Volume 27 Number 3

ACTA CYBERNETICA

Szeged, 2026

ACTA CYBERNETICA

Information for authors. Acta Cybernetica publishes only original papers in the field of Computer Science. Manuscripts must be written in good English. Contributions are accepted for review with the understanding that the same work has not been published elsewhere. Papers previously published in conference proceedings, digests, preprints are eligible for consideration provided that the author informs the Editor at the time of submission and that the papers have undergone substantial revision. If authors have used their own previously published material as a basis for a new submission, they are required to cite the previous work(s) and very clearly indicate how the new submission offers substantively novel or different contributions beyond those of the previously published work(s). There are no page charges. An electronic version of the published paper is provided for the authors in PDF format.

Manuscript Formatting Requirements. All submissions must include a title page with the following elements: title of the paper; author name(s) and affiliation; name, address and email of the corresponding author; an abstract clearly stating the nature and significance of the paper. Abstracts must not include mathematical expressions or bibliographic references.

References should appear in a separate bibliography at the end of the paper, with items in alphabetical order referred to by numerals in square brackets. Please prepare your submission as one single PostScript or PDF file including all elements of the manuscript (title page, main text, illustrations, bibliography, etc.).

When your paper is accepted for publication, you will be asked to upload the complete electronic version of your manuscript. For technical reasons we can only accept files in LaTeX format. It is advisable to prepare the manuscript following the guidelines described in the author kit available at https://cyber.bibl.u-szeged.hu/index.php/actcybern/about/submissions even at an early stage.

Submission and Review. Manuscripts must be submitted online using the editorial management system at https://cyber.bibl.u-szeged.hu/index.php/actcybern/submission/wizard. Each submission is peer-reviewed by at least two referees. The length of the review process depends on many factors such as the availability of an Editor and the time it takes to locate qualified reviewers. Usually, a review process takes 6 months to be completed.

Subscription Information. Acta Cybernetica is published by the Institute of Informatics, University of Szeged, Hungary. Each volume consists of four issues, two issues are published in a calendar year. From 2024, issues are published online only, and articles are made available as soon as they are accepted and copyedited. The content is available free of charge.

Contact information. Acta Cybernetica, Institute of Informatics, University of Szeged. P.O. Box 652, H-6701 Szeged, Hungary. Tel: +36 62 546 396, Fax: +36 62 546 397, Email: acta@inf.u-szeged.hu.

Web access. The above information along with the contents of past and current issues are available at the Acta Cybernetica homepage https://cyber.bibl.u-szeged.hu/.

Tibor Gyimóthy
Department of Software Engineering, University of Szeged, Hungary
gyimothy@inf.u-szeged.hu

Gabriel Istrate
Faculty of Mathematics and Computer Science, University of Bucharest, Romania
gabriel.istrate@unibuc.ro

Zoltan Kato
Department of Image Processing and Computer Graphics, University of Szeged, Hungary
kato@inf.u-szeged.hu

Dragan Kukolj
RT-RK Institute of Computer Based Systems, Novi Sad, Serbia
dragan.kukolj@rt-rk.com

László Lovász
Department of Computer Science, Eötvös Loránd University, Budapest, Hungary
lovasz@cs.elte.hu

Kálmán Palágyi
Department of Image Processing and Computer Graphics, University of Szeged, Hungary
palagyi@inf.u-szeged.hu

Andreas Rauh
School II – Department of Computing Science, Group Distributed Control in Interconnected Systems, Carl von Ossietzky Universität Oldenburg, Germany
andreas.rauh@uni-oldenburg.de

György Vaszil
Department of Computer Science, Faculty of Informatics, University of Debrecen, Hungary
vaszil.gyorgy@inf.unideb.hu

Special Issue of the Summer Workshops on Interval Methods 2024 and 2025

Guest Editors

Sébastien Lahaye
Université d'Angers, Angers, France
sebastien.lahaye@univ-angers.fr

Luc Jaulin
Robex, Lab-STICC, ENSTA-Bretagne, France
lucjaulin@gmail.com

Andreas Rauh
Carl von Ossietzky Universität Oldenburg, Germany
andreas.rauh@uni-oldenburg.de

Preface

The Summer Workshop on Interval Methods (SWIM) was initiated in 2008. Since then, it has become an annual keystone meeting with a special focus on the latest scientific developments in interval analysis and its applications. SWIM attracts a broad community of researchers working in the fields of (control) engineering, computer science, and mathematics.

This special issue contains peer-reviewed articles based on work that has been presented during SWIM 2024, held in Maastricht, The Netherlands, and SWIM 2025, held in Rennes, France. During these two events, in total around 40 talks were given.

Out of these contributions, a collection of 11 papers were selected for publication after a thorough peer-review process. These paper focus on the following topics:

- Solving robotics tasks in search and rescue missions, optimal path planning, localization, and pose estimation for smart wheel chairs.
- Systematic tuning of interval observers with applications to set-based state estimation for battery systems.
- Methods for verification on the basis of signal temporal logic, reachability analysis, and uncertainty propagation in dynamic systems.
- Novel methods for computing with sets that represent bounded uncertainty.

This diverse list of contributions highlights the interdisciplinarity of the activities of the community of interval and set-based methods, ranging from fundamental methods and the implementation of associated software tools in the field of computer algebra to the solution of a broad range of engineering applications.

We as the organizers of SWIM 2024 and SWIM 2025 and the guest editors of this Special Issue would like to thank all authors as well as all reviewers for their invaluable contributions which help to make the series of SWIM workshops a success in the community of interval methods and their applications.

Pieter Collins,
Vincent Drevelle

Organizers of SWIM 2024 and 2025

Sébastien Lahaye, Luc Jaulin,
Andreas Rauh

Guest Editors

Acta Cybernetica **27** (2026) 267–292.

A Discrete Search and Rescue Problem Under Uncertain Interval Parameters*

Joël Bougron[ab], Julien Alexandre dit Sandretto[ac], Bruno Ricaud[de], Stéphane Cardon[fg], and Aline Hufschmitt[fh]

Abstract

The Search and Rescue Problem (SARP) can be formulated in an environment subject to both objective uncertainty (randomness inherent to nature) and subjective uncertainty (lack of knowledge about the state of the world). In this paper, we present an interval arithmetic interpretation of the uncertainty problem. A crisis scenario is modeled as an assignment and optimization problem with interval-valued parameters and constraints. These intervals capture uncertainty over the problem data. A branch and bound algorithm is used to explore the solution space. Interval arithmetic is employed to compute bounds and obtain feasible assignments. From the resulting assignments, residual injury intervals are derived to assess the impact of uncertainty on each wounded person. Parallel computing techniques are also investigated to reduce execution times in the solution process.

Keywords: intervals, Search And Rescue, WTA, uncertainty

1 Introduction

In crisis-response and humanitarian assistance scenarios, efficiently allocating limited medical resources is critical. Coordinating caregivers with diverse skills to meet heterogeneous needs under urgent conditions requires structured decision-making models. We are interested in this type of problem because it appears in various

*This work was supported by KNDS France, a Franco-German company of the defense industry. This project also received funding from the French National Association for Research and Technology (ANRT).

[a]ENSTA Paris, Institut Polytechnique de Paris, Palaiseau, France
[b]E-mail: joel.bougron@ensta.fr, ORCID: 0009-0004-2062-2832
[c]E-mail: julien.alexandre-dit-sandretto@ensta.fr, ORCID: 0000-0002-6185-2480
[d]Innovation and International Directorate (D2I), KNDS France, Versailles, France
[e]E-mail: bruno.ricaud@knds.fr, ORCID: 0009-0009-6360-0808
[f]Research Center (CReC), Académie Militaire de Saint-Cyr Coëtquidan, Guer, France
[g]E-mail: stephane.cardon@st-cyr.terre-net.defense.gouv.fr, ORCID: 0000-0002-2475-1365
[h]ORCID: 0000-0003-1649-3180

DOI: 10.14232/actacyb.323748

applications such as allocating technicians to jobs, vehicles to deliveries, servers to requests, weapons to targets, etc.

1.1 Assignment under uncertainties

In this paper, we consider the **Search and Rescue Problem** (SARP), which includes several subjects of study such as information perception and fusion, path planning, and more particularly, what interests us here, the assignment of caregivers to wounded individuals.

The problem of **assigning resources to tasks** has been widely studied in the literature and concerns many domains where resources must be optimized, such as industry, logistics, computing, military domain, etc. However, in all these domains, resources are assigned to tasks under various **constraints** (e.g., a resource cannot be assigned to two tasks simultaneously).

Under these constraints, it is necessary to calculate ideally the optimal assignments according to several parameters. However, in emergency situations, some parameters are not precisely known and are therefore uncertain. To model this uncertainty, we propose an interval-based approach [22], rather than a probabilistic approach. The goal is then to solve this problem of assigning caregivers to wounded individuals using a constraint optimization algorithm with interval parameters.

In Section 1.2, we formally specify the problem. We show in Section 2 why it remains an open problem in the existing literature. To address it, we have designed a Branch and Bound type algorithm that directly manipulates intervals, which we explain in Section 3. Then, we apply and analyze it in Section 4. The conclusion of this paper is given in Section 5.

1.2 A caregivers-to-wounded assignment problem with interval parameters

We consider a crisis scenario involving m **caregivers**, organized into t **teams**, whose objective is to minimize the injuries of n **wounded individuals**. An example is given in Figure 1. Each wounded person has a specific injury level, which influences the likelihood that a caregiver will engage with them. Caregivers possess **different skills**, while the wounded present **diverse needs**. The degree of adequacy between a caregiver's skills/equipment and a wounded person's injury type also affects the likelihood of engagement, and an assignment is only possible if at least one caregiver has characteristics that match those types. The crisis scenario can be modeled as an assignment problem which has specific characteristics. First, each caregiver can be assigned to at most one wounded person. In other words, the assignment function is **injective** from the wounded to the caregivers. Second, each wounded person can receive assistance from zero, one, or multiple caregivers — the assignment is thus **many-to-one**. It is relevant for combining the effects of the caregivers. Third, the numbers of caregivers and wounded individuals are arbitrary, making the problem **unbalanced**. Moreover, caregivers are required to remain within predefined teams in order to stay organized in this critical context.

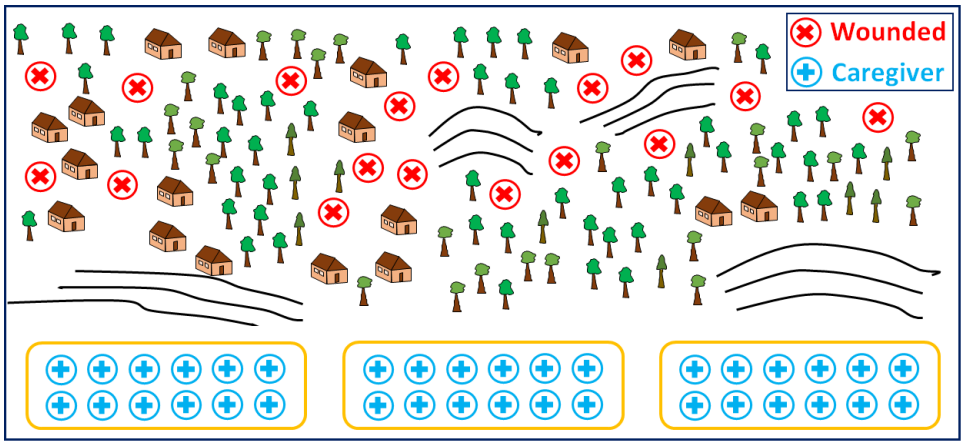


Figure 1: A crisis scenario

Each assignment — or non-assignment — is represented by a **binary decision variable** $x_{ij} \in \{0,1\}$. Caregiver i is assigned to wounded person j if $x_{ij} = 1$; otherwise, $x_{ij} = 0$. The solution space is therefore the Cartesian product of $m \times n$ binary sets $\{0,1\}$.

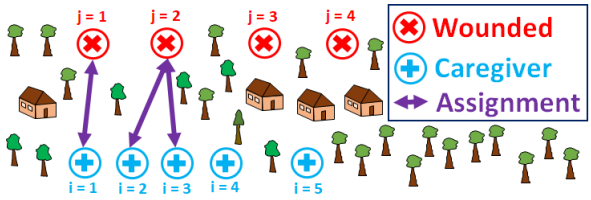


Figure 2: Illustration of the Assignment Problem

For illustration, Figure 2 shows an example with three assignments corresponding to $x_{11} = 1$, $x_{22} = 1$, and $x_{32} = 1$. The non-assignments correspond to $x_{ij} = 0$.

Two parameters model the problem, as input data, depending on the circumstances, derived from professional knowledge and habits, provided by experts or within the command and control (C2) framework. On the one hand, for a wounded person j, v_j denotes their initial injury level, i.e. the **severity level** of their health condition (e.g., severity of a snake bite, severity of bleeding). This value, intrinsic to the wounded person, is what we aim to minimize. On the other hand, for each caregiver-wounded pair (i, j), p_{ij} represents the probability of successful treatment, i.e. the adequacy between the skills and equipment of caregiver i and the **type of injury** of wounded person j (e.g., snake bite, bleeding wound), independently of their severity level. It indicates the reduction ratio of v_j if caregiver i treats wounded person j. Consequently, $1 - p_{ij}$ is the **probability of failed treatment**, which we aim to minimize.

These variables are considered uncertain, then we represent them using intervals [22]. The initial injury level is modeled as an interval $[v_j] = [\underline{v_j}, \overline{v_j}] \in \mathbb{IR}$, and the probability of failed treatment is also represented as an interval $1-[p_{ij}] \in \mathbb{IR}$, which we seek to keep as low as possible, by minimizing its upper bound.

We observe that if $x_{ij} = 0$, then $(1 - p_{ij}) \cdot x_{ij} = 0$, so the injury is not reduced. But if $x_{ij} = 1$, then $(1 - p_{ij}) \cdot x_{ij} = 1 - p_{ij}$, and the injury is reduced by $(1 - p_{ij})$ ratio. In the entire scenario, if these ratios are assumed to be independent, then they are **cumulative**. The product over the m caregivers assigned to wounded person j gives the uncertain cumulative probability interval that all caregivers fail to treat wounded j, which we aim to minimize. The product of this cumulative probability of failed treatment with the initial injury v_j yields the residual injury of wounded j. More precisely, the cumulative probability of failed treatment for wounded j is given by

$$\prod_{i=1}^{m}(1 - [p_{ij}])^{x_{ij}} \in \mathbb{IR} : \text{ cumulative probability of failed treatment} \tag{1}$$

and, consequently, the residual injury of wounded j is expressed as

$$[v_j]\prod_{i=1}^{m}(1 - [p_{ij}])^{x_{ij}} \in \mathbb{IR} : \textbf{ residual injury } \text{of wounded } j. \tag{2}$$

The objective is to minimize the overall injuries of the wounded, represented as an interval quantity equal to the sum of their residual injuries. SARP is modeled as the minimization of the total injury interval $[F] = [\underline{F}, \overline{F}] \in \mathbb{IR}$. The optimization problem seeks the **binary assignment matrix** (x^*_{ij}) minimizing:

$$(x^*_{ij})_{1\le i\le m,\ 1\le j\le n} \in \arg\min_{x_{ij}\in\{0,1\}} \sum_{j=1}^{n}[v_j] \cdot \prod_{i=1}^{m}(1 - [p_{ij}])^{x_{ij}} \tag{3}$$

This nonlinear multiplicative formulation captures the accumulation of treatment probabilities and is equivalent to the classical Weapon Target Assignment Problem (WTAP [21]) which is found in military applications. Indeed, the two approaches share analogies as resource allocation problems to tasks [23]. Thus, the caregiver is considered as a weapon, a wounded person as a target, the initial injury level of a wounded person v_j corresponds to the initial threat level of a target, the probability of successful treatment of a wounded person corresponds to the probability of neutralizing a target, and finally, minimizing the total injury of all wounded individuals corresponds to minimizing the total threat of the targets. For this reason, SARP is viewed as an instance of WTAP in the remainder of the paper.

In the WTA problem, the two parameters v_j and p_{ij} are independent since the former is intrinsic to j, while the latter represents the adequacy between i and j. Indeed, in an assignment phase, even if several caregivers are assigned to the same wounded person, the effects of these assignments are modeled as simultaneous (we

do not consider the movements). However, it can be noted that the problem is iterative, each iteration being independent of the preceding one. But, for each new assignment phase, the parameters v_j and p_{ij} are updated according to the effects of the previous iteration.

Intervals are used to encode input uncertainties, with v_j and p_{ij}. Interval arithmetic enables consistent propagation of these uncertainties to the output objective $[F]$. Interval arithmetic is also employed to check constraints. To complete our optimization problem, we need to consider five constraints.

First, the **binary constraint** (C.1), which reflects the domain of the decision variables. Second, the **at-most-one-wounded-per-caregiver constraint** (C.2), which follows from the definition of the assignment problem (this constraint is enforced by design in the algorithm). For example, in Figure 2, caregiver 1 cannot engage with another wounded at the same time as wounded 1. Third, the **minimum care constraint** (C.3): if caregiver i is assigned to wounded person j, then the expected care $[v_j] \cdot p_{ij}$ must be greater than or equal to a minimum threshold of minimum care $w_j^{\min}$ (gathered in the vector $W^{\min}$). This is an active filtering constraint to avoid negligible treatment. For example, if $\overline{v_3} = 10.0$, $\overline{p_{23}} = 0.4$, and $w_2^{\min} = 5$, then caregiver 2 will not engage with wounded 3. Fourth, the **team exclusivity constraint** (C.4): if caregivers from team k are assigned, then no caregiver from any other team $\ell \neq k$ may be assigned (this is also an active filtering constraint). The caregivers' teams (of equal size) are stored in the vector T, where $T[i] = t_k$ means that caregiver i belongs to team t_k. Thus, three teams are shown in Figure 1. For example, if wounded person 09 is treated by team 1 (e.g., caregivers 01 and 02), it can't be treated by any other team (this example corresponds to the results of Section 4). Fifth, the **maximum residual injury constraint** (C.5): the total sum must be less than or equal to the maximum threshold of residual injuries $w_j^{\max}$ (gathered in the vector $W^{\max}$).

$$\forall (i,j) \in \{1,\ldots,m\} \times \{1,\ldots,n\}, \quad x_{ij} \in \{0,1\} \tag{C.1}$$

$$\forall i \in \{1,\ldots,m\}, \quad \sum_{j=1}^{n} x_{ij} \leq 1 \tag{C.2}$$

$$\forall (i,j), \quad x_{ij} = 1 \Rightarrow [v_j] \cdot [p_{ij}] \subset [w_j^{\min}, +\infty] \tag{C.3}$$

$$\forall j, \forall k, \quad \left(\sum_{i \in t_k} x_{ij}\right) \cdot \left(\sum_{i \in t_l} x_{ij}\right) = 0 \tag{C.4}$$

$$\forall j, \quad [v_j] \prod_{i=1}^{m} (1 - [p_{ij}])^{x_{ij}} \subset [0.0, w_j^{\max}] \tag{C.5}$$

Since, as described above, the objective of this minimization problem is interval-valued, we choose to follow the Hurwicz optimism–pessimism criterion [19] to scalarize it using the upper and lower bounds of each interval, denoted respectively by $\mathrm{ub}(\cdot)$ and $\mathrm{lb}(\cdot)$, as follows. The function $F_\alpha(X)$ is defined by:

$$\alpha \cdot \mathrm{ub}\left(\sum_{j=1}^{n}[v_j]\cdot\prod_{i=1}^{m}(1-[p_{ij}])^{x_{ij}}\right) \;+\; (1-\alpha)\cdot \mathrm{lb}\left(\sum_{j=1}^{n}[v_j]\cdot\prod_{i=1}^{m}(1-[p_{ij}])^{x_{ij}}\right) \quad (4)$$

where the parameter $\alpha \in [0,1]$ expresses the relative weight given to the optimistic (lower bound) versus pessimistic (upper bound) evaluation of the interval objective of the minimization problem. When $\alpha = 1$, the decision is fully robust and based only on the worst case; when $\alpha = 0$, it is fully optimistic and relies only on the best case; intermediate values yield a compromise between these two extremes. In this work, we adopt the robust case $\alpha = 1$ (pessimistic approach), so that:

$$F(X) \;=\; \mathrm{ub}\left(\sum_{j=1}^{n}[v_j]\cdot\prod_{i=1}^{m}(1-[p_{ij}])^{x_{ij}}\right) \quad (5)$$

This implies that, as illustrated in Figure 3, our approach selects the green intervals, which are considered greater than the red ones in all three configurations, regardless of their widths.

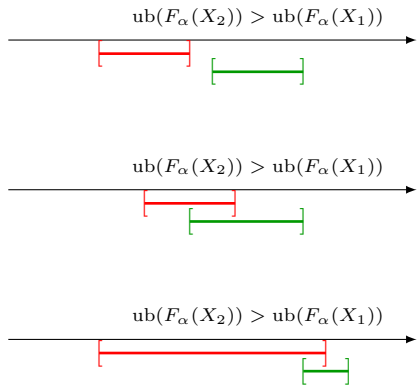


Figure 3: Interval comparisons for red $F_\alpha(X_1)$ and green $F_\alpha(X_2)$

Instead of minimizing the worst-case, it is possible to minimize the best-case (i.e., selecting the red intervals). This is the choice of optimism, which can be seen as taking a risk. If this optimism is indeed aligned with reality, it would allow us to use fewer caregivers for the same set of assignments, with these resources being redeployed elsewhere.

2 Survey and contribution

The context of epistemic uncertainty makes classical optimization methods on real numbers inoperative and requires approaches adapted to the manipulation of inter-

vals, their propagation, and their consideration in the evaluation of solution quality. In this section, we present several approaches identified in the literature that could fit our problem.

2.1 Prior publications

Our problem is a generalization of an unbalanced assignment problem, with several caregivers assignable to one wounded (many-to-one), structured in teams, with a nonlinear objective function and parameter coefficients represented by intervals.

We designate our problem as *Interval Parameters Unbalanced Many-to-one by Teams NonLinear Assignment Problem* or **Interval Weapon Target Assignment Problem** (IWTAP) because of its proximity with WTA problem designation.

No publication considering intervals in the WTA problem has been identified. The state of the art is therefore oriented on the most closely related problems in terms of characteristics, such as the Assignment Problem (AP), which is one-to-one and balanced.

It is then a matter of characterizing and analyzing existing contributions according to several axes: the structure of the objective function, the nature of the decision variables, the nature of the parameters, the resolution methods, and above all the way intervals are manipulated in the resolution. We examine below the approaches that, on the one hand, use interval parameters, and on the other hand have points in common with IWTAP according to the definition. Thus, we first present continuous or binary-continuous problems. Then, we discuss discrete problems reduced to continuous ones. Next, we focus on discrete problems where intervals are not directly considered but via representative real numbers. Then we discuss some links with fuzzy logic and with probabilities. Finally, we present application contexts with the same structure.

2.2 Continuous problem with interval parameters

Our problem is discrete. In this section we detail several types of problems involving continuous decision variables and interval parameters, to identify approaches from continuous settings that can be reused or adapted.

2.2.1 Expression of a single-objective interval problem as a bi-objective problem on a bound and the center

As initiated and formalized in [15], a continuous linear single-objective problem with interval parameters can be transformed into a bi-objective problem on a bound and the center of the intervals. This is quite far from our problem, which is nonlinear and discrete. However, the idea is reused for linear and discrete problems as discussed in Sections 2.4.2 and 2.4.3.

In this framework, order relations are defined in [15] that allow comparing objectives. This is applied in [16] for a center-radius order relation, with a weighting

factor between the center and the radius. But the solving method (genetic algorithm) cannot provide an optimal solution.

2.2.2 Interval analysis to solve continuous optimization problems

In general, interval analysis makes it possible to generate lower and upper bounds for a function. Thus, for the continuous domain, in [13] are described algorithms that use interval analysis to solve nonlinear optimization problems. In particular, interval analysis allows reducing the search space while keeping the box wide enough to contain the solution. For example, the study in [6] finds value intervals of photovoltaic cell models (continuous decision variables) in an optimization problem, solved using a *Depth-First Search with Backtracking* and an interval library (here, `Ibex` [14]).

Such solving methods require continuous decision variables and cannot handle discrete ones, as in our case.

2.2.3 Interval analysis to solve binary-continuous optimization problems

Other sources such as [1] and [29] which address cost minimization for solvent mixing, consider **binary** decision variables y_i (whether the solvent is selected or not) and **continuous** decision variables x_i (the solvent rate in the composition as real decision variables x_i). In this case, interval analysis is used to reduce the search space of the continuous variables x_i, the interval extension function taking as input the x_i and not the y_i.

So, such solving methods still require continuous decision variables and cannot handle discrete ones.

2.3 Assignment problems with interval parameters reduced to continuous problems

In this section, we describe problems with interval parameters reformulated as continuous optimization problems in order to identify approaches suitable for our discrete one.

2.3.1 Binary variables considered as continuous variables

Assignment problems with interval parameters can sometimes be reformulated as continuous optimization problems. For example, [12] addresses a problem of maximizing a linear multi-objective function with interval parameters, transformed into a single-objective one by moving the other objectives into constraints. The decision variables are binary for an unbalanced many-to-one location assignment problem. Interval parameters are reconsidered as exact via a penalty function. An **equivalent probability matrix** allows, instead of binary decision variables, to consider the probability in $[0, 1]$ that the binary decision variable equals 1. Decision variables

thus become continuous (the rest being solved by a particle swarm optimization metaheuristic).

But these approaches are not adapted to the binary nature of the decision, nor to the nonlinear form of the Equation (3).

2.3.2 Substitution of binary variables into continuous parameter variables

In our problem, interval parameters make the resolution more complex because of their presence in the body of the objective function. One can be led to make a **substitution of variables** by moving the binary variables of the objective function to the constraints. At the same time, the problem is reformulated so that the new decision variable is the upper bound of the interval parameter (and not the lower bound which is in the constraints) and therefore is real. [28] presents such an approach for a linear, balanced, one-to-one multi-objective problem (reformulated and solved as a min-max problem).

However, this approach requires the objective function to be linear, unlike in our case.

2.4 Assignment problems with interval parameters reduced to assignment problems with real parameters

Problems with interval parameters are often simplified by replacing intervals with real values. In this section, we present some examples. No known solvers directly handle assignment problems combining binary variables and interval uncertainty. That is why interval uncertainty is often treated indirectly: the problem is solved by considering real parameters. In a first very direct approach, only the centers of the intervals can be considered. In a second, less direct approach, the single-objective problem is transformed into a multi-objective one: a disjunction can then be made on bounds and/or centers (which can also be combined with several objectives if the function is already multi-objective).

2.4.1 Assignment problem with Hungarian method on interval centers

The simplest of these approaches consists in replacing interval parameters with real values to obtain an equivalent problem.

Thus, a balanced, single-objective, linear assignment problem with interval parameters is treated in [20] via midpoints by applying the Hungarian algorithm on the matrix of interval centers. A numerical application is provided in [26]. This approach is extended to unbalanced cases in [27] via dummy rows/columns. This method, for other order relations, is also reused in [7].

However, these approaches are not compatible with the multiplicative form of the objective function in Equation (3).

2.4.2 Case disjunction methods with a bound and the center

As seen in [15], approaches disjoin the cases according to several interval characteristics (lower bound, upper bound, or center), and according to the components of the objective function. Thus, in [18], for a minimization problem of a linear bi-objective function (unbalanced one-to-one assignment), one ends up with four single-objective problems by disjunction of cases, according to the two objectives, and according to the upper bound and the center. Similarly, these approaches are not compatible with Equation (3).

2.4.3 Case disjunction methods on the bounds

In [2, 3], the same principle is applied, but with disjunction according to each of the lower and upper bounds, then solved by genetic algorithms or solvers.

In [24], under the close name *Grey Assignment Problem* (parameters being intervals), with *Grey Arithmetic* operations, operations on parameter matrices lead on the same principle to two problems (one on lower bounds, the other on upper bounds), each being solved by the Hungarian algorithm.

Similarly, these approaches are not compatible with the form of the objective function.

2.5 Links with fuzzy logic and probabilities

We chose to represent uncertainty with intervals, but other forms of uncertainty modeling also exist. For instance, a Fuzzy Assignment Problem can be turned into an interval one using the $\alpha - cut$ method [7].

More generally, the notion of *probability bounds analysis* (PBA) is introduced in [9], combining probability theory and interval arithmetic. A formalism is proposed in [17] where some parameters are described by intervals, while others are described by probability laws.

2.6 Application Contexts with Similar Structure

Minimizing a cost with such a multiplicative form is not specific to our crisis scenario. Several problem domains exhibit similar structural features, where entities are assigned to minimize uncertain risk or loss, or to maximize a benefit: assigning stimuli to brain regions to maximize human satisfaction [25], allocating budgets to maximize advertising reach [5], assigning treatments to minimize the probability of cancer damage [4], dispatching ambulances to emergency locations [11], and, as discussed, assigning weapons to targets to minimize residual threat.

2.7 Contribution

In this survey, we have shown that existing works focus on continuous problems or on simplified assignment problems, using indirect methods to compute the intervals.

Our conclusion is that Unbalanced Many-to-one by Teams NonLinear Assignment problem assimilable to WTA problem has not been solved using interval parameters.

As defined, WTA is a problem of constrained optimization. For discrete-continuous domains with interval parameters, interval-based Constraint Solving Problems (CSP) are generally solved using Constraint Programming (CP) techniques. For large and complex constraint instances, there are now effective techniques, such as SAT modulo theory (SMT) [10]. This would be relevant for problems with more complex constraints and could be useful for the search for counterexamples. For optimization problems, STM-Opt prototypes are known and are competitive, especially for combinatorial optimization problems. However, these prototypes do not seem to be publicly available yet. Moreover, note that the problem is potentially symmetric because parameters related to caregivers and wounded can coincide, thus defining equivalence classes. Symmetry could then reduce the search space, for example by integrating symmetry-breaking constraints [8]. This could be an interesting and original option for our problem (not developed here).

We therefore contribute with Algorithm 1, which directly computes the intervals (through the products of p_{ij}, the products of v_j by p_{ij}, the sums of s_j, and the comparison of upper bounds) for unbalanced and many-to-one assignments, with a multiplicative form of the objective function (a feature of WTA problem).

We have then implemented this algorithm in `C++` with the `Ibex` library [14]. In the next section, we analyse our approach, and study the variations of parameters (residual injuries and the upper bound of a wounded's initial injury level).

3 A Branch and Bound algorithm for Assignment problem with intervals parameters

To detail this contribution, this section presents the exact optimization Algorithm 1 used to explore all feasible caregiver-to-wounded assignments. The recursive `OPTIMIZE` function performs a depth-first Branch and Bound search. Equivalent to the assignment matrix introduced in Section 1, each component $A[i]$ is the wounded assigned to the caregiver i.

3.1 Branch and Bound Optimization function

The main `SOLVE` function takes as input the parameters of our problem: the number of caregivers m and wounded n, the intervals of the initial injury levels $[v_j] \in \mathbb{IR}$ gathered in the vector $\mathbf{V}$, the intervals of the treatment success probabilities $[p_{ij}] \in \mathbb{IR}$ gathered in the matrix $\mathbf{P}$, the parameters ($W^{\min}$, T, $W^{\max}$) associated with the constraints (C.3)–(C.4)-(C.5), a greedy heuristic first solution A_{gr} and its interval objective $[F_{\text{gr}}]$.

It calls once the recursive `OPTIMIZE` function with the current partial assignment A initialized as $A = [-1, -1, \ldots, -1]$ (meaning that no caregiver is yet assigned). At each recursion, if caregiver i has no valid wounded candidate, $A[i] = -2$. The

objective is to minimize the total residual injury interval, as defined in Equation (3) in Section 1.

Before calling main `SOLVE` function, a **greedy heuristic** is first used to build an initial solution A_{gr}. Caregivers $i \in \{1, \ldots, m\}$ are sorted according to the maximum amount of care they can potentially give $S_i = \sum_{j \in \mathcal{C}_i} \overline{[v_j]\,[p_{ij}]}$ among the admissible wounded $\mathcal{C}_i = \{\, j \mid x_{ij}$ valid according to (C.3) and (C.4) $\}$. Those with lower S_i (i.e. limited overall care potential) are considered first. For each wounded, the candidate $j^\star \in \mathcal{C}_i$ is then selected as the one minimizing

$$j^\star = \arg\min_{j \in \mathcal{C}_i} \overline{F(A \text{ s.t. } A[i] = j)} \tag{6}$$

and the resulting solution is considered only if it satisfies constraint (C.5). The initial greedy solution will serve as a first best assignment as a minimum comparison baseline (line 4) during the first call to the `OPTIMIZE` function.

At **each recursion**, the `OPTIMIZE` function takes as input the parameters of our problem, the current partial assignment A describing which caregivers have already been assigned, the best solution found so far and its interval objective $[F^*] = [\underline{F^*}, \overline{F^*}]$.

When an assignment is complete, the maximum residual injury constraint (C.5) is satisfied, and the upper bound $\overline{F}$ of the new solution is smaller than that of the current best solution (line 4), and then the best solution is updated (lines 2–3). The function then returns A^* and $[F^*]$, which ends the current branch of the search tree (line 28).

Otherwise, the **evaluation function** f computes interval **bounds** of the total residual injury that can still be reached from a partial assignment A. For each wounded j, the evaluation first gathers the contribution of the caregivers already assigned to j by multiplying their failed-treatment probabilities. It then extends this partial product by including the possible influence of all unassigned caregivers, which gives

$$[P_j] = [P_j^{\min}, P_j^{\max}] \times \prod_{i=-1} (1 - [p_{ij}]). \tag{7}$$

This interval represents all failed-treatment probabilities still compatible with the current partial assignment. This provides the residual injury intervals $[v_j] \times [P_j]$. If the test $\underline{[f(A)]} < \overline{[F^*]}$ succeeds, the partial assignment A can yet improve the best solution found so far. Then, the current branch is explored (lines 5-26). If it fails, the function then returns A^* and $[F^*]$, which ends the current branch of the search tree (line 28).

During this search (lines 5–26), at each recursion, the `OPTIMIZE` function selects the next caregiver following this heuristic (line 5). The caregiver, $i^\star$, having the lowest total potential treatment,

$$i^\star = \arg\min_{i=-1} \sum_j \overline{[v_j], [p_{ij}]}, \tag{8}$$

is selected first, and it ensures that caregivers with lower overall effectiveness are

Algorithm 1 An interval B&B algorithm for IWTAP.

Input: $\Sigma = (m, n, V, P, W^{\min}, T, W^{\max})$, A_{gr}, $[F_{\text{gr}}]$
Output: best assignment A^* and best objective value $[F^*]$

Function SOLVE($\Sigma, A_{\text{gr}}, [F_{\text{gr}}]$)
if $A_{\text{gr}} \neq \perp$ **then**
  return OPTIMIZE($\Sigma, [-1, \ldots, -1], A_{\text{gr}}, [F_{\text{gr}}]$)
else
  return OPTIMIZE($\Sigma, [-1, \ldots, -1], [-1, \ldots, -1], [+\infty, +\infty]$)
end if

Function OPTIMIZE($\Sigma, A, A^*, [F^*]$)
if (A is complete **and** (C.5) satisfied **and** $\overline{[F(A)]} < \overline{[F^*]}$ **then**
  $A^* \leftarrow A$
  $[F^*] \leftarrow [F(A)]$
else if $\underline{[f(A)]} < \overline{[F^*]}$ **then**
  Choose a caregiver following Heuristic (8)
  if no such caregiver exists **then**
    return $(A^*, [F^*])$
  end if
  Wounded-Candidates $\leftarrow \emptyset$
  for all wounded j **do**
    if (C.3) and (C.4) are satisfied **then**
      Wounded-Candidates $\leftarrow$ *Wounded-Candidates* $\cup \{j\}$
    end if
  end for
  if *Wounded-Candidates* $= \emptyset$ **then**
    $A[i^\star] = -2$ {caregiver non-assignment}
    $(A^*, [F^*]) \leftarrow$ OPTIMIZE($\Sigma, A, A^*, [F^*]$)
    $A[i^\star] = -1$ {backtracking: caregiver cancel}
  end if
  Sort *Wounded-Candidates* according to Heuristic (6)
  for all $j \in$ *Wounded-Candidates* **do**
    $A[i^\star] = j$ {caregiver assignment}
    Update team assigned to wounded j
    $(A^*, [F^*]) \leftarrow$ OPTIMIZE($\Sigma, A, A^*, [F^*]$)
    $A[i^\star] = -1$ {backtracking: caregiver cancel}
  end for
end if
return $(A^*, [F^*])$

processed before the most capable ones (this selection order was empirically found to give the best performance).

Candidate wounded are then generated for that caregiver (lines 9-14), restricted to admissible pairs and sorted according to the Heuristic (6) and its same order (line 20).

Each candidate is then tried **recursively** (lines 21-26). The algorithm assigns wounded j to caregiver i, explores the resulting subproblem through a recursive call (line 24), and then backtracks by cancelling this caregiver assignment (line 25). If a caregiver has no valid wounded candidate (i.e. $C = \emptyset$), it is explicitly marked as $A[i] = -2$ (line 16) and, similarly, the recursion continues (line 17).

Overall, this exhaustive function **evaluates all feasible solutions**, and keeps the one with the best objective value.

3.2 Order relation and pruning strategy

In our Branch and Bound algorithm, all operations and comparisons are performed with interval arithmetic. Each assignment therefore generates an objective interval. Complete solutions are compared to each other using their upper bounds (or, as a secondary criterion, their interval widths), as explained in 3.1. In particular, when $\alpha = 1$, optimization is driven by the worst-case scenario.

However, the evaluation of a partial branch cannot be treated in the same way. A branch represents a set of possible solutions, not a single computed solution. To ensure that no potentially better solution is lost, the pruning test must guarantee that a branch is never discarded if it could contain an improvement. For this reason, we compare the upper bound of the best solution already found with the lower bound attainable by the new branch (corresponding to the best case still possible within that branch), as performed in line 4.

Bounding and pruning are intended to accelerate the search. Yet, in this implementation, exactness is deliberately prioritized over execution speed. This pruning strategy is therefore less aggressive, but it guarantees correctness.

4 Application and impact of uncertainty

4.1 Use case with $m = 36$ and $n = 18$

We consider a use case with $m = 36$ caregivers and $n = 18$ wounded individuals. Both the initial injury levels $[v_j] = [\underline{v_j}, \overline{v_j}] \in \mathbb{IR}$ and the treatment success probabilities $[p_{ij}] = [\underline{p_{ij}}, \overline{p_{ij}}] \in \mathbb{IR}$ are uncertain and therefore modeled as intervals. The underlying estimated initial injury levels v_j are:

$$[7,\ 6,\ 9,\ 6,\ 9,\ 7,\ 6,\ 9,\ 7,\ 9,\ 7,\ 6,\ 6,\ 9,\ 7,\ 6,\ 6,\ 7,\ 9,\ 7,\ 6,\ 6,\ 9,\ 7].$$

Each v_j is considered as an interval with an asymmetric uncertainty margin around its value: $[v_j] = [v_j - 0.10,\ v_j + 0.25]$.

Similarly, each p_{ij} is represented as an interval of fixed width 0.1: $[p_{ij}] = [p_{ij} - 0.05,\ p_{ij} + 0.05]$. The set of all p_{ij} values is represented in Table 1, where we report the midpoints of the intervals for p_{ij}. For example, $[p_{11}] = [0.75, 0.85]$ is represented by its midpoint, 0.8.

Table 1: Estimated treatment success probabilities p_{ij} for $i = 1, \ldots, 36$

	j=1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
i=1	0.8	0.3	0.2	0.8	0.1	0.2	0.2	0.5	0.8	0.8	0.3	0.5	0.1	0.4	0.5	0.4	0.8	0.5
i=2	0.2	0.2	0.1	0.4	0.1	0.8	0.1	0.4	0.8	0.5	0.4	0.1	0.4	0.4	0.1	0.3	0.5	0.1
i=3	0.3	0.4	0.4	0.3	0.2	0.8	0.3	0.3	0.1	0.5	0.3	0.1	0.1	0.1	0.5	0.2	0.1	0.5
i=4	0.2	0.8	0.3	0.3	0.2	0.8	0.1	0.4	0.3	0.1	0.1	0.4	0.1	0.4	0.8	0.1	0.3	0.1
i=5	0.2	0.8	0.3	0.1	0.4	0.8	0.8	0.5	0.1	0.8	0.2	0.4	0.3	0.8	0.5	0.2	0.3	0.8
i=6	0.3	0.4	0.8	0.5	0.1	0.4	0.8	0.1	0.8	0.8	0.1	0.2	0.5	0.4	0.8	0.5	0.5	0.2
i=7	0.5	0.3	0.2	0.8	0.8	0.8	0.4	0.5	0.8	0.2	0.1	0.1	0.5	0.2	0.2	0.5	0.4	0.2
i=8	0.4	0.4	0.8	0.5	0.1	0.8	0.3	0.1	0.1	0.3	0.1	0.8	0.5	0.1	0.4	0.8	0.4	0.4
i=9	0.1	0.8	0.8	0.1	0.5	0.1	0.8	0.3	0.1	0.4	0.8	0.1	0.4	0.1	0.1	0.2	0.4	0.2
i=10	0.2	0.4	0.1	0.2	0.4	0.1	0.5	0.4	0.2	0.3	0.8	0.8	0.3	0.5	0.8	0.3	0.1	0.1
i=11	0.3	0.1	0.8	0.4	0.5	0.1	0.5	0.4	0.4	0.5	0.4	0.4	0.1	0.4	0.8	0.2	0.2	0.1
i=12	0.5	0.4	0.5	0.5	0.1	0.3	0.4	0.1	0.3	0.8	0.2	0.4	0.4	0.4	0.3	0.8	0.3	0.3
i=13	0.2	0.4	0.4	0.8	0.5	0.2	0.1	0.8	0.5	0.4	0.1	0.4	0.8	0.3	0.3	0.1	0.5	0.2
i=14	0.8	0.4	0.8	0.1	0.2	0.1	0.5	0.1	0.8	0.3	0.1	0.1	0.2	0.2	0.1	0.8	0.3	0.2
i=15	0.4	0.3	0.1	0.4	0.5	0.3	0.2	0.3	0.2	0.3	0.3	0.2	0.8	0.5	0.4	0.3	0.4	0.2
i=16	0.1	0.3	0.8	0.3	0.3	0.5	0.2	0.8	0.4	0.3	0.1	0.5	0.2	0.8	0.4	0.2	0.4	0.2
i=17	0.1	0.2	0.2	0.4	0.3	0.5	0.3	0.2	0.5	0.1	0.1	0.4	0.5	0.8	0.1	0.1	0.5	0.8
i=18	0.5	0.3	0.3	0.4	0.4	0.2	0.2	0.3	0.1	0.4	0.8	0.1	0.3	0.1	0.2	0.8	0.4	0.3
i=19	0.8	0.5	0.1	0.1	0.3	0.8	0.5	0.5	0.4	0.2	0.8	0.8	0.8	0.4	0.2	0.5	0.2	0.5
i=20	0.1	0.4	0.5	0.3	0.5	0.8	0.2	0.4	0.8	0.1	0.1	0.3	0.8	0.5	0.1	0.1	0.4	0.8
i=21	0.4	0.4	0.4	0.4	0.3	0.4	0.8	0.5	0.4	0.1	0.1	0.8	0.8	0.5	0.3	0.1	0.1	0.3
i=22	0.8	0.3	0.2	0.8	0.3	0.1	0.2	0.3	0.3	0.5	0.5	0.4	0.8	0.5	0.4	0.2	0.8	0.8
i=23	0.1	0.5	0.2	0.5	0.1	0.4	0.2	0.8	0.5	0.8	0.4	0.8	0.5	0.4	0.5	0.8	0.8	0.2
i=24	0.2	0.2	0.5	0.3	0.4	0.2	0.1	0.5	0.2	0.4	0.3	0.5	0.2	0.2	0.1	0.2	0.2	0.2
i=25	0.8	0.2	0.2	0.8	0.4	0.5	0.1	0.5	0.5	0.3	0.3	0.4	0.1	0.2	0.2	0.1	0.8	0.2
i=26	0.1	0.4	0.5	0.4	0.4	0.8	0.4	0.1	0.4	0.2	0.5	0.2	0.8	0.1	0.8	0.1	0.3	0.5
i=27	0.3	0.8	0.4	0.1	0.3	0.2	0.5	0.1	0.8	0.3	0.8	0.1	0.8	0.5	0.1	0.2	0.8	0.4
i=28	0.5	0.8	0.8	0.2	0.2	0.1	0.3	0.8	0.4	0.3	0.8	0.8	0.5	0.3	0.1	0.1	0.2	0.1
i=29	0.2	0.2	0.4	0.1	0.3	0.8	0.1	0.2	0.5	0.4	0.3	0.1	0.5	0.3	0.5	0.5	0.4	0.4
i=30	0.5	0.8	0.8	0.5	0.1	0.2	0.5	0.1	0.3	0.1	0.1	0.4	0.5	0.1	0.1	0.3	0.1	0.5
i=31	0.4	0.3	0.1	0.8	0.3	0.4	0.2	0.5	0.5	0.2	0.3	0.4	0.1	0.3	0.1	0.2	0.2	0.5
i=32	0.2	0.2	0.5	0.3	0.3	0.2	0.1	0.5	0.1	0.8	0.3	0.1	0.4	0.5	0.8	0.2	0.8	0.5
i=33	0.2	0.1	0.1	0.8	0.1	0.1	0.8	0.8	0.4	0.5	0.8	0.3	0.4	0.5	0.8	0.1	0.2	0.4
i=34	0.1	0.1	0.3	0.5	0.1	0.8	0.2	0.8	0.2	0.5	0.1	0.2	0.5	0.5	0.2	0.5	0.8	0.8
i=35	0.4	0.2	0.3	0.3	0.2	0.8	0.4	0.2	0.5	0.2	0.5	0.5	0.3	0.1	0.8	0.4	0.2	0.5
i=36	0.4	0.5	0.8	0.2	0.1	0.2	0.8	0.1	0.4	0.2	0.3	0.4	0.2	0.1	0.4	0.2	0.8	0.3

The objective of our optimization problem is to minimize the overall residual injury of the wounded,

$$\arg \min_{x_{ij} \in \{0,1\}} \sum_{j=1}^{18} [v_j] \cdot \prod_{i=1}^{36} (1 - [p_{ij}])^{x_{ij}}\,, \tag{9}$$

subject to constraints (C.1)–(C.5), with $w_j^{\min} = 4.0$ and $w_j^{\max} = 5.0$.

Our problem was implemented in `C++`, using the `Ibex` library to handle interval arithmetic, compute rigorous bounds on the objective function, and return the solutions as interval-valued residual injuries.

4.2 Output Representations

The implementation produces four equivalent forms of output, each offering a different perspective on the same solution: a human-readable textual representation of the best assignment by team in Table 2, a matrix representation of the decision variables x_{ij} in Table 3, an m-tuple where the i-th entry indicates the index of the wounded person assigned to caregiver i or -1 if caregiver i is unassigned in Table 4, and possibly an assignment graph (a bipartite graph with caregivers on one side and the wounded on the other). For example, caregiver 03 is assigned to wounded 08: there is a 1 in the corresponding entry of Table 3, and for $i = 4$ there is an 8 in Table 4.

Table 2: Best assignment by team

Caregiver	Wounded
00	13
01	09
02	09
03	08

Team t_1.

Caregiver	Wounded
04	∅
05	∅
06	12
07	12

Team t_2.

Caregiver	Wounded
08	∅
09	01
10	∅
11	01

Team t_3.

Caregiver	Wounded
12	11
13	14
14	11
15	11

Team t_4.

Caregiver	Wounded
16	05
17	16
18	05
19	17

Team t_5.

Caregiver	Wounded
20	∅
21	∅
22	15
23	06

Team t_6.

Caregiver	Wounded
24	03
25	03
26	10
27	10

Team t_7.

Caregiver	Wounded
28	∅
29	07
30	07
31	00

Team t_8.

Caregiver	Wounded
32	∅
33	04
34	04
35	∅

Team t_9.

4.3 Residual Injury Intervals

An example output of the algorithm is illustrated in Figure 4. The uncertainties in resulting injuries (interval widths) are computed using interval arithmetic based on input uncertainties. The initial injury intervals are shown as green segments.

As expected, the residual injury intervals (blue segments in the figure) are all upper-bounded by the maximum injury constraint (C.5) (red dots), meaning that the wounded are sufficiently treated. The overall residual injury defined in Equation (9) has been globally minimized.

Table 3: Assignment matrix (first 10 caregivers × first 10 wounded). CXX = caregiver, WYY = wounded.

	W00	W01	W02	W03	W04	W05	W06	W07	W08	W09
C00	0	0	0	0	0	0	0	0	0	0
C01	0	0	0	0	0	0	0	0	0	**1**
C02	0	0	0	0	0	0	0	0	0	**1**
C03	0	0	0	0	0	0	0	0	**1**	0
C04	0	0	0	0	0	0	0	0	0	0
C05	0	0	0	0	0	0	0	0	0	0
C06	0	0	0	0	0	0	0	0	0	0
C07	0	0	0	0	0	0	0	0	0	0
C08	0	0	0	0	0	0	0	0	0	0
C09	0	**1**	0	0	0	0	0	0	0	0

Table 4: Assignment as an m-tuple.

13, 9, 9, 8, ∅, ∅, 12, 12, ∅, 1, ∅, 1, 11, 14, 11, 11, 5, 16, 5, 17, ...

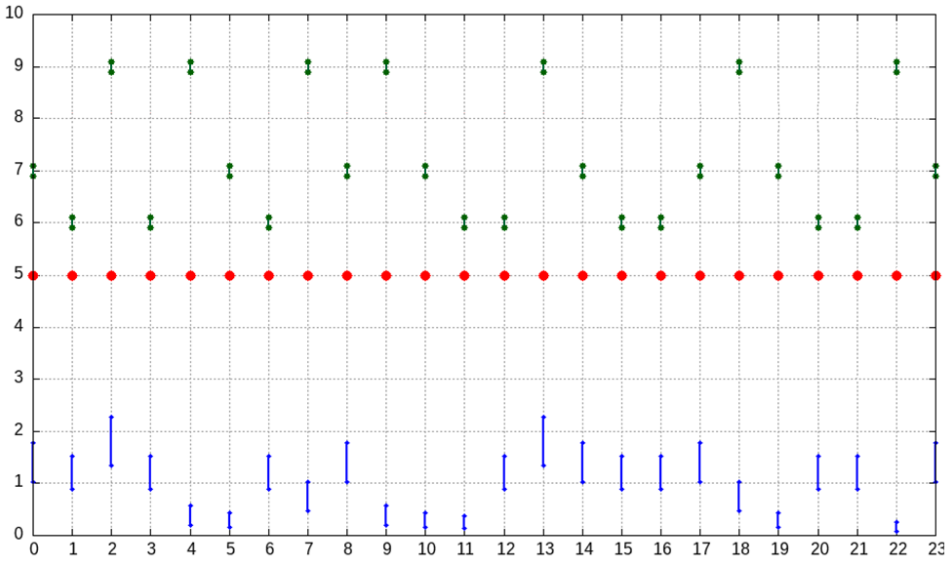


Figure 4: Initial injury intervals (green segments), residual injury intervals (blue segments), maximum injury constraints (red dots).

4.4 Impact of uncertainty about one wounded

We further examined how the pessimistic estimate of an initial injury level (represented by the worst-case bound $\overline{v_j}$) could influence caregiver assignments. To explore this, for $m = 12$ and $n = 12$, and with an initial value $\overline{v_j} = 0$, we focused on wounded person 2 and progressively increased its pessimistic bound $\overline{v_j}$ by a given step δ. In Figure 5, we show the variations of $\overline{v_j}$ (left), $\overline{v_j} + \delta$ with δ ranging from 0 to 58, as well as the resulting assignments (right).

As expected, wounded 2 was **selected more frequently** (in green) as the uncertainty increased: once from the extra margin $\delta = 2$, twice from 10, three times from 34, and four times from 57. The thresholds where an additional assignment occurred are highlighted in orange.

Interestingly, at some uncertainty values (e.g., in blue: 34, 35, 55, and 56), the total number of assignments (here 3) remained unchanged, although the specific caregivers involved differed. This suggests that the resolution can **switch between alternative solutions** without increasing the overall caregiver effort.

To illustrate this effect, Figure 6 shows the worst-case residual injury (y-axis, in green) and the best-case residual injury (y-axis, in purple) as a function of the worst-case initial injury for wounded 2 (x-axis).

As anticipated, the addition of a caregiver reduces the worst-case injury (i.e., the thresholds discussed earlier).

However, between such transitions, increasing uncertainty does not trigger new assignments, which leads to a gradual rise in the worst-case residual injury. Eventually, a saturation effect occurs: although the worst-case injury keeps worsening, the number of caregivers allocated to wounded 2 begins to decline, reflecting the limitation of caregiver resources.

The lower bound of the injuries (in purple), on the other hand, decreases after each assignment and does not increase between assignments. This behavior is expected, since the lower bound is never increased, while each assignment directly contributes to reducing its value.

4.5 Execution times and parallelism

We present the execution times as a function of the variables m (caregivers) and n (wounded), examine the algorithmic complexity, and study the efficiency of the parallelization.

4.5.1 Execution times

In the 3D plot of Figure 7, execution times of our algorithm are shown as a function of m (right-hand axis) and n (left-hand axis). We observe that execution times increase significantly with both parameters, as expected due to the combinatorial nature of our assignment problem. However, the growth rate is greater with respect to m, indicating that the algorithm becomes harder to solve when the number of caregivers increases.

δ	**Assignments as 12-tuple**
0	$(\emptyset, 5, 9, 8, 7, 9,\ \ 5, 11, 3, 1, 10, 4)$
1	$(\emptyset, 5, 9, 8, 7, 9,\ \ 5, 11, 3, 1, 10, 4)$
2	$(2, 5, 9, 8, 7, 9,\ \ 5, 11, 3, 1, 10, 4)$
3	$(2, 5, 9, 8, 7, 9,\ \ 5, 11, 3, 1, 10, 4)$
...	...
8	$(2, 5, 9, 8, 7, 9,\ \ 5, 11, 3, 1, 10, 4)$
9	$(2, 5, 9, 8, 7, 9,\ \ 5, 11, 3, 1, 10, 4)$
10	$(4, 5, 9, 8, 7, 9, 10, 11, 3, 1,\ \ 2, 2)$
11	$(4, 5, 9, 8, 7, 9, 10, 11, 3, 1,\ \ 2, 2)$
...	...
23	$(4, 5, 9, 8, 7, 9, 10, 11, 3, 1,\ \ 2, 2)$
24	$(4, 5, 9, 8, 7, 9, 10, 11, 3, 1,\ \ 2, 2)$
25	$(2, 5, 9, 2, 7, 9,\ \ 5, 11, 3, 1, 10, 4)$
26	$(2, 5, 9, 2, 7, 9,\ \ 5, 11, 3, 1, 10, 4)$
...	...
33	$(2, 5, 9, 2, 7, 9,\ \ 5, 11, 3, 1, 10, 4)$
34	$(2, 5, 9, 2, 7, 2,\ \ 5, 11, 3, 1, 10, 4)$
35	$(2, 5, 2, 2, 7, 9,\ \ 5, 11, 3, 1, 10, 4)$
...	...
55	$(2, 5, 2, 2, 7, 9,\ \ 5, 11, 3, 1, 10, 4)$
56	$(2, 5, 9, 2, 7, 2,\ \ 5, 11, 3, 1, 10, 4)$
57	$(2, 2, 2, 2, 7, 9,\ \ 5, 11, 3, 1, 10, 4)$
58	$(2, 2, 2, 2, 7, 9,\ \ 5, 11, 3, 1, 10, 4)$

Figure 5: Evolution of assignments A as a function of δ

To further illustrate this trend, Figure 8 shows 2D execution curves for several fixed values of m, as a function of n. We observe that execution times grow faster with n when m is large. Again, this is consistent with the exponential increase in the size of the search space when both dimensions grow.

Figure 9 shows two theoretical curves plotted to frame the measurements: a lower bound proportional to $m\,n$, and an upper bound proportional to $(n+1)^m$. These curves provide an **empirical enclosure** of the complexity, since all measured points lie between them. This indicates that the computation becomes significantly harder as m increases, with a growth lying between linear and exponential.

4.5.2 Reduced by parallelism

To leverage multicore embedded architectures for real-time decision-making, we parallelized the exploration by distributing the first-level assignment of the initial caregiver across multiple threads. Each thread independently explores a different

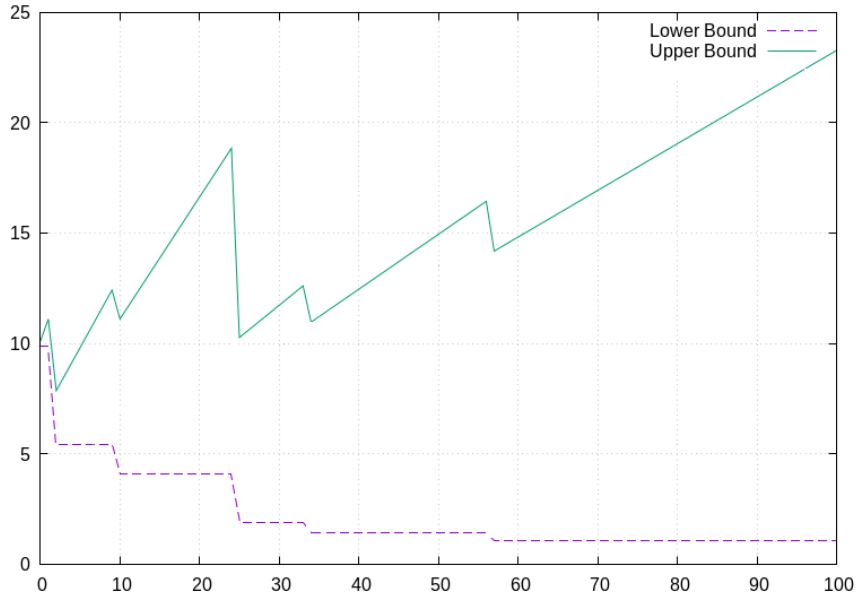


Figure 6: Worst-case residual injury vs uncertainty on wounded person 2

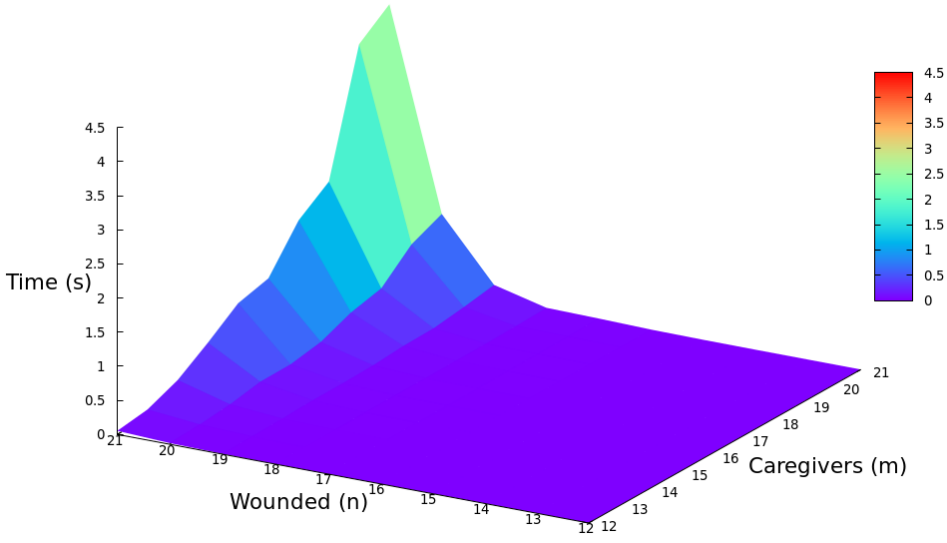


Figure 7: Execution time (in seconds) as a function of the number of caregivers m and wounded n

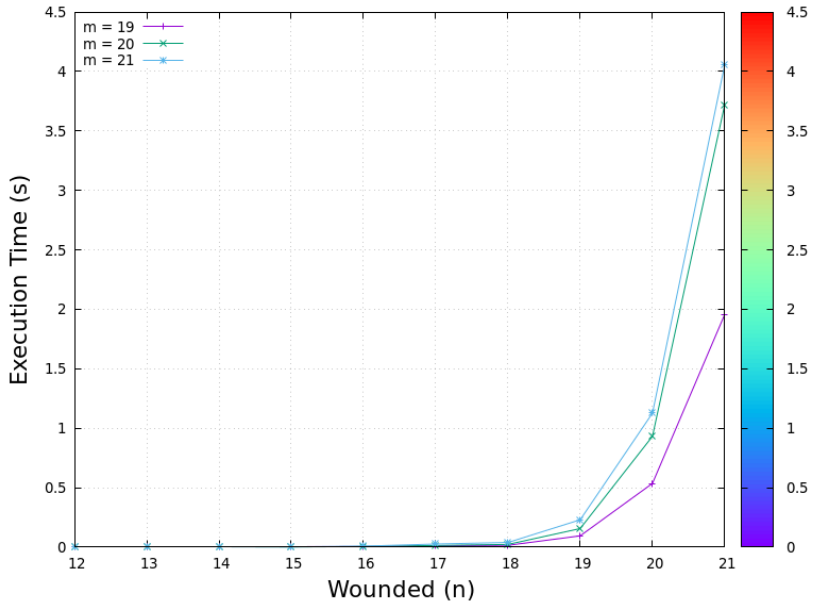


Figure 8: Execution time vs number of wounded n, for three different numbers of caregivers m

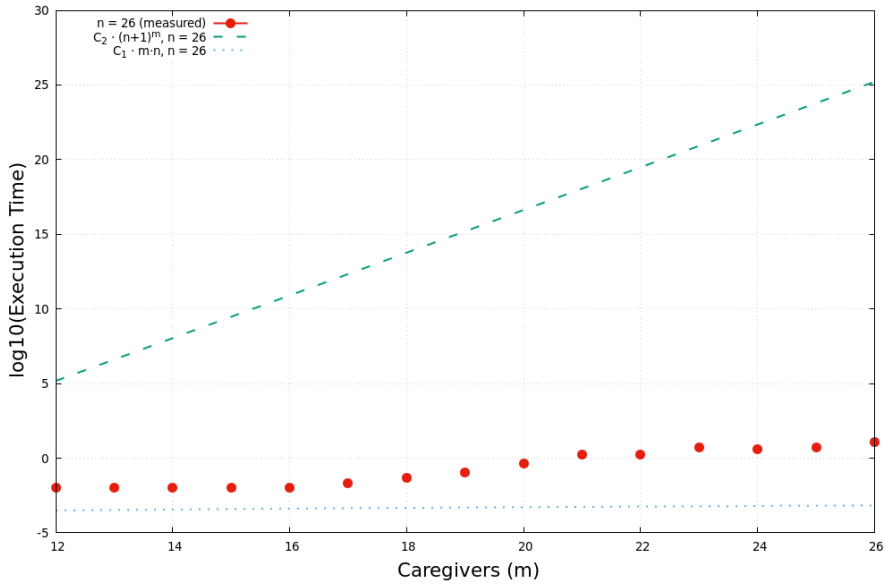


Figure 9: Empirical complexity vs number of caregivers m

initial choice, while the remaining recursive search proceeds sequentially within the thread. This preserves the structure of the algorithm while enabling substantial speedups under strict timing constraints. Figure 10 illustrates the performance gains from this parallelization on an instance with m=36 caregivers and n=18 wounded. On the Figure 10, execution time is shown as a function of the number of cores. It decreases significantly up to 8 cores, then more gradually up to 16 cores. While the execution times decreases with the number of cores, it eventually saturates. This is expected, as the deeper levels of the recursive search rely on sequential backtracking, bounding, and constraint checks that are harder to parallelize efficiently.

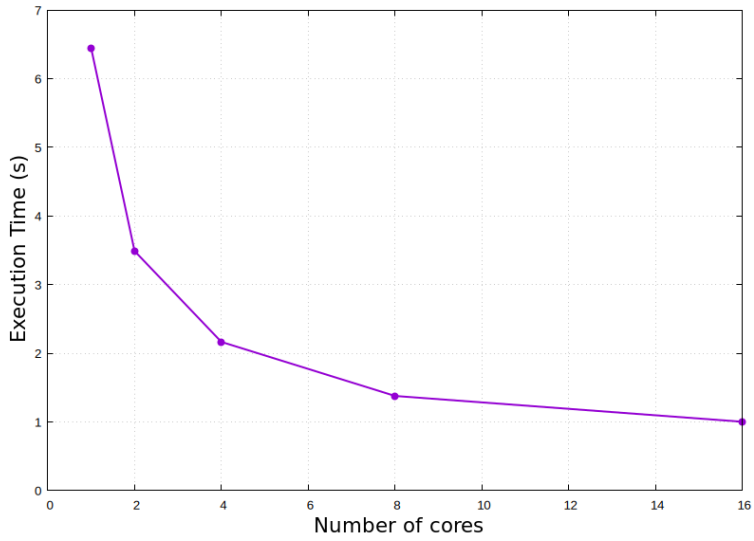


Figure 10: Execution time vs number of cores

5 Conclusion and future works

In this paper, we address a humanitarian scenario in which caregivers are assigned to the wounded, and we show how it corresponds to an instance of the IWTAP. Our problem is characterized by being **unbalanced**, **many-to-one**, with a **multiplicative objective function** defined under interval uncertainty. The state of the art showed that this type of problem had not yet been solved. We solved it using a parallelized Branch and Bound algorithm that directly manipulates the intervals. In an example of resolving this problem, we showed that we ensure a rigorous propagation of uncertainty over the parameters in residual injury intervals. The analysis also highlighted the impact of increasing pessimism on a single wounded individual's injury level, and how uncertainty may trigger changes in assignments.

As future work, we aim to investigate the interdependence between planning and assignment. For example, caregiver movements may be linked to assignments, as illustrated in Figure 11. The process depends on an initial state, which itself results from a planning phase, while planning is in turn influenced by assignment outcomes. This creates a loop of mutual dependence, further complicated by uncertainty. This interplay will be explored in the context of planning under uncertainty, integrating Hierarchical Task Network planning, assignment, and uncertainty into a coherent decision-making framework.

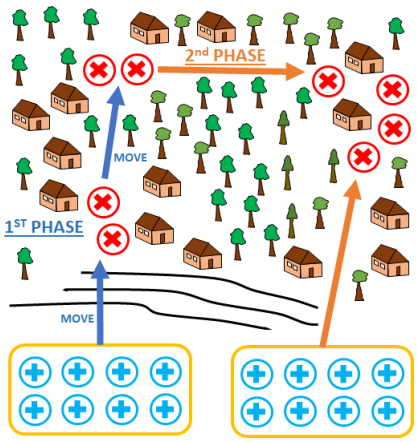


Figure 11: Planning and assignments

References

[1] Achenie, L. E. K. and Sinha, M. Interval global optimization in solvent design. *Reliable Computing*, 9(5):317–338, 2003. DOI: 10.1023/A:1025158512652.

[2] Buvaneshwari, T. and Anuradha, D. Solving bi-objective interval assignment problem using genetic approach. *Advances in Mathematics: Scientific Journal*, 10:759–768, 2021. DOI: 10.37418/amsj.10.2.7.

[3] Buvaneshwari, T. and Anuradha, D. On solving bi-objective interval valued neutrosophic assignment problem. *Neutrosophic Sets and Systems*, 64:212–229, 2024. DOI: 10.5281/zenodo.10729919.

[4] Çetin, E. A queuing theoretical model for anticancer tool selection. In *International Mathematical Forum*, Volume 2-54, pages 2675–2685, 2007. DOI: 10.12988/imf.2007.07238.

[5] Cetin, E. and Esen, S. T. A weapon–target assignment approach to media allocation. *Applied Mathematics and Computation*, 175(2):1266–1275, 2006. DOI: 10.1016/j.amc.2005.08.041.

[6] Chenouard, R. and El-Sehiemy, R. A. An interval branch and bound global optimization algorithm for parameter estimation of three photovoltaic models. *Energy Conversion and Management*, 205:112400, 2020. DOI: 10.1016/j.enconman.2019.112400.

[7] Elsisy, M. A., Elsaadany, A. S., and El Sayed, M. A. Using interval operations in the hungarian method to solve the fuzzy assignment problem and its application in the rehabilitation problem of valuable buildings in egypt. *Complexity*, 2020(1):9207650, 2020. DOI: 10.1155/2020/9207650.

[8] Fahle, T., Schamberger, S., and Sellmann, M. Symmetry breaking. In *International Conference on Principles and Practice of Constraint Programming*, pages 93–107. Springer, 2001. DOI: 10.1007/3-540-45578-7_7.

[9] Ferson, S. and Ginzburg, L. R. Different methods are needed to propagate ignorance and variability. *Reliability Engineering & System Safety*, 54(2-3):133–144, 1996. DOI: 10.1016/S0951-8320(96)00071-3.

[10] Fränzle, M., Herde, C., Teige, T., Ratschan, S., and Schubert, T. Efficient solving of large non-linear arithmetic constraint systems with complex boolean structure. *Journal on Satisfiability, Boolean Modelling and Computation*, 1(3-4):209–236, 2006. DOI: DOI:10.3233/SAT190012.

[11] Gelenbe, E., Timotheou, S., and Nicholson, D. Fast distributed near-optimum assignment of assets to tasks. *The Computer Journal*, 53(9):1360–1369, 2010. DOI: 10.1016/j.cor.2016.07.005.

[12] Gong, D.-w. and Zhang, Y. Particle swarm optimisation for multi-project location problems with interval profits. *International Journal of Modelling, Identification and Control*, 8(4):335–343, 2009. URL: https://www.inderscienceonline.com/doi/abs/10.1504/IJMIC.2009.03008.

[13] Hansen, E. and Walster, G. W. *Global optimization using interval analysis: revised and expanded*, Volume 264. CRC press, 2003. DOI: 10.1604/9780824740597.

[14] IBEX Team. IBEX, 2007-2020. URL: https://ibex-team.github.io/ibex-lib/.

[15] Ishibuchi, H. and Tanaka, H. Multiobjective programming in optimization of the interval objective function. *European Journal of Operational Research*, 48(2):219–225, 1990. DOI: 10.1016/0377-2217(90)90375-L.

[16] Jiang, C., Han, X., Liu, G. R., and Liu, G. A nonlinear interval number programming method for uncertain optimization problems. *European Journal of Operational Research*, 188(1):1–13, 2008. DOI: 10.1016/j.ejor.2007.03.031.

[17] Jiang, C., Zheng, J., and Han, X. Probability-interval hybrid uncertainty analysis for structures with both aleatory and epistemic uncertainties: A review. *Structural and Multidisciplinary Optimization*, 57(6):2485–2502, 2018. DOI: 10.1007/s00158-017-1864-4.

[18] Kagade, K. and Bajaj, V. Fuzzy method for solving multi-objective assignment problem with interval cost. *Journal of Statistics and Mathematics*, 1(1):1, 2010. URL: https://bioinfopublication.org/pages/article.php?id=BIA0001583.

[19] Kreinovich, V. Decision making under interval uncertainty (and beyond). In *Human-centric decision-making models for social sciences*, pages 163–193. Springer, 2013. DOI: 10.1007/978-3-642-39307-5_8.

[20] Majumdar, S. Interval linear assignment problems. *Universal Journal of Applied Mathematics*, 1(1):14–16, 2013. DOI: 10.13189/ujam.2013.010103.

[21] Manne, A. S. A target-assignment problem. *Operations Research*, 6(3):346–351, 1958. DOI: 10.1287/opre.6.3.346.

[22] Moore, R. E., Kearfott, R. B., and Cloud, M. J. *Introduction to interval analysis*. SIAM, 2009. DOI: 10.2307/40590421.

[23] Naidoo, S. Applying military techniques used in threat evaluation and weapon assignment to resource allocation for emergency response — A literature survey, 2008. Presented as Seminar for Operations Research Society of South Africa. URL: https://www.researchgate.net/publication/228356931_Applying_Military_Techniques_used_in_Threat_Evaluation_and_Weapon_Assignment_to_Resource_Allocation_for_Emergency_Response-A_Literature_Survey.

[24] Nasseri, H., Darvishi Salokolaei, D., and Yazdani, A. A new approach for solving grey assignment problems. *Control and Optimization in Applied Mathematics*, 2(1):15–28, 2017. URL: https://mathco.journals.pnu.ac.ir/article_4820.html.

[25] Onay, O. A mathematical approach to neuromarketing: A weapon–target assignment model. *International Journal of Academic Research in Business and Social Sciences*, 6(1):164–173, 2016. DOI: 10.6007/IJARBSS/v6-i1/1986.

[26] Ramesh, G. and Ganesan, K. Assignment problem with generalized interval arithmetic. *International Journal of Scientific and Engineering Research*, 6(3):82–85, 2015.

[27] Ramesh, G., Sudha, G., and Ganesan, K. Method of finding an optimal solution for interval balanced and unbalanced assignment problem. In *IOP Conference Series: Materials Science and Engineering*, Volume 912-6, page 062031. IOP Publishing, 2020. DOI: 10.1088/1757-899X/912/6/062031.

[28] Salehi, K. An approach for solving multi-objective assignment problem with interval parameters. *Management Science Letters*, 4(9):2155–2160, 2014. DOI: 10.5267/j.msl.2014.7.031.

[29] Sinha, M., Achenie, L. E. K., and Gani, R. Blanket wash solvent blend design using interval analysis. *Industrial & Engineering Chemistry Research*, 42(3):516–527, 2003. DOI: 10.1016/B978-0-323-95931-5.00006-3.

Acta Cybernetica **27** (2026) 293–311.

Optimal Local Path Planner Over Receding Horizon Using Open Interval B-Spline*

Lucas Si Larbi[abc], Eric Lucet[ad], and Julien Alexandre dit Sandretto[be]

Abstract

A local path planning algorithm aims to provide a global, deterministic and safe solution for the dynamic navigation of wheeled robots with limited visibility. To this end, a novel approach based on open interval B-spline curves computed over a receding horizon is exposed in this paper. Real-time performances are ensured by using an interval branch and bound algorithm. The resulting path is smooth, obstacle-avoidant, and continuously connects local paths without requiring additional computation. Furthermore, this approach offers a guaranteed understanding of the solution's state. A large set of simulations, adapted for a wheeled differential robot on several scenarios, is finally carried out to assess parameters impact and performances.

Keywords: mobile robot, local path planner, interval B-Spline, non-linear problem, global optimization

1 Introduction

Autonomous off-road navigation of wheeled mobile robots requires rigorous control adapted to the robot's local environment. From classical methods such as sampling-based methods [21] or velocity-based methods [7]; to learning approaches such as genetic algorithms [13] or end-to-end machine learning methods [3], various approaches have been proposed in literature applied to autonomous navigation on uneven terrains. Main requirements in this context are: (i) The ability to consider multiple constraints such as geometrical constraints (*e.g.* curvature), or spatial constraints (*e.g.* avoid obstacles); (ii) The possibility to optimize the local path

*This work was carried out in the scope of REFLEX project, as part of the MOBILEX Challenge. This project received funding from the French Defense Innovation Agency (AID) and the French National Research Agency (ANR) and, in partnership with the French National Center for Space Studies (CNES) and the French Agency for Transport Innovation (AIT).

[a]Université Paris Saclay, CEA, List, F-91120 Palaiseau, France
[b]U2IS, ENSTA Paris, Institut Polytechnique de Paris, 91120 Palaiseau, France
[c]E-mail: lucas.silarbi@cea.fr, ORCID: 0009-0004-6045-9259
[d]E-mail: eric.lucet@cea.fr, ORCID: 0000-0002-9702-3473
[e]E-mail: julien.alexandre-dit-sandretto@ensta-paris.fr, ORCID: 0000-0002-6185-2480

DOI: 10.14232/actacyb.315468

with regard to the robot's local environment; (iii) The ability to guarantee an output solution, when one exists, or else the information of its non-existence; (iv) Finally, the ability to control the computation time to deal with real-time performances. In this context, and because of their rigorous properties, interval global optimization-based methods [9] appear to be a suitable choice [26]. Local planning usually comes within a more general architecture involving global planning and tracking control. Global planning aims to join the starting point with the end point, a goal, using the shortest path which often doesn't respect any particular constraint except avoiding *a priori* known obstacles. The role of tracking control is to generate motor commands in order to follow the solution generated by the motion planner. In many applications, a simpler architecture, only including a global planning level with a high refresh rate (A* [8], Rapidly exploring Random Tree [21]) and a tracking control level (Pure Pursuit [18], Model-based Predictive Controller [17]) is sufficient to obtain acceptable results. However, in unstructured environments, the gap between global planning and tracking control is important. Local planning level is therefore essential to generate a more complete solution over a finite horizon around the robot. This solution can take several forms: a path; a trajectory; a motion direction; or a velocity. Unlike the trajectory, a path is a succession of points that are not temporally linked. A robot following a path is then able to change its velocity, to stop and even reverse without having to regenerate it. This can be interesting in complex situations. This article is therefore focusing on path generation.

1.1 Related Work and Motivation

Some previous studies already considered B-Spline Curves (BSCs) in the generation of an optimized trajectory over a receding horizon [6, 19]. B-Spline Curves where chosen as the solution support due to their properties: a local modification; a definition of the entire curve only with several control points; a setting of the degree of continuity of the curve, and therefore the smoothness of the trajectory generated. The local planning problem has been formalized as a Numerical Constrained global Optimization Problem (NCOP). Sequential least squares quadratic programming [14] were used to find a solution of such problems. Two main issues were raised: the inability to guarantee a solution if one exists; and the inability to control computation time and therefore to ensure real-time performance. In this article, we focus on finding the right solver for our particular problem. There are several types of solver, in particular those based on artificial intelligence. However, these solvers do not address the previous issues. Also, interval methods applied to constrained global optimization allows one to find the global optimum and provide bounds on its value and location [9]. Moreover, interval branch and bound optimizers [2] can prematurely stop the optimization and provide a sub-optimal solution which respects constraints and therefore ensure real-time performance. Path planners using interval methods are already widely developed [1, 12]. The path planner proposed in this article addresses the real-time requirement for local motion planning. As previous studies have shown, reducing the search space is often essential

to limit the convergence time. In our approach, instead of using only the robot model as constraints to define the search space, we propose an adapted solution support. BSCs appear to be interesting for their properties. However, they are functions from real arithmetic, to be compatible with interval branch and bound, it is therefore required to used their interval extension.

1.2 Contribution

After an intuitive introduction to interval arithmetic and to B-spline curves in Section 2. We provide the first application of interval BSCs (introduced in Section 3) applied to path generation for mobile robots in Section 4. The formalization onto a NCOP and the optimization using an interval branch and bound are presented through three scenarios and a deep analysis based on 10K runs. Each figure of this article is rigorously and symbolically computed using Python and C++ programs.

2 Preliminaries

Interval arithmetic and B-splines need to be introduced to smoothly explain the new approach presented in this article.

2.1 Interval Arithmetic

The interval arithmetic is an extension of real arithmetic. Intervals (Definition 2.1) are considered instead of real numbers.

Definition 2.1. *Let* $\boldsymbol{x} = [\underline{x}, \overline{x}]$ *be an* ***interval*** *with its two bounds* $(\underline{x}, \overline{x}) \in \mathbb{R}^2$, $\underline{x} \leq \overline{x}$. $\boldsymbol{x}$ *contains all real numbers between* $\underline{x}$ *and* $\overline{x}$ *including them. The set of intervals is denoted* $\mathbb{IR}$. *If* $\underline{x} = \overline{x}$, $\boldsymbol{x}$ *is said to be degenerate and can be seen as a real number* $x = \underline{x} = \overline{x}$. *The width of* $\boldsymbol{x}$ *is* $w(\boldsymbol{x}) = \underline{x} - \overline{x}$. *The middle of* $\boldsymbol{x}$ *is denoted* $m(\boldsymbol{x}) = \frac{\underline{x}+\overline{x}}{2}$. *Finally, a box* $\boldsymbol{X} = (\boldsymbol{x_0}, ..., \boldsymbol{x_n})$ *is a vector of intervals.*

The interval arithmetic was first proposed in [20]. It was then fully extended to deal with real arithmetic issues and limitations such as the division by zero or the definition and representation of infinite numbers. Moreover, the interval arithmetic appears to be a strong solution to perform robust and guaranteed computer calculation [29]. In particular, it offers a solution to the classic rounding error occurring with numerical representation of real numbers in a finite number of bits: the outward rounding [11, 22]. Finally, like other arithmetics, it provides a wide range of operations and tools making it interesting for many applications [12]. For our use-case, we use both interval objects and tools such as respectively interval B-spline curve [25] and interval branch and bound optimizer [2].

Notation 1. *For clarity and to avoid mismatch between real numbers, intervals and boxes, the following notations are used in this article:* x *is a real number;* $\boldsymbol{x}$ *is an interval;* X *is a vector; and* $\boldsymbol{X}$ *is a box.*

2.1.1 Operations

The interval arithmetic includes interval extension of real operations which are subject to a theorem (Theorem 2.1).

Theorem 2.1. *A real operation* $\diamond \in \{+, -, \times, \div, ...\}$ *can be extended to the interval arithmetic if* $\boldsymbol{x} \diamond \boldsymbol{y}$ *is an interval containing* $\{x \diamond y \mid x \in \boldsymbol{x}, y \in \boldsymbol{y}\}$.

$$\boldsymbol{x} \diamond \boldsymbol{y} = [\{x \diamond y \mid x \in \boldsymbol{x}, y \in \boldsymbol{y}\}] \tag{1}$$

For example, some operations are defined as follows:

- Addition:
$$\boldsymbol{x} + \boldsymbol{y} = [\underline{x} + \underline{y}, \overline{x} + \overline{y}] \tag{2}$$
- Subtraction:
$$\boldsymbol{x} - \boldsymbol{y} = [\underline{x} - \overline{y}, \overline{x} - \underline{y}] \tag{3}$$
- Multiplication:
$$With\ \mathcal{D} = \{\underline{x}\underline{y}, \underline{x}\overline{y}, \overline{x}\underline{y}, \overline{x}\overline{y}\},\ \boldsymbol{x}\boldsymbol{y} = [min(\mathcal{D}), max(\mathcal{D})] \tag{4}$$

As interval arithmetic deals with sets, operations such as the intersection $\cap$, the union $\cup$ and the inclusion $\subset$ are also available.

2.1.2 Dependency

The interval arithmetic suffers from an important weakness called the dependency problem. This problem appears because interval operations are too conservative.

Example 2.1. Let $\boldsymbol{x}$ be a non degenerate interval. Then, $\boldsymbol{x} - \boldsymbol{x} = [\underline{x} - \overline{x}, \overline{x} - \underline{x}]$. Therefore, $\boldsymbol{x} - \boldsymbol{x} \neq [0, 0]$.

Although $0 \in \boldsymbol{x} - \boldsymbol{x}$, $\boldsymbol{x} - \boldsymbol{x}$ is not equal to $[0, 0]$ (Example 2.1) because, according to the Theorem 2.1, the two operation members are considered independent. In other words, the operation $\boldsymbol{x} - \boldsymbol{x}$ is equivalent to $[\{a - b \mid a \in \boldsymbol{x}, b \in \boldsymbol{x}\}]$. More generally, the dependency problem appears when the number of occurrences of a variable in an equation is greater than one [20]. The dependency problem is well known and other arithmetics, such as affine arithmetic [5] or generalized interval arithmetic [9], were proposed to reduce its impact [15]. In Section 3, we explain how the definition of interval B-spline curve gets around the dependency problem.

2.1.3 Interval Function

As well as real numbers, the interval arithmetic provides an interval extension of functions.

Definition 2.2. *Considering* $f : \mathbb{R}^n \to \mathbb{R}^m$*, an extension or inclusion function is the* ***interval function*** $\boldsymbol{f} : \mathbb{IR}^n \to \mathbb{IR}^m$ *if:*

$$\forall \boldsymbol{x} \in \mathbb{IR}^n, f(\boldsymbol{x}) \subset \boldsymbol{f}(\boldsymbol{x}) \tag{5}$$

In this article, we consider the natural interval extension, which consists in substituting real operators with their interval equivalents presented in Section 2.1.1.

Remark 2.1. Because of the dependency problem, the syntax of the equation has an important impact on the result interval width. It is therefore interesting to use symbolic computation in order to reduce the dependency problem.

2.2 Interval Branch and Bound

The interval arithmetic opens the way to an interesting class of optimizers called Interval Branch and Bound (IB&B) [2]. These optimizers use a succession of bisections and contractions on boxes to optimize NCOPs with the guarantee to provide bounds which enclose the global optimum of the cost function [9] (Remark 2.2). A bisection consists in separating a box into two child boxes. A contraction is an operation which, using the analytical expression of constraints, reduces the widths of the NCOPs variables. Such optimization can be stopped prematurely (Remark 2.3). Although sub-optimal, the resulting solution satisfies all the constraints. This capability enables to deal with real-time performance. Moreover, both parts of a box resulting from a bisection can be processed independently. IB&B are therefore parallelizable.

Remark 2.2. When the optimization process ends within the allocated time, IB&B guarantees the following returns:

- Bounds of the cost function global optimum and corresponding optimized variables, if they exist.
- An empty set, if no valid solution exists.

Remark 2.3. When the optimization is prematurely stopped, IB&B can deliver two different results:

- If no point validating constraints was found during the allocated time: the algorithm returns infinite bounds for the global optimum of the cost function.
- If at least one point validating constraints is found during the allocated time: the IB&B returns sub-optimal bounds of the cost function global optimum and corresponding optimized variables.

2.3 Bézier and B-spline curves

Bézier and B-spline Curves (BCs & BSCs), their generalization, are part of the polynomial curves family. They are widely used by the computer aided design community; due to their numerous properties and because they offer a powerful

basis for shapes representations. In this section, we present some interesting properties of BCs and BSCs for path construction and optimization application. More BCs and BSCs properties, rigorous demonstrations and references can be found in books [23, 24]. As well as every polynomial curves, BCs and BSCs can be represented by either an implicit equation or a parametric function. Unlike implicit equation, the parametric function form separately describes each variable with a function depending on a joint parameter[3]. BCs and BSCs can be represented by both implicit equation and parametric function forms. In this article, we only consider the parametric function form because it simplifies programming and makes curve handling more intuitive and flexible. BCs must be introduced (Definition 2.3) in order to better understand BSCs properties and advantages.

Definition 2.3. *A **Bézier curve** is a polynomial given by:*

$$C(u) = \sum_{i=0}^{n} B_{i,n}(u) P_i, \ u \in [0,1] \tag{6}$$

with:

- *the Berstein Polynomial Basis Functions (BPBFs):*

$$B_{i,n}(u) = \frac{n!}{i!(n-i)!} u^i (1-u)^{n-i} \tag{7}$$

- *the control points:*

$$P_i = \begin{bmatrix} x_i \\ y_i \end{bmatrix} \tag{8}$$

BCs are defined by a finite number of control points which allow, when their coordinates are modified, to change the shape of the curve. Moreover, as BCs are constructed from Bernstein polynomials, curves' degree of continuity is controlled by the degree of polynomials. The number of control points thus directly influences the curve degree. Figure 1 shows an example of a three-degree BC (four control points), starting with its BPBFs, influenced by the attraction of control points, on lateral planes (xu, yu). The sum of the BPBFs over x and y can be respectively identified on both planes by a blue and a red dotted curve. The projection of these two sums onto the xy plane (visible at the back in green) represents the final BC. BPBF computation details are provided in Table 1.

BCs suffer from an important drawback: the degree of the curve is directly related to the number of control points. Thus, the degree of the curve quickly increases when a more precise control is required. Furthermore, by definition, BCs consist of a single segment: each point of the curve is influenced by every control points. Consequently, when a control point is moved, the entire curve is modified.

[3] One parameter per dimension. For example two parameters for a surface.

Table 1: Bézier curve construction example (related to Figure 1)

$B_{i,n}(u) * P_i$	$x_0 = 0.5$ $y_0 = 0.2$	$x_1 = 1.5$ $y_1 = 1.0$	$x_2 = 2.3$ $y_2 = 0.5$	$x_3 = 2.8$ $y_3 = 1.0$
$B_{1,n}(u) = -(u-1)^3$	(0) $-0.5(u-1)^3$ (0) $-0.2(u-1)^3$	-	-	-
$B_{2,n}(u) = 3u(u-1)^2$	-	(1) $4.5u(u-1)^2$ (1) $3u(u-1)^2$	-	-
$B_{3,n}(u) = -3u^2(u-1)$	-	-	(2) $-6.9u^2(u-1)$ (2) $-1.5u^2(u-1)$	-
$B_{4,n}(u) = u^3$	-	-	-	(3) $2.8u^3$ (3) u^3
$C(u) = \sum_{i=0}^{n} B_{i,n}(u)P_i$	$(\sum_x)$ $\frac{-(u^3+6u^2-30u-5)}{10}$ $(\sum_y)$ $\frac{(23u^3-39u^2+24u+2)}{10}$			

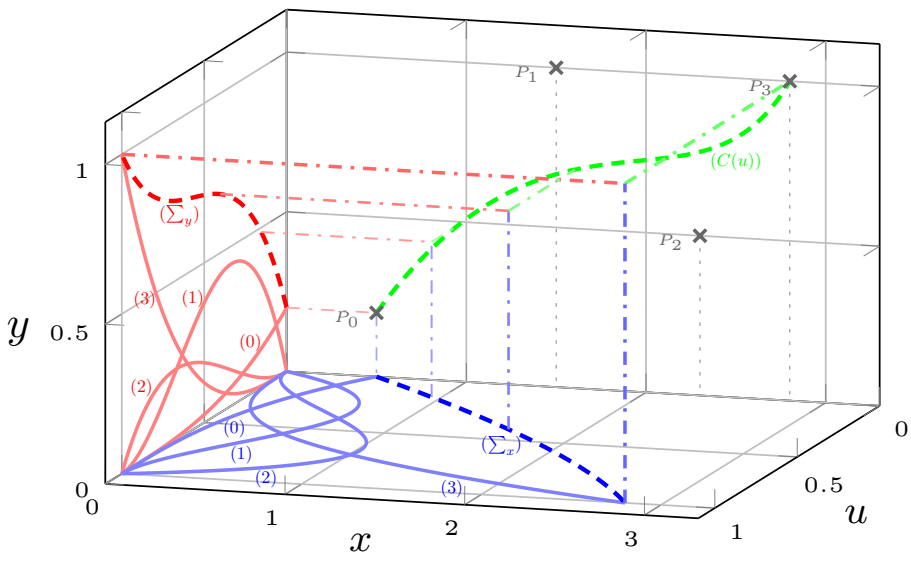


Figure 1: Bézier curve construction example (caption in Table 1)

BSCs were introduced in order to address these issues. Instead of BPBFs, they are constructed from the sum of B-Spline Basis Functions (BSBFs) computed by recurrence (Definition 2.4).

Definition 2.4. *A **B-spline curve** is a connection of polynomials given by:*

$$C(u) = \sum_{i=0}^{n} N_{i,p}(u) P_i, \ u \in [u_p, u_{n+1}] \tag{9}$$

with:

- *the B-spline basis functions:*

$$N_{i,0}(u) = \begin{cases} 1 & if \quad u_i \leq u \leq u_{i+1} \\ 0 & otherwise \end{cases} \tag{10}$$

$$N_{i,p}(u) = \frac{u - u_i}{u_{i+p} - u_i} N_{i,p-1}(u) + \frac{u_{i+p+1} - u}{u_{i+p+1} - u_{i+1}} N_{i+1,p-1}(u) \tag{11}$$

- *the knot vector:*

$$\mathcal{U} = \{u_0, u_1, ..., u_{n+p+1}\}, \ u_i \leq u_{i+1} \tag{12}$$

- *the control points:*

$$P_i = \begin{bmatrix} x_i \\ y_i \end{bmatrix} \tag{13}$$

As the parameter u moves in the sequence $\mathcal{U}$, a BSC is made up of connected segments, hence the name: nodes, for the terms of $\mathcal{U}$. A pth-degree BSC controlled by $n + 1$ control points has $n + 1 - p$ segments. Unlike BCs, the degree p of BSCs is not linked to the number of control points $(n + 1)$. Thus, when a control point is moved, only $p + 1$ segments are modified. It allows local modification and drastically reduces the amount of computation. Moreover, the continuity degree and therefore the smoothness of the curve are fully controllable. Figure 2 shows four BSCs. The three solid lines represent a three, five and eight-degree BSCs. Thus, the degree impact on curve deployment is observable. Moreover, in order to show the local modification property of BSCs, a control point has been moved (P_4 to P_4'), generating the three-degree dashed curve. Finally, each color on visible BSCs represents a segment.

Remark 2.4. BCs and BSCs can be seen as an approximation of their control points. An important and well-known property follows: curves are included in the envelope built from their control points.

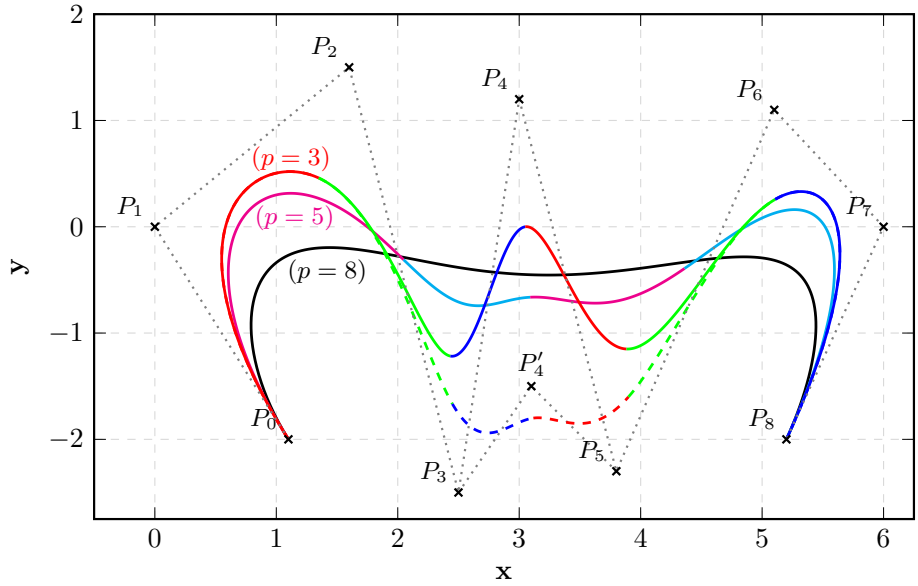


Figure 2: View of three BSC of different degrees (3, 5, 8). View of the impact of control points positioning on curve deployment (dashed curve).

2.4 Clamped, Semi-Open and Open B-spline

The knot vector $\mathcal{U}$ has an important impact on the spatial expansion and the continuity degree of the BSC. The curve switches from one segment to another. Each segment corresponds to the parameter displacement between two nodes. Thus, the knot vector directly influences the position and the continuity degree at segment connections. When a node is repeated multiple times in $\mathcal{U}$, this node multiplicity increases. A node of multiplicity m on a pth-degree BSC means that polynomials to the left and right of the node are of continuity equal to $p-m$. An higher multiplicity therefore implies a loss of continuity. In most applications, the multiplicity of the first and last nodes are fixed to $p+1$. The rest of the knot vector is chosen to be uniform (each node has a multiplicity of one and is evenly distributed in $\mathcal{U}$). This generates well-known clamped BSCs (Figure 2). Clamped means that the curve touches the first and last control points. Thus, the BSC end positions are better controlled. However, when it comes to connecting several clamped BSCs, additional operations are required to ensure continuity (pth-degree derivatives, curvatures and curvature evolution must be equal on the left and right of the connection).

Remark 2.5. A pth-degree clamped BSC controlled by $p+1$ control points is a BC. Moreover, its knot vector has only two nodes (0 and 1) of multiplicity $p+1$. *E.g.*: The 8th-degree clamped BSC visible in the Figure 2 (black curve) is a BC.

When it comes to making a large number of connections between BSCs, it is therefore better to keep the knot vector fully uniform and thus generate open BSCs.

Indeed, as the open BSC has a uniform knot vector, the continuous connection is therefore automatic and does not require additional computation. In fact, in order to connect two pth-degree open BSCs, p consecutive control points need to be equal between the two BSCs. For example, the Figure 3(2) shows two connected open BSCs of degree three. To do so, the first, second and third control points of the red BSC were chosen equal to the second, third and fourth control points of the blue one. In this case, the connection takes place at the end of the first segment of the blue BSC. In general, by definition, connections are necessarily made at the junction between segments. There's also an hybrid version where only the first node of $\mathcal{U}$ has a multiplicity of $p+1$, resulting in a semi-open BSC. Three-degree clamped, semi-open and open BSCs, generated with the same control polygon (list of control points) are visible in the Figure 3(1).

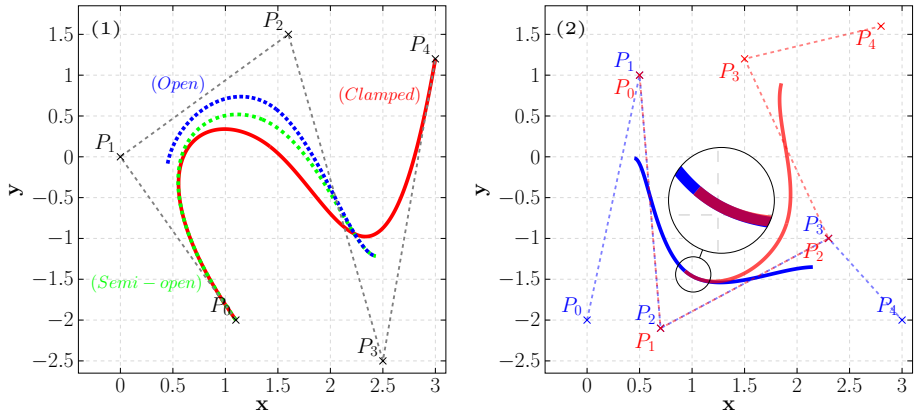


Figure 3: Clamped, semi-open and open B-splines (left). Connection of two open B-spline curves(right).

3 Interval B-spline

Interval B-spline Curves (IBSCs) were introduced in [25] (Definition 3.1) and mainly used for data approximation in various contexts: reverse modeling in computer aided design [28]; curve reconstruction [16]; or more recently for hyperspectral imaging [4].

Definition 3.1. *An **interval B-spline curve** is a B-spline curve in which the control points are boxes (Figure 4):*

$$\boldsymbol{C}(u) = \sum_{i=0}^{n} N_{i,p}(u)\boldsymbol{P_i} \tag{14}$$

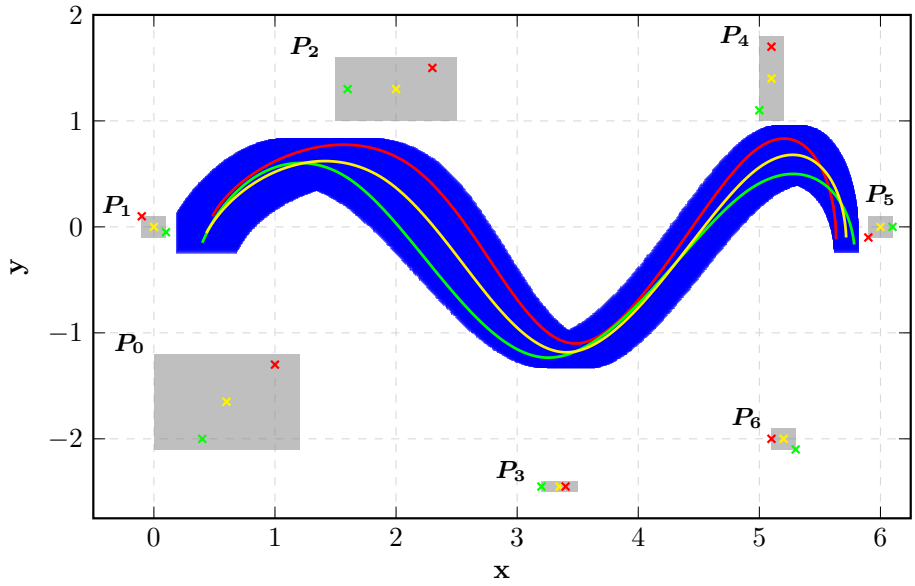


Figure 4: IBSC with three included BSCs including it's midpoint

Property 1. *The coordinates of an IBSC are obtained by algebraic operations defined in the interval arithmetic (Section 2.1.1). Consequently, considering an IBSC* $\boldsymbol{C_Q}$ *build with control points included in control points of another IBSC* $\boldsymbol{C_P}$*.* $\boldsymbol{C_Q}$ *is thus included in* $\boldsymbol{C_P}$ *and is an IBSC:*

$$\forall \boldsymbol{Q_i} \subset \boldsymbol{P_i} \implies \boldsymbol{C_Q}(u) = \sum_{i=0}^{n} N_{i,p}(u)\boldsymbol{Q_i} \subset \boldsymbol{C_P}(u) = \sum_{i=0}^{n} N_{i,p}(u)\boldsymbol{P_i} \tag{15}$$

Remark 3.1. A curve included in an IBSC is not necessarily an IBSC.

The definition 3.1 is interesting as it avoids the dependency problem. Indeed, as the parameter u is a real number, the equation of the curve can be symbolically simplified. In its simplest form, each variable, corresponding to the control points coordinates, appears only once in the equation. The size of the box resulting from the curve evaluation is then minimal [20]. The dependency problem is overcome.

4 IBSC-based Path Planner

Our local planning approach consists in optimizing successive open IBSC using an IB&B to generate paths over a receding horizon. Each optimized IBSC is automatically connected with the previous one using the method presented in Section 2.4. Our application uses circular obstacles to simplify constraints. Distances between curves and obstacles are therefore simpler to compute. We use the Ibex library [10]

and its plugin IbexOpt, which respectively implement the interval arithmetic and an IB&B in C++. Symbolic calculations are made with SymEngine [27] in C++.

4.1 NCOP for path planner

The objective is to connect the starting point S to the end point E with a finite number of optimizations n_{opt}. The environment contains n_{obs} circular obstacles of center Ωo_j and radius ro_j. During the k^{th} optimization, the robot has a known perception distance resulting in a circular planning horizon H_k, centered on the robot position Ωh_k, and of radius rh_k. Only visible obstacles: V_{obs} (included in H_k), are considered for the k^{th} optimization.

Notation 2. *The euclidean distance between A and B is denoted* $d(A, B)$.

The k^{th} NCOP is defined as:
Variables:

$$\boldsymbol{P_{i,k}} = \left| \begin{array}{l} S,\ if \left\{ \begin{array}{l} i \in \{0, ..., p-1\} \\ k = 0 \end{array} \right. \\ \boldsymbol{P_{i+s,k-1}},\ with\ s \in \{1, ..., n-p\},\ if \left\{ \begin{array}{l} i \in \{0, ..., p-1\} \\ k \in \{1, ..., n_{opt}-1\} \end{array} \right. \\ \left[\begin{array}{l} [-\infty, +\infty] \\ [-\infty, +\infty] \end{array} \right] \ otherwise \end{array} \right. \tag{16}$$

Cost function:

$$\boldsymbol{z_k} = d^2(\boldsymbol{C}(u_{n+1}), E) + \sum_{i=0}^{n-1} d^2(\boldsymbol{P_{i,k}}, \boldsymbol{P_{i+1,k}}) \tag{17}$$

Constraints:

$$\Gamma_k = \left\{ \begin{array}{l} \forall (\Omega o_j, ro_j) \in V_{obs},\ \forall u,\ d^2(\boldsymbol{C}(u), \Omega o_j) \subset \left[ro_j^2, +\infty\right] \\ d^2(\Omega h_k, \boldsymbol{C}(u_{n+1})) \subset \left[rh_k^2 - \varepsilon, rh_k^2\right],\ \varepsilon > 0 \end{array} \right. \tag{18}$$

Remark 4.1. All distances are squared to avoid square roots in the symbolic form of functions, reducing the number of operations during the contraction.

Variables of the k^{th} optimization correspond to the control points of the k^{th} IBSC (Eq. (16)). By definition, the first point of the curve is affected by p control points. During initialization, setting the latter equal to the starting point S ensures that the IBSC starts at S. For other optimizations, to ensure a continuous connection, the p first control points must correspond to p successive control points of the previous optimized IBSC. As discussed in Section 2.4, the connection between two curves necessarily takes place at the junction between segments. The variable s is used to specify after which segment of the k^{th} curve the connection with the $k+1^{th}$

one takes place. For example, if $s = 1$, the connection takes place at the end of the first segment of the k^{th} curve. Secondly, we want to minimize: (i) The distance between the last point of the curve and the end point E, to gradually connect it to E; (ii) The distance between successive control points to reduce the envelop (control polygon) area and therefore the spatial expansion of the curve. The cost function is thus built from these two elements (Eq. (17)). Finally, all curve coordinates must be outside obstacles and the last point of the curve must be close to the planning horizon. These two conditions generate the NCOP constraints (Eq. (18)). In order to exactly reach the end point, when E is inside the planning horizon, the planning horizon is set equal to the distance between the robot and E. The end of the curve thus does not exceed E. The planner succeeds when the last point of the curve is inside an area around E.

4.2 Path classification

When an IB&B provides a solution, it guarantees that constraints are validated. In the same way, it is possible to detect when optimization has failed (Remarks 2.2 and 2.3). Strategies in case of failure can then be considered. In our approach, as an optimization takes place in a finite time ($t < timeout$), three scenarios can be considered, leading to four path planner states:

- The solution is valid: it is stored and the following optimization is triggered. If the end point is reached, the path planner succeeded ($state = \boldsymbol{valid}$). If the generated path returns to the same location several times, the path planner is blocked in a concavity formed by obstacles. In practice, to detect this phenomenon, distances between the center of the new horizon to the center of all the previous horizons are observed. If the next planning horizon is close to several previous horizons (several of these distances are smaller than a threshold): the path planner is considered blocked ($state = \boldsymbol{blocked}$).
- There is no solution (empty set): optimizations are stopped ($state = \boldsymbol{fail}$).
- There is no solution in $t < timeout$ ($w(\boldsymbol{z_k}) = +\infty$): a local start point is defined as $S_k = \Omega h_k$. A new optimization is initiated with more time allocated. If it succeeds, the next optimization is triggered ($state = \boldsymbol{fail\ recovered}$). The fail recovered state should be seen as an attempt to take into account rare situations occurring in real-life conditions. Other solutions could be considered, such a request for assistance involving human intervention. Our approach disrupts the smoothness of the path, introducing a continuity break. This recovery approach reduces the scope of suitable robots to differential and holonomic. If it fails again or if the number of fail recovered exceeds a threshold value: optimizations are stopped ($state = \boldsymbol{fail}$).

4.3 Experiments

In this section we analyze this approach in two experiments. Firstly, three general scenarios are presented to analyze the behavior of the local planner. Secondly, an

analysis of a large number of simulations involving random obstacles is proposed to study the states distribution.

4.3.1 Three scenarios analysis

Results of three scenarios are displayed in Figure 5 to examine how the time allocated to optimization (timeout) and the planning horizon (rh_k) affect the path planner behavior. Values of other parameters are set as follows:

- $p = 3$: A 3-degree IBSC is C^2, which ensures a continuous curvature and a smooth path.
- $n = 4$: The number of control points affects directly the problem dimension (e.g. $n = 4$: problem of dimension $2(n + 1) = 10$). Furthermore, the number of control points impacts the number of segments. The IBSC must have a minimum of 2 segments to use the principle generation over receding horizon. A 3-degree IBSC built from 5 control points ($n = 4$) has 2 segments and therefore limits the problem dimension.
- $s = 1$: A 3-degree IBSC build from 5 control points has 2 segments, s is necessarily equal to 1. The connection between the k^{th} and the $k+1^{th}$ IBSC therefore takes place after the first segment of the k^{th} curve.
- u is discretized 25 times per segment. This gives an average offset of $20cm$ between each curve evaluation (assumption of a straight curve over a $10m$ planning horizon).

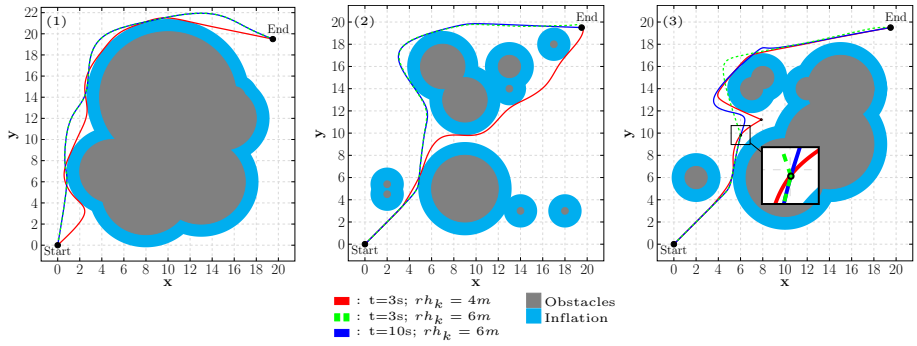


Figure 5: Timeout (t) and planning horizon (rh_k) impacts on the local planner behavior

Each scenario has a specific environment: (1) Environment composed of large obstacles forming a single cluster. (2) Cluttered environment with small obstacles. (3) Cluttered environment where obstacles form a dead end. Three parameter sets were tested in different environments to observe the timeout and planning horizon impact on the path planner behavior. The shorter the planning horizon, the closer

the path is to obstacles. An IBSC with a small number of control points generated over a large horizon means that a single control point causes deformation over a large area, resulting in some quite bulging areas close to obstacles. Furthermore, a shorter horizon allows the planner to pass through less accessible areas. Scenario (3) shows two ***fail recovered*** states. In both cases, when entering the dead end, the optimizer did not obtain a solution within the allocated time. A breaking point was set up (equivalent to a new starting point), and a new optimization was started with more time allocated. The blue curve does not contain a break. Indeed, due to a larger timeout, the optimizer found a more complex solution. Scenarios (1) and (2) show that the timeout has little impact in environments without dead ends.

4.3.2 Simulated states distribution

A large number of simulations have been carried out with random obstacles and different parameter sets. An environment, similar to those shown in Figure 5, is randomly generated for each simulation. Results are shown as four matrices in Figure 6. One hundred simulations were carried out for each pair of parameters. For a given pair on a given matrix, the resulting cell represents the state distribution. For example, with the pair $(t = 10, rh_k = 1)$, we obtained 56 % valid; 0 % fail; 0 % fail recovered; and 44 % blocked. This study shows that the choice of parameters

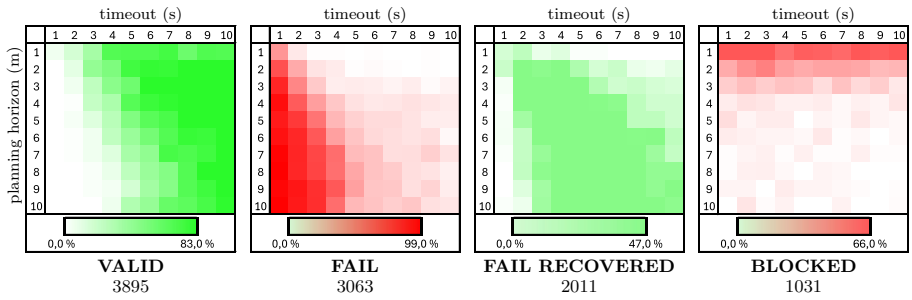


Figure 6: 10K simulations analysis

is important to obtain reliable results. In fact, for a timeout greater than $3s$ and a planning horizon greater than $3m$, more than 77 % of simulations reach the target while avoiding obstacles. In cases of low timeouts, IB&B doesn't have time to obtain a solution, thus the number of fails is important. Furthermore, it is logical to observe that scenarios with a small planning horizon tend to get blocked. As part of a complete mobile robotics application, a global planner would provide intermediate waypoints, helping the local planner to avoid dead ends. A larger planning horizon implies more calculation, as more obstacles are taken into account. Thus, for the same timeout, the number of valid solutions decreases when the horizon increases. In this case, as the fail recovery procedure increases the timeout, many fails are saved and the target is still reached.

4.4 Discussion

A limitation comes from the discretization resolution of the parameter u in the IBSC definition. The gap between two IBSC evaluations makes obstacle avoidance constraints less rigorous. In theory, the obstacle avoidance constraint (Eq. (18)) guarantees that for all parameters u, the IBSC is outside obstacles. However, the coordinate of a point resulting from the curve evaluation for a parameter somewhere between two successive values of u is not necessarily outside obstacles. The lower the discretization resolution, the greater the risk of slightly entering an obstacle. Moreover, the amount of computations increases proportionally with the discretization resolution of u. This resolution must therefore be chosen according to the size of the obstacles. An improvement would be to no longer consider u as a real but as an interval. In this case, obstacle avoidance would be ensured. However, the problem of dependency would no longer be overcome. A new definition of IBSC would be therefore required. Another limitation comes from the circular obstacle assumption. Ideally, all types of obstacle should be considered. One improvement would be to apply a cost function directly on a cost map and use a binary map for constraints. In this way, all obstacles would be taken into account with the resolution of the binary map.

5 Conclusion

Autonomous off-road navigation of wheeled robots requires rigorous control and planning approaches. The local planning level is essential to fill the gap between the global planning and the tracking control levels. Interval methods are interesting in this context for their reliability and provided guarantees. We propose a new interval-based local path planner. The local path is obtained by successive optimization of an open interval B-spline curve over receding horizon using a branch & bound optimizer. We thus propose the first application of open interval B-spline curves for path generation in robotics. After introducing B-splines and interval arithmetic, interval B-splines were intuitively presented. The formalization, a demonstration in three scenarios, and an in-depth study involving 10K simulations have been proposed. This new approach has proved its efficiency in complex situations involving dead-ends and cluttered environments. Furthermore, our method provides a guaranteed verdict on the solution state obtained after optimization (valid, fail, fail recovered, or blocked). This technique is general and modular. Various constraints and optimization criteria can be applied to customize the resulting path. The requirements of navigation on uneven terrains are then respected.

References

[1] Alexandre Dit Sandretto, J., Chapoutot, A., and Mullier, O. Formal verification of robotic behaviors in presence of bounded uncertainties. In *Proceedings*

of the First IEEE International Conference on Robotic Computing, pages 81–88. IEEE Computer Society, 2017. DOI: 10.1109/IRC.2017.17.

[2] Araya, I. and Reyes, V. Interval branch-and-bound algorithms for optimization and constraint satisfaction: A survey and prospects. *Journal of Global Optimization*, 65(4):837–866, 2016. DOI: 10.1007/s10898-015-0390-4.

[3] Benatti, S., Young, A., Elmquist, A., Taves, J., Tasora, A., Serban, R., and Negrut, D. End-to-end learning for off-road terrain navigation using the Chrono open-source simulation platform. *Multibody System Dynamics*, 54(4):399–414, 2022. DOI: 10.1007/s11044-022-09816-1.

[4] Boukezzoula, R., Coquin, D., and Jacq, K. A possibilistic regression based on gradual interval B-Splines: Application for hyperspectral imaging lake sediments. *Information Sciences*, 510:183–204, 2020. DOI: 10.1016/j.ins.2019.09.031.

[5] de Figueiredo, L. H. and Stolfi, J. Affine arithmetic: Concepts and applications. *Numerical Algorithms*, 37(1):147–158, 2004. DOI: 10.1023/B:NUMA.0000049462.70970.b6.

[6] Defoort, M., Palos, J., Kokosy, A., Floquet, T., and Perruquetti, W. Performance–based reactive navigation for nonholonomic mobile robots. *Robotica*, 27(2):281–290, 2009. DOI: 10.1017/S0263574708004700.

[7] Douthwaite, J. A., Zhao, S., and Mihaylova, L. S. A comparative study of velocity obstacle approaches for multi-agent systems. In *Proceedings of the UKACC 12th International Conference on Control*, pages 289–294, 2018. DOI: 10.1109/CONTROL.2018.8516848.

[8] Foead, D., Ghifari, A., Kusuma, M. B., Hanafiah, N., and Gunawan, E. A systematic literature review of A* pathfinding. *Procedia Computer Science*, 179:507–514, 2021. DOI: 10.1016/j.procs.2021.01.034.

[9] Hansen, E. and Walster, G. W. *Global Optimization Using Interval Analysis: Revised And Expanded*. CRC Press, 2004. DOI: 10.1201/9780203026922.

[10] Ibex Team. Ibex documentation. URL: https://ibex-team.github.io/ibex-lib/intro.html#.

[11] IEEE. Standard for binary floating-point arithmetic. *ANSI/IEEE Std 754-1985*, pages 1–20, 1985. DOI: 10.1109/IEEESTD.1985.82928.

[12] Jaulin, L., Kieffer, M., Didrit, O., and Walter, E. *Applied Interval Analysis with Examples in Parameter and State Estimation, Robust Control and Robotics*. Springer London, 2001. DOI: 10.1007/978-1-4471-0249-6.

[13] Katoch, S., Chauhan, S. S., and Kumar, V. A review on genetic algorithm: past, present, and future. *Multimedia Tools and Applications*, 80(5):8091–8126, 2021. DOI: 10.1007/s11042-020-10139-6.

[14] Kraft, D. *A Software Package for Sequential Quadratic Programming*. Deutsche Forschungs- und Versuchsanstalt für Luft- und Raumfahrt Köln: Forschungsbericht. Wiss. Berichtswesen d. DFVLR, 1988. url: https://books.google.fr/books?id=4rKaGwAACAAJ.

[15] Krämer, W. Generalized intervals and the dependency problem. *Proceedings in Applied Mathematics and Mechanics*, 6(1):683–684, 2006. DOI: 10.1002/pamm.200610322.

[16] Lin, H., Chen, W., and Wang, G. Curve reconstruction based on an interval B-Spline curve. *The Visual Computer*, 21:418–427, 2005. DOI: 10.1007/s00371-005-0304-4.

[17] Lucet, E., Micaelli, A., and Russotto, F.-X. Accurate autonomous navigation strategy dedicated to the storage of buses in a bus center. *Robotics and Autonomous Systems*, 136:103706, 2021. DOI: 10.1016/j.robot.2020.103706.

[18] Macenski, S., Singh, S., Martín, F., and Ginés, J. Regulated pure pursuit for robot path tracking. *Autonomous Robots*, 47(6):685–694, 2023. DOI: 10.1007/s10514-023-10097-6.

[19] Mendes Filho, J. M., Lucet, E., and Filliat, D. Real-time distributed receding horizon motion planning and control for mobile multi-robot dynamic systems. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 657–663, 2017. DOI: 10.1109/ICRA.2017.7989081.

[20] Moore, R. E. *Interval Analysis*. Prentice-Hall series in automatic computation. Prentice-Hall, 1966. ISSN: 2577-9435.

[21] Orthey, A., Chamzas, C., and Kavraki, L. E. Sampling-based motion planning: A comparative review. *Annual Review of Control, Robotics, and Autonomous Systems*, 7:285–310, 2023. DOI: 10.1146/annurev-control-061623-094742.

[22] Palmer, J. F. and Morse, S. P. *The 8087 Primer*. Wiley, 1984. ISBN: 9780471875697.

[23] Piegl, L. and Tiller, W. *The NURBS Book*. Monographs in Visual Communication. Springer Berlin Heidelberg, 1996. DOI: 10.1007/978-3-642-59223-2.

[24] Risler, J. J. *Mathematical Methods for CAD*. Recherches en mathématiques appliquées. Cambridge University Press, 1992. ISBN: 9780521436915.

[25] Shen, G. and Patrikalakis, N. M. Numerical and geometrical properties of interval B-Splines. *International Journal of Shape Modeling*, 04(01n02):35–62, 1998. DOI: 10.1142/S0218654398000040.

[26] Si Larbi, L., Lucet, E., and Alexandre dit Sandretto, J. Overview of motion planning techniques and their suitability for an off-road navigation use-case. In *Proceedings of the 18th International Conference on Control, Automation,*

Robotics and Vision, pages 1054–1061, 2024. DOI: 10.1109/ICARCV63323.2024.10821670.

[27] SymEngine Team. SymEngine documentation. URL: https://symengine.org/.

[28] Tuohy, S. T., Maekawa, T., Shen, G., and Patrikalakis, N. M. Approximation of measured data with interval B-Splines. *Computer-Aided Design*, 29(11):791–799, 1997. DOI: 10.1016/S0010-4485(97)00025-0.

[29] Van den Berg, J. B. and Queirolo, E. Rigorous validation of a Hopf bifurcation in the Kuramoto–Sivashinsky PDE. *Communications in Nonlinear Science and Numerical Simulation*, 108:106133, 2022. DOI: 10.1016/j.cnsns.2021.106133.

Acta Cybernetica **27** (2026) 313–336.

Online Interval Depth Localization of an Underwater Robot with Ballast

Luc Jaulin[ab]

Abstract

This paper presents an efficient online method to simulate a dynamical system with interval uncertainties. These uncertainties can be either on the initial state vector, on the time-dependent inputs, or on the evolution function. Compared to other techniques used for the guaranteed integration of differential inclusion, the presented approach is online and requires a small and fixed number of operations at each sampling time. An illustration related to underwater robotics will be provided. The application involves a robot with a ballast that can move from the surface to the sea floor. We would like to guarantee that the robot will reach a given depth at a given time.

Keywords: differential inclusion, ballast, underwater robot, interval analysis, interval integration, reachability

1 Introduction

Reachability has been studied by many authors using set-membership tools [4, 6, 9, 10, 11, 21, 27]. Often, the objective of reachability is to predict the future of a dynamical system under uncertainties [26]. In this paper, we will focus on an underwater robot equipped with a ballast, namely a *float*, shown in Figure 1.

The float can only move upward to the surface and downward to the seafloor. The state equations are given by

$$\left\{\begin{array}{rcl} \dot{s} & = & u \\ \dot{v} & = & \frac{g\beta s}{1+\beta s} - \frac{c_x}{2(1+\beta s)\ell}\, v\cdot|v| \\ \dot{d} & = & v \end{array}\right. \tag{1}$$

where the state variables are:

[a]Robex, Lab-STICC, ENSTA-Bretagne, France
[b]E-mail: lucjaulin@gmail.com, ORCID: 0000-0002-0938-0615

DOI: 10.14232/actacyb.314709

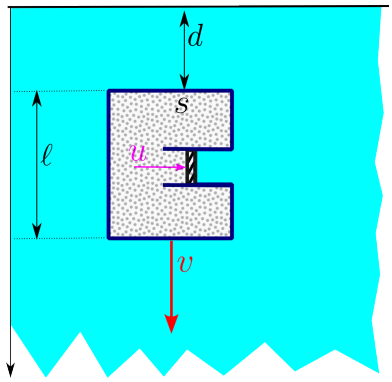


Figure 1: The buoyancy of the robot may change depending of the position of the piston

- The sinking coefficient s (or buoyancy) which corresponds to the position of the piston of the ballast. When $s = 0$, the density of the float is exactly that of the water ρ_0. When $s > 0$, the float sinks and when $s < 0$, it surfaces. The derivative of s can be controlled by a motor and corresponds to the input u. Equivalently, u corresponds to rate of fluid which enters in the ballast.
- The depth d expressed in meters. It is the derivative of the vertical speed.
- The vertical speed v. Its evolution depends of the forces (gravitational, buoyant, drag).

The parameters, assumed to be constant, are:

- The acceleration due to gravity g
- The amplification rate of the piston β: When the piston of the ballast is at position s, the average density of the robot is $(1 + \beta s)$.
- The drag coefficient c_x with respect to the vertical.
- The length ℓ with respect to the vertical position

To get this model, it suffices to apply the Newton's second Law:

$$m\dot{v} = \underbrace{mg}_{\text{gravitational force}} - \underbrace{\rho_0 A\ell g}_{\text{buoyant force}} - \underbrace{\frac{1}{2}c_x\rho_0 A \cdot v \cdot |v|}_{\text{drag force}} \tag{2}$$

where ρ_0 is the density of the water, and A is the cross-sectional area of the robot. Since the mass of the float is $m = (1 + \beta s)\,\rho_0 A\ell$, we get

$$\dot{v} = g - \frac{\rho_0 A\ell g}{(1 + \beta b)\,\rho_0 A\ell} - \frac{1}{2}\frac{c_x\rho_0 A v \cdot |v|}{(1 + \beta b)\,\rho_0 A\ell} \tag{3}$$

which corresponds to (1). Note that much more accurate models can be found in [18].

We assume that we know intervals containing the initial state variables and a tube (*i.e.*, an interval of trajectories) containing the input $u(t)$. Our goal is to find a tube for the three state variables. The notion of tube is illustrated by Figure 2. A tube can be seen as an array containing two lists of intervals: the *gates* $[x](k)$ and the *slices* $[\![x]\!](k)$ (see [24]). A tube is the set of all trajectories that cross all gates and that are always enclosed in the slices. More formally, the slices and the gates are intervals that should satisfy

$$\begin{array}{ll} \forall k, x(k\delta) \in [x](k) & \text{(for the gates)} \\ \forall t \in [(k-1)\delta, k\delta], x(t) \in [\![x]\!](k) & \text{(for the slices)} \end{array} \tag{4}$$

For simplicity, this tube is denoted by $[x](t)$.

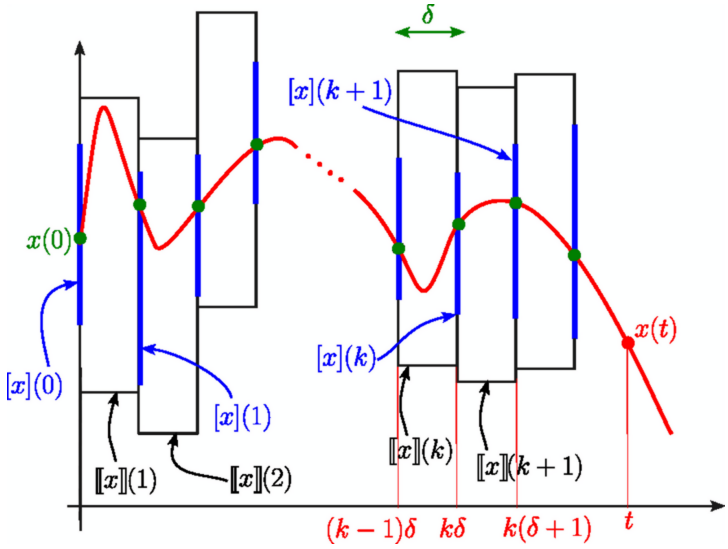


Figure 2: A tube which encloses the trajectory $x(t)$

We can now be more precise on the problem we want to solve. Assume that

- we know an initial box $[s_0^-, s_0^+] \times [v_0^-, v_0^+] \times [d_0^-, d_0^+]$ containing the state (s, v, d) at time $t = 0$.
- we know a tube $[u](t)$ containing $u(t)$ for all $t \geq 0$.

We want to find a tube for each state variable. Moreover, we want the method to be online [22]. More precisely, as illustrated by Figure 3, we want an interval

estimator of the form

$$\begin{pmatrix} [s](k) \\ [v](k) \\ [d](k) \end{pmatrix} = \mathcal{F}\left(\begin{pmatrix} [s](k-1) \\ [v](k-1) \\ [d](k-1) \end{pmatrix}, [\![u]\!](k)\right)$$
$$\begin{pmatrix} [\![s]\!](k) \\ [\![v]\!](k) \\ [\![d]\!](k) \end{pmatrix} = \mathcal{G}\left(\begin{pmatrix} [s](k-1) \\ [v](k-1) \\ [d](k-1) \end{pmatrix}, [\![u]\!](k)\right) \quad (5)$$

such that the corresponding tubes enclose the signals $s(t), v(t), d(t)$. Note that we have gates $[s](k), [v](k), [d](k)$ for s, v, d and a slice $[\![u]\!](k)$ for u. The interval state estimator should execute a fixed set of operations at each sampling time, otherwise we do not have an online predictor. As a consequence, interval methods based on Picard fixed point methods ([2, 13, 20]), bisection based methods (see, e.g., [14]) are not allowed. Moreover, the memory used by estimator should be fixed and thus zonotope approaches [1, 3, 7] should be avoided. In our case, only the three gates for s, v, d will be memorized.

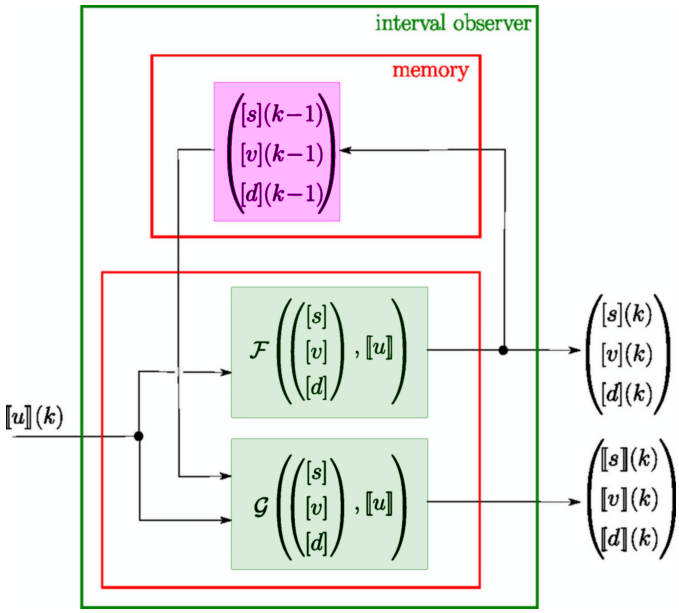


Figure 3: Online interval state estimator

We will take advantage of the fact that the float is composed of three SISO (Single-Input Single-Output) systems in series, as illustrated by Figure 4.

The paper is organized as follows. Section 2 recalls some classical results for differential inclusion for systems with a single state variable. Section 3 introduces the notion of *interval flow* which will be main operator used for a reliable discretisa-

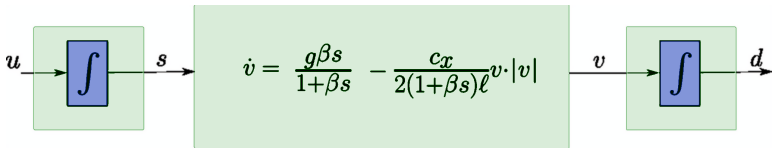


Figure 4: The float is composed of three SISO systems in series

tion of differential inclusions. Section 4 studies the well-known Riccati differential equation. This part will be needed for the resolution of the sinking body problem proposed in Section 5. Section 6 combines all these tools to derive an online interval predictor for the float with interval uncertainties. Section 7 concludes the paper.

2 Differential inclusion

This section recalls some classical results related to differential inclusions [5]. These results will be used further in order to build a reliable procedure to predict the evolution of the float with interval uncertainties.

We consider the scalar system

$$\begin{array}{c} \dot{v}(t) = f(v(t), \mathbf{u}(t)) \\ v(0) = v_0 \in [v_0] \\ \mathbf{u}(t) \in [\![\mathbf{u}]\!] \subset \mathbb{R}^n \end{array} \tag{6}$$

The signal $\mathbf{u}(t)$ is inside the box $[\![\mathbf{u}]\!] \subset \mathbb{R}^m$. Note that $\mathbf{u}(t)$ in chosen as a vector and this is why is it written bold. It varies with time in contrast to the box $[\![\mathbf{u}]\!]$ which is assumed to be constant with time. We have here a differential inclusion [5] with many solutions, as many as we have different signals $\mathbf{u}(t)$ in the box $[\![\mathbf{u}]\!]$. Finding an envelope for the set of all solutions $v(t)$ is a difficult problem which can be solved using optimal control theory [15] for some cases.

2.1 Comparison theorem

We recall here a theorem which can be used directly to find an envelope for a differential inclusion with one state variable [8]. It takes into account the monotonicity of the subsystems [25] to facilitate the interval integration.

Proposition. *Assume that the initial condition satisfies* $v_0 \in [v_0^-, v_0^+]$ *and denote by* $[f]$ *an inclusion function for* f *[17]. An envelope for any solution* $v(t)$ *is* $[v^-(t), v^+(t)]$*, where:*

$$\begin{array}{rcl} \dot{v}^- & = & lb([f](v^-, [\mathbf{u}])) \quad , \quad v^-(0) = v_0^- \\ \dot{v}^+ & = & ub([f](v^+, [\mathbf{u}])) \quad , \quad v^+(0) = v_0^+ \end{array} \tag{7}$$

The operator lb takes the lower bound of its interval input and ub returns its upper bound.

Proof. The minimal and maximal solutions for (6) satisfy [16]:

$$\begin{array}{rcll} \dot{v} & = & f(v, \underset{\mathbf{u}\in[\mathbf{u}]}{\text{argmin}} f(v,\mathbf{u})) & , \quad v(0) = v_0^- \\ \dot{v} & = & f(v, \underset{\mathbf{u}\in[\mathbf{u}]}{\text{argmax}} f(v,\mathbf{u})) & , \quad v(0) = v_0^+ \end{array} \tag{8}$$

It is a consequence of the *Hamilton-Jacobi-Bellman* theorem in the scalar case [15]. Let us recall the comparison theorem for scalar differential equations

$$\left.\begin{array}{c} \dot{x}_1 = \varphi_1(x_1) \\ \dot{x}_2 = \varphi_2(x_2) \\ \varphi_1 \leq \varphi_2 \\ x_1(0) \leq x_2(0) \end{array}\right\} \Rightarrow \forall t, x_1(t) \leq x_2(t). \tag{9}$$

Now, for all v, we have

$$\begin{array}{rcl} \text{lb}([f](v^-,[\mathbf{u}])) & \leq & f(v^-, \underset{\mathbf{u}\in[\mathbf{u}]}{\text{argmin}} f(v^-,\mathbf{u})) \\ \text{ub}([f](v^+,[\mathbf{u}])) & \geq & f(v^+, \underset{\mathbf{u}\in[\mathbf{u}]}{\text{argmax}} f(v^+,\mathbf{u})) \end{array} \tag{10}$$

Therefore, using the comparison theorem, we conclude the proof of the proposition. □

2.2 Example: the sinking body

We consider a body totally immersed in the ocean as represented by Figure 5. As it will be seen later this example is chosen since it is an important component of our underwater robot.

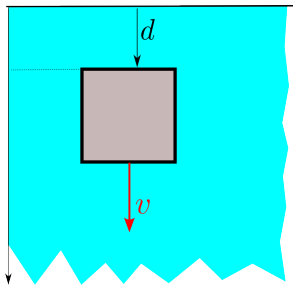


Figure 5: Sinking body

The speed v of the body satisfies the following differential equation

$$\dot{v} = a - bv|v| \tag{11}$$

where $b > 0$ corresponds to a dumping coefficient. If the body has a negative buoyancy, the coefficient a is positive and the body sinks toward the bottom. If it

has a positive buoyancy, a is negative and the body goes up toward the surface. We assume that both a and b are constant and belong to the intervals $[a^-, a^+]$, and $[b^-, b^+]$, respectively.

First, note that the system is stable and $v(t)$ converges to

$$\bar{v} = \text{sign}(a)\sqrt{\frac{|a|}{b}}. \tag{12}$$

Taking into account Proposition 2.1, we get a tube $[v^-, v^+]$ containing $v(t)$. The two bounds v^- and v^+ are defined by the two differential equations

$$\begin{array}{rcl} \dot{v}^- & = & \underbrace{\text{lb}\left([a] - [b]v^-|v^-|\right)}_{f^-_{[a],[b]}(v^-)} \quad , \quad v^-(0) = v_0^- \\ \dot{v}^+ & = & \underbrace{\text{ub}\left([a] - [b]v^+|v^+|\right)}_{f^+_{[a],[b]}(v^+)} \quad , \quad v^+(0) = v_0^+ \end{array} \tag{13}$$

The tube $[v^-, v^+]$ is minimal, *i.e.*, it is the smallest with respect to the inclusion which contains all feasible $v(t)$. As a consequence, we can find a good approximation for the two bounds $v^-(t)$ and $v^+(t)$ using any Runge-Kutta method. For example:

$$\begin{array}{rcl} v^-(t+\delta) & = & v^-(t) + \delta \cdot f^-_{[a],[b]}\left(v^-(t) + \frac{\delta}{2} \cdot f^-_{[a],[b]}(v^-(t))\right) \\ v^+(t+\delta) & = & v^+(t) + \delta \cdot f^+_{[a],[b]}\left(v^+(t) + \frac{\delta}{2} \cdot f^+_{[a],[b]}(v^+(t))\right) \end{array} \tag{14}$$

If δ is small, this approximation is very close to the minimal tube, but it does not provide any guarantee. Guaranteed bounds will be given later in Section 5.

We consider four illustrative cases, illustrated by Figure 6.

Case a We have $(v_0, a, b) \in [0.9, 1.1] \times [0.9, 1.1] \times [1.9, 2.1]$. This means that for $t = 0$, the body goes to the bottom. Since $a > 0$, it sinks (as represented by stones in the cube of the subfigure at the top). The two trajectories $v^-(t)$,$v^+(t)$ in red are obtained by the Runge-Kutta integration (14). We observe that the velocity interval $[v^-(t), v^+(t)]$ converges to the interval $[\bar{v}] = \text{sign}\,([a])\sqrt{\frac{|[a]|}{[b]}}$.

Case b We have $(v_0, a, b) \in [-1.1, -0.9] \times [-1.1, -0.9] \times [1.9, 2.1]$. This means that for $t = 0$, the body goes to the surface. Since $a < 0$, it floats (as represented by the bubbles in the cube of the Subfigure (b)). Again, the two trajectories $v^-(t)$,$v^+(t)$ in red are obtained by the Runge-Kutta integration (14). And again, we observe that $v(t)$ converges to a value $\bar{v}$.

Case c We have $(v_0, a, b) \in [0.9, 1.1] \times [-1.1, -0.9] \times [1.9, 2.1]$. For $t = 0$, the body is thrown toward the bottom. Since $a < 0$, the body floats. We observe that after approximately 1 sec, the body stops sinking and then starts its course to the surface. For the simulation, we need to compute the time at which $v(t)$ changes its sign.

Case d We have $(v_0, a, b) \in [-1.1, -0.9] \times [0.9, 1.1] \times [1.9, 2.1]$. For $t = 0$, the body is thrown toward the surface. Since $a > 0$, the body sinks. We observe that after approximately 1 sec, the body stops surfacing and then starts its course to the bottom.

3 Interval flow

In the previous section, we have shown how an integration of a differential inclusion can be performed in case of interval uncertainties. However, no guarantee was provided, mainly with respect to the time discretisation. In order to get a reliable integration approach, this section presents the new notion of interval flow.

3.1 Interval flow

Given a sampling time $\delta > 0$, an *interval flow* associated with (6) is a function Φ_f which satisfies

$$\Phi_f : \begin{array}{ccc} \mathbb{R} \times \mathbb{IR} \times \mathbb{IR}^m & \to & \mathbb{IR} \times \mathbb{IR} \\ (\delta, [v_0], [\![\mathbf{u}]\!]) & \to & ([v], [\![v]\!]) \end{array} \tag{15}$$

with

$$\left.\begin{array}{c} v(0) \in [v_0] \\ \forall t \in [0, \delta], \mathbf{u}(t) \in [\![\mathbf{u}]\!] \\ \dot{v}(t) = f(v(t), \mathbf{u}(t)) \\ ([v], [\![v]\!]) = \Phi_f(\delta, [v_0], [\![\mathbf{u}]\!]) \end{array}\right\} \Rightarrow \left\{\begin{array}{c} v(\delta) \in [v] \\ \forall t \in [0, \delta], v(t) \in [\![v]\!] \end{array}\right. \tag{16}$$

The interval flow will be used for the discretisation of a differential inclusion. Indeed, if we know an interval for the state $v(t_k)$ at time $t_k = k\delta$, and if we know an interval for the input $u(t)$ for all $t \in [t_k, t_k + \delta]$ the interval flow returns an interval containing $v(t)$, $t \in [t_k, t_k + \delta]$ and an interval for $v(t_k + \delta)$.

3.2 Example: the integrator

Consider the integrator with an uncertain input $u(t)$ and initial state v_0:

$$\left\{\begin{array}{c} \dot{v}(t) = u(t) \\ v(0) = v_0 \in [v_0] \\ u(t) \in [\![u]\!] = [u^-, u^+] \end{array}\right. \tag{17}$$

From Proposition 2.1, we know that any solution $v(t)$ is inside $[v^-(t), v^+(t)]$, where:

$$\begin{array}{rclcl} \dot{v}^- & = & u^- & , & v^-(0) = v_0^- \\ \dot{v}^+ & = & u^+ & , & v^+(0) = v_0^+ \end{array} \tag{18}$$

As a consequence, an interval flow is

$$\Phi_f(\delta, [v_0], [\![\mathbf{u}]\!]) = \left(\begin{array}{c} [v_0] + \delta [\![u]\!] \\ [v_0] + [0, \delta] \cdot [\![u]\!] \end{array} \right) \tag{19}$$

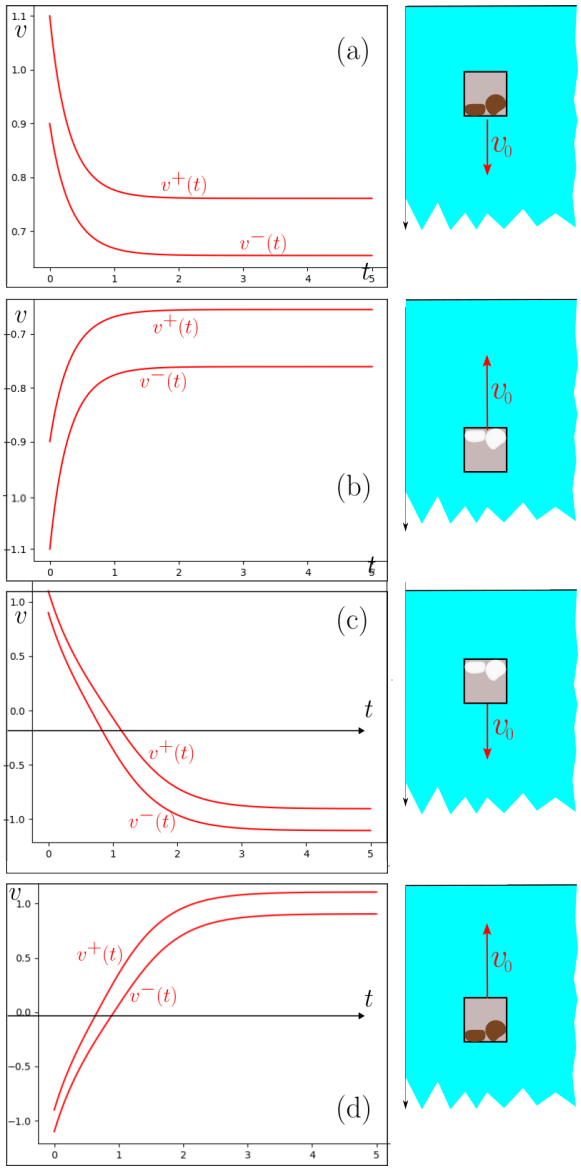


Figure 6: Sinking body (with stones inside) or floating body (with bubbles inside) for different initializations. There is no guarantee that the tubes contain the trajectory $v(t)$

3.3 Example: first order system

Consider a first order linear system with uncertain input u and initial state v:

$$\left\{ \begin{array}{c} \dot{v}(t) = av(t) + u(t) \\ v(0) = v_0 \in [v_0] \\ u(t) \in [\![u]\!] = [u^-, u^+] \end{array} \right. \tag{20}$$

From Proposition 2.1, we know that any solution $v(t)$ is inside $[v^-(t), v^+(t)]$, where:

$$\begin{array}{rcll} \dot{v}^- & = & av^- + u^- & , \quad v^-(0) = v_0^- \\ \dot{v}^+ & = & av^+ + u^+ & , \quad v^+(0) = v_0^+ \end{array} \tag{21}$$

i.e.

$$\begin{array}{rcl} v^-(t) & = & e^{at} v_0^- + \int_0^t e^{a(t-\tau)} u^-(\tau) d\tau \\ & = & e^{at} \left(v_0^- + \int_0^t e^{-a\tau} u^-(\tau) d\tau \right) \\ & = & e^{at} \left(v_0^- + u^- \int_0^t e^{-a\tau} d\tau \right) \\ & = & e^{at} \left(v_0^- + u^- \left[-\frac{1}{a} (e^{-a\tau}) \right]_0^t \right) \\ & = & e^{at} \left(v_0^- - \frac{u^-}{a} (e^{-at} - 1) \right) \\ v^+(t) & = & e^{at} \left(v_0^+ - \frac{u^+}{a} (e^{-at} - 1) \right) \end{array} \tag{22}$$

As a consequence, an interval flow for the scalar first order system is

$$\Phi_f(\delta, [v_0], [\![\mathbf{u}]\!]) = \left(\begin{array}{c} e^{a\delta} \left([v_0] - \frac{[\![u]\!]}{a} \left(e^{-a\delta} - 1 \right) \right) \\ e^{a[0,\delta]} \left([v_0] - \frac{[\![u]\!]}{a} \left(e^{-a[0,\delta]} - 1 \right) \right) \end{array} \right) \tag{23}$$

3.4 Real time interval integration

Recall that our goal is to integrate the equation of the float (1) with some interval uncertainties. Now, it will be shown later that the float is a serial composition of several subsystems for which we have an analytical interval flow. To show how this real-time interval integration can be done, we consider two compositions: serial and parallel, as illustrated by Figure 7. Note that the parallel composition will not we used for our application, but is given here to illustrate that our approach is not limited to serial systems.

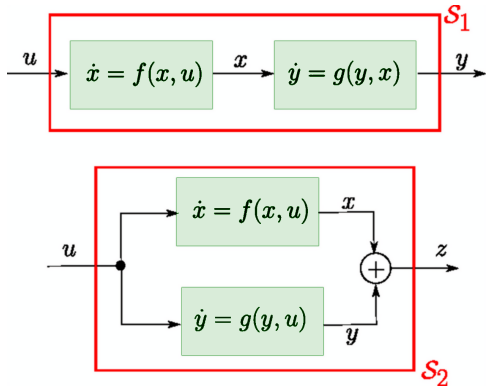


Figure 7: $\mathcal{S}_1$ is serial composition of two systems; $\mathcal{S}_2$ is a parallel composition

Serial systems Consider the system $\mathcal{S}_1$ (see Figure 7)

$$\mathcal{S}_1 : \left\{ \begin{array}{rcl} \dot{x} & = & f(x,u) \\ \dot{y} & = & g(y,x) \end{array} \right. \tag{24}$$

The following algorithm computes a tube for the output $y(t)$.

in: $[x_0], [y_0]$

1 $[x] = [x_0]$

2 $[y] = [y_0]$

3 for $k = 1$ to $k_{\max}$

4 Read $[\![u]\!] = [\![u]\!](k)$

5 $\begin{pmatrix} [x] \\ [\![x]\!] \end{pmatrix} = \Phi_f(\delta, [x], [\![u]\!])$

6 $\begin{pmatrix} [y] \\ [\![y]\!] \end{pmatrix} = \Phi_g(\delta, [y], [\![x]\!])$

7 write$(k, [y], [\![y]\!])$

Proof. Assume that,

$$\begin{array}{rcl} x(t_{k-1}) & \in & [x](k-1) \\ x([t_{k-1}-\delta, t_{k-1}]) & \in & [\![x]\!](k-1) \\ y(t_{k-1}) & \in & [y](k-1) \\ y([t_{k-1}-\delta, t_{k-1}]) & \in & [\![y]\!](k-1) \end{array} \tag{25}$$

with $t_k = k\delta$. Now, from Step 5,

$$\begin{pmatrix} [x](k) \\ [\![x]\!](k) \end{pmatrix} = \Phi_f(\delta, [x](k-1), [\![u]\!](k)) \tag{26}$$

From Equation (16),

$$\begin{cases} x(t_k) \in [x](k) \\ \forall t \in [t_k - \delta, t_k], x(t) \in [\![x]\!](k) \end{cases} \tag{27}$$

Moreover, from Step 6,

$$\begin{pmatrix} [y](k) \\ [\![y]\!](k) \end{pmatrix} = \Phi_f(\delta, [y](k-1), [\![x]\!](k)) \tag{28}$$

Thus, from (16),

$$\begin{cases} y(t_k) \in [y](k) \\ \forall t \in [t_k - \delta, t_k], y(t) \in [\![y]\!](k) \end{cases} \tag{29}$$

□

Parallel systems Consider the system $\mathcal{S}_2$ of Figure 7:

$$\mathcal{S}_2 : \begin{cases} \dot{x} &= f(x,u) \\ \dot{y} &= g(y,u) \\ z &= x+y \end{cases} \tag{30}$$

The following algorithm computes a tube for the output $z(t)$.

in: $[x_0], [y_0]$

1 $[x] = [x_0]$

2 $[y] = [y_0]$

3 for $k = 1$ to $k_{\max}$

4 Read $[\![u]\!] = [\![u]\!](k)$

5 $\begin{pmatrix} [x] \\ [\![x]\!] \end{pmatrix} = \Phi_f(\delta, [x], [\![u]\!])$

6 $\begin{pmatrix} [y] \\ [\![y]\!] \end{pmatrix} = \Phi_g(\delta, [y], [\![u]\!])$

7 $\begin{pmatrix} [z] \\ [\![z]\!] \end{pmatrix} = \begin{pmatrix} [x]+[y] \\ [\![y]\!]+[\![y]\!] \end{pmatrix}$

8 write$(k, [z], [\![z]\!])$

Proof. The proof is similar to that provided for serial systems. □

4 Analytical solution of the Riccati equation

To be able to simulate our float with an interval uncertainty, we need to find an interval flow for each of the three blocks of Figure 4. For the first and the last blocks which are both integrators, the interval flow has been given in Subsection 3.2. For the block of the middle, the interval flow needs a specific analytical resolution. Now, this resolution can be derived from the analytical solution of a Riccati equation that is considered in this section. All results given here are taken from [19] but only those that are useful for our application have been extracted from this book.

A Riccati equation is given by

$$\dot{v} = a - bv^2. \tag{31}$$

We assume here that $v_0 > 0$.

Proposition. *If $a > 0$ then the solution of (31) is*

$$\begin{array}{rcl} v(t) & = & \bar{v}\frac{ce^{2\sqrt{ab}t}-1}{ce^{2\sqrt{ab}t}+1} \\ c & = & \frac{\bar{v}+v_0}{\bar{v}-v_0} \\ \bar{v} & = & \sqrt{\frac{a}{b}} \end{array} \tag{32}$$

Proof. Set $E(t) = ce^{2\sqrt{ab}t}$, we have $\dot{E} = 2\sqrt{ab}cE$. We have

$$\begin{array}{rl} & \dot{v} = a - bv^2 \\ \Leftrightarrow & \bar{v}\frac{d}{dt}\left(\frac{E-1}{E+1}\right) = a - b\left(\bar{v}\frac{E-1}{E+1}\right)^2 \\ \Leftrightarrow & \bar{v}\left(\frac{\dot{E}(E+1)-\dot{E}(E-1)}{(E+1)^2}\right) = a - b\left(\bar{v}\frac{E-1}{E+1}\right)^2 \\ \Leftrightarrow & \bar{v}\left(\frac{2\sqrt{ab}cE(E+1)-2\sqrt{ab}cE(E-1)}{(E+1)^2}\right) = a - b\left(\bar{v}\frac{E-1}{E+1}\right)^2 \\ \Leftrightarrow & \sqrt{\frac{a}{b}}\left(2\sqrt{ab}E\,(E+1) - 2\sqrt{ab}E\,(E-1)\right) = a\,(E+1)^2 - a\,(E-1)^2 \\ \Leftrightarrow & 4aE = E^2 + 2aE + 1 - \left(E^2 + 2aE + 1\right) \end{array} \tag{33}$$

which is true. □

The solution of the Riccati equation, as given by Proposition 4 is singular when $a = 0$ and numerically ill-conditioned a is near zero. Now, this singularity has no physical reason and can be avoided by the using the *exponential cardinal* function expc(ν) defined by

$$\mathrm{expc}(\nu) = \frac{e^{\nu} - 1}{\nu} \tag{34}$$

with expc(0) = 1. This function is continuous and differentiable everywhere. It is a monotonic function, strictly positive and its graph is similar to that of $\exp \nu$. The singularity we observe in the expression for $\nu = 0$ is artificial and should not be considered as such.

Proposition. *If $a \geq 0$ then the solution of (31) is*

$$v(t) = \frac{e^{2b\bar{v}t}(\bar{v}+v_0)+v_0-\bar{v}}{1+e^{2b\bar{v}t}+2v_0bt\cdot \mathrm{expc}(2b\bar{v}t)} \tag{35}$$

Note that, thanks to the use of the expc function, this expression for $v(t)$ has no more singularity for $t=0$.

Proof. Let us first check that the formula is correct when $a>0$. We have

$$\begin{array}{rcccl} v(t) & = & \bar{v}\frac{\frac{\bar{v}+v_0}{\bar{v}-v_0}e^{2b\bar{v}t}-1}{\frac{\bar{v}+v_0}{\bar{v}-v_0}e^{2b\bar{v}t}+1} & = & \bar{v}\frac{(\bar{v}+v_0)e^{2b\bar{v}t}-(\bar{v}-v_0)}{(\bar{v}+v_0)e^{2b\bar{v}t}+(\bar{v}-v_0)} \\ & = & \bar{v}\frac{\bar{v}\left(e^{2b\bar{v}t}-1\right)+v_0\left(e^{2b\bar{v}t}+1\right)}{\bar{v}\left(e^{2b\bar{v}t}+1\right)+v_0\left(e^{2b\bar{v}t}-1\right)} & = & \frac{\bar{v}\left(e^{2b\bar{v}t}-1\right)+v_0\left(e^{2b\bar{v}t}+1\right)}{\left(e^{2b\bar{v}t}+1\right)+v_0\left(\frac{e^{2b\bar{v}t}-1}{\bar{v}}\right)} \\ & = & \frac{\bar{v}\left(e^{2b\bar{v}t}-1\right)+v_0\left(e^{2b\bar{v}t}+1\right)}{\left(e^{2b\bar{v}t}+1\right)+2v_0bt\cdot\mathrm{expc}(2b\bar{v}t)} & & \end{array} \tag{36}$$

Let is now check that the formula is correct when $a=0$. Since $\bar{v}=\sqrt{\frac{a}{b}}=0$, we have

$$v(t) = \frac{v_0\left(e^0+1\right)}{1+e^0+2v_0bt} = \frac{v_0}{1+v_0bt}. \tag{37}$$

Thus

$$\begin{array}{rcc} & \dot{v}=a-bv^2 & \\ \Leftrightarrow & \dot{v}=-bv^2 & \\ \Leftrightarrow & v_0\frac{-v_0b}{(1+v_0bt)^2} = -b\left(\frac{v_0}{1+v_0bt}\right)^2 & \end{array} \tag{38}$$

which is true. □

Proposition. *If $a<0$ then the solution of (31) is*

$$\begin{array}{rcl} v(t) & = & \bar{v}\tan\left(\mathrm{atan}\,\frac{v_0}{\bar{v}}-b\bar{v}t\right) \\ \bar{v} & = & -\sqrt{\frac{-a}{b}} \end{array} \tag{39}$$

where

$$t<t_1=\frac{1}{-b\bar{v}}\left(\frac{\pi}{2}-\mathrm{atan}\,\frac{v_0}{\bar{v}}\right). \tag{40}$$

The change of sign for $v(t)$ is obtained for

$$t_2=\frac{1}{b\bar{v}}\,\mathrm{atan}\,\frac{v_0}{\bar{v}}. \tag{41}$$

Proof. First, note that $a=-\bar{v}^2b$. We have

$$\begin{array}{rcc} & \dot{v}=a-bv^2 & \\ \Leftrightarrow & \frac{d}{dt}\left(\bar{v}\tan\left(\mathrm{atan}\,\frac{v_0}{\bar{v}}-b\bar{v}t\right)\right) = -\bar{v}^2b-b\bar{v}^2\tan^2\left(\mathrm{atan}\,\frac{v_0}{\bar{v}}-b\bar{v}t\right) & \\ \Leftrightarrow & \left(1+\tan^2\left(\mathrm{atan}\,\frac{v_0}{\bar{v}}-b\bar{v}t\right)\right)\cdot\underbrace{\frac{d}{dt}\left(\mathrm{atan}\,\frac{v_0}{\bar{v}}-b\bar{v}t\right)}_{-b\bar{v}} = -\bar{v}b\left(1+\tan^2\left(\mathrm{atan}\,\frac{v_0}{\bar{v}}-b\bar{v}t\right)\right) & \\ \Leftrightarrow & 1+\left(\tan\left(\mathrm{atan}\,\frac{v_0}{\bar{v}}-b\bar{v}t\right)\right)^2 = 1+\left(\tan\left(\mathrm{atan}\,\frac{v_0}{\bar{v}}-b\bar{v}t\right)\right)^2 & \end{array}$$

which is true. The integration is possible until

$$\operatorname{atan}\frac{v_0}{\bar{v}} - b\bar{v}t \in \left]-\frac{\pi}{2}, \frac{\pi}{2}\right[\tag{42}$$

This condition is satisfied for $t = 0$. It will still be satisfied until

$$\begin{array}{rl} & \operatorname{atan}\frac{v_0}{\bar{v}} - b\bar{v}t \leq \frac{\pi}{2} \\ \Leftrightarrow \qquad & t \leq t_1 = \frac{1}{-b\bar{v}}\left(\frac{\pi}{2} - \operatorname{atan}\frac{v_0}{\bar{v}}\right) \end{array} \tag{43}$$

For this t_1, the solution is at infinity. For the initialization, we need to have $\operatorname{atan}\frac{v_0}{\bar{v}} \in]-\frac{\pi}{2}, \frac{\pi}{2}[$ which is always the case. To get the time of change of sign for $v(t)$, we solve:

$$\operatorname{atan}\frac{v_0}{\bar{v}} - b\bar{v}t_2 = 0. \tag{44}$$

Thus

$$t_2 = \frac{1}{b\bar{v}}\operatorname{atan}\frac{v_0}{\bar{v}}. \tag{45}$$

□

Corollary. *The solution of the Riccati equation* $\dot{v} = a - bv^2$ *for* $b > 0$ *is*

$$\begin{array}{rcccl} v(t) & = & \psi^+_{a,b,v_0}(t) & = & \frac{e^{2b\bar{v}t}(\bar{v}+v_0)+v_0-\bar{v}}{1+e^{2b\bar{v}t}+2v_0bt\,\mathrm{expc}(2b\bar{v}t)} \qquad \text{if } a \geq 0 \\ & = & \psi^-_{a,b,v_0}(t) & = & \bar{v}\tan\left(\operatorname{atan}\frac{v_0}{\bar{v}} - b\bar{v}t\right) \qquad \text{if } a < 0 \end{array} \tag{46}$$

where $\bar{v} = \operatorname{sign}(a)\sqrt{\frac{|a|}{b}}$.

5 Sinking body problem

We consider again the equation of the sinking body given by

$$\dot{v} = a - bv|v| \tag{47}$$

where $b > 0$. This equation is close to the Riccati equation $\dot{v} = a - bv^2$. Equivalently (47) can be seen as a piecewise Riccati equation. In this section, we propose to find an analytic solution for the solution $v(t)$. This expression is needed to build an interval flow for (47) which will then be used to integrate our float with interval uncertainties.

5.1 Analytical expression of the solution of the sinking body motion

From the analytical solution of the Riccati equation, we can get an analytical expression of the sinking body motion in the case where the parameters a, b are constant.

Proposition. *The solution of* $\dot{v} = a - bv|v|$ *is*

$$\varphi_{a,b,v_0}(t) = \begin{cases} \operatorname{sign}(a) \cdot \psi^+_{|a|,b,\operatorname{sign}(a)\cdot v_0}(t) & \text{if } av_0 \geq 0 \\ -\operatorname{sign}(a) \cdot \psi^-_{-|a|,b,-\operatorname{sign}(a)\cdot v_0}(t) & \text{if } av_0 < 0 \text{ and } t \leq t_2 \\ \operatorname{sign}(a) \cdot \psi^+_{|a|,b,0}(t - t_2) & \text{if } av_0 < 0 \text{ and } t > t_2 \end{cases} \tag{48}$$

where $t_2 = -\frac{\operatorname{sign}(a)}{\sqrt{|a|b}} \operatorname{atan}\left(v_0\sqrt{\frac{b}{|a|}}\right)$, *where* ψ^- *and* ψ^+ *are defined by (46).*

Proof. If $a = 0$ or $v_0 = 0$, we have $av_0 \geq 0$ and we easily check that the Proposition is valid. We need to check four cases.

Case 1 $a > 0, v_0 > 0$. We have $\dot{v} = a - bv^2$ which is a Riccati equation. From Corollary 4, $v(t) = \psi^+_{a,b,v_0}(t)$ for all $t \geq 0$.

Case 2 $a < 0, v_0 < 0$. We have $\dot{v} = a - bv|v| = a + bv^2$. Set $w = -v$. We have $-\dot{w} = a + bw^2$, *i.e.*, $\dot{w} = (-a) - bw^2$. Thus $w(t) = \psi^+_{-a,b,w_0}(t)$ and finally

$$v(t) = -\psi^+_{-a,b,-v_0}. \tag{49}$$

Case 3 $a < 0, v_0 > 0$. We have $\dot{v} = a - bv^2$, which is again a Riccati equation. From Corollary 4, we get

$$\begin{array}{rcll} v(t) & = & \psi^-_{a,b,v_0}(t) & \text{if } t \leq t_2 = \frac{1}{b\bar{v}} \operatorname{atan} \frac{v_0}{\bar{v}} \\ v(t) & = & -\psi^+_{-a,b,0}(t - t_2) & \text{if } t > t_2 \end{array} \tag{50}$$

Case 4 $a > 0, v_0 < 0$. We have $\dot{v} = a - bv|v| = a + bv^2$. We get

$$\begin{array}{rcll} v(t) & = & -\psi^-_{-a,b,-v_0}(t) & \text{if } t \leq t_2 = -\frac{1}{b\bar{v}} \operatorname{atan} \frac{v_0}{\bar{v}} \\ v(t) & = & \psi^+_{a,b,0}(t - t_2) & \text{if } t > t_2 \end{array} \tag{51}$$

□

5.2 Interval flow of the sinking body motion

Corollary. *If* $v_0 \in [v_0^-, v_0^+]$, $a \in [a^-, a^+], b \in [b^-, b^+]$ *and* $t \in [t^-, t^+]$, *we have*

$$\varphi_{a,b,v_0}(t) \in [\varphi]_{[a],[b],[v_0]}([t]) \tag{52}$$

where

$$(i) \quad [\varphi]_{[a],[b],[v_0]}([t])) = [\varphi]_{a^-,[b],v_0^-}([t]) \sqcup [\varphi]_{a^+,[b],v_0^+}([t]) \tag{53}$$

$$(ii) \quad [\varphi]_{a,[b],v_0}([t]) = [\varphi]_{a,[b],v_0}(\{t^-, t^+\}) \tag{54}$$

$$(iii) \quad [\varphi]_{a,[b],v_0}(t) = \begin{cases} \sigma \cdot [\psi^+]_{|a|,[b],\sigma v_0}(t) & \text{if } a \cdot v_0 \geq 0 \\ [\hat{\varphi}]_{a,[b],v_0}(t) & \text{if } a \cdot v_0 < 0 \end{cases} \tag{55}$$

$$(iv) \quad [\hat{\varphi}]_{a,[b],v_0}(t) = \begin{cases} \varphi_{a,\{b^-,b^+\},v_0}(t)) & \text{if } t \notin [t_2] \\ \sigma \cdot [\psi^+]_{|a|,[b],0}(t - [t_2]) & \text{if } t \in [t_2] \\ [t_2] = t_2(v_0, a, \{b^-, b^+\}) \end{cases} \tag{56}$$

$$(v) \quad [\psi^+]_{a,[b],v_0}(t) = \psi^+_{a,\{b^-,b^+\},v_0}(t) \tag{57}$$

where

$$\begin{array}{rcl} t_2(v_0, a, b) & = & -\frac{\sigma}{\sqrt{|a| \cdot b}} \operatorname{atan}\left(v_0 \sqrt{\frac{b}{|a|}}\right) \\ \sigma & = & \operatorname{sign}(a) \end{array}$$

and $\sqcup$ *denotes the interval hull operator.*

Remark. In the previous formulas, we use an enumeration notation with braces. The resulting calculus returns the smallest interval which contains all possibilities obtained from the list. For instance

$$\begin{array}{rcll} \sin(\{0,1\}) & = & [0, \sin(1)] & \\ [\varphi]_{a,[b],v_0}(\{t^-, t^+\}) & = & [\varphi]_{a,[b],v_0}(t^-) \sqcup [\psi]_{a,[b],v_0}(t^+) & \text{(see Corollary 5.2, (ii))} \\ \psi^+_{a,\{b^-,b^+\},v_0}(t) & = & \left[\left\{\psi^+_{a,b^-,v_0}(t), \psi^+_{a,b^+,v_0}(t)\right\}\right] & \text{(see Corollary 5.2, (v))} \end{array}$$

Proof. (i) Using the comparison theorem, we have

$$v_0 \in [v_0^-, v_0^+], a \in [a^-, a^+] \Rightarrow \varphi_{a,b,v_0}(t) \in \left[\varphi_{a^-,b,v_0^-}(t), \varphi_{a^+,b,v_0^+}(t)\right].$$

It suffices to enclose the two quantities $\psi_{a^-,b,v_0^-}(t)$ and $\psi_{a^+,b,v_0^+}(t)$.

(ii) The signal $\dot{v}(t)$ can never change of sign. Indeed, $\dot{v}(t) = 0$ if $a - bv|v| = 0$ and in this case, $\dot{v}(t) = 0$ for all t. As a consequence, the extreme values for $\varphi_{a,b,v_0}(t)$ are obtained for $t \in \{t^-, t^+\}$.

(iii) Assume that (a, v_0, t) is fixed. If $a \cdot v_0 \geq 0$, we have no stop point. Thus $\varphi_{a,b,v_0}(t) = \sigma \cdot {\psi^+}_{|a|,[b],\sigma v_0}(t)$ as seen in 48. Otherwise, we are in a situation with a stop point.

(iv) We have a stop point. This stop point can be inside or outside the time window $[t]$. We have

$$\frac{\partial \varphi_{a,b,v_0}(t)}{\partial b} = 0 \Leftrightarrow t = t_2 = -\frac{\operatorname{sign}(a)}{\sqrt{|a|b}} \operatorname{atan}\left(v_0 \sqrt{\frac{b}{|a|}}\right). \tag{58}$$

Thus, if $t \notin [t_2]$, where $[t_2] = t_2(v_0, a, \{b^-, b^+\})$, $\varphi_{a,b,v_0}(t)$ is monotonic in t and thus

$$\varphi_{a,b,v_0}(t) \in \varphi_{a,\{b^-,b^+\},v_0}(t) \tag{59}$$

otherwise

$$\varphi_{a,b,v_0}(t) \in \sigma \cdot {\psi^+}_{|a|,[b],0}(t - t_2(v_0, a, \{b^-, b^+\})). \tag{60}$$

(v) The result comes from the monotonicity of ψ^+ with respect to b. □

Corollary. *An interval flow of the sinking body motion is:*

$$\Phi_f : \begin{array}{ccc} \mathbb{IR} \times \mathbb{IR}^2 & \to & \mathbb{IR} \times \mathbb{IR} \\ ([v_0], [\![a]\!], [\![b]\!]) & \to & \begin{pmatrix} [v] \\ [\![v]\!] \end{pmatrix} = \begin{pmatrix} [\varphi]_{[\![a]\!],[\![b]\!],[v_0]}(\delta) \\ [\varphi]_{[\![a]\!],[\![b]\!],[v_0]}([0,\delta]) \end{pmatrix} \end{array} \tag{61}$$

5.3 Example

We take again four cases already treated in Subsection 2.2 (Figure 8).

Case a We have $(v_0, a, b) \in [0.9, 1.1] \times [0.9, 1.1] \times [1.9, 2.1]$. In red, we have the envelope already obtained by the Runge-Kutta method. No pessimism can be observed which is consistent with the fact that $\varphi_{a,b,v_0}(t)$ is monotonic. The magenta bar corresponds to the initial interval $[0.9, 1.1]$ for v_0.

Case b We have $(v_0, a, b) \in [-1.1, -0.9] \times [-1.1, -0.9] \times [1.9, 2.1]$. The envelope is symmetrical to that obtained for Case a. Again, due to the monotonicity $\varphi_{a,b,v_0}(t)$, no pessimism can be observed.

Case c We have $(v_0, a, b) \in [0.9, 1.1] \times [-1.1, -0.9] \times [1.9, 2.1]$. The pessimism of the enclosure is too small to be observed; compared to the trajectories obtained by a Runge Kutta integration (red). From the tube, we can conclude that the speed of the float will cancel and the float will come back.

Case d We have $(v_0, a, b) \in [-1.1, -0.9] \times [0.9, 1.1] \times [1.9, 2.1]$. The situation is similar to that given in Case c.

In the figures, the units are t(sec) and v (m/sec).

6 Online integration of the float

Consider again the float described by Equation (1). Due to the serial structure of the system, we can integrate the differential inclusion using interval flows for each component, as explained in Subsection 3.4. The corresponding decomposition is expressed by Figure 9 and is consistent with the initial goal (see Equation 5). For each sampling time, five steps have to be performed sequentially in the right order. Between sampling times k to $k+1$, three intervals have to be transmitted through the memory: $[s](k), [v](k), [d](k)$.

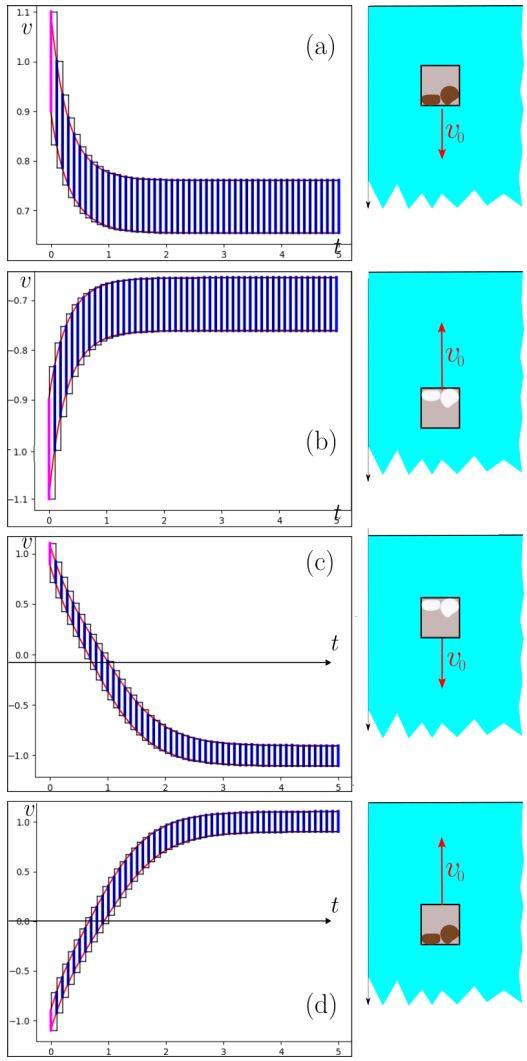


Figure 8: Sinking body for four different initializations. The tubes contains the trajectory $v(t)$

At Step 1, we read the input slice $[\![u]\!](k)$ which contains all $u(t)$ for $t \in [(k-1)\delta, k\delta]$. At Step 2, we integrate $[\![u]\!]$ using the interval flow for the integrator presented in Subsection 3.2. As a result, we get a slice $[\![s]\!](k)$ for $s(t)$. Using a static interval evaluation, we then get at Step 3 two slices $[\![a]\!](k)$ for $a(t)$ and $[\![b]\!](k)$ for $b(t)$. These two slices will then feed the interval flow $[\varphi]_{[\![a]\!],[\![b]\!],[v]}$ which yields the slice $[\![v]\!](k)$ and the gate $[v](k)$ at Step 4. The slice $[\![v]\!](k)$ is then used at Step 5 by the last block to generate the slice $[\![d]\!](k)$ and the gate $[d](k)$.

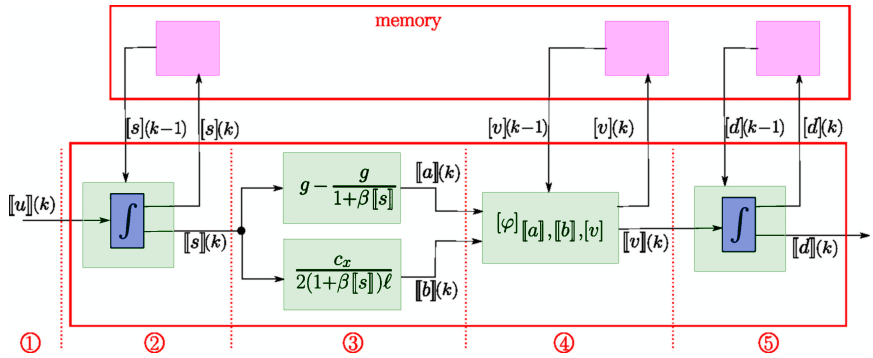


Figure 9: Sequence to be followed for one step interval integration

The resulting computations correspond to the following algorithm.

$$
\begin{array}{ll}
\text{In:} & [d_0], [v_0], [s_0] \\
\text{Init} & [d] = [d_0] \\
& [v] = [v_0] \\
& [s] = [s_0]
\end{array}
$$

$$
\begin{array}{ll}
\text{Main loop} & \text{For } k = 1 \text{ to } k_{\max} \\
\text{Step 1} & \text{Read } [\![u]\!] = [\![u]\!](k) \\
\text{Step 2} & [\![s]\!] = [s] + [\![u]\!] \cdot [0, \delta] \\
& [s] = [s] + [\![u]\!] \cdot \delta \\
\text{Step 3} & [\![a]\!] = g \cdot \left(1 - \frac{1}{1+\beta[\![s]\!]}\right) \\
& [\![b]\!] = \frac{c_x}{2(1+\beta[\![s]\!])\ell} \\
\text{Step 4} & [\![v]\!] = [\varphi]_{[\![a]\!],[\![b]\!],[v]}([0, \delta]) \\
& [v] = [\varphi]_{[\![a]\!],[\![b]\!],[v]}(\delta) \\
\text{Step 5} & [\![d]\!] = [d] + [\![v]\!] \cdot [0, \delta] \\
& [d] = [d] + [\![v]\!] \cdot \delta \\
& \text{write}(k, [d], [\![d]\!], [v], [\![v]\!], [s], [\![s]\!])
\end{array}
$$

The only memory needed by this interval simulator are the three gates $[s], [v], [d]$.

The behavior of the interval simulator is illustrated by Figure 10. We took $g_0 = 9.81m \cdot s^{-2}, \ell = 1m, \beta = 0.1, c_x = 0.9$ for the parameters and $[s_0] = [v_0] = [d_0] = [0, 0.1]$ for the initial conditions. For the input, we took $[\![u]\!](k) = \exp(-[(k-1)\delta, k\delta])$. In the figures, the chosen units are t(sec), d(m) and v (m/sec).

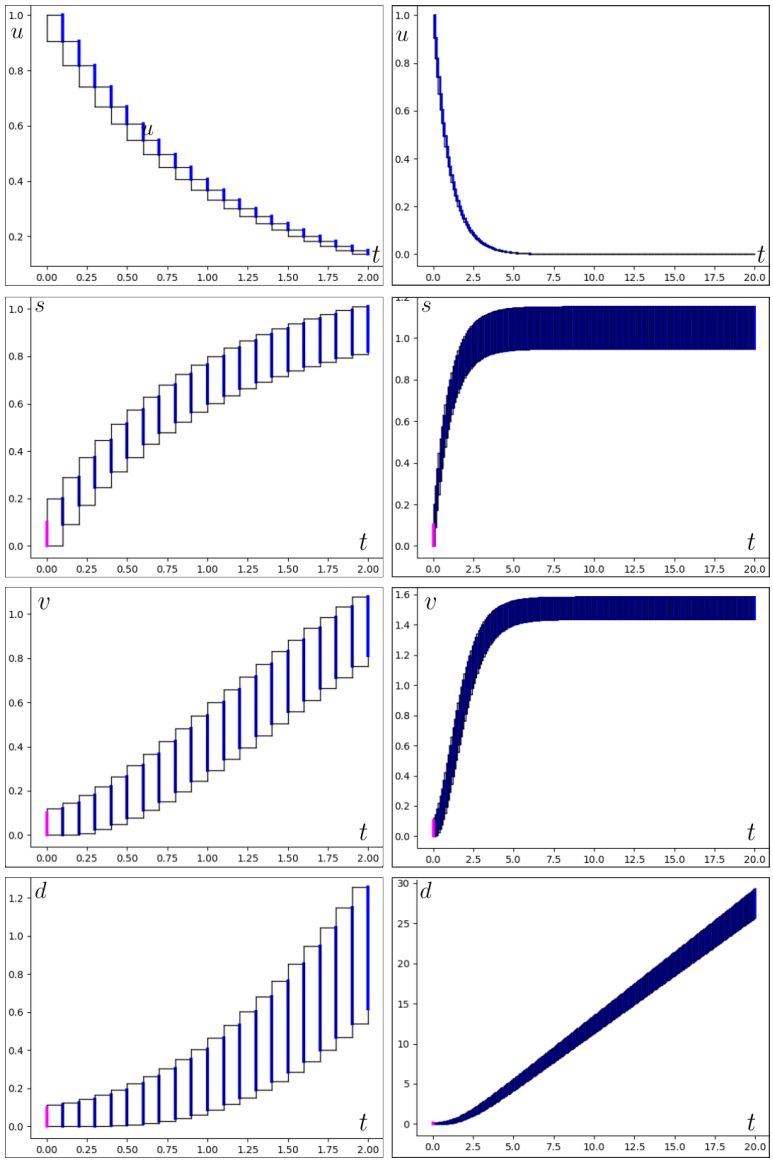


Figure 10: Sinking float. The tubes contain the trajectories $u(t), s(t), v(t), d(t)$ for $t \in [0, 2]$ (left) and for $t \in [0, 20]$ (right)

Even if the system is unstable in the Lyapunov sense (indeed the variable d tends to infinity), we do not observe any exponential explosion of the pessimism, unlike other existing interval methods dealing with differential inclusions.

The implementation is done using the Codac library [23] and the source codes are available at https://www.ensta-bretagne.fr/jaulin/reachfloat.html.

7 Conclusion

In this paper, a new interval estimator has been proposed for online state prediction. For this, we have introduced the concept of interval flow that has to be found analytically for each component of the whole system. Combining the interval flows of all subsystems, we have shown that an interval estimator containing the state variables in a guaranteed way could be derived. For simplicity, but also to evaluate the accuracy of the approach, only the reachability problem has been addressed. This means that no exteroceptive measurements (*i.e.*, collected by a sensor able to interact with the environment, such as a camera, a lidar or a radar) have been taken into account in order to contract the domains for the state variables. The interval state estimator that has been obtained has a fixed number of operations to be performed at each sampling time. This is a strong requirement rarely considered by classical interval algorithms. Indeed, existing interval algorithms dealing with differential inclusions use fixed point procedures that are not consistent with real-time issues. Through an example taken from robotics (an underwater robot with a ballast), we have shown that it was possible to deal with engineering systems efficiently.

The presented approach can be applied to a complex system as soon as it can be built using a parallel and a serial composition of specific scalar systems [12]. More precisely, these scalar systems should have a single state variable, may have several inputs, and an analytical solution should be available for constant inputs. The existence of such an analytical solution could be relaxed if we accept to use an interval resolution of a differential equation based on the Picard operator [13].

References

[1] Alamo, T., Bravo, J., and Camachoa, E. Guaranteed state estimation by zonotopes. *Automatica*, 41(6):1035–1043, 2005. DOI: 10.1016/j.automatica.2004.12.008.

[2] Alexandre dit Sandretto, J. and Chapoutot, A. Validated simulation of differential algebraic equations with Runge-Kutta methods. *Reliable Computing*, 22:56–77, 2016. DOI: 10.1007/s11155-016-0001-2.

[3] Althoff, M. and Krogh, B. Zonotope bundles for the efficient computation of reachable sets. In *Proceedings of the 2011 50th IEEE Conference on Decision and Control and European Control Conference*, pages 6814–6821, 2011. DOI: 10.1109/CDC.2011.6160872.

[4] Asarin, E., Dang, T., and Girard, A. Reachability analysis of nonlinear systems using conservative approximation. In Maler, O. and Pnueli, A., editors, *Hybrid*

Systems: Computation and Control, Volume 2623 of *Lecture Notes in Computer Science*, pages 20–35. Springer, 2003. DOI: 10.1007/3-540-36580-X_5.

[5] Aubin, J. and Frankowska, H. *Set-Valued Analysis*. Birkhäuser, Boston, Boston, MA, 1990. DOI: 10.1007/978-0-8176-4848-0.

[6] Collins, P. and Goldsztejn, A. The reach-and-evolve algorithm for reachability analysis of nonlinear dynamical systems. *Electronic Notes in Theoretical Computer Science*, 223:87–102, 2008. DOI: 10.1016/j.entcs.2008.12.033.

[7] Combastel, C. A state bounding observer for uncertain non-linear continuous-time systems based on zonotopes. In *Proceedings of the 44th IEEE Conference on Decision and Control*, pages 7228–7234. IEEE, 2005. DOI: 10.1109/CDC.2005.1583327.

[8] Efimov, D. and Raïssi, T. Design of interval observers for uncertain dynamical systems. *Automation and Remote Control*, 77(2):191–225, 2016. DOI: 10.1134/S0005117916020016.

[9] Frehse, G. PHAVer: Algorithmic verification of hybrid systems past HyTech. *International Journal on Software Tools for Technology Transfer*, 10:263–279, 2008. DOI: 10.1007/s10009-007-0062-x.

[10] Goubault, E., Mullier, O., Putot, S., and Kieffer, M. Inner approximated reachability analysis. In *Proceedings of the 17th International Conference on Hybrid Systems: Computation and Control*, pages 163–172. ACM, 2014. DOI: 10.1145/2562059.2562113.

[11] Guernic, C. L. and Girard, A. Reachability analysis of linear systems using support functions. *Nonlinear Analysis: Hybrid Systems*, 4(2):250–262, 2010. DOI: 10.1016/j.nahs.2009.03.003.

[12] Jaulin, L. Integral algebra for simulating dynamical systems with interval uncertainties. *International Journal of Approximate Reasoning*, 178, 2025. DOI: doi.org/10.1016/j.ijar.2024.109353.

[13] Kapela, T., Mrozek, M., Wilczak, D., and Zgliczynski, P. CAPD: DynSys, A flexible C++ toolbox for rigorous numerical analysis of dynamical systems. *Communications in Nonlinear Science and Numerical Simulation*, 101:105578, 2021. DOI: 10.1016/j.cnsns.2020.105578.

[14] Kieffer, M., Jaulin, L., and Walter, E. Guaranteed recursive nonlinear state estimation using interval analysis. In *Proceedings of the 37th IEEE Conference on Decision and Control*, pages 3966–3971, Tampa, FL, USA, 1998. IEEE. DOI: 10.1109/CDC.1998.761926.

[15] LaValle, S. *Planning Algorithm*. Cambridge University Press, 2006. DOI: 10.1017/CBO9780511546877.

[16] Mitchell, I., Bayen, A., and Tomlin, C. Validating a Hamilton-Jacobi approximation to hybrid system reachable sets. In Benedetto, M. and Sangiovanni-Vincentelli, A., editors, *Hybrid Systems: Computation and Control*, Volume 2034 of *Lecture Notes in Computer Science*, pages 418–432. Springer Berlin Heidelberg, 2001. DOI: 10.1007/3-540-45351-2_34.

[17] Moore, R. *Methods and Applications of Interval Analysis*. Society for Industrial and Applied Mathematics, 1979. DOI: 10.1137/1.9781611970906.

[18] Munson, B., Young, D., Okiishi, T., and Huebsch, W. *Fundamentals of Fluid Mechanics*. John Wiley & Sons, 7th edition, 2013. DOI: 10.1002/9781118912652.

[19] Polyanin, A., Andrei, D., and Zaitsev, V. *Handbook of Exact Solutions for Ordinary Differential Equations*. Chapman and Hall, CRC, 2nd edition, 2003. DOI: 10.1201/9781420035330.

[20] Ramdani, N. and Nedialkov, N. Computing reachable sets for uncertain nonlinear hybrid systems using interval constraint-propagation techniques. *Nonlinear Analysis: Hybrid Systems*, 5(2):149–162, 2011. DOI: 10.1016/j.nahs.2010.05.010.

[21] Rauh, A., Kersten, J., and Aschemann, H. Techniques for verified reachability analysis of quasi-linear continuous-time systems. In *Proceedings of the 2019 24th International Conference on Methods and Models in Automation and Robotics*, pages 18–23, 2019. DOI: 10.1109/MMAR.2019.8864648.

[22] Rauh, A., Lahme, M., Rohou, S., Jaulin, L., Dinh, T., Raïssi, T., and Fnadi, M. Offline and online use of interval and set-based approaches for control and state estimation: A review of methodological approaches and their application. *Logical Methods in Computer Science*, 2023. DOI: 10.48550/arXiv.2309.11622.

[23] Rohou, S., Desrochers, B., and Bars, F. L. The codac library. *Acta Cybernetica*, 26(4):871–887, 2024. DOI: 10.14232/ACTACYB.302772.

[24] Rohou, S., Jaulin, L., Mihaylova, L., Bars, F. L., and Veres, S. *Reliable Robot Localization*. Wiley, 2019. DOI: 10.1002/9781119680970.

[25] Smith, H. Monotone dynamical systems: An introduction to the theory of competitive and cooperative systems. *Mathematical Surveys and Monographs*, 41, 1995. DOI: 10.1090/surv/041.

[26] Taha, W. and Duracz, A. Acumen: An open-source testbed for cyber-physical systems research. In *Conference on CYber physiCaL systems, iOt and sensors Networks*, 2015. DOI: 10.1007/978-3-319-47063-4_11.

[27] Wan, J. *Computationally reliable approaches of contractive model predictive control for discrete-time systems*. PhD dissertation, Universitat de Girona, Girona, Spain, 2007. URL: http://www.tdx.cat/TDX-1008107-141828.

Acta Cybernetica **27** (2026) 337–357.

Ultra-Wideband Smart Wheelchair Pose Estimation Using Interval Analysis

Théo Le Terrier[ab], Marie Babel[ac], and Vincent Drevelle[ad]

Abstract

In this paper, we introduce a reliable positioning method for a smart wheelchair by using ultra-wideband (UWB) technology. This method provides confidence domains of the pose by assuming bounded measurement errors and proprioceptive information, without any assumptions of independence. Exploiting interval analysis and constraint propagation techniques, we characterize the sets of all feasible poses that are consistent with the measurements. Our method has been validated through experiments with a smart power wheelchair equipped with UWB sensors in realistic conditions. The results demonstrate that our approach consistently provides guaranteed uncertainty domains with 100% integrity across the tested dataset, even when faced with inconsistent measurements. In addition, we compare our interval-based method with M-estimator approaches, and show that while achieving slightly worse positioning accuracy, our method offers superior consistency.

Keywords: localization, interval analysis, ultra-wideband, smart wheelchair

1 Introduction

Preserving independence and access to social activities is essential for wheelchair users. Indeed, a decrease in mobility can lead to reduced self-esteem, isolation and fear of abandonment [9]. However, operating power wheelchairs requires significant driving skills, which might prove infeasible for people with cognitive impairments or visual deficiencies. To make this assistive technology more accessible, *smart wheelchairs* have been proposed by enhancing standard power wheelchairs with sensors and embedded systems. These advanced robotic assistive devices aim to provide safe navigation assistance, thereby preserving the user's independence and social engagement. Both autonomous [27, 24] and semi-autonomous [25, 7] navigation systems have emerged to provide the right level of assistance to the user.

[a]Univ Rennes, INSA Rennes, Inria, CNRS, IRISA – UMR 6074, F-35000 Rennes, France
[b]E-mail: theo.le-terrier@irisa.fr, ORCID: 0009-0000-8800-7159
[c]E-mail: marie.babel@irisa.fr, ORCID: 0000-0001-6425-389X
[d]E-mail: vincent.drevelle@irisa.fr, ORCID: 0000-0001-9579-7793

DOI: 10.14232/actacyb.315261

Fully autonomous navigation systems face the challenge of localization, which can be addressed with integrated smart wheelchair sensors. For widespread adoption, the localization system should offer low cost, easy attachment to any smart wheelchair, robustness and efficiency [20]. Also, for privacy concerns and acceptability reasons, visual sensors are avoided.

This paper proposes a reliable positioning method for a smart wheelchair, using a set of ultra-wideband (UWB) sensors [3]. The latter have gained popularity for indoor robotics where GNSS signals are unavailable, and commercially available sensors such as the DWM1001 by Qorvo meet the aforementioned requirements. A typical UWB-based positioning setup involves fixed UWB nodes (*anchors*), measuring their distance from UWB nodes on the robot (*tags*). Statistical approaches have been commonly considered to design UWB based positioning systems: nonlinear least squares [2, 17], or Kalman filters [21, 22]. In these papers, UWB measurement error distributions are modeled as Gaussian, and independence between each measurement from one time to another is assumed. Such assumptions are reasonable when considering line-of-sight (LOS) signal propagation. The case of non-line-of-sight (NLOS) signal propagation in multipath environments was also widely explored in the field of probabilistic methods [5, 16], with e.g., particle filters [11]. Furthermore, with access to the UWB channel estimation data and additional prior knowledge on the initial receiver position and moving direction, the localization process can be based on the estimated multipath components [10].

An alternative to make as less assumptions as possible is to assume only a *bounded-error model*, meaning that measurement errors and model parameters are within known bounds. Set-membership estimators then compute a domain of solutions consistent with the model and the measurements without removing any feasible solution. Interval analysis [14] provides tools to compute such domains and has been used for reliable pose estimation [23, 26] in various contexts, including aerial robotics [18], marine applications [28], and underwater robotics [13, 29]. These systems typically use sensors like cameras or sonars. GNSS receivers have also been employed for reliable positioning of outdoor vehicles [8, 30], sometimes in combination with UWB [1]. However, to our knowledge, interval methods have not been applied to indoor UWB localization.

The main contribution of this paper is a reliable UWB based positioning method using interval analysis and an experimental validation of the latter in a realistic scenario, involving a user in a smart wheelchair. Our approach is compared with M-estimator methods (classical Huber [12], and the half-Cauchy of [16]).

This paper is structured as follows. We first provide an overview of the pose estimation problem and introduce interval analysis and robust set inversion. We then present our proposed method and conclude with experimental results using UWB sensors mounted on a smart wheelchair.

2 Problem Statement

In this paper, we aim to provide a reliable 2-D pose estimation of a smart wheelchair equipped with UWB sensors. The wheelchair is operated by a user-controlled joystick. The wheelchair is assumed to move in a horizontal planar world without slipping. The robot configuration is denoted $\mathbf{p} = (x, y, \theta)^T$ where (x, y) are the planar coordinates, and θ is the heading angle between the frame $\mathcal{F}_w$ and the local frame attached to the wheelchair $\mathcal{F}_r$. Assuming that the wheelchair's kinematics is similar to those of a unicycle, its continuous-time model is

$$\begin{cases} \dot{x}(t) = v(t) \cdot \cos\theta(t) \\ \dot{y}(t) = v(t) \cdot \sin\theta(t) \\ \dot{\theta}(t) = \omega(t). \end{cases} \tag{1}$$

The input commands are the linear velocity, denoted v, and the angular velocity, denoted ω. These inputs are estimated by the embedded control system (*power module*) of the power wheelchair from the joystick position. A set of UWB anchors is installed in a room, and distance measurements to UWB tags installed on the wheelchair are performed. An overview of the system is presented in Figure 1.

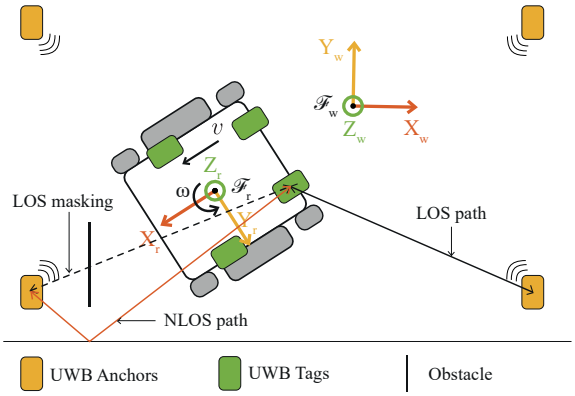


Figure 1: System Overview. Each tag (green) measures its distance to each anchor (yellow), thus estimates its own position in $\mathcal{F}_w$. The wheelchair pose is then deduced, since the positions of the tags are known in $\mathcal{F}_r$

Let n_T be the number of tags installed on the wheelchair, and n_A the number of anchors. We denote $T_{i,\, i\in\{1,\ldots,n_T\}}$, the i^{th} tag, and $A_{j,\, j\in\{1,\ldots,n_A\}}$, the j^{th} anchor. The coordinates of T_i, expressed in $\mathcal{F}_r$, are denoted ${}^r\mathbf{x}_{T_i} = ({}^rx_{T_i}, {}^ry_{T_i}, {}^rz_{T_i})$. The coordinates of T_i in $\mathcal{F}_w$ are denoted $\mathbf{x}_{T_i} = (x_{T_i}, y_{T_i}, z_{T_i})$. The fixed coordinates of A_j in $\mathcal{F}_w$, are denoted $\mathbf{x}_{A_j} = (x_{A_j}, y_{A_j}, z_{A_j})$. The measured range between T_i and A_j is denoted $r_{i,j}$ and is defined as

$$r_{i,j} = \|\mathbf{x}_{T_i} - \mathbf{x}_{A_j}\|_2 + \beta_{i,j}, \tag{2}$$

where $\beta_{i,j}$ is the range measurement error.

3 Interval Analysis and Robust Set Inversion

3.1 Interval Analysis

Interval analysis [14] relates to intervals $[x] = [\underline{x}, \overline{x}]$ and their multidimensional extension, interval vectors or boxes $[\mathbf{x}] = [\underline{\mathbf{x}}, \overline{\mathbf{x}}]$. $\underline{\mathbf{x}}$ and $\overline{\mathbf{x}}$ respectively represent the lower and upper bounds of $[\mathbf{x}]$. The *width* of an interval is defined as the difference between its upper and lower bounds. The width of a box is the width of its largest interval components. The *radius* of an interval, or box, is defined as its half-width. The set of real intervals is denoted $\mathbb{IR}$, and the set of n-dimensional boxes is $\mathbb{IR}^n$.

Let $\mathbf{f}$ be a function from $\mathbb{R}^n$ to $\mathbb{R}^m$. The image by $\mathbf{f}$ of an n-dimensional box $[\mathbf{x}]$ is a set $\mathbf{f}([\mathbf{x}])$ which is not necessarily a box (Figure 2). The *inclusion function* $[\mathbf{f}]$ from $\mathbb{IR}^n$ to $\mathbb{IR}^m$ is then defined to enclose $\mathbf{f}([\mathbf{x}])$ by a box $[\mathbf{f}]([\mathbf{x}])$ such that

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

The smallest box containing $\mathbf{f}([\mathbf{x}])$ is returned by the minimal inclusion function $[\mathbf{f}]^*$. The smallest box enclosing a set $\mathcal{S}$ is denoted $\square\, \mathcal{S}$ and is called the *interval hull* of $\mathcal{S}$. Thus, $[\mathbf{f}]^*([\mathbf{x}]) = \square\, \mathbf{f}([\mathbf{x}])$.

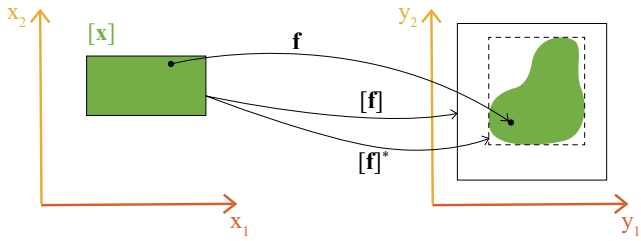


Figure 2: Image of a box $[\mathbf{x}]$ by a function $\mathbf{f}$, an inclusion function $[\mathbf{f}]$, and the minimal inclusion function $[\mathbf{f}]^\star$.

3.2 Contractors

Let $\mathbf{x}$ be a vector which variables are linked by relations, or constraints. A *Constraint Satisfaction Problem* (CSP) can be formulated $\mathcal{H} : \{\mathbf{f}(\mathbf{x}) = \mathbf{0}, \mathbf{x} \in [\mathbf{x}]\}$, where $\mathbf{f}(\mathbf{x}) = \mathbf{0}$ represents the constraints, and $[\mathbf{x}]$ the variables' domain.

Let $\mathcal{S} = \{\mathbf{x} \in [\mathbf{x}] \mid \mathbf{f}(\mathbf{x}) = \mathbf{0}\}$ be the solution set of $\mathcal{H}$. A *contractor* $\mathcal{C}$ [4] is a mapping from $\mathbb{IR}^n$ to $\mathbb{IR}^n$ such that

- $\forall [\mathbf{x}] \in \mathbb{IR}^n, \mathcal{C}([\mathbf{x}]) \subseteq [\mathbf{x}]$ (*contraction*),
- $\forall [\mathbf{x}] \in \mathbb{IR}^n, \mathcal{C}([\mathbf{x}]) \cap \mathcal{S} = [\mathbf{x}] \cap \mathcal{S}$ *(completeness).*

In other words, the variables' domain $[\mathbf{x}]$ can be reduced without excluding any solution by applying a contractor. A contractor $\mathcal{C}^\star$ is *minimal* if $\mathcal{C}^\star([\mathbf{x}])$ is the smallest box containing $\mathcal{S} \cap [\mathbf{x}]$, i.e., $\mathcal{C}^\star([\mathbf{x}]) = \square\, \mathcal{S} \cap [\mathbf{x}]$. In Figure 3 (a), $\mathcal{C}^\star$ is the minimal contractor.

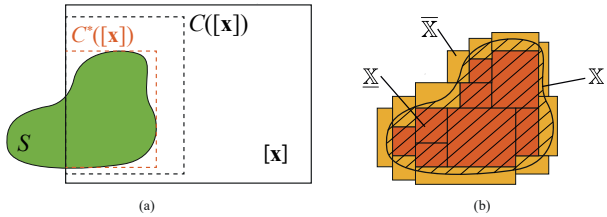


Figure 3: (a) Contractors reducing the variables' domain of a box $[\mathbf{x}]$. (b) Inner (orange) and outer (yellow) approximation of a set $\mathbb{X}$ by subpavings.

3.3 Robust Set Inversion

Let $\mathbb{Y}$ be a known subset of $\mathbb{R}^m$, e.g., obtained from m measurements. Set inversion problem [14] consists in characterizing

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

Two subpavings (sets of non-overlapping boxes) that provide an *inner* and *outer* approximation of $\mathbb{X}$, i.e. such that $\underline{\mathbb{X}} \subset \mathbb{X} \subset \overline{\mathbb{X}}$, can be computed with the *Set Inversion via Interval Analysis* (SIVIA) algorithm [14]. Such subpavings are shown in Figure 3 (b). The SIVIA algorithm employs a branch and bound strategy. Starting from an initial domain $[\mathbf{x_0}]$ to which $\overline{\mathbb{X}}$ is guaranteed to belong, SIVIA performs successive contractions and bisections to refine the inner and outer approximations of the solution set. Algorithm 1 describes our *anytime* implementation, with a breadth-first exploration strategy that can be stopped after a *timeout*.

Algorithm 1 Anytime SIVIA(in: $[\mathbf{x_0}]$, $\mathcal{C}$, ϵ, *timeout* ; out: $\overline{X}$)

```
1:  X̄ ← ∅                                          Outer subpaving of the solution set
2:  L ← ([x0])                                     Add initial domain to the FIFO queue
3:  while L ≠ ∅ and time() ≤ timeout do
4:     ([x], L) ← L                                Dequeue a box from the FIFO
5:     [xc] ← C([x])                               Contract the box
6:     if [xc] ≠ ∅ then
7:        if width([xc]) > ε then                  Bisect large boxes
8:           ([x1], [x2]) ← bisect([xc])          Bisect the box
9:           L ← (L, [x1], [x2])                  Enqueue the subboxes
10:       else                                     Don't further process small boxes
11:          X̄ ← X̄ ∪ [xc]
12:       end if
13:    end if
14: end while
15: X̄ ← X̄ ∪ L                                     Add pending boxes to the solution
```

Sensor measurements are affected by noise and may contain outliers due, for instance, to NLOS signal propagation. The set $\mathbb{X}$ of all parameters consistent with the measurements might then be empty, or do not contain the true configuration. A solution to overcome this situation is to define $\mathbb{X}$ as the set of all parameters consistent with at least $m-q$ measurements. This method is the so-called *q-relaxed intersection* [19], denoted $\bigcap^{\{q\}}$ and represented in Figure 4.

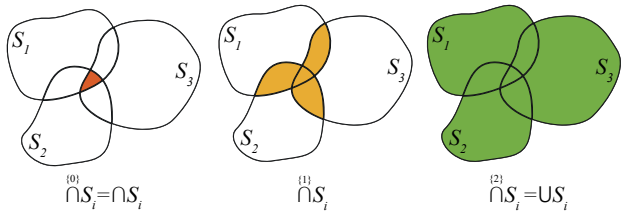


Figure 4: q-relaxed intersection for $q \in \{0, 1, 2\}$

Computing a reliable pose confidence domain, i.e., a box in which the wheelchair is guaranteed to belong, only requires an outer approximation of $\mathbb{X}$. The *Robust Set Inversion via Interval Analysis* (RSIVIA) algorithm [14] computes an outer subpaving of the q-relaxed solution. It is computed the same way as in Algorithm 1, but with a contractor for the q-relaxed intersection as input. The value of q can be estimated at each measurement epoch, using the *Guaranteed Outlier Minimal Number Estimator* (GOMNE) [15, 6]. This algorithm increases the number of relaxed constraints q until a non-empty solution set is found with RSIVIA. Thus, it returns the minimal number of constraints to relax. This section has presented the basic concepts needed to develop our robust interval-based positioning method. The results of Algorithm 1 are presented in Section 5.

4 Interval-Based Reliable State Estimation

In this section, we present our interval-based method to solve the state estimation problem, in the presence of outliers. Only a bounded-error model is assumed. No other assumptions about the distribution and independence of measurement errors are made, contrary to probabilistic methods.

The problem is defined as a CSP, and the constraints are presented in the following subsections. The variables of the problem, and domains we seek to reduce, are the position of the tags in $\mathcal{F}_w$, denoted $\mathbf{x}_{T_i} \in [\mathbf{x}_{T_i}]$, and the wheelchair configuration $\mathbf{p} \in [\mathbf{p}]$. The larger the domains, the higher the uncertainties. Thus, we seek to contract them by removing all the configurations that are not consistent with the constraints.

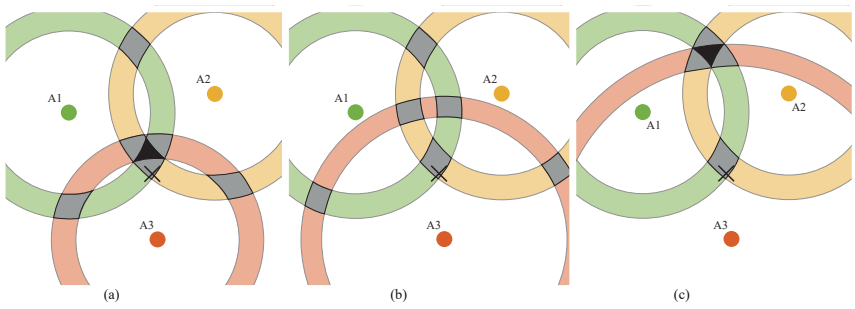


Figure 5: Set-membership localization with three anchors, with one unidentifiable outlier. The solution set is represented in black. The black cross is the actual tag position. The 1-relaxed solution set is represented in gray.

4.1 Constraints over a single epoch

At a given time k or *epoch*, the wheelchair is in a configuration $\mathbf{p}_k = (x_k, y_k, \theta_k)$. Assuming known bounds on range measurement errors, the UWB ranges are represented by a box of interval distances $[\mathbf{r}]_k = ([r_{1,1}]_k, [r_{1,2}]_k, \ldots, [r_{n_A,n_T}]_k)^{\mathbf{T}}$. Considering all these ranges, and the geometry of the problem, a set of constraints can be derived at each epoch k (epoch index k is omitted in the notation). These constraints are presented in this subsection.

UWB constraints in LOS conditions

$\mathcal{L}_{r_{i,j}}$ expresses the distance constraint between the tag T_i and the anchor A_j. It is defined by

$$\mathcal{L}_{r_{i,j}} : \{r_{i,j} = \|\mathbf{x}_{T_i} - \mathbf{x}_{A_j}\|_2\},\ r_{i,j} \in [r_{i,j}],\ i \in \{1 \ldots n_T\},\ j \in \{1 \ldots n_A\}. \tag{3}$$

These constraints are written considering LOS signal propagation for $[r_{i,j}]$ measurement error bounds.

Robustness scheme for the multipath case

The multipath case is handled by the constraint relaxation technique with GOMNE. In the case of an identifiable outlier (i.e., inconsistent with all other measurements), the constraint relaxation strategy can be seen as a measurement rejection scheme. Indeed, in such a case, the non-relaxed solution is empty, and 1-relaxed solution corresponds to the set of all tag positions that are consistent with all measurements except the identifiable outlier.

The case of an unidentifiable outlier is illustrated in Figure 5. In such conditions, the solution set resulting from all constraints is either empty (b), or does not contain the true solution ((a) and (c)), whereas it is the case for the relaxed solution set. The GOMNE algorithm is used to compute the minimal number $q_{\min}$ of constraints

to relax in order not to compute an empty solution set. Thus, a value of $q_{\min} = 0$ will be returned for conditions (a) and (c), and the true solution will not belong to the corresponding solution set. To prevent such a situation, the value computed by GOMNE is then incremented by one to give an additional protection layer against an undetected outlier, such that $q = q_{\min} + 1$.

Wheelchair geometry constraints

The positions of the tags and the wheelchair configuration are linked by geometrical constraints

$$\mathcal{L}_{\mathbf{p},\mathbf{x}_{T_i}} : \left\{ \begin{array}{l} x_{T_i} = x + {}^r x_{T_i} \cos\theta - {}^r y_{T_i} \sin\theta \\ y_{T_i} = y + {}^r x_{T_i} \sin\theta + {}^r y_{T_i} \cos\theta \end{array} \right\}, i \in \{1 \dots n_T\}. \tag{4}$$

The two above constraints are fundamental, since they enable propagating constraints from UWB range measurements to the wheelchair configuration. However, redundant constraints can be added to better contract the variables' domains, thus reducing the uncertainties. Coordinates of the tags, expressed in $\mathcal{F}_r$, are known. The inter-tag distances d_{T_i,T_j} can also be computed. Let ${}^r x_{T_{i,j}} = {}^r x_{T_i} - {}^r x_{T_j}$ and ${}^r y_{T_{i,j}} = {}^r y_{T_i} - {}^r y_{T_j}$ be the difference between the coordinates of i^{th} and j^{th} tag expressed in $\mathcal{F}_r$. This leads to these redundant constraints:

$$\mathcal{L}_{\mathbf{x}_{T_i},\mathbf{x}_{T_j}} : \left\{ \begin{array}{ll} x_{T_i} & = x_{T_j} + {}^r x_{T_{i,j}} . \cos\theta - {}^r y_{T_{i,j}} . \sin\theta \\ y_{T_i} & = y_{T_j} + {}^r x_{T_{i,j}} . \sin\theta + {}^r y_{T_{i,j}} . \cos\theta \\ d_{T_i,T_j} & = \|\mathbf{x}_{T_i} - \mathbf{x}_{T_j}\|_2 \end{array} \right\}, \\ (i,j) \in \{1 \dots n_T\}^2, i < j. \tag{5}$$

Single epoch pose-domain computation

All the constraints related to UWB sensors for a given epoch define a *constraint graph*, presented in Figure 6. We define $\mathcal{L}^k_{\text{UWB}}$ as the composition of (3), (4), and (5) at a given epoch. A contractor is created from $\mathcal{L}^k_{\text{UWB}}$, using the *forward-backward* algorithm [14]. We then set an initial domain for the wheelchair configuration. Finally, we apply GOMNE and the RSIVIA algorithm with the defined contractor, to compute an outer subpaving of all feasible wheelchair poses that are consistent with all measurements except q of them. Notice that, for each epoch k, the computation of the subpaving w.r.t UWB constraints graph is independent from previous epochs, and can be seen as an initial localization problem.

4.2 Pose Estimation from Input Commands

As explained in section 4.1, the wheelchair-configuration domain at epoch k is computed using Algorithm 1 with q-relaxed contractor $\mathcal{L}_{\text{UWB}}$, independently from the previous epoch. To better reduce the estimated pose domain, we use proprioceptive

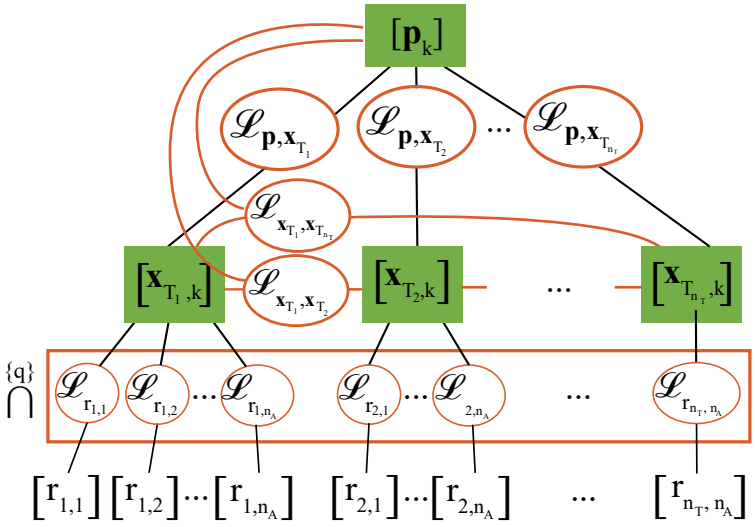


Figure 6: $\mathcal{L}^k_{\text{UWB}}$: UWB-related constraints over a single epoch k . The tag positions are contracted with a robust q-relaxed constraint propagation of the range measurements. Geometrical constraints then contract the wheelchair configuration domains.

information, in the form of linear and angular velocities (v_k, ω_k) directly read from the wheelchair power module.

Discretizing model of (1) using Euler's method, with a sampling period T, the evolution constraints are defined by

$$\mathcal{L}_{\mathbf{p}_{k-1},\mathbf{p}_k} : \left\{ \begin{array}{l} x_{k+1} = x_k + T \cdot v_k \cdot \cos\theta_k \\ y_{k+1} = y_k + T \cdot v_k \cdot \sin\theta_k \\ \theta_{k+1} = \theta_k + T \cdot \omega_k \end{array} \right\}, v_k \in [v_k],\ \omega_k \in [\omega_k]. \tag{6}$$

Integration errors are insignificant w.r.t. uncertainties on the input commands that are handled by our bounded-error model. Evolution constraints are forward-backward propagated in a window containing a horizon of N previous epochs. If $[[\mathbf{p}_{k-N}], ..., [\mathbf{p}_{k-1}], [\mathbf{p}_k]]$ are the pose domains in the horizon at epoch k, then the next epoch horizon is $[[\mathbf{p}_{k-N+1}], ..., [\mathbf{p}_k], [\mathbf{p}_{k+1}]]$. The advantage of such a method is that the evolution constraint propagation related to a wrong configuration domain will only affect domains that share the same horizons. Therefore, N has to be chosen to provide better domain contraction, while not propagating erroneous data over a long period. Evolution constraints graph over one horizon is shown in Figure 7.

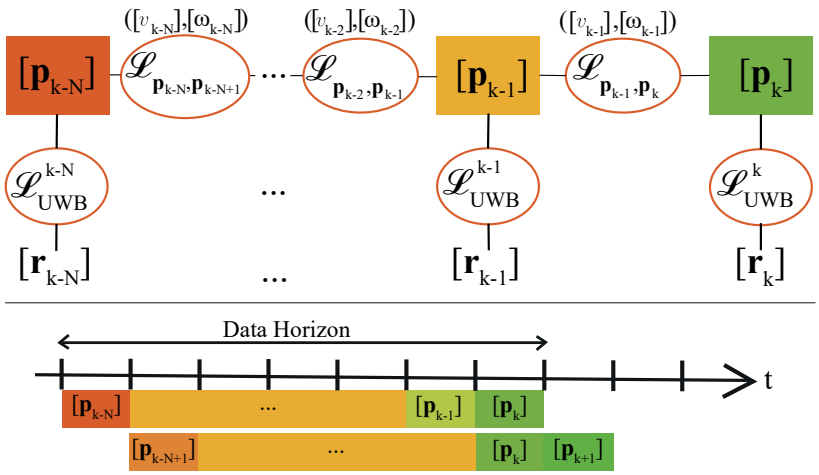


Figure 7: Sliding horizon overview. Robot configuration is computed at each epoch w.r.t. UWB-related constraints. Evolution constraints are then propagated to all variables sharing the same horizon.

5 Experimental Results

5.1 Experimental Setup

To validate our positioning process, we conducted two 4-minute trials using a power wheelchair from Medical Sunrise, which features six wheels: two powered and four caster wheels. Its kinematic is described by the unicycle model of (1). The UWB network was configured by placing 4 UWB anchors in a room and attaching 2 front and 2 rear UWB tags to the wheelchair, as illustrated in Figure 8. UWB range measurements were collected at a frequency of 10 Hz, thus resulting in a total of 4,800 data epochs. Additionally, 8 Qualisys motion capture cameras provide ground truth data for both the wheelchair pose and the positions of the anchors and tags.

During these trials, a user was instructed to navigate the wheelchair using a joystick along a predefined path (see reference trajectory in Figure 12). To mimic real-life activities, the user was tasked to move an object from one table to another located at the opposite end of the room. In the first trial, the user was alone in the wheelchair. In the second trial, a pedestrian walked alongside the wheelchair, occasionally obstructing the line-of-sight (LOS) between some tags and anchors (Figure 9).

We computed the theoretical distances between the anchors and tags and compared them to the measured distances. Figure 10 presents a histogram of the UWB range measurement errors across all trials. The data shows that 85% of the range errors fit a Gaussian distribution with $\sigma = 0.08$ m, corresponding to LOS conditions, while the remaining 15% fit a mixture of Rayleigh distributions, indicative of NLOS conditions.

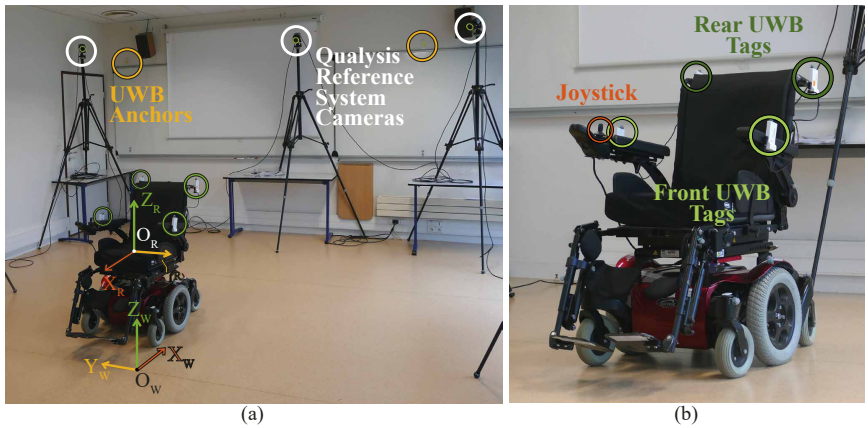


Figure 8: Experimental setup. World frame $\mathcal{F}_w$ and robot frame $\mathcal{F}_r$ are shown in (a).

Figure 9: Experiments without (a) and with (b) a pedestrian. The user had to move the objects from one table to another.

5.2 Pose Domain Computation

The interval-based localization method is tested over both trials. Considering the standard deviation σ estimated in Section 5.1, the UWB range measurement error bounds are set to ± 2.5-σ, i.e., ± 20 cm. With these bounds, 12.82% of the measurements are considered as outliers. UWB anchors' positions are known. Input command error bounds are set to $\pm 0.05 m.s^{-1}$ for linear velocity, and $\pm 0.1 rad.s^{-1}$ for angular velocity. The horizon length is set to $N = 10$, corresponding to 1 s of data. Indeed, a greater N does not provide better domain contraction, thus only increases the risk of propagating an erroneous estimation over a longer time. We assume no prior knowledge of the wheelchair configuration; thus for each epoch, the initial pose domain is set to $[\mathbf{x_0}] = ([-\infty, \infty], [-\infty, \infty], [0, 2\pi])$.

As explained in Section 4.1, all UWB-related constraints, over a single epoch, are

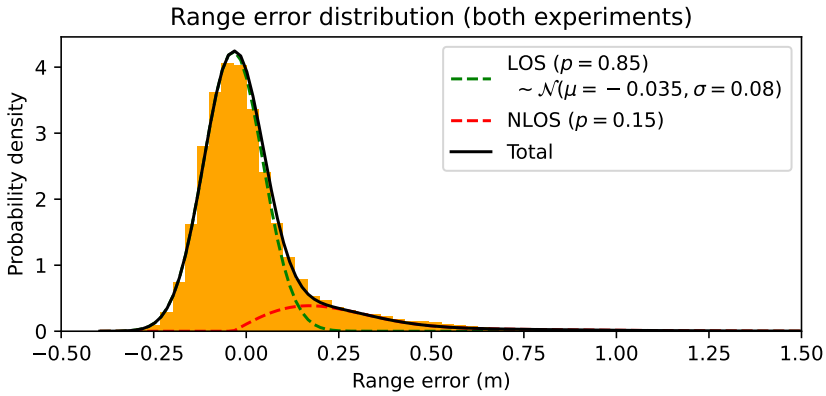


Figure 10: Histogram of the UWB range measurement errors over all trials. 85% of the error distribution is approximated by a Gaussian (green dotted line) in LOS conditions, and 15% by a Rayleigh (red dotted line) in NLOS conditions. The sum of the two distributions fits our range error distribution.

used to create a contractor for the q-relaxed intersection. The q-relaxed contractor is given as input to the RSIVIA algorithm (Algorithm 1), to characterize the set of all feasible poses consistent with the measurements.

At each epoch, 100 ms in total are available to first determine the q value with GOMNE and then compute the pose domain with RSIVIA. A 10 ms budget is allotted to each GOMNE iteration, and all remaining time is then used by RSIVIA to refine the pose domain estimation.

Figure 11 shows outer subpavings (orange boxes) resulting from the RSIVIA algorithm, at epochs corresponding to the minimum (a) and maximum (b) horizontal position error, as well as a disconnected solution set (c). One can observe that these subpavings enclose the ground truth pose (white dots).

Furthermore, the evolution constraints over the horizon enable to significantly reduce our uncertainty domains, as shown by the green boxes in Figure 11. Pose domain contraction using the horizon requires only 3 ms of computation. Thus, the frequency of pose estimation can be increased to provide high-rate predicted poses between two measurement epochs.

At each epoch, as shown in Figure 11, the center of the box $[\mathbf{p}_k]$, denoted $\hat{\mathbf{p}}_k$, is used as an estimation of the unknown wheelchair configuration $\mathbf{p}_k^\star$. Figure 12 shows the estimated trajectory and the ground truth position for the trial with an accompanying pedestrian, with UWB measurements from all four tags.

Our method was tested, for both trials, using data from either all four tags, only the two front tags, or only the two rear tags. Figure 13 shows the cumulative distribution function (CDF) of the horizontal position error (HPE), for each above condition. Slightly higher errors are observed with the presence of a pedestrian

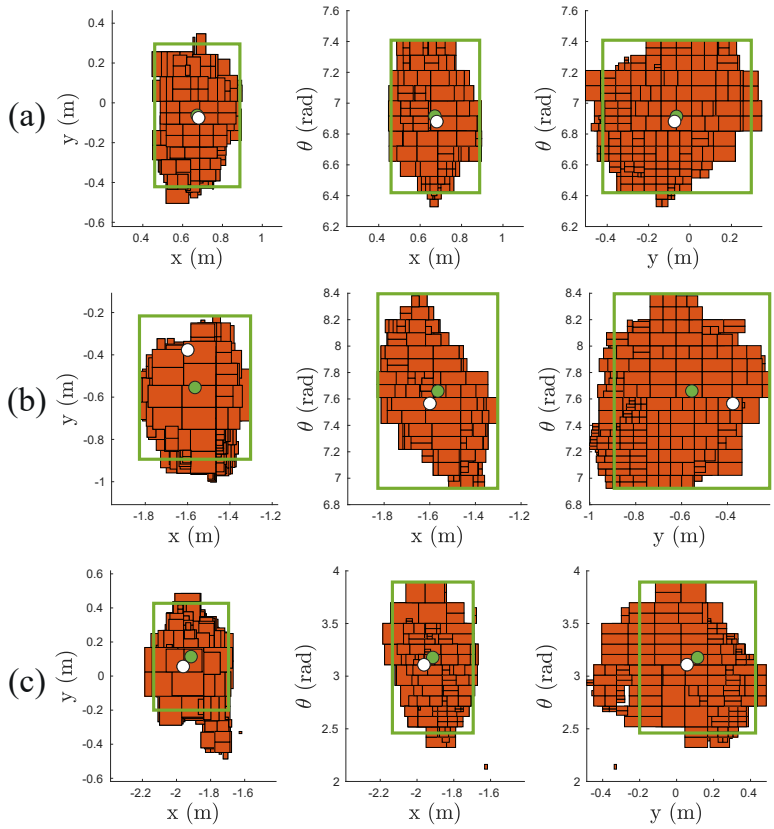


Figure 11: RSIVIA outer subpavings (orange boxes) at $t = 149.4$ s (a), $t = 133.6$ s (b), and $t = 7$ s (c), for the trial with a pedestrian, with four tags. Projections on the x,y plane (*left*), x,θ plane (*middle*) and y, θ plane (*right*). The green boxes represent the box resulting from the contraction of the hull of the orange subpavings, with respect to the evolution constraints. Pose estimate is represented with green dots. Ground truth is represented with white dots.

when using only the two rear tags. This is because the pedestrian was walking next to these particular tags during the experiment. However, when using front or all tags, the CDF are very similar. Thus, the presence of a pedestrian next to the wheelchair has no significant impact on the pose estimation accuracy.

The four-tag solution, with a mean HPE of 6.65 cm (Table 1), is unsurprisingly more accurate than the two-tag solutions. However, the experiments clearly show that using only the two front tags yields better position accuracy (11.44 cm) than the two rear tags (19.46 cm). This is because the position estimate from the rear tags is highly dependent on the orientation estimate, while it is not the case with the two front tags, almost aligned with the robot's y-axis.

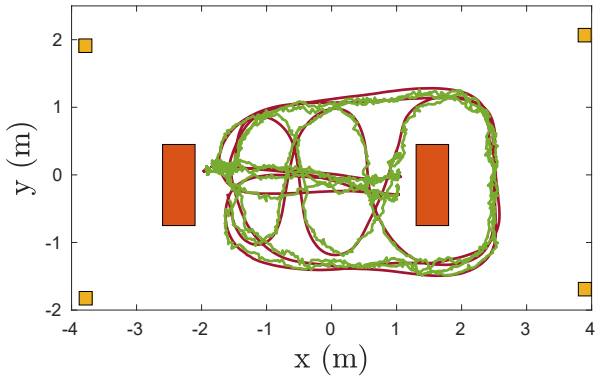


Figure 12: Reference trajectory (red) measured by Qualisys motion capture system, and estimated trajectory (green) with the interval-based localization method, for the trial with a pedestrian, four tags. Tables (orange) and UWB anchors (yellow) are displayed.

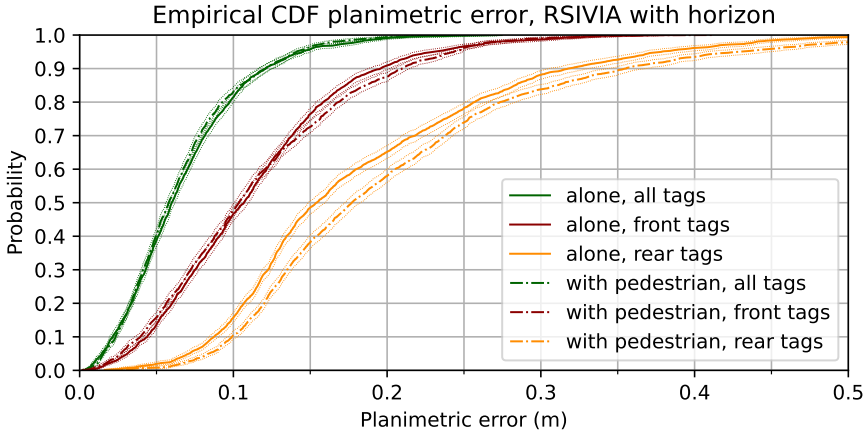


Figure 13: Cumulative Distribution Function of horizontal position error over both trials, with various tag configurations.

To ensure the safety of a task to be performed, each computed pose domain should enclose the true solution. Figure 14 shows the lower and upper bounds of each pose domain, w.r.t. the ground truth, which is at 0. Integrity is obtained if each domain's lower (resp. upper) bound is under (resp. above) the reference line. It can be seen that our method provides confidence domains which are consistent with ground truth, and this is true over the whole dataset. Uncertainties along the x-axis are lower than along the y-axis, due to the geometry of the UWB anchors in the room.

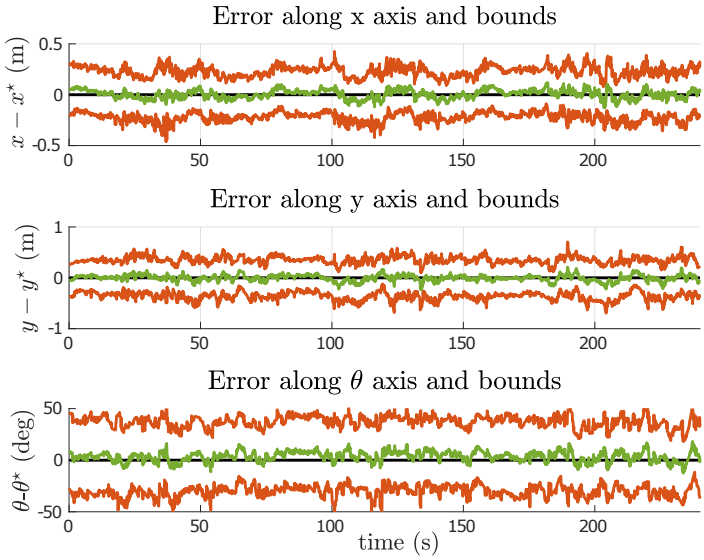


Figure 14: Set-membership position estimation error (green), and confidence lower and upper bounds (orange), for the trial with a pedestrian, with four tags. The reference is at zero (black).

5.3 Comparison with M-Estimator Method

To demonstrate the accuracy of our solution, a comparison with two M-estimators has been carried out, using the same dataset with a horizon of 10 measurement epochs. The first implements the classical Huber loss [12] with 2-σ threshold, and the second a half-Cauchy distribution based loss function, well suited for UWB localization with NLOS [16]. Statistics on the horizontal position error (HPE) and absolute yaw error are given in Table 1. It can be seen that the half-Cauchy M-estimator method is more precise than the classical Huber loss, as shown in [16].

It can be seen that, except when only the two rear tags are used, our interval-based method presents HPE accuracy results close to the half-Cauchy M-estimator method (mean HPE error is 1.76 cm worse). The yaw estimation accuracy is also close (mean absolute yaw error is 0.86° worse) when using all tags, but significantly decreases for both methods when using only two tags. Our method is primarily designed to compute confidence domain in the worst-case scenario, thus is not focused on pose estimate accuracy as it is the case for the M-estimator methods. This explains the better HPE accuracy results for the latter.

We also compared our uncertainty domains with those of the M-estimator method. Results are shown in Table 2. The half-Cauchy M-estimator 3σ bounds on position and angular estimates are significantly smaller than the radius of the computed interval domains (e.g., a mean 3σ radius over x estimate of 4 cm, instead

Table 1: Horizontal position error (HPE) and absolute yaw error $|\tilde{\theta}|$ statistics over both trials, with a horizon of 10 epochs

Method	RSIVIA			M-estimator (Huber)			M-est. (half-Cauchy)		
Tag group	rear	front	all	rear	front	all	rear	front	all.
Mean HPE (cm)	19.46	11.44	**6.65**	11.43	9.03	**5.10**	10.77	8.07	**4.89**
Med. HPE (cm)	16.55	10.50.	**5.90**	10.11	8.67	**4.36**	9.85	7.76	**4.26**
95% HPE (cm)	40.42	23.85	**14.35**	23.88	16.20	**11.86**	21.79	14.65	**11.07**
Max HPE (cm)	70.89	41.17	**27.45**	65.66	28.52	**20.62**	60.64	30.96	**21.14**
Mean $\|\tilde{\theta}\|$ (deg)	13.27	20.12	**5.12**	9.07	8.49	**4.49**	8.77	7.47	**4.26**
Med. $\|\tilde{\theta}\|$ (deg)	10.72	13.92	**4.66**	7.79	7.22	**4.08**	7.64	6.19	**3.84**
95% $\|\tilde{\theta}\|$ (deg)	39.04	58.10	**11.58**	20.79	20.14	**9.93**	19.55	18.32	**9.46**
Max $\|\tilde{\theta}\|$ (deg)	83.18	76.71	**19.56**	45.78	49.18	**22.13**	43.85	57.83	**21.22**

Table 2: Estimation Uncertainty (four tags, both trials) with a horizon of 10: RSIVIA interval domain radius, half-Cauchy (HC) M-estimator with 3σ bounds

Method	RSIVIA: radius			HC M-est.: 3σ			*Consistent HC M-est.:* 3σ		
Domain	x (cm)	y (cm)	θ (deg)	x (cm)	y (cm)	θ (deg)	x (cm)	y (cm)	θ (deg)
min	12.20	21.26	21.48	2.37	3.82	4.17	9.31	15.01	16.37
mean	22.46	35.69	33.43	3.02	4.97	4.95	11.86	19.55	19.46
95%	27.66	43.67	38.49	3.99	5.87	5.27	15.70	23.07	20.70
max	44.05	58.50	51.65	8.13	13.81	16.06	31.95	54.27	63.13

of 22.5 cm with intervals). However, our method provides consistent confidence domains over the whole tested dataset, which is not the case for the half-Cauchy M-estimator method, where the 99% confidence ellipsoid contains the ground truth pose in only 41.7% of the epochs. The latter assumes that measurement errors are independent between two epochs, which is not verified since multipath effects are spatially and therefore temporally correlated, and UWB nodes also have small ranging biases. The 99% confidence ellipsoid needs to be inflated 3.9 times to be consistent with ground truth (i.e. considering a increased measurement error standard deviation of 33 cm). The corresponding uncertainty domains are shown in the last three columns of Table 2. They remain smaller than those provided by our method, but in the same order of magnitude (e.g., a mean 3σ radius over θ estimate of 19.5°, instead of 33.4° with intervals). However, our interval-based method does not need over-tuning to become consistent. It only assumes a bounded-error model, without any independence assumption. Thus, our confidence domains are bigger than those obtained from M-estimator, but no overconfidence is given to the

estimated pose when using realistic measurement error bounds.

The conservative behavior of our method is mostly related to constraint relaxation. Indeed, using the SIVIA algorithm without q-relaxed intersection instead of RSIVIA in our methods results in smaller confidence domains (Figure 15). However, as shown by the blank spaces in Figure 15, the SIVIA method returns an empty solution-set over 86.7% of the dataset, where pose estimation is thus unavailable. It is remarkable that the unavailability of the SIVIA method occurs for large periods of time. This was predictable since multipath are temporally correlated. Our method was designed to provide a high integrity level while ensuring the availability of the localization process. It is then a step toward robust pose estimation for the specific case of UWB for smart wheelchairs.

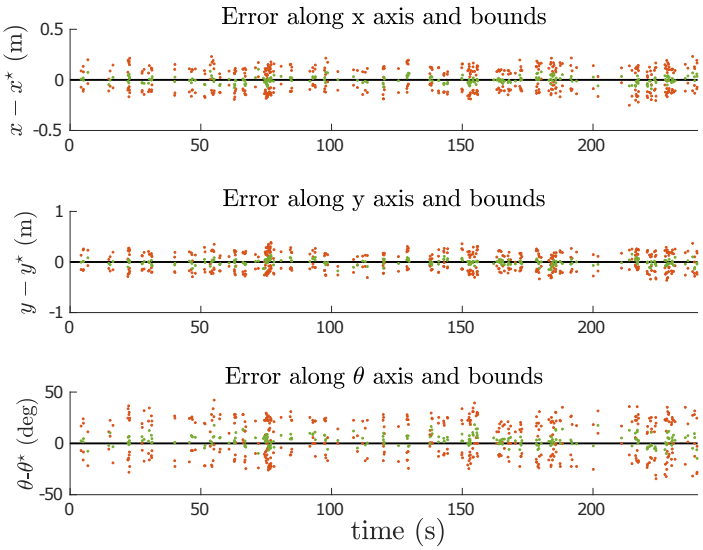


Figure 15: Results of single epoch SIVIA method without constraint relaxation, for the trial with a pedestrian, with four tags. Estimation error in green and bounds in orange (as in Figure 14). The blank spaces correspond to empty solution sets.

6 Conclusion

In this paper, an interval-based localization method for a smart wheelchair has been presented. It provides a real-time robot pose estimate, as well as reliable confidence domains, from bounded-error UWB range measurements and proprioceptive information.

The pose estimation is performed using robust q-relaxed constraint propagation over a horizon of data. The set of the feasible robot poses is computed at each

epoch, using UWB-related constraints. These pose domains are then refined by propagating evolution constraints over the data horizon.

Experiments have been conducted on the smart wheelchair equipped with UWB sensors. Results exhibit consistency of the pose estimation w.r.t. ground truth, despite the large presence of outliers. Close results in terms of accuracy were highlighted when comparing interval-based and M-estimator methods. However, without measurement noise variance inflation, the M-estimator yields inconsistent uncertainty ellipsoids, while our method provides confidence domains that enclose the ground truth 100% of the time over the tested dataset.

The method has been designed to provide worst-case confidence domains, hence its conservative behavior. In order to be exploitable in a fully autonomous navigation and control loop, a trade-off between integrity and confidence domain size could be performed, given a tolerated risk level. Smaller confidence domains could also be computed by developing more efficient contraction methods, and by allocating more time to the GOMNE and RSIVIA algorithms. Also, it seems acceptable to integrate a gyroscope to the smart wheelchair to directly measure the yaw rate with smaller uncertainties, and improve horizon contraction efficiency.

References

[1] Bolting, J. and Fergani, S. Ellipsoidal set membership filtering with reduced conservatism applied to relative localization between aircraft based on GNSS and UWB ranging. In *Proceedings of the 2019 18th European Control Conference*, pages 1810–1815. IEEE, 2019. DOI: 10.23919/ECC.2019.8796256.

[2] Brunacci, V., De Angelis, A., Costante, G., and Carbone, P. Development and analysis of a UWB relative localization system. *IEEE Transactions on Instrumentation and Measurement*, 72:1–13, 2023. DOI: 10.1109/TIM.2023.3305661.

[3] Cazzorla, A., De Angelis, G., Moschitta, A., Dionigi, M., Alimenti, F., and Carbone, P. A 5.6-GHz UWB position measurement system. *IEEE Transactions on Instrumentation and Measurement*, 62(3):675–683, 2013. DOI: 10.1109/TIM.2012.2219139.

[4] Chabert, G. and Jaulin, L. Contractor programming. *Artificial Intelligence*, 173(11):1079–1100, 2009. DOI: 10.1016/j.artint.2009.03.002.

[5] Dardari, D., Conti, A., Ferner, U., Giorgetti, A., and Win, M. Z. Ranging with ultrawide bandwidth signals in multipath environments. *Proceedings of the IEEE*, 97(2):404–426, 2009. DOI: 10.1109/JPROC.2008.2008846.

[6] Desrochers, B., Lacroix, S., and Jaulin, L. Set-membership approach to the kidnapped robot problem. In *Proceedings of the 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3715–3720. IEEE, 2015. DOI: 10.1109/IROS.2015.7353897.

[7] Devigne, L., Narayanan, V. K., Pasteau, F., and Babel, M. Low complex sensor-based shared control for power wheelchair navigation. In *Proceedings of the 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5434–5439. IEEE, 2016. DOI: 10.1109/IROS.2016.7759799.

[8] Drevelle, V. and Bonnifait, P. Localization confidence domains via set inversion on short-term trajectory. *IEEE Transactions on Robotics*, 29(5):1244–1256, 2013. DOI: 10.1109/TRO.2013.2262776.

[9] Finlayson, M. and Van Denend, T. Experiencing the loss of mobility: Perspectives of older adults with MS. *Disability and rehabilitation*, 25(20):1168–1180, 2003. DOI: 10.1080/0963828031000159618O.

[10] Gentner, C., Jost, T., Wang, W., Zhang, S., Dammann, A., and Fiebig, U.-C. Multipath assisted positioning with simultaneous localization and mapping. *IEEE Transactions on Wireless Communications*, 15(9):6104–6117, 2016. DOI: 10.1109/TWC.2016.2578336.

[11] González, J., Blanco, J., Galindo, C., Ortiz-de Galisteo, A., Fernández-Madrigal, J., Moreno, F., and Martínez, J. Mobile robot localization based on Ultra-Wide-Band ranging: A particle filter approach. *Robotics and Autonomous Systems*, 57(5):496–507, 2009. DOI: 10.1016/j.robot.2008.10.022.

[12] Huber, P. J. Robust estimation of a location parameter. *The Annals of Mathematical Statistics*, 35(1):73–101, 1964. DOI: 10.1007/978-1-4612-4380-9_35.

[13] Jaulin, L. Robust set-membership state estimation; application to underwater robotics. *Automatica*, 45(1):202–206, 2009. DOI: 10.1016/j.automatica.2008.06.013.

[14] Jaulin, L., Kieffer, M., Didrit, O., and Walter, É. *Interval Analysis*. Springer, 2001. DOI: 10.1007/978-1-4471-0249-6_2.

[15] Jaulin, L., Kieffer, M., Walter, E., and Meizel, D. Guaranteed robust nonlinear estimation with application to robot localization. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 32(4):374–381, 2002. DOI: 10.1109/TSMCC.2002.806747.

[16] Jiang, F., Caruso, D., Dhekne, A., Qu, Q., Engel, J. J., and Dong, J. Robust indoor localization with ranging-imu fusion. In *Proceedings of the 2024 IEEE International Conference on Robotics and Automation*, pages 11963–11969. IEEE, 2024. DOI: 10.1109/ICRA57147.2024.10611274.

[17] Jiang, H., Wang, W., Shen, Y., Li, X., Ren, X., Mu, B., and Wu, J. Efficient planar pose estimation via UWB measurements. In *Proceedings of the*

2023 IEEE International Conference on Robotics and Automation, pages 1954–1960, London, United Kingdom, 2023. IEEE. DOI: 10.1109/ICRA48891.2023.10161456.

[18] Kenmogne, I.-F., Drevelle, V., and Marchand, E. Interval-based cooperative uavs pose domain characterization from images and ranges. In *Proceedings of the 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 6349–6356. IEEE, 2018. DOI: https://doi.org/10.1109/IROS.2018.8593742.

[19] Kieffer, M., Jaulin, L., Walter, E., and Meizel, D. Robust autonomous robot localization using interval analysis. *Reliable Computing*, 6(3):337–362, 2000. DOI: 10.1023/A:1009990700281.

[20] Leaman, J. and La, H. M. A comprehensive review of smart wheelchairs: past, present, and future. *IEEE Transactions on Human-Machine Systems*, 47(4):486–499, 2017. DOI: 10.1109/THMS.2017.2706727.

[21] Ledergerber, A., Hamer, M., and D'Andrea, R. A robot self-localization system using one-way ultra-wideband communication. In *Proceedings of the 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3131–3137, Hamburg, Germany, 2015. IEEE. DOI: 10.1109/IROS.2015.7353810.

[22] Magnago, V., Corbalan, P., Picco, G. P., Palopoli, L., and Fontanelli, D. Robot localization via odometry-assisted ultra-wideband ranging with stochastic guarantees. In *Proceedings of the 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1607–1613, Macau, China, 2019. IEEE. DOI: 10.1109/IROS40897.2019.8968019.

[23] Merlinge, N. Set inversion and box contraction on Lie groups using interval analysis. *Automatica*, 165:111688, 2024. DOI: 10.1016/j.automatica.2024.111688.

[24] Morales, Y., Kallakuri, N., Shinozawa, K., Miyashita, T., and Hagita, N. Human-comfortable navigation for an autonomous robotic wheelchair. In *Proceedings of the 2013 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pages 2737–2743. IEEE, 2013. DOI: 10.1109/IROS.2013.6696743.

[25] Morbidi, F. et al. Assistive robotic technologies for next-generation smart wheelchairs: Codesign and modularity to improve users' quality of life. *IEEE Robotics & Automation Magazine*, 30(1):24–35, 2023. DOI: 10.1109/MRA.2022.3178965.

[26] Mustafa, M., Stancu, A., Guteirrez, S. P., Codres, E. A., and Jaulin, L. Rigid transformation using interval analysis for robot motion estimation. In *Proceedings of the 2015 20th International Conference on Control Systems and Computer Science*, pages 24–31. IEEE, 2015. DOI: 10.1109/CSCS.2015.98.

[27] Pasteau, F., Narayanan, V. K., Babel, M., and Chaumette, F. A visual servoing approach for autonomous corridor following and doorway passing in a wheelchair. *Robotics and Autonomous Systems*, 75:28–40, 2016. DOI: 10.1016/j.robot.2014.10.017.

[28] Rauh, A., Gourret, Y., Lagattu, K., Hummes, B., Jaulin, L., Reuter, J., Wirtensohn, S., and Hoher, P. Experimental validation of ellipsoidal techniques for state estimation in marine applications. *Algorithms*, 15(5):162, 2022. DOI: 10.3390/a15050162.

[29] Rohou, S., Jaulin, L., Mihaylova, L., Le Bars, F., and Veres, S. M. *Reliable Robot Localization: A Constraint-Programming Approach Over Dynamical Systems*. Wiley, 1 edition, 2019. DOI: 10.1002/9781119680970.

[30] Wang, Z. and Lambert, A. A low-cost consistent vehicle localization based on interval constraint propagation. *Journal of Advanced Transportation*, 2018(1):2713729, 2018. DOI: 10.1155/2018/2713729.

Acta Cybernetica **27** (2026) 359–382.

Approaches Towards A Systematic Tuning of Interval Observers via Norm-Bounded Error Dynamics and LMIs

Marit Lahme[ab] and Andreas Rauh[ac]

Abstract

Interval observers typically rely on performance criteria that specify the estimation accuracy. Inappropriately parameterized performance criteria can lead to linear matrix inequalities (LMIs) that do not yield feasible solutions. Retuning the output specification of the error dynamics allows for exploiting structural feasibility. The overall goal of this paper is to design a TNL interval observer for systems for which the LMIs do not yield a feasible solution with the standard specification of the performance criteria. As an example, the proposed tuning methods are applied to the state estimation of a lithium-ion battery cell. The LMIs introduced together with this observer structure initially do not have a feasible solution for the investigated battery model. To obtain feasible solutions, we propose extended design conditions for the TNL interval observer based on a virtual output equation for the error dynamics that is utilized to reduce the influence of uncertainties on the estimation results. Additionally, we present two techniques to enforce interpretable structures in the observer gains to enhance the estimation accuracy. The first technique enforces a desired ratio between the elements of selected observer gains, the second technique influences the estimation accuracy by specifying the eigenvalues of the scaled observer system matrix. To demonstrate the fundamental application of the tuning methods, the proposed techniques are initially applied to the state estimation of a mass-spring-damper system. Subsequently, the effectiveness of the proposed techniques is shown for the state estimation of the lithium-ion battery cell. With both techniques, the estimation accuracy can be significantly enhanced. Furthermore, they provide a framework for systematically designing TNL interval observers.

Keywords: tuning of performance criteria, LMI, TNL interval observer, norm-bounded error dynamics

[a]Distributed Control in Interconnected Systems Group, Department of Computing Science, Carl von Ossietzky Universität Oldenburg, D-26111 Oldenburg, Germany
[b]E-mail: marit.lahme@uni-oldenburg.de, ORCID: 0000-0001-8633-1908
[c]E-mail: andreas.rauh@uni-oldenburg.de, ORCID: 0000-0002-1548-6547

DOI: 10.14232/actacyb.318090

1 Introduction

In practical control applications, reliable state estimation is essential, because oftentimes not all state variables can be measured. A well known technique to estimate the corresponding state variables based on available measurements is the use of interval observers. The advantage of interval observers over classical observers like the Luenberger observer is that during the design process measurement noise, process uncertainties and parametric uncertainties can be taken into account. Interval observers have been a focus of research for more than two decades. They are recurrently designed based on the monotone systems theory [23]. Consequently, in addition to stability conditions, there are positivity conditions that have to be fulfilled, which exacerbates finding suitable observer gains. In the literature, multiple methods are presented that address this problem. When the design conditions can not be fulfilled directly, commonly a coordinate transformation is performed, where the interval observer is designed for the transformed system [13, 28]. Besides a coordinate transformation an alternative observer structure that provides more design degrees of freedom can be employed to relax the design conditions. A specific example of this is the TNL interval observer, which provides two additional design degrees of freedom, in particular the observer gains $\mathbf{T}$ and $\mathbf{N}$ besides the observer gain $\mathbf{L}$ [27]. In this paper, we will focus on the latter method. We aim to design a TNL interval observer for a quasi-linear system model of a lithium-ion battery cell. Designing a TNL interval observer for this model is challenging. The state variables to be estimated represent the state of charge as well as voltages in an equivalent circuit model and cannot be measured directly. Instead, the measurement represents a linear combination of the state variables and other state-dependent equivalent circuit parameters. In addition, the state variables are scaled differently, and the system matrix exhibits a diagonal structure.

The design of the TNL interval observer in [27] is based on a linear matrix inequality (LMI) approach, which is very common in controller and observer design. Numerous papers employ LMI techniques to ensure the stability of the error dynamics and to guarantee that the positivity conditions are satisfied [1, 11, 12]. Moreover, additional performance criteria are frequently incorporated in the set of LMIs to influence the observer dynamics. For instance, pole placement techniques are utilized to guarantee desired performance characteristics such as specific exponential decay rates and damping ratios [7, 19, 22]. In addition, H_∞ and L_∞ optimization criteria are commonly included in the set of LMIs to attenuate the effect of uncertainties on the estimation result [8, 18, 27]. While those LMI techniques are widely employed to design interval observers, it is not always trivial to solve the LMIs, especially if the variables to be estimated are scaled differently or the dynamics are described by stiff differential equations. This similarly applies to the battery model. Applying the LMIs of [27] to the battery model does not yield a feasible solution, because of inappropriately parameterized performance criteria.

In the literature, there exist various methods to alleviate difficulties that are associated with solving LMIs. For example, LMI relaxation techniques are used to simplify complex optimization problems by transforming them into a more tractable

form. In [26], LMI relaxation techniques are used to transform parameterized LMIs, which are known to be NP-hard, into a finite set of LMIs to reduce the computational effort for finding feasible solutions. The authors in [21] review classical relaxation techniques based on the S-procedure and make use of the Lagrange Duality Theory to investigate under which conditions those relaxation techniques are exact. To improve numerical stability, to reduce conservatism or to obtain tractable conditions, where standard LMIs may not lead to tractable ones, slack variable LMIs or dilated LMIs can be utilized [5, 15]. A different method is presented in [3], where an iterative scheme is applied to obtain feasible or approximately feasible solutions to robust LMIs, which are intractable by standard exact LMI methods. Similarly, the authors in [20] propose iterative semidefinite programming methods to solve a nonlinear matrix inequality problem for robust state feedback control. An iterative approach is likewise utilized in [4] to solve bilinear matrix inequalities to simultaneously optimize controller and observer gains.

With regard to the battery model, the above-mentioned methods are either not applicable or not effective, due to the structure of the state-space matrices and the resulting LMIs. The challenge with regard to the battery model is that there exist feasible solutions for the LMIs that cannot be identified by the solver because of inappropriately parameterized performance criteria. Inspired by the aforementioned pole placement techniques and iterative methods, we therefore propose extended design conditions for the TNL interval observer to obtain feasible solutions for the LMIs. Additionally, we present two techniques based on observer gain tuning to systematically enhance the estimation accuracy. Those techniques are based on enforcing interpretable structures in the matrices $\mathbf{T}$, $\mathbf{N}$, and $\mathbf{L}$, that can lead to improved estimation results with regard to interval width. We achieve this by adding additional design degrees of freedom as well as additional constraints to the set of LMIs and solving them by taking into account appropriate cost functions. To the best of our knowledge, there do not exist such methods in the current literature.

This paper is structured as follows. Section 2 summarizes the problem formulation including assumptions and preliminaries as well as the general structure of the TNL interval observer and the design conditions. The main results are presented in Section 3. At first, the derivation of the extended design conditions is shown. Afterward, two techniques to enforce interpretable structures in the observer gains that can lead to improved estimation results are presented. Simulation results are presented in Section 4. This paper is concluded with a brief summary and an outlook on future work in Section 5.

Notation. In this paper, bold letters represent matrices and vectors, where capitalized letters represent matrices and lowercase letters represent vectors. By default, lowercase bold letters (e.g. $\mathbf{m}$) represent column vectors, while row vectors are represented by the transposed form (e.g. $\mathbf{m}^{\mathrm{T}}$). Upper and lower bounds on a matrix or a vector are denoted by over- and underlines (e.g. $\overline{\mathbf{M}}$, $\underline{\mathbf{M}}$) and are meant to be employed elementwise. Likewise, the operators $\geq$, $>$, $\leq$, $<$ are employed elementwise. Positive and negative parts of a matrix $\mathbf{M}$ are denoted by $\mathbf{M}^{+}$ and $\mathbf{M}^{-}$ defined by $\mathbf{M}^{+} = \max\{\mathbf{0}, \mathbf{M}\}$ and $\mathbf{M}^{-} = \mathbf{M}^{+} - \mathbf{M}$. Letters marked with a circumflex represent estimated values, for example $\hat{\mathbf{x}}$ is an estimate for the true value

$\mathbf{x}$. $\mathbf{I}_n$ denotes the $n \times n$ identity matrix and $\mathbb{R}_{\geq 0}$ denotes the non-negative orthant of $\mathbb{R}$. $\| \, \|$ denotes the L_2-norm and $\mathbf{M} \succ 0$ ($\prec 0$) denotes a positive (negative) definite matrix. The elementwise defined Hadamard product and Hadamard division are represented by $\odot$ and $\oslash$ respectively and an asterisk * is used to indicate terms that are induced by symmetry.

2 Problem Formulation

Consider the discrete-time system

$$\begin{aligned} \mathbf{x}_{k+1} &= \mathbf{A}_\mathrm{d}\mathbf{x}_k + \mathbf{B}_\mathrm{d}\mathbf{u}_k + \mathbf{E}_\mathrm{d}\boldsymbol{w}_k \ , \\ \mathbf{y}_k &= \mathbf{C}\mathbf{x}_k + \boldsymbol{v}_k \ , \end{aligned} \tag{1}$$

with the state vector $\mathbf{x}_k \in \mathbb{R}^n$, the input vector $\mathbf{u}_k \in \mathbb{R}^p$, the output vector $\mathbf{y}_k \in \mathbb{R}^m$, the process uncertainty $\boldsymbol{w}_k \in \mathbb{R}^q$ and the measurement noise $\boldsymbol{v}_k \in \mathbb{R}^m$. The matrices $\mathbf{A}_\mathrm{d}$, $\mathbf{B}_\mathrm{d}$, $\mathbf{E}_\mathrm{d}$ and $\mathbf{C}$ are of appropriate dimensions.

Assumption 1. *The matrices $\mathbf{A}_\mathrm{d}$, $\mathbf{B}_\mathrm{d}$, $\mathbf{E}_\mathrm{d}$ and $\mathbf{C}$ are constant.*

Assumption 2. *The initial estimate for the state vector is chosen so that the true value is included and $\hat{\underline{\mathbf{x}}}_0 \leq \mathbf{x}_0 \leq \hat{\overline{\mathbf{x}}}_0$ holds.*

Assumption 3. *The process and measurement uncertainties are unknown but bounded with $\underline{\boldsymbol{w}}_k \leq \boldsymbol{w}_k \leq \overline{\boldsymbol{w}}_k$ and $\underline{\boldsymbol{v}}_k \leq \boldsymbol{v}_k \leq \overline{\boldsymbol{v}}_k$.*

Lemma ([6, 9])**.** *System (1) with $\mathbf{u}_k \in \mathbb{R}^p_{\geq 0}$, $\boldsymbol{w}_k \in \mathbb{R}^q_{\geq 0}$ and $\boldsymbol{v}_k \in \mathbb{R}^m_{\geq 0}$ is positive if and only if $\mathbf{A}_\mathrm{d} \in \mathbb{R}^{n\times n}_{\geq 0}$, $\mathbf{B}_\mathrm{d} \in \mathbb{R}^{n\times p}_{\geq 0}$, $\mathbf{E}_\mathrm{d} \in \mathbb{R}^{n\times q}_{\geq 0}$ and $\mathbf{C} \in \mathbb{R}^{m\times n}_{\geq 0}$, provided that $\mathbf{x}_0 \geq 0$.*

Lemma ([27])**.** *The discrete-time system*

$$\begin{aligned} \boldsymbol{x}_{k+1} &= \boldsymbol{\mathcal{A}}\boldsymbol{x}_k + \boldsymbol{\mathcal{B}}\boldsymbol{d}_k \\ \boldsymbol{z}_k &= \boldsymbol{\mathcal{C}}\boldsymbol{x}_k + \boldsymbol{\mathcal{D}}\boldsymbol{d}_k \end{aligned} \tag{2}$$

is stable and satisfies $\|\boldsymbol{z}\| < \gamma \|\boldsymbol{d}\|$ for a given scalar $\gamma > 0$, if and only if there exists a matrix $\boldsymbol{\mathcal{P}} \succ 0$ such that

$$\begin{bmatrix} \boldsymbol{\mathcal{A}}^T\boldsymbol{\mathcal{P}}\boldsymbol{\mathcal{A}} + \boldsymbol{\mathcal{C}}^T\boldsymbol{\mathcal{C}} - \boldsymbol{\mathcal{P}} & \boldsymbol{\mathcal{A}}^T\boldsymbol{\mathcal{P}}\boldsymbol{\mathcal{B}} + \boldsymbol{\mathcal{C}}^T\boldsymbol{\mathcal{D}} \\ \boldsymbol{\mathcal{B}}^T\boldsymbol{\mathcal{P}}\boldsymbol{\mathcal{A}} + \boldsymbol{\mathcal{D}}^T\boldsymbol{\mathcal{C}} & \boldsymbol{\mathcal{B}}^T\boldsymbol{\mathcal{P}}\boldsymbol{\mathcal{B}} + \boldsymbol{\mathcal{D}}^T\boldsymbol{\mathcal{D}} - \gamma^2\boldsymbol{\mathcal{I}} \end{bmatrix} \prec 0 \ . \tag{3}$$

We aim to design a TNL interval observer for system (1). This observer has the following structure [27]

$$\begin{aligned} \overline{\zeta}_{k+1} &= \mathbf{T}\mathbf{A}_\mathrm{d}\hat{\overline{\mathbf{x}}}_k + \mathbf{T}\mathbf{B}_\mathrm{d}\mathbf{u}_k + \mathbf{L}(\mathbf{y}_k - \mathbf{C}\hat{\overline{\mathbf{x}}}_k) + \overline{\boldsymbol{\Delta}}_k \ , \\ \hat{\overline{\mathbf{x}}}_k &= \overline{\zeta}_k + \mathbf{N}\mathbf{y}_k \ , \\ \underline{\zeta}_{k+1} &= \mathbf{T}\mathbf{A}_\mathrm{d}\hat{\underline{\mathbf{x}}}_k + \mathbf{T}\mathbf{B}_\mathrm{d}\mathbf{u}_k + \mathbf{L}(\mathbf{y}_k - \mathbf{C}\hat{\underline{\mathbf{x}}}_k) + \underline{\boldsymbol{\Delta}}_k \ , \\ \hat{\underline{\mathbf{x}}}_k &= \underline{\zeta}_k + \mathbf{N}\mathbf{y}_k \ , \end{aligned} \tag{4}$$

with $\overline{\zeta}_k \in \mathbb{R}^n$ and $\underline{\zeta}_k \in \mathbb{R}^n$ as intermediate variables and $\overline{\boldsymbol{\Delta}}_k$ and $\underline{\boldsymbol{\Delta}}_k$ given as

$$\begin{aligned}\overline{\boldsymbol{\Delta}}_k =&(\mathbf{T}\mathbf{E}_\mathrm{d})^+ \overline{\boldsymbol{w}}_k - (\mathbf{T}\mathbf{E}_\mathrm{d})^- \underline{\boldsymbol{w}}_k + \mathbf{L}^+ \overline{\boldsymbol{v}}_k - \mathbf{L}^- \underline{\boldsymbol{v}}_k + \mathbf{N}^+ \overline{\boldsymbol{v}}_{k+1} - \mathbf{N}^- \underline{\boldsymbol{v}}_{k+1} \ , \\ \underline{\boldsymbol{\Delta}}_k =&(\mathbf{T}\mathbf{E}_\mathrm{d})^+ \underline{\boldsymbol{w}}_k - (\mathbf{T}\mathbf{E}_\mathrm{d})^- \overline{\boldsymbol{w}}_k + \mathbf{L}^+ \underline{\boldsymbol{v}}_k - \mathbf{L}^- \overline{\boldsymbol{v}}_k + \mathbf{N}^+ \underline{\boldsymbol{v}}_{k+1} - \mathbf{N}^- \overline{\boldsymbol{v}}_{k+1} \ .\end{aligned} \tag{5}$$

Note that the TNL interval observer reduces to a Luenberger structure if $\mathbf{T} = \mathbf{I}_n$ and $\mathbf{N} = \mathbf{0}$. By defining bounds for the state estimation errors as $\overline{\mathbf{e}}_k = \hat{\overline{\mathbf{x}}}_k - \mathbf{x}_k$ and $\underline{\mathbf{e}}_k = \hat{\underline{\mathbf{x}}}_k - \mathbf{x}_k$, the error dynamics result in [27]

$$\begin{aligned}\overline{\mathbf{e}}_{k+1} &= (\mathbf{T}\mathbf{A}_\mathrm{d} - \mathbf{L}\mathbf{C})\overline{\mathbf{e}}_k + \tilde{\mathbf{B}}_\mathrm{d}\overline{\mathbf{d}}_k \ , \\ \underline{\mathbf{e}}_{k+1} &= (\mathbf{T}\mathbf{A}_\mathrm{d} - \mathbf{L}\mathbf{C})\underline{\mathbf{e}}_k + \tilde{\mathbf{B}}_\mathrm{d}\underline{\mathbf{d}}_k \ ,\end{aligned} \tag{6}$$

with $\overline{\mathbf{d}}_k = \left[(\overline{\boldsymbol{\Delta}}_k - \mathbf{T}\mathbf{E}_\mathrm{d}\boldsymbol{w}_k)^T \quad \boldsymbol{v}_k^T \quad \boldsymbol{v}_{k+1}^T\right]^T$, $\underline{\mathbf{d}}_k = \left[(\underline{\boldsymbol{\Delta}}_k - \mathbf{T}\mathbf{E}_\mathrm{d}\boldsymbol{w}_k)^T \quad \boldsymbol{v}_k^T \quad \boldsymbol{v}_{k+1}^T\right]^T$ and $\tilde{\mathbf{B}}_\mathrm{d} = \begin{bmatrix}\mathbf{I}_n & \mathbf{L} & \mathbf{N}\end{bmatrix}$. For the observer design, we take into account the following virtual output equation for the error dynamics

$$\mathbf{z}_{\mathrm{e},k} = \mathbf{C}_\mathrm{e}\mathbf{e}_k + \mathbf{D}_\mathrm{e}\mathbf{d}_k \ , \tag{7}$$

where the matrices $\mathbf{C}_\mathrm{e}$ and $\mathbf{D}_\mathrm{e}$ characterize the influence of individual error terms, as well as process and measurement noise on the virtual output $\mathbf{z}_\mathrm{e}$. We utilize this virtual output equation to impose performance objectives on the transfer function between $\mathbf{d}$ and $\mathbf{z}_\mathrm{e}$ to reduce the influence of uncertainties on the estimation error [16]. In order to precisely estimate the state variables, it is necessary that the error dynamics are stable. Therefore, the observer system matrix $(\mathbf{T}\mathbf{A}_\mathrm{d} - \mathbf{L}\mathbf{C})$ has to be Schur stable. The concept of the interval observer is based on the theory of positive systems [10]. Hence, a positivity condition according to Lemma 2 has to be fulfilled additionally to reliably bound the true value of the states with lower and upper bounds so that $\hat{\underline{\mathbf{x}}}_k \leq \mathbf{x}_k \leq \hat{\overline{\mathbf{x}}}_k$ holds. To reduce the influence of uncertainties on the estimation result, an H_∞ technique is included in the design process, so that $\|\overline{\mathbf{z}}_\mathrm{e}\| < \gamma \left\|\overline{\mathbf{d}}\right\|$ holds for an arbitrary positive scalar γ (analogously for the lower bound). Those design conditions can be formulated as a set of LMIs. Applying the LMIs from [27] to the battery model does not yield a feasible solution, although the LMIs are not infeasible by structure. This implies that a feasible solution exists, that can not be identified by the solver. In the following section, we therefore propose extended design conditions for the TNL interval observer that allow for identifying feasible solutions for these LMIs. Additionally, we present two techniques that provide a framework to systematically enhance the estimation accuracy.

Remark. In [27], the matrix $\mathbf{C}_\mathrm{e}$ is set to the identity matrix $\mathbf{I}_n$ and the matrix $\mathbf{D}_\mathrm{e}$ is set to zero. This results in equal weights for the individual error terms for imposing performance objectives on the transfer function between $\mathbf{z}_\mathrm{e}$ and $\mathbf{d}$. Additionally, the error dynamics fulfill $\|\overline{\mathbf{e}}\| < \gamma \left\|\overline{\mathbf{d}}\right\|$ and $\|\underline{\mathbf{e}}\| < \gamma \|\underline{\mathbf{d}}\|$. With regard to the battery model, the LMIs with this choice of $\mathbf{C}_\mathrm{e}$ and $\mathbf{D}_\mathrm{e}$ do not yield a feasible solution. We utilize the matrices $\mathbf{C}_\mathrm{e}$ and $\mathbf{D}_\mathrm{e}$ as additional degrees of freedom to allow for specifying different convergence rates for the individual error terms and to identify a feasible solution. Thus, the error dynamics fulfill $\|\overline{\mathbf{z}}_\mathrm{e}\| < \gamma \left\|\overline{\mathbf{d}}\right\|$ and $\|\underline{\mathbf{z}}_\mathrm{e}\| < \gamma \|\underline{\mathbf{d}}\|$.

This, however, does not necessarily provide a high estimation accuracy. By choosing $\mathbf{C}_\mathrm{e}$ and $\mathbf{D}_\mathrm{e}$ as additional design degrees of freedom, we obtain an estimation result as shown in Figure 5a for $\alpha = 1$, where α is chosen according to Section 3.2. This estimation result is not suitable for estimating all state variables because of the low estimation accuracy characterized by the large interval width between the estimated lower and upper bounds. We therefore propose two techniques to enhance the design of the observer gains $\mathbf{T}$, $\mathbf{N}$, and $\mathbf{L}$.

3 Main Results

In this section, we at first introduce the design conditions that have to be fulfilled to design a TNL interval observer for system (1) including the matrices $\mathbf{C}_\mathrm{e}$ and $\mathbf{D}_\mathrm{e}$ as additional design degrees of freedom. Afterward, we propose two techniques to enforce interpretable structures in the matrices $\mathbf{T}$, $\mathbf{N}$, and $\mathbf{L}$, that can lead to improved estimation results with regard to the interval width. With the first technique, we enforce a desired ratio between the elements of selected observer gain matrices. The second technique is used to influence the observer dynamics by specifying desired eigenvalues for the scaled observer system matrix.

3.1 Design conditions

Based on (6) and (7), the stability, positivity, and performance conditions are satisfied if the following set of LMIs is fulfilled:

$$\begin{bmatrix} -\mathbf{P} & * & * & * & * & * \\ 0 & -\gamma^2\mathbf{I}_n & * & * & * & * \\ 0 & 0 & -\gamma^2\mathbf{I}_m & * & * & * \\ 0 & 0 & 0 & -\gamma^2\mathbf{I}_m & * & * \\ \mathbf{C}_\mathrm{e} & \mathbf{D}_{\mathrm{e},1} & \mathbf{D}_{\mathrm{e},2} & \mathbf{D}_{\mathrm{e},3} & -\mathbf{I}_n & * \\ \Phi & \mathbf{P} & \mathbf{W} & \mathbf{X} & 0 & -\mathbf{P} \end{bmatrix} \prec 0 \ , \qquad \mathbf{P} \succ 0 \ , \quad \Phi \geq 0 \ , \quad \gamma > 0 \ , \tag{8}$$

with $\Phi = \mathbf{PM} = (\mathbf{P} - \mathbf{XC})\,\mathbf{A}_\mathrm{d} - \mathbf{WC}$, $\mathbf{M} = (\mathbf{TA}_\mathrm{d} - \mathbf{LC})$, $\mathbf{W} = \mathbf{PL}$, $\mathbf{X} = \mathbf{PN}$, $\mathbf{D}_\mathrm{e} = [\mathbf{D}_{\mathrm{e},1}\ \mathbf{D}_{\mathrm{e},2}\ \mathbf{D}_{\mathrm{e},3}]$, $\mathbf{T} = \mathbf{I}_n - \mathbf{NC}$ and an arbitrary diagonal matrix $\mathbf{P} \in \mathbb{R}^n$, where $\mathbf{T}$, $\mathbf{N}$, and $\mathbf{L}$ are the observer gains to be designed. The matrix $\mathbf{C}_\mathrm{e}$ is an arbitrary diagonal matrix and the matrix $\mathbf{D}_\mathrm{e}$ is an arbitrary full matrix.

Proof. Based on Lemma 2, the error dynamics according to (6) and (7) are stable and satisfy $\|\mathbf{z}_\mathrm{e}\| < \gamma \|\mathbf{d}\|$ if

$$\begin{bmatrix} \mathbf{M}^T\mathbf{PM} + \mathbf{C}_\mathrm{e}^T\mathbf{C}_\mathrm{e} - \mathbf{P} & \mathbf{M}^T\mathbf{P}\tilde{\mathbf{B}}_\mathrm{d} + \mathbf{C}_\mathrm{e}^T\mathbf{D}_\mathrm{e} \\ \tilde{\mathbf{B}}_\mathrm{d}^T\mathbf{PM} + \mathbf{D}_\mathrm{e}^T\mathbf{C}_\mathrm{e} & \tilde{\mathbf{B}}_\mathrm{d}^T\mathbf{P}\tilde{\mathbf{B}}_\mathrm{d} + \mathbf{D}_\mathrm{e}^T\mathbf{D}_\mathrm{e} - \gamma^2\mathbf{I}_{n+2m} \end{bmatrix} \prec 0 \ , \tag{9}$$

with $\mathbf{M} = (\mathbf{T}\mathbf{A}_\mathrm{d} - \mathbf{L}\mathbf{C})$, $\tilde{\mathbf{B}}_\mathrm{d} = \begin{bmatrix}\mathbf{I}_n & \mathbf{L} & \mathbf{N}\end{bmatrix}$, $\gamma > 0$, and $\mathbf{P} \succ 0$. By applying the Schur complement [2] twice (9) is equivalent to

$$\begin{bmatrix} -\mathbf{P} & * & * & * \\ \mathbf{0} & -\gamma^2\mathbf{I}_{n+2m} & * & * \\ \mathbf{M} & \tilde{\mathbf{B}}_\mathrm{d} & -\mathbf{P}^{-1} & * \\ \mathbf{C}_\mathrm{e} & \mathbf{D}_\mathrm{e} & \mathbf{0} & -\mathbf{I}_n \end{bmatrix} \prec 0 \ . \tag{10}$$

With $\mathbf{D}_\mathrm{e} = [\mathbf{D}_{\mathrm{e},1}\ \mathbf{D}_{\mathrm{e},2}\ \mathbf{D}_{\mathrm{e},3}]$, $\tilde{\mathbf{B}}_\mathrm{d} = \begin{bmatrix}\mathbf{I}_n & \mathbf{L} & \mathbf{N}\end{bmatrix}$ and multiplying (10) from the left and right with $\begin{bmatrix} \mathbf{I}_{2n+2m} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I}_n \\ \mathbf{0} & \mathbf{I}_n & \mathbf{0} \end{bmatrix}$, the inequality (10) can be written as

$$\begin{bmatrix} -\mathbf{P} & * & * & * & * & * \\ 0 & -\gamma^2\mathbf{I}_n & * & * & * & * \\ 0 & 0 & -\gamma^2\mathbf{I}_m & * & * & * \\ 0 & 0 & 0 & -\gamma^2\mathbf{I}_m & * & * \\ \mathbf{C}_\mathrm{e} & \mathbf{D}_{\mathrm{e},1} & \mathbf{D}_{\mathrm{e},2} & \mathbf{D}_{\mathrm{e},3} & -\mathbf{I}_n & * \\ \mathbf{M} & \mathbf{I}_n & \mathbf{L} & \mathbf{N} & \mathbf{0} & -\mathbf{P}^{-1} \end{bmatrix} \prec 0 \ . \tag{11}$$

By multiplying (11) from the left and right with $\begin{bmatrix} \mathbf{I}_{3n+2m} & \mathbf{0} \\ \mathbf{0} & \mathbf{P} \end{bmatrix}$ and $\begin{bmatrix} \mathbf{I}_{3n+2m} & \mathbf{0} \\ \mathbf{0} & \mathbf{P}^T \end{bmatrix}$, respectively, it is equivalent to

$$\begin{bmatrix} -\mathbf{P} & * & * & * & * & * \\ 0 & -\gamma^2\mathbf{I}_n & * & * & * & * \\ 0 & 0 & -\gamma^2\mathbf{I}_m & * & * & * \\ 0 & 0 & 0 & -\gamma^2\mathbf{I}_m & * & * \\ \mathbf{C}_\mathrm{e} & \mathbf{D}_{\mathrm{e},1} & \mathbf{D}_{\mathrm{e},2} & \mathbf{D}_{\mathrm{e},3} & -\mathbf{I}_n & * \\ \mathbf{P}\mathbf{M} & \mathbf{P} & \mathbf{P}\mathbf{L} & \mathbf{P}\mathbf{N} & \mathbf{0} & -\mathbf{P} \end{bmatrix} \prec 0 \ . \tag{12}$$

Due to the products of two unknown matrices in $\mathbf{PM}$, $\mathbf{PL}$, and $\mathbf{PN}$, inequality (12) is nonlinear. With the change of variables $\Phi = \mathbf{PM}$, $\mathbf{W} = \mathbf{PL}$, and $\mathbf{X} = \mathbf{PN}$ we obtain the LMI

$$\begin{bmatrix} -\mathbf{P} & * & * & * & * & * \\ 0 & -\gamma^2\mathbf{I}_n & * & * & * & * \\ 0 & 0 & -\gamma^2\mathbf{I}_m & * & * & * \\ 0 & 0 & 0 & -\gamma^2\mathbf{I}_m & * & * \\ \mathbf{C}_\mathrm{e} & \mathbf{D}_{\mathrm{e},1} & \mathbf{D}_{\mathrm{e},2} & \mathbf{D}_{\mathrm{e},3} & -\mathbf{I}_n & * \\ \Phi & \mathbf{P} & \mathbf{W} & \mathbf{X} & \mathbf{0} & -\mathbf{P} \end{bmatrix} \prec 0 \ . \tag{13}$$

For the design of a TNL interval observer, the following relation between the observer gains holds [27]

$$\mathbf{T} + \mathbf{N}\mathbf{C} = \mathbf{I}_n \ . \tag{14}$$

Rearranging (14) for $\mathbf{T}$ and substituting into Φ leads to

$$\begin{aligned} \Phi = \mathbf{P}\mathbf{M} &= \mathbf{P}(\mathbf{T}\mathbf{A}_\mathrm{d} - \mathbf{L}\mathbf{C}) \\ &= (\mathbf{P} - \mathbf{X}\mathbf{C})\mathbf{A}_\mathrm{d} - \mathbf{W}\mathbf{C} \ . \end{aligned} \tag{15}$$

To design a TNL interval observer for system (1), the observer state matrix $\mathbf{TA}_\mathrm{d} - \mathbf{LC}$ has to be chosen so that it is Schur stable and non-negative. The stability is guaranteed if there exists a matrix $\mathbf{P} \succ 0$ so that (13) is fulfilled. Additionally, if $\mathbf{P}$ is chosen as a diagonal matrix, $\mathbf{TA}_\mathrm{d} - \mathbf{LC} \geq 0$ holds if $\Phi \geq 0$ [27]. Furthermore, we choose $\mathbf{C}_\mathrm{e}$ as an arbitrary diagonal matrix and $\mathbf{D}_\mathrm{e}$ as an arbitrary full matrix to allow for different convergence rates for the individual error terms. □

The choice of the matrices $\mathbf{C}_\mathrm{e}$ and $\mathbf{D}_\mathrm{e}$ of the performance output $\mathbf{z}_\mathrm{e}$ is user-defined. The matrix $\mathbf{C}_\mathrm{e}$ serves as a weighting matrix for the estimation errors, thereby quantifying the relative importance of minimizing the sensitivity to disturbances associated with each individual estimation error. The matrix $\mathbf{D}_\mathrm{e}$ characterizes the direct feedthrough from the disturbances to the performance output and consequently imposes a lower bound on the H_∞ norm, which can not be smaller than the direct feedthrough gain. The specification of $\mathbf{C}_\mathrm{e}$ and $\mathbf{D}_\mathrm{e}$ inherently involves a trade-off. When these matrices are subject to stringent constraints, for example $\mathbf{C}_\mathrm{e} = \mathbf{I}_n$ and $\mathbf{D}_\mathrm{e} = \mathbf{0}$, then the LMI solver may not find a feasible solution. Relaxing these constraints can resolve this problem, but at the expense of a suboptimal solution with regard to the H_∞ approach. Hence, determining $\mathbf{C}_\mathrm{e}$ and $\mathbf{D}_\mathrm{e}$ involves balancing the degree of relaxation required for achieving a feasible solution for the LMIs against the desired level of performance of the H_∞ approach.

3.2 Tuning of the estimation accuracy based on the ratio between selected observer gain matrices

This technique relies on influencing the elementwise ratio between the observer gains $\mathbf{L}$ and $\mathbf{N}$ by iteratively solving the LMIs with respect to a cost function that enforces the desired ratio to enhance the estimation accuracy. However, the matrices $\mathbf{L}$ and $\mathbf{N}$ do not explicitly appear in the LMIs (8). This is the result of the change of variables to express the set of bilinear matrix inequalities (12) by a set of LMIs (13). Accordingly, it is not possible to explicitly include $\mathbf{L}$ and $\mathbf{N}$ in the cost function to enforce the desired ratio. In compliance with the change of variables $\mathbf{W} = \mathbf{PL}$ and $\mathbf{X} = \mathbf{PN}$, $\mathbf{W}$ and $\mathbf{X}$ explicitly appear in the LMIs (8) and are therefore incorporated in the cost function. Consequently, we enforce a desired elementwise ratio between the matrices $\mathbf{W}$ and $\mathbf{X}$. By keeping $\mathbf{P}$ constant, the desired ratio between $\mathbf{L}$ and $\mathbf{N}$ can be enforced indirectly. To define the desired elementwise ratio, a design matrix $\mathbf{Y}(\alpha)$ is utilized, where α is an optimization parameter to be chosen. For example, the parameter α can be chosen based on a line search where α is adapted iteratively until suitable observer gains are found. The design process is shown in Algorithm 1.

Remark. Note that by applying Algorithm 1, it is not guaranteed that the exact ratio defined by α is enforced. This results from enforcing this ratio by minimizing the cost function in line 5 of Algorithm 1. The exact ratio is achieved only if the cost function is equal to zero and therefore $\mathbf{W} = \mathbf{W}_\mathrm{init}$ and $\mathbf{X} = \mathbf{Y}(\alpha) \odot \mathbf{X}_\mathrm{init}$ hold. If the cost function is not equal to zero, $\mathbf{W} \approx \mathbf{W}_\mathrm{init}$ and/or $\mathbf{X} \approx \mathbf{Y}(\alpha) \odot \mathbf{X}_\mathrm{init}$ hold, so that the desired ratio can only be achieved approximately.

Algorithm 1 Influencing the elementwise ratio between $\mathbf{L}$ and $\mathbf{N}$

Input: $\mathbf{Y}(\alpha)$, $\alpha \in \mathcal{L}$ *Design matrix, List* $\mathcal{L}$ *of all* α *to be evaluated*
Output: $\mathbf{T}$, $\mathbf{N}$, $\mathbf{L}$

1: Solve LMIs (8) without the additional cost function
2: $\mathbf{P}_{\text{init}} \leftarrow \mathbf{P}$, $\mathbf{W}_{\text{init}} \leftarrow \mathbf{W}$, $\mathbf{X}_{\text{init}} \leftarrow \mathbf{X}$
3: **for** each α in $\mathcal{L}$ **do**
4: $\quad \mathbf{P} \leftarrow \mathbf{P}_{\text{init}}$
5: $\quad$ Solve LMIs with minimization of the cost function $\|\mathbf{W} - \mathbf{W}_{\text{init}} + (\mathbf{X} - \mathbf{Y}(\alpha) \odot \mathbf{X}_{\text{init}})\|$
6: $\quad$ **if** Solution is feasible **then**
7: $\quad\quad \mathbf{L} = \mathbf{P}^{-1}\mathbf{W}$, $\mathbf{N} = \mathbf{P}^{-1}\mathbf{X}$, $\mathbf{T} = \mathbf{I}_n - \mathbf{NC}$
8: $\quad\quad \mathbf{L}_\alpha \leftarrow \mathbf{L}$, $\mathbf{N}_\alpha \leftarrow \mathbf{N}$, $\mathbf{T}_\alpha \leftarrow \mathbf{T}$
9: $\quad$ **end if**
10: **end for**
11: Evaluate the estimation accuracy based on time domain simulations for all obtained observer gains $\{(\mathbf{T}_\alpha, \mathbf{N}_\alpha, \mathbf{L}_\alpha) \mid \alpha \in \mathcal{L}\}$
12: Select $(\mathbf{T}, \mathbf{N}, \mathbf{L})$ from $\{(\mathbf{T}_\alpha, \mathbf{N}_\alpha, \mathbf{L}_\alpha) \mid \alpha \in \mathcal{L}\}$, so that the highest estimation accuracy is achieved

Remark. In this paper, lines 1 to 10 of Algorithm 1 are evaluated automatically based on a line search for α within a defined interval. On the contrary, the lines 11 to 12 are evaluated manually as shown in Section 4. Future work will deal with automatizing the evaluation of the estimation accuracy (line 11) and the selection of the most suitable observer gains (line 12) to include both tasks in the for-loop and to obtain a fully automatized Algorithm.

Remark. In Algorithm 1, the matrices $\mathbf{C}_\text{e}$ and $\mathbf{D}_\text{e}$ are adjusted by the solver without additional constraints according to the predefined structure. If required, additional constraints can be imposed on those matrices to obtain a desired performance. Moreover, the numeric values of $\mathbf{C}_\text{e}$ and $\mathbf{D}_\text{e}$ can provide insight for performance comparisons for different observer parameterizations.

As an example, consider two matrices $\boldsymbol{\mathcal{W}}_{\text{init}}$, $\boldsymbol{\mathcal{X}}_{\text{init}} \in \mathbb{R}^{2\times 2}$ that are obtained by solving the LMIs (8) without the additional cost function.

If $\mathbf{Y}(\alpha)$ is chosen as

$$\mathbf{Y}(\alpha) = \begin{bmatrix} \alpha & 1 \\ 1 & 1 \end{bmatrix} \ , \tag{16}$$

we aim to design two matrices $\boldsymbol{\mathcal{W}}$, $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{2\times 2}$ by minimizing the cost function of line 5 of Algorithm 1 with

$$\boldsymbol{\mathcal{W}} = \boldsymbol{\mathcal{W}}_{\text{init}} \ , \ \boldsymbol{\mathcal{X}} = \mathbf{Y}(\alpha) \odot \boldsymbol{\mathcal{X}}_{\text{init}} \ , \tag{17}$$

to enforce the following ratio between the elements of $\boldsymbol{\mathcal{W}}$ and $\boldsymbol{\mathcal{X}}$

$$\begin{aligned}(\boldsymbol{\mathcal{W}} \oslash \boldsymbol{\mathcal{X}})_{11} &= \frac{1}{\alpha}(\boldsymbol{\mathcal{W}}_{\text{init}} \oslash \boldsymbol{\mathcal{X}}_{\text{init}})_{11} \ , \\ (\boldsymbol{\mathcal{W}} \oslash \boldsymbol{\mathcal{X}})_{12} &= (\boldsymbol{\mathcal{W}}_{\text{init}} \oslash \boldsymbol{\mathcal{X}}_{\text{init}})_{12} \ , \\ (\boldsymbol{\mathcal{W}} \oslash \boldsymbol{\mathcal{X}})_{21} &= (\boldsymbol{\mathcal{W}}_{\text{init}} \oslash \boldsymbol{\mathcal{X}}_{\text{init}})_{21} \ , \\ (\boldsymbol{\mathcal{W}} \oslash \boldsymbol{\mathcal{X}})_{22} &= (\boldsymbol{\mathcal{W}}_{\text{init}} \oslash \boldsymbol{\mathcal{X}}_{\text{init}})_{22} \ .\end{aligned} \tag{18}$$

The observer gains $\mathbf{L}_{\text{init}}$ and $\mathbf{N}_{\text{init}}$ can be calculated by

$$\mathbf{L}_{\text{init}} = \mathbf{P}_{\text{init}}^{-1}\mathbf{W}_{\text{init}} \ , \ \mathbf{N}_{\text{init}} = \mathbf{P}_{\text{init}}^{-1}\mathbf{X}_{\text{init}} \ . \tag{19}$$

Accordingly, $\mathbf{L}$ and $\mathbf{N}$ are calculated by

$$\mathbf{L} = \mathbf{P}^{-1}\mathbf{W} \ , \ \mathbf{N} = \mathbf{P}^{-1}\mathbf{X} \ . \tag{20}$$

The matrix $\mathbf{P}$ is kept constant, therefore $\mathbf{P} = \mathbf{P}_{\text{init}}$ and $\mathbf{P}^{-1} = \mathbf{P}_{\text{init}}^{-1}$ hold. Then from (19) and (20), it follows that

$$\begin{aligned}(\mathbf{L} \oslash \mathbf{N})_{11} &= \frac{1}{\alpha}(\mathbf{L}_{\text{init}} \oslash \mathbf{N}_{\text{init}})_{11} = (\boldsymbol{\mathcal{W}} \oslash \boldsymbol{\mathcal{X}})_{11} \ , \\ (\mathbf{L} \oslash \mathbf{N})_{12} &= (\mathbf{L}_{\text{init}} \oslash \mathbf{N}_{\text{init}})_{12} = (\boldsymbol{\mathcal{W}} \oslash \boldsymbol{\mathcal{X}})_{12} \ , \\ (\mathbf{L} \oslash \mathbf{N})_{21} &= (\mathbf{L}_{\text{init}} \oslash \mathbf{N}_{\text{init}})_{21} = (\boldsymbol{\mathcal{W}} \oslash \boldsymbol{\mathcal{X}})_{21} \ , \\ (\mathbf{L} \oslash \mathbf{N})_{22} &= (\mathbf{L}_{\text{init}} \oslash \mathbf{N}_{\text{init}})_{22} = (\boldsymbol{\mathcal{W}} \oslash \boldsymbol{\mathcal{X}})_{22} \ .\end{aligned} \tag{21}$$

3.3 Tuning of the estimation accuracy based on the eigenvalues of the scaled observer system matrix

This technique is inspired by the LMI-based pole placement method, and it relies on influencing the eigenvalues of the observer system matrix to ensure a desired performance. Likewise to Section 3.2, the observer system matrix does not explicitly appear in the LMIs (8). According to (8), the equality

$$\Phi = \mathbf{PM} = (\mathbf{P} - \mathbf{XC})\,\mathbf{A}_{\text{d}} - \mathbf{WC} \tag{22}$$

holds, where $\mathbf{M} = (\mathbf{TA}_{\text{d}} - \mathbf{LC})$ is the observer system matrix. Φ explicitly appears in the LMIs. Due to this, the eigenvalues of $\mathbf{M}$ can be indirectly influenced by specifying the eigenvalues of Φ leading to the desired performance of the designed interval observer. To achieve this, additional constraints are added to the set of LMIs, that enforce the eigenvalues of Φ to be in a desired range. Note that $\mathbf{P}$ is a diagonal matrix and $\mathbf{M}$ is a full matrix. Hence, Φ is also a full matrix. The expressions for the eigenvalues of Φ get more complex with higher dimensions, which exacerbates considering those expressions as additional design conditions. To circumvent this, we propose to simplify the expressions for the eigenvalues of Φ by diagonalizing Φ with the minimization of a corresponding cost function. If Φ and $\mathbf{P}$ are diagonal matrices, then $\mathbf{M}$ is also a diagonal matrix. Accordingly, the

eigenvalues of Φ can be calculated by multiplying the diagonal elements of $\mathbf{P}$ and $\mathbf{M}$. To define the range of the desired eigenvalues, an optimization parameter α is introduced. This parameter can be for example chosen based on a line search, where it is adapted iteratively until suitable observer gains are found. Additionally, a precision $\epsilon \geq 0$ is defined with which the desired eigenvalue has to be achieved. The design process is shown in Algorithm 2.

Algorithm 2 Specifying eigenvalues for Φ

Input: $\alpha \in \mathcal{L}$, ϵ *List $\mathcal{L}$ of desired eigenvalues to be evaluated, precision*
Output: $\mathbf{T}$, $\mathbf{N}$, $\mathbf{L}$

1: Add additional constraints for the desired eigenvalues to the LMIs (8)
2: **for** each α in $\mathcal{L}$ **do**
3: Solve LMIs (8) with minimization of the cost function $\|\Phi - (\mathbf{I}_n \odot \Phi)\|$
4: **if** Solution is feasible **then**
5: $\mathbf{L} = \mathbf{P}^{-1}\mathbf{W}$, $\mathbf{N} = \mathbf{P}^{-1}\mathbf{X}$, $\mathbf{T} = \mathbf{I}_n - \mathbf{NC}$
6: $\mathbf{L}_\alpha \leftarrow \mathbf{L}$, $\mathbf{N}_\alpha \leftarrow \mathbf{N}$, $\mathbf{T}_\alpha \leftarrow \mathbf{T}$
7: **end if**
8: **end for**
9: Evaluate the estimation accuracy based on time domain simulations for all obtained observer gains $\{(\mathbf{T}_\alpha, \mathbf{N}_\alpha, \mathbf{L}_\alpha) \mid \alpha \in \mathcal{L}\}$
10: Select $(\mathbf{T}, \mathbf{N}, \mathbf{L})$ from $\{(\mathbf{T}_\alpha, \mathbf{N}_\alpha, \mathbf{L}_\alpha) \mid \alpha \in \mathcal{L}\}$, so that the highest estimation accuracy is achieved

Remark. Note that by minimizing the cost function in line 3 of Algorithm 2, it is not guaranteed that the matrix Φ is exactly diagonalized. Φ only exhibits a diagonal structure if the cost function is equal to zero. If the cost function is not equal to zero, the desired eigenvalues can be placed only approximately in the desired range.

Remark. In this paper, lines 2 to 8 of Algorithm 2 are evaluated automatically based on a line search for α within a defined interval. On the contrary, the lines 1 and 9 - 10 are evaluated manually as shown in Section 4. Future work will deal with automatizing the evaluation of the estimation accuracy (line 9) and the selection of the most suitable observer gains (line 10) to include both tasks in the for loop. Additionally, we will work on automatically adding the additional constraints to the LMIs (line 1) to obtain a fully automatized algorithm.

Remark. In Algorithm 2, the matrices $\mathbf{C}_\mathrm{e}$ and $\mathbf{D}_\mathrm{e}$ are adjusted by the solver without additional constraints. In general, the same options as in Remark 3.2 apply for performance tuning and quantification.

As an example, consider a matrix $\Phi \in \mathbb{R}^{2\times 2}$ with

$$\Phi = \begin{bmatrix} p_{11} & 0 \\ 0 & p_{22} \end{bmatrix} \begin{bmatrix} m_{11} & 0 \\ 0 & m_{22} \end{bmatrix} = \begin{bmatrix} p_{11}m_{11} & 0 \\ 0 & p_{22}m_{22} \end{bmatrix} , \tag{23}$$

that was diagonalized by minimizing the cost function of line 3 of Algorithm 2. Because of Φ being a diagonal matrix, the eigenvalues of Φ are its diagonal elements. Consider specifying a range for the first eigenvalue $\lambda_1 = p_{11}m_{11}$. Following Algorithm 2, we add the following constraints to the existing set of LMIs

$$p_{11}m_{11} \geq \alpha - \epsilon, \; p_{11}m_{11} \leq \alpha + \epsilon \;\; , \tag{24}$$

so that $p_{11}m_{11} \in [\alpha - \epsilon \; ; \; \alpha + \epsilon]$ holds, where α defines the desired eigenvalue and ϵ defines the corresponding precision.

4 Simulation Example

In this section, the proposed methods are evaluated through two test cases. To illustrate the application of the tuning methods, we first apply them to the state estimation of a mass-spring-damper system. Subsequently, as previously outlined, the approach is extended to the state estimation of a lithium-ion battery cell.

4.1 Mass-spring-damper system

Consider a mass-spring-damper system with the continuous-time state-space representation

$$\mathbf{A} = \begin{bmatrix} 0 & 1 \\ -\frac{c}{m} & -\frac{d}{m} \end{bmatrix}, \; \mathbf{b} = \begin{bmatrix} 0 \\ \frac{1}{m} \end{bmatrix}, \; \mathbf{c}^T = \begin{bmatrix} 1 & 0 \end{bmatrix}, \; \mathbf{d} = \mathbf{0} \;\; , \tag{25}$$

with $m = 1$, $c = 10$ and $d = 0.5$. The process and measurement noise are characterized by $w = 0$ and $v = 0.025$. All variables are expressed in normalized units. The continuous-time system is discretized with the forward Euler method with the step size $\Delta t = 0.01$. To design the observer gains of the TNL interval observer, the LMIs are solved using MATLAB R2022a [25], YALMIP R20141218 [17], and SeDuMi 1.3 [24]. The resolution of the LMIs as well as the simulation are computed on an Intel(R) Core i7 processor operating at 3GHz and 32GB of RAM. Solving the LMIs without including additional constraints and objective functions for the observer gains $\mathbf{L}$ and $\mathbf{N}$ as described in Sections 3.2 and 3.3 leads to the estimation results shown in Figure 1 for an input signal $u = 0.1 \sin(t) + 0.1$. In the following, we design the observer gains $\mathbf{T}$, $\mathbf{N}$ and $\mathbf{L}$ based on the ratio between $\mathbf{L}$ and $\mathbf{N}$ as described in Section 3.2. From the approach without including any additional constraints or objective functions, we obtain the observer gains $\mathbf{L} = \begin{bmatrix} 0.7377 & -0.8316 \end{bmatrix}^T$ and $\mathbf{N} = \begin{bmatrix} 0.0287 & 0.6008 \end{bmatrix}^T$, which leads to a ratio $\mathbf{L} \oslash \mathbf{N} = \begin{bmatrix} 25.6908 & -1.3841 \end{bmatrix}^T$. Usually, the estimation accuracy can be improved with observer gains, where the elementwise ratio is similar. Therefore, we define the design matrix

$$\mathbf{Y}(\alpha) = \begin{bmatrix} \alpha & 1 \end{bmatrix}^T \;\; , \;\; \alpha \in [20 \; ; \; 50] \;\; , \tag{26}$$

to decrease the ratio $(\mathbf{L} \oslash \mathbf{N})_{11}$. The simulation results in Figure 2 indicate a slight improvement in estimation accuracy. Here, the ratio $(\mathbf{L} \oslash \mathbf{N})_{11}$ varies in

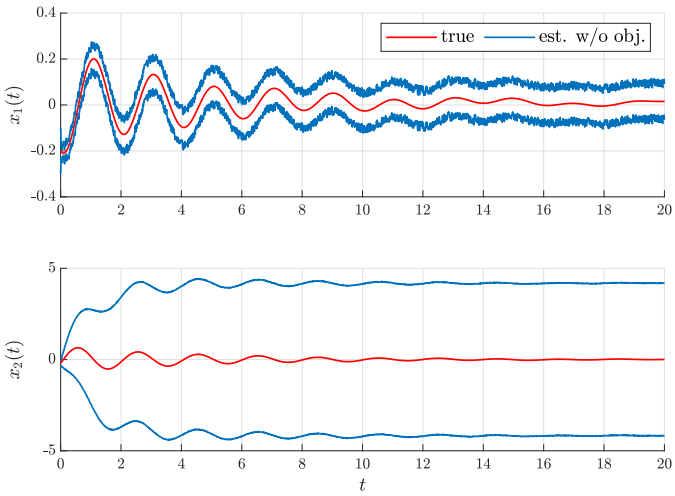


Figure 1: State estimation for the mass-spring-damper system (25) without including additional constraints and objective functions.

the interval [5.6594 ; 7.6987]. Although a marginal improvement in estimation accuracy was observed, the enhancement is negligible in practical terms. In the following, the tuning method based on the eigenvalues of the scaled observer system matrix as described in Section 3.3 is applied to assess whether a more substantial improvement in estimation accuracy can be achieved. From the approach without including any additional constraints or objective functions, the obtained eigenvalues are $\text{eig}\,(\Phi) = \begin{bmatrix}0.2693 & 0.0324\end{bmatrix}^T$ and $\text{eig}\,(\mathbf{M}) = \begin{bmatrix}0.2360 & 0.9908\end{bmatrix}^T$. To enhance the estimation accuracy, we choose to increase the first eigenvalue of the matrix $\mathbf{M}$. Based on (22), (25), and

$$\mathbf{P} = \begin{bmatrix} p_{11} & 0 \\ 0 & p_{22} \end{bmatrix} \ , \ \mathbf{K} = \begin{bmatrix} k_{11} \\ k_{21} \end{bmatrix} \ , \ \mathbf{K} \in \{\mathbf{L},\ \mathbf{N},\ \mathbf{W},\ \mathbf{X}\} \ , \tag{27}$$

it can be calculated as

$$\begin{aligned} \lambda_1 &= p_{11} - 1.0 \cdot p_{11} l_{11} - 1.0 \cdot p_{11} n_{11} \\ &= p_{11} - 1.0 \cdot w_{11} - 1.0 \cdot x_{11} \ , \end{aligned} \tag{28}$$

where the coefficients are rounded to the nearest integer. This leads to the additional constraints

$$\lambda_1 \geq \alpha - \epsilon \ , \ \lambda_1 \leq \alpha + \epsilon \ , \ \alpha \in [0\ ;\ 2] \ , \ \epsilon = 10^{-3} \ , \tag{29}$$

which are added to the existing LMIs. Figures 3 and 4 show the simulation results. The estimation accuracy can be enhanced significantly by applying this tuning method, especially for the second state variable. With regard to the computational effort, the average runtime per iteration for LMI resolution and the simulation were 0.29s and 0.04s, respectively.

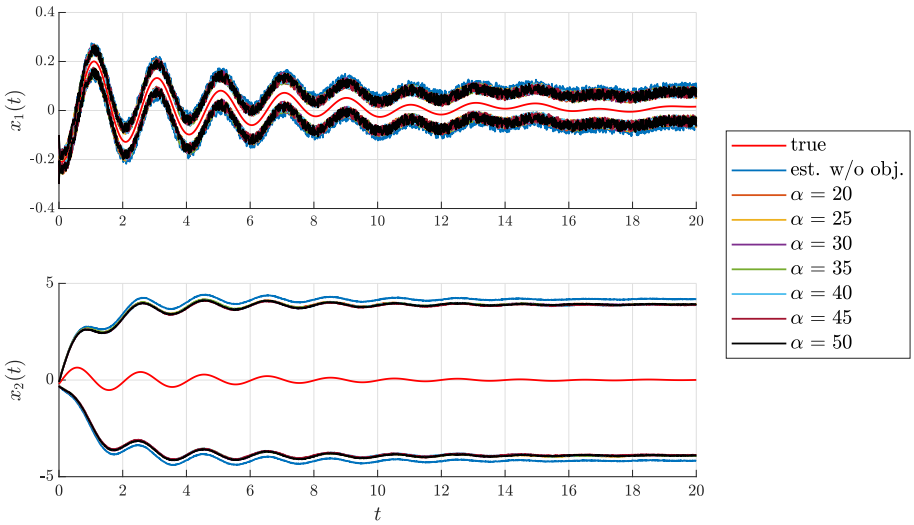


Figure 2: State estimation for the mass-spring-damper system (25) based on the ratio between **L** and **N**.

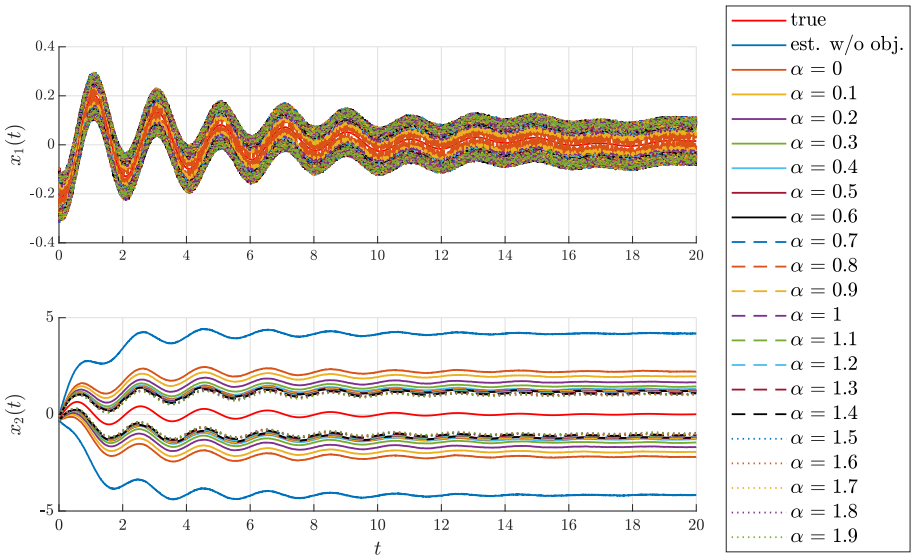


Figure 3: State estimation for the mass-spring-damper system (25) based on the eigenvalues of the scaled observer system matrix.

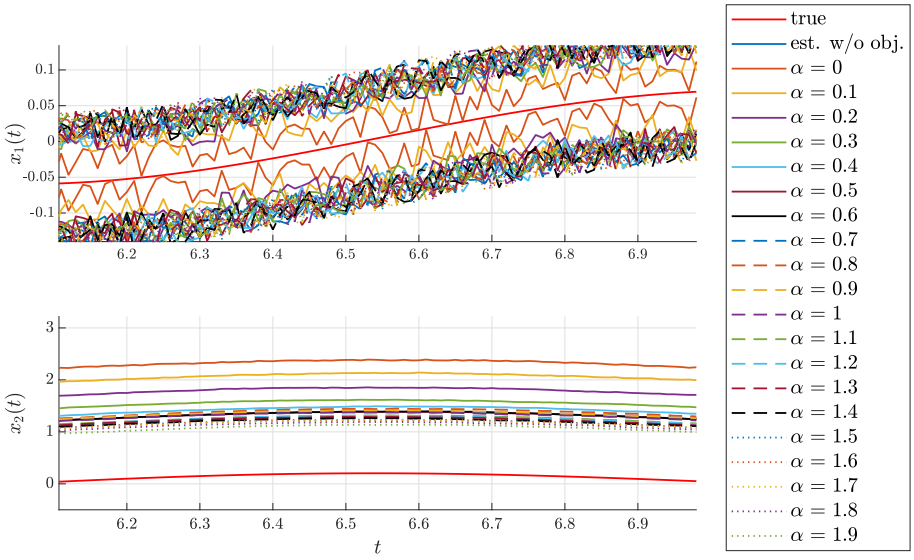


Figure 4: Detailed view of Figure 3.

4.2 Lithium-ion battery cell

The design process is shown for a quasi-linear model of a lithium-ion battery cell. The modeling of the battery cell is described in detail in [14]. We will use the model taken from [14, Section III-B] as an example. The parameters used in this example are the same as the parameters in the referenced paper, so that the nominal model is evaluated for a state of charge of 0.6. In this case, we obtain the matrices

$$\mathbf{A}_\mathrm{d} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 - 0.92 \cdot 10^{-2} & 0 & 0 \\ 0 & 0 & 1 - 0.02 \cdot 10^{-2} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \text{and} \tag{30}$$

$$\mathbf{C} = \begin{bmatrix} 2.33 & -1 & -1 & 1 \\ 48.56 & -99.91 & -99.99 & 100 \end{bmatrix} .$$

Our goal is to design a TNL interval observer to estimate the state variables, which cannot be measured directly. The state vector consists of the state of charge σ and the equivalent circuit voltages v_TS, v_TL and z. At first, we design the matrices $\mathbf{T}$, $\mathbf{N}$, and $\mathbf{L}$ based on the ratio of the matrices $\mathbf{L}$ and $\mathbf{N}$. Therefore, we choose the matrix $\mathbf{Y}(\alpha)$ as

$$\mathbf{Y}(\alpha) = \begin{bmatrix} 1 & 1 & \alpha & 1 \\ 1 & 1 & \alpha & 1 \end{bmatrix}^T , \quad \alpha \in [0\,;\,1] \quad . \tag{31}$$

Figure 5 shows the estimation results of the state variables of the battery model, where infeasible solutions are excluded. By varying α from 0 to 1, the estimation

accuracy can be influenced. In Figure 5a, the estimation results based on $\alpha = 1$ and $\alpha = 0.8$ are not suitable for estimating all state variables, because of the low estimation accuracy characterized by the large interval width between the estimated lower and upper bounds. The estimation accuracy can be increased significantly by choosing α so that $\alpha \in \{0.1, 0.2, 0.3, 0.4, 0.5, 0.6\}$. Figure 5b shows that in the latter case all values of α lead to similar estimation accuracies for the respective state variables. In the following, we design the matrices $\mathbf{T}$, $\mathbf{N}$, and $\mathbf{L}$ based on the eigenvalues of Φ. Therefore, we choose to specify the third eigenvalue. Based on (22), (30), and

$$\mathbf{P} = \begin{bmatrix} p_{11} & 0 & 0 & 0 \\ 0 & p_{22} & 0 & 0 \\ 0 & 0 & p_{33} & 0 \\ 0 & 0 & 0 & p_{44} \end{bmatrix} \ , \ \mathbf{K} = \begin{bmatrix} k_{11} & k_{12} \\ k_{21} & k_{22} \\ k_{31} & k_{32} \\ k_{41} & k_{42} \end{bmatrix} \ , \ \mathbf{K} \in \{\mathbf{L},\ \mathbf{N},\ \mathbf{W},\ \mathbf{X}\} \ , \tag{32}$$

it can be calculated as

$$\begin{aligned} \lambda_3 &= p_{33} + p_{33} l_{31} + 100 \cdot p_{33} l_{32} + p_{33} n_{31} + 100 \cdot p_{33} n_{32} \\ &= p_{33} + w_{31} + 100 \cdot w_{32} + x_{31} + 100 \cdot x_{32} \ , \end{aligned} \tag{33}$$

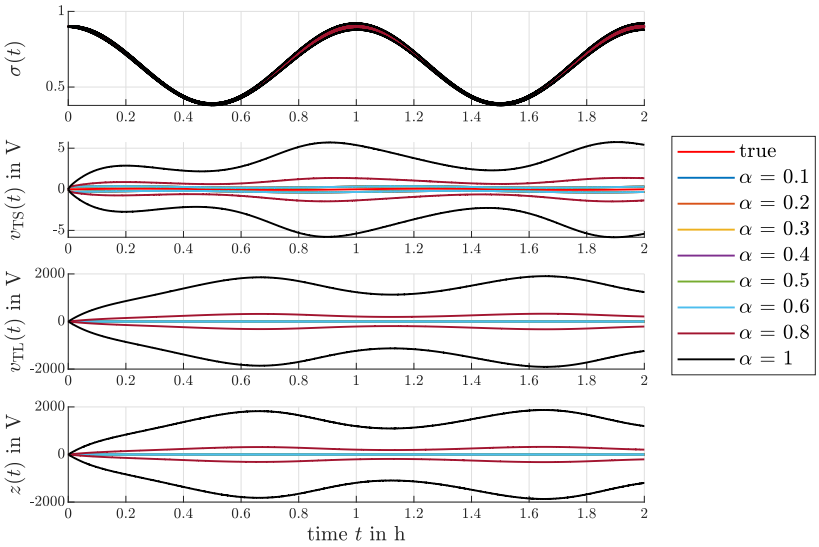
where the coefficients are rounded to the nearest integer. This leads to the additional constraints:

$$\lambda_3 \geq \alpha - \epsilon \ , \ \lambda_3 \leq \alpha + \epsilon \ , \ \alpha \in [0\ ;\ 2] \ , \ \epsilon = 10^{-3} \ . \tag{34}$$

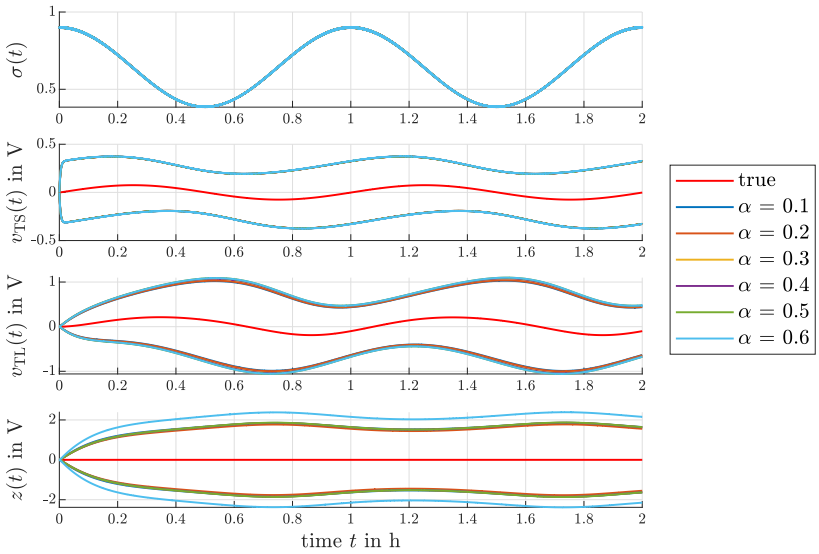
Figure 6 shows the simulation results. Again, infeasible solutions are excluded. It can be seen that different values of α lead to different estimation accuracies. Figures 6b, 6c and 6d show that selected values of α result in similar estimation results and therefore can be grouped. This is illustrated in Table 1. From Table 1 it is clear that different values of α lead to different estimation accuracies for the individual state variables v_{TS}, v_{TL}, and z. This is shown in Figure 7 for three selected values of α. Figure 7a shows that a value of $\alpha = 1.4$ leads to the best estimation accuracy for v_{TS}, while it results in a lower estimation accuracy for v_{TL} and a good estimation accuracy for z. In Figure 7b, the opposite is observed. A value of $\alpha = 0.6$ leads to the best estimation accuracy for z, while it results in a lower estimation accuracy for v_{TS} and a medium accuracy for v_{TL}. Figure 7c shows the estimation results for $\alpha = 0.4$, which represents a compromise between the estimation qualities of all three state variables. This value leads to a very high estimation accuracy, which is not necessarily the best estimation accuracy, for all three state variables. With regard to the computational effort, the average runtime per iteration for LMI resolution and the simulation were 0.41s and 12.43s, respectively.

5 Conclusions and Outlook

In this paper, we design a TNL interval observer for systems for which the LMIs do not yield a feasible solution with the standard specification of the performance

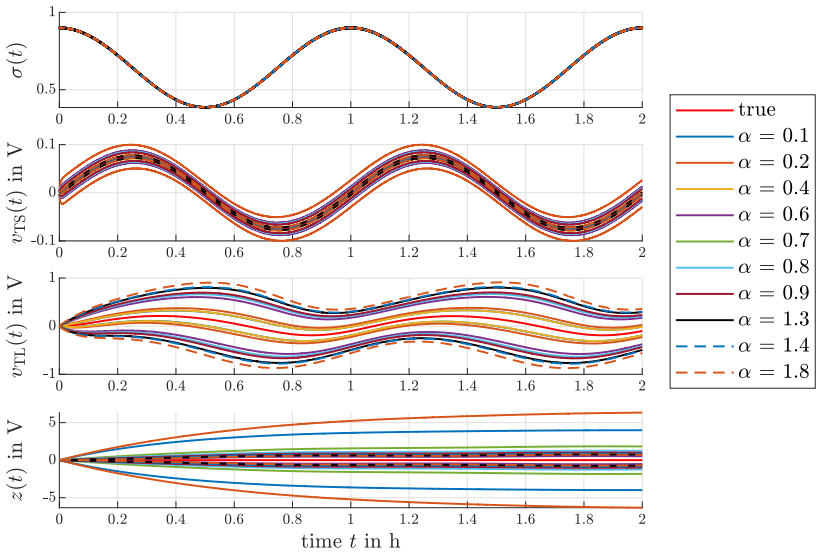


(a) Estimation results for $\alpha \in [0\ ;\ 1]$.

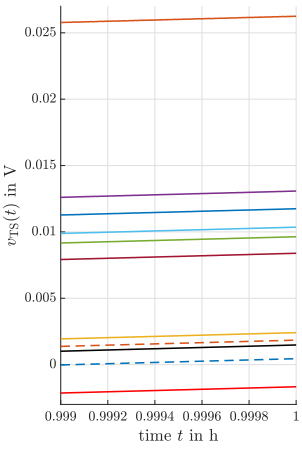


(b) Detailed view for Figure 5a.

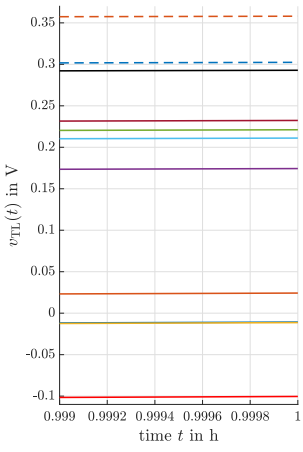
Figure 5: Estimated state variables, where the observer gains are designed based on the ratio of $\mathbf{L}$ and $\mathbf{N}$ defined by the matrix $\mathbf{Y}(\alpha)$ according to (31). Infeasible solutions are excluded. Figure 5a shows an overview for all feasible solutions of multiples of $\alpha = 0.1$. Figure 5b provides a detailed view for Figure 5a.



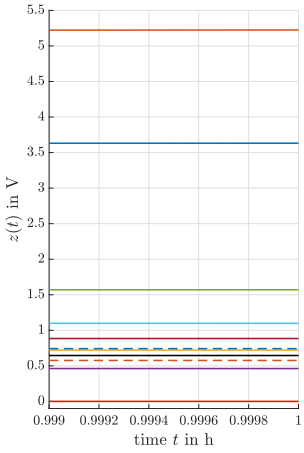
(a) Estimation results for $\alpha \in [0\ ;\ 2]$.



(b) Detailed view of v_{TS} in Figure 6a.



(c) Detailed view of v_{TL} in Figure 6a.



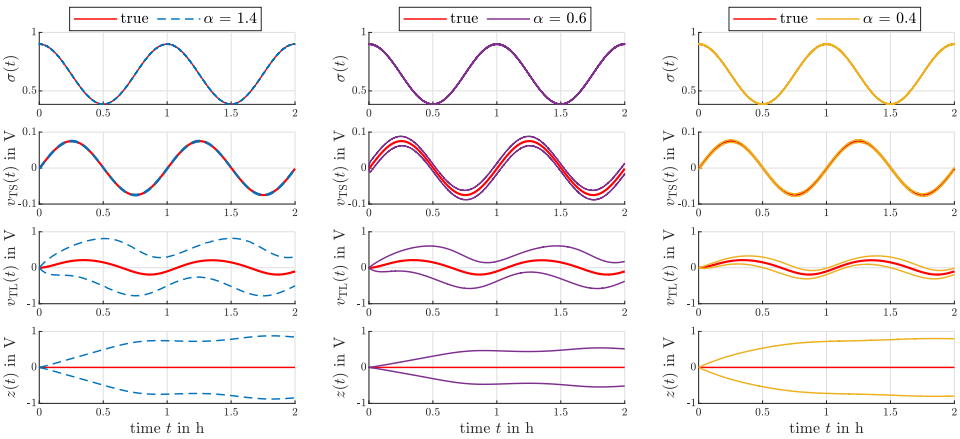
(d) Detailed view of z in Figure 6a.

Figure 6: Estimated state variables, where the observer gains are designed based on the eigenvalues of Φ according to (34). Infeasible solutions are excluded. Figure 6a shows an overview for all feasible solutions of multiples of $\alpha = 0.1$. Figures 6b, 6c, and 6d provide a detailed view for Figure 6a.

Table 1: Values of α according to Figure 6 sorted by estimation accuracy for the state variables v_{TS}, v_{TL} and z with respect to interval widths, where a tight interval between the lower and upper bound represents a high estimation accuracy. Values marked within the same group result in a similar estimation accuracy.

interval widths	$v_{\mathrm{TS}}(t)$		$v_{\mathrm{TL}}(t)$		$z(t)$	
tight	1.4		0.4		0.6	
	1.3	Group 1	0.1	Group 1	1.8	
	1.8		0.2		1.3	
	0.4		0.6		0.4	Group 1
	0.9		0.8	Group 2	1.4	
↓	0.7		0.7		0.9	
	0.8	Group 2	0.9		0.8	
	0.1		1.3	Group 3	0.7	
	0.6		1.4		0.1	
wide	0.2		1.8		0.2	

criteria. The design conditions, that include stability conditions, positivity conditions, and an H_∞ performance criterion, can be formulated as a set of LMIs which together with this observer structure initially do not yield a feasible solution for the battery model because of inappropriately parameterized performance criteria. This implies that a feasible solution exists, that cannot be identified by the solver. Therefore, we propose extended design conditions that allow for identifying feasible solutions for the LMIs and two techniques that provide a framework to systematically tune the observer gains to influence the estimation accuracy. The extended design conditions take into account the matrices of the virtual output equation of the error dynamics as additional degrees of freedom for the LMIs to exploit structural feasibility while the virtual output equation is used to impose performance objectives on the transfer function between the input uncertainty and the virtual output to reduce the influence of uncertainties on the estimation error. We additionally propose two techniques based on enforcing interpretable structures in the observer gains that can lead to an enhanced estimation quality. This is achieved by adding additional design degrees of freedom and additional constraints to the set of LMIs and solving the LMIs by minimizing appropriate cost functions. With the first technique shown in Section 3.2, we enforce a desired ratio between the elements of selected observer gains. The second technique shown in Section 3.3 is used to influence the observer dynamics by specifying desired eigenvalues for the scaled observer system matrix. The proposed techniques are initially applied to the state estimation of a mass-spring-damper system to illustrate the fundamental application of those methods. A second example, where the state variables of a lithium-ion battery model are estimated, illustrates the effectiveness of the proposed techniques. Based on the elementwise ratio between the observer gains, the estimation accuracy varies from a high estimation accuracy represented by tight interval width between the estimated lower and upper bounds to a low estimation



(a) $\alpha = 1.4$. (b) $\alpha = 0.6$. (c) $\alpha = 0.4$.

Figure 7: Estimated state variables, where the observer gains are designed based on the eigenvalues of Φ according to (34) for selected values of α; Figure 7a; $\alpha = 1.4$, Figure 7b; $\alpha = 0.6$, Figure 7c; $\alpha = 0.4$.

accuracy. In the lithium-ion battery example, we observed that selected ratios lead to similar estimation accuracies for the respective state variables. Regarding the eigenvalues of the scaled observer system matrix, the estimation accuracy is influenced by the chosen eigenvalues. Here, the estimation accuracy additionally varies between the individual state variables. An eigenvalue that leads to a high estimation accuracy for one state variable can lead to a low estimation accuracy for another state variable. Additionally, we observed that selected eigenvalues lead to similar estimation results and can therefore be grouped. Both techniques take into account cost functions that are minimized when solving the LMIs. Accordingly, the desired ratios between selected observer gains and the desired eigenvalues of the scaled observer system matrix are only achieved exactly if the cost functions are equal to zero. If the cost functions are not equal to zero, the desired ratios and the desired eigenvalues are only achieved approximately.
The computational effort required for solving the LMIs and for simulating the system depends on both the choice of the LMI solver and the implementation of the underlying differential equations. In the test cases presented in Section 4, the average runtime per iteration for solving the LMIs was on the order $10^{-1}s$, while the average simulation time per iteration was on the order of $10^{1}s$. In applications where the LMIs can be solved offline, these computational costs are typically negligible. However, in scenarios requiring real-time computation of the LMIs, the corresponding runtimes must be carefully taken into consideration. Both test cases presented in this paper are applicable at runtime.

In conclusion, by taking into account the matrices of the virtual output equation of the error dynamics as additional design degrees of freedom for the LMIs

allows for exploiting structural feasibility so that feasible solutions for the LMIs can be obtained. By applying the two techniques shown in Sections 3.2 and 3.3 the estimation accuracy can be significantly enhanced. The algorithms of the two techniques are not yet fully automatized and do not yet guarantee an optimal solution of the LMIs with regard to estimation accuracy. The evaluation of the estimation accuracy for the obtained observer gains as well as the selection of the most suitable observer gains has to be done manually by the designer for both techniques. Additionally, for the technique shown in Section 3.3, the additional constraints for the desired eigenvalues have to be added manually to the LMIs.

Future work will deal with fully automatizing the proposed techniques to provide a basis for systematizing the approaches shown in this paper. We will work on a framework to systematically design the observer gains of a TNL interval observer to obtain optimal observer gains with regard to estimation accuracy. For example, a clustering approach can be investigated to provide a systematic analysis for specifying eigenvalues of the scaled observer system matrix by taking into account the sensitivity of the state variables on the LMIs. Furthermore, a switching observer approach can be investigated to enhance the estimation accuracy for all state variables. Here, multiple observers can be designed that lead to a high estimation accuracy for a specific state variable. By switching between those observers, the aim is to achieve a high estimation accuracy for all state variables. Alternatively, the different interval observers can be operated in parallel. The resulting estimated intervals can be intersected with each other due to the guaranteed enclosure of the true value. Furthermore, a promising direction of future research is the observer design during system operation. Initially, an offline-designed interval observer is established. Subsequently, this interval observer is redesigned during system operation to adapt to changing system conditions. This approach is particularly relevant in scenarios where initial parameter uncertainties can be mitigated through interval estimations or contractor-based techniques. By leveraging these refined uncertainty bounds, the interval observer can be redesigned online to enhance the estimation accuracy. Another relevant application of online interval observer design are systems with time-varying parameters. For instance, lithium-ion battery systems subject to aging phenomena. In such scenarios, the interval observer can be redesigned online to adapt to evolving system parameters, thereby maintaining estimation accuracy and robustness. A prerequisite for online interval observer design is the real-time computation of the LMIs. An online interval observer design is feasible for both test cases shown in Section 4, due to the computationally fast resolution of the LMIs.

References

[1] Avilés, J. D., Moreno, J. A., Dávila, J. A., Becerra, G., Flores, F., Chávez, C. A., and Márquez, C. Stability radii-based interval observers for discrete-time nonlinear systems. *IEEE Access*, 10:3216–3227, 2022. DOI: 10.1109/ACCESS.2021.3139244.

[2] Boyd, S., El Ghaoui, L., Feron, E., and Balakrishnan, V. *Linear matrix inequalities in system and control theory*, Volume 15 of *SIAM Studies in Applied Mathematics*. SIAM, Philadelphia, USA, 1994. DOI: 10.1137/1.9781611970777.

[3] Calafiore, G. C. and Polyak, B. T. Stochastic algorithms for exact and approximate feasibility of robust LMIs. *IEEE Transactions on Automatic Control*, 46(11):1755–1759, 2001. DOI: 10.1109/9.964685.

[4] Dehnert, R., Damaszek, M., Lerch, S., Rauh, A., and Tibken, B. Robust feedback control for discrete-time systems based on iterative LMIs with polytopic uncertainty representations subject to stochastic noise. *Frontiers in Control Engineering*, 2, 2022. DOI: 10.3389/fcteg.2021.786152.

[5] Ebihara, Y. and Hagiwara, T. A dilated LMI approach to robust performance analysis of linear time-invariant uncertain systems. In *Proceedings of the 2003 American Control Conference*, pages 839–844, Denver, CO, USA, 2003. IEEE. DOI: 10.1109/ACC.2003.1239126.

[6] Efimov, D. and Raïssi, T. Design of interval observers for uncertain dynamical systems. *Automation and Remote Control*, 77(2):191–225, 2016. DOI: 10.1134/S0005117916020016.

[7] Ellero, N., Gucik-Derigny, D., and Henry, D. Multiobjective interval observer via LMI techniques for fault detection. In *Proceedings of the 2016 3rd Conference on Control and Fault-Tolerant Systems*, pages 485–490, Barcelona, Spain, 2016. IEEE. DOI: 10.1109/SYSTOL.2016.7739796.

[8] Ethabet, H., Dadi, L., Raïssi, T., and Aoun, M. L∞ set-membership estimation for continuous-time switched linear systems. In *Proceedings of the IEEE International Workshop on Mechatronics Systems Supervision*, pages 1–6, Hammamet, Tunisia, 2023. IEEE. DOI: 10.1109/IW_MSS59200.2023.10369668.

[9] Farina, L. and Rinaldi, S. *Positive Linear Systems: Theory and Applications*, Volume 50 of *Pure and Applied Mathematics*. Wiley, 2000. DOI: 10.1002/9781118033029.

[10] Gouzé, J.-L., Rapaport, A., and Hadj-Sadok, M. Z. Interval observers for uncertain biological systems. *Ecological Modelling*, 133(1-2):45–56, 2000. DOI: 10.1016/S0304-3800(00)00279-9.

[11] He, Z. and Xie, W. Control of non–linear switched systems with average dwell time: interval observer–based framework. *IET Control Theory & Applications*, 10(1):10–16, 2016. DOI: 10.1049/iet-cta.2015.0285.

[12] Ifqir, S., Ichalal, D., Ait-Oufroukh, N., and Mammar, S. A new switched interval observer design for vehicle lateral dynamics estimation. In *Proceedings*

of the 2024 American Control Conference, pages 1–6, Toronto, ON, Canada, 2024. IEEE. DOI: 10.23919/ACC60939.2024.10644308.

[13] Khan, A., Xie, W., Zhang, B., and Liu, L.-W. A survey of interval observers design methods and implementation for uncertain systems. *Journal of the Franklin Institute*, 358(6):3077–3126, 2021. DOI: 10.1016/j.jfranklin.2021.01.041.

[14] Lahme, M., Rauh, A., and Defresne, G. Interval observer design for an uncertain time-varying quasi-linear system model of lithium-ion batteries. In *Proceedings of the 2024 European Control Conference*, Stockholm, Sweden, 2024. DOI: 10.23919/ECC64448.2024.10591102.

[15] Lee, K.-H., Lee, J. H., and Kwon, W. H. Sufficient LMI conditions for H∞ output feedback stabilization of linear discrete-time systems. *IEEE Transactions on Automatic Control*, 51(4):675–680, 2006. DOI: 10.1109/TAC.2006.872766.

[16] Levine, W. S., editor. *The Control Handbook: Control System Advanced Methods*. The Electrical Engineering Handbook Series. CRC Press, Boca Raton, second edition, 2011. DOI: 10.1201/b10384.

[17] Löfberg, J. YALMIP, 2025. Accessed: Feb 02, 2025. URL: https://yalmip.github.io/).

[18] Mizouri, H., Lamouchi, R., and Amairi, M. Robust fault detection based on functional interval observers for multivariable systems. In *Proceedings of the 2024 21st International Multi-Conference on Systems, Signals & Devices*, pages 620–625, Erbil, Iraq, 2024. IEEE. DOI: 10.1109/SSD61670.2024.10548557.

[19] Montes de Oca, S. and Puig, V. Adaptive threshold generation for robust fault detection using interval LPV observers. *IFAC Proceedings Volumes*, 42(8):444–449, 2009. DOI: 10.3182/20090630-4-ES-2003.00074.

[20] Rong, Q. and Irwin, G. W. Robust control design of linear systems with polytopic time-varying uncertainty: An iterative SDP approach. In *Proceedings of the 2003 European Control Conference*, pages 1774–1779, Cambridge, UK, 2003. IEEE. DOI: 10.23919/ECC.2003.7085222.

[21] Scherer, C. W. LMI relaxations in robust control. *European Journal of Control*, 12(1):3–29, 2006. DOI: 10.3166/ejc.12.3-29.

[22] Sehli, N., Taarit, K. I., Raïssi, T., and Ksouri, M. Interval observer design for uncertain discrete-time polytopic systems. In *Proceedings of the 17th International Multi-Conference on Systems, Signals & Devices*, pages 85–90, Monastir, Tunisia, 2020. IEEE. DOI: 10.1109/SSD49366.2020.9364217.

[23] Smith, H. L. *Monotone dynamical systems: An introduction to the theory of competitive and cooperative systems*, Volume 41 of *Mathematical surveys and monographs*. American Mathematical Society, Providence, Rhode Island, USA, 1995.

[24] Sturm, J. F. SeDuMi, 1998. Accessed: Feb 02, 2025. URL: https://sedumi.ie.lehigh.edu/).

[25] The MathWorks Inc. MATLAB, 2025. Accessed: Feb 02, 2025. URL: https://de.mathworks.com/).

[26] Tuan, H. D. and Apkarian, P. Relaxations of parameterized LMIs with control applications. In *Proceedings of the 37th IEEE Conference on Decision and Control*, pages 1747–1752, Tampa, FL, USA, 1998. IEEE. DOI: 10.1109/CDC.1998.758548.

[27] Wang, Z., Lim, C.-C., and Shen, Y. Interval observer design for uncertain discrete-time linear systems. *Systems & Control Letters*, 116:41–46, 2018. DOI: 10.1016/j.sysconle.2018.04.003.

[28] Zhang, Z. and Shen, J. A survey on interval observer design using positive system approach. *Franklin Open*, 4:100031, 2023. DOI: 10.1016/j.fraope.2023.100031.

Acta Cybernetica **27** (2026) 383–403.

Guaranteed Satisfaction of a Signal Temporal Logic Formula on Tubes*

Joris Tillet[ab], Antoine Besset[ac],
and Julien Alexandre dit Sandretto[ad]

Abstract

This paper considers the issue of how to deal with Signal Temporal Logic (STL) when taking into account uncertainties. The STL is a formalism with a large expressiveness to describe real-time properties on real-value signals. It is particularly used for system verification. This work focuses on extensions of STL that handle bounded uncertainties on predicates or on the signal itself, by using tubes to represent the sets of signals. In this way, it becomes possible to robustly check the satisfaction of specifications for a noisy system. However, some cases are undecidable due to uncertainty, and other ones are too complex to determine. Mainly, this paper provides a literature review and compares the few state-of-the-art STL monitors able to deal with tubes. In addition, it proposes to go further by introducing Boolean intervals to formalize undecidable cases, and by implementing a new STL formalism applied to sets in DynIbex, a guaranteed integration tool. Thus, STL specifications can be validated in a guaranteed way for a simulated system. As a result, we obtain the same reliable result as the state-of-the-art, but faster. A robotic application with a drone is proposed to illustrate the concept.

Keywords: STL, interval methods, system verification, tubes

1 Introduction

Cyber-physical systems (CPS) are engineered, physical or biological systems (continuous and real-value systems) with a numerical attached system (discrete states) as a monitor or a controller. Designers and users of such systems often require guarantees on the system behavior, hence the need of a runtime verification process. This formal verification can be done using temporal logic, a formalism able to specify the system requirements with temporal constraints.

*This work was supported by the CIEDS (French Interdisciplinary Center for Defense and Security) within the STARTS project.

[a]ENSTA, Institut Polytechnique de Paris, Palaiseau Cedex, France
[b]E-mail: joris.tillet@ensta.fr, ORCID: 0000-0002-1955-6725
[c]E-mail: antoine.besset@ensta.fr, ORCID: 0009-0004-2662-306X
[d]E-mail: alexandre@ensta.fr, ORCID: 0000-0002-6185-2480

DOI: 10.14232/actacyb.315054

In 1977, Linear Temporal Logic (LTL) [24] is introduced, allowing to describe discrete time properties on a discrete signal. This is mainly used for formal verification of software or digital hardware, but reaches its limits when considering real-time constraints. Metric Temporal Logic [15] goes further by providing a semantics to speak of real-time properties on discrete signal (Boolean traces over continuous time). When considering CPS, man needs an extension of LTL able to handle real-time properties on real-value signal. This is the purpose of Signal Temporal Logic (STL) [22].

Let $\mathbf{x} = (x_1, \ldots, x_n) : \mathbb{R}^+ \to \mathbb{R}^n$ be a signal of dimension n and depending on time t. Let φ be an STL formula. We note the satisfaction of the formula φ by the trace of the signal $\mathbf{x}$ starting at time t by: $(\mathbf{x}, t) \vDash \varphi$.

The Signal Temporal Logic syntax is defined iteratively by:

$$\varphi := \top \mid \mu \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \, \mathcal{U}_{[t_1,t_2]} \, \varphi_2 \tag{1}$$

with $\top$ denoting the *true* Boolean value, and μ is an atomic predicate: it is a constraint on the signal at a time t:

$$(\mathbf{x}, t) \vDash \mu \iff f(x_1(t), \ldots, x_n(t)) > 0. \tag{2}$$

$\neg$ is the logical *not* operator, $\wedge$ the logical *and* and $\mathcal{U}$ is the temporal *until* operator, defined below.

We also have:

$$\bot = \neg\top \qquad \textit{(False)} \tag{3}$$

$$\varphi_1 \vee \varphi_2 = \neg(\neg\varphi_1 \wedge \neg\varphi_2) \qquad \textit{(Or, using De Morgan's law)} \tag{4}$$

Example 1. Let $\mathbf{x} : \mathbb{R}^+ \to \mathbb{R}^6$ be the state of a drone in 3 dimension (position and speed):

$$\mathbf{x} = \begin{pmatrix} x \\ y \\ z \\ \dot{x} \\ \dot{y} \\ \dot{z} \end{pmatrix}. \tag{5}$$

We can express a limited speed when the drone is too close to the ground as follows:

$$\varphi = \mu_{\text{low\_height}} \implies \mu_{\text{speed}} \tag{6}$$

$$= \neg\mu_{\text{low\_height}} \vee \mu_{\text{speed}} \tag{7}$$

with

$$\mu_{\text{speed}} = \sqrt{\dot{x}^2 + \dot{y}^2 + \dot{z}^2} \leq 1 \tag{8}$$

$$\mu_{\text{low\_height}} = z \leq 10. \tag{9}$$

Then, if we consider the signal of a drone flying from time $t = 0$ s, and we check the satisfaction of the formula φ, we obtain (from (2)):

$$(\mathbf{x}, 0) \vDash \varphi \iff \begin{cases} z(0) > 10 \\ \text{or } \sqrt{\dot{x}(0)^2 + \dot{y}(0)^2 + \dot{z}(0)^2} \leq 1. \end{cases} \tag{10}$$

Our specification is only checked at a specific time of the signal (here at 0 s, *i.e.* the beginning of the signal). If we want to specify modalities with respect to time, we have to use the *until* operator of the STL.

Until The temporal operator $\mathcal{U}$ is the *until* operator defined on the time interval $[t_1, t_2]$ with $t_1, t_2 \in \mathbb{R}^+$ and $t_1 \leq t_2$, as follows:

$$(\mathbf{x}, t) \vDash \varphi_1 \, \mathcal{U}_{[t_1,t_2]} \, \varphi_2 \iff \exists t' \in [t_1, t_2] \; (x, t + t') \vDash \varphi_2 \text{ and } \forall t'' \in [t, t'] \; (x, t'') \vDash \varphi_1. \tag{11}$$

In other words, the until operator states that φ_1 must stay true until a time t' within $[t_1, t_2]$ when φ_2 is true. Note that nothing is specified on what happen after t'.

We can derive other useful operators from this operator:

$$\mathcal{F}_{[t_1,t_2]} \, \varphi = \top \, \mathcal{U}_{[t_1,t_2]} \, \varphi \qquad \textit{(Finally)} \tag{12}$$

$$\mathcal{G}_{[t_1,t_2]} \, \varphi = \neg \left(\mathcal{F}_{[t_1,t_2]} \, \neg \varphi \right) \qquad \textit{(Globally)} \tag{13}$$

Finally The *finally* operator $\mathcal{F}$ corresponds to the satisfaction of φ at a time t' within $[t_1, t_2]$:

$$(\mathbf{x}, t) \vDash \mathcal{F}_{[t_1,t_2]} \, \varphi \iff \exists t' \in [t_1, t_2] \; (\mathbf{x}, t + t') \vDash \varphi. \tag{14}$$

Globally The *globally* operator $\mathcal{G}$ corresponds to the satisfaction of φ during the whole period $[t_1, t_2]$:

$$(\mathbf{x}, t) \vDash \mathcal{G}_{[t_1,t_2]} \, \varphi \iff \forall t' \in [t_1, t_2] \; (\mathbf{x}, t + t') \vDash \varphi. \tag{15}$$

Example 2. If we consider again the example of the drone with limited speed, we can now express the same constraint but on the whole duration of the flight (final time is noted T), using the globally operator:

$$\varphi = \mathcal{G}_{[0,T]} \left(\mu_{\text{low\_height}} \implies \mu_{\text{speed}} \right). \tag{16}$$

STL enables a system designer to express properties on real-time and real-valued signals. This formalism is very flexible and can adapt easily to many systems, and yet it is concise.

Example 3. Now we can express more subtle specifications for our drone. For instance:

$$\mathcal{F}_{[0,60]}\ \mu_{\text{goal}} \quad \textit{(The drone must be at less than 5 m from the goal within 1 min)} \tag{17}$$

$$\mathcal{F}_{[0,60]}\ (\mathcal{G}_{[0,10]}\ \mu_{\text{goal}}) \quad \textit{(Same as (17) but the drone must stay around the goal during 10 s at least)} \tag{18}$$

$$\varphi_{\text{takeoff\_area}}\ \mathcal{U}_{[5,10]}\ \neg\mu_{\text{low\_height}} \quad \textit{(The drone must stay in its takeoff area until it reaches 10 m height wihtin the period 5 s – 10 s)} \tag{19}$$

with

$$\mu_{\text{goal}} = \sqrt{(x-20)^2 + (y+50)^2} \leq 5 \tag{20}$$

$$\varphi_{\text{takeoff\_area}} = |x| \leq 2 \wedge |y| \leq 2. \tag{21}$$

There are two ways to monitor a system based on an STL specification. Firstly, the *qualitative* monitoring returns a boolean corresponding to the satisfaction of a signal with an STL formula φ, like in equation (2). Secondly, the *quantitative* monitoring results in a real number ρ which is called the *robustness degree* [8]. The sign of ρ gives the qualitative information, and the absolute value is the robustness: the higher this number, the more robust. It can be viewed as the distance to the boundary between the set of signals satisfying φ and the one violating it.

This quantitative monitoring with its robustness is the mainly used approach to deal with aleatoric uncertainty of systems. However, if the signals are not known and only bounds are available, this approach becomes insufficient. Indeed, in the context of system verification based on STL, we should be able to deal with tubes, which is a robust representation for uncertain signals.

In this paper, a comprehensive overview of the state-of-the-art is proposed to show the need of a new formalism able to handle bounded uncertainties. The ability to consider sets of trajectories instead of a single one allows providing guaranteed results even when the system is too uncertain. The formalism proposed in this paper paves the way for using set-membership approaches when dealing with STL specifications. It can address every CPS whose trajectories can be bounded. There is no assumption of continuity. Theoretically, there is no limitation for this formalism. The limitations of the proposed approach are linked to the implementation. Indeed, nested temporal operators in the STL formula increase the complexity exponentially, and pessimism is added by the abstraction of sets.

The next section (2) of this paper proposes a literature review on STL in the context of handling uncertainties. Then, Section 3 presents how to deal with uncertainties inherent to tubes and introduces Boolean intervals, and an STL formalism adapted to set-membership approaches. The Section 4 is about the implementation of this newly introduced STL formalism and proposes a robotic application with the monitoring of a drone.

2 Literature review

Specifications of CPS have been widely studied. In particular, a survey can be found in [4] where qualitative and quantitative monitoring are reviewed, along with existing tools and applications in different domains. However, it does not deeply review the uncertainty issue. In this section, we propose to review more specifically how to deal with uncertainties, based on quantitative satisfaction, stochastic methods or set-membership approaches.

2.1 Quantitative satisfaction

Uncertainties have already been studied when considering the satisfaction of an STL formula (monitoring). The quantitative satisfaction can be used to handle some uncertainties, as in [6] where spatial and temporal uncertainties are considered, sometimes with the help of intervals. The space robustness is roughly the distance to border between satisfaction and violation, and the time robustness is the period around a time during which a formula is satisfied, *i.e.* the times θ^-, θ^+ representing the duration of the satisfaction of the formula at time t. Then the space-time robustness is the largest rectangle of height c around t. To compute the robustness degree on continuous time and space, the signal is discretized into a piecewise linear function. The complexity is linear with respect to the input signal size and the formula length. If we define the space robustness $\rho(\varphi, \mathbf{x}, t)$ for an STL formula φ and a trace $(\mathbf{x}, t)$, then we have the following Theorem, from [8]:

$$\rho(\varphi, \mathbf{x}, t) > 0 \implies (\mathbf{x}, t) \vDash \varphi \tag{22}$$

$$(\mathbf{x}, t) \vDash \varphi \text{ and } \|\mathbf{x} - \mathbf{x}'\|_\infty < \rho(\varphi, \mathbf{x}, t) \implies (\mathbf{x}', t) \vDash \varphi. \tag{23}$$

Equation (23) ensures us that if the considered signal satisfies the specifications with a robustness greater than the distance with the actual signal, then the real system satisfies the specifications.

In a few papers, intervals have been used to represent these uncertainties. For instance, in [30], an offline monitoring algorithm with intervals for finite time STL formulas is proposed. At each time step, an interval containing the estimated robustness value is computed. In [5], an online monitoring is proposed using an interval of all the possible quantitative satisfaction of a partial signal with unbounded future.

Finally, the robustness as described in the temporal logic literature corresponds to how far, in space or time, a signal is from violating or satisfying a property. In [6], a robustness sensitivity is also defined, so that we can compute the sensitivity of a formula to a given parameter. This robustness is a great tool to get a metric for evaluating signals, or do some specification mining (falsification problem to tune specifications and elicitation). However, this formalism corresponds to the robustness of one specification to one signal. It is not about the robustness to an uncertain hybrid system which presents uncountable many traces. In this paper, we are interested in describing temporal properties for CPS while considering noises and uncertain signals.

There are two main representations for uncertain systems: stochastic and set-membership approaches, whose literature is reviewed in the two next sections.

2.2 Stochastic STL

In this section, we focus on approaches to specify temporal properties over uncertain systems by using probabilities to represent errors due to noises and estimations. The main idea is to add probabilities on predicates, such that an STL formula can be considered as satisfied if the probability of violation is low enough.

For instance, in [27], a probabilistic STL is defined to allow violation of a specification with a given probability. A threshold ϵ on probability is defined to set the tolerance level in satisfaction of the properties. The paper deals with control design only, and monitoring is not addressed.

Another formalism to express probabilities on predicates can be found in [13]. The introduced temporal logic is called "chance-constrained temporal logic". However, the approach is restricted to deterministic and linear dynamical system. In [20], random STL is also proposed, with random predicates. Robot dynamics is modelled by a discrete-time, continuous space Markov decision process. Monitoring a stochastic system with STL is dealt within [10]. It gives a robustness measure (intervals) of stochastic trajectories. A monitor and motion planning are proposed using a sampling-based approach. The stochastic intervals of linear predicates are propagated on the STL.

To go further with the stochastic approach, some researchers have considered a risk-based STL which is able to consider the probability a property is violated weighted by the cost of such a violation. For instance, in [28], stochastic dynamical systems are studied. Constraints on atomic predicates are reformulated into deterministic affine constraints, using axiomatic risk theory. Other papers [18, 21] handle stochastic signals: the STL robustness risk is estimated for the value-at-risk. Some applications to data-driven approach are proposed.

2.3 Set-membership STL

Some researchers have worked on using set-membership approaches to deal with STL. Most papers propose to build robustness intervals, which hold every possible quantitative value for a given signal and an STL formula. It is the case in [30], where a monitoring algorithm with intervals for finite time STL formulas is proposed. It considers spatial deviation and time delay in the signal, but not the uncertainty in the STL predicates. On the other hand, the paper [3] uses intervals on predicates and on traces. It is based on a three-valued logical semantics, and relies on inclusion functions from interval arithmetic. The satisfaction of the STL formula is given by the resulting robustness interval I as follows: it is `true` if $I \subset [0, +\infty]$, `false` if $I \subset [-\infty, 0]$, and `undef` otherwise. This approach is also used in [5] in the context of online monitoring.

Finkbeiner paper's [9] introduces an offline monitoring algorithm, tailored for handling prevalent sensor uncertainty within the framework of STL. The approach

integrates error models, including bounded measurement inaccuracies, directly into the evaluation process of an STL formula. By systematically accounting for uncertainties, the algorithm provides robust verdicts regarding the satisfaction or violation of temporal properties. In [29], the authors introduce an online monitoring approach known as model predictive monitoring or model-based monitoring. This approach uses a dynamical model to estimate future states and provides qualitative evaluations for partially observed signals. This enables early certification or falsification of STL specifications.

Tubes Up to this point, presented set-membership approaches consider single signals as input, and then compute sets to consider uncertainties. However, in the context of real systems, signals cannot be exact, and we might want to consider a set of signals to be sure the actual trace is taken into account. Thus, if we are sure the actual trace is included in our set of signals, we can guarantee the final result. Such a set of signals can be represented by a *tube*, often used to represent intervals of trajectories [26]. In the remaining of this paper, a tube is noted $[x](t)$, and tube vectors are noted $[\mathbf{x}](t)$. A tube is represented by intervals of value at each time interval, as explained in Figure 1. The main drawback of this representation is that some additional traces might be taken into account, making the approach pessimistic. For instance, the orange signal in the figure is taken into account as it is included in the tube, even if it is physically impossible. So, a tube is not a signal with some added uncertainties, it is a set containing an infinite number of signals, but including every possible signal.

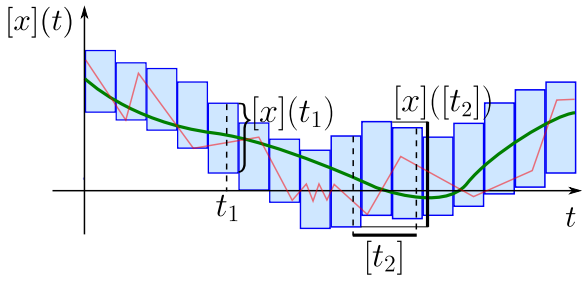


Figure 1: Representation of a set of traces by a tube. Two traces belonging to the tube are represented in green and in red. The red trace is considered whereas it might be physically impossible.

In the literature, a few papers consider sets of traces instead of single traces. Firstly, the Interval STL introduced in [3] considers uncertainties in predicates. Signals of intervals are handled, and quantitative satisfaction is returned as an interval.

In [25], Roehm *et al.* introduce a "Reachset Temporal Logic" (RTL) to address infinite set of traces. It requires to compute the reachable set. Then, the STL is sampled in time and converted into RTL. The tubes are cut in the time dimension

and STL formulas are interpreted as successive discrete states. When there exists a trace in the set of trace that does not satisfy the constraint, then the proposed monitor returns false (it is conservative).

At last, the approach proposed by Ishii *et al.* in [11] is closer to the first proposed STL monitor for single traces [22]: it searches times when properties become true or false, and then propagates the information in a bottom up manner to deduce the result for the whole STL formula. To deal with uncertain results, the algorithm can return `unknown` when there is an ambiguity on the satisfaction or the violation of every trace.

3 Monitoring tubes

When considering tubes instead of single traces, the monitor algorithm to check the satisfiability of an STL formula must take into account every possible trace within the tubes, which is infinite. The two papers introduced in last in the previous section are the only ones able to handle tubes, to our knowledge. These papers are reviewed more deeply in the following, along with an example.

3.1 Uncertainty on satisfaction of tubes

The main difficulty when considering the satisfaction of an STL formula for a tube (instead of a single trace) is that the tube may contain traces satisfying and traces not satisfying the formula at the same time. Then the result is neither completely false, nor completely true. To stay as conservative as possible, these undetermined cases can be considered as false. This is the choice of Roehm *et al.* [25], even if it introduces some pessimism, and prevents from knowing when every trace is false. The other option is to keep the undetermined case as a possible result, leading to a three-valued logic, as done by Ishii *et al.* [11].

However, in some cases it can be intricate to determine the true result. Indeed, there are an infinite number of potential traces in a tube, and there are also an infinite times to check for most STL formulas. This is why the state-of-the-art algorithms can fail to find the true answer and just consider it as an undetermined case even if it is not the case. This is illustrated in the following example.

Consider the tube $[x](t)$ represented in the Figure 2. We have a set point $\rho \in \mathbb{R}$ that we want our system to reach in the time interval $[t_1, t_2] \subset \mathbb{R}^+$. It means that we want the system to stay below ρ before t_1, and be equal or greater than ρ at least at one time during $[t_1, t_2]$. We first consider the STL atomic predicate:

$$\psi = ([x] \geq \rho). \tag{24}$$

In this example, during the period $[0, t_1]$, the predicate ψ is never satisfied for every possible trace included in the tube (every point in the blue area and before t_1 is below the ρ value). Similarly, after time t_2, the predicate ψ is always satisfied. On the other hand, during the period $[t_1, t_2]$, the satisfaction of the predicate ψ is ambiguous: some traces always satisfy ψ, others always violate it, and the remaining

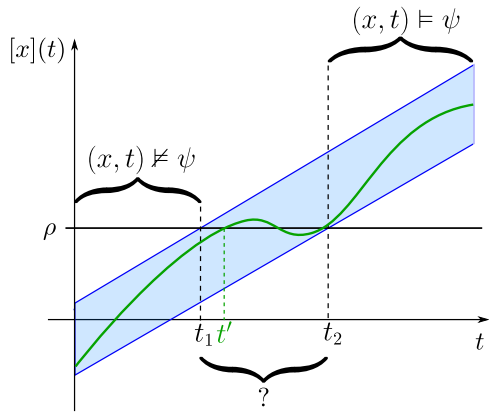


Figure 2: Simple tube representing a set of traces. A trace belonging to the tube is painted in green.

ones violate the predicate before satisfying it. This is typically an undetermined case.

Now, if we consider the full STL formula φ, corresponding to our problem:

$$\varphi = \neg\psi \, \mathcal{U}_{[t_1,t_2]}\psi, \tag{25}$$

it is well satisfied by any trace taken in the tube $[x](t)$ from time zero. Indeed, if we look at the definition of the *until* operator (see (11)), we have:

$$\begin{aligned}(x,0) \vDash \varphi &\overset{(25)}{\iff} (x,0) \vDash \neg\psi \, \mathcal{U}_{[t_1,t_2]} \, \psi \\ &\overset{(11)}{\iff} \exists t' \in [t_1,t_2] \begin{cases} (x,t') \vDash \psi \\ \wedge \, \forall t'' \in [0,t'] \, (x,t'') \nvDash \psi \end{cases}.\end{aligned}$$

It means that for every trace $(x,0)$ in the tube, we can find a time $t' \in [t_1,t_2]$ such that $x(t') \geq \rho$ and for every time t'' in $[0,t']$, we have $x(t'') < \rho$. Such a t' is given for the example of the green trajectory in the Figure 2. We can remark that the until formula does not specify anything on what happen after the time t'. Thus, we have the following proposition:

Proposition 1. *The STL formula $\varphi = \neg\psi \, \mathcal{U}_{[t_1,t_2]}\psi$, with $\psi = [x] \geq \rho$, is* `true` *for every signal in the blue tube of Figure 2.*

Proof. Let $(x,0)$ be a trace in the tube. Let t' be the smallest time within $[t_1,t_2]$ such that $x(t') \geq \rho$. We know such a t' exists because we have $x(t_2) \geq \rho$. Indeed, the interval of possible values for x at time t_2 is above ρ, according to Figure 2. Then we have $\forall t'' \in [0,t'], x(t'') < \rho$. □

However, the difficulty in finding an algorithm checking such a formula lies in the fact that the time t' may be different for every possible trace included in the tube. So, there is an infinite number of trace and time to check.

The two approaches proposed in the state-of-the-art both failed on this example to provide the expected `true` result.

Firstly, when applying the approach proposed by Ishii *et al.* [11] (Figure 3, left), we obtain the three-Boolean signal for the predicate ψ as already described. Then we apply the logical `not` operator ($\neg$), and finally we compute the three-Boolean signal for the complete formula from the two previously computed signals using a logical `and` ($\wedge$) operator and a time shifting. As we have an undefined result for the predicate ψ (depicted by the dark area in the figure), the ambiguity is propagated until the complete formula and is never removed. Thus, the final result is the undefined value for the tube at time zero. This result is not wrong, but it is not the expected one.

Secondly, the proposed approach of Roehm *et al.* [25] (Figure 3, right) uses only binary Boolean values. The time is discretized, and the predicate satisfaction is computed for every slice, resulting in a `false` value when there is a doubt (traces both validating and violating the formula in a same slice). Then the logical `not` operator ($\neg$) is easy to compute, and finally the `until` operator ($\mathcal{U}$) is transformed into a disjunction of cases depending only on the slices. At the end, the obtained result is `false` for this example, which is the conservative result of this approach.

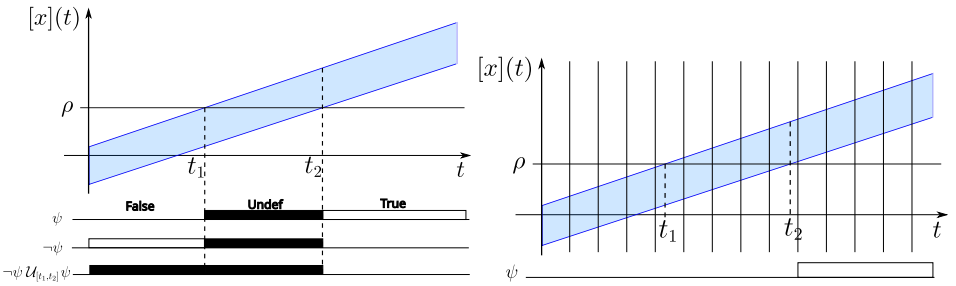


Figure 3: Illustration of the approaches from Ishii *et al.* [11] (left) and Roehm *et al.* [25] (right).

This simple example shows that the monitoring of a tube can easily become intricate, even with a small formula and a linear tube. In this example, and even more generally, in order to conclude the satisfaction rather than an unknown or false result, a solution is to use reachability analysis [16]. However, it is not in the scope of this paper, as reachability requires dealing directly with dynamical equations, and the computation cost can increase quickly when considering high-dimensional systems.

3.2 Boolean intervals

As stated above, the three-valued logic is more suitable when dealing with tubes to separate the cases where we are sure that every trace satisfies or every trace violates the formula, and when it is undetermined. In this paper, we propose to

use the formalism of Boolean intervals to represent such a three-valued logic. It is a well established formalism, based on interval analysis, with an arithmetic already known [12]. In addition, intervals are used as an abstraction for representing tubes, so the arithmetic stays the same.

Definition 1. *A Boolean interval is a subset of the set of Boolean* $\{true, false\}$*, i.e. an element of* $\mathbb{IB} = \{\emptyset, [0], [1], [0,1]\}$ *with:*

- $\emptyset$ *means impossible;*
- $[0]$ *means false* ($\bot$)*;*
- $[1]$ *means true* ($\top$)*;*
- $[0,1]$ *means undetermined (uncertain).*

The following Table 1 presents the different results with respect to the logical and temporal operators for the $[0,1]$ interval.

Table 1: Logical and temporal operators on the Boolean interval $[0,1]$.

$\neg[0,1] = [0,1]$	$[0,1] \vee 1 = 1$	$[0,1]\,\mathcal{U}_{[t_1,t_2]}\,0 = 0$
$[0,1] \wedge 1 = [0,1]$	$[0,1] \vee 0 = [0,1]$	$1\,\mathcal{U}_{[t_1,t_2]}\,[0,1] = [0,1]$
$[0,1] \wedge 0 = 0$	$[0,1]\,\mathcal{U}_{[t_1,t_2]}\,1 = [0,1]$	$0\,\mathcal{U}_{[t_1,t_2]}\,[0,1] = 0.$

What is important in this table is that some operations remove the uncertainty (*e.g.* $[0,1] \wedge 0 = 0$). So, depending on the form of the STL formula, the final result can be more or less pessimistic. Indeed, in practice, we hope to not obtain this uncertain case, as it does not contain actual information.

3.3 Set-Membership STL Formalism

In order to properly deal with tubes using STL specifications, we need to establish a specific formalism based on sets. Thus, we propose to slightly change the STL syntax to directly handle tubes and use Boolean intervals. As tubes are based on sets, the natural atomic predicates are the set operators.

In this way, we define the set predicate $\mu = \mathcal{S} \mid \mathcal{I}$ for boxes $[\mathbf{x}], [\mathbf{A}] \in \mathbb{IR}^n$ with:

- $\mathcal{S}$ the subset test ($[\mathbf{x}] \overset{?}{\subset} [\mathbf{A}]$);
- $\mathcal{I}$ the empty intersection test ($[\mathbf{x}] \cap [\mathbf{A}] \overset{?}{=} \emptyset$).

Finally, the new set-membership STL syntax can be written as:

$$\varphi := \beta \mid \mu \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1\,\mathcal{U}_{[t_1,t_2]}\varphi_2. \tag{26}$$

If we abstract sets by using level sets, this new syntax is not less general than previously. For example, we can represent a predicate $\mu = x > 0$ by $\mathcal{X}^\mu = \{x \in \mathbb{R} \mid x > 0\}$. Then, we have, for $t \in \mathbb{R}$:

$$([\mathbf{x}], t) \vDash \mu = \begin{cases} 1 & \text{if } [\mathbf{x}](t) \subset \mathcal{X}^\mu \quad (\mathcal{S}); \\ 0 & \text{if } [\mathbf{x}](t) \cap \mathcal{X}^\mu = \emptyset \quad (\mathcal{I}); \\ [0,1] & \text{otherwise.} \end{cases} \tag{27}$$

The differences introduced in this STL are only on Boolean values and atomic predicates. Thus, there are no changes on the temporal operators.

4 Application

Using the set-membership STL, it becomes possible to implement a monitor for tubes and use it on real systems. This is presented in this section.

4.1 Implementation

Tubes are used to represent sets of traces. The main advantage is to consider bounded uncertainties in a guaranteed way. Tubes are well-suited for addressing dynamical systems, and several set-based libraries already exist, as Vnode [23], DynIbex [2] or CAPD [14].

DynIbex offers a set of validated numerical integration methods based on Runge-Kutta schemes to solve initial value problem. In the library, there are already available objects and functions required for the set-membership STL. We build a subset of the set-membership STL syntax. Indeed, in this syntax, nested temporal operators are not allowed, contrary to (1). For instance, an STL formula like the following one[1] is not allowed here as there is a *finally* composed with an *until* operator:

$$(x \leq 10)\, \mathcal{U}_{[0,20]}\, (\mathcal{F}_{[0,5]}\, (x \geq 15)).$$

The full syntax is a work in progress. The definition is spread on several lines, with different variables to prevent nested temporal operators:

$$\beta := \emptyset \mid [0] \mid [1] \mid [0,1] \tag{28}$$
$$\mu := \mathcal{S} \mid \mathcal{I} \mid \beta \tag{29}$$
$$\pi := \neg\mu \mid \mu \tag{30}$$
$$\sigma := \mathcal{G}_{[t_1,t_2]}\, \pi \mid \mathcal{F}_{[t_1,t_2]}\, \pi \mid \pi_1\, \mathcal{U}_{[t_1,t_2]}\, \pi_2 \mid \pi \tag{31}$$
$$\theta := \sigma \wedge \theta \mid \sigma \vee \theta \mid \sigma \tag{32}$$

with $\mathcal{S}, \mathcal{I}$ the subset and empty intersection tests, respectively.

[1] This formula states that x must reach 10 within 20 s, and as soon as it happens x must become greater than 15 in less than 5 s.

All these operators have been implemented in DynIbex. For the sake of efficiency, some compositions of operators are directly implemented as one function. The following Table 2 gives the corresponding functions for each operator, based on the $\mathcal{S}$ predicate (it is very similar for the $\mathcal{I}$ predicate). It corresponds to the terms σ in the syntax (Equation 31).

Table 2: Corresponding DynIbex functions for each available set-membership STL. I is an interval of time.

Set-membership STL operator	**DynIbex function**
$\mathcal{S}$	`subset`
$\mathcal{G}_I\,\mathcal{S}$	`globally_subset`
$\mathcal{G}_I\,\neg\mathcal{S}$	`globally_not_subset`
$\mathcal{F}_I\,\mathcal{S}$	`finally_subset`
$\mathcal{F}_I\,\neg\mathcal{S}$	`finally_not_subset`
$\mathcal{S}_1\,\mathcal{U}_I\,\mathcal{S}_2$	`until`

4.2 Example with a simple tube

If we take again the example used in Section 3 and implement it in DynIbex, we obtain the same result as with the algorithm proposed by Ishii *et al.* in [11], *i.e.* `undefined` result ($[0, 1]$ Boolean interval). The simulation time (to generate the tube) takes less than 7 ms in average, and the verification of the STL formula takes less than 0.02 ms on a classical laptop (Intel i5-1335U, up to 4.6 GHz).

Figure 4 illustrates the obtained result. Blue boxes have been estimated in a guaranteed way using DynIbex: the real tube (represented in transparency as a reference) is well included inside the boxes.

Discussion This example enables comparing the two state-of-the-art methods able to deal with tubes (Ishii *et al.* [11] and Roehm *et al.* [25], see Section 3.1) and our approach. We obtain the same uncertain result as Ishii *et al.*, which is more accurate than the conservative false result of Roehm *et al.* The main advantage of our approach is that we evaluate only the satisfaction of the STL formula at the required specific time, which allows to be faster than Ishii *et al.* when the formula does not require to study all the signal duration. Indeed, in their work, Ishii *et al.* always compute the whole satisfaction signal for every sub-formula. The following Table 3 recaps the differences between presented approaches.

In order to obtain the expected true result, we may use tools from reachability analysis. However, it is not in the scope of this paper, as it requires the knowledge of the system dynamical equations, and often asks heavy computations. In our context, we do not need the system model since the tube may have been obtained only with measurements.

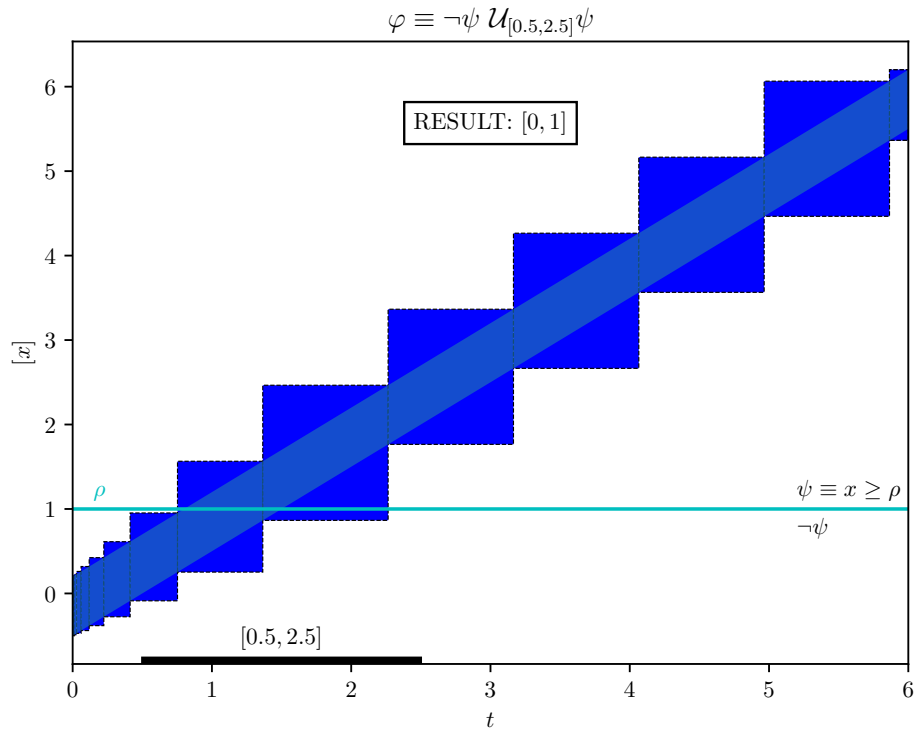


Figure 4: Result of the simulation of the simple tube example using DynIbex. The cyan line represents the ρ parameter. The result is "uncertain" ($[0, 1]$).

Table 3: Comparison with the state-of-the-art approaches on this example

Method	**Result**	**Running time for computing satisfaction**
Roehm *et al.* [25]	False	not implemented
Ishii *et al.* [11]	Undef	12 ms
Ours	[0,1]	0.02 ms

4.3 A test-case: monitoring of a drone

In this section is presented a test-case example where a drone (DJI Tello) is monitored with respect to an STL formula. This is a real-time monitoring: while the drone is flying, its state is estimated for the next 4 seconds and the STL specifications are verified on this estimation. Thus, if the satisfaction is not guaranteed, the drone can be slowed down or even stopped soon enough. In this example, tubes are used to represent uncertainties, mostly due to model approximation. Thus, STL

specifications, which are necessary to guarantee the safety of a critical system like a drone, have to be verified on the tubes. Finally, the generation of the tubes and the verification of the STL must be done in real-time, which requires short runtimes (with respect to the drone velocity).

Setup and modelling The drone state is denoted by y and its control is u. The drone is commanded in velocity along x, y and z axis. The control is based on a switched model, which consists on motion primitives that guide one robot movement, ensuring transitions between trajectories while adhering to its dynamical constraints [19]. Trajectories in the experiment are composed of 8 possible modes, and modes switching are only possible at every τ seconds.

We denote $y_{\text{ref}}(t)$ a reference trajectory and $u_{\text{ref}}(t)$ a reference control. They represent the precomputed movement and control of each motion primitive.

To track the given trajectory, the drone velocity control is:

$$u(t) = K_{\text{p}}(y_{\text{ref}}(t) - y(t)) + u_{\text{ref}}(t), \tag{33}$$

with K_{p} a P corrector.

Even though the behavior of a quadcopter can be represented by a highly nonlinear dynamical model, we consider the drone to be holonomic and use a simplified dynamical model for model-based monitoring. The model is a first-order system controlled by speed along each axis. This model is simple compared to dynamical models that account for the rotation of the quadrotor [1].

$$\dot{y} = u. \tag{34}$$

The main advantage of using a simple model is reducing computation time and complexity. To account for the simplifications in the dynamical models, we will introduce interval parameters as in [17].

$$[\dot{y}] = [\alpha_1] \cdot [u] + [\beta_1] \tag{35}$$

The presence of $[\alpha_1]$:

- Adjusts the assumption of immediate command responses by incorporating the drone speed profile,
- Accounts for uncertainties related to forces that depend on velocity,
- Considers uncertainties associated with the tuning of the corrector to track the trajectory.

The presence of $[\beta_1]$:

- Corrects uncertainties arising from sudden perturbations,
- Accounts for ambient noise and bias in the control inputs, including noise induced by the drone itself [17].

Furthermore, when reading sensor values used in the initial state of the dynamic equation, the initial state $\mathbf{y}_0$ becomes $[\mathbf{y}_0]$ as we introduce bounded uncertainties in the measurement or state estimation from filtering.

Monitoring The monitoring process aims to recognize if faults occur. In our case, it is done by verifying some constraints. Consider the following specification: "the drone must avoid obstacles within a given time horizon $h = N \times \tau$ and remain inside a designated area". The corresponding STL formula is given by:

$$\varphi_{\text{flag}} = \mathcal{G}_{[0,h]} \neg \mathcal{S}_{\text{collision}} \wedge \mathcal{G}_{[0,h]} \mathcal{S}_{\text{environment}}. \tag{36}$$

To monitor the system, the dynamical model is used to predict the future states within the time horizon [7]. The predicted state $[\tilde{\mathbf{y}}]$ is evaluated using a set-based simulation approach to reliably determine if a fault occurs. For example, it is important to ensure that the system stays within safe operating limits. In our setup, we have for a time $t \in \mathbb{R}$:

$$[\tilde{\mathbf{y}}](t) \subset [\mathbf{A}] \implies ([\tilde{\mathbf{y}}], t) \vDash \mathcal{S}_{\text{environment}}, \tag{37}$$

with $[\mathbf{A}]$ the bounding box of the authorized environment. If the predicted state violates such a constraint, the monitor flags a potential fault.

Finally, the satisfaction function C, which now represents the monitor flag for the STL formula φ_{flag}, is defined for an n-dimensional system as:

$$\begin{aligned} C : (\mathbb{R}^+ \to \mathbb{IR}^n) &\to \mathbb{IB} \\ [\tilde{\mathbf{y}}](t) &\mapsto \begin{cases} 1 & \text{if } [\tilde{\mathbf{y}}](t) \text{ satisfies the formula,} \\ 0 & \text{if } [\tilde{\mathbf{y}}](t) \text{ violates the formula,} \\ [0,1] & \text{otherwise (uncertain result).} \end{cases} \end{aligned} \tag{38}$$

Results The Figure 5 presents the results of the experimentation. The monitoring process is operated in real time at a given rate of 1 Hz. Thus, each second, the next 4 seconds of time horizon are estimated and the obtained tube is used to check the satisfaction of the STL formula. The "Monitor Go_flag" is the result of the satisfaction function. It has been implemented in the ROS[2] framework. The drone is localized using a motion capture system (OptiTrack); the actual trajectory is represented by green dots. The authorized environment is the green frame. Obstacles are in gray. The reference trajectory is drawn in thick red strokes (following the possible modes). Finally, the estimated trajectory is the tube starting at the last known position (end of the green dots) with a small blue box filled in transparent red. This box has been integrated using the dynamic model of the drone, along with the uncertainties, leading to bigger boxes.

The performance benchmark was conducted over the first four seconds of the trajectory ($h = 4$ s). We measured the total execution time on a laptop (Intel i5-8265u, 8gb RAM). It took 0.211 s, making it suitable for online monitoring.

[2]ROS: Robot Operating System (see https://www.ros.org/).

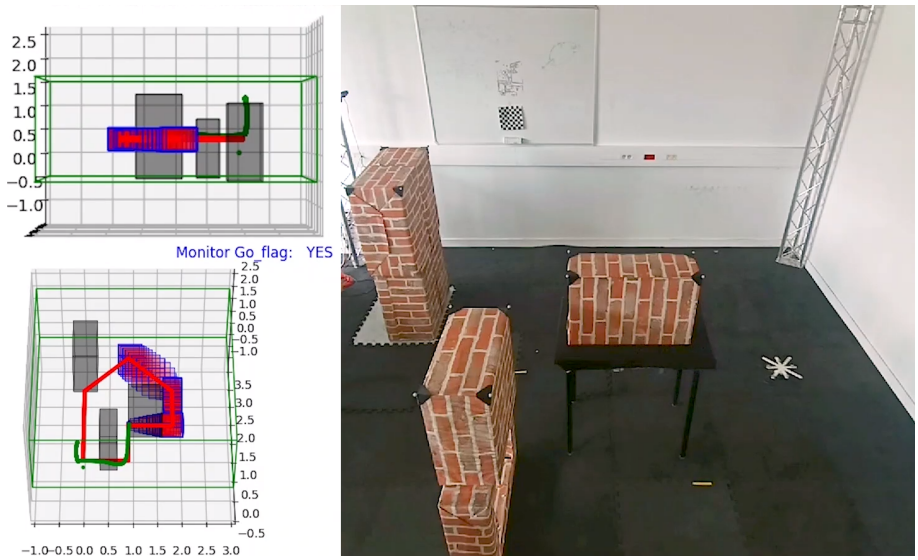


Figure 5: Experimentation results. Left: 2 different views of the computed tube on a given time horizon (red boxes); Right: actual drone flying.

Discussion A simple but realistic STL-based monitor has been tested on a real system for illustration purpose. This experimentation proves the interest of the approach: i) the use of a tube to take into account uncertainties and model simplification provides a guaranteed result; ii) the proposed set-based approach for STL verification is sound and sufficiently fast for real-time application. This example can be generalized, and more complex STL formulas can be tested because no limitations have been imposed.

5 Conclusion

In this paper, literature about dealing with uncertainties has been reviewed in the context of STL. We focused on how to handle tubes instead of single traces, as it allows considering sets of traces from real systems with a strong robustness to uncertainties. Only a few papers have already addressed this issue, and there is still room for improvements, as shown with a simple example whose true result is difficult to obtain with existing approaches. Then, a formalism directly applied on sets is proposed, along with Boolean intervals to rigorously deal with ineluctable uncertain results. This formalism is used as a base for an implementation within the DynIbex tool, in order to have a direct link between STL and a guaranteed integration tool. It is also the base for future works, including dealing with nested formulas and links with reachability analysis.

Acknowledgments

The authors would also like to thank the anonymous referees for their valuable comments and helpful suggestions.

References

[1] Abadi, A., El Amraoui, A., Mekki, H., and Ramdani, N. Guaranteed trajectory tracking control based on interval observer for quadrotors. *International Journal of Control*, 93(11):2743–2759, 2020. DOI: 10.1080/00207179.2019.1610903.

[2] Alexandre Dit Sandretto, J. and Chapoutot, A. Validated explicit and implicit Runge-Kutta methods. *Reliable Computing electronic edition*, 22, 2016. URL: https://hal.science/hal-01243053.

[3] Baird, L., Harapanahalli, A., and Coogan, S. Interval Signal Temporal Logic From Natural Inclusion Functions. *IEEE Control Systems Letters*, 7:3555–3560, 2023. DOI: 10.1109/LCSYS.2023.3337744.

[4] Bartocci, E., Deshmukh, J., Donzé, A., Fainekos, G., Maler, O., Ničković, D., and Sankaranarayanan, S. Specification-Based Monitoring of Cyber-Physical Systems: A Survey on Theory, Tools and Applications. In Bartocci, E. and Falcone, Y., editors, *Lectures on Runtime Verification*, Volume 10457, pages 135–175. Springer International Publishing, Cham, 2018. DOI: 10.1007/978-3-319-75632-5_5.

[5] Deshmukh, J. V., Donzé, A., Ghosh, S., Jin, X., Juniwal, G., and Seshia, S. A. Robust online monitoring of signal temporal logic. *Formal Methods in System Design*, 51(1):5–30, 2017. DOI: 10.1007/s10703-017-0286-7.

[6] Donzé, A. and Maler, O. Robust Satisfaction of Temporal Logic over Real-Valued Signals. In Hutchison, D., Kanade, T., Kittler, J., Kleinberg, J. M., Mattern, F., Mitchell, J. C., Naor, M., Nierstrasz, O., Pandu Rangan, C., Steffen, B., Sudan, M., Terzopoulos, D., Tygar, D., Vardi, M. Y., Weikum, G., Chatterjee, K., and Henzinger, T. A., editors, *Formal Modeling and Analysis of Timed Systems*, Volume 6246, pages 92–106. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010. DOI: 10.1007/978-3-642-15297-9_9.

[7] Dvorak, D. and Kuipers, B. Model-based monitoring of dynamic systems. In *Proceedings of the 11th International Joint Conference on Artificial Intelligence*, Volume 2 of *IJCAI'89*, pages 1238–1243, San Francisco, CA, USA, 1989. Morgan Kaufmann Publishers Inc. URL: https://www.ijcai.org/Proceedings/89-2/Papers/062.pdf.

[8] Fainekos, G. E. and Pappas, G. J. Robustness of temporal logic specifications for continuous-time signals. *Theoretical Computer Science*, 410(42):4262–4291, 2009. DOI: 10.1016/j.tcs.2009.06.021.

[9] Finkbeiner, B., Fränzle, M., Kohn, F., and Kröger, P. A Truly Robust Signal Temporal Logic: Monitoring Safety Properties of Interacting Cyber-Physical Systems under Uncertain Observation. *Algorithms*, 15(4):126, 2022. DOI: 10.3390/a15040126.

[10] Ilyes, R. B., Ho, Q. H., and Lahijanian, M. Stochastic robustness interval for motion planning with Signal Temporal Logic. In *Proceedings of the 2023 IEEE International Conference on Robotics and Automation*, pages 5716–5722, 2023. DOI: 10.1109/ICRA48891.2023.10161409.

[11] Ishii, D., Yonezaki, N., and Goldsztejn, A. Monitoring Temporal Properties using Interval Analysis. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, E99.A(2):442–453, 2016. DOI: 10.1587/transfun.E99.A.442.

[12] Jaulin, L., Kieffer, M., Didrit, O., and Walter, E. Applied Interval Analysis. In Jaulin, L., Kieffer, M., Didrit, O., and Walter, E., editors, *Applied Interval Analysis: With Examples in Parameter and State Estimation, Robust Control and Robotics*, pages 11–43. Springer, London, 2001. DOI: 10.1007/978-1-4471-0249-6_2.

[13] Jha, S., Raman, V., Sadigh, D., and Seshia, S. A. Safe autonomy under perception uncertainty using Chance-Constrained Temporal Logic. *Journal of Automated Reasoning*, 60(1):43–62, 2018. DOI: 10.1007/s10817-017-9413-9.

[14] Kapela, T., Mrozek, M., Wilczak, D., and Zgliczyński, P. CAPD::DynSys: A flexible C++ toolbox for rigorous numerical analysis of dynamical systems. *Communications in Nonlinear Science and Numerical Simulation*, 101:105578, 2021. DOI: 10.1016/j.cnsns.2020.105578.

[15] Koymans, R. Specifying real-time properties with metric temporal logic. *Real-Time Systems*, 2(4):255–299, 1990. DOI: 10.1007/BF01995674.

[16] Kurzhanski, A. B. and Varaiya, P. *Dynamics and Control of Trajectory Tubes: Theory and Computation*, Volume 85 of *Systems & Control: Foundations & Applications*. Springer International Publishing, Cham, 2014. DOI: 10.1007/978-3-319-10277-1.

[17] Largent, S. and Alexandre Dit Sandretto, J. Trajectory monitoring for a drone using interval analysis. In *Proceedings of the Workshop on Planning, Perception and Navigation for Intelligent Vehicles*, Kyoto, Japan, 2023. Philippe Martinet. URL: https://hal.science/hal-04194830.

[18] Lars, L., Lejun, J., Nikolai, M., and J., P. G. Risk of stochastic systems for Temporal Logic Specifications. *ACM Transactions on Embedded Computing Systems*, 2023. DOI: 10.1145/3580490.

[19] Le Coënt, A., Alexandre dit Sandretto, J., Chapoutot, A., and Fribourg, L. An improved algorithm for the control synthesis of nonlinear sampled switched systems. *Formal Methods in System Design*, 53(3):363–383, 2018. DOI: 10.1007/s10703-017-0305-8.

[20] Lee, K. M. B., Yoo, C., and Fitch, R. Signal Temporal Logic synthesis as probabilistic inference. In *Proceedings of the 2021 IEEE International Conference on Robotics and Automation*, pages 5483–5489, 2021. DOI: 10.1109/ICRA48506.2021.9560929.

[21] Lindemann, L., Matni, N., and Pappas, G. J. STL robustness risk over Discrete-Time Stochastic processes. In *Proceedings of the 2021 60th IEEE Conference on Decision and Control*, pages 1329–1335, 2021. DOI: 10.1109/CDC45484.2021.9683305, ISSN: 2576-2370.

[22] Maler, O. and Nickovic, D. Monitoring temporal properties of continuous signals. In Lakhnech, Y. and Yovine, S., editors, *Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems*, Volume 3253 of *Lecture Notes in Computer Science*, pages 152–166, Berlin, Heidelberg, 2004. Springer. DOI: 10.1007/978-3-540-30206-3_12.

[23] Nedialkov, N. S. and Jackson, K. R. ODE software that computes guaranteed bounds on the solution. In Langtangen, H. P., Bruaset, A. M., and Quak, E., editors, *Advances in Software Tools for Scientific Computing*, pages 197–224, Berlin, Heidelberg, 2000. Springer. DOI: 10.1007/978-3-642-57172-5_6.

[24] Pnueli, A. The temporal logic of programs. In *Proceedings of the 18th Annual Symposium on Foundations of Computer Science*, pages 46–57. IEEE, 1977. DOI: 10.1109/SFCS.1977.32.

[25] Roehm, H., Oehlerking, J., Heinz, T., and Althoff, M. STL model checking of continuous and hybrid systems. In Artho, C., Legay, A., and Peled, D., editors, *Automated Technology for Verification and Analysis*, Volume 9938 of *Lecture Notes in Computer Science*, pages 412–427, Cham, 2016. Springer International Publishing. DOI: 10.1007/978-3-319-46520-3_26.

[26] Rohou, S., Jaulin, L., Mihaylova, L., Le Bars, F., and Veres, S. M. Guaranteed computation of robot trajectories. *Robotics and Autonomous Systems*, 93:76–84, 2017. DOI: 10.1016/j.robot.2017.03.020.

[27] Sadigh, D. and Kapoor, A. Safe control under uncertainty with Probabilistic Signal Temporal Logic. In *Robotics: Science and Systems XII*. Robotics: Science and Systems Foundation, 2016. DOI: 10.15607/RSS.2016.XII.017.

[28] Safaoui, S., Lindemann, L., Dimarogonas, D. V., Shames, I., and Summers, T. H. Control design for risk-based Signal Temporal Logic specifications. *IEEE Control Systems Letters*, 4(4):1000–1005, 2020. DOI: 10.1109/LCSYS.2020.2998543.

[29] Yu, X., Dong, W., Li, S., and Yin, X. Model predictive monitoring of dynamical systems for signal temporal logic specifications. *Automatica*, 160:111445, 2024. DOI: 10.1016/j.automatica.2023.111445.

[30] Zhong, B., Jordan, C., and Provost, J. Extending Signal Temporal Logic with quantitative semantics by intervals for robust monitoring of cyber-physical systems. *ACM Transactions on Cyber-Physical Systems*, 5(2):1–25, 2021. DOI: 10.1145/3377868.

Acta Cybernetica **27** (2026) 405–428.

A Muller Approach for Reachability

Luc Jaulin[ab]

Abstract

This paper presents an approach to deal with the prediction of dynamical systems in case of interval uncertainties. These uncertainties can be both on the initial state vector, on the time-dependent inputs and on the evolution function. The approach is based on the Muller theorem often used in an interval context to perform the prediction of cooperative systems. We show here that the Muller approach can be used for general non-cooperative systems. We also show the benefit we can obtain by using a conditioning approach in order to reduce the overestimation.

Keywords: differential inclusion, interval analysis, interval integration, reachability, Muller theorem

1 Introduction

Reachability has been studied by many authors using set-membership tools [4, 6, 9, 11, 18, 28]. Often, the objective of reachability is to predict the future of a dynamical system under uncertainties [27].

The dynamical system we consider has the form $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u})$, where $\mathbf{x}$ is the state vector and $\mathbf{u}$ is the input vector. The input satisfies $\mathbf{u}(\cdot) \in [\mathbf{u}](\cdot)$ which means that, for all t, $\mathbf{u}(t) \in [\mathbf{u}](t)$, where $[\mathbf{u}](t)$ is a box which depends on t. The initial state vector satisfies $\mathbf{x}(0) \in \mathbb{X}_0$. We want to compute the set

$$\mathbb{X}(t) = \left\{ \mathbf{a} \middle| \exists \mathbf{x}(0) \in \mathbb{X}_0, \exists \mathbf{u}(\cdot) \in [\mathbf{u}](\cdot), \mathbf{a} = \varphi_{t,\mathbf{u}(\cdot)}(\mathbf{x}(0)) \right\} \tag{1}$$

where $\varphi_{t,\mathbf{u}(\cdot)}$ is the flow, as illustrated by Figure 1. In this figure, three feasible trajectories are represented. The red is *true* one. The blue trajectory starts from the true $\mathbf{x}(0)$, but deviates from the red due to the uncertainty on $\mathbf{u}$. The black trajectory has the same input but starts from a feasible $\mathbf{x}(0)$ which is not the true one.

From a mathematical point of view, solving a reachability problem amounts to integrating a differential inclusion [5] and requires the use of interval analysis [19]. A general approach to solve this problem has been proposed in [14] but

[a] Robex, Lab-STICC, ENSTA-Bretagne, France
[b] E-mail: lucjaulin@gmail.com, ORCID: 0000-0002-0938-0615

DOI: 10.14232/actacyb.323265

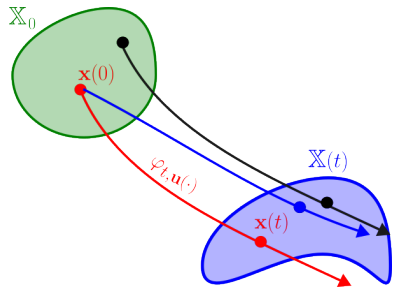


Figure 1: Reach set $\mathbb{X}(t)$ that we want to enclose

the understanding of the approach is not easy and requires the computation of logarithmic norms. Another tool for reachability is CORA [1] with a nice user interface and detailed tutorials. General bounding results for computing reachable sets using differential inequalities can also be found in [26]. See also [7] for an approach based on using mixed monotonicity. A comparison between several tools for reachability can be found in [10].

Probably the first approach to solve this problem is proposed in 1927 by Muller [20]. Due to the fact that interval computation were not know at these times, the use of the Muller's approach was limited to cooperative systems, even if the Muller results are valid for much more general systems. The cooperative property means that the evolution function $\mathbf{f}(\mathbf{x}, \mathbf{u})$ has some monotony properties that are met by only few systems. Since, several people have used the Muller approach, [23, 15, 12] but again, these contributions remain in a context of cooperative systems, even if some change of coordinates are needed to meet cooperativity (see, e.g., [22])).

The contributions of this paper are the following:

- Clearly underline the cooperativity property is not required at all, when using a Muller approach.
- Show a way to transform the reachability problem into a simple ordinary differential equations (ODE).
- Show that conditioning techniques, classically used in the interval algorithms, could reduce drastically the pessimism, in case of small uncertainties.
- Provide an enclosure which is asymptotically minimal [13], *i.e.*, which is minimal when the uncertainties are infinitesimal.

The paper is organized as follows. Section 2 introduces *interval dynamical systems* (IDS) which is a class of differential inclusions that can be integrated as an ODE. Section 3 shows how interval computation combined with the Muller can provide an enclosure of the reachability problem. Section 4 defines conditioners, combines them with a Muller enclosure method and gives some theoretical results on the asymptotic minimality of conditioned IDS. Section 5 gives an application in robotics. Section 6 concludes the paper.

2 Interval dynamical system

In this section, we give the definition of interval dynamical systems, show how they can be used to represent a dynamical system with interval uncertainties and explain how they can be simulated using the Muller theorem.

2.1 Principle

An *interval* $[x] = [x^-, x^+]$ is a connected closed subset of $\mathbb{R}$. The *lower bound* of $[x]$ is $\text{lb}([x]) = x^-$ and its *upper bound* is $\text{ub}([x]) = x^+$. A *box* of $\mathbb{R}^n$ is the Cartesian product of n intervals:

$$[\mathbf{x}] = [x_1^-, x_1^+] \times \cdots \times [x_n^-, x_n^+]. \tag{2}$$

Denote by $\mathbb{IR}$ the set of intervals and by $\mathbb{IR}^n$ the set of all boxes of $\mathbb{R}^n$. The width of a box $[\mathbf{x}]$ is denoted by $w([\mathbf{x}])$.

Definition 2.1. *As illustrated by Figure 2, an* interval dynamical system *(IDS) is a system of the form*

$$\begin{array}{rcl} \dot{\mathbf{z}}(t) & = & \mathbf{f}_z(\mathbf{z}(t), \mathbf{u}^-(t), \mathbf{u}^+(t), t) \\ [\mathbf{x}](t) & = & \mathbf{h}(\mathbf{z}(t), t) \end{array} \tag{3}$$

where $\mathbf{h} : \mathbb{R}^q \times \mathbb{R} \mapsto \mathbb{IR}^n$ *is a continuous one to one function. The vector* $\mathbf{z} \in \mathbb{R}^q$ *is called the* epistemic state vector*. It is qualified as* epistemic*, since, as we will see later, it represents the knowledge we have of the real state* $\mathbf{x}$ *of a dynamical system. The evolution function for* $\mathbf{z}$*, denoted by* $\mathbf{f}_z$*, is assumed to be continuous and locally Lipschitz in* $\mathbf{x}$*. It should not be confused with the evolution function* $\mathbf{f}$ *for* $\mathbf{x}$ *introduced in the previous section. The quantities* $[\mathbf{u}](t) = [\mathbf{u}^-(t), \mathbf{u}^+(t)]$ *and* $[\mathbf{x}](t) = [\mathbf{x}^-(t), \mathbf{x}^+(t)]$ *are time dependent boxes or* tubes *[17, 25]. The IDS should satisfy the* thinness *property*

$$\left.\begin{array}{l} w([\mathbf{x}](0)) = 0 \\ \forall t, w([\mathbf{u}](t)) = 0 \end{array}\right\} \Rightarrow \forall t, w([\mathbf{x}](t)) = 0 \tag{4}$$

which means that if both the initial state $\mathbf{x}(0)$ *and the input* $\mathbf{u}(0)$ *are perfectly known, the IDS does not produce any uncertainty.*

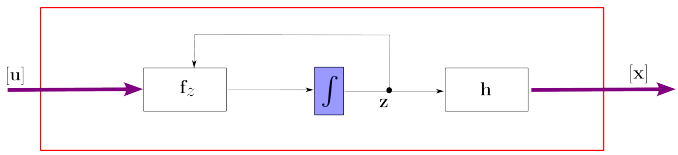


Figure 2: Interval dynamical system (IDS). Thick arrows are interval valued

Definition 2.2. *Consider a dynamical system*

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}) \tag{5}$$

where $\mathbf{f}$ *is continuous and differentiable in* $\mathbf{x}, \mathbf{u}$*, locally Lipschitz in* $\boldsymbol{x}$ *and where* $\mathbf{u}$ *is piecewise continuous, so that a single solution exists for any initial state vector. The IDS (3) is said to be* an enclosing IDS *of (5) if we have*

$$\left.\begin{array}{c} \mathbf{x}(0) \in [\mathbf{x}](0) \\ \forall t, \mathbf{u}(t) \in [\mathbf{u}^-(t), \mathbf{u}^+(t)] \end{array}\right\} \Rightarrow \forall t, \mathbf{x}(t) \in [\mathbf{x}](t). \tag{6}$$

Note that the condition $\mathbf{x}(0) \in [\mathbf{x}](0)$ implies that $\mathbf{h}(\mathbf{z}(0)) \in [\mathbf{x}](0)$.

Example 2.1. The IDS

$$\begin{array}{rcl} \dot{z}_1 & = & z_1 + u^- \\ \dot{z}_2 & = & z_2 + u^+ \\ [x] & = & h(\mathbf{z}) = [z_1, z_2] \end{array} \tag{7}$$

encloses the system $\dot{x} = x + u$ for $u \in [u] = [u^-, u^+]$.

2.2 Muller IDS

In the literature, IDS are generally given in a more specific case which we call here the Muller form [20].

Definition 2.3. *An IDS is of* Muller type *if it has the form*

$$\left\{\begin{array}{rcl} \underbrace{\begin{pmatrix} \dot{\mathbf{x}}^-(t) \\ \dot{\mathbf{x}}^+(t) \end{pmatrix}}_{\dot{\mathbf{z}}(t)} & = & \underbrace{\begin{pmatrix} \mathbf{f}^-(\mathbf{x}^-(t), \mathbf{x}^+(t), \mathbf{u}^-(t), \mathbf{u}^+(t)) \\ \mathbf{f}^+(\mathbf{x}^-(t), \mathbf{x}^+(t), \mathbf{u}^-(t), \mathbf{u}^+(t)) \end{pmatrix}}_{\mathbf{f}_z(\mathbf{z}(t), \mathbf{u}^-(t), \mathbf{u}^+(t))} \\ [\mathbf{x}](t) & = & \underbrace{[\mathbf{x}^-(t), \mathbf{x}^+(t)]}_{\mathbf{h}(\mathbf{z}(t))} \end{array}\right. . \tag{8}$$

The thinness *property becomes*

$$\mathbf{f}^-(\mathbf{x}, \mathbf{x}, \mathbf{u}, \mathbf{u}) = \mathbf{f}^+(\mathbf{x}, \mathbf{x}, \mathbf{u}, \mathbf{u}). \tag{9}$$

IDS generate interval trajectories $[\mathbf{x}^-(t), \mathbf{x}^+(t)]$ (represented by the epistemic state vector $\mathbf{z}$) that are supposed to enclose the trajectory of an uncertain dynamical system. It corresponds to an ordinary differential equation (ODE) which can be integrated using a Runge-Kutta method or using an interval integration [14, 21].

The enclosure property (see Definition 2.2) is illustrated for Muller IDS by Figure 3. The *true* trajectory (blue) $\mathbf{x}(t)$ is between the two red trajectories $\mathbf{x}^-(t)$ and $\mathbf{x}^+(t)$, or equivalently,

$$\forall t, \mathbf{x}(t) \in [\mathbf{x}^-(t), \mathbf{x}^+(t)] = [\mathbf{x}](t). \tag{10}$$

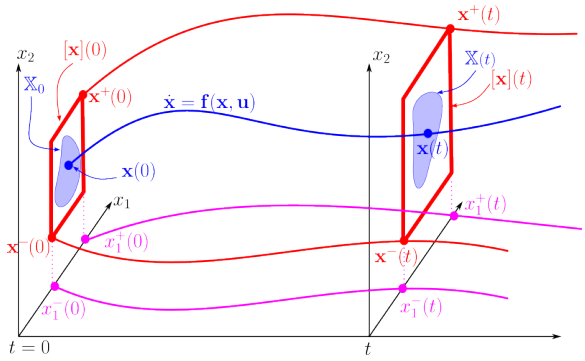


Figure 3: A Muller IDS enclosing a dynamical system

Proposition 2.1. *Consider a Muller IDS (see Equation 8) which satisfies*

$$\frac{\partial f_i^-}{\partial x_i^+} = 0, \quad \frac{\partial f_i^+}{\partial x_i^-} = 0 \tag{11}$$

$$\frac{\partial f_i^-}{\partial x_j^-} \geq 0, \quad \frac{\partial f_i^-}{\partial x_j^+} \leq 0, \quad \frac{\partial f_i^+}{\partial x_j^-} \leq 0, \quad \frac{\partial f_i^+}{\partial x_j^+} \geq 0, \quad \textit{for } i \neq j \tag{12}$$

and

$$\frac{\partial f_i^-}{\partial u_\ell^-} \geq 0, \quad \frac{\partial f_i^-}{\partial u_\ell^+} \leq 0, \quad \frac{\partial f_i^+}{\partial u_\ell^-} \leq 0, \quad \frac{\partial f_i^+}{\partial u_\ell^+} \geq 0 \tag{13}$$

as soon as $\mathbf{x}^- \leq \mathbf{x}^+$ *and* $\mathbf{u}^- \leq \mathbf{u}^+$. *The IDS encloses the system* $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u})$ *with*

$$\mathbf{f}(\mathbf{x}, \mathbf{u}) = \mathbf{f}^-(\mathbf{x}, \mathbf{x}, \mathbf{u}, \mathbf{u}) \quad = \quad \mathbf{f}^+(\mathbf{x}, \mathbf{x}, \mathbf{u}, \mathbf{u}). \tag{14}$$

Note that these conditions on $\mathbf{f}$ are close to the notion of *mixed monotonicity*, introduced in [7]. Before giving the proof, let us give a simple illustrative example.

Example 2.2. An enclosing IDS for the dynamical system

$$\begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \end{pmatrix} = \begin{pmatrix} x_1 x_2 \\ x_1 + u \end{pmatrix} \tag{15}$$

is

$$\begin{pmatrix} \dot{x}_1^- \\ \dot{x}_1^+ \\ \dot{x}_2^- \\ \dot{x}_2^+ \end{pmatrix} = \begin{pmatrix} \min(x_1^- x_2^-, x_1^- x_2^+) \\ \max(x_1^+ x_2^-, x_1^+ x_2^+) \\ x_1^- + u^- \\ x_1^+ + u^+ \end{pmatrix}. \tag{16}$$

Indeed,

$$\mathbf{f}^-(\mathbf{x}, \mathbf{x}, \mathbf{u}, \mathbf{u}) = \mathbf{f}^+(\mathbf{x}, \mathbf{x}, \mathbf{u}, \mathbf{u}) = \mathbf{f}(\mathbf{x}, \mathbf{u}) = \begin{pmatrix} x_1 x_2 \\ x_1 + u \end{pmatrix}. \tag{17}$$

Moreover, we have

$$\left\{\begin{array}{lll}
\frac{\partial f_1^-}{\partial x_2^-} = \max(x_1^-,0) & \geq 0 \\
\frac{\partial f_1^-}{\partial x_2^+} = \min(x_1^-,0) & \leq 0 \\
\frac{\partial f_1^+}{\partial x_2^-} = \min(x_1^+,0) & \leq 0 \\
\frac{\partial f_1^+}{\partial x_2^+} = \max(x_1^+,0) & \geq 0 \\
\frac{\partial f_2^-}{\partial x_1^-} = 1 & \geq 0 \\
\frac{\partial f_2^-}{\partial x_1^+} = 0 & \leq 0 \\
\frac{\partial f_2^+}{\partial x_1^-} = 0 & \leq 0 \\
\frac{\partial f_2^+}{\partial x_1^+} = 1 & \geq 0
\end{array}\right. \text{ and } \left\{\begin{array}{lll}
\frac{\partial f_1^-}{\partial u^-} = 0 & \geq & 0 \\
\frac{\partial f_1^-}{\partial u^+} = 0 & \leq & 0 \\
\frac{\partial f_1^+}{\partial u^-} = 0 & \geq & 0 \\
\frac{\partial f_1^+}{\partial u^+} = 0 & \leq & 0 \\
\frac{\partial f_2^-}{\partial u^-} = 1 & \geq & 0 \\
\frac{\partial f_2^-}{\partial u^+} = 0 & \leq & 0 \\
\frac{\partial f_2^+}{\partial u^-} = 0 & \leq & 0 \\
\frac{\partial f_2^+}{\partial u^+} = 1 & \geq & 0
\end{array}\right. . \tag{18}$$

Proof. (of Proposition 2.1). The proof is by induction. First, by assumption (see (2)), we have $\mathbf{x}(0) \in [\mathbf{x}^-(0), \mathbf{x}^+(0)]$. Assume that the proposition is true for t. Let us prove that it is also true for $t + dt$, where dt is infinitesimal. Thus, we have to prove that

$$\mathbf{x}(t+dt) \in [\mathbf{x}^-(t+dt), \mathbf{x}^+(t+dt)] \tag{19}$$

i.e., for all i,

$$x_i^-(t+dt) \leq x_i(t+dt) \leq x_i^+(t+dt). \tag{20}$$

or equivalently

$$\begin{array}{lll}
(i) & x_i^+(t+dt) - x_i(t+dt) & \geq 0 \\
(ii) & x_i^-(t+dt) - x_i(t+dt) & \leq 0.
\end{array} \tag{21}$$

We just check (i) for $i = 1$, *i.e.*, we want to prove that

$$x_1^+(t+dt) - x_1(t+dt) \geq 0. \tag{22}$$

The same reasoning can be done to prove (ii) and for $i > 1$.

Since $x_1^+(t) - x_1(t) \geq 0$, we have to consider two cases

• Case 1. $x_1^+(t) - x_1(t) > 0$. By continuity of the function $e_1^+(t) = x_1^+(t) - x_1(t)$ with respect to t, we deduce that $e_1^+(t+dt) = x_1^+(t+dt) - x_1(t+dt) > 0$, since dt is infinitesimal.

• Case 2. $x_1^+(t) - x_1(t) = 0$. In this case, $x_1^+(t+dt) - x_1(t+dt) \geq 0$ if

$$f_1^+(\mathbf{x}^-(t), \mathbf{x}^+(t), \mathbf{u}^-(t), \mathbf{u}^+(t)) - f_1(\mathbf{x}(t), \mathbf{u}(t)) \geq 0 \tag{23}$$

This is what we will now prove. From (9), we have

$$f_1^+\left(\begin{array}{c} x_1, x_2, \ldots, x_n \\ x_1, x_2, \ldots, x_n \\ u_1, u_2, \ldots, u_m \\ u_1, u_2, \ldots, u_m \end{array}\right) - f_1\left(\begin{array}{c} x_1, x_2, \ldots, x_n \\ u_1, u_2, \ldots, u_m \end{array}\right) = 0. \tag{24}$$

Moreover, since

$$\begin{cases} \frac{\partial f_1^+}{\partial x_j^-} \leq 0, & \frac{\partial f_1^+}{\partial x_j^+} \geq 0 & \text{for } j \neq 1 & \text{(see (12))} \\ \frac{\partial f_1^+}{\partial u_\ell^-} \leq 0, & \frac{\partial f_1^+}{\partial u_\ell^+} \geq 0 & \forall \ell & \text{(see (13))} \\ x_j^- \leq x_j \leq x_j^+, & & \forall j & \\ u_\ell^- \leq u_\ell \leq u_\ell^+, & & \forall \ell & \end{cases}$$

we get

$$f_1^+ \begin{pmatrix} x_1, x_2^-, \ldots, x_n^- \\ x_1, x_2^+, \ldots, x_n^+ \\ u_1^-, u_2^-, \ldots, u_m^- \\ u_1^+, u_2^+, \ldots, u_m^+ \end{pmatrix} - f_1 \begin{pmatrix} x_1, x_2, \ldots, x_n \\ u_1, u_2, \ldots, u_m \end{pmatrix} \geq 0. \tag{25}$$

Since $x_1^+ = x_1$, and since f_1^+ does not depend on x_1^- ($\partial f_1^+ / \partial x_1^- = 0$, see (11)), we get Inequality (23). □

3 Interval analysis to get an enclosing interval dynamical system

In this section, we show how we can use interval computation to get a Muller dynamical system which encloses a given dynamical system.

3.1 Inclusion function

An *inclusion function* $[\mathbf{f}]$ for $\mathbf{f} : \mathbb{R}^n \mapsto \mathbb{R}^m$, is a function

$$[\mathbf{f}] : \begin{array}{ccc} \mathbb{IR}^n & \rightarrow & \mathbb{IR}^m \\ [\mathbf{x}] & \rightarrow & [\mathbf{f}]([\mathbf{x}]) \end{array} \tag{26}$$

such that [19]

$$\mathbf{f}([\mathbf{x}]) \subset [\mathbf{f}]([\mathbf{x}]). \tag{27}$$

An inclusion function $[\mathbf{f}]$ is *thin* if for any singleton $[\mathbf{x}] = \{\mathbf{x}\}$, $[\mathbf{f}]([\mathbf{x}])$ is also a singleton. It is *inclusion monotonic* if

$$[\mathbf{x}] \subset [\mathbf{y}] \Rightarrow [\mathbf{f}]([\mathbf{x}]) \subset [\mathbf{f}]([\mathbf{y}]). \tag{28}$$

3.2 Epistemic transform

We present here the epistemic transform which will be used to reformulate a differential inclusion into an ODE. The main idea is to explore the boundary of $[\mathbf{x}]$ to compute a box enclosing $\mathbf{f}([\mathbf{x}])$. This makes sense when the function is known to be one to one in $[\mathbf{x}]$. The idea of using the boundary has been explored by several authors in the context of dynamical systems (see, *e.g.*, [29]).

Definition 3.1. *The* epistemic transform *of the interval function*

$$[\mathbf{f}]: \begin{array}{ccc} \mathbb{IR}^n & \to & \mathbb{IR}^m \\ [\mathbf{x}] & \to & [\mathbf{f}]([\mathbf{x}]) \end{array} \tag{29}$$

is given by

$$\mathbf{\Gamma}([\mathbf{f}]): \begin{pmatrix} x_1^- \\ x_1^+ \\ x_2^- \\ x_2^+ \\ \vdots \\ x_n^- \\ x_n^+ \end{pmatrix} \mapsto \begin{pmatrix} lb(& [f_1](x_1^-,[x_2],[x_3],\ldots,[x_n]) &) \\ ub(& [f_1](x_1^+,[x_2],[x_3],\ldots,[x_n]) &) \\ lb(& [f_2]([x_1],x_2^-,[x_3],\ldots,[x_n]) &) \\ ub(& [f_2]([x_1],x_2^+,[x_3],\ldots,[x_n]) &) \\ & \vdots & \\ lb(& [f_m]([x_1],[x_2],[x_3],\ldots,x_n^-) &) \\ ub(& [f_m]([x_1],[x_2],[x_3],\ldots,x_n^+) &) \end{pmatrix}. \tag{30}$$

We will write

$$\widehat{[\mathbf{f}]} = \mathbf{\Gamma}([\mathbf{f}]). \tag{31}$$

As defined above, the epistemic transform is a functional operator which transforms an interval function $[\mathbf{f}]: \mathbb{IR}^n \mapsto \mathbb{IR}^m$ into another function $\widehat{[\mathbf{f}]}: \mathbb{R}^{2n} \mapsto \mathbb{R}^{2n}$.

Example 3.1. The epistemic transform of

$$[\mathbf{f}]([\mathbf{x}]) = \begin{pmatrix} [x_1]\cdot[x_2] \\ \exp([x_1]-\sin[x_2]) \end{pmatrix} \tag{32}$$

is

$$\widehat{[\mathbf{f}]} \begin{pmatrix} x_1^- \\ x_1^+ \\ x_2^- \\ x_2^+ \end{pmatrix} = \begin{pmatrix} \min(x_1^- x_2^-, x_1^- x_2^+) \\ \max(x_1^+ x_2^-, x_1^+ x_2^+) \\ \exp\left(x_1^- - \sin x_2^-\right) \\ \exp\left(x_1^+ - \sin x_2^+\right) \end{pmatrix}. \tag{33}$$

Definition 3.2. *Define the* epistemic canonical injection *as the function*

$$\boldsymbol{\gamma}: \begin{array}{ccc} \mathbb{IR}^n & \to & \mathbb{R}^{2n} \\ [\mathbf{x}] & \to & \left(x_1^-, x_1^+ \quad ,\ldots, x_n^-, x_n^+ \right)^T \end{array}. \tag{34}$$

We will write

$$\widehat{[\mathbf{x}]} = \boldsymbol{\gamma}([\mathbf{x}]). \tag{35}$$

which means that the hat *operator transforms a box of* $\mathbb{R}^n$ *into an equivalent* $2n$*-vector. The epistemic canonical injection* $\boldsymbol{\gamma}$ *generates a vector containing the bounds of its interval inputs. This function has a unique reciprocal* $\boldsymbol{\gamma}^{-1}$ *for which*

$$dom(\boldsymbol{\gamma}^{-1}) = \left\{ \left(x_1^-, x_1^+ \quad ,\ldots, x_n^-, x_n^+ \right) \ | \forall i, x_i^- \le x_i^- \right\} \tag{36}$$

i.e., $\boldsymbol{\gamma}^{-1} \circ \boldsymbol{\gamma}([\mathbf{x}]) = [\mathbf{x}]$, *for all* $[\mathbf{x}] \in \mathbb{IR}^n$. *Note that*

$$\boldsymbol{\gamma}^{-1} \circ \widehat{[\mathbf{f}]} \circ \boldsymbol{\gamma}([\mathbf{x}]) \subset [\mathbf{f}]([\mathbf{x}]). \tag{37}$$

The reason for this inclusion is that $\boldsymbol{\gamma}^{-1} \circ \widehat{[\mathbf{f}]} \circ \boldsymbol{\gamma}$ *evaluates the box* $[\mathbf{x}]$ *its boundary (via each of its* $2n$ *faces) whereas* $[\mathbf{f}]$ *evaluates* $[\mathbf{x}]$ *in its boundary and its interior. As a consequence, there may exist a* $\mathbf{x} \in [\mathbf{x}]$ *such that* $\mathbf{f}(\mathbf{x})$ *is not inside* $\boldsymbol{\gamma}^{-1} \circ \widehat{[\mathbf{f}]} \circ \boldsymbol{\gamma}([\mathbf{x}])$*. It may happen if for one* i*, the function* f_i *has a local extremum inside the interior of* $[\mathbf{x}]$*.*

3.3 Enclosing IDS

The following proposition shows that from a general differential inclusion $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u})$,$\mathbf{u} \in [\mathbf{u}]$, we can derive an enclosing IDS.

Proposition 3.1. *An enclosing IDS for*

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}, t) \tag{38}$$

is

$$\begin{pmatrix} \dot{x}_1^- \\ \dot{x}_1^+ \\ \dot{x}_2^- \\ \dot{x}_2^+ \\ \vdots \\ \dot{x}_n^- \\ \dot{x}_n^+ \end{pmatrix} = \begin{pmatrix} lb(& [f_1](x_1^-, [x_2], [x_3], \ldots, [x_n], [\mathbf{u}], t) &) \\ ub(& [f_1](x_1^+, [x_2], [x_3], \ldots, [x_n], [\mathbf{u}], t) &) \\ lb(& [f_2]([x_1], x_2^-, [x_3], \ldots, [x_n], [\mathbf{u}], t) &) \\ ub(& [f_2]([x_1], x_2^+, [x_3], \ldots, [x_n], [\mathbf{u}], t) &) \\ & \vdots & \\ lb(& [f_n]([x_1], [x_2], [x_3], \ldots, x_n^-, [\mathbf{u}], t) &) \\ ub(& [f_n]([x_1], [x_2], [x_3], \ldots, x_n^+, [\mathbf{u}], t) &) \end{pmatrix} \tag{39}$$

or equivalently

$$\begin{array}{rcl} \dot{\mathbf{z}} & = & \widehat{[\mathbf{f}]}(\mathbf{z}, [\mathbf{u}], t) \\ \mathbf{z} & = & \boldsymbol{\gamma}([\mathbf{x}]) \end{array} \tag{40}$$

where $[f_i]$ *are thin inclusion functions for the* $f_i's$ *that are assumed to be inclusion monotonic. The box* $[\mathbf{u}]$ *and time* t *are seen as parameters and are not involved in the epistemic transform.*

Figure 4 illustrates the equation $\dot{x}_1^- = \mathrm{lb}([f_1](x_1^-, [x_2], \mathbf{u}))$ in a two dimensional state space. We see that the evolution of the left face of box $[\mathbf{x}]$ only depends on the vector field $\mathbf{f}(\mathbf{x}, \mathbf{u})$ on the left face $(x_1^-, [x_2])$. More than this, the evolution depends only on the first component $f_1(\mathbf{x}, \mathbf{u})$ of $\mathbf{f}(\mathbf{x}, \mathbf{u})$. We take the lower bound to be sure that the left face will not go faster than any feasible trajectory.

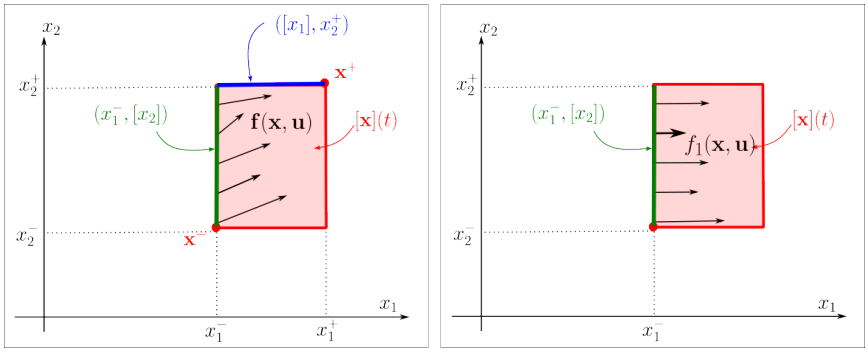


Figure 4: The left face should not go faster than the bold arrow

Proof. To prove the proposition, we apply Proposition 2.1. First, in (39), since for all i, $\dot{x}_i^-$ do not depend on x_i^+ and $\dot{x}_i^+$ do not depend on x_i^-, Condition 11 is satisfied. To prove (12) and (13), consider first an interval function $[f] = [f^-, f^+]$ which is inclusion monotonic, we have

$$\begin{array}{lll} [a,b^-] \subset [a,b^+] & \Rightarrow & [f]([a,b^-]) \subset [f]([a,b^+]) \\ & \Leftrightarrow & [f^-([a,b^-]), f^+([a,b^-])] \subset [f^-([a,b^+]), f^+([a,b^+])] \\ & \Leftrightarrow & \left\{ \begin{array}{lll} f^-([a,b^-]) & \geq & f^-([a,b^+]) \\ f^+([a,b^-]) & \leq & f^+([a,b^+]) \end{array} \right. \end{array}$$

which means that f^- is decreasing with respect to the upper bound b of its interval argument and f^+ is increasing with respect to this bound. Equivalently, we get that f^+ is increasing with respect to the lower bound a of its interval argument and f^+ is decreasing with respect to a. If we apply this reasoning to $[f_i]$ which is inclusion monotonic we get (12) and (13). Property 5 comes from the fact that the inclusion function is thin. □

3.4 Example: the pendulum

An enclosing IDS for the pendulum

$$\begin{array}{lll} \dot{x}_1 & = & x_2 \\ \dot{x}_2 & = & -\sin x_1 - x_2 + u \end{array} \tag{41}$$

is

$$\begin{array}{lll} \dot{\mathbf{z}} & = & \widehat{[\mathbf{f}]}(\mathbf{z}, [u]) \\ \mathbf{z} & = & \boldsymbol{\gamma}([\mathbf{x}]) \end{array} \tag{42}$$

where

$$\begin{array}{rcl} \mathbf{f}(\mathbf{x},u) & = & \begin{pmatrix} x_2 \\ -\sin x_1 - x_2 + u \end{pmatrix} \\ \widehat{[\mathbf{f}]}(\mathbf{z},[u]) & = & \underbrace{\begin{pmatrix} z_3 \\ z_4 \\ \mathrm{lb}(-\sin([z_1,z_2])) - z_3 + u^- \\ \mathrm{ub}(-\sin([z_1,z_2])) - z_4 + u^+ \end{pmatrix}}_{f_z(z,[u])} \\ \mathbf{z} & = & \begin{pmatrix} x_1^- & x_1^+ & x_2^- & x_2^+ \end{pmatrix}^{\mathrm{T}} \\ [\mathbf{x}] & = & [z_1,z_2] \times [z_3,z_4]. \end{array} \tag{43}$$

We simulate our IDS for $t \in [0,6]$, a sampling time $dt = 0.001$ and an initial state box $[\mathbf{x}](0) = [1-\varepsilon, 1+\varepsilon], \times [0.2-\varepsilon, 0.2+\varepsilon]$ where $\varepsilon = 0.01$ is a small positive number representing some uncertainties. For the input box, we have taken $[u] = [-\varepsilon^2, \varepsilon^2]$. The simulation yields Figures 5 and 6. Even if the pendulum is a stable system, we observe an instability of the IDS, where the all bounds $x_1^-, x_1^+, x_2^-, x_2^+$ diverge.

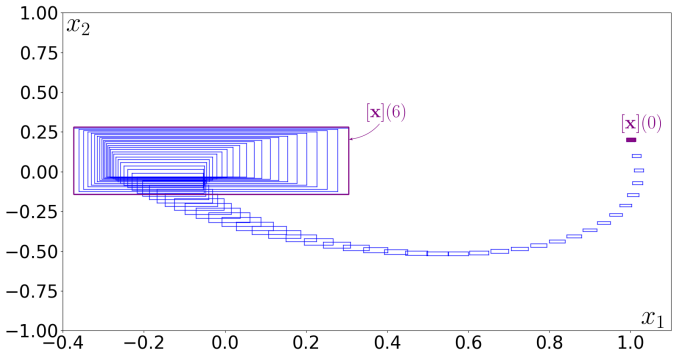


Figure 5: Boxes $[\mathbf{x}](t)$ generated by the Muller IDS

4 Conditioner

In this section, we propose a method to get an accurate IDS, that we call *asymptotically minimal* (see [13] for more details concerning the asymptotic minimality). More precisely, if the width $w([\mathbf{x}](0))$ of the initial box $[\mathbf{x}](0)$ is in $O(\varepsilon)$ (where ε is a tiny positive number), if the width of the input box is small compared to ε, *i.e.*, $w([\mathbf{u}]) = o(\varepsilon)$, then we will have an overestimation introduced by the IDS which is small compared to ε, *i.e.,* in $o(\varepsilon)$.

A conditioner can be seen as a change of coordinates moving along a reference trajectory. The principle is classical when we want to use linear enclosure methods to reduce the conservatism [14, 16, 8].

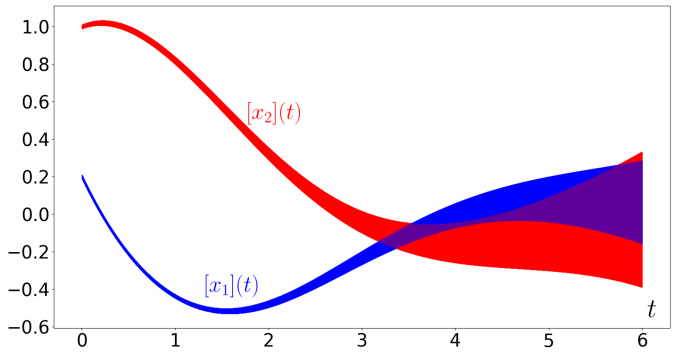


Figure 6: Tubes $[x_1](t)$ and $[x_2](t)$ generated by the Muller IDS

4.1 Principle

Consider again the dynamical system

$$\begin{array}{c} \dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}) \\ \mathbf{x}(0) \in [\mathbf{x}](0) \\ \mathbf{u}(t) \in [\mathbf{u}](t) \end{array} \tag{44}$$

for which we want to compute get an IDS.

We propose to add a *conditioner* as illustrated by Figure 7. Here, conditioner has to be understood in its generic form, *i.e.*, something that improves the enclosure, making it more accurate.

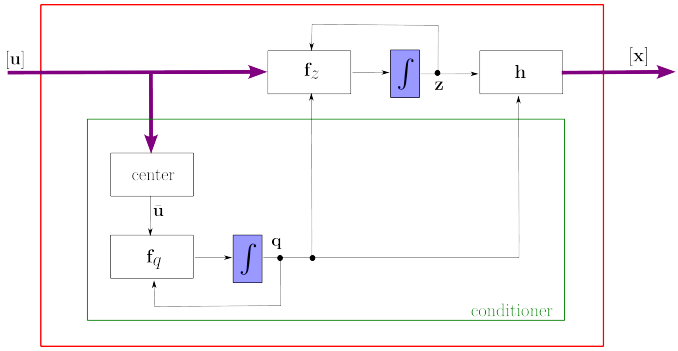


Figure 7: The conditioner improves the accuracy of the interval integration

As a results, we get

$$\begin{array}{rcl} \dot{\mathbf{z}} & = & \mathbf{f}_z(\mathbf{z}, \mathbf{q}, [\mathbf{u}]) \\ \dot{\mathbf{q}} & = & \mathbf{f}_q(\mathbf{q}, \text{center}([\mathbf{u}])) \\ [\mathbf{x}] & = & \mathbf{h}(\mathbf{z}, \mathbf{q}) \end{array} \tag{45}$$

i.e.,

$$\begin{array}{rcl} \dot{\mathbf{z}} & = & \mathbf{f}_z(\mathbf{z}, [\mathbf{u}], t) \\ [\mathbf{x}] & = & \mathbf{h}(\mathbf{z}, t) \end{array} \tag{46}$$

where $\mathbf{f}_z(\mathbf{z}, [\mathbf{u}], t)$ and $\mathbf{h}(\mathbf{z}, t)$ correspond to $\mathbf{f}_z(\mathbf{z}, \mathbf{q}(t), [\mathbf{u}])$ and $\mathbf{h}(\mathbf{z}, \mathbf{q}(t))$, respectively. This is clearly an IDS which can be integrated as an ODE, as it will be shown now.

4.2 IDS with the conditioner

To find the right conditioner, we compute one trajectory $\mathbf{a}(t)$ which corresponds to an approximation of the feasible trajectories. For instance, we can chose $\mathbf{a}(t)$ such that

$$\begin{array}{c} \dot{\mathbf{a}} = \mathbf{f}(\mathbf{a}, \bar{\mathbf{u}}) \\ \mathbf{a}(0) = \text{center}([\mathbf{x}](0)) \\ \bar{\mathbf{u}}(t) = \text{center}([\mathbf{u}](t)). \end{array} \tag{47}$$

Then we will make a change of coordinates at the neighborhood of the trajectory $\mathbf{a}(t)$.

The following theorem corresponds to the main contribution of the paper. It shows that for a large class of differential inclusion $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u})$,$\mathbf{u} \in [\mathbf{u}]$, we can generate an enclosing IDS with some nice asymptotic properties. The main idea of this theorem is to define an error $\mathbf{e}$ between $\mathbf{x}$ and $\mathbf{a}$ in the new local frame. Then we build an IDS for the evolution of the error $\mathbf{e}$ introducing the epistemic state vector $\mathbf{z}$ associated with the box $[\mathbf{e}]$. This will allow us to build a parallelepiped $\langle \mathbf{x} \rangle(t)$ which encloses the trajectory $\mathbf{x}(t)$.

Theorem 4.1. *An enclosing IDS for the system* $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u})$ *is (see Figure 8)*

$$\begin{array}{rcl} \dot{\mathbf{J}} & = & \left(\frac{\partial \mathbf{f}}{\partial \mathbf{x}}(\mathbf{a}, \bar{\mathbf{u}})\right) \cdot \mathbf{J} \\ \dot{\mathbf{a}} & = & \mathbf{f}(\mathbf{a}, \bar{\mathbf{u}}) \\ \dot{\mathbf{z}} & = & \widehat{[\mathbf{f}_e]}(\mathbf{z}, [\mathbf{u}]) \\ [\mathbf{e}] & = & \boldsymbol{\gamma}(\mathbf{z}) \\ [\mathbf{x}] & = & \mathbf{J} \cdot [\mathbf{e}] + \mathbf{a} \end{array} \tag{48}$$

where

$$\begin{array}{rcl} \widehat{[\mathbf{f}_e]} & = & \boldsymbol{\Gamma}([\mathbf{f}_e]) \\ [\mathbf{f}_e]([\mathbf{e}]) & = & \mathbf{f}_e(\bar{\mathbf{e}}) + \mathbf{J}_e \cdot ([\mathbf{e}] - \bar{\mathbf{e}}) \\ \mathbf{f}_e(\mathbf{e}) & = & \mathbf{J}^{-1}\left(-\dot{\mathbf{J}}\mathbf{e} + \mathbf{f}(\mathbf{J}\mathbf{e} + \mathbf{a}, [\mathbf{u}]) - \dot{\mathbf{a}}\right) \\ \mathbf{J}_e([\mathbf{e}]) & = & -\mathbf{J}^{-1}\dot{\mathbf{J}} + \left(\frac{\partial \mathbf{f}}{\partial \mathbf{x}}(\mathbf{J} \cdot [\mathbf{e}] + \mathbf{a}, [\mathbf{u}])\right) \cdot \mathbf{J} \\ \bar{\mathbf{e}} & = & \textit{center}([\mathbf{e}]). \end{array} \tag{49}$$

For the initialization, we take $\mathbf{J}(0) = \mathbf{I}$, $\mathbf{a}(0) = center([\mathbf{x}](0))$ *and* $[\mathbf{e}](0) = [\mathbf{x}](0) - \mathbf{a}(0)$.

Moreover if $w([\mathbf{x}](0)) = O(\varepsilon)$ *and* $w([\mathbf{u}](t)) = o(\varepsilon)$, *where* ε *is infinitesimal, then, for a given* t,

(i) The parallelepiped $\langle \mathbf{x} \rangle(t) = \{\mathbf{x} | \exists \mathbf{e} \in [\mathbf{e}](t), \mathbf{x} = \mathbf{J}(t) \cdot \mathbf{e} + \mathbf{a}(t)\}$ *is such that the Hausdorff distance* $H([\mathbf{x}](t)], \mathbb{X}(t))$ *between the box* $[\mathbf{x}](t)]$ *and the reach set*

$$\mathbb{X}(t) = \left\{\mathbf{a} \middle| \exists \mathbf{x}(0) \in [\mathbf{x}](0), \exists \mathbf{u}(\cdot) \in [\mathbf{u}],\ \mathbf{a} = \varphi_{t,\mathbf{u}(\cdot)}(\mathbf{x}(0))\right\} \tag{50}$$

at time t *is* $o(\varepsilon)$.

(ii) The box $[\mathbf{x}](t)$ *generated by the conditioned IDS (48) is such that the Hausdorff distance* $H([\mathbf{x}](t)], [\mathbb{X}(t)])$ *between* $[\mathbf{x}](t)]$ *and the hull box* $[\mathbb{X}(t)]$ *of* $\mathbb{X}(t)$ *at time* t *is* $o(\varepsilon)$.

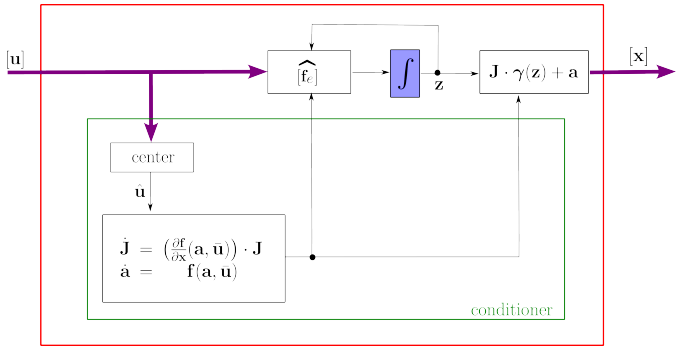


Figure 8: The conditioner increases the accuracy of the interval integrator

Remark 4.1. The two first equations of (48) are named the *variational equations* [2]. The pair $(\mathbf{a}, \mathbf{J})$ follows the flow, *i.e.*, the function $\mathbf{\Phi}_t(\mathbf{x}_0)$ which returns the state vector $\mathbf{x}(t)$ reached at time t assuming that $\mathbf{x}(0)$ has been initialized at $\mathbf{x}_0$. The matrix $\mathbf{J}$ satisfies $\mathbf{J}(t) = \frac{\partial \mathbf{\Phi}_t(\mathbf{x}_0)}{\partial \mathbf{x}}$. The principle of the conditioner is to store all uncertainties in the error $\mathbf{e}$ and propagate them through a single box $[\mathbf{e}]$ which will be shown to be almost static. More precisely, the inflation is in $o(\varepsilon)$.

Proof. Consider a trajectory of the system $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u})$. Since $\mathbf{f}$ is C_1 in $\mathbf{x}$, the Jacobian matrix $\mathbf{J}(t)$ is invertible for all t [3]. This is a consequence of Liouville's formula for linear systems:

$$\frac{d}{dt} \det \mathbf{J}(t) = \mathrm{tr}\left(\frac{\partial \mathbf{f}}{\partial \mathbf{x}}(\mathbf{a}, \bar{\mathbf{u}})\right) \cdot \det \mathbf{J}(t). \tag{51}$$

This scalar differential equation yields

$$\det \mathbf{J}(t) = \exp\left(\int_0^t \mathrm{tr}\left(\frac{\partial \mathbf{f}}{\partial \mathbf{x}}(\mathbf{a}, \bar{\mathbf{u}})\right) ds\right). \tag{52}$$

which never vanishes. Define the vector (seen as an error)

$$\mathbf{e} = \mathbf{J}^{-1}(t)\,(\mathbf{x} - \mathbf{a}(t)) \tag{53}$$

as illustrated by Figure 9. The first step of the proof is to show that $\mathbf{e}$ satisfies $\dot{\mathbf{e}} = \mathbf{f}_e(\mathbf{e})$ where $\mathbf{f}_e$ is given in (49), and to show that $\dot{\mathbf{e}} = o(\varepsilon)$.

By differentiating the relation $\mathbf{Je} = \mathbf{x} - \mathbf{a}$ with respect to t, we get:

$$\begin{array}{rcl} \dot{\mathbf{J}}\mathbf{e} + \mathbf{J}\dot{\mathbf{e}} & = & \dot{\mathbf{x}} - \dot{\mathbf{a}} \\ & = & \mathbf{f}(\mathbf{x},\mathbf{u}) - \dot{\mathbf{a}} \\ & = & \mathbf{f}(\mathbf{Je} + \mathbf{a},\mathbf{u}) - \dot{\mathbf{a}}. \end{array} \tag{54}$$

Thus,

$$\dot{\mathbf{e}} \;=\; \underbrace{\mathbf{J}^{-1}\left(-\dot{\mathbf{J}}\mathbf{e} + \mathbf{f}(\mathbf{Je} + \mathbf{a},\mathbf{u}) - \dot{\mathbf{a}}\right)}_{\mathbf{f}_e(\mathbf{e})}. \tag{55}$$

Note that

$$\begin{array}{rcl} \mathbf{f}_e(\mathbf{e}) & = & \mathbf{J}^{-1} \cdot \left(- \underbrace{\dot{\mathbf{J}}}_{=\left(\frac{\partial \mathbf{f}}{\partial \mathbf{x}}(\mathbf{a},\bar{\mathbf{u}})\right)\cdot\mathbf{J}} \mathbf{e} + \underbrace{\mathbf{f}(\mathbf{Je} + \mathbf{a},\mathbf{u})}_{=\mathbf{f}(\mathbf{a},\mathbf{u}) + \frac{\partial \mathbf{f}}{\partial \mathbf{x}}(\mathbf{a},\mathbf{u})\cdot(\mathbf{Je}) + o(\|\mathbf{e}\|)} - \mathbf{f}(\mathbf{a},\bar{\mathbf{u}}) \right) \\ & = & \mathbf{J}^{-1} \cdot \underbrace{(\mathbf{f}(\mathbf{a},\mathbf{u}) - \mathbf{f}(\mathbf{a},\bar{\mathbf{u}}))}_{o(\varepsilon)} + \underbrace{\left(\frac{\partial \mathbf{f}}{\partial \mathbf{x}}(\mathbf{a},\mathbf{u}) - \frac{\partial \mathbf{f}}{\partial \mathbf{x}}(\mathbf{a},\bar{\mathbf{u}})\right)}_{o(\varepsilon)} \cdot (\mathbf{Je}) + o(\|\mathbf{e}\|) \\ & = & o(\varepsilon). \end{array}$$

A direct consequence of the fact that $\mathbf{f}_e(\mathbf{e}) = o(\varepsilon)$ is that $\mathbf{\Gamma}([\mathbf{f}_e]) = o(\varepsilon)$, since we used the centered form. We would have $\mathbf{\Gamma}([\mathbf{f}_e]) = O(\varepsilon)$ if we had used the natural interval extension [19]. Thus $\dot{\mathbf{z}} = o(\varepsilon)$, whereas $\mathbf{z}(0) = O(\varepsilon)$.

Define the error set $\mathbb{E}(t) = \{\mathbf{e} | \exists \mathbf{x} \in \mathbb{X}(t), \mathbf{e} = \mathbf{J}^{-1}(t)(\mathbf{x} - \mathbf{a}(t))$. Due to the fact that $\dot{\mathbf{z}} = o(\varepsilon)$, we have $H(\mathbb{E}(t), [\mathbf{e}](t)) = o(\varepsilon)$ and therefore, $H(\mathbb{X}(t), \langle \mathbf{x} \rangle(t)) = o(\varepsilon)$ which corresponds to (ii). We immediately get that $H([\mathbb{X}(t)], [\mathbf{x}](t)) = o(\varepsilon)$ which corresponds to (i). □

4.3 Pendulum

Consider again the system given in Subsection 3.4, with $t_{\max} = 20$ instead of $t_{max} = 6$. Our conditioned IDS yields Figure 10. The trajectories painted yellow have been generated with $u = 0$ and starting with each corner of the initial box $[\mathbf{x}](0)$. With two different level of zooms (red and magenta), we illustrate the asymptotic minimality of the enclosure. For large t, we also observe a stability at the neighborhood of $\mathbf{0}$. More precisely, we can show that the state will enter inside small zone at the neighborhood of the origin and will stay in this zone forever.

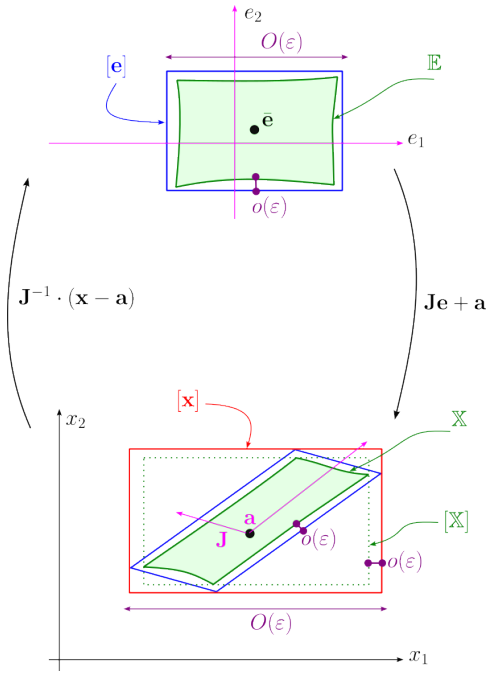


Figure 9: The conditioner increases the accuracy of the interval integrator

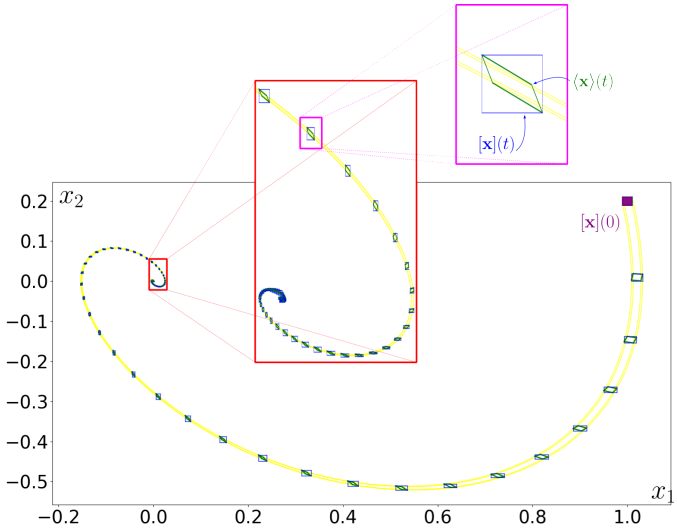


Figure 10: Boxes $[\mathbf{x}](t)$ and parallelepipeds $\langle\mathbf{x}\rangle(t)$ generated by our asymptotically minimal IDS

5 Application

5.1 Dead reckoning

Consider a dynamical system (for instance a vehicle such as a car or a boat) described by the state equations

$$\begin{array}{rcl} \dot{\mathbf{x}}_v & = & \mathbf{f}_v(\mathbf{x}_v, \mathbf{u}_v) \\ \mathbf{y}_v & = & \mathbf{g}_v(\mathbf{x}_v) + \mathbf{e}_v, \end{array} \tag{56}$$

where $\mathbf{x}_v$ is the state vector of the vehicle, $\mathbf{u}$ is the input vector, $\mathbf{y}$ is the measured output and $\mathbf{e}$ is the measured noise. Assume that we have a controller of the form

$$\begin{array}{rcl} \dot{\mathbf{x}}_c & = & \mathbf{f}_c(\mathbf{x}_c, \mathbf{y}_v) \\ \mathbf{u}_v & = & \mathbf{g}_c(\mathbf{x}_c) + \mathbf{e}_u. \end{array} \tag{57}$$

We have the extended system

$$\underbrace{\begin{pmatrix} \dot{\mathbf{x}}_v \\ \dot{\mathbf{x}}_c \end{pmatrix}}_{\dot{\mathbf{x}}} = \underbrace{\begin{pmatrix} \mathbf{f}_v(\mathbf{x}_v, \mathbf{g}_c(\mathbf{x}_c) + \mathbf{e}_u) \\ \mathbf{f}_c(\mathbf{x}_c, \mathbf{g}_v(\mathbf{x}_v) + \mathbf{e}_y) \end{pmatrix}}_{\mathbf{f}(\mathbf{x},\mathbf{u})} \tag{58}$$

where $\mathbf{x} = (\mathbf{x}_v, \mathbf{x}_c)$ is the extended state vector, $\mathbf{u} = (\mathbf{e}_u, \mathbf{e}_y)$ is the extended state input and $\mathbf{f} = (\mathbf{f}_v, \mathbf{f}_c)$ is the extended evolution function.

We want to show that the vehicle with its controller will enter a given zone $\mathbb{A}$ while avoiding a zone $\mathbb{B}$. This corresponds to a reachability problem for which we can use the preconditioned IDS.

5.2 Dubins car

The vehicle we consider is a Dubins car described by the state equations

$$\begin{array}{ll} (i) & \left\{ \begin{array}{rcl} \dot{x}_{v1} & = & x_{v4} \cos x_{v3} \\ \dot{x}_{v2} & = & x_{v4} \sin x_{v3} \\ \dot{x}_{v3} & = & u_{v1} + e_{u1} \\ \dot{x}_{v4} & = & u_{v2} + e_{u2} \end{array} \right. \\ (ii) & \left\{ \begin{array}{rcl} y_{v1} & = & x_{v3} + e_{y1} \\ y_{v2} & = & x_{v4} + e_{y2} \end{array} \right. \end{array} \tag{59}$$

where (i) is the evolution equation and (ii) is the output equation. The position is (x_{v1}, v_{v2}), the heading is x_{v3} and the speed is x_{v4}. The vehicle only measures its heading and its speed. At time $t = 0$, the state $\mathbf{x}_v$ is known to be inside a box $[\mathbf{x}_v](0)$. The noise of the actuators e_{u1}, e_{u2} are unknown and are assumed to be inside known intervals. The noise of the sensors e_{y1}, e_{y2} are also unknown and assumed to be inside known intervals.

5.3 Controller

We want to find a controller such that at time t_1 the position is inside the set $\mathbb{A} \subset \mathbb{R}^2$ and such that for all t, the position is never inside the set $\mathbb{B} \subset \mathbb{R}^2$. To find such a controller, we use a feedback linearization method. For this purpose, we ask the robot to follow a target point $\mathbf{c}(t)$ chosen such that (see Figure 11):

- $\mathbf{c}(0)$ is approximately the position of the car at $t = 0$,
- at time t_1 the point $\mathbf{c}$ is deep inside $\mathbb{A}$,
- $\mathbf{c}(t)$ clearly avoids $\mathbb{B}$.

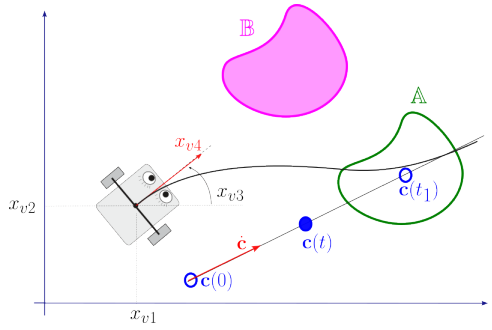


Figure 11: The Dubins car, has to reach $\mathbb{A}$ at time t_1 and to avoid $\mathbb{B}$

We get the following state feedback controller

$$\mathbf{u}_v = \begin{pmatrix} -(c_1(t) - x_{v1}) \frac{\sin x_{v3}}{x_{v4}} + (c_2(t) - x_{v2}) \frac{\cos x_{v3}}{x_4} \\ (c_1(t) - x_{v1}) \cos x_{v3} + (c_2(t) - x_{v2}) \sin x_{v3} - 2x_{v4} \end{pmatrix}. \tag{60}$$

Now, this controller cannot be implemented in this form since the position x_{v1}, x_{v2} are not measured. We need to use a predictor. It can be taken as (see the first two equation of 59)

$$\begin{array}{rcl} \dot{x}_{c1} & = & y_{v2} \cos y_{v1} \\ \dot{x}_{c2} & = & y_{v2} \sin y_{v1}. \end{array} \tag{61}$$

This prediction only uses a measure y_{v1} of the heading and a measure y_{v2} of the speed. This poor estimation can now be used by the controller (11), as illustrated by Figure 12.

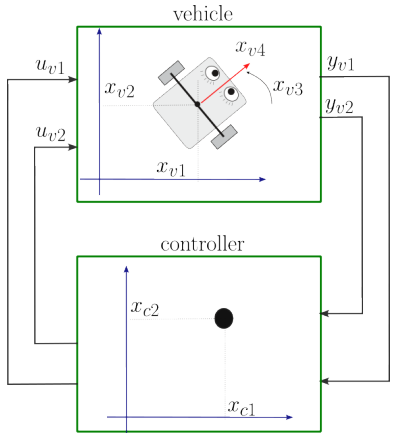


Figure 12: The Dubins car, has two outputs (heading and speed) used by the controller

The state equations of the controller are now

$$(iii) \qquad \begin{cases} \dot{x}_{c1} &= y_{v2}\cos y_{v1} \\ \dot{x}_{c2} &= y_{v2}\sin y_{v1} \end{cases}$$

$$(iv) \qquad \begin{cases} u_{v1} &= -(c_1(t)-x_{c1})\frac{\sin y_{v1}}{y_{v2}} + (c_2(t)-x_{c2})\frac{\cos y_{v1}}{y_{v2}} \\ u_{v2} &= (c_1(t)-x_{c1})\cos y_{v1} + (c_2(t)-x_{c2})\sin y_{v1} - 2y_{v2}. \end{cases}$$

5.4 Closed loop system

If we take the notation $\mathbf{x} = (\mathbf{x}_v, \mathbf{x}_c)$ and $\mathbf{u} = (\mathbf{e}_u, \mathbf{e}_y)$, the state equations of the closed loop system, including both the vehicle and the controller, are

$$\begin{array}{rcl} \dot{x}_1 &=& x_4\cos x_3 \\ \dot{x}_2 &=& x_4\sin x_3 \\ \dot{x}_3 &=& -(c_1(t)-x_5)\frac{\sin(x_3+u_3)}{x_4+u_4} + (c_2(t)-x_6)\frac{\cos(x_3+u_3)}{x_4+u_4} + u_1 \\ \dot{x}_4 &=& (c_1(t)-x_5)\cos(x_3+u_3) + (c_2(t)-x_6)\sin(x_3+u_3) - 2(x_4+u_4) + u_2 \\ \dot{x}_5 &=& (x_4+u_4)\cos(x_3+u_3) \\ \dot{x}_6 &=& (x_4+u_4)\sin(x_3+u_3). \end{array}$$

5.5 Reachability using the conditioned IDS

Assume that the initial state and the input $\mathbf{u}$ satisfy;

$$\mathbf{x}(0) \in \underbrace{[-\bar{\varepsilon},\bar{\varepsilon}]\times[-\bar{\varepsilon},\bar{\varepsilon}]\times[-2,2]\times[1-\bar{\varepsilon},1+\bar{\varepsilon}]\times[0,0]\times[0,0]}_{=[\mathbf{x}](0)}$$

$$\mathbf{u}(t) \in \underbrace{[-\bar{\varepsilon}^2,\bar{\varepsilon}^2]\times[-\bar{\varepsilon}^2,\bar{\varepsilon}^2]\times[-\bar{\varepsilon}^2,\bar{\varepsilon}^2]\times[-\bar{\varepsilon}^2,\bar{\varepsilon}^2]}_{=[\mathbf{u}]}, \forall t$$

where $\bar{\varepsilon} = 10^{-3}$. For the target point, take:

$$\mathbf{c}_i(t) = \begin{pmatrix} 1.8 + 0.9 \cdot t \\ 0.4 + 0.2 \cdot t \end{pmatrix}. \tag{62}$$

The Muller integration of the IDS presented in Section 3 generates Figure 13. The blue set encloses all feasible positions for $t \in [0, 4]$. The red zones contain the position of the car for $t \in \{1, 2, 3\}$. The black curves represents a feasible trajectory.

We were able to prove that the position of the car has reached the set $\mathbb{A} = [2, 3] \times [0.2, 0.8]$ at time $t = 3$. Moreover, for all $t \in [0, 4]$ we proved that the car has never entered inside the forbidden zone $\mathbb{B} = [1, 2] \times [0.8, 1]$.

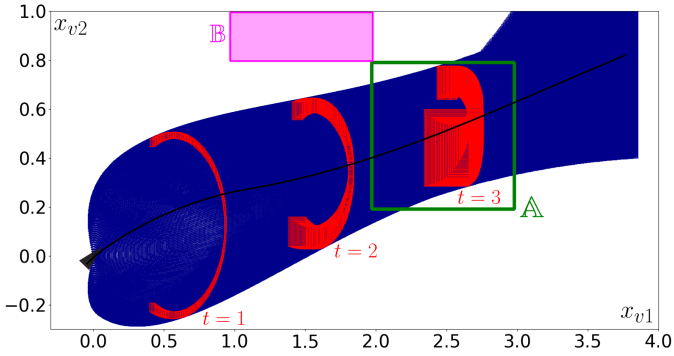


Figure 13: At time $t = 3$, the car reaches $\mathbb{A}$ and always avoid $\mathbb{B}$

With the same accuracy and using a conditioner, as presented in Section 4, the predictor yields Figure 3.

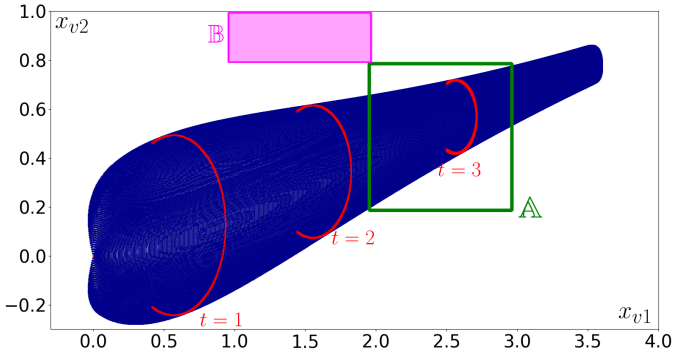


Figure 14: At time $t = 3$, the car has reached $\mathbb{A}$ always avoiding $\mathbb{B}$

For both, the results have been obtained by bisecting the initial box into 40 small boxes. For both, the resulting computing time was less than 500 sec.

Figure 15 shows the superposition of many feasible trajectories obtained by sampling. The pessimism (distance between the yellow and the blue approximations) can be considered as low.

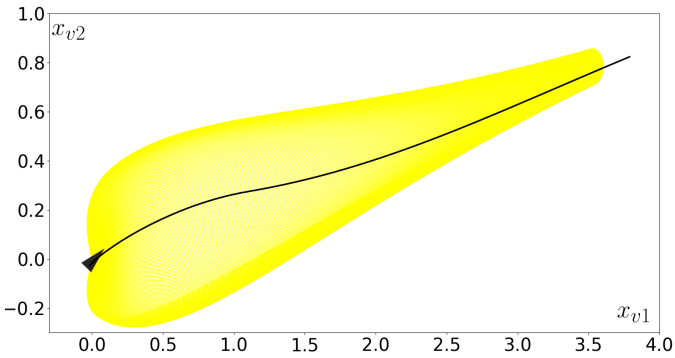


Figure 15: Many trajectories generated to get an inner approximation of the set of reachable positions

6 Conclusion

This paper has proposed a method to reformulate a reachability problem into an ordinary differential equation (ODE) using the Muller theorem. Solving numerically this ODE yields a interval tube which contains all feasible trajectories. One important intuition of using the Muller transform is to take into account the bijective property of the flow function. Indeed, if we know that a function $\mathbf{f} : \mathbb{R}^n \mapsto \mathbb{R}^n$ is a bijection, computing the range of $\mathbf{f}([\mathbf{x}])$, for a box $[\mathbf{x}]$, amounts to compute the range of $\mathbf{f}(\partial[\mathbf{x}])$, where $\partial[\mathbf{x}]$ is the boundary of $[\mathbf{x}]$.

Although our approach suffers from a wrapping effect, due to the interval approximation, we have shown that the approximation is asymptotically minimal if the right conditioner is chosen. This means that when the uncertainties are infinitesimal ($O(\varepsilon)$ for the initial state and $O(\varepsilon^2)$ for the state noise) then almost no overestimation is introduced (in $O(\varepsilon^2)$).

The implementation is done using the Codac library [24] and the source codes are available at `https://webperso.ensta.fr/jaulin/muller.html`.

References

[1] Althoff, M. An Introduction to CORA. *CPS Week*, pages 120–151, 2015. URL: `https://mediatum.ub.tum.de/doc/1280439/document.pdf`.

[2] Arnold, V. *Geometrical Methods In The Theory Of Ordinary Differential Equations*. Springer-Verlag, 1988. DOI: `10.1007/978-3-662-11832-0`.

[3] Arnold, V. I. *Ordinary Differential Equations*. Springer, 2 edition, 1992. URL: https://link.springer.com/book/9783540345633.

[4] Asarin, E., Dang, T., and Girard, A. Reachability analysis of nonlinear systems using conservative approximation. In Maler, O. and Pnueli, A., editors, *Hybrid Systems: Computation and Control*, Volume 2623 of *Lecture Notes in Computer Science*, pages 20–35. Springer, 2003. DOI: 10.1007/3-540-36580-X_5.

[5] Aubin, J. P. *Viability Theory*. Birkhäuser Boston, 2009. DOI: 10.1007/978-0-8176-4910-4.

[6] Collins, P. and Goldsztejn, A. The reach-and-evolve algorithm for reachability analysis of nonlinear dynamical systems. *Electronic Notes in Theoretical Computer Science*, 223:87–102, 2008. DOI: 10.1016/j.entcs.2008.12.033.

[7] Coogan, S. Mixed monotonicity for reachability and safety in dynamical systems. In *Proceedings of the 59th IEEE Conference on Decision and Control*, pages 5074–5085. IEEE, 2020. DOI: 10.1109/CDC42340.2020.9304391.

[8] Cyranka, J., Islam, A., Byrne, G., Jones, P., Smolka, S., and Grosu, R. Lagrangian reachabilty. In *Proceedings of the 29th International Conference on Computer Aided Verification, Part I*, Volume 10426 of *Lecture Notes in Computer Science*, pages 379–400. Springer, 2017. DOI: 10.1007/978-3-319-63387-9_19.

[9] Frehse, G. PHAVer: Algorithmic verification of hybrid systems. *International Journal on Software Tools for Technology Transfer*, 10(3):23–48, 2008. DOI: 10.1007/978-3-540-31954-2_17.

[10] Geretti, L., dit Sandretto, J. A., Althoff, M., Benet, L., Chapoutot, A., Chen, X., Collins, P., Forets, M., D.Freire, Immler, F., Kochdumper, N., Sanders, D., and Schilling, C. ARCH-COMP20 category report: Continuous and hybrid systems with nonlinear dynamics. In *Proceedings of the 7th International Workshop on Applied Verification of Continuous and Hybrid Systems*, pages 49–21, Berlin, Germany, 2020. DOI: 10.29007/zkf6.

[11] Goubault, E., Mullier, O., Putot, S., and Kieffer, M. Inner approximated reachability analysis. In *Proceedings of the 17th International Conference on Hybrid Systems: Computation and Control*, pages 163–172. ACM, 2014. DOI: 10.1145/2562059.2562113.

[12] Gouzé, J. L., Rapaport, A., and Hadj-Sadok, M. Z. Interval observers for uncertain biological systems. *Ecological Modelling*, 133(1):45–56, 2000. DOI: 10.1016/S0304-3800(00)00279-9.

[13] Jaulin, L. Asymptotically minimal interval contractors based on the centered form. *Acta Cybernetica*, 26(4):933–954, 2024. DOI: 10.14232/actacyb.306222.

[14] Kapela, T., Mrozek, M., Wilczak, D., and Zgliczynski, P. CAPD::DynSys: A flexible C++ toolbox for rigorous numerical analysis of dynamical systems. *Communications in Nonlinear Science and Numerical Simulation*, 101:105578, 2021. DOI: 10.1016/j.cnsns.2020.105578.

[15] Kieffer, M. and Walter, E. Guaranteed nonlinear state estimation for continuous-time dynamical models from discrete-time measurements. In *Proceedings of the 17th IFAC World Congress*, Seoul, Republic of Korea, 2006. DOI: 10.3182/20060709-3-KR-2910.00020.

[16] Kurzhanski, A. and Varaiya, P. Ellipsoidal techniques for reachability under state constraints. *SIAM Journal on Control and Optimization*, 45(4):1369–1394, 2006. DOI: 10.1137/S0363012903437605.

[17] Le Bars, F., Sliwka, J., Jaulin, L., and Reynet, O. Set-membership state estimation with fleeting data. *Automatica*, 48(2):381–387, 2012. DOI: 10.1016/j.automatica.2011.11.004.

[18] Le Guernic, C. and Girard, A. Reachability analysis of linear systems using support functions. *Nonlinear Analysis: Hybrid Systems*, 4(2):250–262, 2010. DOI: 10.1016/j.nahs.2009.03.003.

[19] Moore, R. *Methods and Applications of Interval Analysis*. Society for Industrial and Applied Mathematics, 1979. DOI: 10.1137/1.9781611970906.

[20] Müller, M. Über das Fundamentaltheorem in der Theorie der gewöhnlichen Differentialgleichungen. *Mathematische Zeitschrift*, 26:619–645, 1927. DOI: 10.1007/BF01475477.

[21] Nedialkov, N. S., Jackson, K. R., and Corliss, G. F. Validated solutions of initial value problems for ordinary differential equations. *Applied Mathematics and Computation*, 105(1):21–68, 1999. DOI: 10.1016/S0096-3003(99)00004-8.

[22] Raissi, T., Efimov, D., and Zolghadri, A. Interval state estimation for a class of nonlinear systems. *IEEE Transactions on Automatic Control*, 57(1):260–265, 2012. DOI: 10.1109/TAC.2011.2164730.

[23] Rauh, A., Westphal, R., and Aschemann, H. Verified simulation of control systems with interval parameters using an exponential state enclosure technique. In *Proceedings of the 18th International Conference on Methods and Models in Automation and Robotics*, pages 241–246, 2013. DOI: 10.1109/MMAR.2013.6669913.

[24] Rohou, S., Desrochers, B., and Le Bars, F. The codac library. *Acta Cybernetica*, 26(4):871–887, 2024. DOI: 10.14232/actacyb.302772.

[25] Rohou, S., Jaulin, L., Mihaylova, L., Le Bars, F., and Veres, S. *Reliable Robot Localization*. Wiley, 2019. DOI: 10.1002/9781119680970.

[26] Scott, J. K. and Barton, P. I. Bounds on the reachable sets of nonlinear control systems. *Automatica*, 49(1):93–100, 2013. DOI: 10.1016/j.automatica.2012.09.020.

[27] Taha, W. et al. Acumen: An open-source testbed for cyber-physical systems research. In *Proceedings of the Conference on CYber physiCaL systems, iOt and sensors Networks*, Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, pages 118–130, Cham, 2015. Springer International Publishing. DOI: 10.1007/978-3-319-47063-4_11.

[28] Wan, J. *Computationally reliable approaches of contractive model predictive control for discrete-time systems*. PhD dissertation, Universitat de Girona, Girona, Spain, 2007. URL: https://www.tdx.cat/handle/10803/7740#page=1.

[29] Xue, B., Mosaad, P. N., Franzle, M., Chen, M., Li, Y., and Zhan, N. Safe over- and under-approximation of reachable sets for delay differential equations. In *Proceedings of the 15th International Conference on Formal Modeling and Analysis of Timed Systems*, Volume 10419 of *Lecture Notes in Computer Science*, pages 281–299. Springer, 2017. DOI: 10.1007/978-3-319-65765-3_16.

Acta Cybernetica **27** (2026) 429–451.

On Propagating Uncertainties for Estimation and Association using Validated Interval Simulation

Elliot Brendel[ab], Julien Alexandre dit Sandretto[cd], and Charlotte Govignon[ef]

Abstract

We present a new uncertainty propagation algorithm based on interval arithmetic. The goal is to explore the benefits of a set-based approach for estimation and data association using validated simulation. The presented algorithm capitalises on the measures of a dynamical system to improve its estimation and reduce uncertainties on its trajectory. Our approach also contributes to data association by computing the precision required for a measure to belong to a given track with confidence levels. Mainly interested in space surveillance, we illustrate the contributions of this new algorithm with several scenarios of orbit determination and satellite tracking and their numerical simulations.

Keywords: estimation, association, potential clouds, validated simulation, satellite tracking

1 Introduction

Estimation and association are crucial steps needed in various critical applications such as robotics, autonomous vehicles and radar surveillance. When a set of measures is provided with the respective uncertainties, the estimation problem consists in using this information to compute the state of a dynamical system of interest, for example its position and velocity coordinates. Usually, uncertainties on this estimated state are also computed. Estimation techniques which minimize the state's uncertainties, for instance in terms of Cramér-Rao bound, are often chosen rather than less accurate ones. The estimation accuracy can also be improved by the

[a]Thales Land and Air Systems, Paris, France
[b]E-mail: elliot.brendel@thalesgroup.com, ORCID: 0000-0002-0458-4993
[c]ENSTA Paris, Institut Polytechnique de Paris, 828 Boulevard des Maréchaux, Palaiseau 91120, France
[d]E-mail: julien.alexandre-dit-sandretto@ensta-paris.fr, ORCID: 0000-0002-6185-2480
[e]École des Ponts ParisTech, Cité Descartes, 6 et 8 avenue Blaise Pascal, 77420 Champs-sur-Marne, France
[f]E-mail: charlotte.govignon@ponts.org, ORCID: 0009-0003-7703-0785

DOI: 10.14232/actacyb.314226

simulation of the considered dynamical system, which often requires computing the solution of a differential equation representing its dynamics. When multiple systems evolve in the observed environment, the sensor providing the measures can produce data coming from different objects. Then, filtering of the measures between those coming from the system of interest and the rest is mandatory to prevent the estimate from corruption. This problem is called association. In this paper, estimation and association techniques are presented with a focus on dealing with dynamical systems subject to potentially large uncertainties (such as low-earth satellites), while providing guarantees and confidence levels on the computed results and constraining conservatism.

Data association techniques can be divided in two categories. First, delayed decision techniques wait until several measures are received to perform an association using the set of the measures. For example, Multi-Hypothesis Tracking (MHT) [5, 17] approaches consist in building and updating a set of association hypotheses until their respective estimated probabilities lead to the removal of one of the previous hypotheses. These techniques are efficient but show high combinatorial cost, which can lead to a long time before decision, and can be a drawback in an operating environment. Many other delayed decision approaches consist in the combination of gating [6] and estimation [13, 19, 20] steps. A gating step consists in the selection of a subset of measures satisfying a condition (e.g. all the measures whose distance with the estimate is under a chosen threshold). Then, estimation methods, such as Extended Kalman Filter [5, 13, 19, 20] or Least Squares optimization [10, 19, 20] perform an update of the estimate with the subset of previously selected measures. At last, the new estimate is used to realize a final gating step, and the new selected measures are considered as associated with the object.

Instantaneous decision techniques allocate a measure to an object at the time when, or very shortly after, the measure is received. In these techniques, Nearest Neighbor (NN) [5] methods consist in allocating the considered measure to its closest object by computing a chosen quantity representing a distance between the measure and the objects in its environment. These quantities can be deterministic (e.g. the Cartesian distance) or statistical (e.g. the Mahalanobis distance [20]). When multiple measures and multiple objects are considered at the same time, Global Nearest Neighbor (GNN) methods [5] perform optimization algorithms, such as the Hungarian method [14], to maximize association scores between measures and objects. In general, when the number of measures and objects increases, NN and GNN's efficiency decreases. Probabilistic Data Association (PDA) [3, 5, 8] techniques are instantaneous decision techniques performing the computation of the probability of each association hypothesis and the selection of a subset of these hypotheses. Then, a combination of the selected hypotheses is computed, allocated to an object and used to improve its estimate. This approach is usually suitable for environments with multiple objects and risk of confusion.

Most of the described association techniques are probabilistic approaches which are often based on the hypothesis that Gaussian distributions describe well enough the considered uncertainties (from the measures and the estimates) all along the trajectories of the observed objects. However, in multiple documented examples

such as satellite tracking [11], a Gaussian distribution propagated along a trajectory can lose its Gaussian nature, in particular with large uncertainties, which can happen when the tracked object generated a limited number of measures. The previously presented methods then become unsuitable. Conversely, deterministic set-based propagation and association approaches, using classical set theory functions such as intersection, inclusion and contractors, provide a guaranteed enclosure of all the possible values of the objects states. The obtained enclosures can be used to decide, with guarantee, if a measure belongs to an object or not.

The main tool used for deterministic association and state estimation is the well known interval arithmetic (IA). For example, the latter has been applied to the localization of a robot in [12]. In real world, measures come with uncertainties, but in some cases they can be completely wrong. This is the problem of outliers. They must be carefully handled. With IA, the relaxed intersection is efficient for the outlier management as in [7, 12]. In this paper, we focus on dynamical objects (satellites) and thus, an approach handling dynamics is required. In [18], a method for data association and state estimation with dynamics has been proposed. However, all these techniques require continuous, frequent or regular observations of landmarks, which is not the case of stallites tracking. Observations are rare and tracking is highly uncertain. If IA based techniques seem relevant in a space surveillance context, some new methods are needed. The approaches developed in this paper are applied to association, estimation and tracking of satellites observed by a ground radar. Performances and benefits of these approaches are discussed in connection to this space surveillance application. Indeed, classical IA techniques for the propagation of uncertain trajectories will provide a too large approximation, because of the long time between observations of a satellite by a radar, due to Earth revolution and the dynamics of satellites. The result will not be usable for discrimination between two satellites. To deal with this issue, we propose a confidence-based approach that allows us to make varying the well known conservatism associated with IA.

Section 2 provides preliminary notions required in this paper. Our approach for estimation and association from rare and uncertain measures is presented in Section 3. Then, Section 4 presents numerical results for the simulation in four different scenarios. Conclusion and perspectives are given in Section 5.

2 Preliminary Notions

One of the most used tool to handle bounded uncertainties is the interval arithmetic. In this section, we present the notions used in the paper.

2.1 Interval arithmetic

An interval $[a] = [\underline{a}, \overline{a}]$ defines the set of $a \in \mathbb{R}$ such that $\underline{a} \leq a \leq \overline{a}$. $\mathbb{IR}$ denotes the set of all intervals over $\mathbb{R}$. Usual mathematical operators are extended to intervals: $[a] * [b] = \{a * b \,|\, a \in [a], b \in [b]\}$ with $*$ a binary operator on real numbers. Such

operations can sometimes be expressed via the bounds of the intervals, but not always. For instance, the sum of two intervals $[a]$ and $[b]$ gives $\left[\underline{a}+\underline{b}, \overline{a}+\overline{b}\right]$, but the division of an interval $[a]$ by an interval $[b]$ gives different expressions depending on interval $[b]$ containing 0 or not. Elementary functions can also be extended to interval arithmetic. A *tube* denotes the image of a function $f : t \in \mathbb{R} \mapsto [f(t)] \in \mathbb{IR}$. The Cartesian product of n intervals is called an interval vector or a *box* $[a] \in \mathbb{IR}^n$.

An *interval contractor* associated with the set $A \subset \mathbb{R}^n$ is an operator $C : [a] \in \mathbb{IR}^n \mapsto C([a]) \in \mathbb{IR}^n$ satisfying the contractance property $C([a]) \subseteq [a]$ and the correctness property $C([a]) \cap A = [a] \cap A$. Associated with a *constraint*, a contractor allows to reduce the size of an interval while preserving the subset of real numbers verifying the constraint.

2.2 Validated Simulation

Numerical integration consists in approximating a solution of an ordinary differential equation (ODE), $\dot{x} = f(x)$, using an integration scheme (such as Euler or Runge-Kutta). Interval arithmetic can be applied to numerical integration to provide validated numerical integration methods, also named reachability analysis or guaranteed simulation.

Such validated integration methods use guaranteed integration schemes, providing a discretization of time, $t_0 \leqslant \cdots \leqslant t_{\text{end}}$, and a computation of enclosures of the set of states of the system $x_0, \ldots, x_{\text{end}}$.

A guaranteed integration scheme consists in an integration method $\Phi(f, x_j, t_j, h)$, approximating the exact solution $x(t_{j+1}; x_j)$, i.e., $x(t_{j+1}; x_j) \approx \Phi(f, x_j, t_j, h)$, where x_j is an initial value, t_j the initial time, and h the step-size ($t_{j+1} = t_j + h$), and a truncation error function $\text{LTE}_\Phi(f, x_j, t_j, h)$, such that $x(t_{j+1}; x_j) = \Phi(f, x_j, t_j, h) + \text{LTE}_\Phi(f, x_j, t_j, h)$.

Taking into account the approximation of the used integration scheme, in contrast with usual integration methods, a validated numerical integration method is a two-step method which, firstly, computes an enclosure $[\widetilde{x}_j]$ of the solution over the time interval $[t_j, t_{j+1}]$ to bound $\text{LTE}_\Phi(f, x_j, t_j, h)$, then, computes a tight enclosure of the solution for the final time instant t_{j+1}. Multiple methods exist to perform these steps, such as Taylor series and Runge-Kutta, see [2, 15] and the references therein for more details. The step-size can be fixed (h is a constant) or adaptive. Finally, functions R and $\widetilde{R}$ are provided by validated numerical integration methods

$$R : \left\{ \begin{array}{ccc} \mathbb{R} & \to & \mathbb{IR}^n \\ t & \mapsto & [x] \end{array} \right. , \ \widetilde{R} : \left\{ \begin{array}{ccc} \mathbb{IR} & \to & \mathbb{IR}^n \\ [\underline{t}, \overline{t}] & \mapsto & [\widetilde{x}] \end{array} \right. \tag{1}$$

which for a given t_i, $R(t_i) = \{x(t_i; x_0) : \forall x_0 \in [x_0]\} \subseteq [x]$, and $\widetilde{R}([\underline{t}, \overline{t}]) = \{x(t; x_0) : \forall x_0 \in [x_0] \wedge \forall t \in [\underline{t}, \overline{t}]\} \subseteq [\widetilde{x}]$.

2.3 Confidence Contractor

Intervals and probabilities are strongly connected. Indeed, the universe of a distribution is an interval, and a given confidence level is associated with a confidence

interval. A *confidence interval* is a set $\mathcal{S}$ for which the probability of the given random variable to be in this set is equal to a given probability P. In other words, let $\widehat{x}$ be a single observed sample of a quantity X. In statistics, an observed data allows to compute a confidence interval [16], that is to say an interval which may contain the actual value, with respect to a given confidence level. For example, considering a confidence level CL = 95%, one can define the confidence interval $C_{95\%}$. This interval can be obtained by observation (statistical approach) or with the help of a known distribution (probabilistic approach). A new measure $\widehat{x}$ coming from the (same) experiment will be in the associated confidence interval such that $\widehat{x} \in C_{95\%}$ 95% of the time.

We use in our algorithm the confidence contractor proposed in [1]:

$$\begin{array}{rccc} Cbc([x]|f_X, \text{cc}) : \mathbb{IR} & \rightarrow & \mathbb{IR} \\ [x] & \mapsto & [x] \cap [y] \end{array} \tag{2}$$

with $[y]$ defined such that $Pr(x \in [y]) = \int_{[y]} f_X(x)\, dx = \text{cc}$ ($[y]$ is the confidence interval), $Pr(x \in [y])$ stands for "probability that x is in $[y]$", with x following the distribution f_X, and cc being the confidence coefficient ($0 \leq \text{cc} \leq 1$). For example, cc = 0.68 for a 68% confidence level. In the domain of ODEs, confidence contractor has been used to produce potential clouds [9] in [1]. A potential cloud obtained from the validated simulation of an ODE is a set of reachable tubes computed from the most conservative initial states (the support) to the less one (the mean).

3 Association and Estimation of Measures

Based on interval arithmetic and validated simulation, our approach uses measures of the observed object's state to improve the estimation of its trajectory and rule out incompatible measures and tracks, where a track is a set of measures associated together and the corresponding state estimate and uncertainties.

3.1 Definition of Measures

Measuring instruments or sensors provide measures and their corresponding uncertainties, in either a deterministic (3) or a probabilistic (4) approach. In the deterministic case, the measure $x_{\text{mes}} \in \mathbb{R}^n$ and uncertainty $\Delta x_{\text{mes}} \in \mathbb{R}^n_+$ generate an interval vector that necessarily includes the true quantity x_{true}:

$$x_{\text{true}} \in [x_{\text{mes}} - \Delta x_{\text{mes}}, x_{\text{mes}} + \Delta x_{\text{mes}}]. \tag{3}$$

In the probabilistic approach, the measure is represented as a random variable X_{mes} normally distributed with mean x_{true} and covariance $P \in \mathbb{R}^{n \times n}$:

$$X_{\text{mes}} \sim \mathcal{N}(x_{\text{true}}, P). \tag{4}$$

In the probabilistic approach, the uncertainty is carried by the covariance matrix P. The probabilistic approach is often preferred to represent uncertain measures.

However, the downside of this model is its lack of accuracy to describe rare events and singularities that have a low probability to be drawn. In contrast, the deterministic, or set-based approach, allows to take the entire measure interval into consideration, which is significantly more robust with worst case scenarios. For this reason, we choose to consider measures as real valued intervals.

When sensors provide probabilistic uncertainties, a conversion from the uncertainty covariance P to a deterministic uncertainty Δx_{mes} can be performed (see Subsection 4.2).

3.2 Estimation with Interval Measures

In an ideal case, complete observation of the system is provided. However, it is often impossible (e.g. position is measured but not velocity), therefore a more realistic case has to be considered with an only partial observation of the system.

3.2.1 Complete Observation of the System

A dynamical system of state $x_{\text{true}}(t) \in \mathbb{R}^n$ at time $t \in \mathbb{R}$, following the dynamics $\dot{x}_{\text{true}} = f(x_{\text{true}})$ with $f : \mathbb{R}^n \to \mathbb{R}^n$ is considered. The state is measured with a deterministic approach, as in (3).

From the sensor point of view, measures of the dynamical system with their corresponding uncertainties provide an enclosure of the state. At time t, the measure $x_t^m \in \mathbb{R}^n$ and its corresponding uncertainty $\Delta x_t^m \in \mathbb{R}_+^n$ are received, and lead to the following inclusion:

$$x_{\text{true}}(t) \in [x_t^m - \Delta x_t^m, x_t^m + \Delta x_t^m]. \tag{5}$$

From the observed object point of view, validated numerical integration of the dynamical system provides an interval estimate encompassing the state of the system. At time t, the estimate $[\widehat{x}_t]$ of the system is given by:

$$[\widehat{x}_t] := R(t) \in \mathbb{IR}^n, \tag{6}$$

where $R(\cdot)$ is defined in (1), and $R(t_0) = [\widehat{x}_0]$ with $[\widehat{x}_0]$ the initial value of the system estimate, at time t_0. The true state at time t is also included in $[\widehat{x}_t]$:

$$x_{\text{true}}(t) \in [\widehat{x}_t]. \tag{7}$$

Using (5) and (7), the state of the dynamical system is included in the intersection of the measure and the estimate:

$$x_{\text{true}}(t) \in [x_t^m - \Delta x_t^m, x_t^m + \Delta x_t^m] \cap [\widehat{x}_t]. \tag{8}$$

Furthermore, if the exact time $t \in \mathbb{R}$ of the measure is not known precisely, it can be approximated by a time interval. Assuming that the exact time t is bounded by $\underline{t} \leq t \leq \overline{t}$, then instead of using $R(\cdot)$ to provide the estimate as in (6), the function $\widetilde{R}(\cdot)$ generates an estimate $[\widehat{x}_t] := \widetilde{R}(t) \in \mathbb{IR}^n$, taking into account the uncertainty on the time of the measure.

3.2.2 Partial Observation of the System

In some cases, the sensor provide measures from the dynamical system in a different reference frame or partial measures with respect to the state. In this section, it is considered that the sensor can only make observations $y_t^m \in \mathbb{R}^p$ of the system $x_{\text{true}}(t) \in \mathbb{R}^n$, with $p \leq n$. An observation function $g : \mathbb{R}^n \to \mathbb{R}^p$, allowing the conversion of a vector from the state frame to the measure frame, gives the following theoretical constraint for a measure without uncertainty

$$g(x_{\text{true}}(t)) - y_t^m = 0, \tag{9}$$

which represents the compatibility of the system x_{true} with the measure y_t^m at time t. If a measure y_t^m does not verify this constraint, then it cannot come from the system x_{true}. Further, if the measure y_t^m comes from the system x_{true}, the constraint in (9) provides more information on the state of system $x_{\text{true}}(t)$.

The constraint in (9) can be written as a double inclusion between two singletons: $\{g(x_{\text{true}}(t))\} \subset \{y_t^m\}$ and $\{y_t^m\} \subset \{g(x_{\text{true}}(t))\}$. This last constraint can be written as $\left\{g^{-1}\left(y_t^m\right)\right\} \subset \{x_{\text{true}}(t)\}$ when g is bijective. Therefore, given a partial measure interval $[y_t^m] \in \mathbb{IR}^p$ and an estimated interval $[\widehat{x}_t] \in \mathbb{IR}^n$ of a system at time t, the interval counterpart of (9) is as follows:

$$\begin{cases} g([\widehat{x}_t]) \subset [y_t^m], \\ g^{-1}([y_t^m]) \subset [\widehat{x}_t]. \end{cases} \tag{10}$$

The smallest subset of $[\widehat{x}_t]$ that respects (10) is computed with a forward-backward contractor [4], providing a solution without computing g^{-1}, as g is usually not bijective.

As a result, when the measures are only partial and cannot be used to directly intersect the estimated interval (8), interval contractors allow to use the partial information provided by the measures (10).

3.3 Association of Measures

Association is performed by comparing the estimate of a dynamical system with new measures. If the association decision is positive, the estimation step is computed using the information of the measures and their uncertainties to reduce the uncertainty of the estimate. Our algorithm (see Algorithm 1) performs these steps for the complete observation case (see Subsection 3.2.1). A similar algorithm has been implemented for the partial observation case (see Subsection 3.2.2) using (10) instead of (8) as an association rule and improvement of the estimate.

Our algorithm (see Algorithm 1) ensures that a measure is compatible with the simulated dynamical system to perform their association. Since the simulation is validated, the estimated interval vector necessarily includes every possible states of the dynamical system. So, if a measure interval does not intersect with the simulated interval corresponding to the same instant, this necessarily means that the measurement is incompatible with the evaluated track. Nonetheless, it is

possible that certain measures give a non-empty intersection with the simulation estimate, but do not really correspond to the same system. Hence the significance of introducing confidence levels in the measurements.

Algorithm 1 Interval Measure Association with Complete Observation Algorithm.

Funct IMACO($[\widehat{x}_t]$, x_t^m, Δx_t^m)

if $[\widehat{x}_t] \cap [x_t^m - \Delta x_t^m, x_t^m + \Delta x_t^m] \neq \emptyset$ **then** *Association rule*
  Association because of compatibility
  Improvement of the estimate: $[\widehat{x}_t] \leftarrow [\widehat{x}_t] \cap [x_t^m - \Delta x_t^m, x_t^m + \Delta x_t^m]$ (8)
else
  Incompatibility, no association
end if
return $[\widehat{x}_t]$ and association decision

---

for some values of confidence $0 < \mathrm{cc}_i < 1$ **do**
  Compute $[\widehat{x}_t]_{\mathrm{cc}_i}$
  IMACO($[\widehat{x}_t]_{\mathrm{cc}_i}$, x_t^m, Δx_t^m)
end for

Once the algorithm ruled out incompatible measures, it uses the additional information of the measure to increase the precision on the state of the dynamical system using Section 3.2. This gives an upgrade of the estimate by reducing the uncertainty on the state of the system after the measure.

4 Application to Satellite Tracking

Space surveillance of low-earth objects has become a major challenge, notably in a military context. It can be addressed by ground radars, providing detection and tracking of satellites of interest. Estimation and association are crucial steps for the tracking of these objects.

It is important to notice that a ground radar can only provide measures for low-earth satellites when these objects are in its field of view. Therefore, due to earth rotation and satellite dynamics (see Subsection 4.1.2), measures of an observed satellite can be spaced out by several hours. This potentially important time range between measures can cause a loss of precision in the estimate of the satellite state, leading to increased uncertainties and, possibly, confusion with other objects. Moreover, the fact that a Gaussian distribution propagated along an orbital trajectory can lose its Gaussian nature [11] has been documented, in particular when uncertainties are large, for instance when the object generated a limited number of measures.

Therefore, the deterministic set-based propagation and association approaches developed in this paper can be relevant in a space surveillance context, allowing to relax the Gaussian hypothesis. In this section, useful preliminary notions of orbit determination are presented. Then, in order to demonstrate the benefits of our

approach in various substeps of the tracking task, our approach is studied through propagation, estimation and association scenarios. Finally, the performances of our approach are compared to a probabilistic NN method in a propagation and association scenario.

4.1 Preliminary Notions of Orbit Determination

4.1.1 Orbit Description

The state of a dynamical system can be given with its vectors of position and velocity. For a satellite, these six coordinates can be expressed in earth-centered Cartesian references such as *Earth-Centered Intertial* (ECI) or *Earth-Centered Earth-Fixed* (ECEF). Alternatively, it can be convenient to describe a satellite state in the *Keplerian* or the *Equinoctial* frames [19], especially when the dynamics are unperturbed, since these frames provide a more direct handling of the parameters of the ellipse. *Modified equinoctial elements* [21] (mEOE) are derived from Keplerian elements $(a, e, i, \Omega, \omega, \theta)$ as follows:

$$\begin{cases} p = a\left(1 - e^2\right), \\ f = e\cos\left(\omega + \Omega\right), \\ g = e\sin\left(\omega + \Omega\right), \\ h = \tan\left(\frac{i}{2}\right)\cos\Omega, \\ k = \tan\left(\frac{i}{2}\right)\sin\Omega, \\ L = \Omega + \omega + \theta, \end{cases} \tag{11}$$

where a denotes the semi-major axis, e the eccentricity, i the inclination, Ω the longitude of the ascending node, ω the argument of the periapsis, θ the true anomaly. The element p is called the semi-parameter and the element L is called the true longitude. This reference is useful for trajectory analysis because of the stability of the state components in time (see Subsection 4.1.2).

4.1.2 Dynamics of a Satellite

The dynamics of a satellite in mEOE is given by (12) [21]:

$$\begin{cases} \dot{p} = \frac{2p}{w}\sqrt{\frac{p}{\mu}}\Delta_t, \\ \dot{f} = \sqrt{\frac{p}{\mu}}\left(\frac{((w+1)\cos L + f)\Delta_t - gj\Delta_n}{w} + \Delta_r \sin L\right), \\ \dot{g} = \sqrt{\frac{p}{\mu}}\left(\frac{((w+1)\sin L + g)\Delta_t + fj\Delta_n}{w} - \Delta_r \cos L\right), \\ \dot{h} = \sqrt{\frac{p}{\mu}}\frac{s^2\Delta_n}{2w}\cos L, \\ \dot{k} = \sqrt{\frac{p}{\mu}}\frac{s^2\Delta_n}{2w}\sin L, \\ \dot{L} = \frac{\sqrt{\mu p}}{r^2} + \frac{1}{w}\sqrt{\frac{p}{\mu}}(h\sin L - k\cos L)\Delta_n, \end{cases} \tag{12}$$

where $w = 1 + f\cos L + g\sin L$, $s^2 = 1 + h^2 + k^2$, and $j = h\sin L - k\cos L$. Δ_t, Δ_r and Δ_n are the non-two-body perturbations in the radial, tangential and normal directions, respectively. These perturbations aggregate all other orbital perturbations acting on the satellite, like the J_k effects ($k \in \{2, 3, ...\}$) which come from the non-sphericity of the Earth, or the atmospheric drag for instance. For low-earth orbit satellites, J_2 effects induce the strongest perturbation. Therefore, the considered perturbations are:

$$\begin{cases} \Delta_r^{J_2} = -\frac{3\mu J_2 R_e^2}{2r^4}\left(1 - \frac{12j^2}{s^4}\right), \\ \Delta_t^{J_2} = -\frac{12\mu J_2 R_e^2}{r^4}\left(\frac{j(h\cos L + k\sin L)}{s^4}\right), \\ \Delta_n^{J_2} = -\frac{6\mu J_2 R_e^2}{r^4}\left(\frac{(1-h^2-k^2)j}{s^4}\right), \end{cases} \tag{13}$$

where J_2 is the second order zonal coefficient, R_e the earth radius, μ the standard gravitational parameter and $r = \frac{p}{w}$ is the radius. Without perturbations, all equinoctial elements are constant except the true longitude L which is close to linear (12).

4.1.3 Satellite Tracking by Ground Radar

The space is periodically scanned by a ground radar trying to detect new satellites. Then, every produced measure is compared to the estimates of known objects in order to decide if the measure was generated by a known object or by a previously unknown one: this is the measure association step. If the measure belongs to a known object, its estimate is updated and improved by the information of the measure. If the measure belongs to a new object, an estimate of this object is computed and the object will be considered as known for the future measures performed by the radar.

4.2 Simulation Hypotheses

In this paper, the considered radar produces measures of the position of detected objects in the ECI frame. This simplification is justified by the fact that measures performed in a Radar frame, such as Azimuth-Elevation-Range, can easily be converted in a Cartesian frame using classical formulas [19].

The measures provided by the radar come with a covariance matrix, also in ECI, representing the uncertainties of the associated measures. Mathematically, each of these measures can be modelled by a random variable $X_{\text{mes}} \in \mathbb{R}^3$ normally distributed with mean $x_{\text{true}} \in \mathbb{R}^3$ (the true position of the satellite in the ECI frame) and covariance $P \in \mathbb{R}^{3\times3}$ (depending on the radar performances and the satellite position), as in (4). The confidence contractor presented in (2) provides measure boxes, as in (3), corresponding to a chosen confidence level (in this paper, 99%).

The presented method has been implemented in the DynIbex framework [2]. It provides interval arithmetic tools such as contractors and offers a validated numerical integration procedure based on Runge-Kutta methods with adaptive step-size. The presented simulations were computed using the *implicit midpoint* method. The tolerance on the local truncation error [2] was set to 10^{-7}. This choice of parameters granted a good trade-off between computation time and precision.

The objects were simulated in the mEOE frame. The semi-parameter p was normalised by the Earth radius and the time unit for the integration was hours instead of seconds to facilitate the computations.

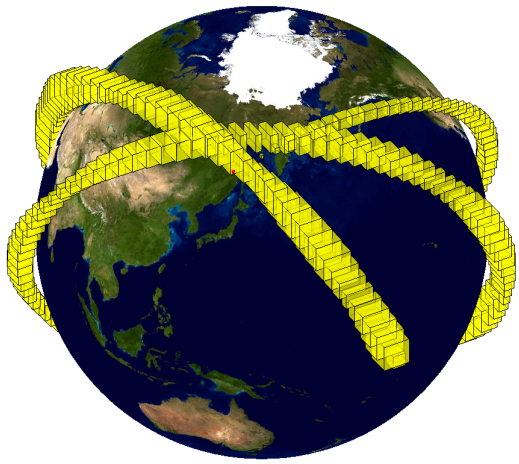


Figure 1: Simulation of a trajectory of a satellite for 6 hours, represented by the boxes of uncertainties on its position, with the starting box in red.

The initial estimate of each simulated satellite is supposed to be normally distributed with a mean vector x_0 and a covariance matrix P. x_0 depends on the scenario. P is common to scenarios 1 to 3 (see Subsections 4.3, 4.4 and 4.5) and chosen diagonal such that $P_{11} = 594.817$, $P_{66} = 1.258 \cdot 10^{-11}$ and $P_{ii} = 5.948 \cdot 10^{-14}$ for $i = \{2, \ldots, 5\}$. This covariance matrix is representative of the uncertainties on the state of a satellite after a few measures. It is satisfactory to demonstrate the contribution of our algorithm but in operating environment, distinct covariance matrices for the satellites of interest would be considered, depending on the measures used to compute their respective estimate, and a given time at which the matrices are evaluated.

Then, a reference box is computed using the confidence contractor of (2) with a 99% confidence level: $\Delta x_{\text{ref}} = [-x_{\text{ref}}, x_{\text{ref}}]$ where $x_{\text{ref},1} = 100$, $x_{\text{ref},6} = 1.45408 \cdot 10^{-5}$ and $x_{\text{ref},i} = 10^{-6}$ for $i = \{2, \ldots, 5\}$. This represents an uncertainty of $[-100, 100]$ meters on the semi-parameter p, and approximately $[-100, 100]$ meters on the arc of circle represented by the true longitude L when the altitude of the satellite is 500 kilometers.

The potential clouds associated with the 5% and 95% confidence levels are given by $\Delta x_{5\%} = 0.21\Delta x_{\text{ref}}$ and $\Delta x_{95\%} = 0.645\Delta x_{\text{ref}}$. Then, with x_0 an initial state of a satellite, the initial box associated with the $n\%$ potential cloud ($n = 5, 95$) is given by $x_0 + \Delta x_{n\%}$.

4.3 Scenario 1: Propagation of Uncertainties

The first scenario consists in the propagation of the 5% and 95% potential clouds of a satellite for 6 hours, using the function R defined in (1).

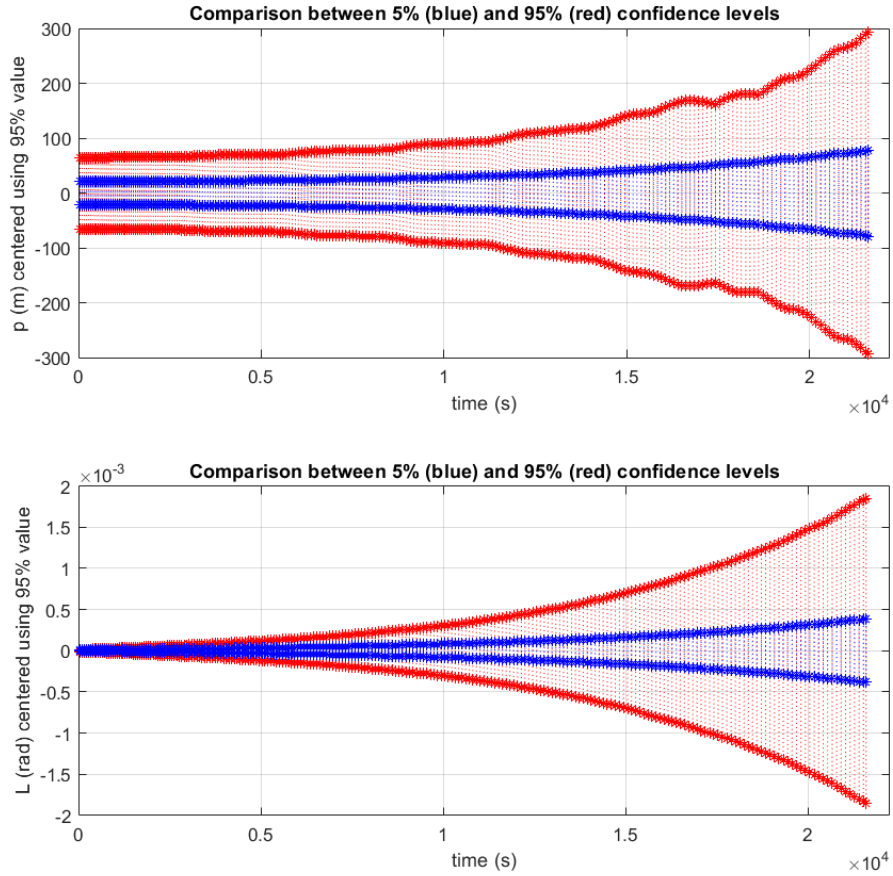


Figure 2: Propagation of the 5% (blue) and 95% (red) potential clouds on the p and L components.

Figure 2 shows the results on the components p and L. For an easier comparison between the two potential clouds, the intervals are translated using the center of the 95% interval. After 6 hours, the 95% potential cloud associated with the semi-

parameter p is about 600 meters wide, three times greater than the 5% potential cloud. Along the propagation, the order of magnitude stays the same for the two potential clouds associated with p. In contrast, the widths of the potential clouds associated with L increase faster and after 6 hours, the circular arcs corresponding to such intervals are about 7 kilometers for the 5% potential cloud and 27 km for the 95% potential cloud. The specificity of the component L is discussed in 4.4. Finally, it can be observed that, as expected, the 5% intervals are always included in the 95% intervals.

4.4 Scenario 2: Effect of Measures on Estimation Uncertainties

In this scenario, a ground radar provided an estimate of the state of the satellite at an initial time, and propagated its estimate for 12 hours with the function R defined in equation (1) before being able to measure its position again. When the satellite gets in the field of view of the radar again, a series of 10 measures of its position over 5 minutes are taken. Using our algorithm, these measures are added to the estimation process.

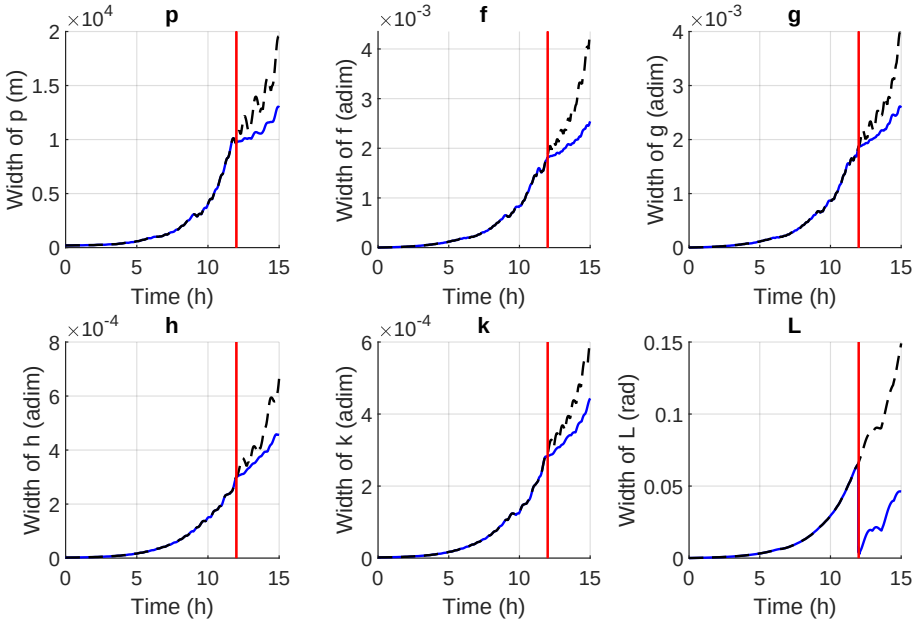


Figure 3: Propagation of the uncertainties, with (blue) and without (black) addition of 10 measures after 12 hours (red).

Figure 3 shows that L is significantly impacted by the series of measures, in comparison to other components of the state of the satellite. Indeed, the uncertainty over L drops with the addition of measures. The precision of the other components

also improves after the addition of measures, compared to the scenario where no measure was added. This comes from the precision gain of L spreading to the other state's components through the dynamics equations.

The impact of the measures on the component L is due to L encoding the position of the satellite along the orbit. Therefore, this component is strongly linked to the position of the satellite, hence the steep decrease of its uncertainties when a position measure is added. Further, the true longitude L is the component which is the most subject to variations over time, since it is the only non-constant one when the dynamics is unperturbed (12). Therefore, it makes the propagated uncertainties grow faster for the true longitude L.

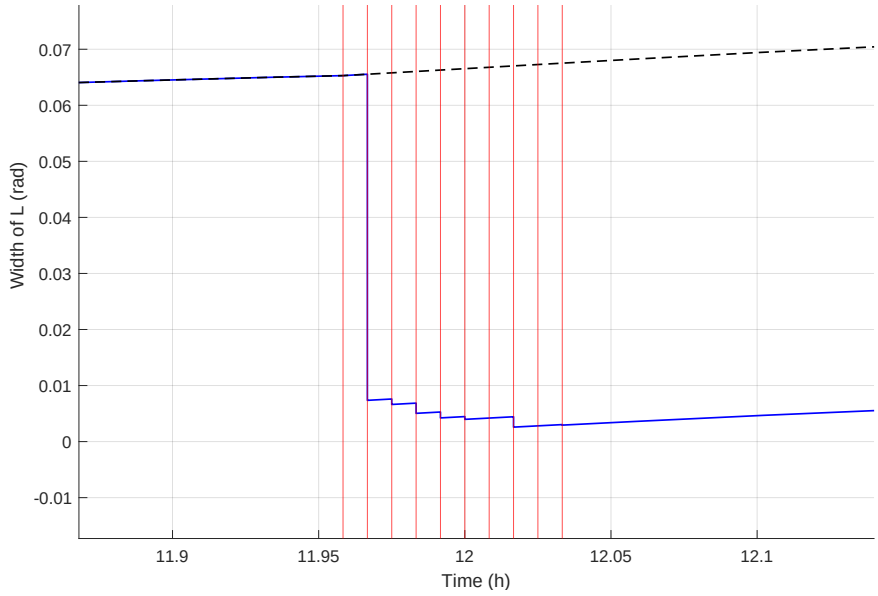


Figure 4: Focus on the effect of each measure on L.

By focusing on the impact of each measure of the series on L, it gives Figure 4, which helps raising two points. First, not every measure helps improving the estimate of the satellite's position. Here, the first measure of the series does not reduce the estimate's uncertainty. Mathematically, it means that the estimation interval was included in the measure interval. The radar is not precise enough to bring new information. Secondly, most of the precision given by the series of measures comes from the first measures. This could allow computing the number of measures that would be necessary to achieve a certain precision on the position of the tracked satellite, depending on the accuracy of the ground radar.

4.5 Scenario 3: Potential Clouds for the Collision Issue and Consequences on Sensor Design

The 5% and 95% potential clouds for two satellites are propagated using validated simulation and the $\widetilde{R}$ function (1) with the initial states:

$$\begin{cases} p_1 = 6890939, & p_2 = 6961989, \\ f_1 = 0.0014125, & f_2 = 0.0014182, \\ g_1 = -0.0014160, & g_2 = -0.0014102, \\ h_1 = 0.57763, & h_2 = -0.57763, \\ k_1 = 0.0095098, & k_2 = -0.0095068, \\ L_1 = -0.32425, & L_2 = -0.31903. \end{cases} \tag{14}$$

At last, the trajectories of the two satellites can be compared as in Figures 5a and 5b. These figures show a sample of the trajectories where the propagated boxes intersect for a 95% confidence level but not for a 5% confidence level. In that respect, the sensor performances can be evaluated in terms of required confidence level and decisions can be made depending on this analysis. These decisions can be design decisions such as improving the sensor's accuracy or multiplying the sources producing measures. In that case, if a sensor can only ensure that a collision will never happen with a 5% confidence level, one can decide to use complementary sources of measures in order to improve the estimation of the satellites states. An other decision could be to maneuver one of the satellites in order to increase the confidence level associated with the collision.

4.6 Scenario 4: Comparison with NN method

In this scenario, two close satellites are considered, but only one of them is the satellite of interest whose state needs to be estimated. Two methods are compared: a NN method [5] whose performance is evaluated with a Monte-Carlo simulation, and a set-based approach using validated simulation and interval arithmetic.

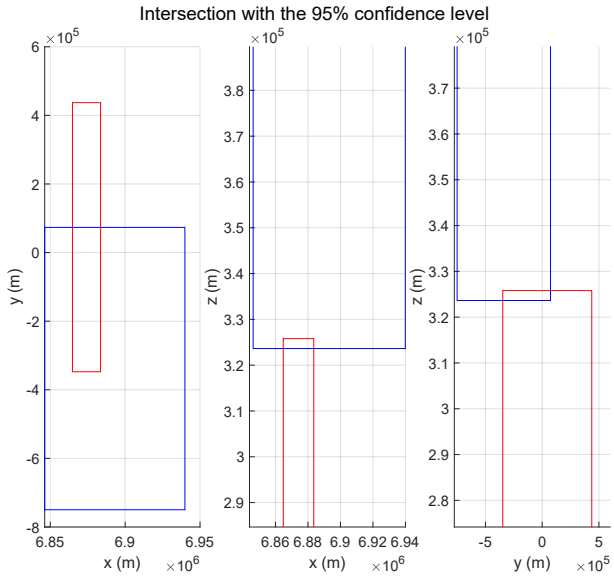
4.6.1 Scenario inputs

The true initial states of the 2 satellites are such that they belong to the same orbit, but satellite 2 is about 200 meters behind satellite 1 on this orbit:

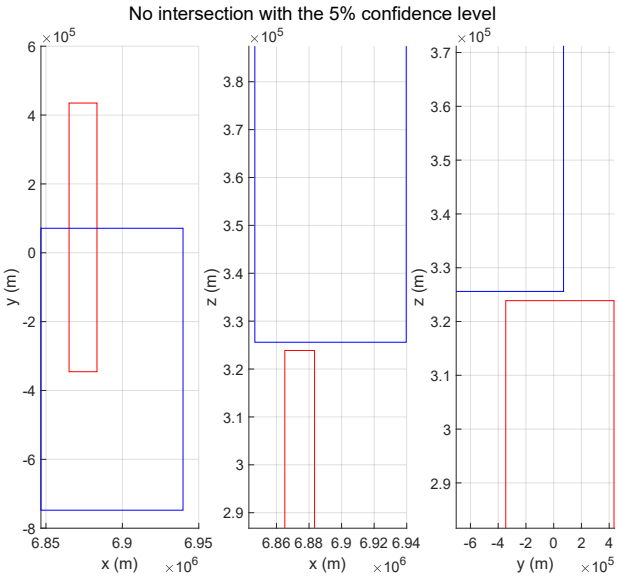
$$\begin{cases} p_1 = 6878130, & p_2 = p_1, \\ f_1 = 0.001, & f_2 = f_1, \\ g_1 = 0, & g_2 = g_1, \\ h_1 = 0.57735, & h_2 = h_1, \\ k_1 = 0, & k_2 = k_1, \\ L_1 = 0, & L_2 = -0.000029078. \end{cases} \tag{15}$$

The estimation step is assumed to produce a normally distributed estimated state $\widehat{x}_1$ of mean $x_1 = (p_1, f_1, g_1, h_1, k_1, L_1)$ and covariance matrix P:

$$\widehat{x}_1 \sim \mathcal{N}(x_1, P), \tag{16}$$



(a) Intersection for 95% potential clouds.



(b) No intersection for 5% potential clouds.

Figure 5: Position boxes at the same time in the ECI frame for the propagation of the 5% and 95% potential clouds of two satellites (blue boxes for satellite 1 and red boxes for satellite 2).

where P is chosen diagonal such that $P_{11} = 594.817$, $P_{66} = 1.258 \cdot 10^{-11}$ and $P_{ii} = 5.948 \cdot 10^{-7}$ for $i = \{2, \ldots, 5\}$. This covariance matrix is representative of the uncertainties on the state of a satellite after a few measures, with amplified uncertainties for $i = \{2, \ldots, 5\}$ in order to enhance the comparison with Monte-Carlo simulation.

4.6.2 NN method and Monte-Carlo simulation

Using (16), 1000 estimated states are randomly drawn and propagated for one hour with a Runge-Kutta method of order 4. The true states of satellites 1 ($x_{1,\text{prop}}$) and 2 ($x_{2,\text{prop}}$) are also propagated using the same method. The covariance matrix P is propagated using the formula

$$P_{\text{prop}} = JPJ^T, \tag{17}$$

where J is the numerically computed Jacobian matrix of the propagation function evaluated at x_1 (as performed in the Extended Kalman Filter [5, 13, 19, 20]). The association metric used to perform the NN method is the Mahalanobis distance [20]. For each simulated estimated state $\widehat{x}^i_{1,\text{prop}}$ and for $j \in \{1, 2\}$, this distance is computed using the formula

$$d^{i,j}_{\text{M}} = \left(x_{j,\text{prop}} - \widehat{x}^i_{1,\text{prop}}\right)^T P^{-1}_{\text{prop}} \left(x_{j,\text{prop}} - \widehat{x}^i_{1,\text{prop}}\right). \tag{18}$$

Then, for each $i \in [1, 1000]$, if $d^{i,1}_{\text{M}} < d^{i,2}_{\text{M}}$, then the ith estimate is associated the the satellite 1. Otherwise, the ith estimate is associated with the satellite 2. The same computations are made in the ECI frame, by converting every state and covariance using the formulas from [21], and (17) with the Jacobian matrix of the conversion function.

Finally, confusion rates are computed: in the mEOE frame, starting from a 0% confusion rate (i.e. every particle is correctly associated with satellite 1), a 20.3% confusion rate is obtained after a one hour propagation (see Figure 6). In the ECI frame, this phenomenon is amplified and the confusion rate is about 83.4% after a one hour propagation (see Figure 7), which means that in a large majority of cases, the NN method would perform the wrong association in this scenario. Considering 10 consecutive measures with these confusion rates, there would only be a 10.3% probability that the batch of 10 measures contains only measures of satellite 1 in the mEOE frame, and 0.0000015% in the ECI frame. Therefore, the estimates of the state of satellite 1 computed with such corrupted batch of measures would be very often corrupted as well.

4.6.3 Set-based approach

The initial states of the two satellites and the estimate of satellite 1 are propagated using validated simulation and the R function (1). The initial states of the two satellites are the same as in (15), with no uncertainty. The initial state of the

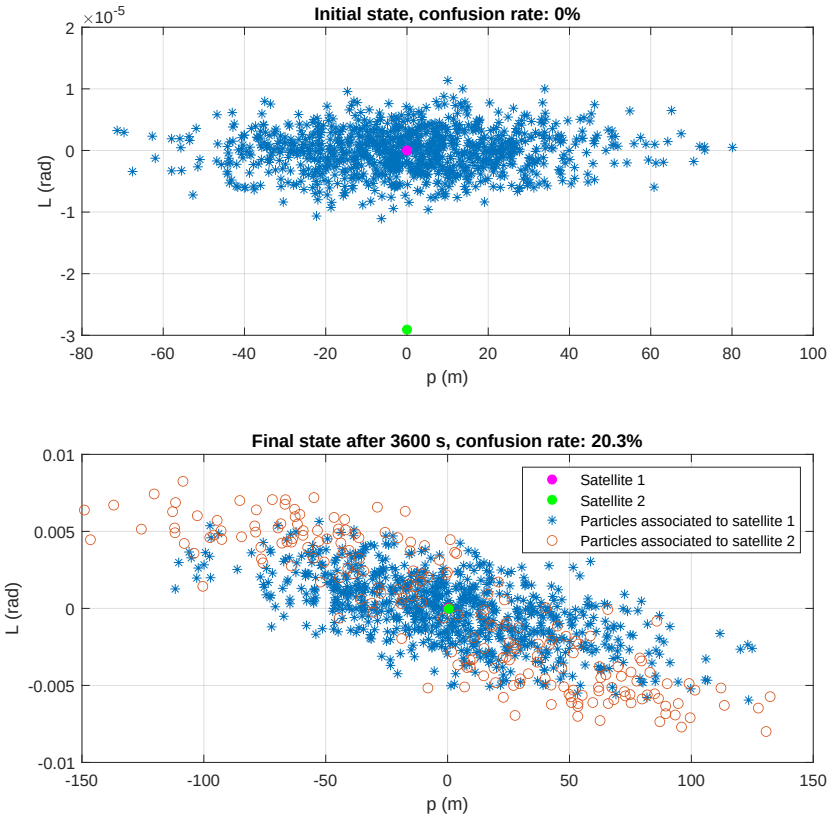


Figure 6: Confusion rate in mEOE after one hour of propagation with 1000 Monte-Carlo simulations.

estimate of satellite 1 is drawn using (16):

$$\begin{cases} \widehat{p_1} = 6878138, \\ \widehat{f_1} = 0.000935524, \\ \widehat{g_1} = -0.001727603, \\ \widehat{h_1} = 0.578592956, \\ \widehat{k_1} = 0.000283935, \\ \widehat{L_1} = 0.000004533, \end{cases} \tag{19}$$

and the corresponding box of uncertainties is computed using the confidence contractor of (2) with a 99% confidence level.

Unlike the NN method which in this case fails to associate the right satellite in about 20% of the Monte-Carlo simulations in mEOE (and 83% in ECI), the box provided by the set-based approach is guaranteed to include the true state

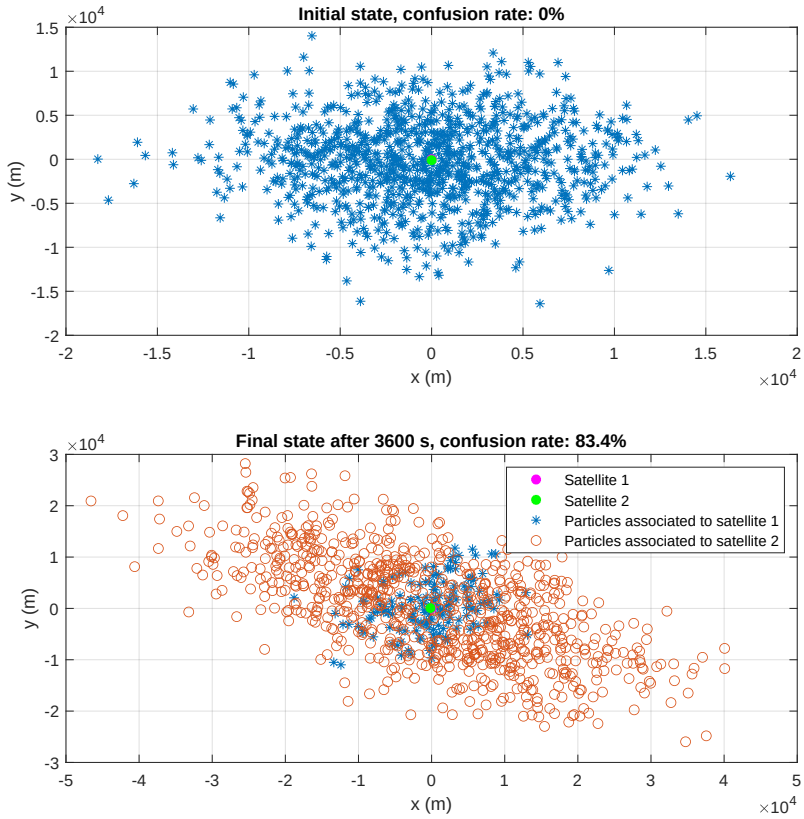


Figure 7: Confusion rate in ECI after one hour of propagation with 1000 Monte-Carlo simulations.

of satellite 1 at least 99% of the time. In this case, both true states of satellites 1 and 2 are included (see Figure 8 and Figure 9) and no association hypotheses can be definitively chosen or left out. However, it is important to keep in mind that multiple measures and backward propagation could enable to decide between these hypotheses with a certain time delay and then improve the accuracy of the state estimate. There is no need to perform a conversion of the tubes from the mEOE frame to the ECI frame here, since the inclusions are preserved by interval arithmetic, Figure 8 and Figure 9 already show that the conclusion would be the same in the ECI frame.

4.7 Discussion

The considered scenarios demonstrate the contribution of our method to various steps of the tracking of low-earth orbit satellites. Scenarios 1 and 3 emphasize

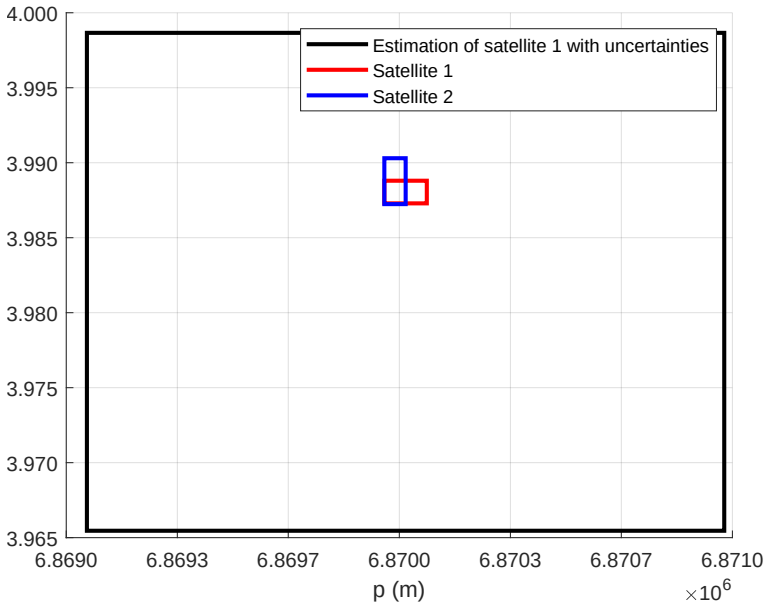


Figure 8: Propagated uncertainty boxes after one hour: satellites 1 and 2 are included in the estimate of satellite 1.

on our ability to compute trajectories of satellites using validated simulation and confidence levels: the prediction step. Scenario 2 shows an association and update step using Algorithm 1. Our set-based approach is compared to a classical NN method in scenario 4, especially in terms of confusion rate, showing the benefits of a set-based approach regarding association decision making in particular.

The missing step for a complete tracking would be the initialization of a track, i.e. the first computation of a satellite state using one or very few measures. Track initialization, also known as initial orbit determination in this context, usually relies on different techniques, such as Gauss's method [19], which remain to be studied and eventually adapted in an interval counterpart.

5 Conclusion and Future Works

In this paper, we proposed a validated approach for association and estimation of dynamical systems, taking account of uncertainties on the initial state, measures and parameters. Our approach uses measures and their uncertainties to quantify likelihood to belong to a certain track. Unlike Monte Carlo methods, our approach provides mathematically guaranteed results related to confidence levels using potential clouds and computed with contractors, provided that these confidence levels reflect the performances of the sensor. Therefore, our results do not ignore low-probability cases, while taking advantage of probability distributions to supplement

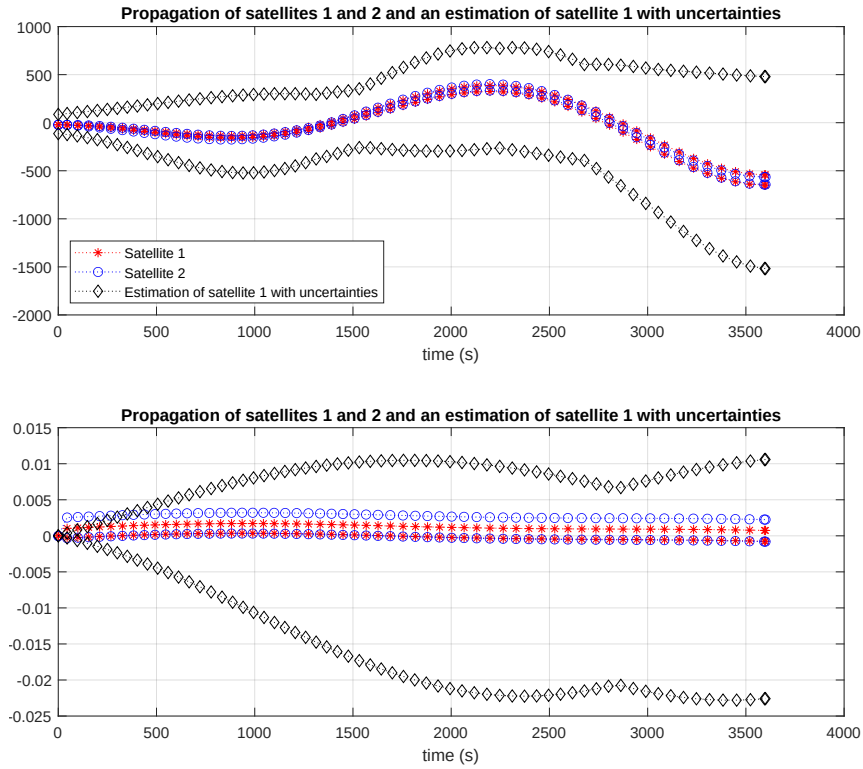


Figure 9: Centered uncertainty tubes for one hour of propagation: the tubes of satellites 1 and 2 are almost always included in the tube of the estimate of satellite 1, and always intersect with it.

the information provided by measures.

Furthermore, Monte Carlo methods require a substantial number of simulations to be reliable, compared to set-based validated methods which only need one simulation to have as much information with more guarantees. Then associated measures allow to improve the state estimate, using interval arithmetic tools such as intersection and contractors. Three scenarios showed promising results on various steps of the tracking problem, and a fourth scenario demonstrated advantages of such an approach compared to classical techniques such as NN.

As future works, implementing the retro-propagation feature of DynIbex to our algorithm would allow using the information given by a measure to reduce the uncertainty of the track beforehand. Moreover, studying the impact of a series of measures depending on the number of measures, the state of the system, the precision of the estimate and the precision of the sensor could help computing the optimal times for taking measures, to gain a maximum of information on the

system. Since operational data are available through various sources online, it would be interesting to apply our approach to such data, and then implement it on an operating ground radar. New developments could then be made in order to deal with outliers or false alarm measures which are common in this context.

References

[1] Alexandre Dit Sandretto, J. Confidence-based contractor, propagation and potential cloud for differential equations. *Acta Cybernetica*, 25(1):49–68, 2021. DOI: 10.14232/actacyb.285177.

[2] Alexandre dit Sandretto, J. and Chapoutot, A. Validated explicit and implicit Runge-Kutta methods. *Reliable Computing*, 22, 2016. URL: https://hal.science/hal-01243053/.

[3] Bar-Shalom, Y. and Tse, E. Tracking in a cluttered environment with probabilistic data association. *Automatica*, 11(5):451–460, 1975. DOI: 10.1016/0005-1098(75)90021-7.

[4] Benhamou, F., Goualard, F., Granvilliers, L., and Puget, J.-F. Revising hull and box consistency. In *Proceedings of the 16th International Conference on Logic Programming*, page 230–244. The MIT Press, 1999. DOI: 10.7551/mitpress/4304.003.0024.

[5] Blackman, S. S. and Popoli, R. F. *Design and Analysis of Modern Tracking Systems*. Artech House, 1999. ISBN: 9781580530064.

[6] Collins, J. B. and Uhlmann, J. K. Efficient gating in data association with multivariate Gaussian distributed states. *IEEE Transactions on Aerospace and Electronic Systems*, 28(3):909–916, 1992. DOI: 10.1109/7.256316.

[7] Drevelle, V. and Bonnifait, P. A set-membership approach for high integrity height-aided satellite positioning. *GPS Solutions*, 15(4):357–368, 2011. DOI: 10.1007/s10291-010-0195-3.

[8] Fortmann, T., Bar-Shalom, Y., and Scheffe, M. Sonar tracking of multiple targets using joint probabilistic data association. *IEEE journal of Oceanic Engineering*, 8(3):173–184, 1983. DOI: 10.1109/JOE.1983.1145560.

[9] Fuchs, M. and Neumaier, A. Autonomous robust design optimisation with potential clouds. *International Jorunal of Reliability and Safety*, 3:23–34, 2009. DOI: 10.1504/IJRS.2009.026833.

[10] Gill, P. E. and Murray, W. Algorithms for the solution of the nonlinear least-squares problem. *SIAM Journal on Numerical Analysis*, 15(5):977–992, 1978. DOI: 10.1137/0715063.

[11] Horwood, J. T. and Poore, A. B. Gauss von Mises distribution for improved uncertainty realism in space situational awareness. *SIAM/ASA Journal on Uncertainty Quantification*, 2(1):276–304, 2014. DOI: 10.1137/130917296.

[12] Jaulin, L., Kieffer, M., Walter, E., and Meizel, D. Guaranteed robust nonlinear estimation with application to robot localization. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 32(4):374–381, 2002. DOI: 10.1109/TSMCC.2002.806747.

[13] Julier, S. J. and Uhlmann, J. K. Unscented filtering and nonlinear estimation. *Proceedings of the IEEE*, 92(3):401–422, 2004. DOI: 10.1109/JPROC.2003.823141.

[14] Mills-Tettey, G. A., Stentz, A., and Dias, M. B. The dynamic Hungarian algorithm for the assignment problem with changing costs. Technical Report CMU-RI-TR-07-27, Robotics Institute, Pittsburgh, PA, USA, 2007. URL: https://kilthub.cmu.edu/articles/The_Dynamic_Hungarian_Algorithm_for_the_Assignment_Problem_with_Changing_Costs/6561212/files/12043517.pdf.

[15] Nedialkov, N. S., Jackson, K. R., and Corliss, G. F. Validated solutions of initial value problems for ordinary differential equations. *Applied Mathematics and Computation*, 105(1):21–68, 1999. DOI: 10.1016/S0096-3003(98)10083-8.

[16] Neyman, J. Outline of a theory of statistical estimation based on the classical theory of probability. *Philosophical Transactions of the Royal Society A*, 236(767):333–380, 1937. DOI: 10.1098/rsta.1937.0005.

[17] Reid, D. An algorithm for tracking multiple targets. *IEEE Transactions on Automatic Control*, 24(6):843–854, 1979. DOI: 10.1109/TAC.1979.1102177.

[18] Rohou, S., Desrochers, B., and Jaulin, L. Set-membership state estimation by solving data association. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4393–4399. IEEE, 2020. DOI: 10.1109/ICRA40945.2020.9197039.

[19] Vallado, D. A. *Fundamentals of Astrodynamics and Applications*. Springer Science & Business Media, 2001. ISBN: 0792369033.

[20] Verhagen, S. and Teunissen, P. J. G. Least-squares estimation and Kalman filtering. In Teunissen, P. J. G. and Montenbruck, O., editors, *Springer Handbook of Global Navigation Satellite Systems*. Springer International Publishing, 2017. DOI: 10.1007/978-3-319-42928-1_22.

[21] Walker, M. J. H., Ireland, B., and Owens, J. A set modified equinoctial orbit elements (errata 1986). *Celestial Mechanics*, 36(4):409–419, 1985. DOI: 10.1007/BF01227493.

Acta Cybernetica **27** (2026) 453–473.

Boundary Approach to Characterize the Inner and Outer Approximation of the Image of a Disk*

Maël Godard[ab], Luc Jaulin[ac], and Damien Massé[de]

Abstract

Calculating directly the inner and outer approximation of the image of a set by a function can be challenging. Then, it is sometimes preferred to compute the image of the boundary of the set instead. However, boundary-based methods are subject to the apparition of fake boundaries in the image set. These fake boundaries add pessimism when characterizing the inner approximation of the image set. This paper then introduces the notion of Box Chains to simplify the detection and the suppression of the fake boundaries. The characterization of the inner and outer approximation of the image set in the case of a function from the unit disk $\mathcal{D}$ to $\mathbb{R}^2$ will be considered, with two examples.

Keywords: fake boundaries, box chains, boundary approach

1 Introduction

Calculating the image of a set by a function has many applications in robotics [7]: Image of a subpaving [9], estimation of the area covered by a sensor [3], state estimation [1, 12]. The interval analysis tools have then been shown to be efficient for roboticians [6, 11].

The methods to calculate the image of a set by a function can be divided into two subclasses. The set-based methods compute directly an outer, and sometimes an inner, approximation of the image of the whole set. They can rely on Interval Analysis for the operations on sets. The boundary-based methods compute instead the image of the boundary of the set, and from this image an inner and outer approximation of the image set can be deduced. These methods are lighter to calculate, but they are subject to the appearance of fake boundaries in the image set.

*This work was supported by the Defense Innovation Agency (AID) and the Brittany Region.
[a]Lab-STICC, ROBEX Team, ENSTA Bretagne, France
[b]E-mail: mael.godard@ensta-bretagne.org, ORCID: 0009-0002-2822-6215
[c]E-mail: luc.jaulin@ensta.fr, ORCID: 0000-0002-0938-0615
[d]Lab-STICC, ROBEX Team, UBO, France
[e]E-mail: damien.masse@univ-brest.fr, ORCID: 0000-0002-4485-8936

DOI: 10.14232/actacyb.314985

These fake boundaries are the result of the difference between the boundary of the image set, and the image of the boundary. These fake boundaries are parts of the image of the boundary that are in facts inside of the image set and should be classified as such. As they add an unwanted pessimism to the estimation of the inner approximation of the image set, these fake boundaries have to be removed.

This problematic has often been addressed as different applications favor different solutions. The papers [2, 16] both expose the fact that the result obtained when using interval methods depends on the way the problem itself is formulated. For example the union of adjacent but non-overlapping sets is subject to the appearance of fake boundaries, whereas a reformulation of the problem can avoid this issue. In [16] a first solution is presented by using a DNF (Disjonctive Normal Form) / CNF (Conjonctive Normal Form) to prevent the fake boundaries from appearing. The paper [2] is a direct continuation of this work and proposes to rely on Karnaugh maps [8] to highlight how an outer approximation of the boundary can be constructed part by part. From this outer approximation of the boundary without the fake boundaries, an inner and outer approximation of the whole set can be obtained.

The problematic of fake boundaries can also be seen from a topological point of view. In [3], the notion of winding number is used to detect the fake boundaries. More precisely, it extends the works from [15] and [5] in order to compute the winding number, i.e. the topological degree, of a whole area. In the context of a coverage measure, the winding number represents how many times a given area was seen. The boundaries then have an interval of winding number, for example $[0, 1]$ if the boundary is between an area seen one time and an unseen area. Fake boundaries can finally be spotted as their winding number does not contain 0.

The method presented in this article is a continuation of these works as the problem considered here can easily be seen as the computation of a covered or visible area without fake boundaries. While the works presented earlier are limited to $\mathbb{R}^2$, the work presented in this article aims to be usable in higher dimensions while remaining as computationally light as possible. To do so a first step is to delete the fake boundaries before computing an inner and an outer approximation of the image set.

Section 2 presents the notions and notations that will be used for the remainder of the article. Section 3 introduces the notion of Box Chains that will be used in the boundary simplification algorithm of Section 4. For this article two examples from the unit disk $\mathcal{D}$ to $\mathbb{R}^2$ will be considered. Finally Section 5 concludes the paper.

2 Problem presentation

2.1 Notations and definitions

Let $\mathcal{D}$ be the unit disk with $\mathcal{S}^1$, the unit circle, its contour. As computing the image of the whole disk $\mathcal{D}$ can give a result too pessimistic to be usable, it is often

prefered to compute the image of its boundary $\mathcal{S}^1$ instead. For this article we will consider functions of the form $\mathbf{f} : \mathcal{D} \to \mathbb{R}^2$ that satisfies :

- $\mathbf{f}$ is $\mathcal{C}^1$, we will note $\mathbf{J_f}$ its Jacobian function.
- $\mathbf{f}$ has no singularity on $\mathcal{D}$, i.e. the determinant of its Jacobian is never null.

Thanks to these assumptions, cases where the image of the set contains a cusp or a fold, as depicted Figure 1, will not be considered.

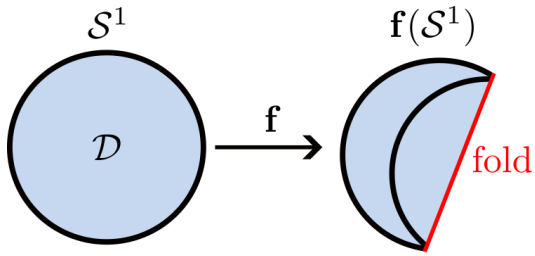


Figure 1: Fold in the image of the set

Considering a set X, we will further note ∂X its boundary. For the unit disk $\mathcal{D}$, its boundary $\partial\mathcal{D}$ is the unit circle $\mathcal{S}^1$.

We will then denote $\mathbf{f}(\mathcal{D})$ the image of the unit disk by the function $\mathbf{f}$ and $\partial\mathbf{f}(\mathcal{D})$ its boundary. Similarly, $\mathbf{f}(\mathcal{S}^1)$ will denote the image of the unit circle by the function $\mathbf{f}$.

As depicted in Figure 2a, we then have:

$$\partial\mathbf{f}(\mathcal{D}) \subseteq \mathbf{f}(\mathcal{S}^1) \tag{1}$$

Proof. Consider a function $\mathbf{f} : \mathcal{D} \to \mathbb{R}^2$. Under the assumptions cited earlier:

- $\mathbf{f}$ is $\mathcal{C}^1$.
- The determinant of its Jacobian is never null.

We can say that $\mathbf{f}$ is locally bijective at every point of $\mathcal{D}$ and is an open mapping. This mean that the image of an open set by $\mathbf{f}$ is an open set:

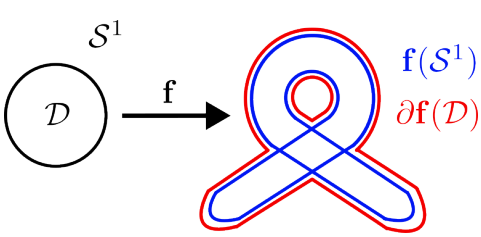
$$\forall \mathbf{x} \in \mathcal{D}, \mathbf{x} \notin \mathcal{S}^1 \implies \mathbf{f}(\mathbf{x}) \notin \partial\mathbf{f}(\mathcal{D}) \tag{2}$$

Then,

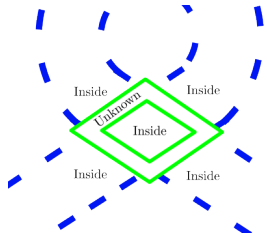
$$\forall \mathbf{y} \in \mathbb{R}^2, \mathbf{y} \in \partial\mathbf{f}(\mathcal{D}) \implies \exists \mathbf{x} \in \mathcal{S}^1, \mathbf{y} = \mathbf{f}(\mathbf{x}) \tag{3}$$

Finally, all the points of $\partial\mathbf{f}(\mathcal{D})$ are points of $\mathbf{f}(\mathcal{S}^1)$. □

Calculating the image of $\mathcal{S}^1$ instead of the image of $\mathcal{D}$ makes fake boundaries appear. These fake boundaries are defined by:



2(a) Fake boundaries



2(b) Unknown area due to the fake boundaries

Figure 2: Pessimism induced by the fake boundaries

$$FB_{\mathbf{f}}(\mathcal{D}) = \mathbf{f}(\mathcal{S}^1) \setminus \partial\mathbf{f}(\mathcal{D}) \tag{4}$$

When calculating the image of a disk, or any other closed outline, we often want to compute the inner and the outer approximation of the image set to verify if they satisfy a given condition (e.g. obstacle avoidance [4, 10]).

Given $\partial\mathbf{f}(\mathcal{D})$ it is possible to compute the inner and the outer approximation of the image set. However using $\mathbf{f}(\mathcal{S}^1)$ will give a more pessimistic result as the neighborhood of $FB_{\mathbf{f}}(\mathcal{D})$ will be considered as part of the boundary as depicted on Figure 2b.

Remark. As $\mathbf{f}$ is continuous, the image of $\mathcal{S}^1$ by $\mathbf{f}$ is a closed outline in $\mathbb{R}^2$.

For practical reasons we define the function displayed Figure 3 $\phi : [0, 2\pi] \to \mathcal{S}^1$ by :

$$\forall t \in [0, 2\pi], \phi(t) = \begin{pmatrix} cos(t) \\ sin(t) \end{pmatrix} \tag{5}$$

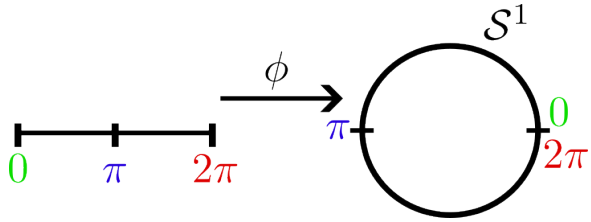


Figure 3: Function ϕ from $[0, 2\pi]$ to $\mathcal{S}^1$.

Remark. This function ϕ is bijective over $[0, 2\pi[$ and $\phi(0) = \phi(2\pi)$. This means that ϕ is bijective over any strict subset of $[0, 2\pi]$.

For the sake of simplicity, let us denote $\mathbf{g}$ the function $\mathbf{f} \circ \phi$, function from $[0, 2\pi]$ to $\mathbb{R}^2$. We then have:

$$\forall t \in [0, 2\pi], \mathbf{g}(t) = \mathbf{f}(\phi(t)) \tag{6}$$

Considering an interval $[t] \subset [0, 2\pi]$, studying the properties of $\mathbf{g}$ over $[t]$ or the properties of $\mathbf{f}$ over $\phi([t])$ will then give the same result as ϕ is bijective on this interval (see Remark 2.1). For instance, if $\mathbf{g}$ is injective over $[t]$, then $\mathbf{f}$ is injective over $\phi([t])$. For the remaining of this article the focus will then be on the function $\mathbf{g}$ to detect and remove the fake boundaries in the image of the unit disk $\mathcal{D}$ by $\mathbf{f}$. Note that as for $\mathbf{f}$, $\mathbf{g}$ is a $\mathcal{C}^1$ function and we note $\mathbf{J_g}$ its Jacobian.

Definition. *Consider a set X. A collection of sets $\{U_\alpha\}_{\alpha \in A}$ is a cover of X if*

$$X \subset \bigcup_{\alpha \in A} U_\alpha \tag{7}$$

As computing the exact image of $[0, 2\pi]$ by $\mathbf{g}$ is not possible, in this article we will consider a cover C of $[0, 2\pi]$ that does not contain $[0, 2\pi]$, and compute $[\mathbf{g}([t])]$ for each interval $[t] \in C$. The result is a set of boxes that contains the true boundary, image of $[0, 2\pi]$ by $\mathbf{g}$. Figure 4 shows the computation of the image of $[0, 2\pi]$ by $\mathbf{g}$ in the case where $\mathbf{f}$ is the identity. In this example all the intervals of C have a width of 0.1.

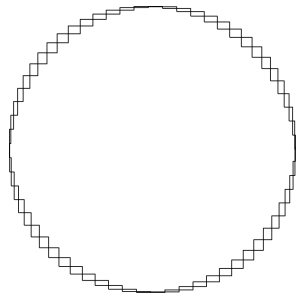


Figure 4: Image of $[0, 2\pi]$ by $\mathbf{g}$ when $\mathbf{f}$ is the identity.

2.2 Problem formulation

The objective of this article is to compute the inner and the outer approximation of the image of the unit disk $\mathcal{D}$ by a function $\mathbf{f}$ using a boundary approach. We will limit our study to the cases where the function $\mathbf{f}$ respects the assumption presented in Subsection 2.1.

As suggested in Subsection 2.1, the study of the boundary will rely on the function $\mathbf{g} : [0, 2\pi] \to \mathbb{R}^2$ defined by $\mathbf{g} = \mathbf{f} \circ \phi$, ϕ being the function defined by Equation (5).

As we are using a boundary approach, the method presented here is subject to the appearance of fake boundaries. As these fake boundaries add an unwanted pessimism to the computation of the image set, the first step is to detect and remove them.

Section 3 introduces the notion of Box Chains to decompose $[0, 2\pi]$ into subsets where $\mathbf{g}$ is injective in order to facilitate the detection of self-intersections in the

boundary. This definition will later be used in the boundary simplification algorithm presented in Section 4. Once the fake boundaries have been removed, the computation of the inner and outer approximation of the image set can be done with less pessimism.

3 Box Chains

3.1 Neighborhood relation

To define the notion of Box Chain we first need to introduce the relation of neighborhood.

Definition. *Let* $[t_i] \in \mathbb{IR}$ *and* $[t_j] \in \mathbb{IR}$ *be two intervals. We define the neighborhood relation noted* $\mathcal{R}_n$ *between* $[t_i]$ *and* $[t_j]$ *as :*

$$[t_i] \, \mathcal{R}_n \, [t_j] \Leftrightarrow [t_i] \cap [t_j] \neq \emptyset \tag{8}$$

This relation can be represented for real-value intervals in the t-plane [14] or by its logical matrix as shown Figure 5.

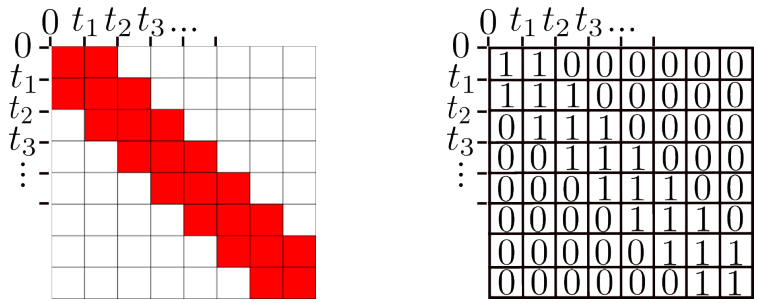


Figure 5: Neighborhood Relation in the t-plane and logical matrix.

The t-plane represents real-value intervals $[0, t_1], [t_1, t_2], ...$ on the abscissa and on the ordinate. The grid is then colored where the corresponding intervals verify the relation, and is left blank otherwise. The resulting grid can be interpreted as a logical matrix, say M, where the blank boxes are null, and the colored boxes are ones. This graphical representation of the relation highlights its properties:

- **Reflexivity** As the main diagonal of the grid has no blank box, i.e. the identity matrix is included in M, it means that the relation is reflexive.
- **Symmetry** As M is symmetric, the relation itself is symmetric.

However we can see that $(t_i \cap t_j \neq \emptyset) \wedge (t_j \cap t_k \neq \emptyset) \not\Longrightarrow t_i \cap t_k \neq \emptyset$, meaning that this relation is not transitive.

3.2 Box Chain relation

Definition. *Let* $[t_i] \in \mathbb{IR}$ *and* $[t_j] \in \mathbb{IR}$ *be two intervals and* $\mathbf{g} : \mathbb{R} \to \mathbb{R}^2$. *We define the Box Chain relation for* $\mathbf{g}$, *noted* $\mathcal{R}_{BC,\mathbf{g}}$ *between* $[t_i]$ *and* $[t_k]$, *as:*

$$[t_i]\,\mathcal{R}_{BC,\mathbf{g}}\,[t_k] \iff \exists m \in \mathbb{N}, ([t_{j_1}], \ldots, [t_{j_m}]) \in \mathbb{IR}^m, \begin{cases} [t_i]\,\mathcal{R}_n\,[t_{j_1}] \wedge \cdots \wedge [t_{j_m}]\,\mathcal{R}_n\,[t_k] \\ \mathbf{g}_{|[t_i] \cup [t_{j_1}] \cup \ldots \cup [t_{j_m}] \cup [t_k]} \text{ is injective} \end{cases} \tag{9}$$

Equivalently, $[t_i]$ *and* $[t_k]$ *are in Box Chain relation for* $\mathbf{g}$ *if a trajectory from neighbor to neighbor exists between* $[t_i]$ *and* $[t_k]$ *on which* $\mathbf{g}$ *is injective.*

The t-plane representation and the logical matrix of the Box Chain relation depends on both the expression of $\mathbf{g}$ and the chosen $t_1, t_2, \cdots$. However this relation is always **symmetric** as the neighborhood relation and the union of sets are symmetric.

3.3 Box Chain decomposition

As depicted in Figure 2a, fake boundaries appear when the considered contour crosses itself. This means that the considered function is not globally injective as two distincts inputs give the same output.

As mentioned in Section 2, the boundary we are dealing with is not a line, but a set of boxes. Finding the self intersections in the contour can then be hard as each box of the contour crosses at least two other boxes as shown Figure 6. This is the result of the continuity of the function $\mathbf{g}$.

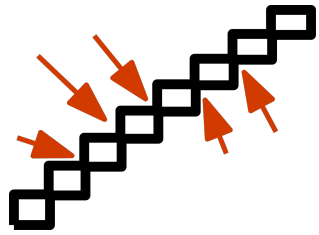


Figure 6: Intersections between the boxes.

3.3.1 Decomposition algorithm

To make this detection easier, Box Chains can be used to decompose the domain of any function into subsets on which the function is locally injective. Looking for the self-intersections in the boundary will then come down to looking for the intersections between different Box Chains. Algorithm 1 is suggested to perform this decomposition. It takes three inputs :

- The studied function $\mathbf{g}$

- An injectivity criterion h
- A cover of $[0, 2\pi]$, C

The injectivity criterion is defined over the powerset of $[0, 2\pi]$, $\mathcal{P}([0, 2\pi])$. For a given interval in $[0, 2\pi]$ it outputs 1 if **g** is injective over this interval, 0 otherwise. This criterion must be sufficient, but does not need to be necessary. An example of injectivity criterion will be given in Section 3.3.3.

The algorithm takes the element of the cover C one by one and try to group them into Box Chains. As soon as an element can not be added to the Box Chain without loosing the injectivity of **g** on it, a new Box Chain is created. Finally the algorithm sorts the intervals of the cover C into a list of Box Chains.

Algorithm 1 Box Chain decomposition.

Input : a function $\mathbf{g} : [0, 2\pi] \to \mathbb{R}^2$,
an injectivity criterion $h : \mathcal{P}([0, 2\pi]) \to \{0, 1\}$ and C a cover of $[0, 2\pi]$
Output : $\mathcal{L}_{BC}$ a list of Box Chains

Set the working list $\mathcal{L}_W := C$ and the final list $\mathcal{L}_{BC} := \{\}$
Pop E from $\mathcal{L}_W$
while ($\mathcal{L}_W \neq \emptyset$) **do**
  Initialize the Box Chain $\mathcal{L}_B := \{E\}$
  while ($\mathcal{L}_W \neq \emptyset$) **do**
    Pop E from $\mathcal{L}_W$
    if $h(\mathcal{L}_B \cup E)$ **and** $\mathcal{L}_B \mathcal{R}_n E$ **then** *injectivity criterion*
      Store E in $\mathcal{L}_B$
    else
      break
    end if
  end while
  Store $\mathcal{L}_B$ in $\mathcal{L}_{BC}$
end while
return $\mathcal{L}_{BC}$

3.3.2 Complexity

To get the best result possible, i.e. as few Box Chains as possible, the cover C will be sorted in increasing order of lower bound, noted lb below, and the popped element will be the first of the list. If we consider a cover of n elements $C = \{[c_i]\}_{i \in [\![1,n]\!]}$, we then have :

$$\forall (i, j) \in [\![1, n]\!]^2, i < j \implies lb([c_i]) \leq lb([c_j]) \tag{10}$$

This sorting and popping process allows to have successive elements in neighborhood relationship.

In terms of complexity the tests $h(\mathcal{L}_B \cup E)$ and $\mathcal{L}_B \mathcal{R}_n E$ require the computation of m jacobians and $m-1$ unions and intersections, m being the number of elements in the list $\mathcal{L}_B$. These tests then have a complexity in $\mathcal{O}(m)$.

The worse case happens when all the elements can be added to the same Box Chain except the last one. The tests mentionned earlier are then runned with $m \in [\![1, n-1]\!]$ elements. Finally, the complexity of the algorithm in the worse case is $\mathcal{O}(n^2)$.

3.3.3 Injectivity criterion

An example of injectivity criterion could rely on the tangent to the contour. The Jacobian of a function represents this tangent when defined, see Figure 7. Note that as the domain of the studied function $\mathbf{g}$ is $\mathbb{R}$, its Jacobian is a vector.

Theorem. *Let $\mathbb{T}$ be a subset of $\mathbb{R}$, $\mathbf{g} : \mathbb{R} \to \mathbb{R}^2$ be a $\mathcal{C}^1$ function and $\mathbf{J_g} : \mathbb{R} \to \mathbb{R}^2$ its Jacobian. If $0_{\mathbb{R}^2} \notin \left[\bigcup_{t \in \mathbb{T}} \mathbf{J_g}(t)\right]$, then $\mathbf{g}$ is injective over $\mathbb{T}$.*

Proof. Let us demonstrate the contrapositive.

If $\mathbf{g}$ is not injective, $\exists (t_1, t_2) \in \mathbb{T}^2$, $t_1 \neq t_2$ so that $\mathbf{g}(t_1) = \mathbf{g}(t_2)$ as shown Figure 7. Let us denote $g_1 : \mathbb{R} \to \mathbb{R}$ and $g_2 : \mathbb{R} \to \mathbb{R}$ the functions that give respectively the first and second component of $\mathbf{g}$. t_1 and t_2 then verify

$$\forall i \in \{1, 2\}, g_i(t_1) = g_i(t_2) \tag{11}$$

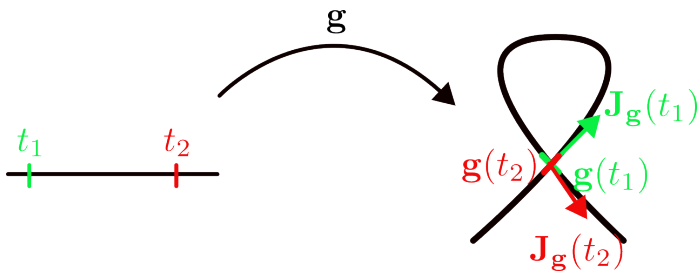


Figure 7: Loop in $\mathbb{R}^2$, $\mathbf{g}$ is not injective.

The mean value theorem can then be applied to each of the g_i functions, giving

$$\exists \tau_i \in [t_1, t_2], J_{g,i}(\tau_i) = 0 \tag{12}$$

Meaning that

$$\forall i \in \{1, 2\}, 0 \in J_{g,i}([t_1, t_2]) \tag{13}$$

Then,

$$0_{\mathbb{R}^2} \in J_{g,1}([t_1, t_2]) \times J_{g,2}([t_1, t_2]) \tag{14}$$

Finally per definition of the cartesian product and as shown Figure 8,

$$J_{g,1}([t_1,t_2]) \times J_{g,2}([t_1,t_2]) = \left[\bigcup_{t\in[t_1,t_2]} \mathbf{J_g}(t) \right] \tag{15}$$

As $(t_1,t_2) \in \mathbb{T}^2$, $[t_1,t_2] \subset \mathbb{T}$. Then $0_{\mathbb{R}^2} \in \left[\bigcup_{t\in\mathbb{T}} \mathbf{J_g}(t) \right]$.

Finally, we get that if $\mathbf{g}$ is not injective over $\mathbb{T}$, $0_{\mathbb{R}^2} \in \left[\bigcup_{t\in\mathbb{T}} \mathbf{J_g}(t) \right]$.

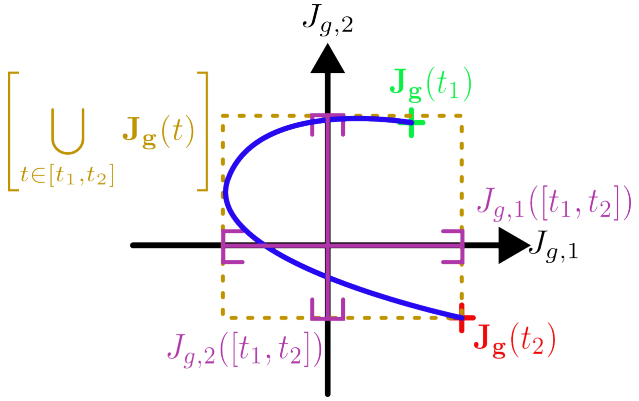


Figure 8: The brown box contains the origin, we can't conclude on the injectivity.

□

4 Determination of the inner and outer approximation of the image a disk

To limit the pessimism while determining the inner and outer approximation of the image of the whole set, the first step is to remove the fake boundaries. To do so, Box Chains presented in the previous section will be used to detect the boxes that belong to the self intersections in the boundary. Once these boxes have been set aside, the fake boundary can be removed before computing the inner and outer approximation of the image set.

4.1 Boundary Simplification algorithm

In this section, the function $\mathbf{f}$ defined in Equation (16) will be considered for the illustrations. However the algorithm presented works with any function as long as the assumptions from Section 2 are satisfied.

$$\forall \mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \in \mathcal{D}, \mathbf{f}(\mathbf{x}) = \begin{pmatrix} x_1^2 - x_2^2 + x_1 \\ 2x_1x_2 + x_2 \end{pmatrix} \tag{16}$$

As explained in Section 2, when studying this function on its boundary $\mathcal{S}^1$, we will work with the function $\mathbf{g} : [0, 2\pi] \to \mathbb{R}^2$ for illustration purposes. It is defined by:

$$\forall t \in [0, 2\pi], \mathbf{g}(t) = \begin{pmatrix} cos(t)^2 - sin(t)^2 + cos(t) \\ 2cos(t)sin(t) + sin(t) \end{pmatrix} \tag{17}$$

If we consider a cover C of $[0, 2\pi]$ where each element has a diameter of 0.01, Figure 9 shows the image of C by the function $\mathbf{g}$ of Equation (17) with and without fake boundaries. As mentioned earlier, the boundary is here a set of boxes that constitutes an over-approximation of the real boundary.

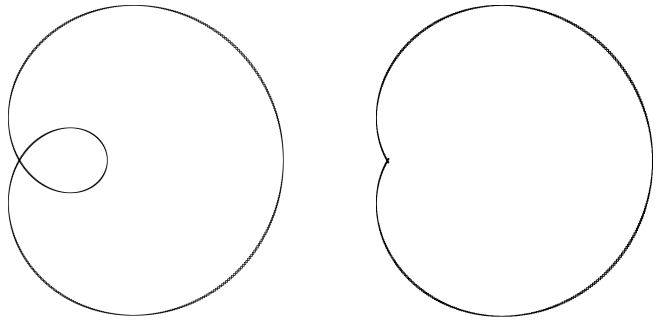


Figure 9: Boundary with (left) and without (right) fake boundaries.

The different steps to eliminate the fake boundaries in the image of C by $\mathbf{g}$ are described below.

4.1.1 Step 1: Box Chain decomposition

The first step before removing the fake boundaries is to look for the self-intersections in the boundary. Indeed the boxes that belong to the intersections have to be kept in the resulting boundary.

The first step is then to decompose the cover C into Box Chains as suggested in Subsection 3.3. This decomposition will make the detection of the fake boundaries easier. Algorithm 1 can be used to this end and Theorem 3.3.3 gives an injectivity criterion h:

$$\forall [t] \subset [0, 2\pi], h([t]) = \begin{cases} 0 \; if \; \mathbf{0}_{\mathbb{R}^2} \in [\mathbf{J}_\mathbf{g}([t])] \\ 1 \; otherwise \end{cases} \tag{18}$$

C can then be decomposed into Box Chains thanks to Algorithm 1 to get the result shown on Figure 10. As expected the self intersections in the boundary appear between different Box Chains.

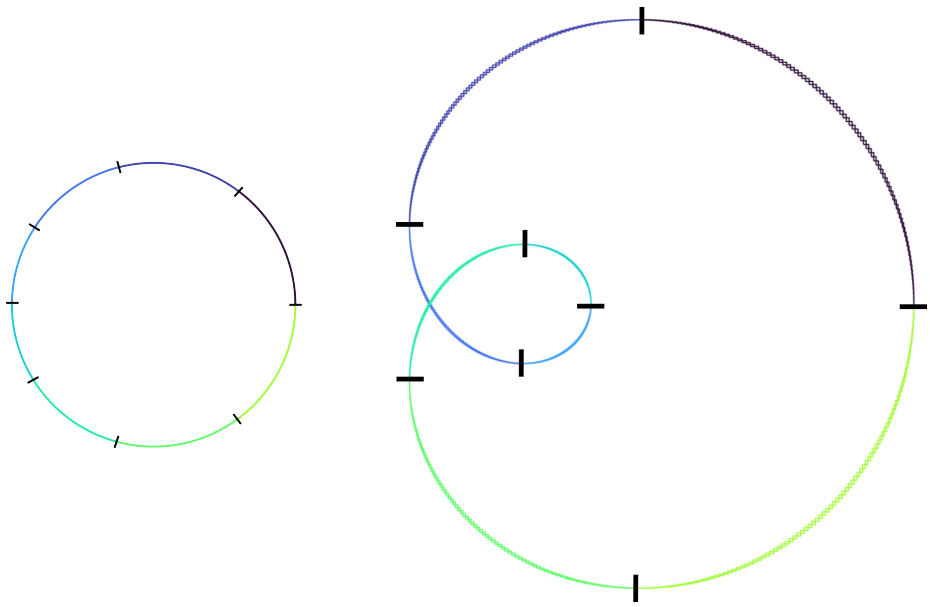


Figure 10: Eight Box Chains in $\mathcal{S}^1$ (left) and their image by $\mathbf{g}$ (right).

4.1.2 Step 2: Finding the self intersections

Once the Box Chain decomposition of C has been done, we can look for intersections between the images of the Box Chains by $\mathbf{g}$. Two cases of intersections can be observed Figure 11.

- If two Box Chains are in neighborhood relation, their images by $\mathbf{g}$ intersect each other at their junction point.
- If fake boundaries exist, the boundary crosses itself and the image of two different Box Chains intersect each other.

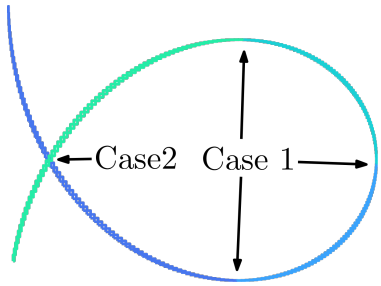


Figure 11: Two cases of intersections between the images of two Box Chains.

Remark. The neighborhood relation can be extended to Box Chains : Let there be two Box Chains B_1 and B_2, $B_1 \mathcal{R}_n B_2 \Leftrightarrow \exists([t_1],[t_2]) \in B_1 \times B_2, [t_1]\mathcal{R}_n[t_2]$

The injectivity criterion h can be used to distinguish these two cases. Indeed in the first case the function **g** is injective around the junction point between the two Box Chains, which is not the case when the boundary really crosses itself. Algorithm 2 sums up this step. When applied, this algorithm outputs a list of intervals where the studied function **g** is not injective. The corresponding intersections can be then computed as visible in red on Figure 12.

Algorithm 2 Finding self intersections.

Input : a function $\mathbf{g} : [0, 2\pi] \to \mathbb{R}^2$, a list of Box Chains $\mathcal{L}_{BC}$ and an injectivity criterion $h : \mathcal{P}([0, 2\pi]) \to \{0, 1\}$

Output : a list of intervals $\mathcal{L}_I$

Set the working list $\mathcal{L}_I := \{\}$.
for $i = 1$ to $len(\mathcal{L}_{BC})$ **do**
  for $j = i + 1$ to $len(\mathcal{L}_{BC})$ **do**
    for t_i in $(\mathcal{L}_{BC}[i])$ **do**
      for t_j in $(\mathcal{L}_{BC}[j])$ **do**
        if $\mathbf{g}(t_i) \cap \mathbf{g}(t_j) \neq \emptyset$ **and not** $h(t_i \cup t_j)$ **then** *self intersection*
          Store individually t_i and t_j in $\mathcal{L}_I$
        end if
      end for
    end for
  end for
end for
return $\mathcal{L}_I$

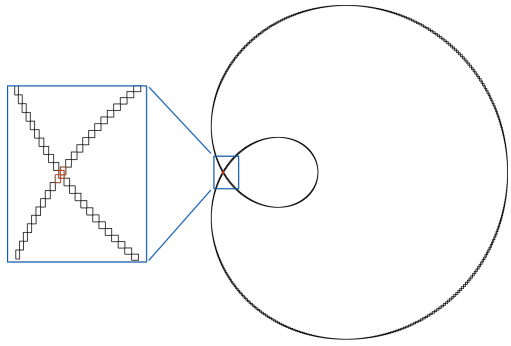


Figure 12: Self intersection in the boundary (in red).

4.1.3 Step 3: Determination of the inner areas

To determine the inner approximation of the image set an initial paving of the set $S = \bigcup_{[t]\in C} [\mathbf{g}([t])]$ is needed. To achieve this the SIVIA algorithm [7] was used with an accuracy of 0.04 to get the result shown on Figure 13. By doing so, the union of the yellow boxes is an outer approximation of S, and each yellow box has a width smaller than 0.04.

The blue boxes then represent the complementary of the yellow boxes. Their union is an inner approximation of the complementary of S. They can be divided into connected subsets as shown on Figure 13. For this work we will rely on the connected subset decomposition implemented in the Codac library [13].

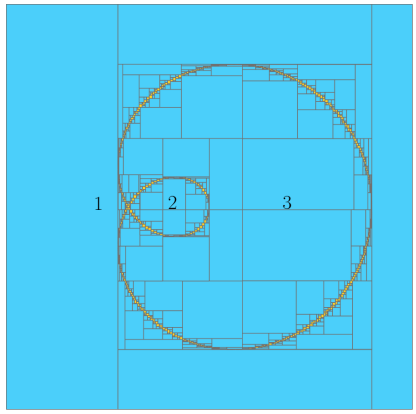


Figure 13: Paving with three connected subsets.

Then instead of qualifying each box individually as inside or outside the image set, we can qualify the whole connected subset. This means that once a box is proven to be inside, the corresponding connected subset can directly be classified as inside.

A solution to distinguish the boxes that are inside and outside is to look at the normal to the boundary. As depicted Figure 15 the normal of a real boundary points towards the exterior. This means that the opposite corner is inside.

Remark. Note that in the case of a fake boundary the normal vector points toward the interior of the set, but the opposite corner is still inside. This is due to the fact that a fake boundary is inside the set it should bound.

Remark. For a given interval $[t] \in [0, 2\pi]$ the tangent to the oriented boundary evaluated in $[\mathbf{g}]([t])$ belongs to $[\mathbf{J_g}([t])]$, and rotating $[\mathbf{J_g}([t])]$ by $-\frac{\pi}{2}$ gives a box containing the normal to the boundary. Note that the sign of the rotation depends on the orientation chosen for the unit circle (here counterclockwise). Figure 14 displays the oriented unit circle with a tangent and the associated normal vector.

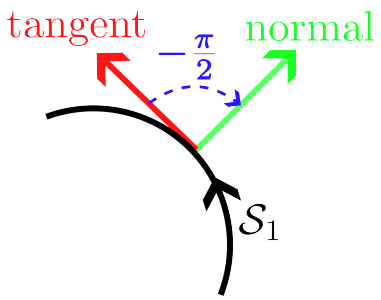


Figure 14: A tangent and the associated normal to the unit circle.

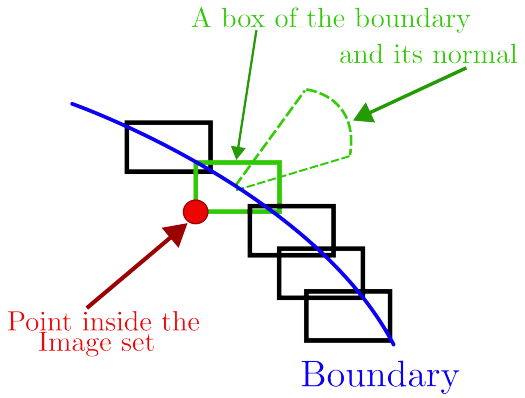


Figure 15: Determination of the inner areas.

Applying this to every box of the boundary, except the ones that were previously proven to belong to the self intersections, gives a set of points that are inside. The connected subsets containing at least one of these points can be marked as inside, and the other can be marked as outside. Figure 16 shows an example of output for this step.

4.1.4 Step 4: Suppressing the fake boundaries

As mentioned earlier, the normal to the boundary is supposed to point outwards. Figure 17 illustrates the fact that in the case of fake boundaries this normal points towards an area that was classified as inside in the last step. This can be seen as the fact that the normal of a fake boundary aims towards a fake exterior.

Suppressing the boxes with a normal pointing towards an inside area allows us to remove the fake boundaries to obtain a less pessimistic approximation of $\partial\mathbf{f}(\mathcal{D})$ as visible on Figure 18. Note that as the self-intersections in the contour have been detected in Step 2, it is possible to propagate the information of a box belonging to the fake boundary from neighbor to neighbor until an intersection is reached.

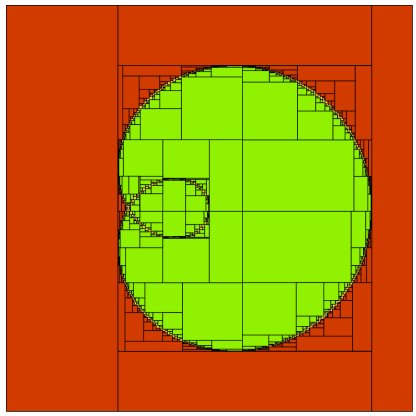


Figure 16: Inner (green) and outer (green + yellow) approximation of the image set with fake boundaries.

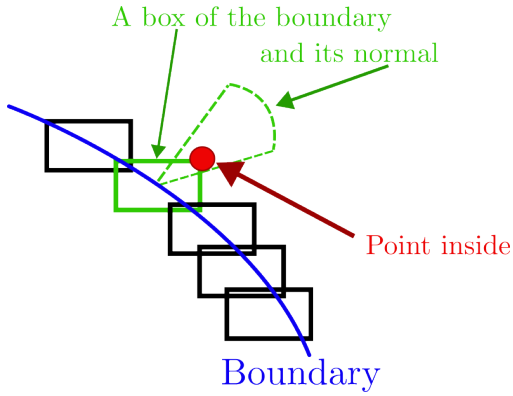


Figure 17: Case of a fake boundary.

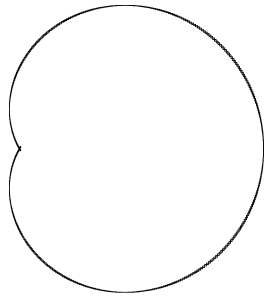


Figure 18: Fake boundaries removed.

4.2 Inner and outer approximation of the image set

Once the fake boundaries have been removed, Step 3 can be applied again with the remaining boxes to get the inner and outer approximation of the image set without fake boundaries.

To do so if we denote by C_r the set of remaining intervals of the cover C after the boundary simplification, we first proceed to an initial paving of $S = \bigcup_{[t]\in C_r} [\mathbf{g}([t])]$ by using the SIVIA algorithm a second time. Then the boxes that are not part of the boundary are sorted into connected subsets and are finally classified as inside or outside the image set with the criteria illustrated Figure 15.

Finally, we are able to characterize the inner and outer approximation of the image of the unit disk. Figure 19 shows the result obtained with the function $\mathbf{f}$ defined by Equation (16) and an initial cover where all the elements have a diameter of 0.01. The whole process, from the Box Chain decomposition to the final result took approximatly 0.6 seconds.

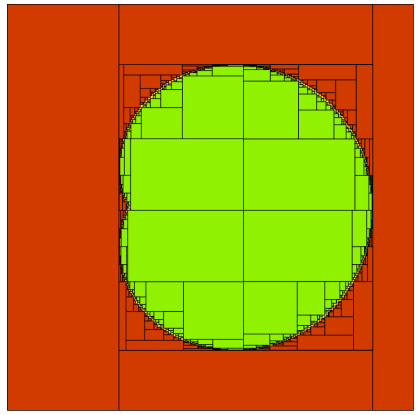


Figure 19: Inner (green) and outer (green + yellow) approximation of the image set without fake boundaries.

4.3 Additional example

The algorithm presented here also work in more complex cases. Algorithm 3 gives another example of a function $\mathbf{f} : \mathcal{D} \to \mathbb{R}^2$.

As earlier a function $\mathbf{g} : [0, 2\pi] \to \mathbb{R}^2$ can be defined by $\forall t \in [0, 2\pi], \mathbf{g}(t) = \mathbf{f}(\phi(t))$. Figure 20 shows the image of the unit circle $\mathcal{S}^1$ by this function $\mathbf{f}$. This result is obtained by considering a cover C of $[0, 2\pi]$ and computing $[\mathbf{g}([t])]$ for each interval $[t] \in C$. For this application each element of the cover has a diameter of 0.005.

With this function, Algorithm 1 gave 15 Box Chains as shown on Figure 21.

Algorithm 3 Pseudocode of the implementation of the **f** function.

Define $s(\tau, a, b, c) = a + (b - a) \cdot 0.5 \cdot (1 + \tanh(10 \cdot (\tau - c)))$

Input: $\mathbf{p} = (p_1, p_2)$ a point in the unit disk

$\rho = \sqrt{p_1^2 + p_2^2}$
$\alpha = arctan2(p_2, p_1)$
$\tau = 0.03 + \alpha \cdot \frac{1.01}{2\pi}$
$t_1 = 1 - \cos(2\pi\tau)$
$\mathbf{d} = \begin{pmatrix} 5 - 50 \cdot \cos(5 \cdot t_1) \\ 30 \cdot \sin(5 \cdot t_1) \end{pmatrix}$
$\theta = s(\tau, -3 \cdot \pi/2, -\pi/2, 0) + s(\tau, 0, \pi, 0.5) + s(\tau, 0, \pi, 1)$
$\mathbf{y} = \begin{pmatrix} 5 \cdot t_1 - 5 \cdot \sin(5 \cdot t_1) \\ 2 - 3 \cdot \cos(5 \cdot t_1) \end{pmatrix} + \frac{\rho}{\sqrt{d_1^2 + d_2^2}} \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \mathbf{d}$

Output: $\mathbf{y}$

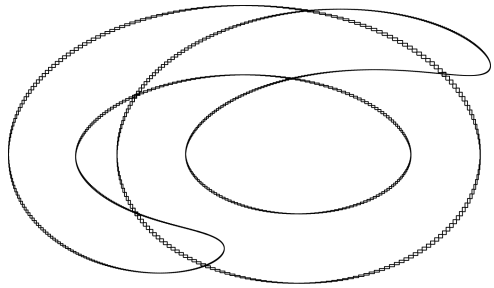


Figure 20: Boundary with fake boundaries.

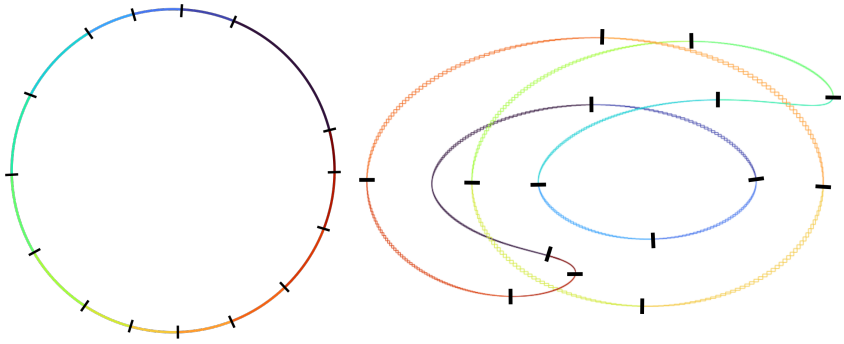


Figure 21: Fifteen Box Chains in $\mathcal{S}^1$ (left) and their image by **g** (right).

Thanks to this Box Chain decomposition, we were able to detect the self-intersections in the boundary as shown in red on Figure 22.

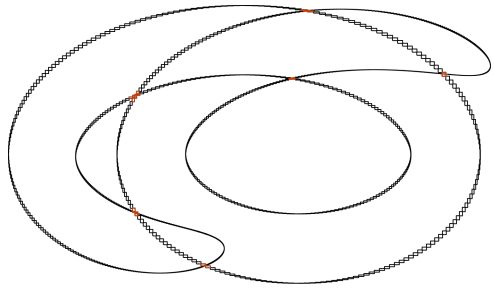
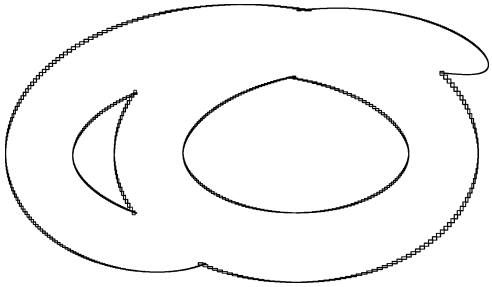


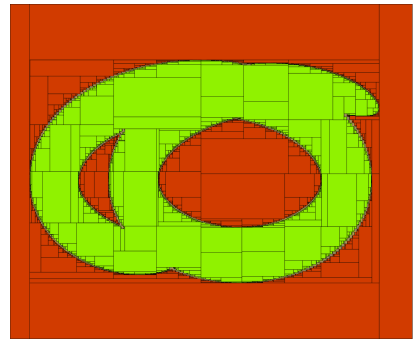
Figure 22: Self-intersections in the boundary (in red).

Finally, the fake boundaries are detected and removed before computing the inner and outer approximation of the image set without fake boundaries as displayed on Figures 23a and 23b.

For this application the whole process took 14 seconds. This increase in the computation time is due to two factors. First the higher resolution of the cover leads to an increase in the computation time as the Box Chain decomposition algorithm has a complexity in $\mathcal{O}(n^2)$ (see Section 3.3.2). In addition, the area to pave in this example is larger while the same accuracy was used in the SIVIA algorithm. Indeed in the first example the area to cover is a 5x5 square, whereas in this example the area is a 16x12 rectangle. As the SIVIA paving is called twice, this second factor influences even more the computation time.



23(a) Fake boundaries removed



23(b) Inner (green) and outer (green + yellow) approximation of the image set

Figure 23: Fake boundaries are detected and removed before computing the inner and outer approximation of the image set without fake boundaries

5 Conclusion

This paper presented the problematic of fake boundaries in the computation of the image of a set by a function. A method to remove them in the case of a function from $\mathcal{D}$ to $\mathbb{R}^2$ was presented. The method was finally applied with two examples where it was indeed able to remove the fake boundaries.

The notion of Box Chains was introduced with the definition of the neighborhood and the Box Chain relations. This Box Chain relation relies on an injectivity criterion and an example for the case of a function from $\mathbb{R}$ to $\mathbb{R}^2$ was proposed. These Box Chains can be used to make the detection of self-intersections in the boundary easier. In addition, a Box Chain decomposition algorithm in $\mathcal{O}(n^2)$ was presented.

Finally a boundary simplification algorithm was displayed. It relies on Box Chains to facilitate the detection of self intersections in the contour. Once the fake boundaries have been removed, a better inner approximation of the image set can be obtained.

References

[1] Alamo, T., Bravo, J. M., and Camacho, E. F. Guaranteed state estimation by zonotopes. *Automatica*, 41(6):1035–1043, 2005. DOI: 10.1016/j.automatica.2004.12.008.

[2] Brateau, Q., Le Bars, F., and Jaulin, L. Considering adjacent sets for computing the visibility region. *Acta Cybernetica*, 2025. DOI: 10.14232/actacyb.314640.

[3] Costa Vianna, M., Goubault, E., Jaulin, L., and Putot, S. Estimating the coverage measure and the area explored by a line-sweep sensor on the plane. *International Journal of Approximate Reasoning*, 169:109162, 2024. DOI: 10.1016/j.ijar.2024.109162.

[4] Dabadie, C., Kaynama, S., and Tomlin, C. J. A practical reachability-based collision avoidance algorithm for sampled-data systems: Application to ground robots. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4161–4168. IEEE, 2014. DOI: 10.1109/IROS.2014.6943149.

[5] Franek, P. and Ratschan, S. Effective topological degree computation based on interval arithmetic. *Mathematics of Computation*, 84(293):1265–1290, 2015. DOI: 10.1090/S0025-5718-2014-02877-9.

[6] Guyonneau, R., Lagrange, S., Hardouin, L., and Lucidarme, P. Guaranteed interval analysis localization for mobile robots. *Journal on Advanced Robotics*, 28(16):1067–1077, 2014. DOI: 10.1080/01691864.2014.908742.

[7] Jaulin, L., Kieffer, M., Didrit, O., and Walter, E. *Applied Interval Analysis: With Examples in Parameter and State Estimation, Robust Control and Robotics*. Engineering Online Library. Springer-Verlag, London, 2001. DOI: 10.1007/978-1-4471-0249-6.

[8] Karnaugh, M. The map method for synthesis of combinational logic circuits. *Transactions of the American Institute of Electrical Engineers, Part I: Communication and Electronics*, 72(5):593–599, 1953. DOI: 10.1109/TCE.1953.6371932.

[9] Kieffer, M., Jaulin, L., Braems, I., and Walter, E. Guaranteed set computation with subpavings. *Scientific Computing, Validated Numerics, Interval Methods*, pages 167–178, 2000. DOI: 10.1007/978-1-4757-6484-0_14.

[10] Malone, N., Chiang, H.-T., Lesser, K., Oishi, M., and Tapia, L. Hybrid dynamic moving obstacle avoidance using a stochastic reachable set-based potential field. *IEEE Transactions on Robotics*, 33(5):1124–1138, 2017. DOI: 10.1109/TRO.2017.2705034.

[11] Merlet, J.-P. Interval analysis and robotics. In *Robotics Research*, Volume 66 of *Springer Tracts in Advanced Robotics*, pages 147–156, 2007. DOI: 10.1007/978-3-642-14743-2_13.

[12] Rego, B. S., Raffo, G. V., Scott, J. K., and Raimondo, D. M. Guaranteed methods based on constrained zonotopes for set-valued state estimation of nonlinear discrete-time systems. *Automatica*, 111:108614, 2020. DOI: 10.1016/j.automatica.2019.108614.

[13] Rohou, S., Desrochers, B., and Le Bars, F. The Codac Library. *Acta Cybernetica*, 26(4):871–887, 2024. DOI: 10.14232/actacyb.302772.

[14] Rohou, S., Jaulin, L., Mihaylova, L., Le Bars, F., and Veres, S. M. *Reliable Robot Localization: A Constraint-Programming Approach Over Dynamical Systems*. Wiley-ISTE, 1 edition, 2020. ISBN: 9781848219700.

[15] Stenger, F. Computing the topological degree of a mapping in $\mathbb{R}^n$. *Numerische Mathematik*, 25(1):23–38, 1975. DOI: 10.1007/BF01419526.

[16] Welte, A., Jaulin, L., Ceberio, M., and Kreinovich, V. Avoiding fake boundaries in set interval computing. *Journal of Uncertain Systems*, 11(2):137–148, 2017. URL: https://ensta-bretagne.hal.science/hal-01698416/file/jusVol11No2paper07.pdf.

Acta Cybernetica **27** (2026) 475–492.

Considering Adjacent Sets for Computing the Visibility Region*

Quentin Brateau[ab], Fabrice Le Bars[ac], and Luc Jaulin[ad]

Abstract

This paper explores the problem of the paving of the union of adjacent contractors. The focus is first put on the analysis of the topology of a set operator, which can be stable or not stable. Then, depending on the stability of the union operator, solutions are proposed to avoid fake boundaries in stable and non-stable union of sets. For stable unions of sets, a boundary preserving form will be developed to add a set overlapping the fake boundary in the expression of the union, whereas for non-stable union of sets, a boundary approach will be developed to avoid fake boundaries. Some problem-specific solutions are also developed to avoid fake boundaries. As an example, an enhancement of the separator on the visibility constraint is proposed. This avoids fake boundaries while characterizing the set of non-visible points from an observation point relative to a polygon.

Keywords: set methods, interval analysis, contractors, set inversion, topology

1 Introduction

Interval Arithmetic and contractor programming have emerged as powerful tools in the field of robotics [16, 5, 10, 8], offering robust methods for handling uncertainty and performing set-based computations. These techniques have been widely applied in robotics area such as in localization [5, 11], in path planning [3, 6], and in control of systems [19, 14].

Interval analysis is a subset of set methods where sets are represented by intervals. Some operators are defined in classical set theory, such as union, intersection, complementary of sets and so on [17]. As intervals are representing sets, these operators are also defined for intervals by interval arithmetic [8].

*This work has been supported by the French Government Defense procurement and technology agency (AID)

[a]LabSTICC UMR 6285, Robex team, ENSTA Bretagne, 2 rue Francois Verny, 29200, Brest, France

[b]E-mail: quentin.brateau@ensta-bretagne.org, ORCID: 0000-0002-1553-9549

[c]E-mail: fabrice.le_bars@ensta-bretagne.fr, ORCID: 0000-0001-9413-4621

[d]E-mail: lucjaulin@gmail.com, ORCID: 0000-0002-0938-0615

DOI: 10.14232/actacyb.314640

Contractors are a mathematical function acting on intervals. They are used to contract a domain of feasible values relative to a constraint. Denoting by $\mathbb{IR}^n$ the set of axis-aligned boxes of $\mathbb{R}^n$, the operator $\mathcal{C} : \mathbb{IR}^n \to \mathbb{IR}^n$ is a contractor for $\mathbb{X} \subseteq \mathbb{R}^n$ if it meets the condition Equation (1).

$$\forall [\mathbf{x}] \in \mathbb{IR}^n, \begin{cases} \mathcal{C}([\mathbf{x}]) \subseteq [\mathbf{x}] & (\textit{Contractance}) \\ \mathcal{C}([\mathbf{x}]) \cap \mathbb{X} = [\mathbf{x}] \cap \mathbb{X} & (\textit{Completeness}) \end{cases} \tag{1}$$

Figure 1 shows a graphical representation of a contractor. The contractor $\mathcal{C}_{\mathbb{A}}$ is contracting the provided box $[\mathbf{b}]$ relative to the set $\mathbb{A}$ in green. The resulting box is shown in purple and represent the contracted box $\mathcal{C}_{\mathbb{A}}([\mathbf{b}])$. To meet the conditions of Equation (1), the contractor is only able to remove points outside the set $\mathbb{A}$.

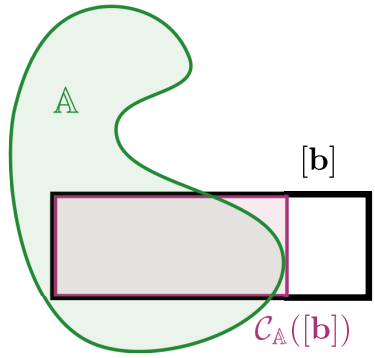


Figure 1: Graphical representation of a contractor

A contractor represents a set as defined in [2]. Set operators are then defined for contractors as it is for sets. Contractors can be combined by computing their union, their intersection, their cartesian product, and so on.

However, while most of these operations are well-defined and straightforward to implement, the union operation presents unique challenges, particularly when dealing with non-overlapping sets. The union of adjacent sets (i.e. sets that share a common boundary) using contractors can sometimes result in the appearance of fake boundaries at the interface of these sets. This phenomenon, was first highlighted in [20], and a solution was proposed using appropriate Disjunctive Normal Forms (DNF) and Conjunctive Normal Forms (CNF) [1] to avoid these fake boundaries. However, this solution is not always applicable as sets are not always defined as unions and intersections of sets.

For instance, the set of visible points from an observation point relative to an obstacle is defined and implemented in [4]. By paving this set, fake boundaries may

appear in the non-visible area. Figure 2 shows an example of the set of visible points from an observation point relative to a polygon obstacle. The observation point is shown in red. The set of visible points from this observation point is shown in blue, the set of non-visible points is shown in pink, and the set of uncertain points is shown in yellow. The obstacle is outlined in black. Fake boundaries appear within the pink area, as lines of yellow boxes in the non-visible area.

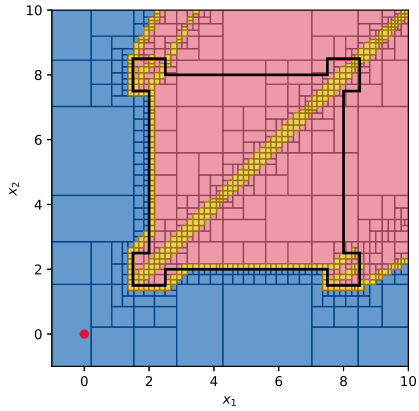


Figure 2: Separator on the visibility constraint

This article aims to address the problem of union operations on adjacent contractors, with a focus on eliminating fake boundaries. By developing new techniques for performing union operations on adjacent sets, we seek to minimize the added pessimism to the results, to improve the efficiency of the paving algorithm, and to enhance the accuracy and reliability of set-based computations in robotic applications. Added pessimism comes from bad classified boxes around the fake boundary. These uncertain boxes might suggest a boundary, but they are clearly inside the considered set.

This paper is organized as follows. Section 2 present the problem of the union of adjacent contractors by an introducing example. Then, Section 3 analyze the problem from a topological point of view, and distinguish stable and non-stable set operators. Section 4 and Section 5 present the solutions to avoid fake boundaries in both cases. Section 6 presents an application of the boundary approach on the separator on the visibility constraint. Finally, Section 7 concludes the paper.

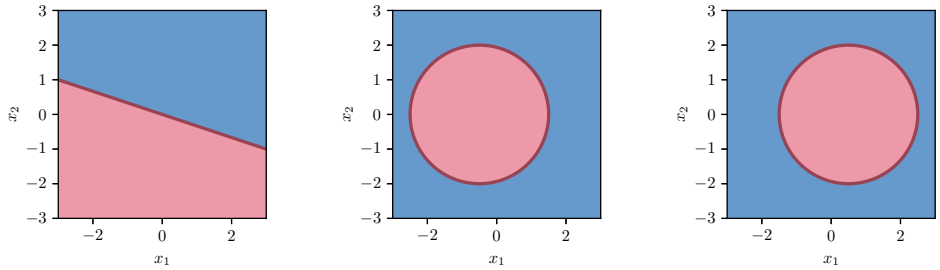
2 Problem Statement

2.1 Illustrative example

Consider three sets $\mathbb{A}$, $\mathbb{B}$ and $\mathbb{C}$ defined below (2):

$$\begin{aligned}\mathbb{A} &: \{x_1 + 3 \cdot x_2 \in [-\infty, 0]\} \\ \mathbb{B} &: \{(x_1 + 0.5)^2 + x_2^2 \in [-\infty, 4]\} \\ \mathbb{C} &: \{(x_1 - 0.5)^2 + x_2 \in [-\infty, 4]\}\end{aligned} \tag{2}$$

These sets are shown in Figure 3. The interior of the set is shown in pink, and the exterior is shown in blue.



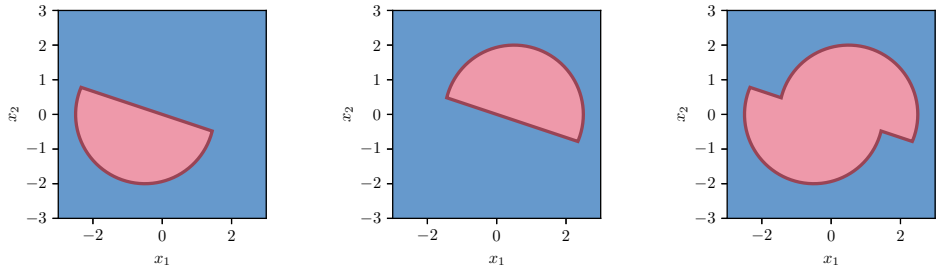
(a) Set $\mathbb{A}$ (b) Set $\mathbb{B}$ (c) Set $\mathbb{C}$

Figure 3: Sets $\mathbb{A}$, $\mathbb{B}$ and $\mathbb{C}$

Define a set $\mathbb{Z}$, computed using $\mathbb{A}$, $\mathbb{B}$ and $\mathbb{C}$, as shown in Equation (3).

$$\mathbb{Z} = (\mathbb{A} \cap \mathbb{B}) \cup (\overline{\mathbb{A}} \cap \mathbb{C}) \tag{3}$$

Set $\mathbb{Z}$, shown in Figure 4c, is built as the union of sets $\mathbb{A} \cap \mathbb{B}$ and $\overline{\mathbb{A}} \cap \mathbb{C}$ represented in Figure 4a, and Figure 4b.



(a) Set $\mathbb{A} \cap \mathbb{B}$ (b) Set $\overline{\mathbb{A}} \cap \mathbb{C}$ (c) $(\mathbb{A} \cap \mathbb{B}) \cup (\overline{\mathbb{A}} \cap \mathbb{C})$

Figure 4: Construction of set $\mathbb{Z}$ from $\mathbb{A}$, $\mathbb{B}$ and $\mathbb{C}$

Note that the two sets $\mathbb{A} \cap \mathbb{B}$ and $\overline{\mathbb{A}} \cap \mathbb{C}$ share a common and non-overlapping boundary. While paving set $\mathbb{Z}$ using the SIVIA algorithm [8], this common boundary appears as shown in Figure 5. This boundary is called a fake boundary [20] as it is not supposed to belong to $\mathbb{Z}$.

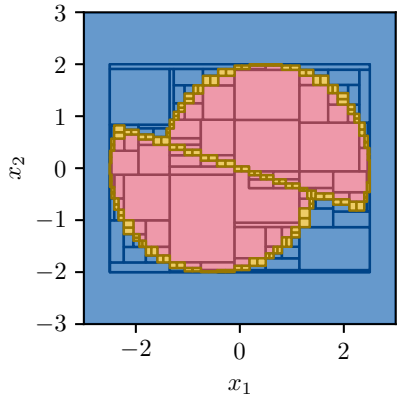


Figure 5: Paving of set $\mathbb{Z}$

2.2 Paving point of view

As shown in Figure 6, the paving algorithm is unable to classify an inner box $[\mathbf{b}]$ overlapping the fake boundary as fully inside $\mathbb{Z}$. Using contractors defined for $\mathbb{Z}$, inner parts $[\mathbf{b}] \setminus [\mathbf{b}_1] = [\mathbf{b}] \setminus \mathcal{C}_{\mathbb{A}\cap\mathbb{B}}([\mathbf{b}])$, and $[\mathbf{b}] \setminus [\mathbf{b}_2] = [\mathbf{b}] \setminus \mathcal{C}_{\overline{\mathbb{A}}\cap\mathbb{C}}([\mathbf{b}])$ are well classified. The remaining part $[\mathbf{b}_3] = [\mathbf{b}] \setminus [\mathbf{b}_1] \setminus [\mathbf{b}_2]$ is classified as unknown and is bisected until the paving algorithm reaches the desired precision.

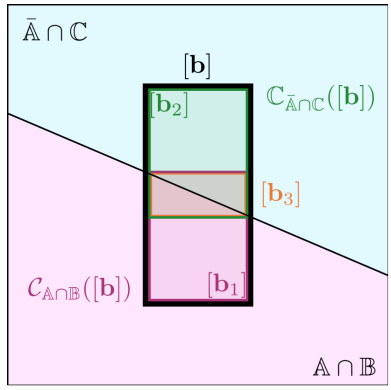
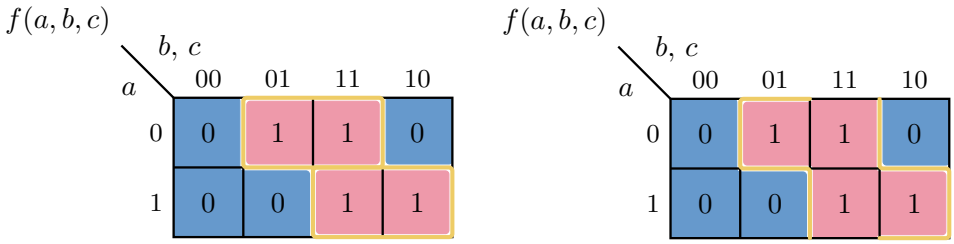


Figure 6: Paving of the fake boundary

To avoid this issue, the paving algorithm has to take into account the fact that $\mathbb{A} \cup \overline{\mathbb{A}} = \mathbb{R}^n$. With this piece of information, the box $[\mathbf{b}]$ can be classified as fully inside $\mathbb{Z}$ in one step.

2.3 Karnaugh map point of view

Karnaugh maps for $(\mathbb{A} \cap \mathbb{B}) \cup (\overline{\mathbb{A}} \cap \mathbb{C})$ and $\mathbb{Z}$ are respectively shown in Figure 7a and Figure 7b. The interior is shown in pink, the exterior is shown in blue, and the boundary is shown in yellow. Although the interior and the exterior of these two sets are equal, the boundaries differ. By denoting by $\delta\mathbb{A}$ the boundary of a set $\mathbb{A}$, the fake boundary appearing on the paving in Figure 5 is $\partial\mathbb{A} \cap \mathbb{B} \cap \mathbb{C}$ and is exactly the difference between the boundaries of $(\mathbb{A} \cap \mathbb{B}) \cup (\overline{\mathbb{A}} \cap \mathbb{C})$ and $\mathbb{Z}$.



(a) Karnaugh map of $(\mathbb{A} \cap \mathbb{B}) \cup (\overline{\mathbb{A}} \cap \mathbb{C})$ (b) Karnaugh map of $\mathbb{Z}$

Figure 7: Comparing Karnaugh maps of $(\mathbb{A} \cap \mathbb{B}) \cup (\overline{\mathbb{A}} \cap \mathbb{C})$ and $\mathbb{Z}$

2.4 Raised issues

Fake boundaries raise two issues. First they add pessimism to the results by classifying boxes around the common boundary as uncertain, whereas these boxes clearly belong to the union of the two sets. Secondly, fake boundaries slow down the paving algorithm by causing unnecessary box bisections around them.

3 Stability of Set Operators

3.1 Topological analysis of set operators

To better understand the issue around the union of adjacent sets, we need to define some tools to analyze the origin of these fake boundaries. Actually, although it turns out that fake boundaries may occur regardless of whether a set operator is Hausdorff-stable, solutions to avoid these fake boundaries are not the same in the two cases.

3.2 Hausdorff distance

Let $(\mathbb{S}, d)$ be a metric space. Define the ϵ-fattening [12] of a set $\mathbb{X}$ of $\mathbb{S}$ by Equation (4).

$$\mathbb{X}_\epsilon = \bigcup_{x \in \mathbb{X}} \{z \in \mathbb{S} \mid d(z, x) \leq \epsilon\} \tag{4}$$

The ϵ-fattening of a set $\mathbb{X}$ is the set of all points in $\mathbb{S}$ that are at most ϵ away from a point in $\mathbb{X}$ relative to the distance d of the metric space, as shown in Figure 8.

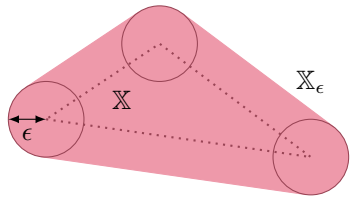


Figure 8: ϵ-fattening of a set

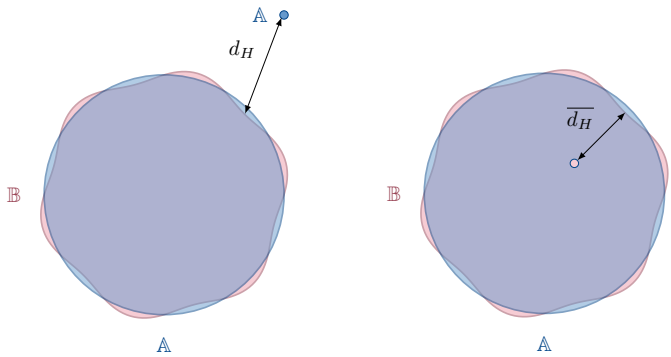
The Hausdorff distance [12] between two subsets $\mathbb{X}$ and $\mathbb{Y}$ of $\mathbb{S}$ is defined by Equation (5):

$$d_H(\mathbb{X}, \mathbb{Y}) = inf\{\epsilon \in \mathbb{R}^+ \mid \mathbb{X} \subseteq \mathbb{Y}_\epsilon \quad and \quad \mathbb{Y} \subseteq \mathbb{X}_\epsilon\} \tag{5}$$

We also introduce the complementary Hausdorff distance defined in Equation (6):

$$\overline{d_H}(\mathbb{X}, \mathbb{Y}) = d_H(\overline{\mathbb{X}}, \overline{\mathbb{Y}}) \tag{6}$$

Example. Figure 9 illustrate cases where Hausdorff distance and complementary Hausdorff distance are significant. Figure 9a shows an example of two sets $\mathbb{A}$ and $\mathbb{B}$ with $d_H(\mathbb{A}, \mathbb{B})$ large because of the small part of $\mathbb{A}$ far from the main part, but $\overline{d_H}(\mathbb{A}, \mathbb{B})$ is tiny, whereas Figure 9b shows an example where $d_H(\mathbb{A}, \mathbb{B})$ is tiny and $\overline{d_H}(\mathbb{A}, \mathbb{B})$ is large because of the hole in $\mathbb{A}$.



(a) Large $d_H(\mathbb{A}, \mathbb{B})$ (b) Large $\overline{d_H}(\mathbb{A}, \mathbb{B})$

Figure 9: Illustration of large Hausdorff and complementary Hausdorff distances

To take into account the general topology of sets, and to be able to compare it, the generalized Hausdorff distance is introduced and defined in Equation (7). It is

the maximum between the Hausdorff distance and the complementary Hausdorff distance.

$$H_d(\mathbb{X}, \mathbb{Y}) = max\{d_H(\mathbb{X}, \mathbb{Y}), \overline{d_H}(\mathbb{X}, \mathbb{Y})\} \tag{7}$$

3.3 Hausdorff stability

Consider two subsets $\mathbb{X}$ and $\mathbb{Y}$ of $\mathbb{S}$. Then a binary operator $\diamond$ acting on set $\mathbb{X}$ and $\mathbb{Y}$ is stable if it meets condition of Equation (8).

$$\forall \eta \in \mathbb{R}_+, \quad \exists \epsilon \in \mathbb{R}_+^*, \quad \begin{cases} H_d(\mathbb{X}, \widetilde{\mathbb{X}}) & \leq \epsilon \\ H_d(\mathbb{Y}, \widetilde{\mathbb{Y}}) & \leq \epsilon \end{cases} \implies H_d(\mathbb{X} \diamond \mathbb{Y}, \widetilde{\mathbb{X}} \diamond \widetilde{\mathbb{Y}}) \leq \eta \tag{8}$$

Example. Consider two subsets $\mathbb{A}$ and $\mathbb{B}$ shown in Figure 10a and two other sets $\widetilde{\mathbb{A}}$ and $\widetilde{\mathbb{B}}$ shown in Figure 10b.

For the union operator, $d_H(\mathbb{A} \cup \mathbb{B}, \widetilde{\mathbb{A}} \cup \widetilde{\mathbb{B}})$ is small, but $\overline{d_H}(\mathbb{A} \cup \mathbb{B}, \widetilde{\mathbb{A}} \cup \widetilde{\mathbb{B}})$ is large as the union of $\widetilde{\mathbb{A}}$ and $\widetilde{\mathbb{B}}$ generates holes at the common boundary of $\mathbb{A}$ and $\mathbb{B}$. Then $H_d(\mathbb{A} \cup \mathbb{B}, \widetilde{\mathbb{A}} \cup \widetilde{\mathbb{B}})$ is large, and the union operator is not Hausdorff-stable for these sets, as it does not meet the condition of Equation (8).

Example. Consider two sets $\mathbb{A}$ and $\mathbb{B}$ shown in Figure 10a and two other sets $\widetilde{\mathbb{A}}$ and $\widetilde{\mathbb{B}}$ shown in Figure 10b.

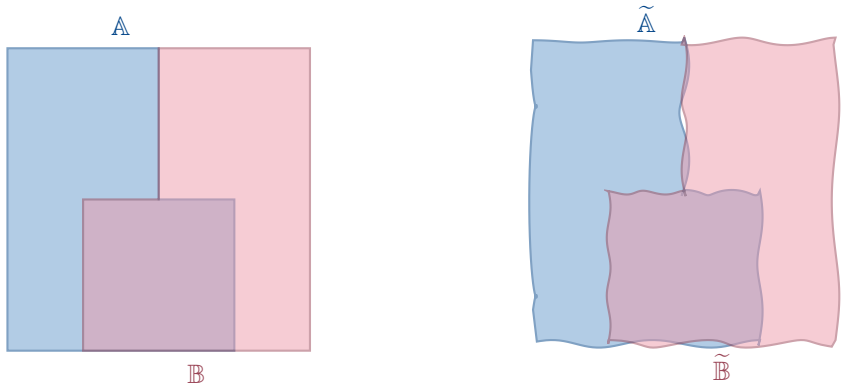
For the intersection operator, $\overline{d_H}(\mathbb{A} \cap \mathbb{B}, \widetilde{\mathbb{A}} \cap \widetilde{\mathbb{B}})$ is small, but $d_H(\mathbb{A} \cap \mathbb{B}, \widetilde{\mathbb{A}} \cap \widetilde{\mathbb{B}})$ is large as the intersection of $\widetilde{\mathbb{A}}$ and $\widetilde{\mathbb{B}}$ generates residual sets at the common boundary of $\mathbb{A}$ and $\mathbb{B}$. Then $H_d(\mathbb{A} \cap \mathbb{B}, \widetilde{\mathbb{A}} \cap \widetilde{\mathbb{B}})$ is large, and the intersection operator is not Hausdorff-stable for these sets, as it does not meet the condition of Equation (8).

Example. Consider the illustrative example presented in Section 2. The union operator between $\mathbb{A} \cap \mathbb{B}$ and $\overline{\mathbb{A}} \cap \mathbb{C}$ is Hausdorff stable as the generalized Hausdorff distance is small. This come from the fact that the same set $\mathbb{A}$ is involved in the computation of $H_d(\mathbb{A} \cap \mathbb{B}, \widetilde{\mathbb{A}} \cap \widetilde{\mathbb{B}})$ and $H_d(\overline{\mathbb{A}} \cap \mathbb{C}, \overline{\widetilde{\mathbb{A}}} \cap \widetilde{\mathbb{C}})$.

This Hausdorff stability condition characterizes the fact that a small perturbation on sets will change the topology of the result by opening boundaries or creating additional ones. It allows identifying topologically different problems. Adapted solutions for Hausdorff-stable and non Hausdorff-stable problems will be proposed in the following sections.

4 Stable Case Solution: Boundary-Preserving Form

In the Hausdorff-stable case, it is possible to change the expression of the computed set $\mathbb{Z}$ by adding a set overlapping the fake boundary. This set helps in the classification of boxes around the fake boundary in the paving algorithm. It must



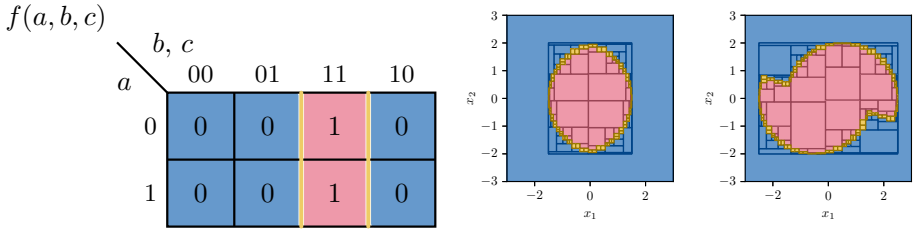
(a) Sets $\mathbb{A}$ and $\mathbb{B}$ (b) Sets $\widetilde{\mathbb{A}}$ and $\widetilde{\mathbb{B}}$

Figure 10: $\mathbb{A}$ and $\mathbb{B}$ are not Hausdorff-stable for union and intersection operators

be chosen such that the interior and the exterior of $\mathbb{Z}$ are preserved, but also its boundary. In this example, the set $\mathbb{D} = \mathbb{B} \cap \mathbb{C}$ is added to the expression of $\mathbb{Z}$ which becomes $\mathbb{Z}'$ Equation (9).

$$\mathbb{Z}' = (\mathbb{A} \cap \mathbb{B}) \cup (\overline{\mathbb{A}} \cap \mathbb{C}) \cup (\mathbb{B} \cap \mathbb{C}) \tag{9}$$

The Karnaugh map of the set $\mathbb{D}$ is shown in Figure 11a, and the paving of $\mathbb{D}$ is shown in Figure 11b. This set ensures that the Karnaugh map of $\mathbb{Z}'$ is the same as the Karnaugh map of $\mathbb{Z}$ shown in Figure 7b. The resulting paving of $\mathbb{Z}'$ is shown in Figure 11c. There is no more fake boundaries.



(a) Karnaugh map of $\mathbb{D}$ (b) Set $\mathbb{D}$ (c) Set $\mathbb{Z}'$

Figure 11: Boundary preserving form

Using the boundary preserving form leads to a correct paving without any fake boundaries. Therefore, to use this solution, the set boundaries have to be analyzed to find the fake boundaries and add a set overlapping these fake boundaries in the expression of the set to pave. This approach is working but is problem-specific and needs to be adapted on a case-by-case basis. This method works well in the Hausdorff-stable case, as there is the possibility to add a set that overlap the fake

boundary. For non Hausdorff-stable operators, the boundary preserving form is not possible as the Karnaugh map is not highlighting any sets that can be added to the expression of the paved set to avoid fake boundaries, and another approach is needed.

5 Non-Stable Case: Boundary Approach

5.1 Topology of the boundary

Let $T = (\mathbb{S}, \tau)$ be a topological space. $\forall \mathbb{X} \in \mathbb{S}$, denote by $\overline{\mathbb{X}}$ the complementary of $\mathbb{X}$ in $\mathbb{S}$, by $cl_{\mathbb{S}}(\mathbb{X})$ the closure of $\mathbb{X}$ in $\mathbb{S}$, by $int_{\mathbb{S}}(\mathbb{X})$ the interior of $\mathbb{X}$ in $\mathbb{S}$, and by $\partial\mathbb{X}$ the boundary of $\mathbb{X}$ in $\mathbb{S}$.

Theorem. *Let $T = (\mathbb{S}, \tau)$ be a topological space. Then*

$$\forall (\mathbb{A}, \mathbb{B}) \in \mathbb{S}^2 \quad \partial(\mathbb{A} \cup \mathbb{B}) \subseteq \partial\mathbb{A} \cup \partial\mathbb{B}$$

Proof. By definition of the boundary

$$\forall \mathbb{A} \in \mathbb{S}, \quad \partial\mathbb{A} = cl_{\mathbb{S}}(\overline{\mathbb{A}}) \cap cl_{\mathbb{S}}(\mathbb{A})$$

By property, intersection is a subset of each set

$$\forall (\mathbb{A}, \mathbb{B}) \in \mathbb{S}^2, \quad \begin{cases} \mathbb{A} \cap \mathbb{B} \subseteq \mathbb{A} \\ \mathbb{A} \cap \mathbb{B} \subseteq \mathbb{B} \end{cases}$$

Then

$$\begin{aligned} \partial(\mathbb{A} \cup \mathbb{B}) &= cl_{\mathbb{S}}(\overline{\mathbb{A} \cup \mathbb{B}}) \cap cl_{\mathbb{S}}(\mathbb{A} \cup \mathbb{B}) \\ &= cl_{\mathbb{S}}(\overline{\mathbb{A}} \cap \overline{\mathbb{B}}) \cap cl_{\mathbb{S}}(\mathbb{A} \cup \mathbb{B}) \\ &= cl_{\mathbb{S}}(\overline{\mathbb{A}} \cap \overline{\mathbb{B}}) \cap (cl_{\mathbb{S}}(\mathbb{A}) \cup cl_{\mathbb{S}}(\mathbb{B})) \\ &= (cl_{\mathbb{S}}(\overline{\mathbb{A}} \cap \overline{\mathbb{B}}) \cap cl_{\mathbb{S}}(\mathbb{A})) \cup cl_{\mathbb{S}}(\overline{\mathbb{A}} \cap \overline{\mathbb{B}}) \cap cl_{S}(\mathbb{B}) \\ &\subseteq (cl_{\mathbb{S}}(\overline{\mathbb{A}}) \cap cl_{\mathbb{S}}(\mathbb{A})) \cup (cl_{\mathbb{S}}(\overline{\mathbb{B}}) \cap cl_{\mathbb{S}}(\mathbb{B})) \\ &= \partial\mathbb{A} \cup \partial\mathbb{B} \end{aligned}$$

□

Section 5.1 demonstrates that the boundary is not preserved over union of sets as $\partial(\mathbb{A} \cup \mathbb{B}) \subseteq \partial\mathbb{A} \cup \partial\mathbb{B}$. This is why the paving of the union of contractors leads to fake boundaries.

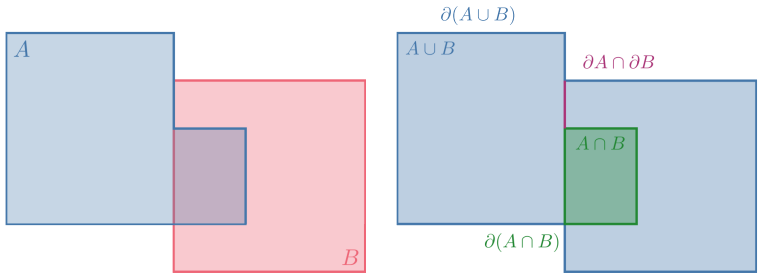
Section 5.1 present the general formula for the union of the boundary of two sets.

Theorem. *Let (S, τ) be a topological space. Then*

$$\forall (\mathbb{A}, \mathbb{B}) \in \mathbb{S}^2 \qquad \partial\mathbb{A} \cup \partial\mathbb{B} = \partial(\mathbb{A} \cup \mathbb{B}) \cup \partial(\mathbb{A} \cap \mathbb{B}) \cup (\partial\mathbb{A} \cap \partial\mathbb{B})$$

Proof. Section 5.1 is proven in [9]. □

From Section 5.1, it is noticeable that the union of boundaries is not the boundary of union. This is the reason why $\mathbb{Z}$ and $(\mathbb{A} \cap \mathbb{B}) \cup (\overline{\mathbb{A}} \cap \mathbb{C})$ do not have the same boundaries when paving these sets. An illustration of Section 5.1 is shown in Figure 12.



(a) Set $\mathbb{A}$ and $\mathbb{B}$ (b) Decomposition of $\partial\mathbb{A} \cup \partial\mathbb{B}$

Figure 12: Illustration of Section 5.1

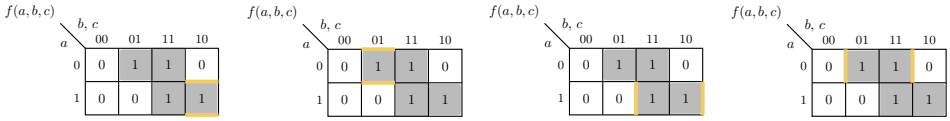
Remark. When $\mathbb{A} \cap \mathbb{B} = \emptyset$ and $\partial\mathbb{A} \cap \partial\mathbb{B} = \emptyset$ in Section 5.1, the union of boundaries is the boundary of union. This is the case where the sets are non-overlapping with no common boundary.

5.2 Boundary approach

A boundary approach can be used to get rid of this fake boundary. This will help to solve this purely computational problem, as the mathematical expression of the set $\mathbb{Z}$ do not have any fake boundaries. This approach consists in computing the boundary of the set $\mathbb{Z}$. This boundary will separate an inner and an outer subpaving. The classification of the resulting subpavings as inside or outside is done using a predicate. The boundary approach method was first introduced in [7] to speed up the solving of set inversion problems.

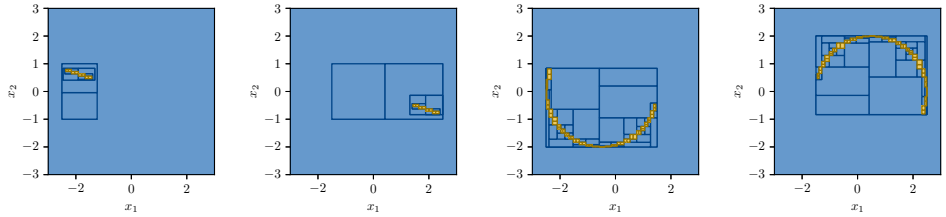
First, $\partial\mathbb{Z}$ has to be expressed from set $\mathbb{A}$, $\mathbb{B}$, and $\mathbb{C}$ without the fake boundary. Figure 13 and Figure 14 respectively show Karnaugh maps and paving of intermediate sets involved in the building of $\partial\mathbb{Z}$. Then, $\partial\mathbb{Z}$ is computed as the union of these boundaries, and it matches the Karnaugh map of $\mathbb{Z}$ shown in Figure 7b.

Then using a predicate, the connected subsets separated by $\partial\mathbb{Z}$ are classified as inside or outside. This predicate is based on the expression of $\mathbb{Z}$ of Equation (3), and is tested on box corners until an in and an out points are found. Then, boxes containing each point are classified as in and out boxes, and the information



(a) $\partial\mathbb{A} \cap \mathbb{B} \cap \overline{\mathbb{C}}$ (b) $\partial\mathbb{A} \cap \overline{\mathbb{B}} \cap \mathbb{C}$ (c) $\mathbb{A} \cap \partial\mathbb{B}$ (d) $\overline{\mathbb{A}} \cap \partial\mathbb{C}$

Figure 13: Karnaugh map of the boundaries

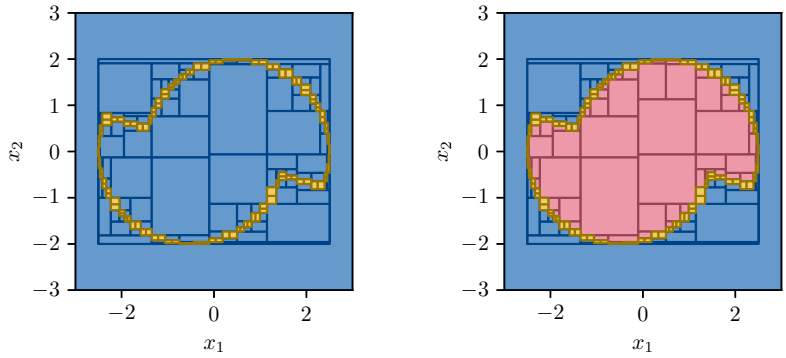


(a) $\partial\mathbb{A} \cap \mathbb{B} \cap \overline{\mathbb{C}}$ (b) $\partial\mathbb{A} \cap \overline{\mathbb{B}} \cap \mathbb{C}$ (c) $\mathbb{A} \cap \partial\mathbb{B}$ (d) $\overline{\mathbb{A}} \cap \partial\mathbb{C}$

Figure 14: Building the boundary of $\mathbb{Z}$

is propagated near to near without crossing the boundary. Finally, each box is classified as in, out, or uncertain.

Figure 15a shows $\partial\mathbb{Z}$ built from boundaries shown in Figure 14 using the proposed method. The resulting paving of Z is shown in Figure 15b, which is classified using the subpaving coloration method.



(a) ∂Z (b) Z

Figure 15: Boundary approach

This boundary approach is efficient to get rid of fake boundaries. Set $\mathbb{Z}$ is computed from the union of two separators, $\mathcal{S}_{\mathbb{A}\cap\mathbb{B}}$, and $\mathcal{S}_{\overline{\mathbb{A}}\cap\mathbb{C}}$, and this union is reinforced by a contractor on the boundary $\mathcal{C}_{\partial\mathbb{Z}}$.

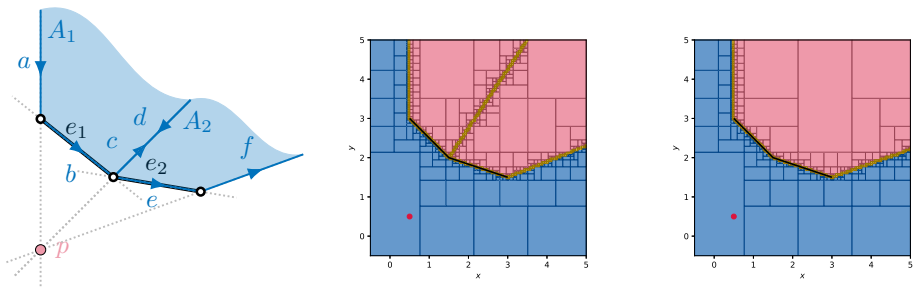
Remark. This method is also working for the Hausdorff-stable case, but it is more efficient to use the boundary preserving form presented in Section 4, as the contractor on the boundary is not easy to define, and the subpaving coloration method is not needed.

6 Application

6.1 Boundary approach application to the separator on the visibility constraint

Separator over the visibility constraint, as implemented in [4], suffer from this fake boundaries when it deals with polygon obstacles. In fact, the contractor on the visibility constraint is defined for an obstacle segment. The extension to polygons involves the union of non-visible areas relative to each segment, and this union leads to fake boundaries.

Figure 16a shows an illustration of the separator on the visibility constraint as implemented in [4]. For each obstacle segment, three segments are defined, which separate the visible and non-visible parts of the space. For segment e_1 is defined relative to the observation point p, the oriented half space on the left of segment a, the one on the left of segment b, and the same for segment c. It is the same for the set of visible points for segment e_2 defined by half planes on the left of segments d, e, and f. The set of masked points from p by e_1 and e_2 i then the union of these two sets A_1 and A_2. Paving this separator shows a fake boundary as shown in Figure 16b.



(a) Separator construction (b) Classical approach (c) Boundary approach

Figure 16: Separator on the visibility constraint using the boundary approach

To avoid this problem, the boundary approach can be applied. The set of masked points from observation point p relative to segments e_1 and e_2 should be defined by half planes on the left of segments a, b, e, and f. The simplification of $c = -d$ has to be taken into account while contracting to avoid this fake boundary. This simplification is based on algebraic topology [18] in which boundary simplifications

are defined and used. Figure 16c shows the paving of this separator using the boundary approach. There is no longer fake boundaries appearing.

Remark. For now, fake boundaries have to be identified and removed by hand, as it not the main topic of this paper. Neither [4] nor this work propose an automatic boundary simplification to avoid fake boundaries in union of adjacent sets. Therefore, it is necessary to find solutions that are problem-specific in order to avoid fake boundaries, as developed in the next subsection.

6.2 Toward a generic implementation of the separator on the visibility constraint

In the case of the visibility constraint another approach to solve this problem can be proposed. The set of visible points from an observation point placed at $(0, 0)$ relative to a shape $\mathbb{Y}$ can be defined by :

$$\mathbb{S} = \{x \in \mathbb{R}^2, \exists \alpha \in \mathbb{R} \mid \alpha \cdot \mathbf{x} \in \mathbb{Y}\} \tag{10}$$

Denoting by f the homothety defined in Equation (11).

$$\begin{aligned} f : \mathbb{R}^3 &\mapsto \mathbb{R}^2 \\ (\mathbf{x}, \alpha) &\to \alpha \cdot \mathbf{x} \end{aligned} \tag{11}$$

Remark. If the observation point is not placed at (0, 0), a simple translation of the problem leads to the presented solution.

The set $\mathbb{S}$ can then be defined as the projection of $f(\mathbb{Y})$ for $\alpha \in [0, 1]$. Listing 1 show the implementation of this separator using the Codac Library [15]. Figure 17a shows the paving of this implementation of the visibility constraint. The comparison with Figure 17b, where the classical implementation of this constraint from [4] on the same obstacle polygon is shown, validates that the problem of fake boundaries is avoided with this method. Figure 17a requires 391 bisections whereas Figure 17b requires 321 bisections. The complexity of these two approaches is quite similar, although all the tests carried out lead to a slightly higher number of bisections for the proposed method, with the benefit of a set without fake boundaries. This is mainly due to the projection algorithm which induces bisections in the dimension of the homothety factor α.

Figure 18 shows the paving of the visibility separator on the same obstacle presented in Figure 2, but the proposed method avoid fake boundaries.

Remark. The separator representing the obstacle could be any separator. However, the separator must be in a closed form with an interior. This method is not applicable to segments or open polygons for instance. But the advantage of this approach is that it can be applied to an ellipsis obstacle.

Remark. There are polygons for which the separator on the visibility constraint does not generate fake boundaries. In these cases the classical implementation proposed in [4] is more efficient than the proposed approach, as the algorithm used to project a separator is based on contractors over quantifiers which requires bisections [2, 13].

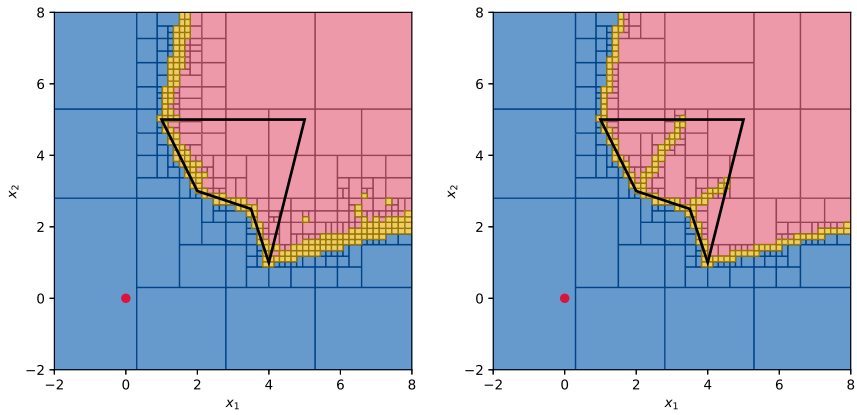
```python
import codac as cd

# Set Y definition
polygon = [[2, 3], [3.5, 2.5], [4, -1], [5, 5], [1, 5], [2, 3]]
Sy = cd.SepPolygon(polygon)

# Set Z definition
f = cd.Function("x", "y", "a", "(a*x,a*y)")
Sz = cd.SepInverse(Sy, f)

# Projection of for a in [0, 1]
epsilon = 0.1
Sx = cd.SepProj(Sz, cd.Interval(0, 1), epsilon)
```

Listing 1: Separator on the visibility constraint using Codac Library



(a) Proposed implementation (b) Classical implementation

Figure 17: Generic SepVisible implementation

Figure 19 shows the comparison between the classical and the projection approaches on the paving of a visibility separator without fake boundaries. The proposed implementation shown in Figure 19a requires 419 bisections whereas the classical implementation shown in Figure 19b requires only 278 bisections.

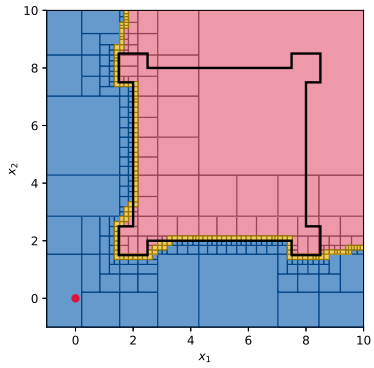
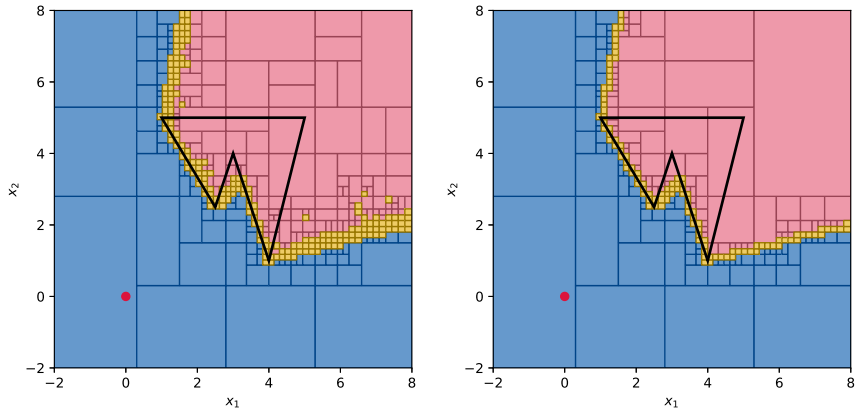


Figure 18: Separator on the visibility constraint on a room



(a) Proposed implementation (b) Classical implementation

Figure 19: Generic implementation of the separator on the visibility constraint

7 Conclusion

In conclusion, this work has highlighted the problem associated with the union of adjacent contractors. Paving the union of these contractors creates fake boundaries that add pessimism to the results and increase the computation time.

This problem occurs in two cases: when an operator applied to sets is Hausdorff-stable, and when it is non-Hausdorff-stable. An approach for Hausdorff-stable is to use a boundary preserving form by adding sets overlapping the fake boundary in the expression of the paved set. For non-Hausdorff-stable operators, a boundary approach is proposed to get rid of this fake boundary.

The result shows that both these approaches are efficient in fake boundary avoidance. The drawback of these methods is that they are problem-specific, they

need to be tuned for each problem, and this paper does not provide an automatic way to remove fake boundaries.

Finally, a generic implementation of the separator on the visibility constraint was proposed. This approach shows that sometimes the fake boundary problem can also be avoided by expressing the problem differently. There are, nevertheless, cases in which no false boundary appears in the classical implementation of the separator on the visibility constraint. In these cases, the classical implementation is more efficient than the proposed method, since the latter involves the projection of a separator, which is computationally time-consuming.

References

[1] Andrews, P. B. *An Introduction to Mathematical Logic and Type Theory: To Truth Through Proof*. Applied Logic Series. Springer Netherlands, 2013. DOI: 10.1007/978-94-015-9934-4.

[2] Chabert, G. and Jaulin, L. Contractor programming. *Artificial Intelligence*, 173:1079–1100, 2009. DOI: 10.1016/j.artint.2009.03.002.

[3] Chauhan, A., Vyas, P., Vachhani, L., and Maity, A. Optimal path planning for a non-holonomic robot using interval analysis. In *Proceedings of the Indian Control Conference*, pages 184–189, 2018. DOI: 10.1109/INDIANCC.2018.8307975.

[4] Guyonneau, R. *Méthodes Ensemblistes Pour La Localisation En Robotique Mobile*. PhD Thesis, Angers, 2013. URL: https://theses.hal.science/tel-00961501.

[5] Guyonneau, R., Lagrange, S., Hardouin, L., and Lucidarme, P. Guaranteed interval analysis localization for mobile robots. *Advanced Robotics*, 28(16):1067–1077, 2014. DOI: 10.1080/01691864.2014.908742.

[6] Jaulin, L. Path planning using intervals and graphs. *Reliable Computing*, 7(1):1–15, 2001. DOI: 10.1023/A:1011400431065.

[7] Jaulin, L. A boundary approach for set inversion. *Engineering Applications of Artificial Intelligence*, 100:104184, 2021. DOI: 10.1016/j.engappai.2021.104184.

[8] Jaulin, L., Kieffer, M., Didrit, O., Walter, E., Jaulin, L., Kieffer, M., Didrit, O., and Walter, É. *Interval analysis*. Springer, 2001. DOI: 10.1007/978-1-4471-0249-6.

[9] Law Wysocki, M. and Darmochwa, A. Subsets of topological spaces. *Journal of Formalized Mathematics*, 1, 1989. URL: https://mizar.uwb.edu.pl/JFM/Vol1/tops_1.html.

[10] Merlet, J.-P. Interval analysis and robotics. In Kaneko, M. and Nakamura, Y., editors, *Robotics Research*, pages 147–156, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg. DOI: 10.1007/978-3-642-14743-2_13.

[11] Mourad, F., Snoussi, H., Abdallah, F., and Richard, C. Anchor-based localization via interval analysis for mobile ad-hoc sensor networks. *IEEE Transactions on Signal Processing*, 57(8):3226–3239, 2009. DOI: 10.1109/TSP.2009.2020018.

[12] Munkres, J. *Topology*. Featured Titles for Topology. Prentice Hall, Incorporated, 2000. ISBN: 9780131816299.

[13] Ninin, J. Global optimization based on contractor programming: An overview of the IBEX library. In Kotsireas, I. S., Rump, S. M., and Yap, C. K., editors, *Mathematical Aspects of Computer and Information Sciences*, pages 555–559, Cham, 2016. Springer International Publishing. DOI: 10.1007/978-3-319-32859-1_47.

[14] Rauh, A. and Hofer, E. P. Interval methods for optimal control. In *Variational Analysis and Aerospace Engineering*, pages 397–418, New York, NY, 2009. Springer New York. DOI: 10.1007/978-0-387-95857-6_22.

[15] Rohou, S., Desrochers, B., and Le Bars, F. The codac library. *Acta Cybernetica*, 26(4):881–887, 2024. DOI: 10.14232/actacyb.302772.

[16] Seignez, E., Kieffer, M., Lambert, A., Walter, E., and Maurin, T. Experimental vehicle localization by bounded-error state estimation using interval analysis. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1084–1089, 2005. DOI: 10.1109/IROS.2005.1545155.

[17] Stoll, R. R. *Set Theory and Logic*. Dover books on advanced mathematics. Dover Publications, 1979. ISBN: 9780486638294.

[18] Tom Dieck, T. *Algebraic topology*, Volume 8. European Mathematical Society, 2008. DOI: 10.4171/048.

[19] Vehí, J., Ferrer, I., and Ángel Sainz, M. A survey of applications of interval analysis to robust control. *IFAC Proceedings Volumes*, 35(1):389–400, 2002. DOI: 10.3182/20020721-6-ES-1901.00399.

[20] Welte, A., Jaulin, L., Ceberio, M., and Kreinovich, V. Avoiding fake boundaries in set interval computing. *Journal of Uncertain Systems*, 11(2):137 – 148, 2017. URL: https://ensta-bretagne.hal.science/hal-01698416/file/jusVol11No2paper07.pdf.

Acta Cybernetica **27** (2026) 493–514.

Effective Representation and Fast Computing With Polyarc Bounded Intervals

Gábor Geréb[ab] and András Sándor[cd]

Abstract

Complex interval arithmetic is a powerful tool for the analysis of computational errors. The naturally arising rectangular, polar, and circular interval types yield overly relaxed bounds. The later introduced polygonal type allows for arbitrarily precise representation for a higher computational cost. We propose the polyarc interval type as an effective generalization of the abovementioned types. The polyarc interval can represent all types and most of their arithmetic combinations precisely and has a better approximation capability with that of the polygonal interval. In particular, in specific cases of antenna tolerance analysis and robot localization it can achieve perfect accuracy for lower computational cost then the polygonal type, which we show in a relevant case study.

Keywords: interval arithmetic, computational geometry, antennas, robotics

1 Introduction

Interval analysis is an effective mathematical technique that allows numerical analysis of problems involving sets [23]. It has long been used to put bounds on computational errors and has recently been applied in robotics, robust control, and antenna design. It also proved to be useful for finding all the solutions of nonlinear equations and inequalities [19]. Interval arithmetic studies the properties of the numerical representation and arithmetic operations of intervals, and therefore it is an essential part of interval analysis.

Pioneered by Moore [22], the interval arithmetic for analyzing real-valued computational errors was soon extended to complex numbers in the form of rectangular and polar intervals by Boche [4], and as circular intervals by Gargantini and Henrici [13]. Hansen [16] studied the linear algebra of complex intervals and introduced a generalized interval arithmetic. The field received increased attention in the 1990s: Ohta [24] introduced polygon interval arithmetic, a journal titled Interval

[a]Department of Informatics, University of Oslo, Norway
[b]E-mail: gaborge@uio.no, ORCID: 0000-0002-1972-4784
[c]Budapest University of Technology and Economics, Budapest, Hungary
[d]E-mail: sandor.andras@renyi.hu, ORCID: 0000-0002-8707-9182

DOI: 10.14232/actacyb.314364

Computations (later renamed Reliable Computing) was established [21, 20], the Matlab/Octave software package called INTLAB was published by [27], and the BLAS FORTRAN package received an interval extension [9]. Books on interval analysis and arithmetic were published by Petkovic and Petkovic [26], Jaulin et al. [19], Moore et al. [23] and Dawood [6].

Complex interval arithmetic greatly benefited from its interconnections with geometric algebra, which is widely used in the fields of computer-aided design, image processing, mathematical morphology, geometrical optics, and dynamical stability analysis. The application of the Minkowski algebra to complex intervals opened up the possibility of the representation and arithmetic combination of intervals bounded by arbitrary explicit and implicit curves in the complex plane [12, 10]. Efficient algorithms for calculating the Minkowski sum of polygons [7], borrowed from computational geometry, have been successfully applied in the design of robust control systems [24, 25] and the tolerance analysis of antenna arrays [28]. The polygonal representation typically produced much more accurate results than the original representations, given that the vertex count was high enough (Figure 1). However, a high vertex count came with a high computational cost.

In the tolerance analysis of sensor arrays, the polar intervals defined by independent amplitude and phase intervals are typically the primary operands of evaluation. Motivated by the success of the polygonal representation and its shortcomings in representing polar and circular intervals and their arithmetic combinations, we set out to find a more suitable interval type. Replacing vertices by circular arcs came as a natural extension and led to a new interval type, the polyarc-bounded (polyarcular) interval. By providing perfect representation for a much wider set of intervals at a similar complexity as the polygonal interval, the new interval type suited our application well.

Polyarc is an explicit curve type, defined by an ordered set of circular arcs and consists of the defining arcs and implicit edges between them (Figure 2). It allows the exact representation of the boundary of the (bounded) rectangular, polar, circular, and polygonal intervals. It is closed under addition, negative, reciprocal, union, and intersection operations, and in some cases under multiplication, too.

In this paper we present the previously unpublished polyarc interval type. In Section 2 we summarize the necessary background theory and the existing interval arithmetic methods. We present the polyarc interval type in Section 3, along with the discussion of its properties and a description of our implementation, which is available as an open source code in the form of a Matlab package (see Supplementary materials). To demonstrate the applicability of our method, we show two case-studies in Section 4, one from antenna design and another from robotics, both intuitive yet current questions of academic and industrial interest. Finally, we summarize our findings in Section 5 and discuss future research opportunities.

For the sake of brevity, the typesetting of symbols in equations carry specific meanings, which is summarized in Table 1.

Table 1: Typesetting in mathematical expressions

Typesetting	Meaning	Example
Calligraphic $(\mathcal{I}, \mathcal{R})$	Sets of curves or intervals	$\mathcal{I}(\mathbb{R}) = \{\boldsymbol{t} = [\underline{t}, \overline{t}] \big\vert \underline{t} \le \overline{t}\}$
Bold-italic $(\boldsymbol{a}, \boldsymbol{A})$	Sets	$\boldsymbol{A} = \{A \in \mathbb{C} \big\vert \mathrm{Re}(A) = 1\}$
Italic (a, A)	Numbers	$A \in \boldsymbol{A} = \{t + it \big\vert t \in \boldsymbol{t}\}$
Lowercase $(a, \boldsymbol{a})$	Real numbers and sets	$a \in \boldsymbol{a} \subset \mathbb{R}$
Uppercase $(A, \boldsymbol{A})$	Complex numbers and sets	$A \in \boldsymbol{A} \subset \mathbb{C}$
Sans-sherif (n, N)	Natural numbers and sets	$\mathsf{n} \in \{1..\mathsf{N}\} \subset \mathbb{N}$
Under-, overlined $(\underline{t}, \overline{t})$	Infimum and supremum	$[\underline{t}, \overline{t}] \ni t,\ \underline{t} \le t \le \overline{t}$

2 Background

2.1 Complex intervals

Complex intervals are certain subsets of the complex plane $\mathbb{C}$, which we will consider our universe. Since there is no common definition for complex intervals (some prefer the term: complex sets), for this discussion we assume that all of them are connected and bounded sets. We also assume that they each have a piece-wise smooth, simple, closed (Jordan) boundary curve. The set of Jordan curves is $\mathcal{J}(\mathbb{C})$.

Definition. *A complex set belongs to the set $\mathcal{I}(\mathbb{C})$ of complex intervals if it is bounded by a piecewise smooth Jordan curve:*

$$\boldsymbol{A} \in \mathcal{I}(\mathbb{C}) \iff \partial \boldsymbol{A} \in \mathcal{J}(\mathbb{C}).$$

The three most common complex interval types have emerged as an extension of real intervals $\mathcal{I}(\mathbb{R})$ to the complex plane. These are the rectangular, polar [4] and circular [13] intervals. Illustrative examples can be found in Figure 1.

Definition. *A rectangular interval*

$$\boldsymbol{a} + \boldsymbol{b}i = \{Z \in \mathbb{C} \mid \mathrm{Re}(Z) \in \boldsymbol{a}, \mathrm{Im}(Z) \in \boldsymbol{b}\} \in \mathcal{R}(\mathbb{C})$$

(with $\boldsymbol{a}, \boldsymbol{b} \in \mathcal{I}(\mathbb{R})$) is the Cartesian product of two real intervals.

Definition. *A polar interval is defined by the polar product of two real intervals, that is*

$$\boldsymbol{r}e^{i\boldsymbol{\varphi}} = \{Z \in \mathbb{C} \mid |Z| \in \boldsymbol{r}, \angle Z \in \boldsymbol{\varphi}\} \in \mathcal{P}(\mathbb{C})$$

where $\boldsymbol{r} \in \mathcal{I}(\mathbb{R})$ is the radial and $\boldsymbol{\varphi} \in \mathcal{I}(\mathbb{R})$ is the angular interval, and the center is the origin.

Definition. *A circular interval is a closed disk in the complex plane, that is*

$$O + [0, r] \cdot e^{i[-\pi, \pi]} = \{Z \in \mathbb{C} \mid |Z - O| \le r\} \in \mathcal{C}(\mathbb{C})$$

with center $O \in \mathbb{C}$ and radius $r \in \mathbb{R}$.

Polygon interval arithmetic was introduced in [24, 25] to represent uncertainty in robust control systems. Since a polygon can approximate any simple closed curve with a finite dataset, it can also represent any complex interval with arbitrary precision limited only by computational constraints. The set of polygonal curves is $\bar{\mathcal{J}}(\mathbb{C})$.

Definition. *Polygonal intervals are complex intervals bounded by polygonal curves.*

$$\boldsymbol{A} \in \mathcal{G}(\mathbb{C}) \subset \mathcal{I}(\mathbb{C}) \iff \partial \boldsymbol{A} \in \bar{\mathcal{J}}(\mathbb{C})$$

Each interval type above has a finite data representation and specific algorithms for arithmetic and set operations. An interval can be represented by different types, and the representation can be cast from one type to the other. We consider an interval to be a member of a given type if it can be exactly represented by it. If an interval is not a member of the corresponding type, then we assume the smallest inclusive interval of the type, in which case the representation will not be tight.

Definition. *The tightness of a representation* $\boldsymbol{A}^{\mathcal{X}} \in \mathcal{X}(\mathbb{C})$ *of the complex interval* $\boldsymbol{A} \in \mathcal{I}(\mathbb{C})$ *is the ratio of its original size and its represented size.*

$$\tau(\boldsymbol{A}^{\mathcal{X}}) = \frac{\mu(\boldsymbol{A})}{\mu(\boldsymbol{A}^{\mathcal{X}})} \in [0, 1],$$

where $\mu(\cdot)$ *is the Lebesgue measure on* $\mathbb{C}$.

2.2 Interval arithmetic

Arithmetic operations on intervals are typically defined by the Minkowski algebra, which is a collection of pointwise operations on sets.

Definition. *For* $\boldsymbol{A}, \boldsymbol{B} \in \mathcal{I}(\mathbb{C})$

$$\begin{aligned}
\boldsymbol{A} \oplus \boldsymbol{B} &= \{A + B \,|\, A \in \boldsymbol{A}, B \in \boldsymbol{B}\}, \\
\boldsymbol{A} \otimes \boldsymbol{B} &= \{A \times B \,|\, A \in \boldsymbol{A}, B \in \boldsymbol{B}\}, \\
-\boldsymbol{A} &= \{-A \,|\, A \in \boldsymbol{A}\}, \\
\boldsymbol{A}^{-1} &= \{A^{-1} \,|\, A \in \boldsymbol{A}\}, \\
\boldsymbol{A} \ominus \boldsymbol{B} &= \{A + (-B) \,|\, A \in \boldsymbol{A}, B \in \boldsymbol{B}\} = \boldsymbol{A} \oplus (-\boldsymbol{B}), \\
\boldsymbol{A} \oslash \boldsymbol{B} &= \{A \times B^{-1} \,|\, A \in \boldsymbol{A}, B \in \boldsymbol{B}\} = \boldsymbol{A} \otimes (\boldsymbol{B}^{-1}).
\end{aligned}$$

Minkowski addition ($\oplus$) and multiplication ($\otimes$) are both closed binary operations. The element-wise negation is a closed unary operation. This means that the results of the sum, product and negative of complex intervals are also complex intervals. However, the reciprocal is only a partial unary operation because the complex zero has no inverse on the complex plane and hence intervals including zero have unbounded inverses. The element-wise subtraction ($\ominus$) and division ($\oslash$) can be defined as the combination of the operations mentioned above and consequently they are closed and partial binary operations respectively. Similarly to real intervals, the Minkowski addition and multiplication of complex intervals are commutative and associative but not distributive [15, 26].

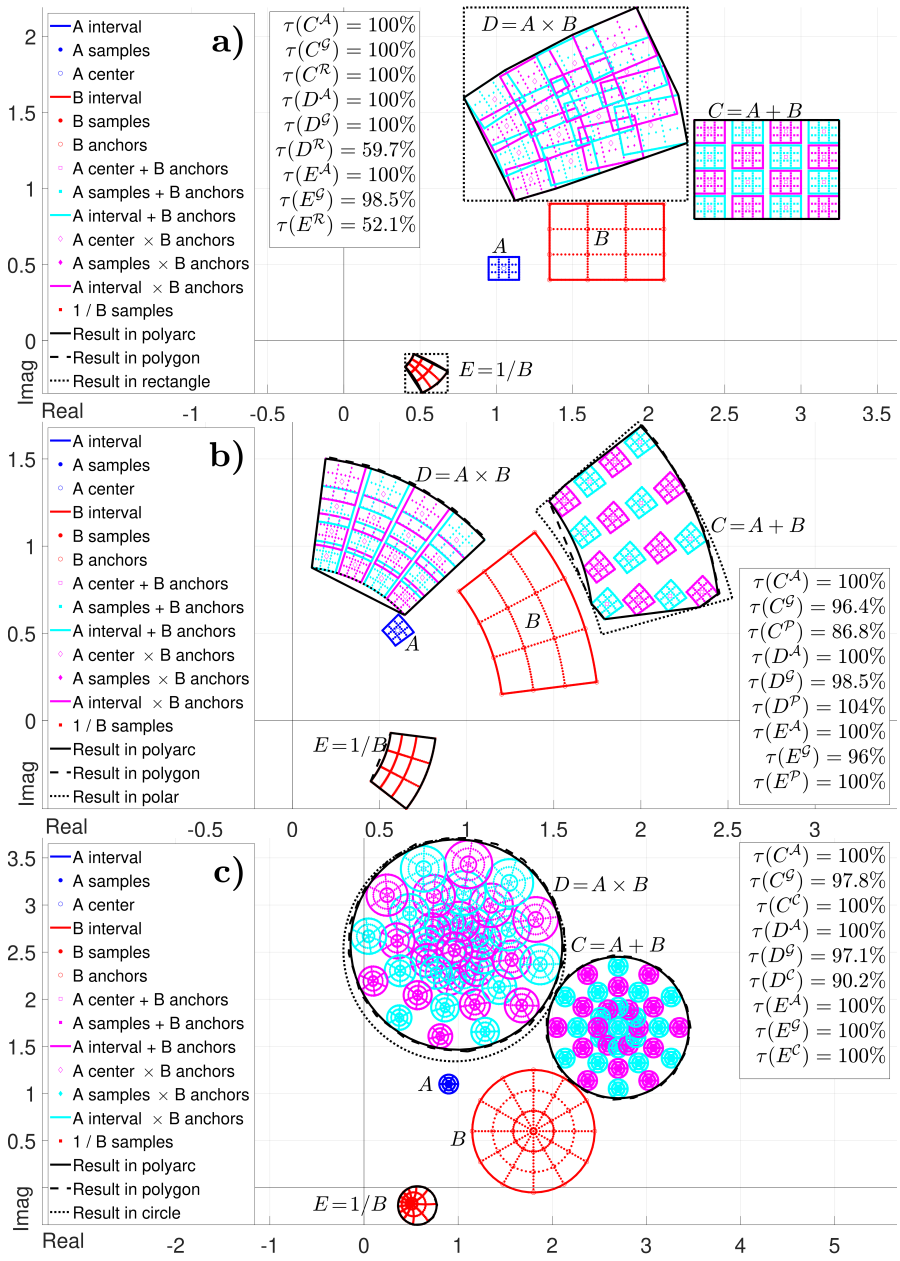


Figure 1: Sum, product, and reciprocal of **a)** rectangular ($\mathcal{R}$), **b)** polar ($\mathcal{P}$) and **c)** circular ($\mathcal{C}$) intervals. Each interval type is casted to polygonal ($\mathcal{G}$) and polyarc ($\mathcal{A}$) types and the tightness ($\tau(\cdot)$) of the arithmetic results are compared.

Table 2: Arithmetic and set properties of complex interval types. For example the polar type is not closed under addition, so $\boldsymbol{A}, \boldsymbol{B} \in \mathcal{P}(\mathbb{C}) \implies \boldsymbol{A}+\boldsymbol{B} \in \mathcal{A}(\mathbb{C})$. Blue symbols indicate when the operation points outside the operand type, but can be represented by another type. In this case we indicate the most specific type that includes the result considering that $\mathcal{R} \subset \mathcal{G}$ and $\mathcal{R}, \mathcal{P}, \mathcal{C}, \mathcal{G} \subset \mathcal{A}$. Red symbols indicate when none of the types can represent the result. (* If one operand is a polar interval, the result is polyarcular, which has a relevance in our first case study in Section 4. † The intersection operation assumes that the operands are connected, while the union operation assumes that the result is connected.) For formal proof of these properties, see Supplementary materials.

$\boldsymbol{A}, \boldsymbol{B}$	$\mathcal{R}(\mathbb{C})$	$\mathcal{P}(\mathbb{C})$	$\mathcal{C}(\mathbb{C})$	$\mathcal{G}(\mathbb{C})$	$\mathcal{A}(\mathbb{C})$
$\boldsymbol{A}+\boldsymbol{B}$	$\mathcal{R}(\mathbb{C})$	$\mathcal{A}(\mathbb{C})$	$\mathcal{C}(\mathbb{C})$	$\mathcal{G}(\mathbb{C})$	$\mathcal{A}(\mathbb{C})$
$\boldsymbol{A} \times \boldsymbol{B}$	$\mathcal{I}(\mathbb{C})$	$\mathcal{P}(\mathbb{C})$	$\mathcal{I}(\mathbb{C})$	$\mathcal{I}(\mathbb{C})$	$\mathcal{I}(\mathbb{C})^*$
$-\boldsymbol{A}$	$\mathcal{R}(\mathbb{C})$	$\mathcal{P}(\mathbb{C})$	$\mathcal{C}(\mathbb{C})$	$\mathcal{G}(\mathbb{C})$	$\mathcal{A}(\mathbb{C})$
$\boldsymbol{A}^{-1}$	$\mathcal{A}(\mathbb{C})$	$\mathcal{P}(\mathbb{C})$	$\mathcal{C}(\mathbb{C})$	$\mathcal{A}(\mathbb{C})$	$\mathcal{A}(\mathbb{C})$
$\boldsymbol{A} \cap \boldsymbol{B}^\dagger$	$\mathcal{R}(\mathbb{C})$	$\mathcal{P}(\mathbb{C})$	$\mathcal{A}(\mathbb{C})$	$\mathcal{G}(\mathbb{C})$	$\mathcal{A}(\mathbb{C})$
$\boldsymbol{A} \cup \boldsymbol{B}^\dagger$	$\mathcal{G}(\mathbb{C})$	$\mathcal{A}(\mathbb{C})$	$\mathcal{A}(\mathbb{C})$	$\mathcal{G}(\mathbb{C})$	$\mathcal{A}(\mathbb{C})$

Theorem. *For* $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C} \in \mathcal{I}(\mathbb{C})$

$$\boldsymbol{A} \oplus \boldsymbol{B} = \boldsymbol{B} \oplus \boldsymbol{A}, \qquad \boldsymbol{A} \otimes \boldsymbol{B} = \boldsymbol{B} \otimes \boldsymbol{A},$$
$$(\boldsymbol{A} \oplus \boldsymbol{B}) \oplus \boldsymbol{C} = \boldsymbol{A} \oplus (\boldsymbol{B} \oplus \boldsymbol{C}), \qquad (\boldsymbol{A} \otimes \boldsymbol{B}) \otimes \boldsymbol{C} = \boldsymbol{A} \otimes (\boldsymbol{B} \otimes \boldsymbol{C}),$$
$$\boldsymbol{A} \otimes (\boldsymbol{B} \oplus \boldsymbol{C}) \subseteq (\boldsymbol{A} \otimes \boldsymbol{B}) \oplus (\boldsymbol{A} \otimes \boldsymbol{C}).$$

Also, there is no additive and multiplicative inverse element for non-degenerate complex intervals, and therefore no inverse operations either. (Intervals consisting of a single non-zero complex number have inverses.) We only have the following trivial inclusions.

$$\begin{aligned} \boldsymbol{A} \oplus (-\boldsymbol{A}) &\ni 0, \quad & (\boldsymbol{A} \oplus \boldsymbol{B}) \ominus \boldsymbol{B} &\supseteq \boldsymbol{A}, \\ \boldsymbol{A} \otimes \boldsymbol{A}^{-1} &\ni 1, \quad & (\boldsymbol{A} \otimes \boldsymbol{B}) \oslash \boldsymbol{B} &\supseteq \boldsymbol{A}. \end{aligned} \tag{1}$$

Interval types are not closed under all operations. In other words, performing an operation on intervals of a given type may result in an interval that cannot be exactly represented with that type. Table 2 summarizes the arithmetic properties of our complex interval types. These properties can be formally proven using geometric algebra, in fact, they are direct consequences of the well-known arithmetic properties of lines and circles [12]. The rigorous proof of the properties exceeds the scope of the present paper, but a review paper is under preparation, where it will be presented. In practice, the interval arithmetic algorithms determine the smallest inclusive interval around the result [4, 13, 26].

3 Polyarc interval

3.1 Concept

The boundaries of rectangular, polar, and circular intervals are all piecewise smooth Jordan curves consisting of edges and arcs. Similarly, polygonal intervals are bounded by edges by definition.

Definition. *An edge* $\mathbf{\Gamma} \in \bar{\mathcal{O}}(\mathbb{C})$ *between the points* $P_1, P_2 \in \mathbb{C}$ *can be given by the* $[0,1] \to \mathbb{C}$ *parametrization*

$$\bar{\Gamma}(P_1, P_2)(t) = (1-t)P_1 + tP_2.$$

Definition. *An arc* $\mathbf{\Gamma} \in \mathring{\mathcal{O}}(\mathbb{C})$ *centered at* $O \in \mathbb{C}$*, with radius* $r \in \mathbb{R}$ *and angular interval* $\boldsymbol{\varphi} \in \mathcal{I}(\mathbb{R})$ *is given by the* $[0,1] \to \mathbb{C}$ *parametrization*

$$\mathring{\Gamma}(O, r, \boldsymbol{\varphi})(t) = O + r \exp\left((1-t)\, i\, \underline{\varphi} + t\, i\, \overline{\varphi}\right).$$

A curve consisting of edges and arcs can precisely represent all the intervals of the mentioned types. The polyarc curve is a direct extension of the polygonal curve, where we replace the vertices with arcs keeping the implicit edges between them (Figure 2). In turn, we propose the corresponding new interval type that contains all the complex interval types above and has the same or better approximation capability for arbitrary complex intervals as the polygonal intervals.

Definition. *A polyarc curve (or polyarc)* $\mathbf{\Gamma} \in \mathring{\mathcal{J}}(\mathbb{C})$ *can be given by the* $[0, 2\mathsf{N}] \to \mathbb{C}$ *parametrization*

$$\Gamma(t) = \begin{cases} \mathring{\Gamma}_{\mathsf{n}}(t - 2\mathsf{n} + 2) & \text{for } t \in [2\mathsf{n}-2, 2\mathsf{n}-1], \\ \bar{\Gamma}_{\mathsf{n}}(t - 2\mathsf{n} + 1) & \text{for } t \in [2\mathsf{n}-1, 2\mathsf{n}], \end{cases}$$

for $\mathsf{n} \in \{1..\mathsf{N}\}$*, with alternating arc and edge pieces (either can be a single point if necessary). Precisely*

$$\begin{aligned} \mathring{\Gamma}_{\mathsf{n}}(t) &= \mathring{\Gamma}(O_{\mathsf{n}}, r_{\mathsf{n}}, \boldsymbol{\varphi}_{\mathsf{n}})(t), \\ \bar{\Gamma}_{\mathsf{n}}(t) &= \bar{\Gamma}(P_{2\mathsf{n}-1}, P_{2\mathsf{n}})(t), \end{aligned}$$

satisfying

$$P_{2\mathsf{n}-1} = O_{\mathsf{n}} + r_{\mathsf{n}} e^{i\overline{\varphi}_{\mathsf{n}}}, \quad P_{2\mathsf{n}} = O_{\mathsf{n}+1} + r_{\mathsf{n}+1} e^{i\underline{\varphi}_{\mathsf{n}+1}},$$

with $O_{\mathsf{N}+1} = O_1$, $r_{\mathsf{N}+1} = r_1$, $\boldsymbol{\varphi}_{\mathsf{N}+1} = \boldsymbol{\varphi}_1$, $P_{2\mathsf{N}} = P_0$.

The set of polyarc curves is $\mathring{\mathcal{J}}(\mathbb{C}) \subset \mathcal{J}(\mathbb{C})$.

Definition. *The polyarc intervals* $\mathcal{A}(\mathbb{C})$ *are complex intervals bounded by polyarcs.*

$$\boldsymbol{A} \in \mathcal{A}(\mathbb{C}) \iff \partial \boldsymbol{A} \in \mathring{\mathcal{J}}(\mathbb{C})$$

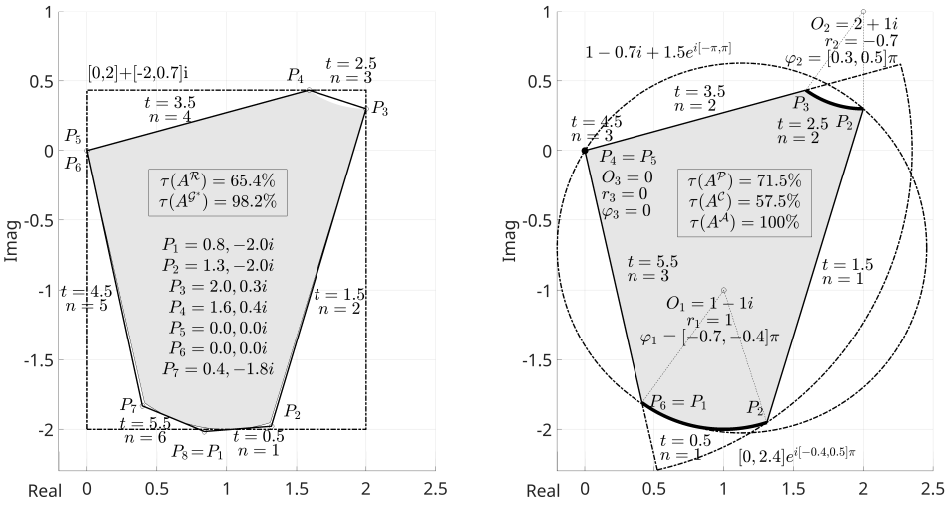


Figure 2: Example of a complex interval represented by various complex interval types. The gray area identifies the given complex interval selected to be perfectly representable by the polyarc type; solid black lines identify the convex polygonal ($\mathcal{G}^*$) and polyarc ($\mathcal{A}$) boundary curves. The grey dash-dotted lines indicate the boundaries of inclusive rectangular ($\mathcal{R}$), polar ($\mathcal{P}$) and circular ($\mathcal{C}$) interval type objects. Representation tightness $\tau(\cdot)$ is listed in text-boxes.

3.2 Properties

The properties of the polyarc interval type are directly related to that of the boundary curve, which consists of arcs and edges. Since an interval is uniquely defined by its boundary curve, arithmetic and set operations can be performed by determining the result boundary. Moreover, the result boundary can be directly calculated from the operand boundaries using the Minkowski algebra. Set operations can be performed by selecting the appropriate subset of the merged operand boundary segments forming the outer or inner boundary of the joint set, so it is trivial that the polyarc intervals will be closed under these operations. The arithmetic properties of arcs and edges can be derived from the properties of lines and circles, which has been thoroughly analyzed by Farouki et al. in [12], starting with Theorem 2.2.

Even without a rigorous analysis, which would exceed the scope of the present paper, we can assume the following properties. Since lines and circles are closed under negation, arcs and edges will be also closed under it, and thus the negative of a polyarc interval also belongs to this type. The reciprocal operation turns non-zero-crossing lines into zero-crossing circles and vice versa, therefore arcs and edges are not closed under it, but since the result boundary will consist of arcs and edges, it will also be of the polyarc type – with the obvious caveat of the complex zero we

have discussed earlier (see Figure 1).

When it comes to binary operations, the key question is whether the envelope of the curve sum or product contains anything else than arcs and edges [11]. The sum of lines has either no envelope or a line envelope when they are parallel. The sum of a line and a circle has an envelope of two lines. Last, the sum of circles has an envelope of one or two circles. We can therefore conclude that polyarc intervals are closed under the addition operation. However, the product of lines and circles may involve envelopes of parabola, hyperbole, ellipse and Cartesian oval curves, except when at least one operand is a zero-crossing line or a zero-centered circle. Therefore, polyarc intervals are not closed under multiplication. Table 2 summarizes the arithmetic and set properties of polyarc intervals. A more detailed description of these properties including their formal proof is available in the Supplementary materials.

3.3 Implementation

The polyarc interval arithmetic has been implemented using the following data types. A set of arithmetic and set operations including addition, multiplication, negative, reciprocal, union and intersection have been implemented for each data type. We used double precision floating point type (double) to represent real numbers, and two real numbers to represent complex numbers.

In our implementation real intervals are represented by storing their endpoints (2 doubles); edges are represented by storing their complex endpoints (4 doubles); arcs are represented by storing their complex type center, real type radius and real interval type argument (5 doubles). Polygonal intervals are represented by storing the ordered set of complex type vertices of the smallest bounding polygon of a given vertex count N ($2\mathsf{N}$ doubles). Finally, polyarc intervals are represented by storing the ordered set of defining arcs (see Remark 3.3) of the smallest bounding polyarc of a given arc count N ($5\mathsf{N}$ doubles).

Remark. polyarc curves consist of arcs defined by their data set and the implicit edges connecting the end-points of adjacent arcs. It is possible to suppress circular segments by setting $r_{\mathsf{n}} = 0$, while the linear segment can be suppressed by making sure that the endpoints are equal. Concave arcs can be created using negative radius values.

A polyarc interval can be efficiently stored by its defining arcs only (some of which might have zero radius[1]). However, for the complete representation of the boundary curve – which will be necessary for the arithmetic operations – the implicit edges and vertices have to be extracted (see Definition 3.1).

Other interval types (rectangular, polar, circular and polygonal) can be casted to the polyarc type using Algorithm 1. Figure 2 gives a demonstrative example of a complex interval represented by the polyarc interval type.

[1] We allow the radius of an arc to be zero to allow the representation of vertices with arc-type objects, which simplifies the implementation of the algorithms.

Algorithm 1 Casting an interval to the polyarc type

1: Decompose the interval boundary to an ordered set of edges and arcs.
2: Convert the vertices at the intersection of edges to zero-radius arcs[1].
3: Store the arcs (including the converted vertices) in a counter-clockwise order.

Remark. Roughly speaking, edges ensure the continuity of the curve, whereas vertices ensure the continuity of the direction of the curve. Since the derivative jumps at the vertices, its value there should be represented by a real interval. Therefore, we consider a vertex as a combination of a point and an angular interval, which is the same as a zero-radius arc. We can then consider vertices as implicit arcs and represent them as arc type variables. Zero-length edges, which occur when consecutive arcs meet at the same point, and redundant vertices, which can occur when a defining arc has zero radius, are removed from the curve.

Let us see how did we implement operations of polyarcs.

Negative is the simplest unary operation, since edges are closed under it and the curve derivative does not have to be considered. It is sufficient to negate the defining arcs to get the result curve. The negative of an arc is

$$-\mathring{\mathbf{\Gamma}} = -\mathring{\mathcal{O}}(\mathbb{C}\big|O, r, \boldsymbol{\varphi}) = \mathring{\mathcal{O}}(\mathbb{C}\big| -O, r, \boldsymbol{\varphi} + \pi). \tag{2}$$

Reciprocal is slightly more involved, because the reciprocal of a non-zero-crossing edge is an arc, and the reciprocal of an arc that is on a zero touching circle is an edge. The result boundary curve can be calculated following Algorithm 2.

Algorithm 2 Determining the reciprocal of a polyarc interval

1: Exclude the defining arcs with zero radius.
2: Add implicit edges to the list according to the segment order in the curve.
3: Calculate the reciprocal of each segment.
4: Store the arc type result segments and the vertices at the intersection of edges.

The reciprocal of an arc is

$$1/\mathring{\mathbf{\Gamma}} = 1/\mathring{\mathcal{O}}(O, r, \boldsymbol{\varphi}) = \begin{cases} \bar{\mathcal{O}}(\mathbb{C}\big|1/\mathring{\Gamma}(0), 1/\mathring{\Gamma}(1)) \\ \mathring{\mathcal{O}}(\mathbb{C}\big|O^*, r^*, [\angle(1/\mathring{\Gamma}(0) - O^*), \angle(1/\mathring{\Gamma}(1) - O^*)]), \end{cases} \tag{3}$$

where $O^* = \mathring{\nu}/(1 - (r|\mathring{\nu}|)^2)$ and $r^* = -r|\mathring{\nu}|^2/(1 - (r|\mathring{\nu}|)^2)$ are the center and radius of the reciprocal arc respectively, and $\mathring{\nu} = 1/O$ is the normalization coefficient that scale-rotates the operand arc on a real one centered circle. The reciprocal of an edge is

$$1/\bar{\mathbf{\Gamma}} = 1/\bar{\mathcal{O}}(\mathbb{C}\big|P_1, P_2) = \begin{cases} \bar{\mathcal{O}}(\mathbb{C}\big|1/P_1, 1/P_2) & \text{if zero-crossing,} \\ \mathring{\mathcal{O}}(\bar{\nu}/2, |\bar{\nu}|/2, [\angle(1/P_1 - \bar{\nu}/2), \angle(1/P_2 - \bar{\nu}/2)] & \text{else,} \end{cases} \tag{4}$$

where $a = \mathrm{Im}(P_1 - P_2)/\mathrm{Re}(P_1 - P_2)$ is the slope of the operand edge and $\bar{\nu} = \exp(-i \arctan(a + \pi/2)/(P_1 \cos(\arctan(a + \pi/2)))$ is the normalization coefficient that scale-rotates the operand edge on a real one crossing vertical line.

Trimming is a special unary operation that turns a self-intersecting curve into a Jordan one by extracting the inner or outer boundary. It is an indispensable part of the implementation of the binary operations: addition, multiplication, union and intersection. For our purposes, we implemented a simplified trimming method shown in Algorithm 3. The method is based on the idea of following the boundary curve in the counter-clockwise direction and turning right at each intersection if the outer boundary is required; and turning left if the inner boundary is required. This method cannot identify holes in the interval, which can occur for example when two polar intervals with wide angular intervals are multiplied. The resulted annulus is turned into a circle by the simplified trimming algorithm. For a more sophisticated algorithm see [11].

Algorithm 3 Trimming a self-intersecting polyarc curve

1: Split the boundary curve segments where they intersect each other.
2: Select and store the curve segment containing the smallest real valued point.
3: Find segments with start-points equal to the end-point of the selected one.
4: Select and store the segment with the lowest or highest starting Gauss map value depending whether the outer or inner boundary is to be extracted.
5: Repeat step 3 and 4 until the end-point of the selected segment equals the start-point of an already stored segment.
6: If the inner boundary is extracted, remove the stored segments until the one that is connected to the last one.
7: Keep the stored arcs and the vertices at the intersection of stored edges as defining arcs for the result.

Union can be directly implemented using the trimming method by choosing the outer boundary option. The result is only valid if the operand intervals are connected.

Intersection is implemented by extracting the inner boundary with the trimming method.

Addition is the simpler of the two binary operations, because the result boundary consists solely of arcs and edges. It requires the extraction of the implicit edges and vertices (arcs). The result boundary segments are a subset of the pair-wise sum of the operand boundary segments including the edges and vertices. We identify this subset using the Gauss map matching and trimming operations.

Gauss map matching identifies those pairs of operand boundary segments that contribute to the result boundary. It is based on the observation that when two curve segments (edge or arc) are added, the normal angle interval of the result is the intersection of that of the operands. The Gauss map of a curve can be seen as

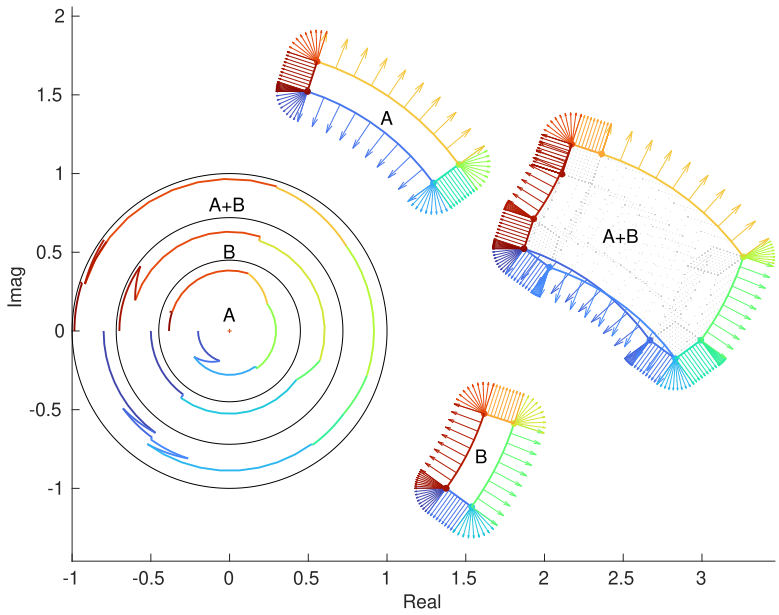


Figure 3: Two polar intervals, $\boldsymbol{A}$ and $\boldsymbol{B}$, and their polyarc sum before trimming. The color of the arcs and the normal vectors indicate the Gauss map value (e.g. $\gamma(\Gamma_A, t)$). The Gauss maps are presented as three concentric curves in the unit circle (with the result outside), where the angle and color indicates the Gauss map value γ, and the radius indicates the curve parameter t. (The radii has been offset to form the concentric rings on the left.)

its normal angle as a function of the curve parameter. A more detailed discussion about Gauss map matching is available in the Supplementary materials.

Definition. *The Gauss map interval of a regular curve segment* $\Gamma(\boldsymbol{t})$ *is*

$$\boldsymbol{\gamma}_{\Gamma,\boldsymbol{t}} = \gamma(\Gamma, \boldsymbol{t}) = \angle i\Gamma'(\boldsymbol{t}).$$

Proposition. *For two boundary segments* $\boldsymbol{\Gamma}_{A,\mathsf{n}} \subset \partial \boldsymbol{A}$ *and* $\boldsymbol{\Gamma}_{B,\mathsf{k}} \subset \partial \boldsymbol{B}$

$$\boldsymbol{\Gamma}_{A,\mathsf{n}} \oplus \boldsymbol{\Gamma}_{B,\mathsf{k}} \subset \partial(\boldsymbol{A} \oplus \boldsymbol{B}) \implies \boldsymbol{\gamma}_{\Gamma_{A,\mathsf{n}}} \cap \boldsymbol{\gamma}_{\Gamma_{B,\mathsf{k}}} \neq \emptyset.$$

Since the Gauss-map value is constant along edges, the sum of two edges $\bar{\boldsymbol{\Gamma}}_A = \bar{\mathcal{O}}(\mathbb{C}\big|P_{A,1}, P_{A,2})$ and $\bar{\boldsymbol{\Gamma}}_B = \bar{\mathcal{O}}(\mathbb{C}\big|P_{B,1}, P_{B,2})$ is only part of the result boundary if they are parallel and have the same direction, in which case the result edge is

$$\bar{\boldsymbol{\Gamma}}_A + \bar{\boldsymbol{\Gamma}}_B \underset{\gamma_{\Gamma_A}=\gamma_{\Gamma_B}}{=} \bar{\mathcal{O}}(\mathbb{C}\big|P_{A,1} + P_{B,1}, P_{A,2} + P_{B,2}). \tag{5}$$

The Gauss-map value of an arc (or vertex) $\mathring{\mathbf{\Gamma}}_B = \mathring{\mathcal{O}}(\mathbb{C}\big|O_B, r_B, \boldsymbol{\varphi}_B)$ is always changing monotonously along the segment, therefore it is only a single point on the arc that matches the Gauss map of the operand edge, in which case the result segment is the operand edge translated by this point on the arc.

$$\bar{\mathbf{\Gamma}}_A + \mathring{\mathbf{\Gamma}}_B \underset{\gamma_{\Gamma_A} \in \gamma_{\Gamma_B}}{=} \bar{\mathcal{O}}(\mathbb{C}\big|P_{A,1}+O_B+r_B e^{i\gamma_{\Gamma_B}}, P_{A,2}+O_B+r_B e^{i\gamma_{\Gamma_B}}) \tag{6}$$

If we consider the Gauss map of an arc (or vertex) as an interval, then the matching of two arcs results the intersection of these intervals, which in some cases can consist of two disconnected intervals due to the angular wrapping (e.g. $[0.8, 2.2]\pi \cap [-0.2, 1.2]\pi$). The Gauss-map matched sum of two arcs $\mathring{\mathbf{\Gamma}}_A$ and $\mathring{\mathbf{\Gamma}}_B$ then results zero, one or two arcs that is

$$\mathring{\mathbf{\Gamma}}_A + \mathring{\mathbf{\Gamma}}_B \underset{\gamma_{\Gamma_A} \cap \gamma_{\Gamma_B} \neq \emptyset}{=} \mathring{\mathcal{O}}(\mathbb{C}\big|O_A + O_B, r_A + r_B, \boldsymbol{\varphi}_A \cap \boldsymbol{\varphi}_B) \tag{7}$$

Figure 3 shows the addition of two polyarc intervals.

A simplified addition algorithm has been implemented for convex operands. A convexity test can be implemented by checking if any of the arcs have negative radius and if any of the vertices have higher Gauss map value at the preceding segment than the following one. The algorithm can then follow the *MinkowskiSum* algorithm in [7, Ch. 13] with the only modification that it uses the Gauss map values instead of the edge angles and instead of adding vertices it adds arcs as in (7). Since the sum of convex intervals has a simple boundary, the trimming operation is unnecessary and the Gauss-map matching is simplified from the quadratic time complexity to linear. Also, because the edges resulting from adding an operand edge to a curve segment of the other operand implicitly results from adding the operand vertices to the other operand segments, the extraction of the implicit edges is unnecessary. In our code implementation, we created a special interval type optimized for fast addition of convex polyarc intervals. The type, called *polyarx interval*, uses a different storage method to avoid the extraction step of the implicit vertices.

Multiplication is similar to the addition, because the product can be seen as addition in logarithmic domain. However, while the envelope [5] of the set formed by the addition of arcs and edges is always bounded by arcs or edges, this is not true for the set formed by multiplying arcs and edges [12]. Therefore, the boundary of two intervals' product can contain segments that are neither arcs nor edges, where the arc-edge product set touches the envelope. This interval cannot be exactly represented by a polyarc curve, but needs to be approximated. To find the subset of pair-wise product of operand boundary segments that are on the result boundary, we used the log-Gauss map matching and trimming algorithms. The log-Gauss matching is based on the fact that the log-Gauss map interval of a given segment in the product interval boundary is the intersection of the log-Gauss map intervals of the corresponding operand curve segments. Similar to the addition algorithm, this allows to extract the boundary curve segments by eliminating operand pairs with

non-overlapping log-Gauss map intervals. Our algorithm also contains the analytical functions for determining whether a given curve segment touches the envelope, in which case it will be apporoximated by an arc. Figure 4 shows an example of the product of two polyarc intervals in both the cartesian and logarithmic domains, and the log-Gauss map of the boundary curve of both operands and the result.

Definition. *The log-Gauss map interval of a regular curve segment* $F(\boldsymbol{t})$ *is*

$$\mathring{\boldsymbol{\gamma}}_{F,\boldsymbol{t}} = \mathring{\gamma}(F,\boldsymbol{t}) = \gamma(\log F,\boldsymbol{t}) = \angle\frac{F'(\boldsymbol{t})}{F(\boldsymbol{t})} = \boldsymbol{\gamma_{F,t}} - \angle F(\boldsymbol{t}).$$

Proposition. *For two boundary segments* $\boldsymbol{\Gamma}_{A,\mathsf{n}} \subset \partial\boldsymbol{A}$ *and* $\boldsymbol{\Gamma}_{B,\mathsf{k}} \subset \partial\boldsymbol{B}$

$$\boldsymbol{\Gamma}_{A,\mathsf{n}} \otimes \boldsymbol{\Gamma}_{B,\mathsf{k}} \subset \partial(\boldsymbol{A} \otimes \boldsymbol{B}) \implies \mathring{\boldsymbol{\gamma}}_{\Gamma_{A,\mathsf{n}}} \cap \mathring{\boldsymbol{\gamma}}_{\Gamma_{B,\mathsf{k}}} \neq \emptyset.$$

Algorithm 4 Determining the product of arcs and edges in the general case

1: Determine the normalized product curve from the operand parameters.
2: Determine the intersection of the operands' log-Gauss map intervals.
3: Find the curve segment corresponding to the log-Gauss map intersection.
4: If the result segment does not touch the envelope, it is an arc, jump to step 6.
5: Fit an arc to the curve, so it is touching from the outside (see Algorithm 5).
6: Scale-rotate the result arc using the reciprocal of the normalization factor.

The special cases of edge and arc multiplication (when at least one operand is on a zero-crossing line or a zero-centered circle) can be simply implemented using addition in the logarithmic domain, because the logarithmic image of a zero-crossing edge and a zero-centered arc are both edges. The only difficulty is finding the point on the curve segment that corresponds to a given log-Gauss map, which is less straightforward than in case of the Gauss map value. In our implementation we used the built-in Matlab function *vpasolve* to numerically solve the inverse parametric equation of the curve segment[2]. The implementation of the general cases of edge and arc products require the geometric algebra described by Farouki et al in [12], and the normalization uses Theorem 2.2. Here, the product function is derived for the normalized (real one crossing vertical) line and (real one centered) circle. Since the normalization does not change the log-Gauss map value, we can determine the result curve the following Algorithm 4.

A simplified multiplication algorithm can be implemented when both operands are convex in the logarithmic domain. A convexity test can be implemented by checking if any of the arcs have negative radius, any of the edges have decreasing absolute value, or any of the vertices have higher log-Gauss map value at the preceding segment than the following one. The simplified algorithm works similar to the *MinkowskiSum* algorithm in [7, Ch. 13] with the modification that it should use the log-Gauss map value instead of the edge angle, and instead of adding vertices

[2] We solve the $\mathring{\gamma}(F,t_0) = \gamma_0$ equality to get t_0.

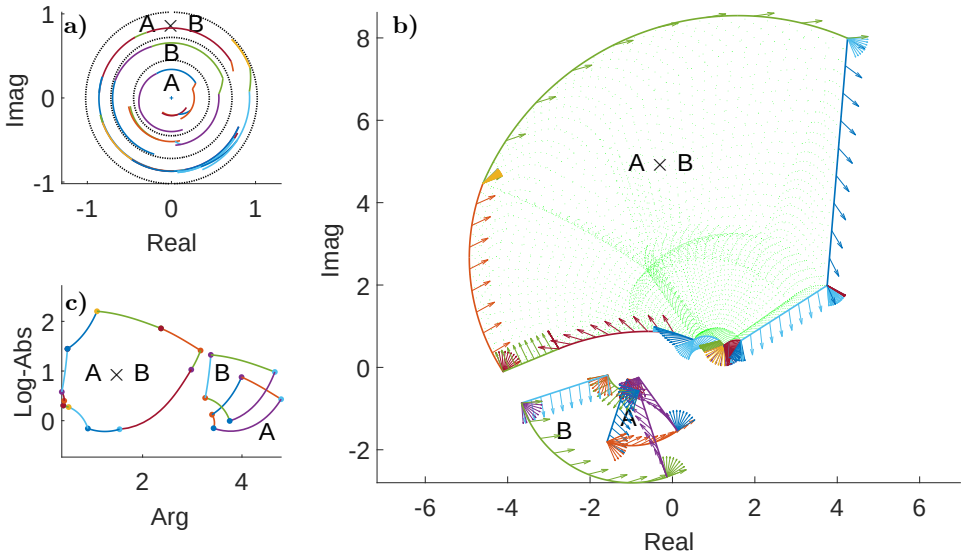


Figure 4: Two polar intervals , $\boldsymbol{A}$ and $\boldsymbol{B}$, and and their polyarc product after trimming. **a)** shows the log-Gauss maps presented as three concentric curves in the unit circle, where the angle and color indicates the log-Gauss map value $\mathring{\gamma}$, and the radius indicates the position on the curve t. (The radii has been offset to form the concentric rings.) One can observe that the result segments located in the outer ring are at the angular intersection of the operand segments in the inner rings. **b)** shows the intervals in the complex plane, where color of the arcs indicate the log-Gauss map value (e.g. $\mathring{\gamma}(\Gamma_A, t)$), and the arrows indicate the log-normal of the curves, which is the normal angle minus the angle of the vector pointing to the point of the curve from the origin. The colors and arrows allow the identification of the operand segment pairs that contribute to a given result boundary segment. The green point cloud indicates samples from the result interval formed by multiplying samples from the operand boundaries. **c)** shows the intervals in the complex logarithmic space, where $\log(\boldsymbol{A}\times\boldsymbol{B}) = \log(\boldsymbol{A}) + \log(\boldsymbol{B})$. One can observe that the result in this domain is the sum of the operands. While not indicated, the log-normal angle of a curve at a given point in this domain is the normal angle of the curve.

it should multiply operand segments with log-Gauss-map matching. The products of arcs and edges would be too long to present in this paper. The method used for fitting an arc to the product envelope curve is described in Algorithm 5 [3] . The implementation was derived from the results in [12] and can be found in the open-source code (see Supplementary materials).

[3]To ensure inclusive bounds, the arc is fitted from the "outside", which should be interpreted as the right hand side of the curve while facing in the increasing curve parameter direction.

Algorithm 5 Fitting an arc to a product curve segment touching the envelope

1: Determine the implicit function of the envelope curve.
2: Find the endpoints of the curve segment by numerically solving it for the log-Gauss map parameter bounds of the operands.
3: Find the center of the circle that goes through the endpoints and touches the envelope segment from the outside by numerically solving an inequality.
4: The fitted arc is defined by the center and the two endpoints.

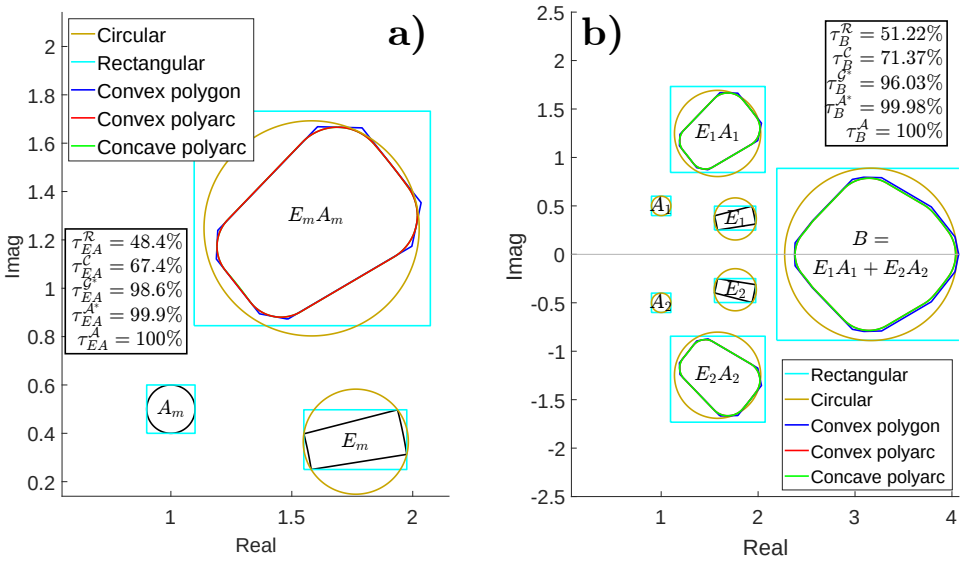


Figure 5: Example of a sensor array tolerance analysis using various complex interval types. **a)** shows the multiplication of a circular ($\boldsymbol{A}_{\mathsf{m}}^{\mathcal{C}} = \sum_{\mathsf{n}} \boldsymbol{A}_{\mathsf{m,n}}^{\mathcal{C}}$) and a polar ($\boldsymbol{E}_{\mathsf{m}}^{\mathcal{P}}$) interval. **b)** shows the summation of two such circular-polar products. Each plot contains a text-box with the tightness of the rectangular ($\mathcal{R}$), circular ($\mathcal{C}$), convex polygonal ($\mathcal{G}^*$), convex polyarc ($\mathcal{A}^*$) and concave polyarc ($\mathcal{A}$) representations of the result intervals ($\boldsymbol{E}_{\mathsf{m}}\boldsymbol{A}_{\mathsf{m}}$ and $\boldsymbol{B}$).

4 Case studies

4.1 Sensor array tolerance analysis

In this section, we present a case study that motivated the development of the polyarc interval type. The tolerance analysis of antenna arrays using interval analysis is an active research area within sensor array design and signal processing [18, 17]. We have previously analyzed the worst-case spatial response of acoustic arrays affected by calibration errors and mutual coupling using rectangular, circular and

polygonal interval [2]. None of these types could provide an exact representation of the complex interval of the array response, which resulted in relaxed bounds. Looking for a tighter solution, we found that given the physical model, the polyarc interval type yields an exact bound, which we demonstrate in the following.

Let $\{\boldsymbol{E}_{\mathsf{m}} \in \mathcal{P}(\mathbb{C}) \big| \mathsf{m} \in \{1..\mathsf{M}\}\}$ represent the combined amplitude and phase sensitivity interval of the individual elements of a sensor array, and let $\{\boldsymbol{A}_{\mathsf{m},\mathsf{n}} \in \mathcal{C}(\mathbb{C}) \big| \mathsf{m},\mathsf{n} \in \{1..\mathsf{M}\}\}$ represent the coupling coefficient interval of each pair of elements, including the self-coupling $\boldsymbol{A}_{\mathsf{n},\mathsf{n}} = 1$. Then, assuming a narrow-band, far-field operation, the complex response interval of the array is

$$\boldsymbol{B} = \sum_{\mathsf{m}} \boldsymbol{E}_{\mathsf{m}} \sum_{\mathsf{n}} \boldsymbol{A}_{\mathsf{m},\mathsf{n}}. \tag{8}$$

Since the boundary segments of the polar intervals are all from zero crossing lines or zero-centered circles, the product of a circular and a polar interval can be exactly represented using edges and arcs. Finally, the sum of polyarc intervals is polyarcular; therefore, the complex response interval is of the polyarc type: $\boldsymbol{B} \in \mathcal{A}(\mathbb{C})$.

While rectangular and circular types are computationally light, they introduce a significant loss of tightness at the type-casting and through the multiplication [29, 1]. The convex polygonal type can approximate the convex arcs with arbitrary precision, but this comes with a price of increased computational complexity [28]. We found that the polyarc type is an ideal choice for this application, because its concave implementation can provide perfect tightness, while its convex implementation provides tighter and faster calculation than the polygonal interval arithmetic. Figure 5 shows an example of the complex beampattern interval representation. A more detailed version of this case study, including run-time and accuracy comparison of the polyarc interval arithmetic with a number of other methods is available in [14].

4.2 Robot localization

Desrochers et al. introduce a measurement problem, in which the localization of a robot is performed by measuring the distance and direction of three landmarks with some tolerance [8]. In their paper, they use contractor programming, which is an interval analysis technique – for the outer approximation of the solution set, which is the region containing the possible locations. Since the problem can be also formulated as a complex interval arithmetic problem, in this section, we show a solution using the polyarc interval.

Let $\boldsymbol{r}_{\mathsf{n}} \in \mathcal{I}(\mathbb{C})$ represent the interval of range measurement and $\boldsymbol{\varphi}_{\mathsf{n}} \in \mathcal{I}(\mathbb{C})$ represent the interval of direction measurement of the n^{th} landmark at the known location $P_{\mathsf{n}} \in \mathbb{C}$. We can then represent the set of possible relative locations of the n^{th} landmark as a polar interval $\boldsymbol{A}_{\mathsf{n}} = \boldsymbol{r}_{\mathsf{n}} e^{i\boldsymbol{\varphi}_{\mathsf{n}}} \in \mathcal{P}(\mathbb{C})$. Given N landmarks, we can determine the set of the possible locations of the robot based on the combination of the measurement as

$$\boldsymbol{B} = \bigcap_{\mathsf{n}=1}^{\mathsf{N}} (P_{\mathsf{n}} - \boldsymbol{A}_n) \in \mathcal{A}(\mathbb{C}). \tag{9}$$

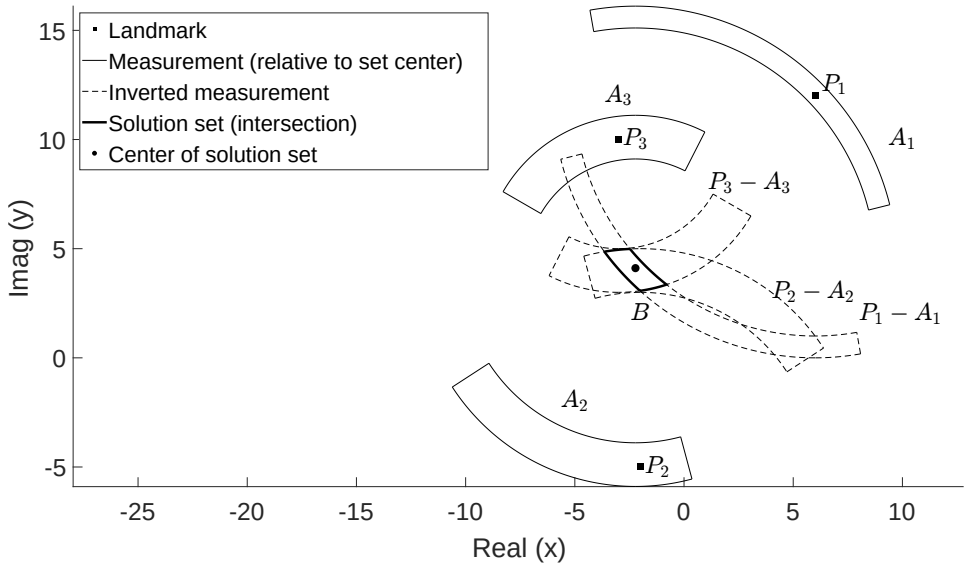


Figure 6: Example of a robot location problem solved with polyarc interval type given the landmark locations $P = \{6 + 12i, -2 + 5i, -3 + 10i\}$, range measurements $\boldsymbol{r} = \{[11, 12], [8, 10], [5, 7]\}$, and direction measurements $\boldsymbol{\varphi} = \{[14, 100], [-147, -75], [63, 150]\}^\circ$.

Since the result boundary consist of the boundary segments of translated polar intervals, it can be exactly representing using a polyarc curve. Figure 6 the example based on the paper [8]. It has to be noted that the intersection operation can result multiple non-connected intervals, as would be the case in this example if the third landmark would not be present. Such cases can be handled by combining contractors with a branch and prune algorithm. This, however, is not available in our current reference implementation.

5 Conclusion

In this paper, we showed that all commonly used complex interval types can be represented and arithmetically combined using the polyarc interval type with improved tightness and similar complexity as of the polygonal method. This makes the polyarc type a valuable option for performing calculations in various cases of interval analysis. We have shown that, similar to the polygonal type, simple arithmetic operations on convex polyarc intervals can be also performed with a computational complexity linearly dependent on the number of elements constituting the operand boundaries.

We presented an implementation of the polyarc interval arithmetic and provided an open-source code library as a reference (see Supplementary materials). We presented a case study from antenna design that served as our motivation to develop this method. We showed that in this particular case polyarc interval outperforms the existing methods, when high accuracy is desired. A second case study has been presented from robotics, where polyarc intervals provide an alternative to contractors. These examples show that the polyarc interval type is applicable in practical design tasks.

The reference implementation provided with this paper is for demonstrating the algorithms and generating reproducible experiments, but not intended for production use. The development of an open-source package in a non-proprietary language (e.g. Python) that would conform with the 1788-2015 - IEEE Standard for Interval Arithmetic is desirable, and would be natural continuation of the presented work.

When large number of intervals are combined by binary operations, the simplification of the result boundary is often desired to avoid the explosion of boundary element count. The removal of collinear vertices is an efficient technique to simplify polygons, which could be extended to polyarcs in the future by identifying joinable edges and arcs.

In our current implementation of the polyarc multiplication, we use the built-in Matlab function *vpasolve* to determine points on the result curves with a given log-Gauss map value, and to find the tightest arc approximating the non-polyarcular segments. Finding closed form solutions to these functions could significantly increase the speed of the algorithm.

Ironically, one of the most significant challenges in the implementation is the propagation of numerical errors, which is the origin of interval arithmetic. Such errors can make tests of inclusion and equality unreliable, which are indispensable for the geometric calculations with arcs and edges. While we could handle this problem for the majority of cases by setting tolerances based on our observations, a thorough analysis of an error propagation could result in a more robust implementation.

The derivation and implementation of non-linear functions on complex intervals is also desirable, and a potential direction for future research. Furthermore, a more detailed study of the robitic localization problem could be of interest.

Supplementary materials

More details about the polyarc arithmetic, including the formal proof of its arithmetic properties is available in a preprint document at https://arxiv.org/abs/2402.06430.

Our reference implementation of real and complex interval arithmetic methods is available as a Matlab package both in the live repository at https://github.com/unioslo-mn/ifi-complex-interval-arithmetic and in the frozen release at [3]. The package contains classes for real intervals, rectangular, polar, circular, convex polygonal, convex polyarc and concave polyarc type complex intervals.

References

[1] Anselmi, N., Salucci, M., Rocca, P., and Massa, A. Power pattern sensitivity to calibration errors and mutual coupling in linear arrays through circular interval arithmetics. *Sensors*, 16(6):791, 2016. DOI: 10.3390/s16060791.

[2] Arnestad, H. K., Geréb, G., Lønmo, T. I. B., Kirkebø, J. E., Austeng, A., and Näsholm, S. P. Worst-case analysis of array beampatterns using interval arithmetic. *The Journal of the Acoustical Society of America*, 153(6):1–7, 2023. DOI: 10.1121/10.0019715.

[3] Arnestad, H. K. and Geréb, G. Beampattern Interval Analysis Toolbox, 2022. DOI: 10.5281/zenodo.6856232.

[4] Boche, R. E. Complex interval arithmetic with some applications. Technical Report LMSC4-22-66-1, Locheed Missiles and Space, 1966. URL: https://www.reliable-computing.org/archive/Moores_early_papers/Boche_complex.pdf.

[5] Bruce, J. W. and Giblin, P. J. What is an envelope? *The Mathematical Gazette*, 65(433):186–192, 1981. DOI: 10.2307/3617131.

[6] Dawood, H. *Theories of Interval Arithmetic: Mathematical Foundations and Applications*. LAP LAMBERT Academic Publishing Gmbh & Co. KG, 2011. ISBN: 9783846501542.

[7] de Berg, M., Cheong, O., van Kreveld, M., and Overmars, M. *Computational Geometry: Algorithms and Applications*. Springer, Berlin Heidelberg, 2008. DOI: 10.1007/978-3-540-77974-2.

[8] Desrochers, B. and Jaulin, L. A minimal contractor for the polar equation: Application to robot localization. *Engineering Applications of Artificial Intelligence*, 55:83–92, 2016. DOI: 10.1016/j.engappai.2016.06.005.

[9] Dongarra, J. J. BLAS (Basic Linear Algebra Subprograms), 1995. URL: https://netlib.org/blas/.

[10] Farouki, R. T., Han, C. Y., and Hass, J. Boundary evaluation algorithms for Minkowski combinations of complex sets using topological analysis of implicit curves. *Numerical Algorithms*, 40(3):251–283, 2005. DOI: 10.1007/s11075-005-4565-9.

[11] Farouki, R. T., Moon, H. P., and Ravani, B. Algorithms for Minkowski products and implicitly-defined complex sets. *Advances in Computational Mathematics*, 13(3):199–229, 2000. DOI: 10.1023/A:1018910412112.

[12] Farouki, R. T., Moon, H. P., and Ravani, B. Minkowski geometric algebra of complex sets. *Geometriae Dedicata*, 85(1):283–315, 2001. DOI: 10.1023/A:1010318011860.

[13] Gargantini, I. and Henrici, P. Circular arithmetic and the determination of polynomial zeros. *Numerische Mathematik*, 18(4):305–320, 1971. DOI: 10.1007/BF01404681.

[14] Geréb, G., Arnestad, H. K., Lønmo, T. I. B., Kirkebø, J. E., Näsholm, S. P., and Austeng, A. Exact power pattern bounds for arrays affected by calibration errors and mutual coupling. *IEEE Transactions on Antennas and Propagation*, pages 1–1, 2025. DOI: 10.1109/TAP.2025.3570168.

[15] Giardina, C. and Dougherty, E. *Morphological methods in image and signal processing*. Society for Industrial and Applied Mathematics, 1988. DOI: 10.1137/1032073.

[16] Hansen, E. R. A generalized interval arithmetic. In Nickel, K., editor, *Interval Mathematics*, Lecture Notes in Computer Science, pages 7–18, Berlin, Heidelberg, 1975. Springer. DOI: 10.1007/3-540-07170-9_2.

[17] He, G., Gao, X., and Zhang, R. Impact analysis and calibration methods of excitation errors for phased array antennas. *IEEE Access*, 9:59010–59026, 2021. DOI: 10.1109/ACCESS.2021.3073222.

[18] He, G., Gao, X., Zhou, H., and Zhu, H. Comparison of three interval arithmetic-based algorithms for antenna array pattern upper bound estimation. *Electronics Letters*, 55(14):775–776, 2019. DOI: 10.1049/el.2019.1229.

[19] Jaulin, L., Kieffer, M., Didrit, O., and Walter, E. *Applied Interval Analysis: With Examples in Parameter and State Estimation, Robust Control and Robotics*. Springer, Berlin, 2001. DOI: 10.1007/978-1-4471-0249-6.

[20] Kearfott, R. E., Musaev, E. A., Nesterov, V. M., and Yakovlev, A. G. Reliable Computing | Volumes and issues, 1995. URL: https://link.springer.com/journal/11155/volumes-and-issues.

[21] Kreinovich, V. and Lauter, C. Interval Computations, 2023. URL: https://www.cs.utep.edu/interval-comp/main.html.

[22] Moore, R. E. *Interval arithmetic and automatic error analysis in digital computing*. PhD thesis, Stanford University California, 1963.

[23] Moore, R. E., Kearfott, R. B., and Cloud, M. J. *Introduction to Interval Analysis*. SIAM, Philadelphia, 2009. ISBN: 9780898716696.

[24] Ohta, Y., Gong, L., and Haneda, H. Polygon interval arithmetic and design of robust control systems. In *Proceedings of the 29th IEEE Conference on Decision and Control*, pages 1065–1067 vol.2, 1990. DOI: 10.1109/CDC.1990.203765.

[25] Ohta, Y. Nonconvex polygon interval arithmetic as a tool for the analysis and design of robust control systems. *Reliable Computing*, 6(3):247–279, 2000. DOI: 10.1023/A:1009926413485.

[26] Petkovic, M. and Petkovic, L. D. *Complex Interval Arithmetic and Its Applications*. John Wiley & Sons, Berlin, 1998. ISBN: 9783527401345.

[27] Rump, S. M. INTLAB — INTerval LABoratory. In Csendes, T., editor, *Developments in Reliable Computing*, pages 77–104. Springer Netherlands, Dordrecht, 1999. DOI: 10.1007/978-94-017-1247-7_7.

[28] Tenuti, L., Anselmi, N., Rocca, P., Salucci, M., and Massa, A. Minkowski sum method for planar arrays sensitivity analysis with uncertain-but-bounded excitation tolerances. *IEEE Transactions on Antennas and Propagation*, 65(1):167–177, 2017. DOI: 10.1109/TAP.2016.2627548.

[29] Zhang, Y., Zhao, D., Wang, Q., Long, Z., and Shen, X. Tolerance analysis of antenna array pattern and array synthesis in the presence of excitation errors. *International Journal of Antennas and Propagation*, 2017. DOI: 10.1155/2017/3424536.

Contents

ISSN 0324—721 X (Print)
ISSN 2676—993 X (Online)

Editor-in-Chief: Boglárka G.-Tóth