Nonlinear bounded-error state estimation of continuous-time systems

Luc Jaulin

Nonlinear bounded-error state estimation of continuous-time systems

Luc Jaulin
Laboratoire d'Ingénierie des Systèmes Automatisés,
62 avenue Notre Dame du Lac 49 000 Angers
jaulin@univ-angers.fr

Abstract: This paper presents a first study on the application of interval analysis and consistency techniques to state estimation of continuous-time systems described by nonlinear ordinary differential equations. The approach is presented in a bounded-error context and the resulting methodology is illustrated on an example.

Keywords: bounded-error estimation, consistency, constraint propagation, CSP, identification, interval analysis, nonlinear state estimation, set estimation.

1 Introduction

The problem to be considered in this paper is the state estimation of a nonlinear continuous-time system in a bounded-error context. The system is assumed to be described by

$$\left\{ \begin{array}{lcl} \dot{\mathbf{x}}(t) & = & \mathbf{f}(\mathbf{x}(t)) \\ \mathbf{y}(t) & = & \mathbf{g}(\mathbf{x}(t)). \end{array} \right. \quad (1)$$

where $t \in [\underline{t}, \overline{t}] \subset \mathbb{R}$ is the time, $\mathbf{x}(t) \in \mathbb{R}^n$ and $\mathbf{y}(t) \in \mathbb{R}^m$ are the state and the output vectors at time t. When the evolution and the observation functions $\mathbf{f}$ and $\mathbf{g}$ are linear, many efficient methods can be found in the literature (see, *e.g.*, [19], [5]). To my knowledge, when $\mathbf{f}$ is nonlinear, state estimation has never been studied in the bounded-error and continuous-time context. This paper proposes a first attempt in that direction. The basic tools to be used are *interval analysis* and *consistency techniques*.

In a bounded-error context, interval analysis [16] has often been used for parameter estimation [17], [12], [3], [15] and state estimation of discrete-time systems [4], [13]. Consistency techniques have been proved to increase the efficiency of interval methods when the number of variables is high (see [6], [8], [18]) and have recently been applied to parameter estimation [9] and state

estimation [11], [10]. Nevertheless, interval and consistency techniques have never been used to estimate state variable of continuous-time systems.

Here, we shall only assume that $\mathbf{f}$ is continuous and differentiable. It is also assumed that for all t, domains $[x_i](t)$ and $[y_j](t)$ containing the variables $x_i(t)$ and $y_j(t)$ are available. These domains can be arbitrarily large, *e.g.*, $[x_i](t) =]-\infty, \infty[$ if no information is available on $x_i(t)$. If a measurement $\hat{y}_j(t_0)$ of $y_j(t_0)$ has been collected at time $t = t_0$ and if an upper bound $\bar{e}$ on the absolute value for the error $\hat{y}_j(t_0) - y_j(t_0)$ is known, then the domain for $y_j(t_0)$ will be $[y_j](t) = [\hat{y}_j(t_0) - \bar{e}, \hat{y}_j(t_0) + \bar{e}]$. For simplicity, these domains will be assumed to be intervals, but union of intervals could also be considered.

Section 2 recalls the basic notions of interval analysis to be used. Section 3 introduces a basic interval method to deal with ordinary differential equation. Section 4 presents an algorithm based for noncausal state estimation. An illustrative example is given in Section 5.

2 Interval computation

An *interval* $[x]$ is a closed and connected subset of $\mathbb{R}$. A *box* $[\mathbf{x}]$ of $\mathbb{R}^n$ is a Cartesian product of n intervals. The set of all boxes of $\mathbb{R}^n$ is denoted by $\mathbb{IR}^n$. Note that $\mathbb{R}^n =]-\infty, \infty[\times \cdots \times]-\infty, \infty[$ is an element of $\mathbb{IR}^n$. Basic operations on real numbers or vectors can be extended to intervals in a natural way.

Example 1 *If* $[t] = [\underline{t}, \overline{t}]$ *is an interval and* $[\mathbf{x}] = [\underline{x}_1, \overline{x}_1] \times [\underline{x}_2, \overline{x}_2]$ *is a box, then the product* $[t] * [\mathbf{x}]$ *is defined as follows*

$$[\underline{t}, \overline{t}] * \begin{pmatrix} [\underline{x}_1, \overline{x}_1] \\ [\underline{x}_2, \overline{x}_2] \end{pmatrix} = \begin{pmatrix} [\underline{t}, \overline{t}] * [\underline{x}_1, \overline{x}_1] \\ [\underline{t}, \overline{t}] * [\underline{x}_2, \overline{x}_2] \end{pmatrix} = \begin{pmatrix} [\min(\underline{t}\underline{x}_1, \underline{t}\overline{x}_1, \overline{t}\underline{x}_1, \overline{t}\overline{x}_1), \max(\underline{t}\underline{x}_1, \underline{t}\overline{x}_1, \overline{t}\underline{x}_1, \overline{t}\overline{x}_1)] \\ [\min(\underline{t}\underline{x}_2, \underline{t}\overline{x}_2, \overline{t}\underline{x}_2, \overline{t}\overline{x}_2), \max(\underline{t}\underline{x}_2, \underline{t}\overline{x}_2, \overline{t}\underline{x}_2, \overline{t}\overline{x}_2)] \end{pmatrix}.$$

The function $[\mathbf{f}](.) : \mathbb{IR}^n \to \mathbb{IR}^p$ is an *inclusion function* of a function $\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^p$ if

$$\forall [\mathbf{x}] \in \mathbb{IR}^n, \mathbf{f}([\mathbf{x}]) \triangleq \{\mathbf{f}(\mathbf{x}) \mid \mathbf{x} \in [\mathbf{x}]\} \quad \subset \quad [\mathbf{f}]([\mathbf{x}]).$$

Interval computation makes it possible to obtain inclusion functions of a large class of nonlinear functions, as illustrated by the following example.

Example 2 : *If* $\mathbf{f}(x_1, x_2) \triangleq ((1 - 0.01x_2)\, x_1\ ;\ (-1 + 0.02x_1)\, x_2)$, *a methodology to obtain an enclosure of the image set* $\mathbf{f}([10, 20], [40, 50])$ *is as follows:*

$$\mathbf{f}\begin{pmatrix} [40,50] \\ [10,20] \end{pmatrix} \subset \begin{pmatrix} (1 - 0.01 * [40,50]) * [10,20] \\ (-1 + 0.02 * [10,20]) * [40,50] \end{pmatrix} = \begin{pmatrix} (1 - [0.4,0.5]) * [10,20] \\ (-1 + [0.2,0.4]) * [40,50] \end{pmatrix}$$
$$= \begin{pmatrix} [0.5,0.6] * [10,20] \\ [-0.8,-0.6] * [40,50] \end{pmatrix} = \begin{pmatrix} ([5,12]) \\ ([-40,-24]) \end{pmatrix}.$$

This methodology can easily be applied for any box $[x_1] \times [x_2]$ *and the resulting algorithm corresponds to an inclusion function for* $\mathbf{f}$. ■

The *interval union* $[\mathbf{x}] \sqcup [\mathbf{y}]$ of two boxes $[\mathbf{x}]$ and $[\mathbf{y}]$ is the smallest box which contains the union $[\mathbf{x}] \cup [\mathbf{y}]$. The *width* $w([\mathbf{x}])$ of a box $[\mathbf{x}]$ is the length of its largest side. The ε-*inflation* of a box $[\mathbf{x}] = [\underline{x}_1, \overline{x}_1] \times \cdots \times [\underline{x}_n, \overline{x}_n]$ is defined by

$$\text{inflate}\,([\mathbf{x}], \varepsilon) \triangleq [\underline{x}_1 - \varepsilon, \overline{x}_1 + \varepsilon] \times \cdots \times [\underline{x}_n - \varepsilon, \overline{x}_n + \varepsilon]. \tag{2}$$

An enclosure of the set $[\mathbf{x}] \cap \mathbf{g}^{-1}([\mathbf{y}])$ where $\mathbf{g} : \mathbb{R}^n \to \mathbb{R}^p$, $[\mathbf{x}] \in \mathbb{IR}^n$ and $[\mathbf{y}] \in \mathbb{IR}^p$ can be obtained by the forward-backward propagation algorithm [1], [10]. Its principle, based on interval computation, is illustrated by the following example.

Example 3 *If* $g(x_1, x_2) = x_1 * x_2 + \exp(x_1)$, *an enclosure of* $[\mathbf{x}] \cap g^{-1}([y])$ *is obtained by the following algorithm.*

Algorithm: ENCLOSE (*in:* $[y]$, *inout:* $[\mathbf{x}]$)

1 $[z_1] := [x_1] * [x_2]$;
2 $[z_2] := \exp([x_1])$;
3 $[y] := ([z_1] + [z_2]) \cap [y]$;
3' $[z_1] := ([y] - [z_2]) \cap [z_1]$; $[z_2] := ([y] - [z_1]) \cap [z_2]$;
2' $[x_1] := \log([z_2]) \cap [x_1]$;
1' $[x_1] := ([z_1] \,/\, [x_2]) \cap [x_1]$; $[x_2] := ([z_1] \,/\, [x_1]) \cap [x_2]$.

The z_i*'s are used to decompose* g *into binary or ternary statements. Steps 1, 2 and 3, corresponding to the* forward propagation, *compute an interval evaluation of* $g([\mathbf{x}])$. *Steps 3', 2' and 1' correspond to the backward propagation of 1, 2 and 3, respectively. Backward statements*

are obtained by rewriting the constraints of the forward statements in a different form and in the reverse order. For instance, Step 3 corresponds to the constraint $y = z_1 + z_2$ *which can be rewritten* $z_1 = y - z_2$ *or* $z_2 = y - z_1$ *to make Step 3'.* ■

3 Picard theorem

Interval analysis for ordinary differential equations were introduced by Moore [16] (See [2] for a description and a bibliography on this topic). These methods provide numerically reliable enclosures of the exact solution at sample times $t_0, t_1, \ldots$ These techniques are based on Picard Theorem.

Theorem 1 *Let* t_1 *and* t_2 *be two real numbers. Assume that* $\mathbf{x}(t_1)$ *is known to belong to the box* $[\mathbf{x}](t_1)$*. Let* $[\mathbf{w}]$ *be a box (that is expected to enclose the trajectory* $\mathbf{x}(\tau)$, $\tau \in [t_1, t_2]$*). If*

$$[\mathbf{x}](t_1) + [0, t_2 - t_1] * [\mathbf{f}]([\mathbf{w}]) \subset [\mathbf{w}], \tag{3}$$

where $[\mathbf{f}](.)$ *is an inclusion function of* $\mathbf{f}$ *and where* $[0, t_2 - t_1]$ *is the smallest interval which contains* 0 *and* $t_2 - t_1$*, then*

$$\begin{array}{l} \textit{(i)}\ \forall \tau \in [t_1, t_2],\ \mathbf{x}(\tau) \in [\mathbf{x}](t_1) + [0, t_2 - t_1] * [\mathbf{f}]([\mathbf{w}]), \\ \textit{(ii)}\ \mathbf{x}(t_2) \in [\mathbf{x}](t_1) + (t_2 - t_1) * [\mathbf{f}]([\mathbf{w}]). \end{array} \tag{4}$$

From this theorem, one can build an algorithm computing an enclosure $[\mathbf{x}](t_2)$ for $\mathbf{x}(t_2)$ from an enclosure $[\mathbf{x}](t_1)$ for $\mathbf{x}(t_1)$.

Algorithm $[\phi]$ (in: $t_1, t_2, [\mathbf{x}](t_1)$, out $[\mathbf{x}](t_2)$)

1 $[\hat{\mathbf{x}}](t_2) := [\mathbf{x}](t_1) + (t_2 - t_1) * [\mathbf{f}]([\mathbf{x}](t_1)))$;
2 $[\mathbf{v}] := [\mathbf{x}](t_1) \sqcup [\hat{\mathbf{x}}](t_2)$;
3 $[\mathbf{w}] :=$ inflate($[\mathbf{v}]$,$\alpha.w([\mathbf{v}]) + \beta$).
4 if $[\mathbf{x}](t_1) + [0, t_2 - t_1] * [\mathbf{f}]([\mathbf{w}])) \not\subseteq [\mathbf{w}]$ $\{[\mathbf{x}](t_2) := \mathbb{R}^n$; return$\}$;
5 $[\mathbf{x}](t_2) := [\mathbf{x}](t_1) + (t_2 - t_1) * [\mathbf{f}]([\mathbf{w}])$.

Comments: Step 1 computes an estimation $[\hat{\mathbf{x}}](t_2)$ for the domain of all $\mathbf{x}(t_2)$ consistent with the fact that $\mathbf{x}(t_1) \in [\mathbf{x}](t_1)$. Note that, it is not certain that $[\hat{\mathbf{x}}](t_2)$ contains $\mathbf{x}(t_2)$. Step 2

computes the smallest box $[\mathbf{v}]$ containing $[\mathbf{x}](t_1)$ and $[\hat{\mathbf{x}}](t_2)$. At Step 3, $[\mathbf{v}]$ is inflated (see (2)) to provide a good candidate for $[\mathbf{w}]$. α and β are small positive numbers (in the examples treated in paper they are chosen as $\alpha = 0.1$ and $\beta = 0.0001$). Step 4 checks the condition of Theorem 1. If the condition is not satisfied, no bounds can be computed for $\mathbf{x}(t_2)$ and $\mathbb{R}^n$ is returned. Otherwise, Step 5 computes a box containing $\mathbf{x}(t_2)$ using (4). Note that, if $|t_2 - t_1|$ is small enough, the Picard condition (1) holds true. ■

A direct consequence of the Picard theorem is the following implication

$$\mathbf{x}(t_1) \in [\mathbf{x}](t_1) \quad \Rightarrow \quad \mathbf{x}(t_2) \in [\boldsymbol{\phi}](t_1, t_2, [\mathbf{x}](t_1)). \tag{5}$$

The operator $[\boldsymbol{\phi}]$ can be used to compute guaranteed enclosures of the state vector at times $\delta, 2\delta, \ldots, \bar{k}\delta$, where δ is the sampling time and $\overline{k}$ is the largest integer smaller than $\overline{t}/\delta$, from a given box $[\mathbf{x}](0)$ containing $\mathbf{x}(0)$. This is performed by the following algorithm PICARD, where $[\mathbf{x}](k)$ is a short notation for $[\mathbf{x}](k\delta)$.

Algorithm PICARD(in: $t_1, t_2, [\mathbf{x}](0)$, out: $[\mathbf{x}](1), \ldots, [\mathbf{x}](\bar{k})$)

1 for $k := 0$ to $\bar{k} - 1$

2 $\quad [\mathbf{x}](k+1) := [\boldsymbol{\phi}](k\delta, (k+1)\delta, [\mathbf{x}](k))$

This algorithm is known to provide very rough enclosures of the $\mathbf{x}(k)$'s as illustrated by the following example.

Example 4 *Consider the Lotka-Volterra predator-prey model given by the following equations*

$$\begin{cases} \dot{x}_1 &= (1 - 0.01x_2)\, x_1 \\ \dot{x}_2 &= (-1 + 0.02x_1)\, x_2 \end{cases} \tag{6}$$

For $[\mathbf{x}](0) = [49.5; 50.5] \times [49.5; 50.5], \delta = 0.005$ *and* $\bar{k} = 400$, PICARD *generates a sequence of boxes* $[\mathbf{x}](k)$*, the superposition of which is depicted in Figure 1. For all* $k \in \{1, \ldots, \bar{k}\}$*, we are certain to have* $\{\mathbf{x}(k) \mid \mathbf{x}(0) \in [\mathbf{x}](0)\} \subset [\mathbf{x}](k)$*. The* $\mathbf{x}(k)$*'s in white in the picture are obtained for* $\mathbf{x}(0) = (50, 50)$*.*

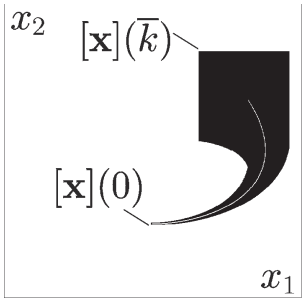

Figure 1: Set of all boxes generated by PICARD; the frame box is $[0, 100] \times [0, 200]$

Example 4 illustrates the major problem of interval methods in ordinary differential equations: the explosion of the size of the $[\mathbf{x}](k)$'s. There are mainly two reasons for this explosion. On the one hand, step methods have a tendency to accumulate errors from point to point. On the other, the approximation of an arbitrary region by a box, called the *wrapping effect*, may introduce considerable imprecision after number of steps. This explosion can be reduced by using the Lohner method [14].

4 Consistency algorithms for state estimation

Consider again the system (1), where domains $[x_i](t)$ and $[y_j](t)$ for the variables $x_i(t)$ and $y_j(t)$ are available. Recall that these domains may be equal to $]-\infty, \infty[$ is no information is available on the associated variable. At sampling times $t = k\delta$, the variables $x_i(t)$ and $y_j(t)$ will be denoted by $x_i(k)$ and $y_j(k)$. The basic idea of consistency techniques is to contract the domains for the variables by pruning parts that are inconsistent with (1) and the domains for the other variables. Using consistency for ordinary differential equation has first been presented in [7]. The contractions to be considered are based on the following implications.

$$
\begin{array}{lllll}
\text{(i)} & \mathbf{x}(k) \in [\mathbf{x}](k) & \Longrightarrow & \mathbf{x}(k+1) \in [\boldsymbol{\phi}](k\delta, (k+1)\delta, [\mathbf{x}](k)) & \text{(see (5))} \\
\text{(ii)} & \mathbf{x}(k) \in [\mathbf{x}](k) & \Longrightarrow & \mathbf{y}(k) \in \mathbf{g}([\mathbf{x}](k)) & \\
\text{(iii)} & \mathbf{y}(k) \in [\mathbf{y}](k) & \Longrightarrow & \mathbf{x}(k) \in \mathbf{g}^{-1}([\mathbf{x}](k)) & \\
\text{(iv)} & \mathbf{x}(k+1) \in [\mathbf{x}](k+1) & \Longrightarrow & \mathbf{x}(k) \in [\boldsymbol{\phi}]((k+1)\delta, k\delta, [\mathbf{x}](k+1)) & \text{(see (5))}
\end{array}
$$

Each of these implications corresponds to a contracting statement of the following algorithm. Note that no consistent values can be lost.

Algorithm CONTRACT(inout: $[\mathbf{x}](0), \ldots, [\mathbf{x}](\bar{k}), [\mathbf{y}](0), \ldots, [\mathbf{y}](\bar{k})$)

1 for $k := 0$ to $\bar{k}$

2 $\quad [\mathbf{x}](k+1) := [\mathbf{x}](k+1) \cap [\boldsymbol{\phi}](k\delta, (k+1)\delta, [\mathbf{x}](k))$; // see (i)

3 $\quad [\mathbf{y}](k) := [\mathbf{y}](k) \cap [\mathbf{g}]([\mathbf{x}](k))$; // see (ii)

4 endfor

5 for $k := \bar{k}$ to 0

6 $\quad [\mathbf{x}](k) := [\mathbf{x}](k) \cap [\mathbf{g}^{-1}]([\mathbf{y}](k))$; // see (iii)

7 $\quad [\mathbf{x}](k) := [\mathbf{x}](k) \cap [\boldsymbol{\phi}]((k+1)\delta, k\delta, [\mathbf{x}](k+1))$; // see (iv)

8 endfor

At Step 3, $[\mathbf{g}](.)$ is an interval extension of $\mathbf{g}(.)$. The contraction of $[\mathbf{x}](k)$ at Step 6 is performed as in Example 3. The procedure can be called several times until no more significant can be observed. When the fixed point is reached, domains may still contains large parts of inconsistent values. Bisections should thus be performed to eliminate them. The strategy to be followed is to partition $[\mathbf{x}](k_0)$, for a given k_0, with smaller boxes in order to eliminate inconsistent values in all the other domains. The resulting algorithm is described below. ε is a small positive number.

.

Algorithm STRANGLE(in: k_0, ε, inout: $[\mathbf{x}](0), \ldots, [\mathbf{x}](\bar{k}), [\mathbf{y}](0), \ldots, [\mathbf{y}](\bar{k})$)

1 partition $[\mathbf{x}](k_0)$ into ℓ boxes $[\mathbf{x}_1](k_0), \ldots, [\mathbf{x}_\ell](k_0)$ of width smaller than ε;

2 for $i := 1$ to ℓ,

3 $\quad$ for $k := 1$ to $\bar{k}$ do $\{[\mathbf{x}_i](k) := [\mathbf{x}](k);\ [\mathbf{y}_i](k) := [\mathbf{y}](k)\}$; endfor k

4 $\quad$ while a significant improvement is obtained do

5 $\quad\quad$ CONTRACT$\big([\mathbf{x}_i](0), \ldots, [\mathbf{x}_i](\bar{k}), [\mathbf{y}_i](0), \ldots, [\mathbf{y}_i](\bar{k})\big)$;

6 endfor i

7 for $k := 0$ to $\bar{k}$

8 $\quad [\mathbf{x}](k) := [\mathbf{x}_1](k) \sqcup \cdots \sqcup [\mathbf{x}_\ell](k)$;

9 $\quad [\mathbf{y}](k) := [\mathbf{y}_1](k) \sqcup \cdots \sqcup [\mathbf{y}_\ell](k)$;

10 endfor k

Note that even for infinitely small ε, the domains $[\mathbf{x}](k)$ and $[\mathbf{y}](k)$ may contain large part of inconsistent values. The principle of the algorithm STATEBOUND to be now given is to

call STRANGLE for different k_0 and for decreasing values of ε in order to contract efficiently the domains. Again, even when the fixed point is reached, the domains may still contain inconsistent value, but no consistent values has been lost.

:

Algorithm STATEBOUND(in: ε, inout: $[\mathbf{x}](0), \ldots, [\mathbf{x}](\bar{k}), [\mathbf{y}](0), \ldots, [\mathbf{y}](\bar{k})$)

1 do
2 do
3 chose a random integer k_0 in $\{0, 1, \ldots, \bar{k}\}$;
4 STRANGLE$(k_0, \varepsilon, [\mathbf{x}](0), \ldots, [\mathbf{x}](\bar{k}), [\mathbf{y}](0), \ldots, [\mathbf{y}](\bar{k}))$;
5 while the improvement is significant;
6 decrease ε;
7 while the improvement is significant.

Remark 1 *For simplicity, in the example of Section 5, the external* do-while *loop is replaced by the loop "for $n_1 := 1$ to $\bar{n}_1$", where $\bar{n}_1$ is chosen by the user. The internal* do-while *loop is replaced by the loop "for $n_2 := 1$ to 30". The statement "decrease ε" is replaced by $\varepsilon := \varepsilon/5$.* ■

5 Example

Consider the system (1) where the evolution equation is given by (6) and the observation equation is $y = x_1$. The data $\hat{y}(t)$ have been obtained by simulating the system with $\mathbf{x}(0) = (50, 50)$ and by adding to the noise-free output $y(t)$ a random white noise with a uniform distribution inside the interval $[-1, -1]$. The domains $[y](t)$ for $y(t)$ are taken as $[y](t) = [\hat{y}(t) - 1.5, \hat{y}(t) + 1.5]$, to take into account that accurate bounds for the feasible errors is rarely available. The available domains for the $\mathbf{x}(t)$'s are all taken equal to $[0, 100] \times [0, 200]$. For $\delta = 0.03$; and $\overline{k} = 200$, STRANGLE$(k_0 = 0, \varepsilon = 4)$ generates the sequence $[\mathbf{x}](k)$ depicted in Figure 2 (a), in 1.33 sec. on a Pentium 300. Note that for small k's, the domains $[\mathbf{x}](k)$ are much more accurate than for large k's. Let us now call STRANGLE$(k_0 = \bar{k}, \varepsilon = 4)$. For large k's the domains have now been considerably contracted (see Figure 2 (b)). For $\bar{n}_1 = 1$ (*i.e.*, the external loop is run once), STATEBOUND$(\varepsilon = 4)$ generates the domains of Figure 2 (c) in 11.4 sec.

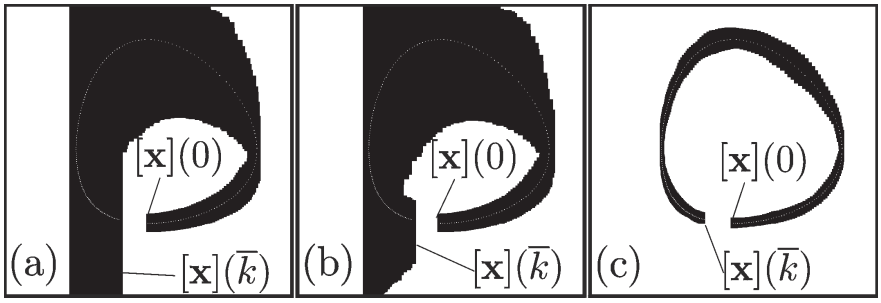

Figure 2: Superposition of all domains $[\mathbf{x}](k)$ contracted by the algorithm STRANGLE; measurements for $y(k)$ are available

Assume now that the measurements are no more available, *i.e.*, all domains for the $y(t)$'s are taken as $]-\infty,\infty[$. All domains for $x(t)$ are taken as $[0,100]\times[0,200]$. The only variable to be approximately known is $\mathbf{x}_0$, the domain of which is taken as $[\mathbf{x}](0)=[49.5,50.5]\times[49.5,50.5]$. For $\delta=0.005$, $\overline{k}=400$, $\bar{n}_1=1$, STATEBOUND($\varepsilon=50$) generates the domains $[\mathbf{x}](k)$ depicted on Figure 3 (a). For $\bar{n}_1=2$, it generates the domains of Figure 3 (b) and for $\bar{n}_1=3$, those of Figure 3 (c). These results are more accurate than those of Figure 1 obtained by the PICARD algorithm.

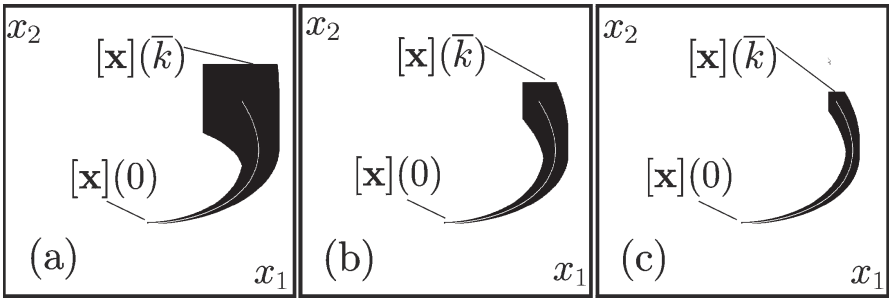

Figure 3 : Superposition of the domains $[\mathbf{x}](k)$ contracted by STATEBOUND; no measurement is available and the initial state vector is approximately known

6 Conclusion

For the first time, this paper studies the application of interval analysis to state estimation of nonlinear continuous-time systems. In a bounded-error and in a noncausal context, a new algorithm to enclose efficiently all consistent values for the state vector inside a box has been presented. This algorithm could be used to build a recursive and causal state estimator. Its

efficiency could be improved by using more efficient consistency techniques [7] and more accurate interval simulation procedures [14].

The source code in C++ (which takes less that 100 lines) corresponding to the example is available on request.

References

[1] F. Benhamou, F. Goualard, L. Granvilliers, and J. F. Puget. Revising hull and box consistency. In *International Conference on Logic Programming*, 1999.

[2] M. Berz, C. Bischof, G. Corliss, and G. A., editors. *Computational Differentiation: Techniques, Applications and Tools*. SIAM, Philadelphia, Penn., 1996.

[3] M. Candev. *Scientific Computation and Validated Numerics*, chapter On the Application of an Interval Algorithm for Set Inversion, pages 140–146. Akademie Verlag, 1996.

[4] G. Chen, J. Wang, and L. S. Shieh. Interval Kalman filtering. *IEEE Trans. on Aerospace and Electronic Systems*, 33(1):250–258, 1997.

[5] F. L. Chernousko. *State Estimation for Dynamic Systems*. CRC Press, Boca Raton, 1994.

[6] E. Davis. Constraint propagation with interval labels. *Artificial Intelligence*, 32:281–331, 1987.

[7] Y. Deville, M. Janssen, and P. V. Hentenryck. Consistency techniques in ordinary differential equations. In *Proceedings of the Fourth International Conference on Principles and Practice of Constraint Programming*, Lecture Notes in Computer Science. Springer Verlag, 1998.

[8] E. Hyvönen. Constraint reasoning based on interval arithmetic; the tolerance propagation approach. *Artificial Intelligence*, 58:71–112, 1992.

[9] L. Jaulin. Interval constraint propagation with application to bounded-error estimation. *Automatica*, 36:1547–1552, 2000.

[10] L. Jaulin, I. Braems, M. Kieffer, and E. Walter. Nonlinear state estimation using forward-backward propagation of intervals. In *SCAN 2000*, 2000.

[11] L. Jaulin, M. Kieffer, I. Braems, and E. Walter. Guaranteed nonlinear estimation using constraint propagation on sets. *International Journal of Control (accepted for publication)*, 2000.

[12] L. Jaulin and E. Walter. Set inversion via interval analysis for nonlinear bounded-error estimation. *Automatica*, 29(4):1053–1064, 1993.

[13] M. Kieffer, L. Jaulin, E. Walter, and D. Meizel. Robust autonomous robot localization using interval analysis. *Reliable Computing*, 6:337–362, 2000.

[14] R. Lohner. *Computer Arithmetic: Scientific Computation and Programming Languages*, chapter Enclosing the solutions of ordinary initial and boundary value problems, pages 255–286. E. Kaucher and U. Kulisch and Ch. Ullrich, 1987.

[15] S. A. Malan, M. Milanese, and M. Taragna. Robust analysis and design of control systems using interval arithmetics. In *Proceedings of the IFAC 13th Triennial World Congress*, pages 25–30, San Francisco, 1996.

[16] R. E. Moore. *Interval Analysis.* Prentice-Hall, Englewood Cliffs, New Jersey, 1966.

[17] R. E. Moore. Parameter sets for bounded-error data. *Mathematics and Computers in Simulation*, 34:113–119, 1992.

[18] D. Sam-Haroud and B. Faltings. Consistency techniques for continuous constraints. *Constraints*, 1:85–118, 1996.

[19] F. C. Schweppe. Recursive state estimation: unknown but bounded errors and system inputs. *IEEE Trans. on Automatic Control*, 13(1):22–28, 1968.