Nonlinear state estimation using forward-backward propagation of intervals in an algorithm

Luc Jaulin, Isabelle Braems, Michel Kieffer, Eric Walter

Nonlinear state estimation using forward-backward propagation of intervals in an algorithm

L. Jaulin [1] [2], I. Braems[1] , M. Kieffer[1] and E. Walter[1]

Abstract: The paper deals with the estimation of the state vector of a discrete-time model from interval output data. When the model outputs are affine in the initial state vector, a number of methods are available to enclose all estimates that are consistent with the data by simple sets such as ellipsoids, orthotopes or parallelotopes, thereby providing guaranteed set estimates. In the nonlinear case, the situation is much less developed and there are very few methods that produce such guaranteed estimates. In this paper, the state estimation of a discrete-time model is performed by combining a set-inversion algorithm with a forward-backward propagation of intervals through the model. The resulting methodology is illustrated on an example.

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

1 Introduction

This paper presents a new approach for the guaranteed estimation of the state vector of a nonlinear discrete-time model in a bounded-error context. Consider a nonlinear discrete-time system described by

$$\left\{\begin{array}{lcll} x_1(k) & = & f_1(x_1(k-1),\ldots,x_{n_{\rm x}}(k-1),k) & \\ \vdots & \vdots & \vdots & \\ x_{n_{\rm x}}(k) & = & f_{n_{\rm x}}(x_1(k-1),\ldots,x_{n_{\rm x}}(k-1),k) & \\ y_1(k) & = & g_1(x_1(k),\ldots,x_{n_{\rm x}}(k),k) & k=1,\ldots,\bar{k}, \\ \vdots & \vdots & \vdots & \\ y_{n_{\rm y}}(k) & = & g_{n_{\rm y}}(x_1(k),\ldots,x_{n_{\rm x}}(k),k) & \end{array}\right. \qquad (1)$$

where k is the time index, $x_1(k),\ldots,x_{n_{\rm x}}(k)$ are the state variables, $y_1(k),\ldots,y_{n_{\rm y}}(k)$ the outputs and the f_i's and the g_i's are known function given by the model. In a vector form, (1) can also

[1]Laboratoire des Signaux et Systèmes, UMR 8506, CNRS-Supélec-Université de Paris Sud, 91192 Gif-sur-Yvette, France

[2]on leave from Laboratoire d'Ingénierie des Systèmes Automatisés, Université d'Angers, France.

be written by:

$$\left\{ \begin{array}{rcl} \mathbf{x}(k) & = & \mathbf{f}(\mathbf{x}(k-1),k) \\ \mathbf{y}(k) & = & \mathbf{g}(\mathbf{x}(k),k). \end{array} \right. \qquad k=1,\ldots,\bar{k}. \tag{2}$$

The set of all variables involved in this problem is

$$\begin{array}{ll} \{x_1(0),\ldots,x_{n_{\rm x}}(0), & \\ x_1(1),\ldots,x_{n_{\rm x}}(1), & y_1(1),\ldots,y_{n_{\rm y}}(1) \\ \vdots & \vdots \\ x_1(\bar{k}),\ldots,x_{n_{\rm x}}(\bar{k}), & y_1(\bar{k}),\ldots,y_{n_{\rm y}}(\bar{k})\}. \end{array} \tag{3}$$

In a bounded-error context, it is generally assumed that the following variables,

$$x_1(0),\ldots,x_{n_{\rm x}}(0),y_1(1),\ldots,y_{n_{\rm y}}(1),\ldots,y_1(\bar{k}),\ldots,y_{n_{\rm y}}(\bar{k}), \tag{4}$$

belong to known prior intervals, denoted by

$$[x_1](0),\ldots,[x_{n_{\rm x}}](0),[y_1](1),\ldots,\left[y_{n_{\rm y}}\right](1),\ldots,[y_1](\bar{k}),\ldots,\left[y_{n_{\rm y}}\right](\bar{k}), \tag{5}$$

respectively. The interval $[x_i](0)$ represents the prior knowledge on the initial state variable $x_i(0)$ and may be arbitrarily large. The interval $[y_i](k)$ represents the ith interval measurement at time k. $[y_i](k)$ is taken as $]-\infty,\infty[$ if, for any reason, the ith measurement at time k could not be collected. The problem of interest is to find intervals (if possible the smallest) enclosing all variables that are consistent with the equations (1) and the domains (5). If $\mathbf{f}$ and $\mathbf{g}$ are linear in $\mathbf{x}$, many methods have been proposed for this purpose [14], [12], [13], [10]. Recently, interval analysis has been introduced to deal with this problem in a guaranteed way when $\mathbf{f}$ and $\mathbf{g}$ are nonlinear in $\mathbf{x}$ [4], [8], [9] and [6]. In this paper, the efficiency of the interval methods is improved by combining them with constraint propagation for the first time in the context of nonlinear state estimation. Note that the combination of constraint propagation with interval analysis have been independently proposed by [3] and [2].

2 Contractors

First, let us remark that the variables $x_1(0),\ldots,x_{n_{\rm x}}(0)$ have a special status: if they are known, all other variables of the problem can be deduced by simulating (1). This section deals with the following much more general problem:

Problem 1: Given a function $\mathbf{f}$ with $n_{\rm p}$ input variables $\mathbf{p}=(p_1,p_2,\ldots,p_{n_{\rm p}})^{\rm T}$ and $n_{\rm y}$ output variables $\mathbf{y}=(y_1,y_2,\ldots,y_{n_{\rm y}})^{\rm T}$, given the box domains $[\mathbf{p}]=[p_1]\times\cdots\times[p_{n_{\rm p}}]$ and $[\mathbf{y}]=$

$[y_1] \times \ldots \times [y_{n_y}]$, contract the $[p_i]$'s and the $[y_i]$'s by removing values in the domains that are inconsistent with the other domains. ■

The problem will be denoted in a short form by

$$\mathcal{H}: \ \mathbf{y} = \mathbf{f}(\mathbf{p}), \mathbf{p} \in [\mathbf{p}], \mathbf{y} \in [\mathbf{y}]. \tag{6}$$

Recall that a value for a given variable is *consistent* with $\mathcal{H}$, if it is possible to instantiate the other variables in their domains such that the relation $\mathbf{y} = \mathbf{f}(\mathbf{p})$ is satisfied. If all inconsistent values for the variables have been removed, the contraction will be called *optimal.* The function $\mathbf{f}$ is assumed to be given by an algorithm and will be called the *simulator*. In our state-estimation problem, $\mathbf{p}$ plays the role of the initial state vector and $\mathbf{y}$ of the set of all measurements:

$$\begin{array}{rcl} \mathbf{p} & \leftrightarrow & x_1(0), \ldots, x_{n_x}(0) \\ \mathbf{y} & \leftrightarrow & y_1(1), \ldots, y_{n_y}(1), \ldots, y_1(\bar{k}), \ldots, y_{n_y}(\bar{k}) \end{array} \tag{7}$$

The simulator $\mathbf{f}$ represents the algorithm which computes all outputs from the initial state vector. It is given by

Algorithm: $\mathbf{f}$ (8)

```
input:    x_1(0), ..., x_{n_x}(0);
1         for k := 1 to k̄
2             for i := 1 to n_x
3                 x_i(k) := f_i(x_1(k-1), ..., x_{n_x}(k-1), k);
4             endfor
5             for j := 1 to n_y
6                 y_j(k) := g_j(x_1(k), ..., x_{n_x}(k), k);
7             endfor
8         endfor
output:   y_1(1), ..., y_{n_y}(1), ..., y_1(k̄), ..., y_{n_y}(k̄)
```

(9)

This section proposes a methodology, based on interval constraint propagation, to build an algorithm, namely the *contractor*, which performs the contractions of the domains $[p_1], \ldots, [p_{n_p}]$ and $[y_1], \ldots, [y_{n_y}]$. The contractor alternates two types of interval propagations:

- *The forward propagation:* Using interval analysis [11], run the simulator with the interval inputs $[p_1], \ldots, [p_{n_p}]$. All intermediate variables z_i involved in $\mathbf{f}$ will thus be bounded, *i.e.,* an interval $[z_i]$ for z_i will be obtained and used to bound or to contract the available domain for z_i. The new intervals obtained for the y_i's will then be used to refine the $[y_i]$'s.

- *The backward propagation:* From the output intervals $[y_1], \ldots, [y_{n_y}]$, sweep **f** backwardly to refine all intermediate variables. After completion of the backward propagation, the $[p_i]$'s have generally been contracted.

The methodology to be now given will make it possible to generate by hand from the simulator **f** (referred as the algorithm **A0**), the contractor (algorithm **A7**) which performs the backward and forward propagations. The methodology can be decomposed into seven steps. These steps will be illustrated on an academic example where the simulator **f** is given by the following algorithm **A0**.

$$\begin{array}{ll} \text{algorithm: } \mathbf{A0} & \\ \text{input:} & p_1, p_2; \\ 1 & z_1 := p_2; \\ 2 & y_1 = z_1.\left(p_2 - \exp(p_1)\right); \\ \text{output:} & y_1 \end{array} \tag{10}$$

Step 1: *Decomposition of the simulator*: Decompose **A0** in order to get only elementary operations, *i.e.*, only one operator or function should be involved at each statement. Intermediate variables may be added. For our example, one gets

$$\begin{array}{ll} \text{Algorithm: } \mathbf{A1} & \\ \text{input:} & p_1, p_2; \\ 1 & z_1 = \exp(p_1); \\ 2 & z_2 = p_2 - z_1; \\ 3 & z_1 := p_2; \\ 4 & y_1 = z_1.z_2; \\ \text{output:} & y_1 \end{array} \tag{11}$$

The intermediate variable used at statement 2 of **A1** is named z_1, which seems rather dangerous since z_1 it is also used at statement 3 of **A1**. This choice has voluntarily been made to illustrate some complications that generally occur for more complicated simulators.

Step 2: *Rename multi-occurring variables*: Rename some variables to avoid that a variable occurs more than once in the left hand side of **A1**. For our example, z_1 is affected at Steps 1 and 3 of **A1**. Thus at statement 3 and after statement 3, z_1 is renamed with a new name (here z_3). We get

$$\begin{array}{ll} \text{Algorithm: } \mathbf{A2} & \\ 1 & z_1 = \exp(p_1); \\ 2 & z_2 = p_2 - z_1; \\ 3 & z_3 := p_2; \\ 4 & y_1 = z_3.z_2; \end{array} \tag{12}$$

Step 3: *Generation of the backward simulator*: Read **A2** from the end to the beginning. At each iteration, isolate each variables of the right hand side. Write the corresponding statement in **A3**. We get

$$\begin{array}{l}\text{Algorithm: } \mathbf{A3}\\ 4 \quad z_3 = y_1/z_2; \;\; z_2 = y_1/z_3;\\ 3 \quad p_2 := z_3;\\ 2 \quad p_2 := z_2 + z_1; \;\; z_1 := p_2 - z_2;\\ 1 \quad p_1 := \ln(z_1);\end{array} \tag{13}$$

Note that if a *For* loop exists in **A2**, the same loop should be rewritten in **A3**, but in the reverse order.

Step 4: *Intervalize the forward simulator*: Rewrite **A2** into **A4** by replacing each variable and each operator or function by their interval counterpart. At each iteration, the interval domain computed has to be intersected with its previous value. For our example, we get:

$$\begin{array}{l}\text{Algorithm: } \mathbf{A4}\\ 1 \quad [z_1] := [z_1] \cap \exp([p_1]);\\ 2 \quad [z_2] := [z_2] \cap ([p_2] - [z_1]);\\ 3 \quad [z_3] := [z_3] \cap [p_2];\\ 4 \quad [y_1] := [y_1] \cap ([z_3] * [z_2]);\end{array} \tag{14}$$

Step 5: *Intervalize the backward simulator*: From **A3**, we get the following interval algorithm **A5**:

$$\begin{array}{l}\text{Algorithm: } \mathbf{A5}\\ 4 \quad [z_3] := [z_3] \cap ([y_1] \,/\, [z_2]); \;\; [z_2] = [z_2] \cap ([y_1] \,/\, [z_3]);\\ 3 \quad [p_2] := [p_2] \cap [z_3];\\ 2 \quad [p_2] := [p_2] \cap ([z_2] + [z_1]); \;\; [z_1] := [z_1] \cap ([p_2] - [z_2]);\\ 1 \quad [p_1] := [p_1] \cap \ln([z_1]);\end{array} \tag{15}$$

Step 6: *Merge the intervalized forward and backward simulators.* Merge algorithms **A4** and

A5. For our example, we get:

Algorithm: **A6**

$$
\begin{array}{ll}
1 & [z_1] := \exp([p_1]) \cap [z_1]\,; \\
2 & [z_2] := ([p_2] - [z_1]) \cap [z_2]\,; \\
3 & [z_3] := [p_2] \cap [z_3]\,; \\
4 & [y_1] := ([z_3] * [z_2]) \cap [y_1]\,; \\
4 & [z_3] = ([y_1] \,/\, [z_2]) \cap [z_3]\,; \quad [z_2] = ([y_1] \,/\, [z_3]) \cap [z_2]\,; \\
3 & [p_2] := [z_3] \cap [p_2]\,; \\
2 & [p_2] := ([z_2] + [z_1]) \cap [p_2]\,; \quad [z_1] := ([p_2] - [z_2]) \cap [z_1]\,; \\
1 & [p_1] := \ln([z_1]) \cap [p_1]\,;
\end{array}
\tag{16}
$$

Step 7: *Repeat several times the merged algorithm*: To generate the contractor for **A0**, initialize the domains for the intermediate variables to $]-\infty, \infty[$. Then, we repeat **A6** while the contraction takes place significantly. Note that the contractor has for inputs and outputs the domains associated with the inputs and outputs of **A0**.

Algorithm: **A7**

$$
\begin{array}{ll}
\text{inputs} & [p_1]\,, [p_2]\,, [y_1]\,; \\
\text{init} & [z_1] := [z_2] := [z_3] =]-\infty, \infty[; \\
 & \text{repeat} \\
1 & \qquad [z_1] := [z_1] \cap \exp([p_1]); \\
2 & \qquad [z_2] := [z_2] \cap ([p_2] - [z_1])\,; \\
3 & \qquad [z_3] := [z_3] \cap [p_2]\,; \\
4 & \qquad [y_1] := [y_1] \cap ([z_3] * [z_2])\,; \\
4 & \qquad [z_3] := [z_3] \cap [y_1] \,/\, [z_2]\,; \quad [z_2] := [z_2] \cap [y_1] \,/\, [z_3]\,; \\
3 & \qquad [p_2] := [p_2] \cap [z_3]\,; \\
2 & \qquad [p_2] := [p_2] \cap ([z_2] + [z_1])\,; \quad [z_1] := [z_1] \cap ([p_2] - [z_2])\,; \\
1 & \qquad [p_1] := [p_1] \cap \ln([z_1]); \\
 & \text{while the contraction is significant} \\
\text{output} & [p_1]\,, [p_2]\,, [y_1]\,;
\end{array}
\tag{17}
$$

Remark 1 *To improve the precision, one can use the centered-form interval arithmetic or the slope interval arithmetic.* ■

Remark 2 *Other contractors could be used. Even if the contractor we have proposed is efficient for a large class of nonlinear problems, but can be totally inefficient for simple linear simulators.*

For instance, if $\mathbf{f}$ *is given by*

$$\begin{array}{ll} \textit{inputs} & p_1, \ldots, p_4; \\ \textit{1} & y_1 = p_1 + p_2 + p_3 + p_4; \\ \textit{2} & y_2 = p_1 + p_2 + p_3 + p_4; \\ \textit{outputs} & y_1, y_2; \end{array} \tag{18}$$

and for $[p_1] = [p_2] = [p_3] = [p_4] = [-1, 1]$, $[y_1] = [0, 0]$ *and* $[y_2] = [0.1, 0.1]$, *the contractor, presented in this section, is unable to contract the domains. Nevertheless contractors based on linear techniques can find immediately that the domains can be contracted to the emptyset. Generally, a collaboration of different contractors can lead to a much more efficient contractor.*■

3 Bisecting

Using a branch-and-prune algorithm, it is possible to control the precision for the contractions of the domains. As an example, let us now present the following algorithm SIVIA (for Set Inversion Via Interval Analysis [7]). $\mathcal{L}$ is list of pairs of the form $([\mathbf{p}], [\mathbf{y}])$ which is initialized as the empty list. The procedure CONTRACT($[\mathbf{p}]$,$[\mathbf{y}]$, $\mathbf{y} = \mathbf{f}(\mathbf{p})$), is the contractor **A7** implemented as explained in the previous section. ε is the required accuracy.

Algorithm: SIVIA($[\mathbf{p}]$,$[\mathbf{y}]$)
CONTRACT($[\mathbf{p}]$,$[\mathbf{y}]$, $\mathbf{y} = \mathbf{f}(\mathbf{p})$);
if ($[\mathbf{p}] = \emptyset$), then return;
if ($\mathrm{w}([\mathbf{p}]) < \varepsilon$) then $\{\mathcal{L} = \mathcal{L} \cup \{([\mathbf{p}], [\mathbf{y}])\}$; return$\}$;
bisect ($[\mathbf{p}]$) into $[\mathbf{p}_1]$ and $[\mathbf{p}_2]$;
SIVIA($[\mathbf{p}_1]$,$[\mathbf{y}]$); SIVIA($[\mathbf{p}_2]$,$[\mathbf{y}]$);

(19)

After completion of SIVIA, $\mathcal{L}$ contains a list of pairs of the form

$$\mathcal{L} = \{([\mathbf{p}(1)], [\mathbf{y}(1)]), ([\mathbf{p}(2)], [\mathbf{y}(2)]), ([\mathbf{p}(3)], [\mathbf{y}(3)]), \ldots\}. \tag{20}$$

The union $\mathbb{S}_{\mathrm{p}}^{+}$ of all $[\mathbf{p}(i)]$'s provides an outer estimation for the set

$$\mathbb{S}_{\mathrm{p}} = \mathbf{f}^{-1}([\mathbf{y}]) \cap [\mathbf{p}], \tag{21}$$

and the union $\mathbb{S}_{\mathrm{y}}^{+}$ of all $[\mathbf{y}(i)]$'s provides an outer estimation for the set

$$\mathbb{S}_{\mathrm{y}} = \mathbf{f}([\mathbf{p}]) \cap [\mathbf{y}]. \tag{22}$$

From the list $\mathcal{L}$ it is also possible to get an accurate outer approximation of the smallest domains for the p_i's and the y_i's that are consistent with the prior domains $[\mathbf{p}]$,$[\mathbf{y}]$ and the equation $\mathbf{y} = \mathbf{f}(\mathbf{p})$.

4 Test case

As an illustration of SIVIA combined with forward backward interval propagation, consider the state estimation of the autonomous discrete-time system:

$$\left\{ \begin{array}{lcl} \left(\begin{array}{c} x_1(k) \\ x_2(k) \end{array} \right) & = & \left(\begin{array}{c} 0.1.x_1(k-1) + x_2(k-1).\exp(x_1(k-1)) \\ x_1(k-1) + 0.1.x_2^2(k-1) + \sin(k) \end{array} \right) \\ y(k) & = & x_2(k)/x_1(k) \end{array} \right. \tag{23}$$

with $k \in \{1, \ldots, 15\}$.

The interval data have been generated as follows. *(i)* For the unknown true value of the initial state vector $\mathbf{x}^*(0) = (-1\ 0)^{\mathrm{T}}$, we have computed by simulation the values for $\mathbf{x}^*(k)$ and $y^*(k), k \in \{1, \ldots, 15\}$. *(ii)* We have added to each $y^*(k)$ a random error with a uniform distribution in the interval $[-e, e]$ to generate the data $\hat{y}(k)$, where e is assumed to be known. *(iii)* Then, we set $[y(k)] = [\hat{y}(k) - e, \hat{y}(k) + e]$. We are thus certain that the interval data $[y(k)]$ contain the unknown true data $y^*(k)$. The problem to be solved is:

Problem 2: Given the equations of the system (23), given the interval data $[y(k)]$, given some bounded intervals $[x_1](0), [x_2](0)$ containing the initial state variables $x_1(0), x_2(0)$, compute accurate interval enclosure for the unknown true values for the $x_1^*(k)$'s, $x_2^*(k)$'s and $y^*(k)$'s. ■

The variables involved are $x_1(0)$, $x_2(0)$, $x_1(1)$, $x_2(1)$, $y(1)$, ..., $x_1(\bar{k})$, $x_2(\bar{k})$, $y(\bar{k})$. The inputs of the simulator algorithm are $\mathbf{p} = (x_1(0), x_2(0))^{\mathrm{T}}$. The prior domains for the initial state vector are given by

$$[x_1](0) = [-1.2, -0.8]; [x_2](0) = [-0.2, 0.2]; \tag{24}$$

The simulator $\mathbf{f}$ is given by

```
    Algorithm: A0
1   input: x1(0), x2(0);
2   for k := 1 to k̄,
3         x1(k) := 0.1.x1(k-1) + x2(k-1).exp(x1(k-1));
4         x2(k) := x1(k-1) + 0.1.x2^2(k-1) + sin(k);
5         y(k) := x2(k)/x1(k);
6   endfor
7   output: y(1), ..., y(k̄);
```

After performing the transformations described at Steps 1 and 2 of Section 2, we get

```
      Algorithm: A2
1     input: x_1(0), x_2(0);
2     for k := 1 to k̄,
3           z_1 (k) := exp(x_1 (k − 1));
4           z_2 (k) = x_2 (k − 1) .z_1 (k) ;
5           x_1 (k) := 0.1.x_1 (k − 1) +z_2 (k) ;
6           z_3 (k) := 0.1.sqr(x_2 (k − 1));
7           z_4 (k) := z_3 (k) + sin(k);
8           x_2 (k) := x_1 (k − 1) +z_4 (k) ;
9           y (k) := x_2 (k) /x_1(k);
10    endfor
11    output: y(1), . . . , y(k̄);
```

After the transformations described at Steps 3 to 7 of Section 2, we get the contractor **A7** given

by

$$
\begin{array}{ll}
 & \text{Algorithm: } \textsc{contract}([x_1](0),[x_2](0),[y](1),\ldots,[y](\bar{k}),\mathbf{y}=\mathbf{f}(\mathbf{x})); \\
\text{input:} & [x_1](0),[x_2](0),\ [y](1),\ldots,[y](\bar{k}) \\
\text{init:} & \text{for } k:=1 \text{ to } \bar{k} \\
 & \quad [x_1](k):=]-\infty,\infty[;\ [x_2](k):=]-\infty,\infty[; \\
 & \quad [z_1](k):=]-\infty,\infty[;\ [z_2](k):=]-\infty,\infty[; \\
 & \quad [z_3](k):=]-\infty,\infty[;\ [z_4](k):=]-\infty,\infty[; \\
 & \text{endfor} \\
\text{repeat} & \\
 & \text{for } k:=1 \text{ to } \bar{k}, \\
 & \quad [z_1](k):=[z_1](k)\cap\exp([x_1](k-1)); \\
 & \quad [z_2](k):=[z_2](k)\cap[x_2](k-1).[z_1](k); \\
 & \quad [x_1](k):=[x_1](k)\cap 0.1.[x_1](k-1)+[z_2](k); \\
 & \quad [z_3](k):=[z_3](k)\cap 0.1.\mathrm{sqr}([x_2](k-1)); \\
 & \quad [z_4](k):=[z_4](k)\cap[z_3](k)+\sin(k); \\
 & \quad [x_2](k):=[x_2](k)\cap[x_1](k-1)+[z_4](k); \\
 & \quad [y](k):=[y](k)\cap[x_2](k)/[x_1](k); \\
 & \text{endfor} \\
 & \text{for } k:=\bar{k} \text{ downto } 1, \\
 & \quad [x_2](k):=[x_2](k)\cap[y](k)*[x_1](k); \\
 & \quad [x_1](k):=[x_1](k)\cap[x_2](k)/[y](k); \\
 & \quad [x_1](k-1):=[x_1](k-1)\cap[x_2](k)-[z_4](k); \\
 & \quad [z_4](k):=[z_4](k)\cap[x_2](k)-[x_1](k-1); \\
 & \quad [z_3](k):=[z_3](k)\cap[z_4](k)-\sin(k); \\
 & \quad [x_2](k-1):=[x_2](k-1)\cap 0.1.\mathrm{sqrt}([z_3](k)); \\
 & \quad [x_1](k-1):=[x_1](k-1)\cap 10.([x_1](k)-[z_2](k)); \\
 & \quad [z_2](k):=[z_2](k)\cap[x_1](k)-0.1[x_1](k-1); \\
 & \quad [x_2](k-1):=[x_2](k-1)\cap[z_2](k)/[z_1](k); \\
 & \quad [z_1](k):=[z_1](k)\cap[z_2](k)/[x_2](k-1); \\
 & \quad [x_1](k-1):=[x_1](k-1)\cap\log([z_1](k)); \\
 & \text{endfor;} \\
\text{while} & \text{the contraction is significant} \\
\text{output:} & [x_1](0),\ldots,[x_1]\left(\bar{k}\right),\ [x_2](0),\ldots,[x_2]\left(\bar{k}\right),\ [y](1),\ldots,[y](\bar{k});
\end{array}
\tag{25}
$$

Note that the intermediate domains $[x_1](1),\ldots,[x_1]\left(\bar{k}\right)$, $[x_2](1),\ \ldots,\ [x_2]\left(\bar{k}\right)$ have been put as outputs of the contractor, because they are of interest in our context of state estimation.

In the case where there is no noise (*i.e.,* $e=0$), The contractor is able find all true values for the variables with an accuracy of 8 digits in 0.1 seconds. No bisections have been generated

by SIVIA. The boxes drawn on the left subfigure of Figure 1 are the boxes obtained after one forward-backward propagation.

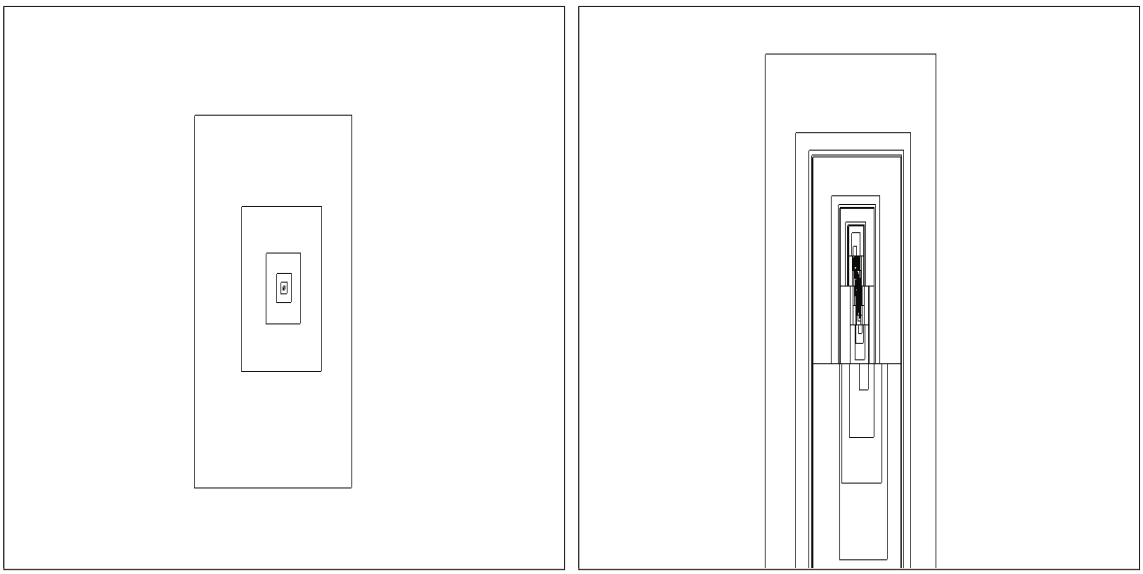

Figure 1: Left: contractions generated in a noise-free context; Right: Contractions and bisections generated in a noisy context. The two frame boxes are $[-1.2, -0.8] \times [-0.2, 0.2]$.

For $e = 0.5$, the volume of the set $\mathbb{S}_{\mathbf{x}(0)}$ of all $\mathbf{x}(0)$'s that are consistent is not equal to zero anymore, and thus, even if the contractor works perfectly, a large number of bisections have to be performed (see the right subfigure of Figure 1). The computing time is about 3 seconds for $\varepsilon = 0.001$. The *prior* data intervals are on the right part of Figure 2 and the corresponding contracted intervals obtained by SIVIA are in the left part of 2. The domains obtained for the state variable $x_1(k)$ and $x_2(k)$ are given in Figure 3.

5 Conclusion

Contractors based on forward and backward propagation of intervals in an algorithm (see [1]) have been used here, for the first time in the context of state estimation, to contract the feasible domains to the time variables involved in the model. A branch-and-prune algorithm has also been proposed to control the accuracy of the contractions. Contrarily to other interval based methods such as the one presented in [8], [9] and [6], the bisections have to be done only is the space of $\mathbf{x}(0)$ and not with respect to all state variables. An illustrative example has shown the efficiency of the approach. On this example, it has been shown that when the volume of the set $\mathbb{S}_{\mathbf{x}(0)}$ of all feasible initial state vectors is not equal zero, the boxes generated tend to

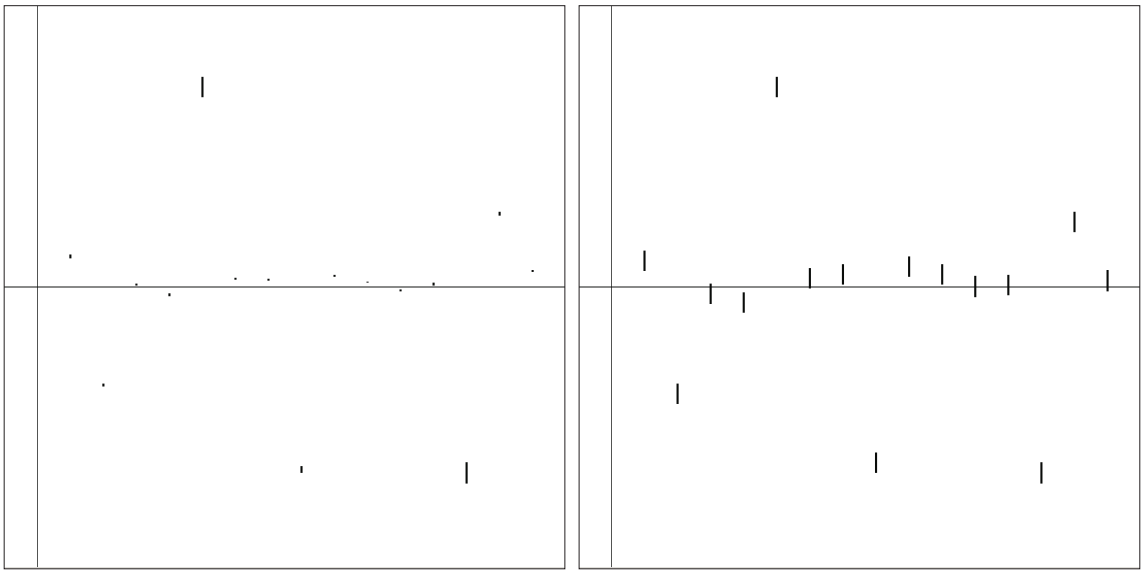

Figure 2: Left: contracted intervals containing the noise-free data; Right: initial domains for the data. The two frame boxes are $[-1, 16] \times [-2, 2]$.

accumulate and the whole feasible set. Such an expensive accumulation could be avoided by using the algorithm HULL presented in [5].

The source code in C++ Builder 4, and an executable program for IBM-compatible PCs corresponding to the example are 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*, pages 230–244, Las Cruces, USA, 1999. MIT Press.

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

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

[4] L. Jaulin. *Solution globale et garantie de problèmes ensemblistes ; application à l'estimation non linéaire et à la commande robuste*. PhD dissertation, Université Paris-Sud, Orsay, 1994.

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

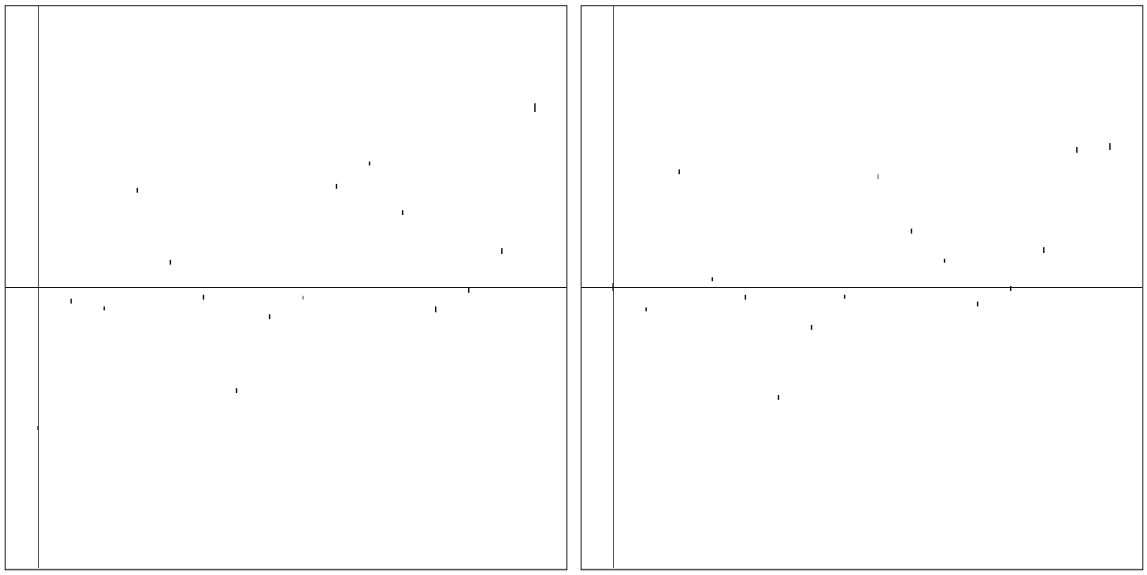

Figure 3: Left: contracted domains for $x_1(k)$; Right: contracted domains for $x_2(k)$. The two frame boxes are $[-1, 16] \times [-15, 15]$.

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

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

[8] M. Kieffer, L. Jaulin, and E. Walter. Guaranteed recursive nonlinear state estimation using interval analysis. In *Proc. 37th IEEE Conference on Decision and Control*, pages 3966–3971, Tampa, December 16-18, 1998.

[9] M. Kieffer, L. Jaulin, E. Walter, and D. Meizel. Guaranteed mobile robot tracking using interval analysis. In *Proc. MISC'99 Workshop on Application of Interval Analysis to Systems and Control*, pages 347–359, Girona, February 24-26, 1999.

[10] M. Milanese, J. Norton, H. Piet-Lahanier, and E. Walter (Eds). *Bounding Approaches to System Identification*. Plenum Press, New York, 1996.

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

[12] J. P. Norton (Ed.). Special issue on bounded-error estimation: Issue 1. *Int. J. of Adaptive Control and Signal Processing*, 8(1):1–118, 1994.

[13] J. P. Norton (Ed.). Special issue on bounded-error estimation: Issue 2. *Int. J. of Adaptive Control and Signal Processing*, 9(1):1–132, 1995.

[14] E. Walter (Ed.). Special issue on parameter identification with error bounds. *Mathematics and Computers in Simulation*, 32(5&6):447–607, 1990.