Solving non-linear constraint satisfation problems involving time-dependant functions

Aymeric Bethencourt and Luc Jaulin

Abstract. In this paper, we consider the resolution of non-linear constraint satisfaction problems where the variables of the systems are trajectories (functions from $\mathbb{R}$ to $\mathbb{R}^n$). We introduce the notion of *tubes* as *intervals of functions*, for which the lower and upper bounds are trajectories with respect to the inclusion. We then define basic operators and prove propositions verified by tubes. We show the possibility to build contractors on tubes and propagate constraints to solve problems involving time-dependant functions as the unknown variables. We show that the approach is particularly powerful when inter-temporal equations (e.g. delays) are involved. Finally, in order to illustrate the principle and efficiency of the approach, several test cases are provided.

ENSTA-Bretagne
LABSTICC
2 Rue Francois Verny
29200 Brest, France

1. Introduction

Interval analysis [1][2] has become over the past few years a strong alternative to traditional probabilistic approaches [3] to solve complex systems of equations, e.g. Simultaneous Localization And Mapping [4][5][6], 3D Reconstruction [7], path planning [8], or more generally the characterization of the state evolution of dynamic systems [9][10]. Interval analysis has been proven particularly efficient when the number of equations (namely *constraints*) is superior to the number of unknown variables [11]. The idea is to cast the system as a constraint satisfaction problem by considering intervals enclosing the solution and use *contractors* (built from the constraints) to successively contract these intervals until a fixed point is reached.

As more and more applications of Interval Analysis involve trajectories (functions from $\mathbb{R}$ to $\mathbb{R}^n$) [12], this paper will introduce the notion of *intervals of trajectories* (namely *tubes*), inspired from Taylor models [13], and define basic operations on tubes in section 2. We then show how to build contractors on tubes and demonstrate a few minimal contractors in section 3. We define the notion of *Synchronous* and *Asynchronous constraints* (constraints that involve inter-temporal relations) and show that our approach is particularly powerful for asynchronous constraints. Finally section 4 provides three non-linear test cases to illustrate the methodology, and section 5 concludes the paper.

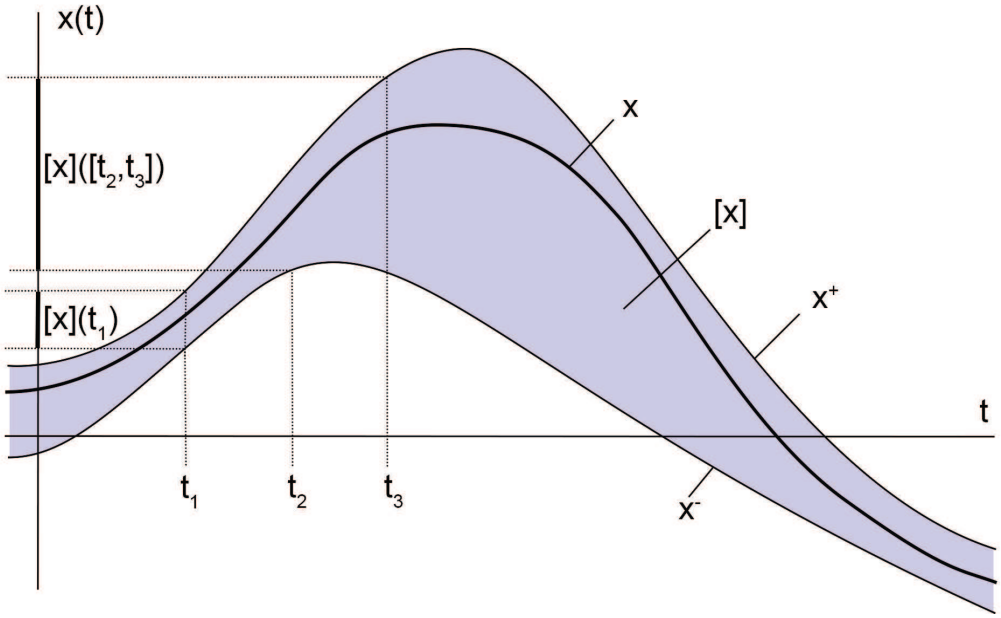

FIGURE 1. A tube $[x]$ of $\mathbb{R}$ which encloses the function x

2. Intervals of functions (or *tubes*)

In Interval Analysis, the unknown variables are usually boolean numbers, integers or real numbers, but the originality of this paper is to consider trajectories. In this section we consider the reader to be familiar with interval analysis [2]. We do however recall the basic notions of *lattices* and *contractors* that are used for working with *tubes*.

2.1. Lattices

A *lattice* $(\mathcal{E}, \leq)$ is a partially ordered set, closed under least upper and greatest lower bounds [14]. The least upper bound of x and y is called the *join* or *supremum* and is denoted by $x \vee y$. The greatest lower bound is called the *meet* or *infimum* and is written as $x \wedge y$.

Example : The set $\mathbb{R}^n$ is a lattice with respect to the partial order relation given by $\mathbf{x} \leq \mathbf{y} \Leftrightarrow \forall i \in \{1, \ldots, n\}, x_i \leq y_i$. We have

$$\begin{aligned} \mathbf{x} \wedge \mathbf{y} &= (x_1 \wedge y_1, \ldots, x_n \wedge y_n) \text{ and} \\ \mathbf{x} \vee \mathbf{y} &= (x_1 \vee y_1, \ldots, x_n \vee y_n) \end{aligned} \tag{2.1}$$

where $x_i \wedge y_i = \min(x_i, y_i)$ and $x_i \vee y_i = \max(x_i, y_i)$. A lattice $\mathcal{E}$ is *complete* if for all (finite or infinite) subsets $\mathcal{A}$ of $\mathcal{E}$, the least upper bound (denoted $\vee\mathcal{A}$) and the greatest lower bound (denoted $\wedge\mathcal{A}$) belong to $\mathcal{A}$.

Notice that the concept of lattice only works when using multidimensional intervals as set representation since min() and max() can only be defined element wise. This is not the case when using other set representations, such as e.g. ellipsoids [15].

2.2. Tubes

A *Tube* (or interval of a trajectory) [16] [17] is an set-membership vision of a random signal. The set $\mathcal{F}^n$ of all functions from $\mathbb{R}$ to $\mathbb{R}^n$ is a complete lattice with partial order $\mathbf{x} \leq \mathbf{y} \Leftrightarrow \forall t \in \mathbb{R},\ \mathbf{x}(t) \leq \mathbf{y}(t)$. A *tube* $[\mathbf{x}]$ is an interval $[\mathbf{x}^-, \mathbf{x}^+]$ of $\mathcal{F}^n$, i.e., a pair of two trajectories $\mathbf{x}^-, \mathbf{x}^+$ such that for all t, $\mathbf{x}^-(t) \leq \mathbf{x}^+(t)$. The set of all tubes of $\mathcal{F}^n$ is denoted by $\mathbb{I}\mathcal{F}^n$.

An element $\mathbf{x}$ of $\mathcal{F}^n$ belongs to the tube $[\mathbf{x}]$ if $\forall t, \mathbf{x}(t) \in [\mathbf{x}](t)$. Fig. 1 illustrates a function $x \in \mathcal{F}^1$ which is inside the tube $[x]$. This tube gives us information related to the unknown function x.

If $\mathbf{x}$ is a function from $\mathbb{R}$ to $\mathbb{R}^n$ (i.e., $\mathbf{x} \in \mathcal{F}^n$), we define

$$\mathbf{x}([t]) = \{\mathbf{x}(t), t \in [t]\}. \tag{2.2}$$

Numerically, a tube $[\mathbf{x}]$ is defined by

$$[\mathbf{x}]([t]) = \bigsqcup_{t \in [t]} [\mathbf{x}](t), \tag{2.3}$$

i.e., $[\mathbf{x}]([t])$ is the smallest box which encloses all boxes $[\mathbf{x}](t), t \in [t]$. It is easy to prove that

$$\mathbf{x} \in [\mathbf{x}], t \in [t] \Rightarrow \mathbf{x}(t) \in [\mathbf{x}]([t]), \tag{2.4}$$

and that no box smaller than $[\mathbf{x}]([t])$ satisfies this property.

2.3. Tube arithmetic

We can extend operations on lattices to intervals of this lattice. The extension is the smallest interval which encloses all possible values. Therefore we can extend operations on $\mathcal{F}^n$, such as sums, multiplication, image by a function, etc. to tubes. We use the rules of interval arithmetic and inclusion functions [1]. An arithmetic on tubes is thus a direct extension of interval arithmetic. As it is the case for interval computation, the result of an operation on tubes contains all results of the same operation performed on the enclosed elements of $\mathcal{F}^n$.

Integral. Consider two numbers t_1, t_2 such that $t_2 \geq t_1 \geq 0$. The integral of a tube $[\mathbf{x}]$ over an interval $[t_1, t_2]$ is defined [18] by

$$\int_{t_1}^{t_2} [\mathbf{x}](\tau)\, d\tau = \left\{ \int_{t_1}^{t_2} \mathbf{x}(\tau)\, d\tau \text{ such that } \mathbf{x} \in [\mathbf{x}] \right\}. \tag{2.5}$$

Since $t_2 \geq t_1$, we deduce from the monotonicity of the integral operator that

$$\int_{t_1}^{t_2} [\mathbf{x}](\tau)\, d\tau = \left[\int_{t_1}^{t_2} \mathbf{x}^-(\tau)\, d\tau, \int_{t_1}^{t_2} \mathbf{x}^+(\tau)\, d\tau \right]. \tag{2.6}$$

where $[x] = [x^-, x^+]$. From the definition of tube integrals, we have

$$\mathbf{x} \in [\mathbf{x}] \Rightarrow \int_{t_1}^{t_2} \mathbf{x}(\tau)\, d\tau \in \int_{t_1}^{t_2} [\mathbf{x}](\tau)\, d\tau. \tag{2.7}$$

Moreover, the *interval primitive* defined by $\int_0^t [\mathbf{x}](\tau)\, d\tau$ defines a tube that vanishes for $t = 0$.

Extension of operators. If $\diamond$ is a binary operator in $\mathbb{R}^n$(such as $+$,$-$,the multiplication $*$ when $n = 1$ or the vector product when $n >= 2$) then it can be extended to $\mathcal{F}^n$ (in the Minkowski sense) and to $\mathbb{I}\mathcal{F}^n$ as follows

$$A \diamond B = \{a \diamond b, a \in A, b \in B\} \tag{2.8}$$

From the monotony of the operator, we have

$$[x] \diamond [y] = [\vee([x] \diamond [y]), \wedge([x] \diamond [y])] \tag{2.9}$$

E.g., for $[\mathbf{x}]$ and $[\mathbf{y}] \in \mathbb{I}\mathcal{F}^n$ and $a \in \mathbb{R}^+$

$$\begin{array}{ll} \text{(i)} & [x] + [y] = [x^- + y^-, x^+ + y^+] \\ \text{(ii)} & a[x] = [ax^- \vee ay^-, ax^+ \wedge ay^+] \end{array} \tag{2.10}$$

Tube envelope. Consider a collection $\{f_i, i \in \mathbb{N}\}$ of functions. The tube envelope $\square\{f_i, i \in \mathbb{N}\}$ is the smallest tube enclosing all f_i. We have

$$\square\{f_i, i \in \mathbb{N}\} = [\vee_{i\in\mathbb{N}} f_i; \wedge_{i\in\mathbb{N}} f_i)] \tag{2.11}$$

For instance,

$$\text{for } t \in [0,\infty], \square\{t, \cos(t), -1\} = [-1; t] \tag{2.12}$$

Inclusion. We define the tube inclusion as follows

$$[x] \sqsubset [y] \iff y^- \leqslant x^- \leqslant x^+ \leqslant y^+ \tag{2.13}$$

Intersection. We define the tube intersection as follows

$$[x] \sqcap [y] \iff [[x] \vee [y], [x] \wedge [y]] \tag{2.14}$$

2.4. Constraint propagation on tubes

Tube contractor. Consider a constraint $\mathcal{L}(x)$ on a trajectory x. A contractor associated with the constraint $\mathcal{L}$ is an operator

$$[y] \xrightarrow{C_{\mathcal{L}}} [x] \tag{2.15}$$

where $[x]$ and $[y]$ are tubes, such that

$$\begin{array}{ll} \forall t, [x](t) \subset [y](t) & \text{(contraction)} \\ \left(\begin{array}{c} \mathcal{L}(x) \\ x \in [x] \end{array} \right) \Longrightarrow x \in [y] & \text{(completeness)} \end{array} \tag{2.16}$$

We call C* the *minimal contractor* for $\mathcal{L}$ that returns the tube $[y]$ with the smallest width and is consistent with the constraint $\mathcal{L}$.

Many problems of estimation, control, robotics, etc. can be represented as *constraint satisfaction problems* (CSP) [19],[20] and many minimal contractors can be applied to optimally contract the domains depending on the class of the problem [11]. Several minimal contractors have been developed over the years: Gauss elimination, Gauss-Seidel Algorithm, Krawczyk method, Newton algorithm, etc. [2].

We can also propose minimal contractors for tubes, but let us first separate our constraints into two categories:

• Synchronous constraint: Constraint that is set to happen at the same time, typically $x(t) = y(t)$ is synchronous.

• Asynchronous constraint: Constraint that is set to happen at different times, typically $x(t) = y(t-2)$ is asynchronous. This requires to have the history of the variable y and this is where tubes will be particularly powerful.

We are now proposing a few minimal contractors for both types.

Proposition 1. The minimal contractor associated with the synchronous constraint $x \leqslant y$ or $\forall t \in [0,\infty[, x(t) \leqslant y(t)$ is

$$C_{\leqslant} \left(\begin{array}{c} [x] \\ [y] \end{array} \right) = \left(\begin{array}{c} [x^-, x^+ \wedge y^+] \\ [x^- \vee y^-, y^+] \end{array} \right) \tag{2.17}$$

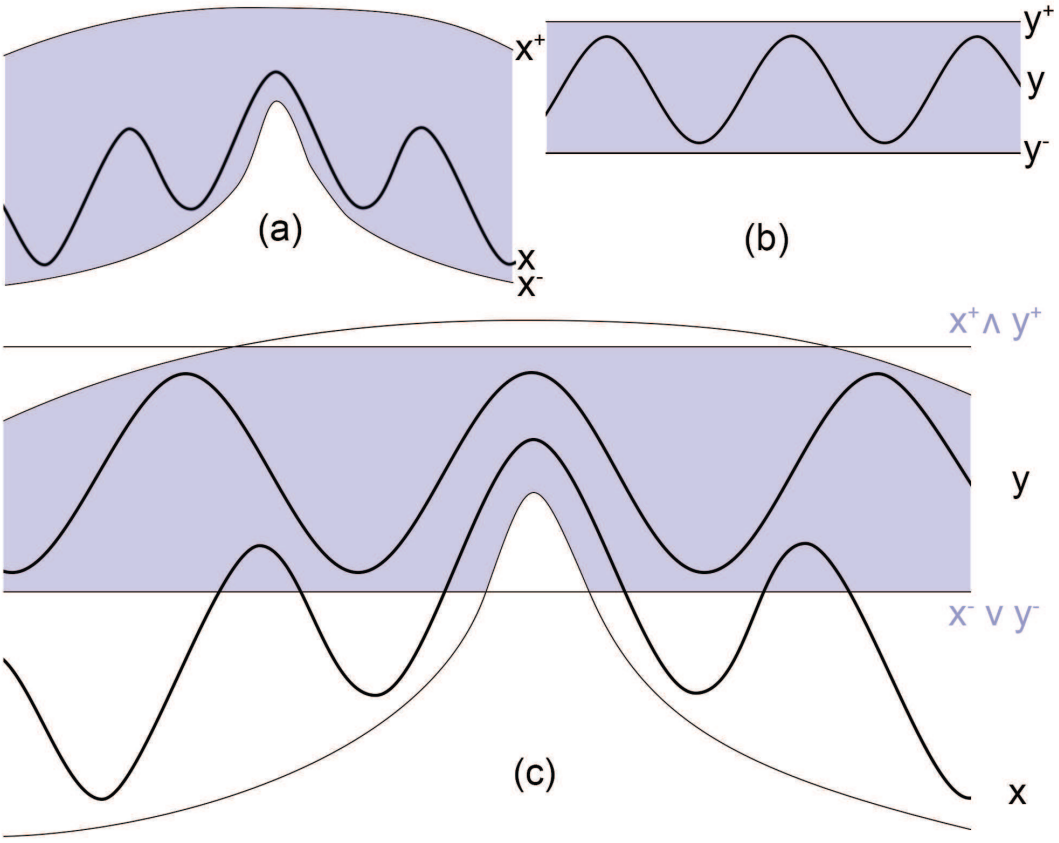

FIGURE 2. (a) represents the trajectory x and its tube $[x]$, (b) the trajectory y and its tube $[y]$, and (c) illustrates the meet and join of them.

Proof. $x \leqslant y$ so according to the independence theorem, we can find a such that $y = x + a$ with $a \geqslant 0$. We can therefore contract the associated tube.

$$\begin{aligned} & [y] = [y] \cap ([x] + [a]) \\ \iff & [y] = [y^- \vee (x^- + a^-), y^+ \wedge (x^+ + a^+)] \\ \iff & [y] = [y^- \vee (x^- + 0), y^+ \wedge (x^+ + \infty)] \\ \iff & [y] = [y^- \vee x^-, y^+] \end{aligned} \tag{2.18}$$

since $[a] = [0, \infty[$. By considering $x = y + a$ with $a \leqslant 0$, we can prove that $[x] = [x^-, x^+ \wedge y^+]$ the same way.

Figure 2 illustrates the contractions.

Proposition 2. The minimal contractor associated with the synchronous constraint $x = y$ (which means that $\forall t \in [0, \infty], x(t) = y(t)$) is

$$C_{=} \left(\begin{array}{c} [x] \\ [y] \end{array} \right) = \left(\begin{array}{c} [x^- \vee y^-, x^+ \wedge y^+] \\ [x^- \vee y^-, x^+ \wedge y^+] \end{array} \right) \tag{2.19}$$

Proof. As $x = y$, any given elements of x and y cannot exceed each other's range. Both variables will share a new interval equal to the intersection of their respective intervals

$$\begin{aligned} & [x] = [x] \cap [y] \\ \iff & [x] = [x^- \vee y^-, x^+ \wedge y^+] \end{aligned} \tag{2.20}$$

The same is proven for y.

Proposition 3. The minimal contractor associated with a translation or delay $\forall t \in [0, \infty[, x(t) = y(t-\tau)$ is

$$C_{delay} \begin{pmatrix} [x] \\ [y] \end{pmatrix} = \begin{pmatrix} \forall t, & [x^-(t) \vee y^-(t-\tau), x^+(t) \wedge y^+(t-\tau)] \\ & [x^-(t+\tau) \vee y^-(t), x^+(t+\tau) \wedge y^+(t)] \end{pmatrix} \tag{2.21}$$

Proof. The proof is immediate considering Proposition 2.

Fig. 3 illustrates this notion. The purple area represents the tube $[x^-(t) \vee y^-(t-\tau), x^+(t) \wedge y^+(t-\tau)]$

Let us notice that this constraint is asynchronous as it involves two different times: t and $t-\tau$.

Special case. If the constraint is $x(t) = x(t-\tau)$, then x is τ-periodic and $\forall k \in \mathbb{N}, C_{periodic}([x]) = ([x^-(t) \vee x^-(t-k\tau), x^+(t) \wedge x^+(t-k\tau)])$

Proposition 4. The minimal contractor associated with an axial symmetry around the axis $t = \tau$ is

$$C_{ASym}([x(\tau+t)]) = [x^-(\tau+t) \vee x^-(\tau-t); x^+(\tau+t) \wedge x^+(\tau-t)] \tag{2.22}$$

Proof. Let's define the axial symmetry operator S such that $S(f(\tau+t)) = f(\tau-t)$. S is increasing since

$$\begin{aligned} f(\tau+t_a) \leqslant f(\tau+t_b) & \\ \Leftrightarrow S(f(\tau+t_a)) \leqslant S(f(\tau+t_b)) & \\ \Leftrightarrow f(\tau-t_a) \leqslant f(\tau-t_b) & \end{aligned} \tag{2.23}$$

Moreover,

$$\begin{aligned} [x(\tau+t)] = [x^-(\tau+t), x^+(\tau+t)] & \\ \Leftrightarrow S([x(\tau+t)]) = [S(x^-(\tau+t)), S(x^+(\tau+t))] & \\ \Leftrightarrow S([x(\tau+t)]) = [x^-(\tau-t), x^+(\tau-t)] & \end{aligned} \tag{2.24}$$

Therefore $C_{ASym}([x(\tau+t)]) = [x(\tau+t)] \sqcap S([x(\tau+t)]) = [x^-(\tau+t) \vee x^+(\tau-t); x^+(\tau+t) \wedge x^-(\tau-t)]$

Special case : When τ=0, then x is even and $C_{even}([x(t)]) = [x^-(t) \vee x^-(-t); x^+(t) \wedge x^+(-t)]$

Fig.4 illustrates this case.

Proposition 5. The minimal contractor associated with a central symmetry around the axis $t = \tau$ is

$$C_{CSym}([x(\tau+t)]) = [x^-(\tau+t) \vee (-x^+(\tau-t)); x^+(\tau+t) \wedge (-x^-(\tau-t))] \tag{2.25}$$

Proof. The proof is identical to proposition 4 considering S decreasing.

Special case : When $\tau = 0$, then x is odd and $C_{odd}([x(t)]) = [x^-(t) \vee (-x^+(-t)); x^+(t) \wedge (-x^-(-t))]$

Notice that both symmetries are asynchronous constraints.

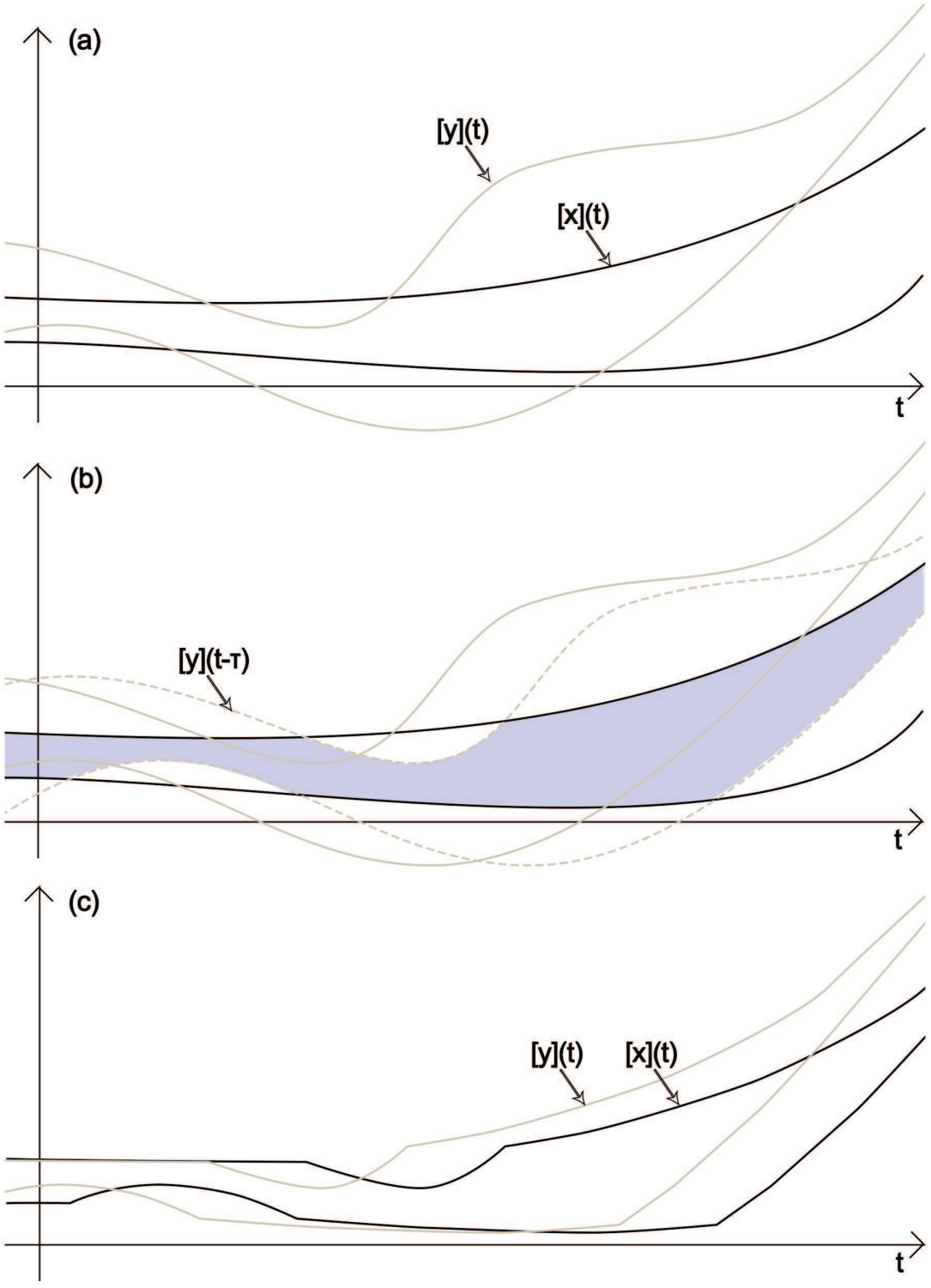

FIGURE 3. Asynchronous constraint propagation on tubes.

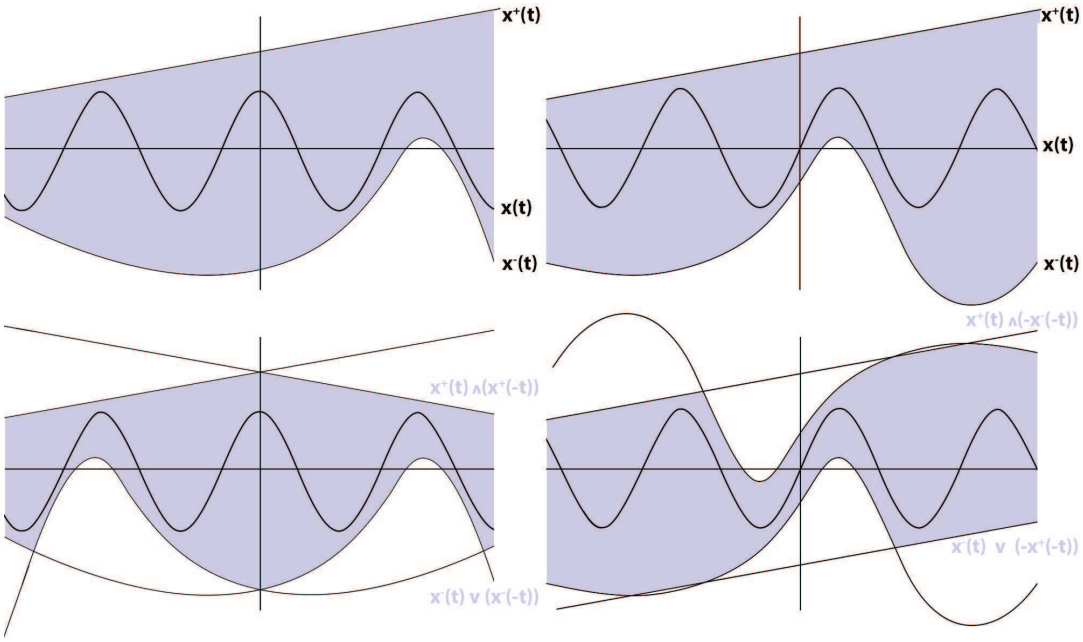

FIGURE 4. On the left, we illustrate the axial symmetry. On the right, the central symmetry.

2.5. State estimation

The problem to be considered in this section is the state estimation of a non-linear continuous-time systems in a bounded error context. Usually we describe the system as follows:

$$\begin{aligned} \dot{\mathbf{x}}(t) &= \mathbf{f}(\mathbf{x}(t)) \\ \mathbf{y}(t) &= \mathbf{g}(\mathbf{x}(t)) \end{aligned} \tag{2.26}$$

where $t \in [0, \infty[$ is the time, $\mathbf{x}(t) \in \mathbb{R}^n$ and $\mathbf{y}(t) \in \mathbb{R}^n$ are respectively the state and the output vectors at time t, and $\mathbf{f}$ and $\mathbf{g}$ the evolution and observation functions. To solve the state estimation problem, many efficient methods can be found in the literature when $\mathbf{f}$ and $\mathbf{g}$ are linear [21][22]. When $\mathbf{f}$ is non-linear, there are only to our knowledge a few approaches [23][24][25] that propose an efficient state estimation.

The originality of our approach is to consider that $\mathbf{g}$ is no longer the observation function at a given state, but a relationship between two states of the system at different times. $\mathbf{g}$ is therefore an inter-temporal equation between the state of the system at time t_1 and its state at time t_2 . (2.26) becomes

$$\begin{aligned} \dot{\mathbf{x}}(t) &= \mathbf{f}(\mathbf{x}(t)) \\ \mathbf{y}(t_1, t_2) &= \mathbf{g}(\mathbf{x}(t_1), \mathbf{x}(t_2)) \end{aligned} \tag{2.27}$$

This representation allows us to express inter-temporal constraints, and use them to solve time-dependent problems as presented in the next section. An alternative solution is to formulate the inter-temporal constraints in the framework of *DAEs (Differential Algebraic Equations)*. Reachability analysis of DAE systems has already been considered in [26] and [27]. In our approach, we propose the following way to solve such systems. We first start with a with a large tube enclosing $\mathbf{x}$ and contract $[\mathbf{x}]$ for all t using the output vectors, i.e. $\mathbf{y}(t_1, t_2)$ will contract $[\mathbf{x}](t_1)$ and $[\mathbf{x}](t_2)$. We then

	Algorithm $C_{CONTRACT}$ (in : k_0, $[\mathbf{x}_0]$, inout: $[\mathbf{x}]$)
1	for $k := 0$ to $\bar{k}$
2	if($k == k_0$)
3	$[\mathbf{x}](k) = [\mathbf{x}_0]$;
5	endif
6	$[\mathbf{x}](k+1) = [\mathbf{x}](k+1) \cap [\phi]([\mathbf{x}](k), [\dot{\mathbf{x}}](k))$;
13	endfor
14	for $k := \bar{k}$ to 0
15	if($k == k_0$)
16	$[\mathbf{x}](k) = [\mathbf{x}_0]$;
18	endif
19	$[\mathbf{x}](k) = [\mathbf{x}](k) \cap [\tilde{\phi}]([\mathbf{x}](k+1), [\dot{\mathbf{x}}](k+1))$;
26	endfor

TABLE 1. Contract algorithm

apply a variant of the $STRANGLE$ and $CONTRACT$ algorithms from [23]. We first apply the $STRANGLE$ around a time k_0 given an initial interval $[\mathbf{x}_0]$ and a precision ϵ. This algorithm bisects the interval $[\mathbf{x}_0]$ into two sub-intervals and propagates each sub-interval using the $CONTRACT$ algorithm. If the contraction of the tube $[\mathbf{x}]$ with the sub-interval possesses empty intervals, then the sub-interval does not contain the solution and we discard it. If the tube has no empty interval, then the sub-interval contains the solution and we bisect it again by recursively calling $CONTRACT$ until the precision is reached. We then take the union of all the tubes that possesses no empty interval. This union is guaranteed to contain the solution.

The $CONTRACT$ algorithm uses the operator $[\phi]$ as defined in [23] to compute enclosures of the state vector at times $\delta, 2\delta, ...\bar{k}\delta$ where δ is the sampling time and $\bar{k}$ is the largest integer smaller than $\bar{t}/\delta$, from a given box $[\mathbf{x}]$ containing $\mathbf{x}$. $\bar{t}$ is the end time of the problem. $[\phi]$ is the operator for the time increasing and computes the state of the system at time $k+1$ from k. $[\tilde{\phi}]$ is the operator for the time decreasing and computes the state of the system at time k from $k+1$. For the sake of simplicity in the following examples, we are using a basic Euler method for $[\phi]$ and thus for solving the ODE and computing the successive states of the system. Notice that using an Euler method means that we lose the guarantee in the results below but do not impact the proof of concept presented in this article. Moreover, there exists multiple libraries for computing rigorous bounds on the solution of ordinary differential equations, e.g. *VNODE* [28] and multiple methods especially developed for studying the evolution of dynamic systems [29].

3. Examples

3.1. Example 1: Sinusoidal signal

Let's consider an unknown signal $a(t)$ where t is the time. We consider to know only a few properties about a and want to find the smallest tube enclosing the solution. For example, let us consider that a verifies :

	Algorithm $C_{STRANGLE}$ (in : $k_0, [\mathbf{x}_0], \epsilon$, inout : $[\mathbf{x}]$)
1	if($[\mathbf{x}_0].width > \epsilon$)
2	$Bisect([\mathbf{x}_0], [\mathbf{x}_1], [\mathbf{x}_2])$;
3	$[\mathbf{z}] = CONTRACT(k_0, [\mathbf{x}_1], [\mathbf{x}])$
4	if($[\mathbf{z}]$ has no empty interval)
5	$CONTRACT(k_0, [\mathbf{x}_1], [\mathbf{z}])$;
6	endif
7	$[\mathbf{z}] = CONTRACT(k_0, [\mathbf{x}_2], [\mathbf{x}])$
8	if($[\mathbf{z}]$ has no empty interval)
9	$CONTRACT(k_0, [\mathbf{x}_2], [\mathbf{z}])$;
10	endif
11	else
12	$[\mathbf{z}] = CONTRACT(k_0, [\mathbf{x}_0], [\mathbf{x}])$
13	if($[\mathbf{z}]$ have no empty interval)
14	$[\mathbf{x}] = [\mathbf{x}] \sqcup [\mathbf{z}]$;
15	endif
16	endif
17	return $[\mathbf{x}]$

TABLE 2. Strangle algorithm

$$a([-\infty;\infty]) \subset [-1;1] \tag{3.1}$$

$$a([\frac{\pi}{2},\pi]) \subset [-0.7(t-\frac{\pi}{2})+0.99, \tag{3.2}$$

$$-0.1(t-\frac{\pi}{2})+1.01] \tag{3.3}$$

$$a(t+\pi) = -a(t) \tag{3.4}$$

$$a(t+2\pi) = a(t) \tag{3.5}$$

$$b(t-\frac{\pi}{2}) = a(t) \tag{3.6}$$

$$b(t) = \dot{a}(t) \tag{3.7}$$

We first define the tube $[a](t) = [a^-(t), a^+(t)] = [-\infty, \infty]$ then apply contractors 3.1 and 3.2. Each inclusion represents two contractors. For instance, 3.2 is equivalent to

$$\forall t \in [\frac{\pi}{2},\pi], \begin{array}{l} a(t) \leqslant -0.1(t-\frac{\pi}{2})+1.01 \\ a(t) \geqslant -0.7(t-\frac{\pi}{2})+0.99 \end{array} \tag{3.8}$$

Therefore, we apply the superiority contractor presented in section 3. The result is presented Fig. 5(a). Contractors 3.4 and 3.5 respectively shows that our signal is symmetrical with respect to the point $(\pi, 0)$ and is 2π periodic. Results of these contractions are presented Fig. 5(b) and Fig. 5(c).

Then we apply the time delayed contractor from 3.6 to create a new tube b :

$$[b(t)] = [b(t)] \cap [a(t+\frac{\pi}{2})] \tag{3.9}$$

Finally, from the differential equation 3.7, we can integrate $[b]$ and contract $[a]$ using the equality contractor :

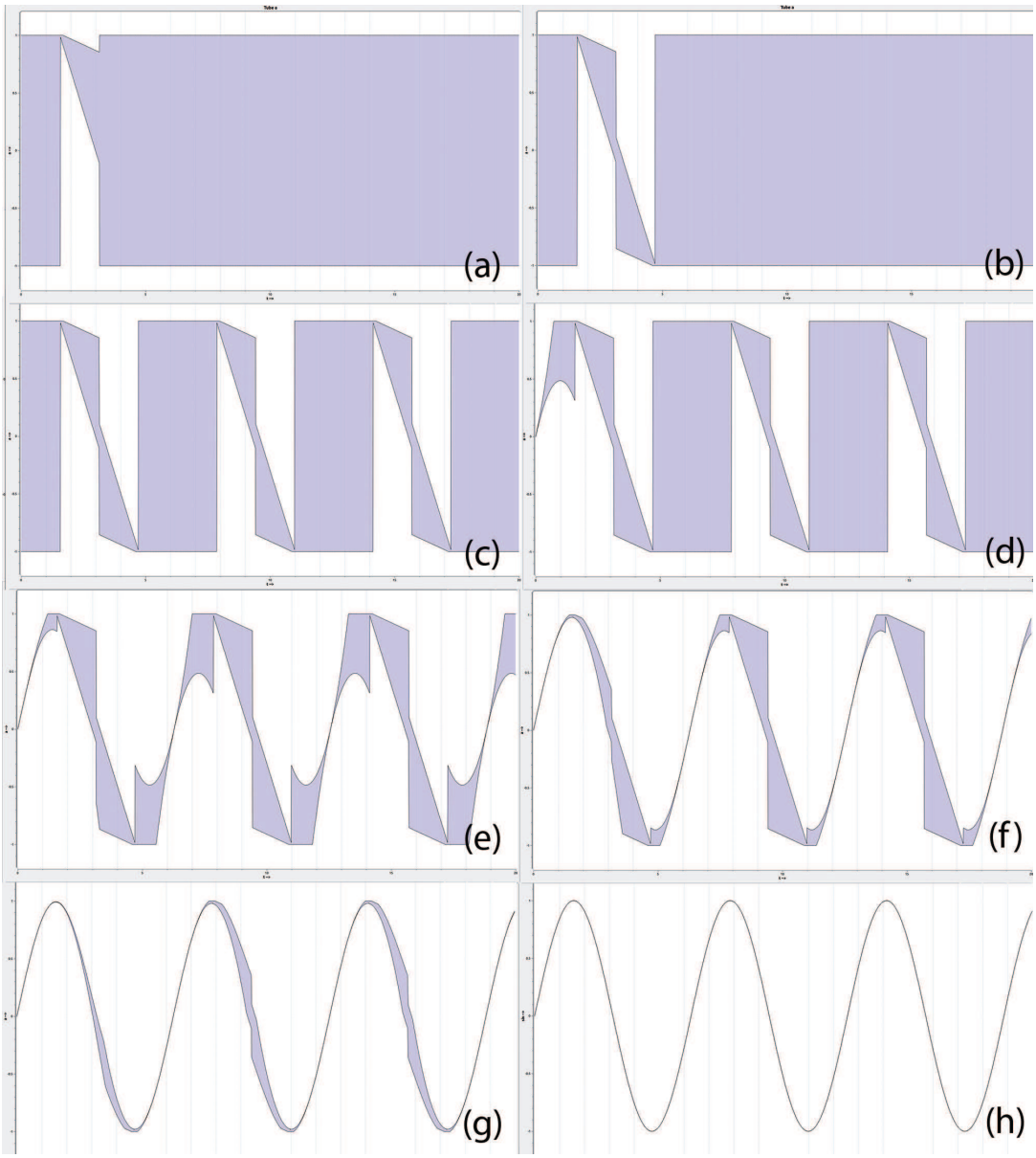

FIGURE 5. Successive contraction of the tube $[a]$.

$$[a(t)] = [a(t)] \cap [\int_{\tau=0}^{t} \frac{db^+(\tau)}{dt} d\tau, \int_{\tau=0}^{t} \frac{db^-(\tau)}{dt} d\tau] \tag{3.10}$$

The results is illustrated in Fig. 5(d). To contract the tube even more, we can re-apply all the contractors one time (Fig. 5(e)), three times (Fig. 5(f)), five times (Fig. 5(g)) or even ten times (Fig. 5(h)) until the width of the tube is satisfying or until a fixed point is reached. We then clearly recognize the sine signal.

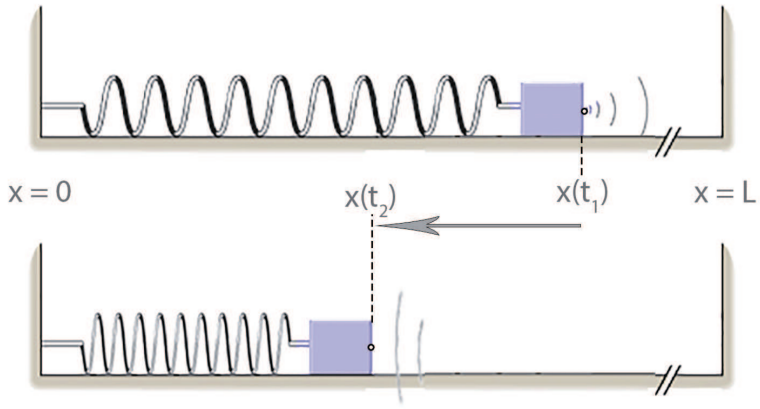

FIGURE 6. Mass-Spring-Sonar system

Computation time for this example on a single core 3.2 Ghz processor is on average 7 ms for Fig. 5(d) and 55 ms for Fig. 5(h).

3.2. Example 2: Non-linear mass-spring system

Let us now consider an example based on the famous academic problem that is the mass-spring system presented in Fig.6. In actual systems, there is a progressive stiffening or weakening of the spring as it is elongated or compressed. This causes the response of the system to be nonlinear. Its dynamic is given by Newton's second law.

$$m.\ddot{x} + \gamma.\dot{x} + \kappa.x - \beta x^3 = 0 \tag{3.11}$$

where β is the stiffness of the spring, m is the mass, x is the displacement, κ is the spring elasticity and γ is the damping constant.

Knowing the initial conditions, this system could be easily solved using standard numerical methods. However, the particularity of our approach is to consider that we do not know the initial conditions. In return, we equip the mass with a sonar that sends a ping every second. We consider the sound wave sent at t_1 to travel at $c = 100m.s^{-1}$ to the right wall, where it is reflected and received back to the sonar at t_2. In our simulation, we place the right wall at $L = 10m$. This means that the mass moves significantly between the emission and the reception of pings. Each ping is represented by an inter-temporal equation:

$$\begin{aligned} (L - x(t_1)) + (L - x(t_2)) &= c.(t_2 - t_1) \\ \Leftrightarrow x(t_2) + x(t_1) &= 2L - c.(t_2 - t_1) \end{aligned} \tag{3.12}$$

To our knowledge, we cannot easily solve this nonlinear system with standard numerical methods. We thus consider this problem as a state estimation problem of a nonlinear delayed system and use our constraint propagation approach defined in the previous section. We first rewrite the system as state equations :

$$\begin{aligned} &\frac{d}{dt}\begin{pmatrix} x \\ \dot{x} \end{pmatrix} = \begin{pmatrix} \dot{x} \\ \frac{(\beta x^2 - \kappa).x - \gamma.\dot{x}}{m} \end{pmatrix} \quad \text{(Evolution equation)} \\ &x(t_i) + x(t_j) = 2L - c.(t_i - t_j) \quad \text{(Observation equations)} \end{aligned} \tag{3.13}$$

where the t_j are sending times and t_i receiving times. Each ping represents an asynchronous constraint that translates into a contractor for $[x]$ and $[\dot{x}]$.

We developed a simulator using MATLAB's Ode45 and defined an initial state with the evolution equations. The result is presented Fig.7. We then simulated sonar pings every seconds. We also simulated errors in the measurements up to $+/-0.05m$ to stay consistent with guaranteed results. We also purposefully "lost" a few pings to simulate sporadic measurements. For the sake of transparency, we used $m = 8kg$, $\gamma = 1$, $\kappa = 2$, $\beta = -0.5$ and $L = 100m$. Initial conditions are $x_0 = 10m$ and $\dot{x}_0 = 0m.s^{-1}$. The simulated sonar output has the following form :

$$\begin{aligned} x(3.00) + x(3.48) &= 34.33 \\ x(9.00) + x(9.54) &= 15.46 \\ x(12.00) + x(12.55) &= 13.06 \\ x(20.00) + x(20.51) &= 24.38 \\ &etc. \end{aligned} \tag{3.14}$$

As we are using a simple Euler method, 3.13 numerically becomes :

$$\begin{aligned} x(k+1) &= x(k) + dt.\dot{x}(k) \\ \dot{x}(k+1) &= \dot{x}(k) + dt.\frac{(\beta x^2(k) - \kappa).x(k) - \gamma.\dot{x}(k)}{m} \\ x(i) + x(j) - l &= 0 \end{aligned} \tag{3.15}$$

where $l = 2L - c.(i-j)$ which is a small measured interval. Therefore we have :

$$\begin{aligned} x(j) &= x(j-1) + dt.\dot{x}(j-1) \\ &= x(j-2) + dt.\dot{x}(j-2) + dt.\dot{x}(j-1) \\ &= x(j-3) + dt.\dot{x}(j-3) + dt.\dot{x}(j-2) + dt.\dot{x}(j-1) \\ &= x(i) + dt.\sum_{k=i}^{j-1} \dot{x}(k) \end{aligned} \tag{3.16}$$

and

$$x(j) - x(i) = v \tag{3.17}$$

$$x(j) + x(i) = l \tag{3.18}$$

where $v = \sum_{k=i}^{j-1} \dot{x}(k)$ which is equivalent to

$$\begin{aligned} x(i) &= \frac{1}{2}(l - v) \\ x(j) &= \frac{1}{2}(l + v) \end{aligned} \tag{3.19}$$

To sum up, we have the following contractors :
∘ *State equation*. Tubes $[x]$ and $[\dot{x}]$, contracted using the evolution equation from 3.13:

$$[\dot{x}(t)] = [\dot{x}(t)] \cap \int_{\tau=0}^{t} \left[\frac{(\beta x^2 - k).x - c.\dot{x}}{m} \right] (\tau).d\tau \tag{3.20}$$

$$[x(t)] = [x(t)] \cap \int_{\tau=0}^{t} [\dot{x}] (\tau).d\tau$$

∘ *Movements*. Using the speed tube $[\dot{x}]$ of the mass, we can compute all the $n_{\max}$ variables $v_n = dt.\sum_{k=i_n}^{j_n-1} \dot{x}(k)$, where n corresponds to the n-th sonar emission and v_n corresponds to the movement of the mass between the n-th sonar emission and n-th sonar reception. We have the following contractor :

$$[v_n] = [v_n] \cap dt.\sum_{k=i_n}^{j_n-1} [\dot{x}](k) \tag{3.21}$$

∘ *Positions.* From 3.19, we get two more contractors:

$$[x](i_n) = [x](i_n) \cap \frac{1}{2}([l_n] - [v_n]) \tag{3.22}$$
$$[x](j_n) = [x](j_n) \cap \frac{1}{2}([l_n] + [v_n])$$

We present in table 3 and 4 the $CONTRACT$ and $STRANGLE$ algorithms modified specifically for this example. Given an initial tube for $[x]$ and $[\dot{x}]$, the algorithm $STRANGLE2$ will apply the contractors 3.21 and 3.22, 3.20. The algorithm imposes intervals $[box](1)$ to $[x]$ and $[box](2)$ to $[\dot{x}]$ when the time reaches k_0 and propagates the intervals forward then backward along the tubes according to the states equations. If the contracted tubes contain empty intervals, the given box at k_0 does not contain the solution. In this case, we discard the given box (white area on Fig.9). If the contracted tubes have no empty interval, then the tubes might contain the solution, and we keep it stored in memory (blue area on Fig.9). In the end, we take the union of all non-empty tubes. The union contains the solution.

Table 3 presents the algorithm where $\phi_i([x](k), [\dot{x}](k))$ is the numerical estimation method of the state equation 3.13. For instance, using an Euler method, $\phi_2([x](k), [\dot{x}](k)) = [x](k) + dt.\frac{(\beta[x]^2(k)-\kappa).[x](k)-\gamma.[\dot{x}](k)}{m}$. Again, the result can be guaranteed using more complex methods and/or VNODE. Our algorithm is only guaranteed in respect to the numerical estimation method used. Because of the wrapping effect on each step of the propagation, the tubes $[x]$ and $[\dot{x}]$ rapidly explode. It is therefore impossible to contract efficiently the tubes from beginning ($k_0 = 0$) to end ($k_0 = \bar{k}$) in one passing. We therefore bisect box until a required width is reached and use $CONTRACT2$ to contract the tubes. We define $box = ([x](k_0), [\dot{x}](k_0))$ and apply the iterative algorithm proposed in table 4 to bisect, propagate and take the union of the non-empty tubes until the required precision is reached.

Fig.8 shows the result of the algorithm. (1p) represents the position tube $[x]$ after applying the contractors 3.22 and (1v) represents the speed tube $[\dot{x}]$. (2p) and (2v) represent the same tubes after applying $STRANGLE2$ for k_0=0s. (3p) and (3v) for k_0=0s, k_0=25s and k_0=50s. The final result is shown in (4p) and (4v) for which $STRANGEL2$ is applied every $t\%5 = 0s$.

	Algorithm $C_{CONTRACT2}$ (in : k_0, $[box]$, inout: $[x]$, $[\dot{x}]$)
1	for $k := 0$ to $\bar{k}$
2	if($k == k_0$)
3	$[x](k) = [box](1)$;
4	$[\dot{x}](k) = [box](2)$;
5	endif
6	$[x](k+1) = [x](k+1) \cap [\phi_1]([x](k), [\dot{x}](k))$;
7	$[\dot{x}](k+1) = [\dot{x}](k+1) \cap [\phi_2]([x](k), [\dot{x}](k))$;
8	if($k == i_n$ or $k == j_n$)
9	$[v_n] = [v_n] \cap dt. \sum_{p=i_n}^{j_n-1} [\dot{x}](p)$;
10	$[x](i_n) = [x](i_n) \cap \frac{1}{2}([l_n] - [v_n])$;
11	$[x](j_n) = [x](j_n) \cap \frac{1}{2}([l_n] + [v_n])$;
12	endif
13	endfor
14	for $k := \bar{k}$ to 0
15	if($k == k_0$)
16	$[x](k) = [box](1)$;
17	$[\dot{x}](k) = [box](2)$;
18	endif
19	$[x](k) = [x](k) \cap [\tilde{\phi}_1]([x](k+1), [\dot{x}](k+1))$;
20	$[\dot{x}](k) = [\dot{x}](k) \cap [\tilde{\phi}_2]([x](k+1), [\dot{x}](k+1))$;
21	if($k == i_n$ or $k == j_n$)
22	$[v_n] = [v_n] \cap dt. \sum_{p=i_n}^{j_n-1} [\dot{x}](p)$;
23	$[x](i_n) = [x](i_n) \cap \frac{1}{2}([l_n] - [v_n])$;
24	$[x](j_n) = [x](j_n) \cap \frac{1}{2}([l_n] + [v_n])$;
25	endif
26	endfor

TABLE 3. Contract algorithm for Ex. 2

Notice that if the inter temporal constraints are posed such that the problem is infeasible, the solution set becomes empty. Computation time for this example on a single core 3.2 Ghz processor is on average 2.5s.

3.3. Example 3: Group of AUVs

For the last example, let us consider a group of robotic *Autonomous Underwater Vehicles* (AUVs). When not submerged, the AUVs are able to use the *Global Positioning System* to accurately compute their position. However, when they are underwater, they can no longer use the GPS and have to estimate their position using their state equations. To illustrate the method, we developed a 3D simulator that generates a set of data from a simulated group of 6 robots following given trajectories.

	Algorithm $C_{STRANGLE2}$ (in : ϵ, $[k0]$, $[box]$, inout : $[x]$, $[\dot{x}]$)
1	if($box.width > \epsilon$)
2	$Bisect([box], [subbox1], [subbox2])$;
3	$([z], [\dot{z}]) = CONTRACT2([subbox1], [z], [\dot{z}])$
4	if($[z]$ and $[\dot{z}]$ have no empty interval)
5	$CONTRACT2([subbox1], [z], [\dot{z}])$;
6	endif
7	$([z], [\dot{z}]) = CONTRACT2([subbox2], [z], [\dot{z}])$
8	if($[z]$ and $[\dot{z}]$ have no empty interval)
9	$CONTRACT2([subbox2], [z], [\dot{z}])$;
10	endif
11	else
12	$([z], [\dot{z}]) = CONTRACT2([box], [z], [\dot{z}])$
13	if($[z]$ and $[\dot{z}]$ have no empty interval)
14	$[x] = [x] \sqcup [z]$;
15	$[\dot{x}] = [\dot{x}] \sqcup [\dot{z}]$;
16	endif
17	endif
18	return $([x], [\dot{x}])$

TABLE 4. Strangle algorithm for Ex. 2

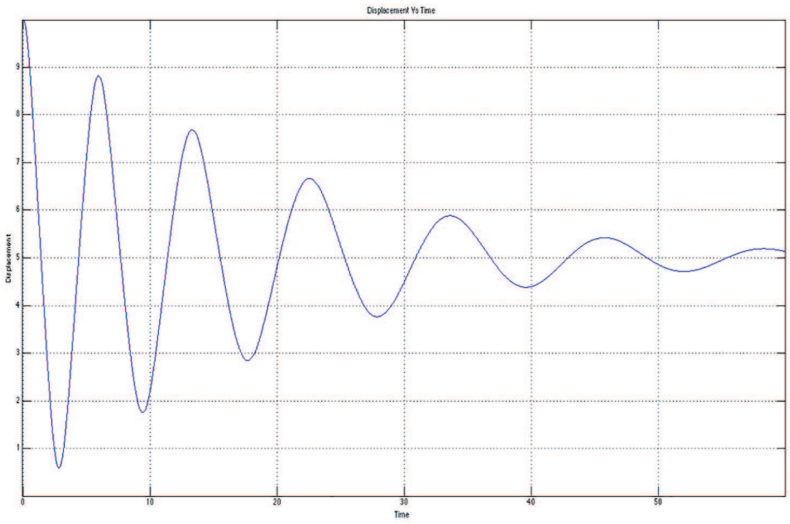

FIGURE 7. MATLAB's Ode45 solution.

$$Vi \in \{1,2,3\}, \hat{\mathbf{x}}_i = \begin{pmatrix} 100\sin t \\ 100\cos t \\ (10\cos t) - 10 \end{pmatrix} \tag{3.23}$$

$$Vi \in \{4,5,6\}, \hat{\mathbf{x}}_i = \begin{pmatrix} 100\sin t \\ 100\cos t \\ (10\cos t) - 30 \end{pmatrix} \tag{3.24}$$

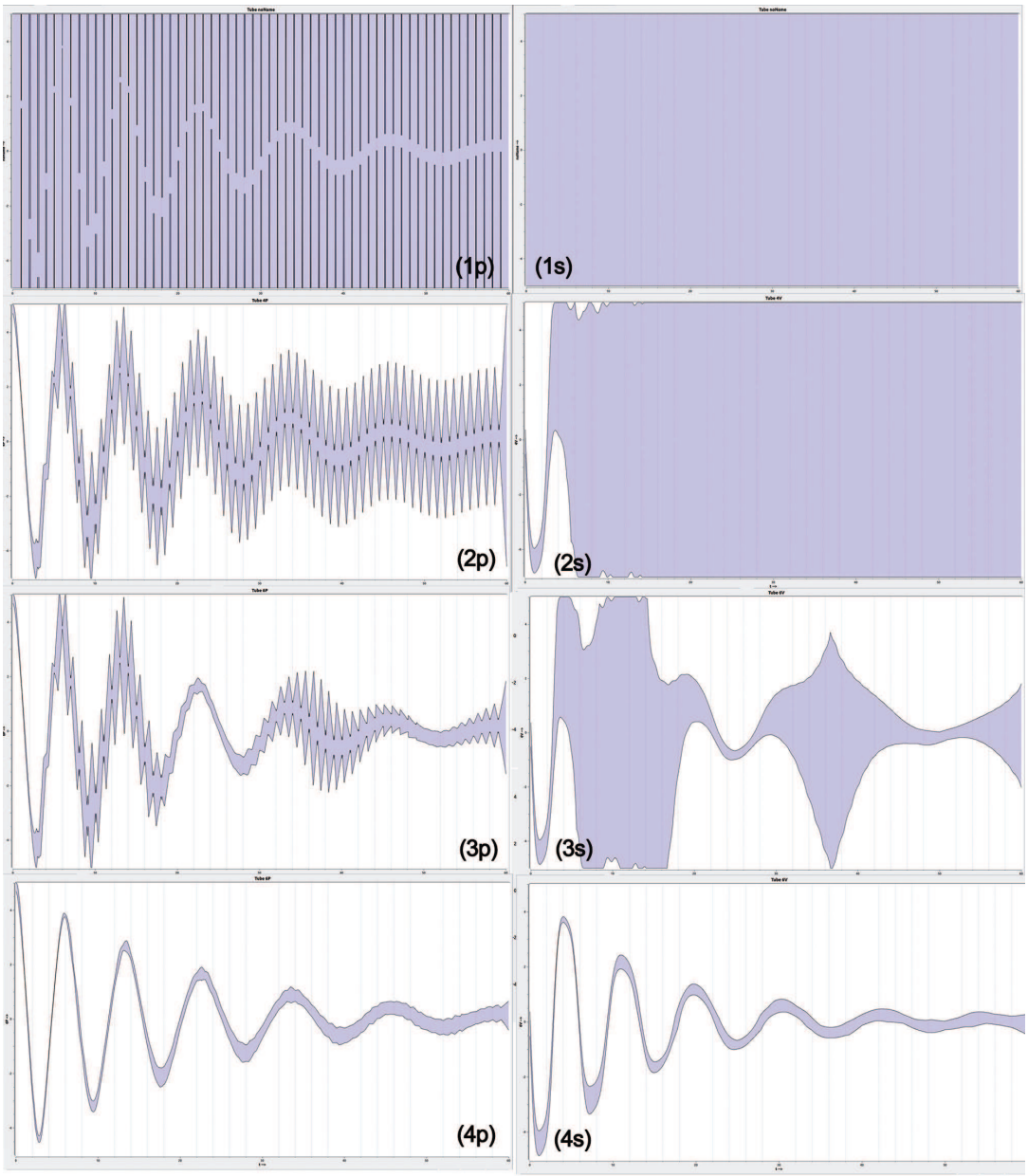

FIGURE 8. Successive results of the $STRANGLE2$ algorithm.

We assume that the state of each AUV is described by the following equations :

$$\begin{aligned} \dot{x}_1 &= u_1 \cos u_2 \\ \dot{x}_2 &= u_1 \sin u_2 \end{aligned} \tag{3.25}$$

where the speed u_1 and heading u_2 are measured every $0.01s$ for a duration of $100s$. The initial state is not known. Because of the noise in the measurements, the uncertainty of the state is increasing over time. The estimated position is represented by a box enclosing the real position

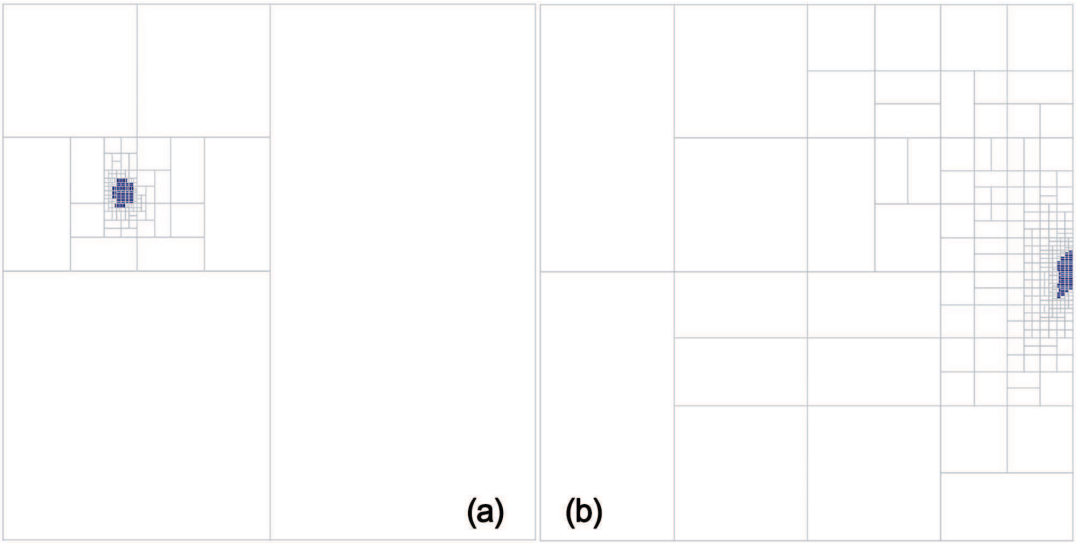

FIGURE 9. Result on a $x, \dot{x}$ plane of the $STRANGLE2$ bisections for (a) t=10s and (b) t=0s. The white boxes represent tubes that become empty at some point. The blue boxes represent tubes that have no empty interval for all t. The solution is in the union of all these tubes.

of the AUV. This box is obviously very thin when the AUV uses the GPS, then gets bigger and bigger underwater. The idea is to contract these boxes when two or more AUVs are in range of communication with each other. We equipped each submarine with a sonar that can broadcast the estimated position box of the AUV every second. The originality of this example is to consider that the displacement of the robot while the sonar wave is moving is superior to the precision of the localization. Therefore we cannot suppose that we measure a true distance between AUVs at the same time, but between AUVs at different times. Note that the communication is one-way only and does not have to be synchronized between the AUVs. When AUV i received at t_i a sonar ping from robot j emitted at t_j, we get a constraint of the form :

$$||\mathbf{p}_i(t_i) - \mathbf{p}_j(t_j)|| = c.(t_i - t_j) \tag{3.26}$$

where c is the celery of the sound in water and $\mathbf{p}_i$ is the position vector of a submarine i in x-y-z coordinates. This way, we have a system of constraints similar to example 2. Fig.10 presents the 3D simulator developed to generate the constraints for each AUV.

$$||p_1(1.00) + p_2(1.32)|| = 128.44 \tag{3.27}$$
$$||p_1(9.00) + p_3(9.74)|| = 284.38 \tag{3.28}$$
$$||p_3(9.00) + p_5(9.15)|| = 72.49 \tag{3.29}$$
$$||p_5(17.00) + p_4(18.13)|| = 682.98 \tag{3.30}$$
$$etc.$$

Using the same approach than for example 2, we can solve this problem using constraint propagation on tubes. Fig. 11 shows the result of the contractions for the position x of one AUV.

Computation time for this example on a single core 3.2 Ghz processor is on average 15 ms for all AUVs. Finally, we can further consider a more realistic model where the clocks of the AUVs are

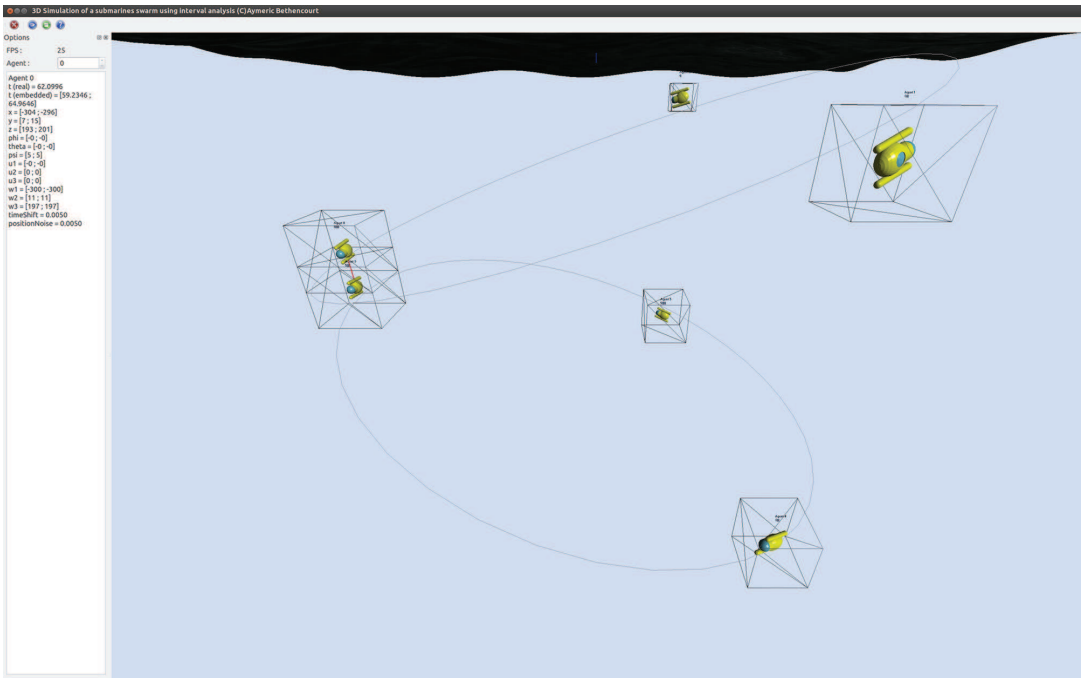

FIGURE 10. 3D simulation of a group of 6 AUVs.

unsynchronized, meaning that the constraints are defined on uncertain times. [30] shows that it is still possible to easily solve the problem.

4. Conclusion

Solving nonlinear systems involving differential equations is a difficult problem, especially when the initial conditions are unknown or when the problem is ill-conditioned, e.g. involving inter-temporal measurements. To numerically solve this class of problems, this paper has first introduced the notion of tube which encompasses the informations needed to guarantee associations upon trajectories. Then, an arithmetic was developed around this notion, and a contractor-based approach has been followed. As a result, a method has been proposed to contract tubes that enclose the solution. Further work will include the study of outliers and the study of the scalability of the approach when applied to a swarm of hundreds of AUVs [31]. As most interval-based methods, this approach can be combined with probabilistic methods [10] and made robust with respect to outliers by relaxing a given number of constraints [12].

Finally, in order to share our research with the community, we integrated tubes, their properties, operators and minimal contractors presented below with *IBEX (Interval Based EXplorer)*, a powerful library for interval computation. You can download *IBEX* and all the source codes for the examples presented in this article on http://AymericBethencourt.com/tubes/

References

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

[2] L. Jaulin, M. Kieffer, O. Didrit, and E. Walter, "Applied interval analysis," *Springer-Verlag*, 2001.

[3] S. Thrun, W. Bugard, and D. Fox, *Probabilistic Robotics*. Cambridge, M.A.: MIT Press, 2005.

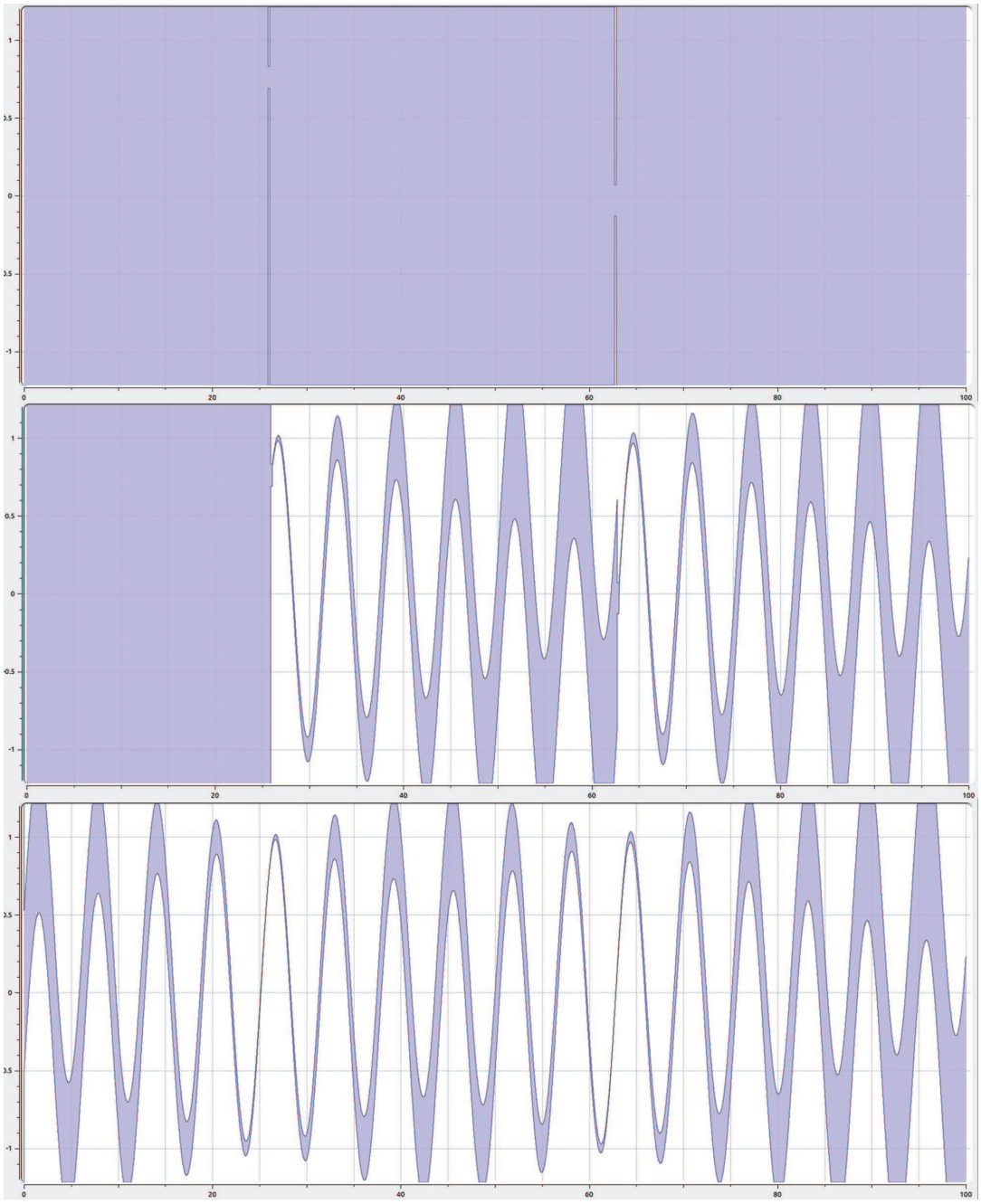

FIGURE 11. Result of the contractions for $[x]$, (a) after using position contraction upon receiving a sonar communication from another robot at t=26s and 63s, (b) after propagating the state equation from 0 to $\overline{k}$ and (c) from $\overline{k}$ to 0.

[4] L. Jaulin, "Range-only SLAM with occupancy maps; A set-membership approach," *IEEE Transaction on Robotics*, vol. 27, no. 5, pp. 1004–1010, 2011.

[5] F. L. bars, A. Bertholom, J. Sliwka, and L. Jaulin, "Interval slam for underwater robots; a new experiment," in *NOLCOS 2010*, Italy, 2010.

[6] C. Drocourt, L. Delahoche, B. M. E. Brassart, and A. Clerentin, "Incremental construction of the robot's environmental map using interval analysis," *Global Optimization and Constraint Satisfaction: Second International Workshop, COCOS 2003*, vol. 3478, pp. 127–141, 2005.

[7] A. Bethencourt and L. Jaulin, "3d reconstruction using interval analysis on the kinect device coupled with an imu," *International Journal of Advanced Robotic Systems*, vol. 10, 2013.

[8] N. Delanoue, L. Jaulin, and B. Cottenceau, "Using interval arithmetic to prove that a set is path-connected," *Theoretical Computer Science, Special issue: Real Numbers and Computers*, vol. 351, no. 1, pp. 119–128, 2006.

[9] J. Aubin and H. Frankowska., *Set-Valued Analysis*. Birkhäuser, Boston, 1990.

[10] F. Abdallah, A. Gning, and P. Bonnifait, "Box particle filtering for nonlinear state estimation using interval analysis," *Automatica*, vol. 44, no. 3, pp. 807–815, 2008.

[11] G. Chabert and L. Jaulin, "Contractor Programming," *Artificial Intelligence*, vol. 173, pp. 1079–1100, 2009.

[12] V. Drevelle and P. Bonnifait, "High integrity gnss location zone characterization using interval analysis," in *ION GNSS*, 2009.

[13] M. Berz and K. Makino, "Verified integration of odes and flows using differential algebraic methods on high-order taylor models," *Reliable Computing*, vol. 4, no. 3, pp. 361–369, 1998.

[14] B. A. Davey and H. A. Priestley, *Introduction to Lattices and Order*. (ISBN 0521784514): Cambridge University Press, 2002.

[15] S. Gollamudi, S. Nagaraj, S. Kapoor, and Y.-F. Huang, "Set-membership state estimation with optimal bounding ellipsoids," in *Int. Symposium on Information Theory and its Applications*, 1996.

[16] A. Kurzhanski and I. Valyi, *Ellipsoidal Calculus for Estimation and Control*. Boston, MA: Birkhäuser, 1997.

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

[18] C. Aubry, R. Desmare, and L. Jaulin, "Loop detection of mobile robots using interval analysis," *Automatica*, vol. 49, no. 2, pp. 463–470, 2013.

[19] I. Araya, B. Neveu, and G. Trombettoni, "Exploiting Common Subexpressions in Numerical CSPs," in *Proc. CP, Constraint Programming*, LNCS 5202, 2008, pp. 342–357.

[20] M. Ceberio and L. Granvilliers, "Solving nonlinear systems by constraint inversion and interval arithmetic," in *Artificial Intelligence and Symbolic Computation*, vol. 1930, LNCS 5202, 2001, pp. 127–141.

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

[22] F. C. Schweppe, "Recursive state estimation: unknown but bounded errors and system inputs," *"IEEE Transactions on Automatic Control"*, vol. 13, no. 1, pp. 22–28, 1968.

[23] L. Jaulin, "Nonlinear bounded-error state estimation of continuous-time systems," *Automatica*, vol. 38, pp. 1079–1082, 2002.

[24] J. Bravo, T. Alamo, and E. Camacho, "Robust mpc of constrained discrete-time nonlinear systems based on approximated reachable sets," *Automatica*, vol. 42, pp. 1745–1751, 2006.

[25] C. Combastel, "A state bounding observer for uncertain non-linear continuous-time systems based on zonotopes," in *CDC-ECC '05*, 2005.

[26] E. A. Cross and I. M. Mitchell, "Level set methods for computing reachable sets of systems with differential algebraic equation dynamics," in *American Control Conference, 2008*. IEEE, 2008, pp. 2260–2265.

[27] M. Althoff and B. Krogh, "Reachability analysis of nonlinear differential-algebraic systems," 2013.

[28] N. Nedialkov, "Interval tools for odes and daes," in *Scientific Computing, Computer Arithmetic and Validated Numerics, 2006. SCAN 2006. 12th GAMM - IMACS International Symposium on*, 2006, pp. 4–4.

[29] O. Bouissou, A. Chapoutot, and A. Djoudi, "Enclosing temporal evolution of dynamical systems using numerical methods," in *5th NASA Formal Methods Symposium, NFM 2013*, NASA Ames Research Center, Moffett Field, CA, USA, 2013.

[30] A. Bethencourt and L. Jaulin, "Cooperative localization of underwater robots with unsynchronized clocks," *Paladyn, Journal of Behavioral Robotics.*, vol. 4, no. 4, pp. 233–244, 2013.

[31] A. Bethencourt, "Interval analysis for swarm localization. application to underwater robotics." PhD dissertation, Universite de Breatgne Occidental, Brest, France, 2014.