Interval Robotics, 2011-2012

Luc Jaulin

February 22, 2012

Part I

Interval methods

Chapter 1

Intervals

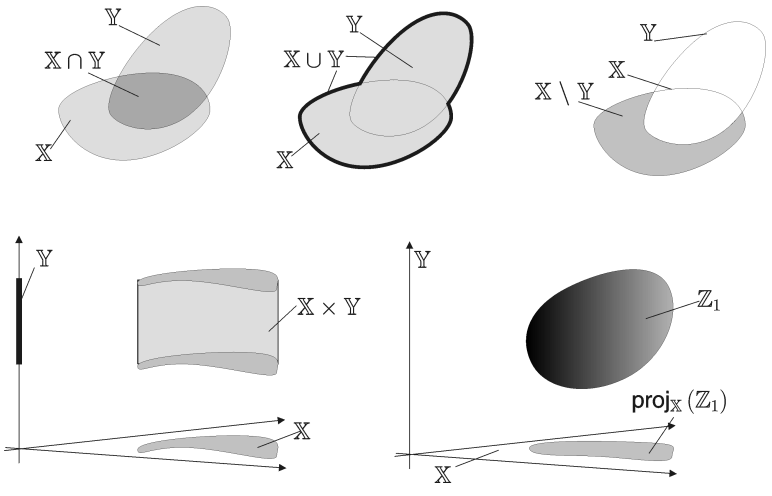
1.1 Basic notions on set theory

Consider two sets $\mathbb{X}$ and $\mathbb{Y}$. We define the following operations.

$$\begin{array}{lcll} \mathbb{X} \cap \mathbb{Y} & \triangleq & \{x \mid x \in \mathbb{X} \text{ and } x \in \mathbb{Y}\} & \text{(intersection)} \\ \mathbb{X} \cup \mathbb{Y} & \triangleq & \{x \mid x \in \mathbb{X} \text{ or } x \in \mathbb{Y}\} & \text{(union)} \\ \mathbb{X} \setminus \mathbb{Y} & \triangleq & \{x \mid x \in \mathbb{X} \text{ and } x \notin \mathbb{Y}\} & \text{(deprivation)} \\ \mathbb{X} \times \mathbb{Y} & \triangleq & \{(x, y) \mid x \in \mathbb{X} \text{ and } y \in \mathbb{Y}\} & \text{(Cartesian product)} \end{array}$$

If $\mathbb{Z} = \mathbb{X} \times \mathbb{Y}$, then the *projection* of a subset $\mathbb{Z}_1$ of $\mathbb{Z}$ onto $\mathbb{X}$ (with respect to $\mathbb{Y}$) is defined as

$$\text{proj}_{\mathbb{X}}(\mathbb{Z}_1) \triangleq \{x \in \mathbb{X} \mid \exists y \in \mathbb{Y} \text{ such that } (x, y) \in \mathbb{Z}_1\}.$$



Consider a function $f : \mathbb{X} \to \mathbb{Y}$. If $\mathbb{X}_1 \subset \mathbb{X}$, the *direct image* of $\mathbb{X}_1$ by f is

$$f(\mathbb{X}_1) \triangleq \{f(x) \mid x \in \mathbb{X}_1\}.$$

If $\mathbb{Y}_1 \subset \mathbb{Y}$, the *reciprocal image* of $\mathbb{Y}_1$ by f is

$$f^{-1}(\mathbb{Y}_1) \triangleq \{x \in \mathbb{X} \mid f(x) \in \mathbb{Y}_1\}.$$

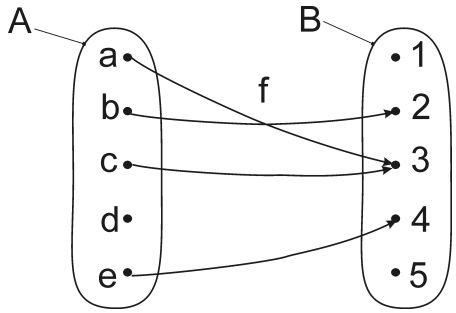
If $\emptyset$ denotes the empty set, then the previous definitions imply that

$$f(\emptyset) = f^{-1}(\emptyset) = \emptyset.$$

It is trivial to show that if $\mathbb{X}_1$ and $\mathbb{X}_2$ are subsets of $\mathbb{X}$ and if $\mathbb{Y}_1$ and $\mathbb{Y}_2$ are subsets of $\mathbb{Y}$, then

$$\begin{gathered}
f(\mathbb{X}_1 \cap \mathbb{X}_2) \subset f(\mathbb{X}_1) \cap f(\mathbb{X}_2), \\
f(\mathbb{X}_1 \cup \mathbb{X}_2) = f(\mathbb{X}_1) \cup f(\mathbb{X}_2), \\
f^{-1}(\mathbb{Y}_1 \cap \mathbb{Y}_2) = f^{-1}(\mathbb{Y}_1) \cap f^{-1}(\mathbb{Y}_2), \\
f^{-1}(\mathbb{Y}_1 \cup \mathbb{Y}_2) = f^{-1}(\mathbb{Y}_1) \cup f^{-1}(\mathbb{Y}_2), \\
f\left(f^{-1}(\mathbb{Y})\right) \subset \mathbb{Y}, \\
f^{-1}(f(\mathbb{Y})) \supset \mathbb{Y}, \\
\mathbb{X}_1 \subset f^{-1}(\mathbb{Y}) \Rightarrow f^{-1}(f(\mathbb{X}_1)) \supset \mathbb{X}_1, \\
\mathbb{X}_1 \subset \mathbb{X}_2 \Rightarrow f(\mathbb{X}_1) \subset f(\mathbb{X}_2), \\
\mathbb{Y}_1 \subset \mathbb{Y}_2 \Rightarrow f^{-1}(\mathbb{Y}_1) \subset f^{-1}(\mathbb{Y}_2), \\
\mathbb{X} \subset \mathbb{Y}_1 \times \mathbb{Y}_2 \Rightarrow \mathbb{X} \subset \mathrm{proj}_{\mathbb{Y}_1}(\mathbb{X}) \times \mathrm{proj}_{\mathbb{Y}_2}(\mathbb{X}).
\end{gathered}$$

For instance, if f is defined as follows



Example 1 *then*

$$\begin{aligned}
f(A) &= \{2,3,4\} = \mathrm{Im}(f). \\
f^{-1}(B) &= \{a,b,c,e\} = dom(f). \\
f^{-1}(f(A)) &= \{a,b,c,e\} \subset A \\
f^{-1}(f(\{b,c\})) &= \{a,b,c\}.
\end{aligned}$$

If $f(x) = x^2$, *then*

$$\begin{aligned}
f([2,3]) &= [4,9] \\
f^{-1}([4,9]) &= [-3,-2] \cup [2,3].
\end{aligned}$$

This is consistent with the property

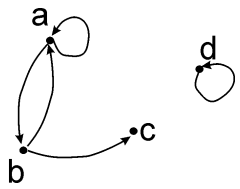
$$f^{-1}(f(\mathbb{Y})) \supset \mathbb{Y}.$$

Relation (or binary constraint)

A relation in $\mathbb{X}$ is a subset of $\mathbb{X} \times \mathbb{X}$. Assume for instance that $\mathbb{X} = \{a, b, c, d\}$, the set

$$\mathcal{C} = \{(a, a), (a, b), (b, a), (b, c), (d, d)\}$$

is a relation or a binary constraint in $\mathbb{X}$. Since $\mathbb{X}$ is finite, it can be represented a directed graph.



or by the matrix

$$\mathbf{C} = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

The matrix notation can be help to compute with relation if the Boolean algebra is used. For instance, the smallest symmetric relation enclosing $\mathbf{C}$ is

$$\mathbf{S} = \mathbf{C} + \mathbf{C}^{\mathrm{T}} = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} + \begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

The smallest equivalence relation enclosing $\mathbf{C}$ is $\left(\mathbf{C} + \mathbf{C}^{\mathrm{T}}\right)^*$ where

$$\mathbf{A}^* = \mathbf{I} + \mathbf{A} + \mathbf{A}^2 + \mathbf{A}^3 + \ldots$$

Here, we get

$$\left(\mathbf{C} + \mathbf{C}^{\mathrm{T}}\right)^* = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} + \begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} + \begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}^2 + \cdots = \begin{pmatrix} 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

Properties of relations

Consider a binary relation $\mathcal{C}$ over a set $\mathbb{X}$.

- $\mathcal{C}$ is *reflexive*: for all x in $\mathbb{X}$ it holds that $(x, x) \in \mathcal{C}$. For example, "greater than or equal to" is a reflexive relation but "greater than" is not.
- $\mathcal{C}$ is *symmetric* if $(x, y) \in \mathcal{C} \Rightarrow (y, x) \in \mathcal{C}$. "Is a blood relative of" is a symmetric relation, because x is a blood relative of y if and only if y is a blood relative of x.
- $\mathcal{C}$ is *antisymmetric*: for all x, y in $\mathbb{X}$ it holds that if $(x, y) \in \mathcal{C}$ and $(y, x) \in \mathcal{C}$ then $x = y$. "Greater than or equal to" is an antisymmetric relation, because if $x \geq y$ and $y \geq x$, then $x = y$.

- $\mathcal{C}$ is *transitive* if $(x,y) \in \mathcal{C}$, $(y,z) \in \mathcal{C} \Rightarrow (x,z) \in \mathcal{C}$. "Is an ancestor of" is a transitive relation, because if x is an ancestor of y and y is an ancestor of z, then x is an ancestor of z.
- $\mathcal{C}$ is *total* if $\forall (x,y) \in \mathbb{X}^2$, $(x,y) \in \mathcal{C}$ or $(y,x) \in \mathcal{C}$. "Is greater than or equal to" is an example of a total relation (this definition for total is different from the one in the previous section).
- $\mathcal{C}$ is an *equivalence relation* if it is reflexive, antisymmetric and transitive.

1.2 Intervals

1.2.1 Intervals in lattices

In order theory in mathematics, a set $\mathbb{X}$ with a binary relation $\leq$ on its elements that is reflexive, antisymmetric and transitive is described as a partially ordered set. If the binary relation is antisymmetric, transitive and also total, then the set is a totally ordered set. A *lattice* $(\mathbb{X}, \leq)$ is a partially ordered set, closed under least upper and greatest lower bounds (see [13], for more details). The least upper bound (or infimum) of x and y is called the *join* and is denoted by $x \vee y$. The greatest lower bound (or *supremum*) is called the *meet* and is written as $x \wedge y$.

Example 2 *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

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

where $x_i \wedge y_i = \min(x_i, y_i)$ and $x_i \vee y_i = \max(x_i, y_i)$. ■

A (possibly empty) subset $\mathbb{D}$ of $\mathbb{X}$ is a *sublattice* of $\mathbb{X}$ if

$$\forall x \in \mathbb{D}, \forall y \in \mathbb{D}, x \vee y \in \mathbb{D} \text{ and } x \wedge y \in \mathbb{D}.$$

Example 3 *The set*

$$\mathbb{D}_1 \triangleq \left\{(x_1, x_2) \in \mathbb{R}^2 | x_1 - x_2 \leq 0\right\}$$

is a sublattice of $\mathbb{R}^2$ whereas the set

$$\mathbb{D}_2 \triangleq \left\{(x_1, x_2) \in \mathbb{R}^2 | x_1 + x_2 \leq 0\right\}$$

is not a sublattice of $\mathbb{R}^2$ (for instance, $(-1,1)$ and $(1,-1)$, both belong to $\mathbb{D}_2$, whereas $(-1,1) \vee (1,-1) = (1,1)$ is not an element of $\mathbb{D}_2$). ■

A lattice $\mathbb{X}$ is *complete* if for all (finite of infinite) subsets $\mathbb{A}$ of $\mathbb{X}$, the least upper bound (denoted $\wedge\mathbb{A}$) and the greatest lower bound (denoted $\vee\mathbb{A}$) belong to $\mathbb{A}$. When a lattice $\mathbb{X}$ is not complete, it is possible to add new elements (corresponding the supremum or infimum of infinite subsets of $\mathbb{X}$ that do not belong to $\mathbb{X}$) to make it complete. By convention, for the empty set, we set $\wedge\emptyset = \vee\mathbb{X}$ and $\vee\emptyset = \wedge\mathbb{X}$.

Example 4 *The set* $\mathbb{R}$ *is not a complete sublattice whereas* $\overline{\mathbb{R}} = \mathbb{R} \cup \{-\infty, \infty\}$ *is.* ■

Consider a complete lattice $(\mathbb{X}, \leq)$. A *closed interval* (or *interval* for short) $[x]$ of $\mathbb{X}$ is a subset of $\mathbb{X}$ which satisfies

$$[x] = \{x \in \mathbb{X} \mid \wedge [x] \leq x \leq \vee [x]\}.$$

Note that since $\mathbb{X}$ is complete, $x^- = \wedge[x]$ and $x^+ = \vee[x]$ belong to $\mathbb{X}$. The set of all intervals of $\mathbb{X}$ will be denoted by $\mathbb{IX}$. An interval is a sublattice of $\mathbb{X}$. An interval $[x]$ of $\mathbb{X}$ will also be denoted by

$$[x] = \left[x^-, x^+\right] = [\wedge[x], \vee[x]]_{\mathbb{X}}.$$

The set $\mathbb{X}$ is an interval of $\mathbb{X}$ which can be denoted by

$$\mathbb{X} = [\bot, \top] \text{ where } \left\{ \begin{array}{lcl} \bot & = & \wedge \mathbb{X} \\ \top & = & \vee \mathbb{X}. \end{array} \right.$$

The empty set $\emptyset$ is also an interval of $\mathbb{X}$ which can be denoted by

$$\emptyset = [\top, \bot].$$

Example 5 *The sets* $\emptyset = [\infty, -\infty]_{\mathbb{R}}$; $\mathbb{R} = [-\infty, \infty]_{\mathbb{R}}$; $[0,1]_{\mathbb{R}}$ *and* $[0, \infty]_{\mathbb{R}}$ *are intervals of* $\mathbb{R}$. *The set* $\{2,3,4,5\} = [2,5]_{\mathbb{N}}$ *is an interval of the set of integers* $\mathbb{N}$. *The set* $\{4,6,8,10\} = [4,10]_{2\mathbb{N}}$ *is an interval of* $2\mathbb{N}$, *the set of even integers.* ■

Interval intersection. The intersection of two intervals $[x]$ and $[y]$, is also an interval. In $\mathbb{IX}$, the intersection is closed with respect to the intersection.

Interval hull. The *interval hull* (or *hull*, for short) of a subset $\mathbb{A}$ of $\mathbb{X}$ is the smallest interval of $\mathbb{X}$ which contains $\mathbb{A}$, *i.e.*,

$$\text{hull}_{\mathbb{X}}(\mathbb{A}) \triangleq \bigcap \{[x] \in \mathbb{IX} \mid \mathbb{A} \subset [x]\} = [\wedge \mathbb{A}, \vee \mathbb{A}]_{\mathbb{X}}.$$

Width. If the lattice $(\mathbb{X}, \leq)$ is equipped with a distance d, we define the *width* of an interval $[x]$ as

$$w([x]) = \left\{ \begin{array}{ll} d(x^-, x^+) & \text{if } [x] \neq \emptyset \\ -\infty & \text{if } [x] = \emptyset. \end{array} \right.$$

Interval union. For two intervals $[x]$ and $[y]$, the interval union is defined by

$$[x] \sqcup [y] = [[x] \cup [y]].$$

Bisection.. Given a metric lattice $(\mathbb{X}, \leq, d)$. A *bisection* is a function

$$\beta : \begin{array}{ccc} \mathbb{IX} & \to & \mathbb{IX} \times \mathbb{IX} \\ [x] & \mapsto & ([x_1], [x_2]) \end{array}$$

such that

$$\begin{array}{ll} [x] = [x_1] \cup [x_2] & \text{(covering)} \\ [x] = [a_1] \cup [a_2] \Rightarrow w([x_1]) + w([x_2]) \leq w([a_1]) + w([a_2]) & \text{(optimality)} \\ [x] = [a_1] \cup [a_2], [a_1] \subset [x_1], [a_2] \subset [x_2] \Rightarrow [a_1] = [x_1], [a_2] = [x_2] & \text{(non overlapping)} \end{array}$$

1.2.2 Real intervals

We shall now consider the case of intervals of $\mathbb{R}$, or real intervals.

Real intervals. A real *interval* is a connected, closed subset of $\mathbb{R}$. The set of all real intervals of $\mathbb{R}$ is denoted by $\mathbb{IR}$.

For example $[1,3]$, $\{1\}$, $]-\infty,6]$, $\mathbb{R}$ and $\emptyset$ are considered as real intervals whereas $]1,3[$, $[3,2]$ and $[1,2]\cup[3,4]$ are not.

Width. The *width* of the interval $[x]$ is defined by

$$w([x]) = x^+ - x^-.$$

Midpoint. The *midpoint* of the interval $[x]$ is defined by

$$\text{mid}([x]) = \frac{x^+ + x^-}{2}.$$

Enveloping interval. The *enveloping interval* of a subset $\mathbb{X}$ of $\mathbb{R}$ is the smallest interval containing $\mathbb{X}$ and is denoted by $[\mathbb{X}]$. For instance

$$[\ [1,3] \cup [6,7[\ ] = [1,7].$$

1.2.3 Boxes

Box. A *box* $[\mathbf{x}]$ (or *vector interval*) is an interval of $\bar{\mathbb{R}}^n$. It corresponds to the Cartesian product of n intervals *i.e.* a vector with interval components:

$$[\mathbf{x}] = [x_1^-, x_1^+] \times \cdots \times [x_n^-, x_n^+] = [x_1] \times \cdots \times [x_n].$$

The set of all boxes of $\mathbb{R}^n$ will be denoted by $\mathbb{IR}^n$.

Width. The *width* $w([\mathbf{x}])$ of a box $[\mathbf{x}]$ is the length of its largest side

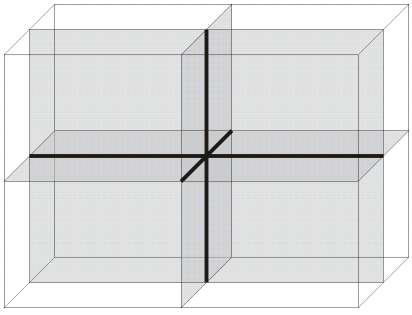
$$w([\mathbf{x}]) = \max_{i\in\{1,\ldots,n\}} w([x_i]).$$

For instance

$$w([1,2]\times[-1,3]) = 4.$$

By convention $w(\emptyset) = -\infty$. If $w([\mathbf{x}]) = 0$, $[\mathbf{x}]$ is said to be *degenerated*. In such a case, $[\mathbf{x}]$ is a singleton of $\mathbb{R}^n$ and will be denoted $\{\mathbf{x}\}$.

Principal plane. The *principal plane* of $[\mathbf{x}]$ is the symmetric plane $[\mathbf{x}]$ perpendicular to its largest side.



A n-dimensional box has n symmetric planes

Bisection. To *bisect* a box $[\mathbf{x}]$ means to split it in the two parts separated by its principal plane. For instance, the bisection of $[\mathbf{x}] = [1,2] \times [-1,3]$ generates the two boxes $[\mathbf{x}](1) = [1,2] \times [-1,1]$ and $[\mathbf{x}](2) = [1,2] \times [1,3]$.

1.2.4 Boolean intervals

A *Boolean number* is an element of

$$\mathbb{B} \triangleq \{false, true\} = \{0,1\}.$$

If we define the relation $\leq$ as

$$0 \leq 0, 0 \leq 1, 1 \leq 1,$$

then, the set $(\mathbb{B}, \leq)$ is a lattice for which intervals can be defined. A *Boolean interval* is an element of

$$\mathbb{IB} = \{\emptyset, 0, 1, [0,1]\},$$

where $\emptyset$ stands for *impossible,* 0 for *false*, 1 for *true*, and $[0,1]$ for *indeterminate*.

1.3 Interval arithmetic

1.3.1 Principle

Given $(\mathbb{X}_1, \leq_1), \ldots, (\mathbb{X}_n, \leq_n), (\mathbb{Y}, \preceq)$, $n+1$ lattices. Consider the n-ary operator $\diamond$ from $\mathbb{X}_1 \times \cdots \times \mathbb{X}_n$ to $\mathbb{Y}$. We extend the operator $\diamond$ to intervals of as follows

$$[x_1] \diamond [x_2] \diamond \cdots \diamond [x_n] \triangleq [\{x_1 \diamond x_2 \diamond \cdots \diamond x_n \mid x_1 \in [x_1], \ldots, x_n \in [x_n]\}].$$

where $[\mathbb{A}]$ is the interval hull of the set $\mathbb{A}$ as defined on page 9. If f an elementary function from $\mathbb{X}$ to $\mathbb{Y}$, we define its interval extension as

$$f([x]) \triangleq [\{f(x) \mid x \in [x]\}].$$

An operator $\diamond$ or a function f is said to be *elementary* if we are able to find efficient polynomial algorithms to compute $[x_1] \diamond \cdots \diamond [x_n]$ or $f([x])$.

1.3.2 Real interval arithmetic

We shall consider the case where $n = 1$ and $\mathbb{X}_1 = \mathbb{X}_2 = \mathbb{Y} = \bar{\mathbb{R}}$. Consider $\diamond \in \{+, -, *, /, \max, \min, \ldots\}$, where $*$ is the multiplication, a binary operation in $\bar{\mathbb{R}}$ We extend the operator $\diamond$ to intervals of $\mathbb{R}$ as follows

$$[x] \diamond [y] \triangleq [\{x \diamond y \mid x \in [x], y \in [y]\}].$$

As a consequence,

$$\begin{aligned}
[x^-, x^+] + [y^-, y^+] &= [x^- + y^-, x^+ + y^+] \\
[x^-, x^+] * [y^-, y^+] &= [\min(x^-y^-, x^+y^-, x^-y^+, x^+y^+), \\
&\quad \max(x^-y^-, x^+y^-, x^-y^+, x^+y^+)] \\
\max([x^-, x^+], [y^-, y^+]) &= [\max(x^-, y^-), \max(x^+, y^+)].
\end{aligned}$$

For instance,

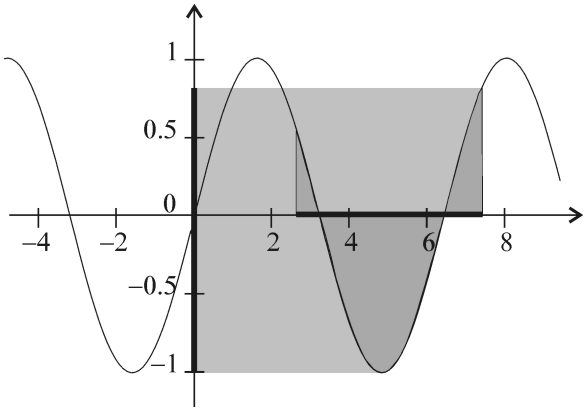
$$\begin{aligned}
[-1, 3] + [2, 5] &= [1, 8], \\
[-1, 3].[2, 5] &= [-5, 15], \\
[-1, 3]/[2, 5] &= [-\tfrac{1}{2}, \tfrac{3}{2}], \\
\max([-1, 3], [2, 5]) &= [2, 5].
\end{aligned}$$

These operators can be combined. For instance

$$([1, 2] + [-3, 4]) * [-1, 5] = [-2, 6] * [-1, 5] = [-10, 30].$$

Some examples of elementary functions extended to intervals are given below.

$$\begin{aligned}
\sin([0, \pi]) &= [0, 1], \\
\mathrm{sqr}([-1, 3]) &= [-1, 3]^2 = [0, 9], \\
\mathrm{abs}([-7, 1]) &= [0, 7], \\
\mathrm{sqrt}([-10, 4]) &= \sqrt{[-10, 4]} = [0, 2], \\
\log([-2, -1]) &= \emptyset.
\end{aligned}$$



The sine function with an interval argument

1.3.3 Boolean interval arithmetic

Operations on Boolean intervals are also defined in the framework of set computation:

$$\begin{aligned}
[a] \vee [b] &= \{a \vee b \mid a \in [a], b \in [b]\}, \\
[a] \wedge [b] &= \{a \wedge b \mid a \in [a], b \in [b]\}, \\
\neg [a] &= \{\neg a \mid a \in [a]\},
\end{aligned}$$

where $\wedge$ and $\vee$ respectively stand for the `AND` and `OR` operators and where $\neg$ is the complementation operator. For instance,

$$([0,1] \vee 1) \wedge ([0,1] \wedge 1) = 1 \wedge [0,1] = [0,1].$$

The dependency effect is still present when the complementation operator is used. For instance,

$$[a] \subset ([a] \wedge [b]) \vee ([a] \wedge \neg [b]),$$

and the values of the two sides of the equation differ for $a = 1$ and $[b] = [0,1]$, whereas $a = (a \wedge b) \vee (a \wedge \neg b)$.

1.3.4 Interval relation

A binary relation on $\mathbb{X}$ can be seen as function from $\mathbb{X} \times \mathbb{X}$ to $\{0,1\}$. For instance, consider the relation $\leq$ on $\mathbb{R}$. We have

$$\left([x^-, x^+] \leq [y^-, y^+]\right) = \begin{cases} 1 & \text{if } x^+ \leq y^- \\ 0 & \text{if } y^+ < x^- \\ [0,1] & \text{otherwise} \end{cases}$$

For instance

$$\begin{array}{lcl} ([1,3] \leqslant [3,5]) & = & 1, \\ ([1,3] \leqslant [-1,0]) & = & 0, \\ ([1,3] \leqslant [2,5]) & = & [0,1] \end{array}$$

If now we consider the unary relation given by $x \in [0,5]$, we get

$$\begin{array}{lcl} ([1,3] \in [0,5]) & = & 1 \\ ([6,9] \in [0,5]) & = & 0 \\ ([3,9] \in [0,5]) & = & [0,1]. \end{array}$$

1.4 Inclusion function

1.4.1 Definitions

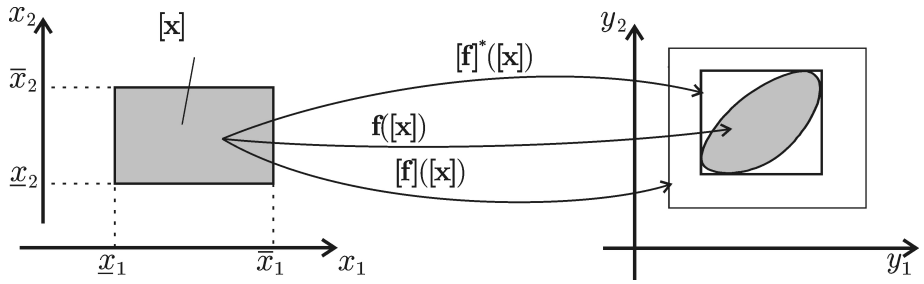
Consider two *lattices* $\mathbb{X}$ and $\mathbb{Y}$ and a function $f : \mathbb{X} \mapsto \mathbb{Y}$. The interval function $[f]$ from $\mathbb{IX}$ to $\mathbb{IY}$ is an *inclusion function* of f if

$$\forall\, [x] \in \mathbb{IX}, \quad f([x]) \subset [f]\,([x]).$$

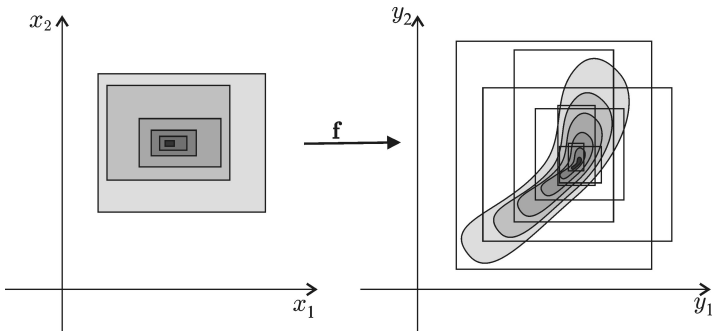
We define the following properties for inclusion functions

$[f]$ is *monotonic*	if	$([x] \subset [y]) \Rightarrow ([f]\,([x]) \subset [f]\,([y]))$
$[f]$ is *minimal*	if	$\forall [x] \in \mathbb{IX},\ [f]\,([x]) = [f\,([x])]$
$[f]$ is *thin*	if	$\forall x \in \mathbb{X},\ [f](\{x\}) = f(\{x\})$
$[f]$ is *convergent*	if	$\lim_{k\to\infty} w([x]\,(k)) = 0 \Rightarrow \lim_{k\to\infty} w([f]([x]\,(k)) = 0$

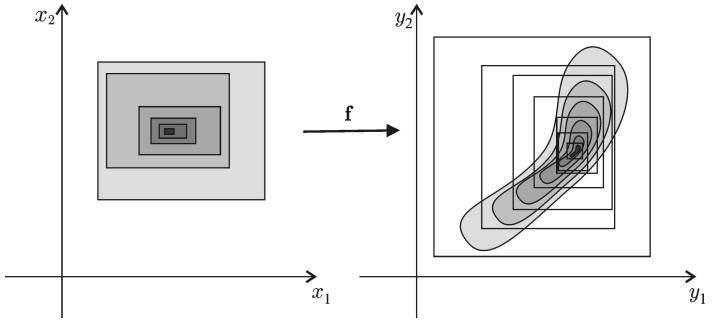
The figure below illustrates these notions in the case where $\mathbb{X}$ and $\mathbb{Y}$ correspond to $\mathbb{R}^2$.



Inclusion functions $[\mathbf{f}]$ and $[\mathbf{f}]^*$; here, $[\mathbf{f}]^*$ is minimal.



Convergent but non-monotonic inclusion function



Convergent and monotonic inclusion function

1.4.2 Natural inclusion function

We shall now consider the case of a function $\mathbf{f}$ from $\mathbb{R}^n$ to $\mathbb{R}^p$. Computing an optimal inclusion function of a polynomial function $\mathbf{f}$ has been proven to be a NP-hard problem [**?**] that is why we shall limit ourselves on trying to find convergent inclusion functions. An easy way to compute an inclusion function $[\mathbf{f}]$ for $\mathbf{f}$ is to replace in the expression of $\mathbf{f}$, all x_i's by $[x_i]$ and all operations on reals by their interval counterpart. It can be proven [64] that if the expression of $\mathbf{f}$ is made as a combination of elementary continuous functions (sqr,sqrt, exp, sin, cos, log,...) or operator $(+, -, \times, \max, \ldots)$, then $[\mathbf{f}]$ is convergent. Moreover if for each component f_i of $\mathbf{f}$, each x_j occurs at most once, then $[\mathbf{f}]$ is minimal.

Example 6 *The natural inclusion function for* $f(x) = x^2 + 2x + 4$ *is*

$$[f]([x]) = sqr([x]) + 2[x] + 4.$$

If $[x] = [-3, 4]$, *we have*

$$[f]([-3,4]) = sqr([-3,4]) + 2[-3,4] + 4 = [0,16] + [-6,8] + 4 = [-2,28].$$

Note that $f([-3,4]) = [3,28]$ *which is a subset of* $[f]([-3,4]) = [-2,28]$. ■

Example 7 *Consider the function*

$$\mathbf{f}: \begin{array}{lcl} \mathbb{R}^2 & \to & \mathbb{R}^3 \\ (x_1, x_2) & \mapsto & \left(x_1 x_2, x_1^2, x_1 - x_2\right). \end{array}$$

Its natural inclusion function is

$$[\mathbf{f}]: \begin{array}{lcl} \mathbb{IR}^2 & \to & \mathbb{IR}^3 \\ ([x_1], (x_2)) & \to & \left([x_1] * [x_2], [x_1]^2, [x_1] - [x_2]\right). \end{array}$$

For instance, $[\mathbf{f}]([-1,1] \times [-1,1]) = [-1,1] \times [0,1] \times [-2,2]$. *Since in each* f_i, *the* x_i*'s occur only once, the natural inclusion function is minimal.* ■

Example 8 *If* $\mathbf{f}$ *is given by the following algorithm*

Algorithm $\mathbf{f}$ *(in:* $\mathbf{x} = (x_1, x_2, x_3)$, *out:* $\mathbf{y} = (y_1, y_2)$*)*	
1	$z := x_1$;
2	*for* $k := 0$ *to* 100
3	$z := x_2(z + kx_3)$;
4	*next;*
5	$y_1 := z$;
6	$y_2 := \sin(zx_1)$;

Its natural inclusion function is

Algorithm $[\mathbf{f}]$ *(in:* $[\mathbf{x}] = ([x_1], [x_2], [x_3])$, *out:* $[\mathbf{y}] = ([y_1], [y_2])$*)*	
1	$[z] := [x_1]$;
2	*for* $k := 0$ *to* 100
3	$[z] := [x_2] * ([z] + k * [x_3])$;
4	*next;*
5	$[y_1] := [z]$;
6	$[y_2] := \sin([z] * [x_1])$;

Here, $[\mathbf{f}]$ *is a convergent, thin and monotonic inclusion function for* $\mathbf{f}$. ■

Example 9 *If* f *is given by*

$$f(x) = \frac{1}{\frac{1}{x}}$$

then the natural inclusion function is

$$[f]([x]) = \frac{1}{\frac{1}{[x]}}.$$

It now, $[x] = [-a, a]$, *then*

$$[f]([x]) = \frac{1}{\frac{1}{[-a,a]}} = \frac{1}{[-\infty, \infty]} = [-\infty, \infty]$$

whereas $f([x]) = f([-a,a]) = [-a,a]$. *As a consequence,* $[f]$ *is neither convergent nor minimal. If* $0 \notin [x]$, *we get a minimal enclosure and the problem appears only if* $0 \in [x]$. *This is due to the fact that the function* $\frac{1}{x}$ *is not continuous around* 0. ■

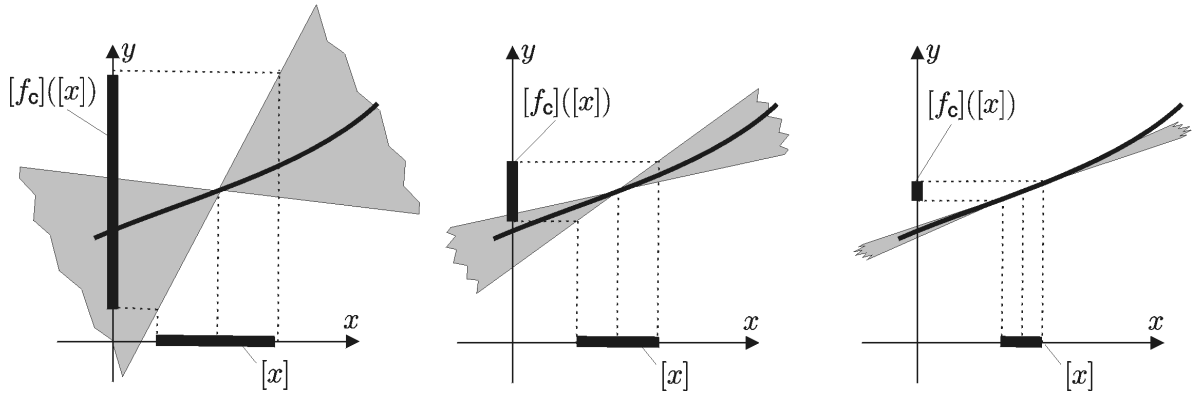


Figure 1.1: Interpretation of the centred inclusion function

1.4.3 Centred inclusion functions

Let $f : \mathbb{R}^n \rightarrow \mathbb{R}$ be a scalar function of a vector $\mathbf{x} = (x_1, \ldots, x_n)^{\mathrm{T}}$. Assume that f is differentiable over the box $[\mathbf{x}]$, and denote mid($[\mathbf{x}]$) by $\mathbf{m}$. The mean-value theorem then implies that

$$\forall \mathbf{x} \in [\mathbf{x}] \, , \exists \mathbf{z} \in [\mathbf{x}] \ \mid f(\mathbf{x}) = f(\mathbf{m}) + \mathbf{g}^{\mathrm{T}}(\mathbf{z})(\mathbf{x} - \mathbf{m}) \, , \tag{1.1}$$

where $\mathbf{g}$ is the *gradient* of f, *i.e.*, a column vector with entries $g_i = \partial f / \partial x_i$, $i = 1, \ldots, n$. Thus,

$$\forall \mathbf{x} \in [\mathbf{x}] \, , f(\mathbf{x}) \in f(\mathbf{m}) + [\mathbf{g}^{\mathrm{T}}]([\mathbf{x}])(\mathbf{x} - \mathbf{m}) \, , \tag{1.2}$$

where $[\mathbf{g}^{\mathrm{T}}]$ is an inclusion function for $\mathbf{g}^{\mathrm{T}}$, so

$$f([\mathbf{x}]) \subseteq f(\mathbf{m}) + [\mathbf{g}^{\mathrm{T}}]([\mathbf{x}])([\mathbf{x}] - \mathbf{m}) \, . \tag{1.3}$$

Therefore, the interval function

$$[f_{\mathrm{c}}]([\mathbf{x}]) \triangleq f(\mathbf{m}) + [\mathbf{g}^{\mathrm{T}}]([\mathbf{x}])([\mathbf{x}] - \mathbf{m}) \tag{1.4}$$

is an inclusion function for $\mathbf{f}$, which we shall call the *centred inclusion function*. To illustrate the interest of this function in the one-dimensional case, consider the function $[f_{\mathrm{c}}](x)$ from $\mathbb{R}$ to $\mathbb{IR}$ defined by

$$[f_{\mathrm{c}}](x) \triangleq f(m) + [f^{'}]([x])(x - m) \tag{1.5}$$

for any given $[x]$. This function can be viewed as affine in x with an uncertain slope belonging to $[f^{'}]([x])$. The graph of $[f_{\mathrm{c}}](x)$ can thus be represented by a cone with centre $(m, f(m))$ as illustrated by Figure 1.1 for decreasing widths of $[x]$. The smaller $w([x])$ is, the better the cone approximates the function. The figure illustrates the fact that

$$\frac{w([f_c]([x]))}{w(f([x]))} \rightarrow 1$$

when the width of $[x]$ tends to 0, which is not the case in general for a natural inclusion function.

When the width of $[\mathbf{x}]$ is small, the effect of the pessimism possibly resulting from the interval evaluation of $[\mathbf{g}]([\mathbf{x}])$ is reduced by the scalar product with $[\mathbf{x}] - \mathbf{m}$, which is a small interval centred on zero.

1.4.4 Inclusion tests

A *test* is a function t from $\mathbb{R}^n$ to $\mathbb{B}$. An *inclusion test* $[t]$ is an inclusion function for t. It means that $[t]$ is a function from $\mathbb{IR}^n$ to $\mathbb{IB}$ such that for any $[\mathbf{x}] \in \mathbb{IR}^n$,

$$\begin{array}{rcl} ([t]([\mathbf{x}]) = 1) & \Rightarrow & (\forall \mathbf{x} \in [\mathbf{x}]\,,\ t(\mathbf{x}) = 1)\,, \\ ([t]([\mathbf{x}]) = 0) & \Rightarrow & (\forall \mathbf{x} \in [\mathbf{x}]\,,\ t(\mathbf{x}) = 0)\,. \end{array}$$

An *inclusion test* $[t_{\mathbb{A}}]$ *for* a set $\mathbb{A}$ of $\mathbb{R}^n$ is an inclusion test for the characteristic test of $\mathbb{A}$ defined by $t_{\mathbb{A}}(\mathbf{x}) \Leftrightarrow (\mathbf{x} \in \mathbb{A})$. We have

$$\begin{array}{rcl} [t_{\mathbb{A}}]([\mathbf{x}]) = 1 & \Rightarrow & (\forall \mathbf{x} \in [\mathbf{x}]\,,\ t_{\mathbb{A}}(\mathbf{x}) = 1) \quad \Leftrightarrow \quad ([\mathbf{x}] \subset \mathbb{A})\,, \\ [t_{\mathbb{A}}]([\mathbf{x}]) = 0 & \Rightarrow & (\forall \mathbf{x} \in [\mathbf{x}]\,,\ t_{\mathbb{A}}(\mathbf{x}) = 0) \quad \Leftrightarrow \quad ([\mathbf{x}] \cap \mathbb{A} = \emptyset)\,. \end{array}$$

When $[t_{\mathbb{A}}]([\mathbf{x}]) = [0,1]$, nothing can be concluded about the inclusion of $[\mathbf{x}]$ in $\mathbb{A}$. Consider an inclusion test $[t]$, we shall say that

$[t]$ is *monotonic*	if	$([\mathbf{x}] \subset [\mathbf{y}]) \Rightarrow ([t]([\mathbf{x}]) \subset [t]([\mathbf{y}]))$
$[t]$ is *minimal*	if	$\forall [\mathbf{x}] \in \mathbb{IR}^n,\ [t]([\mathbf{x}]) = t([\mathbf{x}])$
$[t]$ is *thin*	if	$\forall \mathbf{x} \in \mathbb{R}^n,\ [t](\mathbf{x}) \neq [0,1]$.

If $\mathbb{A}$ and $\mathbb{B}$ are two sets, we have

$$\begin{array}{rcl} t_{\mathbb{A}\cap\mathbb{B}} & = & t_{\mathbb{A}} \wedge t_{\mathbb{B}} \\ t_{\mathbb{A}\cup\mathbb{B}} & = & t_{\mathbb{A}} \vee t_{\mathbb{B}} \\ t_{\neg\mathbb{A}} & = & \neg t_{\mathbb{A}} = 1 - t_{\mathbb{A}} \end{array}$$

The following properties can be used to build inclusion tests for sets defined from elementary set operations such as union, intersection or complementation. If $[t_{\mathbb{A}}]([\mathbf{x}])$ and $[t_{\mathbb{B}}]([\mathbf{x}])$ are thin inclusion tests for the sets $\mathbb{A}$ and $\mathbb{B}$, define

$$\begin{array}{ll} [t_{\mathbb{A}\cap\mathbb{B}}]([\mathbf{x}]) & \triangleq ([t_{\mathbb{A}}] \wedge [t_{\mathbb{B}}])([\mathbf{x}]) = [t_{\mathbb{A}}]([\mathbf{x}]) \wedge [t_{\mathbb{B}}]([\mathbf{x}]), \\ [t_{\mathbb{A}\cup\mathbb{B}}]([\mathbf{x}]) & \triangleq ([t_{\mathbb{A}}] \vee [t_{\mathbb{B}}])([\mathbf{x}]) = [t_{\mathbb{A}}]([\mathbf{x}]) \vee [t_{\mathbb{B}}]([\mathbf{x}]), \\ [t_{\neg\mathbb{A}}]([\mathbf{x}]) & \triangleq \neg\,[t_{\mathbb{A}}]([\mathbf{x}]) = 1 - [t_{\mathbb{A}}]([\mathbf{x}]). \end{array}$$

$[t_{\mathbb{A}\cap\mathbb{B}}]$, $[t_{\mathbb{A}\cup\mathbb{B}}]$ and $[t_{\neg\mathbb{A}}]$ are then thin inclusion tests for the sets $\mathbb{A} \cap \mathbb{B}$, $\mathbb{A} \cup \mathbb{B}$ and $\neg\mathbb{A} \triangleq \{\mathbf{x} \in \mathbb{R}^n \mid \mathbf{x} \notin \mathbb{A}\}$, respectively.

Example 10 *Consider the test*

$$t : \begin{array}{ccc} \mathbb{R}^2 & \rightarrow & \{0,1\} \\ (x_1, x_2)^T & \mapsto & (x_1 + x_2 \leqslant 5), \end{array}$$

which means that

$$t(\mathbf{x}) = \begin{cases} 1 & \text{if} \quad x_1 + x_2 \leqslant 5, \\ 0 & \text{if} \quad x_1 + x_2 > 5. \end{cases}$$

The minimal inclusion test $[t]$ *associated with* t *is given by*

$$[t]([\mathbf{x}]) = \begin{cases} 1 & \text{if} \quad x_1^+ + x_2^+ \leqslant 5, \\ 0 & \text{if} \quad x_1^- + x_2^- > 5, \\ [0,1] & \text{otherwise,} \end{cases}$$

which can be written more concisely as

$$[t]([\mathbf{x}]) \Leftrightarrow ([x_1] + [x_2] \leqslant 5)\,.$$

It is minimal. ■

With the help of interval analysis and the notion of inclusion function, it is easy to build an inclusion test for a large class of sets.

Example 11 *Consider the set*

$$\mathbb{A} \triangleq \{x \in \mathbb{R} | \, (x \leqslant 7) \vee (x \geqslant 6)\}\,.$$

An inclusion test for $\mathbb{A}$ *is* $[t]([x]) = ([x] \leqslant 7) \vee ([x] \geqslant 6)$. *Despite the fact that* $[t]([x])$ *consists of two minimal inclusion tests, this inclusion test is pessimistic. For instance for* $[x] = [5, 8]$,

$$[t]([x]) = ([5, 8] \leqslant 7) \vee ([5, 8] \geqslant 6) = [0, 1] \ \vee \ [0, 1] = [0, 1],$$

whereas $t([x]) = \{t(x) \mid x \in [5, 8]\} = 1$.

Chapter 2

Computing with sets

2.1 Subpavings

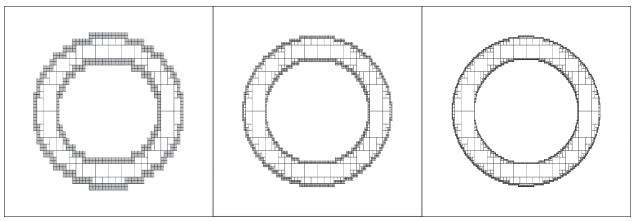
A *subpaving* of $\mathbb{R}^n$ is a set of non-overlapping boxes of $\mathbb{R}^n$. Compact sets $\mathbb{X}$ can be bracketed between inner and outer subpavings:

$$\mathbb{X}^- \subset \mathbb{X} \subset \mathbb{X}^+.$$

The following figure illustrates the bracketing of the set

$$\mathbb{X} = \{(x_1, x_2) \,\big|\, x_1^2 + x_2^2 \in [1, 2]\}$$

between subpavings with an increasing accuracy from left to right. The frame corresponds to the box $[-2, 2] \times [-2, 2]$. The subpaving $\Delta\mathbb{X}$ in grey contains the boundary of $\mathbb{X}$ whereas the subpaving $\mathbb{X}^-$ in white is inside $\mathbb{X}$.



Set operations such as $\mathbb{Z} := \mathbb{X} + \mathbb{Y}$, $\mathbb{X} := \mathbf{f}^{-1}(\mathbb{Y}), \mathbb{Z} := \mathbb{X} \cap \mathbb{Y} \ldots$ can be approximated by subpaving operations.

2.2 Pavers

A paver is an algorithm which generates boxes by bisections and classifies them.

2.2.1 Stack-queue

A *list* is a (possibly empty) ordered finite set. If the list is nonempty, all its elements, except one (called the *last*), have a next element. The element of the list which is the next of nobody is called the *first* element.

A *queue* is a list on which two operations are allowed : (i) add an element at the end (*push*) (ii) remove the first element (*pull*).

A *stack* is a list on which two operations are allowed : (i) add an element at the beginning of the list (*stack*) (ii) remove the first element (*pop*).

Example 12 *Let $\mathcal{L}$ be an empty list. The next table illustrates the evolution of the queue when different operations are performed.*

k	*operation*	*result*
0		$\mathcal{L} = \emptyset$
1	$push(\mathcal{L}, a)$	$\mathcal{L} = \{a\}$
2	$push(\mathcal{L}, b)$	$\mathcal{L} = \{a, b\}$
3	$x := pull(\mathcal{L})$	$x = a, \mathcal{L} = \{b\}$
4	$x := pull(\mathcal{L})$	$x = b, \mathcal{L} = \emptyset$.

For a stack, the table becomes

k	*operation*	*result*
0		$\mathcal{L} = \emptyset$
1	$stack(\mathcal{L}, a)$	$\mathcal{L} = \{a\}$
2	$stack(\mathcal{L}, b)$	$\mathcal{L} = \{a, b\}$
3	$x := pop(\mathcal{L})$	$x = b, \mathcal{L} = \{a\}$
4	$x := pop(\mathcal{L})$	$x = a, \mathcal{L} = \emptyset$.

2.2.2 SIVIA

Given a test t, consider the problem of characterizing the set

$$\mathbb{X} = \{\mathbf{x} \in \mathbb{R}^n \mid t(\mathbf{x}) = 1\} = t^{-1}(1)$$

and assume that we have an inclusion test $[t]$ for t. An enclosure of the form

$$\mathbb{X}^- \subset \mathbb{X} \subset \mathbb{X}^+$$

can be obtained with the algorithm SIVIA (Set Inverter Via Interval Analysis), to be described now. To test if a box $[\mathbf{x}]$ is inside or outside $\mathbb{X}$, we shall use the inclusion test $[t]$. Boxes $[\mathbf{x}]$ for which the test fails, i.e., $[t]([\mathbf{x}]) = [0, 1]$, will be bisected, except if they are too small (*i.e.,* smaller than the required accuracy

ε). The box $[\mathbf{x}](0)$ is a box which is assumed to enclose the solution set $\mathbb{X}$. $\mathcal{L}$ is a queue of boxes.

Algorithm SIVIA(in: $[\mathbf{x}](0), [t]$)	
1	$\mathcal{L} := \{[\mathbf{x}](0)\}$;
2	pull $([\mathbf{x}],\mathcal{L})$;
3	if $[t]([\mathbf{x}]) = 1$, draw($[\mathbf{x}]$, 'red');
4	elseif $[t]([\mathbf{x}]) = 0$, draw($[\mathbf{x}]$, 'blue');
5	elseif $w([\mathbf{x}]) < \varepsilon$, {draw ($[\mathbf{x}]$, 'yellow')};
6	else bisect $[\mathbf{x}]$ into $[\mathbf{x}](1)$ and $[\mathbf{x}](2)$; push $([\mathbf{x}](1),[\mathbf{x}](2),\mathcal{L})$;
7	if $\mathcal{L} \neq \emptyset$, go to 2

If $\Delta\mathbb{X}$ denotes the union of yellow boxes and if $\mathbb{X}^-$ is the union of red boxes then

$$\mathbb{X}^- \subset \mathbb{X} \subset \mathbb{X}^- \cup \Delta\mathbb{X}.$$

2.2.3 Set inversion

Let $\mathbf{f}$ be a possibly non-linear function from $\mathbb{R}^n$ to $\mathbb{R}^m$ and let $\mathbb{Y}$ be a subset of $\mathbb{R}^m$ (for instance, a subpaving). Set inversion is the characterization of

$$\mathbb{X} = \{\mathbf{x} \in \mathbb{R}^n \mid \mathbf{f}(\mathbf{x}) \in \mathbb{Y}\} = \mathbf{f}^{-1}(\mathbb{Y}).$$

For any $\mathbb{Y} \subset \mathbb{R}^m$ and for any function $\mathbf{f}$ admitting a convergent inclusion function $[\mathbf{f}](.)$, two subpavings $\mathbb{X}^-$ and $\mathbb{X}^+$ such that

$$\mathbb{X}^- \subset \mathbb{X} \subset \mathbb{X}^+$$

can be obtained with the algorithm SIVIA (Set Inverter Via Interval Analysis), by choosing the inclusion test $[t]$ as defined below.

$$[t]([\mathbf{x}]) = \begin{cases} 1 & \text{if } [\mathbf{f}]([\mathbf{x}]) \subset \mathbb{Y} \\ 0 & \text{if } [\mathbf{f}]([\mathbf{x}]) \cap \mathbb{Y} = \emptyset. \\ [0,1] & \text{otherwise.} \end{cases}$$

Example 13 *Solving a system of equations of the form* $\mathbf{f}(\mathbf{x}) = \mathbf{0}$ *is a set inversion problem. For instance, the equation*

$$x_1^2 + x_2^2 - 1 = 0,$$

is cast into a set inversion problem if we take $f(\mathbf{x}) = x_1^2 + x_2^2 - 1$ *and* $\mathbb{Y} = \{0\}$. *To characterize the solution set, it suffices to take the inclusion test*

$$[t]([\mathbf{x}]) = \left([x_1]^2 + [x_2]^2 - 1 \in [0,0]\right)$$

and we apply SIVIA *with this test.*

Example 14 *Solving inequalities is also a set inversion problem. To transform*

$$\begin{cases} x_1^2 + x_2^2 \in [1,2] \\ x_1 + \sin x_2 \in [0,1] \end{cases}$$

into a set inversion problem, it suffices to take

$$\mathbf{f}(\mathbf{x}) = \begin{pmatrix} x_1^2 + x_2^2 \\ x_1 + \sin x_2 \end{pmatrix}$$

and $\mathbb{Y} = [1,2] \times [0,1]$. *To apply* SIVIA, *we only have to build the inclusion test*

$$[t]([\mathbf{x}]) = \left(\left([x_1]^2 + [x_2]^2 \in [1,2] \right) \text{ and } ([x_1] + \sin [x_2] \in [0,1]) \right).$$

Example 15 *The composition of two set inversion problems yields a set inversion problem. For instance if*

$$\mathbb{X} = f^{-1}(\mathbb{Y}) \text{ and } \mathbb{Y} = g^{-1}(\mathbb{Z})$$

then

$$\mathbb{X} = f^{-1}\left(g^{-1}(\mathbb{Z})\right) = (g \circ f)^{-1}(\mathbb{Z}).$$

2.2.4 Projection

Consider a set

$$\mathbb{Z} = \{\mathbf{z} = (\mathbf{x}, \mathbf{y}) \in [\mathbf{x}] \times [\mathbf{y}], t(\mathbf{x}, \mathbf{y})\},$$

where $t(\mathbf{x}, \mathbf{y})$ is a test. The projection of $\mathbb{Z}$ onto $\mathbf{x}$ is

$$\mathbb{X} = \{\mathbf{x} \in [\mathbf{x}], \exists \mathbf{y} \in [\mathbf{y}], t(\mathbf{x}, \mathbf{y})\}.$$

The test $t_{\mathbf{x}}(\mathbf{x})$ defined by

$$t_{\mathbf{x}}(\mathbf{x}) \Leftrightarrow \exists \mathbf{y} \in [\mathbf{y}], t(\mathbf{x}, \mathbf{y})$$

is called the projection of t onto $\mathbf{x}$. From an inclusion test $[t]([\mathbf{x}], [\mathbf{y}])$ for $t(\mathbf{x}, \mathbf{y})$, it is possible to compute an inclusion test $[t_{\mathbf{x}}]([\mathbf{x}])$ for $t_{\mathbf{x}}(\mathbf{x})$ using the following test-projection algorithm.

Algorithm $[t_{\mathbf{x}}]$(in: $[\mathbf{x}], [\mathbf{y}], [t]$)	
1	$\mathcal{L} := \{[\mathbf{y}]\}$;
2	while $\mathcal{L} \neq \emptyset$,
3	pull $([\mathbf{y}], \mathcal{L})$;
4	if $[t]([\mathbf{x}]$,center$([\mathbf{y}])) = 1$, return (1);
5	if $[t]([\mathbf{x}], [\mathbf{y}]) = 0$, goto 2;
6	if $w([\mathbf{y}]) < w([\mathbf{x}])$, return $([0,1])$;
	bisect $[\mathbf{y}]$ into $[\mathbf{y}](1)$ and $[\mathbf{y}](2)$; push $([\mathbf{y}](1)$,$[\mathbf{y}](2), \mathcal{L})$;
7	end while;
8	return 0.

Example 16 *Consider the problem of characterizing the set*

$$\mathbb{X} = \{\mathbf{x} \in [\mathbf{x}], \exists \mathbf{y} \in [\mathbf{y}], \mathbf{f}(\mathbf{x}, \mathbf{y}) \leq \mathbf{0}\}.$$

which can also be defined by

$$\begin{aligned} \mathbb{Z} &= \{(\mathbf{x}, \mathbf{y}) \in [\mathbf{x}] \times [\mathbf{y}], \mathbf{f}(\mathbf{x}, \mathbf{y}) \leq \mathbf{0}\} \\ \mathbb{X} &= proj_{\mathbf{x}}(\mathbb{Z}) \end{aligned}$$

We can decompose the problem as follows

$$\begin{array}{lll} \mathbb{X} = \{\mathbf{x} \in [\mathbf{x}], t_{\mathbb{X}}(\mathbf{x})\} & \textit{where} & t_{\mathbb{X}}(\mathbf{x}) \Leftrightarrow (\exists \mathbf{y} \in [\mathbf{y}], \mathbf{f}(\mathbf{x},\mathbf{y}) \leq \mathbf{0}) \\ t_{\mathbb{X}}(\mathbf{x}) \Leftrightarrow (\exists \mathbf{y} \in [\mathbf{y}], t_{\mathbb{Z}}(\mathbf{x},\mathbf{y})) & \textit{where} & t_{\mathbb{Z}}(\mathbf{x},\mathbf{y}) \Leftrightarrow (\mathbf{f}(\mathbf{x},\mathbf{y}) \leq \mathbf{0}) \,. \end{array}$$

To use SIVIA*, we first build an inclusion test* $[t_{\mathbb{Z}}]([\mathbf{x}],[\mathbf{y}])$ *for* $t_{\mathbb{Z}}(\mathbf{x},\mathbf{y})$*. Using the test-projection algorithm, we build a test* $[t_{\mathbb{X}}]([\mathbf{x}])$ *for* $t_{\mathbb{X}}(\mathbf{x})$ *which is used by* SIVIA*.*

Example 17 *Consider the problem of characterizing the set*

$$\mathbb{X} = \{\mathbf{x} \in [\mathbf{x}], \exists \mathbf{y} \in [\mathbf{y}], \forall \mathbf{z} \in [\mathbf{z}], \mathbf{f}(\mathbf{x},\mathbf{y},\mathbf{z}) \leq \mathbf{0}\} \,.$$

We can decompose the problem as follows

$$\begin{array}{lll} \mathbb{X} = \{\mathbf{x} \in [\mathbf{x}], t_1(\mathbf{x})\} & \textit{where} & t_1(\mathbf{x}) \Leftrightarrow (\exists \mathbf{y} \in [\mathbf{y}], \forall \mathbf{z} \in [\mathbf{z}], \mathbf{f}(\mathbf{x},\mathbf{y},\mathbf{z}) \leq \mathbf{0}) \\ t_1(\mathbf{x}) \Leftrightarrow (\exists \mathbf{y} \in [\mathbf{y}], t_2(\mathbf{x},\mathbf{y})) & \textit{where} & t_2(\mathbf{x},\mathbf{y}) \Leftrightarrow (\forall \mathbf{z} \in [\mathbf{z}], \mathbf{f}(\mathbf{x},\mathbf{y},\mathbf{z}) \leq \mathbf{0}) \\ t_2(\mathbf{x},\mathbf{y}) = \neg t_3(\mathbf{x},\mathbf{y}) & \textit{where} & t_3(\mathbf{x},\mathbf{y}) \Leftrightarrow (\exists \mathbf{z} \in [\mathbf{z}], \neg (\mathbf{f}(\mathbf{x},\mathbf{y},\mathbf{z}) \leq \mathbf{0})) \\ t_3(\mathbf{x},\mathbf{y}) \Leftrightarrow (\exists \mathbf{z} \in [\mathbf{z}], t_4(\mathbf{x},\mathbf{y},\mathbf{z})) & \textit{where} & t_4(\mathbf{x},\mathbf{y},\mathbf{z}) \Leftrightarrow \neg (\mathbf{f}(\mathbf{x},\mathbf{y},\mathbf{z}) \leq \mathbf{0}) \\ t_4(\mathbf{x},\mathbf{y},\mathbf{z}) \Leftrightarrow \neg (t_5(\mathbf{x},\mathbf{y},\mathbf{z})) & \textit{where} & t_5(\mathbf{x},\mathbf{y},\mathbf{z}) \Leftrightarrow \mathbf{f}(\mathbf{x},\mathbf{y},\mathbf{z}) \leq \mathbf{0}. \end{array}$$

Example 18 *Consider the optimization problem*

$$a = \min_{\mathbf{x} \in [\mathbf{x}]} f(\mathbf{x}) \,.$$

For a given y*, we have*

$$\begin{array}{rcl} y & \geq & a \Leftrightarrow \exists \mathbf{x} \in [\mathbf{x}], f(\mathbf{x}) \leq y \\ y & \leq & a \Leftrightarrow \forall \mathbf{x} \in [\mathbf{x}], f(\mathbf{x}) \geq y. \end{array}$$

Thus the global minimal belong to the singleton

$$\begin{array}{rcl} \{a\} & = & \{y, y \geq a\} \cap \{y, y \leq a\} \\ & = & \{y, \exists \mathbf{x} \in [\mathbf{x}], f(\mathbf{x}) - y \leq 0\} \cap \{y, \forall \mathbf{x} \in [\mathbf{x}], f(\mathbf{x}) - y \geq 0\} \,. \end{array}$$

To use SIVIA*, we have to perform the following decomposition*

$$\begin{array}{lll} \{a\} = \{y, t_1(y) \wedge t_2(y)\} & \textit{where} & \left\{ \begin{array}{l} t_1(y) \Leftrightarrow (\exists \mathbf{x} \in [\mathbf{x}], f(\mathbf{x}) - y \leq 0) \\ t_2(y) \Leftrightarrow (\forall \mathbf{x} \in [\mathbf{x}], f(\mathbf{x}) - y \geq 0) \end{array} \right. \\ t_1(y) \Leftrightarrow (\exists \mathbf{x} \in [\mathbf{x}], t_3(\mathbf{x},y)) & \textit{where} & t_3(\mathbf{x},y) \Leftrightarrow (f(\mathbf{x}) - y \leq 0) \\ t_2(y) = \neg t_4(y) & \textit{where} & t_4(y) \Leftrightarrow (\exists \mathbf{x} \in [\mathbf{x}], f(\mathbf{x}) - y < 0) \\ t_4(y) \Leftrightarrow (\exists \mathbf{x} \in [\mathbf{x}], t_3(\mathbf{x},y)) \,. & & \end{array}$$

Example 19 *Consider the optimization problem in the case where* $\mathbf{f}$ *is a vector function* $\mathbf{f} : \mathbb{R}^n \mapsto \mathbb{R}^m$*. The problem can be written as*

$$\mathbb{P} = \min_{\mathbf{x} \in [\mathbf{x}]} \mathbf{f}(\mathbf{x})$$

or more precisely by

$$\mathbb{P} = \{\mathbf{y}, \exists \mathbf{x} \in [\mathbf{x}], \mathbf{f}(\mathbf{x}) \leq \mathbf{y}\} \cap \{\mathbf{y}, \forall \mathbf{x} \in [\mathbf{x}], \neg (\mathbf{f}(\mathbf{x}) < \mathbf{y})\}$$

is called the Pareto set. *Here,* $\mathbf{a} < \mathbf{b}$ *means that* $\mathbf{a} \leq \mathbf{b}$ *and* $\mathbf{a} \neq \mathbf{b}$. *It corresponds the generalization of the minimum in the multiobjective case. The corresponding test decomposition is as follows*

$$\begin{array}{lll}
\mathbb{P} = \{\mathbf{y}, t_1(\mathbf{y}) \wedge t_2(\mathbf{y})\} & \textit{where} & \left\{ \begin{array}{l} t_1(\mathbf{y}) \Leftrightarrow (\exists \mathbf{x} \in [\mathbf{x}], \mathbf{f}(\mathbf{x}) \leq \mathbf{y}) \\ t_2(\mathbf{y}) \Leftrightarrow (\forall \mathbf{x} \in [\mathbf{x}], \neg (\mathbf{f}(\mathbf{x}) < \mathbf{y})) \end{array} \right. \\
t_1(\mathbf{y}) \Leftrightarrow (\exists \mathbf{x} \in [\mathbf{x}], t_3(\mathbf{x}, \mathbf{y})) & \textit{where} & t_3(\mathbf{x}, \mathbf{y}) \Leftrightarrow (\mathbf{f}(\mathbf{x}) \leq \mathbf{y}) \\
t_2(\mathbf{y}) = \neg t_4(\mathbf{y}) & \textit{where} & t_4(\mathbf{y}) \Leftrightarrow (\exists \mathbf{x} \in [\mathbf{x}], \mathbf{f}(\mathbf{x}) < \mathbf{y}) \\
t_4(\mathbf{y}) \Leftrightarrow (\exists \mathbf{x} \in [\mathbf{x}], t_5(\mathbf{x}, \mathbf{y})) & \textit{where} & t_5(\mathbf{x}, \mathbf{y}) \Leftrightarrow (\mathbf{f}(\mathbf{x}) < \mathbf{y}).
\end{array}$$

Example 20 *Consider the problem of characterizing the set*

$$\mathbb{X} = \{\mathbf{x} \in [\mathbf{x}], \exists \mathbf{y} \in [\mathbf{y}], f(\mathbf{x}, \mathbf{y}) = 0\}.$$

The set $\{(\mathbf{x}, \mathbf{y}) \in [\mathbf{x}] \times [\mathbf{y}], f(\mathbf{x}, \mathbf{y}) = 0\}$ *has an empty volume and thus the inclusion test associated with* $f(\mathbf{x}, \mathbf{y}) = 0$ *will never return 1 when* $[\mathbf{x}]$ *and* $[\mathbf{y}]$ *are not thin. The previous approach will fail to produce a nonempty inner subpaving* $\mathbb{X}^-$ *for* $\mathbb{X}$. *Now, if* f *is continuous, from the mean-value theorem, we have*

$$(\exists \mathbf{y} \in [\mathbf{y}], f(\mathbf{x}, \mathbf{y}) = 0) \Leftrightarrow (\exists \mathbf{y} \in [\mathbf{y}], f(\mathbf{x}, \mathbf{y}) \geq 0) \wedge (\exists \mathbf{y} \in [\mathbf{y}], f(\mathbf{x}, \mathbf{y}) \leq 0).$$

Thus

$$\mathbb{X} = \{\mathbf{x} \in [\mathbf{x}], \exists \mathbf{y} \in [\mathbf{y}], f(\mathbf{x}, \mathbf{y}) \geq 0\} \cap \{\mathbf{x} \in [\mathbf{x}], \exists \mathbf{y} \in [\mathbf{y}], f(\mathbf{x}, \mathbf{y}) \leq 0\}.$$

The decomposition is thus

$$\begin{array}{lll}
\mathbb{X} = \{\mathbf{x} \in [\mathbf{x}], t_1(\mathbf{x}) \wedge t_2(\mathbf{x})\} & \textit{where} & \left\{ \begin{array}{l} t_1(\mathbf{x}) \Leftrightarrow (\exists \mathbf{y} \in [\mathbf{y}], f(\mathbf{x}, \mathbf{y}) \geq 0) \\ t_2(\mathbf{x}) \Leftrightarrow (\exists \mathbf{y} \in [\mathbf{y}], f(\mathbf{x}, \mathbf{y}) \leq 0) \end{array} \right. \\
t_1(\mathbf{x}) \Leftrightarrow \exists \mathbf{y} \in [\mathbf{y}], t_3(\mathbf{x}, \mathbf{y}) & \textit{where} & t_3(\mathbf{x}, \mathbf{y}) \Leftrightarrow (f(\mathbf{x}, \mathbf{y}) \geq 0) \\
t_2(\mathbf{x}) \Leftrightarrow \exists \mathbf{y} \in [\mathbf{y}], t_4(\mathbf{x}, \mathbf{y}) & \textit{where} & t_4(\mathbf{x}, \mathbf{y}) \Leftrightarrow (f(\mathbf{x}, \mathbf{y}) \leq 0).
\end{array}$$

To use SIVIA, *we first build an inclusion test* $[t_3]([\mathbf{x}], [\mathbf{y}])$ *for* $t_3(\mathbf{x}, \mathbf{y})$ *and* $[t_4]([\mathbf{x}], [\mathbf{y}])$ *for* $t_4(\mathbf{x}, \mathbf{y})$. *Using the test-projection algorithm, we build the inclusion tests* $[t_1]([\mathbf{x}])$ *and* $[t_2]([\mathbf{x}])$ *for* $t_1(\mathbf{x})$ *and* $t_2(\mathbf{x})$. *Then, we call* SIVIA *with the test* $t_1(\mathbf{x}) \wedge t_2(\mathbf{x})$.

2.3 Bounded-error estimation

2.3.1 Example 1

Consider the model defined by $\phi(\mathbf{p}, t) = p_1 e^{-p_2 t}$ where $t \in \mathbb{R}$ corresponds to time and $\mathbf{p} = (p_1, p_2)^{\mathrm{T}}$ is the parameter vector. A prior feasible box for the parameters $[\mathbf{p}] \subset \mathbb{R}^2$ is assumed to be available. Assume that at times $t \in \{t_1, t_2, \ldots, t_m\}$, we have the following data intervals : $[y_1^-, y_1^+], [y_2^-, y_2^+], \ldots, [y_m^-, y_m^+]$. The *posterior feasible set* is

$$\mathbb{S} = \{\mathbf{p} \in [\mathbf{p}], \phi(\mathbf{p}, t_1) \in [y_1^-, y_1^+], \ldots, \phi(\mathbf{p}, t_m) \in [y_m^-, y_m^+]\}.$$

Or equivalently

$$\mathbb{S} = [\mathbf{p}] \cap \boldsymbol{\phi}^{-1}([\mathbf{y}]).$$

where

$$\boldsymbol{\phi}(\mathbf{p}) = \begin{pmatrix} \phi(\mathbf{p}, t_1) \\ \\ \phi(\mathbf{p}, t_m) \end{pmatrix}$$

and

$$[\mathbf{y}] = [y_1^-, y_1^+] \times \cdots \times [y_m^-, y_m^+]$$

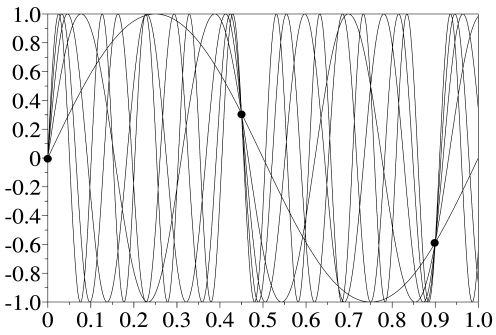
The characterization of $\mathbb{S}$ can thus be done by SIVIA. The software SETDEMO (made by Guillaume Baffet) and available at

`www.ensta-bretagne.fr/jaulin/demo.html`

This software gives an illustration of this characterization for this example.

2.3.2 Example 2

If the model is now given by $\phi(\mathbf{p}, t) = p_1 \sin(2\pi p_2 t)$ where the measurement times are $t_k = k\delta, k \in \mathbb{N}$. As illustrated dynamically by SETDEMO and by the figure below, the posterior feasible set contains an infinite number of connected components.



2.3.3 Robustification against outliers

Consider now the case where outliers occur (e.g., a sensor may fail during data collection, some error bounds turn out to be optimistic, ...)

For this purpose, define a *relaxing function* for the box $[\mathbf{y}] = [y_1] \times \cdots \times [y_n]$

$$\lambda(\mathbf{y}) = \pi_{[y_1]}(y_1) + \cdots + \pi_{[y_n]}(y_n)$$

where

$$\pi_{[a,b]}(x) \begin{cases} = 1 & \text{if} \quad x \in [a, b] \\ = 0 & \text{if} \quad x \notin [a, b]. \end{cases}$$

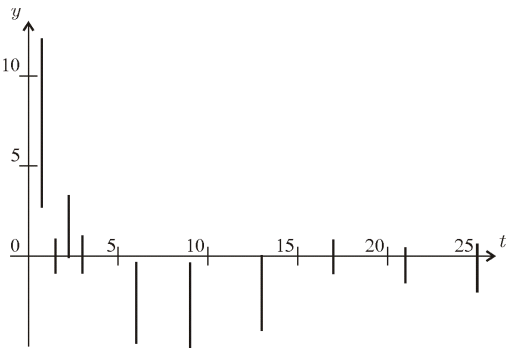
Allow up to q of the n output variables y_i to escape their prior feasible intervals. The posterior feasible set becomes

$$\hat{\mathbb{P}}_q = \{\mathbf{p} \in [\mathbf{p}] \mid \pi_{[y_1]}(\phi_1(\mathbf{p})) + \cdots + \pi_{[y_n]}(\phi_n(\mathbf{p})) \geqslant n - q\}.$$

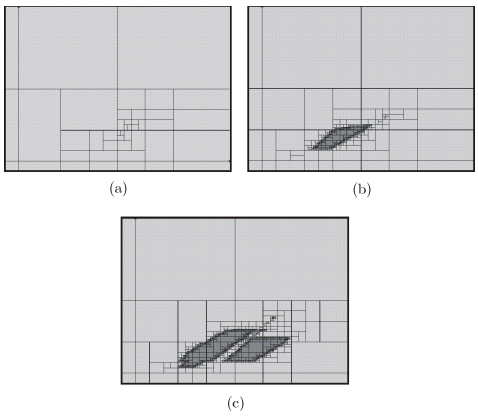
This is a set inversion problem. The set $\hat{\mathbb{P}}_q$ can thus be characterized by SIVIA. As an illustration, consider the model

$$\phi(\mathbf{p}, t) = 20\exp(-p_1 t) - 8\exp(-p_2 t)$$

with the data bars represented on the figure below



SIVIA generates the subpavings depicted the following figure, for $q = 0$ (a), $q = 1$ (b) and $q = 2$, for $\varepsilon = 0.005$ and for the prior search box for the parameters $[\mathbf{p}] = [-0.1, 1.5] \times [-0.1, 1.5]$. $\hat{\mathbb{P}}_0$ turns out to be empty. $\hat{\mathbb{P}}_2$ is disconnected because there are two different strategies to eliminate two interval data in order to be able to be consistent with the eight remaining ones.



(a) no outlier assumed; (b) up to one outlier assumed; (c) up to two outliers assumed; the frames correspond to the parameter box $[-0.1, 1.5] \times [-0.1, 1.5]$

2.4 Robust stability

2.4.1 Routh criterion

Consider a linear system with characteristic polynomial

$$A(s) = a_n s^n + a_{n-1} s^{n-1} + \cdots + a_1 s + a_0$$

Theorem 21 ***(Routh Criterion)***. *Let $P(s) = a_n s^n + \cdots + a_1 s + a_0$ be a polynomial. Its Routh table is defined by*

a_n	a_{n-2}	a_{n-4}	a_{n-6}	$\dots$	0	0
a_{n-1}	a_{n-3}	a_{n-5}	a_{n-7}	$\dots$	0	0
b_1	b_2	b_3			0	0
c_1	c_2	c_3			0	0
$\vdots$	$\vdots$	$\vdots$			$\vdots$	$\vdots$

with

$$\begin{array}{lll} b_1 = \frac{a_{n-1}a_{n-2} - a_n a_{n-3}}{a_{n-1}} & b_2 = \frac{a_{n-1}a_{n-4} - a_n a_{n-5}}{a_{n-1}} & \dots \\ c_1 = \frac{b_1 a_{n-3} - a_{n-1} b_2}{b_1} & c_2 = \frac{b_1 a_{n-5} - a_{n-1} b_3}{b_1} & \dots \\ \dots & \vdots & \end{array}$$

Note that the first two lines are the coefficients of $P(s)$. The other elements of the table t_{ij} are obtained by the following relation

$$t_{ij} = \frac{t_{i-1,1} t_{i-2,j+1} - t_{i-2,1} t_{i-1,j+1}}{t_{i-1,1}}.$$

The roots of $P(s)$ are all stable (i.e., with negative real parts) if and only if all entries of the first column of the table have the same sign.

2.4.2 Stability domain

The *stability domain* $\mathbb{S}_{\mathrm{p}}$ of the polynomial

$$P(s, \mathbf{p}) = s^n + a_{n-1}(\mathbf{p}) s^{n-1} + \ldots + a_1(\mathbf{p}) s + a_0(\mathbf{p})$$

is the set of all $\mathbf{p}$ such that $P(s, \mathbf{p})$ is stable. Consider for instance the polynomial $P(s, \mathbf{p})$ given by

$$s^3 + (p_1 + p_2 + 2)s^2 + (p_1 + p_2 + 2)s + 2p_1p_2 + 6p_1 + 6p_2 + 2 + \sigma^2,$$

$\sigma = 0.5$. Its Routh table is given by

1	$p_1 + p_2 + 2$
$p_1 + p_2 + 2$	$2p_1p_2 + 6p_1 + 6p_2 + 2 + \sigma^2$
$\frac{(p_1+p_2+2)^2 - 2p_1p_2 + 6p_1 + 6p_2 + 2 + \sigma^2}{p_1+p_2+2} = \frac{(p_1-1)^2 + (p_2-1)^2 - \sigma^2}{p_1+p_2+2}$	0
$2p_1p_2 + 6p_1 + 6p_2 + 2 + \sigma^2 = 2(p_1+3)(p_2+3) - 16 + \sigma^2$	0

Its stability domain is thus defined by

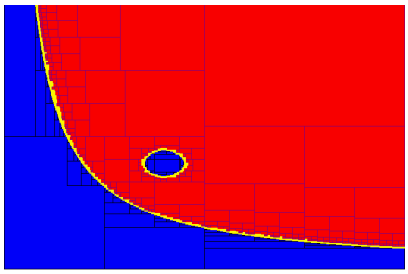
$$\mathbb{S}_{\mathrm{p}} \triangleq \{\mathbf{p} \in \mathbb{R}^n \mid \mathbf{r}(\mathbf{p}) > \mathbf{0}\} = \mathbf{r}^{-1}\left(]0, +\infty[^{\times n}\right).$$

where

$$\mathbf{r}(\mathbf{p}) = \begin{pmatrix} p_1 + p_2 + 2 \\ (p_1 - 1)^2 + (p_2 - 1)^2 - \sigma^2 \\ 2(p_1 + 3)(p_2 + 3) - 16 + \sigma^2 \end{pmatrix}.$$

The corresponding set, obtained by PROJ2D is represented on the following figure. The solver PROJ2D is available at

www.ensta-bretagne.fr/jaulin/demo.html/



Stability domain $\mathbb{S}_p$ generated by PROJ2D

Chapter 3

Contractors

To characterize the set $\mathbb{X} \subset \mathbb{R}^n$, bisection-based algorithms, such as SIVIA, need to bisect all boxes in all directions. Now, for $n = 20$, bisecting a box in all directions generates $2^{20} = 1\,048\,576$ boxes. Therefore, for high dimensions, bisections should be avoided as much as possible. When the solution set $\mathbb{X}$ is small (optimization problem, solving equations), contraction procedures can be used to characterize it, even n is huge. Several bisections can still be performed, but they should be used only as a last resort.

3.1 Definition of a contractor

The degenerated box made with a single point $\mathbf{x}$ will be denoted by $\{\mathbf{x}\}$ or simply by $\mathbf{x}$. The operator $\mathcal{C} : \mathbb{IR}^n \to \mathbb{IR}^n$ is a *contractor* if

$$\begin{array}{lll} \text{(i)} & \forall [\mathbf{x}] \in \mathbb{IR}^n, \mathcal{C}([\mathbf{x}]) \subset [\mathbf{x}] & \text{(contractance)} \\ \text{(ii)} & (\mathbf{x} \in [\mathbf{x}],\ \mathcal{C}(\{\mathbf{x}\}) = \{\mathbf{x}\}) \Rightarrow \mathbf{x} \in \mathcal{C}([\mathbf{x}]) & \text{(consistence)} \\ \text{(iii)} & \mathcal{C}(\{\mathbf{x}\}) = \emptyset \Rightarrow (\exists \varepsilon > 0, \forall [\mathbf{x}] \subset B(\mathbf{x}, \varepsilon),\ \mathcal{C}([\mathbf{x}]) = \emptyset) & \text{(weak continuity)} \end{array} \tag{3.1}$$

where $B(\mathbf{x}, \varepsilon)$ is the ball with center $\mathbf{x}$ and radius ε. A box $[\mathbf{x}]$ is said to be *insensitive* to $\mathcal{C}$ if $\mathcal{C}([\mathbf{x}]) = [\mathbf{x}]$. Form Property (i) boxes can only by contracted. Form Property (ii) an insensitive point $\mathbf{x}$ is never removed by $\mathcal{C}$. From Property (iii), the set off all insensitive $\mathbf{x}$ is closed.

Counterexample. The operator $\mathcal{C} : \mathbb{IR} \to \mathbb{IR}$ defined by

$$\begin{array}{rcl} \mathcal{C}([a,b]) & = & [a, \dfrac{a+b}{2}],\ \text{if } a \neq b \\ & = & \emptyset \quad \text{if } a = b \end{array}$$

does not satisfy the weak continuity. Note that if we consider the sequence $[x](k+1) = \mathcal{C}([x](k))$, $[x](0) = [0,1]$, we have $\lim_{n\to\infty} \mathcal{C}([x]) = [0,0]$ whereas $\mathcal{C}([0,0]) = \emptyset$.

Figure 3.1 illustrates the problem that could arise if the weak continuity is not satisfied, for a sequence of $\mathbb{R}$ defined by $x(k+1) = \mathcal{C}(x(k))$. The operator has a unique fixed point given by $x = 0$.

The *set* (or *constraint*) associated with a contractor $\mathcal{C}$ is the union of all insensitive singletons of $\mathcal{C}$, i.e.,

$$\text{set}(\mathcal{C}) = \{\mathbf{x} \in \mathbb{R}^n, \mathcal{C}(\{\mathbf{x}\}) = \{\mathbf{x}\}\}.$$

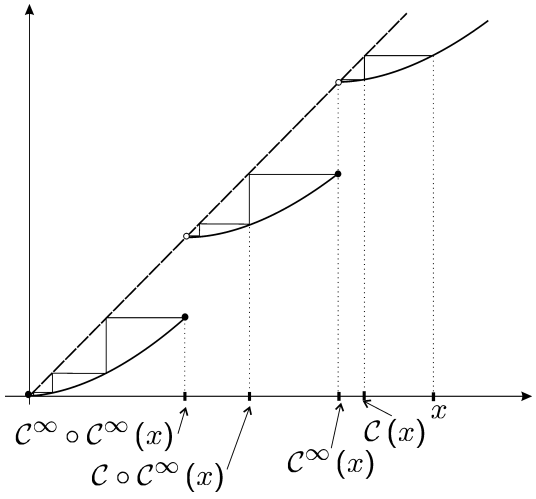


Figure 3.1: Illustration of the type of discontinuity that can be found when dealing with fixed point algorithms

A contractor represents subsets of $\mathbb{R}^n$. Set operations such as intersections, union, inversion, are easy to perform with contractors.

We have the following definitions

$\mathcal{C}$ is *monotonic* if	$[\mathbf{x}] \subset [\mathbf{y}] \Rightarrow \mathcal{C}([\mathbf{x}]) \subset \mathcal{C}([\mathbf{y}])$
$\mathcal{C}$ is *minimal* if	$\forall [\mathbf{x}] \in \mathbb{IR}^n,\ \mathcal{C}([\mathbf{x}]) = [[\mathbf{x}] \cap \text{set}\,(\mathcal{C})]$
$\mathcal{C}$ is *idempotent* if	$\forall [\mathbf{x}] \in \mathbb{IR}^n, \mathcal{C}\,(\mathcal{C}([\mathbf{x}])) = \mathcal{C}([\mathbf{x}]),$
$\mathcal{C}$ is *continuous* if	$\forall [\mathbf{x}] \in \mathbb{IR}^n, \mathcal{C}\,(\mathcal{C}^\infty([\mathbf{x}])) = \mathcal{C}^\infty([\mathbf{x}])$

A contractor satisfies the *completeness* property:

$$\mathcal{C}([\mathbf{x}]) \supset [\mathbf{x}] \cap \text{set}\,(\mathcal{C})$$

which is a consequence (3.1, ii). All minimal contractors are idempotent.

Example. A *precision contractor* is defined by

$$\mathcal{C}_\varepsilon\,([\mathbf{x}]) \begin{cases} = [\mathbf{x}] & \text{if } w\,([\mathbf{x}]) > \varepsilon \\ = \emptyset & \text{otherwise} \end{cases}$$

where $\varepsilon > 0$. This contractor is monotonic and idempotent but is not minimal. Moreover $\text{set}(\mathcal{C}_\varepsilon) = \emptyset$. Note that if $\varepsilon = 0$, then $\mathcal{C}_\varepsilon$ is not a contractor anymore, due to the continuity condition.

3.2 Operations over contractors

3.2.1 Definitions

We define the following operations

intersection	$(\mathcal{C}_1 \cap \mathcal{C}_2)([\mathbf{x}]) \stackrel{\text{def}}{=} \mathcal{C}_1([\mathbf{x}]) \cap \mathcal{C}_2([\mathbf{x}])$
union	$(\mathcal{C}_1 \cup \mathcal{C}_2)([\mathbf{x}]) \stackrel{\text{def}}{=} [\mathcal{C}_1([\mathbf{x}]) \cup \mathcal{C}_2([\mathbf{x}])]$
composition	$(\mathcal{C}_1 \circ \mathcal{C}_2)([\mathbf{x}]) \stackrel{\text{def}}{=} \mathcal{C}_1(\mathcal{C}_2([\mathbf{x}]))$
repetition	$\mathcal{C}^\infty \stackrel{\text{def}}{=} \mathcal{C} \circ \mathcal{C} \circ \mathcal{C} \circ \dots$
repeat intersection	$\mathcal{C}_1 \sqcap \mathcal{C}_2 = (\mathcal{C}_1 \cap \mathcal{C}_2)^\infty$
repeat union	$\mathcal{C}_1 \sqcup \mathcal{C}_2 = (\mathcal{C}_1 \cup \mathcal{C}_2)^\infty$
central symmetry	$(\mathcal{S}_{\mathbf{a}} \circ \mathcal{C})([\mathbf{x}]) = \mathcal{S}_{\mathbf{a}} \circ \mathcal{C} \circ \mathcal{S}_{\mathbf{a}}([\mathbf{x}])$
axial symmetry	$(\mathcal{S}_{\mathbf{u}} \circ \mathcal{C})([\mathbf{x}]) = \mathcal{S}_{\mathbf{u}} \circ \mathcal{C} \circ \mathcal{S}_{\mathbf{u}}([\mathbf{x}])$
translation	$(\mathcal{T}_{\mathbf{u}} \circ \mathcal{C})([\mathbf{x}]) = \mathcal{T}_{\mathbf{u}} \circ \mathcal{C} \circ \mathcal{T}_{-\mathbf{u}}([\mathbf{x}])$
modulo	$(\mathcal{C} \bmod \mathbf{u})([\mathbf{x}]) = \sqcup_i \mathcal{T}_{i.\mathbf{u}} \circ \mathcal{C}([\mathbf{x}])$

It is trivial to prove that these operations always produce contractors which satisfy the properties (3.1).

Proof. We shall limit ourselves to the intersection. We only have to prove that the result of the intersection satisfies the three properties (3.1). The contractance property is satisfied since

$$(\mathcal{C}_1 \cap \mathcal{C}_2)([\mathbf{x}]) \;=\; \mathcal{C}_1([\mathbf{x}]) \cap \mathcal{C}_2([\mathbf{x}]) \;\subset\; [\mathbf{x}] \cap [\mathbf{x}] \;=\; [\mathbf{x}].$$

Let us now prove the consistence. We have

$$\begin{array}{rl}
 & \mathbf{x} \in [\mathbf{x}], (\mathcal{C}_1 \cap \mathcal{C}_2)(\{\mathbf{x}\}) = \{\mathbf{x}\} \\
\Rightarrow & \mathbf{x} \in [\mathbf{x}], \mathcal{C}_1(\{\mathbf{x}\}) \cap \mathcal{C}_2(\{\mathbf{x}\}) = \{\mathbf{x}\} \\
\Rightarrow & \mathbf{x} \in [\mathbf{x}], \mathcal{C}_1(\{\mathbf{x}\}) = \{\mathbf{x}\} \text{ and } \mathcal{C}_2(\{\mathbf{x}\}) = \{\mathbf{x}\} \\
\Rightarrow & \left\{ \begin{array}{l} \mathbf{x} \in [\mathbf{x}], \mathcal{C}_1(\{\mathbf{x}\}) = \{\mathbf{x}\} \text{ and} \\ \mathbf{x} \in [\mathbf{x}], \mathcal{C}_2(\{\mathbf{x}\}) = \{\mathbf{x}\} \end{array} \right. \\
\stackrel{(3.1,\text{ii})}{\Rightarrow} & \mathbf{x} \in \mathcal{C}_1([\mathbf{x}]) \text{ and } \mathbf{x} \in \mathcal{C}_2([\mathbf{x}]) \\
\Rightarrow & \mathbf{x} \in \mathcal{C}_1([\mathbf{x}]) \cap \mathcal{C}_2([\mathbf{x}]) \\
\Rightarrow & \mathbf{x} \in (\mathcal{C}_1 \cap \mathcal{C}_2)([\mathbf{x}]).
\end{array}$$

For the weak continuity, we get :

$$\begin{array}{rl}
 & (\mathcal{C}_1 \cap \mathcal{C}_2)(\{\mathbf{x}\}) = \emptyset \\
\Rightarrow & \mathcal{C}_1(\{\mathbf{x}\}) = \emptyset \text{ or } \mathcal{C}_2(\{\mathbf{x}\}) = \emptyset \\
\stackrel{(3.1,\text{iii})}{\Rightarrow} & \left\{ \begin{array}{l} \exists \varepsilon_1 > 0, \forall [\mathbf{x}] \subset B(\mathbf{x}, \varepsilon_1), \mathcal{C}_1([\mathbf{x}]) = \emptyset \text{ or} \\ \exists \varepsilon_2 > 0, \forall [\mathbf{x}] \subset B(\mathbf{x}, \varepsilon_2), \mathcal{C}_2([\mathbf{x}]) = \emptyset \end{array} \right. \\
\stackrel{\varepsilon=\min(\varepsilon_1,\varepsilon_2)}{\Rightarrow} & \exists \varepsilon > 0, \forall [\mathbf{x}] \subset B(\mathbf{x}, \varepsilon), \mathcal{C}_1([\mathbf{x}]) = \emptyset \text{ or } \mathcal{C}_2([\mathbf{x}]) = \emptyset \\
\Rightarrow & \exists \varepsilon > 0, \forall [\mathbf{x}] \subset B(\mathbf{x}, \varepsilon), \mathcal{C}_1([\mathbf{x}]) \cap \mathcal{C}_2([\mathbf{x}]) = \emptyset \\
\Rightarrow & \exists \varepsilon > 0, \forall [\mathbf{x}] \subset B(\mathbf{x}, \varepsilon), (\mathcal{C}_1 \cap \mathcal{C}_2)([\mathbf{x}]) = \emptyset.
\end{array}$$

■

Note the composition does not commute (i.e., $\mathcal{C}_1 \circ \mathcal{C}_2 \neq \mathcal{C}_2 \circ \mathcal{C}_1$) contrary to the following operators $\cap, \cup, \sqcup, \sqcap$. We defined the inclusion between contractors as follows

$$\mathcal{C}_1 \subset \mathcal{C}_2 \Leftrightarrow \forall [\mathbf{x}] \in \mathbb{IR}^n, \mathcal{C}_1([\mathbf{x}]) \subset \mathcal{C}_2([\mathbf{x}]).$$

Proposition: If $\mathcal{C}$ is a monotonic continuous contractor, the set of all $\mathcal{C}$-insensitive boxes equipped with the inclusion $\subset$ is a complete lattice. Moreover

$$\mathcal{C}^{\infty}([\mathbf{x}]) = \sup_{\subset} \{[\mathbf{a}], \mathcal{C}([\mathbf{a}]) = [\mathbf{a}]\},$$

i.e., $\mathcal{C}^{\infty}([\mathbf{x}])$ corresponds the the largest subbox of $[\mathbf{x}]$ insensitive to $\mathcal{C}$.

Proof. This theorem is a consequence of the Tarski theorem which states that if $\mathcal{L}$ be a complete lattice and let $f : \mathcal{L} \to \mathcal{L}$ be an order-preserving function. Then the set of fixed points of f in $\mathcal{L}$ is also a complete lattice.

Theorem : The set of all idempotent monotonic continuous contractors, equipped with the inclusion, is a complete lattice $\mathcal{T}_C$. The two corresponding operators (*inf* and *sup*) are $\mathcal{C}_1 \sqcup \mathcal{C}_2$ and $\mathcal{C}_1 \sqcap \mathcal{C}_2$. The smallest element $\mathcal{C}^{\perp}$ of $\mathcal{T}_C$ is the empty contractor and the largest element $\mathcal{C}^{\top}$ is the identity:

$$\forall [\mathbf{x}] \in \mathbb{IR}^n,\ \mathcal{C}^{\perp}([\mathbf{x}]) = \emptyset \text{ and } \mathcal{C}^{\top}([\mathbf{x}]) = [\mathbf{x}].$$

Proof : First, let us note that $(\mathcal{T}_C, \subset)$ is partially ordered. In order to show that this set is a lattice, we have to check *(i)* the commutativity, *(ii)* the associativity, *(iii)* the idempotence and *(iv)* the absorption.
(i) Commutativity: Let us first show that $\mathcal{C}_1 \sqcap \mathcal{C}_2 = \mathcal{C}_2 \sqcap \mathcal{C}_1$. Using the monotonicity of contractors, we have

$$(\mathcal{C}_1 \sqcap \mathcal{C}_2)([\mathbf{x}]) \stackrel{\text{def}}{=} (\mathcal{C}_1 \cap \mathcal{C}_2)^{\infty}([\mathbf{x}]) = [\mathbf{a}],$$

where $[\mathbf{a}]$ represents the largest subbox of $[\mathbf{x}]$ such that $\mathcal{C}_1([\mathbf{a}]) = [\mathbf{a}]$ and $\mathcal{C}_2([\mathbf{a}]) = [\mathbf{a}]$. The same reasoning for $\mathcal{C}_2 \sqcap \mathcal{C}_1([\mathbf{x}])$ leads to the same box $[\mathbf{a}]$. Thus, we get $\mathcal{C}_1 \sqcap \mathcal{C}_2 = \mathcal{C}_2 \sqcap \mathcal{C}_1$. In a similar way, we easily prove that $\mathcal{C}_1 \sqcup \mathcal{C}_2 = \mathcal{C}_2 \sqcup \mathcal{C}_1$.
(ii) Associativity : We easily check that $\mathcal{C}_1 \sqcap (\mathcal{C}_2 \sqcap \mathcal{C}_3) = (\mathcal{C}_1 \sqcap \mathcal{C}_2) \sqcap \mathcal{C}_3$ and $\mathcal{C}_1 \sqcup (\mathcal{C}_2 \sqcup \mathcal{C}_3) = (\mathcal{C}_1 \sqcup \mathcal{C}_2) \sqcup \mathcal{C}_3$..
(iii) Idempotence : We have $\mathcal{C}_1 \sqcap \mathcal{C}_1 = \mathcal{C}_1$ and $\mathcal{C}_1 \sqcup \mathcal{C}_1 = \mathcal{C}_1$.
(iii) Absorption : We have $\mathcal{C}_1 \sqcap (\mathcal{C}_1 \sqcup \mathcal{C}_2) = \mathcal{C}_1$ and $\mathcal{C}_1 \sqcup (\mathcal{C}_1 \sqcap \mathcal{C}_2) = \mathcal{C}_1$.
To prove that the lattice is complete, we have to show that any (possibly infinite) collection of contractors, admits a smallest and a greatest element. The intersection of an (infinite) number of boxes always produces a box. For the union, we can converge to an open set. We thus define the union of an infinite number of boxes as

$$\bigsqcup_{i=1}^{\infty} \mathcal{C}_i = \left[\lim_{n=\infty} \mathcal{C}_1 \sqcup \mathcal{C}_2 \sqcup \cdots \sqcup \mathcal{C}_n\right].$$

The union of a collection of boxes will thus always be closed box. ■

Unfortunately, this lattice is not distributive. Instead, we have the subdistributivity property:

$$(\mathcal{C}_1 \sqcap (\mathcal{C}_2 \sqcup \mathcal{C}_3)) \supset (\mathcal{C}_1 \sqcap \mathcal{C}_2) \sqcup (\mathcal{C}_1 \sqcap \mathcal{C}_3).$$

Figure 3.2 provides a counter example of a potential distributivity. From the three sets $\mathbb{S}_1$,$\mathbb{S}_2, \mathbb{S}_3$ we can build the optimal contractors $\mathcal{C}_i([\mathbf{x}]) = [\mathbb{S}_i \cap [\mathbf{x}]]$. We have $(\mathcal{C}_1 \sqcap (\mathcal{C}_2 \sqcup \mathcal{C}_3))([\mathbf{x}]) = [\mathbf{a}]$ whereas $(\mathcal{C}_1 \sqcap \mathcal{C}_2) \sqcup (\mathcal{C}_1 \sqcap \mathcal{C}_3)([\mathbf{x}]) = \emptyset$.

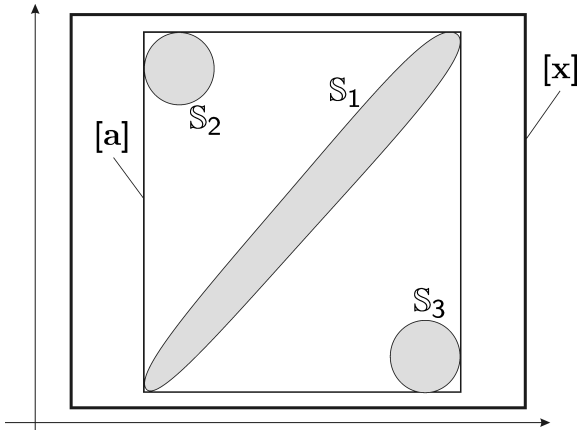


Figure 3.2: Counter-example of a potential distributivity of monotonic idempotent contractors

We have the following properties.

$$\begin{array}{l} \text{set}\,(\mathcal{C}_1 \sqcap \mathcal{C}_2) = \text{set}\,(\mathcal{C}_1 \cap \mathcal{C}_2) = \text{set}\,(\mathcal{C}_1) \cap \text{set}\,(\mathcal{C}_2) = \text{set}\,(\mathcal{C}_1 \circ \mathcal{C}_2) \\ \text{set}\,(\mathcal{C}_1 \sqcup \mathcal{C}_2) = \text{set}\,(\mathcal{C}_1) \cup \text{set}\,(\mathcal{C}_2) \\ \text{set}\,(\mathcal{C}_1^{\infty}) = \text{set}\,(\mathcal{C}_1) \\ \mathcal{C}_1 \sqcap \mathcal{C}_2 \subset \mathcal{C}_1 \circ \mathcal{C}_2 \subset \mathcal{C}_1 \cap \mathcal{C}_2 \text{ (if } \mathcal{C}_1 \text{ is inclusion monotonic).} \end{array} \tag{3.2}$$

These properties have some strong connections with the continuous lattice theory [79].

3.2.2 Unique repeat principle

If $\mathcal{C}_1$ and $\mathcal{C}_2$ are monotonic, we have

$$\mathcal{C}_1^{\infty} \sqcap \mathcal{C}_2^{\infty} \;= (\mathcal{C}_1^{\infty} \circ \mathcal{C}_2^{\infty})^{\infty} = \; \mathcal{C}_1 \sqcap \mathcal{C}_2 = \; (\mathcal{C}_1 \circ \mathcal{C}_2)^{\infty}\,. \tag{3.3}$$

These equations make possible to simplify some expressions involving contractors in order to have a unique repeat. For instance, consider three contractors $\mathcal{C}_1, \mathcal{C}_2$ and $\mathcal{C}_3$. The expression $((\mathcal{C}_1^{\infty} \circ \mathcal{C}_2^{\infty})^{\infty} \sqcap \mathcal{C}_3^{\infty})$ involves five repeats. We can obtain a unique repeat:

$$\begin{array}{rcl} ((\mathcal{C}_1^{\infty} \circ \mathcal{C}_2^{\infty})^{\infty} \sqcap \mathcal{C}_3^{\infty}) & = & ((\mathcal{C}_1 \circ \mathcal{C}_2)^{\infty} \sqcap \mathcal{C}_3^{\infty}) \\ & = & ((\mathcal{C}_1 \circ \mathcal{C}_2) \sqcap \mathcal{C}_3) \\ & = & (\mathcal{C}_1 \circ \mathcal{C}_2 \circ \mathcal{C}_3)^{\infty}\,. \end{array} \tag{3.4}$$

In practice, when we want to have an efficient implementation of a contractor defined by an expression, one should try to transform the expression in order to get a single repeat.

Remark. For the union, we only have the following inclusion

$$\mathcal{C}_1^{\infty} \sqcup \mathcal{C}_2^{\infty} \quad \subset \quad \mathcal{C}_1 \sqcup \mathcal{C}_2.$$

As a consequence, the unique repeat principle does not apply anymore when union operators are involved.

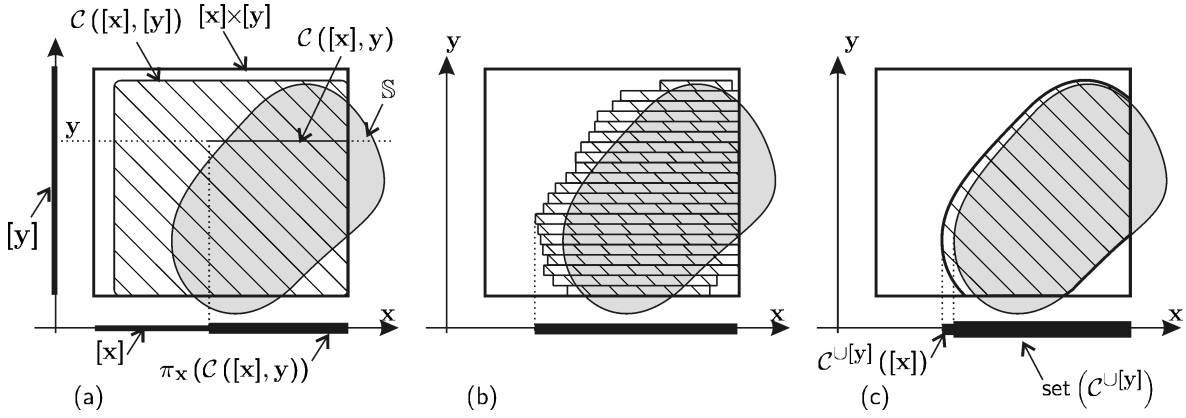


Figure 3.3: Illustration of the union-projection

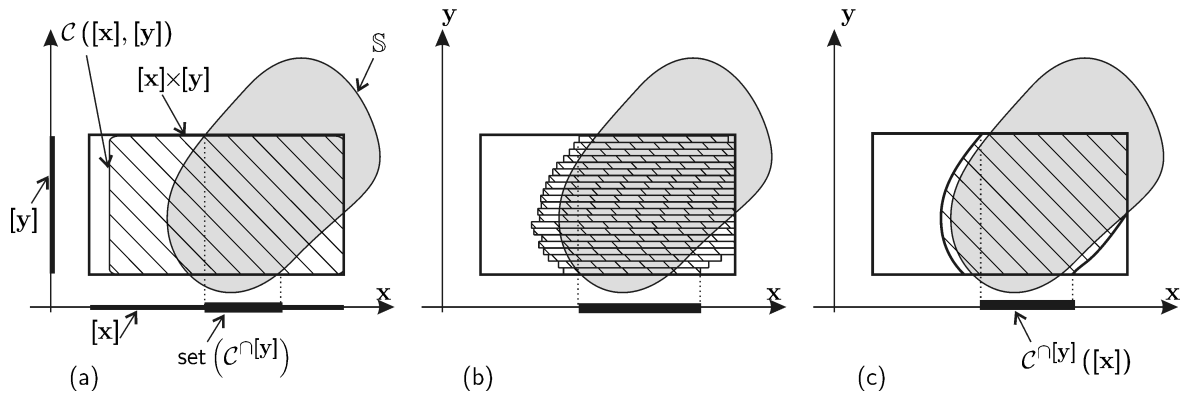


Figure 3.4: Illustration of the intersection-projection

3.2.3 Projections

Consider the contractor $\mathcal{C}([\mathbf{x}],[\mathbf{y}])$, where $[\mathbf{x}] \in \mathbb{R}^n, [\mathbf{y}] \in \mathbb{R}^p$. We define the two following contractors:

$$\begin{cases} \mathcal{C}^{\cup[\mathbf{y}]}([\mathbf{x}]) &= \left[\bigcup_{\mathbf{y}\in[\mathbf{y}]} \pi_{\mathbf{x}}(\mathcal{C}([\mathbf{x}],\mathbf{y}))\right] \\ \mathcal{C}^{\cap[\mathbf{y}]}([\mathbf{x}]) &= \bigcap_{\mathbf{y}\in[\mathbf{y}]} \pi_{\mathbf{x}}(\mathcal{C}([\mathbf{x}],\mathbf{y})), \end{cases} \tag{3.5}$$

where the operator $\pi_x([\mathbf{x}],[\mathbf{y}])$ represents the projection $[\mathbf{x}]$ of the box $[\mathbf{x}]\times[\mathbf{y}]$. The two following operators

$$\begin{array}{lll} \mathcal{C}([\mathbf{x}],[\mathbf{y}]) & \rightarrow & \mathcal{C}^{\cup[\mathbf{y}]}([\mathbf{x}]) \\ \mathcal{C}([\mathbf{x}],[\mathbf{y}]) & \rightarrow & \mathcal{C}^{\cap[\mathbf{y}]}([\mathbf{x}]) \end{array} \tag{3.6}$$

make it possible to build the two contractors $\mathcal{C}^{\cup[\mathbf{y}]}([\mathbf{x}])$ and $\mathcal{C}^{\cap[\mathbf{y}]}([\mathbf{x}])$ from the contractor $\mathcal{C}([\mathbf{x}],[\mathbf{y}])$ are called *union-projection* and *intersection-projection*, respectively. Figure 3.3 illustrates the union-projection. Figure 3.4 illustrates the intersection-projection.

We have the following properties

$$\begin{array}{ll} \text{(i)} & \mathcal{C}^{\cap[\mathbf{y}]} \subset \mathcal{C}^{\cup[\mathbf{y}]}, \\ \text{(ii)} & \mathcal{C}^{\cup[\mathbf{y}]} \text{ and } \mathcal{C}^{\cap[\mathbf{y}]} \text{ are contractors} \\ \text{(iii)} & \text{set}\left(\mathcal{C}^{\cup[\mathbf{y}]}\right) = \{\mathbf{x}, \exists \mathbf{y} \in [\mathbf{y}], (\mathbf{x},\mathbf{y}) \in \text{set}\,(\mathcal{C})\} \\ \text{(iv)} & \text{set}\left(\mathcal{C}^{\cap[\mathbf{y}]}\right) = \{\mathbf{x}, \forall \mathbf{y} \in [\mathbf{y}], (\mathbf{x},\mathbf{y}) \in \text{set}\,(\mathcal{C})\} \end{array}$$

These notions are used to compute with sets defined with quantifiers $\forall, \exists$, such as the projection of a set.

A collection of contractors $\{\mathcal{C}_1, \ldots, \mathcal{C}_m\}$ is said to be *complementary* if

$$\text{set}\,(\mathcal{C}_1) \cap \cdots \cap \text{set}\,(\mathcal{C}_m) = \emptyset.$$

QUIMPER [11][12], is an interpreted language that makes possible to handle easily contractors in order to solve set-valued problems. A QUIMPER program is composed with a collection of complementary contractors. When a QUIMPER program is run, it first builds a collection of contractors. Then a paver is thrown. A *paver* is a simple bisection algorithm which call all available contractors on all current boxes. Only boxes that cannot be contracted by any contractor are allowed to be bisected. The paver explores all the search box $\mathbb{R}^n$ and store all contracted zones onto *subpavings*. To each contractor is associated a single subpaving. Since all contractors are complementary, the paver terminates..

3.2.4 Relaxed intersection

Consider m sets $\mathbb{X}_1, \ldots, \mathbb{X}_m$ of $\mathbb{R}^n$. The q-relaxed intersection denoted by $\bigcap^{\{q\}} \mathbb{X}_i$ is the set of all $\mathbf{x} \in \mathbb{R}^n$ which belong to all $\mathbb{X}_i$'s, except q at most. Fig. 3.5 illustrates this notion for $m = 6$ and $q = 2, 3, 4$. For this example, we have

$$\bigcap^{\{0\}} \mathbb{X}_i = \bigcap^{\{1\}} \mathbb{X}_i = \emptyset, \ \bigcap^{\{5\}} \mathbb{X}_i = \bigcup \mathbb{X}_i \text{ and } \bigcap^{\{6\}} \mathbb{X}_i = \mathbb{R}^2.$$

Example. Consider for instance the 8 intervals $\mathbb{X}_1 = [1,4]$, $\mathbb{X}_2 = [2,4], \mathbb{X}_3 = [2,7], \mathbb{X}_4 = [6,9], \mathbb{X}_5 = [3,4], \mathbb{X}_6 = [3,7]$. We have

$$\begin{array}{lll} \bigcap^{\{0\}} \mathbb{X}_i & = & \emptyset, \ \bigcap^{\{1\}} \mathbb{X}_i = [3,4], \ \bigcap^{\{2\}} \mathbb{X}_i = [3,4], \ \bigcap^{\{3\}} \mathbb{X}_i = [2,4] \cup [6,7], \\ \bigcap^{\{4\}} \mathbb{X}_i & = & [2,7], \ \bigcap^{\{5\}} \mathbb{X}_i = [1,9], \ \bigcap^{\{6\}} \mathbb{X}_i = \mathbb{R}. \end{array}$$

In the case where the $\mathbb{X}_i$'s are intervals, the relaxed intersection can be computed efficiently with a complexity of $n \log n$. Let us now describe a possible method.

- Take all bounds of all intervals with their brackets. For our example, the bounds are

Bounds	1	4	2	4	2	7	6	9	3	4	3	7
Brackets	[	]	[	]	[	]	[	]	[	]	[	]

- Sort the columns with respect the bounds. We get

Bounds	1	2	2	3	3	4	4	4	6	7	7	9
Brackets	[	[	[	[	[	]	]	]	[	]	]	]

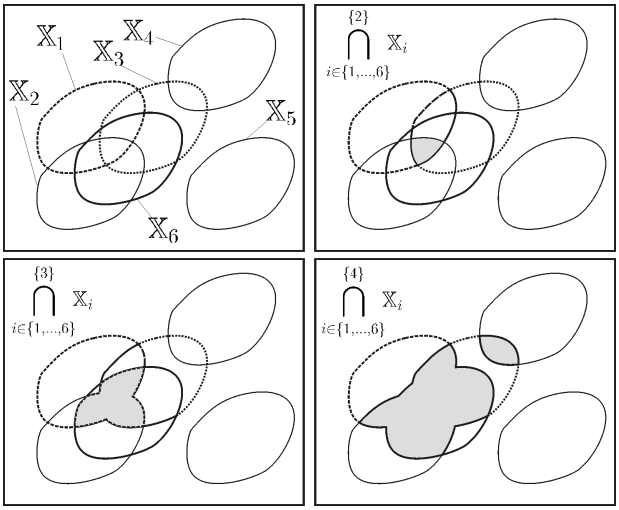


Figure 3.5: Illustration (in gray) of the q-relaxed intersection the 6 sets $\mathbb{X}_1, \ldots, \mathbb{X}_6$ where $q \in \{2, 3, 4\}$

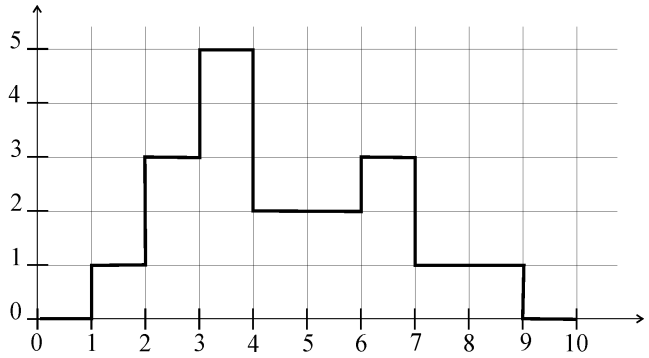


Figure 3.6: Set-membership function associated with the 6 intervals

- Scan these bounds from the left to the right, counting $+1$, when the bound is associated with a left bracket and -1 otherwise. We get

Bounds	1	2	2	3	3	4	4	4	6	7	7	9
Brackets	[	[	[	[	[	]	]	]	[	]	]	]
Sum	1	2	3	4	5	4	3	2	3	2	1	0

- The accumulation corresponds to the set membership function $\mu(x)$ which is the number of intervals to which x belongs (see Figure 3.5). From this function, we directly read the relaxed intersections.

We define the q-relaxed intersection between m contractors

$$\mathcal{C} = \left(\bigcap_{i\in\{1,\ldots,m\}}^{\{q\}} \mathcal{C}_i \right) \Leftrightarrow \forall [\mathbf{x}] \in \mathbb{IR}^n, \mathcal{C}([\mathbf{x}]) = \bigcap^{\{q\}} \mathcal{C}_i([\mathbf{x}]).$$

Computing the q relaxed intersection of m boxes has a polynomial complexity, if the dimension n of the boxes is fixed (see, *e.g.*, [2]), but the complexity of this problem is exponential with respect to n. Fig. 3.7

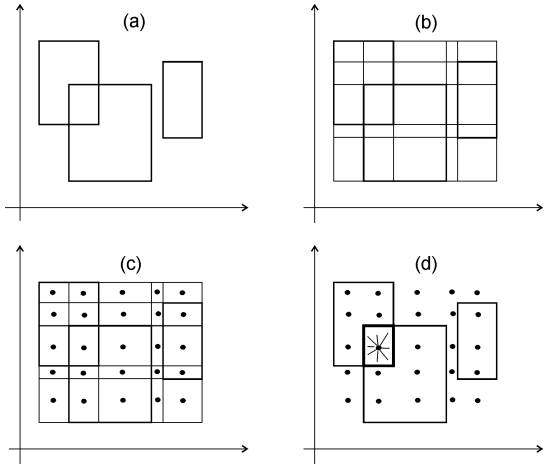


Figure 3.7: Illustration of the principle of polynomial method which computes the smallest box which contains the q-relaxed intersection of m boxes. Here, $q = 1, m = 3$.

illustrates the principle of such an algorithm. First, generate $(2m-1)^n$ boxes as on the subfigure (b). Select all boxes whose centers belong to at least $m-q$ of the m initial boxes Finally, take the enveloping box of all selected boxes.

3.3 Building contractors

3.3.1 Ternary constraints

Consider a constraint $\mathcal{C}$ (*i.e.*, an equation or an inequality), some variables $x_1, x_2, \ldots$ involved in $\mathcal{C}$ and prior interval domains $[x_i]$ for the x_i's. Interval arithmetic makes it possible to contract the domains $[x_i]$ without removing any feasible values for the x_i's. For instance, consider the equation $x_3 = x_1 + x_2$ where the domains for x_1, x_2, x_3 are given by $[x_1] = [-\infty, 5], [x_2] = [-\infty, 4]$ and $[x_3] = [6, \infty]$. These domains can be contracted to $[\bar{x}_1] = [2, 5], [\bar{x}_2] = [1, 4]$ and $[\bar{x}_3] = [6, 9]$. The resulting interval calculation is as follows:

$$\begin{aligned}
x_3 = x_1 + x_2 \Rightarrow \quad z \in \quad & [6,\infty] \cap ([-\infty,5] + [-\infty,4]) \\
& = [6,\infty] \cap [-\infty,9] = [6,9]. \\
x_1 = x_3 - x_2 \Rightarrow \quad x \in \quad & [-\infty,5] \cap ([6,\infty] - [-\infty,4]) \\
& = [-\infty,5] \cap [2,\infty] = [2,5]. \\
x_2 = x_3 - x_1 \Rightarrow \quad y \in \quad & [-\infty,4] \cap ([6,\infty] - [-\infty,5]) \\
& = [-\infty,4] \cap [1,\infty] = [1,4].
\end{aligned}$$

This contraction procedure (which corresponds to a contractor) can be performed with much more complex constraints. The contraction illustrated above can be described by the following procedure, where

PPLUS stands for *Projection of the constraint* PLUS.

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

The projection procedure developed for PLUS can be extended to other ternary constraints such as MULT: $z = x * y$, or equivalently

$$\text{MULT} \triangleq \left\{(x, y, z) \in \mathbb{R}^3 | z = x * y\right\}.$$

The resulting projection procedure becomes

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

For the MAX ternary constraint defined by $z = \max(x, y)$ the optimal contractor can be implemented by using the following algorithm

Algorithm PMAX(inout: $[z], [x], [y]$)
1 $[z] := [z] \cap (\max([x], [y]))$;
2 if $([x] \cap [z] = \emptyset)$, $[y] := [y] \cap [z]$;
3 if $([y] \cap [z] = \emptyset)$, $[x] := [x] \cap [z]$;
4 $[x] := [x] \cap [-\infty, z^+]$;
3 $[y] := [y] \cap [-\infty, z^+]$.

For instance

$$\begin{aligned}
\text{PMAX}([5,10],[0,6],[1,2]) &\rightarrow ([5,6],[0,6],[1,2]) \\
\text{PMAX}([5,10],[3,6],[1,2]) &\rightarrow ([5,6],[5,6],[1,2]) \\
\text{PMAX}([-1,1],[0,6],[1,2]) &\rightarrow ([0,1],[0,1],[1,1])
\end{aligned}$$

3.3.2 Binary constraints

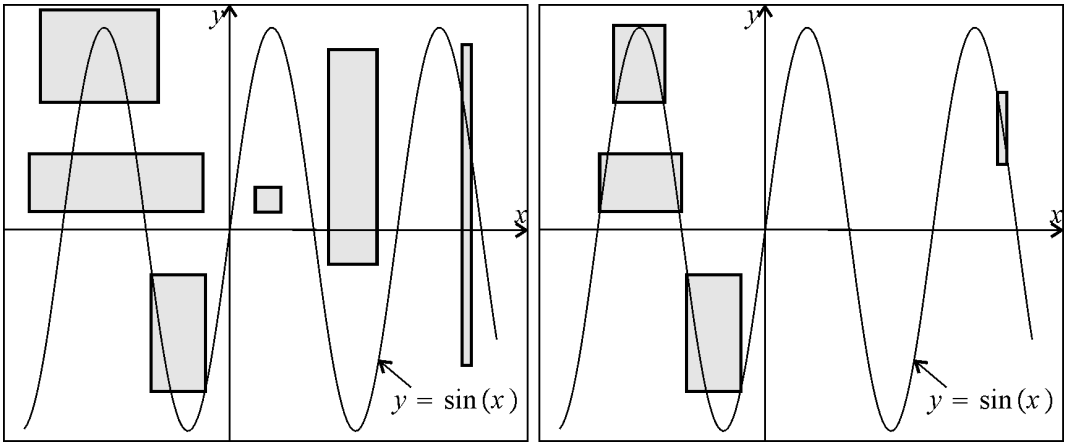
Elementary functions can be interpreted as binary constraints and receive a similar treatment. Consider for instance, the binary constraint

$$\text{EXP} \triangleq \{(x, y) \in \mathbb{R}^n | y = \exp(x)\}.$$

The associated projection procedure becomes

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

Any constraint for which such a projection procedure is available will be called a *primitive constraint*. The figure below illustrates the contraction with respect to the primitive constraint $y = \sin(x)$.



(Left) before contraction; (Right) after contraction

3.4 Propagation

3.4.1 Constraint Satisfaction Problem (CSP)

A CSP is composed of

- a set of variables $\mathcal{V} = \{x_1, \ldots, x_n\}$,
- a set of constraints $\mathcal{C} = \{c_1, \ldots, c_m\}$ and
- a set of interval domains $\{[x_1], \ldots, [x_n]\}$.

The aim of propagation techniques is to contract as much as possible the domains for the variables without loosing any solution. Denote by $[\mathbf{x}]$ the box defined by the Cartesian product of all domains and by $[\mathbf{x}] \sqcap c_j$, the smallest box which contains all points in $[\mathbf{x}]$ that satisfy c_j. The operator $\sqcap$ will be called *square intersection*. The principle generally used to contract the $[x_i]$'s is *arc consistency*. It consists in computing the box

$$((((((([\mathbf{x}] \sqcap c_1) \sqcap c_2) \sqcap \ldots) \sqcap c_m) \sqcap c_1) \sqcap c_2) \ldots,$$

until a steady box (also called the fixed point) is reached.

Consider n variables $x_1, \ldots, x_n$ linked by m primitive constraints $\mathcal{C}_1, \ldots, \mathcal{C}_m$. For each variable x_i, it is assumed that a prior feasible domain $[x_i] = [x_i^-, x_i^+]$ is available (this domain may be equal to $]-\infty, \infty[$ if no information is available on x_i). The principle of interval constraint propagation (ICP) is to perform a projection of all constraints. This operation is repeated until no more significant contraction can be performed.

3.4.2 Example 1

When several constraints are involved, the contractions are performed sequentially, until no more significant contraction can be observed (see [46], for more details). To illustrate the propagation process,

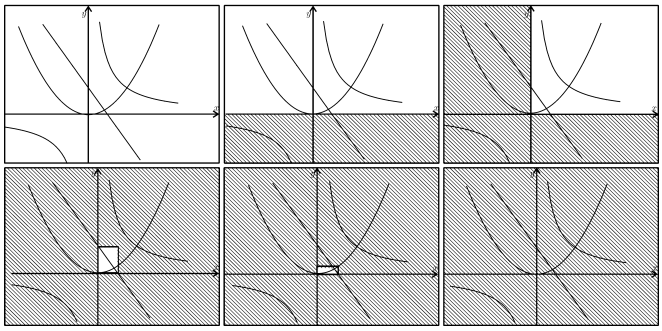


Figure 3.8: Illustration of the propagation procedure

consider the three following equations

$$\begin{cases} (C_1): & y = x^2 \\ (C_2): & xy = 1 \\ (C_3): & y = -2x + 1. \end{cases}$$

Using interval propagation, we want to prove that this system has no solution. To each of the variables, we assign the domain $[-\infty, \infty]$. Then, we contract the domains with respect to the constraints in the following order: C_1, C_2, C_3, C_1, C_2 and we get empty intervals for x and y. A geometric interpretation of the propagation is given on Fig. 3.8. The resulting interval computation is as follows.

$$\begin{array}{lll} (C_1) \Rightarrow & y \in & [-\infty,\infty]^2 \ = \ [0,\infty] \\ (C_2) \Rightarrow & x \in & 1/[0,\infty] \ = [0,\infty] \\ (C_3) \Rightarrow & y \in & [0,\infty] \ \cap \ ((-2) \,.[0,\infty] + 1) \\ & & = [0,\infty] \ \cap \ ([-\infty,1]) = [0,1] \\ & x \in & [0,\infty] \ \cap \ (-[0,1]/2 + 1/2) = [0,\frac{1}{2}] \\ (C_1) \Rightarrow & y \in & [0,1] \cap [0,1/2]^2 = [0,1/4] \\ (C_2) \Rightarrow & x \in & [0,1/2] \ \cap \ 1/[0,1/4] = \emptyset \\ & y \in & [0,1/4] \cap 1/\emptyset = \emptyset. \end{array}$$

3.4.3 Example 2

Consider now the set

$$\begin{array}{rcl} \mathbb{S} & \triangleq & \left\{(x_1,x_2) \in \mathbb{R}^2 \mid x_1 = \sin(x_2) \text{ and } x_1 = x_2^3\right\} \\ & = & \mathbb{S}_1 \cap \mathbb{S}_2 \end{array}$$

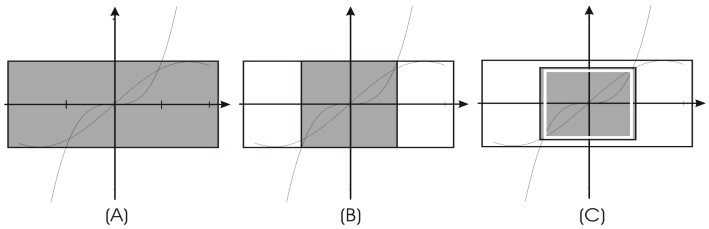
where

$$\begin{array}{rcl} \mathbb{S}_1 & = & \{(x_1,x_2) | x_1 = \sin(x_2)\}, \\ \mathbb{S}_2 & = & \{(x_1,x_2) | x_1 = x_2^3\}. \end{array}$$

For both $\mathbb{S}_1$ and $\mathbb{S}_2$ we assume that minimal contractors $\mathcal{C}^*_{\mathbb{S}_1}$ and $\mathcal{C}^*_{\mathbb{S}_2}$ are available. We have

$$\begin{array}{rcl} [\mathbf{x}] & = &]-\infty,+\infty[\times]-\infty,+\infty[, \\ \mathcal{C}^*_{\mathbb{S}_1}([\mathbf{x}]) & = & [-1,1]\times]-\infty,+\infty[, \\ \mathcal{C}^*_{\mathbb{S}_2}\left(\mathcal{C}^*_{\mathbb{S}_1}([\mathbf{x}])\right) & = & [-1,1]\times[-1,1], \\ \mathcal{C}^*_{\mathbb{S}_2}\left(\mathcal{C}^*_{\mathbb{S}_2}\left(\mathcal{C}^*_{\mathbb{S}_1}([\mathbf{x}])\right)\right) & = & [-0.84,0.85]\times[-1,1], \ldots \end{array}$$

as illustrated by the following figure.



3.5 Local consistency

3.5.1 Problem

If $\mathcal{C}^*_{\mathbb{S}_1}$ and $\mathcal{C}^*_{\mathbb{S}_2}$ are two minimal contractors for $\mathbb{S}_1$ and $\mathbb{S}_2$ then

$$\mathcal{C}_{\mathbb{S}} = \mathcal{C}^*_{\mathbb{S}_1} \circ \mathcal{C}^*_{\mathbb{S}_2} \circ \mathcal{C}^*_{\mathbb{S}_1} \circ \mathcal{C}^*_{\mathbb{S}_2} \circ \dots$$

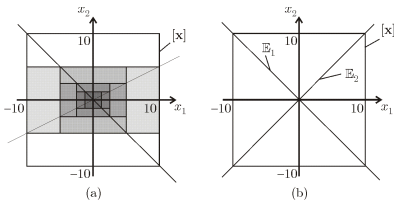
is a contractor for $\mathbb{S} = \mathbb{S}_1 \cap \mathbb{S}_2$, but this contractor may be not optimal. This phenomenon is known as *local consistency effect.*

3.5.2 Example 1

Consider the two following CSPs, both to be contracted with respect to their two equality constraints with $\mathcal{C}_{\downarrow\uparrow}$.

$$\mathcal{H}_1 : \left(\begin{array}{rcl} x_1 + x_2 & = & 0 \\ x_1 - 2x_2 & = & 0 \\ x_1 \in [-10, 10] & & \\ x_2 \in [-10, 10] & & \end{array} \right) \text{ and } \mathcal{H}_2 : \left(\begin{array}{rcl} x_1 + x_2 & = & 0 \\ x_1 - x_2 & = & 0 \\ x_1 \in [-10, 10] & & \\ x_2 \in [-10, 10] & & \end{array} \right).$$

For $\mathcal{H}_2$, the local consistency effect occurs, but not for $\mathcal{H}_1$. Note that when the local consistency effect is present, each set $\mathbb{S}_1$ and $\mathbb{S}_2$ intersects all boundaries of the box $[\mathbf{x}]$.



3.5.3 Example 2

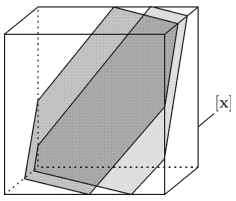
On the following figure, the set $\mathbb{S}_1$ and $\mathbb{S}_2$ are two parallel hyperplanes. We have

$$[\mathbb{S}_1 \cap [\mathbf{x}]] = [\mathbf{x}] \text{ and } [\mathbb{S}_2 \cap [\mathbf{x}]] = [\mathbf{x}]$$

whereas

$$[(\mathbb{S}_1 \cap \mathbb{S}_2) \cap [\mathbf{x}]] = \emptyset.$$

Therefore, we have here a local consistency effect.

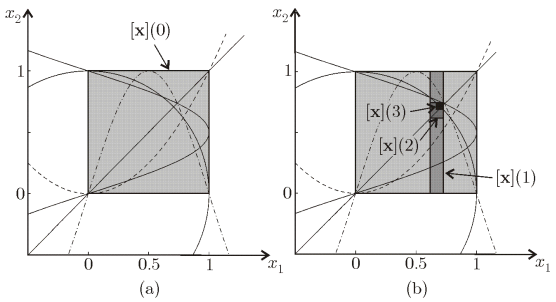


3.5.4 Adding redundant constraints

Consider the CSP

$$\mathcal{H}: \left(\begin{array}{lcl} x_1 - x_2 & = & 0 \\ x_1^2 + x_2^2 - 1 & = & 0 \\ x_2 - \sin(\pi x_1) & = & 0 \\ x_1 - \sin(\pi x_2) & = & 0 \\ x_2 - x_1^2 & = & 0 \\ x_1 \in [0,1], x_2 \in [0,1] & & \end{array} \right).$$

Due to the local consistency effect, constraint propagation is unable to contract $\mathcal{H}$. Now, by summing the first two constraints, one gets the new constraint $x_1^2 + x_2^2 - 1 + x_1 - x_2 = 0$. This new constraint is added to $\mathcal{H}$ to break down the local consistency effect. Constraint propagation can be used to contract optimally $\mathcal{H}$.



3.6 Decomposition into primitive constraints

For more complex constraints, a decomposition is required. For instance, the CSP

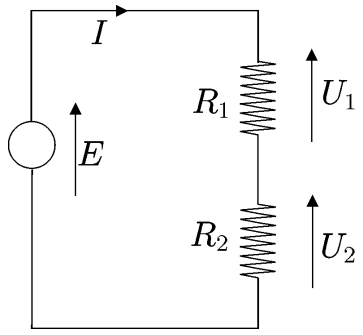
$$\begin{array}{l} x + \sin(xy) \leq 0, \\ x \in [-1,1], y \in [-1,1] \end{array}$$

can be decomposed into the following one.

$$\left\{ \begin{array}{lll} a = xy & x \in [-1,1] & a \in [-\infty,\infty] \\ b = \sin(a) \quad , & y \in [-1,1] & b \in [-\infty,\infty] \\ c = x + b & & c \in [-\infty,0] \end{array} \right.$$

3.7 Application to bounded-error estimation

Consider the electronic circuit consisting of one battery and two resistors represented below



Assume that measures have been collected on this circuit, leading to the following relations :

$$\begin{aligned} E &\in [23V, 26V],\ I \in [4A, 8A],\ U_1 \in [10V, 11V], \\ U_2 &\in [14V, 17V],\ P \in [124W, 130W], \end{aligned}$$

where P is the power delivered by the battery. Nothing is known about the values of the resistors except that they are positive. Thus the prior domains for R_1 and R_2 are $]0, \infty[$. These quantities are related by the following constraints:

$$\begin{aligned} P &= EI;\ E = (R_1 + R_2)\, I; \\ U_1 &= R_1 I;\ U_2 = R_2 I;\ E = U_1 + U_2. \end{aligned}$$

Some of these constraints are redundant, but detecting this is not required by the method. To the contrary, redundancy makes constraint propagation more efficient. Since the second constraint is not primitive, it is decomposed by introducing an auxiliary variable, say R, as follows:

$$E = (R_1 + R_2)\, I \text{ is decomposed into } (E = RI;\ R = R_1 + R_2)\,.$$

Constraint propagation can then be performed by executing several times, say q, the following projections:

$$\begin{aligned} &\text{PMULT}\,(P, E, I)\,;\text{PADD}\,(R, R_1, R_2)\,;\text{PMULT}\,(E, R, U)\,; \\ &\text{PMULT}\,(U_1, R_1, I)\,;\text{PMULT}\,(U_2, R_2, I)\,;\text{PADD}\,(E, U_1, U_2)\,; \end{aligned}$$

This program generates the following results

$$\begin{aligned} R_1 &\in [1.84\Omega, 2.31\Omega]\,,\ R_2 \in [2.58\Omega, 3.35\Omega], \\ I &\in [4.769A, 5.417A]\,,\ U_1 \in [10V; 11V]\,,\ U_2 \in [14V; 16V]\,, \\ E &\in [24V; 26V]\,,\ P \in [124W, 130W]\,. \end{aligned}$$

We got rather accurate intervals containing R_1 and R_2. The domains for I and U_2 have also been contracted, whereas the domains for U_1 and P have been left unchanged.

3.8 Application to robust stability

A CSP is *infallible* if any arbitrary instantiation of all variables in their domains is a solution, *i.e.,* its solution set is equal to the Cartesian production of all its domains. To prove that a CSP is fallible, it suffices to prove that its negation has an empty solution set.

Consider for instance the CSP

$$\begin{aligned} \mathcal{V} &= \{x, y\} \\ \mathcal{D} &= \{[x],[y]\} \\ \mathcal{C} &= \{\ f(x,y) \leq 0,\ g(x,y) \leq 0\}. \end{aligned}$$

The CSP is infallible if

$$\forall x \in [x], \forall y \in [y],\ \ f(x,y) \leq 0 \text{ and } g(x,y) \leq 0,$$

i.e.,

$$\{(x,y) \in [x] \times [y] \mid f(x,y) > 0 \text{ or } g(x,y) > 0\} = \emptyset$$

i.e.,

$$\{(x,y) \in [x] \times [y] \mid \max(f(x,y), g(x,y)) > 0\} = \emptyset.$$

This task can be performed efficiently using interval constraint propagation techniques.

As an illustration, consider a motorbike with a speed of 1m/s. The input of the system is the angle θ of the handlebars and the output is the rolling angle ϕ of the bike. The transfer function is

$$\phi(s) = \frac{1}{s^2 - \alpha_1}\theta(s)$$

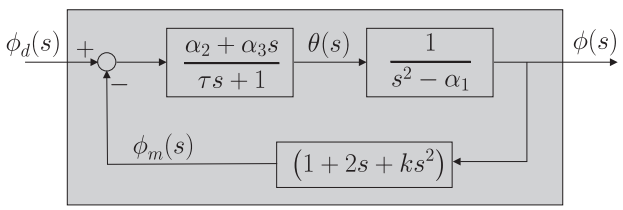
Because of the small velocity of the bike, the system is unstable (the gyroscopic effect of the front wheel is not sufficient to maintain the stability). In order to stabilize the system, we add the following controller

$$\theta(s) = \frac{\alpha_2 + \alpha_3 s}{\tau s + 1}\left(\phi_d(s) - \phi_m(s)\right)$$

where ϕ_d is the wanted rolling angle and ϕ_m is the measured rolling angle. Since the sensor is not perfect ϕ_m is not identical to the rolling angle ϕ of the bike. These quantities are related by the relation :

$$\phi_m(s) = \left(1 + 2s + ks^2\right)\phi(s)$$

The whole system is depicted on the following picture.



The input-output relation of the closed-loop system is :

$$\phi(s) = \frac{\alpha_2 + \alpha_3 s}{(s^2 - \alpha_1)(\tau s + 1) + (\alpha_2 + \alpha_3 s)(1 + 2s + ks^2)}\phi_d(s)$$

Its characteristic polynomial is thus

$$\left(s^2-\alpha_1\right)(\tau s+1)+(\alpha_2+\alpha_3 s)\left(1+2s+ks^2\right)=a_3s^3+a_2s^2+a_1s+a_0,$$

with $a_3=\tau+\alpha_3 k$, $a_2=\alpha_2 k+2\alpha_3+1$, $a_1=\alpha_3-\alpha_1\tau+2\alpha_2$ and $a_0=-\alpha_1+\alpha_2$.

The associated Routh table is :

a_3	a_1
a_2	a_0
$\frac{a_2a_1-a_3a_0}{a_2}$	0
a_0	0

The closed-loop system is thus stable if $a_3, a_2, \frac{a_2a_1-a_3a_0}{a_2}$ and a_0 have the same sign.

Assume that it is known that

$$\begin{array}{rcl}\alpha_1 & \in & [\alpha_1]=[8.8;9.2], \alpha_2\in[\alpha_2]=[2.8;3.2], \alpha_3\in[\alpha_3]=[0.8;1.2],\\ \tau & \in & [\tau]=[1.8;2.2], k\in[k]=[-3.2;-2.8]\end{array}$$

The system is robustly stable if it is stable for all feasible parameters, i.e.,

$$\begin{array}{l}\forall\alpha_1\in[\alpha_1],\forall\alpha_2\in[\alpha_2],\forall\alpha_3\in[\alpha_3],\forall\tau\in[\tau],\forall k\in[k],\\ a_3,\ a_2,\ \frac{a_2a_1-a_3a_0}{a_2} \text{ and } a_0 \text{ have the same sign.}\end{array}$$

Now, we have the equivalence

$$\begin{array}{rl} & b_1,\ b_2,\ b_3 \text{ and } b_4 \text{ have the same sign}\\ \Leftrightarrow & \max\left(\min\left(b_1,b_2,b_3,b_4\right),-\max\left(b_1,b_2,b_3,b_4\right)\right)>0\end{array}$$

The robust stability condition amounts to proving that

$$\begin{array}{l}\exists\alpha_1\in[\alpha_1],\exists\alpha_2\in[\alpha_2],\exists\alpha_3\in[\alpha_3],\exists\tau\in[\tau],\exists k\in[k],\\ \max\left(\min\left(a_3,a_2,\frac{a_2a_1-a_3a_0}{a_2},a_0\right),-\max\left(a_3,a_2,\frac{a_2a_1-a_3a_0}{a_2},a_0\right)\right)\le 0\end{array}$$

is false, i.e., the following CSP

$$\begin{array}{rcl}\mathcal{V} & = & \{a_0,a_1,a_2,a_3,\alpha_1,\alpha_2,\alpha_3,\tau,k\},\\ \mathcal{D} & = & \{[\alpha_0],[\alpha_1],[\alpha_2],[\alpha_3],[\alpha_2],[\alpha_3],[\tau],[k]\},\\ \mathcal{C} & = & \left\{\begin{array}{l}a_3=\tau+\alpha_3k\ ;\ a_2=\alpha_2k+2\alpha_3+1\ ;\ a_1=\alpha_3-\alpha_1\tau+2\alpha_2,\\ a_0=-\alpha_1+\alpha_2\ ;\ b=\frac{a_2a_1-a_3a_0}{a_2};\\ \max\left(\min\left(a_3,a_2,b,a_0\right),-\max\left(a_3,a_2,b,a_0\right)\right)\le 0.\end{array}\right\}\end{array}$$

has no solution.

3.9 Forward-backward propagation

3.9.1 Principle

Forward–backward propagation selects the primitive constraints to be used for contractions in an optimal order in the sense of the size of the domains finally obtained. Consider for instance the equation

$$f(\mathbf{x})\in[y],$$

where

$$f(\mathbf{x}) = x_1 \exp(x_2) + \sin(x_3).$$

The domains for the variables x_1, x_2 and x_3 are denoted by $[x_1], [x_2]$ and $[x_3]$. To obtain an algorithm contracting these domains, first write an algorithm that computes $y = f(\mathbf{x})$, by a finite sequence of elementary operations.

$$\begin{array}{lcl} a_1 & := & \exp(x_2); \\ a_2 & := & x_1 a_1; \\ a_3 & := & \sin(x_3); \\ y & := & a_2 + a_3. \end{array}$$

Then write an interval counterpart to this algorithm:

$$\begin{array}{ll} 1 & [a_1] := \exp([x_2]); \\ 2 & [a_2] := [x_1] * [a_1]; \\ 3 & [a_3] := \sin([x_3]); \\ 4 & [y] := [y] \cap [a_2] + [a_3]. \end{array}$$

If $[y]$ as computed at Step 4 turns out to be empty, then we know that the CSP has no solution. Finally, a backward propagation is performed, updating the domains associated with all the variables to get

$$\begin{array}{ll} 5 & [a_2] := ([y] - [a_3]) \cap [a_2]; \quad \text{// see Step 4} \\ 6 & [a_3] := ([y] - [a_2]) \cap [a_3]; \quad \text{// see Step 4} \\ 7 & [x_3] := \sin^{-1}([a_3]) \cap [x_3]; \quad \text{// see Step 3} \\ 8 & [a_1] := ([a_2]/[x_1]) \cap [a_1]; \quad \text{// see Step 2} \\ 9 & [x_1] := ([a_2]/[a_1]) \cap [x_1]; \quad \text{// see Step 2} \\ 10 & [x_2] := \log([a_1]) \cap [x_2]. \quad \text{// see Step 1} \end{array}$$

At Step 8, $\sin^{-1}([a_3]) \cap [x_3]$ returns the smallest interval containing $\{x_3 \in [x_3] \mid \sin(x_3) \in [a_3]\}$. The associated contractor is given below

Algorithm $\mathcal{C}_{\downarrow\uparrow}$(inout: $[\mathbf{x}]$)	
1	$[a_1] := \exp([x_2])$;
2	$[a_2] := [x_1] * [a_1]$;
3	$[a_3] := \sin([x_3])$;
4	$[y] := [y] \cap ([a_2] + [a_3])$;
5	$[a_2] := ([y] - [a_3]) \cap [a_2]$;
6	$[a_3] := ([y] - [a_2]) \cap [a_3]$;
7	$[x_3] := \sin^{-1}([a_3]) \cap [x_3]$;
8	$[a_1] := ([a_2]/[x_1]) \cap [a_1]$;
9	$[x_1] := ([a_2]/[a_1]) \cap [x_1]$;
10	$[x_2] := \log([a_1]) \cap [x_2]$.

3.9.2 Grouping subexpressions

Grouping subexpressions [1] can improve the computing time for the propagation, but also makes it possible to obtain better contractions. It is important to rewrite our constraints in an optimal way in order to make the propagation more efficient**.** Consider for instance the constraints

$$\begin{array}{lcl} y_1 & = & \cos(i_1 + i_2).\sin(i_1 + i_2), \\ y_2 & = & i_3.\sin^2(i_1 + i_2). \end{array}$$

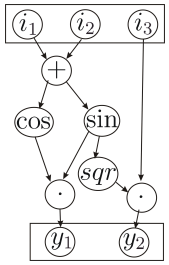
They can be decomposed into primitive constraints as follows

$$\begin{array}{ll} a_1 = i_1 + i_2, & \\ a_2 = \cos(a_1), & a_5 = i_1 + i_2, \\ a_3 = i_1 + i_2, & a_6 = \sin(a_5), \\ a_4 = \sin(a_3), & a_7 = a_6^2, \\ y_1 = a_2.a_4. & y_2 = i_3.a_7. \end{array}$$

A more efficient representation is

$$\begin{array}{ll} a_1 = i_1 + i_2, & \\ a_2 = \cos(a_1), & \\ a_4 = \sin(a_1), & a_7 = a_4^2, \\ y_1 = a_2.a_4. & y_2 = i_3.a_7. \end{array}$$

which is associated to the following DAG



An automatic way to get an optimal decomposition use the notions of DAG (Directed Acyclic Graph) and haching table.

3.10 Existence and uniqueness

Brouwer fixed point theorem: Any continuous function f from the closed unit ball in n-dimensional Euclidean space to itself must have a fixed point (i.e., a point such that $f(x) = x$).

Example. The function $f(x) = \sin x$ is continuous in $[-1, 1]$ and maps it into $[-1, 1]$. Thus $f(x)$ must have a fixed point $[-1, 1]$.

Consider an equation of the form $\mathbf{f}(\mathbf{x}) = \mathbf{0}$. If $\mathbf{M}$ is invertible matrix, we have

$$\mathbf{f}(\mathbf{x}) = \mathbf{0} \Leftrightarrow \underbrace{\mathbf{x} - \mathbf{M}.\mathbf{f}(\mathbf{x})}_{\boldsymbol{\psi}(\mathbf{x})} = \mathbf{x}.$$

As a consequence, if $[\boldsymbol{\psi}]([\mathbf{x}])$ is an inclusion function for $\boldsymbol{\psi}$, we have

$$[\boldsymbol{\psi}]([\mathbf{x}]) \subset [\mathbf{x}] \Rightarrow \exists \mathbf{x} \in [\mathbf{x}], \mathbf{f}(\mathbf{x}) = \mathbf{0}.$$

Banach fixed point theorem. Let $(\mathbb{X}, d)$ be a non-empty complete metric space. Let $f : \mathbb{X} \to \mathbb{X}$ be a contraction mapping on $\mathbb{X}$, i.e.: there is a nonnegative real number $q < 1$ such that

$$\forall (x,y) \in \mathbb{X}^2,\, d(f(x),f(y)) \leq q.d(x,y)$$

Then the map f admits one and only one fixed point $x^* \in \mathbb{X}$. Furthermore, this fixed point can be found as follows: start with an arbitrary element x_0 in $\mathbb{X}$ and define an iterative sequence by $x_n = f(x_{n-1})$. This sequence converges, and its limit is x^*.

Kleene fixed-point theorem. Let $\mathcal{L}$ be a complete partial order, and let $f : \mathcal{L} \to \mathcal{L}$ be a Scott continuous (and therefore monotone) function. Then the least fixed point of f is the supremum of the ascending Kleene chain of f.

$$f \circ f \circ f \circ \cdots \circ f (\bot).$$

3.11 Newton contractor

3.11.1 Newton method

Let us first recall the classical Newton method to solve nonlinear equations $\mathbf{f}(\mathbf{x})$ of n equations with n variables. First, take an approximation $\mathbf{x}_k$ of the solution $\mathbf{x}^*$. Around $\mathbf{x}_k$, we have

$$\mathbf{f}(\mathbf{x}) \simeq \mathbf{f}(\mathbf{x}_k) + \frac{d\mathbf{f}}{d\mathbf{x}}(\mathbf{x}_k).(\mathbf{x}-\mathbf{x}_k).$$

where $\frac{d\mathbf{f}}{d\mathbf{x}}(\mathbf{x}_k)$ denotes the Jacobian matrix of $\mathbf{f}$ at $\mathbf{x}_k$.. In the ideal situation where the approximation is perfect, and since $\mathbf{f}(\mathbf{x}^*) = 0$ we should have

$$\mathbf{f}(\mathbf{x}_k) + \frac{d\mathbf{f}}{d\mathbf{x}}(\mathbf{x}_k).(\mathbf{x}^*-\mathbf{x}_k) = \mathbf{0}$$

or equivalently

$$\mathbf{x}^* = \mathbf{x}_k - \left(\frac{d\mathbf{f}}{d\mathbf{x}}(\mathbf{x}_k)\right)^{-1} \mathbf{f}(\mathbf{x}_k).$$

The Newton methods corresponds to the following sequence

$$\mathbf{x}_{k+1} = \underbrace{\mathbf{x}_k - \left(\frac{d\mathbf{f}}{d\mathbf{x}}(\mathbf{x}_k)\right)^{-1} \mathbf{f}(\mathbf{x}_k)}_{\mathcal{N}(\mathbf{f},\mathbf{J_f},\mathbf{x}_k)}.$$

If the initial condition $\mathbf{x}_k$ is near the solution $\mathbf{x}^*$, the sequence generally quickly converges to $\mathbf{x}^*$.

3.11.2 Interval Newton method

There exists an interval extention of the Newton method. Consider a smooth function $\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^n$ and denote by $\mathbf{J_f}$ its Jacobian matrix. The Newton operator [64] is the operator from $\mathbb{IR}^n$ to $\mathbb{IR}^n$ defined by

$$\mathcal{N}(\mathbf{f},[\mathbf{x}]) = \widehat{\mathbf{x}} - \left[\frac{d\mathbf{f}}{d\mathbf{x}}\right]^{-1} ([\mathbf{x}]).\mathbf{f}(\widehat{\mathbf{x}}),$$

where $\left[\frac{d\mathbf{f}}{d\mathbf{x}}\right]$ is an inclusion function of $\frac{d\mathbf{f}}{d\mathbf{x}}$ and where $\widehat{\mathbf{x}}$ is the center of $[\mathbf{x}]$. Using, the Banach fixed point theorem, Moore [64] has proven that

$$\mathcal{N}(\mathbf{f},[\mathbf{x}]) \subset [\mathbf{x}] \Rightarrow \exists! \mathbf{x} \in [\mathbf{x}], \mathbf{f}(\mathbf{x}) = \mathbf{0},$$

where $\exists!$ means 'there exists a unique'.

3.11.3 Parametric interval Newton method

Consider an equation of the form $\mathbf{f}(\mathbf{x},\mathbf{p})$ where $\mathbf{x}\in\mathbb{R}^n$ and $\mathbf{f}(\mathbf{p},\mathbf{x})\in\mathbb{R}^n$. Define

$$\mathcal{N}(\mathbf{f},[\mathbf{x}],[\mathbf{p}]) = \widehat{\mathbf{x}} - \left[\frac{\partial \mathbf{f}}{\partial \mathbf{x}}\right]^{-1}([\mathbf{x}],[\mathbf{p}]).[\mathbf{f}](\widehat{\mathbf{x}},[\mathbf{p}])$$

where $\left[\frac{\partial \mathbf{f}}{\partial \mathbf{x}}\right]$ and $[\mathbf{f}]$ are inclusion functions for $\frac{\partial \mathbf{f}}{\partial \mathbf{x}}$ and $\mathbf{f}$, respectively. We have

$$\mathcal{N}([\mathbf{f}],[\mathbf{x}],[\mathbf{p}])\subset[\mathbf{x}] \;\Rightarrow\; \forall \mathbf{p}\in[\mathbf{p}],\; \exists!\mathbf{x}\in[\mathbf{x}],\mathbf{f}(\mathbf{x})=\mathbf{0}.$$

3.12 Application to state estimation

As an illustration, consider the non-linear discrete-time system

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

with $k\in\{1,\ldots,15\}$. Interval data have been generated as follows. First, starting from the true value $\mathbf{x}^*(0)=(-1\;\;0)^{\mathrm{T}}$ of the initial state vector, the true values $\mathbf{x}^*(k)$ and $y^*(k), k\in\{1,\ldots,15\}$ were computed by simulation. To each noise-free output $y^*(k)$ a random error was then added, with a uniform distribution in $[-e,e]$, to generate noisy data $\check{y}(k)$. Finally, the prior domain for $y(k)$ was taken equal to $[\check{y}(k)]=[\check{y}(k)-e,\check{y}(k)+e]$. $[\check{y}(k)]$ is thus guaranteed to contain the unknown noise-free output $y^*(k)$. The problem to be solved is then: *given the equations of the system, the interval data* $[\check{y}(k)]$, *and bounded intervals* $[\check{x}_1(0)]$ *and* $[\check{x}_2(0)]$ *containing the initial state variables* $x_1(0)$ *and* $x_2(0)$, *compute (accurate) interval enclosures for the values of the variables* $x_1(k)$, $x_2(k)$ *and* $y(k)$, $k=1,\ldots,15$.

Algorithm ϕ(in: $x_1(0),x_2(0)$; out: $y(1),\ldots,y(15)$)
for $k:=1$ to 15,
$\quad x_1(k):=0.1*x_1(k-1)+x_2(k-1)*\exp(x_1(k-1))$;
$\quad x_2(k):=x_1(k-1)+0.1*x_2^2(k-1)+\sin(k)$;
$\quad y(k):=x_2(k)/x_1(k)$.

This simulator can be decomposed into primitive statements as follows

Algorithm ϕ(in: $x_1(0),x_2(0)$; out: $y(1),\ldots,y(15)$)
for $k:=1$ to 15,
$\quad z_1(k):=\exp(x_1(k-1))$;
$\quad z_2(k)=x_2(k-1)*z_1(k)$;
$\quad x_1(k):=0.1*x_1(k-1)+z_2(k)$;
$\quad z_3(k):=0.1*\mathrm{sqr}(x_2(k-1))$;
$\quad z_4(k):=z_3(k)+\sin(k)$;
$\quad x_2(k):=x_1(k-1)+z_4(k)$;
$\quad y(k):=x_2(k)/x_1(k)$.

The contractor for the set $\hat{\mathbb{X}}(0)$ is in the following table.

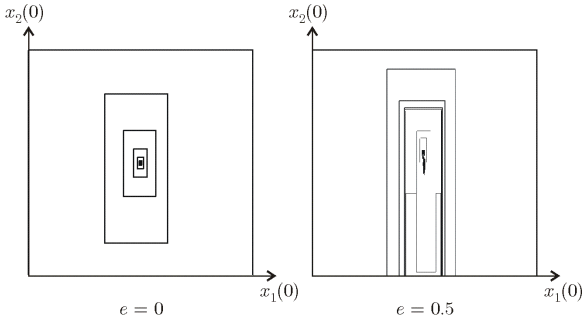
Algorithm $\mathcal{C}_{\hat{\mathbb{X}}(0)}$(in: $[\check{y}(1)], \ldots, [\check{y}(15)]$; inout: $[x_1(0)], [x_2(0)]$)

```
for k := 1 to 15
   [x1(k)] := [-∞, ∞] ; [x2(k)] := [-∞, ∞] ;
   [z1(k)] := [-∞, ∞] ; [z2(k)] := [-∞, ∞] ;
   [z3(k)] := [-∞, ∞] ; [z4(k)] := [-∞, ∞] ;
do
   for k := 1 to 15,                                   // forward
      [z1 (k)] := [z1 (k)] ∩ exp([x1 (k − 1)]);
      [z2 (k)] := [z2 (k)] ∩ ([x2 (k − 1)] ∗ [z1 (k)]) ;
      [x1 (k)] := [x1 (k)] ∩ (0.1 ∗ [x1 (k − 1)]+[z2 (k)]) ;
      [z3 (k)] := [z3 (k)] ∩ (0.1 ∗ sqr([x2 (k − 1)])) ;
      [z4 (k)] := [z4 (k)] ∩ ([z3 (k)] + sin(k)) ;
      [x2 (k)] := [x2 (k)] ∩ ([x1 (k − 1)] + [z4 (k)]) ;
      [y (k)] := [y (k)] ∩ ([x2 (k)]/[x1(k)]) ;
   for k := 15 down to 1,                              // backward
      [x2 (k)] := [x2 (k)] ∩ ([y (k)] ∗ [x1(k)]) ;
      [x1 (k)] := [x1 (k)] ∩ ([x2 (k)]/[y (k)]) ;
      [x1 (k − 1)] := [x1 (k − 1)] ∩ ([x2 (k)] − [z4 (k)]) ;
      [z4 (k)] := [z4 (k)] ∩ ([x2 (k)] − [x1 (k − 1)]) ;
      [z3 (k)] := [z3 (k)] ∩ ([z4 (k)] − sin(k)) ;
      [x2 (k − 1)] := [x2 (k − 1)] ∩ (0.1 ∗ sqr^{-1}([z3 (k)])) ;
      [x1 (k − 1)] := [x1 (k − 1)] ∩ (10 ∗ ([x1 (k)]−[z2 (k)])) ;
      [z2 (k)] := [z2 (k)] ∩ ([x1 (k)] − 0.1 ∗ [x1 (k − 1)]) ;
      [x2 (k − 1)] := [x2 (k − 1)] ∩ ([z2 (k)]/[z1 (k)]) ;
      [z1 (k)] := [z1 (k)] ∩ ([z2 (k)]/[x2 (k − 1)]) ;
      [x1 (k − 1)] := [x1 (k − 1)] ∩ log([z1 (k)]);
while contraction is significant.
```

The prior domains for the components of the initial state vector were taken as

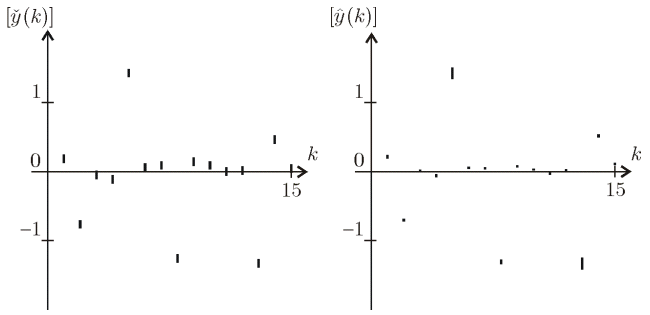
$$[\check{x}_1(0)] = [-1.2, -0.8], \qquad [\check{x}_2(0)] = [-0.2, 0.2].$$

In the absence of noise (*i.e.*, $e = 0$), the contractor is able to find the actual values of all the variables with an accuracy of 8 digits in 0.1 s on a PENTIUM 133. No bisection turned out to be necessary to get this result. The boxes drawn on the left part of the figure are those obtained after each iteration of the contractor $\mathcal{C}_{\downarrow\uparrow}$.

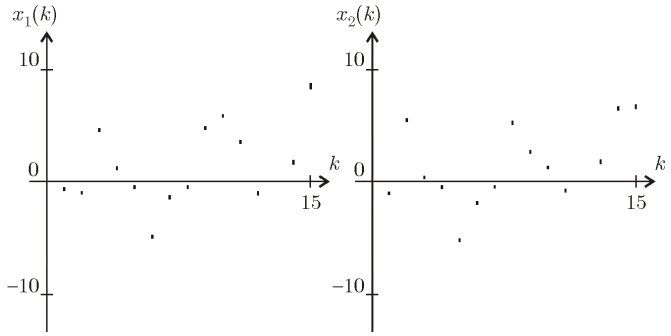


Left: contractions generated in a noise-free context;
right: contractions and bisections generated in a noisy context;
the two frames are $[-1.2, -0.8] \times [-0.2, 0.2]$ in the $(x_1(0), x_2(0))$-space

For $e = 0.5$ (*i.e.*, in the presence of noise), the volume of $\hat{\mathbb{X}}(0)$ is no longer equal to zero, and thus, even with an ideal contractor, bisections have to be performed (see the right part of the figure). The prior interval data $[\check{y}(k)]$ are on the left part of the figure and the corresponding contracted intervals $[\hat{y}(k)]$ are on the right part of the same figure. The following figures presents the initial and contracted domains obtained for the state and output variables.



In the presence of noise ($e = 0.5$), interval output data $[\check{y}(k)]$ (left)
and contracted interval outputs $[\hat{y}(k)]$ containing $y^*(k)$
obtained by taking the constrained set into account (right)



In the presence of noise ($e = 0.5$),
contracted domains for $x_1(k)$ (left) and $x_2(k)$ (right) as functions of k

Part II

Application to robotics

Chapter 4

Image shape extraction

Abstract. This chapter [43] proposes a method for recognition of geometrical shapes (such as lines, circles or ellipsoids) in an image. The main idea is to transform the problem into a bounded error estimation problem and then to use an interval-based method which is robust with respect to outliers. The approach is illustrated on an image taken by an underwater robot where a spheric buoy has to be detected. The results will then be compared to those obtained by the more classical generalized Hough transform.

4.1 Introduction

The problem to be considered in this chapter is to extract known shapes such as ellipses, circles or lines from an image [76] [5]. Fig. 4.1 represents a photo taken by the underwater robot Sauc'isse (Fig. 4.2) that participated in the SAUC'E (Student Autonomous Underwater Competition, European) competition that took place in July 2008 in Brest, France. On Fig. 4.1 we have a spheric buoy that had to be detected by the robot in order to touch it. In this chapter, we search for an automatic method to detect some given geometrical shapes in the image and also return the parameters of the shape. For instance, in Fig. 4.1, the shape to be extracted is a circle and has three parameters: the two coordinates of the center and the radius. The first steps to be performed before the shape extraction are a preprocessing of the image [3] followed by an edge detection [8]. The resulting black-and-white image is represented on Fig. 4.3. The principle of the edge detection is to compute the modulus of the gradient of the image and then to threshold the resulting gradient image. All edge points (in white) of the edge image could potentially belong to the edge of a shape. The approach to be considered here is to find the parameters of a chosen shape that are consistent with a given percentage of the edge points. For our buoy example, the shape will be a circle.

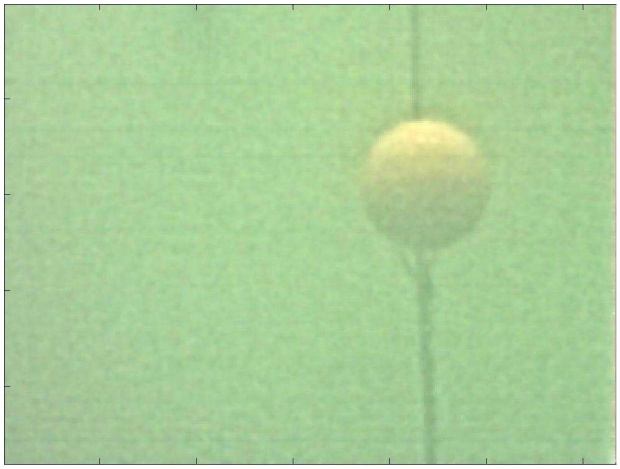


Figure 4.1: A spheric buoy taken by the underwater robot Sauc'isse

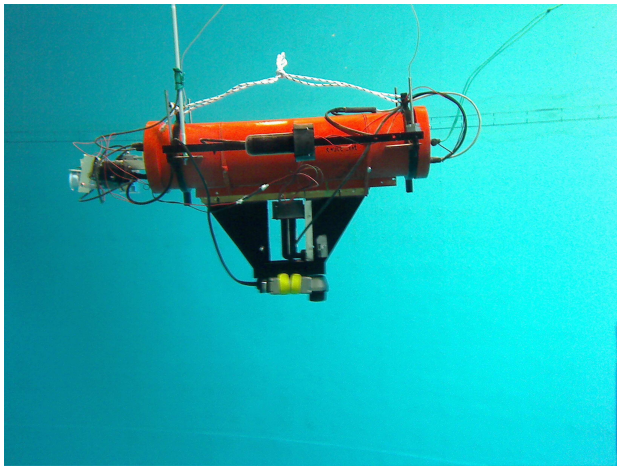


Figure 4.2: Sauc'isse robot inside a swimming pool

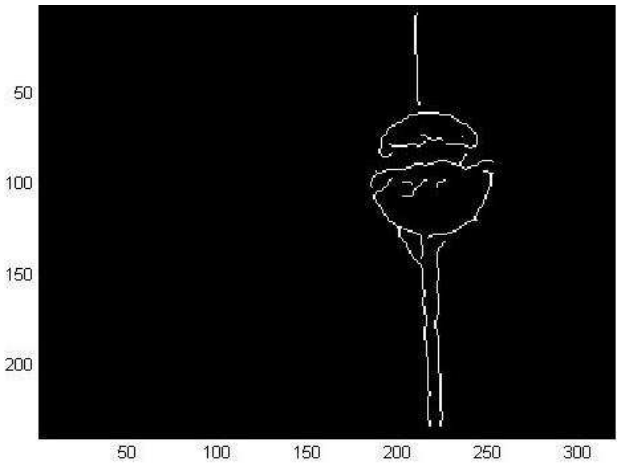


Figure 4.3: Image after an edge detection

4.2 Approach

4.2.1 Set estimation

In a bounded-error estimation context [85], a parameter estimation problem can generally be cast into the problem of characterizing a set defined by

$$\mathbb{P} = \bigcap_{i\in\{1,\ldots,m\}} \underbrace{\{\mathbf{p}\in\mathbb{R}^{n_p}, \exists \mathbf{y}\in[\mathbf{y}](i), \mathbf{f}(\mathbf{p},\mathbf{y})=\mathbf{0}\}}_{\mathbb{P}_i} \tag{4.1}$$

where $\mathbf{p}$ is the parameter vector, $[\mathbf{y}](i)\subset\mathbb{R}^{n_y}$ is the ith measurement box and $\mathbf{f}:\mathbb{R}^{n_p}\times\mathbb{R}^{n_y}\rightarrow\mathbb{R}^{n_f}$ is the model function (latter we will see that n_f can be greater than 1). The set $\mathbb{P}_i$ is the set of all parameter vectors that are consistent with the ith measurement vector. This estimation problem can be qualified as *implicit* because of it involves an implicit equation $\mathbf{f}(\mathbf{p},\mathbf{y})=\mathbf{0}$. In the particular situation where $\mathbf{y}$ can be isolated, *i.e.*, $\mathbf{f}(\mathbf{p},\mathbf{y})=\mathbf{0}\Leftrightarrow\mathbf{g}(\mathbf{p})=\mathbf{y}$, we get

$$\begin{aligned}\mathbb{P} &= \bigcap_{i\in\{1,\ldots,m\}} \underbrace{\{\mathbf{p}\in\mathbb{R}^{n_p}, \mathbf{g}(\mathbf{p})\in[\mathbf{y}](i)\}}_{\mathbb{P}_i}\\ &= \bigcap_{i\in\{1,\ldots,m\}} \mathbf{g}^{-1}([\mathbf{y}](i)),\end{aligned}$$

and the problem becomes a set inversion problem [47].

Example: Consider the example taken from [48] where one wants to find the set of parameter vectors $\mathbf{p}=(p_1,p_2)^{\mathrm{T}}$ such that the graph of the function

$$20\exp(-p_1t)-8\exp(-p_2t),$$

goes through all ten boxes of Fig. 4.4. For this problem, the model function is

$$f(\mathbf{p},\mathbf{y})=20\exp(-p_1y_1)-8\exp(-p_2y_1)-y_2,$$

and the boxes $[\mathbf{y}](1),\ldots,[\mathbf{y}](10)$ are those represented on Fig. 4.4.

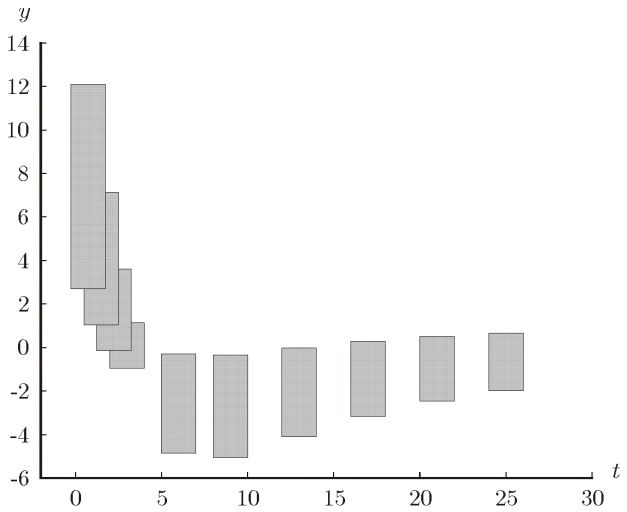


Figure 4.4: Parameter estimation problem

4.2.2 Shape extraction as a set estimation problem

Consider the *shape function* (which is analog to the model function presented above)

$$\mathbf{f}: \left\{ \begin{array}{lll} \mathbb{R}^{n_p} \times \mathbb{R}^2 & \rightarrow & \mathbb{R}^{n_f} \\ (\mathbf{p}, \mathbf{y}) & \rightarrow & \mathbf{f}(\mathbf{p}, \mathbf{y}). \end{array} \right.$$

The $\mathbb{R}^2$ space corresponds to the image, the vector $\mathbf{y}$ to a pixel of this image and $\mathbf{p}$ is the parameter vector of the chosen shape. We define the *shape* associated with $\mathbf{p}$ as follows

$$\mathcal{S}(\mathbf{p}) \stackrel{\text{def}}{=} \left\{ \mathbf{y} \in \mathbb{R}^2, \mathbf{f}(\mathbf{p}, \mathbf{y}) = \mathbf{0} \right\}.$$

Consider a set of (small) boxes in the image

$$\mathcal{Y} = \{[\mathbf{y}](1), \ldots, [\mathbf{y}](m)\}.$$

Each of this box is assumed to intersect the edge of the shape we want to extract.

Remark: In our buoy example, $\mathcal{Y}$ corresponds to small boxes with center the edge points in Fig. 4.3 and with a width of 2 pixels. Since we want to extract a circle, the shape function is

$$f(\mathbf{p}, \mathbf{y}) = (y_1 - p_1)^2 + (y_2 - p_2)^2 - p_3^2.$$

The parameter vector is $\mathbf{p} = (p_1, p_2, p_3)^{\mathrm{T}}$ where p_1, p_2 are the coordinates of the center of the circle and p_3 is its radius.

The feasible set $\mathbb{P}$ is the set of all $\mathbf{p}$ such that the corresponding shape crosses all boxes $[\mathbf{y}](i)$. Now, in our shape extraction problem, a lot of boxes $[\mathbf{y}](i)$ do not correspond to the shape and should then be considered as outliers. As a consequence, except in atypical situations, the resulting feasible set $\mathbb{P}$ is empty. The following section explains how one can robustify the estimation process with respect to some outliers.

4.2.3 Robust set estimation

We define the q relaxed feasible set as

$$\mathbb{P}^{\{q\}} \stackrel{\text{def}}{=} \bigcap_{i \in \{1,\ldots,m\}}^{\{q\}} \left\{ \mathbf{p} \in \mathbb{R}^{n_p}, \exists \mathbf{y} \in [\mathbf{y}](i), \mathbf{f}(\mathbf{p}, \mathbf{y}) = \mathbf{0} \right\}. \tag{4.2}$$

The characterization of $\mathbb{P}^{\{q\}}$ can be done efficiently using interval techniques.

4.3 Interval propagation

With an interval approach, a random variable x of $\mathbb{R}$ is often represented by an interval $[x]$ which encloses the support of its probability function. This representation is of course poorer than that provided by its probability density distribution, but it presents several advantages. *(i)* Since an interval with non zero length is consistent with an infinite number of probability distribution functions, an interval

representation is well adapted to represent random variables with imprecise probability density functions. *(ii)* An arithmetic can be developed for intervals, which makes it possible to deal with uncertainties in a reliable and easy way, even when strong nonlinearities occur. *(iii)* When the random variables are related by constraints (*i.e.*, equations or inequalities) a propagation process (which will be explained later) makes it possible to get efficient polynomial algorithms to compute intervals that are guaranteed to contain all feasible values for the random variables.

4.4 Forward-backward propagation

The interval propagation method converges to a box which contains all solution vectors of our set of constraints. If this box is empty, it means that there is no solution. It can be shown that the box to which the method converges does not depend on the order to which the contractors are applied [46], but the computing time is highly sensitive to this order. There is no optimal order in general, but in practice, one of the most efficient is called *forward-backward propagation*. It consists in writing the equation under the form $\mathbf{f}(\mathbf{p},\mathbf{y}) = \mathbf{0}$. Then, using interval arithmetic, the intervals are propagated from $\mathbf{p},\mathbf{y}$ to $\mathbf{0}$ in a first step (*forward propagation*) and, in a second step, the intervals are propagated from $\mathbf{0}$ to $\mathbf{p},\mathbf{y}$ (*backward propagation*). As an illustration, consider again our shape extraction problem on the buoy image. The following forward-backward contraction algorithm returns the smallest box $[\bar{\mathbf{p}}]$ which encloses the set

$$\left\{\mathbf{p} \in [\mathbf{p}], \exists \mathbf{y} \in [\mathbf{y}], (y_1 - p_1)^2 + (y_2 - p_2)^2 - p_3^2 = 0\right\}.$$

FB(in: $[\mathbf{y}],[\mathbf{p}]$, out: $[\bar{\mathbf{p}}]$)	
1	$[d_1] := [y_1] - [p_1]$;
2	$[d_2] := [y_2] - [p_2]$;
3	$[c_1] := [d_1]^2$;
4	$[c_2] := [d_2]^2$;
5	$[c_3] := [p_3]^2$;
6	$[e] := [0,0] \cap ([c_1] + [c_2] - [c_3])$;
7	$[c_1] := [c_1] \cap ([e] - [c_2] + [c_3])$;
8	$[c_2] := [c_2] \cap ([e] - [c_1] + [c_3])$;
9	$[c_3] := [c_3] \cap ([c_1] + [c_2] - [e])$;
10	$[\bar{p}_3] := [p_3] \cap \sqrt{[c_3]}$;
11	$[d_2] := [d_2] \cap \sqrt{[c_2]}$;
12	$[d_1] := [d_1] \cap \sqrt{[c_1]}$;
13	$[\bar{p}_2] := [p_2] \cap ([y_2] - [d_2])$;
14	$[\bar{p}_1] := [p_1] \cap ([y_1] - [d_1])$;

where $[d_1], [d_2], [c_1], [c_2], [c_3], [e]$ are intermediate interval variables of the algorithm. Steps 1 to 6 form the forward propagation and Steps 7 to 14 form the backward step.

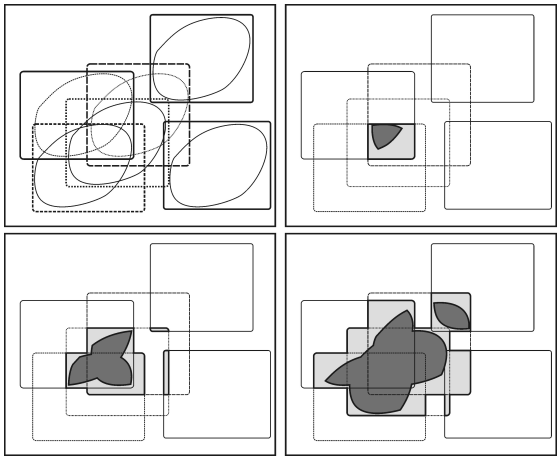


Figure 4.5: Illustration of the inclusion monotonicity of the q-relaxed intersection (in dark grey for the $\mathbb{X}_i$'s and in light grey for the $\mathbb{Y}_i$'s)

4.5 Robust set estimation algorithm

4.5.1 Relaxed intersection

Since the q-relaxed intersection can be written as a combination of unions and intersections, it is inclusion monotonic, *i.e.*,

$$(\mathbb{X}_1 \subset \mathbb{Y}_1, \ldots, \mathbb{X}_m \subset \mathbb{Y}_m) \Rightarrow \bigcap_{i\in\{1,\ldots,m\}}^{\{q\}} \mathbb{X}_i \subset \bigcap_{i\in\{1,\ldots,m\}}^{\{q\}} \mathbb{Y}_i.$$

This inclusion monotonicity is illustrated by Fig. 4.5 in the case where the $\mathbb{Y}_i$'s are boxes. Note that the q-relaxed intersection of m boxes is not necessarily a box, but we can easily compute the smallest box which contains the relaxed intersection.

4.5.2 Algorithm

We shall now present an algorithm to characterize the q-relaxed feasible set $\mathbb{P}^{\{q\}}$ [49], [40]. The principle of the method is illustrated by Fig. 4.6. Fig 4.6,(a) represents the sets $\mathbb{P}_i$ with the solution set $\mathbb{P}^{\{q\}}$ (hatched), representing the q-relaxed intersection we would like to enclose (here, $q = 1$). For each i, we first enclose the sets $[\mathbf{p}] \cap \mathbb{P}_i$ by boxes $[\mathbf{p}](i)$ as represented with dash line boxes on Fig 4.6,(b). On Fig 4.6,(c), the two grey boxes represents the q-relaxed intersection of the boxes $[\mathbf{p}](i)$. We compute a box enclosure (hatched box) of this q-relaxed intersection. On Fig 4.6,(d), we are in the same situation as we were on Fig 4.6,(a). The current box $[\mathbf{p}]$ still encloses $\mathbb{P}^{\{q\}}$ but is now smaller. The process can be iterated once more as illustrated by Fig 4.6,(e) and Fig 4.6,(f). We will then converge to a steady box. The accuracy of the enclosure can be controlled by allowing several bisections of the current box $[\mathbf{p}]$ into subboxes and by iterating the contraction procedure on each subbox. The corresponding algorithm is given by Table 4.1.

Step 1: The list $\mathcal{L}$ contains boxes, the union of which encloses $\mathbb{P}^{\{q\}}$. It is initialized with the single box $[\mathbf{p}]$.

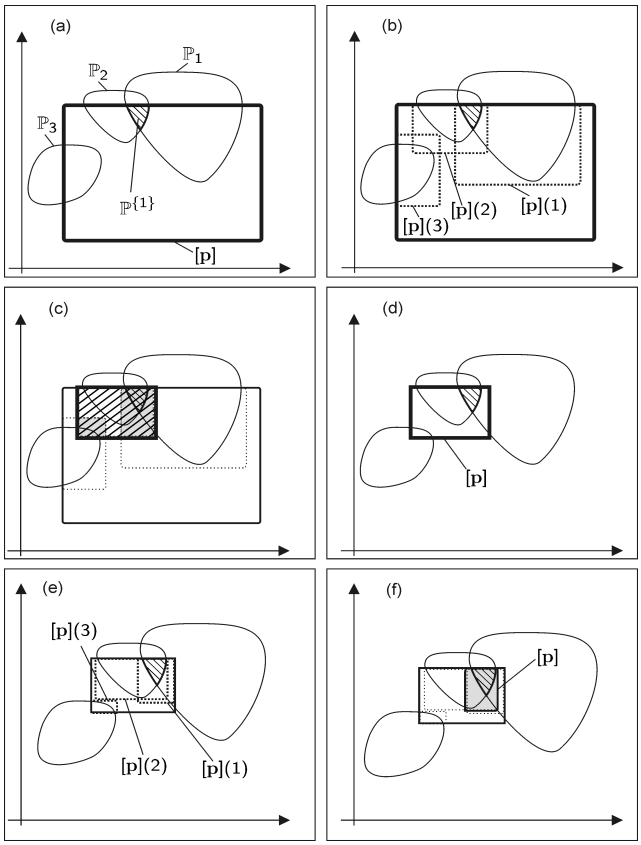


Figure 4.6: Principle of the contraction procedure for the relaxed set inversion problem

Table 4.1: Algorithm for solving the relaxed set inversion problem

Algorithm ENCLOSE(in: $[\mathbf{p}], [\mathbf{y}](1), \ldots, [\mathbf{y}](m), q$, out: $[\bar{\mathbf{p}}]$)

$\mathcal{L} := \{[\mathbf{p}]\}$;
repeat
  pull $([\mathbf{p}], \mathcal{L})$;
  while the contraction are significant
    for $i = 1$ to m, compute $[\mathbf{p}](i)$ enclosing $[\mathbf{p}] \cap \mathbb{P}_i$
    $[\mathbf{p}] := \left[\bigcap_{i \in \{1,\ldots,m\}}^{\{q\}} [\mathbf{p}](i) \right]$
  end repeat
  bisect $[\mathbf{p}]$ and push the resulting boxes into $\mathcal{L}$
until all boxes of $\mathcal{L}$ have a width smaller than ε.
$[\bar{\mathbf{p}}] = \sqcup(\mathcal{L})$

Step 2: A *repeat-until* loop is run until all boxes of $\mathcal{L}$ have a width smaller that a given accuracy ε.

Step 3: The largest box is pulled out from the list.

Step 4: The contraction procedure is iterated until no more significant contraction can be produced.

Step 5: For all i, a box $[\mathbf{p}](i)$ enclosing $[\mathbf{p}] \cap \mathbb{P}_i$ is computed. For the application presented in this chapter, a single forward-backward contraction procedure is implemented.

Step 6: A box enclosing the q-relaxed intersection of the $[\mathbf{p}](i)$'s is computed. Here, $[\mathbb{A}]$ represents a the smallest box enclosing the set $\mathbb{A}$.

Step 8: The current box is bisected into two smaller boxes. These two boxes are pushed at the end of the queue $\mathcal{L}$.

Step 10: The algorithm returns the smallest box $[\bar{\mathbf{p}}]$ enclosing all boxes stored in $\mathcal{L}$ (represented here by the box union operator $\sqcup$).

4.6 Results

Let us apply the interval method to our shape extraction problem for different values of q. We get Fig. 4.7. Subfigures (a),(b),(c) represent circles consistent with $30\%, 20\%, 19\%$ of the data, respectively. The computing time is less than 15 seconds. These results are comparable to those obtain by a classical generalized Hough transform approach devoted to circle extraction [51].

An improvement suggested by O'Gorman and Clowes [69], in the context of the Hough transform [22], is to take into account the fact that local gradient of the image intensity is orthogonal to the edge (See Fig. 4.8).

Let us append the direction y_3 of the gradient (in radian) to the measurement vector $\mathbf{y} = (y_1, y_2)^{\mathrm{T}}$. We thus have a three dimensional measurement vector $\mathbf{y} = (y_1, y_2, y_3)^{\mathrm{T}}$. The gradient condition translates into

$$\det \begin{pmatrix} \frac{\partial f(\mathbf{p},\mathbf{y})}{\partial y_1} & \cos(y_3) \\ \frac{\partial f(\mathbf{p},\mathbf{y})}{\partial y_2} & \sin(y_3) \end{pmatrix} = 0.$$

For the circle detection, since $f(\mathbf{p},\mathbf{y}) = (y_1 - p_1)^2 + (y_2 - p_2)^2 - p_3^2$, this condition becomes

$$(y_1 - p_1)\sin(y_3) - (y_2 - p_2)\cos(y_3) = 0.$$

The model function to be considered for circle detection becomes

$$\mathbf{f}(\mathbf{p},\mathbf{y}) = \begin{pmatrix} (y_1 - p_1)^2 + (y_2 - p_2)^2 - p_3^2 \\ (y_1 - p_1)\sin(y_3) - (y_2 - p_2)\cos(y_3) \end{pmatrix}.$$

This new condition introduces new outliers: the edge points that are on the actual shape, but that do not satisfy the gradient condition. For our problem, we are able to be consistent with 20% of the edge points. But the computing time is now significantly reduced since it is now less that 2 seconds instead of 15 seconds when the gradient condition was not used.

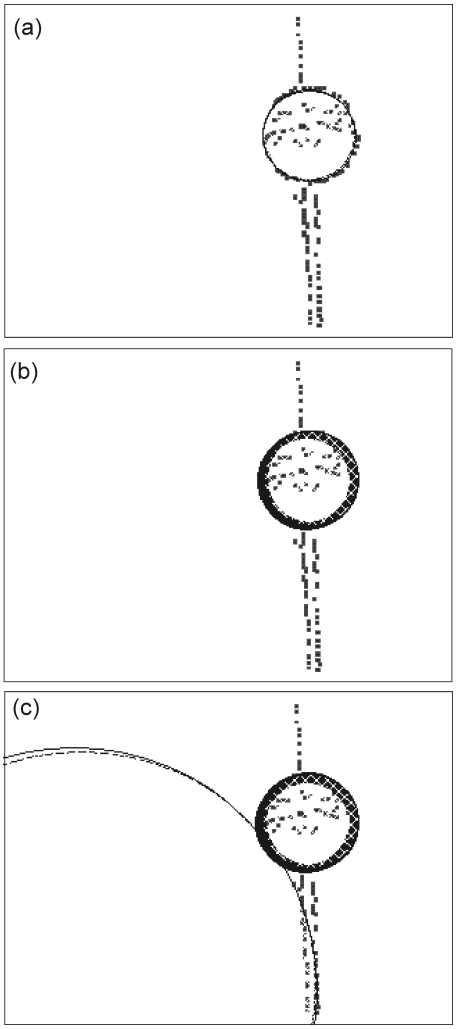


Figure 4.7: Extraction of the circle for different q

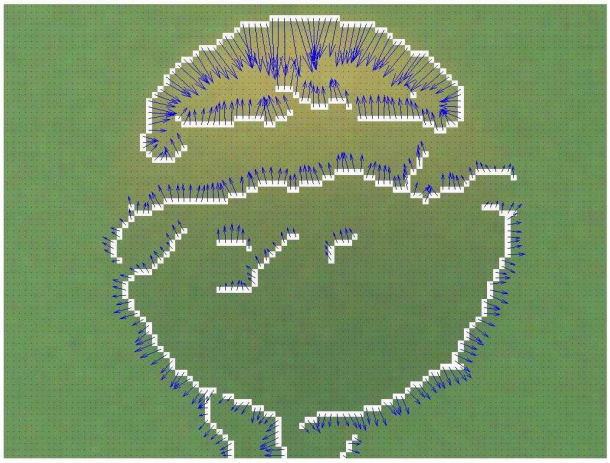


Figure 4.8: Gradient of the buoy image (zoom)

4.7 Conclusion

In this chapter, a method for image shape extraction has been presented. The problem has been transformed into a parameter estimation problem. Since the corresponding estimation problem is nonlinear and is corrupted by many outliers, a classical robust interval method has been used. The Builder 5 C++ code is available at

`www.ensta-bretagne.fr/jaulin/cpp_hough.zip`

A classical approach for extracting geometrical shapes is based on the generalized Hough transform. The method is very similar to that presented in this chapter except that it uses point methods instead of intervals. In short, the principle is to compute the Hough transform defined by

$$\eta(\mathbf{p}) = \text{card}\left\{i \in \{1,\ldots,m\}, \exists \mathbf{y} \in [\mathbf{y}](i), \mathbf{f}(\mathbf{p},\mathbf{y}) = \mathbf{0}\right\},$$

and then to keep all $\mathbf{p}$ such that $\eta(\mathbf{p})$ is higher than a given threshold $m-q$. Instead, our approach directly solves the inequality $\eta(\mathbf{p}) \geq m-q$.

Chapter 5

Robust State Estimation

Abstract. This chapter [40] proposes an observer for estimating the state vector of a nonlinear system. This observer, which is robust with respect to outliers, assumes that the measurement errors as well as the number of outliers that could occur within a given time window are bounded. The principle of the approach is to use interval analysis to deal properly with the nonlinearities involved in the system (without any linearization nor approximation) and to propagate through the time, in a forward and backward manner, the assumptions made about outliers. A testcase related to the localization and control of an underwater robot is also proposed to illustrate the efficiency of the approach.

5.1 Introduction

Consider the nonlinear discrete-time dynamic system

$$\left\{ \begin{array}{lcl} \mathbf{x}_{k+1} & = & \mathbf{f}_k(\mathbf{x}_k) \\ \mathbf{y}_k & = & \mathbf{g}_k(\mathbf{x}_k). \end{array} \right. \tag{5.1}$$

In a bounded-error context, we generally assume that for all k, the output vectors $\mathbf{y}_k$ belong to some known sets $\mathbb{Y}_k$. These sets are obtained from measurements $\tilde{\mathbf{y}}_k$ of the output vector $\mathbf{y}_k$ and take into account some bounded error noises that could corrupt the measurements. In practice, it may happen that some of the $\mathbf{y}_k$, the actual value the output vector at time k, do not belong to their corresponding sets $\mathbb{Y}_k$. Dealing with outliers has already been considered by several authors, in a set membership context (see, *e.g.*, [68], [54], [71]). To robustify bounded error methods against these outliers, we make the following assumption:

MNO (**Minimal Number of Outliers**): Within any time window of length ℓ there are less than q outliers.

Notation: The q-relaxed intersection, denoted by $\bigcap^{\{q\}} \mathbb{X}_i$, of the sets $\mathbb{X}_1, \ldots, \mathbb{X}_m$, , is the set of all $\mathbf{x}$ which belong to all $\mathbb{X}_i$'s, except q at most.

If all $\mathbf{f}_k$ are bijective, the feasible set $\mathbb{X}_k$ for $\mathbf{x}_k$ can recursively be defined by:

$$\mathbb{X}_{k+1} = \mathbf{f}_k\left(\mathbb{X}_k\right) \cap \bigcap_{i\in\{0,\ldots,\ell\}}^{\{q\}} \mathbf{f}_k^i \circ \mathbf{g}_{k-i}^{-1}\left(\mathbb{Y}_{k-i}\right), \tag{5.2}$$

where

$$\mathbf{f}_k^i = \mathbf{f}_k \circ \mathbf{f}_{k-1} \circ \ldots \circ \mathbf{f}_{k-i}. \tag{5.3}$$

If our MNO assumption is true, then for each k, the true value for the state vector belongs to $\mathbb{X}_k$. This chapter proposes an interval constraint propagation approach to recursively compute a box which encloses $\mathbb{X}_k$. Or, equivalently and in a control point of view, we shall build an interval observer robust with respect to outliers. The two integers q and ℓ will be the parameters of this observer. Note that interval constraint propagation methods have been shown successful for several state estimation problems (see, *e.g.*, [45], [72] or [29]). However, these techniques have never been used for state estimation in the context where outliers could occur.

5.2 Relaxed set inversion

The *relaxed set inversion problem* consists in finding a box enclosing the set defined by

$$\mathbb{X} \stackrel{\text{def}}{=} [\mathbf{x}] \cap \bigcap_{i\in\{1,\ldots,\ell\}}^{\{q\}} \mathbf{f}_i^{-1}\left([\mathbf{y}_i]\right). \tag{5.4}$$

This problem is similar to finding a box enclosing $\mathbb{X}_{k+1}$ (see (5.2)). The correspondence between this problem and that of finding a box $[\mathbf{x}_{k+1}]$ enclosing $\mathbb{X}_{k+1}$ is as follows: $\mathbf{f}_i^{-1} \leftrightarrow \mathbf{f}_k^i \circ \mathbf{g}_{k-i}^{-1}$, $[\mathbf{y}_i] \leftrightarrow [\mathbf{y}_{k-i}]$, $\mathbb{X} \leftrightarrow \mathbb{X}_{k+1}$ and $[\mathbf{x}]$ corresponds to a box enclosing $\mathbf{f}_k\left([\mathbf{x}_k]\right)$. The algorithm RSIVIA (for *Relaxed Set Inverter Via Interval Analysis*) solves the relaxed set inversion problem.

Algorithm RSIVIA(in: $[\mathbf{x}], \mathbf{f}_1, \ldots, \mathbf{f}_\ell, q$, out: $[\bar{\mathbf{x}}]$)

```
1   L := {[x]};
2   while allocated time did not elapse and L ≠ ∅
3     pull ([x],L)
4     repeat
5       for i = 1 to ℓ,
            compute [x_i] enclosing [x] ∩ f_i^{-1}([y_i])
6       [x] := [ ⋂_{i∈{1,...,ℓ}}^{{q}} [x_i] ]
7     until no more contraction can be observed
8     if [x] ≠ ∅,bisect [x]
            and push the resulting subboxes into L
9   end while
10  [x̄] = ⊔(L).
```

Step 1: The list $\mathcal{L}$ contains boxes, the union of which encloses $\mathbb{X}$. It is a queue and is initialized with the single box $[\mathbf{x}]$. **Step 2**: The algorithm RSIVIA should take less than the sampling time (between k and $k+1$) for real time applications. This is why we should contract as much as possible the list $\mathcal{L}$

within the allocated time. **Step 3**: The first box (*i.e.*, the one which is waiting to be processed since the longest time) is pulled out from the list. **Step 4**: The contraction procedure is iterated a number of times fixed in advance (for instance 10 times) for real time applications. If no real time implementation is required, we contract until no more significant contractions of $[\mathbf{x}]$ can be observed. **Step 5**: For all i, a box $[\mathbf{x}_i]$ enclosing $[\mathbf{x}] \cap \mathbf{f}_i^{-1}([\mathbf{y}_i])$ is computed. This can be done efficiently using interval analysis [63] combined with constraint propagation methods. For the application presented in this chapter, a single forward-backward contraction procedure (see *e.g.* [46]) is implemented to compute $[\mathbf{x}_i]$. **Step 6**: A box enclosing the q-relaxed intersection of the $[\mathbf{x}_i]$'s is computed. Here $[\mathbb{A}]$ represents a box (as small as possible) enclosing $\mathbb{A}$. Computing the q relaxed intersection of ℓ boxes has a polynomial complexity, if the dimension n of the boxes is fixed (see, *e.g.*, [2]), but the complexity of this problem is exponential with respect to n. **Step 8**: The current box is bisected into two smaller boxes. These two boxes are pushed at the end of $\mathcal{L}$. **Step 10**: The algorithm returns the smallest box $[\bar{\mathbf{x}}]$ enclosing all boxes stored in $\mathcal{L}$ (represented here by the box union operator $\sqcup$).

5.3 Application to the localization and control of an underwater robot

To illustrate the efficiency of the approach, we shall consider the problem of the localization and control of an underwater robot. Note that set-membership methods have often been considered for the localization of robots (see, *e.g.*, [60], [31], in the case where the problem is linear and also [7] when the robot is underwater). In situations where strong nonlinearities are involved, interval analysis has been shown to be particularly useful (see, *e.g.*, [59], where the first localization of an actual robot has been solved with interval methods). Here, the approach is made more efficient by the addition of constraint propagation techniques. Assume the robot is described by

$$\begin{cases} \dot{x} &= v\cos\theta \\ \dot{y} &= v\sin\theta \\ \dot{\theta} &= u_2 - u_1 \\ \dot{v} &= u_1 + u_2 - v, \end{cases} \tag{5.5}$$

where x, y are the coordinates of the robot, θ is its orientation (see Fig. 5.1) and v is its speed. The inputs u_1 and u_2 are the accelerations provided by the left and right propellers. This model corresponds to an underwater robot with a constant depth (the depth regulation of the robot is assumed to be already solved and will not be considered here) and with no roll and pitch. Thus, our robot can be seen as a two-dimensional robot. The localization problem for this type of robot in the presence of outliers is similar to that treated in [59] or [50], but, in these two chapters, the outliers was treated with a static manner, *i.e.*, at each k a lot of measurements were collected (24 sensors were available for the application treated). The robot pose had to be consistent with all measurements made at time k except q of them. Here, the outliers have to be treated in a dynamic manner: the maximum number of allowed outliers is not defined for each k, but for all feasible time windows of a given length.

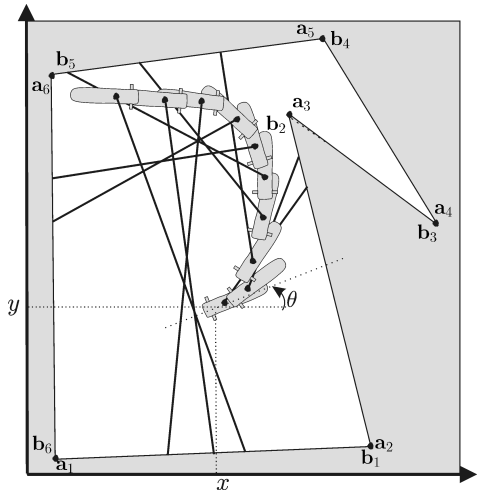


Figure 5.1: When the robot is moving, the sonar turns around the robot

5.3.1 Observer

The system can be discretized by $\mathbf{x}_{k+1} = \mathbf{f}_k(\mathbf{x}_k)$, where $\mathbf{x} = (x, y, \theta, v)$ is the state vector,

$$\mathbf{f}_k \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} x_1 + \delta.x_4.\cos(x_3) \\ x_2 + \delta.x_4.\sin(x_3) \\ x_3 + \delta.u_2(k) - \delta.u_1(k) \\ x_4 + \delta.u_1(k) + \delta.u_2(k) - \delta.x_4 \end{pmatrix} \tag{5.6}$$

and δ is the sampling time. The robot moves inside a swimming pool with a known shape. It is equipped with a sonar which makes it possible to measure the horizontal distance between the robot and the border of the pool following the direction pointed by the sonar. The sonar turns around itself (see Fig. 5.1). Denote by $\alpha(k)$ the angle between the direction of the sonar and the axis of the robot. If the swimming pool is composed with planar vertical walls, the observation equation of the system has the form $d = g_k(\mathbf{x})$. Even if the functions $\mathbf{f}_k$ and g_k are strongly nonlinear, the interval contraction methods required by RSIVIA can be used efficiently to compute a box $[\mathbf{x}_k]$ which encloses the feasible set $\mathbb{X}_k$. The center $\hat{\mathbf{x}}_k$ of this box is returned by the observer as an estimation of the actual state vector for the robot. It is this estimate that will be used by the controller.

5.3.2 Controller

The principle of the controller is described on Fig. 5.2. First, a mission planner sends to the controller a waypoint (x_w, y_w) that has to be reached by the robot. When the current waypoint is considered as reached with a given precision (*i.e.*, $(\hat{x} - x_w)^2 + (\hat{y} - y_w)^2 \leq \varepsilon$), the planner sends the next waypoint. The controller that has been chosen is given by

$$\mathbf{u} = \begin{pmatrix} 1-\omega \\ 1+\omega \end{pmatrix}, \omega = \text{sign}\left(\det \begin{pmatrix} \cos\hat{\theta} & x_w - \hat{x} \\ \sin\hat{\theta} & y_w - \hat{y} \end{pmatrix}\right).$$

The direction to be followed by the robot is given by the vector $\mathbf{e} = (x_w - \hat{x},\ y_w - \hat{y})$. The estimated orientation of the robot is given by the vector $\mathbf{v} = (\cos\hat{\theta},\ \sin\hat{\theta})$. If $\mathbf{v}$ is on the right of $\mathbf{e}$ (i.e., $\det(\mathbf{v}, \mathbf{e}) <$

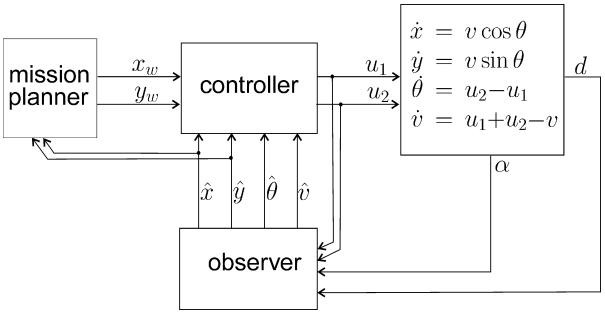


Figure 5.2: Principle of the control of the underwater robot

0), turn right ($\omega = 1$) otherwise, turn left ($\omega = -1$).

5.4 Results

5.4.1 A simulated test-case

To illustrate the behavior of our observer and controller, we consider the problem of localization and control of an underwater robot moving inside a pool with four vertical planar walls and one vertical cylinder (which plays the role of an artificial island inside the pool). Since all walls are vertical, a projection onto the (x, y)-plane is sufficient to characterize them. The coordinates of the corners made by the vertical walls are given by

x	0	13	15	0
y	0	0	10	8

and the circle corresponding to the vertical cylinder has a center at $(10, 5)$ and a radius equal to 2.8m. The mission planner has to send the three following waypoints $(4, 5)$, $(4, 2)$ and $(1.5, 1)$. Once a waypoint is thought to be reached with a precision less than 0.5m, the planner sends the next waypoint, until all waypoints have been sent. The sampling time is chosen as $\delta = 0.0625$s. The length of the sliding time window is chosen as $\ell = 40$, which corresponds to one complete turn of the sonar. The number of allowed outliers inside a time window of length ℓ is chosen as $q = 10$. In our simulation, an outlier is generated with a probability of 0.1. In such a case, the measured distance returned by the simulated robot is fixed at 15m. This choice will facilitate the visualization of the results. Moreover, we added to the measured distance a white noise with a uniform distribution inside the interval $[-0.03, 0.03]$, which correspond to an error of $\pm$3cm.

The results obtained by our observer and controller are illustrated by Fig. 5.3 and Fig. 5.4. The computation time for all the mission takes less than 30sec on classical personal computer, which makes the approach consistent with real time applications. Fig. (5.3) represents the shape of the pool, the trajectory performed by the robot (in black) and the estimated trajectory (small gray squares which represent the centers of the boxes that are proven to enclose the actual position). Point (a) represents the initial position for the robot. Points (b),(c) and (d) are the waypoints that have to be reached by the robot. Note that once point (d) is reached, the mission is finished. A zoom of a part of the trajectory is drawn at point (e). One may see the gray boxes representing the estimated position returned by our

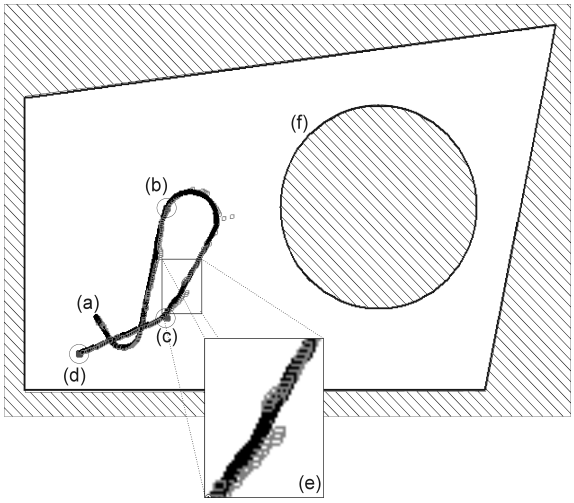


Figure 5.3: Actual (black) and estimated (gray) trajectory of the robot.

observer. At point (f) is represented the vertical cylinder which is inside the pool.

Fig. 5.4 presents the temporal behavior of our interval observer. For each sampling time, the actual state vector $(x(t), y(t), \theta(t), v(t))$ and the measured distance $d(t)$ are depicted. Point (a) represents the initialization step. Its duration corresponds to the length of the sliding window (*i.e.*, $\ell.\delta = 40 * 0.0625 =$ 2.5sec). The vertical bar on (b) indicates the end of the initialization step. A zoom of a sliding window is represented at point (c). The data collected are represented by the black dots and the filtered distance, by the small gray squares. One can see that, inside this window, we have 9 outliers (recall that all of them correspond to a distance of 15m) which is consistent with the assumption that a maximum of 10 outliers could occur inside a window of length ℓ. A typical outlier is represented at point (d). At point (e), we can see a local minimum in the measured distance. This minimum usually corresponds to a situation where the sonar beam is orthogonal to one of the wall, but of course, we don't know which wall it is. At point (f), an outlier has been detected and the estimated distance is far from the measured distance. Point (g) provides some intervals $[x(t)]$ enclosing $x(t)$. When this interval is large, the approximation is less accurate. However, for our testcase, the center is always a good approximation of $x(t)$ and it is the center that is used by the controller. Such enclosing intervals are also provided for $y(t), \theta(t)$ and $v(t)$. All these intervals are proven to enclose the state variables as long as our MNO assumption is satisfied. Note that, for each t, from the current estimated state vector, one is able to reconstruct the past (see Fig. 5.5) on a time window of length ℓ. Since $\ell = 40$, which corresponds to a complete rotation of the sonar, the number of represented sonar beams is also equal to 40. On this figure, the robot looks like a worm. The head (illustrated by the small circle) represents the robot at that time t and the tail represents the robot at time $t - \ell.\delta$.

Remark: The control is computed as if the state vector of the robot were exactly at the center $\hat{\mathbf{x}}_k$ of the box $[\mathbf{x}_k]$. Now, $\hat{\mathbf{x}}_k$ may be infeasible (as it is generally the case when the feasible set corresponds to the union of two disconnected sets) or, even if it is feasible, $\hat{\mathbf{x}}_k$ may correspond to a state which is far from the actual one. In such a case, the robot may be unable to avoid an obstacle. A robust control could be developed in order to provide a control which is safe for all feasible state vectors. In practice, if the swimming pool has no symmetry, $[\mathbf{x}_k]$ is small and its center is near the actual value for the state vector.

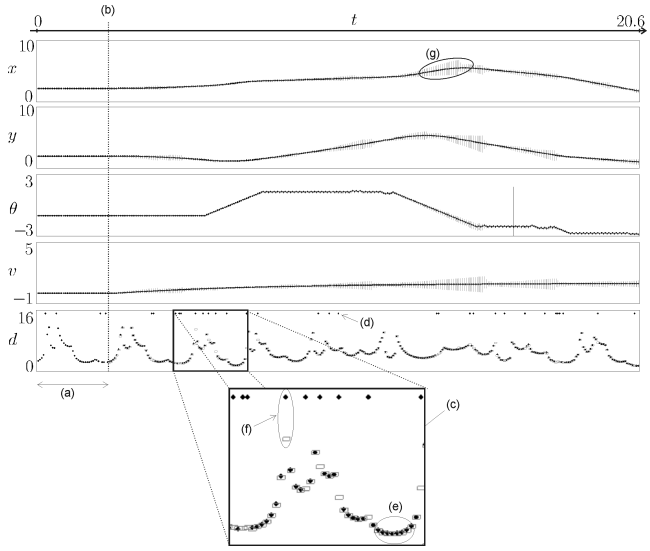


Figure 5.4: The interval observer computes intervals that are guaranteed to enclose all the feasible state variables x, y, θ, v. It also filters the measured distance d.

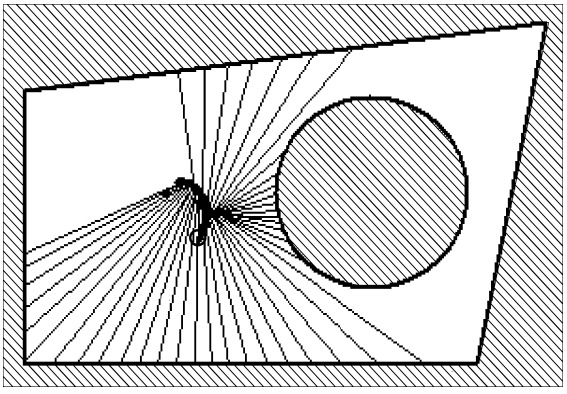


Figure 5.5: Superposition of the poses of the robot with the corresponding sonar beam for $t \in [t - \ell.\delta, t]$

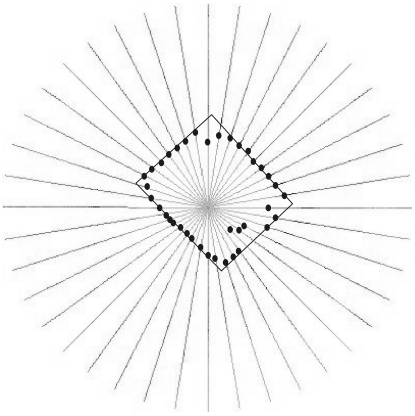


Figure 5.6: A sonar image collected by our TriTech sonar and the corresponding detected distances

To have a safe behavior of the robot, it suffices to check that the obstacles are sufficiently far from the feasible set. If not, we suspect that the robot may knock an obstacle and one should stop the robot.

5.4.2 An actual experiment

The principle of our observer and controller has been implemented on an actual submarine robot on a rectangle swimming pool. Our robot has participated to the SAUC'E (Student Autonomous Underwater Competition, European) competition that took place on July 2007 in Portsmouth, England. Of course, many technical adaptations have to be done in order to take into account the real time requirement, a more complicated state space model, the depth, roll and pitch control, the informations given by the cameras, ... However, the principle of the estimation of (x, y, θ, v) remains the same. The robot, named SAUC'ISSE (SAUCe Interval Super Submarine of ENSTA), has been build by student of the ENSIETA Engineering school, Brest, France. To tune the number q of allowed outliers for each sonar rotation, we took several sonar images (see Fig. 5.6) for different conditions. From the detected distances, we were able count the number of outliers and thus to deduce a realistic upper bound it.

5.5 Conclusion

In this chapter, an observer has been presented with several advantages over classical approaches.

- The observer *is robust with respect to outliers*. By propagating the assumptions on the possible outliers through time, it is robust with respect to a bounded number of outliers, even if we have a small number of outputs in our system. To my knowledge, with existing methods, we were only able to detect outliers in a static way. It should be noted that the principle of propagating outlier assumptions could also be adapted to probabilistic observers (such as Kalman or particle filters [80]) that are not based on some set membership assumptions.
- The observer *is reliable with respect to nonlinearities*. Thanks to interval analysis, it is able to deal with nonlinear (or nondifferentiable and even noncontinuous) state equations, without linearizing

or approximating them.

- The observer *can be used for real time applications.* Constraint propagation techniques, combined with interval analysis, have been used to contract the domains for the variables involved in our problem. These techniques are known to be very efficient even when the number of variables is high. An implementation on an actual robot has also demonstrated the feasibility of our approach when real time is required.

The principle of the approach is based on a methodology that computes recursively the set of all state vectors that are consistent with some bounded errors and a maximal number of outliers within any time window of a given length.

Chapter 6

Interval state observer; application to sailboat robotics

Abstract. This chapter proposes a set-membership observer for estimating the state vector of a nonlinear dynamic system. The method combines flatness concepts with interval set inversion techniques. A testcase related to the state estimation of a sailboat robot is proposed to illustrate the principle and the efficiency of the approach.

6.1 Introduction

This chapter presents an approach for nonlinear state estimation with an application to sailboat robotics. This problem is motivated by the *microtransat challenge* where small autonomous sailboat robots are designed to cross the Atlantic ocean [6]. All components of such robots should be as robust as possible with respect to all situations (heavy weather, waves, salt water, low level of energy, long trip, ...). For sailboat robots, two types of sensors can be considered.

- *Reliable sensors,* which could survive to all situations. Such sensors are the GPS, the compass, the gyrometers and accelerometers. All these sensors are low energy consumers, can be enclosed inside a waterproof tank and can survive for years. The GPS gives us the position of the boat and new generation GPS can also return the speed of the boat with a good accuracy by using the Doppler effect. Since magnetic perturbations inside the ocean can be neglected, the compass measures the north direction with a rather good accuracy. The gyrometer returns the rotational speed and the accelerometers provide the roll and pitch of the robot.

- *Unreliable sensors,* which have a high probability to brake down in case of heavy weather. Anemometers (device that is used for measuring wind speed), weather vane (which returns the direction of the wind), dynamometers which measure the forces on the sail or the rudder are considered as unreliable. They are directly in contact with aggressive natural elements (wind, wave, salt) and can fail down at any time.

On the one hand, to control the robot, it is necessary to know where the wind comes from, what is its power, how strong are the forces on the sail or on the rudder, if the mainsheet is tight or not, ... (see e.g. [78], [34]). On the other hand, a reliable boat should only enclose reliable sensors. The aim of this chapter is twofold.

- The first goal is to show that the variables that could be measured by the unreliable sensors could be reconstructed dynamically from the data collected by the reliable sensors. This is new in a sailboat context.
- The second goal is to give a method which combines nonlinear observation techniques [26], based on flatness concepts, with interval analysis [64]. The first tool makes it possible to transform the observation problem into equations that have to be solved at each time whereas interval analysis provides a systematic way to solve the inversion problem [47] taking into account some interval uncertainties on the measurement data. Note that combining interval analysis with flatness has already been considered for control [30] or source separation [53], but never for state estimation.

6.2 Approach

This section shows how a state estimation problem can be cast into a set of set inversion problems that have to be solved at each instant. Consider the system described by the following state equations

$$\left\{ \begin{array}{rcl} \dot{\mathbf{x}} & = & \mathbf{f}(\mathbf{x},\mathbf{u}) \\ \mathbf{y} & = & \mathbf{g}(\mathbf{x}), \end{array} \right. \tag{6.1}$$

where $\mathbf{u} \in \mathbb{R}^m$ is the control vector, $\mathbf{x} \in \mathbb{R}^n$ is the state vector and $\mathbf{y} \in \mathbb{R}^m$ is the output vector. The dimension of $\mathbf{u}$ and that of $\mathbf{y}$ are assumed to both equal. All vectors depend on the continuous time t. For simplicity, we assume that the evolution function $\mathbf{f}$ and the observation function $\mathbf{g}$ are both smooth with respect to their arguments up to a given order r. The system is said to be *flat* with the flat output $\mathbf{y}$ if there exist two functions $\boldsymbol{\phi}$ and $\boldsymbol{\psi}$ such that for all t, we have

$$\left\{ \begin{array}{rcl} \mathbf{x} & = & \boldsymbol{\phi}\left(\mathbf{y},\dot{\mathbf{y}},\ldots,\mathbf{y}^{(r-1)}\right) \\ \mathbf{u} & = & \boldsymbol{\psi}\left(\mathbf{y},\dot{\mathbf{y}},\ldots,\mathbf{y}^{(r-1)},\mathbf{y}^{(r)}\right). \end{array} \right. \tag{6.2}$$

Now, in practice, $\boldsymbol{\phi}$ and $\boldsymbol{\psi}$ are unknown. To get them, we have to proceed in two steps.

- The *derivation step* (see [35]) computes symbolically $\mathbf{y},\dot{\mathbf{y}},\ldots,\mathbf{y}^{(r)}$ with respect to $\mathbf{x}$ and $\mathbf{u}$, using (6.1), to get an expression of the form

$$\left(\begin{array}{c} \mathbf{y} \\ \dot{\mathbf{y}} \\ \vdots \\ \mathbf{y}^{(r)} \end{array} \right) = \mathbf{h}\left(\begin{array}{c} \mathbf{x} \\ \mathbf{u} \end{array} \right). \tag{6.3}$$

  This can be done automatically without any difficulty using symbolic computation. It suffices to take the equation $\mathbf{y} = \mathbf{g}(\mathbf{x})$ and to compute symbolically its first, second, ... rth derivatives with respect to t. At each step, $\dot{\mathbf{x}}$ should be replaced by $\mathbf{f}(\mathbf{x},\mathbf{u})$.

- The *resolution step* inverses symbolically the function $\mathbf{h}$ to get an expression of the form (6.2). This operation is difficult to obtain except for simple systems.

Example. Consider the system

$$\left\{\begin{array}{lcl} \dot{x}_1 &=& x_1 + x_2 \\ \dot{x}_2 &=& x_2^2 + u \\ y &=& x_1. \end{array}\right.$$

For the derivation step, we compute $y, \dot{y}, \ddot{y}$ with respect to $\mathbf{x}$ and u. We get

$$\left\{\begin{array}{lcl} y &=& x_1 \\ \dot{y} &=& \dot{x}_1 = x_1 + x_2 \\ \ddot{y} &=& \dot{x}_1 + \dot{x}_2 = x_1 + x_2 + x_2^2 + u. \end{array}\right.$$

Thus

$$\mathbf{h}\left(\begin{array}{c} \mathbf{x} \\ u \end{array}\right) = \left(\begin{array}{c} x_1 \\ x_1 + x_2 \\ x_1 + x_2 + x_2^2 + u \end{array}\right).$$

For the resolution step, we have to isolate $\mathbf{x}, u$ to get an expression with respect to $y, \dot{y}, \ddot{y}$. We get

$$\left\{\begin{array}{lcl} x_1 &=& y \\ x_2 &=& \dot{y} - x_1 = \dot{y} - y \\ u &=& \ddot{y} - \left(x_1 + x_2 + x_2^2\right) = \ddot{y} - \dot{y} - (\dot{y} - y)^2. \end{array}\right.$$

As a consequence,

$$\left\{\begin{array}{lcl} \boldsymbol{\phi}(y, \dot{y}) &=& \left(\begin{array}{c} y \\ \dot{y} - y \end{array}\right) \\ \psi(y, \dot{y}, \ddot{y}) &=& \ddot{y} - \dot{y} - (\dot{y} - y)^2. \end{array}\right.$$

■

Equation (6.3) can be rewritten as

$$\mathbf{z} = \mathbf{h}(\mathbf{w}). \tag{6.4}$$

where

$$\mathbf{z} = \left(\begin{array}{c} \mathbf{y} \\ \dot{\mathbf{y}} \\ \vdots \\ \mathbf{y}^{(r)} \end{array}\right) \text{ and } \mathbf{w} = \left(\begin{array}{c} \mathbf{x} \\ \mathbf{u} \end{array}\right), \tag{6.5}$$

Assumption. We assume that for all variables involved in Equation (6.4), membership intervals are available [85]. These intervals can either be punctual if the value of the corresponding variable is known, small if the variable is measured with a good accuracy or equal to $]-\infty, \infty[$ if nothing is known about the variable. For our state estimation problem, we have three types of variables.

- The input variables $u_i, i \in \{1, \ldots, m\}$ can be assumed to be known exactly or with a good precision, i.e., the corresponding interval $[u_i]$ can be assumed to be small or punctual.

- The derivatives $y_i^{(j)}$ of the output variables y_i , $i \in \{1, \ldots, m\}$, $j \in \{0, \ldots, r\}$, are measured with a known error. The intervals $\left[y_i^{(j)}\right]$ containing $y_i^{(j)}$ can be considered as small. For robotic applications, the intervals for derivatives $y_i^{(j)}$ can been often be obtained directly via derivative-based sensors (such as loch-Doppler system, gyrometers or accelerometers). When no such sensor is available and when $\mathbf{y}$ is not too noisy, a robust derivation method (see e.g. [58]) can provide an estimation for the derivatives $y_i^{(j)}$ (but without any estimation of the error). This estimation might help the user to get intervals $\left[y_i^{(j)}\right]$, but without any reliability.
- The state variables x_i, $i \in \{1, \ldots, n\}$ are considered as unknown. The corresponding intervals $[x_i]$ are thus $]-\infty, \infty[$.

Define the boxes

$$[\mathbf{w}] = \underbrace{[x_1] \times \cdots \times [x_n]}_{[\mathbf{x}]} \times \underbrace{[u_1] \times \cdots \times [u_m]}_{[\mathbf{u}]},$$

and

$$[\mathbf{z}] = \underbrace{[y_1] \times \cdots \times [y_m]}_{[\mathbf{y}]} \times \cdots \times \underbrace{\left[y_1^{(r)}\right] \times \cdots \times \left[y_m^{(r)}\right]}_{\left[\mathbf{y}^{(r)}\right]}.$$

The posterior feasible set for $\mathbf{w}$ is

$$\begin{array}{rcl} \mathbb{W} & = & \{\mathbf{w} \in [\mathbf{w}], \exists \mathbf{z} \in [\mathbf{z}], \mathbf{z} = \mathbf{h}(\mathbf{w})\} \\ & = & [\mathbf{w}] \cap \mathbf{h}^{-1}([\mathbf{z}]). \end{array} \tag{6.6}$$

Characterizing the set $\mathbb{W}$ for a given t is thus a set inversion problem [47] which can be solved efficiently using interval analysis. Once $\mathbb{W}$ has been computed, the posterior feasible set $\mathbb{X}$ for $\mathbf{x}$ is easily obtained by a projection of $\mathbb{W}$ onto the $\mathbf{x}$-space.

Remark 1. If the system is observable, if the vectors $\mathbf{u}$ and $\mathbf{y}^{(j)}, j \leq r-1$ are known without any errors, then the set $\mathbb{X}(t)$ is a singleton [19], [28]. Here, since no such assumption is done, the set $\mathbb{X}(t)$ generally encloses an infinite number of elements. However, its size can be small enough to allow us to find a control that fits to all state vectors inside $\mathbb{X}(t)$.

Remark 2. When the system is flat, we may already have an analytical expression for $\mathbf{h}^{-1}$ and thus interval methods are not required anymore for the inversion. However, for our sailing boat, or for many other engineering systems, the inversion cannot be done symbolically and a reliable inversion procedure, such as that provided by interval set inversion, is necessary.

6.3 Set inversion with interval analysis

With an interval approach, a random variable x of $\mathbb{R}$ is represented by an interval $[x]$ which encloses the support of its probability function. This representation is of course poorer than that provided by its probability density distribution, but it presents several advantages. *(i)* Since an interval with non zero length is consistent with an infinite number of probability distribution functions, an interval representation is well adapted to represent random variables with imprecise probability density functions. *(ii)* An arithmetic can be developed for intervals, which makes it possible to deal with uncertainties in a reliable and easy way, even when strong nonlinearities occur. *(iii)* When the random variables are

related by constraints (*i.e.* equations or inequalities) a propagation process (which will be explained later) provides an efficient polynomial algorithm that computes intervals enclosing all feasible values for the random variables.

6.3.1 Interval arithmetic

An interval is a closed and connected subset of $\mathbb{R}$. Consider two intervals $[x]$ and $[y]$ and an operator $\diamond \in \{+, -, ., /\}$, we define $[x] \diamond [y]$ as the smallest interval which contains all feasible values for $x \diamond y$, if $x \in [x]$ and $y \in [y]$ (see [64]). For instance

$$\begin{aligned} [-1,3] + [2,5] &= [1,8], \\ [-1,3].[2,5] &= [-5,15], \\ [-1,3]/[2,5] &= [-\tfrac{1}{2}, \tfrac{3}{2}]. \end{aligned}$$

If f is an elementary function such as $\sin, \cos, \ldots$ we define $f([x])$ as the smallest interval which contains all feasible values for $f(x)$, if $x \in [x]$.

6.3.2 Contractors

Consider a constraint $\mathcal{C}$ (*i.e.*, an equation or an inequality), some variables $x_1, x_2, \ldots$ involved in $\mathcal{C}$ and prior interval domains $[x_i]$ for the x_i's. Interval arithmetic makes it possible to contract the domains $[x_i]$ without removing any feasible values for the x_i's. A contraction operator is called a *contractor*. When several constraints are involved, contractors are called sequentially, until no more significant contraction can be observed (see [46], for more details). The interval propagation method converges to a box which contains all solutions of our set of constraints. If this box is empty, it means that there is no solution. It can be shown that the box toward which the method converges does not depend on the order with which the contractors are applied [46], but the computing time is highly sensitive to this order. There is no optimal order in general, but in practice, one of the most efficient is called *forward-backward propagation*. It consists in writing the equation under the form $\mathbf{y} = \mathbf{h}(\mathbf{x})$. Then, using interval arithmetic, the intervals are propagated from $\mathbf{x}$ to $\mathbf{y}$ in a first step (*forward propagation*) and, in a second step, the intervals are propagated from $\mathbf{y}$ to $\mathbf{x}$ (*backward propagation*).

6.3.3 Algorithm for set inversion

We shall now present an algorithm [47] to characterize the set $\mathbb{W} = [\mathbf{w}] \cap \mathbf{h}^{-1}([\mathbf{z}])$, as required by Equation (6.6). The corresponding algorithm is given by the table below. The inputs of this algorithm, are $[\mathbf{w}]$ which is a (possibly huge) box enclosing all feasible $\mathbf{w} = (\mathbf{x}, \mathbf{u})$ for all t, $[\mathbf{z}]$ is the box defined as the Cartesian product of the intervals $\left[y_i^{(j)}\right]$ enclosing the outputs $y_i^{(j)}$ of our system at time t (see Equation (6.5)). The set $\mathbb{W}^+$ is a subpaving (i.e., a union of boxes) which encloses the feasible set $\mathbb{W}$.

Algorithm SIVIA(in: $[\mathbf{w}], [\mathbf{z}]$, out: $\mathbb{X}^+$)
1 $\mathfrak{L} := \{[\mathbf{w}]\}$
2 repeat
3 pull $([\mathbf{w}], \mathfrak{L})$;
4 while the contractions are significant
5 compute $[\bar{\mathbf{w}}]$ enclosing $[\mathbf{w}] \cap \mathbf{h}^{-1}([\mathbf{z}])$
6 end repeat
7 bisect $[\bar{\mathbf{w}}]$ and push the resulting boxes into $\mathfrak{L}$
8 until all boxes of $\mathfrak{L}$ have a width smaller than ε
9 $\mathbb{W}^+ := \cup\mathfrak{L}$.

The list $\mathfrak{L}$ contains boxes, the union of which encloses $\mathbb{W}$. It is initialized at Step 1 with the single box $[\mathbf{w}]$. At Step 2, a repeat-until loop is run until all boxes of $\mathfrak{L}$ have a width smaller than a given accuracy ε, which is chosen small enough to have a good accuracy on the result and large enough to respect the allowed computing time. At Step 3, the largest box is pulled out from the list. The forward-backward contractor is iterated at Step 4 until no more significant contraction can be observed, i.e., until the Hausdorff distance between the current box and the contracted box is smaller than a given threshold. At Step 7, the current box $[\bar{\mathbf{w}}]$ is bisected into two smaller boxes. These two boxes are pushed at the end of the queue $\mathfrak{L}$. At Step 9, the algorithm returns the subpaving $\mathbb{W}^+$ made by the union of all boxes stored in $\mathfrak{L}$. The properties of SIVIA (time and space complexity, convergence, ...) have been studied in [47]. The complexity has been shown to be exponential with respect to the dimension of $\mathbf{w}$.

6.4 Sailboat

We shall assume that the dynamic of the sailboat represented on Figure 6.1 can be described by the following state equations

$$\left\{\begin{array}{rcl}
\dot{x} &=& v\cos\theta + p_1 a\cos\psi \\
\dot{y} &=& v\sin\theta + p_1 a\sin\psi \\
\dot{\theta} &=& \omega \\
\dot{v} &=& \frac{f_s\sin\delta_s - f_r\sin u_1 - p_2 v}{p_9} \\
\dot{\omega} &=& \frac{f_s(p_6 - p_7\cos\delta_s) - p_8 f_r\cos u_1 - p_3\omega}{p_{10}} \\
\dot{a} &=& 0 \\
\dot{\psi} &=& 0 \\
f_s &=& p_4 a\sin(\theta - \psi + \delta_s) \\
f_r &=& p_5 v\sin u_1 \\
\gamma &=& \cos(\theta-\psi) + \cos(u_2) \\
\delta_s &=& \left\{\begin{array}{ll} \pi - \theta + \psi & \text{if } \gamma \le 0 \\ sign(\sin(\theta-\psi)).u_2 & \text{otherwise.} \end{array}\right.
\end{array}\right. \tag{6.7}$$

where p_1 is the drift coefficient, p_2 is the tangential friction, p_3 is the angular friction, p_4 is the sail lift, p_5 is the rudder lift, p_9 is the mass of the boat and p_{10} is its angular inertia. The distances p_6, p_7, p_8 are represented on Figure 6.1. All parameters p_i are assumed to be known exactly. The sailboat has two inputs: $u_1 = \delta_r$ is the angle between the rudder and the sailboat and $u_2 = \bar{\delta}_s$ is the maximum angle of

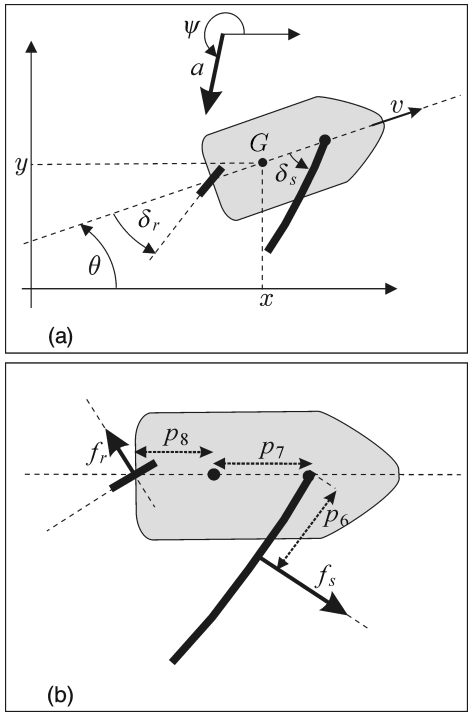


Figure 6.1: Sailboat to be observed

the sail (which is limited by the length of the mainsheet). This model is similar to that described in [37], except that here, *(i)* we added the direction of the wind ψ and its amplitude a as state variables and *(ii)* the control is not anymore the sail angle, but the length of the mainsheet, which is more realistic. In the context of this chapter, the main advantages of this model is its strong nonlinearities, the hybrid behavior due to the fact that the mainsheet may be tight or not. This makes the estimation problem very difficult to solve using existing deterministic approaches. Note that the model that is proposed could be made more realistic by adding ship modelling techniques (see e.g. [4] and [33]).Here, we assume that $x, y, \theta, \dot{x}, \dot{y}, \dot{\theta}, \ddot{x}, \ddot{y}, \ddot{\theta}$ are known with a given error. This assumption is rather realistic if our robot is equipped with a Doppler GPS and accelerometers. Otherwise, robust differentiation methods should be considered [58] to get $\dot{x}, \dot{y}, \ddot{x}, \ddot{y}, \dot{\theta}, \ddot{\theta}$. We also assume that the wind is unknown but constant (at least locally), i.e., $\dot{a} = \dot{\psi} = 0$. From the state equations of the sailboat, it is easy to check that

$$\underbrace{\begin{pmatrix} x \\ y \\ \theta \\ \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \ddot{x} \\ \ddot{y} \\ \ddot{\theta} \end{pmatrix}}_{\mathbf{z}} = \mathbf{h} \underbrace{\begin{pmatrix} x \\ y \\ \theta \\ v \\ \omega \\ a \\ \psi \\ u_1 \\ u_2 \end{pmatrix}}_{\mathbf{w}}$$

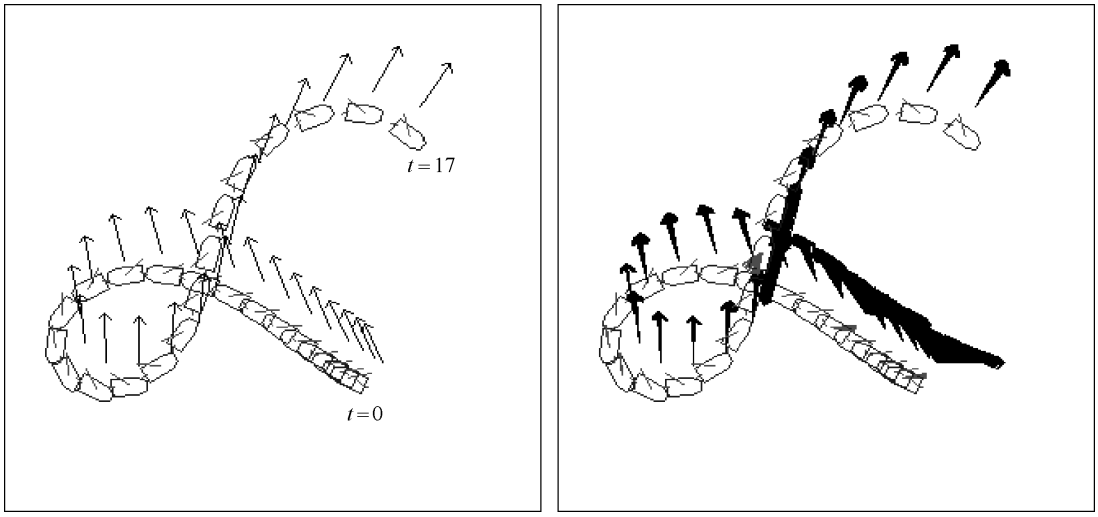


Figure 6.2: Left: Simulated experiment. Right: Estimation of the wind

where $\mathbf{h}$ is given by the following expression

$$\mathbf{h}(\mathbf{w}) = \begin{pmatrix} x \\ y \\ \theta \\ v\cos\theta + p_1 a\cos\psi \\ v\sin\theta + p_1 a\sin\psi \\ \omega \\ \frac{(f_s.\sin\delta_s - f_r\sin u_1 - p_2 v)\cos\theta}{p_9} - \omega v\sin\theta \\ \frac{(f_s\sin\delta_s - f_r\sin u_1 - p_2 v)\sin\theta}{p_9} + \omega v\cos\theta \\ \frac{f_s(p_6 - p_7\cos\delta_s) - p_8 f_r\cos u_1 - p_3\omega}{p_{10}} \end{pmatrix}$$

and $f_s(\mathbf{w}), f_r(\mathbf{w}), \delta_s(\mathbf{w}), \gamma(\mathbf{w})$ are given by (6.7).

6.5 Testcase

We consider the simulated experiment represented on Figure 6.2 (left) which last $t_{\max} = 17s$. The arrows represent the unknown wind vector, which is time dependent (for the simulation, we took $a = 10 + 2\sin(0.1\ t)$ and $\psi = 1 + \cos(0.1\ t)$). The parameters have been chosen as $p_1 = 0.1, p_2 = 100$ Kg.s^{-1}, $p_3 = 500$ N.m.s, $p_4 = 500$ Kg.s^{-1}, $p_5 = 70$ Kg.s^{-1}, $p_6 = 1.1$m, $p_7 = 1.4$ m, $p_8 = 2$m, $p_9 = 1000$Kg and $p_{10} = 2000$ N.m.s^2. For all $x, y, \theta, \dot{x}, \dot{y}, \dot{\theta}, \ddot{x}, \ddot{y}, \ddot{\theta}$ a small uniform noise inside the interval $[-2.10^{-3}, 2.10^{-3}]$ has been added.

The results obtained by our observer are represented on Figure 6.3. The intervals for the variables are $[t] \in [0, 17]$, $[a] \in [0, 15]$, $[\gamma] \in [-2, 2]$, $[\psi] \in [0, 2\pi]$, $[\delta_s] \in [-2, 2]$, $[f_r] \in [-1000, 1000]$ and $[f_s] \in [-10000, 10000]$. At time $t_0 = 0$, the speed of the boat is small and the observer does not provide a good precision. At time t_1, and t_4 two sail angles with different signs fit the data which yield some estimation problems. Inside the interval $[t_2, t_3]$, we have $\gamma \leq 0$, the boat is thus head to wind and the mainsheet

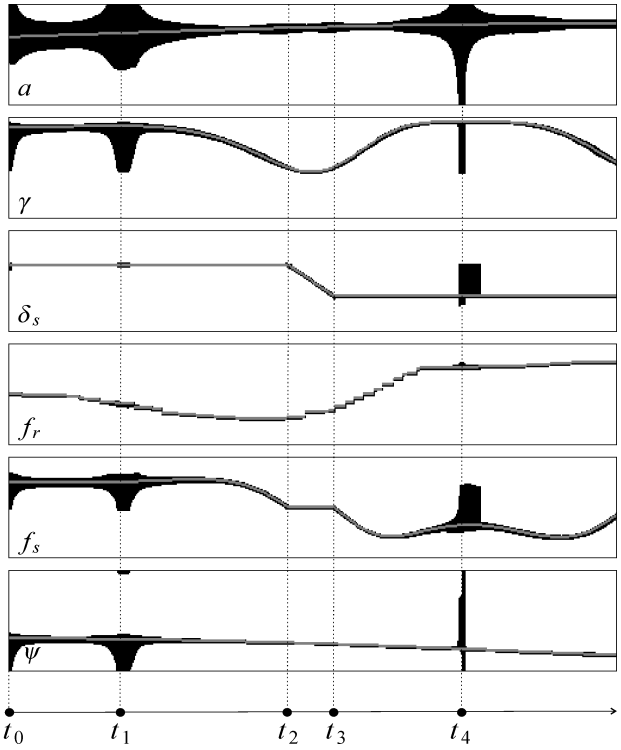


Figure 6.3: Results obtained by the interval observer

is not tight. We checked that the interval envelopes always contain the true signals. The C++ code of the simulation as well as movies illustrating our simulated experiment with the interval observer can be downloaded at

`http://www.ensta-bretagne.fr/jaulin/getwind.html`

6.6 Conclusion

This chapter [39] has presented an approach for nonlinear state estimation. This approach combines some nonlinear state estimation techniques [26] based on flatness [25] with interval set inversion. Flatness makes possible to transform the state estimation problem into set inversion problems parametrized by time t. Interval analysis solves rigorously and efficiently the resulting set estimation problem for each t. The resulting observer has several advantages over classical approaches.

- The observer *is reliable with respect to nonlinearities.* Thanks to interval analysis, it is able to deal with nonlinear (or nondifferentiable and even noncontinuous) state equations, without linearizing (as done by the extended Kalman filter [80]) or approximating them.
- The observer *does not require the interval integration of differential equation.* Such integrations are needed by all other interval state estimation methods [36], [45], [72], [29] which makes them inefficient for high dimensional systems.

- The observer *takes into account bounded noises* on the outputs and their derivatives. To my knowledge, it is not done by existing algebraic nonlinear observers.
- The observer *can be used for real time applications*. For each t, interval set inversion has solved the state estimation of our sailboat problem within a time smaller than 0.1 sec.

The approach has been illustrated on the state estimation of a sailboat. The sailboat estimation problem has several advantages : *(i)* it is motivated by the fact that we want to build a reliable boat without unreliable sensors, *(ii)* it is simple enough to illustrate the principle and the generality of presented approach and *(iii)* it is difficult enough to make all existing other deterministic nonlinear approaches for state estimation fail.

Chapter 7

Simultaneous localization and map building

Abstract. This chapter [39] proposes a set-membership method based on interval analysis to solve the simultaneous localization and map building (SLAM) problem. The principle of the approach is to cast the SLAM problem into a constraint satisfaction problem for which interval propagation algorithms are particularly powerful. The resulting propagation method is illustrated on the localization and map building of an actual underwater robot.

7.1 Introduction

This chapter proposes a set-membership approach to deal with *simultaneous localization and map building* (SLAM) in a submarine context. The SLAM problem [56] for an autonomous robot moving in an unknown environment is to build a map of this environment while simultaneously using this map to compute its location. Most of approaches for the SLAM cast the problem into a state estimation problem, by including the landmark locations among the state variables [10], [20], [62]. See also [86], [75], [24] for the case of autonomous underwater robots. Most of the proposed solutions are based on probabilistic estimation techniques (Kalman filtering, Bayesian estimation, particle filters) [80], [81] which aim at blending data with some state equations of the robot.

In this chapter, a set-membership approach for SLAM (see, *e.g.*, [17], [18]) is considered and it is shown that this approach leads us to a huge set of nonlinear equations which can be solved efficiently using interval analysis and constraint propagation (see, [64], [66] for classical interval analysis, [32] for interval optimization methods and [46] for interval constraint propagation and applications). The approach will be illustrated on an experiment where an actual underwater vehicle is involved. In this problem, we will try to find an envelope for the trajectory of the robot and to compute sets which contain some detected seamarks. Note that for our experiment, the detections of the seamarks are performed by a human operator, after the mission of the robot. Thus, our problem can be considered as an off-line smoothing SLAM problem, *i.e.*, the reconstruction of the robot trajectory and the map building are performed once the measurements have been collected.

Set-membership methods have often been considered for the localization of robots (see, *e.g.*, [60], [31], if the problem is linear and also [7] when the robot is underwater). In situations where strong nonlinearities are involved, interval analysis has been shown to be particularly useful (see, *e.g.*, [59], where the first localization of an actual robot is solved with interval methods). Classical interval analysis has been shown to be efficient in several SLAM applications (see [21] and [70] where it is applied to SLAM of wheeled robots). Now, the resulting techniques perform bisections and thus the complexity of the resulting algorithms is exponential with respect to the number of variables that are bisected. In this chapter the approach is made more efficient by the use of interval propagation. With this approach, no bisection is performed and the resulting complexity becomes polynomial (almost linear in practice for the SLAM problem) with respect to the number of variables. Note that in our case, the number of variables grows with time, since they contain the robot pose at each time instant, as well as the mark position. Although never used in this context, there exist many other robotics applications where interval propagation has been successful (see, *e.g.*, [72] for state estimation, [29] for dynamic localization of robots, [57], [84] for control of robots, [15] for topology analysis of configuration spaces, ...).

7.2 Set-membership approach for SLAM

7.2.1 Principle

We shall describe our SLAM problem as follows

$$\left\{\begin{array}{lcll} \dot{\mathbf{x}} & = & \mathbf{f}(\mathbf{x},\mathbf{u}) & \text{(evolution equation)} \\ \mathbf{y} & = & \mathbf{g}(\mathbf{x},\mathbf{u}) & \text{(observation equation)} \\ \mathbf{z}_i & = & \mathbf{h}(\mathbf{x},\mathbf{u},\mathbf{m}_i) & \text{(mark equation)} \end{array}\right. \tag{7.1}$$

where $\mathbf{x}(t)$ is the state vector of the robot, $\mathbf{u}(t)$ is its input vector, $\mathbf{y}(t)$ is its output vector, $\mathbf{m}_i$ is the location of the ith mark, $\mathbf{z}_i(t)$ is the measurement vector related to the ith mark and $t \in [t_0, t_f]$ is the time. In a set-membership context, we shall assume the following.

- For all $t \in [t_0, t_f]$, we have boxes $[\mathbf{u}](t)$ and $[\mathbf{y}](t)$ enclosing the vectors $\mathbf{u}(t)$ and $\mathbf{y}(t)$. If some components of $\mathbf{y}$ or $\mathbf{u}$ are not measured, then the corresponding interval will be $]-\infty, \infty[$.
- We have a finite subset $\mathcal{M} \subset [t_0, t_f] \times \{1, \ldots, i_{\max}\}$ and bounded boxes $[\mathbf{z}_i](t)$, where $(t,i) \in \mathcal{M}$, such that $\mathbf{z}_i(t) \in [\mathbf{z}_i](t)$. The number of elements of the set $\mathcal{M}$ corresponds to the number of times a mark has been detected. If $(t,i) \in \mathcal{M}$, then, the ith mark has been detected at time t.

In a set-membership context, the objective of SLAM is to contract as much as possible all set-membership domains for $\mathbf{x}(t), \mathbf{m}_1, \ldots, \mathbf{m}_{i_{\max}}$ without removing any feasible value.

7.2.2 A simple example

In this section, an academic problem involving a two dimensional underwater robot is presented. This problem has been chosen for the following reasons *(i)* it illustrates the main idea of SLAM *(ii)* it will be used to show how interval propagation works and *(iii)* it is not easily solved using usual SLAM methods such as the extended Kalman filter.

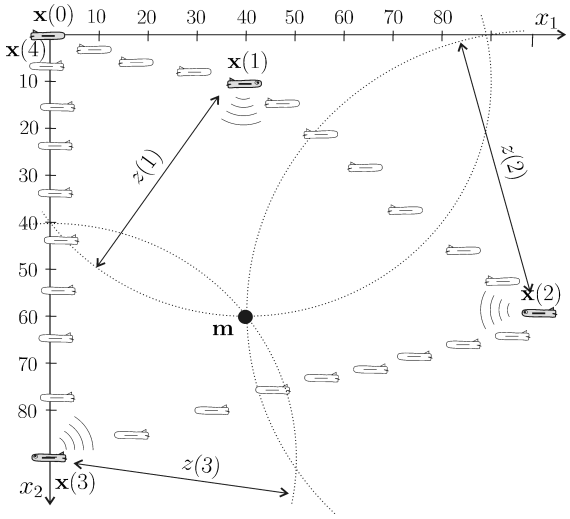


Figure 7.1: The underwater robot with two state variables is able to measure the distance between a seamark and itself

In the environment of the robot, we have one immobile seamark $\mathbf{m}$ (see Figure 7.1). The SLAM problem is described by the following equations

$$\begin{cases} \dot{\mathbf{x}} = \mathbf{u} & \text{(state equation of the robot)} \\ \mathbf{y} = \mathbf{x} & \text{(GPS, when available)} \\ z = ||\mathbf{x} - \mathbf{m}|| & \text{(mark equation)} \end{cases} \tag{7.2}$$

At time $t = 0$, the robot is at the origin and thus $\mathbf{x}(0) = (0, 0)$. At time $t = 4$, the robot is at the surface and is able to measure its location (with the GPS). At time $t \in \{1, 2, 3\}$, the robot measures its distance z to the mark $\mathbf{m}$. During its mission, at each time $t \in [0, 4]$, it also measures the input $\mathbf{u}$ with a known accuracy.

Remark 1 *For our simulation, the seamark is located at* $\mathbf{m} = (40, 60)$ *and the chosen input* $\mathbf{u}(t)$ *is given by:*

t	$[0,1[$	$[1,2[$	$[2,3[$	$[3,4[$
$\mathbf{u}$	$\begin{pmatrix} 40 \\ 10 \end{pmatrix}$	$\begin{pmatrix} 60 \\ 50 \end{pmatrix}$	$\begin{pmatrix} -100 \\ 30 \end{pmatrix}$	$\begin{pmatrix} 0 \\ -90 \end{pmatrix}$

(7.3)

The exact position of the robot as well as the exact distance to the seamark can thus be computed for all t*. As a consequence, we get the following values for* $\mathbf{x}$ *and* z*:*

t	1	2	3	4
$\mathbf{x}$	$\begin{pmatrix} 40 \\ 10 \end{pmatrix}$	$\begin{pmatrix} 100 \\ 60 \end{pmatrix}$	$\begin{pmatrix} 0 \\ 90 \end{pmatrix}$	$\begin{pmatrix} 0 \\ 0 \end{pmatrix}$
z	50	60	50	-

(7.4)

The exact values for $\mathbf{u}(t)$ *and for* $\mathbf{m}$ *are given here for a better understanding of the example and to allow the reader to check the consistency of the results. But in what follows, the location for* $\mathbf{m}$ *is unknown and* $\mathbf{u}(t)$ *is measured with a known accuracy.*

The quantities that can actually be measured are $\mathbf{u}(t)$, for each $t \in [0,4]$, $z(t)$ for $t \in \{1,2,3\}$ and $\mathbf{x}(4)$. In a set-membership context, these measurements translate into the intervals $[\mathbf{u}](t)$, $[z](t)$, $[\mathbf{x}](4)$ which enclose the true values for $\mathbf{u}(t)$, $z(t)$ and $\mathbf{x}(4)$. Define the quantity $\mathbf{v}(t) = \int_t^{t+1} \mathbf{u}(\tau)d\tau, t \in \{0,1,2,3\}$. From the intervals $[\mathbf{u}](t)$ we are able to compute intervals $[\mathbf{v}](t)$ enclosing $\mathbf{v}(t), t \in \{0,1,2,3\}$. Our SLAM problem can then be cast into the following set of equations

$$\begin{cases} \mathbf{x}(1) = \mathbf{x}(0) + \mathbf{v}(0), \ldots, \mathbf{x}(4) = \mathbf{x}(3) + \mathbf{v}(3) \\ z(1) = ||\mathbf{x}(1) - \mathbf{m}||, \ldots, z(3) = ||\mathbf{x}(3) - \mathbf{m}|| \end{cases} \tag{7.5}$$

where some prior membership intervals (or *domains*) are known for $\mathbf{v}(0), \mathbf{v}(1), \mathbf{v}(2), \mathbf{v}(3), \mathbf{x}(4), z(1), z(2), z(3)$. These domains are given by

variable	domain
$z(1)$	$[50 - \delta_z, 50 + 2\delta_z]$
$z(2)$	$[60 - \delta_z, 60 + 2\delta_z]$
$z(3)$	$[50 - \delta_z, 50 + 2\delta_z]$
$\mathbf{v}(0)$	$[40 - \delta_v, 40 + 2\delta_v]$ $[10 - \delta_v, 10 + 2\delta_v]$
$\mathbf{v}(1)$	$[60 - \delta_v, 60 + 2\delta_v]$ $[50 - \delta_v, 50 + 2\delta_v]$
$\mathbf{v}(2)$	$[-100 - \delta_v, -100 + 2\delta_v]$ $[30 - \delta_v 30+, 2\delta_v]$
$\mathbf{v}(3)$	$[-\delta_v, 2\delta_v]$ $[-90 - \delta_v, -90 + 2\delta_v]$
$\mathbf{x}(4)$	$[-\delta_x, 2\delta_x]$ $[-\delta_x, 2\delta_x]$

where δ_z, δ_v and δ_x are positive real numbers. Note that all domains have the form $[a] = [\hat{a} - \delta_a, \hat{a} + 2\delta_a]$, where $\hat{a}$ is the true value for the variable a and δ_a is a tuning error coefficient. These intervals have been chosen in order to satisfy the following properties: *(i)* $[a]$ always encloses the true value $\hat{a}$ for a, *(ii)* if $\delta_a = 0$, then $[a]$ becomes the degenerated interval $[\hat{a}, \hat{a}]$ (*i.e.*, a singleton $\{\hat{a}\}$), *(iii)* if $\delta_a > 0$, then the center of $[a]$ is not $\hat{a}$; this choice makes possible to avoid a particular unrealistic situation which could be favorable to some estimation methods. Any other arbitrary choice for the interval domains which would satisfy these three properties could have been done as well. Remember again that the components of $\mathbf{x}(1), \mathbf{x}(2), \mathbf{x}(3), \mathbf{m}$ are not measured, and thus the associated prior intervals are $]-\infty, \infty[$.

7.3 Interval propagation

With an interval approach, a random variable x of $\mathbb{R}$ is often represented by an interval $[x]$ which encloses the support of its probability function. This representation is of course poorer than that provided by its probability density distribution, but it presents several advantages. *(i)* Since an interval with non zero length is consistent with an infinite number of probability distribution functions, an interval representation is well adapted to represent random variables with imprecise probability density functions. *(ii)* An arithmetic can be developed for intervals, which makes it possible to deal with uncertainties in a reliable and easy way, even when strong nonlinearities occur. *(iii)* When the random variables are related by constraints (*i.e.*, equations or inequalities) a propagation process (which will be explained later) makes it

possible to get efficient polynomial algorithms to compute precise intervals that are guaranteed to contain all feasible values for the random variables.

7.3.1 Complex constraints

Contractors can be developed for complex constraints, but an adaptation is often required. For instance, if we consider the constraint $\dot{\mathbf{p}}(t) = \mathbf{R}(t).\mathbf{v}_r(t)$ where $\forall t \in [t_0, t_1]$, $\mathbf{R}(t) \in [\mathbf{R}]$ and $\mathbf{v}_r(t) \in [\mathbf{v}_r]$. Since

$$\mathbf{p}(t_1) = \mathbf{p}(t_0) + \int_{t_0}^{t_1} \mathbf{R}(t).\mathbf{v}_r(t) \in \mathbf{p}(t_0) + (t_1 - t_0).\,[\mathbf{R}]\,.\,[\mathbf{v}_r]\,,$$

the domains for $\mathbf{p}(t_0)$ and $\mathbf{p}(t_1)$ can be contracted as follows

$$\begin{array}{rcl} [\mathbf{p}]\,(t_1) & = & [\mathbf{p}]\,(t_1) \cap ([\mathbf{p}]\,(t_0) + (t_1 - t_0).\,[\mathbf{R}]\,.\,[\mathbf{v}_r])\,, \\ [\mathbf{p}]\,(t_0) & = & [\mathbf{p}]\,(t_0) \cap ([\mathbf{p}]\,(t_1) + (t_0 - t_1).\,[\mathbf{R}]\,.\,[\mathbf{v}_r])\,. \end{array}$$

For more information on the contraction of constraints described by differential equations, see, *e.g.*, [36]. There exists a lot of sophisticated methods to build contractors adapted to complex constraints . Some of them combine interval methods with formal calculus (for instance by computing the derivatives of the functions involved in the equations). Others have been built for specific important constraints (such as distance or angle constraints).

7.3.2 Forward-backward propagation

The interval propagation method converges to a box which contains all solution vectors of our set of constraints. If this box is empty, it means that there is no solution. It can be shown that the box to which the method converges does not depend on the order to which the contractors are applied [46], but the computing time is highly sensitive to this order. We have already shown that there is no optimal order in general, but in practice, one of the most efficient is called *forward-backward propagation*. It consists in writing the whole set of equations under the form $\mathbf{f}(\mathbf{x}) = \mathbf{y}$ where $\mathbf{x}$ and $\mathbf{y}$ correspond to quantities that can be measured (*i.e.*, some prior interval domains are given for them). Then, using interval arithmetic, the intervals are propagated from $\mathbf{x}$ to $\mathbf{y}$ in a first step (*forward propagation*) and, in a second step, the intervals are propagated from $\mathbf{y}$ to $\mathbf{x}$ (*backward propagation*). The forward-backward propagation is then repeated until no more significant contraction can be observed. To illustrate the principle, consider the equations (7.5). If $\mathbf{x} \stackrel{\text{def}}{=} (\mathbf{x}(0), \mathbf{v}(0), \ldots, \mathbf{v}(3), \mathbf{m})$ and $\mathbf{y} \stackrel{\text{def}}{=} (z(1), z(2), z(3), \mathbf{x}(1), \ldots, \mathbf{x}(4))$, then (7.5) can be rewritten under the form $\mathbf{f}(\mathbf{x}) = \mathbf{y}$. The forward contraction can be described by the following algorithm

FORWARD CONTRACTION	
1	for $k = 0$ to 3
2	$\quad [\mathbf{x}]\,(k+1) = ([\mathbf{x}]\,(k) + [\mathbf{v}]\,(k)) \cap [\mathbf{x}]\,(k+1)$
3	end
4	for $k = 1$ to 3
5	$\quad [\mathbf{d}]\,(k) = [\mathbf{x}]\,(k) - [\mathbf{m}]$
6	$\quad [z]\,(k) = \sqrt{([d_1]\,(k))^2 + ([d_2]\,(k))^2} \cap [z]\,(k)$
7	end

where $[\mathbf{d}](k)$ is the box $[d_1](k) \times [d_2](k)$. The backward propagation is described by

BACKWARD CONTRACTION	
1	for $k = 3$ to 1
2	$[d_1](k) = \sqrt{([z](k))^2 - ([d_2](k))^2} \cap [d_1](k)$
3	$[d_2](k) = \sqrt{([z](k))^2 - ([d_1](k))^2} \cap [d_2](k)$
4	$[\mathbf{x}](k) = ([\mathbf{d}](k) + [\mathbf{m}]) \cap [\mathbf{x}](k)$
5	$[\mathbf{m}] = ([\mathbf{x}](k) - [\mathbf{d}](k)) \cap [\mathbf{m}]$
6	end
7	for $k = 3$ to 0
8	$[\mathbf{x}](k) = ([\mathbf{x}](k+1) - [\mathbf{v}](k)) \cap [\mathbf{x}](k)$
9	$[\mathbf{v}](k) = ([\mathbf{x}](k+1) - [\mathbf{x}](k)) \cap [\mathbf{v}](k)$
10	end.

7.3.3 Results on our simple SLAM example

Consider again the simple SLAM example of Section 7.2.2. If we apply an elementary interval propagation for different values of δ_x, δ_v and δ_z, we get the contracted intervals given in the following table.

	case 1	case 2	case 3
$z(1)$	$[50]$	$[50]$	$[49, 52]$
$z(2)$	$[60]$	$[60]$	$[59, 62]$
$z(3)$	$[50]$	$[50]$	$[49, 52]$
$\mathbf{m}$	$[40]$ $[60]$	$[38, 44.23]$ $[58.93, 62]$	$[36, 45.34]$ $[57.58, 64]$
$\mathbf{v}(0)$	$[40]$ $[10]$	$[39, 42]$ $[9, 12]$	$[39, 42]$ $[9, 12]$
$\mathbf{v}(1)$	$[60]$ $[50]$	$[59, 62]$ $[49, 52]$	$[59, 62]$ $[49, 52]$
$\mathbf{v}(2)$	$[-100]$ $[30]$	$[-101, -98]$ $[29, 32]$	$[-101, -98]$ $[29, 32]$
$\mathbf{v}(3)$	$[0]$ $[-90]$	$[-1, 2]$ $[-91, -88]$	$[-1, 2]$ $[-91, -88]$
$\mathbf{x}(1)$	$[40]$ $[10]$	$[39, 42]$ $[9, 12]$	$[39, 42]$ $[9, 12]$
$\mathbf{x}(2)$	$[100]$ $[60]$	$[98, 104]$ $[58, 64]$	$[98, 104]$ $[58, 64]$
$\mathbf{x}(3)$	$[0]$ $[90]$	$[-3, 6]$ $[87.49, 94.87]$	$[-3, 6]$ $[87, 96]$
$\mathbf{x}(4)$	$[0]$ $[0]$	$[-4, 8]$ $[-4, 6.54]$	$[-4, 8]$ $[-4, 8]$

- Case 1. $\delta_v = 0, \delta_z = 0, \delta_x = \infty$. We are in the unrealistic situation with no noise on the evolution and no noise on the distance measurements. An interval propagation contracts the domains for

$\mathbf{m}, \mathbf{x}(1), \mathbf{x}(2), \mathbf{x}(3), \mathbf{x}(4)$ (which were initially equal to $]-\infty, \infty[^2$) to singletons. The obtained results are consistent with the exact solution.

- Case 2. $\delta_v = 1, \delta_z = 0, \delta_x = \infty$. The evolution is now noisy and we are still able to localize the mark with a rather good accuracy. The localization of the mark allows a small improvement of the accuracy on the trajectory (see the domain for $\mathbf{x}(3)$).
- Case 3. $\delta_v = 1, \delta_z = 1, \delta_x = \infty$. Now, there is also an error on the distance measurement. The location of the mark is less accurate than for Case 2. This additional error prevents any improvement in the accuracy of the trajectory envelope.

Other cases have been considered. For instance, if δ_x is small, the localization of the mark as well as the estimation of the trajectory becomes more accurate. The computing time to get the contracted intervals is always less than 0.1 sec on a standard laptop.

7.4 SLAM in a submarine context

7.4.1 Mine hunters

We shall consider a class of autonomous underwater robots, called *mine hunters.* These robots are used to detect near-shore underwater mines. Most mine hunters are equipped with a loch-Doppler, a gyrocompass, an altimeter and a barometer. They also enclose one or two sidescan sonars to localize seamarks such as rocks or mines. The SLAM problem of mine hunters can be described by the SLAM equations (7.1), as it will now be shown.

7.4.2 Evolution equation

Around the zone covered by the robot, let us build the frame $(\mathbf{O}, \mathbf{i}, \mathbf{j}, \mathbf{k})$ where the vector $\mathbf{i}$ indicates the north, $\mathbf{j}$ indicates the east and $\mathbf{k}$ is oriented toward the center of the earth. Denote by $\mathbf{p} = (p_x, p_y, p_z)$ the coordinates of the robot expressed in the frame $(\mathbf{O}, \mathbf{i}, \mathbf{j}, \mathbf{k})$. The robot motion can be described by

$$\dot{\mathbf{p}} = \mathbf{R}_{\text{Euler}}(\varphi, \theta, \psi).\mathbf{v}_r, \tag{7.6}$$

where $\mathbf{v}_r$ represents the speed vector of the robot measured by a loch-Doppler and $\mathbf{R}_{\text{Euler}}(\varphi, \theta, \psi)$ is the Euler rotation matrix returned by the gyrocompass (the roll is φ, the pitch is θ and the head is ψ). Equation (7.6) corresponds to the first equation of (7.1), where the state vector is $\mathbf{p}$ and the input vector is $\mathbf{u} = (\varphi, \theta, \psi, v_r^x, v_r^y, v_r^z)$.

7.4.3 Observation equation

The observation equation is given by $\mathbf{y} = \mathbf{p}$ and corresponds to the second equation of (7.1). The two first coordinates p_x and p_y of $\mathbf{p}$ can be measured by the GPS (Global Positioning System) with an accuracy less than 2.5 meters. Since electromagnetic waves (here around 1.2 GHz.) do not propagate through the water, the GPS is operational only when the robot is at the surface of the ocean. During a typical

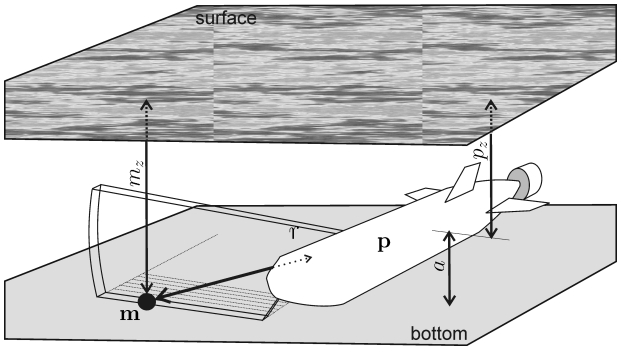


Figure 7.2: Principle of a lateral sonar with an antenna starboard

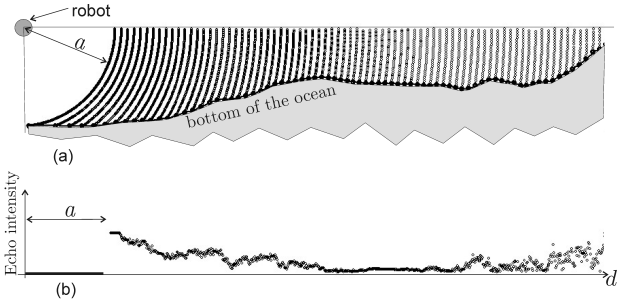


Figure 7.3: When the robot emits a ping (*i.e.,* a short ultrasonic sound), it receives an echo; Subfigure (a) represents the robot and the propagation of the sound, Subfigure (b) represents the intensity of the corresponding echo with respect to the distance. The quantity a is the distance between the robot and the bottom. When $d < a$ no echo can be detected.

underwater mission the robot is only able to measure its location with the GPS before diving and once it comes back to the surface. A barometer is used to measure the depth p_z of the robot (*i.e.*, the distance between the robot and the surface of the ocean).

7.4.4 Mark equation

To detect mines, a mine hunter uses a sidescan sonar that has either an antenna starboard, either an antenna portside. It may also have two antennas on both sides. Sometimes, a mine hunter uses a camera located below it and oriented toward the bottom. A side-scan sonar sends a *ping, i.e.*, a short ultrasonic wave on a thin plane located on the right or on the left of the robot and perpendicular to its main axis (see Figure 7.2). It receives an echo as illustrated by Figure 7.3.

From all these pings, the sidescan sonar builds a long image, an horizontal slice of such an image is represented on Figure 7.4. The image, called a *waterfall,* has a length which corresponds to the distance covered by the robot during its mission. Its width corresponds to the product of the distance covered by the sonar by the number of antennas (one or two). After the mission, a scrolling of the waterfall is performed by a human operator. He is then able to get an estimation $\tilde{r}(t)$ of the distance $r(t)$ between the robot and the seamark detected at time t. From the width of the black vertical strip on the left of

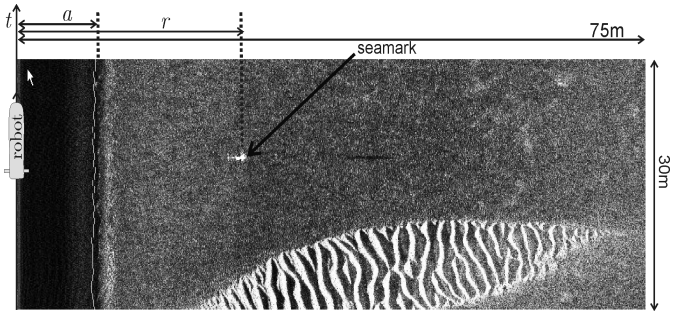


Figure 7.4: Slice of the waterfall built by a sonar with one antenna starboard; this sonar image makes it possible to detect a seamark (here, a mine), to compute the distance r between the seamark and the robot

the picture (called the *water column*), we are also able to compute an estimation $\tilde{a}(t)$ of the altitude $a(t)$ of the robot (distance between the robot and the bottom), but this estimation is less precise and less robust (*i.e.*, it can generate outliers) than that provided by the loch-Doppler. Thus, we will not take it into account for the SLAM.

When the human operator detects a seamark on the waterfall, he is able to see if it is on the right or on the left of the robot and he is also able to measure the distance between the robot and the mine. As a consequence, we shall define the *lateral distance* between the robot at position $\mathbf{p}$ and the ith mark $\mathbf{m}_i$ as follows

$$r(\mathbf{p},\psi,\theta,\varphi,\mathbf{m}_i) \stackrel{\text{def}}{=} \begin{cases} \frac{-||\mathbf{e}_i||}{\text{sign}(\mathbf{j}_r^{\mathrm{T}}.\mathbf{e}_i)} & \text{if } \begin{cases} \mathbf{i}_r^{\mathrm{T}}.\mathbf{e}_i = 0 \\ \mathbf{k}_r^{\mathrm{T}}.\mathbf{e}_i \leq 0 \end{cases} \\ \infty & \text{otherwise,} \end{cases} \tag{7.7}$$

where $\mathbf{i}_r$, $\mathbf{j}_r$ and $\mathbf{k}_r$ are the three column vectors of the rotation matrix $\mathbf{R}_{\text{Euler}}(\varphi,\theta,\psi)$ and

$$\mathbf{e}_i \stackrel{\text{def}}{=} \mathbf{p} - \mathbf{m}_i. \tag{7.8}$$

Let us now explain this definition (see Figure 7.5). The lateral sonar is only able to detect a mark which is on its lateral plane, perpendicular to the main axis of the robot. Thus, if the mark is outside this plane (*i.e.*, $\mathbf{i}_r^{\mathrm{T}}.\mathbf{e}_i \neq 0$), the lateral distance will be undefined (*i.e.*, equal to ∞). Moreover, the detected mark is always below the robot (*i.e.*, $\mathbf{k}_r^{\mathrm{T}}.\mathbf{e}_i \leq 0$) otherwise, the lateral distance is undefined (*i.e.*, equal to ∞). If the mark is starboard (*i.e.*, $\mathbf{j}_r^{\mathrm{T}}.\mathbf{e}_i \leq 0$) then, by convention, the lateral distance r_i is positive and equal to $||\mathbf{e}_i||$. If it is portside (*i.e.*, $\mathbf{j}_r^{\mathrm{T}}.\mathbf{e}_i \geq 0$), then $r_i = -||\mathbf{e}_i||$. Moreover, if we assume that the ocean bottom is approximately flat (at least locally), the altitude of the robot and its depth can be used to get a measurement of the depth of the seamark by the relation

$$m_i^z = a + p_z + s_i, \tag{7.9}$$

where s_i is a variable that could be bounded if we could give a bound on the slopes of the bottom of the ocean in the area covered by the robot. As a consequence, the mark equation is composed with the lateral distance of the seamark and the depth of the seamark. It has the form

$$\mathbf{z}_i = \mathbf{h}(\mathbf{p},\mathbf{u},\mathbf{m}_i) = \begin{pmatrix} r(\mathbf{p},\psi,\theta,\varphi,\mathbf{m}_i) \\ m_i^z \end{pmatrix}. \tag{7.10}$$

Note that, since $r(\mathbf{p},\psi,\theta,\varphi,\mathbf{m}_i)$ is given by (7.7), the function $\mathbf{h}$ is not continuous.

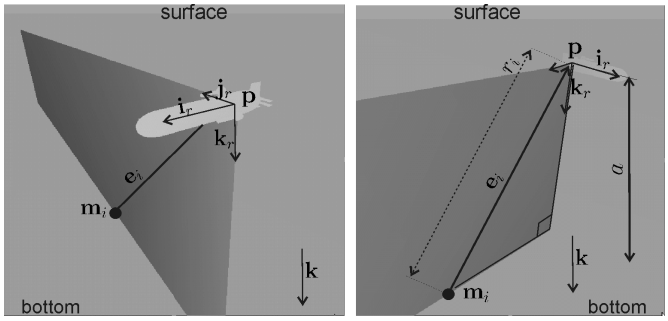


Figure 7.5: Distance between the robot and a seamark obtained using a lateral sidescan sonar with an antenna starboard

7.4.5 SLAM formulation for mine hunters

The previous developments leads to the following SLAM formulation (see (7.1))

$$\left\{ \begin{array}{rcl} \dot{\mathbf{p}} & = & \mathbf{R}_{\text{Euler}}(\varphi,\theta,\psi).\mathbf{v}_r \\ \mathbf{y} & = & \mathbf{p} \\ \mathbf{z}_i & = & \begin{pmatrix} r(\mathbf{p},\psi,\theta,\varphi,\mathbf{m}_i) \\ m_i^z \end{pmatrix} \end{array} \right. \tag{7.11}$$

where $\mathbf{u} = (\varphi, \theta, \psi, v_r^x, v_r^y, v_r^z)$. Recall that in the context of mine hunters, we have accurate interval domains for all components of $\mathbf{u}(t)$, $\forall t \in t \in [t_0, t_f]$. We also have accurate domains for $r_i(t)$ and m_i^z for any pair (t, i) corresponding to a detection. An interval propagation can thus be launched, to find accurate interval domains for all other variables involved in our set of equations.

Remark 2 *If a scalar decomposition of the vector equations is performed, the set of constraints associated with the SLAM problem involves $n_t.n_p+3.n_m$ unknown variables that cannot be measured, where n_t is the number of sampling times, $n_p = 3$ is the dimension of $\mathbf{p}$, and n_m is the number of mines. The number of equations is $n_t.(n_p + n_z)$, where n_z is the dimension of $\mathbf{z}_i$. For the actual experiment to be considered on Section 7.5, we will have $n_t = 60000$, $n_m = 6$ and thus we have a problem with 180018 variables for 300 000 equations. The fact that we have more equations than unknowns creates the redundancy needed by interval propagation methods to solve efficiently and rigorously such high dimension problems.*

7.4.6 GESMI

In Section 7.3, we have shown that a SLAM problem could be cast into a set of nonlinear constraints where interval domains were available for each variable. We have also presented an interval propagation approach to contract domains for the variables and we have illustrated how interval propagation can be used on the simple SLAM problem presented on Section 7.2.2. An interval SLAM solver named GESMI (Guaranteed Estimation of Sea Marks with Intervals) [38] has been developed. The interval propagation performed by GESMI is mainly based on a forward-backward interval propagation. The reason for choosing a forward-backward strategy is mainly due to the large number of constraints involved in our problem. There exists two versions of GESMI: one running under a *Windows* environment with a nice

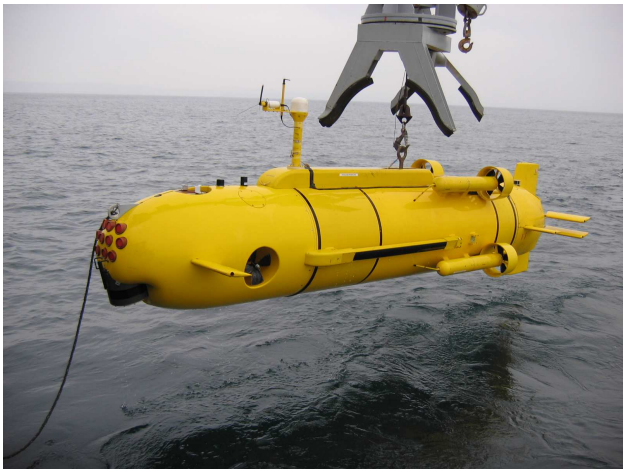


Figure 7.6: The autonomous underwater vehicle, *Redermor*, built by the GESMA (Groupe d'Etude Sous-Marine de l'Atlantique)

OpenGL-based interface and one running under *Linux* without graphical interface. Up to now, the *Windows* version does not implement the outward rounding needed to get the guarantee of the results with respect to rounding errors. On the other hand, the *Linux* version is based on the validated interval library IBEX (`http://ibex-lib.org/`) and performs an outward rounding for each interval operation. The *Windows* solver as well as the data collected during the experiment presented on the next section have been made available [38]. The software is easy to use and allows the reader to get a quick idea about the efficiency of the method.

7.5 Experimental setup

7.5.1 Description of the robot

The robot to be considered in this experiment (see Figure 7.6) is a mine hunter, named *Redermor* (means *sea runner*, in the Breton language). This robot, developed by the GESMA (Groupe d'Etude Sous-Marine de l'Atlantique, a center of the French defense ministry which supervises most of the research in French underwater robotics), has a length of 6 m, a diameter of 1 m and a weight of 3800 Kg. It has powerful propulsion and control system able to provide hovering capabilities. The main purpose of the *Redermor* is to evaluate improved navigation by the use of sonar information. Note that the use of sonars for improving the localization and the navigation has already been proven on several robotics applications (see, *e.g.*, [55]).

7.5.2 Description of the mission and sensors

The experiment to be now presented has been performed by the GESMA in the Douarnenez bay, in Brittany (France). At time $t_0 = 0.0$ s, the robot has been dropped approximately around the position $\tilde{\ell}^0 = (\tilde{\ell}^0_x, \tilde{\ell}^0_y) = (-4.458227931^{\text{o}}, 48.212920614^{\text{o}})$, measured by the GPS, where ℓ^0_x is the west/east lon-

gitude and ℓ_y^0 is the south/north latitude. The error of the GPS is known to be less than 0.0000135^o for ℓ_x^0 and less than 0.0000090^o for ℓ_y^0 in this part of the earth which corresponds approximately to a position error less than 2.5 meters. When the robot surfaces, at time $t_f = 5999.4$ sec, its position is also measured as $\tilde{\ell}^f = (\tilde{\ell}_x^f, \tilde{\ell}_y^f) = (-4.454660760^o, 48.219129760^o)$, with the same accuracy. The loch-Doppler is a *Workhorse Navigator Doppler Velocity Log* from *RD-instruments company*. The frequency of the waves are around 300 kHz. The actual speed (in meters per second) is known to satisfy

$$\mathring{\mathbf{v}}_r \in \widetilde{\mathbf{v}}_r + 0.004 * [-1, 1] . \widetilde{\mathbf{v}}_r + 0.004 * [-1, 1], \tag{7.12}$$

where $\widetilde{\mathbf{v}}_r$ denotes the three dimensional speed vector returned by the sensor (see `www.rdinstruments.com/navigator.html` for more information about these bounds). The loch-Doppler is also able to provide the altitude a of the robot with an error less than 10cm. The gyrocompass that has been used is an Octans III from IXSEA. If we denote by $\left(\tilde{\varphi}, \tilde{\theta}, \tilde{\psi}\right)$, the angles (in radians) returned by our gyrocompass, then the actual Euler angles for our robot should satisfy (see `www.ixsea.com` for the technical characteristics of the sensor)

$$\begin{pmatrix} \mathring{\varphi} \\ \mathring{\theta} \\ \mathring{\psi} \end{pmatrix} \in \begin{pmatrix} \tilde{\varphi} \\ \tilde{\theta} \\ \tilde{\psi} \end{pmatrix} + \begin{pmatrix} 1.75 \times 10^{-4}. [-1, 1] \\ 1.75 \times 10^{-4}. [-1, 1] \\ 5.27 \times 10^{-3}. [-1, 1] \end{pmatrix}. \tag{7.13}$$

If $\tilde{d}$ is the depth (in meters) collected by the barometer, then the actual depth of the robot satisfies $\mathring{p}_z(t) \in [-1.5, 1.5] + \tilde{d} + 0.2 * [-1, 1].\tilde{d}$. The interval $[-1.5, 1.5]$ may change depending on the strength of waves and tides.

Thus, for each time $t \in \mathcal{T} \stackrel{\text{def}}{=} \{0.0, 0.1, 0.2, \ldots, 5999.4 \text{ sec}\}$, the vector of measurements ($\tilde{\varphi}(t)$, $\tilde{\theta}(t)$, $\tilde{\psi}(t)$, $\tilde{v}_r^x(t)$, $\tilde{v}_r^y(t)$, $\tilde{v}_r^z(t)$, $\tilde{a}(t)$, $\tilde{d}(t)$) is collected. This vector can be represented by a point in the 8-dimensional sensor space (where 8 is the number of corresponding sensors) and approximates the actual value

$$\left(\mathring{\varphi}(t), \mathring{\theta}(t), \mathring{\psi}(t), \mathring{v}_r^x(t), \mathring{v}_r^y(t), \mathring{v}_r^z(t), \mathring{a}(t), \mathring{p}_z(t)\right) \tag{7.14}$$

for the measurement vector, with an error which can be bounded by taking into account the characteristics of the sensors. For each $t \in \mathcal{T}$, we are thus able to get a box in $\mathbb{R}^8$ which contains the actual measurement vector.

The sidescan sonar used for detection is a KLEIN 5400 (see `www.l-3klein.com` for detailed characteristics of this sensor). It has a single antenna located starboard (*i.e.* on right-hand side) and its scope is about 75m. Every 0.1 second, the sonar sends a *ping, i.e.*, a short ultrasonic wave with frequency 455KHz, on a thin plane located on its right and perpendicular to the main axis of the robot (see Figure 7.2). From the sonar waterfall, provided by the robot after its mission, six mines $\mathbf{m}_0, \ldots, \mathbf{m}_5$ have been detected

manually. From these detections the following *mark table* can thus be built:

t(sec)	i	$\tilde{r}_i$ (meters)
1054	1	52.42
1092	2	12.47
1374	1	54.40
1748	0	52.68
3038	1	27.73
3688	5	26.98
4024	4	37.90
4817	3	36.71
5172	3	37.37
5232	4	31.03
5279	5	33.51
5688	1	15.05

(7.15)

The table provides *(i)* the time t a mine has been detected starboard, *(ii)* the number i of the detected mine and *(iii)* a measure $\tilde{r}_i(t)$ of the lateral distance between the robot and the mine $\mathbf{m}_i$. The actual distance $\hat{r}_i(t)$ is assumed to satisfy the relation $\hat{r}_i(t) \in [\tilde{r}_i(t) - 1, \tilde{r}_i(t) + 1]$. The set $\mathcal{M}$ of all (t, i) such that the ith mine has been detected at time t can be built from the mark table. We get

$$\mathcal{M} = \{(1054, 1), (1092, 2), (1373, 1), \ldots (5279, 5), (5688, 1)\}\,.$$

For the flatness characteristics of the bottom, we took a maximum slope of 1%, which makes a measurement of the depth of the mine m_i^z with an accuracy less than $0.1 + 2.5 + 0.01 * 75 = 3.35$m (0.1m for the accuracy of a, 2.5m for p_z, 75m for the distance covered by the lateral sonar). Figure 7.4 is related to the detection of a mine (here, it is a spheric floating mine connected to its sinker by a tether).

7.5.3 Results

We applied GESMI to solve the SLAM problem related to this experiment. The results obtained after ten interval forward-backward propagations are illustrated on Figure 7.7. The envelope painted gray encloses the trajectory of the robot, the six black boxes contain the six mines detected by the sonar and the two black circles represent the initial and final poses of the robot. These enclosures result from a merge of the information given by the GPS (available at the beginning and at the end of the mission), the detected mines, the loch-Doppler, the gyrocompass and the barometer. The accuracy of the locations of the robot and the mines is always better than 30 meters. The computing time to get these results is less than 30 seconds with a Pentium III under the *Windows* version of GESMI (without the outward rounding). With the *Linux* version of GESMI, which performs an outward rounding, the computing time is less than 55 seconds. Note that both versions provide exactly the same interval domains (up to five significant digits).

Remark 3 *We are not supposed to know the location of these six mines. However, when we dropped them, we measured our location with a GPS, and we used this information to check the consistency of results obtained by the interval propagation.*

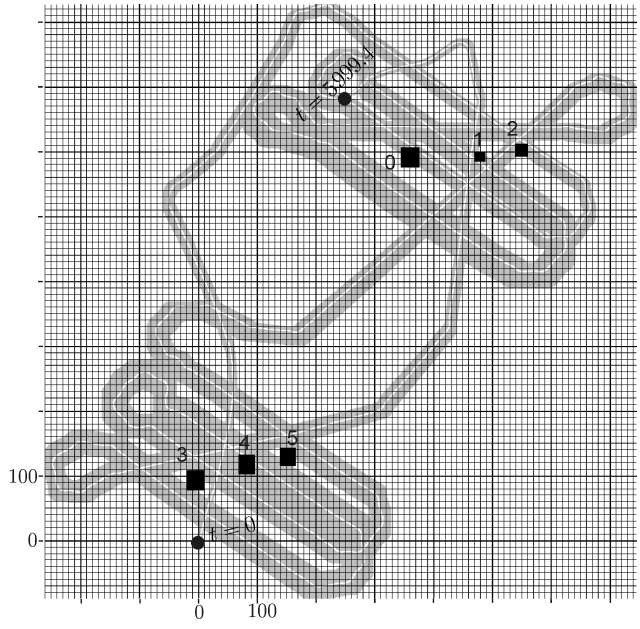


Figure 7.7: Envelope enclosing the trajectory of the robot and the six boxes containing the six mines detected by the sonar

Remark 4 *Remember that our interval estimation has been done after the mission and thus the robot did not use SLAM to control its trajectory. During its mission, the robot localizes itself using a dead reckoning approach, (*i.e.*, it does not use the lateral sonar, but only the loch-Doppler, the gyrocompass and the barometer). The robot tries to follow a trajectory using an elementary PID controller. The trajectory has been defined through a list of waypoints that have to be reached. These waypoints have been given by the user, before the mission. Dead-reckoning generates a drift which can be estimated here to 100 meters for one hour mission and thus, the waypoints have been reached with a bad precision. However, this can be considered as sufficient for controlling the robot for a two-hour mission in the ocean without being lost. Now, the aim of our robot is to detect and localize mines. This localization should be as precise as possible (less than 30 meters) in order to be able to send another teleoperated robot equipped with cameras, to destroy the detected mines. Such a reliable accuracy of 30 meters could not have been obtained without using a SLAM approach.*

7.5.4 Waterfall

Figure 7.8 contains the reconstructed waterfall (left side) and one zoom (right side). Each column corresponds to one of the six mines ($i = 0, \ldots, 5$). The gray areas contain the set of all feasible pairs $(t, \|\mathbf{e}_i(t)\|)$ obtained after the interval propagation. The twelve small black disks correspond to the mines detected by the human operator. From each disk, we can get the time t the mine has been detected (y-axis), the number of the mine and the distance r_i between the robot and the mine (x-axis). Thus, from the twelve black disks of the figure, we are able to reconstruct the mark table. Black areas correspond to all feasible (t, r_i). Some of these areas are tiny and are covered by a black disk. Some are larger and do not contain any black disk. In such a case, an existing mine has been missed by the operator during the scrolling of the waterfall. As a consequence, with the help of Figure 7.8, the operator could scan again the

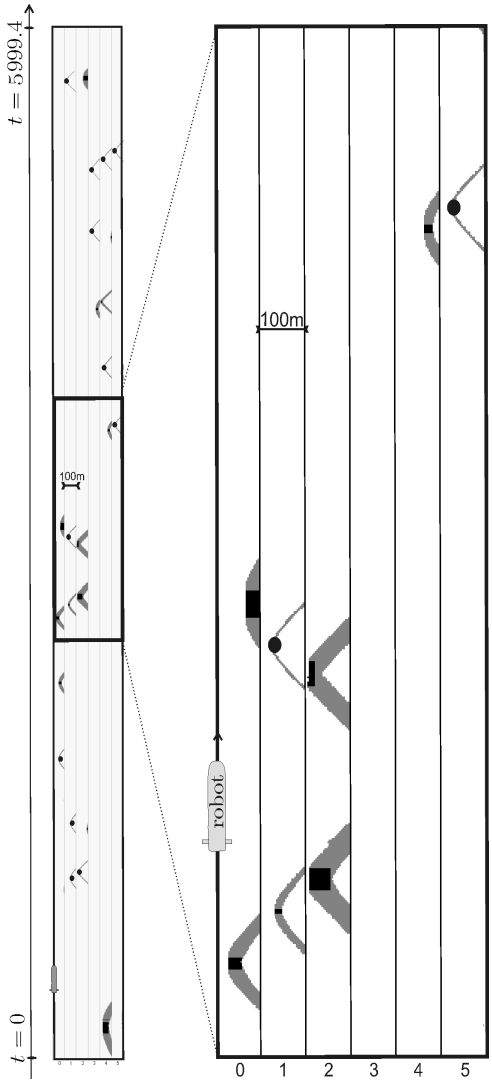


Figure 7.8: The reconstructed waterfalls can help the human operator to find undetected mines.

waterfall and find undetected mines much more efficiently. If the operator is able to detect at least one more mark, then, the propagation algorithm can run again to get a thinner envelope for the trajectory, thinner black areas in the reconstructed waterfall and thus a higher probability to detect new mines on the waterfall. The operator can thus be seen as a contractor inside an interval propagation process. Up to now, the detection procedure is done by hand. Now, new techniques for automatic detections of seamarks are more and more reliable (see, *e.g.*, [73] in the situation where the seamarks are mines). Such automatic detections could of course be included in our propagation process.

7.6 Comparison with existing methods

The class of SLAM problems treated with the interval propagation approach can also be treated using more classical Bayesian smoothers. For instance, in the experiment considered in the previous section,

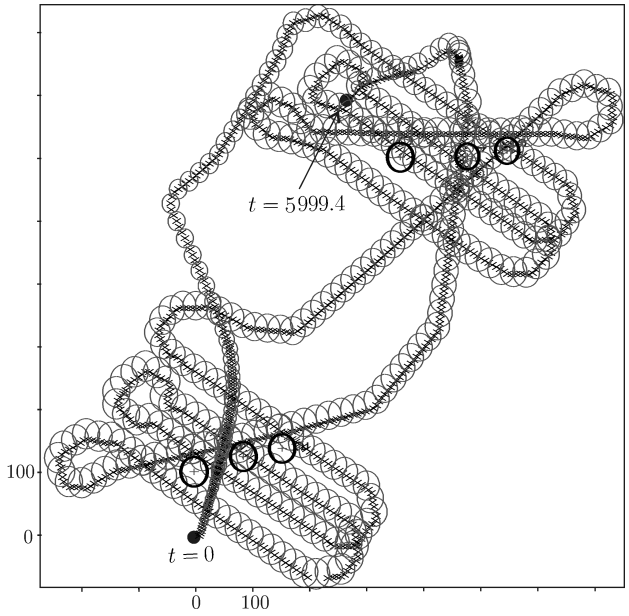


Figure 7.9: Confidence ellipses generated by a classical extended Kalman smoother; the six thick ellipses correspond to the six mines

a Kalman smoother leads to the trajectory envelope represented on Figure 7.9. To apply a Kalman smoother, the Euler angles φ, θ, ψ are assumed to be measured without any error (otherwise the noise would become multiplicative and Kalman filter would not apply) and the mark equation has to be linearized. For our application, we observed that the resulting envelope as well as the ellipses for the marks are accurate and consistent with the results obtained by an interval propagation. This consistency of the results is mainly due to the fact that linearization as well as the assumption on the exact knowledge of the Euler angles do not affect the quality of the results. Let us now give a list a comparisons between the interval propagation approach and other more classical Bayesian approaches for SLAM.

- **Reliability**. If the assumption on both the model and the bound errors are correct, then the interval method will provide guaranteed results. This is not the case of other Bayesian smoothers. The Kalman-based methods do not provide any bound on the linearization errors and particle-based methods do not provide any quantification of the error due to the finite number of particles. Only the interval propagation method is able to provide an envelope which encloses all feasible trajectories of the robot in a nonlinear context.

- **Inconsistency.** When the model is not anymore valid (for instance when the robot meets the bottom), when outliers occur in data, ..., then the interval propagation method generally yields an empty-set. An inconsistency often translates some anomalies during the mission of the robot and should thus been identified in order to realize a reliable navigation system. Detecting inconsistencies can thus be useful in practice to detect conception bugs which affect the behavior of the robot.

- **Validation**. On the one hand, if some sensors are not reliable or if the model is not precise enough, the interval propagation system usually detects an inconsistency. On the other hand, if the interval propagation provides a consistent and small envelope for the trajectory and the marks, then the robot equipped with its sensors can be considered as reliable (this is not the case for the Kalman

filter which often provides a thin envelope even if some assumptions on the sensors are wrong). As a consequence, the interval propagation can be considered as a reliable way to validate the quality of a navigation system.

- **Mark association**. Interval propagation can easily be combined with other propagation methods to deal with problems involving Boolean or integer variables. For instance, if, in the mark table (7.15), we remove all mark numbers in the column 2, and if we only give the information that we have six marks, then the propagation is able to find all correspondences between marks in less than 3 minutes.

- **Approximation.** Interval propagation methods never linearize the equations (contrary to the Kalman smoother); they do not assume that the noise is additive; they do not approximate probability density functions (contrary to most Bayesian smoothers); they do not assume that the computer computes with real numbers instead of floating point numbers. Of course, they consider that the noise is bounded, that reliable boundaries are known, that the model is correct and that no outlier occurs, but these assumptions are already in the definition of the problem. Interval propagation do not bring any further approximation as it is the case for other nonlinear SLAM methods.

7.7 Conclusion

In this chapter, we have shown that interval constraint propagation could be seen as an efficient alternative to Bayesian approaches for solving SLAM problems in a bounded-error context. Of course, as presented here, the SLAM method is offline and requires a human operator to detect seamarks and to make the mark associations. A lot of work is left to obtain an online SLAM method. However, the approach has been demonstrated on a real problem with an actual underwater robot (the *Redermor*).

The main advantages of the interval propagation approach are its generality, its simplicity, its reliability and its efficiency. For instance, if we have several robots communicating together, with different kind of sensors, ..., the SLAM problem can be cast into a huge nonlinear set of constraints. Without any approximation (such as a linearization) or tuning (such as the number of particles), the interval propagation is able to provide an envelope that encloses the trajectories of the robots and to build an interval map for the surrounding marks. Whereas classical Bayesian can be considered as more efficient to get the best trajectory and the best mark positions, an interval propagation is more adequate to find an envelope for them.

In practice, when strong outliers occur, an interval approach generally returns the empty set and we are not able to have an estimation of the location of the robot. If we really need such an estimation, a possible alternative is to maximize the number of constraints that can be satisfied [54]. This can be done efficiently using interval propagation [49]. The corresponding approach has been shown to be robust with respect to outliers.

Chapter 8

Range-only SLAM with occupancy maps

Abstract. This chapter [42] proposes a set-membership method based on interval analysis to solve the simultaneous localization and map building (SLAM) problem. The principle of the approach is to cast the SLAM problem into a constraint satisfaction problem for which interval propagation algorithms are particularly powerful. The resulting propagation method is illustrated on the localization and map building of an actual underwater robot.

8.1 Introduction

The SLAM (Simultaneous Localization And Mapping) problem [56] for an autonomous robot moving in an unknown environment is to build a map of this environment while simultaneously using this map to compute its location. The history and critical issues of SLAM are discussed in [27]. SLAM methods can be classified in two categories [80], referred to as *feature-based* SLAM and *location-based* SLAM. Feature-based SLAM assume that the map is composed of a set of features together with their Cartesian location. The map has thus a parametric structure where the features are points, segments, corners or any other parametric shape [9]. The seminal chapter [74] defines the way to handle uncertain location vectors when using geometric features of the environment as map elements. Many implementations use the segment [67] or the line as the main kind of feature, and some of them use corners or edges modeled as points. Feature-based SLAM problem can be cast into a state estimation problem, by including the feature parameters among the state variables [20], [62]. Probabilistic techniques (Kalman filtering, Bayesian estimation, particle filters) [80] or set membership approaches (where sets can be represented by parallelotopes [17], [18] or by boxes [39]) have been proven to solve efficiently the feature-based SLAM problem. Now feature-based maps not well suited to model non-structured environments, as for underwater robotics where landmarks have no particular geometric shape. Location-based maps offer a label to any location in the world. They contain information not only about obstacles in the environment, but also about the absence of obstacles. A classical location-based map representation is known as *occupancy map* [23] (also called *pose-based* map). They assign to each point of the world an occupancy value (a Boolean number

or a probability of occupancy) which specifies whether or not a pose is occupied by an obstacle. When the occupancy value is binary, the map can be represented by a subset $\mathbb{M}$ (that will be called the *map* in this chapter) which distinguishes free from occupied terrain. The robot's pose must always be in the free space, i.e., outside $\mathbb{M}$. The corresponding SLAM problem contains some unknown variables which are subsets of $\mathbb{R}^q$, where $q = 2$ or 3, depending of the dimension of the robot's environment (2D or 3D) and it cannot be cast into a state estimation problem anymore. The problem has a totally different nature and is much harder to solve than when it is possible to detect parametric shapes around the robot. It becomes even more difficult when the map is only perceived through omnidirectional rangefinders, which only returns the distance to the closest obstacle without any angle information. Using such a range-only sensor precludes easy matching between detected points of the map. In an academic point of view, the omnidirectional range-only posed-based SLAM can be seen as a canonical problem: it is the simplest and most significant representative of a large class of SLAM problems that cannot be solved yet. Developing tools to solve properly and efficiently this problem will be useful to solve many other SLAM problems.

This chapter proposes a set-membership method to deal with the range-only SLAM problem in the case where the map is represented by an occupancy set. It first shows that the SLAM problem can be transformed into a hybrid constraint satisfaction problem where the variables are subsets of $\mathbb{R}^q$. Then the chapter extends existing constraint propagation methods in order to deal with problems involving sets. This can be done thanks to the notion of set intervals recently introduced in [41].

8.2 Formulation of the range-only SLAM

A range-only simultaneous localization and map building (SLAM) problem can be described by

$$\left\{ \begin{array}{lcll} \dot{\mathbf{x}}(t) & = & \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)) & \text{(evolution equation)} \\ z(t) & = & d(\mathbf{x}(t), \mathbb{M}) & \text{(map equation)} \end{array} \right. \tag{8.1}$$

where $t \in [t] \subset \mathbb{R}$ is the time, $\mathbf{x} \in \mathbb{R}^n$ is the state vector, $\mathbf{u} \in \mathbb{R}^m$ is the input vector (in general associated to proprioceptive sensors), $\mathbf{f} : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ is the evolution function and d is the map function. The set $\mathbb{M} \in \mathcal{C}(\mathbb{R}^q)$ is the occupancy map, where $\mathcal{C}(\mathbb{R}^q)$ denotes the set of all compact sets of $\mathbb{R}^q$ and q is the dimension of the map (two or three in practice). The scalar z is an exteroceptive measurement collected by robot (for instance by a sonar telemeter) and provides some information on $\mathbb{M}$. The map $\mathbb{M}$ is unknown and should be reconstructed together with the state $\mathbf{x}$. In this chapter, a set-membership approach will be considered, i.e., we shall assume that $\mathbf{u}(t), z(t)$ are known to belong to some known intervals $[\mathbf{u}](t), [z](t)$. We shall also assume that the map function $d : \mathbb{R}^n \times \mathcal{C}(\mathbb{R}^q) \to \mathbb{R}^+$ corresponds to a rangefinder, i.e., for all $\mathbf{x}, \mathbb{M}_1, \mathbb{M}_2$, we have

$$\left\{ \begin{array}{l} d(\mathbf{x}, \mathbb{M}_1 \cup \mathbb{M}_2) = \min\{d(\mathbf{x}, \mathbb{M}_1), d(\mathbf{x}, \mathbb{M}_2)\} \\ d(\mathbf{x}, \emptyset) = +\infty. \end{array} \right. \tag{8.2}$$

This assumption will provide us some conditions that will be used to characterize the map $\mathbb{M}$. Note that most rangefinders (laser, infrared or ultrasound based) satisfy the condition (8.2) whereas cameras (although often used for SLAM, see, e.g. [14] where Davison proposed a vision-based real-time SLAM) do not. Moreover, since the observations are only made by relative measurements between the robot and the environment, the initial state vector is assumed to be known. Define the function $\delta_{\mathbf{x}} : \mathbb{R}^q \to \mathbb{R}$ as

$$\delta_{\mathbf{x}}(\mathbf{a}) = d(\mathbf{x}, \{\mathbf{a}\}). \tag{8.3}$$

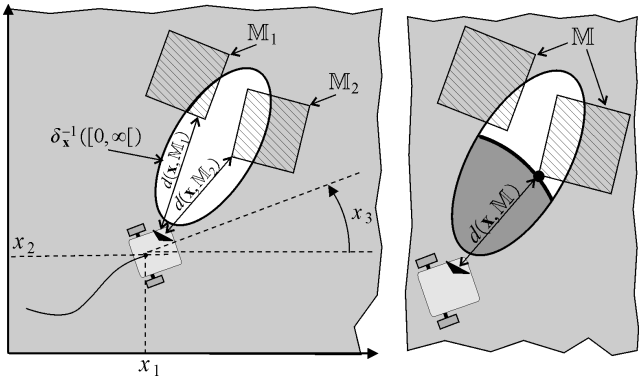


Figure 8.1: Illustration of the impact, covering and dug zones

For a given state vector $\mathbf{x}$ and a measurement z, we define the following sets

covering zone	$\delta_{\mathbf{x}}^{-1}([0,\infty[) = \{\mathbf{a}, \delta_{\mathbf{x}}(\mathbf{a}) < \infty\}$
impact zone	$\delta_{\mathbf{x}}^{-1}(\{z\}) = \{\mathbf{a}, \delta_{\mathbf{x}}(\mathbf{a}) = z\}$
dug zone	$\delta_{\mathbf{x}}^{-1}([0,z[) = \{\mathbf{a}, \delta_{\mathbf{x}}(\mathbf{a}) < z\}.$

(8.4)

Figure 8.1 represents a robot equipped with a rangefinder that can only detect points of the map inside an ellipse. The white ellipse on the left figure corresponds to $\delta_{\mathbf{x}}^{-1}([0,\infty[)$. The thick arc on the right figure corresponds to the set $\delta_{\mathbf{x}}^{-1}(\{z\})$, where $z = d(\mathbf{x},\mathbb{M})$ and $\mathbb{M} = \mathbb{M}_1 \cup \mathbb{M}_2$. The part of the ellipse painted dark gray corresponds to $\delta_{\mathbf{x}}^{-1}([0,z[)$. The two following theorems will be used by our interval method to solve the range-only SLAM problem.

Theorem 1 (dug zone). Denote by $\mathbf{x}$ the state vector of the robot at a given time t. We have,

$$z = d(\mathbf{x},\mathbb{M}) \Rightarrow \delta_{\mathbf{x}}^{-1}([0,z[) \cap \mathbb{M} = \emptyset. \tag{8.5}$$

Proof. The proof is by contradiction. Assume that (i) $z = d(\mathbf{x},\mathbb{M})$, (ii) $\mathbf{a} \in \delta_{\mathbf{x}}^{-1}([0,z[)$ and (iii) $\mathbf{a} \in \mathbb{M}$. We have $\mathbf{a} \in \delta_{\mathbf{x}}^{-1}([0,z[) \Leftrightarrow \delta_{\mathbf{x}}(\mathbf{a}) \in [0,z[\overset{(8.3)}{\Leftrightarrow} d(\mathbf{x},\{\mathbf{a}\}) < z \overset{\text{(i)}}{\Leftrightarrow} d(\mathbf{x},\{\mathbf{a}\}) < d(\mathbf{x},\mathbb{M})$. Now, since $\mathbf{a} \in \mathbb{M}$, we have $\{\mathbf{a}\} \subset \mathbb{M}$, and thus $d(\mathbf{x},\{\mathbf{a}\}) \geq \min\{d(\mathbf{x},\{\mathbf{a}\}), d(\mathbf{x},\mathbb{M})\} \overset{(8.2)}{=} d(\mathbf{x},\mathbb{M}\cup\{\mathbf{a}\}) \overset{\text{(iii)}}{=} d(\mathbf{x},\mathbb{M})$. We have thus proved $d(\mathbf{x},\{\mathbf{a}\}) < d(\mathbf{x},\mathbb{M})$ and that $d(\mathbf{x},\{\mathbf{a}\}) \geq d(\mathbf{x},\mathbb{M})$. ■

The set $\mathbb{D} = \bigcup_{t\in[t]} \delta_{\mathbf{x}(t)}^{-1}([0,z(t)[)$ is called the *dug space*. A direct consequence of Theorem 1 is that $\mathbb{D}$ is inside the free space and thus does not intersect the map.

Theorem 2 (impact zone). For all $\mathbf{x}$, the impact zone intersects the map, i.e,

$$z = d(\mathbf{x},\mathbb{M}) \Rightarrow \delta_{\mathbf{x}}^{-1}(\{z\}) \cap \mathbb{M} \neq \emptyset.$$

Proof. We have $z = d(\mathbf{x},\mathbb{M}) \overset{(8.2)}{=} \min_{\mathbf{m}\in\mathbb{M}} d(\mathbf{x},\{\mathbf{m}\})$ (recall that $\mathbb{M}$ is compact). Denote by $\mathbf{a}$ one minimizer of $d(\mathbf{x},\{\mathbf{m}\})$ over $\mathbb{M}$. We have $z = d(\mathbf{x},\{\mathbf{a}\})$ and thus $\mathbf{a} \in \delta_{\mathbf{x}}^{-1}(\{z\})$. Therefore, $\mathbf{a}$ belongs to both $\delta_{\mathbf{x}}^{-1}(\{z\})$ and $\mathbb{M}$. ■

Figure 8.2 illustrates the principle of the resolution method that will be proposed to solve our SLAM problem. The robot poses painted white represent the actual poses of the robot at times t_1, t_2 with the

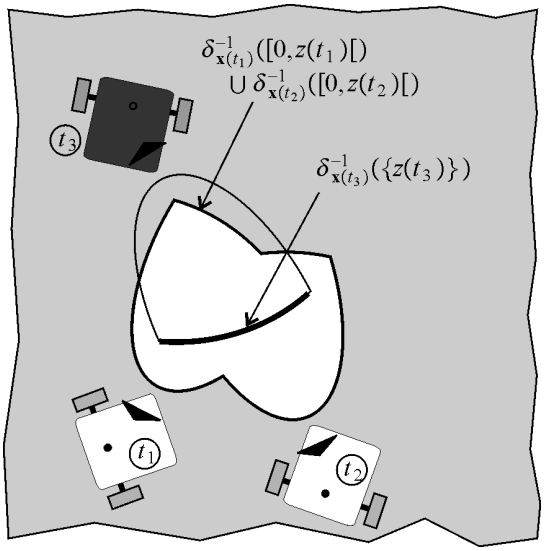


Figure 8.2: An inconsistent situation where the impact zone at time t_3 is enclosed inside the zones dug at times t_1 and t_2

corresponding dug zones. From Theorem 1, we have

$$\left(\delta^{-1}_{\mathbf{x}(t_1)}\left([0,z\left(t_1\right)[\right) \cup \delta^{-1}_{\mathbf{x}(t_2)}\left([0,z\left(t_2\right)[\right)\right) \cap \mathbb{M} = \emptyset$$

which provides an outer approximation $\mathbb{M}^+$ of $\mathbb{M}$ represented by the light gray zone which covers a large part of the workspace. Since

$$\delta^{-1}_{\mathbf{x}(t_3)}\left(\{z\left(t_3\right)\}\right) \subset \delta^{-1}_{\mathbf{x}(t_1)}\left([0,z\left(t_1\right)[\right) \cup \delta^{-1}_{\mathbf{x}(t_2)}\left([0,z\left(t_2\right)[\right),$$

we have $\delta^{-1}_{\mathbf{x}(t_3)}\left(\{z\left(t_3\right)\}\right) \cap \mathbb{M} = \emptyset$ and thus, from Theorem 2 we conclude that the dark gray pose at time t_3 is inconsistent. Theorem 1 will be used for the map building (since it is able to find an outer approximation $\mathbb{M}^+$ of $\mathbb{M}$) whereas Theorem 2 will be used for the localization (since it is able to remove inconsistent poses). This is the main idea of the contraction approach proposed in the next sections.

8.3 Hybrid constraint propagation

Constraint propagation is a numerical method to solve nonlinear problems. In the literature, the unknown variables are Boolean numbers, integers or real numbers. This section explains its principle and extends the classical technique in order to allow solving more general problems (such as those where the variables are functions or sets). This extension is necessary to solve our SLAM problem.

8.3.1 Lattices

A *lattice* $(\mathcal{E}, \leq)$ is a partially ordered set, closed under least upper and greatest lower bounds (see [13], for more details). The least upper bound (or infimum) of x and y is called the *join* and is denoted by $x \vee y$. The greatest lower bound (or *supremum*) is called the *meet* 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

$$\mathbf{x} \leq \mathbf{y} \Leftrightarrow \forall i \in \{1, \ldots, n\}, x_i \leq y_i. \tag{8.6}$$

We have

$$\begin{aligned} \mathbf{x} \wedge \mathbf{y} &= (x_1 \wedge y_1, \ldots, x_n \wedge y_n) \text{ and} \\ \mathbf{x} \vee \mathbf{y} &= (x_1 \vee y_1, \ldots, x_n \vee y_n) \end{aligned} \tag{8.7}$$

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 of infinite) subsets $\mathcal{A}$ of $\mathcal{E}$, the least upper bound (denoted $\wedge\mathcal{A}$) and the greatest lower bound (denoted $\vee\mathcal{A}$) belong to $\mathcal{A}$. When a lattice $\mathcal{E}$ is not complete, it is possible to add new elements (corresponding the supremum or infimum of infinite subsets of $\mathcal{E}$ that do not belong to $\mathcal{E}$) to make it complete. For instance, the set $\mathbb{R}$ is not a complete sub-lattice whereas $\overline{\mathbb{R}} = \mathbb{R} \cup \{-\infty, \infty\}$ is. By convention, for the empty set, we set $\wedge\emptyset = \vee\mathcal{E}$ and $\vee\emptyset = \wedge\mathcal{E}$. The product of two lattices $(\mathcal{E}_1, \leq_1)$ and $(\mathcal{E}_2, \leq_2)$ is the lattice $(\mathcal{E}, \leq)$ defined as the set of all $(a_1, a_2) \in \mathcal{E}_1 \times \mathcal{E}_2$ with the order relation

$$(a_1, a_2) \leq (b_1, b_2) \Leftrightarrow ((a_1 \leq_1 b_1) \text{ and } (a_2 \leq_2 b_2)). \tag{8.8}$$

8.3.2 Intervals

A *closed interval* (or *interval* for short) $[x]$ of a complete lattice $\mathcal{E}$ is a subset of $\mathcal{E}$ which satisfies $[x] = \{x \in \mathcal{E} \mid \wedge[x] \leq x \leq \vee[x]\}$. Both $\emptyset$ and $\mathcal{E}$ are intervals of $\mathcal{E}$. An interval is a sub-lattice of $\mathcal{E}$. An interval $[x]$ of $\mathcal{E}$ will also be denoted by $[x] = [\wedge[x], \vee[x]]_{\mathcal{E}}$. For example, the sets $\emptyset = [\infty, -\infty]_{\mathbb{R}}$; $\mathbb{R} = [-\infty, \infty]_{\mathbb{R}}$; $[0, 1]_{\mathbb{R}}$ and $[0, \infty]_{\mathbb{R}}$ are intervals of $\mathbb{R}$, the set $\{2, 3, 4, 5\} = [2, 5]_{\mathbb{N}}$ is an interval of the set of integers $\mathbb{N}$ and the set $\{4, 6, 8, 10\} = [4, 10]_{2\mathbb{N}}$ is an interval of $2\mathbb{N}$. We shall now introduce the notions of tubes and set intervals that will be used for solving the range-only SLAM problem.

Tubes. The set $\mathcal{F}$ of all functions from $\mathbb{R}$ to $\bar{\mathbb{R}}^n$ is a complete lattice with the following partial order $\mathbf{f} \leq \mathbf{g} \Leftrightarrow \forall t \in \mathbb{R}, \ \mathbf{f}(t) \leq \mathbf{g}(t)$. An interval of $\mathcal{F}$ is called a *tube* [52], [61].

Set intervals. The set $\mathcal{P}(\mathbb{R}^n)$ of all subsets of $\mathbb{R}^n$ is a complete lattice with respect to the inclusion $\subset$.

Figure 8.3 illustrates the notions of tubes and set intervals. On the left subfigure, the function f is bracketed by two stair functions f^-, f^+. In the computer, the interval $[f^-, f^+]$ is represented as a list of boxes. On the right subfigure, the uncertain set $\mathbb{X}$ is approximated by the set interval $[\mathbb{X}^-, \mathbb{X}^+]$, where $\mathbb{X}^-$ is the union of black boxes and $\mathbb{X}^+$ is the union of black and white boxes. The two bounds $\mathbb{X}^-, \mathbb{X}^+$ of the interval $[\mathbb{X}^-, \mathbb{X}^+]$ are represented in the computer as union of boxes (or subpavings [77]).

8.3.3 Contractors

Many problems of estimation, control, robotics, ... can be represented by continuous *constraint satisfaction problems* (CSP) [46]. A CSP [83] is composed of a set of variables $\mathcal{V} = \{x_1, \ldots, x_n\}$, a set of constraints $\mathcal{C} = \{c_1, \ldots, c_m\}$ and a set of interval domains $\{[x_1], \ldots, [x_n]\}$. Each variable x_i should belong to a complete lattice $(\mathcal{E}_i, \leq_i)$. When the lattices $\mathcal{E}_i$ have a different nature the CSP is said to be *hybrid*. The Cartesian product of all domains of a hybrid CSP is called a hybrid box. In the context

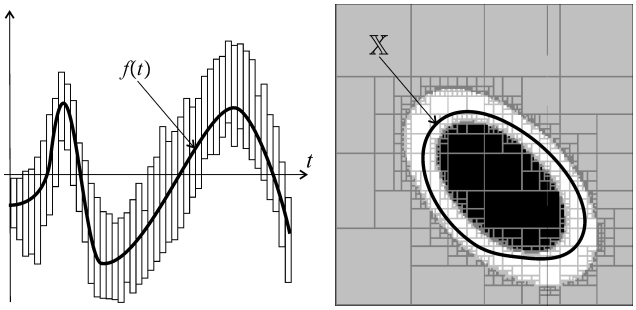


Figure 8.3: An interval function (or tube) and a set interval

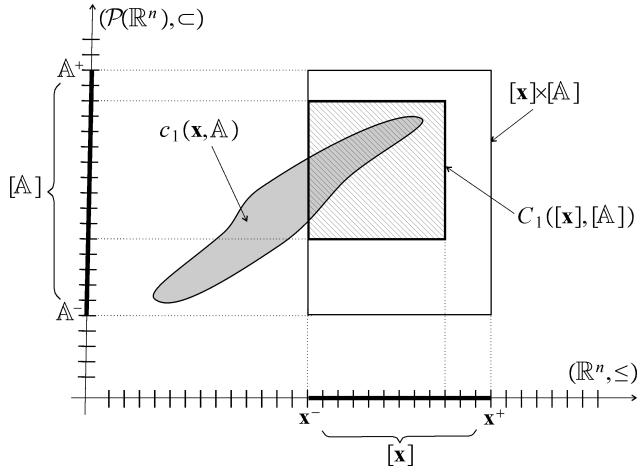


Figure 8.4: Hybrid contractor $\mathcal{C}_1$ associated with the hybrid constraint $c_1(\mathbf{x}, \mathbb{A})$. The hybrid box $[\mathbf{x}] \times \mathbb{A}$ is contracted into the hatched box.

of our chapter, hybrid CSP will be considered and the x_i's will be real numbers, vectors, subsets of $\mathbb{R}^n$ or trajectories (i.e. functions of time). Propagation techniques contract as much as possible the interval domains for the variables without loosing any solution [82, 12]. They have been shown to be efficient in several robotic applications such as localization [59], state estimation [29] or parametric SLAM [39]. Denote by $[\mathbf{x}]$ the Cartesian product of all domains $[x_i]$. A *contractor* associated with the constraint c_i is an operator C_i such that

$$\begin{array}{ll} (c_i \cap [\mathbf{x}]) \subset C_i([\mathbf{x}]) & \text{(completeness)} \\ C_i([\mathbf{x}]) \subset [\mathbf{x}] & \text{(contractance)} \end{array} \tag{8.9}$$

Figure 8.4 shows a hybrid contractor associated with the hybrid constraint $c_1(\mathbf{x}, \mathbb{A})$. The x-axis corresponds to the lattice $\mathbb{R}^n$, whereas the y-axis corresponds $\mathcal{P}(\mathbb{R}^n)$ the set of subsets of $\mathbb{R}^n$. The discretization illustrates that only a finite number of elements of $\mathbb{R}^n$ and $\mathcal{P}(\mathbb{R}^n)$ can be represented by the computer. More precisely, these machine numbers are floating point vectors for $\mathbb{R}^n$ and subpavings (i.e., union of boxes with floating point vectors as vertices). The hybrid box $[\mathbf{x}] \times [\mathbb{A}]$ has two components, namely the box $[\mathbf{x}]$ and the set interval $[\mathbb{A}]$. The bounds of $[\mathbf{x}]$ are $\mathbf{x}^-, \mathbf{x}^+$ whereas the bounds of $[\mathbb{A}]$ are the sets $\mathbb{A}^-, \mathbb{A}^+$. Since $(c_1(\mathbf{x}, \mathbb{A}) \cap ([\mathbf{x}] \times [\mathbb{A}])) \subset C_1([\mathbf{x}], [\mathbb{A}])$ and $C_1([\mathbf{x}], [\mathbb{A}]) \subset [\mathbf{x}] \times [\mathbb{A}]$, the completeness and the contractance properties are satisfied.

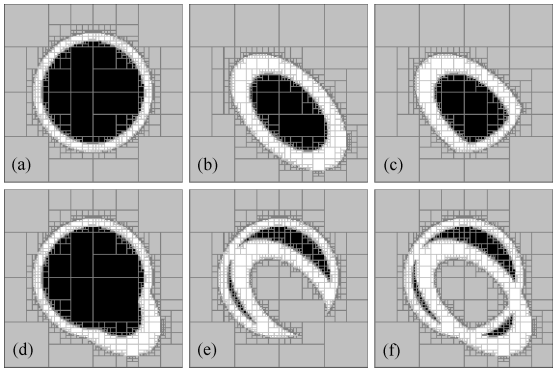


Figure 8.5: Computating with set intervals; (a): $[\mathbb{A}]$, (b) $[\mathbb{B}]$, (c): $[\mathbb{A}] \cap [\mathbb{B}]$, (d) $[\mathbb{A}] \cup [\mathbb{B}]$, (e) $[\mathbb{A}] \setminus [\mathbb{B}]$, (f) $([\mathbb{A}] \cup [\mathbb{B}]) \setminus ([\mathbb{A}] \cap [\mathbb{B}])$

For solving hybrid CSP, the principle of the *propagation* is to contract all hybrid boxes $[\mathbf{x}]$ by calling all available contractors $C_1, \ldots, C_m$ until a fixed point is reached.

8.4 Resolution

The implementation of the contractors described on the previous section requires an implementation of an arithmetic for set intervals [41]. This arithmetic makes it possible to handle easily uncertain sets (such as the map for our SLAM problem) as illustrated by Figure 8.5 (for the graphical representation of a set interval $[\mathbb{A}] = [\mathbb{A}^-, \mathbb{A}^+]$, the black boxes are inside $\mathbb{A}^-$, the grey boxes are outside $\mathbb{A}^+$ and the white boxes are inside $\mathbb{A}^+$ and outside $\mathbb{A}^-$). If the two sets $\mathbb{A}, \mathbb{B}$ belong to the set intervals $[\mathbb{A}]$,$[\mathbb{B}]$ (see Subfigures (a),(b)), then the sets $\mathbb{A} \cap \mathbb{B}$, $\mathbb{A} \cup \mathbb{B}$, $\mathbb{A} \backslash \mathbb{B}$, $(\mathbb{A} \cup \mathbb{B}) \setminus (\mathbb{A} \cap \mathbb{B})$ will belong to the set intervals of Subfigures (c), (d), (e), (f), respectively. The set interval arithmetic can be used to contract set intervals with respect to some constraints. Consider for example the constraint $\mathbb{A} \cap \mathbb{B} = \emptyset$ between the two sets $\mathbb{A}, \mathbb{B}$ and assume that $\mathbb{A} \in [\mathbb{A}], \mathbb{B} \in [\mathbb{B}]$. The contractions of $[\mathbb{A}], [\mathbb{B}]$ are given by the following operations: $[\mathbb{A}] := [\mathbb{A}^-, \mathbb{A}^+ \backslash \mathbb{B}^-]$ and $[\mathbb{B}] := [\mathbb{B}^-, \mathbb{B}^+ \backslash \mathbb{A}^-]$.

Our range-only SLAM problem can be cast into a hybrid CSP. The unknown variables are the trajectory $\mathbf{x}(t)$, the map $\mathbb{M}$ and the dug space $\mathbb{D}$. Since they have a different nature, the resulting CSP is hybrid. The constraints of the hybrid CSP are

$$\left.\begin{array}{ll} (1) & \dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)) \\ (2) & \mathbb{D} = \bigcup_{t \in [t]} \delta_{\mathbf{x}(t)}^{-1}([0, z(t)[) \\ (3) & \mathbb{D} \cap \mathbb{M} = \emptyset \\ (4) & \delta_{\mathbf{x}(t)}^{-1}(\{z(t)\}) \cap \mathbb{M} \neq \emptyset. \end{array}\right\} : z(t) = d(\mathbf{x}(t), \mathbb{M})$$

Constraints (2), (3), (4) correspond to a decomposition of the map constraint $z(t) = d(\mathbf{x}(t), \mathbb{M})$. This decomposition, which is a consequence of Theorems 1 and 2, is necessary to build the corresponding contractors. The prior domains for the set variables $\mathbb{M}$ and $\mathbb{D}$ are the set intervals $[\mathbb{M}] = [\mathbb{D}] = [\emptyset, \mathbb{R}^q]$ which enclose all subsets of $\mathbb{R}^q$. It translates the fact that no prior information on the map and the dug space are available. The prior domain for the trajectory $\mathbf{x}(t)$ is a tube $[\mathbf{x}](t)$. For the SLAM problem to be considered later, we have $[\mathbf{x}](t) = \mathbb{R}^n$ for $t > 0$ and $[\mathbf{x}](0)$ will be a singleton, which means that the

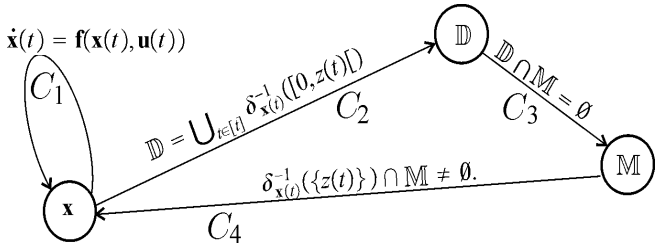


Figure 8.6: Constraint diagram of the SLAM problem

initial state $\mathbf{x}(0)$ is known without any error.

Our CSP is composed of four constraints, the diagram of which is depicted on Figure 8.6. To each constraint, we have to build a contractor. The first one $\mathcal{C}_1([\mathbf{x}](t))$ contracts the tube $[\mathbf{x}](t)$ with respect to the evolution equation $\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t))$. Recall that a bounded tube $[\mathbf{u}](t)$ for $\mathbf{u}(t)$ is assumed to be known. The tube $[\mathbf{x}](t)$ can be contracted without loosing any feasible value by an interval integration [16, 65, 36] using a forward and a backward propagation. The hybrid contractor $\mathcal{C}_2([\mathbb{D}], [\mathbf{x}](t))$ is related to the hybrid constraint $\mathbb{D} = \bigcup_{t\in[t]} \delta^{-1}_{\mathbf{x}(t)}([0, z(t)[)$. It makes possible to contract the set interval $[\mathbb{D}]$. The contractor $\mathcal{C}_3([\mathbb{D}], [\mathbb{M}])$ associated with the constraint $\mathbb{D} \cap \mathbb{M} = \emptyset$ yields contractions of the set interval $[\mathbb{M}]$ (see [41] for more explanations about this contractor). The hybrid contractor $\mathcal{C}_4([\mathbb{M}], [\mathbf{x}](t))$ associated with $\delta^{-1}_{\mathbf{x}(t)}(\{z(t)\}) \cap \mathbb{M} \neq \emptyset$ provides contractions for the tube $[\mathbf{x}]$ (see [44] for more about this contractor).

The resulting propagation algorithm is given on the table below.

	CONTRACT(inout: $[\mathbf{x}](t)$; out: $[\mathbb{D}], [\mathbb{M}]$)
1	$[\mathbb{M}] := [\emptyset, \mathbb{R}^q]$; $[\mathbb{D}] = [\emptyset, \mathbb{R}^q]$;
2	Repeat
3	$[\mathbf{x}](t) := \mathcal{C}_1([\mathbf{x}](t))$;
4	$[\mathbb{D}] := \mathcal{C}_2([\mathbb{D}], [\mathbf{x}](t))$;
5	$[\mathbb{M}] := \mathcal{C}_3([\mathbb{D}], [\mathbb{M}])$;
6	$[\mathbf{x}](t) := \mathcal{C}_4([\mathbb{M}], [\mathbf{x}](t))$;
7	Until no more contraction.

For a given precision, the complexity of contractor-based propagation methods is polynomial if all contractors have a polynomial complexity. Now, some of the contractors that are used have a complexity which is exponential with respect to q, the dimension of the map (which is equal to 2 or 3). Now, since q can be considered as a fixed parameter, the complexity of the method is polynomial with respect to all other parameters of the problem (size of the world, time of the mission, ...).

8.5 Testcases

In order to illustrate the behavior of the algorithm presented in the previous section, consider a mobile robot described by the following range-only SLAM equations

$$\begin{cases} \dot{x}_1(t) &= u_1(t)\cos(u_2(t)) \\ \dot{x}_2(t) &= u_1(t)\sin(u_2(t)) \\ z(t) &= d(\mathbf{x}(t), \mathbb{M}). \end{cases} \tag{8.10}$$

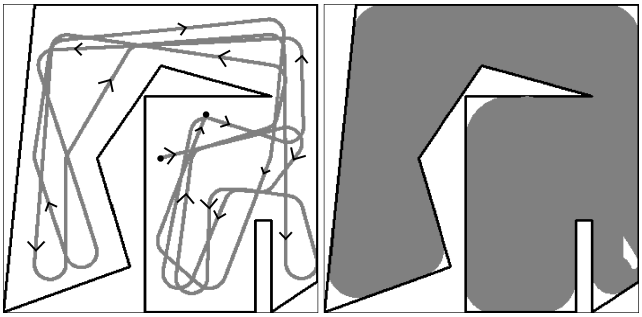


Figure 8.7: Left: actual trajectory of the robot. Right: corresponding dug space. The frame box corresponds to $[-10,10]\times[-10,10]$.

The inputs of the system are the speed u_1 and the heading u_2 of the robot. The measurement z corresponds to the closest distance of the robot to the map which could have been obtained by using a omnidirectional (with angular aperture of 2π) sonar. The quantities u_1, u_2, z are measured every 0.1 sec with an error of 0.01 ms^{-1}, 0.01 rads^{-1}, 0.01m, respectively. The initial state, taken as $\mathbf{x}=(0,0)^{\mathrm{T}}$, is assumed to be known.

Testcase 1. Figure 8.7 provides a simulation of the robot moving inside an unknown map. As shown by the figure, the map is composed by segments but this is not required by the method. The shape of the map could be arbitrary and no parametric representation of the map is needed. The gray zone in the right part of the figure represents the unknown dug space $\mathbb{D}$. It means that, in the ideal situation where the trajectory $\mathbf{x}(t)$ is known exactly, the map $\mathbb{M}$ can be approximated by $\mathbb{D}$ (in the sense that $\mathbb{M}\cap\mathbb{D}=\emptyset$).

An illustration of the interval propagation method is depicted on Figure 8.8 which has been computed in about 15 minutes with a classical laptop. The left subfigures show the computed the tubes $[\mathbf{x}](t)$ (painted grey) with the true path (painted black) of the robot. As expected, the true path is always included inside the tube $[\mathbf{x}](t)$. The right subfigures correspond to inner approximations (painted grey) of the dug space $\mathbb{D}$. The segments of the true map are also represented to illustrate how accurate is the approximation of the map. The width w of the tubes $[\mathbf{x}](t)$ is given on Figure 8.9. After the first call to $\mathcal{C}_1([\mathbf{x}])$ (see Step 3), we get the tube of Figure (8.8,a). The error increases linearly as shown by Subfigure (8.9,a). After running all contractions, we get Subfigure (8.8,b) as an inner approximation of the dug space $\mathbb{D}$. After a second run of the loop, we get Subfigures (8.8,c) and (8.8,d). A third and fourth run yields Subfigures (8.8,e,f,g,h). The fixed point that is reached is depicted on Subfigures (8.8,i) and (8.8,j). As shown by Figure 8.9, the width of the tube $[\mathbf{x}]$ decreases. Oscillations with respect to t are due to the fact that on the right part of the room (which was first observed), the robot succeeds to have a more accurate localization than on the left part of the room. When the right part of the room was observed for the first time, the robot did not accumulate uncertainties in its localization and was thus able to get an accurate map. When the robot came back to the right part, it was then able to take advantage of the accurate mapping to improve its localization. This is consistent with the loop closure effect classically observed in a SLAM context. An educational and easy-to-use windows program associated with this testcase can be downloaded with all C++ codes at

`www.ensta-bretagne.fr/jaulin/dig.html`

Remark. Let us now give more details concerning the very beginning of the contraction procedure. At

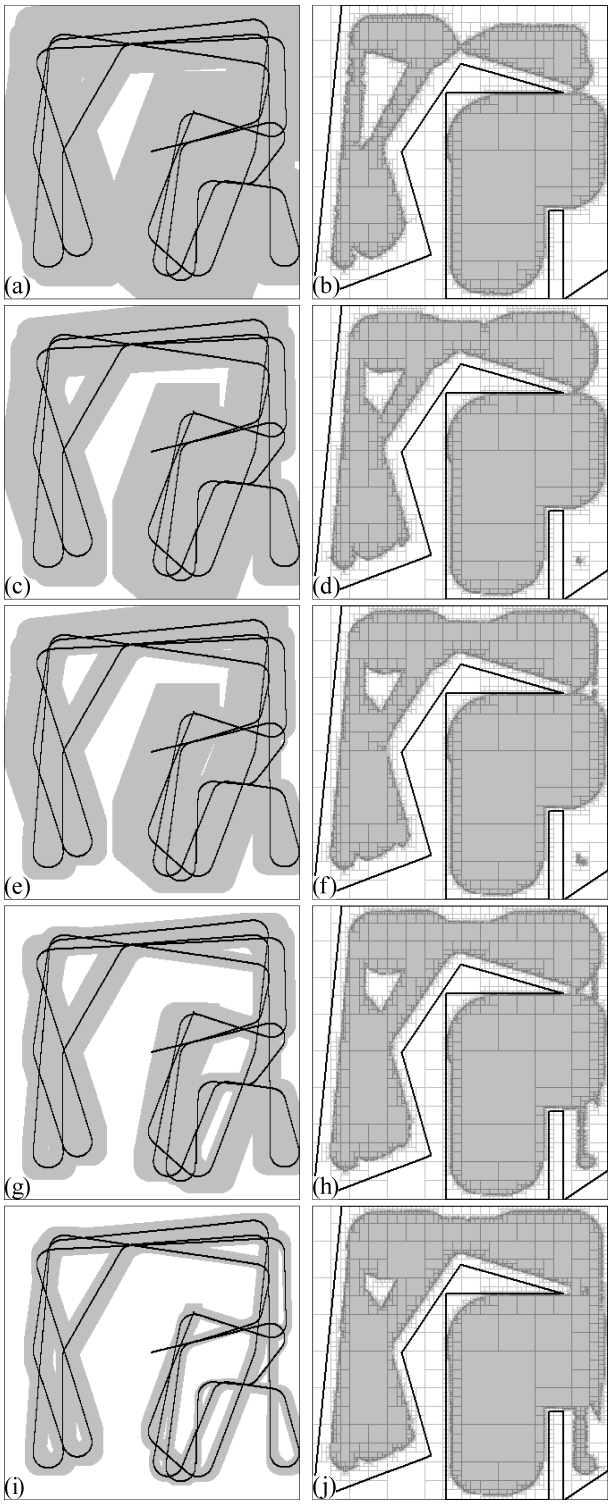


Figure 8.8: Left: Contraction of the tube $[\mathbf{x}](t)$ during the propagation. Right. Evolution of the approximation the dug space $\mathbb{D}$.

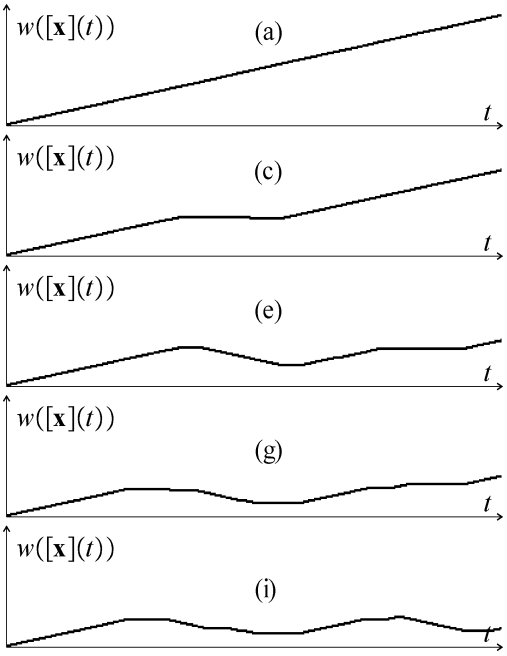


Figure 8.9: Width of the tubes $[\mathbf{x}](t)$ represented on the left part of Figure 8.8

the first step the contractor C_1 only contracts the tube $[\mathbf{x}](t)$ by propagating forward the knowledge of the initial condition $\mathbf{x}(0)$. The contractor C_2, associated to the constraint $\mathbb{D} = \bigcup_{t\in[t]} \delta^{-1}_{\mathbf{x}(t)}([0, z(t)[)$, contracts the set interval $[\mathbb{D}]$. The contractor C_4 associated to the constraint $\delta^{-1}_{\mathbf{x}(t)}(\{z(t)\}) \cap \mathbb{M} \neq \emptyset$ also contracts the tube $[\mathbf{x}](t)$. At this level, the tube $[\mathbf{x}](t)$, represented on Figure 8.10 (top left), appears to be the same as before the call to C_4 (see 8.8,a). This is due to the fact that only small parts of $[\mathbf{x}](t)$ have been contracted, and due to the superposition of all boxes, the contractions are not visible. The width $w([\mathbf{x}](t))$ of the tube is similar to the previous one (almost linear), except for some t (illustrated by the clouds inside the two ellipses at the bottom of the figure). On the subfigure at the top right, is represented the subtube of $[\mathbf{x}](t)$ contracted by C_4. The black zone corresponds to the part that has been removed by C_4 and the grey zone corresponds the subtube after contraction.

In the Figure 8.11, are represented the contraction of the subtubes for five other steps.

Testcase 2. Figure 8.12 represents the true trajectory, the map (segments and circles), the unknown dug space $\mathbb{D}$ (painted gray), on the right subfigure. The robot terminates its mission inside the triangle.

Figure 8.13 presents the principle of resolution. Is also represented the unknown the map (segments and circles) and the true trajectory to demonstrate the correctness of the results. At the bottom, the evolution of width of the tubes $[\mathbf{x}](t)$ is illustrated.

8.6 Conclusion

This chapter has presented a constraint propagation method to solve the SLAM problem in the case where the map cannot be represented by a parametric structure. In such a case, the map can be represented

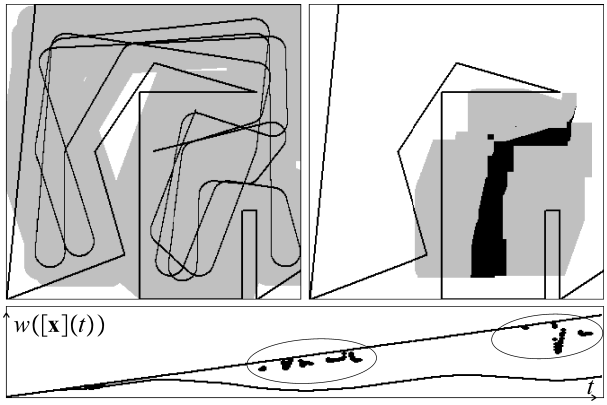


Figure 8.10: Illustration of the first step of the propagation procedure

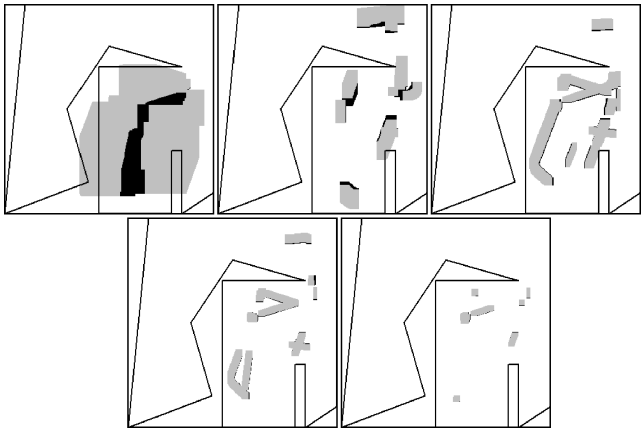


Figure 8.11: Contractions of the trajectory subtubes

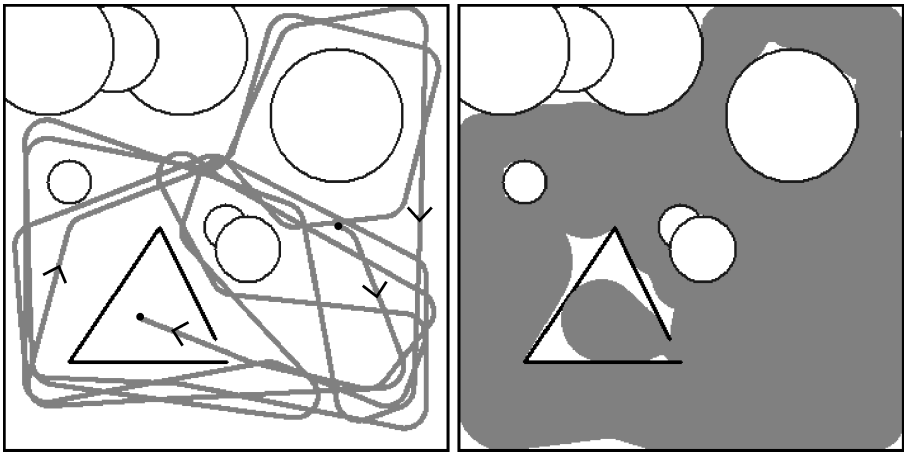


Figure 8.12: True trajectory and dug space

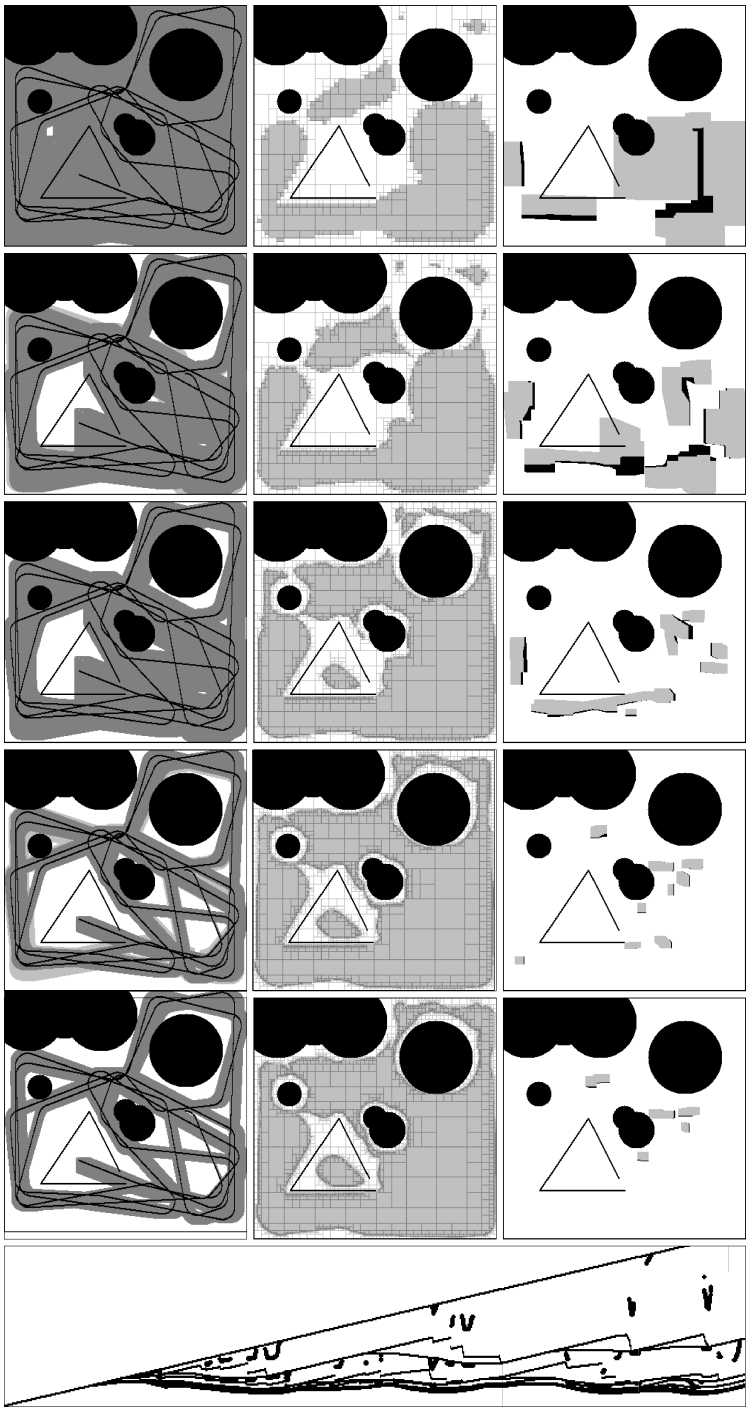


Figure 8.13: Illustration of the resolution associated to Testcase 2

as a set of an infinite number of punctual marks or equivalently by arbitrary compact subsets of $\mathbb{R}^q$. As a consequence, the SLAM problem encloses unknown variables that are sets of $\mathbb{R}^q$ and containing an uncountable number of elements. Solving this type of set-valued nonlinear problems with set-membership methods is not easy and atypical in the robotic or control community. An extension of existing constraint propagation methods has then be proposed in order to allow using contractors associated with set-valued variables. The principle of the resulting hybrid contractor approach for SLAM has been illustrated through a range-only offline SLAM testcase with simulated data.

However, even if the principle of solving the pose-based range-only SLAM in a reliable way has been demonstrated, the proposed technique exhibits some limitations: (1) When outliers occur during the mission, the trajectory tube quickly becomes an empty tube and no more estimation of the map and the trajectory can be produced anymore; (2) The approach cannot be easily extended to situations where moving obstacles exist in the environment; (3) A prior box enclosing the map and the trajectory is needed, which is not well suited for exploration; (4) It is computationally expensive to match representation of the space with subpavings; and (5) The method is only able to perform off-line SLAM, which is not suited for real-time applications. Further researches are therefore necessary to make the approach effective to solve online SLAM problems involving real robots.

Bibliography

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

[2] L. S. B. Rosgen. Complexity results on graphs with few cliques. *Discrete Mathematics and Theoretical Computer Science*, 2007.

[3] S. Bazeille. *Vision sous-marine monoculaire pour la reconnaissance d'objets.* PhD thesis, Université de Bretagne Occidentale, 2008.

[4] V. Bertram. *Practical Ship Hydrodynamics.* F. Vieweg & Sohn, Butterworth - Heinemann, 2000.

[5] A. Bovik. *Handbook of image and video processing.* Academic Press, 2000.

[6] Y. Brière. *The first microtransat challenge* , `http://web.ensica.fr/microtransat`. ENSICA, 2006.

[7] A. Caiti, A. Garulli, F. Livide, and D. Prattichizzo. Set-membership acoustic tracking of autonomous underwater vehicles. *Acta Acustica united with Acustica*, 5(88):648–652, 2002.

[8] J. Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol 8, No. 6*, 1986.

[9] J. Castellanos, J. Neira, and J. Tardós. Multisensor fusion for simultaneous localization and map building. *IEEE Transactions on Robotics and Automation*, 17(6):908–914, 2001.

[10] J. A. Castellanos and J. Tardós. Mobile robot localization and map building: A multisensor fusion approach. *Kluwer*, 1999.

[11] G. Chabert and L. Jaulin. *QUIMPER: QUick Interval Modeling and Programming in a bounded-ERror context, available at* , `http://www.ibex-lib.org/`. ENSIETA, 2008.

[12] G. Chabert and L. Jaulin. Contractor Programming. *Artificial Intelligence*, 173:1079–1100, 2009.

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

[14] A. Davison. Real-time simultaneous localisation and mapping with a single camera. In *International Conference on Computer Vision*, 2003.

[15] 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*, 351(1):119–128, 2006.

[16] 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.

[17] M. Di Marco, A. Garulli, A. Giannitrapani, and A. Vicino. A set theoretic approach to dynamic robot localization and mapping. *Autonomous Robots*, 16(1):23–47, 2004.

[18] M. Di Marco, A. Garulli, S. Lacroix, and A. Vicino. Set membership localization and mapping for autonomous navigation. *International Journal of Robust and Nonlinear Control*, 7(11):709–734, 2001.

[19] S. Diop and M. Fliess. Nonlinear observability, identifiability and persistent trajectories. In *Proc. 36th IEEE Conf. Decision Control*, pages 714–719, Brighton, 1991.

[20] G. Dissanayake, P. Newman, S. Clark, H. F. Durrant-Whyte, and M. Csorba. A solution to the simultaneous localization and map building (SLAM) problem. *IEEE Transactions Robotics and Automation*, 3(17):229–241, 2001.

[21] 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*, 3478:127–141, 2005.

[22] R. O. Duda and P. E. Hart. Use of the hough transform to detect lines and curves in pictures. *Comm. ACM Vol 15.*, 1972.

[23] A. Elfes. Sonar-based real world mapping and navigation. *IEEE Transactions on Robotics and Automation*, pages 249–265, 1987.

[24] R. Eustice, H. Singh, J. Leonard, M. Walter, and R. Ballard. Visually navigating the rms titanic with slam information filters. In *Proceedings of Robotics: Science and Systems (RSS)*, Cambridge, MA, USA, 2005.

[25] M. Fliess, J. Lévine, P. Martin, and P. Rouchon. Flatness and defect of non-linear systems: introductory theory and applications. *International Journal of Control*, (61):1327–1361, 1995.

[26] M. Fliess and H. Sira-Ramirez. Control via state estimations of some nonlinear systems. In *Proc. Symp. Nonlinear Control Systems (NOLCOS 2004)*, Stuttgart, 2004.

[27] U. Frese. A discussion of simultaneous and mapping. *Autonomous Robots*, 20:25–42, 2006.

[28] J. Gauthier and I. Kupka. *Deterministic observation theory and applications*. Cambridge University Press, 2001.

[29] A. Gning and P. Bonnifait. Constraints propagation techniques on intervals for a guaranteed localization using redundant data. *Automatica*, 42(7):1167–1175, 2006.

[30] V. Hagenmeyer and E. Delaleau. Robustness analysis of exact feedforward linearization based on differential flatness. *Automatica*, (39):1941–1946, 2003.

[31] E. Halbwachs and D. Meizel. Bounded-error estimation for mobile vehicule localization. *CESA'96 IMACS Multiconference (Symposium on Modelling, Analysis and Simulation)*, pages 1005–1010, 1996.

[32] E. R. Hansen. *Global Optimization using Interval Analysis*. Marcel Dekker, New York, NY, 1992.

[33] H. Hansen, P. Jackson, and K. Hochkirch. Real-time velocity prediction program for wind tunnel testing of sailing yachts. In *International Conference on the Modern Yacht*, Southampton, 2003.

[34] P. Herrero, L. Jaulin, J. Vehi, and M. A. Sainz. Guaranteed set-point computation with application to the control of a sailboat. *International Journal of Control Automation and Systems*, 8(1):1–7, 2010.

[35] A. Isidori. *Nonlinear Control Systems: An Introduction, 3rd Ed.* Springer-Verlag, New-York, 1995.

[36] L. Jaulin. Nonlinear bounded-error state estimation of continuous-time systems. *Automatica*, 38:1079–1082, 2002.

[37] L. Jaulin. *Représentation d'état pour la modélisation et la commande des systèmes (Coll. Automatique de base)*. Hermès, London, 2005.

[38] L. Jaulin. *solver GESMI (Global Estimation of SeaMarks with Intervals), available at* , `http://www.ensieta.fr/jaulin/gesmi.zip`. ENSIETA-GESMA, 2007.

[39] L. Jaulin. A Nonlinear Set-membership Approach for the Localization and Map Building of an Underwater Robot using Interval Constraint Propagation. *IEEE Transaction on Robotics*, 25(1):88–98, 2009.

[40] L. Jaulin. Robust set membership state estimation ; application to underwater robotics. *Automatica*, 45(1):202–206, 2009.

[41] L. Jaulin. Solving set-valued constraint satisfaction problems. In *SCAN 2010*, Lyon (France), 2010.

[42] L. Jaulin. Range-only SLAM with occupancy maps; A set-membership approach. *IEEE Transaction on Robotics*, 27(5):1004–1010, 2011.

[43] L. Jaulin and S. Bazeille. Image shape extraction using interval methods. In *Sysid 2009*, Saint Malo, France, 2009.

[44] L. Jaulin and G. Chabert. Resolution of nonlinear interval problems using symbolic interval arithmetic. *Engineering Applications of Artificial Intelligence*, 23(6):1035–1049, 2010.

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

[46] L. Jaulin, M. Kieffer, O. Didrit, and E. Walter. *Applied Interval Analysis, with Examples in Parameter and State Estimation, Robust Control and Robotics*. Springer-Verlag, London, 2001.

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

[48] L. Jaulin and E. Walter. Guaranteed bounded-error parameter estimation for nonlinear models with uncertain experimental factors. *Automatica*, 35(5):849–856, 1999.

[49] L. Jaulin and E. Walter. Guaranteed robust nonlinear minimax estimation. *IEEE Transaction on Automatic Control*, 47(11):1857–1864, 2002.

[50] M. Kieffer, L. Jaulin, E. Walter, and D. Meizel. Robust autonomous robot localization using interval analysis. *Reliable Computing*, 6(3):337–362, 2000.

[51] C. Kimme, D. H. Ballard, and J. Sklansky. Finding circles by an array of accumulators. *CACM*, 1975.

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

[53] S. Lagrange, L. Jaulin, V. Vigneron, and C. Jutten. Nonlinear blind parameter estimation. *IEEE TAC*, 53(4):834–838, 2008.

[54] H. Lahanier, E. Walter, and R. Gomeni. OMNE: a new robust membership-set estimator for the parameters of nonlinear models. *Journal of Pharmacokinetics and Biopharmaceutics*, 15:203–219, 1987.

[55] J. J. Leonard and H. F. Durrant-Whyte. *Directed Sonar Sensing for Mobile Robot Navigation*. Kluwer, Boston, 1992.

[56] J. J. Leonard and H. F. Durrant-Whyte. Dynamic map building for an autonomous mobile robot. *International Journal of Robotics Research*, 11(4), 1992.

[57] F. Lydoire and P. Poignet. Nonlinear predictive control using constraint satisfaction. In *In 2nd International Workshop on Global Constrained Optimization and Constraint Satisfaction (COCOS)*, pages 179–188, 2003.

[58] M. Mboup, C. Join, and M. Fliess. A revised look at numerical differentiation with an application to nonlinear feedback control. In *The 15th Mediterrean Conference on Control and Automation - MED'2007*, 2007.

[59] D. Meizel, O. Lévêque, L. Jaulin, and E. Walter. Initial localization by set inversion. *IEEE transactions on robotics and Automation*, 18(6):966–971, 2002.

[60] D. Meizel, A. Preciado-Ruiz, and E. Halbwachs. Estimation of mobile robot localization: geometric approaches. In M. Milanese, J. Norton, H. Piet-Lahanier, and E. Walter, editors, *Bounding Approaches to System Identification*, pages 463–489. Plenum Press, New York, NY, 1996.

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

[62] D. K. Montemerlo, S. Thrun and B. Wegbreit. Fastslam 2.0: An improved particle filtering algorithm for simultaneous localization and mapping that provably converges. *Proc. of the International Joint Conference on Artificial Intelligence (IJCAI)*, 2003.

[63] R. E. Moore. *Interval Analysis*. Prentice-Hall, Englewood Cliffs, NJ, 1966.

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

[65] N. Nedialkov, K. Jackson, and G. Corliss. Validated solutions of initial value problems for ordinary differential equations. *Applied Mathematics and Computation*, 105(1):21–68, 1999.

[66] A. Neumaier. *Interval Methods for Systems of Equations*. Cambridge University Press, Cambridge, UK, 1990.

[67] P. Newman, J. Leonard, J. Tardós, and J. Neira. Explore and return: Experimental validation of real-time concurrent mapping and localization. In *ICRA02*, pages 1802–1809, Washington DC, 2002.

[68] J. Norton and S. Verez. Outliers in bound-based state estimation and identification. *Circuits and Systems*, 1:790–793, 1993.

[69] F. O'Gorman and M. Clowes. Finding Picture Edges Through Collinearity of Feature Points. *IEEE Transactions on Computers*, 25(4):449–456, 1976.

[70] J. Porta. Cuikslam: A kinematics-based approach to slam. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, pages 2436–2442, Barcelona (Spain), 2005.

[71] L. Pronzato and E. Walter. Robustness to outliers of bounded-error estimators and consequences on experiment design. In M. Milanese, J. Norton, H. Piet-Lahanier, and E. Walter, editors, *Bounding Approaches to System Identification*, pages 199–212, New York, 1996. Plenum.

[72] T. Raissi, N. Ramdani, and Y. Candau. Set membership state and parameter estimation for systems described by nonlinear differential equations. *Automatica*, 40:1771–1777, 2004.

[73] S. Reed, Y. Petillot, and J. Bell. An automatic approach to the detection and extraction of mine features in sidescan sonar. *IEEE Journal of Oceanic Engineering*, 28(1):90–105, 2003.

[74] R.Smith, M. Self, and P. Cheeseman. *Autonomous robot vehicles*, volume 8, chapter Estimating uncertain spatial relationships in robotics, pages 167–193. New York, 1990.

[75] I. Ruiz, S. de Raucourt, Y. Petillot, and D. Lane. Concurent mapping and localization using sidescan sonar. *IEEE Journal of Oceonic Engineering*, 39(2):442–456, 2004.

[76] J. C. Russ. *Image processing handbook (The)*. CRC Press, 2002.

[77] D. Sam-Haroud. *Constraint consistency techniques for continuous domains*. PhD dissertation 1423, Swiss Federal Institute of Technology in Lausanne, Switzerland, 1995.

[78] C. Sauze and M. Neal. An autonomous sailing robot for ocean observation. In *proceedings of TAROS 2006*, pages 190–197, Guildford, UK, 2006.

[79] D. Scott. Continuous lattices. In F. L. (ed.), editor, *Toposes, algebraic geometry and logic*, pages 97–136. Springer, 1972.

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

[81] S. Thrun and M. Montemerlo. The GraphSLAM algorithm with applications to large-scale mapping of urban structures. *International Journal on Robotics Research*, 25(5/6):403–430, 2005.

[82] M. van Emden. Algorithmic power from declarative use of redundant constraints. *Constraints*, 4(4):363–381, 1999.

[83] P. van Hentenryck, L. Michel, and Y. Deville. *Numerica - A Modelling Language for Global Optimization*. MIT Press, Cambridge, Massachusetts, 1997.

[84] P. H. Vinas, M. A. Sainz, J. Vehi, and L. Jaulin. Quantified set inversion algorithm with applications to control. *Reliable computing*, 11(5):369–382, 2006.

[85] E. Walter and L. Pronzato. *Identification of Parametric Models from Experimental Data*. Springer-Verlag, London, UK, 1997.

[86] S. Williams, G. Dissanayake, and H. Durrant-Whyte. Towards terrain-aided navigation for underwater robotics. *Advanced Robotics*, 15(5):533–549, 2001.