instance, the value -1 for z is inconsistent with ADD because $\forall x \in [1,5]$ and $\forall y \in [3,8]$, $x + y \neq -1$. It is a very simple matter to contract the feasible domains for the variables by taking into account the constraint ADD and the fact that

$$\begin{aligned} [x^-, x^+] + [y^-, y^+] &= [x^- + y^-, x^+ + y^+], \\ [x^-, x^+] - [y^-, y^+] &= [x^- - y^+, x^+ - y^-]. \end{aligned}$$

(*i*) For the variable z: since $x \in [1,5]$ and $y \in [3,8]$ then $x+y \in [4,13]$. Now, since $z = x+y$ and $z \in [-1,6]$, then $z \in [-1,6] \cap [4,13] = [4,6]$. (*ii*) For the variable x: since $z \in [4,6]$ and $y \in [3,8]$ then $z-y \in [4-8, 6-3] = [-4,3]$. Now, since $x = z - y$ and $x \in [1,5]$, then $x \in [1,5] \cap [-4,3] = [1,3]$. (*iii*) For the variable y: since $z \in [4,6]$ and $x \in [1,3]$ then $z - x \in [4-3, 6-1] = [-1,5]$. Now, since $y = z - x$ and $y \in [3,8]$ then $y \in [3,8] \cap [-1,5] = [3,5]$.

The domains for x, y and z have thus been contracted into $[x] = [1,3]$, $[y] = [3,5]$ and $[z] = [4,6]$. ADD can now be satisfied for any instantiation of anyone of these variables inside its domain, provided that the values of the other two variables are suitably chosen.

The contraction above can be described by the procedure PADD (for *Projection of the constraint* ADD).

Algorithm PADD(inout: $[z], [x], [y]$)	
1	$[z] := [z] \cap ([x] + [y])$;
2	$[x] := [x] \cap ([z] - [y])$;
3	$[y] := [y] \cap ([z] - [x])$.

PADD is a *projection procedure* [5] because it computes the projections onto the x, y and z axes of the set ADD $\cap ([x] \times [y] \times [z])$.

[1]LISA, University of Angiers, 62 avenue Notre Dame du lac,49000 Angiers, France, e-mail: jaulin@univ-angers.fr
[2]L2S, CNRS-Supelec-University of Paris-Sud, 91192 Gif-sur-Yvette, France, Isabelle.Braems,Eric.Walter@lss.supelec.fr

0-7803-7516-5/02/$17.00 ©2002 IEEE

The same approach can be used for other ternary constraints. For instance, the projection of MULT $\triangleq \{(x,y,z) \in \mathbb{R}^n \mid z = x * y\}$ leads to

Algorithm PMULT(inout: $[z],[x],[y]$)
1 $[z] := [z] \cap ([x] * [y])$;
2 $[x] := [x] \cap ([z] * 1/[y])$;
3 $[y] := [y] \cap ([z] * 1/[x])$.

Computations in PMULT are again trivial, as:

$[x^-,x^+] * [y^-,y^+] = [\min(x^-y^-, x^-y^+, x^+y^-, x^+y^+),$
$\max(x^-y^-, x^-y^+, x^+y^-, x^+y^+)],$
$\frac{1}{[x^-,x^+]} = \mathbb{R}$ if $0 \in [x^-,x^+]$ and $[\frac{1}{x^+}, \frac{1}{x^-}]$ otherwise.

Elementary functions can be interpreted as binary constraints and receive a similar treatment. For instance, the projection of the binary constraint EXP $\triangleq \{(x,y) \in \mathbb{R}^n \mid y = \exp(x)\}$ yields

Algorithm PEXP(inout: $[y],[x]$)
1 $[y] := [y] \cap \exp([x])$;
2 $[x] := [x] \cap \ln([y])$.

In PEXP,

$$\begin{aligned} \exp([x^-,x^+]) &= [\exp(x^-), \exp(x^+)], \\ \ln([x^-,x^+]) &= \emptyset \text{ if } x^+ \leq 0, \\ &= [\ln(x), \ln(x^+)] \text{ if } x^- > 0, \\ &=]-\infty, \ln(x^+)] \text{ if } 0 \in [x^-,x^+]. \end{aligned}$$

Any constraint for which such a projection procedure is available will be called a *primitive constraint.*

Consider n variables $x_1,\ldots,x_n$ linked by m primitive constraints $\mathcal{C}_1,\ldots,\mathcal{C}_m$. For each variable x_i, it is assumed that a prior feasible domain $[x_i] = [x_i^-,x_i^+]$ is available (this domain may be equal to $]-\infty,\infty[$ if no prior information is available on x_i). The principle of ICP is to project all constraints as long as contraction is significant. The following recursive algorithm performs this task.

Algorithm ICP(inout: $[x_1],\ldots,[x_n]$)
1 $[a_1] := [x_1],\ldots,[a_n] := [x_n]$;
2 for $j = 1$ to m, project$(\mathcal{C}_j)$;
3 if $\max_i (a_i^+ - x_i^+ + x_i^- - a_i^-) \geq \varepsilon$
then ICP$([x_1],\ldots,[x_n])$.

Step 1 stores the initial domains for the x_is for a later evaluation of the contractions performed at Step 2. At Step 2, all constraints are projected in order to contract the current domains. At Step 3, the algorithm recursively calls itself if the contractions are still deemed significant, with ε a positive stopping parameter to be tuned by the user.

Theorem 1 *The algorithm* ICP *stops in less than* $\frac{mn}{\varepsilon} w([\mathbf{x}])$ *projections, where* $[\mathbf{x}]$ *is the box* $[x_1] \times \cdots \times [x_n]$ *and* $w([\mathbf{x}])$ *is the width of the box* $[\mathbf{x}]$*, i.e.,*

$$w([\mathbf{x}]) \triangleq \max_{i \in \{1,\ldots,n\}} (x_i^+ - x_i^-).$$

Proof: Define ℓ_k as the value of $(x_1^+ - x_1^-) + \cdots + (x_n^+ - x_n^-)$ at the kth iteration of ICP. While the algorithm iterates, $\ell_k \leq \ell_{k-1} - \varepsilon$. Thus, $\forall k$, $\ell_k \leq \ell_0 - k\varepsilon$. Now, $\ell_0 - k\varepsilon$ becomes negative when $\frac{\ell_0}{\varepsilon} < k$. Therefore, since ICP terminates for a positive ℓ_k, it cannot take more than ℓ_0/ε iterations. Since $\ell_0 \leq nw([\mathbf{x}])$, and since at each iteration m projections are performed, no more than $\frac{mn}{\varepsilon} w([\mathbf{x}])$ projections can take place. ■

Remark 2 *Efficiency could be improved, for instance, by choosing a clever order for the projections at Step 2 [6].*

3 Application to bounded-error estimation

Consider the circuit of Figure 1. Assume that measures have been collected leading to the following relations: $E \in [23V,26V], I \in [4A,8A], U_1 \in [10V,11V], U_2 \in [14V,17V]$, $P \in [124W,130W]$, where P is the power delivered by the voltage source. Nothing is known about the values of the resistors except that they are positive. Thus the prior domains for R_1 and R_2 are $]0,\infty[$. These quantities are related by the following constraints: $P = EI$, $E = (R_1+R_2)I$, $U_1 = R_1 I$, $U_2 = R_2 I$, $E = U_1 + U_2$. Some of these constraints are redundant, but this is not a problem for the method. To the contrary, redundancy may make constraint propagation more efficient. Since the second constraint $E = (R_1 + R_2)I$ is not primitive, it is decomposed into primitive constraints by introducing an auxiliary variable, say R, as follows: $E = RI$, $R = R_1 + R_2$. Constraint propagation can then be performed by executing several times, say q, the following projections: PMULT(P,E,I); PADD(R,R_1,R_2); PMULT(E,R,I); PMULT(U_1,R_1,I); PMULT(U_2,R_2,I); PADD(E,U_1,U_2); It can be shown (see page 92 of [7]) that if q tends to infinity the contracted domains that are finally obtained do not depend on the ordering of the projections.

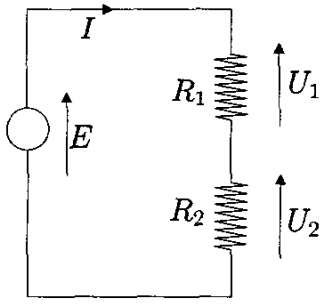

Figure 1: Circuit for bounded-error estimation

The stand-alone Scilab program of Table 1 computes intervals containing R_1 and R_2 by ICP. It is also available

at [8]. Scilab can be freely downloaded from [9].

Table 1: Bounded-error estimation with ICP

```
// interval arithmetic
function z=inter(x,y);
    z(1)=max(x(1),y(1));
    z(2)=min(x(2),y(2));
    endfunction
function z=add(x,y);
    z=x+y; endfunction
function z=dif(x,y);
    z=[x(1)-y(2),x(2)-y(1)];
    endfunction
function z=mult(x,y);
    v=[x(1)*y,x(2)*y];
    z=[min(v),max(v)];
    endfunction
function y=invert(x);
    y=[-%inf,%inf];
     if prod(x)>0, y=[1/x(2),1/x(1)];
    endfunction
function z=div(x,y);
    a=invert(y); z=mult(x,a);
    endfunction
// projection operators
function [c,a,b]=padd(z,x,y);
    c=inter(z,add(x,y));
    a=inter(x,dif(z,y));
    b=inter(y,dif(z,x)); endfunction
function [c;a,b]=pmult(z,x,y)
    c=inter(z,mult(x,y));
    a=inter(x,div(z,y));
    b=inter(y,div(z,x)); endfunction
// Main program
R=[-%inf,%inf]; R1=[0,%inf]; R2=[0,%inf];
E=[23,26]; I=[4,8]; P=[124,130];
U1=[10,11]; U2=[14,17];
for i=1:10, // simplistic stopping criterion
    [R,R1,R2]=padd(R,R1,R2); [P,E,I]=pmult(P,E,I);
    [E,R,I]=pmult(E,R,I); [U2,R2,I]=pmult(U2,R2,I);
    [U1,R1,I]=pmult(U1,R1,I);
    [E,U1,U2]=padd(E,U1,U2);
end;
```

In less that 0.1 second on a Pentium 300, this program generates the following results: $R_1 \in [1.846, 2.307]$, $R_2 \in [2.584, 3.355]$, $I \in [4.769, 5.417]$, $U_1 \in [10, 11]$, $U_2 \in [14, 16]$, $E \in [24, 26]$, $P \in [124, 130]$. ICP made it possible to get rather accurate intervals containing R_1 and R_2. The domains for I and U_2 have also been contracted, whereas the domains for U_1 and P have been left unchanged.

This example is of course very simple and was only meant to facilitate understanding of the concept of ICP. The same type of techniques can be employed for the estimation of the parameters and state of a nonlinear dynamical system [7, 10]. To employ interval techniques in the context of joint identification and robust control, it becomes necessary to distinguish the parameter vector $\mathbf{p}$ of the model of the process from the parameter vector $\mathbf{c}$ of the controller. This will be made possible by the technique presented in Section 4 and applied in Section 5 to robust stabilization.

4 A projection algorithm

Consider the set $\mathbb{S} \triangleq \{(\mathbf{c}, \mathbf{p}) \in [\mathbf{c}] \times [\mathbf{p}] \mid f(\mathbf{c}, \mathbf{p}) \leq 0\}$, where $[\mathbf{c}]$ is a box of $\mathbb{R}^{n_c}$, $[\mathbf{p}]$ is a box of $\mathbb{R}^{n_p}$ and f is a function mapping $\mathbb{R}^{n_c} \times \mathbb{R}^{n_p}$ into $\mathbb{R}$. The projection $\mathbb{S}_c$ of $\mathbb{S}$ onto $\mathbb{R}^{n_c}$ is defined by

$$\mathbb{S}_c \triangleq \{\mathbf{c} \in [\mathbf{c}] \mid \exists \mathbf{p} \in [\mathbf{p}], f(\mathbf{c}, \mathbf{p}) \leq 0\}.$$

This section is devoted to an algorithm computing inner and outer approximations of $\mathbb{S}_c$. To simplify presentation, we shall first introduce the notion of contractors for sets.

4.1 Contractors for sets

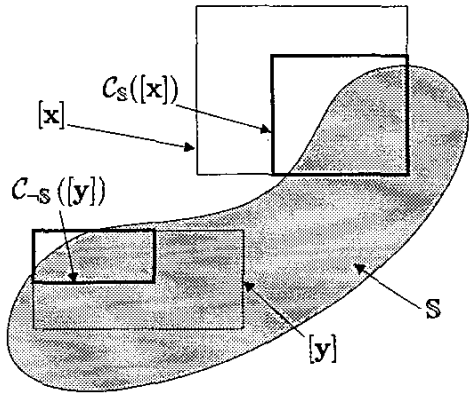

Figure 2: Action of contractors for $\mathbb{S}$ and $\neg\mathbb{S}$

The set of all boxes of $\mathbb{R}^n$ will be denoted by $\mathbb{IR}^n$. The operator $\mathcal{C}_{\mathbb{S}} : \mathbb{IR}^n \to \mathbb{IR}^n$ is a *contractor* for the set $\mathbb{S}$ of $\mathbb{R}^n$ if it satisfies

$$\forall [\mathbf{x}] \in \mathbb{IR}^n, \begin{cases} \mathcal{C}_{\mathbb{S}}([\mathbf{x}]) \subset [\mathbf{x}] & \text{(contractance)}, \\ \mathcal{C}_{\mathbb{S}}([\mathbf{x}]) \cap \mathbb{S} = [\mathbf{x}] \cap \mathbb{S} & \text{(correctness)}. \end{cases} \tag{2}$$

Figure 2 illustrates the action of a contractor for $\mathbb{S}$ operating on $[\mathbf{x}]$ and a contractor for its complementary set $\neg\mathbb{S}$ operating on $[\mathbf{y}]$. ICP can be used to build contractors. Consider, for instance, the set

$$\begin{aligned} \mathbb{S} &\triangleq \{\mathbf{x} \in \mathbb{R}^n | x_1 + x_1 x_2 \leq 1 \text{ and } \sin(x_1 + x_2) \leq 0\} \\ &= \{\mathbf{x} \in \mathbb{R}^n | \max(x_1 + x_1 x_2 - 1, \sin(x_1 + x_2)) \leq 0\}. \end{aligned}$$

The constraint defining $\mathbb{S}$ can be decomposed into primitive constraints, by introducing the auxiliary variables

$a = x_1x_2, b = x_1 + a, c = b - 1, d = x_1 + x_2, e = \sin(d)$ and $f = \max(c, e)$, with

$$a \in]-\infty,\infty[,\ b \in]-\infty,\infty[,\ c \in]-\infty,\infty[,$$
$$d \in]-\infty,\infty[,\ e \in]-\infty,\infty[\text{ and } f \in]-\infty, 0].$$

If ICP is applied to this set of constraints, the prior domains will be contracted. A possible contractor for $\mathbb{S}$ is thus the following procedure.

Algorithm $\mathcal{C}_{\mathbb{S}}$ (inout: $[x_1], [x_2]$)	
1	$[a] := [b] := [c] := [d] := [e] :=]-\infty,\infty[$;
	$[f] :=]-\infty, 0]$;
2	while contraction is significant,
3	PMULT($[a], [x_1], [x_2]$) ; PADD($[b], [x_1], [a]$) ;
4	PADD ($[b], [c], [1, 1]$) ; PADD ($[d], [x_1], [x_2]$) ;
5	PSIN ($[e], [d]$) ; PMAX ($[f], [c], [e]$) ;
6	end while.

This approach can be applied to a vast class of sets defined by nonlinear inequalities. If these inequalities are be linked by Boolean operators, *min* and *max* constraints can be used to obtain expressions that are free of such operators. For instance, the complementary set $\neg\mathbb{S}$ of $\mathbb{S}$ is given by

$$\begin{aligned}\neg\mathbb{S} &= \{\mathbf{x} \in \mathbb{R}^n \mid x_1 + x_1x_2 > 1 \text{ or } \sin(x_1 + x_2) > 0\} \\ &= \{\mathbf{x} \in \mathbb{R}^n \mid \max(x_1 + x_1x_2 - 1, \sin(x_1 + x_2)) > 0\}.\end{aligned}$$

A contractor for $\neg\mathbb{S}$ is thus given by the following procedure.

Algorithm $\mathcal{C}_{\neg\mathbb{S}}$ (inout: $[x_1], [x_2]$)	
1	$[a] := [b] := [c] := [d] := [e] :=]-\infty,\infty[$;
	$[f] :=]0,\infty[$;
2	while contraction is significant,
3	PMULT($[a], [x_1], [x_2]$) ; PADD($[b], [x_1], [a]$) ;
4	PADD ($[b], [c], [1, 1]$) ; PADD ($[d], [x_1], [x_2]$) ;
5	PSIN ($[e], [d]$) ; PMAX ($[f], [c], [e]$) ;
6	end while.

4.2 Projection algorithm

The algorithm PROJECT presented in Table 2 computes two unions of non-overlapping boxes $\mathcal{L}^-$ and $\Delta\mathcal{L}$ such that

$$\mathcal{L}^- \subset \mathbb{S}_c \subset \mathcal{L}^- \cup \Delta\mathcal{L}. \tag{3}$$

Unfamiliar readers may consult Chapter 5 of [7], which contains many such algorithms with an increasing level of difficulty.

In Table 2, to *bisect* [c] means to split it into two boxes along a symmetry plane orthogonal to a side of maximum length. The principle of PROJECT is to update three lists: (*i*) the stack $\mathcal{L}$ containing boxes of $[\mathbf{c}] \times [\mathbf{p}]$ still to be studied, (*ii*) the list $\mathcal{L}^-$ containing boxes of $\mathbb{R}^{n_c}$ that have been proved to be inside $\mathbb{S}_c$ and (*iii*) the list $\Delta\mathcal{L}$ containing boxes of $\mathbb{R}^{n_c}$ for which nothing is known but which are deemed too small to be bisected further (their width is smaller than ε). Let $\text{proj}_c(\mathcal{L})$ be the union of the projections onto c-space of the boxes of $\mathcal{L}$. The property

$$\mathbb{S}_c \subset \text{proj}_c(\mathcal{L}) \cup \mathcal{L}^- \cup \Delta\mathcal{L}.$$

is satisfied whenever Step 2 is reached. When PROJECT terminates $\mathcal{L}$ is empty and thus (3) is satisfied. OUTSIDE, called at Step 4 and given in Table 3, is a contractor for the set $\mathbb{S}$ to be projected. INSIDE, called at Step 6 and given in Table 4, returns a subbox $\overline{[\mathbf{c}]}$ of $[\mathbf{c}]$ such that $[\mathbf{c}] \cap \neg\mathbb{S}_c = \overline{[\mathbf{c}]} \cap \neg\mathbb{S}_c$. Contrary to OUTSIDE, INSIDE does not modify $[\mathbf{c}]$ and $[\mathbf{p}]$.

Table 2: Algorithm bracketing $\mathbb{S}_c$

Algorithm PROJECT(in: $[\mathbf{c}], [\mathbf{p}]$; out: $\mathcal{L}^-, \Delta\mathcal{L}$)	
1	$\mathcal{L} := \{([\mathbf{c}], [\mathbf{p}])\}$; $\mathcal{L}^- := \emptyset$; $\Delta\mathcal{L} := \emptyset$;
2	while $\mathcal{L} \neq \emptyset$,
3	pop a pair of boxes $([\mathbf{c}], [\mathbf{p}])$ out of $\mathcal{L}$;
4	OUTSIDE($[\mathbf{c}], [\mathbf{p}]$) ;
5	if $[\mathbf{c}] = \emptyset$, go to 2;
6	$\overline{[\mathbf{c}]} :=$ INSIDE($[\mathbf{c}], [\mathbf{p}]$) ;
7	if $\overline{[\mathbf{c}]} = \emptyset$ then $\mathcal{L}^- = \mathcal{L}^- \cup \{[\mathbf{c}]\}$; go to 2;
8	if $w([\mathbf{c}]) > \varepsilon$,
9	bisect $[\mathbf{c}]$ into $[\mathbf{c}_1]$ and $[\mathbf{c}_2]$;
10	$\mathcal{L} := \mathcal{L} \cup \{([\mathbf{c}_1], [\mathbf{p}]), ([\mathbf{c}_2], [\mathbf{p}])\}$;
11	else, $\Delta\mathcal{L} := \Delta\mathcal{L} \cup \{[\mathbf{c}]\}$;
12	end while.

In Table 3, $\mathcal{L}$ is a last-in-first-out list of non-overlapping boxes of $\mathbb{R}^{n_c} \times \mathbb{R}^{n_p}$ and, in Step 8, $\sqcup$ is the interval union operator, such that $[\mathbf{c}_0] \sqcup [\mathbf{c}]$ is the smallest box containing $[\mathbf{c}_0] \cup [\mathbf{c}]$. Check that whenever Step 2 is reached, $([\mathbf{c}_0^{\text{init}}] \times [\mathbf{p}_0^{\text{init}}]) \cap \mathbb{S} = (\mathcal{L} \cup ([\mathbf{c}_0] \times [\mathbf{p}_0])) \cap \mathbb{S}$, where $[\mathbf{c}_0^{\text{init}}]$ and $[\mathbf{p}_0^{\text{init}}]$ denote the initial boxes sent as input arguments of OUTSIDE. When OUTSIDE terminates, $\mathcal{L}$ is empty and thus the correctness property $([\mathbf{c}_0^{\text{init}}] \times [\mathbf{p}_0^{\text{init}}]) \cap \mathbb{S} = ([\mathbf{c}_0] \times [\mathbf{p}_0]) \cap \mathbb{S}$ is satisfied, as required by (2). Note that $[\mathbf{c}]$ is never bisected in OUTSIDE.

Table 3: Contractor for $\mathbb{S}$

Algorithm OUTSIDE(inout: $[\mathbf{c}_0], [\mathbf{p}_0]$)	
1	$\mathcal{L} := \{([\mathbf{c}_0], [\mathbf{p}_0])\}$; $\varepsilon_p = w([\mathbf{c}_0])$;
	$[\mathbf{c}_0] := \emptyset$; $[\mathbf{p}_0] := \emptyset$;
2	while $\mathcal{L} \neq \emptyset$,
3	take a pair of boxes $([\mathbf{c}], [\mathbf{p}])$ in $\mathcal{L}$;
4	$\mathcal{C}_{\mathbb{S}}([\mathbf{c}], [\mathbf{p}])$;
5	if $w([\mathbf{p}]) > \varepsilon_p$,
6	bisect $[\mathbf{p}]$ into $[\mathbf{p}_1]$ and $[\mathbf{p}_2]$;
7	$\mathcal{L} := \mathcal{L} \cup \{([\mathbf{c}], [\mathbf{p}_1]), ([\mathbf{c}], [\mathbf{p}_2])\}$;
8	else $[\mathbf{c}_0] := [\mathbf{c}_0] \sqcup [\mathbf{c}]$; $[\mathbf{p}_0] := [\mathbf{p}_0] \sqcup [\mathbf{p}]$;
9	end while.

INSIDE returns a subbox $\overline{[\mathbf{c}]}$ of $[\mathbf{c}]$ such that $[\mathbf{c}] \cap \neg\mathbb{S}_c = \overline{[\mathbf{c}]} \cap \neg\mathbb{S}_c$, *i.e.*, it implements a contractor for the complementary set $\neg\mathbb{S}_c$ of $\mathbb{S}_c$. The values of $\overline{[\mathbf{c}]}$ that have been

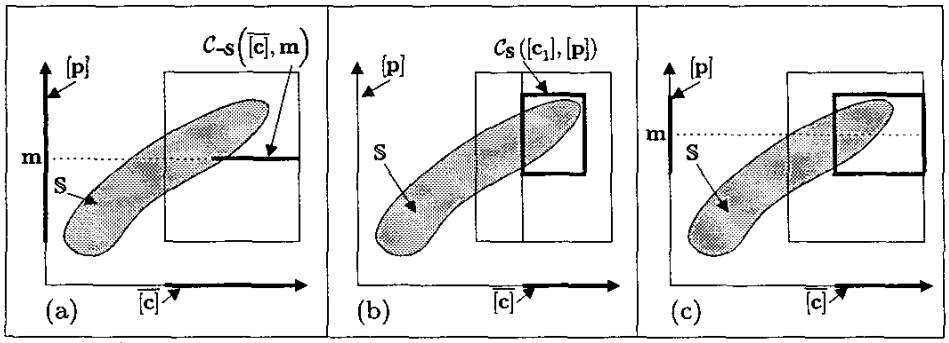

Figure 3: Principle of INSIDE; (a) $[\overline{\mathbf{c}}]$ is contracted without losing a single point of the intersection of $\neg\mathbb{S}_\mathbf{c}$ with the current box; (b) $[\mathbf{p}]$ is now contracted, but $[\overline{\mathbf{c}}]$ remains unchanged; (c) a new iteration can be started

removed inside the *repeat* loop of Step 4 all belong to $\mathbb{S}_\mathbf{c}$. This is illustrated by Figure 3. In this algorithm, only $[\mathbf{p}]$ is bisected. All intermediate boxes are stored inside $\mathcal{L}$, which is here a first-in-first-out list.

Table 4: Contractor for $\neg\mathbb{S}_\mathbf{c}$

	Algorithm INSIDE(in: $[\mathbf{c}_0],[\mathbf{p}_0]$; out $[\overline{\mathbf{c}}]$)
1	$\mathcal{L} := \{[\mathbf{p}_0]\}$; $[\mathbf{c}] := [\mathbf{c}_0]$;
2	while $\mathcal{L} \neq \emptyset$,
3	take the first box $[\mathbf{p}]$ in $\mathcal{L}$;
4	repeat,
5	$\mathbf{m} :=$ center $([\mathbf{p}])$; $\mathcal{C}_{\neg\mathbb{S}}([\overline{\mathbf{c}}],\mathbf{m})$;
6	$[\mathbf{c}_1] := [\overline{\mathbf{c}}]; \mathcal{C}_{\mathbb{S}}([\mathbf{c}_1],[\mathbf{p}])$;
7	while contraction of $[\overline{\mathbf{c}}]$ is significant;
8	if $[\overline{\mathbf{c}}] = \emptyset$, return;
9	if $w([\mathbf{p}]) > w([\mathbf{c}_0])$,
10	bisect $[\mathbf{p}]$ into $[\mathbf{p}_1]$ and $[\mathbf{p}_2]$;
11	store $[\mathbf{p}_1]$ and $[\mathbf{p}_2]$ at the end of $\mathcal{L}$;
12	end while.

5 Application to robust control

Although relatively few papers have been dedicated to controller design via interval analysis see, *e.g.*, [11]-[16], interest is growing, as illustrated by a recent special issue of *Reliable Computing* [17].

The projection algorithm presented in Section 4 will now be used to characterize the set of all parameter vectors $\mathbf{c}$ of a linear controller $\Gamma(\mathbf{c})$ that robustly stabilize a linear time-invariant model depending on an uncertain parameter vector $\mathbf{p}$. The vector $\mathbf{c}$ can be chosen arbitrarily in some given box $[\mathbf{c}] \subset \mathbb{R}^{n_\mathbf{c}}$. The vector $\mathbf{p}$ is only assumed to belong to some known box $[\mathbf{p}] \subset \mathbb{R}^{n_\mathbf{p}}$. The box $[\mathbf{p}]$ may have been obtained by a bounded-error estimation technique similar to that described in Section 3. Denote the closed-loop model by $\Sigma(\mathbf{c},\mathbf{p})$. It is easy to find, for instance via the Routh criterion, a function $r\colon \mathbb{R}^{n_\mathbf{c}} \times \mathbb{R}^{n_\mathbf{p}} \to \mathbb{R}$ such that

$$\Sigma(\mathbf{p},\mathbf{c}) \text{ is stable } \Leftrightarrow r(\mathbf{c},\mathbf{p}) > 0.$$

For the Routh criterion, $r(\mathbf{c},\mathbf{p})$ is the smallest entry in the first column of the Routh table associated with the characteristic polynomial of $\Sigma(\mathbf{p},\mathbf{c})$. Finding all robust controllers $\Gamma(\mathbf{c}), \mathbf{c} \in [\mathbf{c}]$ then amounts to characterizing the set

$$\mathbb{T}_\mathbf{c} = \{\mathbf{c} \in [\mathbf{c}] \mid \forall \mathbf{p} \in [\mathbf{p}], r(\mathbf{c},\mathbf{p}) > 0\} \quad (4)$$

$$= \{\mathbf{c} \in [\mathbf{c}] \mid \neg(\exists \mathbf{p} \in [\mathbf{p}], r(\mathbf{c},\mathbf{p}) \leq 0)\}, \quad (5)$$

where $\neg$ is the negation operator.

The algorithm PROJECT of Section 4 makes it possible to bracket the set

$$\mathbb{S}_\mathbf{c} = \{\mathbf{c} \in [\mathbf{c}] \mid \exists \mathbf{p} \in [\mathbf{p}], r(\mathbf{c},\mathbf{p}) \leq 0\} \quad (6)$$

between the unions of boxes $\mathcal{S}_\mathbf{c}^-$ and $\mathcal{S}_\mathbf{c}^+$ as

$$\mathcal{S}_\mathbf{c}^- \subset \mathbb{S}_\mathbf{c} \subset \mathcal{S}_\mathbf{c}^+.$$

By complementing this equation, we get

$$\neg\mathcal{S}_\mathbf{c}^+ \subset \neg\mathbb{S}_\mathbf{c} \subset \neg\mathcal{S}_\mathbf{c}^-.$$

Therefore

$$[\mathbf{c}]\cap\neg\mathcal{S}_\mathbf{c}^+ \subset [\mathbf{c}]\cap\neg\mathbb{S}_\mathbf{c} \subset [\mathbf{c}]\cap\neg\mathcal{S}_\mathbf{c}^-.$$

Since $\mathbb{T}_\mathbf{c}$ and $\mathbb{S}_\mathbf{c}$ form a partition of $[\mathbf{c}]$, we have $[\mathbf{c}]\cap\neg\mathbb{S}_\mathbf{c} = \mathbb{T}_\mathbf{c}$ and thus

$$[\mathbf{c}]\cap\neg\mathcal{S}_\mathbf{c}^+ \subset \mathbb{T}_\mathbf{c} \subset [\mathbf{c}]\cap\neg\mathcal{S}_\mathbf{c}^-.$$

The projection algorithm of Section 4 can be used to compute $[\mathbf{c}]\cap\neg\mathcal{S}_\mathbf{c}^-$ and $[\mathbf{c}]\cap\neg\mathcal{S}_\mathbf{c}^+$ thereby achieving the bracketing of $\mathbb{T}_\mathbf{c}$.

As an illustration, assume that the transfer functions of the controller and uncertain model of the process are

$$R = \frac{c_2 s + c_1}{s} \text{ and } G = \frac{p_1 p_3^2}{(p_2 s + 1)(s^2 + p_3 s + p_3^2)}$$

The closed-loop model $\Sigma(\mathbf{p}, \mathbf{c})$ is represented in Figure 4. The parameter vector $\mathbf{c}$ of the controller can be chosen arbitrarily in the box $[\mathbf{c}] = [0, 1]^2$. The parameter vector $\mathbf{p}$ of the model of the process is only known to belong to the box $[0.9, 1.1]^3$.

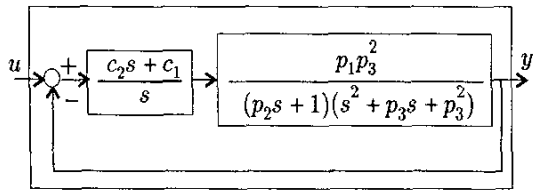

Figure 4: PI controller stabilizing an uncertain model

The first column of the Routh table associated with $\Sigma(\mathbf{p}, \mathbf{c})$ is

$$\begin{pmatrix} p_2 \\ p_2 p_3 + 1 \\ p_2 p_3^2 + p_3 - \frac{p_2(p_3^2 + c_2 p_1 p_3^2)}{p_2 p_3 + 1} \\ p_3^2 + c_2 p_1 p_3^2 - \frac{c_1 p_1 p_3^2 (p_2 p_3 + 1)^2}{(p_2 p_3^2 + p_3)(p_2 p_3 + 1) - p_2(p_3^2 + c_2 p_1 p_3^2)} \\ c_1 p_1 p_3^2 \end{pmatrix}.$$

Since $p_2 > 0$, the closed-loop model $\Sigma(\mathbf{p}, \mathbf{c})$ is asymptotically stable if and only if all the entries of this column are positive. As $r(\mathbf{c}, \mathbf{p})$ denotes the smallest entry of this column, an equivalent condition is $r(\mathbf{c}, \mathbf{p}) > 0$. Inner and outer approximations of $\mathbb{T}_c$ described by Figure 5 have been computed by PROJECT in 470 seconds for $\varepsilon = 0.02$. $\mathcal{T}_c^-$ in white is an inner approximation of the set of all robustly stable controllers while $\Delta\mathcal{T}_c$ in light grey is an uncertainty layer in which no conclusion is reached. The union of $\mathcal{T}_c^-$ and $\Delta\mathcal{T}_c$ is an outer approximation of $\mathbb{T}_c$.

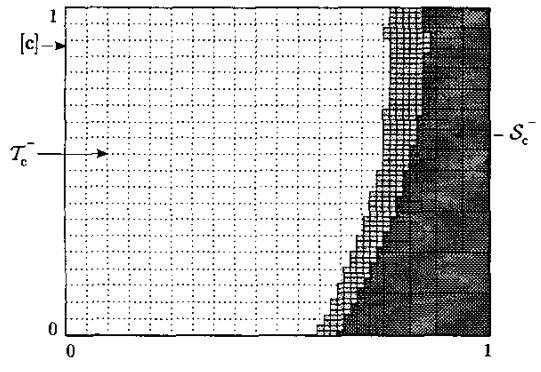

Figure 5: Inner approximation of the set of all controllers that ensure robust stability (in white), and uncertainty layer (in light grey)

References

[1] J. C. Cleary, "Logical arithmetic", *Future Computing Systems*, vol. 2(2) (1987), pp. 125–149.

[2] E. Davis, "Constraint propagation with interval labels", *Artificial Intelligence*, vol. 32(3) (1987), pp. 281–331.

[3] R. E. Moore, *Methods and Applications of Interval Analysis*. SIAM, Philadelphia PA, 1979.

[4] A. K. Mackworth, "Consistency in networks of relations", *Artificial Intel.*, vol. 8(1) (1977), pp. 99–118.

[5] F. Benhamou and L. Granvilliers, "Automatic generation of numerical redundancies for nonlinear constraint solving", *Reliable Computing*, vol. 3(3) (1997), pp. 335–344.

[6] F. Benhamou, F. Goualard, F. Granvilliers, and J. F. Puget, "Revising hull and box consistency", *Proceedings of the International Conference on Logic Programming*, Las Cruces NM, 1999, pp. 230–244.

[7] L. Jaulin, M. Kieffer, O. Didrit, and E. Walter, *Applied Interval Analysis*, Springer, London, 2001.

[8] http://www.istia.univ-angers.fr/~jaulin/propagR1R2.sce

[9] http://www-rocq.inria.fr/scilab/scilab.html

[10] L. Jaulin, M. Kieffer, I. Braems, and E. Walter, "Guaranteed nonlinear estimation using constraint propagation on sets", *International Journal of Control*, vol. 74(18) (2001), pp. 1772–1782.

[11] R. B. Kearfott, "Interval mathematics techniques for control theory computations", in *Progress in Systems and Control Theory*, Birkhäuser, Boston MA, vol. 20 (1989), pp. 169–178.

[12] L. V. Kolev, V. M. Mladenov, and S. S. Vladov, "Interval mathematics algorithms for tolerance analysis", *IEEE Transactions on Circuits and Systems*, vol. 35(8) (1988), pp. 967-974.

[13] N. A. Khlebalin, "Interval automatic systems - theory, computer-aided design and applications", *Interval Computations*, vol. 3 (1992), pp. 106–115.

[14] L.V. Kolev, "An interval first-order method for robustness analysis", *Proceedings of the IEEE International Symposium on Circuits and Systems*, Chicago IL, 1993, pp. 2522–2524.

[15] L. Jaulin and E. Walter, "Guaranteed tuning, with application to robust control and motion planning", *Automatica*, vol. 32(8) (1996), pp. 1217–1221.

[16] S. A. Malan, M. Milanese, and M. Taragna. "Robust analysis and design of control systems using interval arithmetics", *Automatica*, vol. 33(7) (1997), pp. 1363–1372.

[17] J. Garloff and E. Walter, editors, *Special Issue of Reliable Computing* devoted to *Applications to Control, Signals and Systems*, vol. 6(3) (2000), pp 229-362.