Volume 26 Number 4

ACTA CYBERNETICA

Editor-in-Chief: Tibor Csendes (Hungary)

Managing Editor: Boglárka G.-Tóth (Hungary)

Assistant to the Managing Editor: Attila Tanács (Hungary)

Associate Editors:

Michał Baczyński (Poland)
Hans L. Bodlaender (The Netherlands)
Gabriela Csurka (France)
János Demetrovics (Hungary)
József Dombi (Hungary)
Rudolf Ferenc (Hungary)
Zoltán Gingl (Hungary)
Tibor Gyimóthy (Hungary)
Zoltan Kato (Hungary)
Dragan Kukolj (Serbia)
László Lovász (Hungary)
Kálmán Palágyi (Hungary)
Dana Petcu (Romania)
Andreas Rauh (Germany)
György Vaszil (Hungary)

Szeged, 2024

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/.

EDITORIAL BOARD

Editor-in-Chief:

Tibor Csendes
Department of Computational Optimization
University of Szeged, Hungary
csendes@inf.u-szeged.hu

Managing Editor:

Boglárka G.-Tóth
Department of Computational Optimization
University of Szeged, Hungary
boglarka@inf.u-szeged.hu

Assistant to the Managing Editor:

Attila Tanács
Department of Image Processing
and Computer Graphics
University of Szeged, Hungary
tanacs@inf.u-szeged.hu

Associate Editors:

Michał Baczyński
Faculty of Science and Technology,
University of Silesia in Katowice,
Poland
michal.baczynski@us.edu.pl

Hans L. Bodlaender
Institute of Information and
Computing Sciences, Utrecht
University, The Netherlands
h.l.bodlaender@uu.nl

Gabriela Csurka
Naver Labs, Meylan, France
gabriela.csurka@naverlabs.com

János Demetrovics
MTA SZTAKI, Budapest, Hungary
demetrovics@sztaki.hu

József Dombi
Department of Computer Algorithms
and Artificial Intelligence, University of
Szeged, Hungary
dombi@inf.u-szeged.hu

Rudolf Ferenc
Department of Software Engineering,
University of Szeged, Hungary
ferenc@inf.u-szeged.hu

Zoltán Gingl
Department of Technical Informatics,
University of Szeged, Hungary
gingl@inf.u-szeged.hu

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

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

Dana Petcu
Department of Computer Science, West University of Timisoara, Romania
petcu@info.uvt.ro

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 2022 and 2023

Guest Editors

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

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

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

Steffen Schön
Leibniz Universität Hannover, Germany
schoen@ife.uni-hannover.de

Preface

The Summer Workshop on Interval Methods (SWIM) is an annual scientific meeting initiated in 2008 with a special focus on promoting interval analysis techniques and their applications to a broader community of researchers. Since 2008, SWIM has become a multi-disciplinary keystone event for researchers dealing with various aspects of interval and set-based methods.

In 2022 and 2023, the 13th and 14th editions in this workshop series were held at the Leibniz Universität Hannover (Germany) and at Polytech Angers (France), respectively. Both events had a focus on research topics in the fields of (control) engineering, computer science, and mathematics. A total of almost 50 talks were given during both workshops, covering the following areas:

- verified solution of initial value problems for dynamic system models,
- scientific computing with guaranteed error bounds,
- design of robust and fault-tolerant control systems,
- modeling and quantification of errors in engineering tasks,
- implementation of software libraries, and
- usage of the aforementioned approaches for system models in various fields of application such as control engineering, robotics, navigation, data analysis, machine learning, and signal processing.

After a peer-review process, 10 high-quality articles were selected for publication in this special issue. These contributions cover set-valued approaches for the online identification of battery systems, interval-based implementations of nonlinear model-predictive control, robust control and actuator fault detection based on linear matrix inequalities, verified bit and power allocation for MIMO systems, the implementation of software libraries for contractor-based modeling, constraint programming for the simulation of ordinary differential equations, GPU-accelerated interval-based parameter identification, hardware acceleration of interval contractor primitives, and the design of novel approaches for the implementation of complementary contractors as well as asymptotically minimal contractors based on centered form representations.

Luc Jaulin, Sébastien Lahaye,
Andreas Rauh, Steffen Schön

Guest Editors

In memoriam of Nicolas Delanoue 1980–2023

This special issue is dedicated to the memory of Nicolas Delanoue, a brilliant mathematician and good friend, who was the first to propose the use of interval methods for topology with applications to robotics and more generally to the analysis and control of dynamical systems.

Many of us, in the interval community, had the privilege to work with him, to listen to his wonderful seminars and to have long and fruitful discussions with Nicolas.

His sudden departure leaves a void that will never be filled.

However, his availability and talent for explaining mathematical concepts and making them intuitive (using pictures and programs), his willing to formalize complex problems in a simple manner, the passion he shared for the beauty of mathematics, and his positive spirit, will remain a model for many of us.

May his legacy continue to inspire curiosity, creativity, and the promotion of interval tools to solve ambitious problems.

Acta Cybernetica **26** (2024) 755–774.

A Constraint Programming Approach for Polytopic Simulation of Ordinary Differential Equations — A Collision Detection Application*

Julien Alexandre dit Sandretto[ab], Alexandre Chapoutot[ac], Christophe Garion[de], and Xavier Thirioux[df]

Abstract

This paper presents a constraint-based approach to compute the reachable tube of nonlinear differentiable equations. A set of initial values for the equations is considered and defined by a polytope represented as intersections of zonotopes. Guaranteed numerical integration based on zonotopic computation is used to compute reachable tubes. In order to efficiently build polytopes defined by the intersection of several zonotopes, we use a previously developed abstract domain [27] to represent reachable tubes. The proposed contribution allows to compute more expressive reachable tubes more efficiently than methods based only on boxes, and therefore could improve verification/validation processes in robotics application for example. The approach is evaluated on examples taken from literature and we present two applications of this work.

Keywords: constraint programming, abstract domains, ordinary differential equations, cyber-physical systems, abstract interpretation

1 Context and state of the art

Cyber-physical systems (CPS) are systems in which software and physical parts interoperate deeply. The physical part of these systems is often modeled by differential equations. When properties have to be verified on these systems, for instance the feasibility or the safety of a mission assigned to a robot, the solution of such differential equations is generally required. Even if Ordinary Differential Equations

*This work was supported by the Defense Innovation Agency (AID) of the French Ministry of Defense (research project N° 2018 60 0072 00 470 75 01).

[a]ENSTA Paris, Institut Polytechnique de Paris, Palaiseau, France
[b]E-mail: alexandre@ensta.fr, ORCID: 0000-0002-6185-2480
[c]E-mail: chapoutot@ensta.fr, ORCID: 0000-0002-7230-0710
[d]ISAE-SUPAERO, Université de Toulouse, France
[e]E-mail: garion@isae-supaero.fr, ORCID: 0000-0002-4467-2939
[f]E-mail: thirioux@isae-supaero.fr, ORCID: 0009-0002-1126-6835

DOI: 10.14232/actacyb.300771

(ODE) are mostly considered to model cyber-physical systems, obtaining an analytical solution to this class of equations is a complex issue and approximations obtained with numerical methods are sometimes sufficient to check a given property. However, for some applications, such as [9, 14, 17, 23], an approximation is not enough and an enclosure of the exact solution is required.

Starting from a set of possible initial points, the solution of an ODE can be represented by a *reachable tube* describing the evolution of the system from this initial set. To ease the computation of such a tube, *abstract domains* can be used to enclose it: *boxes*, *zonotopes*, *ellipsoids*, and nonconvex sets such as *Taylor models* (see [6] for a recent review of the use of such abstract domains in set-based simulation). The more accurate an abstract domain is, *i.e.* the smallest the difference between the hull of the abstraction and the abstracted set is, the more accurate the enclosure of the reachable tube will be and therefore will be useful for verification purposes for instance.

However, only few of these set abstractions come with an associated arithmetic which is mandatory to evaluate nonlinear expressions[1]. Among such abstractions, boxes are mainly used because the underlying computation process, *i.e.*, interval arithmetic, is easy to use [19]. Zonotopes are also a good representation choice because they rely on affine arithmetic [13] which is also efficiently computable. Polytope enclosure is a promising approach as it is more precise than boxes and zonotopes, but suffers from the expensiveness of its geometrical computation as many linear programs need to be solved to obtain the set described by the sides of the polytope. Considering a polytope as *an intersection of zonotopes* and therefore benefiting from affine arithmetic is a possible solution to overcome the limitations of polytopes. To do so, two main techniques exist: i) zonotope bundles [7], *i.e.*, a set of zonotopes is used so the intersection is not computed; or ii) the intersection is computed when necessary [4].

In this paper, we propose a novel approach based on constraint programming to compute polytopes-based reachable tubes of ODEs in two phases. First a constraint satisfaction problem (mixing disjunctions and conjunctions) is generated from the tubes computed with zonotopes-based simulation. This CSP is thus a novel way to describe a reachable tube (that may or not be evaluated). Our proposed CSP approach also simplifies the computation of intersection of zonotopes. Second, these CSPs can be combined to represent intersection or union of trajectories, collision with obstacles, or other kind of properties that can be modelled by constraints. Finally, the global CSP is solved using a previously designed abstract domains that take advantage of the specificities of ODEs [27], *i.e.*, their continuous aspect and the fact that their abstractions as tubes can be naturally expressed as disjunctions. This approach allows to solve nonlinear ODEs with uncertain initial conditions, and can address several types of problems such as optimal control or safety verification.

However, while several papers deal with constraint programming combined with differential equations, such as [12, 16, 18] which address parameter identification,

[1] Minkowski sum may be then used but comes with all its limitations: only for linear operations, based on convex decomposition, etc

[22] which proposes to solve more complex differential equations, or [3] which considers problems in robotics, we propose in this paper to exploit the CSP framework to obtain a more expressive representation of sets to enclose reachable tubes.

This paper is organized as follows. Section 2 presents the main mathematical tools used in the remaining of the paper. Section 3 presents how to compute intersection of zonotopes using the Constraint Programming framework. Section 4 details experiments on three small size problems and two applications. Finally, Section 5 gives some perspectives for future work.

2 Prerequisite concepts

2.1 Polytopes and zonotopes

Considering the Euclidean space $\mathbb{R}^n$, a *convex hull* of a finite set $\{v_1, \ldots, v_N\}$ of points in $\mathbb{R}^n$ is defined as $\texttt{conv}(v_1, \ldots, v_N) = \left\{x \in \mathbb{R}^n : x = \sum_{i=1}^{N} e_i v_i, e_i \geq 0, \sum_{i=1}^{N} e_i = 1\right\}$.

A *convex polytope* in $\mathbb{R}^n$ is denoted by $\mathcal{P} = \texttt{conv}(v_1, \ldots, v_N)$ where each v_i represents a vertex of the polytope. In the following, polytopes are considered as convex by default. A polytope is a bounded (convex) polyhedron $\mathcal{P} \subset \mathbb{R}^n$, which can be also defined as $\mathcal{P} = \{\mathbf{x} \in \mathbb{R}^n | H\mathbf{x} \leq \mathbf{k}\}$ where H is a matrix of size $m \times n$ and $\mathbf{k}$ is a column vector of dimension m. This latter can be rewritten as a set of constraints $\mathcal{P} = \{x \in \mathbb{R}^n : \langle l_i, x\rangle + a_i \geq 0, i = 1, \ldots, m\}$ where $\langle ., . \rangle$ is scalar product in $\mathbb{R}^n$, $l_i \in \mathbb{R}^n$ and $a_i \in \mathbb{R}$. $\mathcal{P}$ has exactly m facets which are the intersections of $\mathcal{P}$ with the hyperplanes defined in the previous definition. These facets (or similarly the vertices v_i) can be computed with a geometrical approach while the set defined by $\mathcal{P}$ can be paved using a classical branch and prune algorithm.

A *zonotope* is a centrally symmetric polytope. The main advantage of zonotopes is that its set computation (operations, such as addition or product, on variables with zonotopic domains) can be implemented using affine arithmetic [13]. A set of values in this domain is represented by an *affine form* $\hat{x}$, which is a formal expression of the form $\hat{x} = \alpha_0 + \sum_{i=1}^{n} \alpha_i \varepsilon_i$ where the coefficients α_i are real numbers called *noise symbols*, α_0 the *center* of the affine form, and the ε_i are formal variables ranging over the interval $[-1, 1]$. Affine forms encode linear dependencies among variables: if $x \in [a_1, a_2]$ and $y = 2x$, then x will be represented by the affine form $\hat{x}$ above and y will be represented as $\hat{y} = 2\alpha_0 + 2\alpha_1\varepsilon$.

2.2 Polytopes as intersection of zonotopes

A polytope can be represented exactly by the intersection of some zonotopes as proposed in [4, 24]. Once a polytope $\mathcal{P}$ has been represented exactly by the intersection of zonotopes, i.e., $\mathcal{P} = Z_1 \cap \cdots \cap Z_n$, the image of a function f on $\mathcal{P}$ can be computed as $f(\mathcal{P}) = f(Z_1 \cap \cdots \cap Z_n) \subseteq f(Z_1) \cap \cdots \cap f(Z_n)$ where $f(Z_1), \cdots, f(Z_n)$ can be computed using affine arithmetic.

A general procedure to exactly represent a polytope $\mathcal{P} \subset \mathbb{R}^n$ by the intersection of zonotopes is presented in [24]: randomly select n inequality constraints from the pool of all inequality constraints representing the polytope and then use these n inequality constraints to construct a zonotope with the minimal volume containing the polytope until all inequality constraints for the polytope have been used up.

Notice that an intersection of zonotopes is not a zonotope in general. Such an intersection can be computed, or enclosed, with the help of a box based paver or a polytope based method, but the computation cost is usually expensive and prohibitive [15].

2.3 Set-based simulation of ODEs

An *Initial Value Problem* (IVP) for ODEs is defined by[2]

$$\begin{cases} \dot{\mathbf{y}}(t) = \mathbf{f}(t, \mathbf{y}(t)), \\ t \in \mathcal{T} := [0, t_{\text{end}}] \text{ and } \mathbf{y}(0) \in \mathcal{Y}_0, \end{cases} \tag{1}$$

with a nonlinear function $\mathbf{f} : \mathbb{R} \times \mathbb{R}^n \to \mathbb{R}^n$. More precisely, IVP for ODEs are considered over a finite time horizon $[0, t_{\text{end}}]$. Note that a bounded set $\mathcal{Y}_0$ of initial values is considered in this framework. This implies that solution of Equation (1) is a set of trajectories. We assume classical hypotheses on $\mathbf{f}$ to ensure the existence and uniqueness of the solution of Equation (1). The set $\mathcal{Y}(\mathcal{T}, \mathcal{Y}_0)$ stands for $\mathcal{Y}(\mathcal{T}, \mathcal{Y}_0) = \{\mathbf{y}(t; \mathbf{y}_0) : t \in \mathcal{T}, \mathbf{y}_0 \in \mathcal{Y}_0\}$.[3] Intuitively, $\mathcal{Y}(\mathcal{T}, \mathcal{Y}_0)$ gathers all the points reached by the scalar solution $\mathbf{y}(t; \mathbf{y}_0)$ of Equation (1) starting from all scalar initial values $\mathbf{y}_0$. Note that $\mathcal{Y}(\mathcal{T}, \mathcal{Y}_0)$ is hardly computable in general. The goal of a validated or rigorous numerical integration methods is to characterize the set of functions satisfying (1). These functions are represented by the set of values they can reach with their associated time instants, *i.e.*, $\{\mathbf{y}(t; \mathbf{y}_0) : \mathbf{y}_0 \in \mathcal{Y}_0, t \in [0, t_{\text{end}}]\}$. A convenient and often used way to access these values is to use the abstract domain of intervals which uses interval analysis to compute an over approximation of this set [10, 19, 20]. In the following, we consider Lipschitz and continuous ODEs.

Different set abstractions can be considered to tackle the problem of simulation for IVP (boxes, zonotopes, ellipsoids, etc). The main challenge is to be able to compute arithmetic operations with the chosen abstraction, even for nonlinear operations. Box representation is often preferred because of its very simple arithmetic which is also available in many tools. When considering the set of initial conditions as a box $[\mathbf{y}_0] \supset \mathcal{Y}_0$, the use of the interval framework to solve Equation (1) makes possible the design of an inclusion function $[\mathbf{y}](t; [\mathbf{y}_0])$ for the computation of an over approximation of $\mathbf{y}(t; [\mathbf{y}_0])$. To build it, a sequence of time instants $t_1, \ldots, t_s$ such that $t_1 < \cdots < t_s$ and a sequence of boxes $[\mathbf{y}_1], \ldots, [\mathbf{y}_s]$ such that $[\mathbf{y}](t_{i+1}; [\mathbf{y}_i]) \subseteq [\mathbf{y}_{i+1}]$ for all $i \in [0, s-1]$ are computed using a classic iterative

[2] Notice that the $\mathbf{f}$ function generally takes a $\mathbf{q} \in \mathbb{R}^p$ argument representing parameters. We omit it in the current paper as we do not consider parameterized differential equations.

[3] $\mathbf{y}(t; \mathbf{y}_0)$ is a notation representing the function $\mathbf{y}(t)$ with fixed initial values $\mathbf{y}_0$.

two-steps methods [20] producing a *reachable tube* $\{[\tilde{\mathbf{y}}_0], \ldots, [\tilde{\mathbf{y}}_s]\}$ where each $[\tilde{\mathbf{y}}_i]$ is the state interval containing all the values reachable in the time interval $[t_i, t_{i+1}]$.

Zonotopes are another abstraction used in set based simulation. Indeed, zonotopes being the geometrical concretization of affine arithmetic, simulation computed with affine arithmetic allows to obtain a reachable tube made of zonotopes $\mathcal{T} \equiv \{([t_1], \tilde{\mathcal{Z}}_1), \ldots, ([t_{\text{end}}], \tilde{\mathcal{Z}}_{\text{end}})\}$ which is a time-sorted list of pairs (time interval $\times$ zonotope). Tools such as DynIbex [1] or CORA [5] implement this feature. The main advantages of zonotopes are the wrapping effect[4] reduction and a more precise set representation. Considering a closed set $\mathcal{S}$, its optimal box-hull $\mathcal{B}$, its optimal zonotope-hull $\mathcal{Z}$ and its optimal polytope-hull $\mathcal{P}$, the following property holds: $\mathcal{S} \subset \mathcal{P} \subset \mathcal{Z} \subset \mathcal{B}$.

2.4 From reachable tubes to constraint programming

The solution of an IVP-ODE which is given as a set of timed zonotopes in the form $\{([t_1], \tilde{\mathcal{Z}}_1), \ldots, ([t_{\text{end}}], \tilde{\mathcal{Z}}_{\text{end}})\}$ can easily be transcribed as a disjunction of constraints since each pair $([t_i], \tilde{\mathcal{Z}}_i)$ corresponds to a quantified proposition $\forall t \in [t_i]\ \mathbf{y}(t) \in \tilde{\mathcal{Z}}_i$.[5] Eventually, the abstraction of the set of trajectories can be considered as a disjunction of all these constraints since at each time $t \in [t_0, t_{\text{end}}]$, $\mathbf{y}(t)$ verifies exactly one of them[6].

3 CP for polytopic integration

Computing the intersection of zonotopic tubes is a hard task: zonotopes are not closed under intersection or union, and thus computation of the intersection of tubes of zonotopes can be very expensive using a geometrical approach. We propose in this section to tackle this problem from the constraint solving point of view, as the intersection of zonotopic tubes resulting from ODEs can be seen as a Constraint Satisfaction Problem (CSP). Because enumerating all solutions of CSPs is generally impossible when the domains of the variables are continuous, CSP solvers generally compute a set of abstract elements that *covers* the solution space.

We have proposed in [27] a *tree abstract domain* $\mathbb{T}(D)$. This domain can be viewed as a k-d tree [8] in which leaves are defined using the numerical abstract domain D used (e.g. zonotopes) and internal nodes, also called summaries, give information about their subtrees. The tree abstraction exploits the continuous aspect of ODEs solutions to propose a fast pre-computation of the intersection of zonotopes (see Figure 1). It can thus be seen as an incremental powerset that gradually increases its precision (*i.e.* decreasing over-estimation), starting from the

[4] Wrapping effect is the overestimation induced by set abstractions in some iterative computations.

[5] $t \in [t_i]$ means $\underline{[t_i]} \leqslant t \leqslant \overline{[t_i]}$ with $\underline{[t_i]}$ and $\overline{[t_i]}$ the lower and upper bounds of $[t_i]$ respectively.

[6] As two consecutive time intervals $[t_i, t_{i+1}]$ and $[t_{i+1}, t_{i+2}]$ shared one bound (a floating point value t_{i+1}), $\mathbf{y}(t_{i+1})$ is verified for two constraints. However, as $\mathbf{y}(t_{i+1})$ is unique, this degenerated case is not an issue for the following.

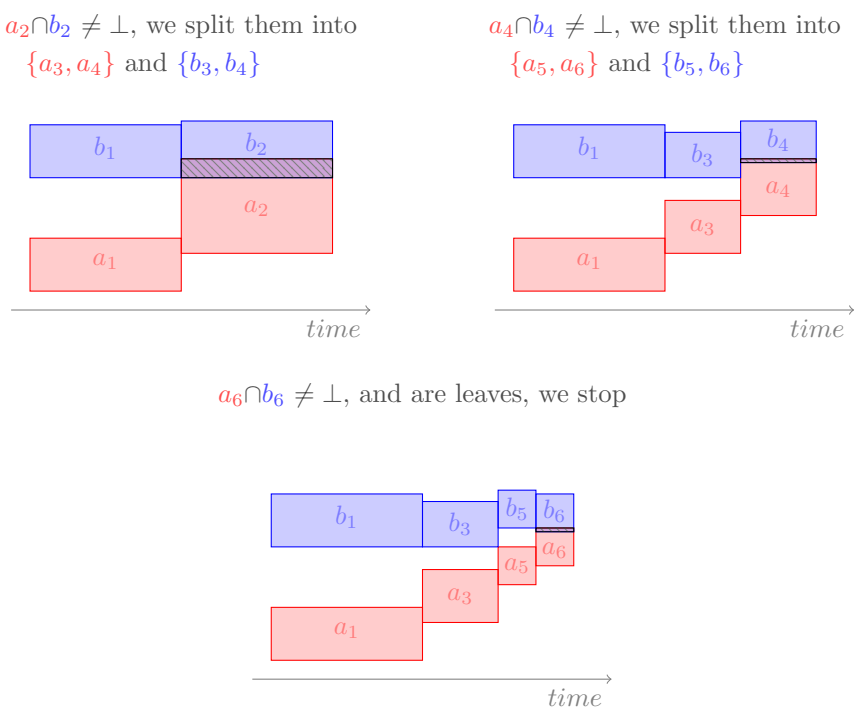


Figure 1: Recursive intersection computation (boxes for illustration).

precision of a base abstract domain D to the precision of its powerset $P(D)$ only if needed. This greatly speeds up the solving process in our experiments.

A tube $\mathcal{T}$ can be defined using a disjunction of predicates as described in Subsection 2.4. $\mathcal{T} = ([t_1] \wedge e_1) \vee ([t_2] \wedge e_2) \cdots \vee ([t_n] \wedge e_n)$ where each e_i represents the set of values of solution functions within time frame $[t_i]$, can be read as: *the solution is either in set* e_1 *during the time frame* $[t_1]$*, or in set* e_2 *during the time frame* $[t_2]$*, etc.* This property is always true, as exactly one of its atoms will be true at a given time (cf. Footnote 6 on Page 759) and defines the reachable tube.

Considering initial values given as a polytope $\mathcal{P}$, $\mathcal{P}$ is decomposed as an intersection of s zonotopes $\mathcal{Z}_i$ as described in Subsection 2.2. The reachable tube of the ODE is therefore described by the conjunction of s tubes $\mathcal{T}^1 \wedge \cdots \wedge \mathcal{T}^i \wedge \cdots \wedge \mathcal{T}^s$, each tube $\mathcal{T}^i$ being obtained by the zonotopic simulation of the ODE with initial value taken as a zonotope $\mathcal{Z}_i$ (cf. Section 2). As the initial polytope is not empty, the s tubes obtained with validated simulation have a non empty intersection due to the Lipschitz and continuous properties of ODEs. This intersection is a correct enclosure of the theoretical tube computed with initial value as polytope $\mathcal{P}$.

Note that there is no synchronization between the tubes, *i.e.* the time frames t^i_* of tube $\mathcal{T}^i$ may be different than the time frames t^j_* of tube $\mathcal{T}^j$, but each time

frame $[t_*^i]$ of $\mathcal{T}^i$ intersects with at least one time frame $[t_*^j]$ of $\mathcal{T}^j$.

Figure 2 shows that the intersection of two zonotope-based tubes $\mathcal{T}^1$ (in blue) and $\mathcal{T}^2$ (in red) produces the polytope-based tube shown in green. Using our proposed tree abstraction we are able to efficiently compute the polytope-based tube from the two zonotope-based tubes without having to explicitly define the constraints corresponding to such an intersection.

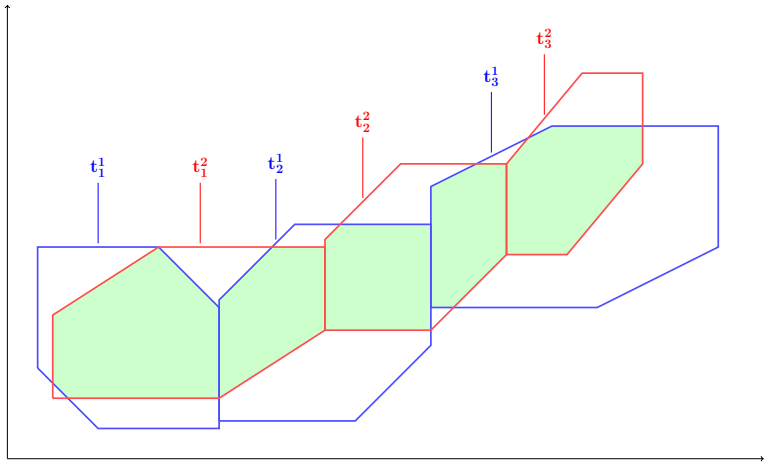


Figure 2: Polytopes as intersections of zonotopes with their own time frames.

Such a CSP C, declared as the intersection of the CSPs describing $\mathcal{T}^1$ and $\mathcal{T}^2$, can be used to:

- pave the reachable tube (with a branch and prune algorithm for example). Notice that such paving may be computationally demanding.
- detect collision with obstacles or other tubes representations by adding constraints. Let us suppose that that we want to verify that the previous tube does not intersect a given obstacle, defined by a polytope $\mathcal{P}_o$. We build the CSP $C \wedge (t \in [t_0, t_{end}] \wedge \mathcal{P}_o)$ and ask our solver to solve the CSP: if there is no solution, then the tube does not intersect the obstacle, otherwise the solver should give us a collision example.

4 Experiments

The following experiments have been conducted using the open-source library DynIbex [1] to compute the zonotope-based reachable tubes and the open-source constraint solver AbSolute [21] equipped with the tree domain presented in Section 3 to check if constraints over the computed tubes hold (no intersection for instance). AbSolute is also used to pave intersections of zonotopes to visualize the corresponding polytope.

4.1 First experiments

We first choose the two classic problems presented in [4] in order to validate our approach. In a third experiment we also compare our polytope based technique with a pointwise Monte-Carlo approach. We analyze these three experiments in Paragraph 4.1.4.

4.1.1 The circle problem

The circle problem is defined by the following equation:

$$\begin{cases} \dot{y}_1 &= -y_2 \\ \dot{y}_2 &= y_1 \end{cases} \tag{2}$$

The initial condition is taken in a polytope given by the five vertices $(-1,-3)$, $(-1.5,3)$, $(0,6)$, $(1.5,4)$, and $(1,-4)$ at $t_0 = 0$. This polytope can be defined as the intersection of three zonotopes. The simulation with these three zonotopes as initial conditions is performed with Kutta's third order validated method (also called RK32 in [11]), with absolute error tolerance $10^{-6}$and till $t = 20$ seconds.

Figures 3a shows the last zonotopes of the three reachable tubes. The polytope defined as the intersection of these three zonotopes is guaranteed to contain the reachable state at the end of simulation horizon ($t = 20$). To compute this polytope, AbSolute is used with a reduced domain for time: we consider $t \in [19.99, 20.00]$. Figure 3b presents the results using the box abstract domain to pave the solution while Figure 3c presents the results using the polytope abstract domain as base domain to pave the solution. The full reachable tube can also be computed with AbSolute (with time domain $t \in [0, 20]$) and is depicted in Figure 4a and Figure 4b.

4.1.2 The Lotka-Volterra problem

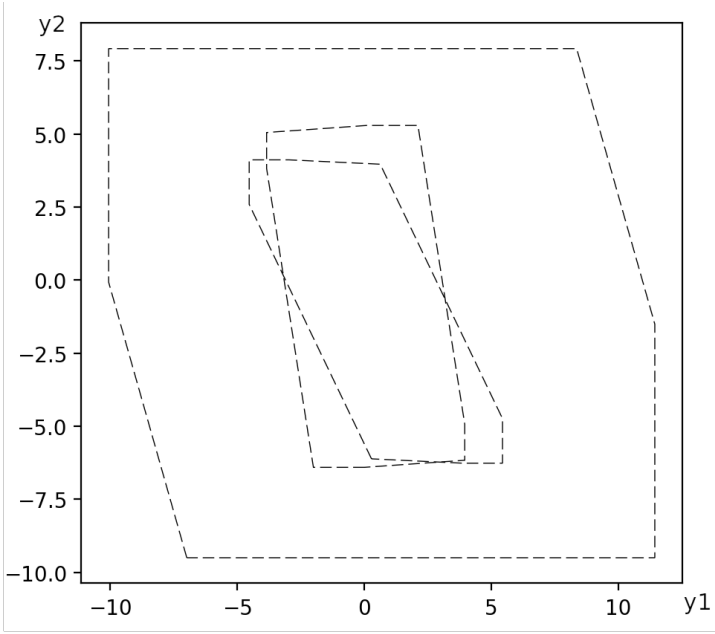
The Lotka-Volterra problem is defined by the following equation:

$$\begin{cases} \dot{y}_1 &= 2y_1(1-y_2) \\ \dot{y}_2 &= -y_2(1-y_1) \end{cases} \tag{3}$$

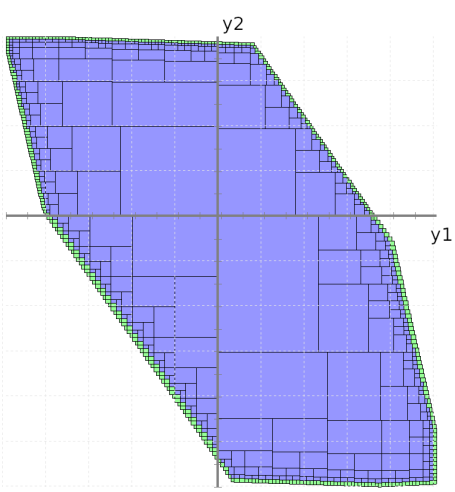
The initial condition is taken in a polytope given by the eight vertices $(1.1035, 3.0457)$, $(1.1041, 3.0386)$, $(1.0981, 3.0366)$, $(1.1039, 3.0358)$, $(1.0983, 3.0339)$, $(1.1020, 3.0320)$, $(1.0989, 3.0498)$ and $(1.0995, 3.0510)$. This polytope is covered by three zonotopes. Simulations with these three zonotopes as initial conditions are performed with Kutta's third order validated method with absolute error tolerance 10^{-6} and till $t = 6$ seconds. The reachable tube is computed as the intersection of the three zonotope-based reachable tubes. The result is shown on Figure 5a and Figure 5b and corresponds to the classic solution of the problem.

4.1.3 Comparison on the Van der Pol problem

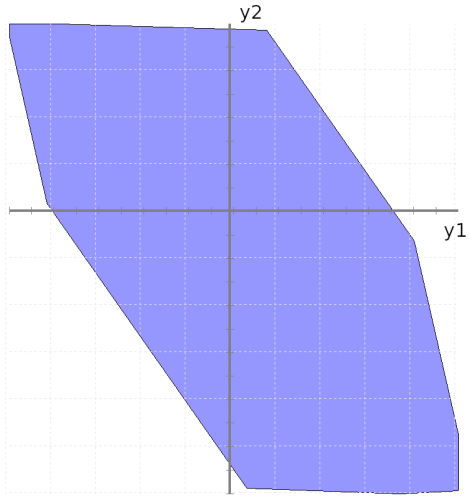
The last problem aims to compare the results obtained with our polytope based technique and a pointwise integration obtained with a Monte-Carlo approach (154



(a) Zonotopes with DynIbex



(b) Polytopes with AbSolute and box domain



(c) Polytopes with AbSolute and polytope domain

Figure 3: Last solution of the tube for Example 4.1.1.

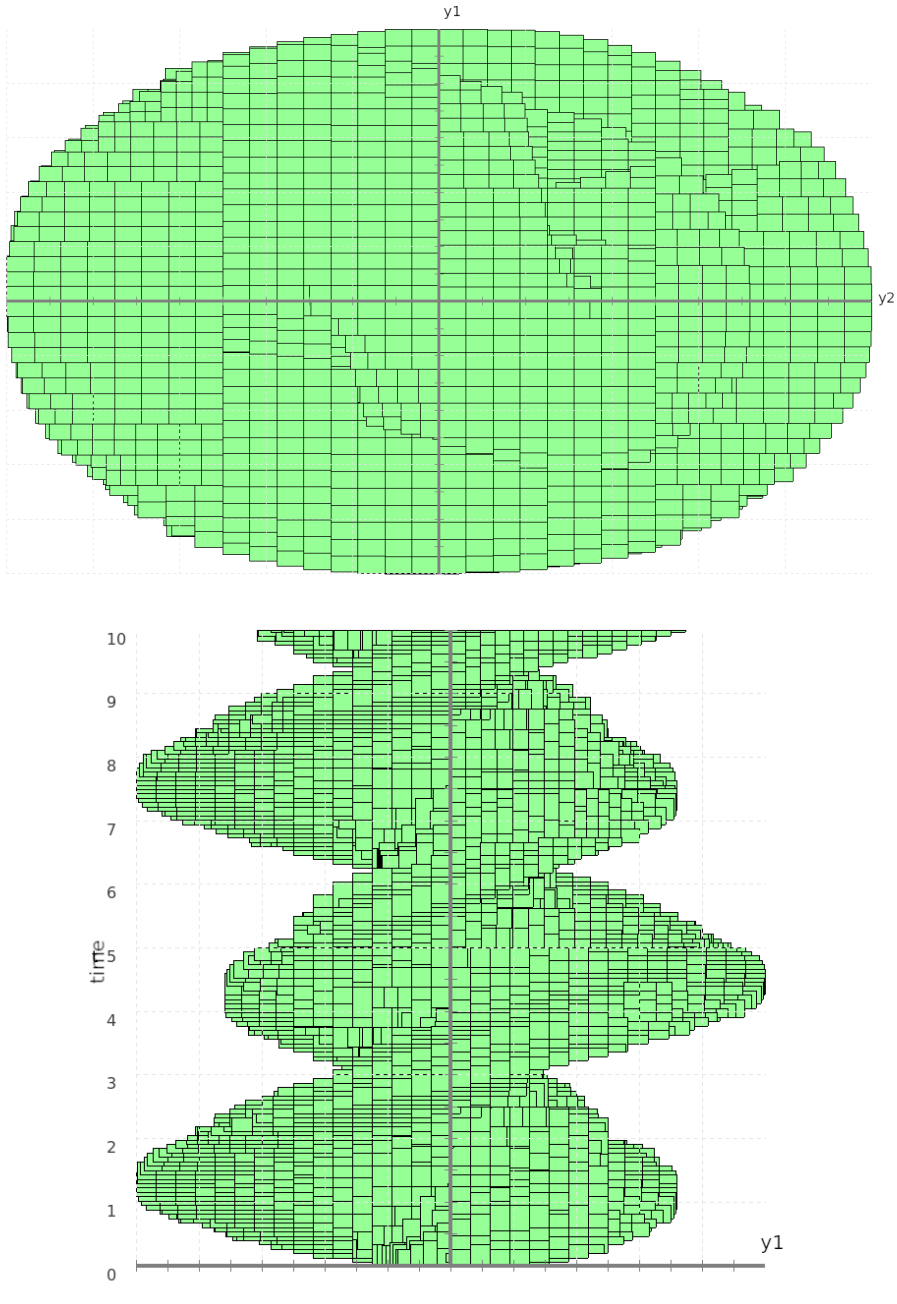
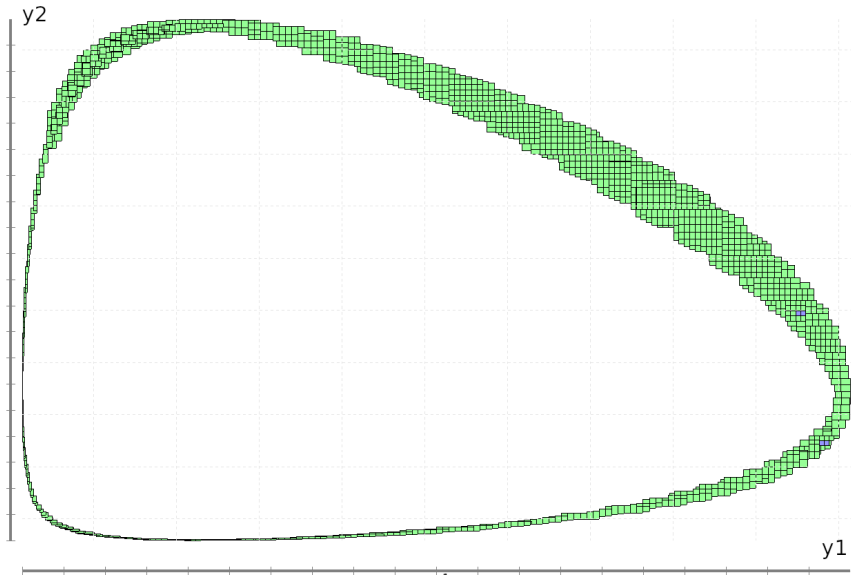
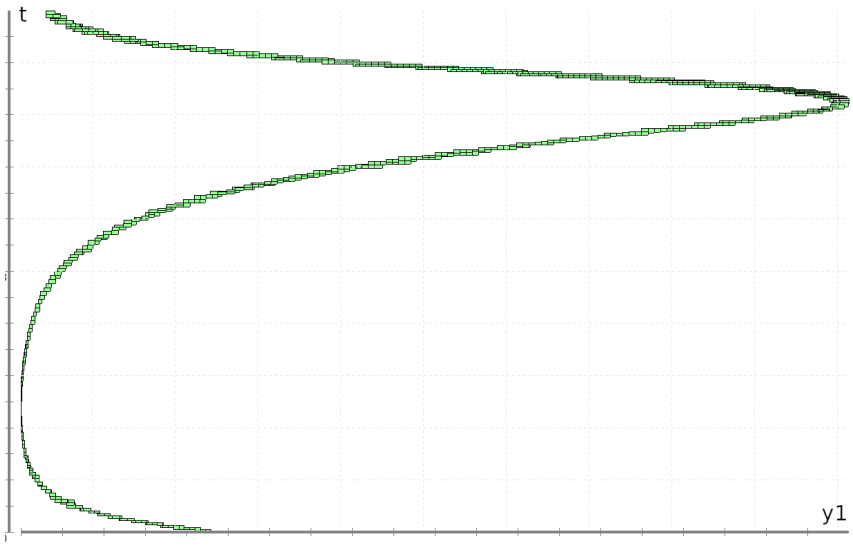


Figure 4: Reachable tube obtained with box domain for Example 4.1.1.



(a)



(b)

Figure 5: Reachable tube obtained with box domain for Example 4.1.2.

points from a regular mesh of the initial polytope, depicted in blue in the following figures). The chosen problem is the Van der Pol oscillator, known to be interesting to challenge tools, as it has a limit cycle which attracts pointwise trajectories but can stress set-based simulators because of system rotation. It is defined by:

$$\begin{cases} \dot{y}_1 &= & y_2 \\ \dot{y}_2 &= & y_2(1-y_1^2)-y_1 \end{cases}. \tag{4}$$

Initial conditions are defined as the intersection of three zonotopes and are plotted in Figure 6 and the whole tube computed with our technique is shown in Figure 7.

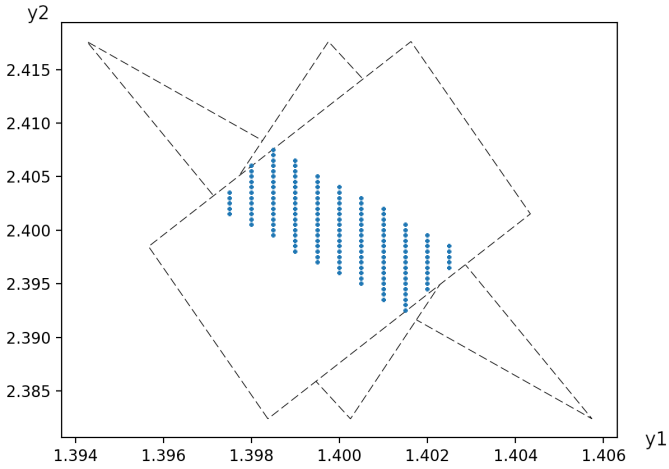


Figure 6: Initial condition, given as the intersection of three zonotopes, for polytope based simulation and initial points for pointwise simulation.

Figures 8a, 8b and 8c present respectively at time $t = 1$, $t = 3$ and $t = 7$ the polytope computed by AbSolute and points obtained with pointwise initial conditions and validated simulation by DynIbex. All simulations (zonotopes-based and pointwise) are performed with Runge-Kutta four order validated method with absolute error tolerance 10^{-8} and till $t = 7$ seconds (time for a revolution).

4.1.4 Appraisal

The circle problem presented in Subsection 4.1.1 shows the results of the polytope computation at some instants and on a whole tube. At a chosen instant, the obtained polytope is similar to the one obtained in [4]. The full tube shows a good stability as the circle is well depicted with a restrained overestimation (property of Hamiltonian conservation is preserved). To verify this conservation, we have

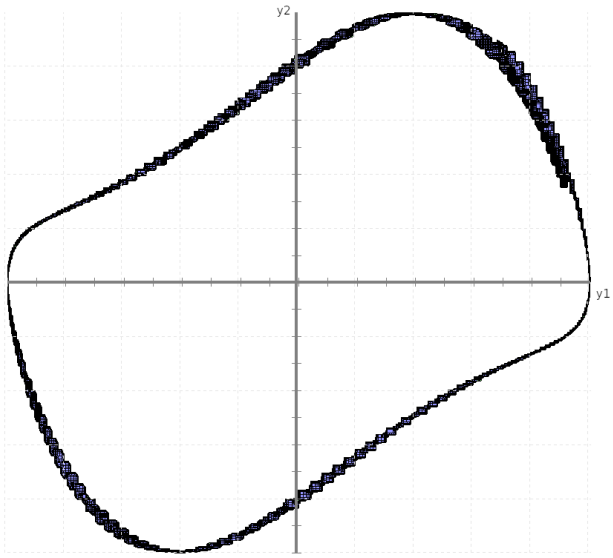


Figure 7: Van der Pol oscillator limit cycle computed with initial condition as a polytope.

computed the maximal distance to the frame center such as max $\sqrt{y_1(t)^2 + y_2(t)^2}$ at different instants ($t = 0, 5, 10, 15, 20$) and we obtained $6.0, 7.58, 6.40, 8.16, 7.63$. The values oscillate as selected times do not follow the system period, but do not show any divergence.

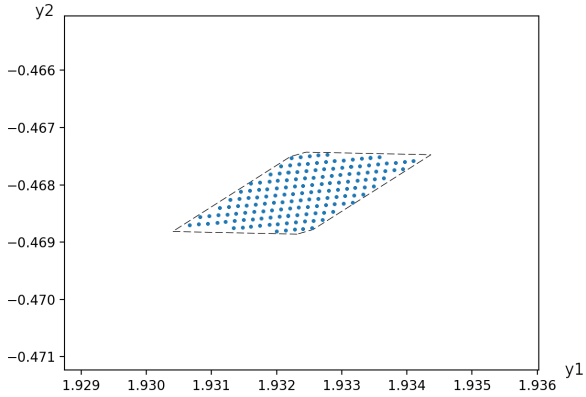
The Lotka-Volterra problem confirms that the contracting instant (bottom left on Figure 5a) is preserved, and that the dissipative instant (top right) has restrained effect on the size of polytopes. The Van der Pol oscillator problem shows that wrapping effect has small effect and that polytopes are able to preserve, at least better than boxes, a limit cycle.

4.2 Application on collision detection and rendez-vous

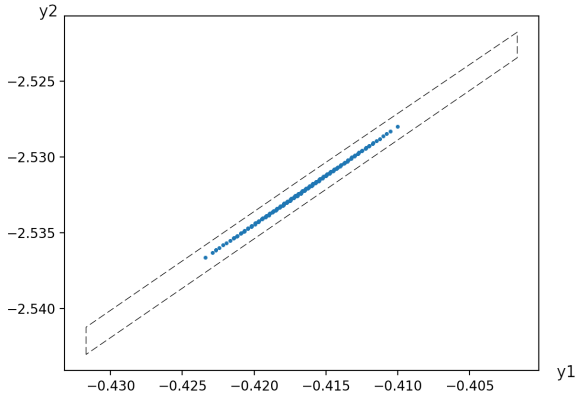
4.2.1 Collision detection

In our previous work [2], we have applied the CP approach with interval domains to check a collision-free property on a state of the art distributed multi-agent formation control protocol [26], briefly recalled in the following.

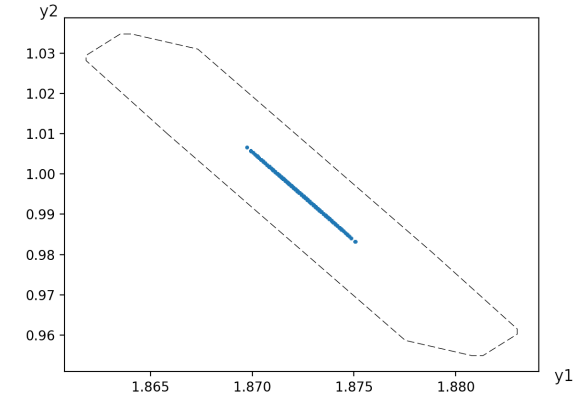
Consider n agents in $\mathbb{R}^d$ ($n \geqslant 2$ and $d \geqslant 2$). Their position at time t is denoted by $p_i(t) \in \mathbb{R}^d$ with $i \in \{1, \ldots, n\}$. Interactions between agents are described by a graph $\mathcal{G}$ with a vertex set $\mathcal{V} = \{1, \ldots, n\}$ and an edge set $\mathcal{E}$. An edge $(i, j) \in \mathcal{E}$ means that agent i can measure the relative bearing of agent j so is a neighbor of Agent j. The set of all neighbors of agent i is denoted by $\mathcal{N}_i = \{j \in \mathcal{V} : (i, j) \in \mathcal{E}\}$. Edges are not directed: if i is a neighbor of j, then j is also a neighbor of i. A



(a) $t = 1$



(b) $t = 3$



(c) $t = 7$

Figure 8: Comparison between the polytope obtained with AbSolute and Monte-Carlo approach.

formation denoted by $\mathcal{G}(p)$ is a graph $\mathcal{G}$ such that each vertex i of $\mathcal{G}$ is associated to $p_i(t)$. The relative bearing of p_j with respect to p_i is defined by $e_{ij} := p_j - p_i$, $g_{i,j} := \frac{e_{ij}}{\|e_{ij}\|}$, with $\| \, . \, \|$ being the Euclidean norm.

Suppose the velocity of n_ℓ agents is given. Such agents are called *leaders* and the other n_f agents are called *followers*. The dynamics of leaders and followers follows a single integrator model that can be used to follow a predefined plan made of different way-points. When dealing with a formation of n agents, we have to define n trajectories $r_1, \ldots r_n$ and the constraint corresponding to the possible collision between agents is expressed by $(r_1 \wedge r_2) \vee \ldots (r_1 \wedge r_n) \vee (r_2 \wedge r_3) \vee \cdots \vee (r_{n-1} \wedge r_n)$: there is a collision if two constraints representing two trajectories are true at the same time and therefore the global formula is true.

We perform a new verification with our polytopic approach. As in [2], we only consider a set of four agents using displacements respecting a square-based formation: translation, scaling and rotation. Composition of these three displacements is performed to generate trajectories for the agents following algorithm presented in [26]. A finite number of values for each displacement is considered: north, south, east, west direction for translation; contraction and dilation for scaling; rotations with angles $\pi/4$, $\pi/2$, $3\pi/4$ and π over the centroid. 108 atomic displacements are thus considered. Two different sources of uncertainties are taken into account: uncertainties on initial positions of agents and on inter-agent distances, allowing four different scenarios.We aim to detect which combination of displacements may generate a collision.

A summary of the verification of collision-free property on atomic displacements is given in Table 1: `NO` stands for no uncertainty, `EI` stands for uncertainty on initial conditions, `ED` stands for uncertainty on distance measure and `EID` stands for uncertainties on both initial position and distance measures. Two values are considered for uncertainties, namely 0.01 and 0.1. The number of satisfiable, unsatisfiable and

Table 1: Collision-free checking on atomic displacements for robot formation.

	T1T2	T1T3	T1T4	T2T3	T2T4	T3T4
`NO`	27/81/0	1/107/0	4/104/0	3/105/0	3/105/0	4/104/0
	0/107/1	0/108/0	0/107/1	0/106/2	0/108/0	0/106/2
`EI` 0.01	27/81/0	1/107/0	9/99/0	9/99/0	3/105/0	17/91/0
	0/107/1	0/108/0	1/105/2	0/108/0	0/108/0	0/105/3
`EI` 0.1	27/81/0	12/96/0	41/67/0	52/56/0	26/82/0	54/54/0
	0/105/3	0/108/3	1/101/6	0/108/0	0/108/0	0/97/11
`ED` 0.01	27/81/0	3/105/0	16/92/0	19/89/0	11/97/0	52/56/0
	0/108/0	0/108/0	0/107/1	0/108/0	0/108/0	0/107/1
`ED` 0.1	27/81/0	31/77/0	80/28/0	48/60/0	50/58/0	65/43/0
	0/107/1	0/108/0	0/107/1	0/108/0	0/108/0	0/106/2
`EID` 0.01	27/81/0	4/104/0	22/86/0	28/80/0	11/97/0	54/54/0
	0/105/3	0/108/0	1/105/2	0/108/0	0/108/0	0/107/1
`EID` 0.1	27/81/0	38/70/0	92/16/0	70/38/0	56/52/0	69/39/0
	0/106/2	0/108/0	1/106/1	0/108/0	0/108/0	0/107/1

inconclusive problems (read SAT/UNSAT/MAYBE) is reported for each scenario and each pair of trajectory between agents (T1T2 stands for trajectory of agent 1 and agent 2). An UNSAT scenario means that no collision have been found and so the atomic displacement is safe while a SAT scenario implies that a potential collision has been found. No conclusion can be done with a MAYBE result.

Looking at this experimental evaluation of the bearing-based formation control, we note that increasing uncertainties increases the number of possible collisions for interval domain but the polytopic approach seems more robust with respect to uncertainties. Moreover, polytopic approach provides sharper trajectory tubes and so reduce the number of possible collisions.

4.2.2 Location of a rendez-vous area

As a second application, we consider the problem of the location of a meeting point or a rendez-vous area. In a mathematical point of view, it is equivalent to the detection of a global attractor, which is a difficult problem. This application could be used to simulate a crowd behavior [25] or study risk of contamination during an epidemic (as where people meet is an important question in epidemiology).

We build a domain with a limit cycle (from Van Der Pol oscillator), a repulsor at position $(1, 1)$ and an attractor at position $(-1, -1)$. The resulting ODE is:

$$\begin{cases} \dot{y}_1 &= \alpha y_2 - \beta(y_1 + 1) + \gamma(y_1 - 1) \\ \dot{y}_2 &= \alpha(y_2(1 - y_1^2) - y_1) - \beta(y_2 + 1) + \gamma(y_2 - 1) \end{cases} \tag{5}$$

with $\alpha = 1$, $\beta = 0.8$ and $\gamma = 0.3$. We randomly drop 10 particles in the corresponding vector field and we detect if a rendez-vous occurs (on a 30 seconds horizon), *i.e.*, if two particles are at the same place at the same time. This is the opposite of the previous problem of collision detection, but the technique is the same.

A zonotopic tube is computed for each particle and two CSP are solved: 1) a disjunction of tubes to depict the trajectories, and 2) a conjunction of tubes to locate a possible rendez-vous area. Results are shown in Figure 9.

An inner region is tagged as a rendez-vous for all the particles in the interval $[-1.69944786755, -1.69888555045]$; $[0.252656244732, 0.253503277614]$ during instant $[11.152624717, 11.1533191391]$, while, between $t = 10.35$ and $t = 30$, a meeting is possible between at least two particles, without certitude, around $(-1.70; 0.25)$ (inside the penumbra). This place seems to be an attractor.

5 Discussion and conclusion

In this paper, we presented a method based on Constraint Programming to compute reachable tubes of Ordinary Differential Equations in the particular case where the initial domain is given as a polytope. This polytopic approach is of interest in the field of cyber-physical system verification, because polytopes offer in general a sharper enclosure than classical boxes. Moreover, initial domains are often defined in engineering processes with constraints producing in general a polytope. The

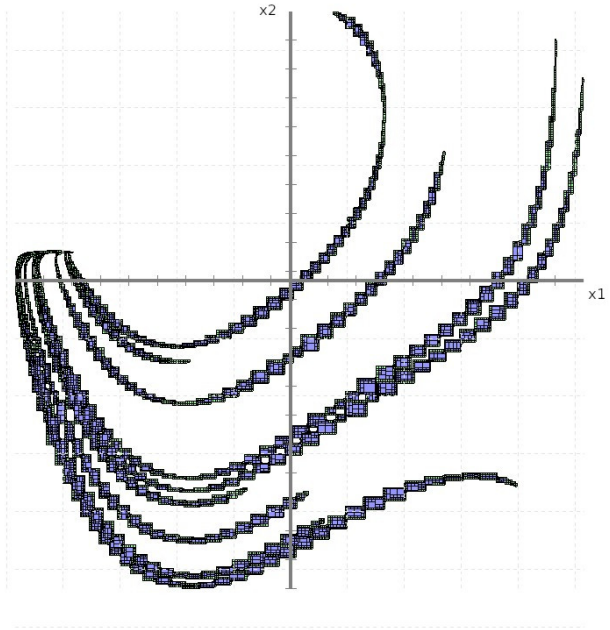


Figure 9: The result for the disjunction of the trajectories of the ten particles.

proposed approach is therefore related to engineering requirements. Our method has been experimented using two open-source tools known to be efficient in their respective communities: DynIbex and AbSolute. The results obtained are good and promising for many applications such as control synthesis (for example with respect to some safety constraints even with uncertainties in sensors), motion planning (for example with respect to uncertainties and obstacle avoidance), etc.

Concerning computation time, the circle problem is solved on a basic laptop with few seconds for the three zonotopic simulations, between few seconds and a minute for the CSP generation (depending on required precision) and few seconds for the CSP solving, which is globally reasonable. CSP generation may be included to the simulation step to optimize execution time.

As future work, even if the method does not present any limitation in term of problems that can be handled, we plan to test and compare it on more complex and higher dimensional problems. After this first step, applications such as properties verification for cyber-physical systems, invariant computation for ODEs or hybrid system simulation will be considered.

References

[1] Alexandre dit Sandretto, J. and Chapoutot, A. Validated explicit and implicit Runge–Kutta methods. *Reliable Computing*, 22(1):79–103, 2016. URL: https:

//hal.science/hal-01243053.

[2] Alexandre dit Sandretto, J., Chapoutot, A., Garion, C., Thirioux, X., and Ziat, G. Constraint-based verification of formation control. In *Proceedings of the 60th IEEE Conference on Decision and Control*, pages 7136–7141. IEEE, 2021. DOI: 10.1109/CDC45484.2021.9683622.

[3] Alexandre dit Sandretto, J., Chapoutot, A., and Mullier, O. Constraint-based framework for reasoning with differential equations. In Koç, Ç. K., editor, *Cyber-Physical Systems Security*, pages 23–41. Springer International Publishing, 2018. DOI: 10.1007/978-3-319-98935-8_2.

[4] Alexandre dit Sandretto, J. and Wan, J. Reachability analysis of nonlinear ODEs using polytopic based validated Runge-Kutta. In *Proceedings of Reachability Problems*, Volume 11123 of *Lecture Notes in Computer Science*, pages 1–14. Springer, 2018. DOI: 10.1007/978-3-030-00250-3_1.

[5] Althoff, M. An introduction to CORA 2015. In *Proceedings of the Workshop on Applied Verification for Continuous and Hybrid Systems*, pages 120–151, 2015. DOI: 10.29007/zbkv.

[6] Althoff, M., Frehse, G., and Girard, A. Set propagation techniques for reachability analysis. *Annual Review of Control, Robotics, and Autonomous Systems*, 4(1), 2021. DOI: 10.1146/annurev-control-071420-081941.

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

[8] Bentley, J. L. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517, 1975. DOI: 10.1145/361002.361007.

[9] Bouissou, O., Goubault, E., Putot, S., Tekkal, K., and Vedrine, F. HybridFluctuat: A static analyzer of numerical programs within a continuous environment. In *Proceedings of Computer Aided Verification*, Volume 5643 of *Lecture Notes in computer Science*, pages 620–626. Springer, 2009. DOI: 10.1007/978-3-642-02658-4_46.

[10] Bouissou, O. and Martel, M. Abstract interpretation of the physical inputs of embedded programs. In *Proceedings of Verification, Model Checking, and Abstract Interpretation*, Volume 4905 of *Lecture Notes in Computer Science*, pages 37–51. Springer, 2008. DOI: 10.5555/1787526.1787534.

[11] Butcher, J. C. *Numerical methods for ordinary differential equations*. John Wiley & Sons, 2008. DOI: 10.1002/9781119121534.

[12] Cruz, J. and Barahona, P. Constraint reasoning over differential equations. *Applied Numerical Analysis and Computational Mathematics*, 1(1):140–154, 2004. DOI: 10.1002/anac.200310012.

[13] de Figueiredo, L. H. and Stolfi, J. *Self-Validated Numerical Methods and Applications*. Brazilian Math. Colloquium monographs. IMPA/CNPq, 1997. URL: https://www.iri.upc.edu/people/thomas/Collection/details/45477.html.

[14] Eggers, A., Ramdani, N., Nedialkov, N., and Fränzle, M. Improving SAT modulo ODE for hybrid systems analysis by combining different enclosure methods. In *Proceedings of Software Engineering and Formal Methods*, Volume 7041 of *Lecture Notes in Computer Science*, pages 172–187. Springer, 2011. DOI: 10.1007/s10270-012-0295-3.

[15] Ghorbal, K., Goubault, E., and Putot, S. A logical product approach to zonotope intersection. In Touili, T., Cook, B., and Jackson, P. B., editors, *Proceedings of the 22th International Conference on Computer Aided Verification*, Volume 6174 of *Lecture Notes in Computer Science*, pages 212–226. Springer, 2010. DOI: 10.1007/978-3-642-14295-6_22.

[16] Goldsztejn, A., Mullier, O., Eveillard, D., and Hosobe, H. Including ordinary differential equations based constraints in the standard CP framework. In *Proceedings of Principles and Practice of Constraint Programming*, Volume 6308 of *Lecture Notes in Computer Science*, pages 221–235. Springer, 2010. DOI: 10.5555/1886008.1886031.

[17] Henzinger, T. A., Horowitz, B., Majumdar, R., and Wong-Toi, H. Beyond HYTECH: Hybrid systems analysis using interval numerical methods. In *Proceedings of Hybrid Systems: Computation and Control*, Volume 1790 of *Lecture Notes in Computer Science*, pages 130–144. Springer, 2000. DOI: 10.1007/3-540-46430-1_14.

[18] Janssen, M., Deville, Y., and Van Hentenryck, P. Multistep filtering operators for ordinary differential equations. In *Proceedings of Principles and Practice of Constraint Programming*, Volume 1713 of *Lecture Notes in Computer Science*, pages 246–260. Springer, 1999. DOI: 10.1007/978-3-540-48085-3_18.

[19] Moore, R. E. *Interval Analysis*. Prentice Hall, 1966. ISBN: 2577-9435.

[20] 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.

[21] Pelleau, M., Miné, A., Truchet, C., and Benhamou, F. A constraint solver based on abstract domains. In *Proceedings of Verification, Model Checking, and Abstract Interpretation*, Volume 7737 of *Lecture Notes in Computer Science*, pages 434–454. Springer, 2013. DOI: 10.1007/978-3-642-35873-9_26.

[22] Rohou, S., Bedouhene, A., Chabert, G., Goldsztejn, A., Jaulin, L., Neveu, B., Reyes, V., and Tromettoni, G. Towards a generic interval solver for differential-algebraic CSP. In *Proceedings of Principles and Practice of Constraint Programming*, Volume 12333 of *Lecture Notes in Computer Science*, pages 548–565. Springer, 2020. DOI: 10.1007/978-3-030-58475-7_32.

[23] Tucker, W. A rigorous ODE solver and Smale's 14th problem. *Foundations of Computational Mathematics*, 2(1):53–117, 2002. DOI: 10.1007/s002080010018.

[24] Wan, J., Sharma, S. K., and Sutton, R. Guaranteed state estimation for nonlinear discrete-time systems via indirectly implemented polytopic set computation. *IEEE Transactions on Automatic Control*, 63(12):4317–4322, 2018. DOI: 10.1109/TAC.2018.2816262.

[25] Wong, K. Y., Thyvetil, M.-A., Machaira, A., and Loscos, C. System for simulating dynamic features of crowd behavior. In *ACM SIGGRAPH 2005 Posters*, SIGGRAPH '05. ACM, 2005. DOI: 10.1145/1186954.1187085.

[26] Zhao, S. and Zelazo, D. Bearing-based formation stabilization with directed interaction topologies. In *Proceedings of the 54th IEEE Conference on Decision and Control*, pages 6115–6120, 2015. DOI: 10.1109/CDC.2015.7403181.

[27] Ziat, G., Mullier, O., Alexandre dit Sandretto, J., Garion, C., Chapoutot, A., and Thirioux, X. Abstract domains for constraint programming with differential equations. In *Proceedings of the 9th International Workshop on Numerical and Symbolic Abstract Domains*, pages 2–11, 2020. DOI: 10.1145/3427762.3429453.

Acta Cybernetica **26** (2024) 775–798.

Verified Bit and Power Allocation for MIMO Systems: A Comparison of SVD Based Techniques With GMD

Ekaterina Auer[ab] and Andreas Ahrens[ac]

Abstract

Methods with result verification have been applied in different practical contexts, for example, in such diverse areas as robotics, computer graphics, or chemistry. Such methods help to verify the result of a computer simulation, additionally taking into account possibly present bounded uncertainty in a deterministic way. Modeling and simulation of multiple-input multiple-output (MIMO) systems has not received much attention from this angle. Nowadays, increasing the capacity of communication links by using the MIMO mechanism is an essential part of various wireless communication standards.

In this paper, we consider the channel separation stage in the overall modeling and simulation process for MIMO systems and compare the singular value decomposition (SVD) and the geometric mean decomposition (GMD) based approaches from the point of view of the achievable bit error ratio (BER) under good and poor scattering conditions. As a special focus, we use interval methods to verify the result and to deal with the appearing epistemic uncertainty. Additionally, we consider resource allocation in detail, which mostly makes sense only for the SVD approach since the goal of the GMD based one is to avoid it. However, this has been studied only asymptotically until now and needs confirmation. We propose a combined analytical-numerical approach to simulate resource allocation relying on verified techniques. The theoretical results are illustrated and the comparison is performed using simulated data for an uncorrelated and a correlated MIMO system with four receiving and four transmitting antennas.

Keywords: MIMO, methods with result verification, SVD, GMD, resource allocation, Lagrange multipliers

[a] University of Applied Sciences Wismar, Germany
[b] E-mail: ekaterina.auer@hs-wismar.de, ORCID: 0000-0003-4059-3982
[c] E-mail: andreas.ahrens@hs-wismar.de, ORCID: 0000-0002-7664-9450

DOI: 10.14232/actacyb.301323

1 Introduction

In the last decades, the multiple-input multiple-output (MIMO) method for transmitting information has become an essential mechanism in wireless communications. This strategy of placing multiple antennas both at the transmitter and receiver sides can be shown to improve the capacity and the integrity of wireless systems [4, 7, 21]. In this paper, we work with simulated systems relying on a simple linear stochastic model for a frequency flat[1] MIMO link consisting of n_{T} transmitting and n_{R} receiving antennas given as

$$\vec{y} = H \cdot \vec{a} + \vec{n}, \quad \vec{y}, \vec{n} \in \mathbb{C}^{n_{\mathrm{R}}},\ \vec{a} \in \mathbb{C}^{n_{\mathrm{T}}},\ H \in \mathbb{C}^{n_{\mathrm{R}} \times n_{\mathrm{T}}} \ . \tag{1}$$

Here, $\vec{y}$ is the received data vector, $\vec{a}$ is the transmitted signal vector, $\vec{n}$ is the vector of the additive white Gaussian noise at the receiver side with the zero mean and the standard deviation σ in both real and imaginary parts. The investigated (frequency flat) channel profile generates interferences between the different antenna's data streams but no intersymbol interference is present at the receiver input. We assume that the standard deviation of the noise is computed as

$$\sigma = \sqrt{\frac{P_{\mathrm{s}}}{2 \cdot 10^{\frac{E_s}{N_0}/10}}} \ .$$

In this formula, P_{s} is the available transmit power and $\frac{E_s}{N_0}$ in dB is the signal-to-noise ratio (SNR), where E_s denotes the symbol energy and N_0 the noise power spectral density. Additionally, we assume that the throughput of the MIMO system is fixed at a certain desired value.

The channel matrix H from Eq. (1) describes each individual path from every transmitting antenna to every receiving antenna. In order to simulate the actual paths, the Rayleigh distribution is used in wireless communications. That is, the coefficients of the $(n_{\mathrm{R}} \times n_{\mathrm{T}})$ matrix H are simulated as independently and identically distributed Rayleigh fading channels [20] with the equal standard deviation δ.

In Figure 1, the general modeling and simulation process for a MIMO system is shown. The first step is to define the structure of the MIMO system. The spatial placement of the antennas is responsible for scattering conditions being good or poor, or, in other words, for creating an uncorrelated (good conditions) or correlated system. In the next step, the channel matrix of the link needs to be identified, which can be done, for example, via least squares optimization using pilot sequences [23]. In this paper, we rely on simulated matrices obtained by the Rayleigh distribution as described, for example, in [1]. The last two steps in the process are in the focus of the present study: interference suppression (channel separation) and resource allocation with the goal of optimizing the quality criterion of the bit error ratio (BER). Additionally, we consider the influence of the scattering conditions.

[1] i.e., a single filter channel tap is enough to represent it; the channel can be described by a single matrix H [22]

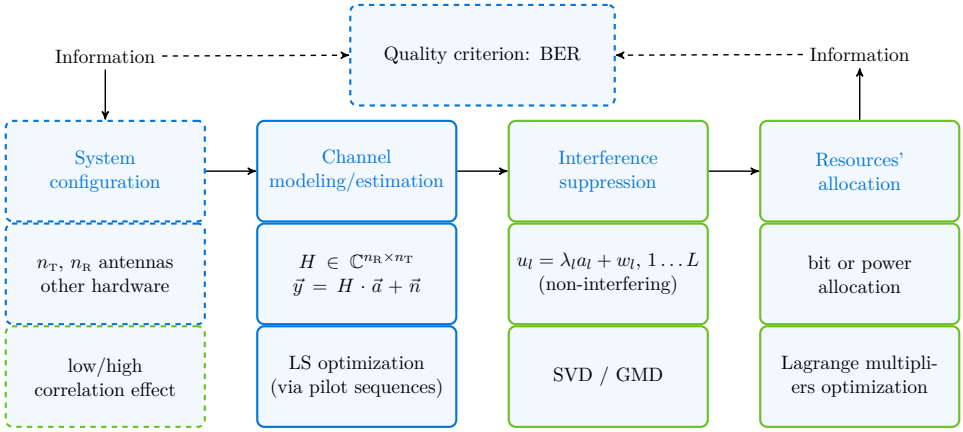


Figure 1: General stages in the MIMO modeling and simulation process. The stages this paper focuses on are shown in green.

The intention behind the interference suppression is to create $n = \min\{n_R, n_R\}$ separate single-input single-output (SISO) communication channels (ideally, without interference) corresponding to the original MIMO channel H using the appropriate precoding and postcoding techniques [9, 18]. The usual approach here is to employ the singular value decomposition (SVD) of the matrix H creating SISO channels of unequal weights, which, therefore, have different performance wrt. the BER. However, it is also possible to use the geometric mean decomposition (GMD) technique producing n *identical* separate channels at the cost of remaining interferences which can be, however, removed from the system by using interference cancellation schemes [10].

Whereas it is advantageous to optimize allocation of resources in the last step of the process in case of the SVD [18], the GMD based approach is reported to be optimal for high SNR, that is, low σ, meaning that resources such as bits per symbol and power can be distributed uniformly among the (active) SISO channels [10]. To our best knowledge, there is no systematic comparison between the SVD and GMD based approaches wrt. the best achievable BER, which additionally considers the difference between correlated and uncorrelated systems. Our intention in this paper is to close the gap with a further focus on verifying the obtained results and quantifying bounded uncertainty via an additional deterministic approach, namely, interval analysis [15].

Interval analysis and other methods with *result verification* can describe and forward-propagate[2] bounded uncertainty in parameters deterministically if an appropriate implementation of a mathematical model is possible. This *computerized model* can be symbolic (mathematical equations describing the system of interest)

[2] Although approaches for inverse propagation exist (a good overview is in [19]), the more usual application is for the forward problem

or algorithmic (code). There are many ready-made implementations of interval counterparts to the usual floating point based software: set arithmetic [5, 13] (basic operations $+, -, \cdot, /$; elementary functions such as the sine) and more complex methods such as those for solving linear and non-linear systems of equations or initial value problems [8, 16]. Many more references can be provided in each case. Originally, methods with result verification were developed to address the question of reliability by proving formally that the outcome of a simulation implemented on a computer was correct (assuming that the underlying implementation was correct). The results are usually sets of floating point numbers which with certainty contain the exact solution to the computerized model. The advantage is that usual numerical assumptions such as truncation or discretization cannot lead to a wrong solution and their negative influence does not remain undetected if result verification is used. A common drawback is the possibility of too conservative bounds for the solution sets (e.g., between $-\infty$ and $+\infty$) caused by the dependency problem or the wrapping effect [14]. Aside from previous work by the authors, interval analysis has been applied to MIMO systems in [24], however, in the context of identification.

The structure of the paper is as follows: In Section 2, the mathematical background of the SVD and GMD decomposition is briefly outlined along with the formulas for the corresponding BER (including quantification of possible bounded epistemic uncertainty). In Section 3, the process of resource allocation is considered from the mathematical point of view. In Section 4, we apply the suggestions from Section 3 to a benchmark MIMO system with four receiving and four transmitting antennas for which a correlated and uncorrelated situations are simulated. We consider bit and power allocation for this system and compare the SVD and GMD channel separation, especially detailed for two active layers. Conclusions are in the last section.

2 Interference Suppression: SVD, GMD, BER

The channel matrix H identified at the second stage in the MIMO modeling and simulation process from Figure 1 is usually dense so that it is not possible to distinguish separate SISO layers. In this section, we give a brief outline of common techniques to determine the non-interfering SISO layers corresponding to the channel described by H using the singular value and the geometric mean decompositions in Subsections 2.1 and 2.2, respectively. The description in Subsection 2.1 relies on [18], in Subsection 2.2 on [11]. Note that the SVD based technique is at the moment the standard one. Moreover, the interference suppression between the SISO channels described here is ideal and works in theory. In practice, there can still be residual interference. In Subsection 2.3, we provide the formulas for the corresponding BER along with its upper bound for the case that certain parameters are not known exactly.

2.1 SVD Based Channel Separation

The channel matrix from Eq. (1) is decomposed as $H = U \cdot \Sigma \cdot V^\dagger$, where U and V are unitary matrices, Σ is the diagonal matrix with real elements and $V^\dagger$ denotes the Hermitian transpose of V. The matrix Σ contains the positive square roots of the eigenvalues ξ_l of $H^\dagger H$ in descending order on the main diagonal (*singular values* denoted by $\lambda_l = \sqrt{\xi_l}$ throughout the paper). If a pre-processed data vector $\vec{x} := V \cdot \vec{a}$ is considered and the corresponding receive signal $\vec{z} := H\vec{x} + \vec{n}$ is post-processed by $U^\dagger$, then the new receive signal is

$$\vec{u} := U^\dagger \vec{z} = U^\dagger \left(U \Sigma V^\dagger\right) V\vec{a} + U^\dagger \vec{n} = \Sigma \vec{a} + \vec{w} \ , \tag{2}$$

where the vector $\vec{a}$ is the transmitted signal vector and $\vec{n}$ is the Gaussian noise vector as explained in the Introduction. In this way, the MIMO link is transformed (ideally) into $n = \min\{n_\mathrm{T}, n_\mathrm{R}\}$ independent, non-interfering SISO layers u_l having (unequal) weights λ_l (satisfying the condition $\lambda_1 > \lambda_2 > \ldots > \lambda_n$):

$$u_l = \lambda_l a_l + w_l \quad \text{for} \quad l = 1 \ldots n \ . \tag{3}$$

Out of those, only $L = 2, \ldots, n$ layers need to be actively used while transmitting information. Since the weights for each layer are different, it is profitable to optimize the allocation of such resources as transmit power. In poor scattering conditions with high antenna correlation, where the weighting of the SISO channels might turn strongly unequal, such optimization gains in importance but is challenging. A unique indicator of the unequal weighting of the MIMO layers is the ratio ϑ between the smallest and the largest singular value which characterizes the correlation effect (and is also the condition number of the matrix H).

2.2 GMD Based Channel Separation

The channel matrix from Eq. (1) is decomposed as $H = Q \cdot R \cdot P^\dagger$, where P and Q are semiunitary[3] matrices, $R \in \mathbb{R}^{n \times n}$ is an upper triangular matrix with identical diagonal elements. After precoding the signal $\vec{a}$ by P at the transmitter side ($\vec{x} = P\vec{a}$) and postcoding $\vec{z} = H\vec{x} + \vec{n}$ by $Q^\dagger$ at the receiver side, the model in Eq. (1) turns into

$$\vec{u} = R\vec{a} + Q^\dagger \vec{n} = R\vec{a} + \vec{\nu} \ . \tag{4}$$

By using appropriate nulling and cancellation approaches [9], it is possible to obtain n parallel, non-interfering SISO links of the form

$$u_l = \lambda \cdot a_l + \nu_l \text{ with equal weights } \lambda = \left(\prod_{l=1}^{n} \lambda_l\right)^{\frac{1}{n}} \ . \tag{5}$$

Since all SISO layers are equal, it should not be necessary to optimize wrt. the amount of bits per symbol or power, which can be chosen to be the same for each

[3] that is, non-square matrices either the rows or columns of which are orthonormal

active layer [10]. However, as will be described in Subsection 2.3, it might still be advantageous to select the number of active layers (i.e., the number of activated MIMO layers $L = 2, \ldots, n$ for the data transmission within an $n_{\mathrm{T}} \times n_{\mathrm{R}}$ MIMO system) to be less than n. The actual value of λ strongly depends on L and can be computed as $\lambda = \left(\prod_{l=1}^{L} \lambda_l \right)^{\frac{1}{L}}$.

Note that we do not consider explicitly the added uncertainty appearing due to the use of precoding/postcoding (SVD) or nulling/cancellation (GMD) in this paper. This point has been addressed, for example, in [3].

2.3 BER for SVD and GMD Separated Channels

For simple MIMO transmission channel and data source models, the BER can be computed analytically [3]. In particular, for quadrature amplitude modulated signals, the bit error probability for a transmission SISO layer l is given as

$$p_{\mathrm{b}}^{(l)} = \frac{2}{m_l} \left(1 - 2^{-\frac{m_l}{2}}\right) \cdot \mathrm{erfc}\left(\frac{\lambda_l}{2\sigma} \sqrt{\frac{3 \cdot P_{\mathrm{s}}^{(l)}}{2^{m_l} - 1}} \right) \tag{6}$$

if the SVD is used. It depends on the amount of bits per symbol m_l (and the constellation size $M_l = 2^{m_l}$), the noise standard deviation σ, the available transmission power per layer $P_{\mathrm{s}}^{(l)}$ and the singular value λ_l corresponding to the considered layer; $\mathrm{erfc}(\cdot)$ is the complementary error function. The desired throughput is denoted by

$$T = \sum_{l=1}^{L} \log_2 M_l = \sum_{l=1}^{L} m_l \tag{7}$$

and considered to be constant throughout the paper. The BER for the whole MIMO link is the sum of probabilities per layer modified with the respective number of bits per layer and the throughput:

$$p_{\mathrm{b}} = \frac{1}{T} \sum_{l=1}^{L} m_l \cdot p_{\mathrm{b}}^{(l)} = \frac{2}{T} \sum_{l=1}^{L} \left(1 - 2^{-\frac{m_l}{2}}\right) \cdot \mathrm{erfc}\left(\frac{\lambda_l}{2\sigma} \sqrt{\frac{3 \cdot P_{\mathrm{s}}^{(l)}}{2^{m_l} - 1}} \right) . \tag{8}$$

The weights λ_l are not necessarily equal for each SISO layer if the SVD is used, which is usually countered by assigning power to layers in the optimal way instead of uniformly ($P_{\mathrm{s}}^{(l)} = \frac{P_{\mathrm{s}}}{L} \rightarrow P_{\mathrm{s}}^{(l)} = \pi_l^2 \cdot \frac{P_{\mathrm{s}}}{L}$). That is, employing the analytical BER representation as a cost function, a MIMO system can be optimized wrt. the parameters π_l, for example, with the help of the Lagrange multipliers approach. The noise variance σ^2 is usually considered to be fixed, but it is possible to optimize the BER still further with the help of bit allocation. Here, the number of bits per symbol m_l for L active layers is computed such that the BER is minimized. Since m_l are natural numbers, the integer optimization problem needs to be solved.

The BER of a correlated system can become significantly higher than that of an uncorrelated one.

Another possibility to deal with the unequal weights and to minimize the BER is to use GMD as described in Subsection 2.2. The formula in Eq. (8) remains almost the same except that, instead of the singular values λ_l of the channel matrix H, their geometric mean λ is used for each active layer to compute $p_{\mathrm{b}}^{(l)}$. Additionally, if the number of bits per symbol is chosen to be the same ($m = m_1 = m_2 = \ldots = m_L$) and the overall power P_{s} is equally distributed among the L active layers, the BER in Eq. (8) simplifies to

$$p_{\mathrm{b,GMD}} = \frac{mL}{T} p_{\mathrm{b}}^{(*)} = \frac{2L}{T}\left(1 - 2^{-\frac{m}{2}}\right) \cdot \operatorname{erfc}\left(\frac{\lambda}{2\sigma}\sqrt{\frac{3 \cdot P_{\mathrm{s}}}{L(2^m - 1)}}\right) \quad . \tag{9}$$

Note that if T is constant, $m = \frac{T}{L}$. Since $m \in \mathbb{N}$, the number of active layers L should be chosen such that $\frac{T}{L} \in \mathbb{N}$.

Since the BER in both (8) and (9) is essentially a sum of positive values of the corresponding $p_{\mathrm{b}}^{(l)}$, it might be profitable to choose the number of active layers L to be less than n, possibly switching off the layer with the highest $p_{\mathrm{b}}^{(l)}$ in the case of SVD.

If the formula in Eq. (8) (or Eq. (9)) is used to compute the overall BER, it is obvious that the major characteristics influencing this quality criterion are the singular values λ_l (layer weights), the standard deviation of the noise σ, the numbers of bits per symbol m_l, the transmit power per layer $P_{\mathrm{s}}^{(l)}$ and the number of activated layers L. As already mentioned, the throughput T is assumed to be constant. Initially, $P_{\mathrm{s}}^{(l)} = P_{\mathrm{s}}/L$ is equal for each layer and is optimized during the stage of power allocation. Optimizing wrt. m_l is the purpose of bit allocation. Choosing the number of active layers L also belongs to the stage of resource allocation. If it holds for the remaining parameters that $\lambda_l \in [\, \underline{\lambda}_l, \overline{\lambda}_l \,]$, where $\underline{\lambda}_l$, $\overline{\lambda}_l$ are known lower and upper bounds, respectively, and the standard deviation $\sigma \in [\, \underline{\sigma}, \overline{\sigma} \,]$, then a conservative upper bound on the BER can be obtained using the rules of interval arithmetic as

$$p_{\mathrm{b}}(\sigma, \lambda_1 \ldots \lambda_L) \leq \frac{2}{T} \sum_{l=1}^{L} \left(1 - 2^{-\frac{m_l}{2}}\right) \cdot \operatorname{erfc}\left(\frac{\underline{\lambda}_l}{2\overline{\sigma}}\sqrt{\frac{3 \cdot P_{\mathrm{s}}^{(l)}}{2^{m_l} - 1}}\right) \tag{10}$$

(cf. [2]). That is, due to monotonicity of the involved functions, it is not necessary to work with actual ranges but with their bounds only, which makes verified optimization easier. Note that the upper bound for σ would be achieved, theoretically, at the SNR below 1dB, which is of no practical interest since the signal would be too 'noisy' to be considered useful. Therefore, it is common practice to choose a certain fixed SNR $\frac{E_s}{N_0}$ in dB (e.g., between 5dB and 20dB) at which the behavior of the MIMO system is studied.

3 Bit and Power Allocation

If the SISO channels are separated via the SVD, the overall system can be optimized wrt. the quality criterion of the BER by appropriate allocation of resources. The first factor to consider is the number of active layers L to use. Next, the amount of information that should be put through each of the active layers $l = 1 \dots L$ can be optimized (bit allocation). Finally, the power assigned to each active layer can optimized (power allocation). In this section, we consider these processes in detail. Where necessary, we mention the behavior of the GMD separated MIMO system in the context.

3.1 Bit Allocation

3.1.1 Bit Allocation with Non-Linear Mixed-Integer Programming (Exact)

The number of bits per transmitted symbol and layer influences the overall BER as can be seen from Eq. (8). Optimizing with respect to bits per symbol (that is, with power equally distributed among the L active layers) is not trivial since m_l should be positive integers fitting the desired throughput T from Eq. (7), which leads to a non-linear mixed-integer programming problem [12]. For small numbers of active layers L and constant throughputs T, the problem can be treated by considering all admissible combinations of $m_l \in \{1, 2, \dots, T - L + 1\}$ satisfying the constraint in Eq. (7) and choosing the combination with the smallest p_b [1]. For small T, it is easy to implement a routine checking the BER produced by all admissible constellations of m_l. For example, an overall of 21 combinations needs to be tested for $T = 8$ (bit/s)/Hz and three active layers ($L = 3$). For high numbers of antennas and higher throughput, this simple routine would be too inefficient wrt. computing time. Even increasing the number of active layers by one (to $L = 4$) results in an increase by 14 combinations (i.e., there are 35 combinations to check). Since $\lambda_1 > \lambda_2 > \ldots > \lambda_L$, that is, the first layer is the strongest (has the smallest BER), the second is the second strongest and so on, it is usually assumed that also $m_1 \geq m_2 \geq \ldots \geq m_L$, which corresponds to the practical consideration that stronger layers should transmit more information. This assumption reduces the number of combinations from 21 to five for the example with the three active layers.

As is obvious from Eq. (6), the error probability $p_\mathrm{b}^{(l)}$, considered as a function of $\lambda_l > 0$, decreases monotonically and has a positive range. It is not easy to answer the same question for $p_\mathrm{b}^{(l)}$ as a function of m_l, even over the limited definition domain $[1,\ T - L + 1]$. It can be monotonically increasing or decreasing with m_l for some values of λ_l, or not be monotonic at all (cf. the example in Figure 2). However, the range is always positive over $[1,\ T - L + 1]$, as can be easily seen from Eq. (6). The behavior wrt. monotonicity is explained by the derivatives wrt. to m_l and λ_l:

$$\frac{\partial p_{\mathrm{b}}^{(l)}}{\partial m_l} = -\frac{2}{m_l^2}\left(1-2^{-\frac{m_l}{2}}\right)\cdot \operatorname{erfc}\left(\frac{\lambda_l}{2\sigma}\sqrt{\frac{3\cdot P_{\mathrm{s}}^{(l)}}{2^{m_l}-1}}\right)+$$

$$\frac{\ln 2}{m_l}\cdot\left(2^{-m_l/2}\cdot \operatorname{erfc}\left(\frac{\lambda_l}{2\sigma}\sqrt{\frac{3P_{\mathrm{s}}^{(l)}}{2^{m_l}-1}}\right)+\right. \tag{11}$$

$$\left.\frac{\lambda_l}{\sigma}\sqrt{\frac{3P_{\mathrm{s}}^{(l)}}{\pi}}\cdot\frac{2^{m_l/2}}{\sqrt{2^{m_l}-1}\left(2^{m_l/2}+1\right)}\cdot\exp\left(-\frac{\lambda_l^2}{4\sigma^2}\frac{3P_{\mathrm{s}}^{(l)}}{(2^{m_l}-1)}\right)\right)$$

$$\frac{\partial p_{\mathrm{b}}^{(l)}}{\partial \lambda_l} = -\frac{2}{m_l\sigma}\sqrt{\frac{3P_{\mathrm{s}}^{(l)}}{\pi\,(2^{m_l}-1)}}\cdot\left(1-2^{-\frac{m_l}{2}}\right)\cdot\exp\left(-\frac{\lambda_l^2}{4\sigma^2}\frac{3P_{\mathrm{s}}^{(l)}}{(2^{m_l}-1)}\right)\ . \tag{12}$$

It holds that $\frac{\partial p_{\mathrm{b}}^{(l)}}{\partial \lambda_l} < 0$ for $m_l \geq 1$, $\sigma > 0$ (which is the case). No such simple statement can be derived for $\frac{\partial p_{\mathrm{b}}^{(l)}}{\partial m_l}$ in Eq. (11) since it contains both positive and negative terms depending on the same parameters and variables. Therefore, it is, strictly speaking, not clear that taking $m_1 \geq m_2 \geq \ldots \geq m_L$ would produce the minimal value for the BER in Eq. (8) under the constraint (7). However, if the BER is computed according Eq. (8), where $p_{\mathrm{b}}^{(l)}$ is multiplied by the corresponding m_l, the negative term disappears from the derivative $\frac{\partial p_{\mathrm{b}}}{\partial m_l}$ making it always positive (cf. Eq. (15)). That is, the overall BER is a sum of functions monotonically increasing with m_l. Since $\operatorname{erfc}(\lambda_1) < \operatorname{erfc}(\lambda_2) < \ldots < \operatorname{erfc}(\lambda_L)$, taking $m_1 \geq m_2 \geq \ldots \geq m_L$ (and therefore $M_1 \geq M_2 \geq \ldots \geq M_L$) seems a good choice, which is also confirmed experimentally in the next Section.

3.1.2 Bit Allocation with Lagrange Multipliers (Approximate)

Another approach to bit allocation is to approximately solve the problem by the Lagrange multipliers method. The task is

$$p_{\mathrm{b}}(m_1,\ldots,m_L) \underset{m_1\ldots m_L}{\longrightarrow} \min \quad \text{s.t.} \sum_{l=1}^{L} m_l = T \quad \text{where } P_{\mathrm{s}}^{(l)} = \frac{P_{\mathrm{s}}}{L}\ . \tag{13}$$

The cost function to consider is then

$$J(m_1,\ldots,m_L,\mu) = p_{\mathrm{b}}(m_1,\ldots,m_L) + \mu\cdot\left(-T+\sum_{l=1}^{L} m_l\right) \tag{14}$$

if we are interested in the number of bits per symbol m_l. This formulation disregards that the numbers m_l should be positive integers and would possibly compute

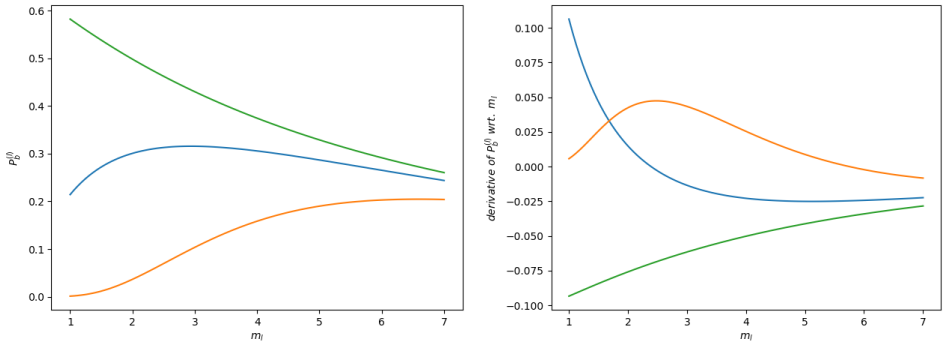


Figure 2: BER $p_{\mathrm{b}}^{(l)}$ per layer l with $P_{\mathrm{s}}^{(l)} = \frac{1}{3}$ as a function of m_l (left) and its derivative wrt. m_l (right) for an example system from Subsection 4.1 for $\lambda_l = 1.3791$ (orange), $\lambda_l = 0.1609$ (blue) and $\lambda_l = 0.0013$ (green).

approximate real values for them. The system we need to solve to obtain candidates for the optimal sets of bits per symbol m_l for $l = 1 \dots L$ is

$$\frac{\partial J}{\partial m_l} = \frac{\ln 2}{T} \cdot \left(2^{-m_l/2} \cdot \operatorname{erfc}\left(\frac{\lambda_l}{2\sigma} \sqrt{\frac{3P_{\mathrm{s}}^{(l)}}{2^{m_l}-1}} \right) + \right. \tag{15}$$

$$\left. \frac{\lambda_l}{\sigma} \sqrt{\frac{3P_{\mathrm{s}}^{(l)}}{\pi}} \cdot \frac{2^{m_l/2}}{\sqrt{2^{m_l}-1}\left(2^{m_l/2}+1\right)} \cdot \exp\left(-\frac{\lambda_l^2}{4\sigma^2} \frac{3P_{\mathrm{s}}^{(l)}}{(2^{m_l}-1)} \right) \right) + \mu = 0$$

$$\frac{\partial J}{\partial \mu} = -T + \sum_{l=1}^{L} m_l = 0 \tag{16}$$

Having computed the enclosures of the approximate real values for m_l, for example, using a solver for systems of non-linear equations based on methods with result verification (e.g., C-XSC Toolbox [8]), we can choose positive integer values that are the closest to them. Here, rounding to the nearest integer number actually provides the results that fulfil the constraint (cf. Table 1). However, we can also round the first $L-1$ powers only and then subtract their sum from T.

Both bit and power allocation are theoretically unnecessary for the GMD. Since all weights are the same for the SISO channels obtained by the GMD, the power can indeed be equally distributed if also all m_l are the same. However, it is not immediately clear why the bit allocation is unnecessary, aside from the practical reason that equally strong layers can transmit the equal amount of information. Besides, these statements are usually true only asymptotically (for high SNRs). Therefore, we seek to study how the GMD behaves under bit allocation also experimentally in Section 4.

3.2 Power Allocation

Once the number of transmitted bits per symbol is fixed, a further approach to BER minimization is to reassign the initially uniformly distributed transmission power. Assigning more power to layers with small λ_l seems a good strategy to improve the overall BER since small λ_l lead to large values of bit error probability $p_{\mathrm{b}}^{(l)}$ per MIMO layer l as implied by the upper bound in Eq. (10) (which is however not always true if both bit and power allocation are performed, cf. Page 793). The task here is

$$p_{\mathrm{b}}(\pi_1,\ldots,\pi_L) \underset{\pi_1\ldots\pi_L}{\longrightarrow} \min \quad \text{s.t.} \quad \sum_{l=1}^{L} \pi_l^2 = L \quad \text{where } P_{\mathrm{s}}^{(l)} = \frac{\pi_l^2 P_{\mathrm{s}}}{L} \ . \tag{17}$$

The Lagrange multipliers method is formulated for this task using the following cost function:

$$\begin{aligned} J(\pi_1\ldots\pi_L,\mu) \quad &= \quad \frac{2}{T}\sum_{l=1}^{L}\left(1-2^{-\frac{m_l}{2}}\right) \\ &\cdot \quad \operatorname{erfc}\left(\frac{\pi_l\lambda_l}{2\sigma}\sqrt{\frac{3\cdot P_{\mathrm{s}}}{L(2^{m_l}-1)}}\right)+\mu\left(\sum_{l=1}^{L}\pi_l^2-L\right)\longrightarrow \min, \end{aligned} \tag{18}$$

where $\pi_1 > 0,\ldots,\pi_L > 0$ are the power allocation parameters with which we modify the initially assigned power $P_{\mathrm{s}}^{(l)}$ in order to improve p_{b} from Eq. (8). With the notations

$$k_l = k_l(m_l,T) \quad := \quad \frac{2}{T}\cdot\left(1-2^{-\frac{m_l}{2}}\right), \tag{19}$$

$$c_l = c_l(m_l,\sigma,P_{\mathrm{s}},L) \quad := \quad \frac{1}{2\sigma}\sqrt{\frac{3\cdot P_{\mathrm{s}}}{L(2^{m_l}-1)}},\ l=1...L\ , \tag{20}$$

the Lagrange multipliers approach produces the nonlinear system of equations (21) for the minimizer candidates of the cost function (18) :

$$\frac{\partial J(\pi_1\ldots\pi_L,\mu)}{\partial\pi_l} = -\frac{2k_l}{\sqrt{\pi}}\left(c_l\lambda_l e^{-c_l^2\lambda_l^2\pi_l^2}\right)+2\mu\pi_l = 0, \quad \sum_{l=1}^{L}\pi_l^2 - L = 0\ , \tag{21}$$

where $\pi_l > 0$, $l \in \{1,\ldots,L\}$. It is clear from the first L equations that μ must be positive. Additionally, the second derivative $\dfrac{\partial^2 J}{\partial\pi_l\partial\pi_m} = 0$ for $l \neq m$ and is positive for $l = m$. The bordered Hessian is symmetric and has the form

$$\begin{pmatrix} 0 & 2\pi_1 & \cdots & 2\pi_L \\ 2\pi_1 & 2\mu + \dfrac{4k_1c_1^2\lambda_1^2}{\sqrt{\pi}}\pi_1 e^{-c_1^2\lambda_1^2\pi_1^2} & \cdots & 0 \\ 2\pi_2 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 2\pi_L & 0 & \cdots & 2\mu + \dfrac{4k_Lc_L^2\lambda_L^2}{\sqrt{\pi}}\pi_L e^{-c_L^2\lambda_L^2\pi_L^2} \end{pmatrix} .$$

It could be shown that the determinants of all relevant $L-1$ leading principal minors are always negative [1]. For one constraint, this means that a stationary point is a local minimum. Using again the solver for systems of non-linear equations from C-XSC Toolbox, we can compute guaranteed enclosures for all stationary points in the search interval $(0, L]$ for each π_l from the system in Eq. (21). Working in a verified way has the advantage of taking care of numerical errors. For power allocation, it has an additional benefit of proving that the solution is really unique if only one possibility is suggested. Combined with the analytical conclusions above, this leads to the computer-assisted proof that the candidate obtained by solving (21) is actually the global minimum.

3.3 Bit and Power Allocation

The approximate bit and exact power allocation can be performed simultaneously:

$$p_{\mathrm{b}}(\pi_1 \ldots \pi_L, m_1, \ldots, m_L) \underset{\pi_1 \ldots \pi_L, m_1, \ldots, m_L}{\longrightarrow} \min \quad \text{s.t.} \sum_{l=1}^{L} m_l = T, \sum_{l=1}^{L} \pi_l^2 = L \ , \tag{22}$$

where $P_{\mathrm{s}}^{(l)} = \frac{\pi_l^2 P_{\mathrm{s}}}{L}$. The cost function combining those from Eq. (14) and (18)

$$\begin{aligned} J(\pi_1 \ldots \pi_L, m_1, \ldots, m_L, \mu_1, \mu_2) = & \frac{2}{T} \sum_{l=1}^{L} \left(1 - 2^{-\frac{m_l}{2}}\right) \cdot \operatorname{erfc}\left(\frac{\pi_l \lambda_l}{2\sigma} \sqrt{\frac{3 \cdot P_{\mathrm{s}}}{L(2^{m_l} - 1)}}\right) \\ & + \mu_1 \left(\sum_{l=1}^{L} m_l - T\right) + \mu_2 \left(\sum_{l=1}^{L} \pi_l^2 - L\right) \end{aligned} \tag{23}$$

needs then to be minimized. The corresponding non-linear system for the stationary points is

$$\begin{aligned} \frac{\partial J}{\partial \pi_l} &= -\frac{2k_l}{\sqrt{\pi}} \left(c_l \lambda_l e^{-c_l^2 \lambda_l^2 \pi_l^2}\right) + 2\mu_1 \pi_l = 0 \qquad (24) \\ \frac{\partial J}{\partial m_l} &= \frac{\ln 2}{T} \cdot \left(2^{-m_l/2} \cdot \operatorname{erfc}(c_l \lambda_l \pi_l) + \frac{2 c_l \lambda_l \pi_l 2^{m_l/2}}{\left(2^{m_l/2} + 1\right)} \cdot \exp\left(-c_l^2 \lambda_l^2 \pi_l^2\right)\right) + \mu_2 = 0 \\ \frac{\partial J}{\partial \mu_1} &= \sum_{l=1}^{L} \pi_l^2 - L = 0, \quad \frac{\partial J}{\partial \mu_2} = -T + \sum_{l=1}^{L} m_l = 0 \quad \text{for } l = 1 \ldots L \end{aligned}$$

having now double the number of equations compared to problems in (14) or (18) individually. In Subsection 4.3, we compare this approach to the approach combining exact power allocation by Eq. (18) and exact bit allocation by brute-force combinatorial approach from points of view of both the BER and computing time for a system with two active layers.

Note that it is possible to solve the problems in Eq. (14) or (18) directly (without having to formulate the respective non-linear systems of equations for the candidates) in the verified way by using global optimization algorithms implemented, for example, within the same C-XSC Toolbox. However, the achieved parameter enclosures are too wide as discussed in [3]. That is why we concentrate on solving the systems of non-linear equations in (15)-(16), (21), (24) using verified methods in this paper, which produces almost point enclosures.

4 Numerical Results

In this section, we first take a closer look at the bit allocation for both SVD and GMD based approaches, since it does not seem to be inherently clear that GMD requires the uniform bit distribution in all cases (Subsection 4.2). For this purpose, we consider both an uncorrelated and correlated MIMO system, with 5000 channel realizations each, this benchmark described in Subsection 4.1. After that, we focus on the case of two active layers and compare both approaches in more detail, including power allocation (Subsection 4.3).

4.1 The Benchmark System

The practical problem we consider in this paper is a wireless frequency flat MIMO link with $n_{\mathrm{T}} = n_{\mathrm{R}} = 4$ antennas, the desired constant throughput $T = 8$ (bit/s)/Hz and the available transmit power $P_{\mathrm{s}} = 1\mathrm{W}$. In this setup, correlated and uncorrelated data sets with 5000 realizations for $\lambda_1, \ldots, \lambda_4$ each were generated in a non-verified simulation with $\delta^2 = \frac{1}{2}$. The correlation coefficients at the transmitter and receiver sides were chosen as $\rho^{(\mathrm{RX})} = \rho^{(\mathrm{TX})} = 0.2375$ (see [1] for details as well as [6] for an overview on models and [17] for validity areas of the employed Kronecker channel model). If not mentioned otherwise, we provide results at the SNR of $\frac{E_s}{N_0} = 10\mathrm{dB}$ ($\sigma^2 = 0.05$). All simulations in this paper are carried out using Intel i7-4790K @ 4.00GHz (8 cores) CPU under Ubuntu 20.04 LTS and are implemented using C++.

4.2 SVD vs GMD: Bit Allocation

In [1], we tested the correlated and uncorrelated MIMO systems described above with their 5000 realizations using verified power allocation and manual bit allocation for the SVD based approach under the restriction $m_1 \geq m_2 \geq \ldots m_L$ with two, three or four active layers. In this subsection, we take a closer look at the bit allocation for both the SVD and GMD based approach, considering two versions.

The first variant (denoted 'NLI' for convenience in the following) implements the brute force approach of computing the BER using interval arithmetic for all possible combinations of $m_l = \log_2 M_l$ (cf. Section 3.1.1). The second variant (denoted 'OPT') solves the system of non-linear equations in Eqs. (15)–(16) using the corresponding verified solver from C-XSC Toolbox to obtain enclosures of approximate real values for m_l (cf. Section 3.1.2). Using methods with result verification, we can, in a guaranteed way, rule out the situation that there exist different, better local minimizers in the considered search interval if the unique solution is obtained.

In Table 1, the results are summarized for four representative examples. First, we consider a randomly chosen uncorrelated data set (№4998, Case 1) and a randomly chosen correlated data set (№101, Case 3) from the available 5000 realizations for each case. For both uncorrelated and correlated MIMO systems, we examine additionally the worst case by taking the smallest values for every of the four λ_l out of the 5000 realizations (Case 2 and 4, respectively). By the formula in Eq. (10), these values provide an upper bound for the achievable BER for all the 5000 realizations considered. Note that this conservative upper bound on the BER of our realizations is not attained by any of the actual realizations since the lower bounds on each of λ_i are not necessarily contained in a single set $\lambda_1, \ldots, \lambda_L$. All possible values of the BER are below this bound. In Columns 2–6 of the Table, the values of m_l and the BER for the SVD based approach are given for a predefined number of active layers L (Column 1) and both variants OPT (the first of the corresponding lines) and NLI (where the integer numbers are given). The structure of Columns 7–11 describing the GMD based approach is the same aside from the additional BER value in parentheses showing, as a comparison, the SVD based BER for the m_l values optimal for the GMD. The results are computed for the SNR of 10dB.

We reproduce the midpoints of the obtained enclosures for m_l and the upper bounds of the obtained intervals for the BER; all values are rounded to five digits. The solver in OPT is used with the tolerance of 10^{-10} and the search interval of $[0.9, 8.1]$ for each of the m_l along with the $[-2, 0]$ for the μ. The width of the obtained intervals has the maximum order of magnitude of 10^{-10}. Sometimes, OPT could not verify a solution in the chosen search interval (denoted by – in the Table). One reason is that values below one are suggested for weaker layers (agreeing also with the results from non-verified solvers). However, this would correspond to switching the weaker layer off, which we want to explicitly control by choosing the number of active layers L manually. That is why we do not provide an OPT solution in these cases.

The values computed by OPT always agree with NLI in the sense that positive integers closest to the real values from OPT are also suggested for m_l by the NLI. This indicates that the system in Eqs. (15)–(16) can be used in combination with verified or non-verified solvers to compute approximate values for m_l if the NLI approach takes too long for the given number of active layers and the throughput. Obviously, the OPT based BER is somewhat better than the corresponding NLI based one. Verified computations using both NLI and OPT confirm that the GMD based approach is indeed on average at its optimal for equal numbers of bit per symbol m_l. Using three active layers $L = 3$ is not a good scenario for GMD for our

benchmark systems since we cannot achieve the throughput of $T = 8$ with equal m_l that are positive integers. It is nonetheless interesting that $m_l = \frac{T}{3}$ is indeed the best choice at a relatively low SNR of 10dB as confirmed by the OPT solution.

In the last example of correlated worst case there were 4 candidates for the system's solution for $L = 3$ (in italics) if the OPT was used. The expected solution $m_l = T/L$ is actually not the only one producing the minimum for the BER, there are three more. Since the real solution needs to be rounded to positive integers, those further candidates can be taken as corresponding to 3 possibilities to assign the numbers 3,3,2 to the three m_l. The order is not important for the GMD based approach since the parameters of the $p_{\mathrm{b}}^{(l)}$ are otherwise equal. The BER for the GMD is the same for all possibilities and is given in the Table. The best BER for these m_l with SVD is actually produced by the sequence $3-2-3$ ($1.9214{\cdot}10^{-1}$). It is, however, higher than that for the optimal set $5-2-1$ reproduced in the Table.

Bit allocation does not bring improvement for the GMD in most of the cases from Table 1 at 10dB. However, the best BER under GMD is achieved for 1,1,1,5 (or any other permutation of these numbers) in the uncorrelated and correlated worst case (0.2990 and 0.3633, respectively). As will be shown in the next subsection, there is also one case at 5dB where bit allocation makes sense for the GMD with $L = 2$ active layers. That is, the bit allocation cannot be ruled out for the GMD and low SNRs.

If bit allocation is performed, the SVD is mostly better than the GMD wrt. the BER at 10dB for the considered examples (if we take into account only positive integer values m_l relevant in the practice). Without the bit allocation, the GMD can be better on average, especially, in the "normal" cases 1 and 3, where it is so independently of the number of the activated layers. Bit allocation is demonstrated to significantly improve the SVD based BER for both uncorrelated and correlated cases. Without bit allocation, the GMD based approach improves the BER especially in the normal uncorrelated case. In the next subsection, we offer a broader comparison taking into account all 5000 realizations of the uncorrelated and correlated system for two active layers.

As a general observation from this subsection, it could be mentioned that it does not make sense to use all four available layers as also followed from our study of SVD under bit and power allocation in [1]. Employing GMD instead of SVD does not seem to change the situation. It is necessary to perform more experiments to substantiate this claim, which is the subject of our future work.

4.3 Detailed Comparison of SVD and GMD for Two Active Layers

In this subsection, we compare in detail how the GMD and SVD based approaches perform for different possibilities in case two layers are switched off in the example system described in Subsection 4.1 ($L = 2$). We consider the same example cases as in the previous subsection at different SNRs. Additionally, we analyze all 5000 realizations of the uncorrelated and correlated channel at different SNRs.

Table 1: Comparison between GMD and SVD wrt. optimal values for $m_l = \log_2 M_l$ without power allocation at 10dB.

	SVD					GMD					
L	m_1	m_2	m_3	m_4	BER	m_1	m_2	m_3	m_4	BER GMD	(BER SVD)
Case 1: uncorrelated data set $\lambda_1 = 4.341226$, $\lambda_2 = 2.178729$, $\lambda_3 = 1.070086$, $\lambda_4 = 0.474591$, $\vartheta \approx 0.11$											
2	4.9113	3.0887			$7.1359{\cdot}10^{-4}$	4	4			$7.8821{\cdot}10^{-4}$	
	5	3			$7.5104{\cdot}10^{-4}$	4	4			$7.8821{\cdot}10^{-4}$	$(55.061{\cdot}10^{-4})$
3	4.3079	2.5510	1.1410		$7.2862{\cdot}10^{-4}$	$\frac{8}{3}$	$\frac{8}{3}$	$\frac{8}{3}$		$14.032{\cdot}10^{-4}$	$(220.67{\cdot}10^{-4})$
	4	3	1		$16.148{\cdot}10^{-4}$	3	3	2		$31.536{\cdot}10^{-4}$	$(78.309{\cdot}10^{-4})$
4	–	–	–	–	–	2	2	2	2	$96.220{\cdot}10^{-4}$	
	4	2	1	1	$149.04{\cdot}10^{-4}$	2	2	2	2	$96.220{\cdot}10^{-4}$	$(680.30{\cdot}10^{-4})$
Case 2: uncorrelated, worst case $\lambda_1 = 2.420786$, $\lambda_2 = 0.917965$, $\lambda_3 = 0.302213$, $\lambda_4 = 0.004891$, $\vartheta \approx 0.002$											
2	5.2343	2.7657			$4.6972{\cdot}10^{-2}$	4	4			$5.1015{\cdot}10^{-2}$	
	5	3			$4.7907{\cdot}10^{-2}$	4	4			$5.1015{\cdot}10^{-2}$	$(7.0148{\cdot}10^{-2})$
3	–	–	–		–	$\frac{8}{3}$	$\frac{8}{3}$	$\frac{8}{3}$		$10.458{\cdot}10^{-2}$	$(13.418{\cdot}10^{-2})$
	5	2	1		$7.1372{\cdot}10^{-2}$	3	3	2		$10.916{\cdot}10^{-2}$	$(11.730{\cdot}10^{-2})$
4	–	–	–	–	–	2	2	2	2	$35.252{\cdot}10^{-2}$	
	4	2	1	1	$13.694{\cdot}10^{-2}$	2	2	2	2	$35.252{\cdot}10^{-2}$	$(22.167{\cdot}10^{-2})$
Case 3: correlated data set $\lambda_1 = 3.9493$, $\lambda_2 = 1.6891$, $\lambda_3 = 0.9149$, $\lambda_4 = 0.3578$, $\vartheta \approx 0.091$											
2	5.1129	2.8871			$3.2611{\cdot}10^{-3}$	4	4			$3.6752{\cdot}10^{-3}$	
	5	3			$3.4050{\cdot}10^{-3}$	4	4			$3.6752{\cdot}10^{-3}$	$(17.115{\cdot}10^{-3})$
3	4.4896	2.3487	1.1616		$3.3103{\cdot}10^{-3}$	$\frac{8}{3}$	$\frac{8}{3}$	$\frac{8}{3}$		$5.6410{\cdot}10^{-3}$	$(34.968{\cdot}10^{-3})$
	5	2	1		$5.6579{\cdot}10^{-3}$	3	3	2		$9.4616{\cdot}10^{-3}$	$(18.886{\cdot}10^{-3})$
4	–	–	–	–	–	2	2	2	2	$27.299{\cdot}10^{-3}$	
	4	2	1	1	$26.776{\cdot}10^{-3}$	2	2	2	2	$27.299{\cdot}10^{-3}$	$(90.895{\cdot}10^{-3})$
Case 4: correlated, worst case $\lambda_1 = 1.3791$, $\lambda_2 = 0.5526$, $\lambda_3 = 0.1609$, $\lambda_4 = 0.0013$, $\vartheta \approx 0.0009$											
2	5.6754	2.3246			$1.3631{\cdot}10^{-1}$	4	4			$1.4350{\cdot}10^{-1}$	
	6	2			$1.3667{\cdot}10^{-1}$	4	4			$1.4350{\cdot}10^{-1}$	$(1.4033{\cdot}10^{-1})$
3	–	–	–			$\frac{8}{3}$	$\frac{8}{3}$	$\frac{8}{3}$		*2.2482*$\cdot 10^{-1}$	$(2.0133{\cdot}10^{-1})$
	5	2	1		$1.7307{\cdot}10^{-1}$	3	3	2		$2.2418{\cdot}10^{-1}$	$(1.9441{\cdot}10^{-1})$
4	–	–	–	–	–	2	2	2	2	$4.2949{\cdot}10^{-1}$	
	4	2	1	1	$2.3086{\cdot}10^{-1}$	2	2	2	2	$4.2949{\cdot}10^{-1}$	$(2.7613{\cdot}10^{-1})$

From Figure 3, it can be seen that the GMD approach is better than the one based on the SVD if resources are allocated uniformly ($P_{\mathrm{s}}^{(l)} = \frac{1}{2}\mathrm{W}, m_l = 4, l = 1, 2$). The Figure shows the comparison for the example systems with 5000 realizations each (uncorrelated on the left, correlated on the right) at 10dB and 15dB. For clarity of the representation, only every 100th result is shown. For frequency plots on λ_l, see [1].

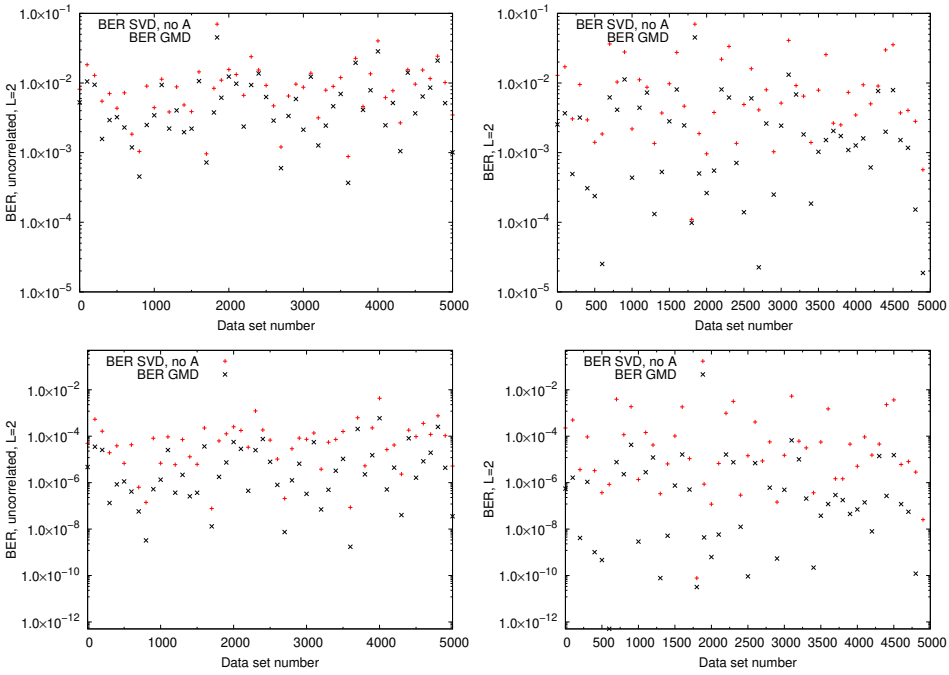


Figure 3: BER for SVD and GMD without any resource allocation for 5000 realizations (with every 100th result shown) of an uncorrelated (left) and correlated (right) MIMO link at 10dB (above) and 15dB (below).

In Figure 4, the comparison between the BER for the GMD based approach with uniformly allocated resources and SVD without power allocation but with bit allocation by OPT is shown, again for 5000 realizations of the uncorrelated and correlated MIMO channels at 10dB and 15dB. The SVD based approach with bit allocation is always better than the GMD one in this case. The unique optimum for m_l cannot always be verified. While optimal solutions can be verified for all 5000 channel realizations in the non-correlated case, 10 cases are not solved for the correlated channel. At 15dB, the corresponding numbers for unsolved cases are 2435 and 3661, respectively. In this subsection, the optimum was considered as not verified if there was no solution to the system in Eqs. (15)–(16) with $L = 2$ in the considered search interval or if there were multiple solutions.

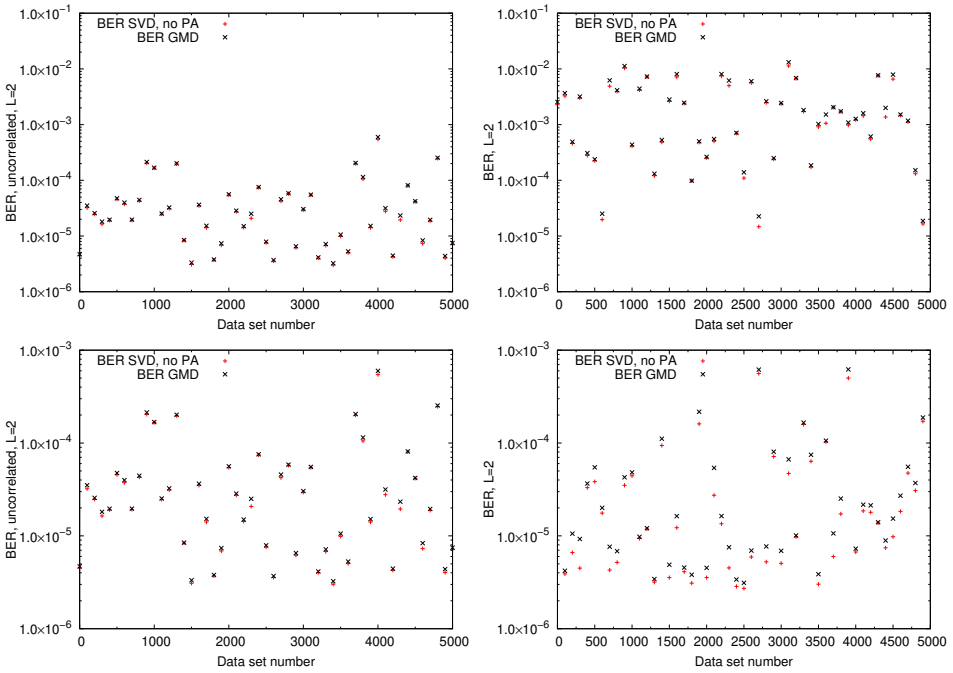


Figure 4: BER for SVD with bit allocation and GMD for 5000 realizations (with every 100th result shown) of an uncorrelated (left) and correlated (right) MIMO link at 10dB (above) and 15dB (below).

Next, we consider how the examples from Table 1 behave if both bit and power allocation (BPA) are performed for the SVD separated channels using the NLI and OPT approaches at different SNRs. Here, NLI means that the system in Eqs. (15)–(16) is solved by C-XSC for each admissible combination of m_l computed by a brute force approach to identify the combination leading to the smallest BER. For OPT, we solve the system in Eq. (24) using C-XSC. In case of the GMD, we perform only bit allocation as a comparison and for the sake of completeness. There is only one case where non-uniform bit allocation is better for the GMD. Therefore, we provide the numbers for m_l in this case only. The results are given in Table 2. For BPA with OPT, the numbers given for m_l are the closest integers. We see that there is no difference in m_l if we do bit and parameter allocation separately by verified non-linear equations solver and optimization (NLI) or together using the system in Eq. (24) by a non-linear solver (OPT) as long as T is fixed. There is, as expected, a difference in the BER, although not very large. From Table 2, it is evident that not only power but also bit allocation depend on both σ and λ_l and cannot be precomputed. Although, at least for smaller σ, the BER of GMD is sometimes better than that of SVD after bit allocation, power allocation makes the BER for

the SVD separated channels the best in all of the considered cases.

Note that the assumption that power allocation assigns more power to weaker layers is not always correct as demonstrated for $L = 2$. For example in Case 3, the power $\pi^2 \cdot P_s^{(1)}$ assigned to the first (and strongest) layer with $\lambda_1 = 3.9493$ is approximately 0.54361 W ($\pi_1 \approx 1.0427$), whereas the second layer with $\lambda_2 = 1.6891$ is assigned 0.45639 W ($\pi_2 \approx 0.95539$) to achieve the minimal BER of $3.2192 \cdot 10^{-3}$ at $m_1 \approx 5.2254$, $m_2 \approx 2.8871$ and the SNR of 10dB given in Table 2 if OPT is used. This is true for all 5000 data sets and is similar for the uncorrelated channel (0.53164 W for the first layer, 0.46836 W for the second in Case 1): the stronger layer is always assigned more power for two active layers at 10dB.

Table 2: Comparison between the GMD and SVD based BER under bit and power allocation for two active layers and the examples from Table 1.

		BER SVD			BER GMD
	dB	BA	BPA (NLI)	BPA (OPT)	BA (4-4)
Case 1	5	$3.0196 \cdot 10^{-2}$(5-3)	$2.9767 \cdot 10^{-2}$(5-3)	$2.9767 \cdot 10^{-2}$(5-3)	$3.1398 \cdot 10^{-2}$
	10	$7.5104 \cdot 10^{-4}$(5-3)	$7.0689 \cdot 10^{-4}$(5-3)	$7.0686 \cdot 10^{-4}$(5-3)	$7.8821 \cdot 10^{-4}$
	15	$1.8489 \cdot 10^{-8}$(5-3)	$1.2815 \cdot 10^{-8}$(5-3)	–	$1.6971 \cdot 10^{-8}$
Case 2	5	$1.4185 \cdot 10^{-1}$(6-2)	$1.3980 \cdot 10^{-1}$(6-2)	$1.3933 \cdot 10^{-1}$(6-2)	$1.5070 \cdot 10^{-1}$
	10	$4.7907 \cdot 10^{-2}$(5-3)	$4.7895 \cdot 10^{-2}$(5-3)	$4.6371 \cdot 10^{-2}$(5-3)	$5.1015 \cdot 10^{-2}$
	15	$3.2918 \cdot 10^{-3}$(5-3)	$2.8633 \cdot 10^{-3}$(5-3)	$2.5074 \cdot 10^{-3}$(5-3)	$3.0105 \cdot 10^{-3}$
Case 3	5	$5.1756 \cdot 10^{-2}$(5-3)	$5.1672 \cdot 10^{-2}$(5-3)	$5.1168 \cdot 10^{-2}$(5-3)	$5.4896 \cdot 10^{-2}$
	10	$3.4050 \cdot 10^{-3}$(5-3)	$3.3502 \cdot 10^{-3}$(5-3)	$3.2192 \cdot 10^{-3}$(5-3)	$3.6753 \cdot 10^{-3}$
	15	$1.9868 \cdot 10^{-6}$(5-3)	$1.3623 \cdot 10^{-6}$(5-3)	–	$1.6392 \cdot 10^{-6}$
Case 4	5	$1.9676 \cdot 10^{-1}$(7-1)	$1.9618 \cdot 10^{-1}$(7-1)	–	*$2.0156 \cdot 10^{-1}$(7-1)*
					$2.3381 \cdot 10^{-1}$
	10	$1.3667 \cdot 10^{-1}$(6-2)	$1.3441 \cdot 10^{-1}$(6-2)	$1.3440 \cdot 10^{-1}$(6-2)	$1.4350 \cdot 10^{-1}$
	15	$4.2405 \cdot 10^{-2}$(5-3)	$4.2404 \cdot 10^{-2}$(5-3)	$4.1433 \cdot 10^{-2}$(5-3)	$4.5213 \cdot 10^{-2}$

The fact that SVD is better than the GMD if both bit and power allocation is performed using OPT at all SNRs, observed for the four examples in Table 2, is confirmed by the simulation considering all 5000 realizations of the uncorrelated and correlated channel (cf. Figure 5). At 10dB, the minimum could not be verified for 11 data sets in the correlated case using OPT. SVD with BPA is better than GMD in all the remaining cases. At 15dB, the result could not be verified for 3667 (correlated) and 2441 (uncorrelated) data sets using OPT. For the remaining data sets, SVD is better. Note that, although the OPT approach is quite helpful if only bit allocation is performed since it identifies the optimal m_l constellations reliably, it is purely theoretical if both bit and power allocation are combined as in

Eq. (24). The optimal power parameters π_l obtained for the real values of m_l do not necessarily retain their optimality for the corresponding positive integer m_l.

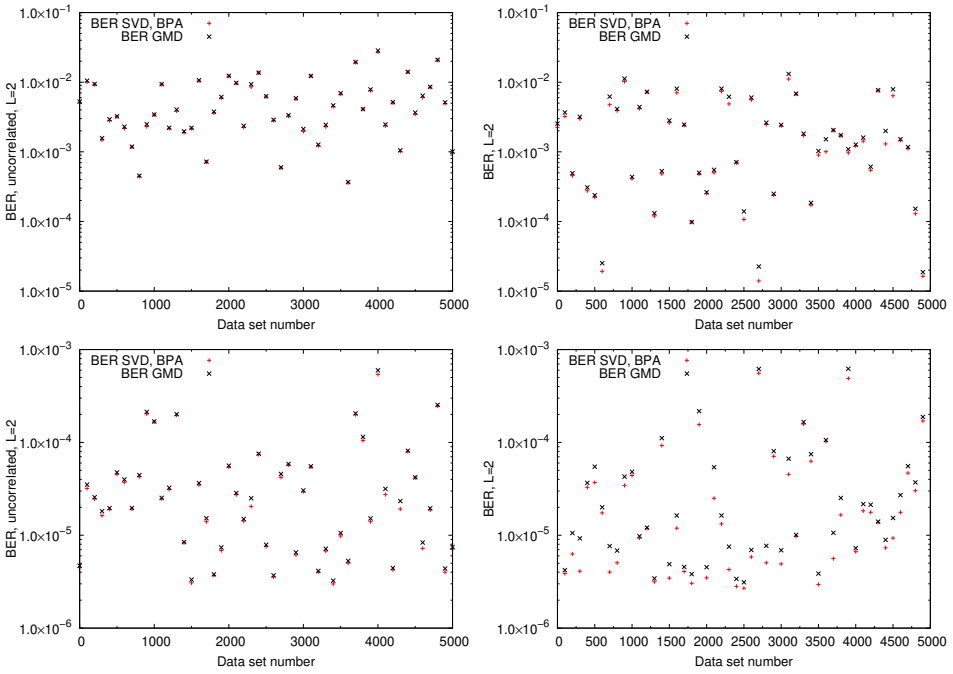


Figure 5: BER for SVD with bit and power allocation and GMD without resource allocation for 5000 realizations (with every 100th result shown) of an uncorrelated (left) and correlated (right) MIMO link at 10dB (above) and 15dB (below).

Using NLI for both bit and power allocation is more relevant in practice. On the one hand, more information can be gained with respect to GMD/SVD comparison since a verified proof of uniqueness is possible for almost all data sets[4]. The reason for this is that only two equations need to be solved for each $m_1 - m_2$ combination in the verified way. On the other hand, the NLI approach uses only positive integer constellations of m_l, leading to realistic values of π_l and the BER.

If BPA for the SVD case is performed via NLI, the SVD based channel separation is not always better as demonstrated in Figure 6, on the left. There, the normalized ratio is shown between the number of cases in which the BER under SVD is better for 5000 realizations of uncorrelated/correlated MIMO systems and the overall number of successful cases. Note that the resources are allocated uniformly in the GMD case. At lower SNRs, SVD is better, especially in the correlated case. At

[4]For example, a verified result cannot be produced for overall 79 combinations in the correlated case using NLI at 15dB, which also includes combinations possibly not leading to the minimal BER

higher SNRs, GMD is better, especially in the uncorrelated case. On the right of Figure 6, the best and worst computed values of the BER are shown at each SNR for correlated and uncorrelated MIMO system under SVD and GMD for 5000 realizations each. We observe that the GMD is mostly better in terms of the upper and lower bounds. That is, out of 5000 realizations, the smallest best case bound (lower) and the smallest worst case bound (upper) are provided by GMD (starting at 12.5dB, at the latest). This does not mean that this is so on average (cf. the Figure on the left). Additionally, it can be seen that the correlated MIMO system has a much broader intervals between the best and the worst achievable BER within the same channel.

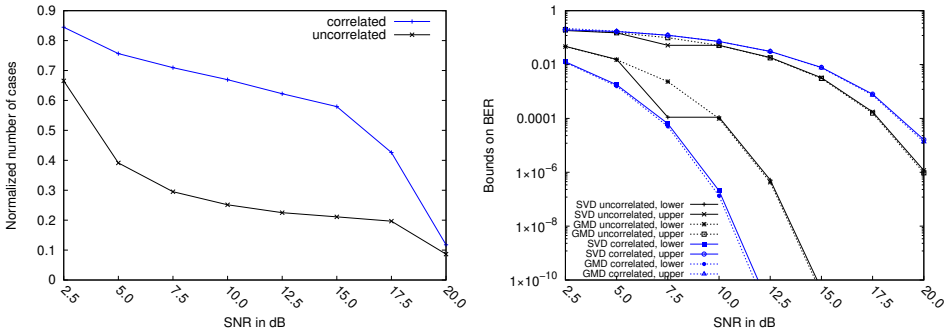


Figure 6: On the left, the normalized number of cases of better BER under SVD and NLI. On the right, lower and upper BER bounds under SVD (solid) and GMD (dashed). In both figures, 5000 realizations of the uncorrelated (black) and correlated (blue) MIMO systems are considered.

To give an idea about the computing times, we provide the user CPU time supplied by the Ubuntu function `time` for the slowest simulation variant (5000 realizations, BPA). While the simulations are run as a matter of seconds for variants with only bit or only power allocation (or, of course, without any resource allocation), the user time is 80 minutes for OPT in the correlated, 52 minutes in the uncorrelated case and on average 17 seconds both for correlated and uncorrelated case using NLI if BPA is performed at 10dB. This time includes output operations (creating a text file with data for Figures 5 or 6, respectively). Note that NLI is much faster because there are only seven possibilities to check for two active layers.

5 Conclusions

In this paper, we studied bit and power allocation for the SVD based channel separation from the verified point of view. Additionally, we compared the results to the GMD-based approach, not only from the theoretical side but also using 5000 realizations of an uncorrelated and correlated MIMO channel. Although the GMD

based approach is considered to require no resource allocation, at least asymptotically for high SNR, it is not always so at lower SNRs, where bit allocation might be profitable in isolated cases. Experimentally, we demonstrated that the GMD is definitely an alternative if no optimization of resource allocation can be carried out. Besides, it is competitive even if there is enough capacity to perform bit and power allocation for the SVD separated channel, at least, for higher SNRs, especially under good scattering conditions. However, the SVD outperforms the GMD wrt. to the quality criterion of the BER on average in bad scattering conditions and lower SNRs. As concerns the SVD based channel separation, we observe that using only bit allocation improves the BER significantly for both uncorrelated and correlated MIMO systems. The BER can be improved even further by subsequent power allocation, the adjustment for the better more visible in the uncorrelated than in the correlated case.

Some of the results suggest that employing GMD instead of SVD does not change the fact that the weakest layer should be switched off, at least, at lower SNRs. More experiments are necessary to substantiate this suggestion, which is the topic for our future work. Additionally, it is not clear beforehand for given singular values and an SNR whether the BER would be better if SVD or GMD is employed. A comprehensible criterion depending on these parameters would help to optimize MIMO systems further wrt. their BER. To study if it is possible to devise such a criterion is a further subject for our future work.

References

[1] Auer, E. and Ahrens, A. Guaranteed minimization of the bit error ratio for correlated MIMO systems. In *Proceedings of the 9th International Workshop on Reliable Engineering Computing*, pages 457–470, 2021. URL: http://ww2new.unime.it/REC2021/proceedings/REC2021_Proceedings.pdf.

[2] Auer, E. and Ahrens, A. Guaranteed minimization of the bit error ratio for MIMO systems: A mathematical viewpoint. *ASCE-ASME Journal of Risk and Uncertainty in Engineering Systems Part B: Mechanical Engineering*, 7(2), 2021. DOI: 10.1115/1.4050161.

[3] Auer, E., Benavente-Peces, C., and Ahrens, A. Solving the power allocation problem using methods with result verification. *International Journal of Reliability and Safety*, 12(1/2):86–102, 2018. DOI: 10.1504/IJRS.2018.092506.

[4] Chiani, M., Win, M. Z., and Zanella, A. On the capacity of spatially correlated MIMO Rayleigh-fading channels. *IEEE Transactions on Information Theory*, 49:2363–2371, 2003. DOI: 10.1109/TIT.2003.817437.

[5] de Figueiredo, L. H. and Stolfi, J. Affine arithmetic: Concepts and applications. *Numerical Algorithms*, 34(1–4):147–158, 2004. DOI: 10.1023/B:NUMA.0000049462.70970.b6.

[6] Ertel, R. B., Cardieri, P., Sowerby, K. W., Rappaport, T. S., and Reed, J. H. Overview of spatial channel models for antenna array communication systems. *IEEE Personal Communications*, 5(1):10–22, 1998. DOI: 10.1109/98.656151.

[7] Foschini, G. Layered space-time architecture for wireless communication in a fading environment when using multi-element antennas. *Bell Labs Technical Journal*, 1(2):41–59, 1996. DOI: 10.1002/bltj.2015.

[8] Hofschuster, W., Krämer, W., and Neher, M. C-XSC and closely related software packages. In Cuyt, A. M., Krämer, W., Luther, W., and Markstein, P. W., editors, *Numerical Validation in Current Hardware Architectures*, Volume 5492 of *Lecture Notes in Computer Science*, pages 68–102, 2008. DOI: 10.1007/978-3-642-01591-5_5.

[9] Jiang, Y., Hager, W. W., and Li, J. The generalized triangular decomposition. *Mathematics of Computation*, 77(262):1037–1056, 2008. DOI: 10.1090/S0025-5718-07-02014-5.

[10] Jiang, Y., Li, J., and Hager, W. W. MIMO transceiver design using geometric mean decomposition. In *Proceedings of the 2004 IEEE Information Theory Workshop*, pages 193–197, San Antonio, TX, USA, 2004. IEEE. DOI: 10.1109/ITW.2004.1405298.

[11] Kuo, P.-H. and Ting, P. Probabilistic behavior analysis of MIMO fading channels under geometric mean decomposition. *Journal of Electrical and Computer Engineering*, pages 1–8, 2012. DOI: 10.1155/2012/340809.

[12] Lee, J. and Leyffer, S. *Mixed Integer Nonlinear Programming*. Springer Publishing Company, Incorporated, 2011. DOI: 10.1007/978-1-4614-1927-3.

[13] Lerch, M., Tischler, G., von Gudenberg, J. W., Hofschuster, W., and Krämer, W. filib++, a fast interval library supporting containment computations. *ACM Transactions on Mathematical Software*, 32(2):299–324, 2006. DOI: 10.1145/1141885.1141893.

[14] Lohner, R. On the ubiquity of the wrapping effect in the computation of error bounds. In Kulisch, U., Lohner, R., and Facius, A., editors, *Perspectives on Enclosure Methods*, pages 201–218. Springer-Verlag, 2001. DOI: 10.1007/978-3-7091-6282-8_12.

[15] Moore, R. E., Kearfott, R. B., and Cloud, M. J. *Introduction to Interval Analysis*. Society for Industrial and Applied Mathematics, Philadelphia, 2009. DOI: 10.1137/1.9780898717716.

[16] Nedialkov, N. S. *Implementing a Rigorous ODE Solver Through Literate Programming*. In Rauh, A. and Auer, E., editors, *Modeling, Design, and Simulation of Systems with Uncertainties*, Volume 3 of *Mathematical Engineering*,

pages 3–19. Springer, Heidelberg, 2011. DOI: 10.1007/978-3-642-15956-5_1.

[17] Oestges, C. Validity of the Kronocker Model for MIMO Correlated Channels. In *Vehicular Technology Conference*, Volume 6, pages 2818–2822, Melbourne, 2006. DOI: 10.1109/VETECS.2006.1683382.

[18] Raleigh, G. G. and Cioffi, J. M. Spatio-temporal coding for wireless communication. *IEEE Transactions on Communications*, 46(3):357–366, 1998. DOI: 10.1109/26.662641.

[19] Shinde, K. *Interval uncertainty method to treat inconsistent measurements in inverse problems*. PhD thesis, Université de Technologie de Compiègne, 2021. URL: https://theses.hal.science/tel-03680994.

[20] Sklar, B. Rayleigh fading channels in mobile digital communication systems. I. Characterization. *IEEE Communications Magazine*, 35(7):90–100, 1997. DOI: 10.1109/35.601747.

[21] Telatar, E. Capacity of multi-antenna Gaussian channels. *European Transactions On Telecommunications*, 10:585–595, 1999. DOI: 10.1002/ett.4460100604.

[22] Tse, D. and Viswanath, P. *The Wireless Channel*. In *Fundamentals of Wireless Communication*, chapter 2. Cambridge University Press, 2005. DOI: 10.1017/CBO9780511807213.

[23] Weikert, O. and Zölzer, U. Efficient MIMO channel estimation with optimal training sequences. In *Proceedings of 1st Workshop on Commercial MIMO Components and Systems*, Duisburg, Germany, 2007.

[24] Zaiser, S., Buchholz, M., and Dietmayer, K. Interval system identification for MIMO ARX models of minimal order. In *53rd IEEE Conference on Decision and Control*, pages 1774–1779, 2014. DOI: 10.1109/CDC.2014.7039655.

Acta Cybernetica **26** (2024) 799–816.

Robust Control and Actuator Fault Detection Based on an Iterative LMI Approach: Application on a Quadrotor

Oussama Benzinane[ab] and Andreas Rauh[ac]

Abstract

Linear Matrix Inequalities (LMIs) have recently gained momentum due to the increasing performance of computing hardware. Many current research activities rely on the advantages of this growth in order to design controllers with provable stability and performance guarantees. To guarantee robustness despite actuator faults, model uncertainty, nonlinearities, and measurement noise, a novel iterative LMI approach is presented to design an observer-based state feedback controller allowing for simultaneous optimization of the control and observer gains. A comparison with a combination of an Extended Kalman Filter (EKF) and a Linear-Quadratic Regulator (LQR) has been conducted, inherently providing guaranteed stability for the closed loop only when the separation principle holds, which is not the case in this study. Both approaches are applied on a quadrotor, where reliable detection and compensation of the faults in the presence of measurement noise is demonstrated.

Keywords: robust control, linear matrix inequalities, interval methods, extended Kalman Filter, linear-quadratic regulator

1 Introduction

Stability, robustness, and fault tolerance are the most challenging purposes that the researchers have tackled by developing different control and estimation techniques. One of the domains of application is aeronautics, where flight control systems play a major role in ensuring the safety of drones when tracking desired trajectories. During the flight, quadrotors face many issues that originate from the inside (such as a suddenly broken rotor or a failed transmission of measurements from GPS) or from the surrounding environment (e.g., lateral wind). Unavoidably, detection and compensation of faults should take place to reduce the effects of such issues.

[a]Distributed Control in Interconnected Systems, Carl von Ossietzky Universität Oldenburg, Germany
[b]ORCID: 0000-0002-8633-1139
[c]E-mail: andreas.rauh@uni-oldenburg.de, ORCID: 0000-0002-1548-6547

DOI: 10.14232/actacyb.301392

In the literature, see [1], [7], [2] and [16], many researchers have exploited benefits of fault-tolerant methods. Particularly, some attention has been paid to the use of the LMI approach for detecting and compensating actuator faults. For instance, in the paper [12], the authors have investigated the LMI approach to reconfigure the controller parameters for a discrete-time switched system in the presence of actuator faults, unstructured uncertainties, and time delay.

Since systems in the real world are nonlinear, the use of a linear control system, tuned for a single operating point, might not ensure stability and performance. Hence, reformulating the modeled nonlinear system into a quasi-linear form, and based on this, into a polytopic representation with bounded parameter uncertainty allows for exploiting LMI approaches during the design of the control system. In [11], the authors have presented a joint optimization of the combination of control laws and filters taking into consideration bounded uncertainty and noise. In [10], the authors have developed a strategy for desensitization of the closed-loop behavior towards stochastic noise in continuous-time scenarios. Then, exploiting the ideas published in the paper [4], the present paper constitutes a contribution to the design of a discrete-time observer-based state feedback controller in the presence of both bounded parameter uncertainty and stochastic noise in order to guarantee robust performance despite actuator faults, model uncertainty, nonlinearities, and measurement noise. The proposed design procedure allows for optimizing controller and observer gains simultaneously. It consists of the following two phases: (i) placement of poles into a desired area within the complex z-plane and (ii) desensitization of the closed loop to stochastic noise.

From the literature, a huge amount of research has been conducted to exploit the Extended Kalman Filter (EKF) to accurately estimate state variables. Moreover, they are often combined with linear-quadratic regulators (LQR) c.f. [8] and [15]. Hence, for the purpose of comparison, the actuator faults are not only estimated in this paper but also using the EKF. To compensate the actuator faults and for stabilizing the system states, a combination with the LQR is further investigated.

This paper is organized as follows. In Section 2, a mathematical model is formulated by employing a first principle approach based on the Newton-Euler equations for the description of the dynamic characteristics of a quadrotor. Section 3 describes the design of the controller and observer based on a polytopic representation. The fourth section introduces the synthesis of the controller and observer gains. The fifth section presents the adapted EKF-LQR method. In Section 6, results are presented with comments before conclusions and an outlook of future work are given in Section 7.

2 Modeling of the Quadrotor

In the literature, research has been conducted to build quadrotor models that take into consideration some parts of the knowledge that humans have acquired about the aerodynamic phenomena, see e.g. [6]. Such models allow designing a corresponding controller using one or a mixture of the control methods that exist in the

literature.

According to [13], we consider the quadrotor in the earth-fixed inertial coordinate frame $(e_{1\mathrm{I}}, e_{2\mathrm{I}}, e_{3\mathrm{I}})$ and body-fixed frame $(e_{1\mathrm{B}}, e_{2\mathrm{B}}, e_{3\mathrm{B}})$ whose origin is at the center of gravity of the quadrotor as shown in Figure 1. The position of the quadrotor is described by its coordinate vector $\mathbf{p}^T = (x, y, z)$. For the rotation from the earth's inertial frame to the body frame, the ZYX convention for roll, pitch, and yaw angles (ϕ, θ, ψ) is chosen. Indeed, there are several existing conventions to describe the transformation based on the successive rotation about these three axes, see e.g. [5].

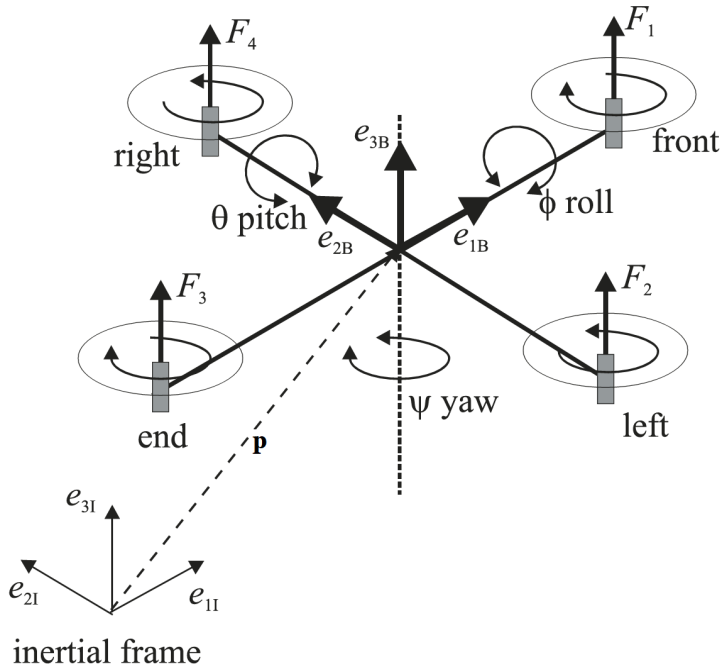


Figure 1: Earth- and body-fixed frame of the quadrotor as introduced in [13].

By applying Newton's law for the rotational and translational motions, the kinematic and dynamic expressions are derived while making the hypothesis that the quadrotor is a rigid body. Considering some assumptions such as neglecting the ground effects and ignoring gyroscopic moments, a set of nonlinear ordinary differential equations is obtained.

After rewriting the previously derived equations into a corresponding state-space form $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u})$, the model can be decomposed into two parts, as addressed in [13], thus describing the attitude dynamics and the velocity dynamics. However,

in this paper, only the first part is treated according to

$$\begin{cases} \dot{\phi} = \dot{\phi} \\ \ddot{\phi} = \dot{\theta}\dot{\psi}\frac{I_y - I_z}{I_x} - \frac{J_R}{I_x}\dot{\theta}\omega_\mathrm{d} + \frac{l}{I_x}\tau_\phi \\ \dot{\theta} = \dot{\theta} \\ \ddot{\theta} = \dot{\phi}\dot{\psi}\frac{I_z - I_x}{I_y} + \frac{J_R}{I_y}\dot{\phi}\omega_\mathrm{d} + \frac{l}{I_y}\tau_\theta \\ \dot{\psi} = \dot{\psi} \\ \ddot{\psi} = \dot{\phi}\dot{\theta}\frac{I_x - I_y}{I_z} + \frac{l}{I_z}\tau_\psi \ , \end{cases} \tag{1}$$

with the state vector $\mathbf{x} = [\phi, \dot{\phi}, \theta, \dot{\theta}, \psi, \dot{\psi}]^T \in \mathbb{R}^n$ and the input vector $\mathbf{u} = [\tau_\phi, \tau_\theta, \tau_\psi]^T \in \mathbb{R}^m$, representing respectively the roll, pitch, and yaw torque, all depending on the rotor speeds; ω_d is a fictitious disturbance that depends on the speeds of the four rotors, and J_R is the rotor inertia, while I_x, I_y, I_z are the diagonal entries of the quadrotor's inertia matrix.

Using an optimized factorization, where β_1, β_2, and $\beta_3 \in \mathbb{R}$ are free optimization variables, the quasi-linear model

$$\begin{aligned} \underbrace{\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \\ \dot{x}_5 \\ \dot{x}_6 \end{bmatrix}}_{\dot{\mathbf{x}}(t)} = & \underbrace{\begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \beta_1 I_1 x_6 & 0 & (1-\beta_1) I_1 x_4 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & \beta_2 I_2 x_6 & 0 & 0 & 0 & (1-\beta_2) I_2 x_2 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & (1-\beta_3) I_3 x_4 & 0 & \beta_3 I_3 x_2 & 0 & 0 \end{bmatrix}}_{\mathbf{A}_\mathrm{c}(\mathbf{x}(\mathbf{t}))} \cdot \underbrace{\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{bmatrix}}_{\mathbf{x}(\mathbf{t})} \\ & + \underbrace{\begin{bmatrix} 0 & 0 & 0 \\ \frac{l}{I_x} & 0 & 0 \\ 0 & 0 & 0 \\ 0 & \frac{l}{I_y} & 0 \\ 0 & 0 & 0 \\ 0 & 0 & \frac{l}{I_z} \end{bmatrix}}_{\mathbf{B}_\mathrm{c}(\mathbf{x}(\mathbf{t}))} \cdot \underbrace{\begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix}}_{\mathbf{u}(\mathbf{t})} + \underbrace{\begin{bmatrix} 0 \\ -\frac{J_R}{I_x} x_4 \\ 0 \\ \frac{J_R}{I_y} x_2 \\ 0 \\ 0 \end{bmatrix}}_{\mathbf{G}_\mathrm{c}(\mathbf{x}(\mathbf{t}))} \cdot \omega_\mathrm{d} \end{aligned} \tag{2}$$

has been obtained, where $x_1 \in [\underline{\phi}, \overline{\phi}]$, $x_3 \in [\underline{\theta}, \overline{\theta}]$, $x_5 \in [\underline{\psi}, \overline{\psi}]$, $x_2 \in [\underline{\dot{\phi}}, \overline{\dot{\phi}}]$, $x_4 \in [\underline{\dot{\theta}}, \overline{\dot{\theta}}]$, and $x_6 \in [\underline{\dot{\psi}}, \overline{\dot{\psi}}]$ are assumed to be bounded by a-priori known intervals. Moreover, u_1,u_2,u_3 are respectively the control signals $\tau_\phi, \tau_\theta, \tau_\psi$. The parameters I_1, I_2, and I_3 depend on the inertia matrix entries; $\mathbf{A}_\mathrm{c}$ is the system matrix, $\mathbf{B}_\mathrm{c}$ is the state-independent input matrix, and $\mathbf{G}_\mathrm{c}$ is the disturbance input matrix, coupling the process noise with the system dynamics.

The exploitation of the optimization variables introduced above allows the computation of an adequate system matrix of the quasi-linear realization in each iteration of the control and observer design in the following section to maximize the provable domain of attraction of the operating point (the equilibrium). For further information, where a similar approach was also used for stability analysis, the

reader is referred to [9]. The obtained model in Eq. (2) represents the exact nonlinear dynamics and is used in the simulation as shown in Figure 2 as well as in the controller and observer design phase.

3 Controller and Observer Design

In the obtained model, the appearance of nonlinearities in the system matrix leads to the fact that the separation principle of control and observer design is no longer valid as explained in the paper [10], which means that the controller and observer can influence each other's stability and, therefore, they must be designed simultaneously.

Consider the discrete-time quasi-linear state-space representation

$$\begin{cases} \mathbf{x}_{k+1} = \mathbf{A}(\mathbf{x}_k)\mathbf{x}_k + \mathbf{B}(\mathbf{x}_k)(\mathbf{u}_k + \mathbf{d}_k) + \mathbf{G}_\mathrm{p}\omega_k \\ \mathbf{y}_k = \mathbf{C}\mathbf{x}_k, \end{cases} \tag{3}$$

where $\mathbf{y}_k \in \mathbb{R}^p$, ω_k, and $\mathbf{d}_k \in \mathbb{R}^m$ are respectively the output vector, process noise vector, and actuator fault vector. The faults could manifest themselves in different manners like a blocking, saturation, or efficiency loss of the physical actuators. Finally, $\mathbf{C}$ is the output matrix.

3.1 Design for Model-Based Actuator Fault Compensation

To be able to detect actuator faults, an augmented system model

$$\begin{cases} \underbrace{\begin{bmatrix}\mathbf{x}_{k+1}\\ \mathbf{d}_{k+1}\end{bmatrix}}_{\mathbf{z}_{k+1}} = \underbrace{\begin{bmatrix}\mathbf{A}(\mathbf{x}_k) & \mathbf{B}(\mathbf{x}_k)\\ \mathbf{0}_{(m,n)} & \mathbf{I}_{(m,m)}\end{bmatrix}}_{\mathbf{A}_\mathrm{e}(\mathbf{x}_k)} \cdot \underbrace{\begin{bmatrix}\mathbf{x}_k\\ \mathbf{d}_k\end{bmatrix}}_{\mathbf{z}_k} + \underbrace{\begin{bmatrix}\mathbf{B}(\mathbf{x}_k)\\ \mathbf{0}_{(m,m)}\end{bmatrix}}_{\mathbf{B}_\mathrm{e}(\mathbf{x}_k)} \cdot \mathbf{u}_k + \underbrace{\begin{bmatrix}\mathbf{G_p}\\ \mathbf{0}_{(m,1)}\end{bmatrix}}_{\mathbf{G}_\mathrm{e}} \cdot \omega_k \\ \mathbf{y}_k = \underbrace{\begin{bmatrix}\mathbf{C} & \mathbf{0}_{(p,m)}\end{bmatrix}}_{\mathbf{C}_\mathrm{c}} \cdot \underbrace{\begin{bmatrix}\mathbf{x}_k\\ \mathbf{d}_k\end{bmatrix}}_{\mathbf{z}_k} \end{cases} \tag{4}$$

is formulated by appending a discrete-time integrator disturbance model $\mathbf{d}_{k+1} = \mathbf{d}_k$ for each of the independent faults to the original state vector.

On this basis, a linear time-invariant full-state observer is designed with the discrete-time state-space representation

$$\underbrace{\begin{bmatrix}\hat{\mathbf{x}}_{k+1}\\ \hat{\mathbf{d}}_{k+1}\end{bmatrix}}_{\hat{\mathbf{z}}_{k+1}} = \underbrace{\begin{bmatrix}\tilde{\mathbf{A}} & \tilde{\mathbf{B}}\\ \mathbf{0}_{(m,n)} & \mathbf{I}_{(m,m)}\end{bmatrix}}_{\tilde{\mathbf{A}}_\mathrm{e}} \cdot \underbrace{\begin{bmatrix}\hat{\mathbf{x}}_k\\ \hat{\mathbf{d}}_k\end{bmatrix}}_{\hat{\mathbf{z}}_k} + \underbrace{\begin{bmatrix}\tilde{\mathbf{B}}\\ \mathbf{0}_{(m,m)}\end{bmatrix}}_{\tilde{\mathbf{B}}_\mathrm{e}} \cdot \mathbf{u}_k + \underbrace{\begin{bmatrix}\mathbf{H}_\mathrm{i}\\ \mathbf{H}_\mathrm{f}\end{bmatrix}}_{\mathbf{H}_\mathrm{e}} \cdot \mathbf{C}_\mathrm{e} \cdot (\mathbf{z}_k - \hat{\mathbf{z}}_k), \tag{5}$$

where $\mathbf{H}_\mathrm{e} \in \mathbb{R}^{(n+m)\times p}$ is the constant observer gain; $\tilde{\mathbf{A}}$ and $\tilde{\mathbf{B}}$ are, respectively, the nominal dynamics and input matrices that are chosen in this current paper as the matrices of $\mathbf{A}(\mathbf{x}_k)$ and $\mathbf{B}(\mathbf{x}_k)$ evaluated for the chosen operating point which corresponds to the hovering state.

The estimated states $\hat{\mathbf{z}}_k$ are fed back by the control law

$$\mathbf{u}_k = -\underbrace{\begin{bmatrix}\mathbf{K} & \mathbf{I}_{(m,m)}\end{bmatrix}}_{\mathbf{K}_\mathrm{e}} \cdot \hat{\mathbf{z}}_k, \tag{6}$$

where the multiplication by $\mathbf{K}_\mathrm{e} \in \mathbb{R}^{m\times(m+n)}$ includes the actual state feedback by means of the gain $\mathbf{K}$ and the compensation of the actuator faults.

The architecture of the closed loop is illustrated in Figure 2. For simulation purposes, it has been implemented in Simulink with ode1 as a solver with a small fixed integration step size.

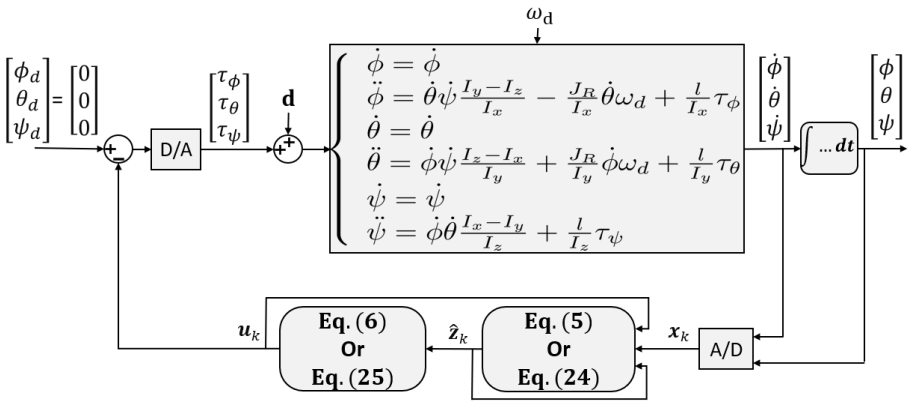


Figure 2: Structure of the linear observer-based state feedback controller

In Figure 2, the block A/D represents the digital-to-analog converter in a first-order-hold mode, and the D/A block is the corresponding analog-to-digital converter.

Considering the error $\mathbf{e}_k = \mathbf{z}_k - \hat{\mathbf{z}}_k$, an augmented closed-loop system representation has been obtained according to

$$\underbrace{\begin{bmatrix}\mathbf{z}_{k+1}\\ \mathbf{e}_{k+1}\end{bmatrix}}_{\mathbf{w}_{k+1}} = \underbrace{\begin{bmatrix}\mathbf{A}(\mathbf{x}_k) - \mathbf{B}(\mathbf{x}_k)\mathbf{K} & \mathbf{B}(\mathbf{x}_k) & \mathbf{B}(\mathbf{x}_k)\mathbf{K} & \mathbf{0}_{(n,m)}\\ \mathbf{0}_{(m,n)} & \xi\cdot\mathbf{I}_{(m,m)} & \mathbf{0}_{(m,n)} & \mathbf{0}_{(m,m)}\\ \mathcal{A}_{31} & (\tilde{\mathbf{B}} - \mathbf{B}(\mathbf{x}_k)) & \mathcal{A}_{33} & \\ \mathbf{0}_{(m,n)} & \mathbf{I}_{(m,m)} & -\mathbf{H}_\mathrm{f}\mathbf{C} & \mathbf{I}_{(m,m)}\end{bmatrix}}_{\mathcal{A}(\mathbf{x}_k)} \cdot \underbrace{\begin{bmatrix}\mathbf{z}_k\\ \mathbf{e}_k\end{bmatrix}}_{\mathbf{w}_k} + \underbrace{\begin{bmatrix}\mathbf{G}_\mathrm{e}\\ \mathbf{G}_\mathrm{e}\end{bmatrix}}_{\mathcal{G}(\mathbf{x}_k)} \cdot \omega_k, \tag{7}$$

with $\mathcal{A}_{31} = \mathbf{A}(\mathbf{x}_k) - \tilde{\mathbf{A}} - (\mathbf{B}(\mathbf{x}_k) - \tilde{\mathbf{B}})\mathbf{K}$ and $\mathcal{A}_{33} = \tilde{\mathbf{A}} - \mathbf{H}_\mathrm{i}\mathbf{C} + (\mathbf{B}(\mathbf{x}_k) - \tilde{\mathbf{B}}(\mathbf{x}_k))\mathbf{K}$.

Remark. To avoid a pure integrator of the actuator fault $\mathbf{d}_k$, a multiplication of the identity matrix by a number $0 < \xi < 1$ is performed during the synthesis stage so that the corresponding mode becomes stabilizable. This parameter choice corresponds to

$$\xi = \begin{cases} 1 & : \text{used for modeling and simulation,} \\ < 1 & : \text{used during synthesis.} \end{cases} \tag{8}$$

Note that this modification only influences the convergence of the following iterative LMI-based control design but leaves the eigenvalues of the closed-loop system model, represented conservatively in a polytopic form, unchanged.

3.2 Simplification of the Augmented System Model for a Control Without Actuator Fault Detection

For the purpose of comparison with the synthesis in which the actuator fault detection is not taken into consideration, the fault $\mathbf{d}_k$ is not estimated and hence not included in the vector $\mathbf{z}_k$ but instead appended to the disturbance vector ω_k. This modification leads to the augmented model

$$\underbrace{\begin{bmatrix} \mathbf{x}_{k+1} \\ \mathbf{e}_{k+1} \end{bmatrix}}_{\mathbf{w}_{k+1}} = \underbrace{\begin{pmatrix} \mathbf{A}(\mathbf{x}_k) - \mathbf{B}(\mathbf{x}_k)\mathbf{K} & \mathbf{B}(\mathbf{x}_k)\mathbf{K} \\ \mathbf{A}(\mathbf{x}_k) - \tilde{\mathbf{A}} - (\mathbf{B}(\mathbf{x}_k) - \tilde{\mathbf{B}})\mathbf{K} & \tilde{\mathbf{A}} - \mathbf{H}_\mathrm{i}\mathbf{C} + (\mathbf{B}(\mathbf{x}_k) - \tilde{\mathbf{B}})\mathbf{K} \end{pmatrix}}_{\mathcal{A}(\mathbf{x}_k)} \cdot \underbrace{\begin{bmatrix} \mathbf{x}_k \\ \mathbf{e}_k \end{bmatrix}}_{\mathbf{w}_k}$$
$$+ \underbrace{\begin{pmatrix} \mathbf{B}(\mathbf{x}_k) & \mathbf{G}_\mathrm{p} \\ \mathbf{B}(\mathbf{x}_k) & \mathbf{G}_\mathrm{p} \end{pmatrix}}_{\mathcal{G}(\mathbf{x}_k)} \cdot \underbrace{\begin{bmatrix} \mathbf{d}_k \\ \omega_k \end{bmatrix}}_{\omega_\mathrm{p}}, \tag{9}$$

which replaces the use of Eq. (7), when required in the following synthesis.

For the same reason, and also during simulation, the fault $\mathbf{d}_k$ is then also removed from $\hat{\mathbf{z}}_k$. Therefore, the state observer that replaces Eq. (5), turns into

$$\hat{\mathbf{x}}_{k+1} = \tilde{\mathbf{A}} \cdot \hat{\mathbf{x}}_k + \tilde{\mathbf{B}} \cdot \mathbf{u}_k + \mathbf{H}_\mathrm{i}\mathbf{C} \cdot (\mathbf{x}_k - \hat{\mathbf{x}}_k), \tag{10}$$

with the simplified control law

$$\mathbf{u}_k = -\mathbf{K} \cdot \hat{\mathbf{x}}_k \tag{11}$$

that replaces Eq. (6).

3.3 Polytopic Uncertainty Representation of the Augmented Closed-Loop System

For control and observer design, the state variables are assumed to be constrained. By chosen limits, the entries of the system matrix and the disturbance input matrix

are also bounded, so that a polytopic domain can be determined to enclose the matrices $\mathcal{A}(\mathbf{x}_k)$ and $\mathcal{G}(\mathbf{x}_k)$ which belong to the convex combination of n_v independent extremal vertex matrices $\mathcal{A}_v$ and $\mathcal{G}_v$ according to

$$[\mathcal{A}(\mathbf{x}_k), \mathcal{G}(\mathbf{x}_k)] \in \left\{ [\mathcal{A}(\zeta), \mathcal{G}(\zeta)] = \sum_{v=1}^{n_v} \zeta_v \cdot [\mathcal{A}_v, \mathcal{G}_v]; \sum_{v=1}^{n_v} \zeta_v = 1; \zeta_v \geq 0 \right\}. \tag{12}$$

Here, ζ_v are the scheduling variables.

During this formulation, identical dependencies in the matrix entries should be identified to reduce the conservativeness of the polytopic model as far as possible.

4 Optimization of the Control and Observer Gains

In the paper [4], the authors have developed an iterative LMI tool that is briefly explained in the following sub-sections. It serves as the basis for the optimization of the gains of the fault-tolerant control structure developed in this paper.

4.1 Eigenvalue Domain Assignment

To realize certain closed-loop characteristics (such as the settling time or maximum overshoot) in combination with robustness against the uncertainty represented by the previous polytopic model, an eigenvalue domain assignment is performed for all extremal system matrices introduced in Eq. (12) with the help of a common Lyapunov function candidate.

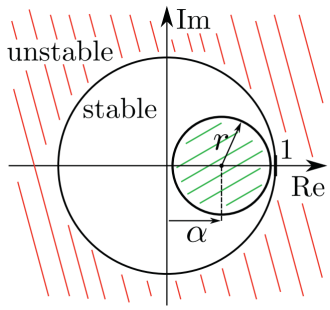


Figure 3: Desired stability domain in the interior of the unit circle of the complex z-plane.

The parameterized sub-region inside the unit circle of the z-plane with radius r and midpoint α, as illustrated in Figure 3, can be expressed according to [4] by the LMI condition

$$\begin{bmatrix} \mathbf{L} & (\mathcal{A}_v - \alpha\mathbf{I}) \\ (\mathcal{A}_v - \alpha\mathbf{I})^T & r^2\mathbf{P} \end{bmatrix} \succ 0, v = 1, ..., n_v. \tag{13}$$

Here, the positive-definite matrix $\mathbf{P} = \mathbf{P}^T$ is a free decision variable and parameterizes a common Lyapunov function for all realizations of the polytopic model. Its inverse, is expressed in terms of $\mathbf{P}^{-1} \succeq 2\hat{\mathbf{P}}^{-1} - \hat{\mathbf{P}}^{-1}\mathbf{P}\hat{\mathbf{P}}^{-1} =: \mathbf{L}$. This approximation has been obtained by a linearization approach using a first-order Neumann series, cf. [3].

In the absence of stochastic noise, a successful solution of Eq. (13) ensures asymptotic stability of the observer-based closed-loop control structure. This solution can be obtained by means of the first algorithm in [4], in which the matrix $\mathbf{P}$ is updated in each iteration step l. Ending with admissible values for r_{end} (typically a predefined value) and α, leading to stability domains in the interior of the unit circle, a preliminary controller and observer design is obtained.

4.2 Desensitization Towards Noise

In the presence of stochastic noise, the solution of Sec. 4.1 is improved in the sense of desensitization towards noise. For that purpose, the discrete-time version of the Itô differential operator has been used to expand the Lyapunov conditions and to express the size of the domain around the equilibrium for which stability cannot be proven due to noise excitation. After some mathematical reformulation, an iteration rule for the optimization task has been derived in [4] that uses a cost function subject to LMI constraints in the form

$$\min J = \sum_{v=1}^{n_v} \frac{\text{trace}\{\mathbf{N}\}}{\det(-\hat{\mathbf{M}}_v)}, \tag{14}$$

with

$$\mathbf{P} \succ 0, \tag{15}$$

$$\mathbf{N} \succ 0, \tag{16}$$

$$\begin{bmatrix} \mathbf{L} & \mathcal{G}_v \\ \mathcal{G}_v^T & \mathbf{N} \end{bmatrix} \succ 0, v = 1, ..., n_v, \tag{17}$$

$$\begin{bmatrix} \mathbf{L} & (\mathcal{A}_v - \alpha\mathbf{I}) \\ (\mathcal{A}_v - \alpha\mathbf{I})^T & r^2\mathbf{P} \end{bmatrix} \succ 0, v = 1, ..., n_v, \tag{18}$$

and $\hat{\mathbf{M}}_v = \hat{\mathcal{A}}_v^T \hat{\mathbf{P}} \hat{\mathcal{A}}_v - \hat{\mathbf{P}}$, where the optimization variables β_1, β_2, and β_3 introduced in Eq. (2) are included as further decision variables. The free matrix variable $\mathbf{N}$ is automatically determined by means of an LMI solver so that $\mathbf{N} \succeq \mathcal{G}_v^T \mathbf{P} \mathcal{G}_v$ holds for all $v = 1, ..., n_v$.

In all expressions in this subsection, the symbol $\hat{(\cdot)}$ means the updated variable from the previous iteration.

Note that a successful minimization of the cost function in Eq. (14) leads to a reduction of sensitivity of the closed-loop system against noise, as it has been originally derived in [10], [11], and [4].

4.3 Summary of the Design Procedure

The procedure to find suitable controller and observer gains and to optimize their numerical values according to the desensitization criteria described in the previous subsection is summarized in the following Nassi-Shneiderman diagram.

Construction of the nonlinear system model Eq. (1)
Derive the quasi-linear model according to Eq. (2)
Discretization using the explicit Euler method
Build the polytopic domain of Eq. (12) for the augmented closed-loop model in Eq. (7)
Initialization with $r = 2$ and the step size $\Delta r = 0.2$ and $\hat{\mathbf{P}}_0^{-1} = \mathbf{I}$, Here: use a fixed value $\alpha = 0$
while $r > r_{\text{end}}$
Decrease $r := r - \Delta r$
Generate Eqs. (2), (7)
Compute the gains $\mathbf{K}$, $\mathbf{H}_{\text{i}}$ and $\mathbf{H}_{\text{f}}$, and the matrix $\mathbf{P}$ by solving the LMI in Eq. (13)
Checking the feasibility
Feasible: Update $\hat{\mathbf{P}}$ and store the matrices
Infeasible: Re-increase $r := r + \Delta r$ and reduce Δr by half
Initialization with the most recent successfully computed matrices obtained from the previous part of the algorithm
Keep r_{end} and $\alpha = 0$ fixed and initialize $\hat{\mathbf{M}}_v = \mathbf{I}$
while $\text{trace}(\mathbf{N}_l) - \text{trace}(\mathbf{N}_{l-1}) > 10^{-7}$
Generate Eqs. (2), (7)
Optimize the cost function in Eq. (14) subject to Eqs. (15), (16), (17), (18)
Update $\hat{\mathbf{P}}^{-1}$ and $\hat{\mathbf{M}}_v$ if an admissible solution has been found, and store the matrices
Use the final controller and observer gains in the closed-loop structure shown in Figure 2

Remark. In this work, the construction of the polytopic representation as seen in the diagram above is made only by a choice of intervals for the state variables. Uncertainty of the system parameters (e.g. the inertia) can be included analogously, leading to an increase in the number of vertices.

5 Alternative Control Parametrization: Extended Kalman Filter-Based Linear Quadratic Regulator Design

In order to analyze the efficiency of the iterative LMI-based method, an LQR approach is implemented additionally that uses states and actuator fault estimates obtained from an EKF as a stochastic filter approach applicable to nonlinear systems. Such an approach does not prove stability in contrast to the iterative LMI-based approach.

5.1 The Extended Kalman Filter

An extension of Eq. (3) with additive Gaussian system and measurement noise $\mathbf{w}_k$ and $\mathbf{v}_k$ is given as

$$\begin{cases} \mathbf{x}_{k+1} = \mathbf{A}(\mathbf{x}_k)\mathbf{x}_k + \mathbf{B}(\mathbf{x}_k)(\mathbf{u}_k + \mathbf{d}_k) + \mathbf{G}_\mathrm{p}\omega_k + \mathbf{W}\mathbf{w}_k \\ \mathbf{y}_k = \mathbf{C}\mathbf{x}_k + \mathbf{v}_k \\ \mathbf{d}_{k+1} = \mathbf{d}_k + \mathbf{E}\mathbf{g}_k, \end{cases} \tag{19}$$

with their expected mean values and covariance matrices $\mu_{\mathbf{w},k} = 0$, $\mathbb{C}_\mathbf{w}$, $\mu_{\mathbf{v},k} = 0$ and $\mathbb{C}_\mathbf{v}$. Here, $\mathbf{W}$ is the additive disturbance input matrix.

To be able to detect actuator faults, an augmented system model

$$\begin{cases} \underbrace{\begin{bmatrix}\mathbf{x}_{k+1}\\ \mathbf{d}_{k+1}\end{bmatrix}}_{\mathbf{z}_{k+1}} = \underbrace{\begin{bmatrix}\mathbf{A}(\mathbf{x}_k) & \mathbf{B}(\mathbf{x}_k)\\ \mathbf{0}_{(m,n)} & \mathbf{I}_{(m,m)}\end{bmatrix}}_{\mathbf{A}_\mathrm{e}(\mathbf{x}_k)} \cdot \underbrace{\begin{bmatrix}\mathbf{x}_k\\ \mathbf{d}_k\end{bmatrix}}_{\mathbf{z}_k} + \underbrace{\begin{bmatrix}\mathbf{B}(\mathbf{x}_k)\\ \mathbf{0}_{(m,m)}\end{bmatrix}}_{\mathbf{B}_\mathrm{e}(\mathbf{x}_k)} \cdot \mathbf{u}_k \\ \qquad + \underbrace{\begin{bmatrix}\mathbf{G}_\mathrm{p} & \mathbf{W} & \mathbf{0}_{(n,m)}\\ \mathbf{0}_{(m,1)} & \mathbf{0}_{(m,n)} & \mathbf{E}\end{bmatrix}}_{\mathbf{G}_\mathrm{d}} \cdot \begin{bmatrix}\omega_k\\ \mathbf{w}_k\\ \mathbf{g}_k\end{bmatrix} \\ \mathbf{y}_k = \underbrace{\begin{bmatrix}\mathbf{C} & \mathbf{0}_{(p,m)}\end{bmatrix}}_{\mathbf{C}_\mathrm{e}} \cdot \underbrace{\begin{bmatrix}\mathbf{x}_k\\ \mathbf{d}_k\end{bmatrix}}_{\mathbf{z}_k} \end{cases} \tag{20}$$

is formulated by appending discrete-time integrator disturbance models to the original state vector, where $\mathbf{E}$ is the input matrix of the additive actuator faults and $\mathbf{g}_k$ is a noise term representing their dynamics by an additive Gaussian noise process; $\mathbb{C}_\mathbf{g}$ is its covariance matrix.

The EKF approach consists of two parts. In the prediction part, the prior mean and covariance are computed according to

$$\underbrace{\begin{bmatrix}\mu^\mathrm{p}_{\mathbf{x},k}\\ \mu^\mathrm{p}_{\mathbf{d},k}\end{bmatrix}}_{\mu^\mathrm{p}_{\mathbf{z},k}} = \begin{bmatrix}\mathbf{A}(\mu^\mathrm{e}_{\mathbf{x},k-1}) & \mathbf{0}_{(n,m)}\\ \mathbf{0}_{(m,n)} & \mathbf{I}_{(m,m)}\end{bmatrix} \cdot \underbrace{\begin{bmatrix}\mu^\mathrm{e}_{\mathbf{x},k-1}\\ \mu^\mathrm{e}_{\mathbf{d},k-1}\end{bmatrix}}_{\mu^\mathrm{e}_{\mathbf{z},k-1}} + \begin{bmatrix}\mathbf{B}(\mu^\mathrm{e}_{\mathbf{x},k-1})\\ \mathbf{0}_{(m,m)}\end{bmatrix} \cdot (\mathbf{u}_k + \mu^\mathrm{e}_{\mathbf{d},k-1}), \tag{21}$$

and

$$\mathbb{C}^{\mathrm{p}}_{\mathbf{z},k} = \underbrace{\begin{bmatrix} \tilde{\mathbf{A}} & \tilde{\mathbf{B}} \\ \mathbf{0}_{(m,n)} & \mathbf{I}_{(m,m)} \end{bmatrix}}_{\tilde{\mathbf{A}}_{\mathrm{e}}} \cdot \mathbb{C}^{\mathrm{e}}_{\mathbf{z},k-1} \cdot \underbrace{\begin{bmatrix} \tilde{\mathbf{A}} & \tilde{\mathbf{B}} \\ \mathbf{0}_{(m,n)} & \mathbf{I}_{(m,m)} \end{bmatrix}^T}_{\tilde{\mathbf{A}}_{\mathrm{e}}^T} + \mathbf{G}_{\mathrm{d}} \cdot \begin{bmatrix} \mathbb{C}_{\omega} & \mathbf{0}_{(1,n)} & \mathbf{0}_{(1,m)} \\ \mathbf{0}_{(n,1)} & \mathbb{C}_{\mathbf{w}} & \mathbf{0}_{(n,m)} \\ \mathbf{0}_{(m,1)} & \mathbf{0}_{(m,n)} & \mathbb{C}_{\mathbf{g}} \end{bmatrix} \cdot \mathbf{G}_{\mathrm{d}}^T. \tag{22}$$

In the equation above, $\tilde{\mathbf{A}}$ and $\tilde{\mathbf{B}}$ are, respectively, the nominal dynamics and input matrices that are chosen in this current paper as the matrices of $\mathbf{A}(\mathbf{x}_k)$ and $\mathbf{B}(\mathbf{x}_k)$ evaluated for the chosen operating point which corresponds to the hovering state. Here, $\mathbb{C}_{\omega}$ is the variance of the process noise that reflects the influence of the rotor speed-dependent nonlinearity in the system model. The superscripts $(.)^{\mathrm{p}}$ and $(.)^{\mathrm{e}}$, respectively, denote the computed prior and posterior values with respect to the current measurement.

The second part is the update of the predicted mean and covariance in the innovation stage which starts with the computations of the Kalman gain

$$\mathbf{L}_k = \mathbb{C}^{\mathrm{p}}_{\mathbf{z},k} \cdot \mathbf{C}_{\mathrm{e}}^T \cdot (\mathbf{C}_{\mathrm{e}} \cdot \mathbb{C}^{\mathrm{p}}_{\mathbf{z},k} \cdot \mathbf{C}_{\mathrm{e}}^T + \mathbb{C}_{\mathbf{v}})^{-1}. \tag{23}$$

Then, estimates for the states and actuator faults are obtained with the following equation

$$\mu^{\mathrm{e}}_{\mathbf{z},k} = \mu^{\mathrm{p}}_{\mathbf{z},k} + \mathbf{L}_k \cdot (\mathbf{C}\mathbf{x}_k - \mathbf{C}_{\mathrm{e}}\mu^{\mathrm{p}}_{\mathbf{z},k}), \tag{24}$$

besides its corresponding covariance

$$\mathbb{C}^{\mathrm{e}}_{\mathbf{z},k} = (\mathbf{I}_{(n+m,n+m)} - \mathbf{L}_k\mathbf{C}_{\mathrm{e}}) \cdot \mathbb{C}^{\mathrm{p}}_{\mathbf{z},k}, \tag{25}$$

which are both fed back to the prediction part to be used for the next step.

5.2 The Linear Quadratic Regulator

For the control implementation, the mean value of the state estimates obtained in Eq. (24) is fed back by using the control law

$$\mathbf{u}_k = -\underbrace{\begin{bmatrix} \mathbf{K} & \mathbf{I}_{(m,m)} \end{bmatrix}}_{\mathbf{K}_{\mathrm{e}}} \cdot \mu^{\mathrm{e}}_{\mathbf{z},k}, \tag{26}$$

where the controller gain

$$\mathbf{K} = (\tilde{\mathbf{B}}^T\mathbf{S}\tilde{\mathbf{B}} + \mathbf{R})^{-1} \cdot (\tilde{\mathbf{B}}^T\mathbf{S}\tilde{\mathbf{A}}) \tag{27}$$

depends on the matrix $\mathbf{S}$ obtained by solving the algebraic Riccati equation

$$\tilde{\mathbf{A}}^T\mathbf{S}\tilde{\mathbf{A}} - \mathbf{S} - (\tilde{\mathbf{A}}^T\mathbf{S}\tilde{\mathbf{B}}) \cdot (\tilde{\mathbf{B}}^T\mathbf{S}\tilde{\mathbf{B}} + \mathbf{R})^{-1} \cdot (\tilde{\mathbf{B}}^T\mathbf{S}\tilde{\mathbf{A}}) + \mathbf{Q} = \mathbf{0}_{(n,n)}. \tag{28}$$

The positive-definite symmetric matrices $\mathbf{R}$ and $\mathbf{Q}$ are the weighting matrices chosen by a trial-and-error approach for the cost function

$$J(\mathbf{u}_k) = \sum_{k=1}^{\infty} (\mathbf{x}_k^T \mathbf{Q} \mathbf{x}_k + \mathbf{u}_k^T \mathbf{R} \mathbf{u}_k) \tag{29}$$

to be minimized.

6 Simulation Results

This section presents the simulation results of the quadrotor model as shown in Figure 2 with the nominal matrices corresponding to the hovering state. The sample time for discretization is set to 1 ms. The initial states are set to $\pi/4$ rad for each angle. As a control goal, the desired hovering state $[\phi_\mathrm{d}, \theta_\mathrm{d}, \psi_\mathrm{d}]^T = [0, 0, 0]^T$ shall be reached.

For the LMI-based approach, the eigenvalue locations are determined by the predefined sub-circle that is chosen with α as the origin of the z-plane and $r_\mathrm{end} = 0.999$. For the LQR, the weighting matrices are chosen as $\mathbf{R} = \mathrm{diag}([2, 2, 2])$ and $\mathbf{Q} = \mathrm{diag}([50, 10, 50, 10, 50, 10])$.

The values of the parameters that were used are listed in Table 1.

Table 1: Values of the parameters of the quadrotor model.

Parameter	Values
l [m]	0.2
J_R [kg·m^2]	$3.36{\cdot}10^{-5}$
$I_x{=}I_y$ [kg·m^2]	$4.85{\cdot}10^{-3}$
I_z [kg·m^2]	$8.81{\cdot}10^{-3}$
m [kg]	0.5

6.1 Simulation Without Output Noise and Without Detection of Actuator Faults

In the first stage, the simulation was made without taking into consideration any disturbances $\omega_k = 0$. Moreover, also the detection of the actuator faults was deactivated. After 3 seconds, an actuator fault occurs, which leads to a deviation in the yaw torque τ_ψ of -12 Nm and in the pitch torque τ_ϕ with -9 Nm.

Figure 4 shows a convergence to the steady hovering state after the initial deviation within the first 3 s. At $t = 3$ s, the occurrence of the actuator fault causes a deviation with an oscillatory behavior of the roll and pitch angles before ending in a non-desired attitude and orientation. The same figure shows the temporal evolution of the roll, pitch, and yaw torques. After the time of 3 s, they show a

strong oscillation before converging to zero, however, without bringing the system to the desired goal.

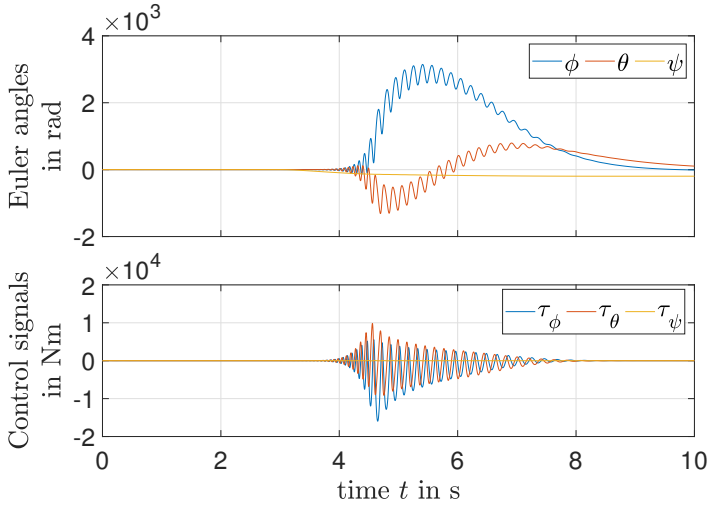


Figure 4: Regulation of the quadrotor's attitude without actuator fault detection

6.2 Simulation With Output Noise and With Compensation of Actuator Faults

To evaluate the robustness against the disturbances, a Gaussian distributed random process noise was taken into consideration with the standard deviation matrix $\mathbf{G}_{\mathrm{e}}$ in Eq. (4), together with a Gaussian distributed measurement noise with a mean of 0 and a standard deviation of $\frac{\pi}{1200}$ rad. The same actuator fault magnitude as in Sec. 6.1 is applied in the current section.

The bounds chosen for the angular velocity states specified in rad/s are $\dot{\theta} \in [-8\pi, 8\pi]$, $\dot{\phi} \in [-8\pi, 8\pi]$, $\dot{\psi} \in [-8\pi, 8\pi]$.

Figure 5 shows a convergence in a short time for the Euler angles even with the stochastic noise in response to the initial states. The deviations of the angular velocities caused by the actuator fault remain within the predefined bounds as seen in Figure 6. The associated control signals are shown in the same Figure.

6.3 Simulation with LQR based on an EKF

With the same noise used for the simulation of the iterative LMI-based method, and choosing $\mathbb{C}_{\omega} = 5$, $\mathbb{C}_{\mathbf{g}} = \mathrm{diag}([10, 10, 10])$, $\mathbb{C}_{\mathbf{v}} = \mathrm{diag}([4, 4, 4])$, and $\mathbb{C}_{\mathbf{w}} = \mathrm{diag}([2, 2, 2, 2, 2, 2])$ besides initializing the filter with $\mathbb{C}_{\mathbf{d},0} = \mathrm{diag}([10, 10, 10])$ and

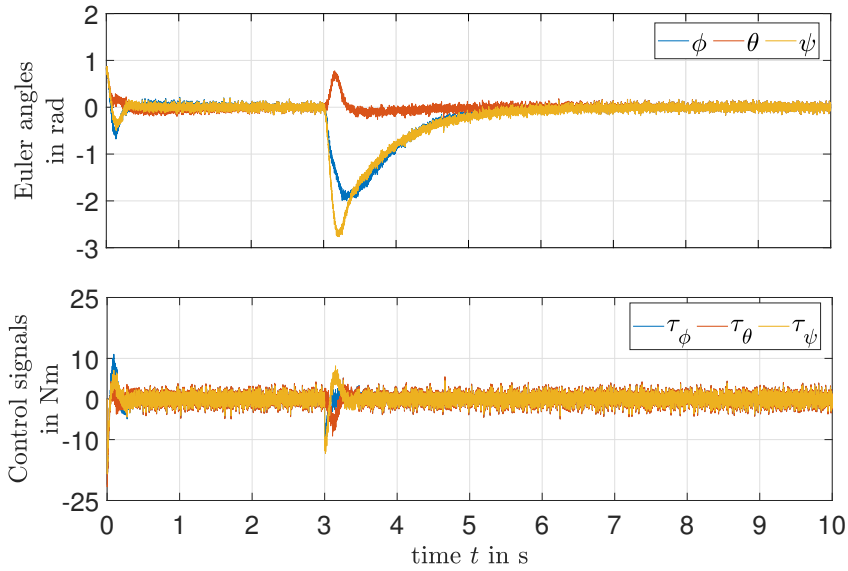


Figure 5: Regulation of the quadrotor's attitude in the presence of Gaussian output noise and actuator faults.

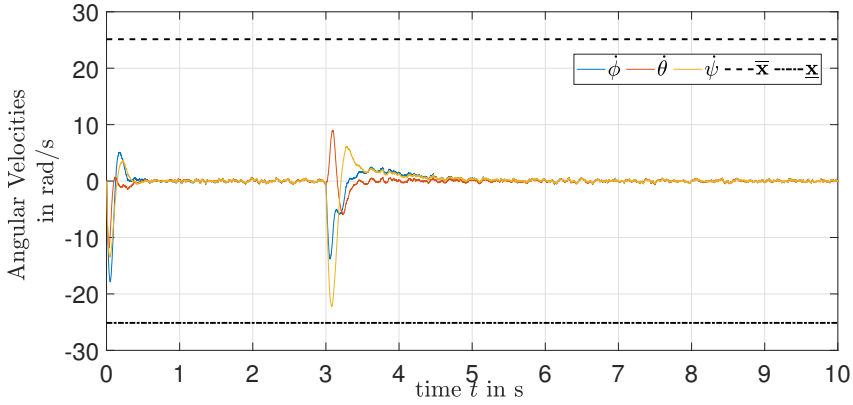


Figure 6: Response of the quadrotor's angular velocities in the presence of Gaussian output noise and actuator faults

$\mathbb{C}_{\mathbf{x},0} = \text{diag}([80, 50, 80, 50, 80, 50])$, a simulation of the attitude response and the corresponding control signals is obtained.

Figure 7 depicts the convergence of the Euler angles in a smooth way in response

to the initial states. After the actuator fault at 3s, the state variables converge within 2s but with a larger amplitude deviation than obtained for the robust LMI solution. The same Figure shows the corresponding control signals.

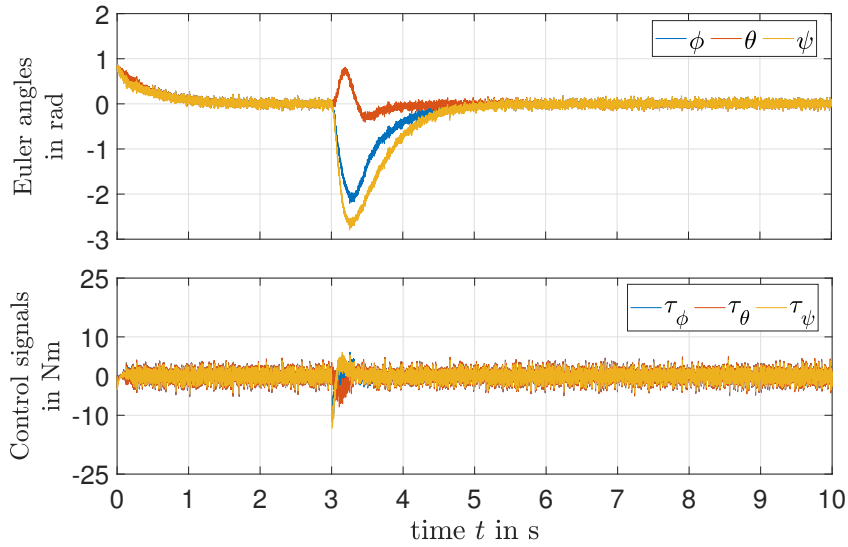


Figure 7: Regulation of the quadrotor's attitude in the presence of Gaussian output noise and actuator faults with EKF-based LQR method

In summary, the response of the LMI-based method takes a shorter time to converge compared with the EKF-based LQR. Additionally, dealing with the actuator fault was better. Due to the fact that the filter gain needs to be recomputed in the update step of the EKF-based LQR, in contrast to time-invariant gains in the LMI-based method, the novel approach helps to significantly reduce the computational effort despite its inherent proof of stability over the operating domain chosen for the parameterization of the polytopic uncertainty model.

7 Conclusions and Outlook on Future Work

The computation of controller and observer gains taking into consideration actuator faults, process noise, and nonlinearities has been possible thanks to the developed iterative LMI approach. In addition to the guaranteed stability, satisfactory time domain behavior has been achieved by a choice of the location of the eigenvalues within the z-plane. The resulting time-domain performance outperformed the one obtained with the EKF-based LQR method which in addition does not inherit a proof of stability for nonlinear models and includes much more parameters to tune.

Although the computation for the iterative LMI-based approach is made offline, attention should be paid to the computational effort. It may turn into an issue if the order of the closed loop and the number of uncertain parameters become larger. To overcome such an obstacle, the following paths can be investigated in future work: On the one hand, the pure polytopic uncertainty model can be replaced by a norm-bounded one. On the other hand, to reduce the conservativeness of the realizations, Chebyshev points can be determined for determining tighter enclosures of nonlinear dependencies. Fundamental work in this direction is published in [14]. Finally, research on using a flatness-based approach for the computation of a feedforward control together with a feedback linearization of the nonlinear plant with nominal parameters is promising to reduce the width of the polytopic domain that needs to be stabilized by a robust feedback controller.

References

[1] Blanke, M., Kinnaert, M., Lunze, J., and Staroswiecki, M., editors. *Diagnosis and fault-tolerant control*. Springer, Berlin; New York, 2nd edition, 2006. DOI: 10.1007/978-3-540-35653-0.

[2] Chan Shi, J. and Dwi, P. Fault detection and identification in Quadrotor system (Quadrotor robot). In *2016 IEEE International Conference on Automatic Control and Intelligent Systems (I2CACIS)*, pages 11–16, Selangor, Malaysia, 2016. IEEE. DOI: 10.1109/I2CACIS.2016.7885281.

[3] Dehnert, R. *Entwurf robuster Regler mit Ausgangsrückführung für zeitdiskrete Mehrgrößensysteme*. Business, Economics, and Law. Springer Fachmedien, Wiesbaden, 2020. DOI: 10.1007/978-3-658-29900-2.

[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*, 2:786152, 2022. DOI: 10.3389/fcteg.2021.786152.

[5] Diebel, J. Representing attitude: Euler angles, unit quaternions, and rotation vectors. *Matrix*, 58(15–16):1–35, 2006. URL: https://www.astro.rug.nl/software/kapteyn-beta/_downloads/attitude.pdf, accessed on June 20, 2023.

[6] Hoffmann, G., Huang, H., Waslander, S., and Tomlin, C. Quadrotor helicopter flight dynamics and control: Theory and experiment. In *AIAA Guidance, Navigation and Control Conference and Exhibit*, Hilton Head, South Carolina, 2007. American Institute of Aeronautics and Astronautics. DOI: 10.2514/6.2007-6461.

[7] Patton, R. and Klinkhieo, S. Actuator fault estimation and compensation based on an augmented state observer approach. In *Proceedings of the 48h*

IEEE Conference on Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference, pages 8482–8487, Shanghai, China, 2009. IEEE. DOI: 10.1109/CDC.2009.5399548.

[8] Raja, M. M. Extended Kalman Filter and LQR controller design for quadrotor UAVs. Master's thesis, Wright State University, 2017. URL: https://corescholar.libraries.wright.edu/etd_all/1761, accessed on June 20, 2023.

[9] Rauh, A., Bourgois, A., and Jaulin, L. Verifying provable stability domains for discrete-time systems using ellipsoidal state enclosures. *Acta Cybernetica*, pages 267–291, 2022. DOI: 10.14232/actacyb.293871.

[10] Rauh, A., Dehnert, R., Romig, S., Lerch, S., and Tibken, B. Iterative solution of linear matrix inequalities for the combined control and observer design of systems with polytopic parameter uncertainty and stochastic noise. *Algorithms*, 14(7):205, 2021. DOI: 10.3390/a14070205.

[11] Rauh, A. and Romig, S. Linear matrix inequalities for an iterative solution of robust output feedback control of systems with bounded and stochastic uncertainty. *Sensors*, 21(9):3285, 2021. DOI: 10.3390/s21093285.

[12] Telbissi, K. and Benzaouia, A. Robust fault tolerant control for uncertain switched systems with time delay. *Journal of Control, Automation and Electrical Systems*, 2023. DOI: 10.1007/s40313-022-00983-2.

[13] Voos, H. Nonlinear state-dependent Riccati equation control of a quadrotor UAV. In *2006 IEEE Conference on Computer Aided Control System Design, 2006 IEEE International Conference on Control Applications, 2006 IEEE International Symposium on Intelligent Control*, pages 2547–2552, Munich, Germany, 2006. IEEE. DOI: 10.1109/CACSD-CCA-ISIC.2006.4777039.

[14] Warthenpfuhl, S. A. and Tibken, B. Guaranteed bounds for robust LMI problems with polynomial parameter dependence. *IFAC Proceedings Volumes*, 41(2):10057–10062, 2008. DOI: 10.3182/20080706-5-KR-1001.01702.

[15] Xu, Y., Yuan, X., and Zhu, J. EKF-LQR-based cooperative optimal control of IPMSM. In *Proceedings of the 9th International Conference on Computer and Automation Engineering*, pages 307–312, Sydney Australia, 2017. ACM. DOI: 10.1145/3057039.3057075.

[16] Yu, X.-H. Actuator fault compensation for a helicopter model. In *Proceedings of 2003 IEEE Conference on Control Applications*, Volume 2, pages 1372–1374, Istanbul, Turkey, 2003. IEEE. DOI: 10.1109/CCA.2003.1223212.

Acta Cybernetica **26** (2024) 817–838.

A New Interval Arithmetic To Generate the Complementary of Contractors

Pierre Filiol[ab], Theotime Bollengier[ac], Luc Jaulin[ad], and Jean-Christophe Le Lann[ae]

Abstract

Contractor algebra is used to characterize a set defined as a composition of sets defined by inequalities. It mainly uses interval methods combined with constraint propagation. This algebra includes the classical operations we have for sets such as the intersection, the union and the inversion. Now, it does not include the complement operator. The reason for this is probably related to the interval arithmetic itself. In this paper, we show that if we change the arithmetic used for intervals adding a single flag, similar to *not a number*, we are able to include easily the complement in the algebra of contractors.

Keywords: intervals, contractors, complement, Not a Number

1 Introduction

Interval analysis [11] is a numerical tool used to solve nonlinear problems such as non convex optimization [7] or solving nonlinear equations [13]. In control or robotics, it is often needed to compute inner and outer approximations for sets [9] [16].

The algorithms we use to characterize a set $\mathbb{X}$ are pavers that classify areas of the search space using contractors [4]. A *contractor* $\mathcal{C}$ for the set $\mathbb{X} \subset \mathbb{R}^n$ is an operator $\mathbb{IR}^n \mapsto \mathbb{IR}^n$ which satisfies

$$\begin{array}{ll} \mathcal{C}([\mathbf{x}]) \subset [\mathbf{x}] & \text{(contractance)} \\ [\mathbf{x}] \subset [\mathbf{y}] \;\Rightarrow\; \mathcal{C}([\mathbf{x}]) \subset \mathcal{C}([\mathbf{y}]) & \text{(monotonicity)} \\ \mathcal{C}([\mathbf{x}]) \cap \mathbb{X} = [\mathbf{x}] \cap \mathbb{X} & \text{(consistency)} \end{array} , \tag{1}$$

where $\mathbb{IR}^n$ is the set of axis-aligned boxes of $\mathbb{R}^n$. The paver bisects boxes and uses $\mathcal{C}$ to eliminate parts of the search space that are outside $\mathbb{X}$.In this paper, sets $\mathbb{X}$ of $\mathbb{R}^n$ will be represented in *mathbb* font and intervals $[x]$ or boxes $[\mathbf{x}]$ within brackets.

[a]ENSTA Bretagne, Brest, France
[b]E-mail: pierre.filiol@netc.fr, ORCID: 0009-0009-6162-9578
[c]E-mail: theotime.bollengier@ensta-bretagne.org, ORCID: 0009-0006-7315-2736
[d]E-mail: lucjaulin@gmail.com, ORCID: 0000-0002-0938-0615
[e]E-mail: jean-christophe.le_lann@ensta-bretagne.fr, ORCID: 0000-0003-2555-1805

DOI: 10.14232/actacyb.300840

If $\mathcal{C}_1$ and $\mathcal{C}_2$ are two contractors, we define the following operations on contractors:

$$(\mathcal{C}_1 \cap \mathcal{C}_2)([\mathbf{x}]) = \mathcal{C}_1([\mathbf{x}]) \cap \mathcal{C}_2([\mathbf{x}]), \tag{2}$$
$$(\mathcal{C}_1 \cup \mathcal{C}_2)([\mathbf{x}]) = \mathcal{C}_1([\mathbf{x}]) \sqcup \mathcal{C}_2([\mathbf{x}]), \tag{3}$$

where $[\mathbf{a}] \sqcup [\mathbf{b}]$ is the smallest box which contains both $[\mathbf{a}]$ and $[\mathbf{b}]$.

We also use the contractors to eliminate parts that are inside the solution set, but we observe some problems in existing solvers as soon as the domains of the functions involved in the problem are restricted. We now illustrate thus on a simple example.

Consider the set

$$\mathbb{X} = \{(x_1, x_2) \,|\, x_2 + \sqrt{x_1 + x_2} \in [1, 2]\}. \tag{4}$$

and let us try to compute an inner and outer approximations of $\mathbb{X}$ using an existing solver. For instance, if we use Codac [15] with the following script:

```
from codac import *
from vibes import *
X0=IntervalVector([[-10,10],[-10,10]])
f = Function(x1,x2,x2+sqrt(x1+x2))
S=SepFwdBwd(f,sqr(Interval(1,2)))
vibes.beginDrawing()
SIVIA(X0,S,0.01)
```

We get the paving illustrated by Figure 1 where the blue boxes are proved to be outside $\mathbb{X}$ and the magenta boxes are supposed to be inside. We observe that this is not the case. Indeed some boxes are wrongly classified as inside whereas they are outside. This phenomenon occurs for all existing solvers which are able to provide an inner approximation. The reasons for this is that contractor-based methods obtain an inner approximation by considering a contractor for the complementary of $\mathbb{X}$ as

$$\{(x_1, x_2) \,|\, x_2 + \sqrt{x_1 + x_2} \notin [1, 2]\}, \tag{5}$$

whereas it should be

$$\overline{\mathbb{X}} = \{(x_1, x_2) \,|\, x_2 + \sqrt{x_1 + x_2} \notin [1, 2] \text{ or } x_1 + x_2 < 0\}. \tag{6}$$

In the figure, some magenta zones are wrongly classified as inside because in these zones, $\sqrt{x_1 + x_2}$ is not defined. The goal of this paper is to provide a rigorous way to build contractors associated with the complementary of a set in the case where functions involved in the constraint are not defined everywhere.

The paper is organized as follows. Section 2 explains the approach that will motivate a new arithmetic. Section 3 presents an extension of the arithmetic on real numbers, named *total real arithmetic*, and shows the role of a flag named ι in the case where partial functions are involved. Section 4 introduces the notion of the total interval arithmetic. Section 4 provides the notion of total contractors and extends the classical forward-backward contractor to total intervals. Section 6 concludes the paper.

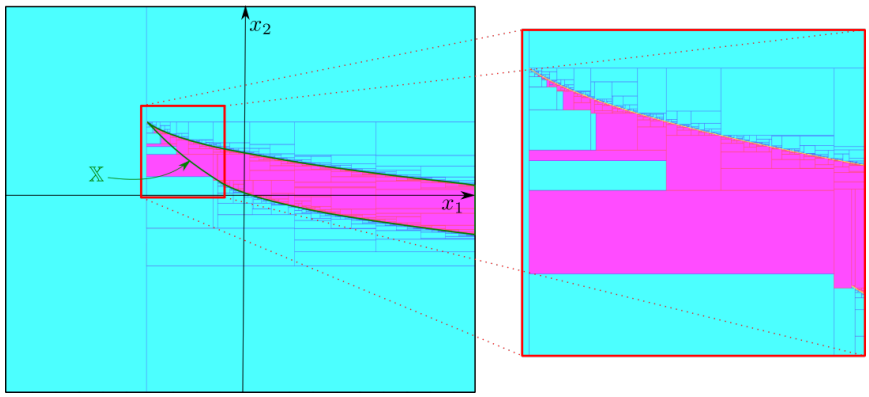


Figure 1: Left: Paving obtained by classical methods to approximate $\mathbb{X}$; Right: A zoom on the red box

2 Approach

Contractor algebra as defined in [4] does not allow any non-monotonic operation. It means that if a contractor $\mathcal{C}$ is defined by an expression $\mathcal{E}$ of other contractors $\mathcal{C}_i$ then we always have

$$\forall i, \mathcal{C}_i \subset \mathcal{C}_i' \Rightarrow \mathcal{E}\left(\mathcal{C}_1, \mathcal{C}_2, \dots\right) \subset \mathcal{E}\left(\mathcal{C}_1', \mathcal{C}_2', \dots\right). \tag{7}$$

As a consequence the complementary $\overline{\mathcal{C}}$ of a contractor $\mathcal{C}$ or the restriction $\mathcal{C}_1 \setminus \mathcal{C}_2$ of two contractors $\mathcal{C}_1, \mathcal{C}_2$ (which both correspond to non-monotonic operations) is not defined.

To be more precise, contractor algebra allows us to construct a contractor for expressions of sets defined by union, intersection and inversion of other sets. Take for instance the set

$$\mathbb{X} = \mathbb{X}_1 \cup \mathbf{f}^{-1}(\mathbb{X}_2 \cap \mathbb{X}_3). \tag{8}$$

We can represent its expression by the tree of Figure 2 or equivalently by the following expressions

$$\begin{array}{rcl} \mathbb{X} & = & \mathbb{X}_1 \cup \mathbb{B} \\ \mathbb{B} & = & \mathbf{f}^{-1}(\mathbb{A}) \\ \mathbb{A} & = & \mathbb{X}_2 \cap \mathbb{X}_3 \end{array}. \tag{9}$$

The intermediate sets $\mathbb{A}$ and $\mathbb{B}$ correspond to nodes of the tree. In practice, the leaves $\mathbb{X}_i$ of the tree are *set reverse* (or equivalently *inequality constraints*) of the form

$$\mathbb{X}_i = \varphi_i^{-1}([\mathbf{y}_i]) = \{\mathbf{x}_i \,|\, \varphi_i(\mathbf{x}_i) \in [\mathbf{y}_i]\}, \tag{10}$$

where φ_i is a function defined by an algorithm and $[\mathbf{y}_i]$ is a box of $\mathbb{R}^n$. A contractor for $\mathbb{X}_i$ is usually built by a forward-backward procedure as for instance $HC4$-revised [1]. The contractor associated with the constraint $\varphi(\mathbf{x}) \in [\mathbf{y}]$ is denoted by $\mathcal{C}^{\updownarrow}_{\varphi^{-1}([\mathbf{y}])}$.

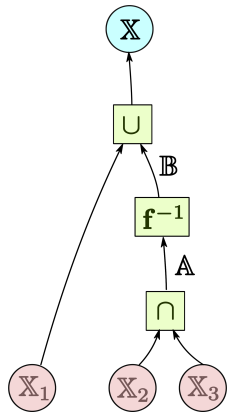


Figure 2: Contractor tree for $\mathbb{X}_1 \cup \mathbf{f}^{-1}(\mathbb{X}_2 \cap \mathbb{X}_3)$

Once the contractor for $\mathbb{X}$ is built from the tree, a paver [9] is called to provide an outer approximation for $\mathbb{X}$. More precisely, the paver generates boxes $[\mathbf{x}]$ of $\mathbb{R}^n$ that have to be contracted by the available contractors. The resulting procedure for contracting the set $\mathbb{X}$ defined by (8) is given by Algorithm 1.

Algorithm 1 Contractor for $\mathbb{X} = \mathbb{X}_1 \cup \mathbf{f}^{-1}(\mathbb{X}_2 \cap \mathbb{X}_3)$

Input: $[\mathbf{x}]$

$[\mathbf{x}_1] = \mathcal{C}_{\mathbb{X}_1}([\mathbf{x}])$
$[\mathbf{b}] = [\mathbf{x}]$
$[\mathbf{a}] = \mathbf{f}([\mathbf{b}])$
$[\mathbf{x}_2] = \mathcal{C}_{\mathbb{X}_2}([\mathbf{a}])$
$[\mathbf{x}_3] = \mathcal{C}_{\mathbb{X}_3}([\mathbf{a}])$
$[\mathbf{a}] = [\mathbf{x}_2] \cap [\mathbf{x}_3]$
$[\mathbf{b}] = \mathcal{C}^{\updownarrow}_{\mathbf{f}^{-1}([\mathbf{a}])}([\mathbf{b}])$
$[\mathbf{x}] = [\mathbf{x}_1] \sqcup [\mathbf{b}]$
return $[\mathbf{x}]$

This procedure is approximately what is performed by IBEX [3] even if IBEX does not admit a set expression as an input.

To express the complement $\overline{\mathbb{X}}$ we need to use the *De Morgan's laws* which states that:

- the complement of the union of two sets is the same as the intersection of their complements
- the complement of the intersection of two sets is the same as the union of their complements

We get

$$\overline{\mathbb{X}} = \overline{\mathbb{X}}_1 \cap \left(\mathbf{f}^{-1}(\overline{\mathbb{X}}_2 \cup \overline{\mathbb{X}}_3) \cup \overline{\text{dom}\mathbf{f}}\right). \tag{11}$$

Note that we had to introduce the domain of $\mathbf{f}$, denoted by dom$\mathbf{f}$, to take into account the fact that $\mathbf{f}$ may be a partial function (*i.e.*, not defined everywhere).

If we define the set-valued function $\mathring{\mathbf{f}}^{-1} : \mathcal{P}(\mathbb{R}^m) \mapsto \mathcal{P}(\mathbb{R}^n)$ as

$$\mathring{\mathbf{f}}^{-1}(\mathbb{Y}) = \mathbf{f}^{-1}(\mathbb{Y}) \cup \overline{\text{dom}\mathbf{f}}, \tag{12}$$

then we have

$$\overline{\mathbb{X}} = \overline{\mathbb{X}}_1 \cap \left(\mathring{\mathbf{f}}^{-1}(\overline{\mathbb{X}}_2 \cup \overline{\mathbb{X}}_3)\right). \tag{13}$$

The decomposition for $\overline{\mathbb{X}}$ is defined by

$$\begin{array}{rcl} \overline{\mathbb{X}} & = & \overline{\mathbb{X}}_1 \cap \overline{\mathbb{B}} \\ \overline{\mathbb{B}} & = & \mathring{\mathbf{f}}^{-1}(\overline{\mathbb{A}}) \\ \overline{\mathbb{A}} & = & \overline{\mathbb{X}}_2 \cup \overline{\mathbb{X}}_3 \end{array} \tag{14}$$

which corresponds to the tree of Figure 3.

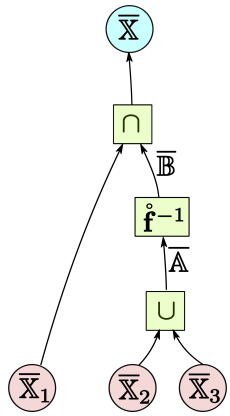


Figure 3: Contractor tree for the complementary of $\mathbb{X}_1 \cup \mathbf{f}^{-1}(\mathbb{X}_2 \cap \mathbb{X}_3)$

Since the sets $\mathbb{X}_i$ were defined by $\varphi_i(\mathbf{x}_i) \in [\mathbf{y}_i]$, the complement is by

$$\overline{\mathbb{X}}_i = \mathring{\varphi}_i^{-1}(\overline{[\mathbf{y}_i]}). \tag{15}$$

To implement, the complementary of a contractor using the *De Morgan's low,* the only brick we need is the forward-backward contractor for the set

$$\mathring{\mathbf{f}}^{-1}([\mathbf{y}]) = \mathbf{f}^{-1}([\mathbf{y}]) \cup \overline{\text{dom}\mathbf{f}}. \tag{16}$$

Now, the set $\mathring{\mathbf{f}}^{-1}([\mathbf{y}])$ is not a set reverse as defined by (10) and thus we cannot apply a forward-backward contractor without an extension which will be proposed in this paper.

3 Total extension

3.1 Definitions

In mathematics, a function $f : X \mapsto Y$ which is defined for all $x \in X$ is said to be *total*. Equivalently, a function $f : X \mapsto Y$ is total if

$$\forall x \in X, \exists y \in Y \text{ such that } f(x) = y. \tag{17}$$

A function f which is not defined for all x is said to be *partial*. Given a partial function f, the *total extension* is obtained by adding an element to Y, say ι which collects all $x \notin \mathrm{dom} f$. To be more precise, we give the following definition.

Definition. *The* total extension *of the partial function* $f : X \mapsto Y$ *is* $\mathring{f} = X \cup \{\iota\} \mapsto Y \cup \{\iota\}$ *with*

$$\mathring{f}(x) = \begin{cases} f(x) & \text{if } x \in dom f \\ \iota & \text{otherwise.} \end{cases} \tag{18}$$

Note that since $\iota \notin \mathrm{dom} f$, we have $\mathring{f}(\iota) = \iota$.

3.2 Illustration

Consider the partial function f as given in Figure 4. We have

$$\begin{array}{rcl} f^{-1}(\mathbb{Y}) & = & \{\beta, \gamma, \varepsilon\} \\ f^{-1}(\overline{\mathbb{Y}}) & = & \{\alpha\} \\ \mathrm{dom} f & = & \{\alpha, \beta, \gamma, \varepsilon\} \end{array} . \tag{19}$$

Now, since $f(\{\gamma, \delta\}) \subset \mathbb{Y}$, some would classify δ inside $f^{-1}(\mathbb{Y})$ which is wrong. This is would be true if f were total.

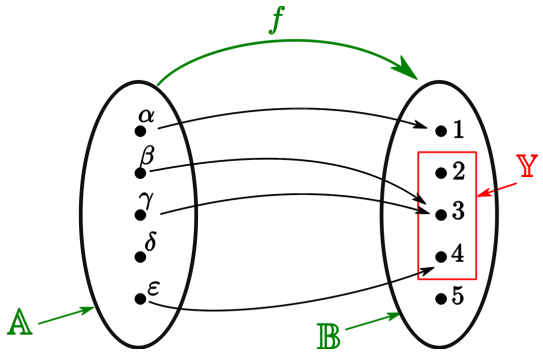


Figure 4: A partial function f

Introducing the indeterminate *NaN* (Not a number), denoted by ι, in the sets allows us to get rid of the problem involved by the partiality of f.

Given a set $\mathbb{A}$, we define the *extended total set* as $\mathring{\mathbb{A}} = \mathbb{A} \cup \{\iota\}$. Thus, $\mathring{f} : \mathring{\mathbb{A}} \mapsto \mathring{\mathbb{B}}$ is the total extension of $f : \mathbb{A} \mapsto \mathbb{B}$ as illustrated by Figure 5. Following Definition 1, extended functions can be used to set as follows:

$$\mathring{f}(\mathbb{X}) = \begin{cases} f(\mathbb{X}) & \text{if } \mathbb{X} \subset \text{dom} f \\ f(\mathbb{X}) \cup \{\iota\} & \text{otherwise} \end{cases}, \tag{20}$$

where $\mathbb{X} \subset \mathring{\mathbb{A}}$. Note that, in the figure, whereas $f(\{\gamma, \delta\}) = \{3\} \subset \mathbb{Y}$, we have $\mathring{f}(\{\gamma, \delta\}) = \{3, \iota\} \not\subset \mathbb{Y}$.

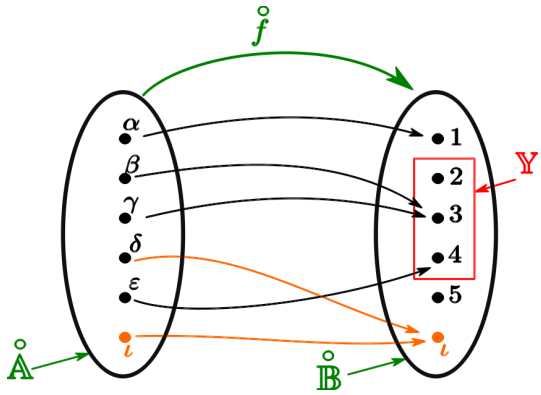


Figure 5: Introduction of Not a Number ι

3.3 Properties

For total functions, we have some properties that will be useful in our algorithms.

Proposition. *If $\mathring{f}$ is a total extension of f, we have*

$$\begin{array}{cc} \mathring{f}^{-1}(\overline{\mathbb{Y}}) = \overline{\mathring{f}^{-1}(\mathbb{Y})} & (i) \\ \mathring{f}(\mathbb{X}) \subset \mathbb{Y} \Rightarrow \mathbb{X} \subset \mathring{f}^{-1}(\mathbb{Y}) & (ii) \\ \mathring{f} \circ \mathring{f}^{-1}(\mathbb{Y}) = \mathbb{Y} & (iii) \end{array} . \tag{21}$$

Proof. Let us prove (ii) only. We have:

$$\begin{array}{rcc} \mathring{f}(\mathbb{X}) \subset \mathbb{Y} & \Leftrightarrow & \mathring{f}(\mathbb{X}) \cap \overline{\mathbb{Y}} = \emptyset \\ & \Leftrightarrow & \underbrace{\mathring{f}^{-1} \circ \mathring{f}(\mathbb{X})}_{\supset \mathbb{X}} \cap \underbrace{\mathring{f}^{-1}(\overline{\mathbb{Y}})}_{= \overline{\mathring{f}^{-1}(\mathbb{Y})}} = \emptyset \\ & \Rightarrow & \mathbb{X} \cap \overline{\mathring{f}^{-1}(\mathbb{Y})} = \emptyset \\ & \Leftrightarrow & \mathbb{X} \subset \mathring{f}^{-1}(\mathbb{Y}). \end{array} \tag{22}$$

□

Proposition. *If $\mathring{f}$, $\mathring{g}$ are total extensions of f, g then the total extension of $f \circ g$ is $\mathring{f} \circ \mathring{g}$.*

Proof. If $h = f \circ g$, we have

$$\mathring{f} \circ \mathring{g}(x) = \begin{cases} f \circ g(x) & \text{if } x \in \text{dom}g \text{ and } g(x) \in \text{dom}f \\ \iota & \text{otherwise.} \end{cases} \tag{23}$$

Now, since

$$\text{dom}h = \text{dom}f \circ g = \text{dom}g \cap g^{-1}(\text{dom}f) \tag{24}$$

we get

$$\mathring{f} \circ \mathring{g}(x) = \begin{cases} h(x) & \text{if } x \in \text{dom}h \\ \iota & \text{if } x \notin \text{dom}h \end{cases} \tag{25}$$

which corresponds to $\mathring{h}(x)$. □

Example. To illustrate the proposition, take

$$\begin{array}{rcl} f(x) & = & \sqrt{1-x} \\ g(x) & = & \sqrt{x-1} \end{array} . \tag{26}$$

Note that

$$\begin{array}{rcl} \text{dom}f & = & (-\infty, 1] \\ \text{dom}g & = & [1, \infty) \end{array} . \tag{27}$$

We have

$$h(x) = f \circ g(x) = \sqrt{1 - \sqrt{x-1}}. \tag{28}$$

Since

$$\begin{array}{rcl} \text{dom}h & = & \text{dom}g \cap g^{-1}(\text{dom}f) \\ & = & [1, \infty) \cap g^{-1}((-\infty, 1]) \\ & = & [1, \infty) \cap [0, 2] = [1, 2] \end{array} \tag{29}$$

we get

$$\mathring{h}(x) = \begin{cases} \sqrt{1 - \sqrt{x-1}} & \text{if } x \in [1, 2] \\ \iota & \text{otherwise} \end{cases} . \tag{30}$$

3.4 Total real arithmetic

We define the total extension of the classical arithmetic on real numbers. Consider the extended total set of reals:

$$\mathring{\mathbb{R}} = \mathbb{R} \cup \{\iota\}. \tag{31}$$

Adding such a special value for real numbers is now classical since it has been introduced by the IEEE 754 floating-point standard in 1985. Operations on real numbers can be extended to $\mathring{\mathbb{R}}$ as follows:

$$\begin{array}{ll} f(x) = \iota & \text{if } x \notin \text{dom}(f) \\ f(\iota) = \iota & \\ \iota \diamond x = \iota & \end{array} , \tag{32}$$

where f is any partial function and $x \in \mathbb{R}$ and any binary operator $\diamond$.

Note that we do not define comparisons, which means that if we have the relation $a \leq b$ then both a and b belong to $\mathbb{R}$ (or equivalently neither a nor b can be equal to ι).

Proposition. *Consider a partial function* $\mathbf{f} : \mathbb{R}^n \mapsto \mathbb{R}^m$ *given by an expression* $\mathbf{f}(x_1, \ldots, x_n)$ *including elementary functions (sin,* $\sqrt{}$*, log,...) and elementary operators* $(+, -, *, /, \ldots)$*. An expression for* $\mathring{\mathbf{f}}$ *can be obtained by the total real arithmetic.*

Proof. The proof is a direct consequence of the fact that the total extension is preserved by composition. □

An element of the Cartesian product $\mathring{\mathbb{R}}^n = \mathring{\mathbb{R}} \times \cdots \times \mathring{\mathbb{R}}$ is called a *total vector.*

3.5 Link with the complex number i

The set of complex numbers $\mathbb{C}$ extends the set of real numbers by adding a number i such that $i^2 = -1$. The extension preserves some properties such that the fact $\mathbb{C}$ is a group with respect to the addition. Due to this, i has an opposite: $-i$. Indeed, $i + (-i) = 0$.

Take now the set $\mathring{\mathbb{R}}$ and let us check if ι has an opposite. We solve $\iota + x = 0$ and we get no solution for x. This means that $\mathring{\mathbb{R}}$ is not anymore a group and many properties we had for $\mathbb{R}$ are lost. As a consequence, symbolic resolution and group-based simplifications are not allowed in $\mathring{\mathbb{R}}$.

Moreover, adding i to build $\mathbb{C}$ involves the addition of many numbers of the form $a + ib$. In $\mathring{\mathbb{R}}$, we just add a single number: ι.

There exists a tiny link between $\mathring{\mathbb{R}}$ and $\mathbb{C}$ in the construction since we add one number. But the link stops here. Whereas complex numbers can be used to build a huge numbers of theorems and theories, the total numbers will be used as a tool to build the complementary of contractors.

4 Total intervals

In this section, we introduce the notion of intervals for $\mathring{\mathbb{R}}$, called *total intervals.*

4.1 Intervals in unions of lattices

On a lattice $(\mathbb{A}, \leq_{\mathbb{A}})$, we can define the notion of intervals, interval hull and contractors. This has been used for several types of lattices such as real numbers, integers, trajectories, graphs, etc. To be able to use interval methods, the lattice structure is required. We show here that it is not strictly necessary by considering union of lattices.

Definition. *Consider two lattices* $(\mathbb{A}, \leq_{\mathbb{A}})$ *and* $(\mathbb{B}, \leq_{\mathbb{B}})$ *that are disjoint. Denote by* $\mathbb{I}\mathbb{A}$ *and* $\mathbb{I}\mathbb{B}$*, the set of all intervals of* $\mathbb{A}$ *and* $\mathbb{B}$*. We can define intervals of* $\mathbb{C} = \mathbb{A} \cup \mathbb{B}$ *as subsets* $\mathbb{C}$ *which have the form*

$$[c] = [a] \cup [b], \tag{33}$$

where $[a] \in \mathbb{I}\mathbb{A}$ *and* $[b] \in \mathbb{I}\mathbb{B}$*.*

Indeed, the set $(\mathbb{C}, \leq_{\mathbb{C}})$ can be equipped with an order relation:

$$x \leq_{\mathbb{C}} y \Leftrightarrow \left\{ \begin{array}{ll} & x \in \mathbb{A}, y \in \mathbb{A}, x \leq_{\mathbb{A}} y \\ \text{or} & x \in \mathbb{B}, y \in \mathbb{B}, x \leq_{\mathbb{B}} y \end{array} \right. . \tag{34}$$

Now, $\mathbb{C}$ is not a lattice, *i.e.*, if $x \in \mathbb{A}$, $y \in \mathbb{B}$ we cannot define $x \wedge y$ and $x \vee y$. This is due to the fact that we cannot provide a common lower or upper bounds for x, y.

Example. Consider the case where $\mathbb{A} = \mathbb{R}$ the set of real numbers and $\mathbb{B} = \{a, b, c, \ldots, z\}$ the set of letters. Both can be equipped with an order relation and both are lattices. Examples of intervals for the set $\mathbb{C} = \mathbb{A} \cup \mathbb{B}$ are

$$\begin{array}{l} [c_1] = [2, 5] \\ [c_2] = \{e, f, g, h\} \\ [c_3] = [2, 5] \cup \{e, f, g, h\} \\ [c_4] = [4, 9] \cup \{g, h, i\} \\ [c_5] = \emptyset \\ [c_6] = \mathbb{A} \cup \mathbb{B} \end{array} . \tag{35}$$

It is easy to check that the intervals of $\mathbb{C}$ is closed under intersection. It is thus a Moore family [2, 10]. As a consequence, contractor methods can be used.

4.2 Total intervals

Consider the singleton $\{\iota\}$ which is equipped with the trivial order relation: $\iota \leq \iota$. The set of all intervals of $\{\iota\}$ is $\{\emptyset, \{\iota\}\}$. The set $\mathring{\mathbb{R}}$ can be equipped with a partial order relation $\leq_{\mathring{\mathbb{R}}}$ derived from $\mathbb{R}$:

$$\begin{array}{c} \iota \leq_{\mathring{\mathbb{R}}} \iota \\ a \in \mathbb{R}, b \in \mathbb{R} \quad \text{then} \quad a \leq_{\mathring{\mathbb{R}}} b \text{ iff } a \leq_{\mathbb{R}} b \end{array} . \tag{36}$$

Total intervals are denoted by $[\mathring{x}]$.

Examples of intervals of $\mathring{\mathbb{R}}$ are:

$$\begin{array}{c} [\mathring{a}] = [1, \infty) \\ [\mathring{b}] = [-1, 0] \cup \{\iota\} \\ [\mathring{c}] = \{\iota\} \\ [\mathring{d}] = \emptyset, \end{array} \tag{37}$$

as illustrated by Figure 10.

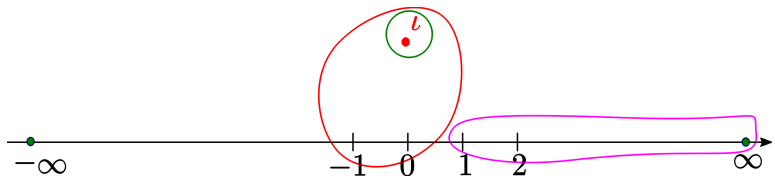


Figure 6: Total intervals are intervals of $\mathring{\mathbb{R}} = \mathbb{R} \cup \{\iota\}$

The set of total intervals is denoted by $\mathbb{I}\mathring{\mathbb{R}}$. We define the *hull* of a subset of $\mathring{\mathbb{X}}$ of $\mathring{\mathbb{R}}$ as the smallest total interval $[\mathring{x}]$ which encloses $\mathring{\mathbb{X}}$. We will write $[\mathring{x}] = \mathring{\mathbb{X}}$. For instance

$$\begin{array}{rcl} \{1,2,3\} & = & [1,3] \\ \{1,2,3,\iota\} & = & [1,3] \cup \{\iota\} \\ \{\iota\} & = & \{\iota\} \end{array} . \tag{38}$$

4.3 Total interval arithmetic

Consider a partial function $f : \mathbb{R} \mapsto \mathbb{R}$. We define its *total interval extension* as follows

$$[\mathring{f}] = \{\mathring{f}(\mathring{x}), \mathring{x} \in [\mathring{x}]\}. \tag{39}$$

For instance $\sqrt{[-1,4]} = [0,2] \cup \{\iota\}$.

In the same manner, if $\diamond \in \{+,-,\cdot,/\}$, we define

$$[\mathring{a}] \diamond [\mathring{b}] = \{\mathring{a} \diamond \mathring{b}, \mathring{a} \in [\mathring{a}], \mathring{b} \in [\mathring{b}]\}. \tag{40}$$

4.4 Total interval vector

The set of interval vectors $\mathring{\mathbb{R}}^n$ is a lattice [5]. We can thus define intervals of $\mathring{\mathbb{R}}^n$. The set of interval vectors has the form $\mathbb{I}\mathring{\mathbb{R}}^n = \mathbb{I}\mathring{\mathbb{R}} \times \cdots \times \mathbb{I}\mathring{\mathbb{R}}$. We define the *hull* of a subset of $\mathring{\mathbb{X}}$ of $\mathring{\mathbb{R}}^n$ as the smallest $[\mathring{\mathbf{x}}]$ which encloses $\mathring{\mathbb{X}}$. We will write $[\mathring{\mathbf{x}}] = \mathring{\mathbb{X}}$. For instance,

$$([1,2] \times \{\iota\}) \cup ([3,4] \times [5,6]) = [1,4] \times ([5,6] \cup \{\iota\}). \tag{41}$$

5 Total contractors

This section extends the notion of contractor to total intervals. We first consider the case of elementary contractors built from elementary functions. Then, we consider the case of contractors defined from elementary operators.

5.1 Total directed contractor for a binary constraint

Consider a constraint of the form $y = f(x)$, where $f : \mathbb{R} \mapsto \mathbb{R}$: is a partial function with domain $\mathrm{dom} f$. We can extend the constraint to $\mathring{\mathbb{R}}$ by the following decomposition

$$\left\{ \begin{array}{c} \mathring{y} = f(\mathring{x}) \\ \mathring{x} \in \mathring{\mathbb{R}} \\ \mathring{y} \in \mathring{\mathbb{R}} \end{array} \right. \Leftrightarrow \left\{ \begin{array}{ll} & \mathring{y} = f(\mathring{x}),\ \mathring{x} \in \mathrm{dom} f,\ \mathring{y} \in \mathbb{R} \\ \text{or} & \mathring{x} \in \mathbb{R} \setminus \mathrm{dom} f\ ,\ \mathring{y} = \iota \\ \text{or} & \mathring{x} = \iota,\ \mathring{y} = \iota \end{array} \right. . \tag{42}$$

This means that $\iota = f(x)$ is considered as true only if and only if $x = \iota$ or if $x \notin \mathrm{dom} f$. We define the *forward directional contractor* as

$$\overrightarrow{\mathcal{C}_f}([\mathring{x}]) = \{\mathring{y} \,|\, \exists \mathring{x} \in [\mathring{x}], \mathring{y} = f(\mathring{x})\} \tag{43}$$

and the *backward directional contractor*

$$\overleftarrow{\mathcal{C}_f}([\mathring{x}], [\mathring{y}]) = \{\mathring{x} \in [\mathring{x}] \,|\, \exists \mathring{y} \in [\mathring{y}],\ \mathring{y} = f(\mathring{x})\}. \tag{44}$$

Proposition. *The forward directional contractor associated with f is*

$$\overrightarrow{\mathcal{C}_f}([\mathring{x}]) = f([\mathring{x}] \cap \mathbb{R}), \cup ([\mathring{x}] \cap \{\iota\}) \cup \iota([\mathring{x}] \cap (\mathbb{R} \setminus dom f)), \tag{45}$$

where ι is the constant function ι, i.e,

$$\iota(\mathbb{A}) = \left\{ \begin{array}{ll} \iota & \text{if } \mathbb{A} \neq \emptyset \\ \emptyset & \text{if } \mathbb{A} = \emptyset \end{array} \right. . \tag{46}$$

Proof. Since

$$\begin{array}{rcl} \mathring{x} \in \mathrm{dom} f & \Rightarrow & \mathring{y} = f(\mathring{x}) \\ \mathring{x} = \iota & \Rightarrow & \mathring{y} = \iota \\ \mathring{x} \in \mathbb{R} \setminus \mathrm{dom} f & \Rightarrow & \mathring{y} = \iota \end{array} \tag{47}$$

we have

$$f([\mathring{x}]) = \underbrace{f([\mathring{x}] \cap \mathrm{dom}\, f)}_{f([\mathring{x}] \cap \mathbb{R})} \cup \underbrace{\iota([\mathring{x}] \cap \{\iota\})}_{= [\mathring{x}] \cap \{\iota\}} \cup \iota([\mathring{x}] \cap (\mathbb{R} \setminus \mathrm{dom} f)). \tag{48}$$

Thus

$$\begin{array}{rcl} \overrightarrow{\mathcal{C}_f}([\mathring{x}]) & = & \{\mathring{y} \,|\, \exists \mathring{x} \in [\mathring{x}], \mathring{y} = f(\mathring{x})\} \\ & = & f([\mathring{x}] \cap \mathbb{R}) \cup ([\mathring{x}] \cap \{\iota\}) \\ & & \cup \iota([\mathring{x}] \cap (\mathbb{R} \setminus \mathrm{dom}\, f)) \end{array} . \tag{49}$$

□

As a consequence, Algorithm 2 implements $\overrightarrow{\mathcal{C}_f}([\mathring{x}])$:

Algorithm 2 Forward directional contractor $\overrightarrow{\mathcal{C}_f}$

Input: $f, [\mathring{x}]$

1: $[\mathring{y}] = f([\mathring{x}] \cap \mathbb{R})$
2: $[\mathring{y}] = [\mathring{y}] \cup ([\mathring{x}] \cap \{\iota\})$
3: **if** $[\mathring{x}] \not\subset \operatorname{dom} f$ **then**
4: $\quad [\mathring{y}] = [\mathring{y}] \cup \{\iota\}$
5: **end if**
6: return $[\mathring{y}]$

Proposition. *The backward directional contractor associated with f is*

$$\overleftarrow{\mathcal{C}_f}([\mathring{x}], [\mathring{y}]) = [\mathring{x}] \cap \left(f^{-1}([\mathring{y}] \cap \mathbb{R}) \cup \mathbb{I}([\mathring{y}])\right), \tag{50}$$

where

$$\mathbb{I}([\mathring{y}]) = \begin{cases} \{\iota\} \cup (\mathbb{R} \setminus dom\, f) & if\ \iota \in [\mathring{y}] \\ \emptyset & otherwise \end{cases}. \tag{51}$$

Proof. We have

$$\begin{array}{rcl} \mathring{y} \in \mathbb{R} & \Leftrightarrow & \mathring{x} \in f^{-1}(\{\mathring{y}\}) \\ \mathring{y} = \iota & \Leftrightarrow & (\mathring{x} = \iota) \vee (\mathring{x} \in \mathbb{R} \setminus \operatorname{dom} f) \\ & \Leftrightarrow & \mathring{x} \in \{\iota\} \cup (\mathbb{R} \setminus \operatorname{dom} f) \end{array}. \tag{52}$$

□

This leads to Algorithm 3 which implements $\overleftarrow{\mathcal{C}_f}([\mathring{x}], [\mathring{y}])$:

Algorithm 3 Backward directional contractor $\overleftarrow{\mathcal{C}_f}$

Input: $f^{-1}, [\mathring{x}], [\mathring{y}]$

1: $[\mathring{r}] = \emptyset$
2: **if** $[\mathring{y}] = \emptyset$ **then**
3: $\quad$ return $[\mathring{r}]$
4: **end if**
5: $[\mathring{r}] = f^{-1}([\mathring{y}] \cap \mathbb{R})$
6: **if** $\iota \in [\mathring{y}]$ **then**
7: $\quad [\mathring{r}] = [\mathring{r}] \cup (\mathbb{R} \setminus \operatorname{dom} f) \cup \{\iota\}$
8: **end if**
9: return $[\mathring{r}] \cap [\mathring{x}]$

Example. *Total contractor for the square root.* Consider the constraint

$$y = \sqrt{x}, \tag{53}$$

where all variables belong to $\mathring{\mathbb{R}}$. The values $(9, 3), (-4, \iota), (\iota, \iota)$ for (x, y) are consistent with the constraint (53) whereas $(9, 2), (-4, 2), (9, \iota), (\iota, 2)$ are inconsistent.

For instance, assume that we have $x \in [\mathring{x}] = [-2, 9], y \in [\mathring{y}] = [-1, 2] \cup \{\iota\}$. We obtain

$$\begin{array}{lcl} \overrightarrow{\mathcal{C}_{\sqrt{\cdot}}}([\mathring{x}]) & = & \sqrt{[-2,9]} = [0,3] \cup \{\iota\} \\ \overrightarrow{\mathcal{C}_{\sqrt{\cdot}}}([\mathring{x}]) \cap [\mathring{y}] & = & ([0,3] \cup \{\iota\}) \cap ([-1,2] \cup \{\iota\}) \\ & = & [0,2] \cup \{\iota\} \\ \overleftarrow{\mathcal{C}_{\sqrt{\cdot}}}([\mathring{x}], [\mathring{y}]) & = & [-2,4] \end{array} . \tag{54}$$

It means that $x \in [-2, 4]$ and $y \in [0, 2] \cup \{\iota\}$.

Assume now that $x \in [\mathring{x}] = [4, 9], y \in [\mathring{y}] = [3, 15] \cup \{\iota\}$. We obtain

$$\begin{array}{lcl} \overrightarrow{\mathcal{C}_{\sqrt{\cdot}}}([\mathring{x}]) & = & \sqrt{[4,9]} = [2,3] \\ \overrightarrow{\mathcal{C}_{\sqrt{\cdot}}}([\mathring{x}]) \cap [\mathring{y}] & = & [2,3] \cap ([3,15] \cup \{\iota\}) = \{3\} \\ \overleftarrow{\mathcal{C}_{\sqrt{\cdot}}}([\mathring{x}], [\mathring{y}]) & = & \{9\} \end{array} . \tag{55}$$

It means that $x = 9$ and $y = 3$.

5.2 Total directed contractor for a ternary constraint

Consider the ternary constraint of $z = x + y$. The case of constraints involving $-, \cdot, /$ can be defined from $+$ and binary constraints already treated in the previous section. The following reasoning can also be done for these operators.

We can extend the constraint $z = x + y$ to $\mathring{\mathbb{R}}$ by the following decomposition

$$\left\{ \begin{array}{l} \mathring{z} = \mathring{x} + \mathring{y} \\ \mathring{x} \in \mathring{\mathbb{R}} \\ \mathring{y} \in \mathring{\mathbb{R}} \\ \mathring{z} \in \mathring{\mathbb{R}} \end{array} \right. \Leftrightarrow \left\{ \begin{array}{ll} & \mathring{z} = \mathring{x} + \mathring{y},\ \mathring{x} \in \mathbb{R}, \mathring{y} \in \mathbb{R},\ \mathring{z} \in \mathbb{R} \\ \text{or} & (\mathring{x} = \iota \vee \mathring{y} = \iota) \wedge \mathring{z} = \iota \end{array} \right. . \tag{56}$$

Note that in $\mathring{\mathbb{R}}$, we do not have

$$\mathring{z} = \mathring{x} + \mathring{y} \Leftrightarrow \mathring{x} = \mathring{z} - \mathring{y}. \tag{57}$$

Indeed, take $\mathring{x} = 1, \mathring{y} = \iota, \mathring{z} = \iota$. We have $\mathring{z} = \mathring{x} + \mathring{y}$ whereas $\mathring{x} \neq \mathring{z} - \mathring{y}$. As a consequence, the values $(2, 3, 5), (2, \iota, \iota), (\iota, \iota, \iota)$ for (x, y, z) are consistent with the constraint whereas $(2, 3, 6), (2, \iota, 4), (2, 3, \iota)$ are inconsistent.

We define the forward directed contractor

$$\overrightarrow{\mathcal{C}_+}([\mathring{x}], [\mathring{y}]) = \{\mathring{z} \,|\, \exists \mathring{x} \in [\mathring{x}], \exists \mathring{y} \in [\mathring{y}], \mathring{z} = \mathring{x} + \mathring{y}\} \tag{58}$$

and the backward directed contractor

$$\overleftarrow{\mathcal{C}_+}([\mathring{x}], [\mathring{y}], [\mathring{z}]) = \{(\mathring{x}, \mathring{y}) \in [\mathring{x}] \times [\mathring{y}] \,|\, \exists \mathring{z} \in [\mathring{z}],\ \mathring{z} = \mathring{x} + \mathring{y}\}. \tag{59}$$

We get Algorithms 4 and 5 for $\overrightarrow{\mathcal{C}_+}$ and $\overleftarrow{\mathcal{C}_+}$.

Algorithm 4 Forward directed contractor $\overrightarrow{\mathcal{C}_+}$

Input: $[\mathring{x}], [\mathring{y}]$

1: $[\mathring{z}] = ([\mathring{x}] \cap \mathbb{R}) + ([\mathring{y}] \cap \mathbb{R})$
2: $[\mathring{z}] = [\mathring{z}] \cup ([\mathring{x}] \cap \{\iota\}) \cup ([\mathring{y}] \cap \{\iota\})$
3: return $[\mathring{z}]$

Step 1 computes to the interval containing of all feasible $z \in \mathbb{R}$.

Step 2 adds ι when $\iota \in [\mathring{x}]$ or when $\iota \in [\mathring{y}]$.

Algorithm 5 Backward directed contractor $\overleftarrow{\mathcal{C}_+}$

Input: $[\mathring{x}], [\mathring{y}], [\mathring{z}]$

1: **if** $\iota \notin [\mathring{z}]$ **then**
2: $\quad [\mathring{x}] = [\mathring{x}] \cap ([\mathring{z}] - [\mathring{y}])$
3: $\quad [\mathring{y}] = [\mathring{y}] \cap ([\mathring{z}] - [\mathring{x}])$
4: **end if**
5: return $[\mathring{x}], [\mathring{y}]$

The implementation for $\overleftarrow{\mathcal{C}_+}$ is simplified by the fact that it is called after $\overrightarrow{\mathcal{C}_+}$.

Remark. The contractor $\overleftarrow{\mathcal{C}_+}$ is often minimal, but not always. Indeed, there exist some rare counterexamples. Consider for instance the case

$$\begin{array}{c} x \in [1,2] \\ y \in [3,4] \cup \{\iota\} \\ z \in [6,9] \cup \{\iota\} \end{array} . \tag{60}$$

If we call $\overrightarrow{\mathcal{C}_+}$, we get

$$\begin{array}{c} x \in [1,2] \\ y \in [3,4] \cup \{\iota\} \\ z \in [6,6] \cup \{\iota\} \end{array} . \tag{61}$$

The backward contractor $\overleftarrow{\mathcal{C}_+}$ (see Algorithm 5) yields no contraction for x and y whereas it should conclude the following contraction for y :

$$y \in [4,4] \cup \{\iota\}.$$

An optimal backward contractor could be obtained by Algorithm 6:

Algorithm 6 An optimal backward contractor

Input: $[\mathring{x}], [\mathring{y}], [\mathring{z}]$

$[\mathring{r}_x] = \emptyset$, $[\mathring{r}_y] = \emptyset$
$[x] = [\mathring{x}] \cap \mathbb{R}$; $[y] = [\mathring{y}] \cap \mathbb{R}$; $[z] = [\mathring{z}] \cap \mathbb{R}$;
if $[x] \neq \emptyset$ *and* $[y] \neq \emptyset$ *and* $[z] \neq \emptyset$ **then**
  $[\mathring{r}_x] = [x] \cap ([z] - [y])$;$[\mathring{r}_y] = [y] \cap ([z] - [x])$
end if
$[y] = [\mathring{y}] \cap \{\iota\}$; $[z] = [\mathring{z}] \cap \{\iota\}$;
if $[x] \neq \emptyset$ and $[y] \neq \emptyset$ and $[z] \neq \emptyset$ **then**
  $[\mathring{r}_y] = [\mathring{r}_y] \cup \{\iota\}$
end if
$[y] = [\mathring{y}] \cap \mathbb{R}$; $[x] = [\mathring{x}] \cap \{\iota\}$; $[z] = [\mathring{z}] \cap \{\iota\}$;
if $[x] \neq \emptyset$ and $[y] \neq \emptyset$ and $[z] \neq \emptyset$ **then**
  $[\mathring{r}_x] = [\mathring{r}_x] \cup \{\iota\}$
end if
$[x] = [\mathring{x}] \cap \{\iota\}$; $[y] = [\mathring{y}] \cap \{\iota\}$; $[z] = [\mathring{z}] \cap \{\iota\}$;
if $[x] \neq \emptyset$ and $[y] \neq \emptyset$ and $[z] \neq \emptyset$ **then**
  $[\mathring{r}_x] = [\mathring{r}_x] \cup \{\iota\}$; $[\mathring{r}_y] = [\mathring{r}_y] \cup \{\iota\}$
end if
return $[\mathring{r}_x]$, $[\mathring{r}_y]$

Now, this algorithm improves the efficiency of a propagation only for rare situations. This is why we will preferred the use of the backward contractor of Algorithm 5, even if not fully minimal.

5.3 Total forward-backward contractor

We show how the forward-backward contractor works on two test-cases.

Test-case 1. Consider the set

$$\mathbb{S} = \{(x, y) \,|\, y + \sqrt{x + y} \in [1, 2]\}. \tag{62}$$

We built the AST (Abstract Syntax Tree) associated with $\mathbb{S}$ as shown in Figure 7(a). We also build the AST for $\overline{\mathbb{S}}$ as in Figure 7(b). Note that the two trees are identical except the images that are complementary in $\mathring{\mathbb{R}}$, i.e.,

$$[1, 2] \cup ((-\infty, 1] \cup [2, \infty) \cup \{\iota\}) = \mathring{\mathbb{R}}. \tag{63}$$

A forward-backward contractor yields Algorithm 7. Note that below the set $\mathbb{Z}$ is not the set of integers (as often used in math books), but an interval of $\mathbb{R}$.

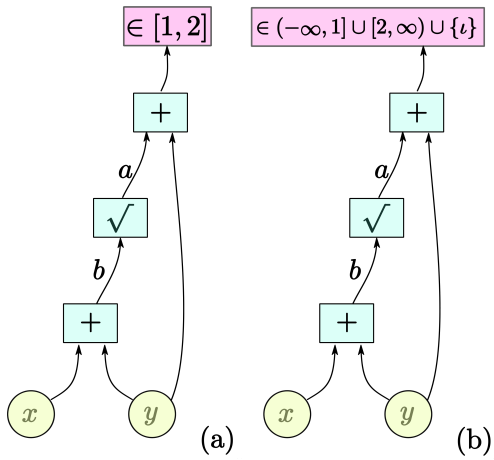


Figure 7: AST for the constraint $y + \sqrt{x+y} \in [1,2]$ (left) and its complementary (right)

Algorithm 7 Contractor for the constraint $y + \sqrt{x+y} \in \mathbb{Z}$

Input: $[\mathring{x}], [\mathring{y}], \mathbb{Z}$

1: $[\mathring{b}] = \overrightarrow{\mathcal{C}_+}([\mathring{x}], [\mathring{y}])$
2: $[\mathring{a}] = \overrightarrow{\mathcal{C}_\surd}([\mathring{b}])$
3: $[\mathring{z}] = \overrightarrow{\mathcal{C}_+}([\mathring{a}], [\mathring{y}])$
4: $[\mathring{z}] = [\mathring{z}] \cap \mathbb{Z}$
5: $[\mathring{a}], [\mathring{y}] = \overleftarrow{\mathcal{C}_+}([\mathring{z}], [\mathring{a}], [\mathring{y}])$
6: $[\mathring{b}] = \overleftarrow{\mathcal{C}_\surd}([\mathring{b}], [\mathring{a}])$
7: $[\mathring{x}], [\mathring{y}] = \overleftarrow{\mathcal{C}_+}([\mathring{b}], [\mathring{x}], [\mathring{y}])$
8: return $[\mathring{x}], [\mathring{y}]$

To have a contractor for $\mathbb{S}$ we call Algorithm 7 with $\mathbb{Z} = [1,2]$. To get a contractor for $\overline{\mathbb{S}}$, we call the algorithm with $\mathbb{Z} = (-\infty, 1] \cup [2, \infty) \cup \{\iota\}$. Using a paver with these two contractors, we are able to generate the approximation illustrated by Figure 8. The frame box is $[-10, 10] \times [-10, 10]$.

An implementation is given in [6].

Test-case 2. Consider the discrete-time state space system, inspired from Henon map, of the form $\mathbf{x}(k+1) = \mathbf{f}(\mathbf{x}(k))$ with

$$\mathbf{f}(\mathbf{x}) = \begin{pmatrix} bx_1 \\ 1 + ax_1^2 + \sqrt{x_2^2 + c} \end{pmatrix}, \tag{64}$$

where $a = -1.4$, $b = 0.3$ and $c = 0.075$. The behavior of this system may not lead to a well defined state $\mathbf{x}(k)$ if the initial state vector $\mathbf{x}(0)$ is not chosen properly.

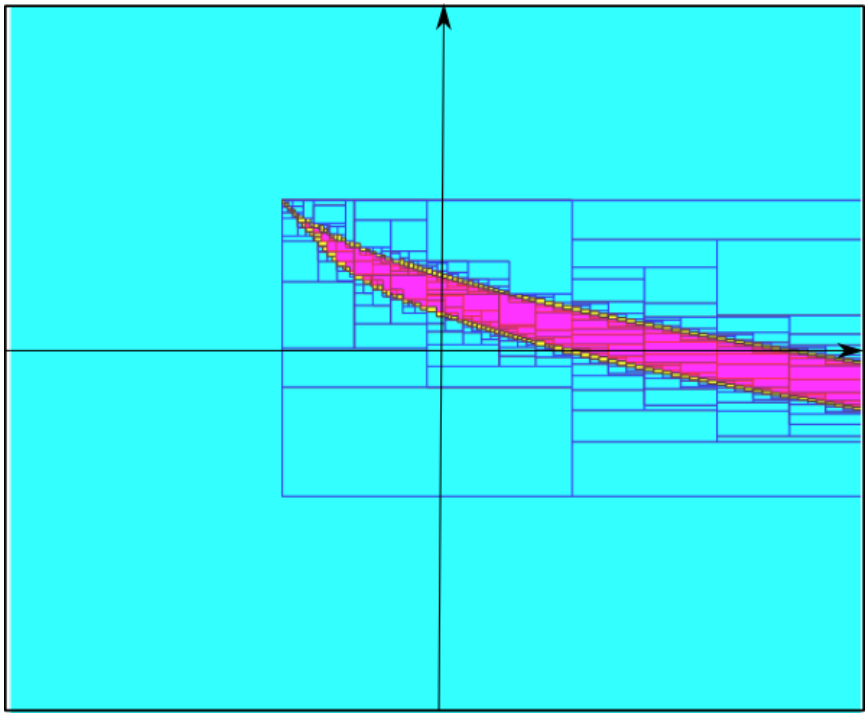


Figure 8: Paving obtained using the contractor for $\mathbb{S}$ and its complementary

We want to compute the set $\mathbb{S}$ of all initial vectors $\mathbf{x}(0)$ which lead to a state vector of $\mathbb{R}^2$ when $k = 5$. We define

$$\begin{array}{rcl} \mathbf{f}^0(\mathbf{x}) & = & \mathbf{x} \\ \mathbf{f}^{k+1}(\mathbf{x}) & = & \mathbf{f}^k \circ \mathbf{f}(\mathbf{x}) \end{array} . \tag{65}$$

We have

$$\mathbb{S} = \{\mathbf{x} \in \mathbb{R}^2 \,|\, \mathbf{f}^5(\mathbf{x}) \in \mathbb{R}^2\}. \tag{66}$$

The forward backward contractor (see Algorithm 8) associated to the constraint $\mathbf{f}(\mathbf{x}) \in \mathbb{Y}$ is based on the AST of Figure 9.

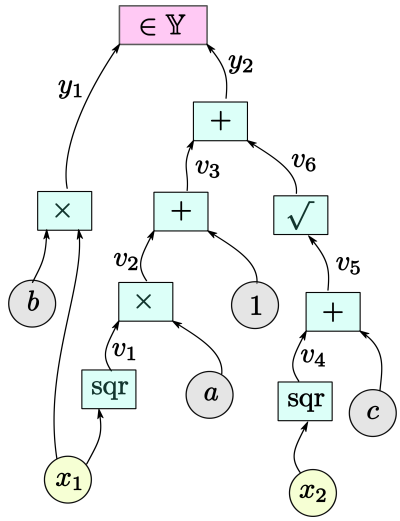


Figure 9: AST for the constraint $\mathbf{f}(\mathbf{x}) \in \mathbb{Y}$

Algorithm 8 Contractor for a constraint $\mathbf{f}(\mathbf{x}) \in \mathbb{Y}$ for Test-case 2

Input: $[\mathring{x}_1], [\mathring{x}_2], \mathbb{Y}$

1: $[\mathring{a}] = -1.4$; $[\mathring{b}] = 0.3$; $[\mathring{c}] = -0.075$
2: $[\mathring{v}_1] = \overrightarrow{\mathcal{C}_{sqr}}([\mathring{x}_1])$
3: $[\mathring{v}_2] = \overrightarrow{\mathcal{C}_{\times}}([\mathring{a}], [\mathring{v}_1])$
4: $[\mathring{v}_3] = 1 + [\mathring{v}_2]$
5: $[\mathring{v}_4] = \overrightarrow{\mathcal{C}_{sqr}}([\mathring{x}_2])$
6: $[\mathring{v}_5] = \overrightarrow{\mathcal{C}_{+}}([\mathring{v}_4], [\mathring{c}])$
7: $[\mathring{v}_6] = \overrightarrow{\mathcal{C}_{\surd}}([\mathring{v}_5])$
8: $[\mathring{y}_1] = \overrightarrow{\mathcal{C}_{\times}}([\mathring{x}_1], [\mathring{b}])$
9: $[\mathring{y}_2] = \overrightarrow{\mathcal{C}_{+}}([\mathring{v}_3], [\mathring{v}_6])$
10: $[\mathring{y}_1] \times [\mathring{y}_2] = ([\mathring{y}_1] \times [\mathring{y}_2]) \cap \mathbb{Y}$
11: $[\mathring{v}_3], [\mathring{v}_6] = \overleftarrow{\mathcal{C}_{+}}([\mathring{y}_2], [\mathring{v}_3], [\mathring{v}_6])$ *(see Step 9)*
12: $[\mathring{x}_1], [\mathring{b}] = \overleftarrow{\mathcal{C}_{\times}}([\mathring{y}_1], [\mathring{x}_1], [\mathring{b}])$ *(see Step 8)*
13: $[\mathring{v}_5] = \overleftarrow{\mathcal{C}_{\surd}}([\mathring{v}_5], [\mathring{v}_6])$ *(see Step 7)*
14: $[\mathring{v}_4], [\mathring{c}] = \overleftarrow{\mathcal{C}_{+}}([\mathring{v}_5], [\mathring{v}_4], [\mathring{c}])$ *(see Step 6)*
15: $[\mathring{x}_2] = \overleftarrow{\mathcal{C}_{sqr}}([\mathring{v}_4], [\mathring{x}_2])$ *(see Step 5)*
16: $[\mathring{v}_2] = ([\mathring{v}_3] - 1) \cap [\mathring{v}_2]$ *(see Step 4)*
17: $[\mathring{a}], [\mathring{v}_1] = \overleftarrow{\mathcal{C}_{\times}}([\mathring{v}_2], [\mathring{a}], [\mathring{v}_1])$ *(see Step 3)*
18: $[\mathring{x}_1] = \overleftarrow{\mathcal{C}_{sqr}}([\mathring{x}_1], [\mathring{v}_1])$ *(see Step 2)*
19: return $[\mathring{x}_1], [\mathring{x}_2]$

To have a contractor for $\mathbb{S}$ we call Algorithm 8 with $\mathbb{Y} = \mathbb{R}^2$. To get a contractor for $\overline{\mathbb{S}}$, we call the algorithm with $\mathbb{Y} = (\mathbb{R} \times \{\iota\}) \cup (\{\iota\} \times \mathbb{R}) \cup (\{\iota\} \times \{\iota\})$ which is the complementary of $\mathbb{R}^2$ in $(\mathbb{R} \cup \{\iota\})^2$ Using a paver with these two complementary contractors, we are able to generate the approximation illustrated by Figure 10. The frame box is $[-5, 5] \times [-5, 5]$.

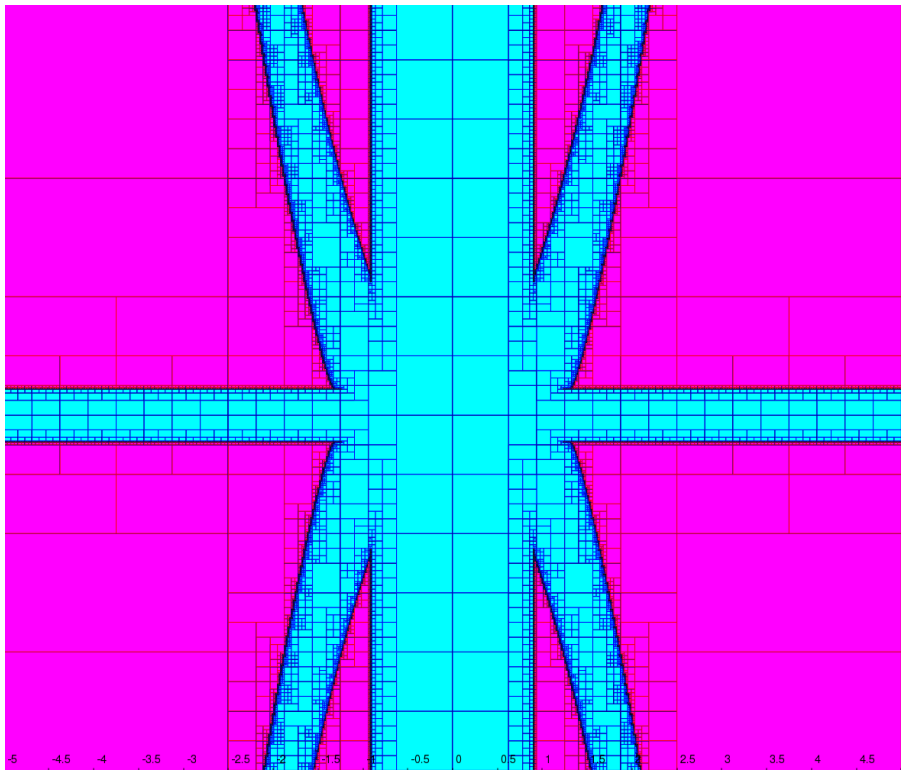


Figure 10: Paving representing the solution set for Test-case 2

6 Conclusion

In this paper, we have proposed to extend the interval arithmetic developed by Moore [12] in order to facilitate the implementation of complementary of contractors. For this purpose, we proposed to add a flag ι to each interval to form *total intervals*. The associated arithmetic has been derived. In our our new interval arithmetic, we have $\sqrt{[-1, 1]} = [0, 1] \cup \{\iota\}$ instead of $\sqrt{[-1, 1]} = [0, 1]$. This is due to the fact that $\sqrt{\cdot}$, which is a partial function, has been made total. The ι number is not seen anymore as an exception, but as a possible value.

The flag ι has similarities with some decorations already used in the context of interval computation [14], [8]. The main advantage of our extension is to allow the interval propagation when some partial functions are involved in the definition of the constraints. We have presented a generalization of the forward-backward propagation to total intervals. The efficiency has been illustrated on two test-cases.

References

[1] Benhamou, F., Goualard, F., Granvilliers, L., and Puget, J. F. Revising hull and box consistency. In *Proceedings of the International Conference on Logic Programming*, pages 230–244, Las Cruces, NM, 1999. DOI: 10.7551/mitpress/4304.003.0024.

[2] Blyth, T. *Lattices and Ordered Algebraic Structures*. Springer, 2005. DOI: 10.1007/b139095.

[3] Chabert, G. Ibex 2.0. Ecole des mines de Nantes, 2013. URL: http://www.emn.fr/z-info/ibex/.

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

[5] Davey, B. A. and Priestley, H. A. *Introduction to Lattices and Order*. Cambridge University Press, 2002. DOI: 10.1017/CBO9780511809088.

[6] Filiol, P., Bollengier, T., Jaulin, L., and Lann, J. L. Codes associated with the paper entitled: A new interval arithmetic to generate the complementary of contractors, 2022. URL: https://www.ensta-bretagne.fr/jaulin/iota.html.

[7] Hansen, E. R. *Global Optimization using Interval Analysis*. Marcel Dekker, New York, NY, 1992. ISBN: 9780824786960.

[8] IEEE Microprocessor Standards Commitee. IEEE 1788-2015 standard for interval arithmetic, 2015. URL: https://standards.ieee.org/ieee/1788/4431/.

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

[10] Moore, E. *Introduction to a form of general analysis*, Volume 2 of *Colloquium publications / American Mathematical Society*. Yale University Press, 1910. URL: https://books.google.hu/books?id=ezvvAAAAMAAJ.

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

[12] Moore, R., Kearfott, R., and Cloud, M. *Introduction to Interval Analysis*. SIAM, Philadelphia, PA, 2009. URL: https://epubs.siam.org/doi/10.1137/1.9780898717716.

[13] Neumaier, A. *Interval Methods for Systems of Equations*. Cambridge University Press, Cambridge, UK, 1990. DOI: 10.1017/CBO9780511526473.

[14] Revol, N. Introduction to the IEEE 1788-2015 standard for interval arithmetic. In *Proceedings of the 10th International Workshop on Numerical Software Verification*, 2017. DOI: 10.1007/978-3-319-63501-9_2.

[15] Rohou, S. Codac (Catalog Of Domains And Contractors). Robex, Lab-STICC, ENSTA-Bretagne, 2021. URL: http://codac.io/.

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

Acta Cybernetica **26** (2024) 839–854.

Exponential State Enclosure Techniques for the Implementation of Validated Model Predictive Control

Mohamed Fnadi[a] and Andreas Rauh[b]

Abstract

The design task of predictive controllers for uncertain systems is commonly formulated on the basis of their kinematic and/or dynamic models. These models are assumed to be expressed as initial value problems (IVPs) for finite-dimensional sets of nonlinear ordinary differential equations (ODEs). If constraints for the admissible state trajectories are formulated, bounds for these trajectories need to be computed by numerical procedures to obtain guaranteed enclosures of all possible states at each time step that contain the solution of the exact IVP-ODEs. Uncertainties in both the initial states and system parameters are considered in this paper by means of bounded interval variables. For this kind of system representation, we apply an exponential enclosure approach to determine guaranteed enclosures of all reachable states. This approach is embedded in a novel manner into the framework of a guaranteed nonlinear model predictive control (NMPC) to acquire optimal and safe control domains along a receding horizon. The NMPC problem is solved at each time step considering several constraints which are crucial for the system's safety and stability, namely, bounds on the state trajectories and the control signals. The capabilities of the combination of the exponential enclosure technique with the set-based NMPC strategy are illustrated through simulations using a nonlinear inverted pendulum.

Keywords: exponential enclosure, ordinary differential equations, model predictive control, guaranteed numerical integration

1 Introduction

Guaranteed numerical integration is a fundamental tool to solve initial value problems of ordinary differential equations (IVP-ODEs) with uncertain initial conditions

[a]Laboratoire d'Informatique, Signal et Image de la Côte d'Opale – LISIC UR 4491, Université du Littoral Côte d'Opale, F-62228, France. E-mail: mohamed.fnadi@univ-littoral.fr, ORCID: 0000-0001-9593-8859

[b]Carl von Ossietzky Universität Oldenburg, Group: Distributed Control in Interconnected Systems, D-26111 Oldenburg, Germany. E-mail: andreas.rauh@uni-oldenburg.de, ORCID: 0000-0002-1548-6547

DOI: 10.14232/actacyb.302013

and parameters in a reliable and validated way[1]. Providing guaranteed solution enclosures to these IVP-ODEs is essential for designing and verifying linear and nonlinear feedback controllers, mainly for predictive control approaches. In the literature, several validated approaches have been designed for systems with uncertain initial conditions and uncertain parameters to compute state enclosures which are guaranteed to contain all possible IVP-ODEs solutions (under the assumption of correct models and correct bounds of all uncertain parameters involved). For instance, set-valued integration methods exploiting interval Taylor series and Taylor model-based techniques (e.g., the solvers VNODE-LP or COSY VI) were used for the verification of uncertain systems [9, 15]. Moreover, Runge–Kutta schemes (implemented in the DynIbex library) have been used to obtain tight state enclosures [1, 2].

Nevertheless, it has been shown that — due to the computational complexity of Runge–Kutta methods, as used in the previous work of the first author [4] — fast convergence and high accuracy of the computed enclosures are not always guaranteed for finitely long integration time spans, possibly leading to an excessive duration to get the IVP-ODEs' solutions. Similar statements also hold for other state-of-the-art techniques exploiting interval-based methods for solving IVP-ODEs in a guaranteed and validated way (e.g., RealPaver [8], CAPD [10]). This leads to the observation that these solvers may become impractical for the task of control optimization in an NMPC framework due to their high computation time. In other words, fast convergence of this kind of solvers cannot be guaranteed over a finite time and extremely long response times may be exhibited when searching for tight enclosures of the solutions for the problem and hand. To tackle this issue of the computational burden, exponential enclosure techniques for IVP-ODE problems seem to be attractive to remarkably reduce the computing time of validated methods and to approach real-time capability [18, 19]. Compared to Taylor series or Runge–Kutta model-based techniques, the exponential enclosure approach allows contracting the computed state enclosures over time for asymptotically stable dynamics, which prevents growing diameters of the interval enclosures [16]. However, the computed interval bounds may be wider than those determined by alternative, more complex, solution techniques.

The primary goal of an NMPC strategy is to deploy a plant model to predict the system behavior along a receding horizon. At each sampling point, the NMPC technique computes the optimal control input that minimizes a cost function and satisfies all safety constraints (e.g., bounds for the actuator outputs as well as constraints for internal system variables such as position, speed, and acceleration). Among the existing works, real-time, constrained NMPC approaches with safety and stability constraints were proposed in [3, 6], where all constraints are expressed

[1] Due to the fact that these integration routines provide guaranteed state enclosures, they are typically denoted as *verified* in the literature. Throughout this paper, however, we use the term *validated* to point out that the underlying models are approximations to the actual dynamics for which parameters are determined by means of experimental identification. Therefore, a full *verification* of state bounds is not possible but rather only the computation of state enclosures that contain the true reachable sets with a high level of confidence.

in terms of inequalities with respect to the optimization variables. This approach is based on the exact linearization of the nonlinear model so as to formulate the optimization problem completely as a quadratic programming task. Such kind of problem can be handled easily by linear solvers to obtain the optimal control variables. Nevertheless, most of the existing predictive control techniques assume that uncertainties, related to internal parameters of the system model as well as to sensor measurements with bounded accuracy, are neglected. To solve this problem, guaranteed control strategies were developed in recent work to ensure robustness toward all the uncertainties occurring in dynamic parameters. Despite the capability to handle constraints in a reliable manner, they need huge time to compute the state enclosures [4, 14]. The time aspect is especially crucial, because at each sampling instant a validated NMPC needs to compute optimal and guaranteed system inputs along a receding horizon that minimize some interval cost function and ensure compatibility constraints (such as the aforementioned actuator saturations and safety constraints on the state trajectories).

Our motivation is to interface the exponential enclosure techniques published in [18, 19] with the validated NMPC developed in [4], which is based on Runge–Kutta schemes, to remarkably speed up the solution. The NMPC combined with exponential state enclosures will allow to speed up the search for admissible control sequences based on tight enclosures of the dynamic model considering bounded uncertainties on both initial conditions and dynamic parameters. The goal of the proposed controller is twofold: First, a branching procedure enables one to find safe control domains that obey the state constraints and ensure the convergence to reference intervals. Second, the optimization procedure allows the computation of an optimal and point-valued control input that will finally be applied to the system's actuators.

The remainder of this paper is organized as follows. Firstly, an overview of the exponential state enclosure technique is given in Sec. 2. Secondly, Sec. 3 introduces the guaranteed NMPC approach based on interval arithmetic and a validated exponential state enclosure approach. Thirdly, simulation results and discussions are reported in Sec. 4, before Sec. 5 gives conclusions and an outlook on future work.

2 An Exponential Enclosure Technique for Computing Guaranteed Solution Enclosures for IVP-ODEs

Classical numerical integration methods compute approximations of solutions of IVP-ODEs, however, without guarantees on the accuracy of the approximation. Thus, verified approaches have been developed that are supposed to compute guaranteed enclosures of the solution. Firstly, we give an overview of general validated numerical integration methods in Sec. 2.1. Secondly, we present the concept of exponential enclosures in Sec. 2.2. Finally, techniques for stability analysis and an underlying stabilization of open-loop unstable plants are presented in Sec. 2.3 to

achieve a cascaded control architecture that allows for using exponential enclosures efficiently in the frame of NMPC.

2.1 Verified Computation of Enclosures of IVP-ODEs

Consider an uncertain dynamical system defined by the IVP-ODEs

$$\begin{cases} \dot{\mathbf{x}}_t = \mathbf{f}(t, \mathbf{x}_t, \mathbf{u}, \mathbf{p}) \\ \mathbf{x}_0 \in [\mathbf{x}_0] \subseteq \mathbb{IR}^n \\ \mathbf{u} \in [\mathbf{u}] \subseteq \mathbb{IR}^m \\ \mathbf{p} \in [\mathbf{p}] \subseteq \mathbb{IR}^p, \end{cases} \tag{1}$$

where the state vector is denoted by $\mathbf{x}_t$, the vector of dynamic parameters by $\mathbf{p}$, and the control vector by $\mathbf{u}$. The sets $[\mathbf{x}_0] = \left[[x_{10}] \quad \dots \quad [x_{n0}]\right]^T$, $[\mathbf{u}] = \left[[u_1] \quad \dots \quad [u_m]\right]^T$, and $[\mathbf{p}] = \left[[p_1] \quad \dots \quad [p_p]\right]^T$, expressed as interval boxes, are respectively the initial condition of the state vector, the interval-bounded input, and the set of feasible dynamic parameters. This IVP-ODE has a unique solution $\mathbf{x}_t(t, \mathbf{x}_0, \mathbf{u}, \mathbf{p})$ at $t > 0$ since $\mathbf{f} : \mathbb{R} \times \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^p \to \mathbb{R}^n$ is continuous in t and Lipschitz in $\mathbf{x}_t$ (assuming that $\mathbf{u}$ and $\mathbf{p}$ are known and constant). For our purpose, we assume additionally that $\mathbf{f}$ is sufficiently smooth, i.e., of class C^k.

Compared to the classical numerical integration for an IVP-ODE problem, validated approaches consist of solving this problem in a complete and validated way (i.e., each actual solution is rigorously returned and enclosed in a (sufficiently) tight interval). These methods are commonly based on Taylor series [15] or Runge–Kutta methods [2]. Basically, the main principle of validated integration methods is to obtain the tight enclosure of the IVP-ODE problem.

As presented in [1], the purpose of a validated numerical algorithm is to solve Eq. (1) so as to obtain a sequence of boxes $[\mathbf{x}_0], \dots, [\mathbf{x}_K]$ at the time instants $t_0 = 0 < \dots < t_K = T$. At each guaranteed integration step, it is assumed that input and parameter boxes ($[\mathbf{u}_j]$ and $[\mathbf{p}_j]$) are known to compute the state sequences. It is achieved in such a way that the inclusion function, denoted by $[\mathbf{F}]$, satisfies the property

$$[\mathbf{x}_{j+1}] \supseteq [\mathbf{F}]\left(t_j, [\mathbf{x}_j], [\mathbf{u}_j], [\mathbf{p}_j]\right), \ \forall j \in \{0, \dots, K\}. \tag{2}$$

Such approaches work in two stages at each integration step to compute the guaranteed solutions. These steps are:

i) Computation of a *prior* enclosure of the solution $[\tilde{\mathbf{x}}_{j+1}]$, such that for all t in the time interval $t \in [t_j \ ; \ t_{j+1}]$, the inclusion property

$$\mathbf{F}\left(t_j, [\mathbf{x}_j], [\mathbf{u}_j], [\mathbf{p}_j]\right) \in [\tilde{\mathbf{x}}_{j+1}] \tag{3}$$

is satisfied. This stage enables proving the existence and the uniqueness of the solution.

ii) Computation of a *tight* enclosure of the state $[\mathbf{x}_{j+1}]$ at the time instant t_{j+1}, such that $\mathbf{F}\left(t_{j+1}, [\mathbf{x}_j], [\mathbf{u}_j], [\mathbf{p}_j]\right) \in [\mathbf{x}_{j+1}]$. This step makes use of the solution $[\tilde{\mathbf{x}}_{j+1}]$ to bound the *truncation error*, i.e., the distance between the exact solution and the numerical approximation.

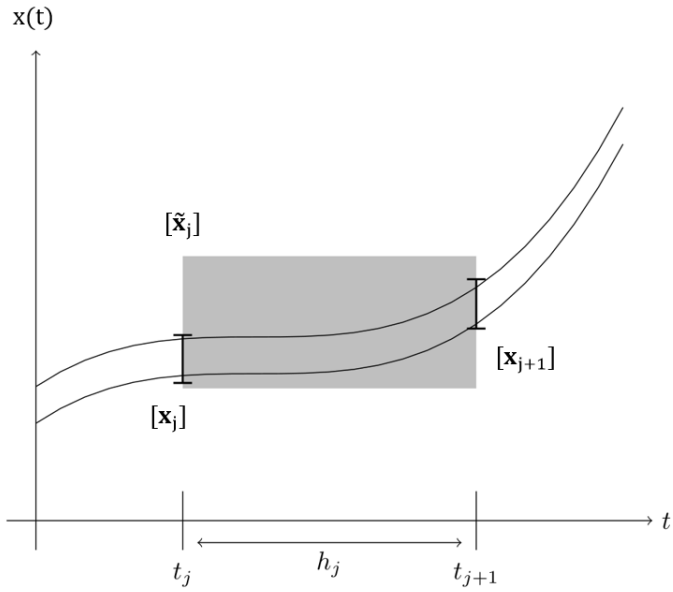


Figure 1: Prior and tight enclosures computed along a single discretization interval using a validated method.

The *tight* and *prior* enclosures calculated along one integration step between t_j and t_{j+1} (step size $h_j = t_{j+1} - t_j > 0$) are visualized exemplarily in Fig. 1.

2.2 Exponential Enclosure Technique

Guaranteed numerical integration methods aim at computing the state enclosure sequence $(t_j, [\mathbf{x}_j])_{j\in\mathbb{N}}$, assuming that the input and parameter boxes $[\mathbf{u}]$ and $[\mathbf{p}]$, respectively, are piecewise constant and known for each run of the validated simulation. To avoid the two-stage evaluation, resulting from a truncated series representation of the solution to the IVP-ODE (1), the exponential enclosure technique [18, 19] is applied to approximate the solutions in a verified manner. It has been shown that this method may improve the accuracy of the computed state enclosures while simultaneously reducing the required computation time for asymptotically stable systems [18, 19].

The dynamic model (1) can be reformulated by considering that the dynamic parameters are represented by constant intervals, and the input variables are assumed to be included in an augmented state vector, i.e., $\left[\mathbf{x}_t^T \quad \mathbf{u}^T(\mathbf{x}_t)\right]^T$, denoted

for brevity again as $\mathbf{x}_t$ with

$$\dot{\mathbf{x}}_t = \mathbf{f}(\mathbf{x}_t). \tag{4}$$

To ensure the (local) asymptotic stability of the system model in the neighborhood of a desired terminal state, we assume — as a prerequisite for the exponential enclosure approach — that a feedback controller $\mathbf{u}_{\mathrm{fb}}(\mathbf{x}_t)$ is included in a cascaded manner in the control law

$$\mathbf{u}(\mathbf{x}_t) = \mathbf{u}_{\mathrm{fb}}(\mathbf{x}_t) + \mathbf{u}_{\mathrm{ff}}(t) \tag{5}$$

so that the NMPC effectively computes a kind of feedforward control sequence $\mathbf{u}_{\mathrm{ff}}(t)$ as shown in Fig. 2.

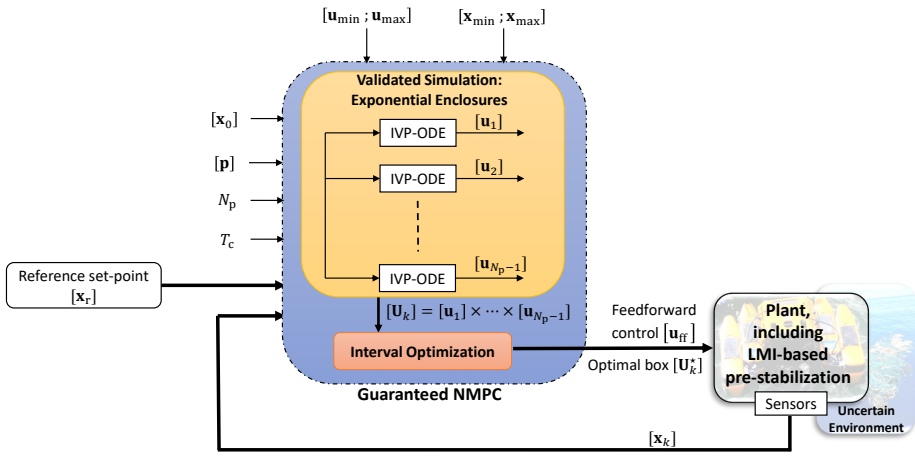


Figure 2: Overall structure of the validated NMPC.

To prevent the growth of the diameters of the intervals $(t_j, [\mathbf{x}_j])_{j\in\mathbb{N}}$ for asymptotically stable systems with a minimum computational effort, the exact solution $\mathbf{x}_t^\star$ can be bracketed by the following exponential state enclosure

$$\mathbf{x}_t^\star \in [\mathbf{x}_e](t) = \exp\Big([\mathbf{\Lambda}]t\Big)[\mathbf{x}_e](0)\ , \quad [\mathbf{x}_e](0) = [\mathbf{x}_0], \tag{6}$$

where $\mathbf{\Lambda}$ represents a yet unknown matrix after a translation of the state space so that the origin $\mathbf{x} = \mathbf{0}$ corresponds of the system's (asymptotically stable) equilibrium. By choosing $[\mathbf{\Lambda}] = \mathrm{diag}\{[\lambda_i]\}$, $i = 1, \ldots, n$, as a diagonal matrix, its elements λ_i need to have negative real parts to describe contracting state enclosures.

Using the exponential state enclosures (6) and a Picard iteration with the iteration index κ, we obtain

$$\begin{aligned}\mathbf{x}_t^\star \in [\mathbf{x}_e]_{(\kappa+1)} &= \exp\Big([\mathbf{\Lambda}]_{(\kappa+1)}\, t\Big)[\mathbf{x}_e](0) \\ &= [\mathbf{x}_e](0) + \int_0^t \mathbf{f}\Big(\exp\Big([\mathbf{\Lambda}]_{(\kappa)}\, s\Big)[\mathbf{x}_e](0)\Big)\,\mathrm{d}s.\end{aligned} \tag{7}$$

The differentiation of (7) with respect to time, belonging to the integration interval $t \in [0\,;\,T]$, leads to

$$\dot{\mathbf{x}}_{[t]}^\star \in [\mathbf{\Lambda}]_{(\kappa+1)} \exp\Big([\mathbf{\Lambda}]_{(\kappa+1)}[t]\Big)[\mathbf{x}_e](0) = \mathbf{f}\Big(\exp\Big([\mathbf{\Lambda}]_{(\kappa)}[t]\Big)[\mathbf{x}_e](0)\Big). \tag{8}$$

Assuming a converging iteration with $[\mathbf{\Lambda}]_{(\kappa+1)} \subseteq [\mathbf{\Lambda}]_{(\kappa)}$ and, thus, $[\lambda_i]_{(\kappa+1)} \subseteq [\lambda_i]_{(\kappa)}$, the iteration formula for $[\lambda_i]_{(\kappa+1)}$ can be expressed as

$$[\lambda_i]_{(\kappa+1)} = \frac{f_i\Big(\exp\Big([\mathbf{\Lambda}]_{(\kappa)}[t]\Big)[\mathbf{x}_{e,i}](0)\Big)}{\exp\Big([\mathbf{\Lambda}]_{(\kappa)}[t]\Big)[\mathbf{x}_{e,i}](0)}, \quad i = 1, \ldots, n. \tag{9}$$

The guaranteed state enclosure at the time instant $T = \sup([t])$ (the chosen end of the prediction window) is given by

$$\mathbf{x}_t^\star \in [\mathbf{x}_e](t) = \exp\Big([\mathbf{\Lambda}]T\Big)[\mathbf{x}_e](0), \tag{10}$$

where $[\mathbf{\Lambda}]$ is the final result of the iteration (9). For a suitable step size control strategy, allowing for a reduction of overestimation in the computed solutions for systems with non-negligible nonlinearities, the reader is referred to [11, Sec. III.C].

2.3 Design of the Subsidiary Robustly Stabilizing Feedback Controller

To design a linear, robustly stabilizing subsidiary feedback controller

$$\mathbf{u}_{\mathrm{fb}}(\mathbf{x}_t) = -\mathbf{K}\mathbf{x}_t, \tag{11}$$

we assume that the state equations

$$\dot{\mathbf{x}}_t = \mathbf{f}(t, \mathbf{x}_t, \mathbf{u}, \mathbf{p}) \tag{12}$$

can be reformulated into a quasi-linear form

$$\dot{\mathbf{x}}_t = \mathbf{A}(\mathbf{x}_t, \mathbf{u}, \mathbf{p})\mathbf{x}_t + \mathbf{B}(\mathbf{x}_t, \mathbf{p})\mathbf{u}. \tag{13}$$

As in the previous subsection, we assume for simplicity of the notation that the state space has been translated so that $\mathbf{x} = \mathbf{0}$ corresponds to the desired equilibrium.

Then, a robust linear state feedback controller (11) can be designed by a linear matrix inequality approach on the basis of the Lyapunov function candidate

$$V(\mathbf{x}_t) = \mathbf{x}_t^T \mathbf{P} \mathbf{x}_t \tag{14}$$

with the yet unknown symmetric, positive-definite matrix $\mathbf{P} = \mathbf{P}^T \succ 0$. For asymptotic stability, the property

$$\dot{V}(\mathbf{x}_t) < 0 \tag{15}$$

needs to hold for $\mathbf{x}_t \neq \mathbf{0}$. To ensure a minimum decay rate γ, this inequality can be replaced by

$$\dot{V}(\mathbf{x}) \leq -\gamma V(\mathbf{x}) \quad \text{with} \quad \gamma > 0. \tag{16}$$

Assuming further that the quasi-linear system model can be embedded into a convex polytopic uncertainty representation

$$\mathcal{D} = \left\{ [\mathbf{A}(\xi), \mathbf{B}(\xi)] \mid [\mathbf{A}(\xi), \mathbf{B}(\xi)] = \sum_{v=1}^{n_v} \xi_v \, [\mathbf{A}_v, \mathbf{B}_v,] \, ; \sum_{v=1}^{n_v} \xi_v = 1; \xi_v \geq 0 \right\}, \tag{17}$$

the bilinear matrix inequalities ($v = 1, \ldots, n_v$)

$$(\mathbf{A}_v - \mathbf{B}_v \mathbf{K})^T \, \mathbf{P} + \mathbf{P} \, (\mathbf{A}_v - \mathbf{B}_v \mathbf{K}) \prec 0 \; , \quad \mathbf{P} \succ 0 \tag{18}$$

need to be solved for the state-independent gain $\mathbf{K}$ to ensure asymptotic stability according to (15). This is typically done by means of a linearizing change of variables such as described in [21].

The more strict condition (16) is satisfied if

$$(\mathbf{A}_v - \mathbf{B}_v \mathbf{K})^T \, \mathbf{P} + \mathbf{P} \, (\mathbf{A}_v - \mathbf{B}_v \mathbf{K}) \prec 2\gamma \mathbf{P} \; , \quad \mathbf{P} \succ 0 \tag{19}$$

holds.

A drawback of this polytopic model approach is a certain degree of conservativeness introduced by treating all matrix entries as independent. This can be reduced by a norm-bounded uncertainty representation similarly used in [17].

In this case, robust asymptotic stability is achieved by satisfying the matrix inequality

$$\begin{aligned} &\begin{bmatrix} \mathbf{A}_{\text{nom}} \mathbf{P} - \mathbf{B}_{\text{nom}} \mathbf{Z} + \mathbf{P} \mathbf{A}_{\text{nom}}^T - \mathbf{Z}^T \mathbf{B}_{\text{nom}}^T & \mathbf{P} \mathbf{N}^T + \mathbf{Z}^T \mathbf{D}_{12}^T \\ \mathbf{N} \mathbf{P} + \mathbf{D}_{12} \mathbf{Z} & -\mu \mathbf{I} \end{bmatrix} \\ &+ \mu \begin{bmatrix} \mathbf{M} \mathbf{M}^T & \mathbf{M} \mathbf{Q}^T \\ \mathbf{Q} \mathbf{M}^T & \mathbf{Q} \mathbf{Q}^T \end{bmatrix} \prec 0 \end{aligned} \tag{20}$$

with $\mathbf{A}_{\text{nom}} = \text{mid}([\mathbf{A}]), \mathbf{B}_{\text{nom}} = \text{mid}([\mathbf{b}]), \mathbf{N} = \text{rad}([\mathbf{A}]), \mathbf{M} = \mathbf{I}, \mathbf{Q} = \mathbf{0}$, $\mathbf{D}_{12} = \text{rad}([\mathbf{B}]), \mathbf{P} \succ 0, \mathbf{K} = \mathbf{Z} \mathbf{P}^{-1}$, where $[\mathbf{A}]$ and $[\mathbf{B}]$ represent interval enclosures of the respective matrices in the polytopic model (17).

Remark. A means for further reduction of conservativeness, to be investigated in future work in combination with the validated NMPC, is the use of parameter-dependent Lyapunov function candidates, leading typically to larger regions of attraction for the equilibrium to be stabilized. In addition, it can be expected that parameter-dependent Lyapunov functions will allow for a more efficient use of the control effort [21]. Moreover, approaches for introducing hard saturations of the control signal and its variations rates as described in [13] can be investigated in future work.

Using the linear matrix inequality approach above, the following conclusions can be drawn:

- The feedforward term $\mathbf{u}_{\text{ff}}$ is mandatory for states outside the polytopic domain (17), where the stability of the feedback controller is not proven and this control part is, therefore, deactivated;
- The feedback controller can be activated as soon as states corresponding to the interior of the polytope (17) are attained;
- The exponential enclosure approach can be expected to find contracting solutions in the interior of this polytope, especially in combination with the comparison lemma [12] detailed subsequently.

The comparison lemma can be exploited to compute bounds for the Lyapunov function $V(\mathbf{x}_t)$ according to

$$\lambda_{\min}(\mathbf{P})\|\mathbf{x}_t\|_2^2 \leq \mathbf{x}_t^T\mathbf{P}\mathbf{x}_t \leq \lambda_{\max}(\mathbf{P})\|\mathbf{x}_t\|_2^2, \tag{21}$$

$$0 \leq \mathbf{x}_t^T\mathbf{P}\mathbf{x}_t \leq \mathbf{x}^T(0)\mathbf{P}\mathbf{x}(0), \tag{22}$$

where $\lambda_{\min}(\mathbf{P})$ and $\lambda_{\max}(\mathbf{P})$ are the smallest and largest eigenvalues of the matrix $\mathbf{P} = \mathbf{P}^T \succ 0$.

The enclosure (21) can be refined to

$$\lambda_{\min}(\mathbf{P})\|\mathbf{x}_t\|_2^2 \leq \mathbf{x}_t^T\mathbf{P}\mathbf{x}_t \leq e^{-\gamma t}\lambda_{\max}(\mathbf{P})\|\mathbf{x}(0)\|_2^2 \tag{23}$$

by exploiting the decay rate defined in (16). It immediately leads to the a-priori state enclosures

$$\|\mathbf{x}\|_2 \leq \sqrt{\frac{\lambda_{\max}(\mathbf{P})}{\lambda_{\min}(\mathbf{P})}}e^{-\frac{\gamma}{2}t}\|\mathbf{x}(0)\|_2 \tag{24}$$

that can always be intersected with the exponential enclosures (6) during the iterative computation of the parameters $[\lambda_i]_{(\kappa+1)}$.

Remark. Pessimism in this a-priori enclosure can be reduced by a computation of $\mathbf{P}$ after an (approximate) decoupling of the quasi-linear state equations by means of a suitable change of coordinates.

Remark. Due to the fact that the validated NMPC approach described in further detail in the following section performs a bisection of the control intervals, it is always possible to initialize the exponential enclosures' iteration formula (9) with the result obtained for the non-bisected interval. This is an obvious result of the enclosure property of interval analysis and helps to speed up the iteration procedure.

3 Validated Nonlinear Model Predictive Control

The purpose of this section is to review the work conducted in [4] where a new validated NMPC was developed on the basis of a Runge–Kutta method for validated integration. The approach uses interval analysis tools to compute a guaranteed control sequence over a receding horizon, taking into account the bounded uncertainties in the parameters of the dynamic system model and the measured data. Control intervals are computed in such a way that convergence to the set-point interval is ensured (i.e., $\mathbf{x}_j \rightarrow [\mathbf{x}_\mathrm{r}], \ \ \forall j$, which must be included in the interior of the state box from which the polytopic uncertainty model has been constructed in the previous section), and all the state and input constraints are satisfied (i.e., $\mathbf{x}_i \in [\mathbf{x}_i]$ and $\mathbf{u}_j \in [\mathbf{u}_j], \ \ \forall i, j$). In summary, the proposed guaranteed NMPC encompasses two stages [4],

- **Filtering and branching:** This first step provides a sequence of guaranteed input interval boxes at each time-step k over the prediction horizon N_p, denoted as $[\mathbf{U}_k] = [\mathbf{u}_k] \times [\mathbf{u}_{k+1}] \times \ldots \times [\mathbf{u}_{k+N_\mathrm{p}-1}]$. Branching and filtering procedures allow the computation of safe input intervals along the receding time horizon that satisfy the state constraints (i.e., $\forall j, [\mathbf{x}_j] \subseteq [\mathbf{x}_{\min,j}, \mathbf{x}_{\max,j}]$, where $\mathbf{x}_{\min,j}$ and $\mathbf{x}_{\max,j}$ are the bounds for the admissible domain for each state variable) and ensure convergence to the reference interval[2] (i.e., $[\mathbf{x}_k] \rightarrow [\mathbf{x}_\mathrm{r}]$).

  From the methodological point of view, the state limits should be verified at each validated simulation of the dynamic model using the exponential enclosures technique. If these limits are not satisfied, the initial input interval is further bisected and the validated simulations are relaunched. Subintervals after the bisection are kept according to the following criteria for selection:

  1. A branch leading to unsafe states is eliminated, i.e., if

  $$[\mathbf{x}_{t+T_\mathrm{c}}] \cap [\mathbf{x}_\mathrm{min} \ ; \ \mathbf{x}_\mathrm{max}] = \emptyset;$$

  2. A branch leading to a state far from the reference interval $[\mathbf{x}_\mathrm{r}]$ is also eliminated. The same holds for candidates partially having a cost greater than the other branch(es).

[2] Although only temporally constant reference values are discussed subsequently, time-varying reference trajectories $[\mathbf{x}_{\mathrm{r},k}]$ can be handled by the same algorithm.

- **Interval optimization:** The optimization algorithm computes safe inputs over a finite time horizon by minimizing a newly formulated interval objective function. This function aims to reduce the norm of the input intervals and the error between the predicted and reference outputs as much as possible, resulting in the computation of a sub-optimal input box $[\mathbf{U}]_k^\star$.

4 Application: Control of an Inverted Pendulum

4.1 Dynamic Modeling of the Inverted Pendulum

The control framework based on validated simulations presented in this paper is applied to the stabilization of the nonlinear inverted pendulum shown in Fig. 3. The pendulum is actuated by a DC motor whose angular speed is the input variable.

Following an Euler–Lagrange equation-based modeling procedure as described, for example, in [7], the dynamics of the rotary nonlinear inverted pendulum can be described by the ODEs

$$\begin{cases} \dot{x}_1 = x_2, \\ \dot{x}_2 = \dfrac{1}{\Delta}\Big\{ -\mu_3\cos(x_3)\left(\mu_1\sin(x_3)\cos(x_3)x_2^2 + \Gamma_2 - \mu_g\sin(x_3)\right) + \\ \qquad\qquad \mu_4\left(\mu_3\sin(x_3)x_4^2 - \mu_1\sin(2x_3)x_2x_4 + \Gamma_1\right)\Big\}, \\ \dot{x}_3 = x_4, \\ \dot{x}_4 = \dfrac{1}{\Delta}\Big\{ \left(\mu_1\sin(x_3)^2 + \mu_2\right)\left(\mu_1\cos(x_3)\sin(x_3)x_2^2 + \Gamma_2 - \mu_g\sin(x_3)\right) - \\ \qquad\qquad \mu_3\cos(x_3)\left(\mu_3\sin(x_3)x_4^2 - 2\mu_1\cos(x_3)\sin(x_3)x_2x_4 + \Gamma_1\right)\Big\}, \end{cases} \tag{25}$$

where x_1 and x_2 are, respectively, the angular position and velocity of the rotatory arm. Likewise, x_3 and x_4 are the angle and angular velocity of the pendulum arm. All of these variables are summarized in the state vector $\mathbf{x} = [x_1, x_2, x_3, x_4] \in \mathbb{R}^4$; $\Gamma_i = f_{vi}\dot{x}_i$ is the viscous friction torque at the joint i. Finally, the following state- and parameter-dependent terms are included in the system model (25)

$$\Delta = -\mu_3^2\cos(x_3)^2 + \mu_1\mu_4\sin(x_3)^2 + \mu_2\mu_4,$$

$$\mu_1 = l_{\mathrm{p}}^2\left(\frac{m_{\mathrm{p}}}{4} + M\right), \quad \mu_2 = l_{\mathrm{a}}^2(m_{\mathrm{p}} + M) + \frac{J_{\mathrm{m}}}{N_{\mathrm{g}}^2} + J_{\mathrm{a}},$$

$$\mu_3 = l_{\mathrm{p}}l_{\mathrm{a}}\left(\frac{m_{\mathrm{p}}}{2} + M\right), \quad \mu_4 = l_{\mathrm{p}}^2\left(\frac{m_{\mathrm{p}}}{4} + M\right) + J_{\mathrm{p}}, \quad \mu_{\mathrm{g}} = \left(\frac{m_{\mathrm{p}}}{2} + M\right)l_{\mathrm{p}}g,$$

where m_{a} and J_{a} denote the horizontal arm's mass and its inertia, respectively. Similarly, the mass and inertia of the pendulum arm are given by m_{p} and J_{p}; M is the mass of the load attached to the pendulum arm, J_{m} is the DC motor inertia, and N_{g} its gear ratio; l_{a}, l_{p}, and r_0 are the system's geometric parameters, and g

is the gravitational acceleration. A guaranteed identification of these parameters was performed in [5] with the help of interval methods. The corresponding results are summarized in Tab. 1.

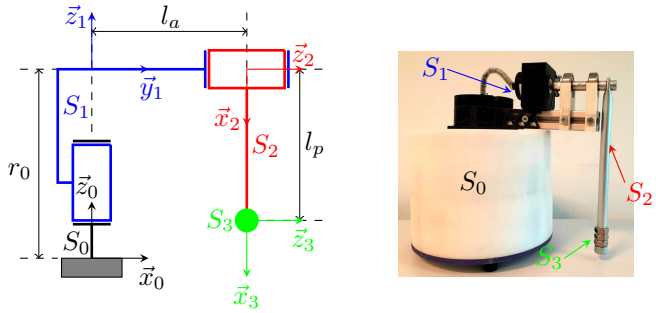


Figure 3: Left: Definition of links frame (configuration $x_1 = x_3 = 0°$); Right: representation of the nonlinear inverted pendulum.

Table 1: Interval bounds for the dynamic parameters of the inverted pendulum [5].

Symbol	**Interval Enclosure**	**Measurement Unit**
μ_1	$[9.63318 \cdot 10^{-4}\ ;\ 1.22604 \cdot 10^{-3}]$	$\mathrm{kg} \cdot \mathrm{m}^2$
μ_2	$[2.19585 \cdot 10^{-3}\ ;\ 2.79471 \cdot 10^{-3}]$	$\mathrm{kg} \cdot \mathrm{m}^2$
μ_3	$[7.227 \cdot 10^{-4}\ ;\ 8.833 \cdot 10^{-4}]$	$\mathrm{kg} \cdot \mathrm{m}^2$
μ_4	$[1.08271 \cdot 10^{-3}\ ;\ 1.37799 \cdot 10^{-3}]$	$\mathrm{kg} \cdot \mathrm{m}^2$
μ_g	$[6.24299 \cdot 10^{-2}\ ;\ 8.44641 \cdot 10^{-2}]$	$\mathrm{kg} \cdot \mathrm{m}^2 \cdot \mathrm{s}^{-2}$
f_{v_1}	$[0.043012\ ;\ 0.13002]$	$\mathrm{Nm} \cdot \mathrm{s} \cdot \mathrm{rad}^{-1}$
f_{v_2}	$[0.000454\ ;\ 0.001174]$	$\mathrm{Nm} \cdot \mathrm{s} \cdot \mathrm{rad}^{-1}$

4.2 Simulation Results

In [4], the NMPC was implemented for the following settings: prediction horizon $N_\mathrm{p} = 10$, control sampling time $T_\mathrm{c} = 16$ ms and the final time of the simulation $T_\mathrm{f} = 0.3$ s. The safety limits of all state variables are: $x_1 \in [-2\pi\ ;\ 2\pi]$, $x_2 \in [-52.4\ \mathrm{rad\ s^{-1}}\ ;\ 52.4\ \mathrm{rad\ s^{-1}}]$, $x_3 \in [-2\pi\ ;\ 2\pi]$, $x_4 \in [-100\ \mathrm{rad\ s^{-1}}\ ;\ 100\ \mathrm{rad\ s^{-1}}]$; the DC motor's torque is limited by the constraints $\tau \in [-8.05\ \mathrm{Nm}\ ;\ 8.05\ \mathrm{Nm}]$. The reference interval of the desired pendulum arm is $x_\mathrm{r} \in [\pi - 0.1\ ;\ \pi + 0.1]$. The tolerance parameter applied for the bisection procedure is adjusted as $tol = 0.25$. Since the filtering and branching algorithm begins from the admissible input domain

$[-8.05\ \text{Nm}\ ;\ 8.05\ \text{Nm}]$, it can lead to approximately 64^{10} branches with $N_\text{p} = 10$. The interval cost function is computed only in the optimization process using the weighting matrices $\mathbf{Q} = \text{diag}\,[1000, 1000, 1000, 1000]$ and $\mathbf{R} = 1$.

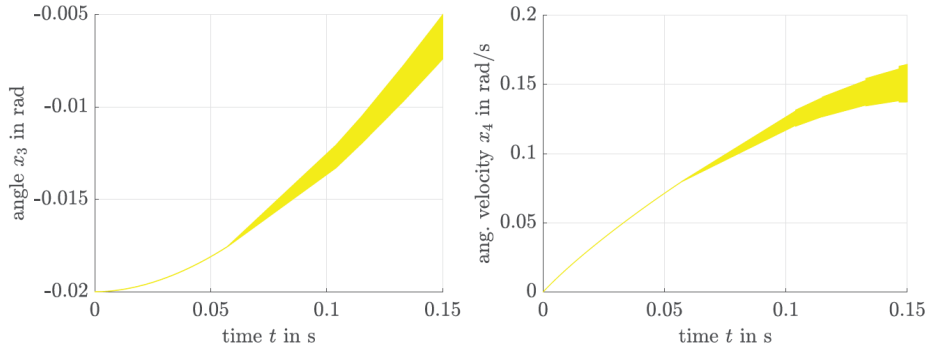


Figure 4: Simulation of the open-loop model using the exponential enclosure approach. Left: Pendulum angle $[\mathbf{x}_3]$; Right: Pendulum arm velocity $[\mathbf{x}_4]$.

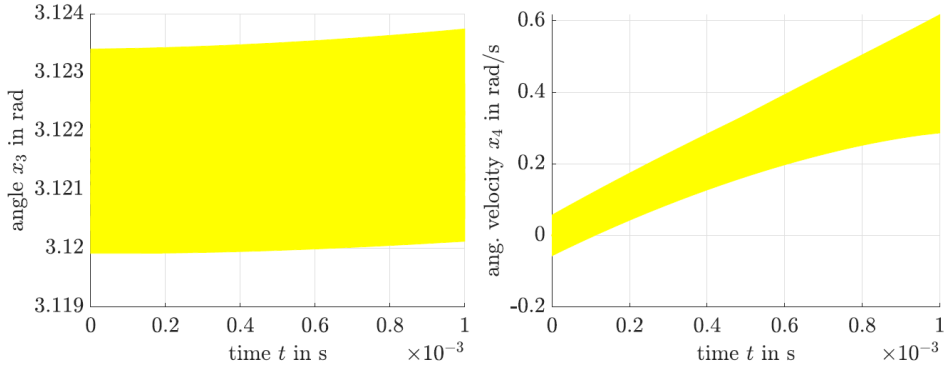


Figure 5: Simulation of the closed-loop model using the exponential enclosure approach. Left: Pendulum angle $[\mathbf{x}_3]$; Right: Pendulum arm velocity $[\mathbf{x}_4]$.

The exponential enclosure approach is now applied to the following two use cases:

1. Simulation of the open-loop dynamics (NMPC-based feedforward controller is switched off) and the pendulum's free motion shall be stabilized at the angle $x_3 = 0$;

2. Simulation of the influence of the subsidiary controller according to Sec. 2.3, after the NMPC has brought the pendulum arm close to its upright position.

For the simulation, the exponential enclosure technique has been implemented in the INTLAB library [20]. Figs. 4 and 5 display the open-loop and closed-loop simulation results of the combination of the exponential state enclosures approach with the control strategy. In Fig. 4, the length of the simulation time horizon is limited to half of the prediction window used for the NMPC.

As it can be seen in Fig. 4, the enclosures of the pendulum angle and its velocity, computed in the open-loop setting, converge to the open-loop stable equilibrium. Moreover, Fig. 5 shows the closed-loop system behavior, for which the computed enclosures of the pendulum angle and its velocity have a tolerable growth of pessimism over time. As already discussed in Sec. 2.3, future work will address the use of parameter-dependent Lyapunov functions for the subsidiary control design to enlarge the region of attraction of the desired equilibrium state and to limit the control effort by an explicit consideration of input and input rate constraints [13].

5 Conclusion and Future Works

This paper focuses on combining a reliable and validated nonlinear model predictive control (NMPC) with an exponential state enclosure technique which is advantageous for the simulation of uncertain dynamic systems with provably asymptotically stable dynamics. The efficiency and robustness of the proposed method were investigated through several numerical simulations using a nonlinear inverted pendulum.

In ongoing works, we focus on studying the following two issues: firstly, the extension of the exponential enclosure technique to ellipsoidal state domain representations, both for real- and complex-valued exponential enclosures, in combination with the extensions of the subsidiary control law discussed in this paper. Secondly, the application of the proposed approach on a real system. For the latter, it will also be required to estimate non-measured system states reliably by means of nonlinear robust observers which exploit interval methods not only to estimate bounds for the states at a specific point in time but also to reconstruct the influence of external disturbances.

References

[1] Alexandre dit Sandretto, J. and Chapoutot, A. DynIBEX: A differential constraint library for studying dynamical systems. In *Conference on Hybrid Systems: Computation and Control (HSCC 2016)*, 2016. https://hal.archives-ouvertes.fr/hal-01297273/file/dynibex-poster.pdf.

[2] Alexandre dit Sandretto, J. and Chapoutot, A. Validated explicit and implicit Runge–Kutta methods. *Reliable Computing electronic edition*, 22, 2016. https://www.cs.utep.edu/interval-comp/reliable-computing-22-pp-079-103.pdf.

[3] Allgöwer, F., Findeisen, R., and Nagy, Z. K. Nonlinear model predictive control: From theory to application. *Journal of the Chinese Institute of Chemical Engineers*, 35(3):299–316, 2004. DOI: 10.6967/JCICE.200405.0299.

[4] Fnadi, M. and Alexandre dit Sandretto, J. Experimental validation of a guaranteed nonlinear model predictive control. *Algorithms*, 14(8):248, 2021. DOI: 10.3390/a14080248.

[5] Fnadi, M., Alexandre dit Sandretto, J., Ballet, G., and Pribourg, L. Guaranteed identification of viscous friction for a nonlinear inverted pendulum through interval analysis and set inversion. In *2021 American Control Conference (ACC)*, pages 3920–3926, New Orleans, LA, USA, 2021. IEEE. DOI: 10.23919/ACC50511.2021.9483185.

[6] Fnadi, M., Du, W., Plumet, F., and Benamar, F. Constrained model predictive control for dynamic path tracking of a bi-steerable rover on slippery grounds. *Control Engineering Practice*, 107:104693, 2021. DOI: 10.1016/j.conengprac.2020.104693.

[7] Gäfvert, M. *Modelling the Furuta pendulum*. Department of Automatic Control, Lund Institute of Technology (LTH), 2016. https://lucris.lub.lu.se/ws/files/4453844/8727127.pdf (accessed: Feb. 12, 2023).

[8] Granvilliers, L. and Benhamou, F. Algorithm 852: Realpaver: An interval solver using constraint satisfaction techniques. *ACM Transactions on Mathematical Software (TOMS)*, 32(1):138–156, 2006. DOI: 10.1145/1132973.1132980.

[9] Henzinger, T. A., Horowitz, B., Majumdar, R., and Wong-Toi, H. Beyond HYTECH: Hybrid systems analysis using interval numerical methods. In *International Workshop on Hybrid Systems: Computation and Control*, pages 130–144. Springer, 2000. DOI: 10.1007/3-540-46430-1-14.

[10] 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.

[11] Kersten, J., Rauh, A., and Aschemann, H. Application-based discussion of verified simulations of interval enclosure techniques. In *2019 24th International Conference on Methods and Models in Automation and Robotics (MMAR)*, pages 209–214, Międzyzdroje, Poland, 2019. DOI: 10.1109/MMAR.2019.8864673.

[12] Khalil, H. K. *Nonlinear systems*. Prentice-Hall, Upper Saddle River, NJ, 3rd edition, 2002. ISBN: 9780130673893.

[13] Lerch, S., Dehnert, R., Rosik, M., and Tibken, B. Minimizing oscillations for magnitude and rate-saturated discrete-time systems by a d_R region pole placement. In *Proceedings of the 10th International Conference on Systems and Control (ICSC)*, pages 244–249, Marseille, France, 2022. DOI: 10.1109/ICSC57768.2022.9993891.

[14] Lydoire, F. and Poignet, P. Nonlinear model predictive control via interval analysis. In *Proceedings of the 44th IEEE Conference on Decision and Control*, pages 3771–3776. IEEE, 2005. DOI: 10.1109/CDC.2005.1582749.

[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] Rauh, A. and Auer, E. Verified simulation of ODEs and DAEs in VALENCIA-IVP. *Reliable Computing*, 15(4):370–381, 2011. https://www.cs.utep.edu/interval-comp/reliable-computing-15-pp-370-381.pdf.

[17] Rauh, A., Bourgois, A., and Jaulin, L. Verifying provable stability domains for discrete-time systems using ellipsoidal state enclosures. *Acta Cybernetica*, 26(2):267–291, 2022. DOI: 10.14232/actacyb.293871.

[18] Rauh, A., Westphal, R., and Aschemann, H. Verified simulation of control systems with interval parameters using an exponential state enclosure technique. In *2013 18th International Conference on Methods & Models in Automation & Robotics (MMAR)*, pages 241–246, Międzyzdroje, Poland, 2013. IEEE. DOI: 10.1109/MMAR.2013.6669913.

[19] Rauh, A., Westphal, R., Aschemann, H., and Auer, E. Exponential enclosure techniques for initial value problems with multiple conjugate complex eigenvalues. In *International Symposium on Scientific Computing, Computer Arithmetic, and Validated Numerics*, pages 247–256. Springer, 2015. DOI: 10.1007/978-3-319-31769-4-20.

[20] Rump, S. INTLAB — INTerval LABoratory. In Csendes, T., editor, *Developments in Reliable Computing*, pages 77–104. Kluver Academic Publishers, 1999. DOI: 10.1007/978-94-017-1247-7_7.

[21] Scherer, C. and Weiland, S. Linear matrix inequalities in control. In Levine, W., editor, *Control System Advanced Methods*, The Electrical Engineering Handbook Series, pages 24–1—24–30. CRC Press, Boca Raton, 2nd edition, 2011. DOI: 10.1201/b10384.

Acta Cybernetica **26** (2024) 855–869.

Set-Valued Approach for the Online Identification of the Open-Circuit Voltage of Lithium-Ion Batteries

Marit Lahme[ab] and Andreas Rauh[ac]

Abstract

To describe the dynamic behavior of lithium-ion batteries using the terminal current and the terminal voltage as input and output of the battery, equivalent circuit models are used, which comprise series resistances, RC sub-networks and a state of charge dependent voltage source. The parameters of the battery model are influenced by aging effects as well as other factors such as the state of charge and the cell temperature. Although those variations can be estimated with the help of an augmented state vector, the typically applied approaches do not allow for a direct identification of nonlinear dependencies of circuit elements on the state of charge or other influence factors. Therefore, a two-stage identification routine for identifying those nonlinear dependencies using an interval observer and an interval contraction scheme is proposed in this paper. The identification routine was successfully applied to identify the open-circuit voltage characteristic of a lithium-ion battery. Numerical simulations are used to evaluate the identification quality of the identification routine.

Keywords: online identification, interval methods, lithium-ion batteries

1 Introduction

The charging/discharging dynamics of lithium-ion batteries can be approximated by using equivalent circuit models. According to [4, 7, 15], these models consist of a finite number of RC sub-networks as well as series resistances and a state of charge (SOC) dependent voltage source which represents the open-circuit voltage. In classical state estimation approaches, the parameters are identified beforehand (cf. [4, 7, 15]). However, the parameters of battery models are subject to aging and temperature induced variations, which is shown in [4]. The aging of battery cells leads to a loss of the total capacity, an increasing Ohmic cell resistance and changes in the charging/discharging efficiency as well as changes of the delay parameters of the aforementioned RC sub-networks. Additionally, many degradation mechanisms

[a] Carl von Ossietzky Universität Oldenburg, Distributed Control in Interconnected Systems
[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.301185

of lithium-ion batteries that lead to capacity or power fading are highly dependent on the cell temperature [3]. Many of these parameter variations can be mapped onto the open-circuit voltage and estimated during system operation with the help of an augmented state vector, but this approach does not allow for estimating nonlinear functional dependencies of the circuit elements on the SOC or other influence factors. The above-mentioned variations can alternatively be detected offline using electrochemical impedance spectroscopy (EIS), as mentioned in [1, 12, 13]. The EIS method is to apply a sinusoidal current (sinusoidal voltage) to the battery over a wide frequency range and to measure the terminal voltage (terminal current). The spectroscopic features of the resulting impedance spectrum can be assigned to the components of the corresponding equivalent circuit model. The impedance is affected by a variety of factors such as the cell temperature, aging, or the SOC. Accordingly, variations resulting from these influence factors can be detected using EIS. The drawback of this method is that it takes a long time typically in the range of hours to days because of the gradual charging/discharging process and resting periods that might be necessary.

To address these difficulties, we propose a two-stage identification of nonlinear dependencies in this paper with the dependency of the open-circuit voltage on the SOC as an example. This two-stage identification is based on interval analysis, which has already been used successfully in parameter estimation, for example in [2]. Therefore, it is a promising method for this online identification routine. The state variables of the dynamic system are estimated in the first stage with an interval observer. In the second stage, the a-priori knowledge of the open-circuit voltage characteristic is corrected using the estimated state variables. For the identification of the nonlinear dependency of the open-circuit voltage on the SOC with underlying aging and temperature induced variations, it is assumed that the other equivalent circuit parameters are known and not yet affected by aging.

This paper is structured as follows. Section 2 outlines the modeling of lithium-ion batteries based on equivalent circuit models. In Section 3, the design of the interval observer is described which is used for the identification routine shown in Section 4. The results of the numerical simulation for evaluating the performance of this set-valued identification approach are presented in Section 5. The paper is concluded with a brief summary of the proposed identification routine as well as with an outlook on future work in Section 6.

Notation. In this paper, matrices and vectors are denoted by bold letters to distinguish them from scalar variables. The notations $\underline{\mathbf{M}}$ and $\overline{\mathbf{M}}$ for an interval matrix $\mathbf{M}$ denote the element-wise lower and upper bounds.

2 Equivalent Circuit Model of Lithium-Ion Batteries

In this paper, an equivalent circuit model containing one series resistance and two RC sub-networks representing processes with short (τ_{TS}) and large (τ_{TL}) time con-

stants resulting from polarization effects and concentration losses is considered as shown in Fig. 1 [4, 15]. Here, v_{OC} is the open-circuit voltage. The terminal current i_{T} is used as the input to charge or discharge the battery and the terminal voltage v_{T} can be measured. The instantaneous voltage drop of the terminal voltage is caused by the serial resistance R_{S}. The two RC sub-networks consisting of C_{TS}, C_{TL}, R_{TS}, and R_{TL} are used to describe the transient behavior. First, the modeling of lithium-ion batteries is introduced with point-valued state variables and crisp equivalent circuit parameters. At the end of this section, the model is extended to interval values.

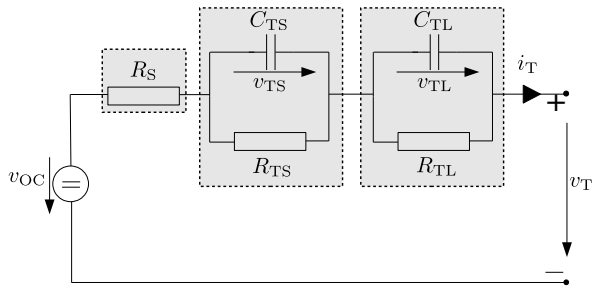


Figure 1: Equivalent circuit model of a lithium-ion battery (modified from [15]).

The SOC $\sigma(t)$ and the voltages across the RC sub-networks $v_{\mathrm{TS}}(t)$ and $v_{\mathrm{TL}}(t)$ are chosen as the state variables. Hence, the state vector

$$\mathbf{x}(t) = \begin{bmatrix} \sigma(t) & v_{\mathrm{TS}}(t) & v_{\mathrm{TL}}(t) \end{bmatrix}^T \tag{1}$$

and the quasi-linear, continuous time state equations

$$\begin{aligned} \dot{\mathbf{x}}(t) &= \mathbf{A}\left(\sigma(t)\right) \cdot \mathbf{x}(t) + \mathbf{b}\left(\sigma(t)\right) \cdot u(t) \\ &= \begin{bmatrix} 0 & 0 & 0 \\ 0 & \frac{-1}{C_{\mathrm{TS}}(\sigma(t)) \cdot R_{\mathrm{TS}}(\sigma(t))} & 0 \\ 0 & 0 & \frac{-1}{C_{\mathrm{TL}}(\sigma(t)) \cdot R_{\mathrm{TL}}(\sigma(t))} \end{bmatrix} \cdot \mathbf{x}(t) + \begin{bmatrix} \frac{-1}{C_{\mathrm{Bat}}} \\ \frac{1}{C_{\mathrm{TS}}(\sigma(t))} \\ \frac{1}{C_{\mathrm{TL}}(\sigma(t))} \end{bmatrix} \cdot u(t) \end{aligned} \tag{2}$$

are obtained, where the system input is given as the terminal current $u(t) := i_{\mathrm{T}}(t)$. The SOC as well as $v_{\mathrm{TS}}(t)$ and $v_{\mathrm{TL}}(t)$ can be point-valued or interval values. The first state equation represents the integrating behavior

$$\dot{\sigma}(t) = -\frac{i_{\mathrm{T}}(t)}{C_{\mathrm{Bat}}} \tag{3}$$

between the SOC $\sigma(t) \in [0\,;\,1]$ and the charging/discharging current $i_{\mathrm{T}}(t)$. Here, the SOC is a normalized value, so that $\sigma = 0$ corresponds to the completely discharged battery and $\sigma = 1$ represents the fully charged battery with the nominal

capacitance C_{Bat}. The other two state equations represent the first-order lag behavior of the RC sub-networks

$$\dot{v}_\iota(t) = \frac{-v_\iota(t)}{C_\iota(\sigma(t)) \cdot R_\iota(\sigma(t))} + \frac{i_\mathrm{T}(t)}{C_\iota(\sigma(t))} \ , \quad \iota \in \{\mathrm{TS}, \mathrm{TL}\} \tag{4}$$

where the equivalent circuit parameters and the delay parameters are dependent on the SOC according to

$$R_\iota(\sigma(t)) = R_{\iota\mathrm{a}} \cdot e^{R_{\iota\mathrm{b}} \cdot \sigma(t)} + R_{\iota\mathrm{c}} \ , \quad \iota \in \{\mathrm{TS}, \mathrm{TL}\} \ , \tag{5}$$

$$C_\iota(\sigma(t)) = C_{\iota\mathrm{a}} \cdot e^{C_{\iota\mathrm{b}} \cdot \sigma(t)} + C_{\iota\mathrm{c}} \ , \quad \iota \in \{\mathrm{TS}, \mathrm{TL}\} \ , \tag{6}$$

$$\tau_\iota(\sigma(t)) = C_\iota(\sigma(t)) \cdot R_\iota(\sigma(t)) \ , \quad \iota \in \{\mathrm{TS}, \mathrm{TL}\} \ . \tag{7}$$

The SOC-dependent parameters have been identified in [17] using experimental parameter identification. The identified parameters are shown in Tab. 1. By applying Kirchhoff's voltage law, the terminal voltage is obtained as

$$v_\mathrm{T}(t) = v_\mathrm{OC}(\sigma(t)) - v_\mathrm{TS}(t) - v_\mathrm{TL}(t) - i_\mathrm{T}(t) \cdot R_\mathrm{S}(\sigma(t)) \ , \tag{8}$$

with the Ohmic resistance

$$R_\mathrm{S}(\sigma(t)) = R_{\mathrm{Sa}} \cdot e^{R_{\mathrm{Sb}} \cdot \sigma(t)} + R_{\mathrm{Sc}} \ . \tag{9}$$

Based on the experimental parameter identification in [17], the open-circuit voltage is assumed to be represented by the nonlinear expression

$$v_\mathrm{OC}(\sigma(t)) = v_0 \cdot e^{v_1 \cdot \sigma(t)} + \sum_{i=0}^{3} v_{i+2} \cdot \sigma^i(t) \ , \tag{10}$$

which can be rewritten in the quasi-linear form

$$\tilde{v}_\mathrm{OC}(\sigma(t)) = \eta_\mathrm{OC}(\sigma(t)) \cdot \sigma(t) = v_\mathrm{OC}(\sigma(t)) - v_0 - v_2 \tag{11}$$

$$= \left(v_0 \cdot \frac{e^{v_1 \cdot \sigma(t)} - 1}{\sigma(t)} + v_3 + v_4 \cdot \sigma(t) + v_5 \cdot \sigma^2(t) \right) \cdot \sigma(t) \tag{12}$$

by subtracting the constant, state-independent terms from the expression for the open-circuit voltage [15]. Using (8) and (12), the output equation can be obtained as

$$\begin{aligned} \mathrm{y}(t) &= \tilde{v}_\mathrm{T}(t) = \mathbf{C}(\sigma(t)) \cdot \mathbf{x}(t) + \mathbf{D}(\sigma(t)) \cdot i_\mathrm{T}(t) \\ &= \tilde{v}_\mathrm{OC}(t) - v_\mathrm{TS}(t) - v_\mathrm{TL}(t) - i_\mathrm{T}(t) \cdot R_\mathrm{S}(t) \ , \end{aligned} \tag{13}$$

and can be rewritten in the quasi-linear input-independent form

$$\begin{aligned} \mathrm{y}^*(t) &= \mathrm{y}(t) - \mathbf{D}(\sigma(t)) \cdot i_\mathrm{T}(t) = \mathbf{C}(\sigma(t)) \cdot \mathbf{x}(t) \\ &= \begin{bmatrix} \eta_\mathrm{OC}(\sigma(t)) & -1 & -1 \end{bmatrix} \cdot \mathbf{x}(t) \in [\mathrm{y_m}] \ , \end{aligned} \tag{14}$$

with the output matrix $\mathbf{C}(\sigma(t)) = \begin{bmatrix} \eta_{\mathrm{OC}}(\sigma(t)) & -1 & -1 \end{bmatrix}$ (which is a row vector in the special case of this paper) and the feedthrough matrix $\mathbf{D}(\sigma(t)) = -R_{\mathrm{S}}(\sigma(t))$. In (14), $\mathrm{y}^*(t)$ represents the measurement that is provided to the observer. It is the terminal voltage of the battery, adjusted in a way that it is expressed according to the quasi-linear output equation. Furthermore, the measurement noise (assumed to be bounded) has to be considered, so that $\mathrm{y}^*(t)$ is an element of the measurement interval $[\mathrm{y_m}]$, defined in (22). Outliers, for example measurements that are wrong, may affect the estimation quality of the interval observer designed in Section 3 and therefore may also affect the identification quality of the open-circuit voltage characteristic. In this paper, we assume that there are no outliers, but managing outliers can be done in future work with applying relaxed set inversion techniques as shown in [2] and [8].

To implement the proposed identification approach, the continuous time system is temporally discretized with a constant step size T and approximated with the help of the explicit Euler method. Because of the small sampling time T compared to the delay parameters τ_{TS} and τ_{TL}, this method is sufficiently accurate. The discretization errors are assumed to be included in the measurement uncertainty. With $\mathbf{x}_k$ and σ_k approximating the exact state values $\mathbf{x}(t_k)$ and $\sigma(t_k)$ in the discretization points, the discretized state equations are then obtained as

$$\mathbf{x}_{k+1} = \mathbf{A}_{\mathrm{d}}(\sigma_k) \cdot \mathbf{x}_k + \mathbf{b}_{\mathrm{d}}(\sigma_k) \cdot u_k \ , \tag{15}$$

$$\mathbf{A}_{\mathrm{d}}(\sigma_k) = \mathbf{I}_{3\times 3} + T \cdot \mathbf{A}(\sigma_k) \ , \tag{16}$$

$$\mathbf{b}_{\mathrm{d}}(\sigma_k) \cdot u_k = T \cdot \mathbf{b}(\sigma_k) \cdot u_k \ . \tag{17}$$

In this paper, interval state variables are considered, so that $\mathbf{x}(t) \in [\underline{\mathbf{x}}(t)\,;\,\overline{\mathbf{x}}(t)]$ is a vector consisting of intervals for each component. Therefore, also the equivalent circuit parameters become intervals, which leads to interval-valued matrices $\mathbf{A}(\sigma(t)) \in \left[\underline{\mathbf{A}}(\sigma(t))\,;\,\overline{\mathbf{A}}(\sigma(t))\right]$, $\mathbf{b}(\sigma(t)) \in \left[\underline{\mathbf{b}}(\sigma(t))\,;\,\overline{\mathbf{b}}(\sigma(t))\right]$, $\mathbf{C}(\sigma(t)) \in \left[\underline{\mathbf{C}}(\sigma(t))\,;\,\overline{\mathbf{C}}(\sigma(t))\right]$ and $\mathbf{D}(\sigma(t)) \in \left[\underline{\mathbf{D}}(\sigma(t))\,;\,\overline{\mathbf{D}}(\sigma(t))\right]$. This also applies to the discretized system.

3 Design of the Interval Observer

The proposed identification approach for nonlinear dependencies of the circuit elements on state variables is a two-stage procedure. In the first stage, state estimation is performed with an interval observer in each time step. Therefore, the most recent estimate for the state vector is used which consists of intervals for each component. The estimated values are then used in the second stage to correct the a-priori knowledge for the open-circuit voltage characteristic. The estimation of the state variables and the open-circuit voltage is shown in Fig. 2.

A prerequisite for this identification approach is that all state variables can be estimated. Hence, the system has to be observable or identifiable. The observer gain matrix $\mathbf{H}$ is designed in such a way that the stability is ensured for the assumed open-circuit voltage characteristic. The system matrix $\mathbf{A}(\sigma(t))$ from equation (2) has the following sign pattern

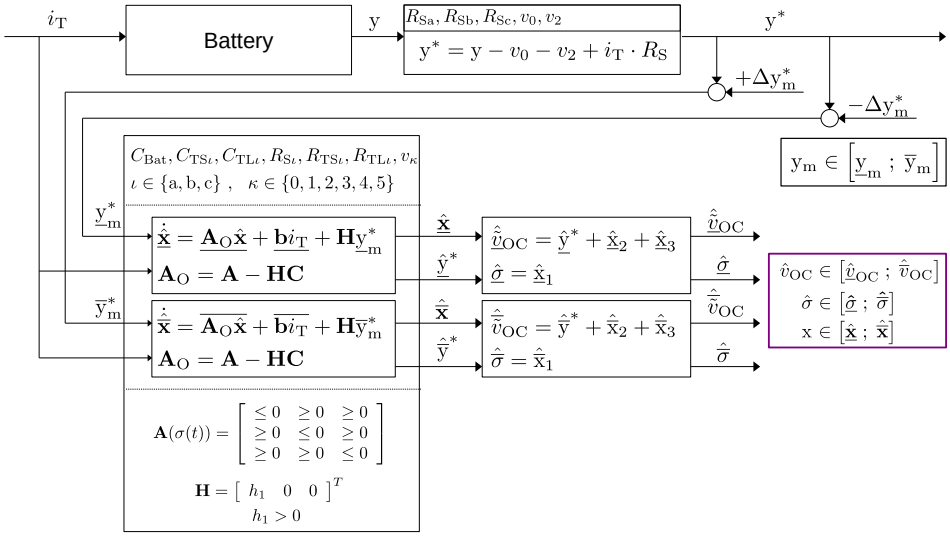


Figure 2: Estimation of the open-circuit voltage and the state of charge.

$$\mathbf{A}\left(\sigma(t)\right) = \begin{bmatrix} \leq 0 & \geq 0 & \geq 0 \\ \geq 0 & \leq 0 & \geq 0 \\ \geq 0 & \geq 0 & \leq 0 \end{bmatrix} \in \left[\underline{\mathbf{A}} \; ; \; \overline{\mathbf{A}}\right] \quad . \tag{18}$$

Based on the design of a robust interval observer shown in [7], the observer matrix $\mathbf{H}$ is hereby assigned as

$$\mathbf{H} = \begin{bmatrix} h_1 & 0 & 0 \end{bmatrix}^T \quad , \quad h_1 > 0 \; , \tag{19}$$

where h_1 is a scalar.

With the bounding systems $\mathbf{x} \in \left[\underline{\mathbf{x}} \; ; \; \overline{\mathbf{x}}\right]$ for the true states and $\hat{\mathbf{x}} \in [\hat{\mathbf{x}}] := \left[\underline{\hat{\mathbf{x}}} \; ; \; \overline{\hat{\mathbf{x}}}\right]$ for their estimates, respectively, $\mathbf{x}$ is given as $\mathbf{x} \in \left[\underline{\hat{\mathbf{x}}} \; ; \; \overline{\hat{\mathbf{x}}}\right]$ and the interval observer is obtained according to [7] and [14] as

$$\underline{\mathbf{A}_\mathrm{O}\hat{\mathbf{x}}} + \underline{\mathbf{bu}} + \mathbf{H}\underline{\mathrm{y}}_\mathrm{m} \leq \dot{\hat{\mathbf{x}}} \leq \overline{\mathbf{A}_\mathrm{O}\hat{\mathbf{x}}} + \overline{\mathbf{bu}} + \mathbf{H}\overline{\mathrm{y}}_\mathrm{m} \quad , \; \hat{\mathbf{x}} \in [\hat{\mathbf{x}}] \tag{20}$$

with the observer system matrix

$$\mathbf{A}_\mathrm{O} = \mathbf{A} - \mathbf{HC} \quad \in \quad \left[\underline{\mathbf{A}}_\mathrm{O} \; ; \; \overline{\mathbf{A}}_\mathrm{O}\right] \tag{21}$$

and the uncertain measurements

$$[\mathrm{y}_\mathrm{m}] := \left[\underline{\mathrm{y}}_\mathrm{m} \; ; \; \overline{\mathrm{y}}_\mathrm{m}\right] = \mathrm{y}_\mathrm{m} + [-\Delta\mathrm{y}_\mathrm{m} \; ; \; \Delta\mathrm{y}_\mathrm{m}] \quad . \tag{22}$$

The observer is parameterized in a cooperativity preserving way [6, 9]. Cooperative dynamic systems have state trajectories that are monotonic with respect to

the initial conditions. Lower and upper bounds for the states of uncertain cooperative systems can be calculated by solving two independent initial value problems, one for each bound. Hence, the computational effort to determine these lower and upper bounds is reduced in comparison to a non-cooperative model. A system

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t)), \quad \mathbf{x} \in \mathbb{R}^n, \tag{23}$$

has to fulfill the following two conditions to be cooperative. All off-diagonal elements of its Jacobian

$$\mathbf{J} = \frac{\partial \mathbf{f}(\mathbf{x})}{\partial \mathbf{x}} \tag{24}$$

have to be non-negative

$$J_{i,j} \geq 0, \quad i,j \in \{1,...,n\}, \quad i \neq j \tag{25}$$

and all state variables have to be non-negative [6, 9]

$$\mathbf{x} \in \mathbb{R}_{\geq 0} \ . \tag{26}$$

Due to the design of the observer gain matrix, the observer system matrix $\mathbf{A}_\mathrm{O}$ and the system matrix $\mathbf{A}$ have the same sign pattern. Therefore, adding this interval observer to a cooperative system preserves the property of cooperativity.
For cooperative systems, guaranteed state enclosures are found by considering $\underline{\mathbf{A}}_\mathrm{O}\hat{\underline{\mathbf{x}}}$ and $\overline{\mathbf{A}}_\mathrm{O}\hat{\overline{\mathbf{x}}}$ for the lower and upper bound respectively in equation (20), if all state variables are element wise non-negative. The system in equation (2), however, does not satisfy the conditions (25) and (26). Hence, it is not cooperative in this simplest form [6]. Therefore, guaranteed lower and upper bounds for all state variables of the system (2) have to be calculated according to Müller's theorem [5], without taking advantage of the property of cooperativity. This is done by considering the infimum (supremum) of $\mathbf{A}_\mathrm{O}\hat{\mathbf{x}}$ for the lower bound (upper bound) in equation (20).

Like the continuous time system (2), the observer is also discretized with a constant step size T and with the help of the explicit Euler method. This leads to the following equations:

$$\hat{\mathbf{x}}_{k+1} = \mathbf{A}_\mathrm{Od}\left(\hat{\sigma}_k\right) \cdot \hat{\mathbf{x}}_k + \mathbf{b}_\mathrm{d}\left(\hat{\sigma}_k\right) \cdot u_k + \mathbf{H}_\mathrm{d} \cdot \mathrm{y}_{\mathrm{m},k} \ , \tag{27}$$

$$\mathbf{A}_\mathrm{Od}\left(\hat{\sigma}_k\right) = \mathbf{I}_{3\times 3} + T \cdot \mathbf{A}_\mathrm{O}\left(\hat{\sigma}_k\right) \ , \tag{28}$$

$$\mathbf{b}_\mathrm{d}\left(\hat{\sigma}_k\right) \cdot u_k = T \cdot \mathbf{b}\left(\hat{\sigma}_k\right) \cdot u_k \ , \tag{29}$$

$$\mathbf{H}_\mathrm{d} \cdot \mathrm{y}_{\mathrm{m},k} = T \cdot \mathbf{H} \cdot \mathrm{y}_{\mathrm{m},k} \ . \tag{30}$$

In order to guarantee stability for the discretized observed system, the magnitudes of all eigenvalues of the matrix $\mathbf{A}_\mathrm{Od}$ have to be less than one. To preserve the derivation of lower and upper bounding systems, the system matrix in (28) needs to be element wise non-negative. The evaluation of this discretized model is performed in analogy to equation (17) in [16].

4 Identification of the open-circuit voltage

With the help of interval methods, the nonlinear dependency of the open-circuit voltage on the SOC can be identified as visualized in Fig. 3. During the charging/discharging process of the battery, the values of the open-circuit voltage and the state of charge are estimated by the interval observer resulting in axis-aligned interval boxes for the SOC and the open-circuit voltage. An interval box $[\mathbf{\Gamma}]$ is defined as the Cartesian product of closed intervals. Here, $[\mathbf{\Gamma}]$ is an element of $\mathbb{IR}^2$ and is defined as $[\mathbf{\Gamma}] = [\sigma(t)] \times [\tilde{v}_{\mathrm{OC}}(t)]$ with $\sigma(t) \in [\underline{\sigma}(t)\,;\,\overline{\sigma}(t)]$ and $\tilde{v}_{\mathrm{OC}}(t) \in \left[\underline{\tilde{v}}_{\mathrm{OC}}(t)\,;\,\overline{\tilde{v}}_{\mathrm{OC}}(t)\right]$. The open-circuit voltage is calculated based on the estimated state variables and equation (14)

$$\tilde{v}_{\mathrm{OC}}(t) = \mathrm{y}_{\mathrm{m}}^{*}(t) + v_{\mathrm{TS}}(t) + v_{\mathrm{TL}}(t) \quad . \tag{31}$$

The intersections of those interval boxes are utilized to improve the approximation of the true $v_{\mathrm{OC}}(\sigma(t))$ characteristic.

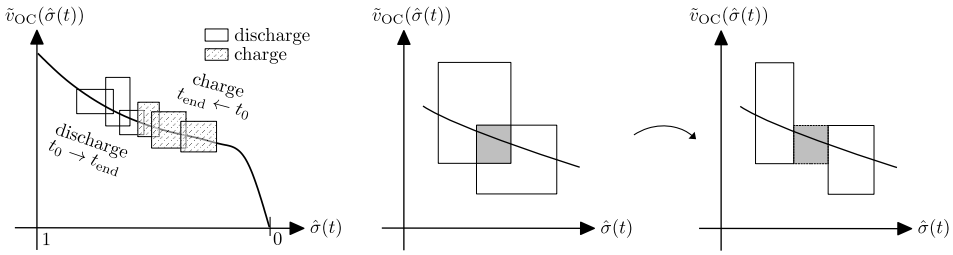


Figure 3: Identification of nonlinear dependencies using interval methods.

To keep the computational effort that is required for this intersection process feasible, merging strategies are necessary. A certain number of overlapping interval boxes are combined with the help of their convex interval hull as shown in Fig. 4. The resulting convex hulls are intersected with each other. The convex interval hull of two interval boxes $[\mathbf{\Gamma}_1]$ and $[\mathbf{\Gamma}_2]$ is denoted by

$$[\mathbf{\Gamma}_{\mathrm{hull}}] = [\mathbf{\Gamma}_1] \cup [\mathbf{\Gamma}_2] \quad . \tag{32}$$

The overapproximation of two interval boxes with the convex interval hull results in an overestimation of the magnitude

$$\delta_{hull} = \frac{area\{[\mathbf{\Gamma}_{\mathrm{hull}}]\} - (area\{[\mathbf{\Gamma}_1]\} + area\{[\mathbf{\Gamma}_2]\}) + area\{[\mathbf{\Gamma}_1] \cap [\mathbf{\Gamma}_2]\}}{area\{[\mathbf{\Gamma}_1]\} + area\{[\mathbf{\Gamma}_2]\} - area\{[\mathbf{\Gamma}_1] \cap [\mathbf{\Gamma}_2]\}} \cdot 100\% \quad , \tag{33}$$

where the area of an interval box is equal to the product of the interval widths of all components [10]. The overestimation for combining more than two interval boxes with the help of the corresponding convex interval hull can be calculated by

recursively applying equation (33), where δ_{hull} represents the exact overestimation, when two interval boxes are combined. Otherwise, it represents the upper bound on the overestimation. During the numerical simulation shown in this paper, the number of interval boxes that are approximated with the convex interval hull is defined beforehand. At the end of the simulation, the overestimation δ_{hull} is calculated for each convex interval hull.

Remark. Instead of using a preset, constant number of interval boxes that should be merged before the intersection process, a limit on the overestimation can be introduced. In this case, the interval boxes are recursively merged as long as the overestimation remains below the threshold $\delta_{hull} \leq \delta_{hull,max}$.

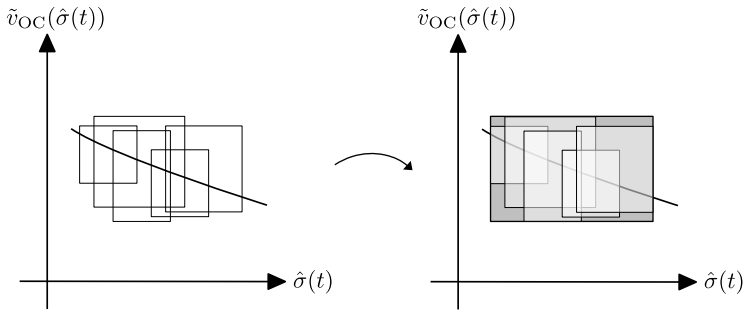


Figure 4: Combining interval boxes with the help of their convex interval hull.

5 Numerical Simulation

To evaluate the identification quality of the proposed identification routine, a numerical simulation was performed for the initial state $\mathbf{x}_0 = \begin{bmatrix} 0.9 & 0 & 0 \end{bmatrix}^T$ and the system parameters shown in Table 1. The discretization step size and the observer gain matrix are chosen as $T = 10\,\mathrm{ms}$ and $\mathbf{H} = \begin{bmatrix} 0.2 & 0 & 0 \end{bmatrix}^T$. The magnitude of the bounded measurement noise is set to $\Delta \mathrm{y_m} = 2.5\,\mathrm{mV}$. The measurements are generated in a simulation using y and uniformly distributed random numbers in the interval of $[-\Delta \mathrm{y_m}\ ;\ \Delta \mathrm{y_m}]$. The input current has an amplitude of 5 A and a period length of 1 h and is given as shown in Fig. 5. To reduce the uncertainty and improve the estimation results, the input current consists of an alternating sequence of sine half-waves and periods where the current is constant and equal to zero. The number of interval boxes that are combined with the help of their convex interval hull is chosen as 6000. This corresponds to merging the interval boxes in a time span of one minute and leads to an overestimation of each interval hull of less than 3.73 %. The true values are calculated based on point-valued state variables, point-valued equivalent circuit parameters and the system parameters shown in

Table 1: System parameters according to the experimental parameter identification in [17].

C_{Bat}	3.100	Ah	R_{TSa}	1	Ω	R_{TLa}	0.01	Ω	v_0	−1	V
			R_{TSb}	−30		R_{TLb}	−4		v_1	−23	
R_{Sa}	0.25	Ω	R_{TSc}	0.015	Ω	R_{TLc}	0.05	Ω	v_2	3.255	V
R_{Sb}	−20		C_{TSa}	−900	F	C_{TLa}	25000	F	v_3	0.8342	V
R_{Sc}	0.07	Ω	C_{TSb}	−2		C_{TLb}	−2		v_4	−0.2905	V
			C_{TSc}	1000	F	C_{TLc}	2000	F	v_5	0	V

Tab. 1. The simulation is implemented in MATLAB. The single core simulation of the interval observer and the intersection process is approximately four times faster than real time, so that this implementation is applicable for online identification purposes.

Figs. 6 and 7 show the estimation results for the three state variables. During the simulation, the estimation uncertainty increases but can be reduced by the periods of constant current, because the integrating behavior between the SOC and the terminal current is the main contribution to the uncertainty. It can only be reduced if the constant current is equal to zero. Otherwise, the uncertainty increases instead of being reduced. The uncertainty of the voltage v_{TS} decreases faster than the uncertainty of the voltage v_{TL} because of the smaller delay parameter τ_{TS} compared to the delay parameter τ_{TL}. The frequency of the input current also affects the estimation uncertainty. This is shown in Figs. 8 and 9. Here, the input currents $i_{\mathrm{T}}(t) = 5\,\mathrm{A} \cdot \sin\left(\frac{2\pi}{3600}t\right)$ and $i_{\mathrm{T}}(t) = 5\,\mathrm{A} \cdot \sin\left(\frac{10\pi}{3600}t\right)$ are compared. With increasing frequency, the estimation uncertainty decreases. However, the achievable SOC range also decreases, which in turn influences the identification process.

The identification result is shown in Fig. 10. Fig. 10(a) shows the identification result after the first sine half-wave of the input current (discharging process). It is obvious that the increasing uncertainty leads to an increasing width of the interval boxes. As mentioned before, the uncertainty is then reduced due to the constant current period so that, during the following charging process (second sine half-wave), the width of the interval boxes is further reduced (Fig. 10(b)). Here, the identification result of the discharging process serves as the a-priori knowledge for the characteristic to be identified. The true value of the $v_{\mathrm{OC}}(\sigma(t))$ characteristic can be approximated well in the range of the SOC that can be achieved during both discharging and charging processes.

6 Conclusions and Future Work

In this paper, a set-valued approach for the online identification of state-dependent characteristics was presented. This identification routine consists of two stages. At first, the state variables of the dynamic system are estimated with an interval

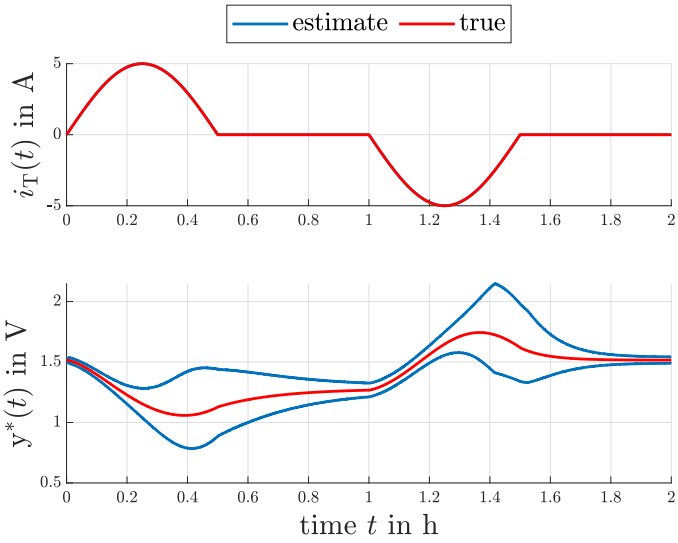


Figure 5: True output value $y^*(t)$ in comparison with the estimated lower and upper bounds resulting from the terminal current $i_T(t)$.

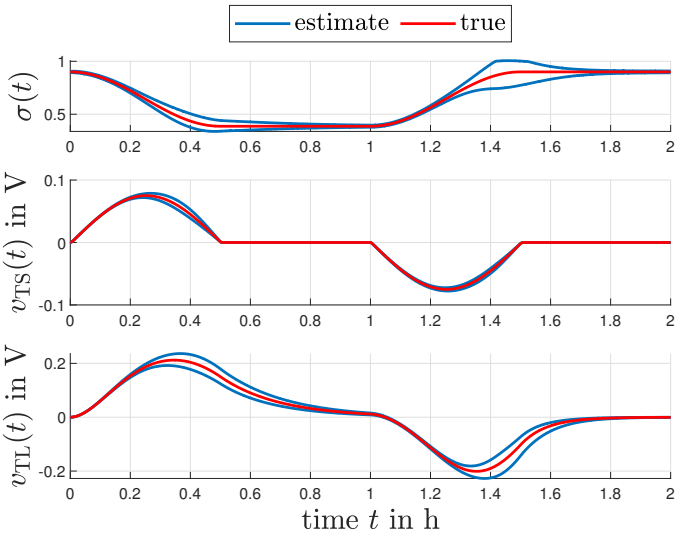
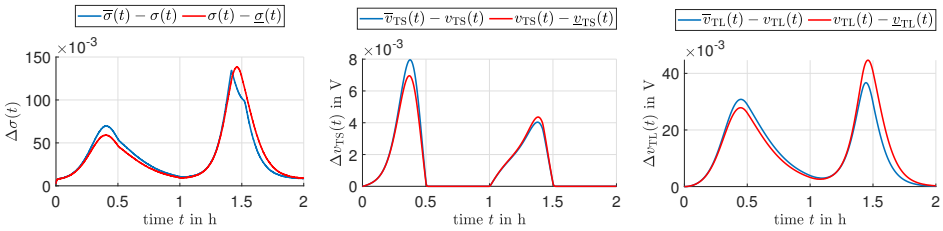
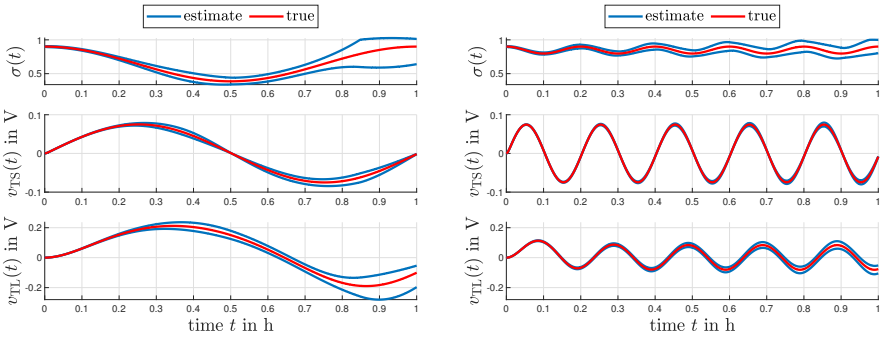


Figure 6: True value of the state variables in comparison with the estimated lower and upper bounds.



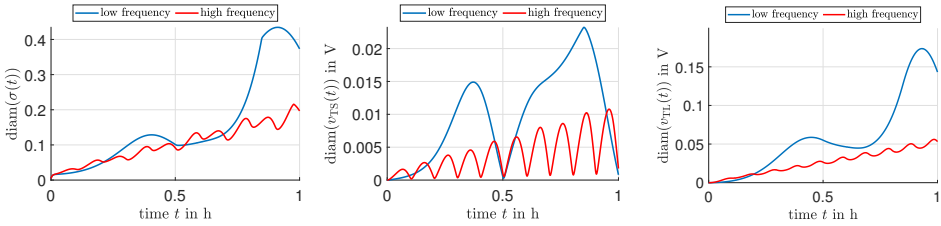
(a) Estimation error of $\sigma(t)$. (b) Estimation error of $v_{\mathrm{TS}}(t)$. (c) Estimation error of $v_{\mathrm{TL}}(t)$.

Figure 7: Estimation errors of the state variables.



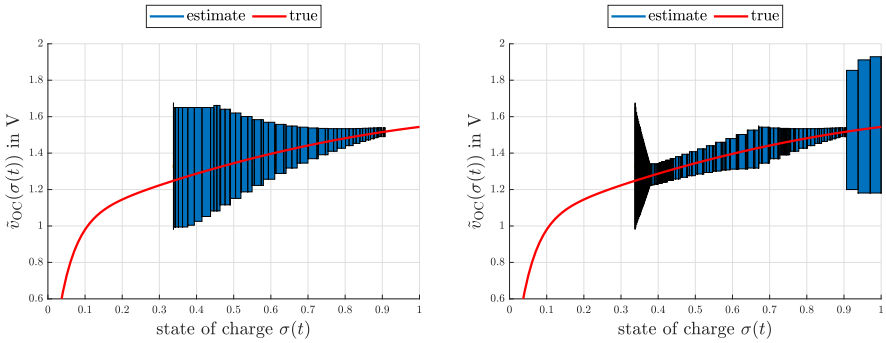
(a) Low frequency input current. (b) High frequency input current.

Figure 8: Comparison of the estimated state variables resulting from low or high frequency input currents.



(a) Interval diameters of $\sigma(t)$. (b) Interval diameters of $v_{\mathrm{TS}}(t)$.(c) Interval diameters of $v_{\mathrm{TL}}(t)$.

Figure 9: Comparison of the interval diameters of the estimated state variables resulting from the input currents according to Fig. 8.



(a) Discharge process (first sine half-wave). (b) Charge process (second sine half-wave).

Figure 10: Approximation of the $v_{OC}(\sigma(t))$ characteristic.

observer, afterwards the estimated values are used to correct an a-priori knowledge for the characteristic to be identified. The proposed identification routine was successfully applied to the identification of the open-circuit voltage characteristic of a lithium-ion battery.

The estimation quality has a major influence on the identification results. It is therefore necessary to reduce the estimation uncertainty as much as possible. As shown above, the input current strongly affects the estimation quality. So the design of optimal experiments by tuning the input current is very important. Furthermore, the design of the interval observer can be optimized to improve the estimation results. For example, a cascaded observer design or a TNL observer design as shown in [18] could be investigated.

In addition, our future work will deal with the extension of this approach to the identification of characteristics that depend on multiple variables (e.g. the open-circuit voltage characteristic of lithium-ion batteries that depends on the SOC and the temperature) and with the investigation of possible combinations of set-valued and stochastic approaches (cf. [11]).

References

[1] Barai, A., Uddin, K., Widanage, W. D., McGordon, A., and Jennings, P. A study of the influence of measurement timescale on internal resistance characterisation methodologies for lithium-ion cells. *Scientific Reports*, 8(1):21, 2018. DOI: 10.1038/s41598-017-18424-5.

[2] dit Sandretto, J. A., Trombettoni, G., Daney, D., and Chabert, G. Certified calibration of a cable-driven robot using interval contractor programming. In Thomas, F., editor, *Computational Kinematics*, Volume 15 of *Mechanisms and Machine Science Series*, pages 209–217. Springer Netherlands, Dordrecht, 2014. DOI: 10.1007/978-94-007-7214-4_24.

[3] Edge, J. S., O'Kane, S., Prosser, R., Kirkaldy, N. D., Patel, A. N., Hales, A., Ghosh, A., Ai, W., Chen, J., Yang, J., Li, S., Pang, M.-C., Bravo Diaz, L., Tomaszewska, A., Marzook, M. W., Radhakrishnan, K. N., Wang, H., Patel, Y., Wu, B., and Offer, G. J. Lithium ion battery degradation: What you need to know. *Physical Chemistry Chemical Physics (PCCP)*, 23(14):8200–8221, 2021. DOI: 10.1039/d1cp00359c.

[4] Erdinç, O., Vural, B., and Uzunoğlu, M. A dynamic lithium-ion battery model considering the effects of temperature and capacity fading. In *Proceedings of International Conference on Clean Electrical Power (ICCEP)*, pages 383–386. IEEE, 2009. DOI: 10.1109/ICCEP.2009.5212025.

[5] Gennat, M. and Tibken, B. Guaranteed bounds for uncertain systems: Methods using linear Lyapunov-like functions, differential inequalities and a midpoint method. In *Proceedings of the 12th GAMM — IMACS International Symposium on Scientific Computing, Computer Arithmetic and Validated Numerics (SCAN)*, page 17. IEEE, 2006. DOI: 10.1109/SCAN.2006.21.

[6] Gennat, M. and Tibken, B. Computing guaranteed bounds for uncertain cooperative and monotone nonlinear systems. In *Proceedings of the 17th IFAC World Congress*, pages 4846–4851. IFAC Proceedings Volumes, 2008. DOI: 10.3182/20080706-5-KR-1001.00814.

[7] Hildebrandt, E., Kersten, J., Rauh, A., and Aschemann, H. Robust interval observer design for fractional-order models with applications to state estimation of batteries. In *Proceedings of the 21st IFAC World Congress*, pages 3683–3688. IFAC-PapersOnLine, 2020. DOI: 10.1016/j.ifacol.2020.12.2052.

[8] 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.

[9] Kersten, J., Rauh, A., and Aschemann, H. Transformation of uncertain linear fractional order differential equations into a cooperative form. In *Proceedings of the 11th IFAC Symposium on Nonlinear Control Systems (NOLCOS)*, pages 646–651. IFAC-PapersOnLine, 2019. DOI: 10.1016/j.ifacol.2019.12.035.

[10] Krasnochtanova, I., Rauh, A., Kletting, M., Aschemann, H., Hofer, E. P., and Schoop, K.-M. Interval methods as a simulation tool for the dynamics of biological wastewater treatment processes with parameter uncertainties. *Applied Mathematical Modelling*, 34(3):744–762, 2010. DOI: 10.1016/j.apm.2009.06.019.

[11] Lahme, M. and Rauh, A. Combination of stochastic state estimation with online identification of the open-circuit voltage of lithium-ion batteries. In *Proceedings of the 1st IFAC Workshop on Control of Complex Systems (COSY)*. IFAC-PapersOnLine, 2022. DOI: 10.1016/j.ifacol.2023.01.055.

[12] Meddings, N., Heinrich, M., Overney, F., Lee, J.-S., Ruiz, V., Napolitano, E., Seitz, S., Hinds, G., Raccichini, R., Gaberšček, M., and Park, J. Application of electrochemical impedance spectroscopy to commercial Li-ion cells: A review. *Journal of Power Sources*, 480:228742, 2020. DOI: 10.1016/j.jpowsour.2020.228742.

[13] Middlemiss, L. A., Rennie, A. J., Sayers, R., and West, A. R. Characterisation of batteries by electrochemical impedance spectroscopy. In *Proceedings of the 4th Annual CDT Conference in Energy Storage & Its Applications*, pages 232–241. Energy Reports, 2019. DOI: 10.1016/j.egyr.2020.03.029.

[14] Raïssi, T. and Efimov, D. Some recent results on the design and implementation of interval observers for uncertain systems. *at - Automatisierungstechnik*, 66(3):213–224, 2018. DOI: 10.1515/auto-2017-0081.

[15] Rauh, A., Chevet, T., Dinh, T. N., Marzat, J., and Raïssi, T. Robust iterative learning observers based on a combination of stochastic estimation schemes and ellipsoidal calculus. In *Proceedings of the 25th International Conference on Information Fusion (FUSION)*, pages 1–8. IEEE, 2022. DOI: 10.23919/FUSION49751.2022.9841329.

[16] Rauh, A. and Kersten, J. Transformation of uncertain linear systems with real eigenvalues into cooperative form: The case of constant and time-varying bounded parameters. *Algorithms*, 14(3):85, 2021. DOI: 10.3390/a14030085.

[17] Reuter, J., Mank, E., Aschemann, H., and Rauh, A. Battery state observation and condition monitoring using online minimization. In *Proceedings of the 21st International Conference on Methods and Models in Automation and Robotics (MMAR)*, pages 1223–1228. IEEE, 2016. DOI: 10.1109/MMAR.2016.7575313.

[18] 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.

Acta Cybernetica **26** (2024) 871–887.

The Codac Library: A Catalog of Domains and Contractors

Simon Rohou[ab], Benoît Desrochers[ac], and Fabrice Le Bars[ad]

Abstract

Codac (Catalog Of Domains And Contractors) is a C++/Python library providing tools for constraint programming over reals, trajectories and sets. It has many applications in parameter estimation, guaranteed integration or robot localization and provides reliable outputs by computing sets of feasible solutions according to the constraints defining the problem. This paper provides a brief overview of the library and its Contractor Network approach, illustrated on a convincing robotic application.

Keywords: constraint programming, interval analysis, state estimation, dynamical systems, solver, robotics, Contractor Network, SLAM

1 Introduction

This paper provides an overview of the Codac library[1] (http://codac.io), that aims at providing a catalog of tools based on interval analysis and constraint programming. The toolbox allows to approximate feasible solutions of non-linear and/or differential systems. Since the solution of these complex systems cannot generally be calculated exactly, Codac uses numerical analysis to compute the bounds of sets of feasible solutions. The assets are *guarantee* (computations are guaranteed to never lose solutions, due to the rigorous interval arithmetic) and *exhaustiveness* (if multiple values are possible, all of them are characterized). In the same way, obtaining an empty set allows to safely disprove properties of a system. In Codac, the variables can be of different types, such as reals, vectors [2], trajectories [35], uncertain sets [9], graphs [18], *etc.*, in order to address a wide range of problems.

Intervals are used to reliably propagate uncertainties (from sensors, models, discretizations), even in the case of non-linearities, provided that they are bounded. Coupled with constraint programming, these methods have been shown to be effective for solving complex problems involving constraints that are generally difficult to

[a]ENSTA Bretagne, Lab-STICC, UMR CNRS 6285, Brest, France
[b]E-mail: simon.rohou@ensta-bretagne.fr, ORCID: 0000-0001-6232-9918
[c]E-mail: benoit.desrochers@ensta-bretagne.org, ORCID: 0000-0001-7910-1119
[d]E-mail: fabrice.le_bars@ensta-bretagne.fr, ORCID: 0000-0001-9413-4621
[1]Codac has been built upon two former libraries: Tubex and pyIbex.

DOI: 10.14232/actacyb.302772

handle (strong uncertainties, differential equations, temporal delays [39], indistinguishable data, inter-temporal measurements [36], hybrid systems, *etc.*). Classical robotic applications such as Simultaneous Localization And Mapping (SLAM), the kidnapped robot problem, or the exploration of unstructured environments, can be formalized as such systems, and are still challenging issues. Conventional methods such as Kalman or particle filters are commonly used for tackling these problems. However, they have limitations related to the context of use, the propagation of error distributions, the reliability of the outputs, or the involved equations describing the system. Besides, constraint programming coupled with interval methods allows to handle a wider class of systems and has been shown to be comfortable with solving several problems known to be difficult, in a very few steps with the simplicity offered by constraint programming [7].

Recent advances in interval methods have been done by the community, and the Codac library gathers a part of related state-of-the-art implementations with the objective to make them easy to combine. This paper provides an overall picture of the library and proposes an application on an academic problem of SLAM.

2 Constraint programming

Constraint programming is a paradigm in which users concentrate on the properties of a solution to be found (*e.g.* the pose of a robot, the location of a landmark, the orbit of a satellite) by stating constraints on variables. Constraints usually come from the equations of the problem, inequalities, or measurements from sensors. The variables appear in the equations. Then, a solver performs *constraint propagation* on the variables and reliably provides feasible solutions corresponding to the problem. In this approach, the user concentrates on *what* the problem is instead of *how* to solve it, thus leaving the computer to deal with the *how*. The strength of this declarative paradigm lies in its simplicity, as it allows one to describe a complex problem without requiring knowledge of resolution tools and their specific parameters to tune. The second asset is genericity: a situation is seen from a high-level point of view and this abstraction enables the resolution of a wide range of problems. The energy is mainly spent on the description of the problem.

Since several decades, the community of constraint programming has brought numerous contributions for dealing with discrete problems. The Prolog language [3] appears to be one of the most famous outcomes with free implementations available to the community. Classical applications of these developments lie in automated planning or interactive theorem proving. While a major effort from the community has been undertaken around this concept, other studies appeared in order to tackle continuous problems with this paradigm [17]. For both discrete and continuous problems, constraint programming can be applied by defining a Constraint Network involving variables $\mathcal{V}_i$, domains $\mathcal{D}_i$, and constraints $\mathcal{L}_j$ [25].

Variables In continuous problems, the members of a system, including the unknown solutions, are reals $x \in \mathbb{R}$ or vectors $\mathbf{x} \in \mathbb{R}^n$. Recent advances have led to

the extension of Constraint Networks in order to tackle a larger number of types of variables, such as sets $\mathbb{X} \in \mathcal{P}(\mathbb{R}^n)$ [11], graphs [18], paths [23], *etc.* In particular, dynamical systems can be drawn as Constraint Networks by introducing so-called trajectories variables, denoted by $\mathbf{x}(\cdot) : \mathbb{R} \to \mathbb{R}^n$, thus allowing to consider the solution of a dynamical system as a single and continuous item.

Domains A variable $\mathcal{V}_i$ is known to be enclosed in some domain $\mathcal{D}_i$ on which we will apply constraints $\mathcal{L}_j$ *via* some operators. Domains define non-empty ranges of feasible values. For instance, domains can be intervals, polytopes, ellipsoids, subpavings, tubes, *etc.*

Constraints Elementary facts and rules apply on variables: so-called constraints. There are very few restrictions on the forms of the constraints: they are understood as the expression of any relation that binds variables, which are known to belong to some domains. In the continuous and differential context, constraints may be non-linear equations such as $x_3 = \cos(x_1 + \exp(x_2))$, inequalities, quantified parameters [13], differential systems expressed as $\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t))$, *etc.* For instance, in the context of robotics, constraints will come from state equations, numerical models, or measurements. Uncertainties from sensors are specified either by other constraints (*e.g.* $x \geqslant 0$) or by restricting the domains of the variables (*e.g.* $[x] = [0, \infty]$, where $[x]$ is the interval domain known to enclose the variable x).

Constraint propagation Elementary constraints are relations that cannot be decomposed, such as $c = a + b$. Then, complex constraints can be considered by combining simpler constraints, in order to increase in complexity, while preserving simplicity. This leads to a propagation process, due to dependencies between constraints sharing the same variables.

The aim of Codac is to easily deal with constraints in order to eventually characterize sets of values compliant with the defined rules. This is done by providing implementations of operators for elementary constraints as well as algorithms for their combination. The related domains and operators depend on tools from interval analysis.

3 Interval methods for constraint programming

In the constraint programming approach, the estimation of a variable consists in reducing its domain. The obtained set is said to be reliable: the resolution must guarantee that no solution will be lost during the solving process, according to the constraints defining the problem. In practice, domains and constraints have to be numerically computed. The use of interval methods is perfectly suited for this, providing intervals for domains and a rigorous arithmetic to implement constraints.

Intervals domains An interval $[x]$ is a closed and connected subset of $\mathbb{R}$. The set of all intervals is denoted $\mathbb{IR}$. An interval vector (a box) $[\mathbf{x}]$ of $\mathbb{IR}^n$ is an axis-aligned box, a closed and connected subset of $\mathbb{R}^n$. These interval sets can be easily represented in a computer. For instance, a box $[\mathbf{x}] = [\mathbf{x}^-, \mathbf{x}^+]$ will be defined by its two bounds $\mathbf{x}^-$ and $\mathbf{x}^+$, that are representable vectors of $\mathbb{R}^n$. Intervals can be extended to trajectories: we then define a *tube* $[\mathbf{x}](\cdot) : \mathbb{R} \to \mathbb{IR}^n$ as an interval of two trajectories $\mathbf{x}^-(\cdot)$ and $\mathbf{x}^+(\cdot)$ such that $[\mathbf{x}](\cdot) = [\mathbf{x}^-(\cdot), \mathbf{x}^+(\cdot)]$. They have also been extended to sets with *thicksets* denoted by $[\![\mathbb{X}]\!] = [\mathbb{X}^-, \mathbb{X}^+]$ where $\mathbb{X}^\pm \in \mathcal{P}(\mathbb{R}^n)$.

Contractors A contractor $\mathcal{C}$ for a constraint $\mathcal{L}$ is an operator designed to reduce a domain without losing any solution consistent with $\mathcal{L}$. A contractor is thus an algorithm that can act on an interval domain for narrowing its bounds in a reliable way. The following definition applies for contractors on boxes [7], and can be easily extended to tubes, thicksets, *etc.*

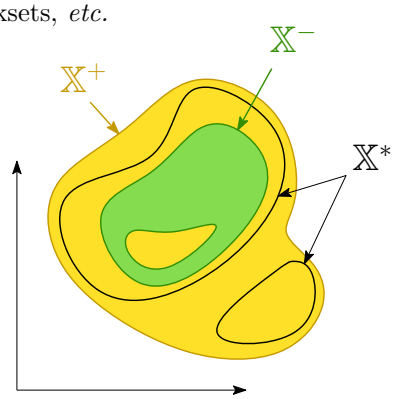


Figure 1: A contractor $\mathcal{C}_\mathbb{X}$ related to a constraint representing a set $\mathbb{X}$ is applied on several boxes. Hatched parts correspond to vectors that are removed after the contraction.

Definition. *A contractor $\mathcal{C}$, associated with a constraint $\mathcal{L}$, on a box $[\mathbf{x}] \in \mathbb{IR}^n$ is a mapping from $\mathbb{IR}^n$ to $\mathbb{IR}^n$ such that:*

$$(i) \quad \forall [\mathbf{x}] \in \mathbb{IR}^n,\ \mathcal{C}([\mathbf{x}]) \subseteq [\mathbf{x}], \qquad \textit{(contraction)}$$

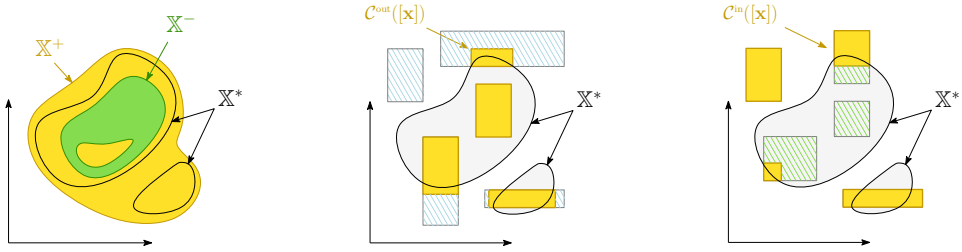
$$(ii) \quad \begin{pmatrix} \mathcal{L}(\mathbf{x}) \\ \mathbf{x} \in [\mathbf{x}] \end{pmatrix} \Longrightarrow \mathbf{x} \in \mathcal{C}([\mathbf{x}]). \quad \textit{(consistency)}$$

The use of contractors allows to enclose complex algorithms in simple black boxes that are only used to contract a domain according to a constraint, in a reliable way. The reliable property (expressed by the *consistency* rule of Definition 3) is important as it allows to combine contractors, call them in any order and as much as necessary, without running the risk of losing solutions. This allows to deal with complex problems providing that contractors are at hand and sufficient to address elementary constraints obtained from a decomposition. Finally, domains

and contractors can be combined in propagation algorithms in order to implement an interval solver.

One may emphasize that interval methods have the reputation of being limited to problems of low dimensions, due to usual bisections of the domains which leads to an exponential complexity order. Contractors may overcome this problem: they are usually given by polynomial-time algorithms and can be employed without performing bisections, which allows to tackle problems of higher dimensions.

Separators A contractor $\mathcal{C}^{\text{out}}$ associated with a set $\mathbb{X}$ aims at removing infeasible solutions (*i.e.* vectors that are not in $\mathbb{X}$) from a domain. When employed in a branch-and-contract algorithm, it allows to compute an outer approximation $\mathbb{X}^+ \supset \mathbb{X}$. However, the same contractor cannot be used for inner approximations $\mathbb{X}^-$, that are sets in which any vector is solution, that is $\mathbb{X}^- \subset \mathbb{X}$. For instance, when no contraction happens on a given box, *i.e.* $\mathcal{C}^{\text{out}}([\mathbf{x}]) = [\mathbf{x}]$, it is not possible to know if $[\mathbf{x}]$ is completely included in $\mathbb{X}$ (see the yellow box on Fig. 1) or if there may exist vectors in $[\mathbf{x}]$ that do not belong to $\mathbb{X}$ (see the red dot in Fig. 1). Inner approximations are particularly important for proving the existence of solutions. Their computation can be done by using a contractor $\mathcal{C}^{\text{in}}$ consistent with the complementary $\overline{\mathbb{X}}$ that removes vectors of $\mathbb{X}$, as depicted in Fig. 2. The pair $\{\mathcal{C}^{\text{in}}, \mathcal{C}^{\text{out}}\}$ defines a new operator called a *separator* [19], which can be employed in a branch-and-contract algorithm in order to characterize simultaneously an inner $\mathbb{X}^-$ and an outer approximation $\mathbb{X}^+$ and therefore enclose $\mathbb{X}$ in a *thickset* $[\![\mathbb{X}]\!] = [\mathbb{X}^-, \mathbb{X}^+]$.



(a) $\mathbb{X}^* \in [\mathbb{X}^-, \mathbb{X}^+]$. (b) Illustration of $\mathcal{C}^{\text{out}}$. (c) Illustration of $\mathcal{C}^{\text{in}}$.

Figure 2: Enclosure of a set $\mathbb{X}^* \subset \mathbb{R}^2$ by two approximated inner and outer sets $\mathbb{X}^-$ and $\mathbb{X}^+$, computable using a pair of contractors (a separator) involved in a branch-and-contract algorithm. Note that any vector of $\mathbb{X}^-$ also belongs to $\mathbb{X}^+$. For computing $\mathbb{X}^+$, resp. $\mathbb{X}^-$, a $\mathcal{C}^{\text{out}}$ contractor, resp. $\mathcal{C}^{\text{in}}$, can be employed as pictured in Subfig. 2b–2c. The green hatched areas are part of $\mathbb{X}^-$ while the blue ones belong to $\overline{\mathbb{X}^+}$. In practice, these algorithms output subpavings made of non-overlapping boxes, not represented in Subfig. 2a.

Interval domains, contractors and separators form a set of items that one can combine in order to describe continuous nonlinear equations, dynamical systems, or measurements, that are usually encountered in physics or robotics. They are the basic components of the Codac library.

4 The Codac library

4.1 A framework of domains and contractors

The API of Codac can be broken down into three layers: (*i*) a list of implemented domains such as intervals, boxes, tubes, thicksets, *etc.*; (*ii*) a catalog of contractors and separators for dealing with a wide variety of constraints; (*iii*) a top-level system solver called Contractor Network. Recent efforts from the community [30, 39, 19] have led to new domains, contractors and separators. The objective of Codac is to gather the related implementations and form a catalog of algorithms associated with publications from the literature.

4.2 Other libraries

Several libraries, see for instance filib++ [27], MPFI [32] and GAOL [14], provide low-level features related to interval arithmetic, most of them including reliable numerical operations with outward rounding. For its interval arithmetic computations, Codac currently stands on GAOL that has shown good performances and a large portability on operating systems. At a higher level of abstraction, the contractor programming approach [7] deployed in Codac is also a cornerstone of the IBEX library [8]. Codac is, however, not limited to constraint processing over reals and is inspired by robotic applications, including the capacity to deal with dynamical systems, inter-temporal constraints and thicksets.

The guaranteed simulation of dynamical systems is also the topic of several set-based libraries, namely Vnode [26], Cosy [31], DynIBEX [1] or CAPD [41]. These libraries have applications in robotics and automatic control, by verifying dynamical properties of non-linear systems [28] or for the computation of reachable sets [15]. They are also used by mathematicians to prove conjectures [12]. While they offer good performances for guaranteed integrations, they are mainly suited for systems expressed under the form of Initial Value Problems (IVPs), that is $\{\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}),\ \mathbf{x}(0) \in [\mathbf{x}_0]\}$, which does not represent the diversity of state estimation problems. This limitation motivated new constraint-based approaches in order to assess for instance inter-temporal relations, time uncertainties, delays, or hybrid systems. Robotics also requires to mix various uncertainties, not only related to dynamical systems, but also involving sets, graphs, paths, to name but a few. Nonetheless, future work may focus on building bridges between these libraries in order to benefit from good performances from each specific tool. The contractor framework could ease the use of these state-of-the-art contributions by enclosing them in contractor operators compatible with each other.

Finally, the above-mentioned libraries are mainly available in C++. The core of Codac is also developed in C++ for performance and historical reasons, and in order to be compliant with many robotic frameworks. In addition, a Python wrapper of the library allows to use Codac in Python 3 while benefiting from C++ performance. The library is actively supported on Ubuntu derivatives, Windows and macOS, with pre-built packages available.

4.3 Implemented domains

The building blocks of this library are intervals and the above-mentioned domains (boxes, tubes, thicksets, *etc.*) are mainly designed as compositions of intervals. However, for specific computation needs and in order to avoid as much as necessary wrapping effects, that are pessimism drawbacks induced by the use of inaccurate enclosures, some implementations in Codac rely on more specific domains, such as cuboids [24], polytopes [40] or ellipsoids [21, 29]. For instance, it has been shown in [33] that the integration of continuous-time linear systems could be computed exactly without pessimism by using polytopes instead of boxes, at the expense of longer computation times. A good trade-off has been studied in [30] with the use of ellipsoids for enclosing the continuous states. These are specific domains also available in Codac, and relevant for guaranteed integration purposes.

Domains for trajectories: tubes $\mathbb{X}(\cdot)$ In Codac, a tube is implemented as a temporal sequence of sets $\mathbb{X}_i$, where the $\mathbb{X}_i$s can be intervals, boxes, ellipsoids [30], polytopes, subpavings, or any domain[2] for which set operations can be computed. More precisely, a tube $\mathbb{X}(\cdot)$ with a sampling time $\delta > 0$ is implemented as a set-valued function which is constant for all t inside intervals $[k\delta, k\delta + \delta]$, $k \in \mathbb{N}$. The set $[k\delta, k\delta + \delta] \times \mathbb{X}(t_k)$, with $t_k \in [k\delta, k\delta + \delta]$, is called the k^{th} *slice* of the tube $\mathbb{X}(\cdot)$ and is denoted by $\mathbb{X}(k)$. For instance, when the $\mathbb{X}_i$s correspond to intervals, the implementation amounts to an interval of step functions $[\underline{x^-}(\cdot), \overline{x^+}(\cdot)]$ such that $\forall t,\ \underline{x^-}(t) \leqslant \overline{x^+}(t)$, as pictured by Fig. 3.

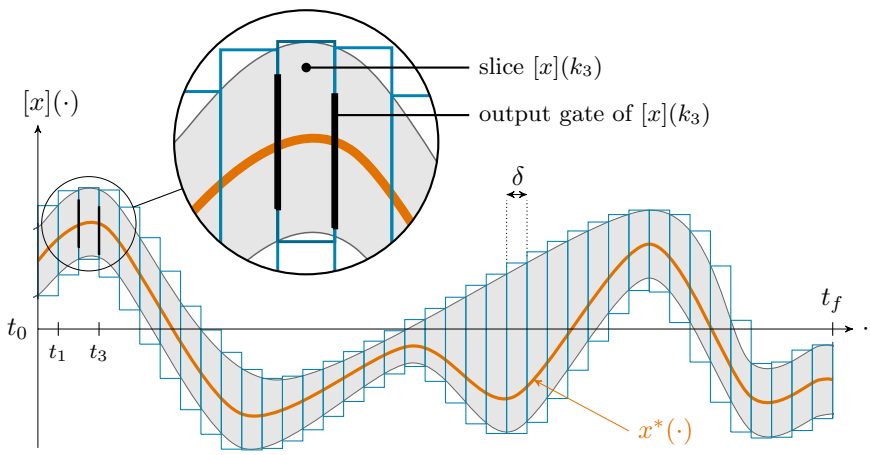


Figure 3: An interval-tube $[x](\cdot)$ implemented as a list of interval-slices. In practice, the sampling δ is not necessarily constant.

[2]Generic programming is enabled for tubes thanks to their C++ implementation providing compile-time templates.

This implementation then takes rigorously into account floating point precision when building a tube, thanks to reliable numerical libraries. Further computations involving $\mathbb{X}(\cdot)$ will be based on its slices, thus giving an outer approximation of the actual solution set. For instance, the lower bound of the integral of a boxed-tube $[\mathbf{x}](\cdot)$ is simply computed as the signed area of the region in the tx-plane that is bounded by the graph of $\underline{\mathbf{x}}^-(t)$ and the t-axis. The lower the slice width δ, the higher the precision of the approximation. Note that the current implementation allows a variable sampling, as well as input and output *gates* on each slice as pictured in Fig. 3 in the case of interval slices. In the literature, these gates are classical restrictions used when dealing with IVPs.

Other representations of tubes are available in Codac, and provide faster evaluations than with a simple sequence of slices. For instance, the computer evaluation $\big[\mathbb{X}([t])\big] = \big[\,\{x(t) \mid x(\cdot) \in \mathbb{X}(\cdot), t \in [t]\}\,\big]$, with $[t]$ a large interval over several slices, can be optimized with a redundant data structure such as binary trees [36, pp. 54–55] or polynomial approximations of the bounds of the tubes. This is relevant when many evaluations have to be performed on tubes that have non-updated values, namely integral calculations or tube inversions [35].

Domains for sets: thicksets $[\![\mathbb{X}]\!]$ A set $\mathbb{X} \subset \mathbb{R}^n$ can be bracketed between two sets $\mathbb{X}^-$ and $\mathbb{X}^+$, forming a so-called thickset $[\![\mathbb{X}]\!] = [\mathbb{X}^-, \mathbb{X}^+]$, [11]. The approximation of $\mathbb{X}^-$ and $\mathbb{X}^+$ is possible using subpavings [20] that are unions of non-overlapping boxes of $\mathbb{R}^n$. As for tubes, optimized implementations based on binary trees allow to speed up evaluations of subpavings. Fig. 4 illustrates an application of Codac involving tubes and thicksets for the computation of the guaranteed zone explored by a robot [9]. This example typically illustrates the need to couple various domains for robotic applications.

4.4 Contractors and separators

A list of contractors (or separators for approximating sets) is available in the library. They aim at contracting domains in a reliable way and do not need to be configured. Most of them implement elementary constraints. Other contractors are compositions of primitive contractors based on syntax trees, see for instance the HC4revise contractor [2] available in IBEX and Codac. This allows one to obtain a high-level contractor built from an analytical expression, and automatically involving primitive contractors. Besides, avoiding constraint decompositions can provide better results if one relies on a dedicated and optimized contractor. For instance, in the case of the polar equation ($x = \rho\cos(\theta)$, $y = \rho\sin(\theta)$), the contractor $\mathcal{C}_{\text{polar}}([x],[y],[\rho],[\theta])$ provided by [10] allows a minimal contraction for $[x]$, $[y]$, $[\rho]$ and $[\theta]$. Additional contractors are designed to deal with inter-temporal and differential constraints, such as linear systems $\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u}$ ([33]), temporal delays $x(t) = y(t-a)$ ([39]), time uncertainties $\{y = x(t), t \in [t]\}$ ([35]), differential nonlinear equations $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x})$ ([5, Chap. 4]), to name but a few. Other contractors focus on geometric constraints [16], or allow contractions robust to outliers [6].

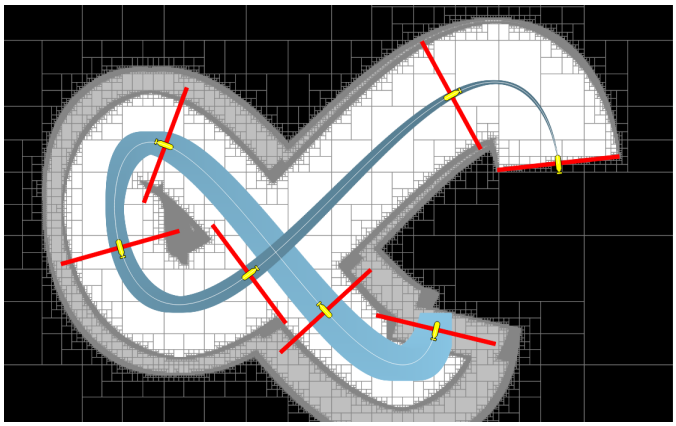


Figure 4: Guaranteed zone explored by a robot [9]. A robot equipped with a side-scan-sonar (the scope of which is pictured by red lines) evolves along a trajectory $\mathbf{x}^*(\cdot)$ (white line) estimated with uncertainties (blue tube $[\mathbf{x}](\cdot)$). The problem consists in computing the explored space $\mathbb{X}$. Taking into account the increasing uncertainty of localization, $\mathbb{X}$ cannot be computed exactly. However, an inner set $\mathbb{X}^-$ (in white) can be approximated, and corresponds to the part that has been surely observed with the sonar. Also, the black set (*i.e.* $\overline{\mathbb{X}^+}$) is computable and related to parts of the environment for which we can reliably state that no observation has been done, for any feasible trajectory $\mathbf{x}(\cdot) \in [\mathbf{x}](\cdot)$. The gray part corresponds to the *penumbra* in which observations may have been done, or not.

5 Contractor Networks (CN)

5.1 Higher degree of abstraction

When several contractors are at stake, there may be interactions between them: a contraction from one contractor may trigger another one, which reveals a constraint propagation process [4, 7]. It becomes necessary to call some contractors several times in order to converge to the best contraction of the domains. This number of contracting iterations cannot be known in advance. It depends on the contractors at stake, their efficiency and their sequencing. Classically, one can implement a loop of contractions in order to process the contractors as long as there is a contraction on one of the domains. The iteration stops when a fixed point has been reached: when nothing can be contracted anymore. Note that because a computer uses floating point numbers, the iterative fixed point will be necessarily reached in a finite number of steps [22, p. 42]. In practice, we may stop the iteration as soon as the contractions are not significant anymore. In any case, even if the algorithm stops before reaching the fixed point, the actual solution will always be enclosed in the domains.

Codac provides a high-level propagation tool, called a Contractor Network (CN),

that aims at managing the propagation process automatically. This simplifies the use of contractors: the user does not have to implement contracting loops and to manage stopping conditions. Instead, he/she only has to design a CN by connecting contractors and domains together. This approach allows to take a higher degree of abstraction by hiding the propagation part. What remain are the domains and contractors, which correspond to the variables and their related rules, as in a pure declarative paradigm.

5.2 Refined propagation process

In addition, the propagation process can be enhanced: some heuristic encoded in the CNs can allow a better sequence of the contractors calls. While the user only states the relations between contractors and domains, the CN defines by itself the sequence of contractions to be run, depending of the types of domains and contractors and some empirical models encoded in the library. The result is a global contraction that runs faster than with a random sequence of contractors.

Numerically, a contractor on a complex domain (such as a tube, a thickset) may amount to several contractors on subdomains (such as a slice, a box). For instance, the $\mathcal{C}_+$ contractor for the constraint $a = b + c$ can be extended to tubes: $\mathcal{C}_+\big([x_1](\cdot), [x_2](\cdot), [x_3](\cdot)\big)$, [35]. This does not correspond to inter-temporal or differential equations: the related constraint applies for all t. This amounts to calling $\mathcal{C}_+$ for all tuples of slices $\big([x_1](k), [x_2](k), [x_3](k)\big)$. In practice, some parts of the tube may not be updated during a propagation and it becomes relevant to avoid further calls of contractors on the involved slices, if we can state that they will be ineffective. While it would be cumbersome to tune a propagation algorithm at this level of granularity, it becomes worthwhile to rely on an automatic tool that will call the contractors only on relevant parts of the complex domains. Hence, CNs are also used to break tubes down into graphs of slices, each of them being connected to the former contractors related to the tube itself. The propagation algorithm then naturally calls the contractors if necessary. It leads to faster computations. To our knowledge, this is the first time that contractors are involved in a propagation network with heterogeneous domains such as tubes or thicksets.

5.3 Graphs of contractors and domains

A Contractor Network is a graph of domains and contractors. Fig. 5 provides an illustration of such graph, with domains pictured by circles and contractors drawn by boxes. A possible propagation sequence is the following: assume that $\mathcal{C}_3$ is first triggered, either manually or because the contractor has been newly added in the graph. $\mathcal{C}_3$ is added in a stack of contractors $\mathscr{L}$ that was empty so far. Then $\mathcal{C}_3$ is called as it is the first (and only) item in the stack. This results in a contraction of $[\mathbf{b}]$, which induces the addition of the connected contractors $\mathcal{C}_2$, $\mathcal{C}_4$, $\mathcal{C}_3$ in $\mathscr{L}$. $\mathcal{C}_2$ is then called (first item in the stack), which contracts again $[\mathbf{b}]$: $\mathcal{C}_2$ is added again in $\mathscr{L}$. $\mathcal{C}_4$ is now called and contracts $[\mathbf{a}]$, which adds $\mathcal{C}_1$, $\mathcal{C}_4$ in the stack. This sequence runs until $\mathscr{L}$ becomes empty; a fixed point has been reached. One can

note that the graph may also be made of directed arcs, depending on the involved constraints. This enhances the propagation and so computation times.

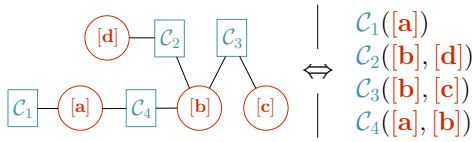


Figure 5: A CN corresponding to four contractors and four domains.

6 Example of use: a SLAM problem

6.1 Formalism

In robotics, Simultaneous Localization And Mapping (SLAM) [37] is a topic that ties together the problem of state estimation and that of mapping an unknown environment. A robot exploring its surroundings will associate localization uncertainties to the observed features of the environment, assigning their location with some error. However, a scene of the environment may be seen several times during the exploration, thus leading to an inter-temporal measurement which could benefit both localization and mapping procedures. Indeed, a robot that recognizes a part of the environment will deduce to be close to a previous position. This chicken-and-egg problem is difficult to solve with recursive algorithms such as Kalman filters, due to inter-temporal relations between the states, corresponding to so-called loop closures [38]. Because SLAM requires a capacity to manage equations involving states from different times and strong uncertainties, it can be more easily dealt with tubes and contractors.

We propose to solve a classical range-only SLAM problem using CNs. Let us consider a robot measuring distances from landmarks for which the position is unknown. This can be formalized with the following state equations:

$$\begin{cases} \mathbf{x}(0) = \mathbf{0}, & \text{(1a)} \\ \dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)), & \text{(1b)} \\ y^{(i)} = g(\mathbf{x}(t_i), \mathbf{b}^{(j)}), & \text{(1c)} \end{cases}$$

with $\mathbf{x}(t) \in \mathbb{R}^n$ and $\mathbf{u}(t) \in \mathbb{R}^m$, the state and input vectors, $y^{(i)} \in \mathbb{R}$ a distance measurement and $\mathbf{b}^{(j)} \in \mathbb{R}^2$ the related landmark of the environment. $\mathbf{f}$ and g are nonlinear functions depicting the evolution of the states and distance measurements. Both $\mathbf{u}(t)$ and $y^{(i)}$ are measured with some bounded errors.

6.2 Building a SLAM-CN

In a few steps, the problem is solved with Codac by (i) defining the initial domains (boxes, tubes) of our variables (vectors, trajectories); (ii) taking contractors from a

catalog of already existing operators, provided in the library, or building contractors for specific constraints; (*iii*) binding the contractors and domains in a CN; (*iv*) letting the CN solve the problem by contracting the sets of feasible values.

Eq. (1b) is a differential equation difficult to solve in the nonlinear case. We will therefore decompose the equation into two constraints $\mathbf{v} = \mathbf{f}(\mathbf{x}, \mathbf{u})$ and $\dot{\mathbf{x}} = \mathbf{v}$ in order to use available contractors, respectively $\mathcal{C}_{\mathbf{f}}$ [2] and $\mathcal{C}_{\frac{d}{dt}}$ [34]. Involved domains are intervals $[y]^{(i)}$, boxes $[\mathbf{b}]^{(j)}$ and tubes $[\mathbf{x}](\cdot)$, $[\mathbf{u}](\cdot)$, $[\mathbf{v}](\cdot)$. The resulting CN is pictured in Fig. 7, providing an illustration of the decomposition of tubes into graphs of slices for allowing a finer propagation. The source code of this SLAM-CN is available on `http://codac.io/slam` for encouraging future comparisons. The SLAM-CN runs contractions in less than 1s, providing the results pictured by Fig. 6. In this example, inter-temporal relations between the states are implicitly managed by the SLAM-CN. The landmarks are first localized and then used to improve the localization of the robot. This scenario is automatically managed by the CN.

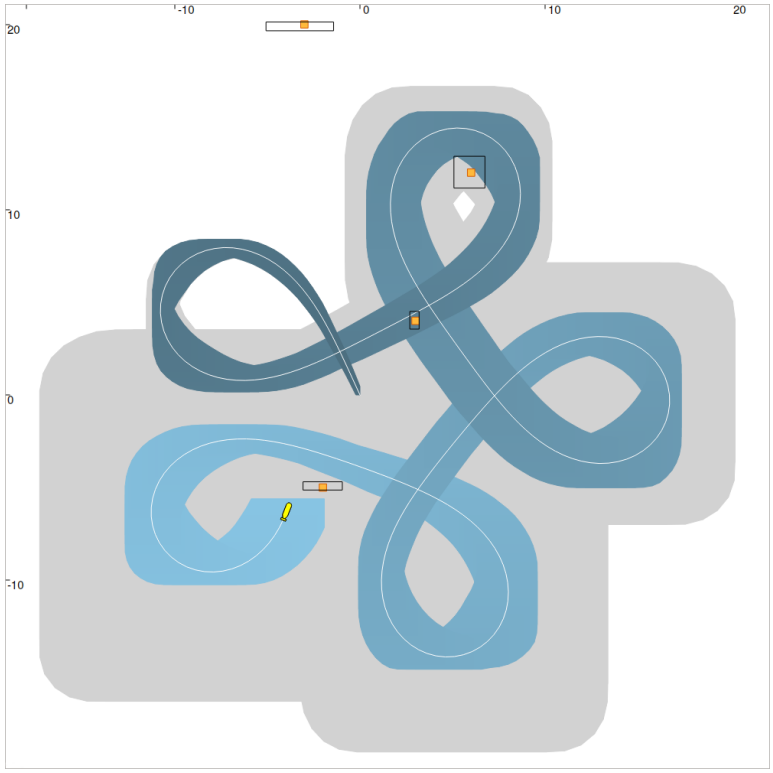


Figure 6: Contracted tube of positions (in blue) resulting from the SLAM-CN. The gray tube provides a reference corresponding to the localization drift without measurements. The estimated position of the landmarks is depicted by black boxes.

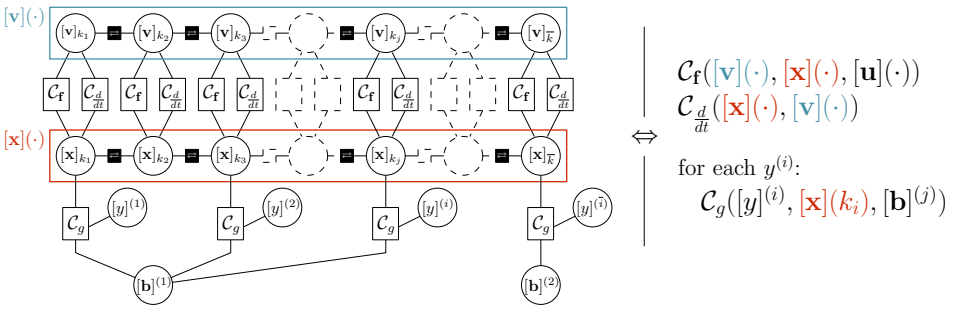


Figure 7: Illustration of a SLAM-CN associated with System (1) and some measurements from two landmarks $\mathbf{b}^{(1)}$, $\mathbf{b}^{(2)}$. The contractors and domains, formalized in a declarative way on the right-hand side, are transformed into a CN partially pictured on the left-hand side. For ease of reading, $[\mathbf{u}](\cdot)$ is not represented on this figure. The graph reveals how inter-temporal relations are implicitly built.

7 Conclusion

The Codac library offers a catalog of domains and contractors that one can combine into Contractor Networks in order to build interval solvers using a declarative paradigm. The assets lie both in the simplicity of the approach and the reliability of the results. It also allows to deal with a wide class of constraints that are classically encountered in real applications. Future work will concentrate on the development of the catalog of domains, contractors and separators, as well as improvements of CNs for real-time implementations.

The library is released under LGPLv3. The list of contributors of Codac is available on the website of the library, with related publications associated with the provided tools: http://codac.io. We encourage anyone interested in contributing to this open-source project to contact us.

References

[1] Alexandre dit Sandretto, J. and Chapoutot, A. Validated explicit and implicit Runge-Kutta methods. *Reliable Computing*, 22, 2016. URL: https://hal.archives-ouvertes.fr/hal-01243053.

[2] Benhamou, F., Goualard, F., Granvilliers, L., and Puget, J. F. Revising hull and box consistency. In *Proceedings of the International Conference on Logic Programming*, pages 230–244, Las Cruces, NM, 1999. DOI: 10.5555/341176.341208.

[3] Benhamou, F. and Touraivane, T. Prolog IV : langage et algorithmes. In *JFPLC*, pages 51–64, 1995. URL: https://www.mcours.net/cours/pdf/leilclic2/leilclic294.pdf.

[4] Bessiere, C. Constraint Propagation. In Rossi, F., van Beek, P., and Walsh, T., editors, *Foundations of Artificial Intelligence*, Volume 2 of *Handbook of Constraint Programming*, pages 29–83. Elsevier, 2006. DOI: 10.1016/S1574-6526(06)80007-6.

[5] Bourgois, A. *Safe and collaborative autonomous underwater docking*. PhD dissertation, ENSTA Bretagne, Brest, France, 2021. URL: https://hal.science/tel-03151588v1.

[6] Carbonnel, C., Trombettoni, G., Vismara, P., and Chabert, G. Q-intersection algorithms for constraint-based robust parameter estimation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 2630–2636. AAAI Press, 2014. DOI: 10.1609/aaai.v28i1.9117.

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

[8] Chabert, G., Ninin, J., and Neveu, B. IBEX: A C++ numerical library based on interval arithmetic and constraint programming. URL: http://www.ibex-lib.org.

[9] Desrochers, B. and Jaulin, L. Computing a guaranteed approximation the zone explored by a robot. *IEEE Transaction on Automatic Control*, 62:425–430, 2016. DOI: 10.1109/TAC.2016.2530719.

[10] Desrochers, B. and Jaulin, L. A minimal contractor for the polar equation; application to robot localization. *Engineering Applications of Artificial Intelligence*, pages 83–92, 2016. DOI: 10.1016/j.engappai.2016.06.005.

[11] Desrochers, B. and Jaulin, L. Thick set inversion. *Artificial Intelligence*, 249:1–18, 2017. DOI: 10.1016/j.artint.2017.04.004.

[12] Goldsztejn, A., Hayes, W., and Collins, P. Tinkerbell is chaotic. *SIAM Journal on Applied Dynamical Systems*, 10(4):1480–1501, 2011. DOI: 10.1137/100819011.

[13] Goldsztejn, A. and Jaulin, L. Inner and outer approximations of existentially quantified equality constraints. In *Principles and Practice of Constraint Programming*, pages 198–212, Nantes (France), 2006. DOI: 10.1007/11889205_16.

[14] Goualard, F. GAOL: Not just another interval library. URL: https://github.com/goualard-f/GAOL.

[15] 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, New York, NY, USA, 2014. ACM. DOI: 10.1145/2562059.2562113.

[16] Guyonneau, R. *Set-membership approaches for mobile robot localization*. PhD dissertation, Université de Nantes Angers Le Mans, France, 2013. URL: https://theses.hal.science/tel-00961501.

[17] Hansen, E. R. and Sengupta, S. Global constrained optimization using interval analysis. In Nickel, K., editor, *Interval Mathematics*, pages 25–47. Academic Press, New York, NY, 1980.

[18] Jaulin, L. Range-only SLAM with indistinguishable landmarks; a constraint programming approach. *Constraints*, 21, 2016. DOI: 10.1007/s10601-015-9231-9.

[19] Jaulin, L. and Desrochers, B. Introduction to the algebra of separators with application to path planning. *Engineering Applications of Artificial Intelligence*, 33:141–147, 2014. DOI: 10.1016/j.engappai.2014.04.010.

[20] Jaulin, L. and Walter, E. Set inversion via interval analysis for nonlinear bounded-error estimation. *Automatica*, 29(4):1053–1064, 1993. DOI: 10.1016/0005-1098(93)90106-4.

[21] Kurzhanski, A. and Valyi, I. *Ellipsoidal calculus for estimation and control*. Birkhäuser, Boston, MA, 1997. URL: https://link.springer.com/book/9780817636999.

[22] Le Mézo, T. *Bracketing largest invariant sets of dynamical systems*. PhD dissertation, Université Bretagne Loire, Brest, France, 2019.

[23] Le Mézo, T., Jaulin, L., and Zerr, B. Bracketing the solutions of an ordinary differential equation with uncertain initial conditions. *Applied Mathematics and Computation*, 318:70–79, 2018. DOI: 10.1016/j.amc.2017.07.036.

[24] Lohner, R. J. Enclosing the solutions of ordinary initial and boundary value problems. *Computer Arithmetic*, pages 225–286, 1987.

[25] Mackworth, A. Consistency in networks of relations. *Artificial Intelligence*, 8(1):99–118, 1977. DOI: 10.1016/0004-3702(77)90007-8.

[26] Nedialkov, N. and Jackson, K. ODE Software that computes guaranteed bounds on the solution. In *Advances in Software Tools for Scientific Computing*, Lecture Notes in Computational Science and Engineering, pages 197–224. Springer, Berlin, Heidelberg, 2000. DOI: 10.1007/978-3-642-57172-5_6.

[27] Nehmeier, M. and von Gudenberg, J. W. filib++ , expression templates and the coming interval standard. *Reliable Computing*, 15:312–320, 2011. URL: https://interval.louisiana.edu/reliable-computing-journal/volume-15/no-4/reliable-computing-15-pp-312-320.pdf.

[28] 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.

[29] Rauh, A. and Jaulin, L. A computationally inexpensive algorithm for determining outer and inner enclosures of nonlinear mappings of ellipsoidal domains. *Applied Mathematics and Computer Science*, 31(3):399–415, 2021. DOI: 10.34768/amcs-2021-0027.

[30] Rauh, A., Rohou, S., and Jaulin, L. An ellipsoidal predictor-corrector state estimation scheme for linear continuous-time systems with bounded parameters and bounded measurement errors. *Frontiers in Control Engineering*, 3, 2022. DOI: 10.3389/fcteg.2022.785795.

[31] Revol, N., Makino, K., and Berz, M. Taylor models and floating-point arithmetic: Proof that arithmetic operations are validated in COSY. *The Journal of Logic and Algebraic Programming*, 64(1):135–154, 2005. DOI: 10.1016/j.jlap.2004.07.008.

[32] Revol, N. and Rouillier, F. Motivations for an arbitrary precision interval arithmetic and the MPFI library. *Reliable Computing*, 11(4):275–290, 2005. DOI: 10.1007/s11155-005-6891-y.

[33] Rohou, S. and Jaulin, L. Exact bounded-error continuous-time linear state estimator. *Systems & Control Letters*, 153, 2021. DOI: 10.1016/j.sysconle.2021.104951.

[34] 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.

[35] Rohou, S., Jaulin, L., Mihaylova, L., Le Bars, F., and Veres, S. M. Reliable non-linear state estimation involving time uncertainties. *Automatica*, 93:379–388, 2018. DOI: 10.1016/j.automatica.2018.03.074.

[36] Rohou, S., Jaulin, L., Mihaylova, L., Le Bars, F., and Veres, S. M. *Reliable robot localization: A constraint-programming approach over dynamical systems*. ISTE Ltd, London, 2019. DOI: 10.1002/9781119680970.

[37] Smith, R., Self, M., and Cheeseman, P. Estimating uncertain spatial relationships in robotics. In *Autonomous Robot Vehicles*, pages 167–193. Springer, New York, NY, 1990. DOI: 10.1007/978-1-4613-8997-2_14.

[38] Thrun, S. and Leonard, J. J. Simultaneous Localization and Mapping. In Siciliano, B. and Khatib, O., editors, *Springer Handbook of Robotics*, pages 871–889. Springer, Berlin, Heidelberg, 2008. DOI: 10.1007/978-3-540-30301-5_38.

[39] Voges, R. *Bounded-error visual-LiDAR odometry on mobile robots under consideration of spatiotemporal uncertainties*. PhD dissertation, Leibniz Universität Hannover, Germany, 2020. DOI: 10.15488/9932.

[40] Walter, E. and Piet-Lahanier, H. Exact recursive polyhedral description of the feasible parameter set for bounded-error models. *IEEE Transactions on Automatic Control*, 34(8):911–915, 1989. DOI: 10.1109/9.29443.

[41] Wilczak, D., Zgliczyński, P., Pilarczyk, P., Mrozek, M., Kapela, T., Galias, Z., Cyranka, J., and Capinski, M. Computer assisted proofs in dynamics group, a C++ package for rigorous numerics, 2017. URL: http://capd.ii.uj.edu.pl.

Acta Cybernetica **26** (2024) 889–912.

RISC-V Based Hardware Acceleration of Interval Contractor Primitives in the Context of Mobile Robotics

Pierre Filiol[ab], Theotime Bollengier[ac], Luc Jaulin[ad], and Jean-Christophe Le Lann[ae]

Abstract

Localization tasks, generally modeled as Constraint Satisfaction Problem (CSP), are recurring problems in mobile robotics. Known approaches rely on software libraries which all have their advantages but also their limitations, among which non-optimal computing performances. This paper proposes a different approach which consists in extending the RISC-V ISA to provide hardware support for interval primitives.

Keywords: intervals, contractor programming, RISC-V, IEEE-1788, mobile robotics, FPGA

1 Introduction

A lot of recurring mobile robotics tasks, such as robust-control or localization, can be modeled as a *Constraint Satisfaction Problem* (CSP) and solved by characterizing the corresponding solution set $\mathbb{S}$. A common way to achieve this is to compute an inner and outer approximation of $\mathbb{S}$, which is usually done with the help of contractor algebra and a paving algorithm such as SIVIA [21]. The paver classifies the search space by recursively calling a contractor to discard parts outside of the target set. In [7, 15, 20] this formalism is illustrated in real-world robotics examples.

A *contractor* C for the set $\mathbb{X} \subset \mathbb{R}^n$ is an operator $\mathbb{IR}^n \to \mathbb{IR}^n$ which satisfies:

$$C([\mathbf{x}]) \subset [\mathbf{x}] \text{ (}\textit{contractance}\text{)}, \tag{1}$$

$$[\mathbf{x}] \subset [\mathbf{y}] \implies C([\mathbf{x}]) \subset C([\mathbf{y}]) \text{ (}\textit{monotonicity}\text{)}, \tag{2}$$

[a]ENSTA Bretagne, Brest, France
[b]E-mail: pierre.filiol@netc.fr, ORCID: 0009-0009-6162-9578
[c]E-mail: theotime.bollengier@ensta-bretagne.org, ORCID: 0009-0006-7315-2736
[d]E-mail: lucjaulin@gmail.com, ORCID: 0000-0002-0938-0615
[e]E-mail: jean-christophe.le_lann@ensta-bretagne.fr, ORCID: 0000-0003-2555-1805

DOI: 10.14232/actacyb.306636

$$C([\mathbf{x}]) \cap \mathbb{X} = [\mathbf{x}] \cap \mathbb{X} \ (\textit{consistency}), \tag{3}$$

where $\mathbb{IR}^n$ is the set of axis-aligned boxes of $\mathbb{R}^n$.

Let C_1 and C_2 be two contractors. The union and intersection operators are defined as follows:

$$(C_1 \cap C_2)([\mathbf{x}]) = C_1([\mathbf{x}]) \cap C_2([\mathbf{x}]), \tag{4}$$

$$(C_1 \cup C_2)([\mathbf{x}]) = C_1([\mathbf{x}]) \sqcup C_2([\mathbf{x}]), \tag{5}$$

where $[\mathbf{a}] \sqcup [\mathbf{b}]$ is the smallest box which contains both $[\mathbf{a}]$ and $[\mathbf{b}]$.

The members of the robotics community usually rely on the use of existing software libraries to implement the contractors required to evaluate solution sets. The available libraries implement the IEEE-1788 standard on interval analysis [18] to various extents and define the most common interval primitives (operators, contractors, ...) for reuse. This paper advocates in favor of a novel approach which adds support for interval primitives directly in RISC-V hardware. We believe that this method can be beneficial to solve some of the portability issues occurring in software approaches, especially for embedded targets. Another advantage is the ability to compute with intervals using a tailored speed/precision compromise which is impossible on general-purpose hardware.

This paper is organized as follows. Section 2 presents a typical mobile robotics localization problem which is used as a reference throughout the article. Then Section 3 discusses the benefits of a hardware approach for intervals. Section 4 compares common hardware acceleration techniques while Section 5 presents the methodology used to create the proof of concept for hardware support of intervals. Section 6 sums up the obtained results and presents future works.

In the rest of the paper, sets $\mathbb{X}$ of $\mathbb{R}^n$ will be represented in *mathbb* font and intervals $[x]$ or boxes $[\mathbf{x}]$ within brackets.

2 State of the art and previous work

Let us consider the following localization problem which will be used for the discussions in this paper. The goal is to solve the CSP which is related to the localization of a robot (green triangle) using 3 landmarks (red circles) with known positions (Figure 1). The robot uses its internal sensors to compute the distances d_i toward landmark i with $i \in \{1, 2, 3\}$. The coordinate of each landmark is defined by a_i with $i \in \{1, 2, 3\}$. Those values are represented by intervals due to measurement uncertainties. The goal is to estimate the position (x_r, y_r).

The state vector of the robot is defined as:

$$\mathbf{x} = (x_r, y_r)^\top \tag{6}$$

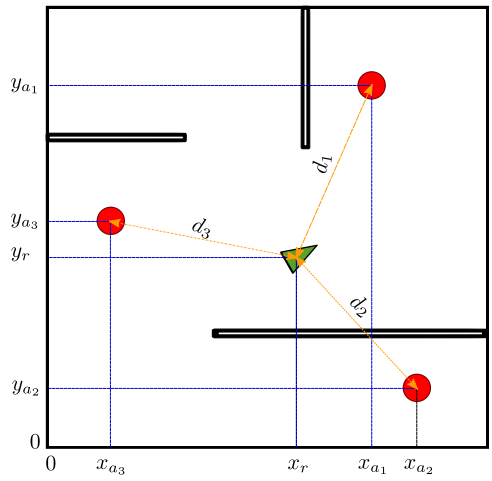


Figure 1: Localization of a robot using 3 landmarks

and its respective domain as:

$$[\mathbf{x}] = [0, \infty] \times [0, \infty]. \tag{7}$$

A couple of coordinates (x_r, y_r) is a potential solution if it satisfies the following constraints (ring equations):

$$a)\ (x_r - x_{a_1})^2 + (y_r - y_{a_1})^2 \in {d_1}^2, \tag{8}$$

$$b)\ (x_r - x_{a_2})^2 + (y_r - y_{a_2})^2 \in {d_2}^2, \tag{9}$$

$$c)\ (x_r - x_{a_3})^2 + (y_r - y_{a_3})^2 \in {d_3}^2. \tag{10}$$

Each constraint gives a corresponding solution sets:

$$\mathbb{S}_1 : \{\mathbf{x} \in [0, \infty]^2 \mid (x_r - x_{a_1})^2 + (y_r - y_{a_1})^2 \in {d_1}^2\}, \tag{11}$$

$$\mathbb{S}_2 : \{\mathbf{x} \in [0, \infty]^2 \mid (x_r - x_{a_2})^2 + (y_r - y_{a_2})^2 \in {d_2}^2\}, \tag{12}$$

$$\mathbb{S}_3 : \{\mathbf{x} \in [0, \infty]^2 \mid (x_r - x_{a_3})^2 + (y_r - y_{a_3})^2 \in {d_3}^2\}. \tag{13}$$

And the solution set for the localization problem becomes:

$$\mathbb{S} = S_1 \cap S_2 \cap S_3. \tag{14}$$

The goal is now to find 3 contractors C_1, C_2, C_3 which respectively characterize sets $\mathbb{S}_1, \mathbb{S}_2, \mathbb{S}_3$ and to perform the intersection of contractors to find $\mathbb{S}$.

C_1 can be computed with a forward-backward procedure [25] such as HC4-revised by using the set $\mathbb{S}_1$ AST (Figure 2). Note that throughout the article forward and backward contractors on operator/function $*$ are respectively defined as $\overrightarrow{C_*}$ and $\overleftarrow{C_*}$. The resulting contractors can be found in Algorithm 1 and 2.

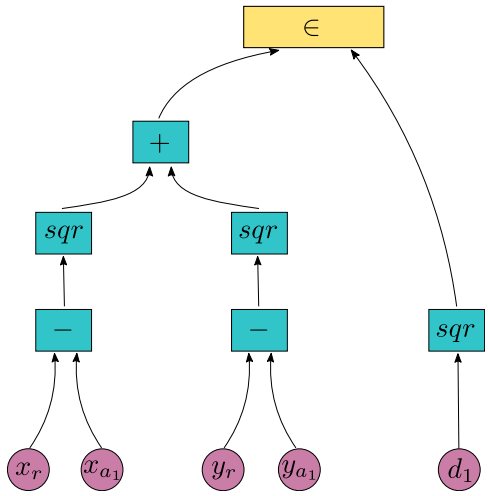


Figure 2: Abstract syntax tree corresponding to $\mathbb{S}_1$

Algorithm 1 Algorithm for $\overrightarrow{C_1}$

Funct $\overrightarrow{C_1}([x_r],[y_r],[x_{a_1}],[y_{a_1}],[d_1])$

1: $[DistX] = \overrightarrow{C_-}([x_r],[x_{a_1}])$
2: $[DistY] = \overrightarrow{C_-}([y_r],[y_{a_1}])$
3: $[DistXSqr] = \overrightarrow{C_{sqr}}([DistX])$
4: $[DistYSqr] = \overrightarrow{C_{sqr}}([DistY])$
5: $[DistSqr] = \overrightarrow{C_{sqr}}([d_1])$
6: **return** $[DistXSqr],[DistYSqr],[DistSqr]$

The following libraries are among the most commonly used in robotics:

- Goualard et al's Gaol [14]. A C++ low-level interval library which is focused on providing most of the common reverse operators for backward propagation.
- Chabert et al's Ibex [4] (IMT Atlantique). A library written in C++ targeting system solving and global optimization problems using interval arithmetic.
- Nehmeier et al's libieep1788 [24]. A C++ template library focused on rigorous implementation of the IEEE-1788 standard.

Algorithm 2 Algorithm for $\overleftarrow{C_1}$

Funct $\overleftarrow{C_1}([DistXSqr], [DistYSqr], [DistSqr])$

1: $[DistXSqr], [DistYSqr] = \overleftarrow{C_+}([DistSqr], [DistXSqr], [DistYSqr])$
2: $[a] = \overleftarrow{C_{sqr}}([DistXSqr], [DistX])$
3: $[b] = \overleftarrow{C_{sqr}}([DistYSqr], [DistY])$
4: $[x_r] = \overleftarrow{C_-}([DistX], [x_r], [x_{a_1}])$
5: $[y_r] = \overleftarrow{C_-}([DistY], [y_r], [y_{a_1}])$
6: **return** $[x_r], [y_r]$

- INRIA's MPFI [29]. A C library implementing interval arithmetic in arbitrary precision by using MPFR reliable floating-points.

The contractor C_1 can be created by implementing Algorithms 1 and 2 using the simple contractor primitives available for example in the aforementioned software libraries.

3 Motivations for a hardware approach

The minimal requirement to perform interval computations in an embedded robotic system is the availability of an hardware *floating-point unit* (FPU) . The industrial standard for that purpose is the IEEE-754 [19] which enables a theoretical portability of the user code to any compliant hardware. However, no guarantee is made about consistency and the final result will not be bit-consistent across multiple {hardware, compiler} pairs. This subject has been extensively researched in contributions such as [9, 13].

This lack of consistency is especially problematic for interval computations which require frequent switches between IEEE-754 rounding modes to perform accurate interval bounds evaluation (respectively round to $-\infty$ and round to $+\infty$ for lower and upper bounds).

Another obstacle lies in the implementation of the interval libraries themselves. Most of them are built using third-party maths libraries which perform fast and accurate floating-point evaluation at the expense of portability. It is frequent for them to inline assembly instructions directly in the C code for optimization but as a consequence the whole library becomes architecture-locked.

For all the reasons detailed above, we believe that relying on a standard floating-point architecture is not a valid approach. This paper presents a dedicated hardware architecture to accelerate the most common interval primitives along with a dedicated compiler. The {hardware, compiler} pair thus tackles the problem of embedded intervals and presents a proof of concept for a hardware accelerator targeted at the most commonly used interval primitives.

3.1 Tailored performances for interval computation

A lot of robotics applications require to repetitively call a contractor to make use of a short-lived sensor input. This is typically the case of the localization example introduced in Section 2 where the distances d_i are used to estimate the robot location. Ideally the contraction fixed-point must be reached before obtaining new inputs which implies to find a compromise between execution speed and result precision. These factors are known to be inversely proportional in the context of floating-point operators [11].

Popular architectures like X86 or ARM use hardware FPUs which are optimized for the IEEE-754 [19] single- and double- precision floating-points. While these formats are good "one size fits all" trade-offs for most of general-purpose computations, they can become a limiting factor in some niche applications such as interval arithmetic. The main reason is that they impose predefined ranges and precisions for floating-point numbers while real-world problems can have varied requirements.

Another drawback is the overall hardware complexity induced by general-purpose FPUs which are required to support all the rounding modes defined in the IEEE-754 standard. An FPU tailored for interval arithmetic, on the other hand, would only require two rounding modes for bound computations (round to $-\infty$ and $+\infty$), thus leading to a lower hardware complexity and power consumption.

3.2 Guaranteed computation at hardware level

Interval arithmetic is especially suited to solve hard non-linear problems in a reliable and efficient way. The IEEE-1788 standard [18] specifies the behavior of interval operators using various flavors which are all organized as depicted in Table 1. From now on, we consider only the set-based flavor which is the most commonly used in robotics.

The mathematical layer describes how interval operations work. It also introduces the notion of decorated intervals comprising a bare interval and a decoration. The latter are meant to implement the standard way of dealing with non-nominal cases during computation. The available decorators for the set-based flavor are presented on Figure 1 and their inclusion relationships are as follows:

$$com \subset dac \subset def \subset trv \supset ill. \tag{15}$$

In a previous article [10] we had shed light on a recurring contractor arithmetic bug which occurs when a function evaluates an interval that is not or partially in its domain of definition. The solution proposed was to mark the incriminated interval with a ι flag, perform the forward propagation normally and execute a modified backward propagation which takes the flag into account. A solution in the spirit of IEEE-1788 would be to create a decorator "out-of-domain" (*ood*) to handle this non-nominal case but doing this without breaking the existing logic depicted in Formula 15 is hard.

Table 1: The set-based flavor decorators (extracted from [18])

Value	Short description	Property	Definition
`com`	common	$p_{\texttt{com}}(f, \boldsymbol{x})$	$\boldsymbol{x}$ is a bounded, nonempty subset of $\mathrm{Dom}(f)$; f is continuous at each point of $\boldsymbol{x}$; and the computed interval $f(\boldsymbol{x})$ is bounded.
`dac`	defined & continuous	$p_{\texttt{dac}}(f, x)$	$\boldsymbol{x}$ is a nonempty subset of $\mathrm{Dom}(f)$, and the restriction of f to $\boldsymbol{x}$ is continuous;
`def`	defined	$p_{\texttt{def}}(f, \boldsymbol{x})$	$\boldsymbol{x}$ is a nonempty subset of $\mathrm{Dom}(f)$;
`trv`	trivial	$p_{\texttt{trv}}(f, \boldsymbol{x})$	always true (so gives no information);
`ill`	ill-formed	$p_{\texttt{ill}}(f, \boldsymbol{x})$	Not an Interval; formally $\mathrm{Dom}(f) = \emptyset$,

The ι flag information could be added to the standard and could be directly used by the hardware implementation.An example of an interval encoding using the ι flag is depicted in Figure 7. This approach comes with a lot of freedom since few things are currently defined in the standard regarding the bit-level representation of intervals (decorated intervals, empty set, NaN handling,...).

4 Evaluation of hardware acceleration strategies for interval arithmetic

4.1 Affordable hardware prototyping with FPGA

The *application specific integrated circuit* (ASIC) technology is nowadays the industrial standard to produce high-end chips. This design process allows to reach the highest performances at the price of tremendous costs in development and manufacturing facilities. The resulting chips are made affordable for customers only through mass-replication which lowers the per-unit cost. As a consequence, this technique is not suitable for hobbyists or small research teams who operate with tight budgets and produce only a few prototypes to evaluate the performances of a specific design.

A solution for this audience lies in the *field programmable gate array* (FPGA) technology which comes at much affordable prices. While ASICs operate directly at transistor level, FPGAs expose an array of interconnected logic cells which contain the digital building blocks for more advanced logic (Figure 3). The desired behavior of the user logic is described through *hardware description languages* (HDL) such as VHDL or VERILOG which are also widely used in ASIC field. With the help of dedicated software, called synthesizers, the user can modify the connections between cells to match the HDL code. This process can be repeated any number of times and allows incremental development whereas ASICs chips are etched once and forever. Table 2 briefly compares both technologies.

From now on, any hardware design mentioned in this paper refers to the production of HDL code for synthesis on an FPGA chip.

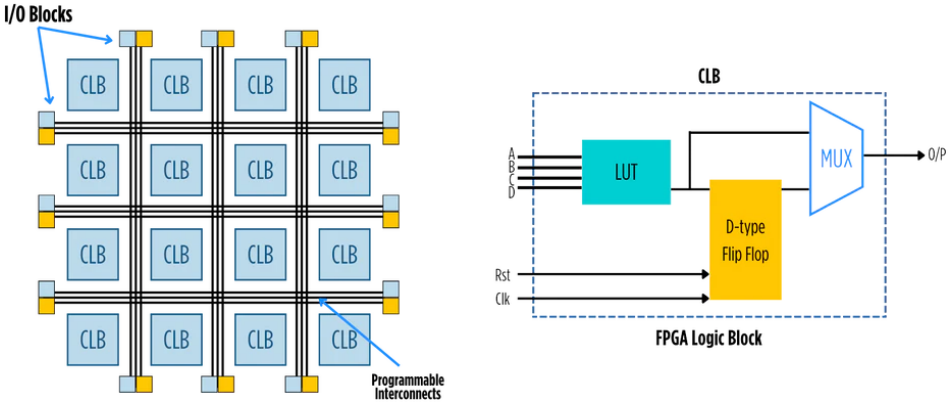


Figure 3: FPGA reconfigurable network and content of a logic cell [6]

Table 2: Comparison of FPGA and ASIC

	FPGA	**ASIC**
Design flow	Reconfigurable circuit (suitable for prototyping)	Permanent circuitery (no room for error)
Design input	HDL	HDL
Entry cost	Low (from 100 $ to 10k dollars)	High (typically millions of dollars)
Typical use	Prototyping of a few units	Mass-production
Energy consumption	Higher than ASIC	Lower than FPGA
Frequencies	Lower than ASIC	Higher than FPGA
Analog designs	Not supported.	Supported (eg transceivers ...).

4.2 Full hardware equation mapping strategy

FPGA are very efficient at implementing highly specialized circuits for niche applications that no manufacturer would mass produce for cost reasons. There are no theoretical issues in translating a set of equations into a dedicated circuit, this approach is very FPGA compliant and has been done very often in literature [8, 12, 16]. While the equations from a contractor such as C_1 are no different and could be implemented in hardware with great performances, this solution will not be the preferred one as the main focus is made on re-usability, ease of prototyping and costs.

Mapping equations from contractors into efficient hardware confronts the designer with difficult FPGA problems such as optimizing the timing and pipelining of

a big combinatorial circuit. The slightest modification of the input contractor (like adding an angular measurement for the landmarks) would require to re-engineer the circuit from scratch and face those difficulties again. In these conditions, it is difficult to imagine that an interval user would be skilled enough both in HDL and in hardware development to be really autonomous with this solution. Moreover, it is frequent to see evolutions in input contractors during the development of a robotic application.

Another drawback of this method is related to high hardware resource consumption. A circuit with no logic folding would require one instantiation for each contractor primitive in the target contractor. For example, Algorithm 1 (forward contraction from C_1) performs 3 square contractions which require duplicated logic. The amount of available hardware resources on a typical FPGA is limited, especially on the low-end and affordable ones. The use of this technique should be reserved to either simple contractors (for which the need of an hardware implementation is dubious) or for production-oriented synthesis on high-end boards.

4.3 Coprocessor strategy

This solution is an alternative to exposing the hardware directly as it was done in previous technique. The user communicates with a hardware device using a software abstraction. In general-purpose computing, this paradigm is often implemented using a coprocessor which performs the computation jobs requested by the main *Central Processing Unit* (CPU). For example, this is the computation model adopted in modern *Graphic Processor Units* (GPUs) where graphic pipeline operations are transferred to a PCI device with the help of APIs such as CUDA or OPENCL.

This technique suits interval computation and a FPGA could be used as a coprocessor to expose data parallelism (SIVIA boxes for example) in a GPU fashion. While this system is theoretically the best compromise between re-usability and performances (logic folding and hardware parallelism), it will not be studied further in this paper as it implies too much complexity for a first approach to hardware intervals. Several problems must be solved to achieve this goal which are beyond building an efficient interval core. The designer must indeed organize the communication between several core instances (for parallelism), optimize host to device communications and produce an efficient compiler.

4.4 Instruction set extension of a general purpose cpu

The two previous subsections have raised the importance of logic folding to lower the hardware complexity and resources utilization. This new strategy aims at adding interval operators directly in the main CPU. An *Instruction Set Architecture* (ISA) defines the supported assembly instructions and their behaviour in an implementation-agnostic way to guarantee binary compatibility between several chip manufacturers. Popular ISAs come with compiler support for high-level

languages such as C thus allowing the development of general-purpose software libraries (libc,...).

This paper aims at extending an existing ISA to support a cleverly chosen set of hardware interval primitives. The main benefit is to produce more efficient libraries that make use of hardware interval instructions instead of the general-purpose ones (Figure 4). This operation requires both the modification of the CPU (circuit) and the compiler (to use the new instructions).

In practice, there is no official mechanisms to alter mainstream CPUs from brand such as INTEL, AMD or ARM to inject new hardware instructions. First and foremost, their architecture design is extremely complex (gate-level design) and the manufacturing process requires industrial tools which are far beyond the budget of hobbyists. Additionally, the corresponding ISAs are distributed under proprietary licenses which prevent any legal modifications. Finally, as older ISAs such as X86 have very limited available opcode space remaining, manufacturers are reluctant to add new instructions to the standard unless there is a major consensus.

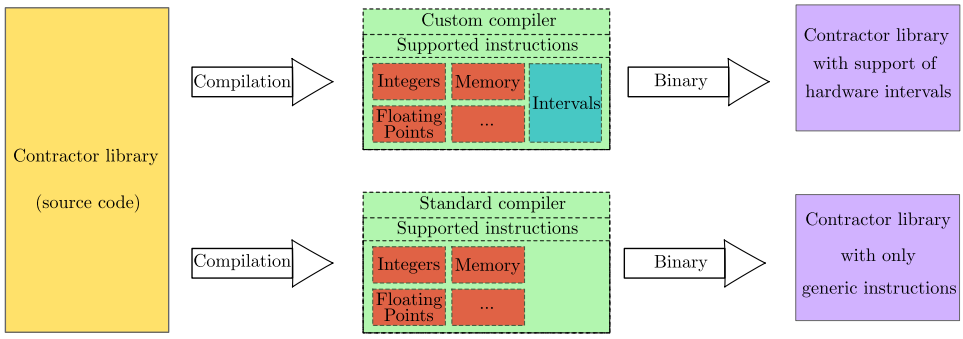


Figure 4: ISA and software stack modification

5 Design of a RISC-V interval custom extension

5.1 The RISC-V standard

The RISC-V standard brings some solutions to the aforementioned problems. Contrary to X86/ARM, it is an open and free ISA [31] whose specification started in 2014 as a purely academic project carried out by the university of Berkeley. As a consequence, anyone is free to implement a core which follows RISC-V standards without paying royalties to any third-party. The main philosophy behind RISC-V is to promote a simple processor model featuring a load-store micro-architecture typical of *Reduced Instruction Set Computer* (RISC) in opposition to the *Complex Instruction Set Computer* (CISC) adopted by X86 processors. The performances and overall simplicity of RISC-V design has gained traction in the last years for

low-power/embedded applications in various fields such as neural-networks [22], cryptography [5] or in our case robotics [32].

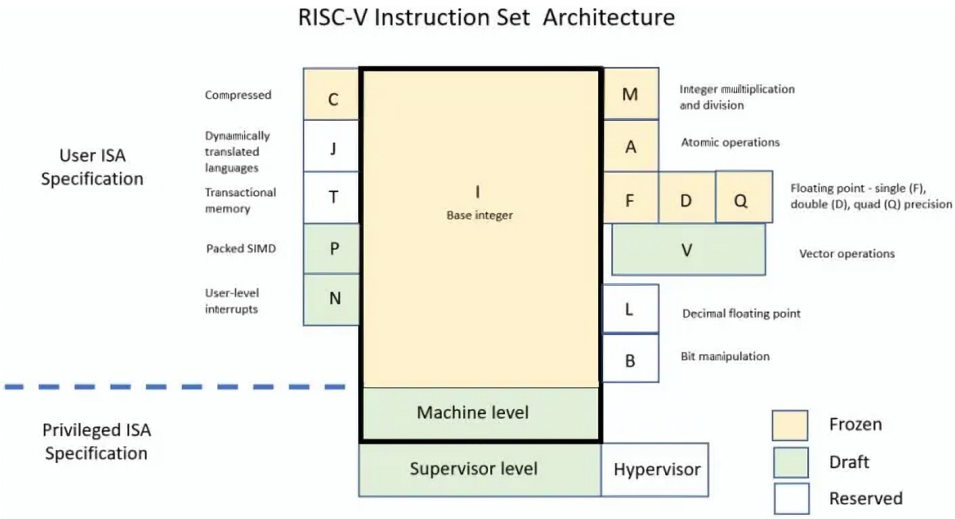


Figure 5: The standard extensions [28]

The standard provides no hints about processor implementation but enforces several design characteristics about the ISA:

- The ISA is designed with modularity in mind. Any RISC-V implementation is composed of a mandatory base ISA (named I) and a number of ISA extensions (identified by a letter). The aim is to allow the user to build a custom processor perfectly tailored for a specific need. All the extensions currently specified in the standard are displayed in Figure 5. A RISC-V core can be described using a naming convention which consists in RV + the register width (in bits) and the supported extensions. For example, a RV32IMFD is a 32 bits core implementing extensions I (base), M (integer multiplication), F (single-precision floating point support) and D (double-precision floating point support).
- The ISA specifies the required registers for each extension as well as their width. For example, base extension (I) comes with 32 registers of width 32 or 64 (depending on the chosen implementation).
- The ISA can be extended with custom extensions to add application-specific operators.

Another strength of the standard lies in its very rich software ecosystem. The user can access a fully-fledged GCC compiler with a dedicated RISC-V toolchain

which takes into account which extensions are actually available in the core to produce optimized binaries.

5.2 Objectives and virtual prototyping

The paper will use the RISC-V standard as a way to demonstrate the feasibility of adding hardware support for intervals and tackle real-world robotic problems. A complete strategy would be as follows:

- Due to the overall simplicity of the standard, it is possible to synthesize an HDL description of a RISC-V core on a middle range FPGA. A lot of core designs can be found in various configurations either in the literature such as [26, 23] or bought from *Intellectual property* (IP) vendors such as Sifive.com. This allows to build a baseline effortlessly and evaluate the performances on the localization problem given in section 2 with only the standard instructions. The target core configuration is set to RV32IMAFD for the reasons exposed in Section 5.3.

- After the initial performance measurement, we take advantage of RISC-V extensibility to add support for a new extension geared toward interval computation named *xinterval*. Two major tasks must be done to achieve this goal. The first one is to implement the *xinterval* instructions in hardware (using an HDL) and to integrate them in the base design synthesized on FPGA (Section 5.3). The second one consists in adding support for instructions in the compiler. To this end, we rely on the available RISC-V GCC which has also been designed with extensibility in mind and allows to define new instructions easily (Section 5.4).

- Evaluate the performances on the localization problem given in Section 2 with the new instructions/compiler and compare with previous results.

The hardware design of a RISC-V extension is a complex topic which demands an extensive knowledge of the underlying micro-architecture and raises various optimization trade-offs. This has been done in literature in works such as [3, 28] and [2]. As a first step we wanted to prove that intervals could be added to RISC-V ISA in an efficient way without worrying too much about hardware implementation. This goal can be achieved through *Virtual Prototyping* (VP) which allows functional evaluation. This approach is often used in system design [1, 17, 27] to decrease development times and help in the upcoming *Register Transfer Level* (RTL) verification step. The development of hardware primitives in an HDL and its integration is a RISC-V core is left as future work and will be the main topic of a next article.

The VP used in this study is organized as depicted in Figure 6. The host computer runs a simulation program where a robot tries to estimate its location using 3 landmarks with fixed coordinates (box ③ on Figure 6). At each simulation

step, the distance measurements between the robot and each landmark are sent in the memory of a simulated RISC-V core (box ① on Figure 6). The processor simulator is another program written in C++ which aims at achieving functional emulation of a binary code execution up to the assembly level. It implements the required RISC-V standard extensions and *xinterval* instructions and is able to execute a binary code compiled with the custom GCC presented in Section 5.4. Practically, the simulated core runs an implementation of Algorithms 1 and 2 from section 2 implemented with *xinterval* instructions instead of traditional floating-point.

5.3 Design principle of the `xinterval` custom extension

The simulated core ① from Figure 6 is defined by RISC-V standard as a RV32IMFD. It implements the following standard extensions and registers:

Table 3: Extensions used in the simulated core

Letter	Name	Number of instructions
I	Base Integer	40
M	Integer Multiplication	8
F	Single-precision floating-point	26
D	Double-precision floating-point	26

Table 4: Registers used in the simulated core

Names	Characteristics	Number
x_0-x_{31}	32-bit integer register	32
f_0-f_{31}	32-bit floating-point register	32

The simulated core relies on standard extensions F and D which bring dedicated registers and instructions (Tables 2 and 3) to enable floating-point support respectively in single (32-bit) and double (64-bit) precisions. An interesting feature of the RISC-V standard is the ability to handle 64-bit floating-point numbers on a 32-bit processor (identified by RV32xxx) by working on pairs of 32-bit floating-point registers. The main idea behind *xinterval* integration is to fit the interval representation into one of the aforementioned register, to take advantage of the standard F/D instructions to handle load/store operations and to develop the missing HDL logic to perform interval computations. Section 3.1 recalled that a hardware approach allowed to tailor the performances according to the needs of a specific application. For intervals, this translates into the ability to configure the floating-point format used to encode the bounds of an interval instead of forcing a sub-optimal IEEE-754

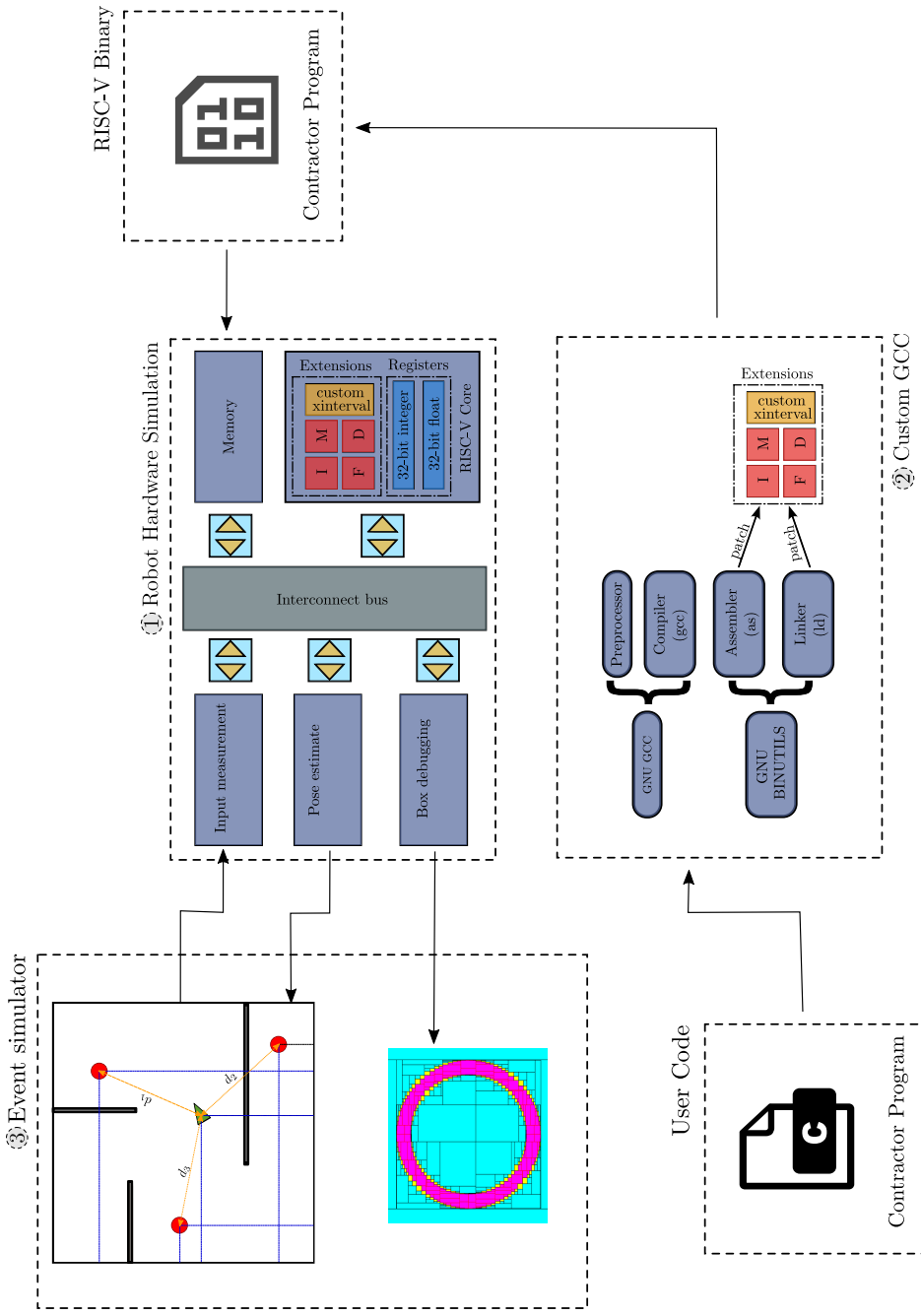


Figure 6: The RISC-V evaluation testbench

format. A direct consequence is the impact on performances in matter of latency, resource utilization and frequency of the resulting hardware design which allows to address a wide range of applications and targets.

In the particular context of RISC-V integration, the interval representation needs to fit a 64-bit double register. We made the choice to use the interval representation depicted in Figure 7. Here, a bound is encoded using 31 bits with a 7 bits exponent field (while 8 are used in IEEE-754 single precision). Practically, this reduces the range of representable numbers (which still remain acceptable for typical robotics applications) but leaves room for two additional 1-bit flags in the 64-bit data. The empty flag marks an empty set and the ι flag is applied to intervals which are subject to the phenomenon illustrated in Section 3.2. If an application requires greater speed at the expense of precision, it would be possible to further reduce the number of bits used for the bounds, even if this means leaving unused offsets in the register.

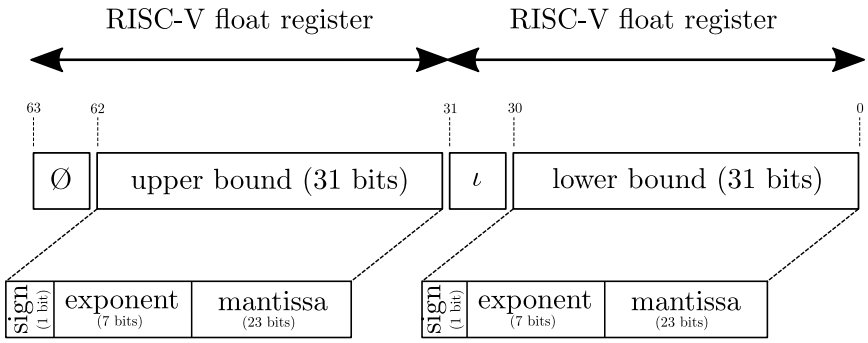


Figure 7: The 64-bit interval representation used in *xinterval*

The custom *xinterval* extension contains instructions to perform the forward and backward contractions of recurring interval primitives. The RISC-V standard introduces limitations that force instructions to adopt one of the encoding shown in Figure 8. Most of the time, we use the R-Type which stands for "register instruction" and encodes a destination register (r_d) and two source registers (r_1 and r_2). Fields *opcode*, $funct_3$, and $funct_7$ are used to discriminate between two R-Type instructions and send them to the right portion of the logic circuit in the processor. For interval primitives with two inputs (addition, ...) the associated backward contractor must have 3 inputs (see Table 5) which forces to use the R4-Type which is a sub-case of R-Type where the $funct_7$ is partly replaced by a third source register r_3. To ease the explanations, instruction formats used in this section will be named using RISC-V GCC terminology (Section 5.4/Table 5).

The general design philosophy used in *xinterval* will now be reviewed through 2 examples:

The addition is a case of a two inputs arithmetic operator for which *xinterval* instructions are summed up in Table 6. The forward contractor performs a simple addition using the input interval operands saved in registers r_{s1},r_{s2} and stores the

Table 5: An extract of GCC terminology for instruction fields

Mnemonic	**Meaning**
D	Fp destination register
S	Fp source register 1
T	Fp source register 2
R	Fp source register 3

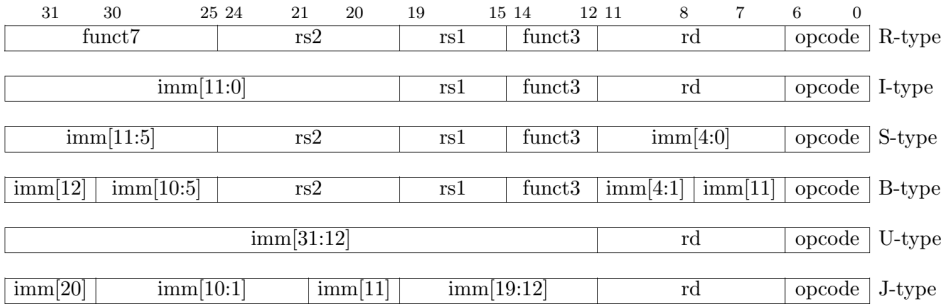


Figure 8: Legal instruction formats in the RISC-V standard

result in destination register r_d. Two corresponding backward-contractors can be defined to contract each input separately. In robotics, the output of the backward contraction is often intersected with the corresponding input pre-propagation to update state variables (x_c and y_c in Table 3). As a consequence, we made the choice to optimize this precise case at the expense of an additional register r_{s3}.

Table 6: Instructions linked to operator +.

Instruction	Prototype	Type	Operation	Register movements
addfwctc	"D,S,T"	+ forward contractor	$z = x + y$	$r_d = r_{s1} + r_{s2}$
addbwctc1	"D,S,T,R"	+ backward contractor 1	$x_c = x \cap (z - y)$	$r_d = r_{s1} \cap (r_{s3} - r_{s2})$
addbwctc2	"D,S,T,R"	+ backward contractor 2	$y_c = y \cap (z - x)$	$r_d = r_{s2} \cap (r_{s3} - r_{s1})$

The principle differs a bit for algebraic and elementary functions, and especially for those which are not defined everywhere and suffer from the bug recalled in Section 3.2. The example of the square root illustrates all these issues. The forward contractor performs a modified version of the square root ($sqrt_\iota$) which sets the ι flag of the output interval when the input is not entirely in the domain of definition.

The backward contractor evaluates the ι flag of the input and performs a modified version of the backward square root function ($sqrt_\iota bw$). It works the same way as the traditional operator but decorates the interval with a iota if the input is partially in the square root domain of definition. This time, square root required only one interval input (in r_{s1}) so the matching prototype in RISC-V assembly for forward and backward are respectively "D,S" and "D,S,T". These operations are described in Table 7.

Table 7: Instructions linked to function $\sqrt{}$.

instruction	Prototype	Type	Operation	Register movements
sqrtfwctc	"D,S"	Square root forward contractor	$y = sqrt_\iota(x)$	$r_d = sqrt_\iota(r_{s1})$
sqrbwctc	"D,S,T"	Square root backward contractor	$x_d = sqrt_\iota bw(x, y)$	$r_d = sqrt_\iota bw(r_{s1}, r_{s2})$

The custom extension *xinterval* proceeds in an analog way for all the operators and algebraic, elementary functions which are often used in robotics applications (Table 8).

Table 8: Supported contractor primitives in *xinterval*.

Addition	Subtraction	Multiplication	Division
Square root	Square	Exponential	Logarithm
Cosine	Sine		

5.4 Modification of the software stack

The GCC RISC-V toolchain is an open-source project bundled as part of the RISC-V software stack. Support for new instructions can be added by modifying the *binutils* module whose purpose is to convert assembly code into an executable binary. This module is then called by GCC as part of the compilation process as depicted in ② in Figure 6. This method is simple because it only requires the modification of *binutils* assembler which is a far less complex code base than the compiler itself.

The encoding of new assembly instructions must be added to the source code of *binutils* and satisfy two rules:

- Uniqueness: two instructions cannot have the same encoding to prevent collisions.
- Format: instruction must have one of the legal types presented in Figure 8.

Let us take the example of two-input forward-contractors of *xinterval* presented in Section 5.3. Since they belong to the R-Type, each new instruction must have a unique $\{opcode,\ funct_3,\ funct_7\}$ combination which does not collide with other standard instructions. Field *opcode* acts as a preliminary filter and is encoded using 7 bits. The standard has officially left some opcode values unused (0xB, 0x2B, 0x5B and 0x7B) by built-in extensions to accommodate custom increments (Table 9). An opcode can only regroup instructions of same formats due to micro-architecture limitations but this is not a problem for us since most of our added operators are R-Type with the opcode value fixed (0xB), all our 2-inputs forward contractors must have unique $\{funct_3,\ funct_7\}$ pairs as depicted in Table 10.

Table 9: The opcode space of the RISC-V standard

inst[1:0] = "11" for RV32

inst[6:5] \ inst[4:2]	000	001	010	011	100	101	110	111
00	LOAD	LOAD FP	custom-0	MISC MEM	OP-IMM	AUIPC	OP-IMM-32	48b
01	STORE	STORE FP	custom-1	AMO	OP	LUI	OP-32	64b
10	MADD	MSUB	NMSUB	NMADD	OP-FP	reserved	custom-2	48b
11	BRANCH	JALR	reserved	JAL	SYSTEM	reserved	custom-3	≥80b

Available for custom extensions

Table 10: Examples of forward-contractor encoding in *xinterval*

instr name	$funct_7$	rs_2	rs_1	$funct_3$	r_d	*opcode*
addfwctc	0000000	-	-	100	-	0001011
subfwctc	0000001	-	-	100	-	0001011
mulfwctc	0000010	-	-	100	-	0001011
divfwctc	0000011	-	-	100	-	0001011

In the source code of *binutils*, the user can add a relationship between an assembly mnemonic (ie the name of the instruction as written in code) and a machine code encoding [30]. This information is used by GCC during the compilation process GCC to produce efficient machine code from the assembly code.

After the patch, RISC-V GCC is now able to compile the following C code using our new instructions (Listing 1). The same process can be repeated for all instructions of *xinterval* to lay the foundations of a hardware-accelerated interval library as presented in Figure 4.

```
/* interval is defined as a double (64 bits) */
typedef interval double;

/* inline function which uses of xinterval instruction addfwctc */
/* inputs loaded from double registers */
/* output stored in double register */
inline interval __attribute__((always_inline))
  _addFwCtc(interval itv1, interval itv2) {

  interval result;;
  asm("addfwctc %0, %1, %2" : "=f"(result) : "f"(itv1), "f"(itv2));
  return result;
}
```

Listing 1: Calling *addfwctc* from C

5.5 Solving the localization problem

In order to solve the localization problem, the robot must compute the intersection of contractors C_1, C_2 and C_3 using the landmark distance measurements. We implemented Algorithms 1 and 2 in C with explicit use of *xinterval* operators to mimic an embedded program running inside a robot. It was then compiled with the modified GCC and executed by the simulated core presented in Section 5.2. (① on Figure 6).

The event simulator shown in ③ on Figure 6 is an external tool which generates coherent sensor measurements to simulate a trajectory between landmarks. These data are periodically written to the robot sensor virtual device to be used by the program running on the ISS. After each localization step, the pose estimate and the SIVIA box are written to dedicated robot devices for *a posteriori* analysis. Figure 9 shows examples of localization paving obtained with our virtual prototype.

6 Results and future works

This paper discussed the advantages of using dedicated hardware to perform interval computations in the context of embedded robotics. This approach differs from the usual one which consists in using software libraries in conjunction to general-purpose hardware. We took advantage of the RISC-V standard to design a custom ISA extension regrouping interval primitives such as forward and backward contractors (Section 5.3). Naturally, the existing software stack and especially the GCC compiler has been modified to expose the new assembly instructions in a high-level language such as C (Section 5.4). Finally, we tested the core in a real-world localization problem using a virtual prototype and performed a successful run of the Algorithm 1 and 2 as shown in Figure 9 (Section 5.5).

As stated in Section 5.1, the goal of this paper was not to implement a full hardware accelerator but to demonstrate the possibility of handling interval primitives directly in hardware. The presented virtual prototype demonstrated the model

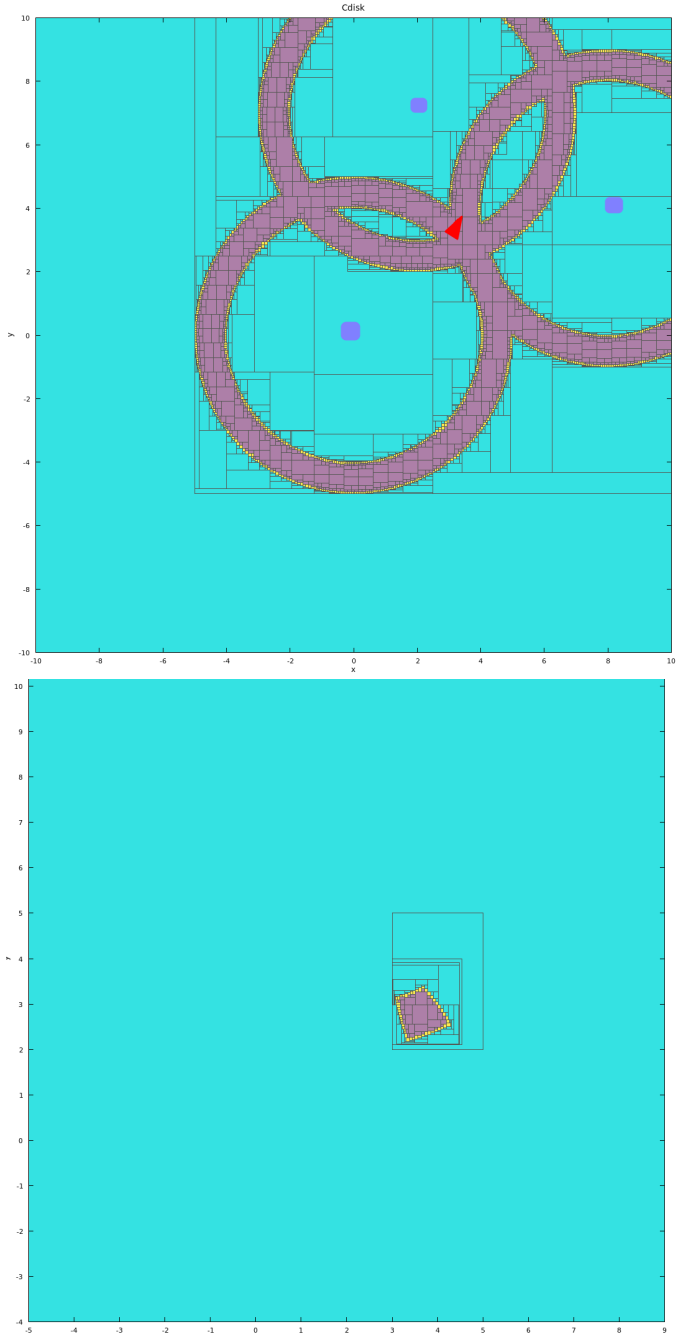


Figure 9: Contractors union and intersection (simulated hardware)

correctness up to the machine code level and is a step toward another refinement of the model to add the RTL layer (under the form of HDL code).

In future works, the focus will be made on the transition between virtual prototype and hardware synthesized on FPGA. The design strategy explained in Section 5.1 will be performed. The main topics are recalled below:

- Synthesis of an existing RISC-V design on a middle-range FPGA. As explained in Section 5, this has been done several times in literature and a lot of designs on shelf are available.

- Execution metrics such as execution time. number of clock cycles and overall frequency will be measured on Algorithm 1 and 2 with only the standard extensions. This process gives a baseline to compare with interval-dedicated instructions.

- Implementation of interval primitives in hardware and integration in RISC-V. The most challenging part is to implement IEEE-754 arithmetic and algebraic operators in an HDL such as VHDL. Once again, such a topic has been widely studied in the literature and the job will be to perform the concatenation of all this work in a dedicated chip. As we target multi-precision, several design trade-offs must be explored for these operators which are mainly hardware resources utilization on FPGA, output frequency and latency.

- Characterization of the performances achieved by the hardware acceleration. We will re-run the Algorithms 1 and 2 (now compiled with *xinterval* support) and compare the results through the prism of the metrics defined in Step 2.

References

[1] Ahmadi-Pour, S., Pieper, P., and Drechsler, R. Virtual-peripheral-in-the-loop : A hardware-in-the-loop strategy to bridge the VP/RTL design-gap. arXiv: 2311.00442, 2023. DOI: 10.48550/arXiv.2311.00442.

[2] Alkim, E., Evkan, H., Lahr, N., Niederhagen, R., and Petri, R. ISA Extensions for Finite Field Arithmetic Accelerating Kyber and NewHope on RISC-V. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2020(3):219–242, 2020. DOI: 10.13154/TCHES.V2020.I3.219-242.

[3] Bandara, S., Ehret, A., Kava, D., and Kinsy, M. BRISC-V: An open-source architecture design space exploration toolbox. In *Proceedings of the 2019 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, page 306–306. ACM, 2019. DOI: 10.1145/3289602.3293991.

[4] Chabert, G. IBEX library, 2007. URL: https://github.com/ibex-team/ibex-lib.

[5] Cheng, H., Großschädl, J., Marshall, B., Page, D., and Pham, T. RISC-V instruction set extensions for lightweight symmetric cryptography. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2023(1):193–237, 2022. DOI: 10.46586/tches.v2023.i1.193-237.

[6] C&T Solutions Inc. What is FPGA (Field Programmable Gate Array)? how does it work? URL: https://www.candtsolution.com/news_events-detail/what-is-fpga-field-programmable-gate-array-and-how-does-it-work/ — Accessed: 2024-03-12.

[7] Delafosse, M., Clerentin, A., Delahoche, L., Brassart, E., and Marhic, B. The mobile robot localization problem treated as a constraint satisfaction problem. *IFAC Proceedings Volumes*, 37(8):394–399, 2004. DOI: 10.1016/S1474-6670(17)32008-6.

[8] El-Kurdi, Y., Giannacopoulos, D., and Gross, W. J. Hardware acceleration for finite-element electromagnetics: Efficient sparse matrix floating-point computations with FPGAs. *IEEE Transactions on Magnetics*, 43(4):1525–1528, 2007. DOI: 10.1109/TMAG.2007.892459.

[9] Farnum, C. Compiler support for floating-point computation. *Software: Practice and Experience*, 18(7):701–709, 1988. DOI: 10.1002/spe.4380180709.

[10] Filiol, P., Bollengier, T., Jaulin, L., and Le Lann, J.-C. A new interval arithmetic to generate the complementary of contractors. In Schön, S., editor, *Proceedings of the Summer Workshop on Interval Methods, Hannover, Germany*. Gottfried Wilhelm Leibniz Universität, 2022. URL: https://hal.science/hal-03859346.

[11] Forget, L., Uguen, Y., and de Dinechin, F. Comparing posit and IEEE-754 hardware cost, 2021. URL: https://hal.science/hal-03195756.

[12] Gac, K., Karpiel, G., and Petko, M. FPGA based hardware accelerator for calculations of the parallel robot inverse kinematics. In *Proceedings of 2012 IEEE 17th International Conference on Emerging Technologies & Factory Automation*, pages 1–4, 2012. DOI: 10.1109/ETFA.2012.6489717.

[13] Goldberg, D. What every computer scientist should know about floating-point arithmetic. *ACM Computing Surveys*, 23(1):5–48, 1991. DOI: 10.1145/103162.103163.

[14] Goualard, F. GAOL — Not just another interval library, 2020. URL: https://github.com/goualard-f/GAOL.

[15] 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.

[16] Gwalani, K. A. and Elkeelany, O. Design and evaluation of FPGA based hardware accelerator for elliptic curve cryptography scalar multiplication. *WSEAS Transactions on Computers*, 8(5):884–893, 2009. URL: https://dl.acm.org/doi/10.5555/1558772.1558786.

[17] Herdt, V., Große, D., Pieper, P., and Drechsler, R. RISC-V based virtual prototype: An extensible and configurable platform for the system-level. *Journal of Systems Architecture*, 109:101756, 2020. DOI: 10.1016/j.sysarc.2020.101756.

[18] IEEE. 1788-2015 — IEEE standard for interval arithmetic, 2015. DOI: 10.1109/IEEESTD.2015.7140721.

[19] IEEE. 754-2019 — IEEE standard for floating-point arithmetic (revision of IEEE Std 754-2008), 2019. DOI: 10.1109/IEEESTD.2019.8766229.

[20] Jaulin, L. Localization of an underwater robot using interval constraints propagation. In Benhamou, F., editor, *Proceedings of the Twelfth International Conference on Principles and Practice of Constraint Programming*. Springer Berlin Heidelberg, 2006. DOI: 10.1007/11889205_19.

[21] Jaulin, L. and Walter, E. Set inversion via interval analysis for nonlinear bounded-error estimation. *Automatica*, 29(4):1053–1064, 1993. DOI: 10.1016/0005-1098(93)90106-4.

[22] Jin, S., Qi, S., Dai, Y., and Hu, Y. Design of convolutional neural network accelerator based on RISC-V. In Abawajy, J. H., Xu, Z., Atiquzzaman, M., and Zhang, X., editors, *Proceedings of the Tenth International Conference on Applications and Techniques in Cyber Intelligence*, Volume 170 of *Lecture Notes on Data Engineering and Communications Technologies,*, pages 446–454. Springer International Publishing, 2023. DOI: 10.1007/978-3-031-29097-8_53.

[23] Khabarov, S. RISC-V VHDL: System-on-Chip, 2023. URL: https://sergeykhbr.github.io/.

[24] Nehmeier, M. libieeep1788: A C++ implementation of the IEEE interval standard P1788. In *Proceedings of the 2014 IEEE Conference on Norbert Wiener in the 21st Century (21CW)*, pages 1–6, 2014. DOI: 10.1109/NORBERT.2014.6893854, URL: https://github.com/nehmeier/libieeep1788.

[25] Neumaier, A. and Schichl, H. Interval analysis on directed acyclic graphs for global optimization. *Journal of Global Optimization*, 33(4):541–562, 2005. DOI: 10.1007/s10898-005-0937-x.

[26] Nolting, S. The NEORV32 RISC-V processor. Zenodo, 2024. DOI: 10.5281/zenodo.5018888, URL: https://github.com/stnolting/neorv32.

[27] Pieper, P., Herdt, V., and Drechsler, R. Advanced embedded system modeling and simulation in an open source RISC-V virtual prototype. *Journal of Low Power Electronics and Applications*, 12(4), 2022. DOI: 10.3390/jlpea12040052.

[28] Quinnell, R. Creating a custom processor with RISC-V. *EE Times Europe*, 2010. URL: https://www.eetimes.eu/creating-a-custom-processor-with-risc-v/.

[29] Revol, N. The MPFI library: Towards IEEE 1788-2015 compliance. In *Proceedings of the 13th International Conference on Parallel Processing and Applied Mathematics*, number 12044 in LNCS, pages 353–363, 2019. DOI: 10.1007/978-3-030-43222-5_31, URL: https://inria.hal.science/hal-02162346.

[30] Sandid, H. R. Adding custom instructions to the RISC-V GNU-GCC toolchain, 2022. URL: https://hsandid.github.io/posts/risc-v-custom-instruction/.

[31] Waterman, A., Lee, Y., Patterson, D. A., and Asanović, K. The RISC-V instruction set manual, Volume I: User-level ISA, version 2.0. Technical Report UCB/EECS-2014-54, EECS Department, University of California, Berkeley, 2014. URL: http://www2.eecs.berkeley.edu/Pubs/TechRpts/2014/EECS-2014-54.html.

[32] Zelensky, A., Alepko, A., Dubovskov, V., and Kuptsov, V. Heterogeneous neuromorphic processor based on RISC-V architecture for real-time robotics tasks. In Dijk, J., editor, *Artificial Intelligence and Machine Learning in Defense Applications II*, Volume 11543, page 115430L. International Society for Optics and Photonics, SPIE, 2020. DOI: 10.1117/12.2574470.

Acta Cybernetica **26** (2024) 913–932.

GPU-Accelerated, Interval-Based Parameter Identification Methods Illustrated Using the Two-Compartment Problem

Lorenz Gillner[ab] and Ekaterina Auer[ac]

Abstract

Interval methods are helpful in the context of scientific computing for reliable treatment of problems with bounded uncertainty. Most traditional interval algorithms, however, were designed for sequential execution while internally depending on processor-specific instructions for directed rounding. Nowadays, many-core processors and dedicated hardware for massively parallel data processing have become the de facto standard for high-performance computers. Interval libraries have yet to adapt to this heterogeneous computing paradigm.

In this article, we investigate the parallelization of interval methods with an emphasis on modern graphics processors. Using a parameter identification scenario in combination with newly developed or enhanced GPU-based interval software, we evaluate different methods for reducing the size of large interval search domains. For the first time, algorithmic differentiation can be used with intervals on the GPU. Different versions of interval optimization algorithms are compared wrt. their functionality, run times, and energy consumption.

Keywords: parameter identification, interval software, parallelization, GPGPU

1 Introduction

Since many decades, interval analysis [27] has been used in the context of scientific computing for obtaining verified solutions to many problems, for example, in computer graphics or in engineering. One of its advantages is the possibility to quantify or propagate bounded uncertainty through systems in simulations in a deterministic way. Many of the major programming languages have built-in interval capabilities; over 20 libraries for interval arithmetic alone are available today. With the emergence of multi-core processors, parallelization of interval methods using well

[a] University of Applied Sciences Wismar, Germany
[b] E-mail: lorenz.gillner@hs-wismar.de ORCID: 0009-0007-8244-5810
[c] E-mail: ekaterina.auer@hs-wismar.de, ORCID: 0000-0003-4059-3982

DOI: 10.14232/actacyb.306774

established libraries such as C–XSC, Boost, or Profil/Bias has been tested in multi-threaded and distributed systems [11, 23, 28, 35]. However, high-performance computing industry is moving towards developing specialized hardware to accelerate repetitive tasks, for example, graphics processing units (GPUs), data processing units (DPUs), or field-programmable gate arrays (FPGAs). Interval software still has to be adapted to such specific co-processors.

Especially general-purpose graphics processing units (GPGPUs) have inspired much interest among the scientific community. One part of the reason is their low cost and high availability compared to conventional supercomputers or large-scale CPU clusters. The application of GPUs for computations with the focus on uncertainty has been investigated in [2, 3, 9, 10, 29, 34], just to name a few. Nonetheless, popular interval libraries cannot typically be used on the GPU directly, because the co-processors' architecture differs significantly from that of conventional CPUs. For example, the switching of rounding modes commonly applied in CPU-based interval libraries, which has a negative influence on computing performance, is not necessary on the GPU, since most mathematical operations are available as specifically rounded versions. Notable examples of custom interval libraries for the GPU are given in [5, 7, 21]. In this paper, we are interested in the usefulness of GPUs in the area of interval computations. We extend traditional (sequential) ideas to the massively parallel computing paradigm of GPUs and apply them in a classic parameter estimation scenario.

The paper is structured as follows. In Section 2, the basics of the employed interval techniques are described and software implementing them on the GPU is highlighted where applicable. In Section 3, parameter identification methods we implement using the GPU are detailed. In Section 4, we employ the described methods within our testing procedure relying on the well-known example of a two-compartment problem and highlight the comparison results. A perspective on the paper's findings and an outlook on our further research are in Section 5.

2 Background on Interval Analysis

In this section, we provide a short overview of the concepts from the area of interval analysis that are applied in the rest of this article. Where necessary, we also highlight the GPU-based software implementing them.

2.1 Interval Analysis and Algorithmic Differentiation

Interval analysis (IA), formally introduced by R. Moore in 1966 [26], is a powerful mathematical tool for *verified* computations, that is, computations with an automatically provided guarantee of correctness. In the context of scientific computing, IA offers a way to compensate for numerical uncertainty caused by finite floating point (FP) number representation as specified by the IEEE 754 standard. Instead of working with a crisp FP number approximating a given real number, IA methods rely on an interval with FP bounds containing it, propagating this uncertainty

through a computation. Almost as a by-product, this approach allows us to propagate uncertainty representable by intervals through systems in a deterministic and verified way. A real interval $\mathbf{x}$ is a closed interval defined as

$$\mathbf{x} = [\underline{x}, \overline{x}] = \{x \in \mathbb{R} \mid \underline{x} \leq x \leq \overline{x}\},$$

whereas a machine interval has to be additionally rounded towards $\pm\infty$ to the next possible representation $[\lfloor \underline{x} \rfloor, \lceil \overline{x} \rceil]$. For two intervals $\mathbf{x}$ and $\mathbf{y}$, arithmetic operations $\circ \in \{+, -, \cdot, /\}$ can be defined as

$$\mathbf{x} \circ \mathbf{y} = \{x \circ y \mid \forall x \in \mathbf{x}, y \in \mathbf{y}\},$$

while elementary functions $\zeta(.)$ (e.g. $\sin x$) can be extended for the use with intervals as

$$\zeta_{[]}(\mathbf{x}) = [\min\{\zeta(a) \mid \forall a \in \mathbf{x}\}, \max\{\zeta(b) \mid \forall b \in \mathbf{x}\}].$$

These simple rules can be used to define (naive) interval arithmetic, that is, a way to evaluate composite functions over intervals directly, without having to solve any optimization problems as in the equation above. There are more involved interval algorithms, for example, those for computing verified enclosures of solutions to (non-linear) systems of (differential) equations.

Although replacing FP values by intervals guarantees an enclosure of the true result, the results can exhibit overestimation, that is, intervals being wider than necessary. This behaviour originates from the so-called *dependency problem*, when multiple occurrences of the same interval variable are interpreted as separate new variables, and the *wrapping effect*, meaning that an interval enclosure of any (multi-dimensional) shape not identical to an axis-aligned box will always contain some amount of superfluous data [27].

The ability to perform automatic differentiation (AD) is essential for any form of (non-naive) interval analysis [14]. For example, the centered form of an interval extension to the real-valued function g, in the univariate case defined as

$$g_{[\tau]}(\mathbf{x}) = g(\tau) + g'(\mathbf{x}) \cdot (\mathbf{x} - \tau), \quad \tau \in \mathbf{x}, \tag{1}$$

requires the evaluation of the first derivative of g. The same is true for interval root-finding algorithms, such as the interval Newton method [5]. The widely used divided differences method

$$g'(x) \approx \frac{g(x+h) - g(h)}{h} \tag{2}$$

yields only an approximation of the true derivative, with the error increasing as the step size decreases, which makes computations using it less reliable [4]. Although symbolic differentiation (SD) produces expressions for exact derivatives, it might exhibit a considerable computational overhead (since it is a top-down approach) or even increase overestimation when used with intervals [16]. By contrast, automatic differentiation allows us to compute the value of a function's exact derivative at a specific point or its enclosure in a bottom-up approach without errors introduced by floating point approximation and with less of the overhead. In general, AD falls into three steps:

1. Decomposition of g into elementary expressions (e.g., $\ln x$)
2. Differentiation of elementary expressions $\left(\text{e.g., } \frac{\partial}{\partial x} \ln x = \frac{1}{x}\right)$
3. Application of the chain rule, i.e.
   $(y(x(t)))' = \frac{\partial y}{\partial t} = \frac{\partial y}{\partial x} \cdot \frac{\partial x}{\partial t} = y'(x(t)) \cdot x'(t)$, e.g., $\frac{\partial}{\partial t} \ln x(t) = \frac{1}{x(t)} x'(t)$.

This allows us to differentiate not only mathematical functions, but also entire code segments containing them. While there is a wide selection of AD libraries available, some of them even compatible with the GPU [13], interval arithmetic is typically not supported by these libraries. Although many popular interval libraries can perform AD of interval-valued functions on the CPU, to the authors' knowledge, the Julia language is the only open-source tool offering a straightforward way of using AD with interval-valued functions on the GPU [31], see Section 2.3 for details.

2.2 Cooperativity

For the so-called *cooperative* dynamical systems, it is possible to perform set-based computations taking into account bounded uncertainty in parameters without necessarily applying methods with result verification. Consider an ODE system in the explicit, autonomous form

$$y' = f(y, p),\ y : \mathbb{R} \mapsto \mathbb{R}^{n_y},\ p \in \mathbb{R}^{n_p},\ f : \mathbb{R}^{n_y + n_p} \mapsto \mathbb{R}^{n_y} \tag{3}$$

depending on a time-invariant parameter p (that is, $p' = 0$) and parameter-invariant initial conditions $y(t_0)$ of the solution function y (that is, $\dfrac{\partial y_i(t_0, p)}{\partial p_j} = 0$), possibly influenced by bounded uncertainty with $y(t_0) \in \mathbf{y_0}$ for a $t_0 \in \mathbb{R}$ and $p \in \mathbf{p}$, where the bold face denotes characteristics described by intervals as introduced in the previous subsection. The property of cooperativity holds for the system in Eq. (3) if

$$\frac{\partial f_i}{\partial y_j} \geq 0 \text{ for all } i \neq j,\ i, j = 1 \dots n_y\,. \tag{4}$$

Müller's theorem [25] combined with the property of cooperativity results in the Smith's theorem [32] that shows the possibility to quantify the uncertainty in the system (3) by working with two systems with crisp parameters that are independent of each other (the bracketing systems)

$$y'_{\mathrm{lb}} = \underline{f}(y_{\mathrm{lb}}, p_{\mathrm{b}}) \quad \text{and} \quad y'_{\mathrm{ub}} = \overline{f}(y_{\mathrm{ub}}, p_{\mathrm{b}}) \tag{5}$$

instead of one system with uncertain parameters. Here, p_{b} means that the crisp values of the lower or upper bounds of the interval $\mathbf{p} = [p_{\mathrm{lb}}\ ,\ p_{\mathrm{ub}}]$ are used (not the values in between them). The true result $y(t)$ lies in the convex hull $[\underline{y}_{\mathrm{lb}}(t), \overline{y}_{\mathrm{ub}}(t)]$ of the enclosures $\mathbf{y}_{\mathrm{lb}}(t)$, $\mathbf{y}_{\mathrm{ub}}(t)$ of the true solutions $y_{\mathrm{lb}}(t)$, $y_{\mathrm{ub}}(t)$ to (5). Considering such enclosures in each point of time t_k of the chosen time grid gives us the verified flow of the uncertain system (3) over this grid. Good approximations $y_{\mathrm{lb}}^{(k)}$, $y_{\mathrm{ub}}^{(k)}$

to the verified solutions $y_{\mathrm{lb}}(t_k)$, $y_{\mathrm{ub}}(t_k)$ are obtained by solving the systems in (5) using traditional floating point arithmetic (e.g., solvers `odeint` available in the C++ library BOOST). This procedure captures the bulk of output uncertainty and can be useful on the GPU (cf. Section 3.2.1) since appropriate GPU implementations of verified ODE solvers are not available at the time of writing.

2.3 Software for IA and IA-Based AD on the GPU

Implementation of interval methods in any programming language requires interval arithmetic as the core component; performing full interval analysis for a practical problem often needs further functionalities and methods. The C–XSC toolbox [20] or INTLAB [30], for example, offer additional modules for AD, root-finding and optimization, but due to their CPU-specific instructions for rounding control, they are not suited for use on the GPU. One very promising tool for GPU computations involving interval-based AD is the Julia language, because its meta-programming approach allows for generating hardware-specific instructions from generically formulated source code. In [31], it was demonstrated that it was indeed possible to compute derivatives of interval functions for both the CPU and GPU, albeit without tight-as-possible rounding and full support for elementary functions. However, the focus of this paper is on native C++ implementations, because the comfort of Julia's generic programming comes at the price of lengthy compile times.

The Compute Unified Device Architecture (CUDA) is the programming framework for NVIDIA-manufactured GPUs. A basic library for interval arithmetic on CUDA-compatible GPUs was presented in [7]. This library has been extended by a selection of elementary set-based functions in [8]. While arithmetic operations such as addition are available in different directed rounding modes in CUDA C++, the same is not true for trigonometric and other elementary functions. Their results, therefore, need to be rounded to the next and previous FP number if the simplest kind of enclosure is to be obtained. Hence, enclosures produced by the GPU are in some cases wider than they would have been on the CPU. We use our own, enhanced version of this interval library for all interval computations in this paper.

To use interval methods requiring derivatives, programmers are required to provide all necessary derivatives directly in the source code at the moment. To enable AD for these libraries, we ported the forward mode of the well-established C++ library FADBAD [6] to the CUDA C++ dialect. It is entirely template-based, which allows it to work both with different FP data types and interval ones, as long as a corresponding arithmetic is defined completely. Compatibility with CUDA-capable devices can be implemented in a fairly straightforward way, at least in the case of forward mode AD. The data structures and functions used by FADBAD are entirely compatible with CUDA-intrinsic functions, so they merely have to be labeled as suitable for CPU (`__host__`) and GPU (`__device__`) use. This modification makes FADBAD the first AD library to fully support IA on both the CPU and the GPU. A topic for our future work is to test the limitations of this implementation on the GPU and to improve it accordingly, if possible.

3 Parameter Identification

In this section, we describe set-based optimization methods — some of them based on IA and fully verified — for the general goal of parameter identification on the GPU. Our approach is not to parallelize a given optimization method; rather, we rely on the massive data parallelism of the GPU to employ brute-force techniques that would be too time-consuming on the CPU.

3.1 General Possibilities for Parameter Identification

The task of identifying n unknown but bounded system parameters can be viewed as a global optimization problem of the form

$$\min_{\mathbf{x}\in\mathbb{R}^n} c(\mathbf{x}),\ c : \mathbb{R}^n \mapsto \mathbb{R}\,, \tag{6}$$

where c is the objective (cost) function wrt. the system parameters $\mathbf{x}$. According to the widely used least-squares principle, the cost function Φ can be employed to identify unknown parameters $p \in \mathbb{R}^{n_p}$ of a dynamic model of the form in Eq. (3) with initial conditions at $t_0 := T_b - 1$, given a search space $\mathbf{p} \in \mathbb{IR}^{n_p}$ and measured data over a discrete time grid $t_k \in \{T_b, T_b + 1, \ldots, T_e\}$:

$$\Phi(p) = \sum_{k=T_b}^{T_e} \sum_{j=1}^{n_m} \left(y_j(t_k, p) - y_{m,j}^{(k)}\right)^2 \xrightarrow{p\in\mathbf{p}} \min\,, \tag{7}$$

where $y_j(t_k, p)$ is the jth component of the solution to the model in Eq. (3); $y_{m,j}^{(k)}$ is the jth component of the measurement made for the solution at the time point $t_k \in [T_b, T_e]$; n_m is the number of the measured solution components; T_e, T_b are the end and start times of data recording. There are different possibilities to tackle this problem, with varying degrees of verification associated with them. The available options are:

F0 Do we use the formula in Eq. (7)?

F0.a (no) Experimental/neural networks/other **F0.b (yes)** Least squares

F1 How exactly is $y(t_k, p)$ obtained?

F1.a A closed-form solution

F1.b Approximation by an expression (e.g., Euler's method)

F1.c Numerical solution from a "black-box" solver

F2 What is the underlying technique for the implementation?

F2.a Fixed or floating point **F2.b** Interval **F2.c** Other verified

F3 How do we represent the measured data $y_{m,j}^{(k)}$?

F3.a As provided by sensors: Hundreds of MB of floating point numbers

F3.b With the help of any reliable means for data reduction

As an example, consider the set of options F0.a-F1.b-F2.b-F3.a. The true solution $y(t,p)$ of the IVP in Eq. (3) can be approximated by an explicit method (e.g., Euler's) in its interval version taking into account numerical errors but not the discretization error. For Euler's method, the formula is $\mathbf{y}^{(k)} := \mathbf{y}^{(k-1)} + h \cdot f(\mathbf{y}^{(k-1)}, \mathbf{p})$ for a constant step size $h := t_k - t_{k-1}$. The interval approximation $\mathbf{y}^{(k)}$ at t_k is then substituted for the exact solution $y(t_k, p)$ in the cost function (7) and the discretization error ignored. For the example we consider in this paper as an illustration (cf. Section 4.3), the step size is chosen to be equal to the sampling time for the data, $h = 1$. In general, for this approach to give good results, that is, for $\mathbf{y}^{(k)}$ to be an acceptable approximation of the true solution $y(t_k, p)$, this sampling time (or the step size) should be significantly smaller than the dominant time constant of the process described by the IVP. The approximated cost function can then be rewritten as

$$\Phi_{app}(p) = \sum_{k=T_b}^{T_e} \sum_{j=1}^{n_m} \left(y_j^{(k-1)} - y_{m,j}^{(k)} + h \cdot f_j(y^{(k-1)}, p) \right)^2 , \tag{8}$$

where $y^{(T_b-1)}$ is the initial condition. Although the whole process is not verified, even if interval optimization procedures are applied, the overall verification degree is high if the first (and second) derivatives of Φ_{app} are computed exactly with the help of AD.

Option F3 deserves a separate discussion. On modern multi-processor systems, high-bandwidth interconnect technologies and large amounts of expandable main memory allow for quick access to terabytes of data, either stored locally or in a distributed manner. By contrast, GPUs typically have a fixed amount of on-board memory that is often limited to a fraction of the capacity available on the CPU. So in case of option F3.a, the question arises how large data sets should be handled on the GPU. Data partitioning and sequential processing of small data blocks at a time involves many expensive memory transfers. One idea is to interpolate the data, for example, by a (piecewise) scalar Bézier curve $b(t)$ depending on time with a predefined degree and enclose them in a verified way by a corresponding interval extension $b_{[]}(t)$. That is, the kernel functions depending on data can be implemented much easier and more efficiently on the GPU using Option F3.b.

Using the CPU as the basis, we explored, for example, the option F0.b-F1.b-F2.b-F3.a for different kinds of solid oxide fuel cell models in [1, 19]. The option F0.a-F1.c-F2.a-F3.a is studied for a distributed heating system on the GPU in [2].

3.2 Optimization Algorithms

In this subsection, we describe in detail different GPU-based optimization approaches depending on the choices made according to the general options from Section 3.1. All of them are brute-force approaches; the availability of the cheap

GPU computing power makes it possible to carry them out in acceptable time. Note that the algorithms proposed here do not try to parallelize a sequential optimization algorithm, but rather execute the sequential approach for each considered data item in parallel.

3.2.1 Experimental Identification

Let us suppose that measurements $y_{m,j}^{(k)}$ for specified solution components $y_j(t)$ are available at all times $t_k \in \{T_b, \dots, T_e\}$. For these, plausibility bounds $\mathbf{y}_{m,j}$ can be derived on the basis, for example, of physical constraints or of tolerances of measurement devices. This information can be used to reduce the initial search box $\mathbf{p}$ for optimal parameters. A general scheme for a brute-force algorithm is

Step 1 Bisect $\mathbf{p} = \bigcup_{j=1}^{l} \mathbf{p}_j$ until a predefined bound on the width of $\mathbf{p}_j$ is reached

Step 2 Compute an enclosure $\mathbf{y}_i^{(k)}$ of $\mathbf{y}_i(t_k, \mathbf{p}_j)$ in parallel

Step 3 Exclude $\mathbf{p}_j$ if $\exists k, i$: $\mathbf{y}_i^{(k)} \cap \mathbf{y}_{m,i}^{(k)} = \emptyset$ (in parallel)

Result List $\mathcal{L} \subseteq \{1, \dots, l\}$ containing indices of suitable boxes

Steps 2 and 3 can be performed on the GPU, for example, in floating-point arithmetic using the Smith theorem [32]. Moreover, the plausibility test in Step 3 can be replaced by any other test mentioned in the next subsection. One possible use for the results is to build the hull of $\mathbf{p}_j$ for $j \in \mathcal{L}$ to obtain a reduced search space. The algorithm can then be reiterated if necessary using the tighter search interval with a lower bisection bound. A possibly better use is to compute the convex hull of enclosures for $\mathbf{y}(t_k, \mathbf{p}_j)$ for $j \in \mathcal{L}$ (again on the GPU). This new enclosure can be good enough even without subsequent least squares.

3.2.2 Preconditioning

When the parameter domain has a large plausibility range, different preliminary tests can be applied to exclude boxes from the search [18, 33]. In the brute-force algorithm in the previous subsection, the test in Step 3 can be replaced by various other preconditioning approaches.

The midpoint test is one possibility for such preconditioning. A box $\mathbf{p}_j$ is excluded from the search space if $\underline{c_{[]}(\mathbf{p}_j)} > \overline{\mathbf{z}^*}$, where $\mathbf{z}^*$ is an enclosure of the global minimum and $c_{[]}$ an interval extension of the cost function c [18]. For example, we know that the least squares cost function $\Phi(p)$ from Eq. (7) has zero as its global, ideal minimum. This global minimum, however, might not be known beforehand in general, or cannot be reached. On the GPU, where all tests are virtually executed at the same time, this value must be chosen carefully.

A more elaborate option is the monotonicity test. A box $\mathbf{p}_i$ is excluded from the search if it fails to satisfy the condition $0 \in \nabla c_{[]}(\mathbf{p}_i)$ (∇ meaning the exact gradient). In this paper, we apply the monotonicity test in parallel on the GPU, see the example in Section 4. We plan to consider the convexity test in our future work.

3.2.3 Set Inversion Approach

Nonlinear parameter identification can be characterized as a set inversion problem. SIVIA — set inversion via interval methods — is an algorithm frequently employed in optimization and parameter estimation tasks involving quantities with bounded uncertainty [15]. There are different variations of the algorithm; Algorithm 1 describes its simplest form. Given an interval function $c_{[]}$, an image $\mathbf{z}$ and an initial search domain $\mathbf{p}_0$, the algorithm tests whether $c_{[]}(\mathbf{p}_i)$ is a real subset of $\mathbf{z}$, $\mathbf{p}_i$ being sub-boxes of $\mathbf{p}_0$ obtained by bisection, until the width of a box is at most ϵ in its width. Boxes that certainly are preimages of $\mathbf{z}$ are part of the solution set $\mathcal{S}$ (also called *paving*), boxes containing no solution are members of $\mathcal{N}$ and boxes with the width smaller than ϵ belong to the boundary set $\mathcal{E}$.

Algorithm 1 Generic SIVIA algorithm.

Require: $c_{[]}$, $\mathbf{z}$, $\mathbf{p}_0$, ϵ
1: $\mathcal{S} \leftarrow \emptyset$, $\mathcal{N} \leftarrow \emptyset$, $\mathcal{E} \leftarrow \emptyset$, $\mathcal{L} \leftarrow \{\mathbf{p}_0\}$
2: **while** $\mathcal{L} \neq \emptyset$ **do**
3: $\quad \mathbf{p} \leftarrow \text{pop}(\mathcal{L})$
4: $\quad$ **if** $c_{[]}(\mathbf{p}) \subset \mathbf{z}$ **then**
5: $\quad\quad \text{push}(\mathcal{S}, \mathbf{p})$
6: $\quad$ **else if** $c_{[]}(\mathbf{p}) \cap \mathbf{z} = \emptyset$ **then**
7: $\quad\quad \text{push}(\mathcal{N}, \mathbf{p})$
8: $\quad$ **else if** $diam(\mathbf{p}) < \epsilon$ **then**
9: $\quad\quad \text{push}(\mathcal{E}, \mathbf{p})$
10: $\quad$ **else**
11: $\quad\quad \mathbf{p}_L, \mathbf{p}_R \leftarrow \text{bisect}(\mathbf{p})$
12: $\quad\quad \text{push}(\mathcal{L}, \mathbf{p}_L)$
13: $\quad\quad \text{push}(\mathcal{L}, \mathbf{p}_R)$
14: $\quad$ **end if**
15: **end while**
16: **return** $\mathcal{S}, \mathcal{N}, \mathcal{E}$

The approach from Algorithm 1 can be parallelized in two basic ways:

PSIVIA: a priori subdivision of $\mathbf{p}_0$ into small boxes with subsequent inclusion test in parallel, or

NSIVIA: coarse subdivision of $\mathbf{p}_0$, then parallel execution of SIVIA on sub-boxes.

In the first version, PSIVIA, the bisection step is eliminated by the pre-computation of a grid, making the remaining part of the algorithm *perfectly parallelizable*. Since all boxes are similar in size, the inclusion of a box into the set $\mathcal{E}$ is no longer determined by the box width. Instead, the remaining boxes, being neither a subset of $\mathbf{z}$ nor having an empty intersection with $\mathbf{z}$, satisfy the condition $\mathbf{p}_i \cap \mathbf{z} \neq \emptyset$, qualifying as boundary boxes. This type of lightweight algorithm benefits the most

Algorithm 2 Parallelized SIVIA.

Require: $c_{[]}$, $\mathbf{z}$, $\mathbf{p}_0$
$\mathcal{S} \leftarrow \emptyset$, $\mathcal{N} \leftarrow \emptyset$, $\mathcal{E} \leftarrow \emptyset$
$\mathcal{L} \leftarrow \{\text{subdivide}(\mathbf{p}_0)\}$
for all $\mathbf{p}_i \in \mathcal{L}$ **do**
  if $c_{[]}(\mathbf{p}_i) \subset \mathbf{z}$ **then**
    push($\mathcal{S}$, $\mathbf{p}_i$)
  else if $c_{[]}(\mathbf{p}_i) \cap \mathbf{z} = \emptyset$ **then**
    push($\mathcal{N}$, $\mathbf{p}_i$)
  else
    push($\mathcal{E}$, $\mathbf{p}_i$)
  end if
end for
return $\mathcal{S}, \mathcal{N}, \mathcal{E}$

from parallel processing, especially on the GPU. Algorithm 2 shows the reformulated, parallelizable SIVIA procedure. In parameter identification scenarios, we are interested in enclosures containing the system parameters. Depending on the subdivision strategy, the optimal enclosure box might be situated directly on the boundary of two boxes in the subdivision. Hence, forming a convex hull of all boxes contained in $\mathcal{S}$ might seem sensible. However, the solution set $\mathcal{S}$ might consist of non-adjacent boxes, or be a set of adjacent boxes enveloping a set that does not contain any solutions. In the worst case, the hull of such a paving will be marginally smaller than, or equal to, the initial search domain $\mathbf{p}_0$. While this box might be guaranteed to contain optimal parameters, it might also get too large to lead to any meaningful conclusions about the problem under study. From this perspective, building a convex hull over solutions $y(t_k, \mathbf{p}_i)$ to Eq. (3) over each $\mathbf{p}_i \in \mathcal{S}$ (or, possibly, $\mathcal{S} \cup \mathcal{E}$) at each t_k might be a more useful option. Another possibility is to treat boxes from $\mathcal{S}$ as entities in an n-dimensional feature space so that we can apply density-based clustering to group suitable boxes into smaller sets as described in Algorithm 3. The result of that algorithm is a set of boxes $\mathcal{C}$, where each box contains at least one box belonging to $\mathcal{S}$, ultimately containing the entire $\mathcal{S}$.

4 Test Application: Two-Compartment Model

In this section, we test the performance of the methods introduced in Section 3 by applying them to the example of the two compartment model similarly to [17].

4.1 Problem Description

The two-compartment model described in [14] has been a standard example in parameter identification since many years. It describes the evolution of two interconnected compartments after an impulse. This dynamic system can be described

Algorithm 3 Fast interval clustering.

Require: $\mathcal{B},\ \theta$
1: $\mathcal{C} \leftarrow \emptyset$
2: **for all** $\mathbf{b}_i \in \mathcal{B}$ **do**
3: $\quad processed \leftarrow false$
4: $\quad$ **for all** $\mathbf{q}_j \in \mathcal{C}$ **do**
5: $\quad\quad$ **if** $\max_k |\mathbf{b}_{i,k} - \mathbf{q}_{j,k}| \leq \theta$ **then**
6: $\quad\quad\quad \mathbf{q}_i \leftarrow \mathbf{b}_i \cup \mathbf{q}_j$
7: $\quad\quad\quad processed \leftarrow true$
8: $\quad\quad$ **end if**
9: $\quad$ **end for**
10: $\quad$ **if** $\neg processed$ **then**
11: $\quad\quad$ push($\mathcal{C}$, $\mathbf{b}_i$)
12: $\quad$ **end if**
13: **end for**
14: **return** $\mathcal{C}$

by a two-dimensional first-order ODE of the form

$$\begin{aligned} \dot{y}_1 &= -(p_3 + p_1) \cdot y_1 + p_2 \cdot y_2 \\ \dot{y}_2 &= p_3 \cdot y_1 - p_2 \cdot y_2 \end{aligned} \tag{9}$$

with the initial condition $\mathbf{y}(0) = (1, 0)^T$. We are interested in the values of p_1, p_2, p_3 given (measurement) data. The problem is formulated such that only the second component y_2 is measured (i.e., the data is given only for y_2). These linear ODEs can be solved analytically:

$$y_2(t) = \alpha \cdot \left(e^{-\lambda_1 \cdot t} - e^{-\lambda_2 \cdot t}\right) \tag{10}$$

with its macro-parameters defined as

$$\begin{aligned} D &= \sqrt{(p_1 - p_2 + p_3)^2 + 4p_2 \cdot p_3}\,, \quad \alpha = \frac{p_3}{D}\,, \\ \lambda_1 &= \frac{1}{2}(p_1 + p_2 + p_3 - D)\,, \quad \lambda_2 = \frac{1}{2}(p_1 + p_2 + p_3 + D)\,. \end{aligned} \tag{11}$$

Following [17], this exact solution can be reformulated to reduce overestimation as

$$y_2(t) = \left(p_3 \cdot e^{\frac{-p_3 \cdot t}{2}}\right)\left(e^{\frac{-p_2 \cdot t}{2}}\right)\left(e^{\frac{-p_1 \cdot t}{2}}\right)\left(e^{\frac{D \cdot t}{2}} - e^{-\frac{D \cdot t}{2}}\right)/D\,. \tag{12}$$

We implemented three GPU-based approaches to reduce the search space for the parameter estimation and applied them to the two-compartment model: the experimental approach from Section 3.2.1 involving the black-box solver `odeint`, taking advantage of the system's cooperativity (I), the monotonicity test from Section 3.2.2 (II) and PSIVIA from Section 3.2.3 with the image $\Phi_{[]}(\mathbf{p}) \subset [0, 0.005]$

(III). All three methods have been tested with raw data points for y_2 in combination with the respective plausibility bounds copied to the GPU's memory and a continuous enclosure of the measurements by interval Bézier curves. Methods II and III were additionally tested with the exact solution from Eq. (12) and an approximation of $y(t)$ by using Euler's method as described in the formula from Eq. (8). The scenarios we considered in this paper are therefore defined as follows with respect to the parameter identification options mentioned in Section 3.1: Ia and Ib correspond to F0.a-F1.c-F2.a-F3.a/b; IIa/IIIa and IIb/IIIb to F0.b-F1.a-F2.b-F3.a/b; IIc/IIIc and IId/IIId to F0.b-F1.b-F2.b-F3.a/b. Note that IIa/IIIa and IIb/IIIb use only verified operations and algorithms and in this way can be considered verified on the GPU.

4.2 Reference System and Testing Setup

With an emphasis on reproducibility of the experiments, measurement data were synthesized by the following procedure. First, system parameters were defined as $\mathbf{p} = (0.232718, 1.925403, 0.145076)^T$. Inserting $\mathbf{p}$ into the exact solution from Eq. (10), we then calculated the values of $y_2(t)$ from $T_b = 1$ to $T_e = 16$ with a step size $h = 1$. Afterwards, small pseudo-random FP numbers were added to each value to achieve a simulated measurement noise. For random number generation, the well-known Classic Mersenne Twister [24] with a seed value of 1788 was used. The resulting uniform distribution was then transformed into a normal distribution with a mean value of $\mu = 0$ and a standard deviation $\sigma = \frac{1}{2^9}$. We used the C++ standard library function `std::normal_distribution`, which is based on the Box-Muller transform. Finally, the results were truncated after the sixth decimal place. Table 1 shows the data points acquired in that way.

Table 1: Artificial measurements for y_2 (with added plausibility bounds of $\pm$ 0.007).

t	1	2	3	4	5	6	7	8
$y_{m,2}$	0.051801	0.046753	0.040787	0.032543	0.024291	0.021511	0.020577	0.012213

t	9	10	11	12	13	14	15	16
$y_{m,2}$	0.008919	0.007634	0.004929	0.008158	0.006349	0.002386	0.005543	0.004355

All computations were performed on

- a CPU system with an Intel Xeon Gold 5215 64 bit CPU with 192 GB of main memory and
- a GPU system with an NVIDIA Quadro RTX 6000 GPU with 24 GB of memory,

running the operating system Ubuntu 22.04 with CUDA 12 and the NVIDIA driver 525.105 installed. Experiments on the GPU were written in the CUDA C++. For interval computations on the CPU, we used the `interval` library from Boost. On the GPU, we used a modified version of the extended `cuda_interval_lib` from [8],

originally published in [7]. Device-side automatic differentiation of interval data types was made possible by a custom port of the forward-mode differentiation from FADBAD [6] to CUDA C++ (both described in more detail in Section 2.3). Function execution times were measured by wrapping the relevant parts of the program between two calls to `std::chrono::high_resolution_clock::now()` and calculating their difference at nanosecond resolution. The execution time of entire programs was captured by the standard UNIX command `time`. Since the testing environment was accessed exclusively via a remote connection, measurements of power consumption were limited to software-based tools. According to [22], the reported sampling time of power usage information on NVIDIA GPUs is 20 ms. Taking into account that a kernel might complete its operation in less time for the considered relatively small example, these measurements are only rough approximations.

4.3 Results

We conducted a series of (standardized) tests to compare the approaches presented in the previous sections. Given the measurements in Table 1, we chose the (hypothetical) plausibility bounds $\mathbf{y}_{m,2} = [y_{m,2} \pm 0.007]$, the search space $\mathbf{p} = [0.01, 1] \times [1, 2] \times [0.05, 2]$, and a maximum box width $w = 0.05$, which resulted in a subdivision containing 2^{16} possible boxes. The search space was chosen in such a way as to exclude possible symmetric parameter values; theoretically, only one optimum is contained in $\mathbf{p}$ if we consider the data generation procedure described in Section 4.2. The averaged results for 100 test trials for each method are shown in Table 2.

The enclosure $\mathbf{p}^*$ is the convex hull of all suitable boxes after search space reduction. To achieve a fair comparison between all methods, the enclosure of the SIVIA-like method includes the boundary set $\mathcal{E}$ as well, since the other two methods make no distinction between uncertain boxes and definite solutions. In Column *Boxes*, the number of boxes in the corresponding solution list is shown. Kernel time t_{kern} represents the execution time of each method on its own, while t_{wall} is the total number of seconds elapsed (wall-clock time) for the entire program execution, including post-processing. During each test, peaks in CPU and GPU power usage were recorded. The highest enduring peaks during kernel execution P_{peak} are listed in the last column.

The scenarios I, II and III were described in Section 4.1. For comparison, we also tested the example with a CPU-based C++ implementation of SIVIA (IV) and with the `@infsup/fsolve` function (V) provided by the `interval` package for GNU Octave [12]. Decimal values in the table are rounded outwards/to nearest to four decimal places for improved readability.

Based only on enclosure size, the best results are obtained by the experimental method. Although its execution time is also fairly low, this method has the highest peak in power consumption. This comes as no surprise since the use of the black-box solver `odeint` adds a significant amount of computations. This impact on energy efficiency might be a factor to consider when solving more complex problems. Moreover, although the bulk of uncertainty is captured by this approach, the

Table 2: Test results for the parameter space $\mathbf{p} = [0.01, 1] \times [1, 2] \times [0.05, 2]$.

Method		$\mathbf{p}^*$	Boxes	t_{kern} (s)	t_{wall} (s)	P_{peak} (W)
GPU	Ia	$[0.0718, 0.505] \times [1, 2] \times [0.05, 0.2024]$	597	0.0579	0.3664	39.2905
	Ib	$[0.0718, 0.505] \times [1, 2] \times [0.05, 0.2024]$	548	0.0568	0.3664	39.1510
	IIa	$[0.01, 1] \times [1, 2] \times [0.05, 2]$	47 531	22.9567	25.3034	90.5923
	IIb	$[0.01, 1] \times [1, 2] \times [0.05, 2]$	47 509	22.9817	25.3283	90.5746
	IIc	$[0.01, 1] \times [1, 2] \times [0.05, 2]$	65 536	28.547	31.8037	90.8255
	IId	$[0.01, 1] \times [1, 2] \times [0.05, 2]$	65 536	28.3206	31.7399	90.5315
	IIIa	$[0.01, 1] \times [1, 2] \times [0.05, 0.6899]$	11 215	0.0056	0.4193	37.6086
	IIIb	$[0.01, 1] \times [1, 2] \times [0.05, 0.6899]$	11 178	0.0288	0.4358	37.6487
	IIIc	$[0.01, 1] \times [1, 2] \times [0.05, 0.8727]$	12 910	0.0038	0.6337	36.2425
	IIId	$[0.01, 1] \times [1, 2] \times [0.05, 0.8727]$	12 899	0.0274	0.625	36.0214
CPU	IVa	$[0.01, 1] \times [1, 2] \times [0.05, 0.5985]$	8 877	0.4384	0.4548	46.7245
	Va	$[0.01, 1] \times [1, 2] \times [0.05, 0.7875]$	1 067	26.781	29.117	49.572

computations are not verified.

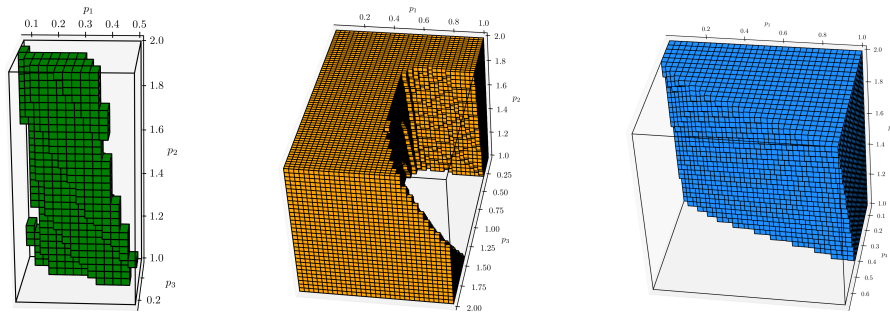
Although the monotonicity test should actually be less computationally expensive, it takes considerably longer than the previous method, while reducing the initial search space only slightly (not even noticeable from the convex hull). Because only a fraction of the search space has been discarded (18 005 out of 65 536), this test alone is not sufficient for search space reduction, at least in this scenario. The high execution time is a result of our GPU version of FADBAD still having essentially the same structure as the version on the CPU, where memory access is not as limited and expensive as on the GPU. Using the Euler-based approximation of the solution to (3) in IIc/d does not help to exclude any boxes.

Out of the three GPU-based methods considered here, PSIVIA shows the best performance in terms of computation time and power usage due its simplicity, even surpassing the traditional CPU-based SIVIA implementation in execution time, despite being a brute-force approach. In Figure 1, visualizations of the reduced parameter space are shown after each of the methods with the exact solution and measurement data provided was applied.

If only the set of guaranteed boxes $\mathcal{S}$ is considered for PSIVIA, the remaining enclosure is

$$[0.2265, 0.4122] \times [1.2187, 2] \times [0.0804, 0.2024],$$

outperforming the experimental approach wrt. the width of the convex hull. When we apply Algorithm 3 with $\theta = 0.25$ as part of the post-processing to the solution set produced by PSIVIA (using the exact solution and no interpolation), the resulting boxes are



(a) $\mathbf{p}^*_{\text{Ia}}$ (b) $\mathbf{p}^*_{\text{IIa}}$ (c) $\mathbf{p}^*_{\text{IIIa}}$

Figure 1: Visualizations of the reduced parameter space after GPU computations.

$$[0.2265, 0.4122] \times [1.7187, 2] \times [0.1109, 0.2024],$$
$$[0.2575, 0.4122] \times [1.4375, 1.7188] \times [0.1109, 0.1718],$$
$$[0.2884, 0.4122] \times [1.2187, 1.4375] \times [0.0804, 0.1415],$$

the first of which is a small enclosure of the original parameters used for synthesizing our measurements (see Section 4.1).

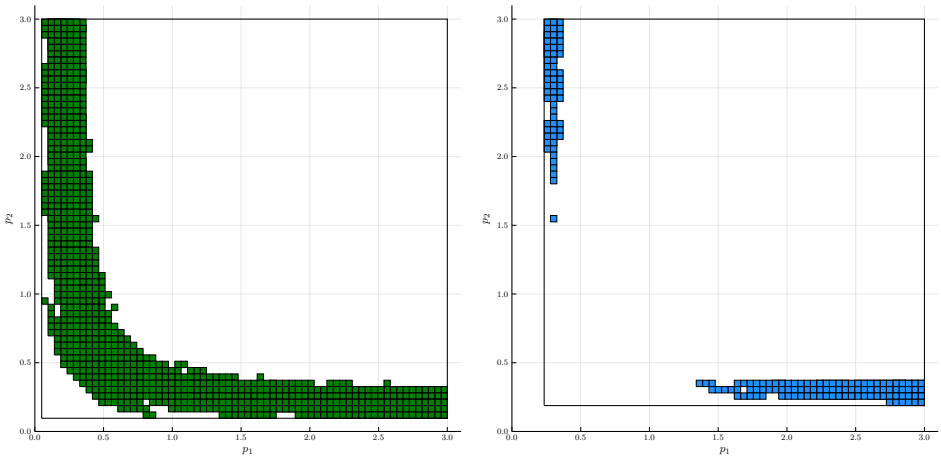
At this scale, preferring interpolated measurement data over raw data points adds some computational overhead, despite global memory access being a performance bottleneck. Furthermore, it can be observed that Euler's method is not suitable for this task. In the case of PSIVIA, the time horizon of 16 steps is long enough for the wrapping effect to significantly impact the resulting enclosure, leading to intervals so large in width that the criterion $\Phi_{app}(\mathbf{p}_i) \subset \mathbf{z}$ cannot be satisfied anymore. As a result, all suitable boxes belong to the boundary set $\mathcal{E}$, while $\mathcal{S}$ remains empty.

Considering a wider search space $\mathbf{p} = [0.05, 3] \times [0.05, 3] \times [0.01, 2]$ with w unchanged, the tests were repeated. Results are shown in Table 3.
Depending on the method used, we observe different increases in execution time. While the experimental approach is more than two times slower than before, both

Table 3: Test results for a larger parameter domain $\mathbf{p} = [0.05, 3] \times [0.05, 3] \times [0.01, 2]$.

	Method	$\mathbf{p}^*$	Boxes	t_{kern} (s)	t_{wall} (s)	P_{peak} (W)
GPU	Ia	$[0.05, 3] \times [0.0960, 3] \times [0.0410, 0.2899]$	1 420	0.1369	0.9377	60.5923
	IIa	$[0.05, 3] \times [0.05, 3] \times [0.01, 2]$	182 603	90.0465	156.2434	95.1462
	IIIa	$[0.05, 3] \times [0.05, 3] \times [0.01, 2]$	102 618	0.0222	1.5053	35.5917
CPU	IVa	$[0.05, 3] \times [0.05, 3] \times [0.01, 2]$	102 618	4.5162	4.6874	60.91

the monotonicity test and PSIVIA need approximately four times of the previous kernel time. However, this larger search space highlights the advantage of using the GPU; now the CPU implementation is significantly slower than both methods I and III. Another factor to take into account is that $\mathbf{p}$ now contains a symmetric solution [14]. In Figure 2, the effectiveness of method III compared to I is highlighted at this scale (under the assumption that only $\mathcal{S}$ and not the boundary $\mathcal{E}$ is considered).



(a) $\mathbf{p}^*_{\mathrm{Ia}}$ (b) $\mathcal{S}_{\mathrm{IIIa}}$

Figure 2: Visual comparison of methods I and III for a larger parameter domain (top view of p_1 and p_2).

5 Conclusions

In this paper, we demonstrated — for the first time — a solid combination of IA with AD capabilities in C++ employed on a graphics processor, using enhanced public domain software. Furthermore, we applied an experimental set-based and two actually verified methods to a well-known example of a two-compartment problem and compared them wrt. computational cost, both in terms of execution time and approximate power consumption, as well as wrt. the quality of the resulting convex hull of the parameter enclosures. In regard to the rising interest in energy-efficient software design and "green computing", the comparison of power consumption is of particular interest. The PSIVIA method seems to be a good compromise between the power consumption and solution quality, even outperforming other considered methods if only the definite solution set $\mathcal{S}$ is taken into account. Using a data reduction method, although easier to implement, could not be shown to lead to better run times in case of the simple test scenario we considered. However, we expect it to be so for more complex examples. Likewise, using the Euler approximation

did not bring any significant reduction of the initial search space in the considered scenario, the cause of which was that the data sampling time of $h = 1$s was too large compared to the dominant time constants of the process. Nonetheless, we expect it to be helpful in more complex scenarios with appropriate sampling times. Finally, we succeeded in obtaining a tight enclosure of the optimum on the GPU.

It remains to be seen how the methods presented in this paper scale up when applied to more complex and close-to-life problems. On the one hand, good performance of our GPU-based set-inversion approach indicates that this type of hardware might allow algorithms like SIVIA to solve problems of higher dimensionality than currently possible. On the other hand, we aim to improve the currently low performance of interval-based AD on the GPU, because our ultimate goal is to implement a GPU-enabled verified solver for differential equations, for which the ability to perform set-based AD is essential.

Our results indicate that AD at run time might not be an ideal concept for the GPU, because it requires every kernel to produce essentially the same derivative, which is expensive computationally. A more elegant approach would be to compute a function's derivatives once and then make them available to all GPU kernels, which we plan to test in our future work. Even better results might be achieved by evaluating derivatives beforehand during compilation. Finally, after an extensive phase of testing the introduced (and extended) GPU-based approaches, we plan to apply the most promising ones in the context of battery systems and fuel cells. To improve our currently purely software-based testing process, we plan to include hardware-based measurement tools for higher accuracy as well as further comparison criteria to assess the performance of our algorithms.

References

[1] Auer, E., Kiel, S., and Rauh, A. Verified parameter identification for solid oxide fuel cells. In *Proceedings of the 5th International Conference on Reliable Engineering Computing*, pages 41–55, 2012. URL: https://rec2012.fce.vutbr.cz/documents/papers/auer.pdf.

[2] Auer, E., Rauh, A., and Kersten, J. Experiments-based parameter identification on the GPU for cooperative systems. *Journal of Computational and Applied Mathematics*, 371:112657, 2020. DOI: 10.1016/j.cam.2019.112657.

[3] Bagóczki, Z. and Bánhelyi, B. A parallel interval arithmetic-based reliable computing method on a GPU. *Acta Cybernetica*, 23(2):491–501, 2017. DOI: 10.14232/actacyb.23.2.2017.4.

[4] Baydin, A. G., Pearlmutter, B. A., Radul, A. A., and Siskind, J. M. Automatic differentiation in machine learning: A survey. *Journal of Marchine Learning Research*, 18:5595–5637, 2017. DOI: 10.5555/3122009.3242010.

[5] Beck, P.-D. and Nehmeier, M. Parallel interval Newton method on CUDA. In *Proceedinngs of the 11th International Conference on Applied Parallel and*

Scientific Computing, pages 454–464. Springer, Berlin, 2013. DOI: 10.1007/978-3-642-36803-5_34.

[6] Bendtsen, C. and Stauning, O. FADBAD, a flexible C++ package for automatic differentiation. Technical report, Department of Mathematical Modelling, Technical University of Denmark, Lyngby, 1996. URL: https://www2.imm.dtu.dk/~kajm/FADBAD/tr17_96.pdf.

[7] Collange, C., Daumas, M., and Defour, D. Interval arithmetic in CUDA. In Hwu, W. W., editor, *GPU Computing Gems Jade Edition*, chapter 9, pages 99–107. Elsevier, 2012. DOI: 10.1016/b978-0-12-385963-1.00009-5.

[8] Eriksen, M. B. and Rasmussen, S. *GPU accelerated parameter estimation by global optimization using interval analysis*. Master's thesis, Aalborg University, 2013. URL: https://projekter.aau.dk/projekter/files/77291483/report.pdf.

[9] Fan, H., Ferianc, M., and Luk, W. Enabling fast uncertainty estimation: Accelerating Bayesian transformers via algorithmic and hardware optimizations. In *Proceedings of the 59th ACM/IEEE Design Automation Conference*, pages 325—330, New York, NY, 2022. Association for Computing Machinery. DOI: 10.1145/3489517.3530451.

[10] González-Arribas, D., Sanjurjo-Rivo, M., and Soler, M. Multiobjective optimisation of aircraft trajectories under wind uncertainty using GPU parallelism and genetic algorithms. *Computational Methods in Applied Sciences*, 2018. DOI: 10.1007/978-3-319-89890-2_29.

[11] Grimmer, M. and Krämer, W. An MPI extension for the use of C–XSC in parallel environments, 2005. Preprint 2005/3, Universität Wuppertal. URL: https://www2.math.uni-wuppertal.de/wrswt/preprints/prep_05_3.pdf.

[12] Heimlich, O. Interval arithmetic in GNU Octave. In *SWIM 2016: 9th Summer Workshop on Interval Methods*, 2016. URL: https://swim2016.sciencesconf.org/data/SWIM2016_book_of_abstracts.pdf.

[13] Ifrim, I., Vassilev, V., and Lange, D. J. GPU accelerated automatic differentiation with Clad. In *Journal of Physics: Conference Series*, page 012043. IOP Publishing, 2023. DOI: 10.1088/1742-6596/2438/1/012043.

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

[15] Jaulin, L. and Walter, E. Guaranteed nonlinear parameter estimation from bounded-error data via interval analysis. *Mathematics and Computers in Simulation*, 35(2):123–137, 1993. DOI: 10.1016/0378-4754(93)90008-I.

[16] Jerrell, M. E. Automatic differentiation and interval arithmetic for estimation of disequilibrium models. *Computational Economics*, 10:295–316, 1997. DOI: 10.1023/A:1008633613243.

[17] Kieffer, M., Markót, M. C., Schichl, H., and Walter, E. Verified global optimization for estimating the parameters of nonlinear models. In *Modeling, Design, and Simulation of Systems with Uncertainties*. Springer, Berlin, 2011. DOI: 10.1007/978-3-642-15956-5_7.

[18] Kieffer, M. and Walter, E. Interval analysis for guaranteed nonlinear parameter estimation. In *MODA 5 — Advances in Model-Oriented Data Analysis and Experimental Design*, pages 115–125, Heidelberg, 1998. Physica-Verlag. DOI: 10.1007/978-3-642-58988-1_13.

[19] Kiel, S., Auer, E., and Rauh, A. Uses of GPU powered interval optimization for parameter identification in the context of SO fuel cells. In *Proceedings of the 9th IFAC Symposium on Nonlinear Control Systems*, pages 558–563. International Federation of Automatic Control, 2013. DOI: 10.3182/20130904-3-FR-2041.00169.

[20] Klatte, R., Kulisch, U., Wiethoff, A., Lawo, C., and Rauch, M. *C–XSC, A C++ Class Library for Extended Scientific Computing*. Springer, Berlin, 1993. DOI: 10.1007/978-3-642-58058-1.

[21] Kozikowski, G. and Kubica, B. J. Interval arithmetic and automatic differentiation on GPU using OpenCL. In *Proceedings of the 11th International Conference on Applied Parallel and Scientific Computing*, pages 489–503. Springer, Berlin, 2013. DOI: 10.1007/978-3-642-36803-5_37.

[22] Lang, J. and Rünger, G. High-resolution power profiling of GPU functions using low-resolution measurement. In *Proceedings of the 19th International Conference on Parallel Processing*, pages 801–812. Springer, Berlin, 2013. DOI: 10.1007/978-3-642-40047-6_80.

[23] Marvel, S. W., de Luis Balaguer, M. A., and Williams, C. M. Parameter estimation in biological systems using interval methods with parallel processing. In *Proceedings of the 8th International Workshop on Computational Systems Biology*, pages 129–132, 2011.

[24] Matsumoto, M. and Nishimura, T. Mersenne twister: A 623-dimensionally equidistributed uniform pseudo-random number generator. *ACM Transactions on Modeling and Computer Simulation*, 8(1):3–30, 1998. DOI: 10.1145/272991.272995.

[25] Meslem, N. and Ramdani, N. Interval observer design based on nonlinear hybridization and practical stability analysis. *International Journal of Adaptive Control and Signal Processing*, 25(3):228–248, 2011. DOI: 10.1002/acs.1208.

[26] Moore, R. E. *Interval Analysis*. Prentice-Hall, Englewood Cliffs, NJ, 1966.

[27] Moore, R. E., Kearfott, R. B., and Cloud, M. J. *Introduction to Interval Analysis*. Society for Industrial and Applied Mathematics, 2009. DOI: 10.1137/1.9780898717716.

[28] Pilarek, M. and Wyrzykowski, R. Solving systems of interval linear equations in parallel using multithreaded model and "interval extended zero" method. In *Proceedings of the 9th International Conference on Parallel Processing and Applied Mathematics*, pages 206–214. Springer, Berlin, 2012. DOI: 10.1007/978-3-642-31464-3_21.

[29] Rebner, G. and Beer, M. CUDA accelerated fault tree analysis with C–XSC. In *Scalable Uncertainty Management*, pages 539–549. Springer, Berlin, 2012. DOI: 10.1007/978-3-642-33362-0_41.

[30] Rump, S. M. *INTLAB — INTerval LABoratory*. In *Developments in Reliable Computing*, pages 77–104. Springer, Dordrecht, 1999. DOI: 10.1007/978-94-017-1247-7_7.

[31] Sanders, D. P. and Churavy, V. Branch-and-bound interval methods and constraint propagation on the GPU using Julia. In *Proceedings of the 19th International Symposium on Scientific Computing, Computer Arithmetic, and Verified Numerical Computations*, page 79, Szeged, Hungary, 2021. URL: https://www.inf.u-szeged.hu/scan2020/sites/default/files/scan2020_proceedings.pdf.

[32] Smith, H. L. *Monotone Dynamical Systems: An Introduction to the Theory of Competitive and Cooperative Systems*. Mathematical Surveys and Monographs. American Mathematical Society, Providence, RI, 1995. DOI: 10.1090/surv/041.

[33] Tucker, W. *Validated Numerics: A Short Introduction to Rigorous Computations*. Princeton University Press, 2011. DOI: 10.1515/9781400838974.

[34] Wojtkiewicz, S. F. and Wojtkiewicz, G. Use of GPU computing for uncertainty quantification in computational mechanics: A case study. *Scientific Programming*, 19(4):199–212, 2011. DOI: 10.3233/SPR-2011-0328.

[35] Zimmer, M. Using C–XSC in a multi-threaded environment, 2011. Preprint 2011/2, Universität Wuppertal. URL: https://www2.math.uni-wuppertal.de/wrswt/preprints/prep_11_2.pdf.

Acta Cybernetica **26** (2024) 933–954.

Asymptotically Minimal Interval Contractors Based on the Centered Form — Application to the Stability Analysis of Linear Time-Delayed Differential Equations

Luc Jaulin[a]

Abstract

This paper proposes a new interval-based contractor for nonlinear equations which is minimal when dealing with narrow boxes. The method is based on the centered form classically used by interval algorithms combined with a Gauss Jordan band diagonalization preconditioning. As an illustration in stability analysis, we propose to compute the set of all parameters of a characteristic function of linear time-delayed equations which have at least one zero in the imaginary axis. Our approach is able compute a guaranteed and accurate enclosure of the solution set faster than existing approaches.

Keywords: interval analysis, contractor, centered form, stability

1 Introduction

Interval analysis is an efficient tool used for solving rigorously complex nonlinear problems involving bounded uncertainties [7, 20, 34]. Many interval algorithms are based on the notion of *interval contractor* [8] (or *contractor* for short) which is an operator which shrinks an axis-aligned box $[\mathbf{x}]$ of $\mathbb{R}^n$ without removing any point of the solution set $\mathbb{X}$. The set $\mathbb{X}$ is assumed to be defined by equations involving the components $x_1, \ldots, x_n$ of a vector $\mathbf{x} \in \mathbb{R}^n$.

Combined with a paver [38] which bisects boxes, the contractor builds an outer approximation of the set $\mathbb{X}$. The resulting methodology can be applied in several domains of engineering such as identification [32], localization [21, 14], SLAM [29, 37], vision [11], reachability [13], control [3, 41], calibration [26], etc.

Centered form is one of the most fundamental brick in interval analysis. It is traditionally used to enclose the range of a function over narrow intervals [28, 30, 16]. The quadratic approximation property, guarantees an asymptotically small

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

DOI: 10.14232/actacyb.306222

overestimation for sufficiently narrow boxes. Now, the centered form is only for the forward interval evaluation of a function. The backward propagation is not treated by the classical centered form. Now, this backward step is mandatory is we want to implement a propagation process. This is why we need to build an interval contractor which contains not only a forward interval evaluation, but also the backward propagation. In this paper, we propose to use the centered form to build efficient contractors [17] that are optimal when the intervals are narrow. To my knowledge, no other contractor with this asymptotic property exists in the literature.

To achieve this goal, we first get a guaranteed first order enclosure of each equation composing our problem using an interval linearization technique. Then, we combine these constraints preserving the first order approximation using interval linear techniques. More particularly, we propose to use a preconditioning method based on a Gauss-Jordan band diagonalization. We show that our approach is guaranteed to enclose all solutions of the problem and may outperform state of the art techniques on an example taken from the literature.

The main contribution of this paper is that the contractor we propose is asymptotically minimal, *i.e.*, it is minimal when the boxes are small. To the best of my knowledge, such a contractor does not exist in the literature even if some use a linear approximation (see the X-Taylor iteration [1] tested on global minimization problems, [6] which is similar to X-Taylor but for solving inequalities, the interval Newton [28] used for solving square nonlinear systems, or the affine arithmetic [12] which has been used for non-square systems but which is not asymptotically minimal).

Section 2 recalls some useful mathematical notions related to the sensitivity of the solution set of a linear system. Section 3 introduces wrappers to approximate accurately a function over a box. Section 4 defines what is an asymptotically minimal contractor and Section 5 gives an algorithm to generate it. The relevance and the efficiency of our approach are shown in Section 6 on the stability analysis of a linear differential equation with delays. Section 7 concludes the paper.

2 Preliminaries

This section recalls some basic definitions and theorems related to the sensitivity of the solution set of a linear system with respect to small perturbations. They will be used later in the paper to define the asymptotic minimality of our approximation for the solution set.

2.1 Proximity

Denote by $L(\mathbf{a}, \mathbf{b})$ the distance between $\mathbf{a}$ and $\mathbf{b}$ of $\mathbb{R}^n$ induced by the L-norm [5]. As illustrated by Figure 1, the *proximity* of $\mathbb{A}$ to $\mathbb{B}$, where $\mathbb{A}$ and $\mathbb{B}$ are closed

subsets of $\mathbb{R}^n$, is defined by

$$h(\mathbb{A},\mathbb{B}) = \sup_{\mathbf{a}\in\mathbb{A}} L(\mathbf{a},\mathbb{B}) \tag{1}$$

where

$$L(\mathbf{a},\mathbb{B}) = \inf_{\mathbf{b}\in\mathbb{B}} L(\mathbf{a},\mathbf{b}). \tag{2}$$

The norm L that will be used later in the algorithm will be the L_∞ norm, even if, in the pictures, for a better visibility, we use the Euclidean L_2 norm.

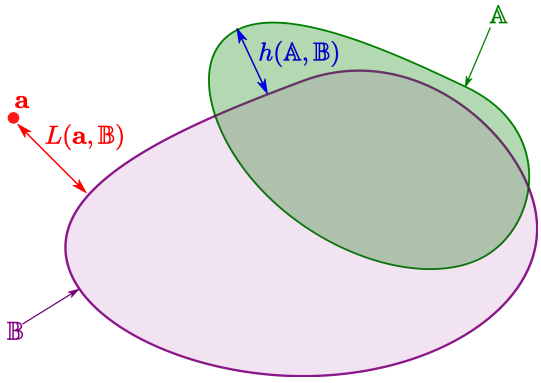


Figure 1: Proximity $h(\mathbb{A},\mathbb{B})$ of $\mathbb{A}$ to $\mathbb{B}$. If we inflate $\mathbb{B}$ by a coefficient of $h(\mathbb{A},\mathbb{B})$, then $\mathbb{B}$ will enclose $\mathbb{A}$

A nested sequence of closed subsets $\mathbb{B}(k) \subset \mathbb{R}^n$, $k \in \mathbb{N}$ is converging to $\mathbf{x}$ if

$$\lim_{k\to\infty} h(\mathbb{B}(k), \{\mathbf{x}\}) = 0. \tag{3}$$

2.2 Linear systems

Consider a system of linear equations of the form $\mathbf{A}\cdot\mathbf{x} = \mathbf{b}$ with more variables than unknows. Denote by $\mathbb{X}$ the solution set. This set can can be a point (if $\mathbf{A}$ is square), a line, a plane, or any affine space. Consider $\bar{\mathbf{x}} \in \mathbb{X}$. If we change just a little the entries for $\mathbf{A}$ and $\mathbf{b}$, the solution set $\mathbb{X}$ will move also. The point $\bar{\mathbf{x}}$ will then probably be outside $\mathbb{X}$, but still close to the new $\mathbb{X}$. The corresponding distance is $L(\bar{\mathbf{x}},\mathbb{X})$. The following proposition allows us to quantify the value for $L(\bar{\mathbf{x}},\mathbb{X})$ or equivalently to provide a sensitivity for the system $\mathbf{A}\cdot\mathbf{x} = \mathbf{b}$.

Proposition. *Consider a point* $\mathbf{x}$ *which satisfies the linear system* $\mathbf{A}\cdot\mathbf{x} = \mathbf{b}$*, where* $\mathbf{A}$ *has independent rows, i.e., which is full rank. Consider a small variation* $d\mathbf{A}$ *of* $\mathbf{A}$*. The quantity*

$$d\mathbf{x} = -\mathbf{A}^\dagger \cdot (d\mathbf{A}\cdot\mathbf{x} + d\mathbf{A}\cdot d\mathbf{x}), \tag{4}$$

where

$$\mathbf{A}^\dagger = \mathbf{A}^T(\mathbf{A}\cdot\mathbf{A}^T)^{-1} \tag{5}$$

is the generalized inverse of $\mathbf{A}$*, satisfies*

$$(\mathbf{A} + d\mathbf{A}) \cdot (\mathbf{x} + d\mathbf{x}) = \mathbf{b}. \tag{6}$$

This proposition tells us that if we move $\mathbf{A}$ a little, then, the solution set for the linear equation moves a little also, at order 1.

Proof. We have

$$\begin{array}{ll} & (\mathbf{A} + d\mathbf{A}) \cdot (\mathbf{x} + d\mathbf{x}) = \mathbf{b} \\ \Leftrightarrow & \mathbf{A} \cdot \mathbf{x} + \mathbf{A} \cdot d\mathbf{x} + d\mathbf{A} \cdot \mathbf{x} + d\mathbf{A} \cdot d\mathbf{x} = \mathbf{b} \end{array} \tag{7}$$

Thus

$$\mathbf{A} \cdot d\mathbf{x} + d\mathbf{A} \cdot \mathbf{x} + d\mathbf{A} \cdot d\mathbf{x} = \mathbf{0} \tag{8}$$

i.e.

$$\mathbf{A} \cdot d\mathbf{x} = -d\mathbf{A} \cdot \mathbf{x} - d\mathbf{A} \cdot d\mathbf{x} \tag{9}$$

Since $\mathbf{A}$ has independent lines, the solution which minimizes $\|d\mathbf{x}\|$ is

$$d\mathbf{x} = \mathbf{A}^{\dagger} \cdot (-d\mathbf{A} \cdot \mathbf{x} - d\mathbf{A} \cdot d\mathbf{x}). \tag{10}$$

□

Corollary. *Consider the hyperplane*

$$\mathcal{P} = \{\mathbf{x} \in \mathbb{R}^n | \mathbf{A} \cdot \mathbf{x} = \mathbf{0}\}, \tag{11}$$

where $\mathbf{A}$ *has independent lines. Consider a small variation* $d\mathbf{A}$ *of* $\mathbf{A}$ *with* $\|d\mathbf{A}\| = O(\varepsilon)$ *where* ε *is small. Take a point* $d\mathbf{x} \in \mathcal{P}$ *with* $\|d\mathbf{x}\| = O(\varepsilon)$*. The distance from* $d\mathbf{x}$ *to* $\tilde{\mathcal{P}} = \{\mathbf{x} \in \mathbb{R}^n | (\mathbf{A} + d\mathbf{A}) \cdot \mathbf{x} = \mathbf{0}\}$ *is* $o(\varepsilon)$*, i.e.,* $O(\varepsilon^2)$*.*

Proof. Denote by $\hat{\mathbf{p}}$ the projection of a point $\mathbf{p} \in \mathcal{P}$ on $\tilde{\mathcal{P}}$. From Proposition 2.2, we have

$$\|\hat{\mathbf{p}} - \mathbf{p}\| = O(\varepsilon). \tag{12}$$

If we take $\mathbf{p} = d\mathbf{x}$. We get

$$\|d\hat{\mathbf{x}} - d\mathbf{x}\| \quad = \quad o(\varepsilon) = O(\varepsilon^2) \tag{13}$$

as illustrated by Figure 2. □

3 Wrappers

The approximation of sets using boxes computed using interval analysis generates a strong wrapping effect. It has been shown by several authors that it was possible to get a linear approximation with a better accuracy using other types of sets such as zonotopes [9, 10], constrained zonotopes [39, 35], ellipsoids [33], or doubleton [19]. Before defining the notion of wrapper to quantify the order of approximation we can get, we first recall what is a contractor.

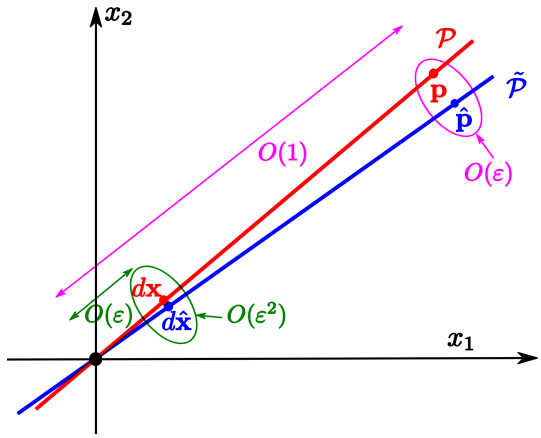


Figure 2: If we move the plane $\mathcal{P}$ of an order ε, a point $\mathbf{p}$ of the plane $\mathcal{P}$ will be at a distance to the new plane $\tilde{\mathcal{P}}$ of an order ε. If we do the same operation with a vector $d\mathbf{x}$ with a norm of order ε, then the distance of $d\mathbf{x}$ to $\tilde{\mathcal{P}}$ is an order ε^2.

Definition. *Denote by* $\mathbb{IR}^n$ *the set of boxes of* $\mathbb{R}^n$. *A* contractor *associated to the closed set* $\mathbb{X} \subset \mathbb{R}^n$ *is a function* $\mathcal{C} : \mathbb{IR}^n \mapsto \mathbb{IR}^n$ *such that*

$$\begin{array}{ll} \mathcal{C}([\mathbf{x}]) \subset [\mathbf{x}] & \textit{(contraction)} \\ [\mathbf{x}] \cap \mathbb{X} \subset \mathcal{C}([\mathbf{x}]) & \textit{(consistency)} \end{array}$$

The contractor $\mathcal{C}$ *for* $\mathbb{X}$ *is* minimal *if* $\mathcal{C}([\mathbf{x}]) = [\![[\mathbf{x}] \cap \mathbb{X}]\!]$ *where* $[\![\mathbb{A}]\!]$ *denotes the smallest box enclosing the set* $\mathbb{A}$.

The following definition of a *wrapper* extends the concept of contractor and will be needed for convergence analysis.

Definition. *A* wrapper *associated to the closed set* $\mathbb{X} \subset \mathbb{R}^n$ *is a function* $\mathcal{W} : \mathbb{IR}^n \mapsto \mathcal{P}(\mathbb{R}^n)$ *such that*

$$\begin{array}{ll} \mathcal{W}([\mathbf{x}]) \subset [\mathbf{x}] & \textit{(contraction)} \\ [\mathbf{x}] \cap \mathbb{X} \subset \mathcal{W}([\mathbf{x}]) & \textit{(consistency)} \\ \mathbf{x} \notin \mathbb{X} \Rightarrow \exists \varepsilon, \forall [\mathbf{x}] \subset B(\mathbf{x}, \varepsilon), \mathcal{W}([\mathbf{x}]) = \emptyset & \textit{(accuracy)} \end{array}$$

where $\mathcal{P}(\mathbb{R}^n)$ *is the set of all subsets of* $\mathbb{R}^n$ *and* $B(\mathbf{x}, \varepsilon)$ *is the box with center* $\mathbf{x}$ *and radius* ε.

An illustration of a wrapper is given by Figure 3. The set $\mathbb{X}$ is a curve which could be given by an equation. For the box $[\mathbf{a}]$, the set $\mathcal{W}([\mathbf{a}])$ encloses the part of $\mathbb{X}$ which is inside $[\mathbf{a}]$. The accuracy property is illustrated by the box $[\mathbf{b}]$, which satisfies $\mathcal{W}([\mathbf{b}]) = \emptyset$. The box $[\mathbf{b}]$ is inside the box $B(\mathbf{b}, \varepsilon)$ with $\mathbf{b} \notin \mathbb{X}$. This translates the fact if a box $[\mathbf{b}]$ is outside $\mathbb{X}$ and sufficiently small then the wrapper will be able conclude that it is indeed outside $\mathbb{X}$.

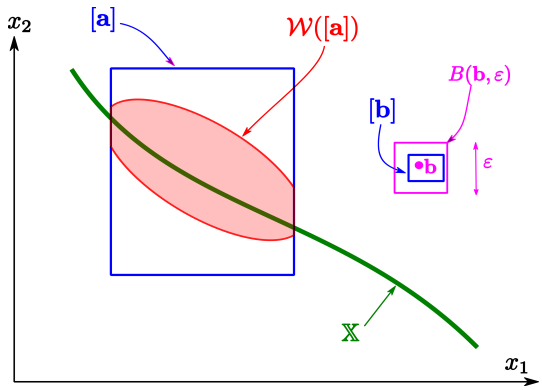


Figure 3: Illustration of a wrapper. $\mathcal{W}([\mathbf{b}])$ is empty which means that $[\mathbf{b}] \cap \mathbb{X} = \emptyset$.

The wrapper $\mathcal{W}$ for $\mathbb{X}$ has an order i at point $\mathbf{x}$ if for all nested sequences of boxes $[\mathbf{x}](k)$ converging to $\mathbf{x}$, we have

$$\lim_{k\to\infty} \frac{h(\mathcal{W}([\mathbf{x}](k)), \mathbb{X})}{(w([\mathbf{x}](k)))^i} = 0 \tag{14}$$

where $w([\mathbf{x}])$ is the width of $[\mathbf{x}]$. In this paper, only the order one will be considered. Denote by $\text{Wrap}(\mathbb{X}, \mathbf{x})$ the set of all wrappers for $\mathbb{X}$ which have an order 1 at point $\mathbf{x}$.

The notion of order is illustrated by Figure 4. The larger is k, the narrower is $[\mathbf{x}](k)$ and more accurate is the approximation.

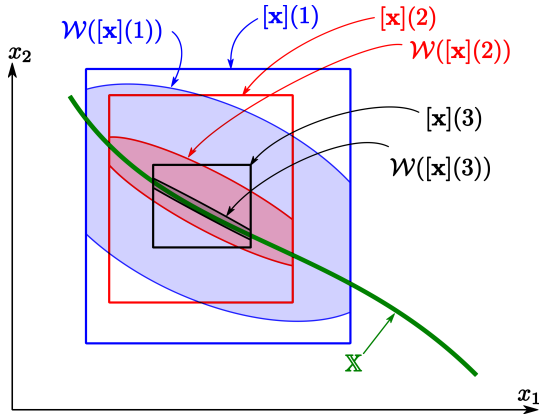


Figure 4: Wrapper of order 1. This wrapper generates a set which fits the shape of the set $\mathbb{X} \cap [\mathbf{x}]$.

Definition. *We define the intersection $\mathcal{W}$ of two wrappers $\mathcal{W}_1$ and $\mathcal{W}_2$ as*

$$\mathcal{W}([\mathbf{x}]) = (\mathcal{W}_1 \cap \mathcal{W}_2)([\mathbf{x}]) = \mathcal{W}_1([\mathbf{x}]) \cap \mathcal{W}_2([\mathbf{x}]). \tag{15}$$

It is trivial to check that if $\mathcal{W}_1$ is a wrapper for $\mathbb{X}_1$ and $\mathcal{W}_2$ is a wrapper for $\mathbb{X}_2$ then $\mathcal{W} = \mathcal{W}_1 \cap \mathcal{W}_2$ is a wrapper for $\mathbb{X}_1 \cap \mathbb{X}_2$. Unfortunately, the order of the approximation is not always preserved. The following proposition gives some conditions which allows us to preserve the order 1.

Proposition. *Given m sets $\mathbb{X}_i = \{\mathbf{x} \in \mathbb{R}^n | f_i(\mathbf{x}) = 0\}$, where $f_i : \mathbb{R}^n \mapsto \mathbb{R}$. Consider $\mathbb{Z} = \bigcap_i \mathbb{X}_i$ and a point $\mathbf{z} \in \mathbb{Z}$.* Assume that all $\frac{df_i}{d\mathbf{x}}(\mathbf{z})$ are independent. *If $\mathcal{W} = \bigcap_i \mathcal{W}_i$, we have*

$$\forall i, \mathcal{W}_i \in \mathit{Wrap}(\mathbb{X}_i, \mathbf{z}) \Rightarrow \textstyle\bigcap_i \mathcal{W}_i \in \mathit{Wrap}(\mathbb{Z}, \mathbf{z}) \tag{16}$$

Figure 5 illustrates that the intersection of two wrappers of order 1 at $\mathbf{z}$ is generally a wrapper of order 1 at $\mathbf{z}$. In the figure, the set $\mathbb{Z} = \mathbb{X}_1 \cap \mathbb{X}_2$ is the singleton $\{\mathbf{z}\}$. The box $[\mathbf{x}]$ should be interpreted as a narrow box containing $\mathbf{z}$.

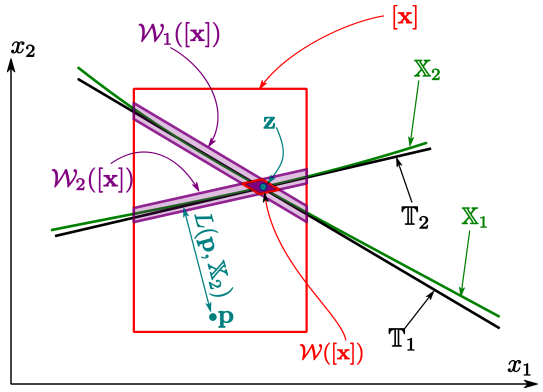


Figure 5: The intersection of two wrappers $\mathcal{W}_1$ and $\mathcal{W}_2$ of order 1 (here red) is a wrapper $\mathcal{W}$ of order 1 for the intersection of the two corresponding sets $\mathbb{X}_1$ and $\mathbb{X}_2$.

Proof. Since $\mathbb{Z} = \bigcap_i \mathbb{X}_i$, $\mathcal{W} = \bigcap_i \mathcal{W}_i$ is a wrapper for $\mathbb{Z}$. We also need to prove that the order of $\mathcal{W}$ is 1 at $\mathbf{z}$. For this, consider a sequence $[\mathbf{x}](k)$ converging to $\mathbf{z}$. When k is large $\varepsilon = w([\mathbf{x}](k))$ is small. For short, let us omit the dependency with respect to k. For all $\mathbf{p} \in [\mathbf{x}]$, we have $\|\mathbf{p} - \mathbf{z}\| = O(\varepsilon)$. If $\mathbb{T}_i$ is the tangent space of $\mathbb{X}_i$ at point $\mathbf{z}$ then

$$L(\mathbf{p}, \mathbb{X}_i) = L(\mathbf{p}, \mathbb{T}_i) + o(\varepsilon). \tag{17}$$

If all $\mathbb{T}_i$ are transverse, we have

$$L(\mathbf{p}, \mathbb{Z}) = L(\mathbf{p}, \textstyle\bigcap_i \mathbb{X}_i) = L(\mathbf{p}, \bigcap_i \mathbb{T}_i) + o(\varepsilon). \tag{18}$$

Take now, $\mathbf{p} \in \mathcal{W}([\mathbf{x}])$. Since $\forall i, L(\mathbf{p}, \mathbb{T}_i) = o(\varepsilon)$ and since the $\mathbb{T}_i$ are transverse, we get that $L(\mathbf{p}, \bigcap_i \mathbb{T}_i) = o(\varepsilon)$. Therefore, from (18), $L(\mathbf{p}, \mathbb{Z}) = o(\varepsilon)$. Since this is true for all $\mathbf{p} \in \mathcal{W}([\mathbf{x}])$, we have

$$h(\mathcal{W}([\mathbf{x}]), \mathbb{Z}) = \sup_{\mathbf{p} \in \mathcal{W}([\mathbf{x}])} L(\mathbf{p}, \mathbb{Z}) = o(\varepsilon) = o(w([\mathbf{x}])). \tag{19}$$

Taking into account the dependency of $[\mathbf{x}]$ in k, we get:

$$\lim_{k \to \infty} \frac{h(\mathcal{W}([\mathbf{x}](k)), \mathbb{Z})}{w([\mathbf{x}](k))} = 0, \tag{20}$$

which proves that $\mathcal{W}$ has an order 1 at point $\mathbf{z}$. □

4 Asymptotically minimal contractor

Consider the special case where wrappers, as defined by Definition 3, generate sets $\mathcal{W}([\mathbf{x}])$ that are boxes of $\mathbb{R}^n$. The order cannot be equal to 1 (it can only be equal to 0), except if $n = 1$. Now, we can use the wrappers of order 1 (which return a set which is not a box, a zonotope, for instance), as an intermediate result, to get contractors with a good accuracy. For this, we will have to compute the smallest possible box which encloses this non-box intermediate approximation.

This section formally defines such accurate contractors which are called *asymptotically minimal*.

Definition. *A contractor for* $\mathbb{X}$ *is* asymptotically minimal *at point* $\mathbf{z} \in \mathbb{X} \subset \mathbb{R}^n$ *if for any nested sequence* $[\mathbf{x}](k)$ *converging to* $\mathbf{z}$*, we have*

$$\lim_{k \to \infty} \frac{h(\mathcal{C}([\mathbf{x}](k)), [\![[\mathbf{x}](k) \cap \mathbb{X}]\!])}{w([\mathbf{x}](k))} = 0. \tag{21}$$

Note that since $\mathcal{C}$ is a contractor the quantity $\mathcal{C}([\mathbf{x}](k))$ is a box.

Proposition. *If* $\mathcal{W} \in \mathit{Wrap}(\mathbb{X}, \mathbf{z})$*, then, the contractor defined by*

$$\mathcal{C}([\mathbf{x}]) = [\![\mathcal{W}([\mathbf{x}])]\!] \tag{22}$$

is an asymptotically minimal contractor for $\mathbb{X}$ *at* $\mathbf{z}$*.*

An illustration of the proposition is given by Figure 6. The gray part corresponds to the pessimism of the contractor which tends to disappear when $[\mathbf{x}]$ becomes narrow.

Proof. The proof is by contradiction. Assume that $\mathcal{C}([\mathbf{x}]) = [\![\mathcal{W}([\mathbf{x}])]\!]$ is not asymptotically minimal in $\mathbf{z}$. From (21), there exists a sequence of nested boxes such converging to $\mathbf{z}$ such that

$$\lim_{k \to \infty} \frac{h([\![\mathcal{W}([\mathbf{x}])(k)]\!], [\![[\mathbf{x}](k) \cap \mathbb{X}]\!])}{w([\mathbf{x}](k))} > 0. \tag{23}$$

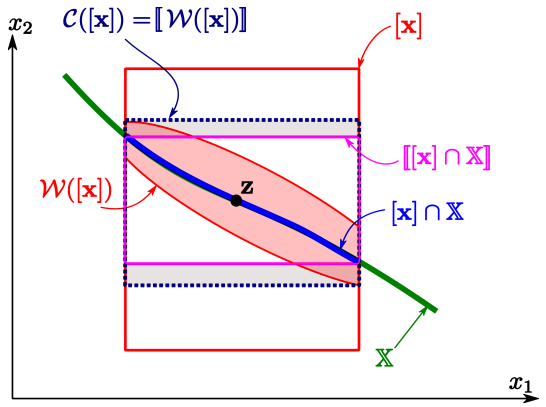


Figure 6: Asymptotic minimal contractor $\mathcal{C}([\mathbf{x}])$. It first computes the set $\mathcal{W}([\mathbf{x}])$ and then encloses in the box $[\![\mathcal{W}([\mathbf{x}])]\!]$.

Since for all $\mathbb{A} \subset \mathbb{R}^n$, and for all box $[\mathbf{b}]$, we have $h([\![\mathbb{A}]\!], [\mathbf{b}]) = h(\mathbb{A}, [\mathbf{b}])$, we have

$$\lim_{k\to\infty} \frac{h(\mathcal{W}([\mathbf{x}])(k), [\![[\mathbf{x}](k) \cap \mathbb{X}]\!])}{w([\mathbf{x}](k))} > 0. \tag{24}$$

Moreover, since h is monotonic decreasing with respect to its second argument, we get

$$\lim_{k\to\infty} \frac{h(\mathcal{W}([\mathbf{x}])(k), [\mathbf{x}](k) \cap \mathbb{X})}{w([\mathbf{x}](k))} > 0.$$

Since the sequence $[\mathbf{x}](k)$ converges to $\mathbf{z}$, if k is sufficiently large, we have $h(\mathcal{W}([\mathbf{x}])(k), [\mathbf{x}](k) \cap \mathbb{X}) = h(\mathcal{W}([\mathbf{x}])(k), \mathbb{X})$. As a consequence,

$$\lim_{k\to\infty} \frac{h(\mathcal{W}([\mathbf{x}](k)), \mathbb{X})}{w([\mathbf{x}](k))} > 0. \tag{25}$$

This is inconsistent with the fact that $\mathcal{W}$ has an order 1 in $\mathbf{z}$ (see (14)). □

5 Centered contractor

In this section, we show how to build an asymptotic minimal contractor using the centered form. We will consider functions $\mathbf{f} : \mathbb{R}^n \mapsto \mathbb{R}^p$ which are all continuous and differentiable. More precisely, the function $\mathbf{f}$ is described by continuous operator of functions such as $+, -, /, \sin, \exp, \ldots$ As a consequence using interval analysis, we are able to enclose the range of $\mathbf{f}$ and of $\frac{d\mathbf{f}}{d\mathbf{x}}$ over a box $[\mathbf{x}]$. In [28], Moore has proved that if $w([\mathbf{x}]) = O(\varepsilon)$ then using interval computation, we get an enclosure $[\mathbf{f}]([\mathbf{x}])$ for $\mathbf{f}([\mathbf{x}])$ and an enclosure $[\frac{d\mathbf{f}}{d\mathbf{x}}]([\mathbf{x}])$ for $\frac{d\mathbf{f}}{d\mathbf{x}}([\mathbf{x}])$ such that $w([\mathbf{f}]([\mathbf{x}])) = O(\varepsilon)$ and $w(\frac{d\mathbf{f}}{d\mathbf{x}}([\mathbf{x}])) = O(\varepsilon)$.

5.1 Scalar case

Proposition. *Consider the equation* $f(\mathbf{x}) = \mathbf{0}$*, where* $f : \mathbb{R}^n \mapsto \mathbb{R}$ *is differentiable. The solution set is*

$$\mathbb{X} \quad = \quad \{\mathbf{x} \in \mathbb{R}^n \,|\, f(\mathbf{x}) = \mathbf{0}\}. \tag{26}$$

Consider a point $\mathbf{z}$ *such that* $f(\mathbf{z}) = 0$*. Consider a nested sequence* $[\mathbf{x}](k)$ *converging to* $\mathbf{z}$*.* The function $\mathcal{L} : \mathbb{IR}^n \mapsto \mathcal{P}(\mathbb{R}^n)$ defined as

$$\begin{array}{rl} \mathcal{L}([\mathbf{x}]) & = \{ \quad \mathbf{x} \in [\mathbf{x}] \,|\, \exists \mathbf{a} \in [\frac{df}{d\mathbf{x}}]([\mathbf{x}]), \\ & \qquad\qquad f(\mathbf{m}) + \mathbf{a} \cdot (\mathbf{x} - \mathbf{m}) = 0\} \end{array} \tag{27}$$

where $\mathbf{m} = \text{center}([\mathbf{x}])$, is a wrapper of order 1*, i.e., it belongs to* $Wrap(\mathbb{X}, \mathbf{z})$*. It will be called the centered wrapper associated with* f*.*

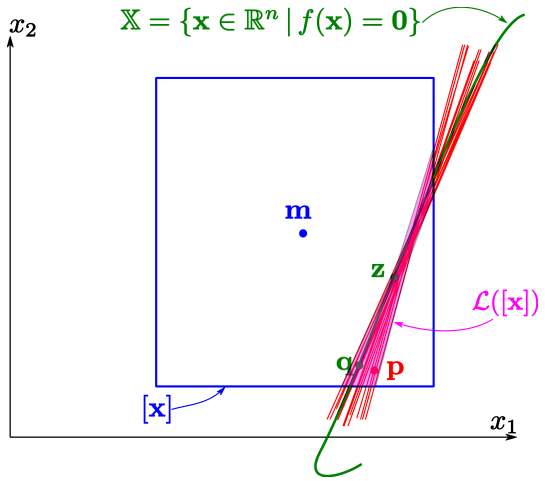


Figure 7: The set $\mathcal{L}([\mathbf{x}])$ (magenta) with a bowtie shape is close to the set $\mathbb{X}$ (here the curve in green). Moreover, $\mathcal{L}([\mathbf{x}])$ encloses $[\mathbf{x}] \cap \mathbb{X}$. The approximation is asymptotically perfect.

Proof. Consider the sequence $[\mathbf{x}](k) \subset \mathbb{R}^n$ converging to $\mathbf{z}$. We assume that $[\mathbf{x}](k)$, or $[\mathbf{x}]$ for short, is narrow, *i.e.*, $w([\mathbf{x}]) = O(\varepsilon)$. If $\mathbf{p} \in \mathcal{L}([\mathbf{x}])$ (see Figure 7) then, for some $\mathbf{a} \in [\mathbf{a}] = [\frac{df}{d\mathbf{x}}]([\mathbf{x}])$, we have

$$f(\mathbf{m}) + \mathbf{a} \cdot (\mathbf{p} - \mathbf{m}) = 0 \tag{28}$$

where $\mathbf{m} = \text{center}([\mathbf{x}])$. From Corollary 2.2, taking $d\mathbf{x} = \mathbf{p} - \mathbf{m} = O(\varepsilon)$ and since $w([\mathbf{a}]) = O(\varepsilon)$, we get that the distance between a point in $\mathcal{L}([\mathbf{x}])$ and the set $\mathbb{X}$ is an $o(\varepsilon)$. We get that

$$h(\mathcal{L}([\mathbf{x}](k)), \mathbb{X}) = o(w([\mathbf{x}](k))) \tag{29}$$

i.e.,

$$\lim_{k\to\infty} \frac{h(\mathcal{L}([\mathbf{x}](k)), \mathbb{X})}{w([\mathbf{x}](k))} = 0. \tag{30}$$

Thus the wrapper $\mathcal{L}$ is of order 1 at $\mathbf{z}$. □

Corollary. *The contractor for* $f(\mathbf{x}) = 0$ *defined by*

$$\begin{array}{rcl} [x_i] & = & [x_i] \cap \left(m_i - \frac{1}{[a_i]} \left(f(\mathbf{m}) + \sum_{j\neq i} [a_j] \cdot ([x_j] - m_j) \right) \right) \\ [a_j] & = & [\frac{\partial f}{\partial x_j}]([\mathbf{x}]) \end{array} \tag{31}$$

is asymptotically minimal.

Remark. Before starting the proof, it is important to recall an important notion on interval propagation. Consider an equation of the form

$$1 + a_1(x_1 - 2) + a_2(x_2 - 3) = 0,$$

with $a_1 \in [a_1], a_2 \in [a_2], x_1 \in [x_1], x_2 \in [x_2]$. The smallest box $[\mathbf{y}] = [y_1] \times [y_2]$ which encloses the set

$$\{(x_1, x_2) \in [\mathbf{x}] \,|\, \exists a_1 \in [a_1], \exists a_2 \in [a_2],\ 1 + a_1(x_1 - 2) + a_2(x_2 - 3) = 0\}$$

where $[\mathbf{x}] = [x_1] \times [x_2]$, is defined by

$$\begin{array}{rcl} [y_1] & = [x_1] \cap & \left(2 - \frac{1}{[a_1]} \left(1 + [a_2]([x_2] - 3) \right) \right) \\ [y_2] & = [x_2] \cap & \left(3 - \frac{1}{[a_2]} \left(1 + [a_1]([x_1] - 2) \right) \right) \end{array}$$

This corresponds to a forward-backward contraction in our special case. As shown in [27], $[\mathbf{y}]$ is indeed the smallest because both x_1 and x_2 occur only once in the equation $1 + a_1(x_1 - 2) + a_2(x_2 - 3) = 0$. It is related to what Moore calls the dependency problem [28]. When we have more than one equation, such as for instance,

$$\begin{array}{rcl} 1 + a_{11}(x_1 - 2) + a_{12}(x_2 - 3) & = & 0 \\ 1 + a_{21}(x_1 - 2) + a_{22}(x_2 - 3) & = & 0 \end{array}$$

the forward-backward contraction will not yield the minimal contraction. This is due to the fact that in the system of two equations, x_1 and x_2 occur twice and not once.

Proof. Define $\mathcal{L}([\mathbf{x}])$ as in (27). From Proposition 4, $\mathcal{L} \in \text{Wrap}(\mathbb{X}, \mathbf{z})$. The contractor $\mathcal{C}([\mathbf{x}]) = [\![\mathcal{L}([\mathbf{x}])]\!]$ is an asymptotically minimal contractor. Now the set $\mathcal{L}([\mathbf{x}])$ can be defined as the set of all $\mathbf{x}$ which satisfy the following constraint

$$\left\{ \begin{array}{c} f(\mathbf{m}) + \mathbf{a} \cdot (\mathbf{x} - \mathbf{m}) = 0 \\ \text{with } \mathbf{a} \in [\frac{df}{d\mathbf{x}}]([\mathbf{x}]) \\ \text{and } \mathbf{m} = \text{center}([\mathbf{x}]) \end{array} \right. \tag{32}$$

Since $\mathbf{x}$ occurs only once in the constraint $f(\mathbf{m}) + \mathbf{a} \cdot (\mathbf{x} - \mathbf{m}) = 0$, an interval forward-backward propagation provides us the minimal contraction [27], *i.e.*, it returns the box $[\![\mathcal{L}([\mathbf{x}])]\!]$. □

5.2 Vector case

Proposition. *Consider the equation* $\mathbf{f}(\mathbf{x}) = \mathbf{0}$*, where* $\mathbf{f} : \mathbb{R}^n \mapsto \mathbb{R}^p$ *is differentiable. The solution set is*

$$\mathbb{X} \quad = \quad \{\mathbf{x} \in \mathbb{R}^n \,|\, \mathbf{f}(\mathbf{x}) = \mathbf{0}\}. \tag{33}$$

Consider a point $\mathbf{z}$ *such that* $\mathbf{f}(\mathbf{z}) = \mathbf{0}$ *and a nested sequence* $[\mathbf{x}](k)$ *converging to* $\mathbf{z}$*.* Assume that all $\frac{df_i}{d\mathbf{x}}(\mathbf{z})$ are independent. *Consider the wrappers* $\mathcal{L}_i : \mathbb{IR}^n \mapsto \mathcal{P}(\mathbb{R}^n)$ *of order 1 for* $f_i(\mathbf{x}) = 0$ *defined by*

$$\mathcal{L}_i([\mathbf{x}]) \quad = \{ \quad \mathbf{x} \in [\mathbf{x}] \,|\, \exists \mathbf{a} \in [\tfrac{df_i}{d\mathbf{x}}]([\mathbf{x}]), f_i(\mathbf{m}) + \mathbf{a} \cdot (\mathbf{x} - \mathbf{m}) = 0\} \tag{34}$$

where $\mathbf{m} = center([\mathbf{x}])$*. The operator* $\bigcap_i \mathcal{L}_i$*, belongs to* $Wrap(\mathbb{X}, \mathbf{z})$*.*

Proof. We have

$$\mathbb{X} = \underbrace{\{\mathbf{x} \in \mathbb{R}^n \,|\, f_1(\mathbf{x}) = 0\}}_{\mathbb{X}_1} \cap \cdots \cap \underbrace{\{\mathbf{x} \in \mathbb{R}^n \,|\, f_p(\mathbf{x}) = 0\}}_{\mathbb{X}_p}.$$

Now, from Proposition 5.1, the $\mathcal{L}_i([\mathbf{x}])$, as defined by 34, belong to $\text{Wrap}(\mathbb{X}_i, \mathbf{z})$. From Proposition 3, we get that $\bigcap_i \mathcal{L}_i$ belongs to $\text{Wrap}(\mathbb{X}, \mathbf{z})$. □

To compute $\bigcap_i \mathcal{L}_i$, the method proposed for the scalar case is not valid anymore. An interval linear method could be used [31, 1] that are based on an interval version of the simplex algorithms. Now, these methods are not proved to be minimal or asymptotically minimal, which may ruin our objective to get an asymptotically minimal contractor. An other possibility is to use a preconditioning method based on the Gauss-Jordan decomposition, which will be minimal in many cases, such as the test-case that will be treated in Section 6.

5.3 Preconditioning

Consider the equation $\mathbf{f}(\mathbf{x}) = \mathbf{0}$, where $\mathbf{f} : \mathbb{R}^n \mapsto \mathbb{R}^p$ is differentiable. Intersecting sets $\mathcal{L}_i([\mathbf{x}])$ as suggested by Proposition 5.2 requires the resolution of interval linear equations. This operation is costly and should be avoided if it has to be repeated a large number of times. Instead of this, we prefer to use a specific preconditioning method.

To understand the principle of the preconditioning, consider the following interval linear system

$$\begin{pmatrix} d_{11} & d_{12} & 0 \\ 0 & d_{22} & d_{23} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \end{pmatrix} \tag{35}$$

where

$$d_{ij} \in [d_{ij}], x_j \in [x_j], b_i \in [b_i] \tag{36}$$

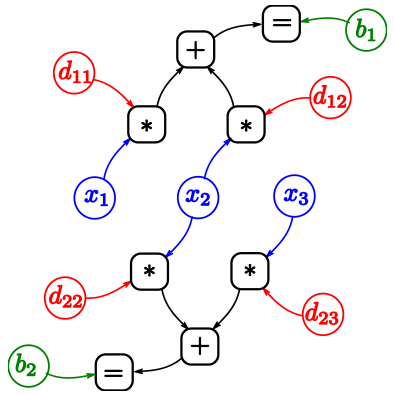


Figure 8: The constraint network has no cycle (it is a tree). Thus the interval propagation is minimal.

The optimal contraction can be obtained by a simple interval propagation. This is due to the fact that the corresponding constraint network has no cycle [27], as illustrated by Figure 8.

Note that no cycle would have been obtained with the following linear system:

$$\begin{pmatrix} d_{11} & d_{12} & 0 & 0 \\ 0 & d_{22} & d_{23} & 0 \\ 0 & 0 & d_{33} & d_{34} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix} \tag{37}$$

A matrix $\mathbf{D}$ such that the system $\mathbf{D} \cdot \mathbf{x} = \mathbf{b}$ has no cycle can be called a *tree matrix*.

Both systems (35) and (37), for which the matrix $\mathbf{D}$ is a *band* matrix [2], could be obtained from a Gauss Jordan transformation of a linear systems [22]. For instance, if we have a system of the form $\mathbf{Ax} = \mathbf{c}$ where $\mathbf{A}$ is of dimension 3×4 with full rank, there exists a matrix $\mathbf{Q}$ of dimension 3×3 such that

$$\mathbf{Ax} = \mathbf{c} \Leftrightarrow \mathbf{Q} \cdot \mathbf{A} \cdot \mathbf{x} = \mathbf{Q} \cdot \mathbf{c} \tag{38}$$

where $\mathbf{D} = \mathbf{Q} \cdot \mathbf{A}$ has the form given by (37).

Proposition. *Consider a set* $\mathbb{X} = \{\mathbf{x} \in \mathbb{R}^n | \mathbf{f}(\mathbf{x}) = \mathbf{0}\}$. *Take a narrow box* $[\mathbf{x}]$ *with center* $\mathbf{m}$. *Assume that* $\frac{d\mathbf{f}}{d\mathbf{x}}(\mathbf{m})$ *is a tree matrix. An interval propagation on the system*

$$\begin{gathered} \mathbf{f}(\mathbf{m}) + \mathbf{A} \cdot (\mathbf{x} - \mathbf{m}) = \mathbf{0} \\ \text{with } \mathbf{A} \in [\tfrac{d\mathbf{f}}{d\mathbf{x}}]([\mathbf{x}]) \\ \text{and } \mathbf{x} \in [\mathbf{x}] \end{gathered} \tag{39}$$

corresponds to an asymptotically minimal contractor for $\mathbb{X}$.

Proof. The interval matrix $[\mathbf{A}] = [\frac{d\mathbf{f}}{d\mathbf{x}}]([\mathbf{x}])$ is such that $w([\mathbf{A}]) = O(\varepsilon)$, where $\varepsilon = w([\mathbf{x}])$. Now, Proposition 2.2 tells us that if we move $\mathbf{A}$ a little (at order 0), then, the solution set for the linear equation moves a little also, at order 1. Due to the fact that the contractor $\mathcal{C}$ resulting from the interval propagation is minimal for $\mathbf{A} = \frac{d\mathbf{f}}{d\mathbf{x}}(\mathbf{m})$, we get that the contractor obtained by an elementary interval propagation is asymptotically minimal. □

Corollary. *Consider a set* $\mathbb{X} = \{\mathbf{x} \in \mathbb{R}^n | \mathbf{f}(\mathbf{x}) = \mathbf{0}\}$. *Take a narrow box* $[\mathbf{x}]$ *with center* $\mathbf{m}$. *Define* $\mathbf{Q}$ *such that* $\mathbf{Q} \cdot \frac{d\mathbf{f}}{d\mathbf{x}}(\mathbf{m})$ *is a tree matrix. An interval propagation on the system*

$$\begin{array}{c} \mathbf{Q} \cdot \mathbf{f}(\mathbf{m}) + \mathbf{Q} \cdot \mathbf{A} \cdot (\mathbf{x} - \mathbf{m}) = \mathbf{0} \\ \textit{with } \mathbf{A} \in [\frac{d\mathbf{f}}{d\mathbf{x}}]([\mathbf{x}]) \\ \textit{and } \mathbf{x} \in [\mathbf{x}] \end{array} \tag{40}$$

corresponds to an asymptotically minimal contractor for $\mathbb{X}$.

Proof. It suffices to apply Proposition 5.3 where $\mathbf{f}(\mathbf{x})$ should be replaced by $\mathbf{Q} \cdot \mathbf{f}(\mathbf{x})$. □

5.4 Algorithm

Consider the system $\mathbf{f}(\mathbf{x}) = \mathbf{0}$ and take a box $[\mathbf{x}]$. We assume that we have an analytical expression for $\mathbf{f}$, so that we have an inclusion function for $\mathbf{f}$ and its Jacobian matrix $\frac{d\mathbf{f}}{d\mathbf{x}}$. The following algorithm corresponds to a centered contractor.

Input:	$\mathbf{f}$,$[\mathbf{x}]$
1	$\mathbf{m} = \text{center}([\mathbf{x}])$
2	Compute the Gauss-Jordan matrix $\mathbf{Q}$ for $\frac{d\mathbf{f}}{d\mathbf{x}}(\mathbf{m})$
3	Define $\mathbf{g}(\mathbf{x}) = \mathbf{Q} \cdot \mathbf{f}(\mathbf{x})$
4	For $i \in \{1, \ldots, p\}$
5	For $j \in \{1, \ldots, n\}$
6	$[\mathbf{a}] = [\frac{\partial g_i}{\partial \mathbf{x}}]([\mathbf{x}])$
7	$[s] = \sum_{k \neq j} [a_k] \cdot ([x_k] - m_k)$
8	$[x_j] = [x_j] \cap \frac{1}{[a_j]} (-g_i(\mathbf{m}) - [s])$
9	Return $[\mathbf{x}]$

- Step 1 takes the center $\mathbf{m}$ of $[\mathbf{x}]$ in order to form a linear approximation for $\mathbf{f}$ in $[\mathbf{x}]$:

$$\mathbf{f}(\mathbf{x}) = \mathbf{f}(\mathbf{m}) + \frac{d\mathbf{f}}{d\mathbf{x}}(\mathbf{m}) \cdot (\mathbf{x} - \mathbf{m}). \tag{41}$$

- Step 2 returns an invertible $m \times m$ matrix $\mathbf{Q}$ such that $\mathbf{A} = \mathbf{Q} \cdot \frac{d\mathbf{f}}{d\mathbf{x}}(\mathbf{m})$ is a band matrix. The matrix $\mathbf{Q}$ is chosen by a Gauss-Jordan algorithm. The new system to be solved is now

$$\mathbf{Q} \cdot \mathbf{f}(\mathbf{x}) = \mathbf{0}. \tag{42}$$

- Step 3 defines the function $\mathbf{g}(\mathbf{x}) = \mathbf{Q} \cdot \mathbf{f}(\mathbf{x})$. We need to solve $\mathbf{g}(\mathbf{x}) = \mathbf{0}$ in the box $[\mathbf{x}] - \mathbf{m}$. The main difference compared to the previous system $\mathbf{f}(\mathbf{x}) = \mathbf{0}$ is that its linear approximation

$$\mathbf{g}(\mathbf{x}) = \mathbf{g}(\mathbf{m}) + \mathbf{A} \cdot (\mathbf{x} - \mathbf{m}) \tag{43}$$

  is such that $\mathbf{A}$ is a band matrix.

- Step 4-9 define the set of constraints

$$\begin{cases} \mathbf{0} = \mathbf{g}(\mathbf{m}) + \mathbf{A} \cdot (\mathbf{x} - \mathbf{m}) \\ \quad \text{with } \mathbf{A} \in [\frac{d\mathbf{g}}{d\mathbf{x}}]([\mathbf{x}]) \\ \quad \text{and } \mathbf{x} \in [\mathbf{x}] \end{cases} \tag{44}$$

  and performs an interval propagation. Due to the fact that the system has no cycle (at first order), from Corollary 5.3, we get that the propagation is asymptotically minimal.

6 Test case

Interval methods have been shown to be very powerful for the stability analysis of linear systems [23]. We have chosen to consider the linear time-delay system [40] given by

$$\ddot{x} + 2\dot{x}(t - p_1) + x(t - p_2) = 0 \tag{45}$$

but other types of linear systems [25] with fractional orders could be considered as well. Its characteristic function is

$$\theta(\mathbf{p}, s) = s^2 + 2se^{-sp_1} + e^{-sp_2}. \tag{46}$$

For a given $\mathbf{p} = (p_1, p_2)$, the location of the roots for $\theta(\mathbf{p}, s)$ provides an information concerning the stability of the system. For instance, if all roots are on the half left of the complex plane, then the system is stable. The stability changes when one root crosses the imaginary line. This is the reason why we are interested in characterizing the set

$$\mathcal{P} = \{\mathbf{p} \,|\, \exists \omega > 0,\, \theta(\mathbf{p}, j\omega) = 0\}. \tag{47}$$

which corresponds to the set of parameters for which the roots are at the stability boundary. Since that for all $\mathbf{p}$ and for all ω, we have $\theta(\mathbf{p}, j\omega) = \theta(\mathbf{p}, -j\omega)$, we classically impose $\omega > 0$. Now

$$\begin{array}{rl} & \theta(p_1, p_2, j\omega) \\ = & -\omega^2 + 2j\omega e^{-j\omega p_1} + e^{-j\omega p_2} \\ = & -\omega^2 + 2j\omega(\cos(\omega p_1) - j\sin(\omega p_1)) \\ & + \cos(\omega p_2) - j\sin(\omega p_2) \\ = & -\omega^2 + 2\omega\sin(\omega p_1) + \cos(\omega p_2) \\ & + j \cdot (2\omega\cos(\omega p_1) - \sin(\omega p_2)) \end{array} \tag{48}$$

We have

$$\theta(p_1,p_2,j\omega)=0 \Leftrightarrow \underbrace{\begin{pmatrix} -\omega^2+2\omega\sin(\omega p_1)+\cos(\omega p_2) \\ 2\omega\cos(\omega p_1)-\sin(\omega p_2) \end{pmatrix}}_{\mathbf{f}(p_1,p_2,\omega)}=0 \tag{49}$$

Take $[p_1]=[0,2.5]$, $[p_2]=[1,4]$, $[\omega]=[0,10]$ and let us characterize the set $\mathcal{P}$ using the centered contractor. Using a branch and prune algorithm such as SIVIA (see e.g. [18]) with an accuracy of $\varepsilon=2^{-8}$ with an HC4 algorithm [7, 4] (the state of the art), we get the paving of Figure 9 in 4 sec. The number of boxes of the approximation is 43173. Similar results were obtained were obtained on the same example in [24].

With an accuracy of $\varepsilon=2^{-4}$ with the centered contractor given in Section 5.4, we get the paving of Figure 10 in 1.2 sec. The number of boxes of the approximation is 282 (instead of 43173), for a more accurate approximation.

With an accuracy of $\varepsilon=2^{-8}$ with the centered contractor, we get the thin curve represented on Figure 11. This curve is made with the small boxes generated by the paver, which shows the quality of the approximation. The big blue boxes are those already painted in the green box [**a**] of Figure 10.

With an accuracy of $\varepsilon=2^{-12}$ with the centered contractor, we get the magenta curve of Figure 12. The big gray boxes are those already painted in the red box [**b**] of Figure 11. The fact that, for a small ε, the boxes of the approximation only overlap on their corners illustrates the minimality of the contractor.

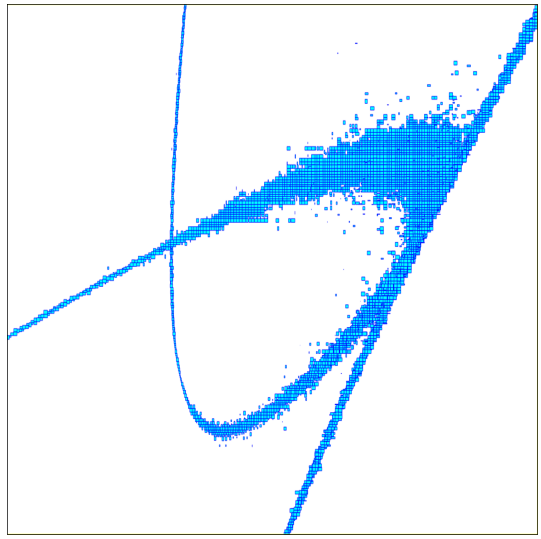


Figure 9: Approximation of the solution set $\mathcal{P}$ with a state of the art contractor (here HC4). The frame box for (p_1,p_2) is $[0,2.5]\times[2,4]$.

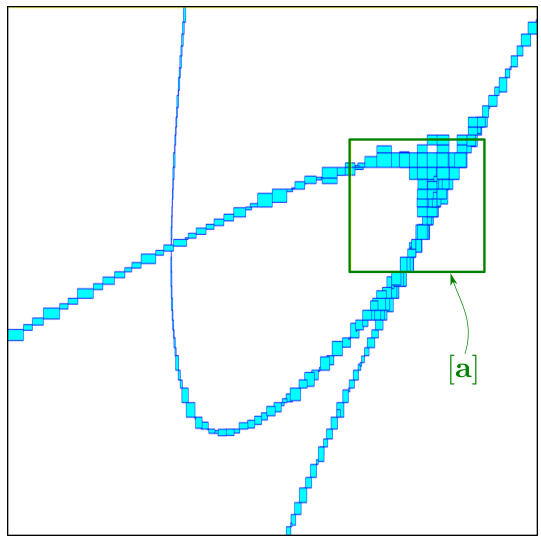


Figure 10: Paving obtained with the centered contractor. The frame box for (p_1, p_2) is $[0, 2.5] \times [2, 4]$.

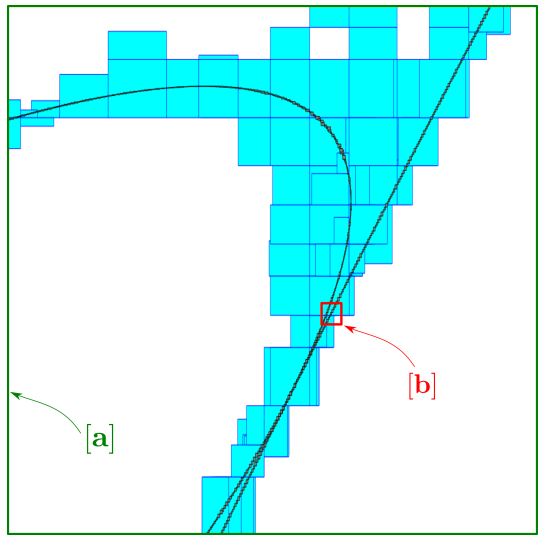


Figure 11: Pavings obtained with the centered contractor in the box $[\mathbf{a}] = [1.3, 1.8] \times [3.0, 3.5]$; Blue: $\varepsilon = 2^{-4}$; Thin: $\varepsilon = 2^{-8}$.

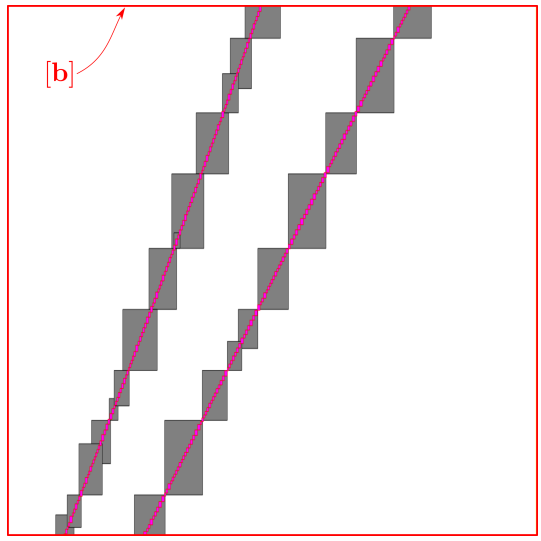


Figure 12: Approximation of the solution set in $[\mathbf{b}] = [1.595, 1.615] \times [3.2, 3.22]$; Gray: $\varepsilon = 2^{-8}$; Magenta: $\varepsilon = 2^{-12}$.

The computing time to get the three Figures 10, 11 and 12 is less than 10 sec. Our results are much more accurate than those obtained in Section 6 of [24].

The code, based on the codac library [36], and an illustrating video are given at `www.ensta-bretagne.fr/jaulin/centered.html`.

7 Conclusion

In this paper, we have proposed a contractor which is asymptotically minimal for the approximation of a curve defined by nonlinear equations. The resulting *centered* contractor is based on the centered form which suppresses the pessimism when the boxes are narrow and when we have a single equation. When we combine several equations, a preconditioning method has been proposed in order to linearize the problem into a system where a tree matrix in involved. The preconditioning has been implemented using a Gauss Jordan band diagonalization method. On an example, we have shown that our centered contractor was able to outperform the state of the art contractor based on a forward-backward propagation.

Other approaches, such as the generalized interval arithmetic [15], the affine arithmetic [12] allows to get first order approximation of the constraints. As for our paper, these arithmetics can obviously model the affine dependencies between quantities with an error that shrinks quadratically with the size of the input intervals. Now, this linear approximation is only valid when we have a single constraint and can thus not be used to build asymptotically minimal contractors without

some improvements. Our approach does not require the implementation of a new arithmetic since it only uses the standard interval arithmetic. Moreover, our approach generates a contractor that can be combined with other existing contractors enforcing the efficiency of the resolution.

References

[1] Araya, I., Trombettoni, G., and Neveu, B. A contractor based on convex interval taylor. In Beldiceanu, N., Jussien, N., and Pinson, E., editors, *Proceedings of the 9th International Conference on Integration of AI and OR Techniques in Contraint Programming for Combinatorial Optimzation Problems*, Volume 7298 of *Lecture Notes in Computer Science*, pages 1–16. Springer, 2012. DOI: 10.1007/978-3-642-29828-8_1.

[2] Atkinson, K. *An Introduction to Numerical Analysis (2nd edition)*. John Wiley, 1989. DOI: 10.1002/0471667196.ess1837.

[3] Auer, E., Rauh, A., Hofer, E., and Luther, W. Validated modeling of mechanical systems with SmartMOBILE: Improvement of performance by ValEncIA-IVP. In Hertling, P., Hoffmann, C. M., Luther, W., and Revol, N., editors, *Revised papers of the International Seminar on Reliable Implementation of Real Number Algorithms: Theory and Practice*, Volume 5045 of *Lecture Notes in Computer Science*, pages 1–27. Springer, 2008. DOI: 10.1007/978-3-540-85521-7_1.

[4] Benhamou, F., Goualard, F., Granvilliers, L., and Puget, J. F. Revising hull and box consistency. In Schreye, D. D., editor, *Proceedings of the 1999 International Conference on Logic Programming*, pages 230–244. MIT Press, 1999. DOI: 10.7551/mitpress/4304.003.0024.

[5] Berger, M. *Geometry I and II*. Springer-Verlag, Berlin, 1987. ISBN: 9783540116585 and ISBN: 9783540170150.

[6] Braems, I., Kieffer, M., Walter, É., and Jaulin, L. *Set Computation, Computation of Volumes and Data Safety*. In Krämer, W. and von Gudenberg, J. W., editors, *Scientific Computing, Validated Numerics, Interval Methods*, pages 267–278. Kluwer Academic Publishers, 2001. DOI: 10.1007/978-1-4757-6484-0_22.

[7] Cébério, M. and Granvilliers, L. Solving nonlinear systems by constraint inversion and interval arithmetic. In Campbell, J. A. and Roanes-Lozano, E., editors, *Proceedings of the International Conference on Artificial Intelligence and Symbolic Computation*, Volume 1930 of *Lecture Notes in Computer Science*, pages 127–141. Springer, 2000. DOI: 10.1007/3-540-44990-6_10.

[8] Chabert, G. and Jaulin, L. Contractor programming. *Artificial Intelligence*, 173:1079–1100, 2009. DOI: 10.1016/J.ARTINT.2009.03.002.

[9] Combastel, C. Zonotopes and Kalman observers: Gain optimality under distinct uncertainty paradigms and robust convergence. *Automatica*, 55:265–273, 2015. DOI: 10.1016/j.automatica.2015.03.008.

[10] Combastel, C. Functional sets with typed symbols: Mixed zonotopes and polynotopes for hybrid nonlinear reachability and filtering. *Automatica*, 143:110457, 2022. DOI: 10.1016/j.automatica.2022.110457.

[11] Ehambram, A., Voges, R., and Wagner, B. Stereo-visual-lidar sensor fusion using set-membership methods. In *Proceedings of the 17th IEEE International Conference on Automation Science and Engineering*, pages 1132–1139. IEEE, 2021. DOI: 10.1109/CASE49439.2021.9551516.

[12] Figueiredo, L. D. and Stolfi, J. Affine arithmetic: concepts and applications. *Numerical Algorithms*, 37(1):147–158, 2004. DOI: 10.1023/B:NUMA.0000049462.70970.B6.

[13] Goubault, E. and Putot, S. Robust under-approximations and application to reachability of non-linear control systems with disturbances. *IEEE Control System Letters*, 4(4):928–933, 2020. DOI: 10.1109/LCSYS.2020.2997261.

[14] Guyonneau, R., Lagrange, S., and Hardouin, L. A visibility information for multi-robot localization. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1426–1431, 2013. DOI: 10.1109/IROS.2013.6696536.

[15] Hansen, E. R. A generalized interval arithmetic. In Nickel, K., editor, *Proceedings of the International Symposium on Interval Mathemantics*, Volume 29, pages 7–18. Springer, 1975. DOI: 10.1007/3-540-07170-9_2.

[16] Hansen, E. R. *Global Optimization using Interval Analysis*. Marcel Dekker, New York, NY, 1992. ISBN: 9780824786960.

[17] Jaulin, L., Kieffer, M., Didrit, O., and Walter, E. *Applied Interval Analysis, with Examples in Parameter and State Estimation, Robust Control and Robotics*. Springer-Verlag, London, 2001. DOI: 10.1007/978-1-4471-0249-6_2.

[18] Jaulin, L. and Walter, E. Set inversion via interval analysis for nonlinear bounded-error estimation. *Automatica*, 29(4):1053–1064, 1993. DOI: 10.1016/0005-1098(93)90106-4.

[19] 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.

[20] Kreinovich, V., Lakeyev, A., Rohn, J., and Kahl, P. Computational complexity and feasibility of data processing and interval computations. *Reliable Computing*, 4(4):405–409, 1998. DOI: 10.1023/A:1024484203503.

[21] Langerwisch, M. and Wagner, B. Guaranteed mobile robot tracking using robust interval constraint propagation. In *Proceedings of the 5th International Conference on Intelligent Robotics and Applications, Part II*, Volume 7507 of *Lecture Notes in Computer Science*, pages 354–365. Springer, 2012. DOI: 10.1007/978-3-642-33515-0_36.

[22] Leon, S. *Linear Algebra with Applications (8th edition)*. Pearson, 2009. ISBN: 9780136009290.

[23] Malan, S. A., Milanese, M., and Taragna, M. Robust analysis and design of control systems using interval arithmetics. *Automatica*, 33(7):1363–1372, 1997. DOI: 10.1016/S0005-1098(97)00028-9.

[24] Malti, R., Rapaić, M., and Turkulov, V. A unified framework for robust stability analysis of linear irrational systems in the parametric space. *Annual Reviews in Control*, 57, 2024. URL: https://hal.archives-ouvertes.fr/hal-03646956.

[25] Mayoufi, A., Malti, R., Chetoui, M., and Aoun, M. System identification of MISO fractional systems: Parameter and differentiation order estimation. *Automatica*, 141:110268, 2022. DOI: 10.1016/j.automatica.2022.110268.

[26] Merlet, J. and Daney, D. Dimensional synthesis of parallel robots with a guaranteed given accuracy over a specific workspace. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, pages 942–947. IEEE, 2005. DOI: 10.1109/ROBOT.2005.1570238.

[27] Montanari, U. and Rossi, F. Constraint relaxation may be perfect. *Artificial Intelligence*, 48(2):143–170, 1991. DOI: 10.1016/0004-3702(91)90059-S.

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

[29] Mustafa, M., Stancu, A., Delanoue, N., and Codres, E. Guaranteed SLAM; an interval approach. *Robotics and Autonomous Systems*, 100:160–170, 2018. DOI: 10.1016/J.ROBOT.2017.11.009.

[30] Neumaier, A. *Interval Methods for Systems of Equations*. Cambridge University Press, Cambridge, UK, 1990. DOI: 10.1002/zamm.19920721114.

[31] Neumaier, A. and Shcherbina, O. Safe bounds in linear and mixed-integer linear programming. *Mathematical Programming*, 99(2):283–296, 2004. DOI: 10.1007/s10107-003-0433-3.

[32] Ramdani, N. and Poignet, P. Robust dynamic experimental identification of robots with set membership uncertainty. *IEEE/ASME Transactions on Mechatronics*, 10(2):253–256, 2005. DOI: 10.1109/TMECH.2005.844703.

[33] Rauh, A. and Jaulin, L. A computationally inexpensive algorithm for determining outer and inner enclosures of nonlinear mappings of ellipsoidal domains. *International Journal of Applied Mathematics and Computer Science*, 31(3):399–415, 2021. DOI: 10.34768/AMCS-2021-0027.

[34] Rauh, A., Senkel, L., Auer, E., and Aschemann, H. Interval methods for real-time capable robust control of solid oxide fuel cell systems. *Mathematics in Computer Science*, 8(3-4):525–542, 2014. DOI: 10.1007/S11786-014-0205-X.

[35] Rego, B., Scott, J., Raimondo, D., and Raffo, G. Set-valued state estimation of nonlinear discrete-time systems with nonlinear invariants based on constrained zonotopes. *Automatica*, 129:109638, 2021. DOI: 10.1016/J.AUTOMATICA.2021.109638.

[36] Rohou, S. Codac (Catalog of domains and contractors). Robex, Lab-STICC, ENSTA-Bretagne, 2021. URL: http://codac.io/.

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

[38] Sainudiin, R. *Machine Interval Experiments: Accounting for the Physical Limits on Empirical and Numerical Resolutions*. LAP Academic Publishers, Köln, Germany, 2010. ISBN: 9783838315997, URL: https://my.lap-publishing.com/catalogue/details/gb/978-3-8383-1599-7/machine-interval-experiments.

[39] Scott, J., Raimondo, D., Marseglia, G., and Braatz, R. Constrained zonotopes: A new tool for set-based estimation and fault detection. *Automatica*, 69:126–136, 2016. DOI: 10.1016/J.AUTOMATICA.2016.02.036.

[40] Turkulov, V., Rapaić, M., and Malti, R. Stability analysis of time-delay systems in the parametric space. *Automatica*, 157, 2023. DOI: 10.1016/J.AUTOMATICA.2023.111220.

[41] Wan, J., Vehi, J., and Luo, N. A numerical approach to design control invariant sets for constrained nonlinear discrete-time systems with guaranteed optimality. *Journal of Global Optimization*, 44:395–407, 2009. DOI: 10.1007/S10898-008-9334-6.

Contents

ISSN 0324—721 X (Print)
ISSN 2676—993 X (Online)

Editor-in-Chief: Tibor Csendes