INTERNATIONAL JOURNAL OF ADAPTIVE CONTROL AND SIGNAL PROCESSING
Int. J. Adapt. Control Signal Process. 2002; **16**:193–218 (DOI: 10.1002/acs.680)

Guaranteed recursive non-linear state bounding using interval analysis

Michel Kieffer[1,*,†], Luc Jaulin[1,‡] and Éric Walter[1]

[1]*Laboratoire des Signaux et Systèmes, CNRS-Supélec-Université Paris-Sud, Plateau de Moulon, 91192 Gif-sur-Yvette, France*

SUMMARY

The problem considered here is state estimation in the presence of unknown but bounded state perturbations and measurement noise. In this context, most available results are for linear models, and the purpose of the present paper is to deal with the non-linear case. Based on interval analysis and the notion of set inversion, a new state estimator is presented, which evaluates a set estimate *guaranteed* to contain all values of the state that are consistent with the available observations, given the perturbation and noise bounds and a set containing the initial value of the state. To the best of our knowledge, it is the first estimator for which this claim can be made. The precision of the set estimate can be improved, at the cost of more computation. Theoretical properties of the estimator are studied, and computer implementation receives special attention. A simple illustrative example is treated. Copyright © 2002 John Wiley & Sons, Ltd.

KEY WORDS: bounded errors; interval analysis; non-linear estimation; set estimation; state estimation

1. INTRODUCTION

This paper is devoted to recursive non-linear state estimation for discrete-time systems in the presence of bounded state perturbation and measurement noise. The problem here is to characterize the set of all the state vectors of a given model that are compatible with the information available. Unknown and possibly time-varying parameters can also be considered.

In the context of bounded errors, the model is usually assumed to be linear, and ellipsoids can then be computed at each step, guaranteed to contain the present state, see, e.g. References [1–7]. The computation of these ellipsoids closely parallels Kalman filtering, alternating prediction and correction phases. For non-linear models, the problem is much more difficult, since the set is usually non-convex and may even be non-connected, and relatively few results are available.

* Correspondence to: Michel Kieffer, Laboratoire des Signaux et Systèmes, CNRS-Supélec-Université Paris-Sud, Plateau de Moulon, 91192 Gif-sur-Yvette, France

† E-mail: kieffer@lss.supelec.fr

‡ On leave from Laboratoire d'Ingénierie des Systèmes Automatisés, Université d'Angers

Contract/grant sponsor: INTAS; Contract/grant number: RFBR-97-10782

Received 21 July 2000
Revised 11 December 2000
Accepted 24 March 2001

Published online 25 January 2002

Assuming a linear observation equation, and linearizing the state equation, Shamma and Kuang-Yang Tu [8] enclose the set of all possible state values in a polytope. The linearization error may be included in the state perturbation. Techniques bounding the state of systems with poorly known state equations and inputs are presented in References [9, 10], with applications in waste processing. These results require a special structure of the state equation and noise-free outputs. Lichtenberg and Lunze [11] propose a description of the state, input and output of the model using qualitative values. A qualitative observer is then obtained, providing a state estimate that may consist of several numerical vectors. The main limitations of this technique are that the transformation of a non-linear model into a qualitative model does not allow the estimates to be guaranteed and that the complexity of the description grows very fast with the number of qualitative values, even when the number of state variables is small.

Our purpose here is to characterize the set of all the state vectors that are consistent with the information available, and to do so recursively, thus facilitating real-time implementation. This characterization will be performed with the help of interval analysis, by computing outer approximations by unions of boxes, or *subpavings*, which can, in principle, be made as accurate as desired as we shall see. Interval techniques have already been applied to building interval Kalman filters for uncertain linear models [12], but no guarantee was given that the state would belong to the set estimate computed. In contrast, the technique to be presented here guarantees its results. To the best of our knowledge, it is the first one that can make this claim in the context of non-linear state bounding when the state is perturbed and the measurements are noisy. Preliminary results concerning this estimator were presented in Reference [13].

In Section 2, an idealized state estimator is presented and an illustrative example is introduced. The basic notions of interval arithmetic are recalled in Section 3. They are then used to build an approximate but guaranteed non-linear state estimator using the notion of subpaving presented in Section 4. The correction and prediction steps of this estimator are, respectively, presented in Sections 5 and 6. Section 7 summarizes the algorithm and establishes theoretical properties of the corresponding estimator, which is then applied to the illustrative example of Section 2. Conclusions are presented in Section 8, followed by three appendices.

2. IDEALIZED STATE ESTIMATOR

This state estimator is based on interval computation. A *scalar interval* $[x]$ is a closed, connected and bounded subset of $\mathbb{R}$. It may be characterized by its lower bound $\underline{x}$ and upper bound $\bar{x}$. Thus, $[x] = \{x \in \mathbb{R} \mid \underline{x} \leqslant x \leqslant \bar{x}\}$. An n-dimensional *box* (or *vector interval*) $[\mathbf{x}]$ is the Cartesian product of n scalar intervals $[x_i]$.

Consider non-linear and possibly time-varying system defined by

$$\begin{aligned} \mathbf{x}_{\ell+1} &= \mathbf{f}_\ell(\mathbf{x}_\ell, \mathbf{w}_\ell), \\ \mathbf{y}_\ell &= \mathbf{h}_\ell(\mathbf{x}_\ell) + \mathbf{v}_\ell, \end{aligned} \qquad \ell = 0, 1, \ldots \tag{1}$$

where $\mathbf{x}_\ell \in \mathbb{R}^n$ and $\mathbf{y}_\ell \in \mathbb{R}^p$ are, respectively, the state and output vectors. The initial state $\mathbf{x}_0$ is assumed to belong to some prior compact set $\mathscr{X}_0 \subset \mathbb{R}^n$. $\{\mathbf{w}_\ell\}$ and $\{\mathbf{v}_\ell\}$ are unknown state perturbation and measurement noise sequences, respectively, assumed to belong to the known box sequences $\{[\mathbf{w}]_\ell\}$ and $\{[\mathbf{v}]_\ell\}$. $\mathbf{f}_\ell$ and $\mathbf{h}_\ell$ are known functions (possibly defined by finite algorithms). The problem to be considered is the recursive estimation of the smallest set guaranteed to contain all values of $\mathbf{x}_\ell$ compatible with the information available just after the ℓth measurement, i.e. with

$$\mathscr{I}_\ell = \{\mathscr{X}_0, \{[\mathbf{w}]_i\}_{i=0}^{\ell-1}, \{\mathbf{y}_i, [\mathbf{v}]_i\}_{i=0}^{\ell}\} \tag{2}$$

Remark 1

The function $\mathbf{f}_\ell$ may, of course, depend on a vector of deterministic inputs $\mathbf{u}_\ell \in \mathbb{R}^m$, omitted in (1) for the sake of simplicity. In such a case, $\mathscr{I}_\ell$ should also include $\mathbf{u}_i$, $i = 0, \dots, \ell - 1$.

Remark 2

To simplify the presentation, the bounded measurement noise is supposed to be additive in (1). Other observation equations such as $\mathbf{y}_\ell = \mathbf{h}_\ell(\mathbf{x}_\ell, \mathbf{v}_\ell)$ or $\mathbf{g}_\ell(\mathbf{y}_\ell, \mathbf{v}_\ell) = \mathbf{h}_\ell(\mathbf{x}_\ell)$ can also be considered, provided that the characterization of the set of all the state vectors at time ℓ that are consistent with the output vector $\mathbf{y}_\ell$ and the bounds on $\mathbf{v}_\ell$ can be formulated as a set-inversion problem, see Section 5.

Remark 3

The more general problem of joint state and parameter estimation can easily be treated. It suffices to replace $\mathbf{x}_\ell$ by an *extended state* $\mathbf{x}_\ell^{\mathrm{e}} = (\mathbf{x}_\ell^{\mathrm{T}}, \mathbf{p}_\ell^{\mathrm{T}})^{\mathrm{T}}$ incorporating the unknown parameter vector $\mathbf{p}_\ell \in \mathbb{R}^q$, assumed to belong to some prior compact set $\mathscr{P}_0 \subset \mathbb{R}^q$. An evolution equation for the parameters is then needed; it may, for instance, be $\mathbf{p}_{\ell+1} = \mathbf{p}_\ell + \mathbf{w}_\ell^p$, with $\{\mathbf{w}_\ell^p\}$ included in some known interval sequence $\{[\mathbf{w}^p]_\ell\}$. Thus, an extended state equation can be written as

$$\begin{aligned} \mathbf{x}_{\ell+1}^{\mathrm{e}} &= \begin{pmatrix} \mathbf{x}_{\ell+1} \\ \mathbf{p}_{\ell+1} \end{pmatrix} = \begin{pmatrix} \mathbf{f}_\ell(\mathbf{x}_\ell^{\mathrm{e}}, \mathbf{w}_\ell, \mathbf{w}_\ell^p) \\ \mathbf{p}_\ell + \mathbf{w}_\ell^p \end{pmatrix} = \mathbf{f}_\ell^{\mathrm{e}}(\mathbf{x}_\ell^{\mathrm{e}}, \mathbf{w}_\ell^{\mathrm{e}}), \\ & \qquad\qquad\qquad\qquad\qquad\qquad \ell = 0, 1, \dots \\ \mathbf{y}_\ell &= \mathbf{h}_\ell^{\mathrm{e}}(\mathbf{x}_\ell^{\mathrm{e}}) + \mathbf{v}_\ell, \end{aligned} \tag{3}$$

where $\mathbf{w}_\ell^{\mathrm{e}} = (\mathbf{w}_\ell^{\mathrm{T}}, (\mathbf{w}_\ell^p)^{\mathrm{T}})^{\mathrm{T}} \in [\mathbf{w}^{\mathrm{e}}]_\ell = ([\mathbf{w}]_\ell^{\mathrm{T}}, [\mathbf{w}^p]_\ell^{\mathrm{T}})^{\mathrm{T}}$ and $\mathbf{x}_0^{\mathrm{e}} \in \mathscr{X}_0^{\mathrm{e}} = \mathscr{X}_0 \times \mathscr{P}_0 \subset \mathbb{R}^n \times \mathbb{R}^q$. When specific knowledge about some deterministic dependency of $\mathbf{p}_{\ell+1}$ in $\mathbf{p}_\ell$ and $\mathbf{x}_\ell$ is available, this knowledge can readily be incorporated in (3).

2.1. Idealized algorithm

Assume that $\mathscr{X}_\ell$ is some set guaranteed to contain $\mathbf{x}_\ell$. Define the *predicted* set $\mathscr{X}_{\ell+}$ as

$$\mathscr{X}_{\ell+} = \mathbf{f}_\ell(\mathscr{X}_\ell, [\mathbf{w}]_\ell) = \{\mathbf{f}_\ell(\mathbf{x}, \mathbf{w}) | \mathbf{x} \in \mathscr{X}_\ell, \mathbf{w} \in [\mathbf{w}]_\ell\} \tag{4}$$

This predicted set is guaranteed to contain $\mathbf{x}_{\ell+1}$. Moreover, let $\mathscr{Y}_{\ell+1}$ be the set of all possible values of the noise-free output, when the value of the measured output is $\mathbf{y}_{\ell+1}$

$$\mathscr{Y}_{\ell+1} = \mathbf{y}_{\ell+1} - [\mathbf{v}]_{\ell+1} = \{\mathbf{y}_{\ell+1} - \mathbf{v} | \mathbf{v} \in [\mathbf{v}]_{\ell+1}\} \tag{5}$$

and let $\mathscr{X}^{\mathrm{o}}_{\ell+1}$ be the set of all values of the state at time $\ell + 1$ that could have led to an observation $\mathbf{y}$ in $\mathscr{Y}_{\ell+1}$

$$\mathscr{X}^{\mathrm{o}}_{\ell+1} = \mathbf{h}^{-1}_{\ell+1}(\mathscr{Y}_{\ell+1}) = \{\mathbf{x} \in \mathbb{R}^n | \mathbf{h}_{\ell+1}(\mathbf{x}) \in \mathscr{Y}_{\ell+1}\} \tag{6}$$

Then, the *corrected* set

$$\mathscr{X}_{\ell+1} = \mathscr{X}_{\ell+} \cap \mathscr{X}^{\mathrm{o}}_{\ell+1} \tag{7}$$

is also guaranteed to contain $\mathbf{x}_{\ell+1}$.

Procedure (4)–(7) can be applied recursively starting from $\mathscr{X}_0$, assumed to be available *a priori*. This is summarized in the following idealized algorithm.

Algorithm 1

```
For $\ell = 0$ to $\bar{\ell}$, do
1. Prediction: $\mathscr{X}_{\ell+} = \mathbf{f}_\ell(\mathscr{X}_\ell, [\mathbf{w}]_\ell)$.
2. Correction: $\mathscr{X}_{\ell+1} = \mathbf{h}^{-1}_{\ell+1}(\mathscr{Y}_{\ell+1}) \cap \mathscr{X}_{\ell+}$.    □
```

It is easy to show that $\mathscr{X}_\ell$ as evaluated by Algorithm 1 is the smallest set guaranteed to contain $\mathbf{x}_\ell$ that can be computed from $\mathscr{I}_\ell$ and (1).

Except in very particular cases, it is not possible to evaluate the sets $\mathscr{X}_{\ell+}$ and $\mathscr{X}_{\ell+1}$ exactly. Our purpose will therefore be to get guaranteed outer approximations of these sets, as accurate as possible. Before presenting the basic blocks of an algorithm for this task, a simple illustrative example will be introduced, which will be treated in Section 7.

2.2. The bouncing-ball example

Estimating the state of a bouncing ball is not as simple as it may seem, because of the discontinuity in the state equation introduced by the bounce. This problem is representative of a number of difficulties encountered, e.g. when describing the motion of walking robots or various hybrid systems [14]. The solution presented, in what follows, could easily be transposed to more complex systems.

Consider an idealized ball with radius ρ bouncing on the floor. The motion of this ball is assumed to be one dimensional, and its state is characterized by its height x_1 and speed $x_2 = \dot{x}_1$. Our purpose is to estimate $\mathbf{x} = (x_1, x_2)^{\mathrm{T}}$ when the ball is dropped with an initial state $\mathbf{x}_0$ only known to belong to $\mathscr{X}_0$. The discrete-time measurements of its height, are assumed to satisfy

$$y_\ell = h_\ell(\mathbf{x}_\ell) + v_\ell = (1 \;\; 0)\mathbf{x}_\ell + v_\ell \tag{8}$$

with v_ℓ belonging to some known interval. Neglecting friction with the atmosphere, the motion of the ball is described by

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= -g \end{aligned} \tag{9}$$

during the fall, and

$$\begin{aligned} x_1 &\to x_1 \\ x_2 &\to -\sqrt{(1-\gamma)}x_2 \end{aligned} \tag{10}$$

when the ball hits the floor. The portion γ of the energy lost during the bounce is unknown but assumed to belong to some known interval. So γ can be viewed as a bounded perturbation entering non-linearly in the state equation. By exact discretization, a finite algorithm $\mathbf{f}$ evaluating $\mathbf{x}_{\ell+1}$ for a given numerical value of $\mathbf{x}_\ell$ and γ is easily constructed, see Table A1 in Appendix A.

3. INTERVAL ARITHMETIC

The prediction and correction steps of Algorithm 1 can be implemented in an approximate but guaranteed way using interval computation and subpavings, which provide powerful tools for the description of sets. The aim of this section is to recall the few basic notions needed. For a more detailed presentation of interval arithmetic, see, e.g. Reference [15] or [16].

An important characteristic of an interval $[x]$ is its *width* $w([x]) = \bar{x} - \underline{x}$. The width of an n-dimensional box $[\mathbf{x}]$ is $w([\mathbf{x}]) = \max_{i=1,\dots,n} w([x_i])$.

Interval arithmetic extends the usual arithmetical operations on real numbers to intervals through the generic formula

$$\forall \circ \in \{+, -, \times, /\}, [x] \circ [y] = \{x \circ y | x \in [x] \quad \text{and} \quad y \in [y]\} \tag{11}$$

All these operations are *inclusion monotonic* [16]:

$$\text{if } [x'] \subset [x] \text{ and } [y'] \subset [y], \text{ then } [x'] \circ [y'] \subset [x] \circ [y] \tag{12}$$

This means that uncertainty on their results cannot deteriorate if uncertainty on their arguments is decreased, which suggests algorithms where intervals may be split into subintervals to increase precision. An extension to vector and matrix intervals, necessary for state estimation, can be found in References [15, 16].

Let f be a real function of a real variable, defined on $\mathscr{D} \subset \mathbb{R}$. The definition of this function is extended to intervals $[x] \subset \mathscr{D}$ as follows:

$$f([x]) = \{f(x) | x \in [x]\} \tag{13}$$

In general, it is not possible exactly to characterize the image set $f([x])$, which is not even necessarily an interval. The fundamental notion of an inclusion function makes it possible to compute *an interval* guaranteed to contain this image set. An *inclusion function* associated with f will be denoted by $f_{[]}$; for any $[x] \subset \mathscr{D}$, it should satisfy

$$f([x]) \subseteq f_{[]}([x]) \tag{14}$$

In addition to being inclusion monotonic, it is desirable that $f_{[]}([x])$ converge to $f([x])$ when $w([x])$ tends to zero. When $f_{[]}([x])$ possesses this property, it is said to be *convergent*. Among the infinitely many inclusion functions associated with any given real function f, it is of interest to obtain one that is as accurate as possible.

Various methods for building a convergent inclusion function from the real function f are available. The simplest one is to replace all the occurrences of x in the formal expression of f by $[x]$ and all operators and elementary functions by their interval counterparts. One thus gets a *natural* inclusion function. Usually, however, this inclusion function is far from accurate, because of the pessimism introduced by interval computations as soon as there are several occurrences of x. The accuracy can often be improved by rewriting f so as to decrease the number of occurrences of x in its formal expression. Various other techniques of obtaining efficient inclusion functions are available, especially when the width of $[x]$ is small enough [16].

4. SUBPAVINGS

4.1. Definition and properties

An n-dimensional *subpaving* $\hat{\mathscr{X}}$ is a union of non-overlapping boxes of $\mathbb{R}^n$. The set of all the subpaving of $\mathbb{R}^n$ is denoted by $\mathscr{K}(\mathbb{R}^n)$. To characterize the precision with which a subpaving may approximate a compact subset of $\mathbb{R}^n$, it is necessary to introduce a distance between such sets. Let $\mathscr{A}$ and $\mathscr{B}$ be compact subsets of $\mathbb{R}^n$. The *Hausdorff distance* based on the infinity norm is $d(\mathscr{A},\mathscr{B}) = \max\{d_0(\mathscr{A},\mathscr{B}), d_0(\mathscr{B},\mathscr{A})\}$, where $d_0(\mathscr{A},\mathscr{B}) = \max_{\mathbf{a}\in\mathscr{A}} d_0(\mathbf{a},\mathscr{B})$, with $d_0(\mathbf{a},\mathscr{B}) = \min_{\mathbf{b}\in\mathscr{B}} d_\infty(\mathbf{a},\mathbf{b})$ and $d_\infty(\mathbf{a},\mathbf{b}) = \max_{i=1,\dots,n}\{|a_i - b_i|\}$. *Hausdorff distances* such as $d(.,.)$ are metrics for the set of compact subsets of $\mathbb{R}^n$ [17].

Let $\mathscr{A}$, $\mathscr{B}$, and $\mathscr{C}$ be three compact subsets of $\mathbb{R}^n$ and let $[\mathbf{x}]$ be a box of $\mathbb{R}^n$. The following properties are straightforward to prove.

Property 1. If $\mathbf{x}\in[\mathbf{x}]$, then $w([\mathbf{x}])/2 \leqslant d(\mathbf{x},[\mathbf{x}]) \leqslant w([\mathbf{x}])$.
Property 2. If $\mathscr{B}\subset[\mathbf{x}]$ and $w([\mathbf{x}])\leqslant\varepsilon$, then $d(\mathscr{B},[\mathbf{x}])\leqslant\varepsilon$.
Property 3. If $w([\mathbf{x}])\leqslant\varepsilon$ and $d_0([\mathbf{x}],\mathscr{A})>\varepsilon$, then $[\mathbf{x}]\cap\mathscr{A}=\emptyset$.
Property 4. If $\mathscr{B}\subset\mathscr{C}$, then $d_0(\mathscr{B},\mathscr{C})=0$ and $d_0(\mathscr{A},\mathscr{C})\leqslant d_0(\mathscr{A},\mathscr{B})$.
Property 5. If $\mathscr{A}\cap\mathscr{B}\neq\emptyset$ and $\mathscr{A}\cap\mathscr{C}\neq\emptyset$, then $d(\mathscr{A}\cap\mathscr{B},\mathscr{A}\cap\mathscr{C})\leqslant d(\mathscr{B},\mathscr{C})$.
Property 6. If $\mathscr{C}\neq\emptyset$, then $d(\mathscr{A}\times\mathscr{C},\mathscr{B}\times\mathscr{C}) = d(\mathscr{A},\mathscr{B})$.

The following result shows that outer approximations of compact subsets of $\mathbb{R}^n$ can be obtained at any desired precision using subpavings.

Proposition 1

For any compact subset $\mathscr{A}$ of $\mathbb{R}^n$ and positive real $\varepsilon>0$, there exists $\hat{\mathscr{X}}_\varepsilon\in\mathscr{K}(\mathbb{R}^n)$ such that $\mathscr{A}\subset\widehat{\mathscr{X}_\varepsilon}$ and $d(\mathscr{A},\widehat{\mathscr{X}_\varepsilon})\leqslant\varepsilon$.

Proof. As $\mathscr{A}$ is compact, it is bounded and there exists a box $[\mathbf{x}]_\mathscr{A}\subset\mathbb{R}^n$ such that $\mathscr{A}\subset[\mathbf{x}]_\mathscr{A}$. Let $\widehat{\mathscr{X}^0_\mathscr{A}}$ be a subpaving built by successive bisections of $[\mathbf{x}]_\mathscr{A}$ into subboxes $[\mathbf{x}]_i$, $i=1,\dots,\bar{i}$, all with width less than ε. Assume $\widehat{\mathscr{X}_\varepsilon}$ is built by the following algorithm

```
For $i = 1$ to $\bar{i}$
    if $d_0([\mathbf{x}]_i, \mathscr{A}) > \varepsilon$, remove $[\mathbf{x}]_i$ from $\widehat{\mathscr{X}^{i-1}_{\mathscr{A}}}$ to get $\widehat{\mathscr{X}^{i}_{\mathscr{A}}}$,
    else, $\widehat{\mathscr{X}^{i}_{\mathscr{A}}} = \widehat{\mathscr{X}^{i-1}_{\mathscr{A}}}$
$\widehat{\mathscr{X}_\varepsilon} = \widehat{\mathscr{X}^{\bar{i}}_{\mathscr{A}}}$
```

If $d_0([\mathbf{x}]_i, \mathscr{A}) > \varepsilon$, then from Property 3, $[\mathbf{x}]_i \cap \mathscr{A} = \emptyset$. For each $i = 1, \ldots, \bar{i}$, $\mathscr{A} \subset \widehat{\mathscr{X}^{i}_{\mathscr{A}}}$, so $d_0(\mathscr{A}, \widehat{\mathscr{X}_\varepsilon}) = 0$ from Property 4. Moreover,

$$\begin{aligned} d_0(\widehat{\mathscr{X}_\varepsilon}, \mathscr{A}) &= \max_{\mathbf{x} \in \widehat{\mathscr{X}_\varepsilon}} d_0(\mathbf{x}, \mathscr{A}) \\ &= \max_{[\mathbf{x}]_i \in \widehat{\mathscr{X}_\varepsilon}} \left(\max_{\mathbf{x} \in [\mathbf{x}]_i} d_0(\mathbf{x}, \mathscr{A}) \right) \leqslant \varepsilon \end{aligned}$$

by construction and thus $d(\mathscr{A}, \widehat{\mathscr{X}_\varepsilon}) \leqslant \varepsilon$. □

As subpavings consist of boxes, interval operations are easily extended to subpavings. The way in which subpavings are described will, however, have a direct impact on the complexity of these operations and must therefore be designed carefully.

4.2. Description

To illustrate how subpavings will be represented, consider the two-dimensional example in Figure 1. The compact set $\mathscr{X}$ (in dark grey), included in the box $[\mathbf{x}_0] = [0, 8]^{\times 2}$, is represented by the subpaving $\hat{\mathscr{X}}$ (in light grey).

The subboxes of $\hat{\mathscr{X}}$, as described by Figure 1, can be enumerated in the list

$$\mathscr{L} = \{[0, 2] \times [0, 4], [4, 6] \times [2, 4], [4, 6] \times [4, 8]\} \tag{15}$$

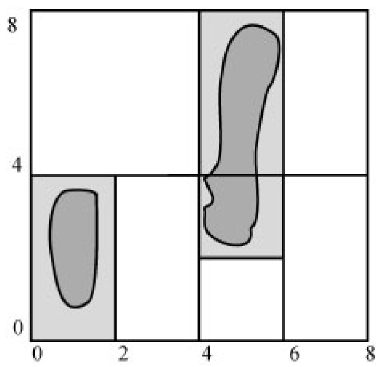

Figure 1. Set $\mathscr{X}$ (in dark grey) described using the subpaving $\hat{\mathscr{X}}$ (in light grey).

As all these boxes are obtained from $[\mathbf{x}_0]$ by a succession of bisections and selections, one may alternatively store the initial box together with the history of these operations. It is then necessary to introduce a notation to describe how boxes are bisected and selected. If $[\mathbf{x}]$ is an n-dimensional box and if the midpoint of $[x]_j$ is denoted by $m([x]_j) = (\overline{x_j} + \underline{x_j})/2$, then

$$L_j[\mathbf{x}] = ([x]_1, \dots, [x]_{j-1}, [\underline{x_j}, m([x]_j)], [x]_{j+1}, \dots, [x]_n) \tag{16}$$

$$R_j[\mathbf{x}] = ([x]_1, \dots, [x]_{j-1}, [m([x]_j), \overline{x_j}], [x]_{j+1}, \dots, [x]_n) \tag{17}$$

will, respectively, be called the *left* and *right subboxes* resulting from the bisection of $[\mathbf{x}]$ across its jth dimension. One may then write the list (15) as

$$\mathscr{L} = \{L_1 L_2 L_1[\mathbf{x}_0], \quad R_2 L_1 L_2 R_1[\mathbf{x}_0], \quad L_1 R_2 R_1[\mathbf{x}_0]\} \tag{18}$$

This description is obviously not unique. For instance, $R_1 L_2 L_1[\mathbf{x}_0] = L_2 R_1 L_1[\mathbf{x}_0]$. This ambiguity can be avoided by agreeing on some canonical bisection rule. The canonical rule used in this paper is to cut any box $[\mathbf{x}] \subset \mathbb{R}^n$ across its *main dimension* j, defined as

$$j = \inf\{i = 1, \dots, n \mid w([x]_i) = w([\mathbf{x}])\} \tag{19}$$

but other canonical rules could of course be used.

A subpaving will be called *regular* if it can be obtained from some initial box by applying the canonical rule. Note that the subpaving built in the proof of Proposition 1 is regular, so this proposition holds true for regular subpavings. For such subpavings, the direction of cut can easily be retrieved, so indexing of L and R is no longer necessary and the list (15) can be written as $\{LLL[\mathbf{x}_0], RLLR[\mathbf{x}_0], LRR[\mathbf{x}_0]\}$.

Any regular subpaving $\hat{\mathscr{X}}$ can be viewed as an *ordered binary-tree* [18], i.e. a finite set of *nodes*, which is either empty or consists of one node, the *root* of the tree, and two disjoint ordered binary trees, the *left* and *right subtrees*. The notion of regular subpaving is particularly interesting for the many interval algorithms that are based on bisection.

The binary tree associated with $\hat{\mathscr{X}}$ of Figure 1 is represented by Figure 2.

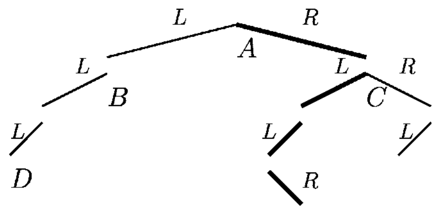

Figure 2. Tree representation of $\hat{\mathscr{X}}$ of Figure 1. The leaf at the end of the branch in bold corresponds to the box $[4, 6] \times [2, 4]$.

Node A is the *root* of the tree. Nodes B and C are, respectively, the *left* and *right children* of A; they are *siblings* because they have the same *parent* A. A has *left* and *right subtrees*; B has an *empty* right subtree. Any node with two empty subtrees, such as D, is called a *leaf.*

This tree is constructed using the list (15). The growth of the branches is determined by the way in which the initial box $[\mathbf{x}_0]$, corresponding to the root, is bisected. For example, the leaf at the end of the bold branch in Figure 2 corresponds to the subbox $RLLR[\mathbf{x}_0] = [4, 6] \times [2, 4]$. A node indicates that the associated subbox has been bisected according to the canonical bisection rule. Any subbox corresponding to a leaf is entirely in the subpaving. The *depth* of a subbox corresponds to the number of bisections applied to get this subbox from the root box. Thus, the depth of the box $[4, 6] \times [2, 4]$ is 4.

A tree (subpaving) is said to be *minimal* if it has no siblings leaves (subboxes). Such siblings leaves should be eliminated, with their parent node becoming a leaf instead.

The notions of binary trees and regular subpavings are equivalent, so the vocabulary for trees will be used for regular subpavings. In what follows, the tree representation will be adopted to take advantage of its natural recursivity. But the equivalence between the tree and its list of boxes should be kept in mind.

Details on how regular subpavings and algorithms for manipulating them are implemented on a computer can be found in Appendix C.

5. IMPLEMENTABLE CORRECTION STEP

The implementable counterpart of Algorithm 1 (Section 2) also alternates prediction and correction. As the correction step is simpler, it will be presented first. It requires characterizing $\mathcal{X}_{\ell+1} = \{\mathbf{x} \in \mathcal{X}_{\ell+} | \mathbf{h}_{\ell+1}(\mathbf{x}) \in \mathcal{Y}_{\ell+1}\}$. This task belongs to the class of *set-inversion* problems, formulated as follows: given two sets $\mathcal{X} \subset \mathbb{R}^n$, $\mathcal{Y} \subset \mathbb{R}^p$ and a function $\mathbf{h} : \mathbb{R}^n \to \mathbb{R}^p$, characterize the set $\mathbf{h}_{\mathcal{X}}^{-1}(\mathcal{Y}) = \{\mathbf{x} \in \mathcal{X} | \mathbf{h}(\mathbf{x}) \in \mathcal{Y}\}$.

This problem is solved in an approximated but guaranteed way by using the algorithm Sivia (for Set Inversion Via Interval Analysis), presented in the context of parameter estimation in References [19, 20]. This algorithm brackets $\mathbf{h}_{\mathcal{X}}^{-1}(\mathcal{Y})$ between inner and outer subpavings. Here, a new recursive version will be presented. This new version evaluates only an outer subpaving but allows the computation on subpavings.

Assume that $\mathcal{X}$ and $\mathcal{Y}$ are, respectively, enclosed in the subpavings $\hat{\mathcal{X}}$ and $\hat{\mathcal{Y}}$. The task of Sivia is to enclose the set $\mathcal{S} = \mathbf{h}_{\hat{\mathcal{X}}}^{-1}(\hat{\mathcal{Y}}) = \{\mathbf{x} \in \hat{\mathcal{X}} | \mathbf{h}(\mathbf{x}) \in \hat{\mathcal{Y}}\}$ in a subpaving $\hat{y}$, with the help of an inclusion function $\mathbf{h}_{[]}$ of $\mathbf{h}$. For this purpose, a recursive structure will be used to scan all nodes of $\hat{\mathcal{X}}$. Let $[\mathbf{x}]$ be the box corresponding to a given node M of $\hat{\mathcal{X}}$. If $\mathbf{h}_{[]}([\mathbf{x}]) \subset \hat{\mathcal{Y}}$, then the whole subpaving stemming from M is included in the solution set $\mathcal{S}$ and stored in $\hat{\mathcal{S}}$. If $\mathbf{h}_{[]}([\mathbf{x}]) \cap \hat{y} = \emptyset$, then $[\mathbf{x}] \cap \mathcal{S} = \emptyset$. Therefore, M and the subpavings stemming from it can be discarded from further consideration. If the results of the two preceding tests are negative and if $w([\mathbf{x}]) > \varepsilon$, then the subtrees stemming from M have to be tested, else the box corresponding to M is considered small enough to be incorporated in $\hat{\mathcal{S}}$. The positive real number ε is chosen by the user to specify the desired accuracy in the description of the set to be characterized.

The C++ code of this recursive version of Sivia is given in Appendix C. Sivia $(\hat{\mathcal{X}}, \mathbf{h}_{[]}, \hat{\mathcal{Y}}, \varepsilon)$ returns a subpaving containing $\mathbf{h}_{\hat{\mathcal{X}}}^{-1}(\hat{\mathcal{Y}})$. This outer approximation of $\mathbf{h}_{\hat{\mathcal{X}}}^{-1}(\hat{\mathcal{Y}})$ can, in principle, be made as accurate as desired, as indicated by the following proposition.

Proposition 2

Consider a subpaving $\hat{\mathscr{Y}} \subset \mathbb{R}^p$ and a function $\mathbf{h}: \mathbb{R}^n \to \mathbb{R}^p$ with a convergent inclusion function $\mathbf{h}_{[]}$ in $\hat{\mathscr{X}} \subset \mathbb{R}^n$. For any $\eta > 0$, there exists $\varepsilon > 0$ such that

$$d(\text{Sivia}(\hat{\mathscr{X}}, \mathbf{h}_{[]}, \hat{\mathscr{Y}}, \varepsilon), \mathbf{h}_{\hat{\mathscr{X}}}^{-1}(\hat{\mathscr{Y}})) \leqslant \eta \tag{20}$$

Proof. Similar to that presented in Reference [20] for the case $\hat{\mathscr{Y}} = [\mathbf{y}]$. □

In practice, there is of course a limit to the accuracy that can be achieved, because of the complexity of the resulting description.

Example 1

Consider the subpavings $\hat{\mathscr{A}} = \{[-3, 3]^{\times 2}\}$ and $\widehat{\mathscr{B}_1} = \{[2, 4]\}$, and the function

$$h_1(x_1, x_2) = (x_1 - 1)^2 + (x_2 - 1)^2$$

The set $\mathscr{C}_1 \subset \hat{\mathscr{A}}$ such that $h_1(\mathscr{C}_1) = \widehat{\mathscr{B}_1}$ is guaranteed to belong to $\widehat{\mathscr{C}_1} = \text{Sivia}(\hat{\mathscr{A}}, h_{1[]}, \widehat{\mathscr{B}_1}, \varepsilon)$, shown in Figure 3. Consider now the subpaving $\widehat{\mathscr{B}_2} = \{[2, 4]\}$ and the function

$$h_2(x_1, x_2) = (x_1 + 1)^2 + (x_2 + 1)^2$$

The set $\mathscr{C}_2 \subset \widehat{\mathscr{C}_1}$ such that $h_2(\mathscr{C}_2) = \widehat{\mathscr{B}_2}$ is guaranteed to belong to $\widehat{\mathscr{C}_2} = \text{Sivia}(\widehat{\mathscr{C}_1}, h_{2[]}, \widehat{\mathscr{B}_2}, \varepsilon)$, shown in Figure 4. In this example, $h_{1[]}$ and $h_{2[]}$ were taken as the natural inclusion functions for h_1 and h_2.

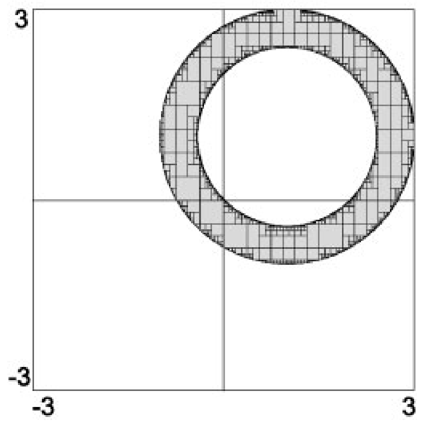

Figure 3. $\widehat{\mathscr{C}_1}$ as computed by Sivia for Example 1.

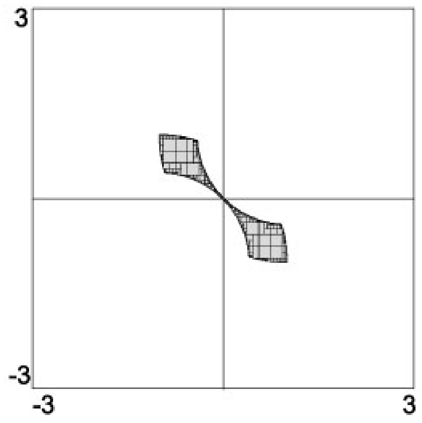

Figure 4. $\widehat{\mathscr{C}_2}$ as computed by Sivia for Example 1.

6. IMPLEMENTABLE PREDICTION STEP

The prediction step requires characterizing

$$\mathscr{X}_{\ell+} = \{\mathbf{f}_\ell(\mathbf{x}, \mathbf{w}) | \mathbf{x} \in \mathscr{X}_\ell, \mathbf{w} \in [\mathbf{w}]_\ell\} \tag{21}$$

This can be included in the more general problem of direct image evaluation, which is at the core of interval arithmetic: given two compact sets $\mathscr{X} \subset \mathbb{R}^n$ and $\mathscr{Y}_0 \subset \mathbb{R}^r$ and a function $\mathbf{f}: \mathbb{R}^n \to \mathbb{R}^r$, characterize the set $\mathscr{Y} \subset \mathscr{Y}_0$ such that $\mathscr{Y} = \{\mathbf{f}(\mathbf{x}) \in \mathscr{Y}_0 | \mathbf{x} \in \mathscr{X}\}$. When $r = 1$, it has been shown [16, 21], that the approximation could be made as precise as desired, provided that the inclusion function $f_{[]}$ be Lipschitz [16]. When $r > 1$, we shall see that it is still possible to approximate the image set with arbitrary precision, under techniques that depend on whether $\mathbf{f}$ can be inverted. In the context of state prediction, $\mathscr{X}_\ell \times [\mathbf{w}]_\ell$ plays the role of $\mathscr{X}$, and $\mathscr{X}_{\ell+}$ that of $\mathscr{Y}$.

6.1. *Prediction when* f *is not invertible*

When $\mathbf{f}$ is not invertible (e.g. for state prediction in the presence of state perturbations), the approximate image of an initial set $\mathscr{X}$ can be computed by using a description of $\mathscr{X}$ by a non-minimal subpaving consisting of P boxes $[\mathbf{x}]_i$ each with width less than ε. The images of all these boxes are evaluated using an inclusion function $\mathbf{f}_{[]}$ of $\mathbf{f}$ and stored in a list $\mathscr{L}_\varepsilon$. One thus gets P *image* boxes, each of which contains the image set of the associated initial box. The image $\mathscr{Y}$ of $\mathscr{X}$ is therefore included in the union of all of them. Finally, these boxes are merged into a subpaving to allow further processing.

The C+ + code of the corresponding function ImageSp (for Image SubPaving evaluation) is given in Appendix C. $\widehat{\mathscr{Y}_\varepsilon} = \text{ImageSp}([\mathbf{s}]^0, \mathbf{f}_{[]}, \widehat{\mathscr{X}}, \varepsilon)$ returns an outer approximation $\widehat{\mathscr{Y}_\varepsilon}$ of the image of the subpaving $\widehat{\mathscr{X}}$ by the function $\mathbf{f}$, or rather of the part of it that is included in some prior search box $[\mathbf{s}]^0$.

The quality of the resulting approximation of the image set depends, of course, on that of the inclusion function $\mathbf{f}_{[]}$ of $\mathbf{f}$ and on the precision parameter ε. The next proposition will show that the precision can, at least in principle, be made arbitrarily good. It involves the following definition of a Lipschitz function, inspired from Reference [16, p. 34].

Definition 1

Consider $\mathbf{f}: \mathbb{R}^n \to \mathbb{R}^r$ and $\mathscr{D}$ a compact subset of $\mathbb{R}^n$. $\mathbf{f}$ is *Lipschitz* in $\mathscr{D}$ if there exists a constant λ such that for all compact sets $\mathscr{A} \subset \mathscr{D}$ and $\mathscr{B} \subset \mathscr{D}$, $d(\mathbf{f}(\mathscr{A}), \mathbf{f}(\mathscr{B})) \leqslant \lambda\, d(\mathscr{A}, \mathscr{B})$.

Let us prove that any function $\mathbf{f}: \mathbb{R}^n \to \mathbb{R}^r$ satisfying the classical Lipschitz condition in a compact set $\mathscr{D} \subset \mathbb{R}^n$ is also *Lipschitz* in $\mathscr{D}$ in the sense of Definition 1. If $\mathscr{A}$ and $\mathscr{B}$ are compact subsets of $\mathscr{D}$ and $\mathbf{f}$ is Lipschitz in $\mathscr{D}$, then $\mathbf{f}(\mathscr{A})$ and $\mathbf{f}(\mathscr{B})$ are compact sets and there exists $\mathbf{a}_0 \in \mathscr{A}$, $\mathbf{b}_0 \in \mathscr{B}$ and a constant λ such that $d(\mathbf{f}(\mathscr{A}), \mathbf{f}(\mathscr{B})) = d_\infty(\mathbf{f}(\mathbf{a}_0), \mathbf{f}(\mathbf{b}_0)) \leqslant \lambda \;\; d_\infty(\mathbf{a}_0, \mathbf{b}_0)$. Since $d_\infty(\mathbf{a}_0, \mathbf{b}_0) \leqslant d(\mathscr{A}, \mathscr{B})$, this implies that $d(\mathbf{f}(\mathscr{A}), \mathbf{f}(\mathscr{B})) \leqslant \lambda d(\mathscr{A}, \mathscr{B})$. Moreover, it is straightforward to prove that if $\mathbf{f}$ is Lipschitz in $\mathscr{D}$, then its natural inclusion function $\mathbf{f}_{[]}$ is also Lipschitz in $\mathscr{D}$ and thus convergent (see Lemma 2).

Proposition 3

Consider a subpaving $\widehat{\mathscr{X}} \subset \mathbb{R}^n$, a function $\mathbf{f}: \mathbb{R}^n \to \mathbb{R}^r$, with an inclusion function $\mathbf{f}_{[]}$ that is Lipschitz in $\widehat{\mathscr{X}}$, and $[\mathbf{s}]^0 \subset \mathbb{R}^r$ such that $\mathbf{f}(\widehat{\mathscr{X}}) \subset [\mathbf{s}]^0$. For any $\eta > 0$, there exists $\varepsilon > 0$ such that $d(\textsc{ImageSp}([\mathbf{s}]^0, \mathbf{f}_{[]}, \widehat{\mathscr{X}}, \varepsilon), \mathbf{f}(\widehat{\mathscr{X}})) \leqslant \eta$.

Proof. See Appendix B. □

For any given function $\mathbf{f}$ with a Lipschitz inclusion function, ImageSp thus provides a convergent *subpaving inclusion function.*

6.2. Prediction when f is invertible

When $\mathbf{f}$ is invertible, prediction may be cast in the formalism of set inversion, as the problem of finding $\mathscr{S} = \{\mathbf{x} \in \mathscr{S}_0 | \mathbf{f}^{-1}(\mathbf{x}) \in \mathscr{X}\}$. The search set $\mathscr{S}_0$ should be taken large enough to be guaranteed to contain the set of interest. If $\widehat{\mathscr{S}_0}$ and $\widehat{\mathscr{X}}$ are subpavings enclosing $\mathscr{S}_0$ and $\mathscr{X}$, then $\mathscr{S}$ can be approximated by

$$\widehat{\mathscr{S}_\varepsilon} = \textsc{Sivia}(\widehat{\mathscr{S}_0}, \mathbf{f}_{[]}^{-1}, \widehat{\mathscr{X}}, \varepsilon) \tag{22}$$

provided that an inclusion function is available for $\mathbf{f}^{-1}$. Assuming that $\mathbf{f}$ is invertible may seem rather strong, but in many models based on physical laws this only amounts to changing the sign of time in the equations.

7. GUARANTEED STATE ESTIMATOR

An approximate but guaranteed version of Algorithm 1 can now be proposed, where it is assumed that $[\mathbf{s}]$ is a (possibly very large) search box in state space where the state vector is assumed to stay from $\ell = 0$ to $\ell = \bar{\ell}$.

Algorithm 2

For $\ell = 0$ *to* $\bar{\ell}$, do

1. *Guaranteed prediction*: Compute the set estimate for the state at step $\ell + 1$ before measurement as

$$\widehat{\mathcal{X}_{\ell+}} = \textsc{ImageSp}([\mathbf{s}], \mathbf{f}_{\ell[]}, \widehat{\mathcal{X}_\ell^e}, \varepsilon) \tag{23}$$

where $\widehat{\mathcal{X}_\ell^e}$ is an extended subpaving containing $\widehat{\mathcal{X}_\ell} \times [\mathbf{w}]_\ell$.
If the state perturbations are negligible and $\mathbf{f}_\ell$ is invertible, then (23) may be replaced by

$$\widehat{\mathcal{X}_{\ell+}} = \textsc{Sivia}([\mathbf{s}], \mathbf{f}_{\ell[]}^{-1}, \widehat{\mathcal{X}_\ell}, \varepsilon) \tag{24}$$

2. *Guaranteed correction*: Discard all elements of $\widehat{\mathcal{X}_{\ell+}}$ that can be proved to be incompatible with measurements at step $\ell + 1$ by computing

$$\widehat{\mathcal{X}_{\ell+1}} = \textsc{Sivia}(\widehat{\mathcal{X}_{\ell+}}, \mathbf{h}_{\ell[]}, \mathcal{Y}_{\ell+1}, \varepsilon) \tag{25}$$

Some properties of the estimator provided by Algorithm 2 will now be established.

7.1. Prediction by ImageSp

Assume that
(A1) A prior search box $[\mathbf{s}]$ in state space is available, large enough to contain $\widehat{\mathcal{X}_\ell}$ for any $\ell \geqslant 0$.
(A2) Inclusion functions are available for $\mathbf{f}_\ell$ and $\mathbf{h}_\ell$ for $\ell = 0, 1, \ldots, \bar{\ell}$, and these inclusion functions are Lipschitz, which implies that $\mathbf{f}_\ell$ and $\mathbf{h}_\ell$ are also Lipschitz.
For the sake of simplicity, it will also be assumed that the set of all possible initial states $\mathcal{X}_0$ is a subpaving, so $\widehat{\mathcal{X}_0}$ can be taken equal to $\mathcal{X}_0$.

Proposition 4 (*Guaranteedness*)

For $\ell = 0, \ldots, \bar{\ell}$, $\mathbf{x}_\ell \in \widehat{\mathcal{X}_\ell}$.

Proof. The proof is by induction. $\widehat{\mathcal{X}_0}$ contains $\mathbf{x}_0$ by assumption. Assume that $\mathcal{X}_\ell \subset \widehat{\mathcal{X}_\ell}$. For the prediction step, ImageSp ensures that $\mathcal{X}_{\ell+} \subset \widehat{\mathcal{X}_{\ell+}}$. For the correction step, Sivia returns $\widehat{\mathcal{X}_{\ell+1}}$, which is guaranteed to contain $\mathbf{h}^{-1}(\mathcal{Y}_{\ell+1}) \cap \widehat{\mathcal{X}_{\ell+}}$. As $\mathcal{X}_{\ell+1} = \mathbf{h}^{-1}(\mathcal{Y}_{\ell+1}) \cap \mathcal{X}_{\ell+}$, it follows from the two preceding inclusions that $\mathcal{X}_{\ell+1} \subset \mathbf{h}^{-1}(\mathcal{Y}_{\ell+1}) \cap \widehat{\mathcal{X}_{\ell+}} \subset \widehat{\mathcal{X}_{\ell+1}}$. Since $\mathcal{X}_{\ell+1}$ is obtained by the idealized algorithm, $\mathbf{x}_{\ell+1}$ belongs to $\mathcal{X}_{\ell+1}$ and thus to $\widehat{\mathcal{X}_{\ell+1}}$. □

Proposition 5 (*Arbitrary precision*)

For $\ell = 0, \ldots, \bar{\ell}$, and for any $\eta > 0$, there exists $\varepsilon > 0$ such that $\widehat{\mathcal{X}_\ell}$ as computed by Algorithm 2 satisfies $d(\widehat{\mathcal{X}_\ell}, \mathcal{X}_\ell) \leqslant \eta$.

Proof. As $\mathbf{f}_\ell$ is Lipschitz in $[\mathbf{s}]\times[\mathbf{w}]_\ell$, for $\ell = 0,\ldots,\bar{\ell}$ there exists $K_\ell \geqslant 0$ such that for any $\mathscr{A} \subset [\mathbf{s}]\times[\mathbf{w}]_\ell$ and $\mathscr{B} \subset [\mathbf{s}]\times[\mathbf{w}]_\ell$, $d(\mathbf{f}_\ell(\mathscr{A}), \mathbf{f}_\ell(\mathscr{B})) \leqslant K_\ell d(\mathscr{A}, \mathscr{B})$. Let

$$K = \max(1, K_0, \ldots, K_{\bar{\ell}}) \tag{26}$$

We shall prove by induction that, for any $\eta > 0$, there exists $\varepsilon_\ell > 0$ such that $d(\widehat{\mathscr{X}_\ell}, \mathscr{X}_\ell) \leqslant \eta_\ell$, with $\eta_\ell = \eta/(3K)^{\bar{\ell}-\ell} \leqslant \eta$. The ε of Proposition 5 will then be taken equal to $\min(\varepsilon_1, \ldots, \varepsilon_{\bar{\ell}})$.

Since $\widehat{\mathscr{X}_0} = \mathscr{X}_0$, $d(\widehat{\mathscr{X}_0}, \mathscr{X}_0) = 0_i$, assume that at time index $\ell \geqslant 1$, there exists $\varepsilon_\ell > 0$ such that $d(\widehat{\mathscr{X}_\ell}, \mathscr{X}_\ell) \leqslant \eta_\ell \leqslant \eta$. We shall then show that at time index $\ell + 1 \leqslant \bar{\ell}$, $d(\widehat{\mathscr{X}_{\ell+1}}, \mathscr{X}_{\ell+1}) \leqslant \eta_{\ell+1} \leqslant \eta$.

Since $\mathbf{f}_\ell$ is Lipschitz in $[\mathbf{s}]\times[\mathbf{w}]_\ell$,

$$d(\mathbf{f}_\ell(\widehat{\mathscr{X}_\ell^{\mathrm{e}}}), \mathbf{f}_\ell(\widehat{\mathscr{X}_\ell^{\mathrm{e}}})) = K\, d(\widehat{\mathscr{X}_\ell^{\mathrm{e}}}, \widehat{\mathscr{X}_\ell^{\mathrm{e}}}) \tag{27}$$

where K is defined by (26), $\mathscr{X}_\ell^{\mathrm{e}} = \mathscr{X}_\ell \times [\mathbf{w}]_\ell$ and $\widehat{\mathscr{X}_\ell^{\mathrm{e}}} = \widehat{\mathscr{X}_\ell} \times [\mathbf{w}]_\ell$. Property 6 implies that

$$d(\widehat{\mathscr{X}_\ell^{\mathrm{e}}}, \mathscr{X}_\ell^{\mathrm{e}}) = d(\widehat{\mathscr{X}_\ell}, \mathscr{X}_\ell) \tag{28}$$

thus

$$d(\mathbf{f}_\ell(\widehat{\mathscr{X}_\ell^{\mathrm{e}}}), \mathbf{f}_\ell(\mathscr{X}_\ell^{\mathrm{e}})) \leqslant K\, d(\widehat{\mathscr{X}_\ell}, \mathscr{X}_\ell) \leqslant K\eta_\ell \tag{29}$$

According to Proposition 3, there exists $\varepsilon_1 > 0$ such that the outer approximation $\widehat{\mathscr{X}_{\ell+}}$ of the prediction set computed by ImageSp satisfies

$$d(\widehat{\mathscr{X}_{\ell+}}, \mathbf{f}_\ell(\widehat{\mathscr{X}_\ell^{\mathrm{e}}})) \leqslant K\eta_\ell \tag{30}$$

with $\widehat{\mathscr{X}_{\ell+}}$ evaluated using (23). According to Proposition 2, there exists $\varepsilon_2 > 0$ such that the outer approximation $\widehat{\mathscr{X}_{\ell+1}}$ of the corrected set computed by Sivia satisfies

$$d(\widehat{\mathscr{X}_{\ell+1}}, \mathbf{h}_{\ell+1}^{-1}(\mathscr{Y}_{\ell+1}) \cap \widehat{\mathscr{X}_{\ell+}}) \leqslant K\eta_\ell \tag{31}$$

where $\widehat{\mathscr{X}_{\ell+1}}$ is computed using (25). Since $\mathscr{X}_{\ell+1} = \mathbf{h}_{\ell+1}^{-1}(\mathscr{Y}_{\ell+1}) \cap \mathscr{X}_{\ell+}$,

$$d(\widehat{\mathscr{X}_{\ell+1}}, \mathscr{X}_{\ell+1}) = d(\widehat{\mathscr{X}_{\ell+1}}, \mathbf{h}_{\ell+1}^{-1}(\mathscr{Y}_{\ell+1}) \cap \mathscr{X}_{\ell+}) \tag{32}$$

The triangular inequality then yields

$$d(\widehat{\mathscr{X}_{\ell+1}}, \mathscr{X}_{\ell+1}) \leqslant d(\widehat{\mathscr{X}_{\ell+1}}, \mathbf{h}_{\ell+1}^{-1}(\mathscr{Y}_{\ell+1}) \cap \widehat{\mathscr{X}_{\ell+}}) + d(\mathbf{h}_{\ell+1}^{-1}(\mathscr{Y}_{\ell+1}) \cap \widehat{\mathscr{X}_{\ell+}}, \mathbf{h}_{\ell+1}^{-1}(\mathscr{Y}_{\ell+1}) \cap \mathscr{X}_{\ell+})$$

and (31) implies that

$$d(\widehat{\mathscr{X}_{\ell+1}}, \mathscr{X}_{\ell+1}) \leqslant K\eta_\ell + d(\mathbf{h}_{\ell+1}^{-1}(\mathscr{Y}_{\ell+1}) \cap \widehat{\mathscr{X}_{\ell+}}, \mathbf{h}_{\ell+1}^{-1}(\mathscr{Y}_{\ell+1}) \cap \mathscr{X}_{\ell+}) \tag{33}$$

Using Property 5, one can deduce from (33) that

$$d(\widehat{\mathscr{X}_{\ell+1}}, \mathscr{X}_{\ell+1}) \leqslant K\eta_\ell + d(\widehat{\mathscr{X}_{\ell+}}, \mathscr{X}_{\ell+}) \tag{34}$$

Apply the triangular inequality again to get

$$d(\widehat{\mathscr{X}_{\ell+1}}, \mathscr{X}_{\ell+1}) \leqslant K\eta_\ell + d(\widehat{\mathscr{X}_{\ell+}}, \mathbf{f}_\ell(\widehat{\mathscr{X}_\ell^{\mathrm{e}}})) + d(\mathbf{f}_\ell(\widehat{\mathscr{X}_\ell^{\mathrm{e}}}), \mathscr{X}_{\ell+}) \tag{35}$$

As $\mathscr{X}_{\ell+} = \mathbf{f}_\ell(\mathscr{X}_\ell^{\mathrm{e}})$, (29) and (30) then imply that

$$d(\widehat{\mathscr{X}_{\ell+1}}, \mathscr{X}_{\ell+1}) \leqslant K\eta_\ell + K\eta_\ell + K\eta_\ell = 3K\eta_\ell \tag{36}$$

The variable η_ℓ should therefore satisfy the recurrence equation $\eta_{\ell+1} = 3\mathbf{K}\eta_\ell$, with the terminal condition $\eta_{\bar{\ell}} = \eta$. The solution of this recurrence equation is then

$$\eta_\ell = \eta/(3K)^{\bar{\ell}-\ell} \leqslant \eta, \quad \ell = 0, \ldots, \bar{\ell} \tag{37}$$

So there exists $\varepsilon_{\ell+1} = \inf(\varepsilon_1, \varepsilon_2)$ such that $d(\widehat{\mathscr{X}_{\ell+1}}, \mathscr{X}_{\ell+1}) \leqslant \eta_{\ell+1} \leqslant \eta$. □

Remark 4

The sizes of the predicted and corrected sets do not depend on the width of the prior search set [**s**], provided that it is large enough for (A1) to be satisfied and that the precision parameter ε is taken small enough.

Remark 5

The properties of guaranteedness and arbitrary precision mean that when η tends to zero the sets $\widehat{\mathscr{X}_\ell}$ computed by the estimation algorithm converge from the outside to the actual sets $\mathscr{X}_\ell$ for any finite value of ℓ. This convergence thus differs from that usually considered in statistics, where one is interested in the asymptotic properties of point estimators when the number of data points tends to infinity.

7.2. Prediction by Sivia

When **f** is invertible, and the state perturbations are negligible, (24) may be used instead of (23) for the prediction step. Propositions 4 and 5 can still be guaranteed under assumptions (A1) and

(A2′) Inclusion functions are available for $\mathbf{f}_\ell^{-1}$ and $\mathbf{h}_\ell$ for $\ell = 0, 1, \ldots, \bar{\ell}$, and these inclusion functions are Lipschitz in their arguments.

The proof of Proposition 4 is unchanged. The poor of Proposition 5 simplifies because it is no longer necessary to introduce $\mathscr{X}_\ell^{\mathrm{e}}$ and $\widehat{\mathscr{X}_\ell^{\mathrm{e}}}$, since, according to Proposition 2, there exists $\varepsilon_1 > 0$ such that the outer approximation $\widehat{\mathscr{X}_{\ell+}}$ of the predicted set computed by (24) satisfies

$$d(\widehat{\mathscr{X}_{\ell+}}, \mathbf{f}_\ell(\widehat{\mathscr{X}_\ell})) \leqslant K\eta_\ell \tag{38}$$

which replaces (30).

Remarks 4 and 5 of course remain valid.

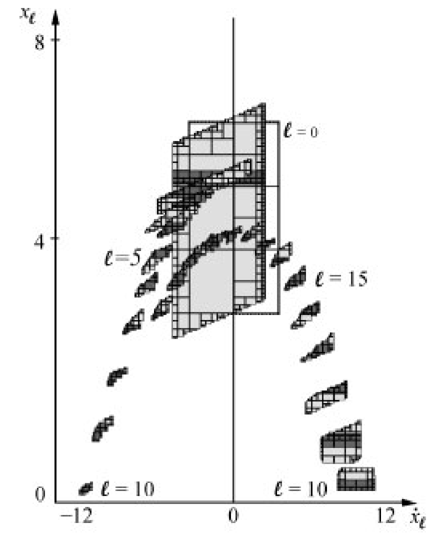

Figure 5. Ball motion. The predicted sets are in light grey and corrected sets in dark grey. At time $t = 1$ s (step $\ell = 10$), the bounce may or may not have taken place, so the speed may be positive or negative.

7.3. Example

Algorithm 2 has been applied to the bouncing ball example. The sampling period is $T = 0.1$ s, $\rho = 0.2$ m, g is taken as $10\,\mathrm{m\,s^{-2}}$. The initial state is only known to belong to $[\mathbf{x}_0] = [3\text{ m}, 6\text{ m}] \times [-3\,\mathrm{m\,s^{-1}}, 3\,\mathrm{m\,s^{-1}}]$ and the state perturbation γ is assumed to belong to $[0.1, 0.25]$. The search box $[\mathbf{s}]$ is taken equal to $[0\text{ m}, 8\text{ m}] \times [-12\,\mathrm{m\,s^{-1}}, 12\,\mathrm{m\,s^{-1}}]$. At each time $t = \ell T$, $\ell = 0, 1, \ldots, 24$, the position of the ball is measured, with the measurement noise v_ℓ assumed to belong to $[-0.1\text{ m}, 0.1\text{ m}]$. The observation equation is $y_\ell = x_\ell + v_\ell$. Table A2 describes an inclusion function for the function of Table A1, evaluating the position at time $(\ell + 1)\,T$ from that at time ℓT.

Figure 5 displays the predicted subpaving $\widehat{\mathcal{X}_{\ell+}}$ in light grey and the corrected subpaving $\widehat{\mathcal{X}_{\ell+1}}$ in dark grey. The data were generated by the algorithm of Table A1 with $\mathbf{x}_0 = (5\text{ m}, 0\,\mathrm{m\,s^{-1}})^{\mathrm{T}}$ and $\gamma = 0.2$.

Remark 6

If $\widehat{\mathcal{X}_0}$ does not contain the actual initial state, or if bounds on the measurement noise or state perturbations are violated, then the set estimate may become empty, thus proving that at least one of the assumptions is erroneous.

8. CONCLUSIONS

A new recursive non-linear discrete-time state estimator has been presented. At any given time, it returns a set guaranteed to enclose all the values of the state vector that are consistent with the information available so far. To the best of our knowledge, it is the first one for which this claim can be made in the presence of bounded state perturbations and measurement noise.

As in classical Kalman filtering, this estimator alternates prediction and correction steps. The prediction step uses either a direct image evaluator (ImageSp) or a procedure for the computation of inverse images based on the algorithm Sivia. ImageSp divides the boxes of the original subpaving into smaller boxes and merges the images of all these boxes into an approximate image subpaving. Sivia proceeds in the opposite direction, starting from the image set, and may be more efficient when applicable. From the predicted subpaving thus obtained, the correction step discards all the values of the state vector that are proved to be inconsistent with the newly available observations. The important theoretical properties of guaranteedness and arbitrary precision have been established. The estimator has been applied to a simple example, nevertheless representative of some difficulties encountered by classical estimators when dealing with hybrid systems.

It seems important to point out that this estimator does not require the set of all the values of the state vector that are consistent with the information available to be connected. Ambiguous situations where the state is not globally observable can therefore be dealt with, without any modification.

A key element for a recursive implementation of the estimator on a computer was the introduction of subpavings, which allow the computation of outer approximations for sets, with tunable precision. It is easy to extend all arithmetic operators and usual functions on floating-point numbers to the class of subpavings, in any language that allows operator overloading (such as C + +). It is thus possible to get a *subpaving computation* that approximates computation on more general sets.

The main limitation of the estimator presented lies in the explosion of complexity with the number of state variables. It is, nevertheless, possible to solve actual tracking problems, see the localization and tracking of a mobile robot described in References [22, 23]. An interesting area for further research is the use of contractors based on interval constraint propagation to struggle against the curse of dimensionality (see References [24–26]).

ACKNOWLEDGEMENTS

The authors wish to thank INTAS for partial support, through Grant RFBR-97-10782.

APPENDIX A: COMPUTATION OF THE MOTION OF THE BALL

The function described by Table A1 computes $\mathbf{x}_{\ell+1}$ as a function of $\mathbf{x}_\ell = (x_\ell, \dot{x}_\ell)^{\mathrm{T}}$ and γ_i g is the acceleration of gravity, T the sampling period and ρ the radius of the ball. $\dot{x}_{bb}$ and $\dot{x}_{ab}$, respectively, denote the speed of the ball just before and after the bounce. The height of the ball at time $(\ell + 1)T$ is evaluated assuming free fall. Provided that it remains greater than ρ, the free-fall model is still valid. Otherwise, the bounce speed $\dot{x}_{ab}$ immediately after the bounce and the time T_b of bounce are evaluated to compute the height and speed at time $(\ell + 1)T$.

When $\mathbf{x}_\ell$ and γ are only known to belong to some intervals, an inclusion function $\mathbf{f}_{[]}$, such as the one described in Table A2, must be used instead of the function $\mathbf{f}$ of Table A1.

Table A1. Function computing $\mathbf{x}_{\ell+1}$ as a function of $\mathbf{x}_\ell = (x_\ell, \dot{x}_\ell)^{\mathrm{T}}$ and γ for the ball motion

```
$\mathbf{f}(x_\ell, \dot{x}_\ell, \gamma)$

const $g, T, \rho$;
$x_{\ell+} = x_\ell + \dot{x}_\ell T - g\frac{T^2}{2}$;
if $(x_{\ell+} \geqslant \rho)$
return $\begin{pmatrix} x_{\ell+} \\ \dot{x}_\ell - gT \end{pmatrix}$;
$\dot{x}_{bb} = \sqrt{2g(x_\ell - \rho) + (\dot{x}_\ell)^2}$;
$T_b = \frac{1}{g}(\dot{x}_\ell + \dot{x}_{bb})$;
$\dot{x}_{ab} = \sqrt{1-\gamma}\,\dot{x}_{bb}$;
return $\begin{pmatrix} \rho + \dot{x}_{ab}(T - T_b) - g\frac{(T-T_b)^2}{2} \\ \dot{x}_{ab} - g(T - T_b) \end{pmatrix}$.
```

The interval $[x_{\ell+}]$ enclosing all possible positions at the next step is first evaluated. If its lower bound $\underline{x_{\ell+}}$ is greater than ρ, then the equations of free fall should be applied. Else, the ball *may* have hit the floor and the algorithm must evaluate all possible speeds $[\dot{x}_{bb}]$ just before the bounce and all possible bounce times $[T_b]$, as well as all possible speeds $[\dot{x}_{ab}]$ after the bounce, taking into account the unknown but bounded portion of energy lost during the bounce. Advantage is taken of the knowledge of the signs of the time of bounce and speed. The fact that the ball *may not have* hit the floor yet should not be forgotten. If the upper bound $\overline{x_{\ell+}}$ of $[x_{\ell+}]$ is lower than ρ, then the ball *has definitely* hit the floor. Its position and speed are then evaluated using $[\dot{x}_{ab}]$ and $[T_b]$. Otherwise, it is not known yet whether the ball has hit the floor and the two hypotheses must be taken into account by sending the union of the corresponding solutions, which only requires a slight modification of SIVIA.

APPENDIX B: PROOF OF PROPOSITION 3

Lemma B1

If an inclusion function $\mathbf{f}_{[]}$ is *Lipschitz* in $\mathscr{D} \subset \mathbb{R}^n$, then there exists a constant K such that $w(\mathbf{f}_{[]}([\mathbf{x}])) \leqslant Kw([\mathbf{x}])$ for every $[\mathbf{x}] \subset \mathscr{D}$.

Proof. Consider $[\mathbf{x}] \subset \mathscr{D}$ and $\mathbf{x}_0 \in [\mathbf{x}]$. As $\mathbf{f}_{[]}$ is an inclusion function, $\mathbf{f}_{[]}([\mathbf{x}])$ is a box containing $\mathbf{f}_{[]}(\mathbf{x}_0)$. Then, from Property 1, $d(\mathbf{x}_0, [\mathbf{x}]) \leqslant w([\mathbf{x}])$ and $d(\mathbf{f}_{[]}(\mathbf{x}_0), \mathbf{f}_{[]}([\mathbf{x}])) \geqslant w(\mathbf{f}_{[]}([\mathbf{x}]))/2$. As $\mathbf{f}_{[]}$ is *Lipschitz* in $\mathscr{D}$, there exists λ such that for any $[\mathbf{x}] \subset \mathscr{D}$ and $\mathbf{x}_0 \in [\mathbf{x}]$,

Table A2. Inclusion function for the function of Table A1

```
$\mathbf{f}_{[]}([x_\ell], [\dot{x}_\ell], [\gamma])$

const $g, T, \rho$;
$[x_{\ell+}] = [x_\ell] + [\dot{x}_\ell]T - g\frac{T^2}{2}$;
if $(\underline{x_{\ell+}} \geqslant \rho)$
    return $\begin{pmatrix} [x_{\ell+}] \\ [\dot{x}_\ell] - gT \end{pmatrix}$;
$[\dot{x}_{bb}] = \sqrt{(2g([x_\ell] - \rho) + ([\dot{x}_\ell])^2) \cap [0, +\infty[}$;
$[T_b] = \frac{1}{g}([\dot{x}_\ell] + [\dot{x}_{bb}]) \cap [0, T[$;
$[\dot{x}_{ab}] = \sqrt{1 - [\gamma]}[\dot{x}_{bb}]$;
if $(\overline{x_{\ell+}} \leqslant \rho)$
    return $\begin{pmatrix} \rho + [\dot{x}_{ab}](T - [T_b]) - g\frac{(T - [T_b])^2}{2} \\ [\dot{x}_{ab}] - g(T - [T_b]) \end{pmatrix}$;
else
    return $\begin{pmatrix} [x_{\ell+}] \\ [\dot{x}_\ell] - gT \end{pmatrix} \cup \begin{pmatrix} \rho + [\dot{x}_{ab}](T - [T_b]) - g\frac{(T - [T_b])^2}{2} \\ [\dot{x}_{ab}] - g(T - [T_b]) \end{pmatrix}$.
```

$d(\mathbf{f}_{[]}(\mathbf{x}_0), \mathbf{f}_{[]}([\mathbf{x}])) \leqslant \lambda d(\mathbf{x}_0, [\mathbf{x}])$. The three previous inequalities imply that $w(\mathbf{f}_{[]}([\mathbf{x}])) \leqslant 2\lambda w([\mathbf{x}])$ and K may thus be taken to be equal to 2λ. □

Lemma B2

If $\mathbf{f}_{[]}$ is an inclusion function of $\mathbf{f}: \mathbb{R}^n \to \mathbb{R}^r$ that is Lipschitz in $\hat{\mathscr{X}} \subset \mathbb{R}^n$, then for any $\alpha > 0$, there exists ε such that for any $[\mathbf{x}] \subset \hat{\mathscr{X}}$ with $w([\mathbf{x}]) \leqslant \varepsilon$, $d(\mathbf{f}_{[]}([\mathbf{x}]), \mathbf{f}([\mathbf{x}])) \leqslant \alpha$.

Proof. Since $\mathbf{f}_{[]}$ is Lipschitz in $\hat{\mathscr{X}}$, according to Lemma B1, there exists K such that, for any $[\mathbf{x}] \subset \hat{\mathscr{X}}$, $w(\mathbf{f}_{[]}([\mathbf{x}])) \leqslant Kw([\mathbf{x}])$. Take $\varepsilon = \alpha/K$, and consider $[\mathbf{x}]_0 \subset \hat{\mathscr{X}}$ such that $w([\mathbf{x}]_0) \leqslant \varepsilon$; then $w(\mathbf{f}_{[]}([\mathbf{x}]_0)) \leqslant Kw([\mathbf{x}]_0) \leqslant \alpha$. Since $\mathbf{f}([\mathbf{x}]_0) \subset \mathbf{f}_{[]}([\mathbf{x}]_0)$, Property 2 implies that $d(\mathbf{f}([\mathbf{x}]_0), \mathbf{f}_{[]}([\mathbf{x}]_0)) \leqslant \alpha$. □

Proof of Proposition 3 (*First part*). IMAGESP of Appendix C divides $\hat{\mathscr{X}}$ into P boxes $[\mathbf{x}]_i$ of width less than ε, and stores their images by $\mathbf{f}_{[]}$ in a list of image boxes $\mathscr{L}_\varepsilon^0 = \{\mathbf{f}_{[]}([\mathbf{x}]_j)\}_{j=1}^P$.

Let us prove that for any given $\eta > 0$, there exists ε_1 such that if $\varepsilon < \varepsilon_1$ then $d(\mathbf{f}(\hat{\mathscr{X}}), \bigcup_{j=1}^{P} \mathbf{f}_{[]}([\mathbf{x}]_j)) \leqslant \eta/2$.

The proof is inspired from References [16, 21]. Since $\mathbf{f}_{[]}$ is a convergent inclusion function of $\mathbf{f}$, Lipschitz in $\hat{\mathscr{X}}$, from Lemma B2, for any $\eta > 0$, there exists ε_1 such that $d_0(\mathbf{f}_{[]}([\mathbf{x}]), \mathbf{f}([\mathbf{x}])) \leqslant \eta/2$ for any $[\mathbf{x}] \subset \hat{\mathscr{X}}$ with $w([\mathbf{x}]) \leqslant \varepsilon_1$. Now, $\hat{\mathscr{X}}$ and $\bigcup_{i=1}^{P} [\mathbf{x}]_i$, the union of the P boxes with width less than ε_1 stemming from $\hat{\mathscr{X}}$ represent the same set, so $\mathbf{f}(\hat{\mathscr{X}}) = \bigcup_{i=1}^{P} \mathbf{f}([\mathbf{x}]_i) \subset \bigcup_{j=1}^{P} \mathbf{f}_{[]}([\mathbf{x}]_j)$ as $\mathbf{f}_{[]}$ is an inclusion function of $\mathbf{f}$. Therefore, according to Property 4,

$$d_0\left(\bigcup_{i=1}^{P} \mathbf{f}([\mathbf{x}]_i), \bigcup_{j=1}^{P} \mathbf{f}_{[]}([\mathbf{x}]_j) \right) = 0 \tag{B1}$$

and

$$d_0\left(\mathbf{f}_{[]}([\mathbf{x}]_i), \bigcup_{j=1}^{P} \mathbf{f}([\mathbf{x}]_j) \right) \leqslant d_0(\mathbf{f}_{[]}([\mathbf{x}]_i), \mathbf{f}([\mathbf{x}]_i)) \tag{B2}$$

$$\leqslant \eta/2 \quad \text{for } i = 1, \ldots, P \tag{B3}$$

Thus

$$d_0\left(\bigcup_{i=1}^{P} \mathbf{f}_{[]}([\mathbf{x}]_i), \bigcup_{j=1}^{P} \mathbf{f}([\mathbf{x}]_j) \right) = \max_{i=1,\ldots,P} d_0\left(\mathbf{f}_{[]}([\mathbf{x}]_i), \bigcup_{j=1}^{P} \mathbf{f}([\mathbf{x}]_j) \right) \leqslant \eta/2 \tag{B4}$$

From (B1) and (B4), as $\mathbf{f}(\hat{\mathscr{X}}) = \bigcup_{j=1}^{P} \mathbf{f}([\mathbf{x}]_j)$, it follows that

$$d\left(\mathbf{f}(\hat{\mathscr{X}}), \bigcup_{j=1}^{P} \mathbf{f}_{[]}([\mathbf{x}]_j) \right) \leqslant \eta/2 \tag{B5}$$

Proof of Proposition 3 (Second part). ImageSp of Appendix C builds a subpaving containing the boxes of $\mathscr{L}_{\varepsilon_1}^{0}$ with the procedure BuildSp $(\mathscr{L}_{\varepsilon_1}^{0}, [\mathbf{s}]^0, \varepsilon_2)$. This subpaving is assumed to be included in some prior box $[\mathbf{s}]^0$. (The superscript k in $\mathscr{L}_{\varepsilon_1}^{k}$ and $[\mathbf{s}]^k$ indicates that these quantities are associated with the kth node of the binary tree describing the image subpaving, with $k = 0$ corresponding to the root.) If the subpaving is not equal to $[\mathbf{s}]^0$, $[\mathbf{s}]^0$ is bisected into $L[\mathbf{s}]^0$ and $R[\mathbf{s}]^0$. BuildSp then recursively calls itself on $L\mathscr{L}_{\varepsilon_1}^{0}$ and $R\mathscr{L}_{\varepsilon_1}^{0}$, the lists containing the intersection of the boxes of $\mathscr{L}_{\varepsilon_1}^{0}$ with $L[\mathbf{s}]^0$ and $R[\mathbf{s}]^0$, respectively. The resulting subpaving consists of the union of the two subpavings returned by BuildSp $(L\mathscr{L}_{\varepsilon_1}^{0}, L[\mathbf{s}]^0, \varepsilon_2)$ and BuildSp $(R\mathscr{L}_{\varepsilon_1}^{0}, R[\mathbf{s}]^0, \varepsilon_2)$. We shall now prove that for any given η, there exists ε_2 such that d(BuildSp$(\mathscr{L}_{\varepsilon_1}^{0}, [\mathbf{s}]^0, \varepsilon_2)$, $\bigcup_{i=1}^{P} \mathbf{f}_{[]}([\mathbf{x}]_i)) \leqslant \eta/2$.

The proof is by induction. The property will first be established for non-recursive calls to BuildSp, i.e. when BuildSp returns a leaf of the subpaving. We shall then show that if the property is true for the calls associated with leaves it remains so for all the earlier calls associated with nodes, up to the root of the tree.

To simplify presentation, all the lists $\mathscr{L}_{\varepsilon_1}^{k} = \{[\mathbf{z}]_i^k\}_{i=1}^{P}$ are assumed to have the same number P of boxes, some of which may be empty. A list will be said to be empty if it contains only empty boxes. In the actual implementation of the algorithm, these empty boxes are of course discarded.

Take $\varepsilon_2 = \eta/2$. Consider first a situation where the Nth call to BuildSp has a non-recursive treatment. Its parameters are $\mathscr{L}^N_{\varepsilon_1} = \{[\mathbf{z}]^N_i\}^P_{i=1}$, $[\mathbf{s}]^N$ and ε_2. Three cases have to be considered, which correspond to the three conditional statements in the algorithm BuildSp of Appendix C.

(i) If $\mathscr{L}^N_{\varepsilon_1}$ is empty ($\bigcup^P_{i=1}[\mathbf{z}]^N_i = \emptyset$) then BuildSp$(\mathscr{L}^N_{\varepsilon_1}, [\mathbf{s}]^N, \varepsilon_2)$ returns an empty subpaving and, by convention,

$$d\left(\text{BuildSp}(\mathscr{L}^N_{\varepsilon_1}, [\mathbf{s}]^N, \varepsilon_2), \bigcup_{i=1}^{P} [\mathbf{z}]^N_i\right) = d(\emptyset, \emptyset) = 0 \tag{B6}$$

(ii) If there exists some i such that $[\mathbf{z}]^N_i = [\mathbf{s}]^N$, then $\bigcup^P_{i=1}[\mathbf{z}]^N_i = [\mathbf{s}]^N$, and BuildSp$(\mathscr{L}^N_{\varepsilon_1}, [\mathbf{s}]^N, \varepsilon_2)$ returns the subpaving corresponding to $[\mathbf{s}]^N$. Therefore

$$d\left(\text{BuildSp}(\mathscr{L}^N_{\varepsilon_1}, [\mathbf{s}]^N, \varepsilon_2), \bigcup_{i=1}^{P} [\mathbf{z}]^N_i\right) = 0 \tag{B7}$$

(iii) At last, if $\mathscr{L}^N_{\varepsilon_1}$ does not represent an empty set, but $w([\mathbf{s}]^N) \leqslant \varepsilon_2 = \eta/2$, then BuildSp$(\mathscr{L}^N_{\varepsilon_1}, [\mathbf{s}]^N, \varepsilon_2)$ returns the subpaving corresponding to $[\mathbf{s}]^N$. As $\bigcup^P_{i=1}[\mathbf{z}]^N_i \subset$ BuildSp$(\mathscr{L}^N_{\varepsilon_1}, [\mathbf{s}]^N, \varepsilon_2) = [\mathbf{s}]^N$, and as $w([\mathbf{s}]^N) \leqslant \eta/2$, it follows from Property 2 that

$$d\left(\text{BuildSp}(\mathscr{L}^N_{\varepsilon_1}, [\mathbf{s}]^N, \varepsilon_2), \bigcup_{i=1}^{P} [\mathbf{z}]^N_i\right) \leqslant \eta/2 \tag{B8}$$

In all three cases, $d(\text{BuildSp}(\mathscr{L}^N_{\varepsilon_1}, [\mathbf{s}]^N, \varepsilon_2), \bigcup^P_{i=1}[\mathbf{z}]^N_i) \leqslant \eta/2$. Assume now that during its kth call BuildSp$(\mathscr{L}^k_{\varepsilon_1}, [\mathbf{s}]^k, \varepsilon_2)$ recursively calls itself and that the two resulting subpavings satisfy

$$d\left(\text{BuildSp}(L\mathscr{L}^k_{\varepsilon_1}, L[\mathbf{s}]^k, \varepsilon_2), \bigcup_{i=1}^{P} [\mathbf{z}]^{k,L}_i\right) \leqslant \eta/2 \tag{B9}$$

and

$$d\left(\text{BuildSp}(R\mathscr{L}^k_{\varepsilon_1}, R[\mathbf{s}]^k, \varepsilon_2), \bigcup_{i=1}^{P} [\mathbf{z}]^{k,R}_i\right) \leqslant \eta/2 \tag{B10}$$

where $L[\mathbf{s}]^k \cup R[\mathbf{s}]^k = [\mathbf{s}]^k$, $\mathscr{L}^k_{\varepsilon_1} = \{[\mathbf{z}]^k_i\}^P_{i=1}$, $L\mathscr{L}^k_{\varepsilon_1} = \{[\mathbf{z}]^{k,L}_i\}^P_{i=1} = \{[\mathbf{z}]^k_i \cap L[\mathbf{s}]^k\}^P_{i=1}$ and $R\mathscr{L}^k_{\varepsilon_1} = \{[\mathbf{z}]^{k,R}_i\}^P_{i=1} = \{[\mathbf{z}]^k_i \cap R[\mathbf{s}]^k\}^P_{i=1}$, and where some $[\mathbf{z}]^{k,L}_i$ and $[\mathbf{z}]^{k,R}_i$ may be empty. Let us prove that $d(\text{BuildSp}(\mathscr{L}^k_{\varepsilon_1}, [\mathbf{s}]^k, \varepsilon_2), \bigcup^P_{i=1}[\mathbf{z}]^k_i) \leqslant \eta/2$.

Let $\widehat{\mathscr{S}^k_{\varepsilon_2}} = $ BuildSp$(\mathscr{L}^k_{\varepsilon_1}, [\mathbf{s}]^k, \varepsilon_2)$, $\widehat{\mathscr{S}^{k,L}_{\varepsilon_2}} = $ BuildSp$(L\mathscr{L}^k_{\varepsilon_1}, L[\mathbf{s}]^k, \varepsilon_2)$, and $\widehat{\mathscr{S}^{k,R}_{\varepsilon_2}} = $ BuildSp$(R\mathscr{L}^k_{\varepsilon_1}, R[\mathbf{s}]^k, \varepsilon_2)$. Then $\widehat{\mathscr{S}^k_{\varepsilon_2}} = \widehat{\mathscr{S}^{k,L}_{\varepsilon_2}} \cup \widehat{\mathscr{S}^{k,R}_{\varepsilon_2}}$. As $[\mathbf{z}]^{k,L}_i = [\mathbf{z}]^k_i \cap L[\mathbf{s}]^k$, we have $[\mathbf{z}]^{k,L}_i \subset [\mathbf{z}]^k_i$ for $i = 1, \ldots, P$ and thus $\bigcup^P_{i=1}[\mathbf{z}]^{k,L}_i \subset \bigcup^P_{i=1}[\mathbf{z}]^k_i$. So, from Property 4, $d_0(\widehat{\mathscr{S}^{k,L}_{\varepsilon_2}}, \bigcup^P_{i=1}[\mathbf{z}]^k_i) \leqslant d_0(\widehat{\mathscr{S}^{k,L}_{\varepsilon_2}}, \bigcup^P_{i=1}[\mathbf{z}]^{k,L}_i)$. Now $d_0(\widehat{\mathscr{S}^{k,L}_{\varepsilon_2}}, \bigcup^P_{i=1}[\mathbf{z}]^{k,L}_i) \leqslant \eta/2$, so $d_0(\widehat{\mathscr{S}^{k,L}_{\varepsilon_2}}, \bigcup^P_{i=1}[\mathbf{z}]^k_i) \leqslant \eta/2$ and similarly $d_0(\widehat{\mathscr{S}_{\varepsilon_2}^{k,R}}, \bigcup_{i=1}^{P}[\mathbf{z}]_i^k) \leqslant \eta/2$. Therefore

$$d_0\Big(\widehat{\mathscr{S}_{\varepsilon_2}^{k}}, \bigcup_{i=1}^{P}[\mathbf{z}]_i^k\Big) = \max\Big(d_0\Big(\widehat{\mathscr{S}_{\varepsilon_2}^{k,L}}, \bigcup_{i=1}^{P}[\mathbf{z}]_i^k\Big), d_0\Big(\widehat{\mathscr{S}_{\varepsilon_2}^{k,R}}, \bigcup_{i=1}^{P}[\mathbf{z}]_i^k\Big)\Big) \leqslant \eta/2 \tag{B11}$$

Similarly, $[\mathbf{z}]_i^k \subset [\mathbf{s}]^k$ and $[\mathbf{z}]_i^{k,L} \cup [\mathbf{z}]_i^{k,R} = [\mathbf{z}]_i^k$ for $i = 1, \ldots, P$, imply that $d_0(\bigcup_{i=1}^{P}[\mathbf{z}]_i^{k,L}, \widehat{\mathscr{S}_{\varepsilon_2}^{k}}) \leqslant \eta/2$ and $d_0(\bigcup_{i=1}^{P}[\mathbf{z}]_i^{k,R}, \widehat{\mathscr{S}_{\varepsilon_2}^{k}}) \leqslant \eta/2$, which in turn imply

$$d_0\left(\bigcup_{i=1}^{P}[\mathbf{z}]_i^k, \widehat{\mathscr{S}_{\varepsilon_2}^{k}}\right) = \max\left(d_0\left(\bigcup_{i=1}^{P}[\mathbf{z}]_i^{k,L}, \widehat{\mathscr{S}_{\varepsilon_2}^{k}}\right), d_0\left(\bigcup_{i=1}^{P}[\mathbf{z}]_i^{k,R}, \widehat{\mathscr{S}_{\varepsilon_2}^{k}}\right)\right) \leqslant \eta/2 \tag{B12}$$

From (B11) and (B12), it follows that $d(\bigcup_{i=1}^{P}[\mathbf{z}]_i^k, \widehat{\mathscr{S}_{\varepsilon_2}^{k}}) \leqslant \eta/2$.

It has thus been proved that the property holds true when BuildSp returns a leaf of the subpaving and remains so for all earlier calls, so it holds true for the root of the tree, i.e.

$$d\left(\text{BuildSp}(\mathscr{L}_{\varepsilon_1}^0, [\mathbf{s}]^0, \varepsilon_2), \bigcup_{i=1}^{P}\mathbf{f}_{[]}([\mathbf{x}]_i)\right) \leqslant \eta/2 \tag{B13}$$

Proof of Proposition 3 (End) The results of the first two parts of the proof must now be combined. Let $\varepsilon = \inf(\varepsilon_1, \varepsilon_2)$. From (B5),

$$d\left(\mathbf{f}(\hat{\mathscr{X}}), \bigcup_{j=1}^{P}\mathbf{f}_{[]}([\mathbf{x}]_j)\right) \leqslant \eta/2$$

and from (B13),

$$d\left(\text{BuildSp}(\mathscr{L}_{\varepsilon_1}^0, [\mathbf{s}]^0, \varepsilon), \bigcup_{i=1}^{P}\mathbf{f}_{[]}([\mathbf{x}]_i)\right) \leqslant \eta/2$$

As ImageSp returns $\widehat{\mathscr{S}_\varepsilon} = \text{BuildSp}(\mathscr{L}_\varepsilon^0, [\mathbf{s}]^0, \varepsilon)$, then the triangular inequality implies that

$$d(\mathbf{f}(\hat{\mathscr{X}}), \widehat{\mathscr{S}_\varepsilon}) = d(\mathbf{f}(\hat{\mathscr{X}}), \text{BuildSp}(\mathscr{L}_{\varepsilon_1}^0, [\mathbf{s}]^0, \varepsilon)) \leqslant \eta/2 + \eta/2 = \eta \tag{B14}$$

APPENDIX C: COMPUTER IMPLEMENTATION

The computer implementation of the algorithms presented in this paper is far from trivial and impacts heavily on their performance. This is why we thought it useful to describe its key points in this appendix. All functions and algorithms have been developed in the C-XSC framework [15], and we choose to describe these algorithms using the syntax of this language. The source code is available on request.

C.1. Representation of subpavings

Subpaving are represented by binary trees, using a C+ + class node. Each subpaving is coded as a pointer to a node. This node contains the description of the root box of the subpaving and pointers to nodes representing the roots of the two subtrees stemming from it.

```
class node;
typedef node* spaving;
class node{
  ivector box;                          // box corresponding to the root
  spaving lchild;                         //pointer to the left subtree
  spaving rchild;                        //pointer to the right subtree
};
```

A NULL pointer corresponds to an empty subtree, thus leaves are identified by NULL pointers lchild and rchild. The function _box(spaving) gives access to the box represented by the root of the subpaving; _lchild(spaving) and _rchild(spaving), respectively return the left and right subtree; _spaving([**s**]) creates a subpaving consisting of the box [**s**].

C.2. Manipulation of subpavings

C.2.1. Basic functions. The following functions and operators are used in SIVIA and IMAGESP. To improve readability, the variables and operators are those used in the text, not always at the C++ standard.

- *union operator* (spaving operator$\cup$(spaving $\hat{\mathscr{X}}$, spaving $\hat{\mathscr{Y}}$)): returns the minimal subpaving corresponding to the union of two subpavings $\hat{\mathscr{X}}$ and $\hat{\mathscr{Y}}$.
- *δ-depth expansion function* (spaving expand(spaving $\hat{\mathscr{X}}$, int δ)): creates a non-minimal subpaving whose leaves are all at depth δ. If the original tree $\hat{\mathscr{X}}$ has deeper leaves, then these leaves are removed, else any leaf at depth $i < \delta$, is replaced by a subtree such that all its leaves are at depth $\delta - i$.
- *ε-width expansion function* (spaving expand(spaving $\hat{\mathscr{X}}$, double ε)): creates a non-minimal subpaving whose leaves are all with width less than ε. This function is similar to the *δ-depth expansion function*; they are differentiated by the type of their second argument.
- *box-inclusion test* (bool operator $\subset$ (ivector [**x**], spaving $\hat{\mathscr{X}}$)): checks whether the box [**x**] is included in the subpaving $\hat{\mathscr{X}}$.

As an example, the following function implements the box-inclusion test, assuming that at the first call of the function, the box [**x**] to be checked is included in the box corresponding to the root of $\hat{\mathscr{X}}$.

```
bool operator ⊂ (ivector [x], spaving X̂)
{
  if ((isempty([x]))||isleaf(X̂)) return true;
  else if isempty(X̂) return false;
// Recursive treatment
  bisect(_box(X̂), lcbox, rcbox);
  return (([x]∩lcbox) ⊂ _lchild(X̂))&&(([x]∩rcbox)) ⊂ _rchild(X̂));
}
```

A classical *preorder traversal* algorithm [18] is used; the root is first considered and then the algorithm is recursively called on the left and right children.

If $[\mathbf{x}]$ is empty, then it is included in $\hat{\mathscr{X}}$, or if $\hat{\mathscr{X}}$ is a leaf, then the subpaving consists of a single box, in which $[\mathbf{x}]$ is included, so the test returns true in both cases. If $[\mathbf{x}]$ is not empty but $\hat{\mathscr{X}}$ is empty, $[\mathbf{x}]$ cannot be included in $\hat{\mathscr{X}}$ and the test returns false. In all other cases, if the intersection of $[\mathbf{x}]$ with the box corresponding to the left child of $\hat{\mathscr{X}}$ is included in the left child of $\hat{\mathscr{X}}$ and if the intersection of $[\mathbf{x}]$ with the box corresponding to the right child of $\hat{\mathscr{X}}$ is included in the right child of $\hat{\mathscr{X}}$, then the test is true, else it is false.

C.2.2. Recursive SIVIA. The main procedure of SIVIA is as follows:

```
spaving SIVIA (spaving $\hat{\mathscr{X}}$, function_ptr $\mathbf{h}_{[]}$, spaving $\hat{\mathscr{Y}}$, float $\varepsilon$)
{
  if isempty($\hat{\mathscr{X}}$) return NULL;
  if disjoint($\mathbf{h}_{[]}$(_box($\hat{\mathscr{X}}$)), $\hat{\mathscr{Y}}$) return NULL;
  if ($\mathbf{h}_{[]}$(_box($\hat{\mathscr{X}}$)) $\subset$ $\hat{\mathscr{Y}}$)||($w$(_box($\hat{\mathscr{X}}$)) $<\varepsilon$) return _spaving(_box($\hat{\mathscr{X}}$));
// Recursive treatment
  if isleaf($\hat{\mathscr{X}}$) $\hat{\mathscr{X}}$ = expand($\hat{\mathscr{X}}$, 1);
  return (SIVIA(_lchild($\hat{\mathscr{X}}$), $\mathbf{h}_{[]}$, $\hat{\mathscr{Y}}$, $\varepsilon$) $\cup$ SIVIA (_rchild($\hat{\mathscr{X}}$), $\mathbf{h}_{[]}$, $\hat{\mathscr{Y}}$, $\varepsilon$));
}
```

C.2.3. IMAGESP. The following conceptual algorithm leads to high-complexity subpavings, only needed to allow the procedure to reach any desired precision. In practice, a number of simplifications can be carried out to keep complexity at a manageable level. The price to be paid is a loss in the precision achievable, but the results remain guaranteed.

The main procedure of IMAGESP is as follows:

```
spaving IMAGESP (ivector $[\mathbf{s}]^0$, function_ptr $\mathbf{f}_{[]}$, spaving $\hat{\mathscr{X}}$, float $\varepsilon$)
{
  $\hat{\mathscr{X}}$ = expand($\hat{\mathscr{X}}$, $\varepsilon$);
  list $\mathscr{L}^0$ = fill_list($\mathbf{f}_{[]}$, $\hat{\mathscr{X}}$);
  return(BUILDSP($\mathscr{L}^0$, $[\mathbf{s}]^0$, $\varepsilon$));
}
```

It first modifies the subpaving $\hat{\mathscr{X}}$ to get a non-minimal subpaving consisting only of boxes with width lower than ε. The image of these boxes by $\mathbf{f}_{[]}$ are put into the list $\mathscr{L}^0$ by the basic function fill_list, which will not be presented. The procedure BUILDSP is then called to merge all image subboxes into a subpaving included in some (possibly very large) prior search box $[\mathbf{s}]^0$ that will constitute the root box of this subpaving. The basic idea of BUILDSP is similar to the *Split and Merge* algorithms used in image segmentation [27]. This algorithm is as follows:

```
spaving BUILDSP (list $\mathscr{L}$, ivector $[\mathbf{s}]$, float $\varepsilon$)
{
  if isempty($\mathscr{L}$) return NULL;
  if ($[s] \in \mathscr{L}$) return _spaving($[\mathbf{s}]$);
  if ($w([\mathbf{s}]) < \varepsilon$) return _spaving($[\mathbf{s}]$);
// Recursive treatment
  bisect($[\mathbf{s}]$, $L[\mathbf{s}]$, $R[\mathbf{s}]$);
```

```
    distribute($\mathscr{L}$, $L[\mathbf{s}]$, $R[\mathbf{s}]$, $L\mathscr{L}$, $R\mathscr{L}$);
    return (BuildSp($L\mathscr{L}$, $L[\mathbf{s}]$, $\varepsilon$) $\cup$ BuildSp($R\mathscr{L}$, $R[\mathbf{s}]$, $\varepsilon$));
}
```

If the list $\mathscr{L}$ is empty, then an empty subpaving is returned. Else, BuildSp checks whether the box $[\mathbf{s}]$ corresponds to an element of the list $\mathscr{L}$, which would mean that $[\mathbf{s}]$ belongs to the image subpaving. Else, if $w([\mathbf{s}])$ is less than ε, $[\mathbf{s}]$ also has to be stored in the approximate image subpaving, because the algorithm has failed to prove that $[\mathbf{s}]$ and the image set are disjoint. If none of these two conditions is fulfilled, the prior box $[\mathbf{s}]$ is bisected into $L[\mathbf{s}]$ and $R[\mathbf{s}]$, and the list $\mathscr{L}$ is split by the procedure distribute into two sublists $L\mathscr{L}$, and $R\mathscr{L}$ containing the intersection of the elements of $\mathscr{L}$ with $L[\mathbf{s}]$ and $R[\mathbf{s}]$, respectively. The procedure BuildSp is then recursively called, with arguments corresponding to the two sublists just created. At last, BuildSp returns the subpaving corresponding to the union of those returned by the two recursive calls to BuildSp.

REFERENCES

1. Chernousko FL. *State Estimation for Dynamic Systems*. CRC Press: Boca Raton, FL, 1994.
2. Durieu C, Walter E, Polyak B. Set-membership estimation with the trace criterion made simpler than with the determinant criterion. *CD-Rom of the IFAC Symposium on System Identification*, 2000.
3. Gollamudi S, Nagaraj S, Kapoor S, Huang Y-F. Set-membership state estimation with optimal bounding ellipsoids. *International Symposium on Information Theory and its Applications*, 1996.
4. Kurzhanski A, Valyi I. *Ellipsoidal Calculus for Estimation and Control*. Birkhäuser: Boston, MA, 1997.
5. Maksarov D, Norton JP. State bounding with ellipsoidal set description of the uncertainty. *International Journal of Control* 1996; **65**(5):847–866.
6. Schweppe FC. Recursive state estimation: unknown but bounded errors and system inputs. *IEEE Transactions on Automatic Control* 1968; **13**(1):22–28.
7. Schweppe FC. *Uncertain Dynamic Systems*. Prentice-Hall: Englewood Cliffs, NJ, 1973.
8. Shamma JS, Kuang-Yang Tu. Approximate set-valued observers for nonlinear systems. *IEEE Transactions on Automatic Control*, 1997; **42**(5):648–658.
9. Alcaraz-González V, Genovesi A, Harmand J, González AV, Rapaport A, Steyer JP. Robust exponential nonlinear interval observer for a class of lumped models useful in chemical and biochemical engineering. Application to a wastewater treatment process. *Proceedings of MISC'99 Workshop on Applications of Interval Analysis to Systems and Control*, Girona, February 24–26, 1999; 225–235.
10. Hadj-Sadok MZ, Gouzé J-L, Rapaport A. State observers for uncertain models of activated sludge processes. *CD-Rom of the IFAC-EurAgEng International Workshop on Decision and Control in Waste Bio-Processing*, Narbonne, February 25–27, 1998.
11. Lichtenberg G, Lunze J. Observation of qualitative state by means of qualitative model. *International Journal of Control* 1997; **66**(6):885–903.
12. Chen G, Wang J, Shieh LS. Interval Kalman filtering. *IEEE Transactions on Aerospace and Electronic Systems* 1997; **33**(1):250–258.
13. Kieffer M, Jaulin L, Walter E. Guaranteed recursive nonlinear state estimation using interval analysis. *Proceedings of 37th IEEE Conference on Decision and Control*, Tampa, December 16–18, 1998; 3966–3971.
14. Grossman RL, Nerode A, Ravn AP, Rischel H. (eds). *Hybrid Systems*. Springer: New York, NY, 1993.
15. Hammer R, Hocks M, Kulish U, Ratz D. *C++ Toolbox for Verified Computing*. Springer: Berlin, 1995.
16. Moore RE. *Methods and Applications of Interval Analysis*. SIAM Publ.: Philadelphia, PA, 1979.
17. Berger M. *Geometry I and II*. Springer: Berlin, 1987.
18. Beidler J. *Data Structures and Algorithms*. Springer: New York, NY, 1996.
19. Jaulin L, Walter E. Guaranteed nonlinear parameter estimation from bounded-error data via interval analysis. *Mathematics and Computers in Simulation* 1993; **35**:123–137.
20. Jaulin L, Walter E. Set inversion via interval analysis for nonlinear bounded-error estimation. *Automatica* 1993; **29**(4): 1053–1064.
21. Neumaier A. *Interval Methods for Systems of Equations*. Cambridge University Press: Cambridge, 1990.
22. Kieffer M. Estimation ensembliste par analyse par intervalles, application à la localisation d'un véhicule. *Ph.D. Dissertation*, Université Paris-Sud, Orsay, 1999.

23. Kieffer M, Jaulin L, Walter E, Meizel D. Localisation et suivi robustes d'un robot mobile grâce à l'analyse par intervalles. *Traitement du Signal* 2001; **17**(3):207–219.
24. Davis E. Constraint propagation with interval labels. *Artificial Intelligence* 1987; **32**:281–331.
25. Hyvönen E. Constraint reasoning based on interval arithmetic; the tolerance propagation approach. *Artificial Intelligence* 1992; **58**(1–3):71–112.
26. Jaulin L. Interval constraint propagation with application to bounded-error estimation. *Automatica* 2000; **36**(10):1547–1552.
27. Klinger A. Experiment in picture representation using regular decomposition. *Computer Graphics Image Processing* 1976; **5**:68–105.