1

Static Set-membership State Estimation

1.1. Introduction

A state estimation problem can be dealt with by set-membership approaches, where both the uncertainties on the model and the measurements errors are known. By *known*, we mean that an unknown actual value is guaranteed to lie within bounds that delineate the uncertainties, thus defining a *solution set* (Walter and Piet-Lahanier 1988, Cerone 1996, Veres and Norton 1996, Maksarov and Norton 1996).

The estimation then consists of reducing this set of feasible values by means of operators, where other approaches would have minimized an error criterion. The computations are not performed in a probabilistic way but based on deterministic operations on the bounds of the set. This approach is significantly different from usual methods: here, we do not assume a probability distribution in the calculations. Furthermore, the bounding property of such an approach is guaranteed even though the system is nonlinear. This quality is of particular importance in mobile robotics where several problems present nonlinearities, the case of a range-only localization being one of them (Caiti *et al*. 2005).

Range-only state estimation

A telling example of set-membership state estimation is the localization of a mobile robot among beacons. This kind of application has already been appropriately presented in Drevelle (2011) to introduce set-membership methods. We will use it as a guiding thread of this book, first in this chapter in

a static context without considering temporal evolutions: the pose of the robot is estimated with synchronous measurements. The same example will be extended to dynamical state estimation in Chapters 2, 3 and 4 based on differential state equations and asynchronous measurements, sometimes obtained with time uncertainties.

A robot $\mathcal{R}$ is described by its state $\mathbf{x} = (x_1, x_2)^\intercal$, representing its 2D location. The robot evolves among a set of beacons $\mathcal{B}_k$ located at (x_1^k, x_2^k) and sending synchronously acoustic signals, as illustrated in Figure 1.1. The distances ρ_k from the $\mathcal{B}_k$ will be used to localize $\mathcal{R}$. The following observation function is used to this end:

$$\rho_k = g_k(\mathbf{x}) = \sqrt{\left(x_1 - x_1^k\right)^2 + \left(x_2 - x_2^k\right)^2}. \qquad [1.1]$$

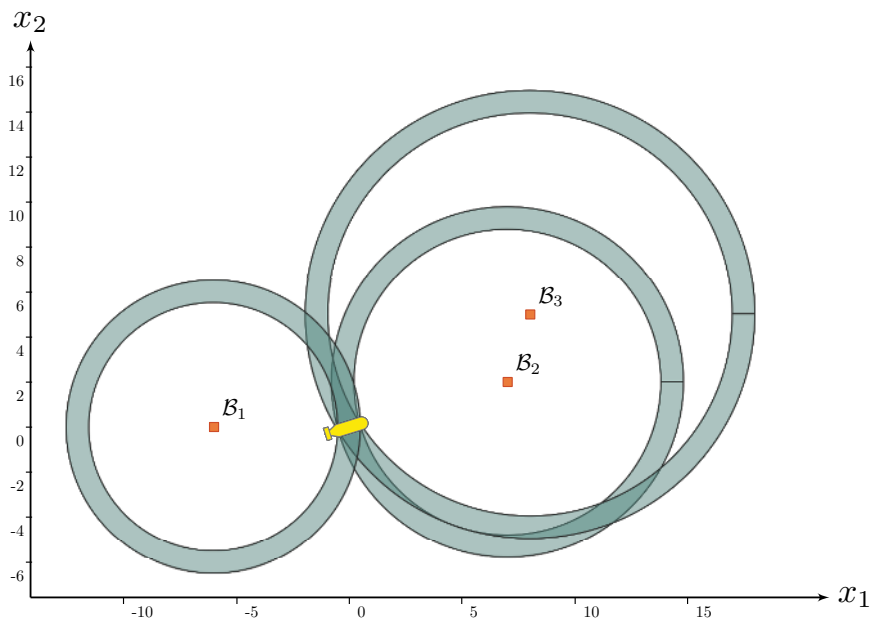
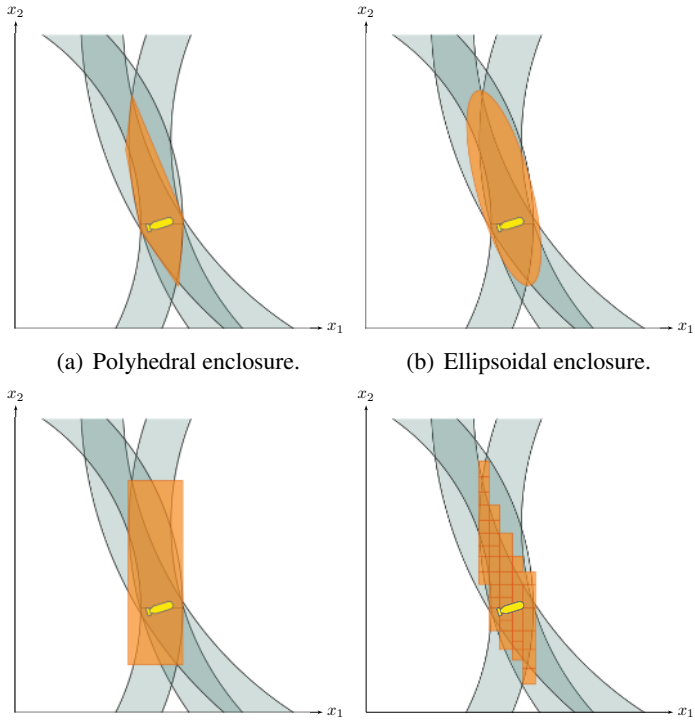


Figure 1.1. *Range-only robot localization among three beacons represented by red boxes ▪. The position of $\mathcal{R}$ in $(0,0)$ is unknown. The state estimation is performed based on uncertain measurements displayed by rings ▪. Several kinds of solution sets are shown in Figure 1.2. For a color version of the figures in this chapter see www.iste.co.uk/rohou/robot.zip*

The robot is located on one of the intersections of the range circles. When at least three beacons are used, the intersection is generally a single point: the horizontal localization is done without ambiguity.

Wrappers

Now, considering uncertainties on the measurements, the circles become *thick* and the intersection then results in a set that can be of any shape, as shown in Figure 1.1.



(a) Polyhedral enclosure. (b) Ellipsoidal enclosure.

(c) Interval enclosure. (d) Paving enclosure.

Figure 1.2. *Reliable set-membership approaches to enclose the set of feasible positions of $\mathcal{R}$ based on uncertain beacons measurements*

The solution set may even be made of several connected subsets or include holes. Figure 1.2 illustrates different methods to represent a set, namely zonotopes or polyhedral enclosures (Combastel 2005, Walter and

Piet-Lahanier 1989), ellipsoids (Rokityanskiy and Veres 2005), intervals or subpavings (Jaulin and Walter 1993a). These latter two have been proven to be efficient when dealing with nonlinear systems or complex solution shapes. The current chapter presents their theoretical basis that will be used then throughout this book. Section 1.2 focuses on interval analysis, and section 1.3 presents tools named *contractors* that aim at reliably reducing bounds of interval sets. Subpavings (Figure 1.2(d)) are introduced in section 1.4 for set-inversion, before discussions about implementation and concrete use of intervals in section 1.5.

1.2. Interval analysis

1.2.1. *Once upon a time*

From the beginning of mathematics, the consideration of irrational numbers has raised the question of their decimal representation. Intervals appear to be a natural solution to provide an accurate approximation of a number with two bounds. For example, Archimedes calculated a reliable enclosure of π: $\frac{223}{71} < \pi < \frac{22}{7}$.

However, the spirit of interval analysis has appeared recently with the advent of numerical computations, shining a new light on intervals. Using computers, real numbers of infinite precision are implemented by floating-point values of finite precision. An approximation of the numbers is then made and can lead to increasing errors during the computations[1].

The following illustration was given in Rump (1988). Let us compute

$$f(x,y) = \frac{1335}{4}y^6 + x^2\left(11x^2y^2 - y^6 - 121y^4 - 2\right) + 5.5y^8 + \frac{x}{2y} \qquad [1.2]$$

with $x = 77,617.0$ and $y = 33,096.0$ that are exactly computer-representable numbers. Almost the *same* results are obtained using single, double and

1 Note that these errors may vary from one computer to another.

extended precisions: the first seven figures are equal.

$$\begin{aligned} \text{single precision:} \quad & f = 1.172603\ldots \\ \text{double precision:} \quad & f = 1.1726039400531\ldots \\ \text{extended precision:} \quad & f = 1.172603940053178\ldots \end{aligned}$$

The exact value[2] $f^* = -0.8273960599468213^4_3$ shows how wrong the computations are, despite similar results among several levels of precision.

Using bounds to represent a value is of interest for computer representation, since these limits can be floating-point numbers of finite precision while ensuring a guaranteed numerical representation of a real number such as π. The counterpart is pessimism, because this arithmetic handles ranges of possibilities in place of unique values. Hence, the evaluation of equation [1.2] using interval arithmetic provides a large enclosure of the actual f^*, but guaranteed to contain it.

This modern need opened a new thread in mathematics. The first main book on this topic, (Moore 1966), brought a tremendous impetus to interval analysis, putting intervals into a new perspective: the uncertainties regarding floating-point precision can be extended to physical uncertainties. This is of high interest for robotic applications dealing with measurement errors and unknown environments. We will show in this chapter that intervals can even be used to represent strong temporal uncertainties. Additionally, interval analysis is appropriate to compute a search space and lends itself for many robotic applications such as forward kinematics (Merlet 2004), trajectory planning (Piazzi and Visioli 1997) or workspace analysis (Chablat *et al.* 2002).

This book will not focus on floating-point precision but on strong physical uncertainties. Moreover, because all the computations are based on rigorous interval analysis, we ensure that the presented outcomes are numerically guaranteed. In this context, results can be used for proof purposes.

2 This value has been computed by a reliable algorithm so that the displayed figures are guaranteed and the sixteenth known to yield between 3 and 4.

1.2.2. *Intervals*

Elementary notions about intervals are given in the following sections, all of them being used in the next chapters. For more information on interval analysis and its applications, the reader may refer to (Jaulin *et al*. 2001).

Basics

An interval $[x]$ is a closed and connected subset of $\mathbb{R}$. The set of all intervals is denoted $\mathbb{IR}$. An interval $[x]$ is delimited by a lower bound x^- and an upper one x^+ that can be infinite[3]:

$$[x] = [x^-, x^+] = \left\{ x \in \mathbb{R} \mid x^- \leqslant x \leqslant x^+ \right\}. \quad [1.3]$$

When $x^- = x^+$, the interval $[x]$ is said to be *degenerate*. In the following computations, any real number can be considered as a degenerate interval for the sake of generality. The same apply for the empty set $\varnothing$. In the problems we are dealing with, $\varnothing$ will represent an absence of solution.

The following are interval examples:

– $[2, 3]$;

– $[5] = \{5\}$;

– $[-\infty, \infty]$;

– $[0, \infty]$;

– $\varnothing$.

In this book, an actual but unknown value to be approximated will be denoted by a star: x^*. Its estimation is expressed by $[x]$, the width of which represents the uncertainty on x^*:

$$\text{width}\left([x]\right) = x^+ - x^-. \quad [1.4]$$

3 In this book, we will sometimes use the notation $\text{lb}\left([x]\right) = x^-$ (respectively $\text{ub}\left([x]\right) = x^+$) to denote the lower (upper) bound of $[x]$.

When some applications need to deal with a scalar value, the center of $[x]$ can be considered to represent an evaluation of the unknown x^*. The relevance of such choice will be discussed for tubes in Chapter 2, section 2.5.2.

$$\text{mid}\,([x]) = \frac{x^- + x^+}{2}. \quad [1.5]$$

Set operations on intervals

The intersection of two intervals is an interval:

$$[x] \cap [y] = \{z \in \mathbb{R} \mid z \in [x] \text{ and } z \in [y]\}\,. \quad [1.6]$$

However, the union of two intervals may not be an interval:

$$[x] \cup [y] = \{z \in \mathbb{R} \mid z \in [x] \text{ or } z \in [y]\}\,. \quad [1.7]$$

The interval union is computed as the *interval hull* of $[x] \cup [y]$ so that the result of the union is a connected subset of $\mathbb{R}$. In this chapter, this union is denoted by $[x] \sqcup [y]$ and defined as

$$[x] \sqcup [y] = [[x] \cup [y]]\,. \quad [1.8]$$

Some specific cases are listed below:

$$[x] \sqcup \varnothing = [x], \quad [1.9]$$

$$[x] \cap \varnothing = \varnothing, \quad [1.10]$$

$$[x] \sqcup [-\infty, \infty] = [-\infty, \infty]\,, \quad [1.11]$$

$$[x] \cap [-\infty, \infty] = [x]\,. \quad [1.12]$$

Interval computations

Interval analysis is based on the extension of all classical real arithmetic operators. Consider two intervals $[x]$ and $[y]$ and an operator $\diamond \in \{+, -, \cdot, /\}$. We define $[x] \diamond [y]$ as the smallest interval containing all feasible values for $x \diamond y$, assuming that $x \in [x]$ and $y \in [y]$, (Moore and Yang 1959):

$$[x] \diamond [y] = [\{x \diamond y \in \mathbb{R} \mid x \in [x], y \in [y]\}]\,, \quad [1.13]$$

$$[x] \diamond \varnothing = \varnothing. \quad [1.14]$$

Dealing with closed intervals, most of the operations can rely on their bounds. It is, for example, the case of addition, difference, union, etc.:

$$[x] + [y] = \left[x^- + y^-, x^+ + y^+\right], \quad [1.15]$$

$$[x] - [y] = \left[x^- - y^+, x^+ - y^-\right], \quad [1.16]$$

$$[x] \sqcup [y] = \left[\min\left(x^-, y^-\right), \max\left(x^+, y^+\right)\right], \quad [1.17]$$

$$\begin{aligned} [x] \cap [y] &= \left[\max\left(x^-, y^-\right), \min\left(x^+, y^+\right)\right] \text{ if } \max\left\{x^-, y^-\right\} \\ &\leqslant \min\left\{x^+, y^+\right\}, \\ &= \varnothing \text{ otherwise.} \end{aligned} \quad [1.18]$$

However, computing some operations is sometimes not as straightforward. Take, for example, the case of the division:

$$1/[y] = \begin{cases} \varnothing & \text{if } [y] = [0,0] \\ [1/y^+, 1/y^-] & \text{if } 0 \notin [y] \\ [1/y^+, \infty] & \text{if } y^- = 0 \text{ and } y^+ > 0 \\ [-\infty, 1/y^-] & \text{if } y^- < 0 \text{ and } y^+ = 0 \\ [-\infty, \infty] & \text{if } y^- < 0 \text{ and } y^+ > 0 \end{cases}, \quad [1.19]$$

$$[x] \,/\, [y] = [x] \cdot (1/\,[y])\,. \quad [1.20]$$

This arithmetic extension also includes the adaptation of elementary functions such as $\cos$, $\exp$ and $\tan$. Sometimes the image of an interval $[x]$ through a function f is not an interval, as it is the case for discontinuous functions. Then, the interval evaluation of $f([x])$, denoted $[f]([x])$, is the smallest interval containing all the images of all defined inputs through the function:

$$[f]([x]) = [\{f(x) \mid x \in [x]\}]\,. \quad [1.21]$$

When f is monotonic, $[f]([x])$ can be evaluated directly from its bounds:

$$[\exp]([x]) = \left[\exp(x^-), \exp(x^+)\right]. \quad [1.22]$$

Otherwise, other expressions or algorithms must be used (Bouron 2002). The cosine function is a typical example of a non-monotonic function:

$$[\cos]([0,2\pi]) = [-1,1] \neq \underbrace{[\cos(0),\cos(2\pi)]}_{[1,1]}. \quad [1.23]$$

Generalization

We may consider extending the notion of intervals of real numbers to other sets, such as intervals of functions, sets (Desrochers and Jaulin 2017), booleans or even graphs (Jaulin 2015b). The main contribution presented in this book is to provide tools to deal with intervals of trajectories, which will be introduced in Chapter 2. The following section focuses on intervals of vectors.

Interval vectors

A Cartesian product of n intervals defines a *box* – also called *interval vector* – belonging to the set $\mathbb{IR}^n$. As for vectors $\mathbf{x}$, boxes will be represented in bold: $[\mathbf{x}]$.

$$\begin{aligned}[\mathbf{x}] &= [x_1] \times \cdots \times [x_n], \\ &= [x_1^-, x_1^+] \times \cdots \times [x_n^-, x_n^+], \\ &= ([x_1], \ldots, [x_n])^\intercal .\end{aligned} \quad [1.24]$$

An interval vector $[\mathbf{x}]$ of $\mathbb{IR}^n$ is an axis-aligned box, closed and connected subset of $\mathbb{R}^n$. The i-th component $[x_i]$ is therefore the projection of $[\mathbf{x}]$ onto the i-th axis, as shown in Figure 1.3. The empty set of $\mathbb{R}^n$ is $(\varnothing \times \cdots \times \varnothing)^\intercal$.

Most operations on intervals are easily scalable to boxes by performing computations on each component. For example, a box $[\mathbf{x}]$ is defined by its bounds such that

$$\mathbf{x}^- = \left(x_1^-, \ldots, x_n^-\right)^\intercal, \qquad \mathbf{x}^+ = \left(x_1^+, \ldots, x_n^+\right)^\intercal . \quad [1.25]$$

These extensions also apply to matrices of intervals, for which each component is an interval, as for boxes.

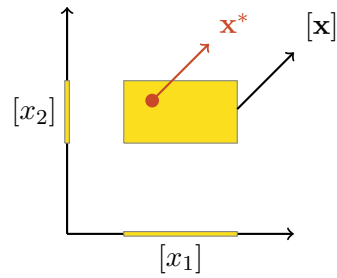


Figure 1.3. *An interval vector $[\mathbf{x}] \in \mathbb{IR}^2$, its components $[x_i]$ being projections onto axes*

1.2.3. *Inclusion functions*

Definition

Considering an n-dimensional box $[\mathbf{x}]$ as the input, a function $\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^m$ will output a set which is not necessarily a box, as shown in Figure 1.4 with an image made of non-connected subsets and a hole. Obtaining an accurate representation of the output set is often a complicated task, sometimes achieved with a computational burden of particular concern.

Instead, we use an *inclusion function* $[\mathbf{f}] : \mathbb{IR}^n \to \mathbb{IR}^m$ to enclose the image of $[\mathbf{x}]$ by $\mathbf{f}$ in a box such that $\forall [\mathbf{x}] \in \mathbb{IR}^n, \mathbf{f}([\mathbf{x}]) \subset [\mathbf{f}]([\mathbf{x}])$. Hence, a reliable enclosure of the image set can be evaluated reasonably quickly. Furthermore, inclusion functions are based on analytical expressions or even algorithms based on datasets.

Properties

An inclusion function is *thin* if the image of any degenerate interval vector $[\mathbf{x}] = \mathbf{x}$ is also punctual: $[\mathbf{f}](\mathbf{x}) = \{\mathbf{f}(\mathbf{x})\}$.

$[\mathbf{f}]$ is said to be *inclusion monotonic* if:

$$[\mathbf{x}] \subset [\mathbf{y}] \implies [\mathbf{f}]([\mathbf{x}]) \subset [\mathbf{f}]([\mathbf{y}]). \qquad [1.26]$$

An infinity of inclusion functions exist for a given $\mathbf{f}$, but only one of them will be *minimal* and denoted $[\mathbf{f}]^*$. $[\mathbf{f}]$ is minimal if $\forall [\mathbf{x}]$, $[\mathbf{f}]([\mathbf{x}])$ is the

smallest box containing $\mathbf{f}([\mathbf{x}])$. Figure 1.4 illustrates this notion. Any non-minimal inclusion function is said to be *pessimistic*.

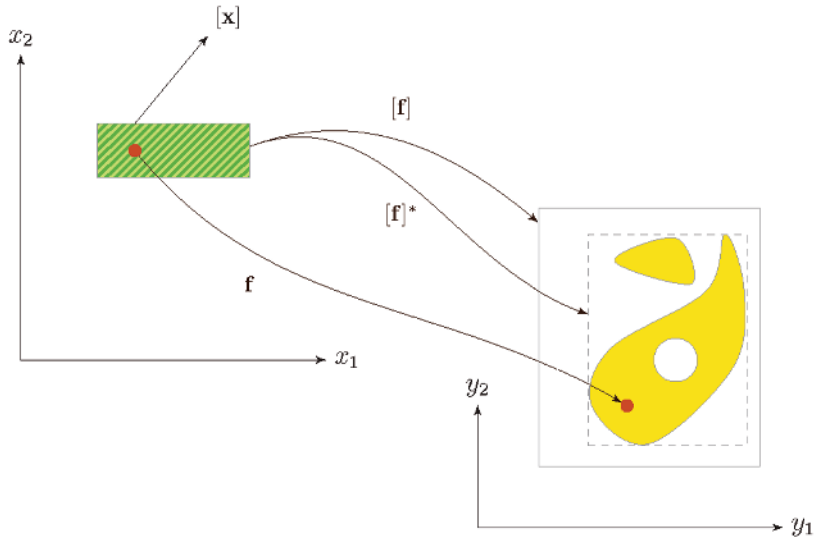


Figure 1.4. *Inclusion functions: the image of a box* $[\mathbf{x}]$ *in green • is an arbitrary set* $\mathbf{f}([\mathbf{x}])$*, shown in yellow •, that can be approximated with the* inclusion function $[\mathbf{f}]$*. The minimal inclusion function* $[\mathbf{f}]^*$ *provides the smallest enclosure of* $\mathbf{f}([\mathbf{x}])$

Natural inclusion functions

When $\mathbf{f}$ is a function made of a finite suite of elementary functions such as $\sin$, $\tan$, $\sqrt{(\cdot)}$, $\min$, ... and operators $+$, $-$, $*$, $/$, then the simplest method to obtain an inclusion function of it is to replace the variables $x_1, x_2, \ldots$ by their interval representation $[x_1], [x_2], \ldots$ and the functions and operators by their interval counterpart: $[\sin]$, $[\tan]$, etc. The obtained $[\mathbf{f}]$ is then said to be *natural inclusion function* of $\mathbf{f}$.

$[\mathbf{f}]$ is inclusion monotonic and thin. In addition, it is convergent if made of continuous functions and operators. However, a natural inclusion function may not be minimal due to dependencies between the variables and some wrapping effect, discussed in section 1.2.4. $[\mathbf{f}]$ will be minimal if each variable only

appears once in its expression and if only continuous functions and operators are involved in its expression.

Let us come back to the range-only beacon localization example and compute the natural inclusion of the distance function $\mathbf{g}$:

$$\begin{array}{l} \mathbf{g} : \mathbb{R}^2 \to \mathbb{R}, \\ \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \mapsto \sqrt{\left(x_1 - x_1^k\right)^2 + \left(x_2 - x_2^k\right)^2}. \end{array} \quad [1.27]$$

Replacing items by their interval counterpart, $[\mathbf{g}]$ is given by:

$$\begin{array}{l} [\mathbf{g}] : \mathbb{IR}^2 \to \mathbb{IR}, \\ \begin{pmatrix} [x_1] \\ [x_2] \end{pmatrix} \mapsto \sqrt{\left([x_1] - x_1^k\right)^2 + \left([x_2] - x_2^k\right)^2}. \end{array} \quad [1.28]$$

As functions $\sqrt{\cdot}$, $(\cdot)^2$ and operators $+$, $-$ are continuous on their definition domain and variables x_1, x_2 appear only once, the natural inclusion function $[\mathbf{g}]$ is minimal.

1.2.4. *Pessimism and wrapping effect*

We have seen that properties of basic operations on intervals may differ from their equivalent in $\mathbb{R}$, sometimes inducing unwanted pessimism. This section briefly presents two causes of overestimation. Although this effect does not impact the reliability of the results, it may lead to meaningless outcomes when intervals are too wide. In most cases, it becomes important to think about how to overcome such pessimism.

Dependencies between the variables

In interval analysis, different analytical expressions of the same function often lead to significantly different performances. As an example, consider the degenerate difference function $f(x) = x - x = 0$, where interval substitutions are made:

$$[x] - [x] = [\{a - b \mid a \in [x], b \in [x]\}] = [x^- - x^+, x^+ - x^-]. \quad [1.29]$$

The result is far from being thin. This issue appears when an expression involves a variable several times. Figure 1.5 provides another telling example (Ceberio and Granvilliers 2002).

Dependencies between the variables may be dealt with analytically, for example, by representing systems of equations into unique directed acyclic graphs (Schichl and Neumaier 2005) or by exploiting and deleting common sub-expressions (Araya *et al*. 2008). To this day, however, no general method has been adopted in existing solvers.

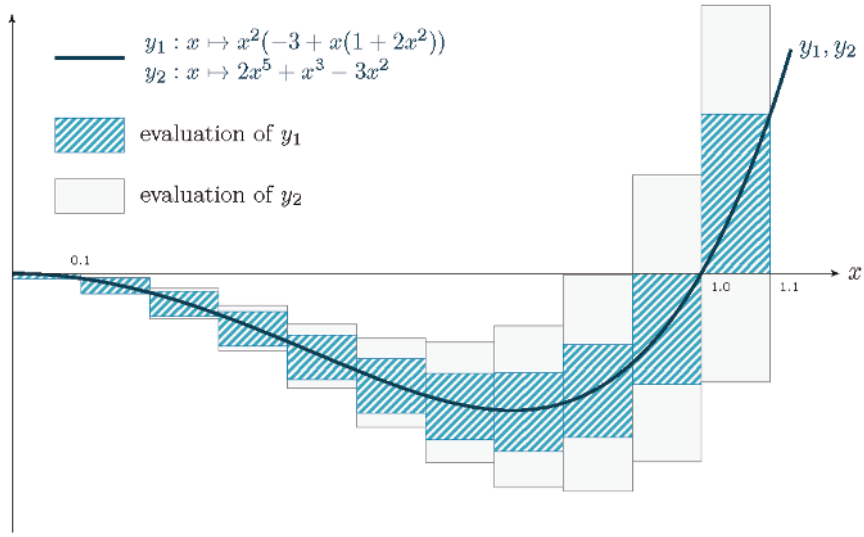


Figure 1.5. *Comparison of different interval evaluations of a same function $y_1 = y_2$. Properties of interval arithmetic sometimes lead to computed intervals that can be much larger than the exact range. This* dependency *problem can be overcome by rewriting expressions and using different factorization schemes. This example shows a classical interval evaluation and its counterpart shown in blue •, using a factorization based on Horner's rule (Ceberio and Granvilliers 2002)*

Wrapping effect

Intervals and interval vectors are axis-aligned items. Therefore, any set that is not a box made of boundaries aligned with axes will suffer from a pessimistic enclosure representation: the so-called *wrapping effect*.

This excessive enclosure, also called over-enclosure, may quickly increase when a set is successively evaluated by a composition of functions, each of which inducing its own wrapping effect. A well-known illustration was given in Moore (1966), presenting a suite of rotations of a box, as shown in Figure 1.6.

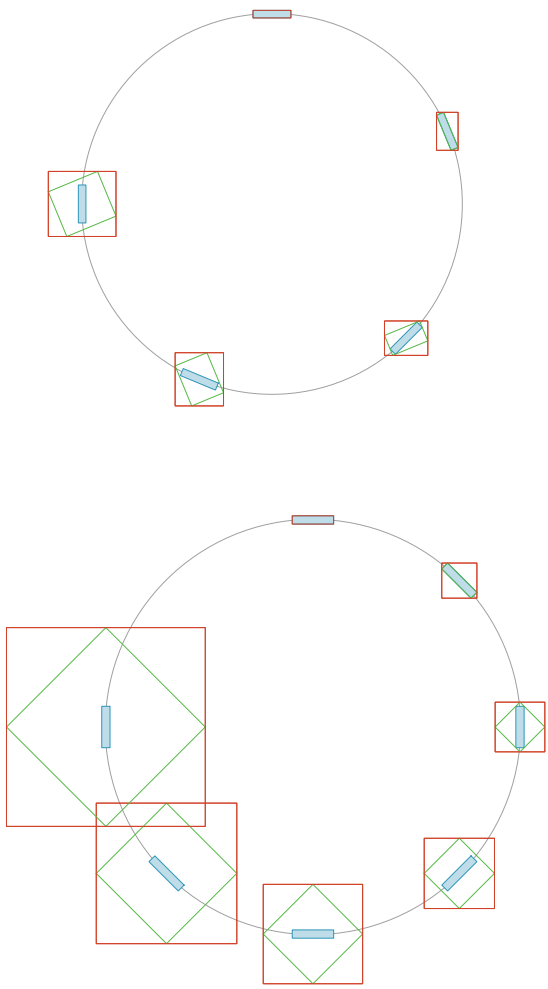


Figure 1.6. *The wrapping effect revealed through several boxes rotations. A blue box • is submitted to a suite of rotating angles resulting in a $\frac{3\pi}{2}$ global rotation. The red boxes • represent interval enclosures, while the green ones • are the results of rotated enclosures. The rotation is performed in four steps in the first case and in six steps in the second one. This highlights how the effect increases with successive computations. Note that a rotation performed in one or three steps would have provided the minimal enclosure of the final blue box since each intermediate evaluation is axis-aligned*

This effect can be overcome by dividing the solution space into a set of non-overlapping boxes (see section 1.4.1) at the expense of longer computation times and increased memory space.

1.3. Constraint propagation

In parallel with the emergence of interval analysis, the artificial intelligence community has developed new approaches based on constraint propagations (Bessiere 2006). These methods are used for systems solving problems involving discrete or continuous variables of real numbers (Benhamou and Older 1997, Van Hentenryck *et al*. 1998) and go well with the use of intervals defining the domains of these variables.

In our applications, a state estimation problem will be represented by a constraint network (CN) and solved by using *contractors* that are tools to reduce the domains of the network's variables.

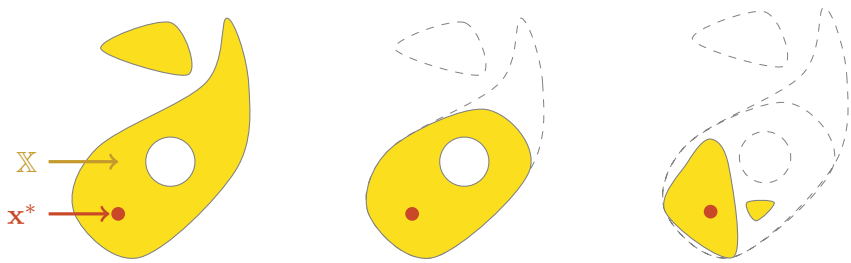
1.3.1. *Constraint networks*

Presentation

A mathematical problem can be presented by means of a CN involving variables $\{x_1, \ldots, x_n\}$ that must satisfy a set of rules or facts, called constraints and denoted by $\{\mathcal{L}_1, \ldots, \mathcal{L}_m\}$, over domains defining a non-empty range of feasible values $\{\mathbb{X}_1, \ldots, \mathbb{X}_n\}$ (Mackworth 1977).

The variables x_i can be symbols, real numbers (Araya *et al*. 2012) or vectors of $\mathbb{R}^n$. As presented in the introduction of this chapter, domains can be intervals, boxes, polytopes, etc. Generally, there are very few restrictions on the forms of the constraints that can be, for instance, nonlinear equations between the variables, such as $x_3 = \cos(x_1 + \exp(x_2))$, inequalities or even quantified parameters (Goldsztejn 2006).

The estimation then consists of computing the smallest variables' domain while satisfying the defined constraints[4]. Figure 1.7 provides a simple view of this approach.



(a) Initial unconstrained $\mathbb{X}$. (b) $\mathbb{X}$ considering $\mathcal{L}_1$. (c) $\mathbb{X}$ considering both $\mathcal{L}_1$, $\mathcal{L}_2$.

Figure 1.7. *In this theoretical view, a domain $\mathbb{X}$ represented in yellow • is known to enclose a solution $\mathbf{x}^*$ shown by a red dot • and consistent with two constraints $\mathcal{L}_1$ and $\mathcal{L}_2$. The estimation of $\mathbf{x}^*$ consists of reducing $\mathbb{X}$ while satisfying $\mathcal{L}_1$ and $\mathcal{L}_2$*

Decomposition

Problems involving complex equations can be broken down into a set of primitive constraints. Here, *primitive* means that the constraints cannot be decomposed anymore. For example, in our range-only localization problem, the observation constraint $\mathcal{L}_{g_k}$ that is based on equation [1.1] can be decomposed into:

$$\mathcal{L}_{g_k} : \rho_k = \sqrt{\left(x_1 - x_1^k\right)^2 + \left(x_2 - x_2^k\right)^2} \iff \begin{cases} a = x_1 - x_1^k, \\ b = x_2 - x_2^k, \\ c = a^2, \\ d = b^2, \\ e = c + d, \\ \rho_k = \sqrt{e}. \end{cases} \quad [1.30]$$

where $a, b, \ldots, e$ are intermediate variables used for ease of decomposition. This constitutes a network made of the $\mathcal{L}_-$, $\mathcal{L}_+$, $\mathcal{L}_{(\cdot)^2}$ and $\mathcal{L}_{\sqrt{\cdot}}$ elementary constraints, that are easily implementable. Complex constraints are then

4 We also often speak about Constraint Satisfaction Problems (CSPs) formulated as $\mathcal{H} : (\mathbf{f}(\mathbf{x}) = \mathbf{0}, \mathbf{x} \in [\mathbf{x}])$ and for which the resolution consists of computing the best approximation of $\mathbf{x}$. However, a set-membership state estimation is not formalized by such $\mathcal{H}$ and it is more appropriate to speak about CNs to represent the constraints of our applications.

affordable using propagation techniques on networks of elementary constraints.

Propagation

When working with finite domains, a propagation technique (Waltz 1972) can be used to simplify a problem. The process is run several times until a fixed point is reached when domains $\mathbb{X}_i$ can no longer be reduced. Interval analysis can be efficiently used for this purpose, taking advantage of interval arithmetic and its capacity to preserve any feasible solution.

Furthermore, using monotonous operators in iterative resolution processes, the constraints can be applied in any order (Apt 1999). The sequence can only impact the computation time, as it could be more interesting to apply one constraint before another.

The approach adopted in this chapter is to apply a given constraint on a box $[\mathbf{x}] \in \mathbb{IR}^n$ by means of a contractor $\mathcal{C}$.

1.3.2. *Contractors*

Formally, a contractor $\mathcal{C}_\mathcal{L}$ associated with the constraint $\mathcal{L}$ is an operator $\mathbb{IR}^n \to \mathbb{IR}^n$ that returns a box $\mathcal{C}_\mathcal{L}([\mathbf{x}]) \subseteq [\mathbf{x}]$ without removing any vector consistent with $\mathcal{L}$. We will use the following definition, adapted from (Chabert and Jaulin 2009):

DEFINITION 1.1.– *A contractor is a mapping $\mathcal{C}_\mathcal{L}$ from $\mathbb{IR}^n$ to $\mathbb{IR}^n$ such that*

(i) $\forall [\mathbf{x}] \in \mathbb{IR}^n$, $\mathcal{C}_\mathcal{L}([\mathbf{x}]) \subseteq [\mathbf{x}]$, *(contraction)*

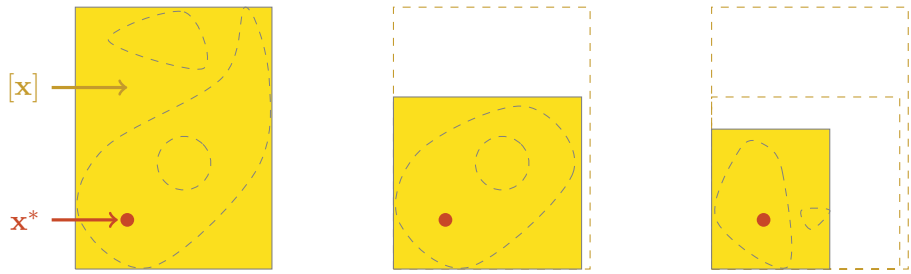
(ii) $\begin{pmatrix} \mathcal{L}(\mathbf{x}) \\ \mathbf{x} \in [\mathbf{x}] \end{pmatrix} \Longrightarrow \mathbf{x} \in \mathcal{C}_\mathcal{L}([\mathbf{x}])$. *(consistency)*

Figure 1.8 gives a simple illustration of contractions.

Property (i) states that a box is reduced when submitted to a contractor, while the second one justifies that a solution consistent with $\mathcal{L}$ cannot be removed. Therefore, contractors can be applied on boxes as many times as necessary without risking losing a solution or being more pessimistic.

In practice, these operators are usually given by polynomial-time algorithms. Constructing a store of contractors such as $\mathcal{C}_+$, $\mathcal{C}_{\sin}$ and $\mathcal{C}_{\exp}$ associated with primitive equations such as $z = x + y$, $y = \sin(x)$,

$y = \exp(x)$ has been the subject of much work (see, for example, (Jaulin *et al.* 2001, Chabert and Jaulin 2009, Desrochers and Jaulin 2016)). A significant part of interval analysis algorithms can also be wrapped into these contractors, as illustrated below.



(a) Initial box $[\mathbf{x}]$. (b) $\mathcal{C}_1\,([\mathbf{x}])$. (c) $\mathcal{C}_2\big(\mathcal{C}_1\,([\mathbf{x}])\big)$.

Figure 1.8. *Application of the constraints $\mathcal{L}_1$ and $\mathcal{L}_2$ of Figure 1.7 by means of respective contractors $\mathcal{C}_1$ and $\mathcal{C}_2$. On this theoretical example, the domain $\mathbb{X}$ is now a subset of a box $[\mathbf{x}]$, easily representable and contractible*

Examples

Let us consider the constraint $\mathcal{L}_+\,(a,x,y)\colon a = x + y$. The related contractor $\mathcal{C}_+$ is defined as

$$\begin{pmatrix} [a] \\ [x] \\ [y] \end{pmatrix} \overset{\mathcal{C}_+}{\longmapsto} \begin{pmatrix} [a] \cap ([x] + [y]) \\ [x] \cap ([a] - [y]) \\ [y] \cap ([a] - [x]) \end{pmatrix}. \qquad [1.31]$$

Thus, information on $[a]$, $[x]$ or $[y]$ can be propagated to the other intervals. For example, $\mathcal{C}_+\big([4,5],[0,3],[-2,2]\big)$ will produce $\big([4,5],[2,3],\,[1,2]\big)$. As another example, let us write the contractor $\mathcal{C}_{\exp}$ for the nonlinear constraint $\mathcal{L}_{\exp}\,(a,b) : a = \exp\,(b)$:

$$\begin{pmatrix} [a] \\ [b] \end{pmatrix} \overset{\mathcal{C}_{\exp}}{\longmapsto} \begin{pmatrix} [a] \cap \exp\,([b]) \\ [b] \cap \log\,([a]) \end{pmatrix}. \qquad [1.32]$$

Properties considered in this chapter

The *minimal* contractor is obtained when a box $[\mathbf{x}]$ is contracted to the smallest box containing the solution set.

A contractor is said to be *monotonous* if

$$[\mathbf{x}] \subseteq [\mathbf{y}] \implies \mathcal{C}([\mathbf{x}]) \subseteq \mathcal{C}([\mathbf{y}]). \quad [1.33]$$

Contractor programming

Although the implementation of an elementary constraint such as $\mathcal{L}_+$ can be trivial, things are further complicated when facing complex ones. The propagation is affordable by implementing a dedicated contractor for the difficult constraint. However this does not allow genericity in the approach, as the solution is hardly scalable for non-advised users: the new operator will be dedicated to this complex constraint, and a good knowledge of its operation may be necessary to extend it to other purposes.

The concept of *contractor programming* introduced in Chabert and Jaulin (2009) carries the approach a step further, proposing a formalism where a contractor $\mathbb{IR}^n \to \mathbb{IR}^n$ can also be interpreted as a subset of $\mathbb{R}^n$. This allows one to consider all the standard operations on sets of contractors such as:

$$\begin{aligned}
(\mathcal{C}_1 \cap \mathcal{C}_2)([\mathbf{x}]) &:= \mathcal{C}_1([\mathbf{x}]) \cap \mathcal{C}_2([\mathbf{x}]) && \textit{(intersection)} \\
(\mathcal{C}_1 \cup \mathcal{C}_2)([\mathbf{x}]) &:= \mathcal{C}_1([\mathbf{x}]) \sqcup \mathcal{C}_2([\mathbf{x}]) && \textit{(union)} \\
(\mathcal{C}_1 \circ \mathcal{C}_2)([\mathbf{x}]) &:= \mathcal{C}_1\big(\mathcal{C}_2([\mathbf{x}])\big) && \textit{(composition)} \\
\mathcal{C}_1^{\infty} &:= \mathcal{C}_1 \circ \mathcal{C}_1 \circ \mathcal{C}_1 \circ \ldots && \textit{(iterated composition)}
\end{aligned} \quad [1.34]$$

Using this formalism allows simple combinations of primitive contractors. Combining these operators can lead to complex ones, which can still provide reliable results, thus allowing one to deal with a wide range of problems.

The direct outcomes of such framework are genericity and simplicity: the user now focuses on the *what* instead of the *how* to build a solver, which is the essence of declarative programming. The energy is spent programming a solver based on mathematical constraints by combining contractors, rather than configuring a dedicated algorithm.

The contractor programming concept has triggered the development of several successful applications (Gning and Bonnifait 2006, Alexandre dit Sandretto *et al*. 2014, Jaulin 2011), thus demonstrating its efficiency. Very recent works proposed extending this concept to dynamical systems, outlining good prospects for this line of research. The two first contributions presented

in this book provide new contractors related to differential equations, opening doors to the estimation of dynamical systems such as those encountered in mobile robotics.

1.3.3. *Application to static range-only robot localization*

In this subsection we illustrate nonlinear state estimation using the aforementioned tools.

Problem statement

The robot $\mathcal{R}$ is located between three beacons $\mathcal{B}_k$, $k \in \{1, 2, 3\}$. Respective synchronous range measurements are ρ_k^*. However, these values are not known precisely and we shall assume the following bounded measurements:

	(x_1^k, x_2^k)	ρ_k^*	$[\rho_k]$
$\mathcal{B}_1$	$(-0.5, 4.0)$	4.03	$[3.63, 4.43]$
$\mathcal{B}_2$	$(-2.5, -2.5)$	3.53	$[3.13, 3.93]$
$\mathcal{B}_3$	$(2.5, -0.5)$	2.55	$[2.15, 2.95]$

Table 1.1. *Beacons' positions and respective measurements. Note that the ρ_k^* values are given indicatively without being used in the resolution process*

Recall that the observation constraint that links a measurement ρ_k to the position $\mathbf{x}$ of $\mathcal{R}$ is:

$$\mathcal{L}_{g_k}(\mathbf{x}, \rho_k) : \rho_k = \sqrt{\left(x_1 - x_1^k\right)^2 + \left(x_2 - x_2^k\right)^2}. \quad [1.35]$$

The problem is synthesized with the following CN:

$$\text{CN:} \begin{cases} \textbf{Variables: } \mathbf{x}, \rho_1, \rho_2, \rho_3 \\ \textbf{Constraints:} \\ \quad 1.\ \mathcal{L}_{g_1}(\mathbf{x}, \rho_1) \\ \quad 2.\ \mathcal{L}_{g_2}(\mathbf{x}, \rho_2) \\ \quad 3.\ \mathcal{L}_{g_3}(\mathbf{x}, \rho_3) \\ \textbf{Domains: } [\mathbf{x}], [\rho_1], [\rho_2], [\rho_3] \end{cases} \quad [1.36]$$

The domains are intervals initialized with the bounded values provided in Table 1.1. The robot's position is considered unknown: $[\mathbf{x}] = [-\infty, \infty]^2$.

State estimation

Each constraint $\mathcal{L}_{g_k}$ is implemented with a combination of primitive contractors, based on the decomposition detailed in equation [1.30]. Three contractors $\mathcal{C}_{g_k}$ are built in this way and applied on the domains:

1) $\mathcal{C}_{g_1}([\mathbf{x}], [\rho_1])$: contraction from $\mathcal{B}_1$'s measurement (Figure 1.9(a), 1.9(d));

2) $\mathcal{C}_{g_2}([\mathbf{x}], [\rho_2])$: contraction from $\mathcal{B}_2$'s measurement (Figure 1.9(b), 1.9(e));

3) $\mathcal{C}_{g_3}([\mathbf{x}], [\rho_3])$: contraction from $\mathcal{B}_3$'s measurement (Figure 1.9(c)).

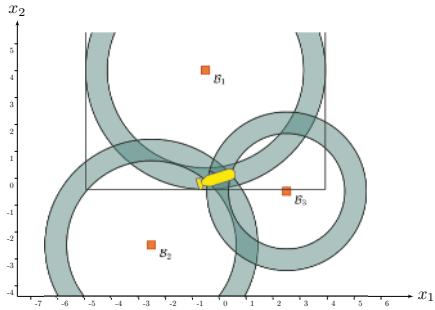
Each contractor will reduce the domain $[\mathbf{x}]$, which may raise new contraction possibilities for the other constraints. It becomes interesting to call again the other contractors in order to benefit from any contraction. An iterative resolution process is then used, where the contractors are called until a fixed point has been reached. By *fixed point* we mean that none of the domains $[\mathbf{x}]$ and $[\rho_k]$ have been contracted during a complete iteration. Figure 1.9 provides the synoptic of this state estimation. In this example, constraints have been propagated over seven iterations in less than 0.01 second.

1.4. Set-inversion via interval analysis

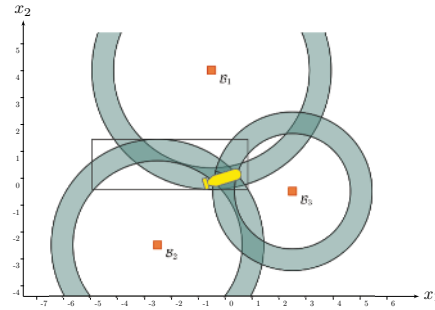
When the solution set is made of holes or several non-connected subsets, its enclosure may suffer from a strong pessimism: the so-called wrapping effect presented in section 1.2.4. A refinement can be obtained by dividing the solution space and test for each subdivision whether it encloses a part of the solution set or not. The result constitutes a new kind of wrapper, called *subpaving*, which is particularly useful for the *set-inversion* problems, presented in this section.

1.4.1. *Subpaving*

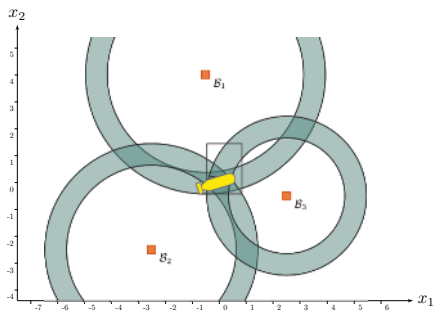
A thinner estimation of a set $\mathbb{X} \subset \mathbb{R}^n$ enclosed by a box $[\mathbf{x}] \in \mathbb{IR}^n$ can be made with a union of non-overlapping boxes $[\mathbf{x}]^{(i)}$ included in $[\mathbf{x}]$. This set of boxes is called *subpaving*.



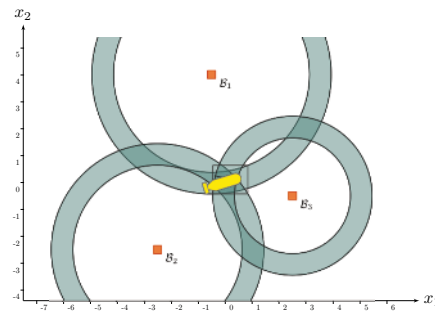
(a) Contraction from $\mathcal{B}_1$'s measurement.



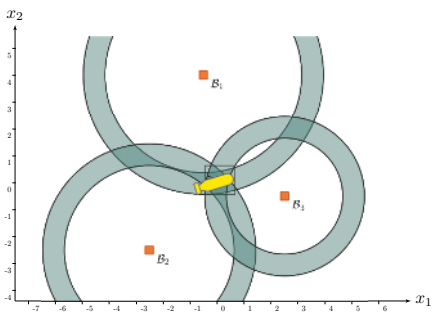
(b) Contraction from $\mathcal{B}_2$'s measurement.



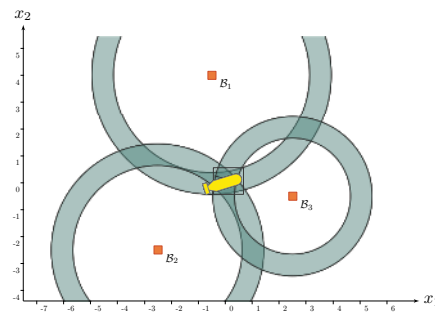
(c) Contraction from $\mathcal{B}_3$'s measurement.



(d) Contraction from $\mathcal{B}_1$'s measurement.



(e) Contraction from $\mathcal{B}_2$'s measurement.



(f) Fixed-point result.

Figure 1.9. *Set-membership localization with range-only measurements. Beacons represented by red boxes • emit signals received by the robot $\mathcal{R}$ drawn in yellow •. The figures illustrate successive contractions of the position box of $\mathcal{R}$ from each range-only bounded measurement represented by rings •*

A subpaving $\mathbb{K}$ of $[\mathbf{x}]$ covering completely $[\mathbf{x}]$ such that

$$[\mathbf{x}] = \bigcup_{[\mathbf{b}]\in\mathbb{K}} [\mathbf{b}] \qquad [1.37]$$

is called the *paving* of $[\mathbf{x}]$ and can be made of a collection of several subpavings.

If a thinner approximation of $\mathbb{X}$ is affordable with a subpaving, we might also expect a qualification of such approximation. This can be done using two subpavings denoted $\mathbb{X}^-$ and $\mathbb{X}^+$ such that

$$\mathbb{X}^- \subset \mathbb{X} \subset \mathbb{X}^+. \qquad [1.38]$$

Figure 1.10 illustrates so-called *inner* and *outer* approximations of a set $\mathbb{X}$ with subpavings respectively denoted $\mathbb{X}^-$ and $\mathbb{X}^+$. An inner approximation gathers boxes that contain only solutions, while an outer approximation is made of boxes that may contain a solution. The precision of the computation is given by the width of the set $[\mathbb{X}^-, \mathbb{X}^+]$ enclosing the boundary $\partial\mathbb{X}$ of the solution set. A thinner splitting of boxes will increase this precision, at the expense of longer computation times and raising memory space.

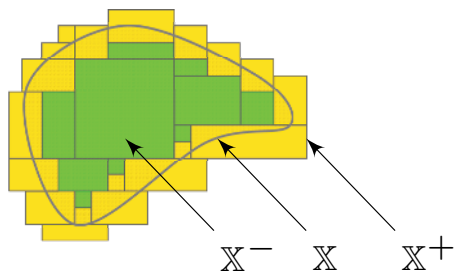


Figure 1.10. *Inner and outer approximations of a set $\mathbb{X}$ (hatched part) with two subpavings $\mathbb{X}^-$ and $\mathbb{X}^+$ represented respectively by green • and the union of green and yellow boxes ••. In any case, the boundary $\partial\mathbb{X}$ must remain enclosed within the visible yellow boxes*

1.4.2. *SIVIA algorithm for set-inversion*

Let us consider the computation of the reciprocal image $\mathbb{X} \subset \mathbb{R}^n$ such that $\mathbb{X} = \mathbf{f}^{-1}(\mathbb{Y})$, where $\mathbb{Y} \subset \mathbb{R}^m$ is the image set of $\mathbb{X}$ by a possibly nonlinear function $\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^m$. This operation is called *set-inversion* and is formalized as the characterization of:

$$\mathbb{X} = \{\mathbf{x} \in \mathbb{R}^n \mid \mathbf{f}(\mathbf{x}) \in \mathbb{Y}\} = \mathbf{f}^{-1}(\mathbb{Y}). \quad [1.39]$$

A SIVIA[5] algorithm (Jaulin and Walter 1993b) can be used to approximate $\mathbb{X}$ from any $\mathbb{Y} \subset \mathbb{R}^m$ and any function $\mathbf{f}$ admitting an inclusion function $[\mathbf{f}] : \mathbb{IR}^n \to \mathbb{IR}^m$. The approximation is made by bracketing $\mathbb{X}$ between two subpavings $\mathbb{X}^-$ and $\mathbb{X}^+$. Starting from an initial box $[\mathbf{x}]^{(0)} \in \mathbb{IR}^n$, SIVIA will apply inclusion tests to decide whether it belongs to $\mathbb{X}^+$, both $\mathbb{X}^-$ and $\mathbb{X}^+$, or none. In the case of undecidability, the strategy is to bisect the box and reapply the tests on the sub-boxes.

A recursive version of SIVIA is given in Algorithm 1 in which four cases are encountered:

1) $[\mathbf{f}]([\mathbf{x}]) \cap \mathbb{Y} = \varnothing$: $[\mathbf{x}]$ does not belong to $\mathbb{X}$;

2) $[\mathbf{f}]([\mathbf{x}]) \subset \mathbb{Y}$: any vector in $[\mathbf{x}]$ is solution, so $[\mathbf{x}]$ belongs to $\mathbb{X}$ and is stored in both $\mathbb{X}^-$ and $\mathbb{X}^+$;

3) $[\mathbf{f}]([\mathbf{x}])$ has a non-empty intersection with $\mathbb{Y}$ while not being a subset of $\mathbb{Y}$. It is an undetermined case where $[\mathbf{x}]$ may contain a solution. Therefore:

a) if $\text{width}([\mathbf{x}]) < \varepsilon$, then the box is considered small enough regarding the expected precision of the algorithm. The process stops here by storing $[\mathbf{x}]$ into $\mathbb{X}^+$;

b) otherwise, a bisection of $[\mathbf{x}]$ is performed, for example, along its largest dimension, and new tests are applied on each resulting box.

These cases are shown in Figure 1.11. Such inversion is easily affordable as it is based only on the inclusion of $\mathbf{f}$: its inverse is not required.

5 Set-inversion via interval analysis (SIVIA)

Algorithm 1 SIVIA(in: $[\mathbf{f}]$, $[\mathbf{x}]$, $\mathbb{Y}$, ε – inout: $\mathbb{X}^-$, $\mathbb{X}^+$)

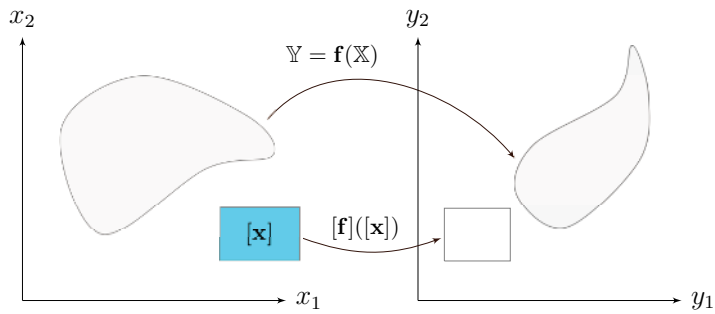
if $[\mathbf{f}]([\mathbf{x}]) \cap \mathbb{Y} \neq \varnothing$ **then**
  if $[\mathbf{f}]([\mathbf{x}]) \subset \mathbb{Y}$ **then**
    $\mathbb{X}^+ \leftarrow \mathbb{X}^+ \cup [\mathbf{x}]$ ▷ *outer set*
    $\mathbb{X}^- \leftarrow \mathbb{X}^- \cup [\mathbf{x}]$ ▷ *inner set*
  else if $\text{width}([\mathbf{x}]) < \varepsilon$ **then**
    $\mathbb{X}^+ \leftarrow \mathbb{X}^+ \cup [\mathbf{x}]$ ▷ *outer set only*
  else
    bisect$([\mathbf{x}])$ into $[\mathbf{x}]^{(1)}$ and $[\mathbf{x}]^{(2)}$
    SIVIA$([\mathbf{f}], [\mathbf{x}]^{(1)}, \mathbb{Y}, \varepsilon, \mathbb{X}^-, \mathbb{X}^+)$
    SIVIA$([\mathbf{f}], [\mathbf{x}]^{(2)}, \mathbb{Y}, \varepsilon, \mathbb{X}^-, \mathbb{X}^+)$
  end if
end if

The precision ε of the approximation, materialized by the width of the interval $[\mathbb{X}^-, \mathbb{X}^+]$, is the only parameter to set with this algorithm. The thinner the interval $[\mathbb{X}^-, \mathbb{X}^+]$, the better the approximation of the inversion. In any case, the true solution set $\mathbb{X}$ remains enclosed within these bounds. Figure 1.12 provides an illustration of subpavings computed by SIVIA, in various accuracy levels.

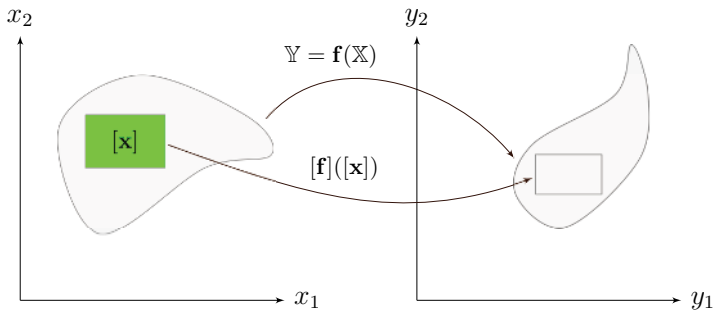
Several optimizations can be done. For example, the SIVIA algorithm is easily parallelizable when several evaluations of $[\mathbf{f}]([\mathbf{x}])$ can be done simultaneously. In addition, the paving of the solution space can be built as a binary tree, each node of which corresponds to a bisection of a box $[\mathbf{x}]$. This regular representation speeds up the access to solution boxes.

1.4.3. *Illustration involving contractions*

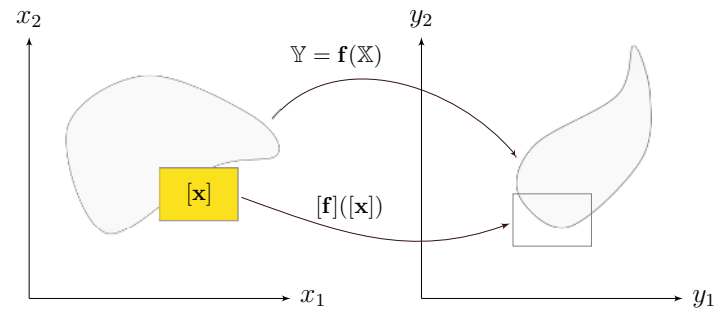
Coming back to the range-only problem, we now intend to consider two beacons instead of three in order to deal with an ambiguous solution set. The robot stays located at $(0, 0)$.



(a) $[\mathbf{f}]([\mathbf{x}]) \cap \mathbb{Y} = \varnothing \implies \mathbf{f}([\mathbf{x}]) \cap \mathbb{Y} = \varnothing$. The box $[\mathbf{x}]$ does not belong to the outer set $\mathbb{X}^+$ and so to the inner one $\mathbb{X}^-$.



(b) $[\mathbf{f}]([\mathbf{x}]) \subset \mathbb{Y} \implies \mathbf{f}([\mathbf{x}]) \subset \mathbb{Y}$. The box $[\mathbf{x}]$ belongs to both the inner and outer sets, respectively $\mathbb{X}^-$ and $\mathbb{X}^+$.



(c) Indefinite case. $[\mathbf{x}]$ is either subdivided or placed in the outer set $\mathbb{X}^+$.

Figure 1.11. *Inclusion tests for set-inversion. The chosen color code is kept in the remainder of this document: green • for inner solution sets, yellow • for boxes belonging to outer sets only and blue • for no-solution sets*

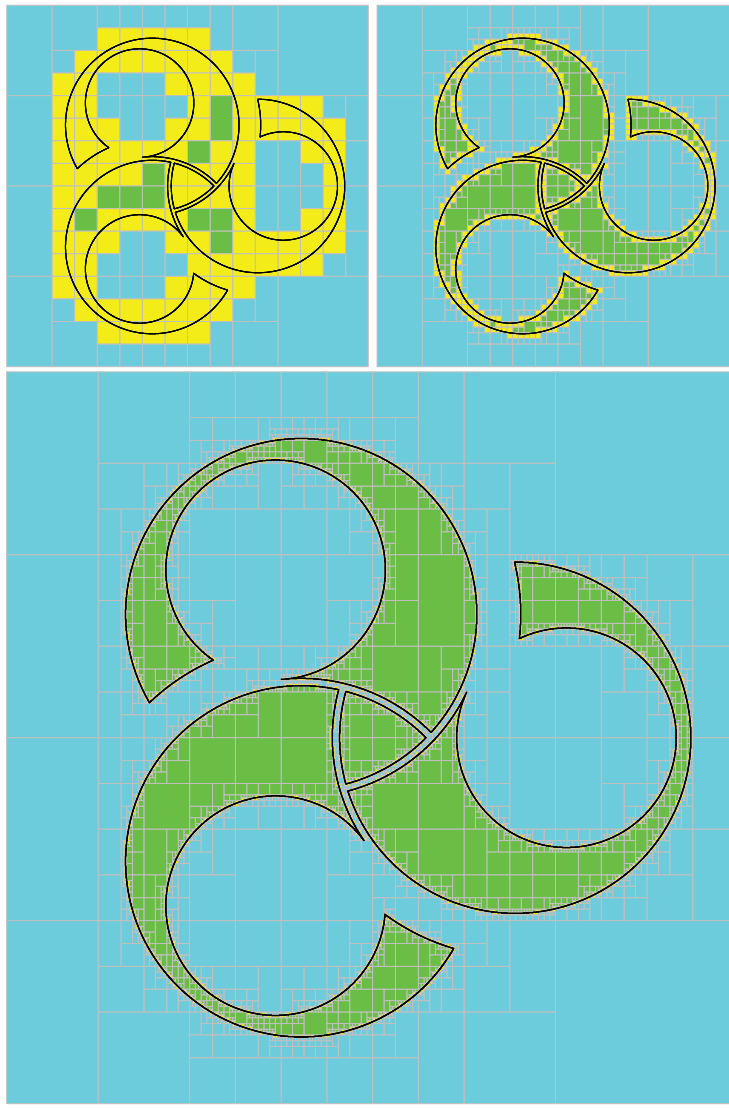
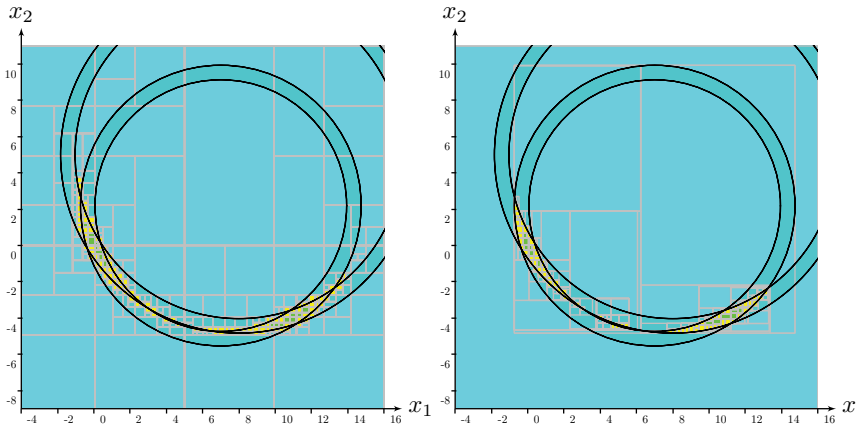


Figure 1.12. *Subpavings computed by a SIVIA algorithm in various accuracy levels. The boundary $\partial\mathbb{X}$ of the true solution set is plotted by a black line. The inner and outer sets are respectively drawn by green • and both yellow and green boxes ••. The part proven to not contain solutions is represented in blue •. This approach enables the estimation of sets of any shape such as this triskelion (Le Gallo 2016)*

SIVIA will provide a thinner approximation of the solution set, while the only use of contractors would enclose it by a single box. Nonetheless, the algorithm can be coupled with contractors in order to decrease the time complexity, or equivalently the space complexity, by reducing the boxes to be evaluated in the subpavings. Figure 1.13 provides a comparison between a classical SIVIA algorithm for this range-only problem (Table 1.2) and its combination with the contractors presented in section 1.3.3. This coupling is simply done by contracting $[\mathbf{x}]$ before its bisection in the inconclusive inclusion test (see line 8 of Algorithm 1). The counterpart of this approach is that the paving is no longer regular.



(a) Classical SIVIA algorithm: 306 boxes. (b) SIVIA coupled with contractors: 112 boxes.

Figure 1.13. *Range-only problem with two beacons. The solution set is complex and its enclosure by a box would present too much pessimism. The use of SIVIA then becomes relevant on this problem. The figures provide a comparison between a classical algorithm and its adaptation including a contraction process*

	(x_1^k, x_2^k)	ρ_k^*	$[\rho_k]$
$\mathcal{B}_1$	$(8.0, 5.0)$	9.43	$[9.03, 9.83]$
$\mathcal{B}_2$	$(7.0, 2.2)$	7.34	$[6.94, 7.74]$

Table 1.2. *Beacons' positions and respective measurements*

1.4.4. *Kernel characterization of an interval function*

The kernel characterization of a function is elementary and can be encountered in many problems under the form $\mathbf{f}(\mathbf{x}) = \mathbf{0}$, $\mathbf{x} \in [\mathbf{x}]$. The kernel $\ker \mathbf{f}$ of a function $\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^n$ is a subset of the domain of $\mathbf{f}$ defined as

$$\ker \mathbf{f} = \{\mathbf{x} \in \mathbb{R}^n \mid \mathbf{f}(\mathbf{x}) = \mathbf{0}\} = \mathbf{f}^{-1}(\mathbf{0}). \qquad [1.40]$$

When $\mathbf{f}$ is known to be bounded by an interval-valued function $[\mathbf{f}]$, the characterization of $\ker[\mathbf{f}]$ can be done with a SIVIA algorithm. This was the object of the work by Aubry *et al.* (2014) with definitions and examples. For the sake of being self-contained, we will briefly recall the concepts that will be used later in this book.

The kernel of an interval function $[\mathbf{f}]$ is defined by

$$\ker[\mathbf{f}] = \bigcup_{\mathbf{f} \in [\mathbf{f}]} \ker \mathbf{f} = \{\mathbf{x} \in [\mathbf{x}] \mid \mathbf{0} \in [\mathbf{f}](\mathbf{x})\}. \qquad [1.41]$$

which is shown in Figure 1.14.

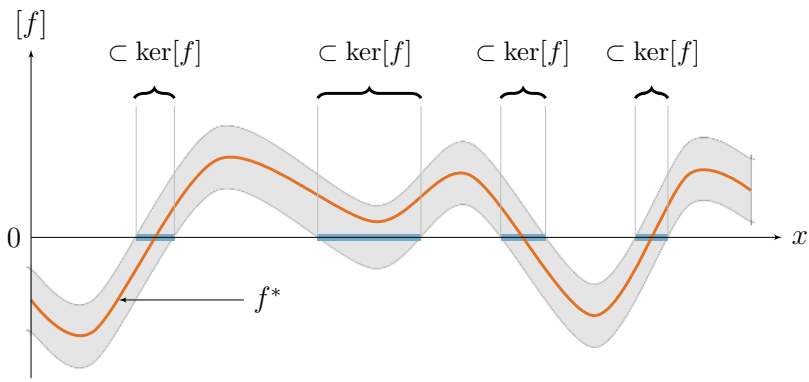


Figure 1.14. *The kernel of an interval function* $[f]$

$\ker[\mathbf{f}]$ is a set $\mathbb{X}$ that can be approximated by two subpavings $\mathbb{X}^-$ and $\mathbb{X}^+$. Therefore, the kernel of the actual but unknown function $\mathbf{f}^*$ can be evaluated

by $\ker \mathbf{f}^* \subset \mathbb{X}^+$. In the following, we will denote $\mathbf{f}^-(\mathbf{x})$ and $\mathbf{f}^+(\mathbf{x})$ as the upper and lower bounds of $[\mathbf{f}](\mathbf{x})$ (see Figure 1.15). We have:

$$\forall \mathbf{x}, [\mathbf{f}](\mathbf{x}) = \left[\mathbf{f}^-(\mathbf{x}), \mathbf{f}^+(\mathbf{x})\right]. \quad [1.42]$$

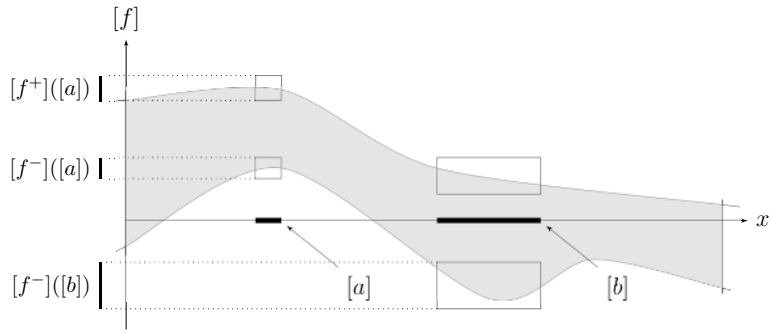


Figure 1.15. *Bounds on an interval function* $[f]$

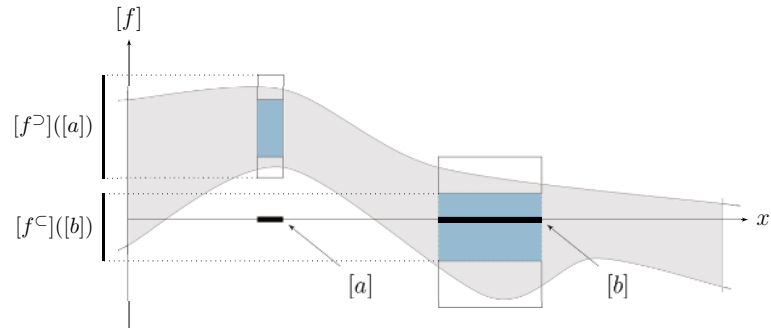


Figure 1.16. *Inclusion functions* $[f^{\subset}]$ *and* $[f^{\supset}]$

We will further assume the following two convergent inclusion functions $[\mathbf{f}^{\subset}]$ and $[\mathbf{f}^{\supset}]$ shown in Figure 1.16 and defined by:

$$\left[\mathbf{f}^{\subset}\right]([\mathbf{x}]) = \left[\text{ub}\left(\mathbf{f}^-([\mathbf{x}])\right), \text{lb}\left(\mathbf{f}^+([\mathbf{x}])\right)\right], \quad [1.43]$$

$$\left[\mathbf{f}^{\supset}\right]([\mathbf{x}]) = \left[\text{lb}\left(\mathbf{f}^-([\mathbf{x}])\right), \text{ub}\left(\mathbf{f}^+([\mathbf{x}])\right)\right]. \quad [1.44]$$

These definitions allow the following inclusion tests:

$$\mathbf{0} \in [\mathbf{f}^{\subset}]([\mathbf{x}]) \implies [\mathbf{x}] \subset \mathbb{X}, \quad [1.45]$$

$$\mathbf{0} \notin [\mathbf{f}^{\supset}]([\mathbf{x}]) \implies [\mathbf{x}] \cap \mathbb{X} = \varnothing. \quad [1.46]$$

Then, the approximation of $\ker[\mathbf{f}] = \mathbb{X}$ can be made based on these tests, as presented in Algorithm 2.

Algorithm 2 kernelSIVIA $(\text{in} : [\mathbf{f}], [\mathbf{x}], \varepsilon, \text{inout} : \mathbb{X}^-, \mathbb{X}^+)$

if $\mathbf{0} \in [\mathbf{f}^{\supset}]([\mathbf{x}])$ **then**
  if $\mathbf{0} \in [\mathbf{f}^{\subset}]([\mathbf{x}])$ **then**
    $\mathbb{X}^+ \leftarrow \mathbb{X}^+ \cup [\mathbf{x}]$ ▷ *outer set*
    $\mathbb{X}^- \leftarrow \mathbb{X}^- \cup [\mathbf{x}]$ ▷ *inner set*
  else if $\text{width}([\mathbf{x}]) < \varepsilon$ **then**
    $\mathbb{X}^+ \leftarrow \mathbb{X}^+ \cup [\mathbf{x}]$ ▷ *outer set only*
  else
    bisect$([\mathbf{x}])$ into $[\mathbf{x}]^{(1)}$ and $[\mathbf{x}]^{(2)}$
    kernelSIVIA$([\mathbf{f}], [\mathbf{x}]^{(1)}, \varepsilon, \mathbb{X}^-, \mathbb{X}^+)$
    kernelSIVIA$([\mathbf{f}], [\mathbf{x}]^{(2)}, \varepsilon, \mathbb{X}^-, \mathbb{X}^+)$
  end if
end if

1.5. Discussions

Today interval methods are yet little known and used by the mobile robotics and automatic control communities: Bayesian approaches are much more common in these fields. The reason is mainly due to the fact that research on the interval topic is new, though promising. In addition, set-membership methods work on sets while most applications expect scalar results, often assessed by some probabilities that are hardly verifiable. It is mainly a matter of how one can appropriately use one method or another according to the context. By going further, the approaches could be combined in order to keep the best of each world, but this is a topic which has entered its initial phase of investigations (Abdallah *et al.* 2008, Neuland *et al.* 2014, De Freitas *et al.* 2016).

This section aims at providing some answers to recurring questions on this approach.

1.5.1. *From sensors to reliable results*

When dealing with real situations, speaking about guaranteed approaches is all based on the nature of the inputs to our algorithms: the datasets. The transition from theoretical computations to real values is a significant matter, which needs to be done rigorously in order to ensure further guaranteed outcomes.

In practice, a measurement error is often modeled by a Gaussian distribution which has an infinite support. Therefore, setting bounds around this measurement already constitutes a theoretical risk of losing the actual value. A choice must be made at this step, considering such risk. After that, however, any algorithm that is based on interval methods is ensured not to increase this risk.

Figure 1.17 presents the interval evaluation of a measurement μ assumed to follow a Gaussian distribution so that we consider the real value enclosed within the interval $[x]$, centered on μ, with a 95% confidence rate. Datasheets usually give sensor specifications such as the standard deviation σ. The bounded value $[x]$ can then be inflated according to this dispersion value.

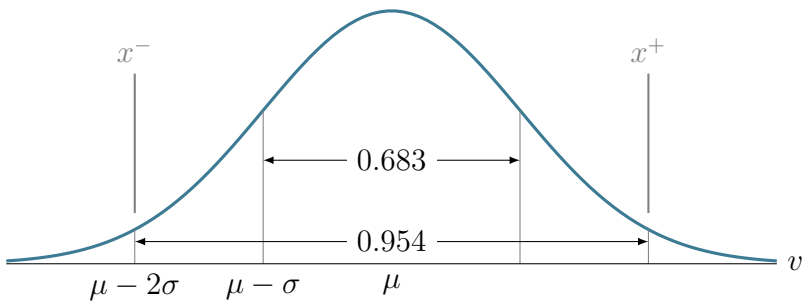


Figure 1.17. *An interval $[x] = [x^-, x^+]$ computed from a Gaussian distribution to guarantee a 95% confidence rate over a measurement μ: $[x] = [\mu - 2\sigma, \mu + 2\sigma]$*

Note that in many applications, measurement bounds are known and can be guaranteed without any distribution information. The problem of outliers is another topic, and needs to be handled separately as these errors seriously spoil bounding methods as they would also disrupt the usual statistical approaches.

1.5.2. *Numerical libraries*

As stated in section 1.2.1, the computer representation of a real number – or an interval defined by real bounds – can also induce errors. Indeed, the nearest floating-point number is generally used by machines to represent a real $x \in \mathbb{R}$. The exact value of x will be lost if it does not exist among the computer numbers. As we have seen when introducing interval analysis, a rigorous containment of x will be assessed using an interval defined by representable bounds. Throughout the calculations, these bounds must be reliably represented, thus preventing any loss of value. This procedure is called *outer rounding* (see Figure 1.18), and needs to be executed for any arithmetic operation on intervals.

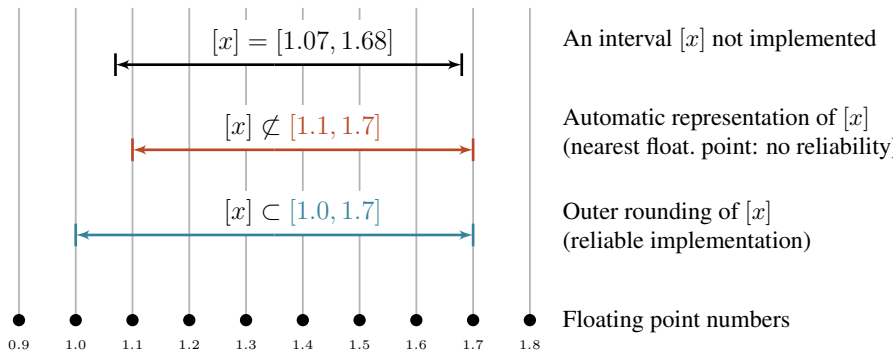


Figure 1.18. *Outer rounding of an interval defined by non-representable bounds. The black dots • represent floating-point numbers depending on the computer precision. Its reliable approximation, represented in blue •, encloses the initial range of values*

Several numerical libraries have been created to this end. For example, computations presented in this chapter rely on the *GAOL*[6] and *filib++*[7]

6 http://frederic.goualard.net/#research-software

7 http://www2.math.uni-wuppertal.de/ xsc/software/filib.html

(Nehmeier and von Gudenberg 2011) libraries that ensure numerical computations such as outer rounding.

At a higher level of abstraction, we use *IBEX*[8] (Chabert 2017): a C++ library for constraint processing over real numbers. It provides a set of tools from the contractor programming paradigm presented in this chapter.

Finally, the contributions presented in this chapter come together with a dedicated open source library called *Tubex*[9], the aim of which is to implement constraints over sets of trajectories. This will be discussed in Chapter 2.

1.5.3. *Reliable tool for proof purposes*

Once any source of error is reliably handled, the following results are both computationally and mathematically guaranteed not to lose solutions. The outcomes of these algorithms can therefore be used for verified computing and so for proof purposes (see, for example, (Tucker 1999, Goldsztejn *et al.* 2011)).

This asset of set-membership methods will be highlighted in Chapter 5, in which we provide an original method to prove loops along an uncertain robot trajectory.

1.6. Conclusion

Interval analysis provides a reliable way to deal with uncertainties over real numbers. This chapter, has shown how simple it is to address nonlinear problems without having to perform any linearization nor approximation, as we do for usual methods such as the Kalman filter. Furthermore, intervals serve as a reliable solution to deal with poor datasets, in which any data is of interest. Chapter 6 will highlight this point by providing a new localization method in poor environments, where other approaches would badly behave. Finally, the reliability of the results makes these methods suitable for areas of engineering where proof of performance and functional verification is required.

8 http://www.ibex-lib.org

9 http://www.simon-rohou.fr/research/tubex-lib

In addition, when coupled with constraint propagation approaches, interval analysis allows one to deal with a wide range of problems in the most simple way. The definition of a set of constraints and their application by contractors on intervals and boxes have proved their worth and are attractive to the communities of constraint programming and set-membership tools. There are still many lines of research to explore in this field, for example, in order to overcome wrapping effects, propose new elementary contractors or even to reliably address problems with respect to outliers outside measurements errors (Norton and Veres 1993, Pronzato and Walter 1996, Carbonnel *et al*. 2014).

The next chapter extends these interval concepts to continuous-time dynamical systems. Our goal is to be able to deal with a wider class of problems such as differential equations and inter-temporal measurements. The forthcoming new methods will still follow the techniques of the contractor programming approach.