Volume 25 Number 1

ACTA CYBERNETICA

Szeged, 2021

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. Subscription rates for one issue are as follows: 5000 Ft within Hungary, €40 outside Hungary. Special rates for distributors and bulk orders are available upon request from the publisher. Printed issues are delivered by surface mail in Europe, and by air mail to overseas countries. Claims for missing issues are accepted within six months from the publication date. Please address all requests to:

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://www.inf.u-szeged.hu/en/kutatas/acta-cybernetica` .

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

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

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

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

Andreas Rauh
ENSTA Bretagne
Brest, France
andreas.rauh@interval-methods.de

Heiko Vogler
Department of Computer Science
Dresden University of Technology
Dresden, Germany
Heiko.Vogler@tu-dresden.de

Gerhard J. Woeginger
Department of Mathematics and
Computer Science
Eindhoven University of Technology
Eindhoven, The Netherlands
gwoegi@win.tue.nl

Summer Workshop on Interval Methods

Guest Editors:

Julien Alexandre dit Sandretto

Olivier Mullier

Alexandre Chapoutot

ENSTA Paris, Institut Polytechnique de Paris, Palaiseau France

{alexandre, mullier, chapoutot}@ensta.fr

Preface

The Summer Workshop on Interval Methods (SWIM) is an annual meeting initiated in 2008 by the French MEA working group on Set Computation and Interval Techniques of the French research group on Automatic Control. A special focus of the MEA group is on promoting interval analysis techniques and applications to a broader community of researchers, facilitated by such multidisciplinary workshops. Since 2008, SWIM has become a keystone event for researchers dealing with various aspects of interval and set-based methods.

In 2019, the 12^{th} edition in this workshop series was held at ENSTA Paris, France, with a total of 25 talks.

Traditionally, workshops in the series of SWIM provide a platform for both theoretical and applied researchers who work on the development, implementation, and application of interval methods, verified numerics, and other related (set-membership) techniques. For this edition, given talks were in the fields of

- the verified solution of initial value problems for ordinary differential equations, differential-algebraic system models, and partial differential equations,
- scientific computing with guaranteed error bounds,
- the design of robust and fault-tolerant control systems,
- the implementation of corresponding software libraries, and
- the usage of the mentioned approaches for a large variety of system models in areas such as control engineering, data analysis, signal and image processing.

Seven papers were selected for submission to this Acta Cybernetica special issue. After a two turn peer-review process, six high-quality articles were selected for publication in this special issue. Three papers propose a contribution regarding differential equations, two papers focus on robust control, and one paper consider fault detection.

Julien Alexandre dit Sandretto
Olivier Mullier
Alexandre Chapoutot

Guest Editors

Acta Cybernetica **25** (2021) 5–20.

Interval-based Simulation of Zélus IVPs using DynIbex*

Jason Brown[a] and François Pessaux[b]

Abstract

Modeling continuous-time dynamical systems is a complex task. Fortunately some dedicated programming languages exist to ease this work. Zélus is one such language that generates a simulation executable which can be used to study the behavior of the modeled system. However, such simulations cannot handle uncertainties on some parameters of the system. This makes it necessary to run multiple simulations to check that the system fulfills particular requirements (safety for instance) for all the values in the uncertainty ranges. Interval-based guaranteed integration methods provide a solution to this problem. The DynIbex library provides such methods but it requires a manual encoding of the system in a general purpose programming language (C++). This article presents an extension of the Zélus compiler to generate interval-based guaranteed simulations of IVPs using DynIbex. This extension is conservative since it does not break the existing compilation workflow.

Keywords: DynIbex, Zélus, compilation, hybrid system, interval, guaranteed integration, simulation

1 Introduction

Hybrid systems are commonly defined as dynamical systems mixing discrete and continuous times. They are widely present in control command systems where a continuous physical process is controlled by software components which run at discrete instants. The implementation of such software components involved in many critical systems has to be verified to ensure that the behavior of the global system does not lead to any critical event. One of the verification techniques is to simulate

*This research benefited from the support of the "Chair Complex Systems Engineering - École Polytechnique, THALES, DGA, FX, DASSAULT AVIATION, DCNS Research, ENSTA ParisTech, Télécom ParisTech, Fondation ParisTech and FDO ENSTA" and it is also partially funded by DGA AID. We warmly thank Julien Alexandre dit Sandretto, Alexandre Chapoutot and Olivier Mullier for the many discussions we had to achieve this work.

[a]University of Melbourne, Australia. E-mail: jason.brown@unimelb.edu.au, ORCID: https://orcid.org/0000-0002-1235-3722.

[b]U2IS, ENSTA Paris, France. E-mail: francois.pessaux@ensta-paris.fr, ORCID: https://orcid.org/0000-0002-4219-3854.

DOI: 10.14232/actacyb.285246

the global system. In such a simulation process, the continuous physical process is modeled as differential equations whose solutions are approximated by dedicated integration algorithms. The discrete processing is the software components. Both parts of the system have to interact, allowing the discrete process to react to events of the continuous one.

Simulations can be very dependent on the initial conditions of the system. Small variations may have important impacts. Moreover, the initial conditions may not always be accurately known. A solution to address these uncertainties is to compute using intervals, hence to rely on interval-based integration tools [10, 14, 2, 3, 4, 17] possibly with guaranteed arithmetic [5, 8, 13].

Domain Specific Languages and tools exist to ease the modeling, development and verification of hybrid systems (Modelica, Simulink/Stateflow, LabVIEW, Zélus and others [7]). These languages provide numerous advantages compared to a manual implementation requiring to explicitly bind the code of the software components with the runtime/library of simulation. They often propose high-level constructs (automata, differential equations, guards) with dedicated static verifications (typechecking, initialization analysis, scheduling, causality analysis) and compile the hybrid model to low-level code (C, C++) to produce an executable simulation.

This work proposes to bind the flexibility of a hybrid programming language, Zélus[6], with the safety of interval-based guaranteed integration using DynIbex[1, 9]. Zélus natively generates imperative OCaml code linked with a point-wise simulation runtime. DynIbex is a plug-in of the C++ Ibex library, bringing various validated numerical integration methods to solve Initial Value Problems (*IVPs*). We do not address the compilation of arbitrary Zélus programs toward DynIbex. We present instead the compilation schema for an IVP described in a subset of Zélus to a C++ simulation code using DynIbex. An example is presented, showing the results obtained with the standard Zélus simulation and with the interval-based one.

The rest of the paper is organized as follows. In Section 2, we briefly present Zélus and the features we handle. Section 3 provides a quick introduction to DynIbex and demonstrates how to encode an IVP using the library. Section 4 addresses the compilation schema. Experimental results are described in Section 5. Section 6 is devoted to related work and we conclude and comment on possible further works in Section 7.

2 Zélus Succinctly, Used Features

Zélus is a synchronous programming language extended with ordinary differential equations (ODEs). It provides a wide range of features like synchronous dataflow equations, hierarchical automata, signals, data-types, pattern-matching, functional features, etc. In this paper, we will only address the constructs required to implement IVPs. Zélus makes it possible to model systems in which there is an interaction between discrete-time and continuous-time dynamics. In this work, we do not address any discrete-time behavior.

A program in Zélus is a hierarchy of *nodes*, possibly parameterized, containing equations relating the inputs and outputs of each node. A node can be instantiated in another one to import the equations of the former in the latter, where parameters are replaced by the effective arguments provided at the instantiation point. This mechanism allows the reuse of sets of equations with different parameters. An IVP is represented by a system of coupled equations coming from the equations defined in a node and those imported from instantiated nodes.

The model of a simple harmonic oscillator with dampening, described by the equation $\ddot{x} + k_2\,\dot{x} + k_1\,x = 0$ with initial values $x(0) = 1,\ \dot{x}(0) = 0$, can be written in Zélus as:

```
let hybrid shm_decay (x0, x'0, k1, k2) = x where
  rec der x = x' init x0
  and der x' = -. k1 *. x -. k2 *. x' init x'0

let hybrid main () = y where
  y = shm_decay (1.0, 0.0, 4.0, 0.4)
```

where `der x` represents $\dot{x}$ and `der x'` is $\ddot{x}$. The node `main` instantiates the node `shm_decay` with specific initial values and k_1 and k_2. Note that floating-point arithmetic operators are suffixed by a dot in Zélus.

3 DynIbex in a Few Words, Used Features

DynIbex is a C++ library that builds on the Ibex library. Ibex provides tools to develop programs for constraint processing over real numbers using interval arithmetic and affine arithmetic. DynIbex adds validated numerical integration methods (including handling of floating-point rounding errors). To describe an IVP one defines objects of predefined classes to represent the initial values and the ODEs of the system, using vector-valued representation. That is, initial values are a vector of intervals and the ODEs are "merged" into one unique function whose domain and codomain are vectors of intervals. Once these objects are defined, a dedicated function is called to perform the simulation with the desired parameters (integration method, duration, precision, etc.).

Using the example introduced in Section 2, we show in Figure 1 the expected result of the compilation into C++ code, where the red parts represent code dependent on the compiled IVP. After instantiation of the node `shm_decay` with the effective arguments, the set of equations is `der` $x = x'$ `init` 1 and `der` $x' = -4x - 0.4x'$ `init` 0.

The variable `dim` represents the number of equations of the system, `y` represents the continuous state of the system, `ydot` encodes the differential equations in the `Return` expression. The mapping from the coupled equations `x` and `x'` to the vector-valued representation assigns `x` to the dimension 0 (`y[0]`) and `x'` to the dimension 1 (`y[1]`). All the numerical constants are transformed into single-point intervals taking rounding issues in account. If a float cannot exactly be represented, the obtained interval is the smallest containing this float. The initial conditions of the problem are stored in `yinit`. The number of noise symbols is set using

```
#define T0 (0.0)
#define TEND (6.0)
#define PREC (1e-8)
#define NOISESYMBS (150)
int main () {
  const int dim = 2 ;
  Variable y (dim) ;
  IntervalVector yinit (dim) ;
  Function ydot =
    Function (y, Return (y[1], ((-Interval (4.0)) * y[0]) - (Interval (0.4) * y[1]))) ;
  yinit[0] = Interval (1.0) ;
  yinit[1] = Interval (0.0) ;
  ibex::AF_fAFFullI::setAffineNoiseNumber (NOISESYMBS) ;
  ivp_ode problem = ivp_ode (ydot, T0, yinit) ;
  simulation simu = simulation (&problem, TEND, LC3, PREC) ;
  simu.run_simulation () ;
  simu.export_y0 ("export") ;
  return 0 ;
}
```

Figure 1: IVP encoding in DynIbex (targeted generated C++ code)

`setAffineNoiseNumber`. An IVP object `problem` is created to group the initial values, the equations and the initial time. A simulation object `simu` is created and run. Finally, the results are exported as plain text, providing for each time interval of the simulation the intervals representing the solution of each equation.

4 Compiling an IVP

4.1 Overview

For several reasons, it is impossible to simply rewrite Zélus's backend to make it generate C++ code instead of OCaml code and get hybrid systems for free with DynIbex.

First, the generated OCaml code is tightly dependent on the ODE solver used by Zélus and the solving runtime is very different from DynIbex's mechanisms. Second, the runtime simulation code is deeply mixed with the IVP problem code, making impossible to distinguish between code to be translated into C++, code to be transformed into intervals and code which should be ignored. Finally, intervals are strongly incompatible with point-wise simulation. General Zélus programs may contain discrete code running on events, using continuous values. When working with intervals, these values are no longer exact which makes, for instance, conditional constructs fuzzy : a test may be true and false at the same time. In such a situation, a usual computation cannot be carried out anymore.

For these reasons, we need a dedicated compilation schema to bind Zélus and DynIbex and decide to address only IPV in this work.

In this context, compiling the Zélus code requires two steps. First, the hierarchy of nodes must be flattened, harvesting all the differential equations and their initial values. During this process, each node instantiation expression is replaced by the body of the node where the occurrences of its parameters are replaced by

the effective expressions provided at the instantiation point. This process implies a recursive inlining mechanism which terminates since Zélus forbids recursive nodes. Since DynIbex simulation only accepts a system of differential equations as input, regular dataflow equations must be transformed into differential equations by symbolic differentiation.

Once the intermediate representation of the flattened system is obtained, the multiple equations have to be aggregated into a unique vector-valued function to finally generate the C++ code. Each differential equation corresponds to one dimension of the DynIbex `Function` data-structure. Initial conditions are also transformed into interval vectors. During this process, Zélus expressions are compiled to C++ expressions. Since nodes are flattened, leading to a list of equations, this process mostly consists of a translation of arithmetic expressions into C++, mapping the identifiers to the appropriate vector component, and converting real constants into trivial proper rounded intervals (i.e. the smallest interval containing the translated float).

4.2 Compiling to the Intermediate Representation

The restricted syntax of programs addressed in this work is given in Figure 2.

p	::=	n^+	Program
n	::=	`hybrid` $f\,(x^*)$ = y `where` eq^+	Node
eq	::=	`der` x = e_1 `init` e_2	Differential equation
	|	$x = e$	Regular dataflow equation
e	::=	r	Real numeric constant
	|	x	Identifier
	|	e_1 `op` e_2	Arithmetic expression
	|	$f\,(e^*)$	Node instantiation

Figure 2: Syntax subset of Zélus for IVP

A program p is a list of nodes. A node n is the definition of a parameterized component returning a value y which is the result of one of the equations eq defined in this node. An equation eq may be a differential equation with an initial value e_2 or a regular dataflow equation binding an expression e to an identifier x. Expressions are numeric constants, identifiers, arithmetic expressions or the instantiation of a node named f with expressions. Node instantiations cannot be self-recursive.

The elaboration of the intermediate representation of an IVP proceeds recursively on the Abstract Syntax Tree of the program. The inlining pass relies on an environment E, a partial function from node names to *node descriptions*. We note $\langle\!\langle\, n, d \,\rangle\!\rangle$ the environment mapping the node name n to its description d and use the + symbol to denote the addition of a new binding to an environment. A node description is a triplet $(\mathcal{I}, y, \mathcal{E})$ where $\mathcal{I}$ is the list of the node's formal parameters, y is the output of the node (i.e. the name of one of its equations), $\mathcal{E}$ is the list of *ivpeqs* describing the ODEs of the node. An *ivpeq* is a triplet $\langle\, x, e_1, e_2 \,\rangle$ where x is the name of the differential equation, e_1 is its expression and e_2 is its initial value.

The node instantiation inlining requires a substitution operation on expressions and *ivpeqs*. Such a substitution is a partial application with a finite domain from identifiers to expressions. We denote by $[x_1 \leftarrow e_1; \ldots; x_n \leftarrow e_n]$ the substitution of the domain $\{x_1, \cdots, x_n\}$ associating the expression e_i to the identifier x_i. We extend a substitution to an application from expressions to expressions and from *ivpeqs* to *ivpeqs* as described in the Figure 3.

$$
\begin{array}{lcl}
\langle\, x, e_1, e_2 \,\rangle[y \leftarrow e] & = & \langle\, x, e_1[y \leftarrow e], e_2[y \leftarrow e] \,\rangle \\
r[y \leftarrow e] & = & r \\
x[y \leftarrow e] & = & x \text{ if } x \neq y \\
x[x \leftarrow e] & = & e \\
(e_1 \texttt{ op } e_2)[x \leftarrow e] & = & e_1[x \leftarrow e] \texttt{ op } e_2[x \leftarrow e] \\
f\,(e_1, \ldots e_n)[x \leftarrow e] & = & f\,(e_1[x \leftarrow e], \ldots e_n[x \leftarrow e])
\end{array}
$$

Figure 3: Substitution in an *ivpeq* and expressions

Figure 4 gives an example of the inlining process. In gray is the Zélus code. When processing the node `foo` no changes occur since it contains no instantiation expression. The node `bar` contains two instantiations. For each of them we import the contents of the node where the formal parameters have been replaced by the effective arguments of the instantiation. Then we add this new equation and we replace the instantiation expression by the name of this new equation. Note that to avoid name conflicts, equations are renamed during this process.

```
let hybrid foo (a, b, c) = v where
  der v = -9.8 - c * a init b
```

```
foo: v0, v1, v2 -> v3
  der v3 = -9.8 - (v2 * v0) init v1
```

```
let hybrid bar (y, z, k) = (x1) where
  rec der x2 = foo (x2 - x1, 0, k + 0.5) init 2 + 3
  and der x1 = foo (x2 * x1, z, k) init y
```

```
bar: v0, v1, v2 -> v4
  der v3 = v5 init (2 + 3)
  der v4 = v6 init v0
  der v5 = -9.8 - ((v2 + 0.5) * (v3 - v4)) init 0
  der v6 = -9.8 - (v2 * (v3 * v4)) init v1
```

```
let hybrid main () = x where
  x = bar (1, (1 / 2), 3.14)
```

```
main: -> v0
  der v0 = v3 init 1
  der v1 = v2 init (2 + 3)
  der v2 = -9.8 - ((3.14 + 0.5) * (v1 - v0)) init 0
  der v3 = -9.8 - (3.14 * (v1 * v0)) init (1 / 2)
```

Figure 4: Node instantiation inlining example

Since the `Function` class of Dynlbex can only encode ODEs, it is impossible to directly express the equations for `u` or `v` in the following example.

```
hybrid foo () = u where
  rec der z = u init 3.0
  and u = 5.0 +. z +. v
  and v = 1. +. z
```

We cannot simply inline the body of `u` at each occurrence in the equations since `u` is the return identifier of the node. The solution is to symbolically differentiate `5 + z + v`, i.e. create the expressions corresponding to $\delta(\texttt{5} + \texttt{z} + \texttt{v})$ and the initial value of `5 + z + v`. Inside this expression, the identifier `z` represents an ODE, so its derivative is exactly the body of `z`. However, `v` represents a regular dataflow equation. Hence `v` must be replaced by the (recursive) differentiation of its body. Thus, the whole equation is replaced by `der u = 0 + u + 0 + u`. The initial value of this new equation is the original body of `u` in which `z` is replaced by the initial value of `der z` and `v` is replaced by its body in which we recursively apply the transformation. The obtained result is `init 5 + 3 + 1 + 3`. In summary, while processing a node definition, any equation defined by `y` = e must be replaced by `der y` = $\mathcal{B}(e)$ `init` $\mathcal{I}(e)$ where $\mathcal{B}(e)$ and $\mathcal{I}(e)$ are given in Figure 5.

$$
\begin{array}{lcl}
\mathcal{B}(r) & = & 0 \\
\mathcal{B}(e_1 + e_2) & = & \mathcal{B}(e_1) + \mathcal{B}(e_2) \\
\mathcal{B}(e_1 * e_2) & = & \mathcal{B}(e_1) * e_2 + e_1 * \mathcal{B}(e_2) \\
\mathcal{B}(e_1 / e_2) & = & (\mathcal{B}(e_1) * e_2 - e_1 * \mathcal{B}(e_2)) \;/\; (e_2 * e_2) \\
\mathcal{B}(x) & = & e_1 \quad \textit{if } x \textit{ is defined by } \texttt{der } x \texttt{ = } e_1 \texttt{ init } e_2 \\
\mathcal{B}(x) & = & \mathcal{B}(e) \quad \textit{if } x \textit{ is defined by } x \texttt{ = } e \\
\\
\mathcal{I}(r) & = & r \\
\mathcal{I}(e_1 + e_2) & = & \mathcal{I}(e_1) + \mathcal{I}(e_2) \\
\mathcal{I}(e_1 * e_2) & = & \mathcal{I}(e_1) * \mathcal{I}(e_2) \\
\mathcal{I}(e_1 / e_2) & = & \mathcal{I}(e_1) \;/\; \mathcal{I}(e_2) \\
\mathcal{I}(x) & = & e_2 \quad \textit{if } x \textit{ is defined by } \texttt{der } x \texttt{ = } e_1 \texttt{ init } e_2 \\
\mathcal{I}(x) & = & \mathcal{I}(e) \quad \textit{if } x \textit{ is defined by } x \texttt{ = } e
\end{array}
$$

Figure 5: $\mathcal{B}(e)$ and $\mathcal{I}(e)$ rules

The compilation rules for expressions are given in Figure 6. The judgment $E \vdash e_1 \rightarrow_4 (e_2, \mathcal{D})$ means that, in the environment E, the expression e_1 is transformed into the expression e_2 and produces the set of *ivpeqs* $\mathcal{D}$.

The rule Num handles a real numeric constant and produces the same constant with an empty set of *ivpeqs*. The rule Id handles identifiers the same way. The rule App handles the node instantiation and is in charge of the effective inlining. The name f is expected to be bound in the environment to a node description. The expression returned by the rule App is the identifier naming the output of f and the *ivpeqs* set is the one of f in which all the occurrences of the formal parameters x_i of f have been substituted by the corresponding effective argument expressions e'_i. In this rule, to simplify the presentation, we omit the renaming of all the identifiers of the inlined node by fresh variables (i.e. not appearing anywhere in the program). This renaming is mandatory to avoid variable capture. Finally, the rule Opp recursively processes the two sub-expressions to rebuild an arithmetic

$$(\textsc{Num})\frac{}{E \vdash r \to_4 (r,\ \varnothing)} \qquad (\textsc{Id})\frac{}{E \vdash x \to_4 (x,\ \varnothing)}$$

$$(\textsc{App})\frac{\begin{array}{c} E(f) = (\{x_1, \ldots, x_n\}, y, \{\langle\, d_1 \,\rangle, \ldots, \langle\, d_m \,\rangle\}) \\ E \vdash e_1 \to_4 (e'_1,\ \mathcal{D}_1) \quad \ldots \quad E \vdash e_n \to_4 (e'_n,\ \mathcal{D}_n) \\ d'_1 = d_1[x_i \leftarrow e'_i]_{i=1\ldots n} \quad \ldots \quad d'_m = d_m[x_i \leftarrow e'_i]_{i=1\ldots n} \end{array}}{E \vdash f\,(e_1, \ldots, e_n) \to_4 (y,\ \{\langle\, d'_1 \,\rangle, \ldots, \langle\, d'_m \,\rangle\} \cup \mathcal{D}_1 \cup \ldots \cup \mathcal{D}_n)}$$

$$(\textsc{Op})\frac{E \vdash e_1 \to_4 (e'_1,\ \mathcal{D}_1) \quad E \vdash e_2 \to_4 (e'_2,\ \mathcal{D}_2)}{E \vdash e_1\, \mathtt{op}\, e_2 \to_4 (e'_1\, \mathtt{op}\, e'_2,\ \mathcal{D}_1 \cup \mathcal{D}_2)}$$

Figure 6: Compilation of expressions

expression and the union of the *ivpeqs* obtained for each sub-expression.

The compilation rules for equations are given in Figure 7. The judgment $E \vdash eq \to_3 \mathcal{D}$ states that the equation eq produces the set of *ivpeqs* $\mathcal{D}$.

$$(\textsc{Eq})\frac{E \vdash e \to_4 \left(e',\ \mathcal{D}\right) \quad e_b = \mathcal{B}\,(e') \quad e_i = \mathcal{I}\,(e')}{E \vdash x = e \to_3 \langle\, x, e_b, e_i \,\rangle + \mathcal{D}}$$

$$(\textsc{Der})\frac{E \vdash e_1 \to_4 \left(e'_1,\ \mathcal{D}_1\right) \quad E \vdash e_2 \to_4 \left(e'_2,\ \mathcal{D}_2\right)}{E \vdash \mathtt{der}\ x = e_1\ \mathtt{init}\ e_2 \to_3 \langle\, x, e'_1, e'_2 \,\rangle + \mathcal{D}_1 \cup \mathcal{D}_2}$$

Figure 7: Compilation of equations

The rule Eq handles regular dataflow equations. It transforms the body e of the equation to obtain the *ivpeqs* representing inlined node instantiation expressions of e. The equation is differentiated and added as a new *ivpeq*. The rule Der performs the inlining transformation on the body and initial value of the differential equation, and returns the set made of the *ivpeqs* obtained for each sub-expression extended with the one created for x.

Finally, the compilation rules for nodes then programs are given in Figure 8.

$$(\textsc{Node})\frac{\begin{array}{c} E \vdash eq_1 \to_3 \mathcal{D}_1 \quad \ldots \quad E \vdash eq_n \to_3 \mathcal{D}_n \\ \langle\!\langle\, f,\ (\{\, x_1, \ldots, x_n \,\},\ y,\ \mathcal{D}_1 \cup \ldots \cup \mathcal{D}_n) \,\rangle\!\rangle + E \end{array}}{E \vdash \mathtt{hybrid}\ f\,(x_1, \ldots, x_m) = y\ \mathtt{where}\ eq_1 \ldots eq_n \to_2}$$

$$(\textsc{Prg})\frac{E \vdash nod_1 \to_2 E_1 \quad \ldots \quad E_{n-1} \vdash nod_n \to_2 E_n}{E \vdash \{nod_1, \ldots, nod_n\} \to_1 E_n}$$

$$(\textsc{Top})\frac{\begin{array}{c} \varnothing \vdash p \to_1 E \quad \exists\, \langle\!\langle\, \mathtt{main},\ (\{\, x_1, \ldots, x_m \,\}, y, \{\, eq_1, \ldots, eq_n \,\}) \,\rangle\!\rangle \in E \\ eq_i = \langle\, x_i, e_i, d_i \,\rangle_{i=1\ldots n} \quad \sigma = [x_i \mapsto i] \end{array}}{t_0, t_f \vdash p \to (n, \{\, e_1, \ldots, e_n \,\}, \{\, d_1, \ldots, d_n \,\}, t_0, t_f),\ \sigma}$$

Figure 8: Rules for intermediate representation synthesis

The rule Node processes one node, harvesting the *ivpeqs* obtained from its equations. It returns the environment extended with the node description of f.

The rule PRG iterates this process on the list of nodes making the program. The description of each processed node is added to the environment to process the remaining ones. Since Zélus imposes that a node is always defined before any use of it, this ensures that we will always find the node description bound to a node identifier present in an instantiation. The rule TOP processes all the nodes of the program p, then looks for a node named `main`. It builds the representation of the IVP as a tuple containing the size of the problem, the list of ODEs expressions, the list of initial values, the initial (t_0) and final (t_f) dates of the simulation and a substitution σ. This substitution maps each identifier binding an equation to an integer representing the rank of its equation. It will be used during the C++ code generation.

Optimization The combination of the instantiation inlining and the differentiation process can produce duplicated equations which is inefficient since this increases the size of the system. Consider the Zélus program given in Section 2. After the inlining in the node `main`, we get the set of equations :

`der` $x = x'$ `init` 1
`der` $x' = -4 * x - 0.4 * x'$ `init` 0
$y = x$

which is transformed during differentiation of y into :

`der` $x = x'$ `init` 1
`der` $x' = -4 * x - 0.4 * x'$ `init` 0
`der` $y = x'$ `init` 1

where `x` and `y` are redundant. To overcome this inefficiency, when processing equations of the form $y = x$, we drop the equation and replace y by x in the node (i.e. in its equations and its return value).

4.3 Generating C++ Code

The C++ code generation mostly consists of generating a template like the one shown in Figure 1, where the dimension of the system, the `Function` object and the initial conditions are instantiated from the intermediate representation of the compiled IVP. Hence, the code generation relies on the intermediate representation of the node `main` and on some data outside the model (duration, integration method, precision, number of noise symbols) provided at compile-time.

The rule TOP already provides the list of initial values and differential expressions. Most of what remains is the translation of arithmetic expressions into C++. The substitution σ returned by this rule is used while generating the initial values and the `Function` object representing the equations. It allows the replacement of identifiers by accesses in the arrays used for DynIbex's vector-valued representation of the equations. Thus, in the `Function` object, an identifier x is translated into `y[`$\sigma(x)$`]`. In the same way, the initial value of x is set in `yinit[`$\sigma(x)$`]`. A numerical constant n is translated into the trivial interval `Interval (`n`)`.

5 Experimental Results

We extended the Zélus compiler to implement the described compilation process. This new backend operates on the intermediate representation obtained after type, causality and initialization analyses and does not interfere with the standard compilation.

The first experiment was to simulate the system with Zélus and with our generated code, then to compare the results. In Figure 9, the Zélus native simulation is represented by the red line and the simulation obtained using the intervals is shown by the green boxes.

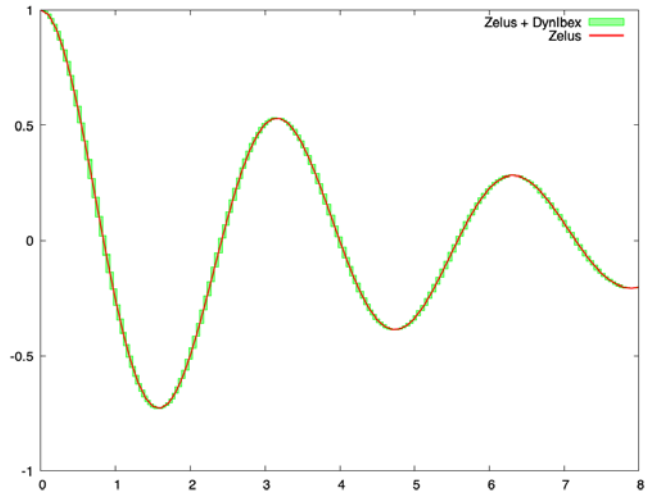


Figure 9: Simulations with/without intervals

The two simulations behave consistently. In particular, the results obtained with the standard integration runtime of Zélus always remain inside the boxes obtained using the intervals mechanism. This suggests that the native integration runtime of Zélus is precise enough in this example to avoid inaccuracies that could be caused by float rounding errors.

The Zélus syntax has been extended to specify an alternative interval value for any float value. This interval is only taken into account when compiling toward DynIbex. Hence, it is possible to add uncertainty on the initial value of `der x`, by writing `y = shm_decay (1.0 [0.9; 1.0], 0.0, 4.0, 0.4)`. The "default" value `1.0` is then ignored and the interval `[0.9; 1.0]` is considered instead. The simulation obtained after this change, displayed in Figure 10, shows that both simulations continue to behave consistently, but the effect of the uncertainty becomes clearly visible.

It is also possible to add uncertainty on the coefficients of the equations. The examples in Figure 11 are based on a value of k_2 in $[0.3; 0.4]$ instead of 0.4. At

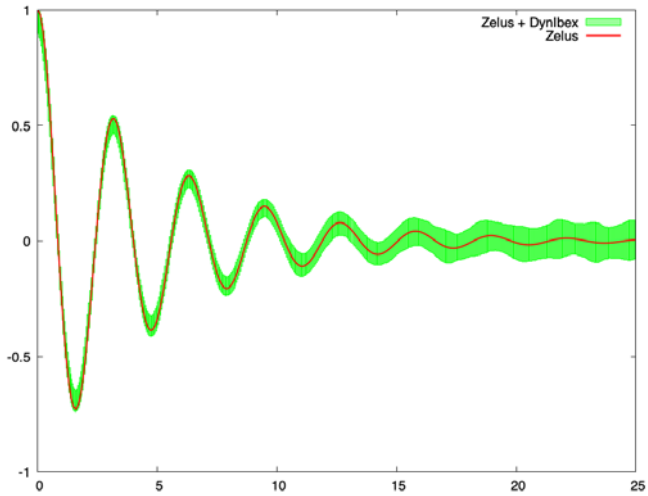


Figure 10: Simulation with initial uncertainty

compile-time, it is possible to select different integration methods, precisions, etc. From left to right, top to bottom, we used third-order Runge-Kutta 10^{-10}, Heun's method 10^{-6}, fourth-order Runge-Kutta 10^{-6} and fourth-order Gauss-Legendre 10^{-12}.

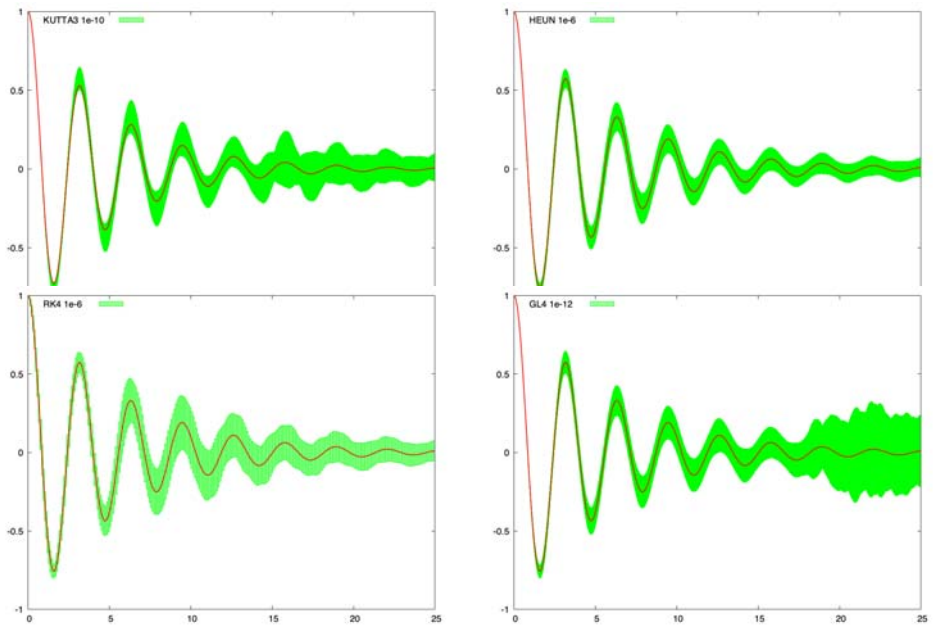


Figure 11: Simulation with parameter uncertainty

Despite a loss of precision, simulating with intervals to model uncertainties allows one to run one unique simulation instead of several point-wise simulations to try to cover the entire range of uncertainty. The simulation result is less accurate but safe and guaranteed. Running several point-wise simulations is not satisfactory: how many must be ran, which values must be chosen? There is no guaranty that the chosen strategy covers all the possible behaviors. The Figure 12 shows the inclusion of several point-wise simulations in a single interval-based simulation. In the top

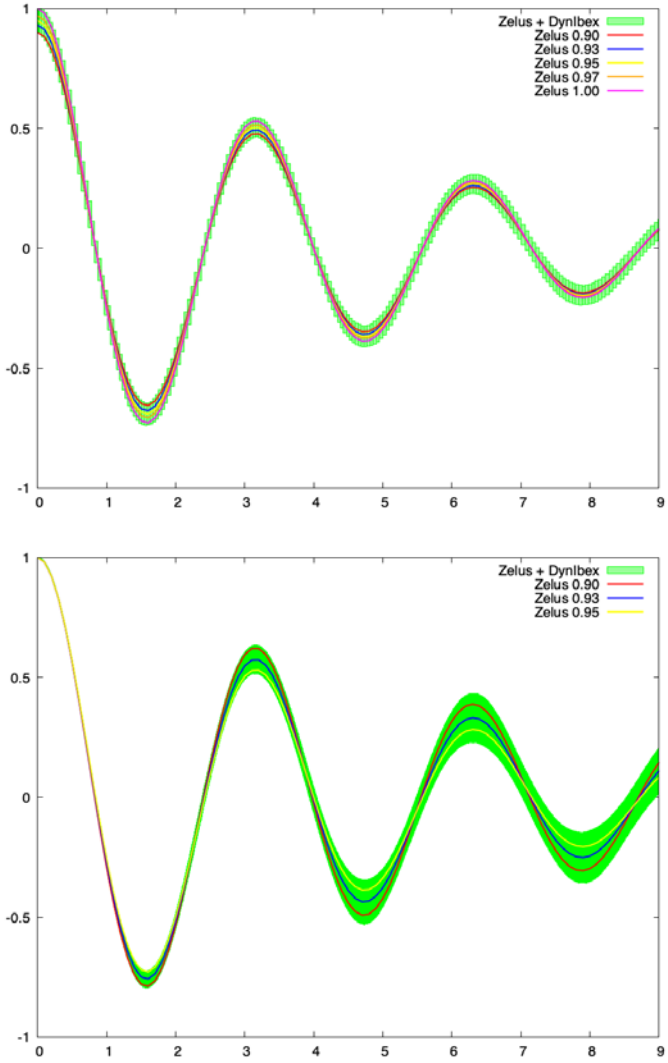


Figure 12: Point-wise simulations vs interval simulation

picture, the interval for the initial condition of `der x` is $[0.9; 1]$. In the bottom picture, the interval for $k2$ is $[0.3; 0.4]$. We plot several point-wise simulations in these ranges of uncertainty. As one expects for safety purposes, the interval-based simulations cover all the point-wise ones.

6 Related Work

Several frameworks exist for reachability analysis of hybrid systems, though few have a high-level programming language in which to directly describe the systems.

- JuliaReach [14] is a Julia library performing set-based reachability analysis on both linear and non-linear systems. The description of the system to analyze has to be manually encoded in Julia using the tools provided by the library.

- CORA [2] is a toolbox written in MATLAB providing advanced data-structures and reachability algorithms for linear and non-linear systems. It has been extended with intervals and Taylor models [3, 4]. The modeling of a system is manually done in MATLAB, hence considering floating-point arithmetic as exact.

- SpaceEx [10] is a verification platform to model linear (or piece-wise linear) hybrid systems and compute the sets of reachable states using different reachability algorithms. It relies on floating-point arithmetic, though it fails to account for rounding errors. Systems are modeled in a dedicated interface (or in a XML file). A graphical WEB interface is available to set parameters, run simulations and visualize the results. A translator from a subset of Simulink to SpaceEx is also available [16].

- FLOW* [8] allows one to model non-linear hybrid systems with uncertainties and compute an over-approximation of reachable states using Taylor models and guaranteed floating-point arithmetic.

- Hyson [5] allows one to perform set-based simulations of Simulink hybrid models (continuous and discrete, linear and non-linear, using a subset of the language). It provides guaranteed integration based on Runge Kutta method with handling of floating-point rounding errors. It can be considered as an ancestor of this work.

- dReal [11, 12] encodes hybrid systems as SMT modulo ODEs problems. A system has to be encoded as a first-order logic formula with the properties it must respect in the SMT-LIB format. It is then able to check if the properties of the system hold. However, it does not provide high-level constructs to write the systems.

- The Acumen [17, 15] framework provides ways to express various kinds of hybrid systems in a Domain Specific Language. It allows point-wise simulations to provide very fancy dynamic visual representations. It also provides an enclosure interpreter supporting intervals to handle uncertainties for system verification purposes. The main differences with our work come from the nature of the input language and the integration mechanism. Using Zélus provides various static analyses and advanced constructs. Though we do not handle some of these constructs here, work is underway to address a larger subset of the Zélus language, including automata, and thus model more complex systems. DynIbex was chosen as a simulation framework as it provides guaranteed integration methods which handles floating-point issues.

7 Future Work

This exploratory work can be extended in three directions. The first direction aims at considering more complex systems. We would like to address IVPs with reset conditions (guards). In such systems, the ODEs are fixed, however the continuous state can be set to particular values on some conditions. To go further, we would like to address systems with several dynamics. In these systems, the ODEs involved in the dynamics may change depending on the state of the system. This will naturally be represented in Zélus by automata. Some obvious issues arise due to the temporal and spacial uncertainties represented by the intervals when changing state in an automaton. Moreover, the compilation schema to implement this will have to remain compatible with the IVPs simulation presented here, while also handling the more elaborate constructs of Zélus.

The second direction is to add contracts in Zélus and to be able to check if they hold during the simulation. First, the form of properties to verify must be chosen since this will have an impact on the logic to use. Then there is the choice of when to verify the contracts: during the simulation or at the end of the simulation. Finally, we need to compile the formulae and represent them with DynIbex constructs.

The last extension of this work is to address the discrete behavior it is possible to model in Zélus. The programs currently supported are restricted to continuous computation. This enables us to model the dynamics of a physical system but not a controller coupled to this system.

8 Conclusion

We presented a mechanism to compile IVPs described in Zélus to C++ code using DynIbex. This allows the simulation of programs written in a high-level programming language with interval-based validated numerical integration methods. Various parameters can be set at compile-time to tune the simulation accuracy. This work is fully implemented in the Zélus compiler. Extensions to handle more complex systems and to compile contracts verification on programs are in progress.

References

[1] Alexandre dit Sandretto, Julien, Chapoutot, Alexandre, and Mullier, Olivier. The DynIbex library. `https://perso.ensta-paris.fr/~chapoutot/dynibex/`.

[2] Althoff, Matthias. An introduction to CORA 2015. In Frehse, Goran and Althoff, Matthias, editors, *ARCH14-15. 1st and 2nd International Workshop on Applied veRification for Continuous and Hybrid Systems*, volume 34 of *EPiC Series in Computing*, pages 120–151. EasyChair, 2015. DOI: `10.29007/zbkv`.

[3] Althoff, Matthias and Grebenyuk, Dmitry. Implementation of interval arithmetic in CORA 2016. In *ARCH@CPSWeek*, volume 43 of *EPiC Series in Computing*, pages 91–105. EasyChair, 2016. DOI: `10.29007/w19b`.

[4] Althoff, Matthias, Grebenyuk, Dmitry, and Kochdumper, Niklas. Implementation of taylor models in CORA 2018. In *ARCH@ADHS*, volume 54 of *EPiC Series in Computing*, pages 145–173. EasyChair, 2018. DOI: `10.29007/zzc7`.

[5] Bouissou, Olivier, Mimram, Samuel, and Chapoutot, Alexandre. HySon: Set-based simulation of hybrid systems. In *Proceedings of IEEE International Symposium on Rapid System Prototyping*, pages 79–85, 2012. DOI: `10.1109/RSP.2012.6380694`.

[6] Bourke, Timothy and Pouzet, Marc. Zélus: A synchronous language with ODEs. In *Proceedings of the 16th International Conference on Hybrid Systems: Computation and Control*, pages 113–118. ACM, 2013. DOI: `10.1145/2461328.2461348`.

[7] Carloni, Luca P., Passerone, Roberto, Pinto, Alessandro, and Angiovanni-Vincentelli, Alberto L. Languages and tools for hybrid systems design. *Found. Trends Electron. Des. Autom.*, 1(1):1–193, 2006. DOI: `10.1561/1000000001`.

[8] Chen, Xin, Ábrahám, Erika, and Sankaranarayanan, Sriram. Flow*: An analyzer for non-linear hybrid systems. In *Proceedings of the 25th International Conference on Computer Aided Verification - Volume 8044*, CAV 2013, pages 258–263, New York, NY, USA, 2013. Springer-Verlag New York, Inc. DOI: `10.1007/978-3-642-39799-8_18`.

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

[10] Frehse, Goran, Le Guernic, Colas, Donzé, Alexandre, Cotton, Scott, Ray, Rajarshi, Lebeltel, Olivier, Ripado, Rodolfo, Girard, Antoine, Dang, Thao, and Maler, Oded. SpaceEx: Scalable verification of hybrid systems. In Ganesh Gopalakrishnan, Shaz Qadeer, editor, *Proc. 23rd International Conference on Computer Aided Verification (CAV)*, LNCS, pages 379–395. Springer, 2011. DOI: `10.1007/978-3-642-22110-1_30`.

[11] Gao, S., Kong, S., and Clarke, E. M. dReal: An SMT solver for nonlinear theories over the reals. In *International Conference on Automated Deduction*, volume 7898 of *Lecture Notes in Computer Science*, pages 208–214. Springer, 2013. DOI: 10.1007/978-3-642-38574-2_14.

[12] Gao, Sicun, Kong, Soonho, and Clarke, Edmund M. Satisfiability modulo ODEs. In *Proceedings of Formal Methods in Computer-Aided Design*. IEEE, 2013. DOI: 10.1109/FMCAD.2013.6679398.

[13] Henzinger, Thomas A., Horowitz, Benjamin, Majumdar, Rupak, and Wong-Toi, Howard. Beyond HyTech: Hybrid systems analysis using interval numerical methods. In *Hybrid Systems: Computation and Control*, pages 130–144. Springer Berlin Heidelberg, 2000. DOI: 10.1007/3-540-46430-1_14.

[14] Immler, Fabian, Althoff, Matthias, Chen, Xin, Fan, Chuchu, Frehse, Goran, Kochdumper, Niklas, Li, Yangge, Mitra, Sayan, Tomar, Mahendra Singh, and Zamani, Majid. ARCH-COMP18 category report: Continuous and hybrid systems with nonlinear dynamics. In Frehse, Goran, editor, *ARCH18. 5th International Workshop on Applied Verification of Continuous and Hybrid Systems*, volume 54 of *EPiC Series in Computing*, pages 53–70. EasyChair, 2018. DOI: 10.29007/mskf.

[15] Konečný, Michal, Taha, Walid, Duracz, Jan, Duracz, Adam, and Ames, Aaron. Enclosing the behavior of a hybrid system up to and beyond a Zeno point. In *2013 IEEE 1st international conference on cyber-physical systems, networks, and applications (CPSNA)*, pages 120–125, United States, 2013. IEEE. DOI: 10.1109/CPSNA.2013.6614258.

[16] Minopoli, Stefano and Frehse, Goran. SL2SX translator: From Simulink to SpaceEx models. In *Proceedings of the 19th International Conference on Hybrid Systems: Computation and Control*, pages 93–98, 04 2016. DOI: 10.1145/2883817.2883826.

[17] Zeng, Yingfu, Rose, Chad G., Brauner, Paul, Taha, Walid, Masood, Jawad, Philippsen, Roland, O'Malley, Marcia K., and Cartwright, Robert. Modeling basic aspects of cyber-physical systems, part II. In *Procceedings of the 2014 IEEE Intl Conf on High Performance Computing and Communications*, 2014. DOI: 10.1109/HPCC.2014.119.

Acta Cybernetica **25** (2021) 21–48.

Toward the Development of Iteration Procedures for the Interval-Based Simulation of Fractional-Order Systems

Andreas Rauh and Julia Kersten[a]

Abstract

In many fields of engineering as well as computational physics, it is necessary to describe dynamic phenomena which are characterized by an infinitely long horizon of past state values. This infinite horizon of past data then influences the evolution of future state trajectories. Such phenomena can be characterized effectively by means of fractional-order differential equations. In contrast to classical *linear* ordinary differential equations, *linear* fractional-order models have frequency domain characteristics with amplitude responses that deviate from the classical integer multiples of $\pm 20\,\mathrm{dB}$ per frequency decade and, respectively, deviate from integer multiples of $\pm\frac{\pi}{2}$ in the limit values of their corresponding phase response. Although numerous simulation approaches have been developed in recent years for the numerical evaluation of fractional-order models with point-valued initial conditions and parameters, the robustness analysis of such system representations is still a widely open area of research. This statement is especially true if interval uncertainty is considered with respect to initial states and parameters. Therefore, this paper summarizes the current state-of-the-art concerning the simulation-based analysis of fractional-order dynamics with a restriction to those approaches that can be extended to set-valued (interval) evaluations for models with bounded uncertainty. Especially, it is shown how verified simulation techniques for integer-order models with uncertain parameters can be extended toward fractional counterparts. Selected linear as well as nonlinear illustrating examples conclude this paper to visualize algorithmic properties of the suggested interval-based simulation methodology and point out directions of ongoing research.

Keywords: interval analysis, fractional-order differential equations, Picard iteration, exponential enclosure techniques, Mittag-Leffler functions

[a]University of Rostock, Chair of Mechatronics, Justus-von-Liebig-Weg 6, D-18059 Rostock, Germany. E-mail: Andreas.Rauh@interval-methods.de, Julia.Kersten@uni-rostock.de. ORCID: https://orcid.org/{0000-0002-1548-6547, 0000-0002-1160-8623}.

DOI: 10.14232/actacyb.285660

1 Introduction

Simulation procedures for fractional-order systems have been investigated in many current research projects. Such simulation procedures involve the numerically efficient and accurate evaluation of functions of the Mittag-Leffler type, the implementation of numerically efficient and robust simulation routines based on the Grünwald-Letnikov differentiation operator for linear and nonlinear system models, Laplace domain representations for the linear case, the development of frequency domain-based procedures and the design of software packages such as CRONE TOOLBOX. This toolbox includes computational routines for fractional-order time- and frequency-domain system identification, fractional-order path planning techniques, and approaches for a fractional-order control synthesis [10, 12, 13, 23, 27, 30].

Among the approaches mentioned above, the *Grünwald-Letnikov operator* for numerically approximating fractional-order derivatives and the corresponding numerical integration of fractional system models is widely used in engineering applications. It is based on a temporal series expansion of the fractional derivative operator and coincides, when setting the fractional differentiation order to one (i.e., considering classical first-order ordinary differential equations) to the well-known temporal Taylor series expansion of the solution to an ordinary differential equation that is typically truncated after some finite order in any numerical simulation of dynamic systems. The general drawback of this numerical evaluation scheme is the necessity for a large number of summands in the series expansion to capture the long-term memory effects of fractional systems with sufficient accuracy. Although the Grünwald-Letnikov operator is generally applicable to linear as well as nonlinear fractional-order models, the large number of required terms in the series expansion prevents its naive use to general system models with uncertain parameters and initial conditions due to the inevitably arising wrapping effect. To avoid this wrapping effect, that also occurs if high-order series expansions are applied in the case of classical ordinary differential equations with purely integer-order derivatives, this paper is focused on using quasi-analytic representations of the enclosure of the systems' time responses by means of Mittag-Leffler functions [10]. Those functions represent a generalization of classical exponential functions and can be exploited — as shown for the first time in [33, 35] — to represent enclosures for the sets of reachable states for fractional-order systems if they are extended to the case of interval arguments.

Besides series expansions in the time domain, also *frequency-domain approximations* can be determined [4, 9, 27, 28, 30]. They make use of approximating the amplitude and phase responses by multiplicative concatenations (i.e., series concatenations in the respective signal flow) of fundamental linear Bode plot elements corresponding to first- and second-order lead and lag elements and, under some circumstances, input-output transport delays. As such, these frequency-domain techniques can be seen as the approximation of the fractional-order Laplace-domain transfer functions by using Taylor, respectively, Laurent series to approximate their numerators and denominators by expressions with integer-order powers of the Laplace variable (except for an isolated classical transport delay operator). Due

to this strong relation to the Taylor and Laurent series expansion in the Laplace domain, the resulting integer-order approximations are only applicable to systems with dominating linear dynamics within a restricted frequency band. Although such approximations are commonly not suitable for accurate system simulations, they are well suited, if control design with sufficiently strong low-pass filter behavior is concerned and, for example, for the experimental identification of fractional models in restricted frequency bands with applications in engineering and biomedical tasks such as impedance spectroscopy for battery systems [2,38,39], rheological material properties [29], or the study of visco-elastic properties of blood cells [5].

Linear control approaches that are described by fractional-order transfer functions can be interpreted as extensions of classical output feedback routines of PID type (proportional, integrating, differentiating) by replacing the integrating and differentiating elements with their respective fractional-order generalizations [23, 27].

As already mentioned in the discussion of the Grünwald-Letnikov operator, the amount of memory required to accurately represent the flow of a fractional-order system model may become prohibitively large if long integration horizons are considered. Hence, techniques for a short-term memory storage, going along with quantifying the errors arising from restarting the numerical integration of a fractional-order system at some specific point in time, are crucial in practice. In this paper, techniques for quantifying the effect of resetting the temporal derivative at some point of time will be used to describe guaranteed interval enclosures of the arising errors. These interval enclosures are then interfaced in a novel manner with the basic iteration approaches published in [33, 35] by combining them with the solution representations in terms of Mittag-Leffler functions. In such a way, the iterative solution scheme developed by the authors can be employed more efficiently for simulation scenarios in which long prediction horizons are of practical interest.

Although the numerical integration routines based on temporal series expansions, such as the Grünwald-Letnikov operator, are practically useful for a large variety of fractional-order system models that are characterized by point-valued system parameters and precisely known initial conditions, research concerning the analysis of uncertain but bounded parameters is still at the very beginning if fractional models are concerned. This problem has not yet received the same amount of attention as for the case of integer-order sets of ordinary differential equations. To the knowledge of the authors, only initial works were performed in this direction which are based on generalizations of the Picard iteration to the fractional case [22]. Using this iteration, it becomes possible to compute guaranteed outer interval enclosures for those states that are reachable over a sufficiently short prediction horizon. However, these enclosures — resulting from the integral formulation of the Picard iteration, see Theorem 3 in this paper — are typically quite conservative due to the fact that the resulting bounding boxes describe *time-invariant* state bounds that are valid for the complete prediction window.

In contrast to fractional-order systems, the task of verified simulation and reachability analysis has been studied extensively over the last decades in the frame of integer-order dynamic system models and corresponding control laws. Such techniques are readily applicable in terms of software-based simulation packages and

can be employed — among others — for the verification of safety constraints of dynamic systems. State-of-the-art general-purpose initial value problem solvers for such tasks make use of so-called verified simulation approaches which are based on either interval analysis, zonotopic representations of the sets of reachable states, or Taylor model arithmetic [26, 33, 36].

By means of set-valued computations, these solvers avoid time-consuming gridding techniques and Monte-Carlo sampling, where it has to be pointed out additionally that neither of these gridding and sampling techniques can provide any guarantee of determining outer solution tubes that contain the exact sets of reachable states of a general dynamic system model with absolute certainty. In contrast to grid- or sampling-based approaches, the fundamental property of those tubes computed with the help of verified approaches is that a guaranteed outer hull of the solutions of the underlying uncertain system model is determined [26]. To account for specific system properties such as a-priori proven asymptotic stability of the system dynamics (which is often verified in advance if a guaranteed stabilizing control design has been performed prior to evaluating the state equations), an exponential enclosure technique was developed by the working group of the authors for the class of integer-order systems [36]. Relations of this exponential enclosure approach to specific system properties such as cooperativity and positivity of a dynamic system [37] were published in [33]. If these latter properties are guaranteed to be satisfied, it becomes possible to evaluate lower and upper bounding trajectories independently during the numerical simulation. Such properties are often exploited during the design of interval observers which can analogously be derived for both, integer-order and fractional-order system representations [3, 8, 20, 31].

If state equations are not a-priori proven to be cooperative, especially the use of the exponential enclosure technique allows for reducing overestimation (i.e., to avoid unphysically wide bounds for the computed state trajectories) in the case of asymptotically stable dynamics. This is caused by the fact that the exponential enclosure technique [33, 35] aims at preserving stability properties in combination with a reduction of the wrapping effect [16]. As shown in previous work, this approach is most successful if a transformation of the state equations into a quasi-linear system representation exists [35]. This transformation then has to ensure that the simulation routine makes use of a set of quasi-linear state equations given by a diagonally dominant form.

In this paper, the exponential enclosure technique is further developed for fractional-order systems, where exponential functions describing the guaranteed state enclosures have to be replaced by functions of the Mittag-Leffler type. The use of Mittag-Leffler functions as the corresponding ansatz for the solution enclosures is motivated by the fact that these functions represent the exact solutions of linear fractional-order models with precisely known parameters, cf. [7, 11]. To allow for an efficient implementation of numeric simulation routines, preconditioning strategies of the state equations into a diagonally dominant form, the influence of truncation errors occurring from a finite-time approximation of the fractional-order systems' memory, and monotonicity properties of Mittag-Leffler functions with respect to its argument and with respect to the non-integer differentiation order need to be

investigated. These aspects are discussed in detail in the current paper together with novel extensions towards the quantification of truncation errors resulting from infinite memory effects of fractional-order dynamics.

Following the summary of preliminaries and the state-of-the-art in Secs. 2 and 3, the exponential enclosure technique for interval-valued uncertain systems, which was so far primarily studied for classical sets of ordinary differential equations, is generalized to fractional-order models in Sec. 4. Here, we rely on the iteration procedure stated already in [33, 35]. In the current paper, relations of this approach to the state-of-the-art, especially the integral formulation of the Picard iteration in Theorem 3, and extensions by a more detailed analysis of monotonicity properties allowing for an efficient implementation in an interval arithmetic framework together with handling temporal truncation errors are worked out as the novel contributions in Sec. 5. Sec. 6 provides illustrating linear and nonlinear examples for the use of the proposed enclosure technique before the paper is concluded with an outlook on future work in Sec. 7.

2 Preliminaries

In the course of this paper, simulation routines for the case of integer-order differential equations are first summarized. These routines are based on an interval-based exponential enclosure technique. According to the corresponding publications in [33, 36], they result from a differential formulation of the Picard iteration. Second, they are generalized toward the counterpart of fractional-order dynamics. This generalization is essentially based on the replacement of exponential functions by suitable Mittag-Leffler functions as already motivated in [33, 35].

For that purpose, the two-parameter Mittag-Leffler function[1] [10, 12, 14] is denoted by

$$E_{\nu,\beta}(\zeta) = \sum_{i=0}^{\infty} \frac{\zeta^i}{\Gamma(\nu i + \beta)} \tag{1}$$

with the general argument $\zeta \in \mathbb{C}$, the gamma function $\Gamma(\nu i + \beta)$, as well as the parameters $\nu \in \mathbb{R}_+$ and $\beta \in \mathbb{R}$.

All system models in this paper are assumed to be given in terms of explicit, autonomous, time-invariant[2] state equations which are re-written — if possible — according to Def. 1 into a quasi-linear form to enhance efficiency of the numerical evaluation.

Definition 1 (Quasi-linear system model)**.** *After factoring out the state vector* $\mathbf{x}(t) \in \mathbb{R}^n$ *of a nonlinear autonomous system, initial value problems for quasi-linear*

[1] The two-parameter Mittag-Leffler function serves as an exact solution representation for linear fractional-order differential equations according to [7, 11].

[2] Note, the restriction to autonomous, time-invariant systems can be removed by the introduction of auxiliary state variables for the time argument as well as for time- and state-dependent expressions included in control inputs. Corresponding procedures, leading to an increase of the system dimension, were discussed exemplarily in [36] for the integer-order case.

models

$$\dot{\mathbf{x}}(t) = \boldsymbol{\mathcal{A}}\left(\mathbf{x}(t)\right) \cdot \mathbf{x}(t) \ , \quad \boldsymbol{\mathcal{A}}\left(\mathbf{x}(t)\right) \in \mathbb{R}^{n \times n} \ , \tag{2}$$

are specified with the vector of initial conditions

$$\mathbf{x}(0) \in [\mathbf{x}]\,(0) \ . \tag{3}$$

Analogously, a commensurate-order set of fractional-order differential equations of Caputo type [27, 30] is defined by

$$\mathbf{x}^{(\nu)}(t) = \boldsymbol{\mathcal{A}}\left(\mathbf{x}(t)\right) \cdot \mathbf{x}(t) \quad \textit{with} \quad 0 < \nu < 1 \ , \tag{4}$$

where initial conditions $\mathbf{x}(0)$ *are defined according to (3).*

For both system models in Def. 1, the initial state vector $\mathbf{x}(0)$ is assumed to be described by the interval representation $[\mathbf{x}]\,(0) = [\underline{\mathbf{x}}(0) \ ; \ \overline{\mathbf{x}}(0)]$, where the inequalities $\underline{x}_i(0) \leq \overline{x}_i(0)$ hold element-wise for each vector component $i \in \{1, \ldots, n\}$.

The existence of a solution to the problem specified according to (4) with the initial conditions (3) is ensured if either of the iteration procedures in Secs. 3 or 4 converges to an appropriate interval enclosure.

Definition 2 (Diagonally dominant model)**.** *Diagonally dominant quasi-linear system models are given by the state-space representations*

$$\begin{aligned} \dot{\mathbf{z}}(t) = \mathbf{f}\left(\mathbf{z}(t)\right) &= \left(\mathbf{T}^{-1} \cdot \boldsymbol{\mathcal{A}}\left(\mathbf{T} \cdot \mathbf{z}(t)\right) \cdot \mathbf{T}\right) \cdot \mathbf{z}(t) \\ &= \mathbf{A}\left(\mathbf{z}(t)\right) \cdot \mathbf{z}(t) \ , \quad \mathbf{A}\left(\mathbf{z}(t)\right) \in \mathbb{R}^{n \times n} \ , \end{aligned} \tag{5}$$

and

$$\mathbf{z}^{(\nu)}(t) = \mathbf{f}\left(\mathbf{z}(t)\right) = \mathbf{A}\left(\mathbf{z}(t)\right) \cdot \mathbf{z}(t) \tag{6}$$

after a suitable similarity transformation

$$\mathbf{x}(t) = \mathbf{T} \cdot \mathbf{z}(t) \ , \quad \mathbf{T} \in \mathbb{R}^{n \times n} \ , \quad \mathbf{z}(t) \in \mathbb{R}^{n} \tag{7}$$

of the systems in Def. 1.

Remark 1. In this paper, we restrict ourselves to the case of real-valued similarity transformations in (7). These transformations lead to the real-valued initial state enclosures

$$\mathbf{z}(0) \in \mathbf{T}^{-1} \cdot [\mathbf{x}]\,(0) \tag{8}$$

for both integer-order and fractional-order system models. As shown in [33, 36] for integer-order system models, also complex-valued similarity transformations are possible. They are advantageous for the case of systems with conjugate-complex eigenvalues and, hence, oscillatory dynamics. For both the real- and complex-valued case with system models having an eigenvalue multiplicity of one, the transformation matrix $\mathbf{T}$ is composed of the eigenvectors of $\boldsymbol{\mathcal{A}}\left(\mathbf{x}_{\mathrm{m}}\right)$, computed at the interval midpoint $\mathbf{x}_{\mathrm{m}} = \frac{1}{2} \cdot \left(\underline{\mathbf{x}}(0) + \overline{\mathbf{x}}(0)\right)$. For generalizations to higher multiplicities, which were so far only investigated for integer-order scenarios, see [36].

Remark 2. Where necessary for a compact notation of the iteration formulas derived in the following sections, it is further assumed that a translation of the state vector has been performed prior to solving the considered simulation task so that the trajectories of the systems under consideration converge to the origin of the state space if the dynamics are asymptotically stable.

Example 1. Fractional-order differential equations appear, as stated in the introduction of this paper, in a variety of engineering applications. For example, series connections of electric subcircuits containing resistors and capacitors can be used for modeling the dynamics of Lithium-Ion batteries. The corresponding impedance (as the quotient between terminal voltage and current) then takes the form of the integer-order (IO) frequency response

$$Z_{\mathrm{IO}}(\jmath\omega) = \frac{\sum\limits_{i=0}^{n} b_i \cdot (\jmath\omega)^i}{\sum\limits_{i=0}^{n} a_i \cdot (\jmath\omega)^i} \tag{9}$$

with the imaginary unit $\jmath$ and the angular frequency $\omega \geq 0$. However, experimental impedance spectroscopy data gives rise to the more general fractional-order (FO) expression, see [2, 38, 39],

$$Z_{\mathrm{FO}}(\jmath\omega) = \frac{\sum\limits_{i=0}^{n} b_i \cdot (\jmath\omega)^{\nu_i}}{\sum\limits_{j=0}^{n} a_j \cdot (\jmath\omega)^{\nu_j}} \ , \tag{10}$$

where ν_i and ν_j are non-negative, not necessarily integer-valued parameters with $0 \leq \nu_0 < \nu_1 < \nu_2 < \ldots$.

Here, numerator expressions of order ν_i are related to fractional derivatives of the terminal current, while the orders ν_j in the denominator are connected with a non-integer derivative of the terminal voltage. Due to the fact that a repeated fractional-order differentiation, first of order ν_a and second of order order ν_b corresponds in total with a derivative of order $\nu_a + \nu_b$, see [27, 30], the type of system model mentioned in this example, can always be transferred into a commensurate-order state-space representation according to Defs. 1 and 2 by setting ν to the greatest common divisor of all fractional orders ν_i and ν_j.

3 State-of-the-Art Techniques Applicable to the Verified Simulation of Fractional-Order System Models

3.1 Exploitation of Differential Inclusions and Cooperativity

Theorem 1 (Differential inclusions for fractional-order differential equations)**.** *Time-varying bounds for a fractional-order system described according to Def. 2*

are given by the interval vector

$$\mathbf{z}(t) \in [\mathbf{v}(t)\ ;\ \mathbf{w}(t)] \tag{11}$$

in which the individual components of the vectors $\mathbf{v}(t)$ *and* $\mathbf{w}(t)$ *are solutions to the coupled lower and upper bounding systems*

$$\mathbf{v}^{(\nu)}(t) = \mathbf{f}_v\left(\mathbf{v}(t), \mathbf{w}(t)\right) \leq \mathbf{z}^{(\nu)}(t) = \mathbf{f}\left(\mathbf{z}(t)\right) \leq \mathbf{w}^{(\nu)}(t) = \mathbf{f}_w\left(\mathbf{v}(t), \mathbf{w}(t)\right) \tag{12}$$

representing differential inclusions for the dynamic system $\mathbf{z}^{(\nu)}(t) = \mathbf{f}\left(\mathbf{z}(t)\right)$.

Proof. Theorem 1 is a straightforward consequence of Müller's theorem originally published for integer-order ordinary differential equations [25]. Substituting the integer-order derivatives in this theorem by their respective fractional-order counterparts completes the proof. □

Corollary 1 (Differential inclusions for cooperative fractional-order differential equations)**.** *Time-varying bounds for cooperative, positive fractional-order systems described according to Def. 2 are given by the interval vector*

$$\mathbf{z}(t) \in [\mathbf{v}(t)\ ;\ \mathbf{w}(t)]\ \ ,\quad v_i(t) \geq 0\ ,\quad i \in \{1, \ldots, n\}\ \ , \tag{13}$$

in which the individual components of the vectors $\mathbf{v}(t)$ *and* $\mathbf{w}(t)$ *are solutions to the mutually decoupled lower and upper bounding systems*

$$\mathbf{v}^{(\nu)}(t) = \mathbf{f}\left(\mathbf{v}(t)\right) \leq \mathbf{z}^{(\nu)}(t) = \mathbf{f}\left(\mathbf{z}(t)\right) \leq \mathbf{w}^{(\nu)}(t) = \mathbf{f}\left(\mathbf{w}(t)\right)\ \ . \tag{14}$$

Proof. Assume a cooperative dynamic system with strictly non-negative states $z_i(t) \geq 0$ satisfying the sufficient criterion for cooperativity [8, 17, 32, 37] given by

$$J_{i,j}\left(\mathbf{z}\right) \geq 0 \quad \text{for all} \quad i \neq j\ ,\quad i,j \in \{1, \ldots, n\} \quad \text{with} \quad \mathbf{J} = \frac{\partial \mathbf{f}\left(\mathbf{z}\right)}{\partial \mathbf{z}}\ \ . \tag{15}$$

An element-wise minimization (respectively, maximization) of the function $\mathbf{f}\left(\mathbf{z}(t)\right)$ over the state interval (13) directly leads to its element-wise defined lower bound $\mathbf{f}\left(\mathbf{v}(t)\right)$ (respectively, upper bound $\mathbf{f}\left(\mathbf{w}(t)\right)$). □

The property exploited in Corollary 1 is widely employed in the frame of observer design for both, integer-order and fractional-order system models. Suitable references concerning observer design as well as for its dual task, namely, cooperativity-preserving control synthesis can be found in [8, 15, 20, 32, 34]. If cooperativity is either directly given after first-principle modeling of the systems in Def. 1 or Def. 2, Corollary 1 provides overestimation-free state bounds if the element-wise minimizations and maximizations mentioned in the proof above coincide with actually reachable operating conditions. It has to be noted that in this case it is not necessary (from a practical point of view) to apply interval-based simulation routines as long as temporal discretization errors in the differential equations for $\mathbf{v}(t)$ and $\mathbf{w}(t)$ are

negligibly small. For several practically relevant system models, such as the interval observer design for a fractional-order battery model in [15], cooperativity can be ensured by design. Alternatively, a cooperativity-enforcing change of variables can be performed to remove the restrictive assumptions imposed by cooperativity if a dynamic system model is initially not cooperative. Details about suitable transformation techniques are given in [18, 19, 21]. However, if cooperativity is either not given directly or cannot be achieved by these similarity transformations, the following alternatives need to be exploited to determine guaranteed bounds for all reachable states.

3.2 Transformation of Fractional Systems into Equivalent Ordinary Differential Equations

Theorem 2 ([6] Solution of fractional-order differential equations by nonlinear time transformations)**.** *Let* $\mathbf{f}(\mathbf{z}(t))$ *be a bounded and continuous function. The solution to the time-invariant fractional-order differential equations considered in Def. 2 according to*

$$\mathbf{z}^{(\nu)}(t) = \mathbf{f}(\mathbf{z}(t)) \tag{16}$$

with the bounded initial conditions $\mathbf{z}(0)$ *is given by*

$$\mathbf{z}(t) = \boldsymbol{\mathfrak{z}}\left(\frac{t^{\nu}}{\Gamma(\nu+1)}\right) \quad , \tag{17}$$

where $\boldsymbol{\mathfrak{z}}(\tau)$ *is determined as the solution to an initial value problem to the set of integer-order differential equations*

$$\frac{\mathrm{d}\boldsymbol{\mathfrak{z}}(\tau)}{\mathrm{d}\tau} = \mathbf{\mathfrak{f}}(\boldsymbol{\mathfrak{z}}(\tau)) \tag{18}$$

with the initial condition

$$\boldsymbol{\mathfrak{z}}(0) = \mathbf{z}(0) \tag{19}$$

and the nonlinear time transformation

$$\tau = t - \left(t^{\nu} - \tau \cdot \Gamma(\nu+1)\right)^{\frac{1}{\nu}} \tag{20}$$

leading to

$$\mathbf{\mathfrak{f}}(\boldsymbol{\mathfrak{z}}(\tau)) = \mathbf{f}\left(\mathbf{z}\left(t - \left(t^{\nu} - \tau \cdot \Gamma(\nu+1)\right)^{\frac{1}{\nu}}\right)\right) \quad , \tag{21}$$

in which τ *is the independent variable and* t *is considered as a parameter.*

Although Theorem 2 provides a quite general approach that makes initial value problem solvers originally developed for the case of integer-order differential equations applicable to the fractional-order case, it has two main drawbacks if uncertain systems are concerned: First, the nonlinear time transformation according to Theorem 2 leads to the fact that even for time-invariant system models, usually time-varying initial value problems need to be solved. In the general case, this

can only be done by augmenting the state vector by the time variable τ according to the procedure discussed in [36], leading inevitably to an increase in the system dimension. Second, this augmentation of the state vector as well as the required backward transformation (17) of the computed solution usually introduce some additional amount of overestimation due to multiple dependencies on common interval variables.

Remark 3. The time-varying characteristics of the transformed system model in (21) with (20) highlights the property of fractional-order differential equations, that restarting the temporal solution procedure at some point of time $T > 0$ purely on the basis of the novel initial conditions $\mathbf{z}(T)$ with simultaneously resetting the time to zero would inevitably lead to truncation errors. Handling of these errors by means of guaranteed error bounds on the derivative operator is discussed further in Secs. 5 and 6 of this paper.

3.3 A Picard Iteration Procedure for Fractional-Order Dynamics

Theorem 3 ([1, 22] Integral formulation of Picard iterations for fractional-order differential equations)**.** *Let* $\mathbf{f}\left(\mathbf{z}(t)\right)$ *be a continuous Lipschitzian function on a bounded state and time domain. The solution to the time-invariant fractional-order differential equations considered in Def. 2 at the point of time* $T > 0$ *can be computed iteratively according to the fixed-point iteration*

$$\mathbf{z}^{\langle\kappa+1\rangle}(T) := \mathbf{z}(0) + \frac{1}{\Gamma(\nu)} \cdot \int_0^T (T-s)^{\nu-1} \cdot \mathbf{f}\left(\mathbf{z}^{\langle\kappa\rangle}(s)\right) \mathrm{d}s \ , \quad \kappa \in \mathbb{N}_0 \ , \tag{22}$$

with the initialization $\mathbf{z}^{\langle 0\rangle} := \mathbf{z}(0)$ *at the iteration step* $\kappa = 0$.

This iteration generalizes to interval bounded initial conditions $\mathbf{z}(0) \in [\mathbf{z}]\,(0) = [\mathbf{z}_0]$ *according to*

$$[\mathbf{z}]^{\langle\kappa+1\rangle} := [\mathbf{z}_0] + \frac{1}{\Gamma(\nu+1)} \cdot [0\ ;\ T^\nu] \cdot \mathbf{f}\left([\mathbf{z}]^{\langle\kappa\rangle}\right) \ , \quad \kappa \in \mathbb{N}_0 \ , \tag{23}$$

where convergence requires $[\mathbf{z}]^{\langle\kappa+1\rangle} \subseteq [\mathbf{z}]^{\langle\kappa\rangle}$*, leading to* $\mathbf{z}(t) \in [\mathbf{z}]^{\langle\kappa+1\rangle}$ *for all* $t \in [0\ ;\ T]$.

Theorem 3 provides the possibility to determine *time-invariant* bounds $[\mathbf{z}]^{\langle\kappa+1\rangle}$ containing all possible states $\mathbf{z}(t)$ that are reachable over the complete time interval $t \in [0\ ;\ T]$. However, the fact that these bounds are time-invariant makes them excessively wide at the single point $t = T$. Hence, generalizations of this iteration are derived in the following section to obtain *time-varying* bounds which — for asymptotically stable dynamics — contract temporally towards the system's equilibrium state.

4 Interval-Based Iteration Procedure: Generalization of Exponential State Enclosures to Fractional-Order Systems

In this section, an interval-based iteration procedure is derived for the computation of guaranteed state enclosures for fractional-order system models. To make this paper self-contained, an already existing variant for integer-order models as well as the initial work [33, 35] for the fractional-order case are briefly reviewed, before a detailed discussion about specific extensions to the fractional-order case is provided.

4.1 Exponential State Enclosures for Integer-Order Ordinary Differential Equations

Definition 3 (Exponential state enclosure)**.** *The time-dependent exponential enclosure function*

$$\mathbf{z}^*(t) \in [\mathbf{z}_e](t) := \exp([\mathbf{\Lambda}] \cdot t) \cdot [\mathbf{z}_e](0) \ , \ \ [\mathbf{z}_e](0) = [\mathbf{z}_0] \tag{24}$$

with the parameter matrix

$$[\mathbf{\Lambda}] := \operatorname{diag}\{[\lambda_i]\} \ , \ i \in \{1, \ldots, n\} \ , \tag{25}$$

is denoted as a verified exponential state enclosure *for the system model (5) with (8) if it is determined according to Theorem 4.*

Theorem 4 ([36] Iteration for exponential state enclosures)**.** *The exponential state enclosure (24) is guaranteed to contain the set of all reachable states* $\mathbf{z}^*(T)$ *at the point of time* $t = T > 0$ *according to*

$$\mathbf{z}^*(T) \in [\mathbf{z}_e](T) := \exp([\mathbf{\Lambda}] \cdot T) \cdot [\mathbf{z}_e](0) \ \ , \tag{26}$$

if $[\mathbf{\Lambda}]$ *is set to the outcome of the converging iteration*

$$[\lambda_i]^{\langle \kappa+1 \rangle} := \frac{f_i\left(\exp\left([\mathbf{\Lambda}]^{\langle \kappa \rangle} \cdot [t]\right) \cdot [\mathbf{z}_e](0)\right)}{\exp\left([\lambda_i]^{\langle \kappa \rangle} \cdot [t]\right) \cdot [z_{e,i}](0)} \ , \tag{27}$$

$i \in \{1, \ldots, n\}$*, with the prediction horizon* $[t] = [0 \ ; \ T]$*.*

Proof. Assume that the integral form of the Picard iteration, see [36] and Theorem 3 with $\nu = 1$,

$$\mathbf{z}^*(t) \in [\mathbf{z}_e]^{\langle \kappa+1 \rangle} := [\mathbf{z}_e](0) + \int_0^t \mathbf{f}\left([\mathbf{z}_e]^{\langle \kappa \rangle}(s)\right) \mathrm{d}s \tag{28}$$

describes a converging iteration that encloses the exact solution $\mathbf{z}^*(t)$ to the initial value problem of an integer-order system as given in Def. 2 in terms of an outer

interval hull over all possible state trajectories over the time horizon $t \in [t] = [0\,;\,T]$ with $T > 0$.

The evaluation of the iteration (28) for the ansatz of an exponential state enclosure (24) with (25) according to Def. 3 yields the relation

$$\begin{aligned}\mathbf{z}^*(t) &\in \exp\Big([\mathbf{\Lambda}]^{\langle\kappa+1\rangle}\cdot t\Big)\cdot[\mathbf{z}_e](0)\\ &= [\mathbf{z}_e](0) + \int_0^t \mathbf{f}\left(\exp\Big([\mathbf{\Lambda}]^{\langle\kappa\rangle}\cdot s\Big)\cdot[\mathbf{z}_e](0)\right)\mathrm{d}s\end{aligned} \tag{29}$$

between the interval matrices $[\mathbf{\Lambda}]^{\langle\kappa\rangle}$ and $[\mathbf{\Lambda}]^{\langle\kappa+1\rangle}$ for the two subsequent iteration steps κ and $\kappa+1$. The differentiation of (29) with respect to time results in the *differential form of the Picard iteration* which is given by

$$\begin{aligned}\dot{\mathbf{z}}^*(t) &\in [\mathbf{\Lambda}]^{\langle\kappa+1\rangle}\cdot\exp\Big([\mathbf{\Lambda}]^{\langle\kappa+1\rangle}\cdot t\Big)\cdot[\mathbf{z}_e](0)\\ &= \mathbf{f}\left(\exp\Big([\mathbf{\Lambda}]^{\langle\kappa\rangle}\cdot t\Big)\cdot[\mathbf{z}_e](0)\right) = \mathbf{f}\left([\mathbf{z}_e]^{\langle\kappa\rangle}(t)\right)\end{aligned} \tag{30}$$

with its corresponding interval extension for the complete prediction window $[t]$ according to

$$\dot{\mathbf{z}}^*([t]) \in [\mathbf{\Lambda}]^{\langle\kappa+1\rangle}\cdot[\mathbf{z}_e]^{\langle\kappa+1\rangle}([t]) = \mathbf{f}\left([\mathbf{z}_e]^{\langle\kappa\rangle}([t])\right)\,. \tag{31}$$

All equivalent expressions (29)–(31) describe a converging iteration process if

$$[\lambda_i]^{\langle\kappa+1\rangle} \subseteq [\lambda_i]^{\langle\kappa\rangle} \tag{32}$$

and hence

$$[\mathbf{\Lambda}]^{\langle\kappa+1\rangle} \subseteq [\mathbf{\Lambda}]^{\langle\kappa\rangle} \tag{33}$$

are satisfied. Due to inclusion monotonicity [16] of the exponential function, the relations (32) and (33) imply

$$\exp\Big([\mathbf{\Lambda}]^{\langle\kappa+1\rangle}\cdot t\Big) \subseteq \exp\Big([\mathbf{\Lambda}]^{\langle\kappa\rangle}\cdot t\Big) \tag{34}$$

for all $t \in [t]$. Overapproximating the left-hand side of (31), cf. [33, 36], in a conservative manner with an interval

$$[\lambda_i]^{\langle\kappa+1\rangle} \subseteq [\tilde{\lambda}_i]^{\langle\kappa+1\rangle} \subseteq [\lambda_i]^{\langle\kappa\rangle} \tag{35}$$

according to

$$\mathrm{diag}\left\{[\tilde{\lambda}_i]^{\langle\kappa+1\rangle}\right\}\cdot[\mathbf{z}_e]^{\langle\kappa\rangle}([t]) =: \mathbf{f}\left([\mathbf{z}_e]^{\langle\kappa\rangle}([t])\right) \tag{36}$$

and solving the equality in (36) for the yet unknown bounds $[\tilde{\lambda}_i]^{\langle\kappa+1\rangle}$ with subsequently renaming this parameter into $[\lambda_i]^{\langle\kappa+1\rangle}$ completes the proof. □

For further discussions concerning the necessary zero-exclusion requirement $0 \notin [z_{e,i}]([t])$ for all components $i \in \{1, \ldots, n\}$ of the state vector as well as for generalizations to multiple real and/or complex eigenvalues, the reader is referred to [36]. Fundamental step-size control strategies and the definition of time-varying transformation matrices leading to less conservative quasi-linear system models than those in Def. 2 are given in [19].

Corollary 2. *For quasi-linear state-space representations according to Def. 1, which are transformed into the diagonally dominant form of Def. 2, the component-wise notation*

$$\dot{z}_i(t) = f_i(\mathbf{z}(t)) = \sum_{j=1}^{n} a_{ij}(\mathbf{z}(t)) \cdot z_j(t) \tag{37}$$

of the state equations allows for a reduction of interval-related dependency problem, the wrapping effect, and the resulting computational effort if formula (27) is reformulated symbolically into

$$\begin{aligned}[\lambda_i]^{\langle\kappa+1\rangle} := a_{ii}\left([\mathbf{z}_e]^{\langle\kappa\rangle}([t])\right) \\ + \sum_{\substack{j=1 \\ j\neq i}}^{n} \left\{ a_{ij}\left([\mathbf{z}_e]^{\langle\kappa\rangle}([t])\right) \cdot e^{\left(\left([\lambda_j]^{\langle\kappa\rangle} - [\lambda_i]^{\langle\kappa\rangle}\right)\cdot[t]\right)} \cdot \frac{[z_{e,j}](0)}{[z_{e,i}](0)} \right\} \quad .\end{aligned} \tag{38}$$

4.2 Mittag-Leffler Type State Enclosures for Fractional-Order Differential Equations

The focus of this subsection is the generalization of the exponential enclosure technique to sets of commensurate fractional-order models. The fundamental iteration summarized in the following Theorem 5 was first published by the authors in [33] and [35]. The novelty of the present paper is the detailed description of relations to the state-of-the-art approaches in Sec. 3 and the in-depth discussion of interval-based numerical evaluation schemes together with the reliable consideration of the infinite-horizon memory property that becomes crucial as soon as the integration time horizon is divided into several temporal subslices of finite duration.

Definition 4 (Mittag-Leffler type state enclosure)**.** *The time-dependent Mittag-Leffler type enclosure function*

$$\mathbf{z}^*(t) \in \mathbf{E}_{\nu,1}([\mathbf{\Lambda}] \cdot t^{\nu}) \cdot [\mathbf{z}_e](0) \ , \ \ [\mathbf{z}_e](0) = [\mathbf{z}_0] \tag{39}$$

with the diagonal parameter matrix $[\mathbf{\Lambda}] := \operatorname{diag}\{[\lambda_i]\}$, $i \in \{1, \ldots, n\}$, *is denoted as a* verified Mittag-Leffler type state enclosure *for the system model (6) with (8) if it is determined according to Theorem 5.*

Theorem 5 ([33, 35] Iteration for Mittag-Leffler type enclosures)**.** *The Mittag-Leffler type state enclosure (39) is guaranteed to contain the set of all reachable states* $\mathbf{z}^*(T)$ *at the point of time* $t = T > 0$ *according to*

$$\mathbf{z}^*(T) \in \mathbf{E}_{\nu,1}([\mathbf{\Lambda}] \cdot T^{\nu}) \cdot [\mathbf{z}_e](0) \quad , \tag{40}$$

if $[\mathbf{\Lambda}]$ *is set to the outcome of the converging iteration*

$$[\lambda_i]^{\langle\kappa+1\rangle} := \frac{f_i\left(\mathbf{E}_{\nu,1}\left([\mathbf{\Lambda}]^{\langle\kappa\rangle}\cdot[t]^{\nu}\right)\cdot[\mathbf{z}_e]\,(0)\right)}{E_{\nu,1}\left([\lambda_i]^{\langle\kappa\rangle}\cdot[t]^{\nu}\right)\cdot[z_{e,i}]\,(0)} \quad , \tag{41}$$

$i \in \{1,\ldots,n\}$*, with the prediction horizon* $[t] = [0\,;\,T]$.

Proof. According to [7,11], the exact solution of a linear fractional-order differential equation

$$z^{(\nu)}(t) = \lambda\cdot z(t) \tag{42}$$

of Caputo type — for which only the initial conditions of the system states at $t = 0$ are specified — is given by the analytic expression

$$z(t) = E_{\nu,1}\left(\lambda t^{\nu}\right)\cdot z(0) \ . \tag{43}$$

As for the case of integer-order differential equations, this relation serves as an ansatz for describing verified state enclosures. By substituting it (cf. (30) in Theorem 5) into the *differential formulation of the Picard iteration*, which results from determining the fractional-order time derivative of the result in Theorem 3, the expression

$$\begin{aligned}\mathbf{z}^{(\nu)}(t) &\in \left([\mathbf{\Lambda}]^{\langle\kappa+1\rangle}\right)\cdot\mathbf{E}_{\nu,1}\left([\mathbf{\Lambda}]^{\langle\kappa+1\rangle}\cdot t^{\nu}\right)\cdot[\mathbf{z}_e]\,(0)\\ &= \mathbf{f}\left(\mathbf{E}_{\nu,1}\left([\mathbf{\Lambda}]^{\langle\kappa\rangle}\cdot t^{\nu}\right)\cdot[\mathbf{z}_e]\,(0)\right) =: \mathbf{f}\left([\mathbf{z}_e]^{\langle\kappa\rangle}\,([t])\right)\end{aligned} \tag{44}$$

is obtained.

Overapproximating the Mittag-Leffler type state enclosure $\mathbf{E}_{\nu,1}\left([\mathbf{\Lambda}]^{\langle\kappa+1\rangle}\cdot t^{\nu}\right)$ in the iteration step $\kappa+1$ by the enclosure $[\mathbf{z}_e]^{\langle\kappa\rangle}\,([t])$ obtained in the previous iteration step on the left-hand side of (44), as it was also done in the proof of Theorem 4 for the integer-order counterpart, leads to

$$\mathrm{diag}\left\{[\tilde{\lambda}_i]^{\langle\kappa+1\rangle}\right\}\cdot[\mathbf{z}_e]^{\langle\kappa\rangle}\,([t]) = \mathbf{f}\left([\mathbf{z}_e]^{\langle\kappa\rangle}\,([t])\right) \ . \tag{45}$$

Solving this expression for $[\tilde{\lambda}_i]^{\langle\kappa+1\rangle}$ with subsequently renaming this parameter into $[\lambda_i]^{\langle\kappa+1\rangle}$ completes the proof of Theorem 5. □

Corollary 3. *Quasi-linear state-space representations of fractional-order differential equations in diagonally dominant form according to Def. 2 can be simulated efficiently by the symbolically simplified iteration scheme*

$$\begin{aligned}[\lambda_i]^{\langle\kappa+1\rangle} &:= a_{ii}\left([\mathbf{z}_e]^{\langle\kappa\rangle}\,([t])\right)\\ &\quad+\sum_{\substack{j=1\\ j\neq i}}^{n}\left\{a_{ij}\left([\mathbf{z}_e]^{\langle\kappa\rangle}\,([t])\right)\cdot\frac{E_{\nu,1}\left([\lambda_j]^{\langle\kappa\rangle}\cdot[t]^{\nu}\right)}{E_{\nu,1}\left([\lambda_i]^{\langle\kappa\rangle}\cdot[t]^{\nu}\right)}\cdot\frac{[z_{e,j}]\,(0)}{[z_{e,i}]\,(0)}\right\} \ .\end{aligned} \tag{46}$$

Remark 4. In contrast to Eq. (38) derived for integer-order systems, where analytic simplifications of exponential functions become possible, the quotient of two Mittag-Leffler functions in (46) cannot be simplified further in the general case. This imposes further restrictions on the numerical evaluation of (46) using techniques from interval arithmetic in the following section.

Remark 5. For the order $\nu = 1$, the iteration formulas in Theorems 4 and 5 become identical due to $E_{1,1}(z) \equiv e^z$.

5 Evaluating Mittag-Leffler Functions for Interval Arguments

As described in [14], rough (outer) bounds for the evaluation of Mittag-Leffler functions with (real-valued) interval arguments can be determined by exploiting the property of a continuous interpolation between Gaussian (exponential) and Lorentzian (rational) functions according to

$$\exp\left(-\zeta\right) < E_{\nu,1}\left(-\zeta\right) \leq \frac{1}{1+\zeta}\ , \quad \zeta \geq 0 \tag{47}$$

in Fig. 1a and

$$\exp\left(-\zeta^2\right) < E_{\nu,1}\left(-\zeta^2\right) \leq \frac{1}{1+\zeta^2}\ , \quad \zeta \geq 0 \tag{48}$$

in Fig. 1b. However, if only subintervals from the range $\nu \in [0\ ;\ 1]$ are required for a specific application scenario, these bounds are usually too conservative for the interval-based evaluation of the iteration formulas presented in the previous section.

Therefore, floating point evaluations of the Mittag-Leffler function using the MATLAB implementation of R. Garrappa [10] are extended in the following subsections to obtain tight guaranteed interval bounds for the case of real-valued arguments. The case of complex interval arguments is a topic for future research.

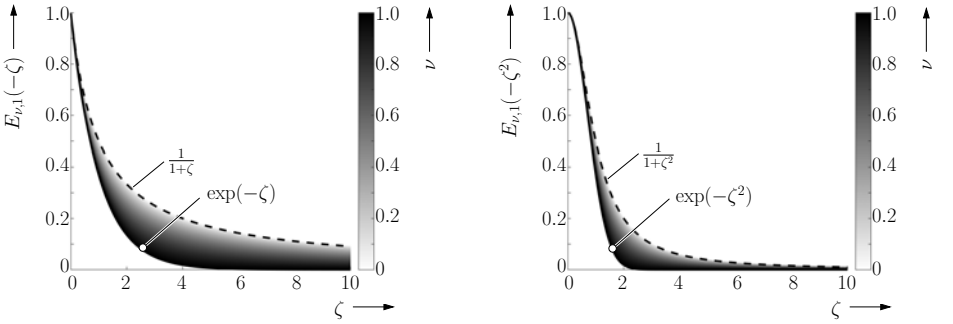
5.1 Interval Evaluation of the Two-Parameter Mittag-Leffler Function with Real Arguments

Theorem 6 ([35] Interval bounds for the Mittag-Leffler function with real arguments)**.** *Interval evaluations of Mittag-Leffler functions with real-valued arguments* $z \in [z] = [\underline{z}\ ;\ \overline{z}]$ *are given by*

$$E_{\nu,\beta}\left([z]\right) \in \left[E^*_{\nu,\beta}\left([z]\right)\right] = \left[\tilde{E}_{\nu,\beta}\left([z]\right)\right] + \frac{\epsilon}{1+\epsilon} \cdot \left(1 + \left|\left[\tilde{E}_{\nu,\beta}\left([z]\right)\right]\right|\right) \cdot [-1\ ;\ 1] \tag{49}$$

with the tolerance value ϵ *of a floating point function evaluation of (1) and the interval definition*

$$\left[\tilde{E}_{\nu,\beta}\left([z]\right)\right] = \left[\bigtriangledown\tilde{E}_{\nu,\beta}\left(\underline{z}\right)\ ;\ \bigtriangleup\tilde{E}_{\nu,\beta}\left(\overline{z}\right)\right]\ , \tag{50}$$



(a) Bounds for the Mittag-Leffer according to (47).

(b) Bounds for the Mittag-Leffer according to (48).

Figure 1: Guaranteed bounds for Mittag-Leffler functions for the parameter range $\nu \in [0\,;\,1]$, where the Lorentzian upper bound is highlighted by the dashed line and the gray color code visualizes the dependence on ν.

where $\bigtriangledown$ and $\bigtriangleup$ denote switchings of the rounding mode of a CPU towards minus and plus infinity, respectively, in the corresponding floating point evaluations.

Proof. The interval extension (50) directly results form the strict monotonicity of the two-parameter Mittag-Leffler function with real-valued arguments. Moreover, a guaranteed tolerance value ϵ ($\epsilon \approx 10^{-15}$ for the case of the implementation by R. Garrappa [10]) allows to express the relative deviation between the floating point approximation $\tilde{E}_{\nu,\beta}(z)$ and the exact function value $E_{\nu,\beta}(z)$ at some value $z \in \mathbb{R}$, representable in floating point arithmetic, according to

$$\frac{\left|E_{\nu,\beta}(z) - \tilde{E}_{\nu,\beta}(z)\right|}{1 + |E_{\nu,\beta}(z)|} = \frac{|\Delta|}{1 + \left|\tilde{E}_{\nu,\beta}(z) + \Delta\right|} \leq \epsilon \ . \tag{51}$$

Solving the inequality (51) for $|\Delta|$ relies on the fact that

$$\frac{|\Delta|}{1 + \left|\tilde{E}_{\nu,\beta}(z) + \Delta\right|} \leq \frac{|\Delta|}{1 + \left|\tilde{E}_{\nu,\beta}(z)\right| - |\Delta|} \tag{52}$$

holds. Assuming

$$\frac{|\Delta|}{1 + \left|\tilde{E}_{\nu,\beta}(z)\right| - |\Delta|} \leq \epsilon \tag{53}$$

in correspondence with (51) leads to the inequality

$$|\Delta| \leq \frac{\epsilon}{1+\epsilon} \cdot \left(1 + \left|\tilde{E}_{\nu,\beta}(z)\right|\right) \tag{54}$$

which characterizes the interval bounds $[-1\,;\,1]\cdot|\Delta|$ of the worst-case approximation error. Adding this tolerance interval to the outward rounded point-valued evaluation of the Mittag-Leffler function in Eq. (50) completes the proof of Theorem 6. □

5.2 Exploitation of Monotonicity in Interval Evaluations of the Mittag-Leffler Function

To reduce overestimation in the interval evaluation of the iteration procedure according to Theorem 5, monotonicity properties of the Mittag-Leffler function with respect to the time t, the solution parameter λ as well as to a derivative order ν specified as an interval variable are investigated in this section.

Theorem 7 ([35] Monotonicity-based interval bounds for the Mittag-Leffler function)**.** *The range of function values for the Mittag-Leffler function with the uncertain real-valued parameters* $\nu \in [\underline{\nu}\,;\,\overline{\nu}]$, $0 < \underline{\nu} \leq 1$, $0 < \overline{\nu} \leq 1$ *and* $\lambda \in \left[\underline{\lambda}\,;\,\overline{\lambda}\right]$, $\overline{\lambda} < 0$ *and the non-negative time argument* $t \in \left[\underline{t}\,;\,\overline{t}\right]$, $\underline{t} \geq 0$, *can be bounded tightly according to the interval enclosure*

$$E_{\nu,1}\left(\lambda t^{\nu}\right) \in \left[E^{*}_{\overline{\nu},1}\left(\inf\left([\boldsymbol{\mathcal{X}}]\right)\right)\,;\,E^{*}_{\underline{\nu},1}\left(\sup\left([\boldsymbol{\mathcal{X}}]\right)\right)\right] \tag{55}$$

with $[\boldsymbol{\mathcal{X}}] := [\lambda]\cdot[t]^{[\nu]}$, *where* $\sup\left([\boldsymbol{\mathcal{X}}]\right) \leq 0$ *holds.*

If monotonicity with respect to ν *can be proven additionally, the relation simplifies to*

$$E_{\nu,1}\left(\lambda t^{\nu}\right) \in \left[E^{*}_{\overline{\nu},1}\left(\underline{\lambda}\cdot\overline{t}^{\overline{\nu}}\right)\,;\,E^{*}_{\underline{\nu},1}\left(\overline{\lambda}\cdot\underline{t}^{\underline{\nu}}\right)\right] \tag{56}$$

for the monotonically decreasing branch in ν *and to*

$$E_{\nu,1}\left(\lambda t^{\nu}\right) \in \left[E^{*}_{\underline{\nu},1}\left(\underline{\lambda}\cdot\overline{t}^{\underline{\nu}}\right)\,;\,E^{*}_{\overline{\nu},1}\left(\overline{\lambda}\cdot\underline{t}^{\overline{\nu}}\right)\right] \tag{57}$$

for the increasing branch; the change of monotonicity occurs on the surface depicted in Fig. 2.

Proof. Formula (55) is a direct consequence of the continuous interpolation property of Mittag-Leffler functions between Gaussian and Lorentzian functions, see the beginning of Sec. 5 and [12, 24]. For a proof of *monotonicity with respect to* t, λ, *and* ν, see [35]. □

Remark 6. For intervals $[\nu]$, $[\lambda]$, and $[t]$ which do not intersect with the surface depicted in Fig. 2, the Eq. (56) holds for ν values below the surface (monotonically decreasing) and (57) for values above (monotonically increasing); otherwise (55) must be applied.

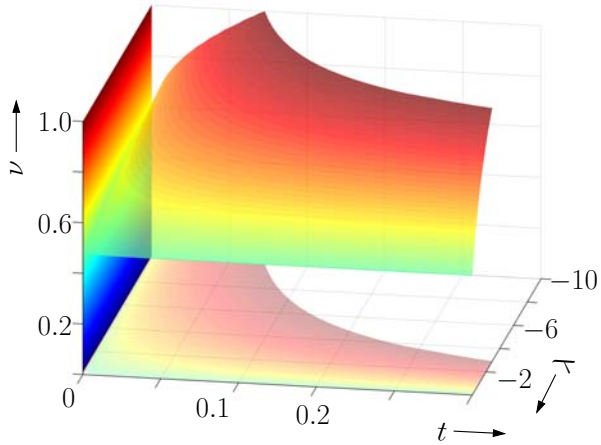


Figure 2: Surface, where the Mittag-Leffler function $E_{\nu,1}\left(\lambda t^{\nu}\right)$ changes its monotonicity with respect to ν.

5.3 Interval Bounds for Temporal Truncation Errors due to the Infinite Memory Property of Fractional-Order Systems

It is well known that fractional-order system models are characterized by an infinite memory of previous states [27,30]. Hence, restarting a simulation at some point of time $t = t_{k+1}$ on the basis of state information $\mathbf{z}(t_{k+1})$ computed by a simulation that was originally initiated at some point $t_k < t_{k+1}$ does not only have to account for these new initial conditions[3]. It also needs to consider the effect of temporal truncation errors which can be expressed by component-wise error bounds resulting from the fact that a fractional-order derivative of order ν with a memory start at $t = t_k$ is replaced with a new starting point $t = t_{k+1}$. The corresponding derivative operators are subsequently denoted by ${}_{t_k}\mathcal{D}_t^{\nu}\mathbf{z}(t)$ and ${}_{t_{k+1}}\mathcal{D}_t^{\nu}\mathbf{z}(t)$, respectively.

Theorem 8 (Bounds for temporal truncation errors)**.** *Resetting the initial point of time of the integration of fractional-order models defined in Def. 2 based on Theorem 5 after completion of a time interval of length T requires the inflation of the right-hand side of the state equations by the symmetric interval $[-\boldsymbol{\mu}\,;\,\boldsymbol{\mu}]$ at the point T with*

$$\boldsymbol{\mu} := \frac{\boldsymbol{\mathcal{Z}} \cdot (t_k + T)^{-\nu}}{|\Gamma(1-\nu)|} \tag{58}$$

[3]For the sake of compactness, the notation in this subsection is based on Def. 2. A transfer towards Def. 1 solely requires to replace all occurrences of the vectors $\mathbf{z}(t)$ with their counterpart $\mathbf{x}(t)$.

and the component-wise defined supremum of the set of reachable states

$$\mathcal{Z}_i = \sup_{t\in[t_0\ ;\ t_{k+1}]} |z_i(t)| \quad . \tag{59}$$

The re-initialized initial value problem is then given by

$${}_{t_k+T}\mathcal{D}_t^{\nu}\mathbf{z}(t) = \mathbf{z}^{(\nu)}(t) = \mathbf{f}\left(\mathbf{z}(t)\right) + [-\boldsymbol{\mu}\ ;\ \boldsymbol{\mu}] =: \tilde{\mathbf{f}}\left(\mathbf{z}(t)\right) \tag{60}$$

with the initial state enclosure $\mathbf{z}(t_k+T) \in [\mathbf{z}]\,(t_k+T)$ *resulting form the solution of*

$${}_{t_0}\mathcal{D}_t^{\nu}\mathbf{z}(t) = \mathbf{z}^{(\nu)}(t) = \mathbf{f}\left(\mathbf{z}(t)\right) \quad \textit{with} \quad \mathbf{z}(t_0) \in [\mathbf{z}]\,(t_0)\ , \quad t_0 = 0\ . \tag{61}$$

Proof. Theorem 8 is a consequence of the component-wise defined error bounds for a general fractional derivative operator of a commensurate system model on the time interval $t_k + T \le t \le t_{k+1}$ that can be computed according to [30] by

$$|{}_{t_k}\mathcal{D}_t^{\nu}\mathbf{z}(t) - {}_{t_k+T}\mathcal{D}_t^{\nu}\mathbf{z}(t)| \le \frac{\boldsymbol{\mathcal{Z}}T^{-\nu}}{|\Gamma(1-\nu)|} =: \boldsymbol{\mu} \tag{62}$$

As presented in [30], Eq. (62) relies on the component-wise defined supremum (59) of the reachable states denoted by the vector $\boldsymbol{\mathcal{Z}}$. □

A visualization of this state resetting procedure, with a corresponding adjustment of the right-hand side of the set of state equations is given in the following section. The following section accounts both for linear and nonlinear system models, as well as for a first possibility to interface the bounding approach according to Eq. (62) with a contractor technique [16] applied to the solution parameters $[\lambda_i]$ that for some system models yields tighter bounds than those given purely by applying the iteration of Theorem 5 after inflating the right-hand sides of the state equation with the vector $\boldsymbol{\mu}$. The reason for these possible enhancements can be seen in the fact that the original bound $\boldsymbol{\mu}$ captures an infinitely long time window starting at $t = t_k$, while in many practical scenarios much shorter windows are sufficient for the reliable forecast of the set of all possible state trajectories.

Theorem 9 (Contractor for the state enclosure of fractional-order systems)**.** *Assume that a reference solution* $\mathbf{z}(t) \in [\mathbf{z}_{\mathrm{ref}}]\,(t)$ *has already been computed for the initial value problem with the initial point of time* $t = t_k$ *that is valid up to the point* $t = t^* > t_k + T$ *and that the application of Theorem 5 to the re-initialized initial value problem (60) in Theorem 8 with the initial point of time* $t = t_k + T$ *has provided the interval bounds* $[\mathbf{z}_e]\,(t) = \exp\left([\boldsymbol{\Lambda}]\cdot(t-(t_k+T))\right)\cdot[\mathbf{z}_e]\,(t_k+T)$ *that are also valid up to* $t = t^*$ *with the associated solution parameters* $[\lambda_i]$, $i \in \{1,\ldots,n\}$, *a contractor is given by*

$$[\lambda_i] := [\lambda_i] \cap [\tilde{\lambda}_i] \tag{63}$$

with

$$[\tilde{\lambda}_i] = \frac{\tilde{f}_i\left([\mathbf{z}_e]\,([t_k+T\ ;\ t^*])\right) \cap \tilde{f}_i\left([\mathbf{z}_{\mathrm{ref}}]\,([t_k+T\ ;\ t^*])\right)}{[z_{e,i}]\,([t_k+T\ ;\ t^*]) \cap [z_{i,\mathrm{ref}}]\,([t_k+T\ ;\ t^*])} \quad . \tag{64}$$

Proof. The validity of Theorem 9 is a direct consequence of the fact that both $[\mathbf{z}_{\mathrm{ref}}](t)$ and $[\mathbf{z}_e](t)$ are verified state enclosures according to $\mathbf{z}(t) \in [\mathbf{z}_{\mathrm{ref}}](t)$ and $\mathbf{z}(t) \in [\mathbf{z}_e](t)$ and, thus, have to satisfy the fractional-order differential equation in the componentwise notation (44). Intersecting the evaluation of (44) for both state enclosures after consideration of the error bounds $\boldsymbol{\mu}$ for the corresponding point of restart and solving the result for the interval of the solution parameter $[\tilde{\lambda}_i]$ yields the relation (64). □

6 Illustrating Examples

6.1 Visualization of Interval Bounds for Temporal Truncation Errors of Linear Fractional-Order Systems

To visualize the influence of temporal truncation errors on the solution quality, consider the Caputo type linear fractional-order differential equation

$$z^{(\nu)}(t) = -z(t) \tag{65}$$

with the differentiation order $\nu = 0.5$ as well as the initial state $z(0) = 1$. According to (42) and (43) its *exact* solution is given by

$$z(t) = E_{0.5,1}\left(-t^{0.5}\right) \quad \text{for} \quad t \geq 0 \ . \tag{66}$$

This solution is visualized in Fig. 3.

In addition, assume that the overall integration time horizon $t \in [0\ ;\ T_{\mathrm{f}}]$ with $T_{\mathrm{f}} = 10$ for this system is split into N equidistant slices $[\tau_k] := [t_{k-1}\ ;\ t_k]$ with $t_0 = 0$ and $t_k = k \cdot \frac{T_{\mathrm{f}}}{N}$, $k \in \{1, \ldots, N\}$. Neglecting the infinite-time horizon memory of this system, approximate solutions $\tilde{z}(t)$ are computed recursively by means of

$$\tilde{z}(t) = \left(E_{0.5,1}\left(-\left(\frac{T_{\mathrm{f}}}{N}\right)^{0.5}\right)\right)^{k-1} \cdot E_{0.5,1}\left(-\left(t - t_{k-1}\right)^{0.5}\right) \ , \tag{67}$$

where $\tilde{z}(0) = z(0)$ and $t \in [\tau_k]$. As shown in Fig. 3, the quality of these approximations becomes worse, the larger the value N is chosen.

To quantify the effect of the infinite-horizon memory, the error quantification according to Sec. 5.3 is included in the iteration scheme for determining the parameter $[\lambda]$ according to Theorem 5. It becomes obvious that the interval enclosures included in Fig. 4 for the time steps $k \geq 2$ contain the exact solution to the initial value problem. Note, the computation of the parameter enclosures $[\lambda]$ according to Theorem 5 was interfaced with intersecting the iteration result with a further conservative expression obtained by

$$[\tilde{\lambda}] := \frac{-\left[z_e\right]\left([\tau_k]\right) + [-1\ ;\ 1] \cdot \mu(t_{k-1})}{\left(E_{0.5,1}\left(-\left[\tau_k\right]^{0.5}\right) \cdot [1-\eta\ ;\ 1+\eta]\right) \cap \left[z_e\right]\left([\tau_k]\right)} \ , \quad 0 < \eta < 1 \ , \tag{68}$$

as a special case of Theorem 9, where the numerator directly results from computing the state enclosure as shown in (41) and the denominator includes some a-priori knowledge on the domain of reachable states in the time interval $[\tau_k]$.

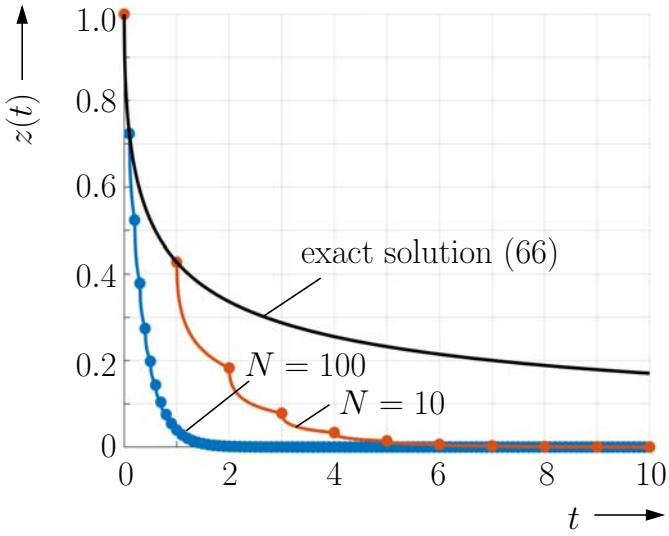


Figure 3: Visualization of truncation errors resulting from the infinite-horizon memory effect of fractional-order systems.

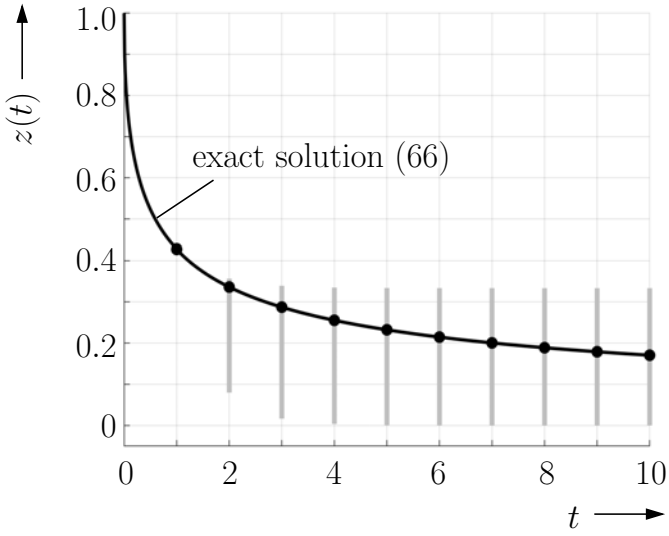


Figure 4: Interval-based quantification of truncation errors resulting from the infinite-horizon memory effect of fractional-order systems with $\eta = 0.1$.

6.2 A Nonlinear Example: Interval Bounds for Different Integration Horizons

As a second, nonlinear example for the application of Theorem 5, the state equation

$$z^{(\nu)}(t) = p \cdot z^3(t) = p \cdot a(z(t)) \cdot z(t) \tag{69}$$

with an uncertain initial state $z(0) \in [z]\,(0)$, the interval parameter $p \in [p]$, and the uncertain differentiation order $\nu \in [\nu]$ is considered. Note that Eq. (69) already includes the reformulation into a quasi-linear system model so that the modified iteration formula (46) becomes applicable.

In the following, two cases differing in the amount of uncertainty according to

Case a: $[z]\,(0) = [0.99\ ;\ 1.0]$, $[p] = [-2\ ;\ -1.99]$, $[\nu] = [0.8\ ;\ 0.81]$

and

Case b: $[z]\,(0) = [0.5\ ;\ 1.0]$, $[p] = [-2\ ;\ -1]$, $[\nu] = [0.8\ ;\ 0.9]$

are distinguished for this example.

Using the integration time horizons $T \in \{0.25, 0.50, 1.0\}$ for both **Case a** and **Case b**, without restarting the integration at any point in the interior of the temporal window $t \in [0\ ;\ T]$, the state enclosures in Figs. 5a and 5b are obtained. It can be noticed that for both cases the iteration describes non-diverging state enclosures despite the fact that constant parameter bounds $[\lambda]$ were determined for the *complete* integration time horizons. Both, for small uncertainty levels in Fig. 5a and large uncertainty in Fig. 5b, these computed interval bounds become wider for increasing lengths of the integration horizon, due to the fact that solutions close to the steady state need to be incorporated. This statement can also be verified by investigating the numerical parameter values produced by Theorem 5 (listed in ascending order of T) for

Case a: $[\lambda] \in \{[-2.0001\ ;\ -0.5261], [-2.0001\ ;\ -0.2276], [-2.0001\ ;\ -0.0684]\}$

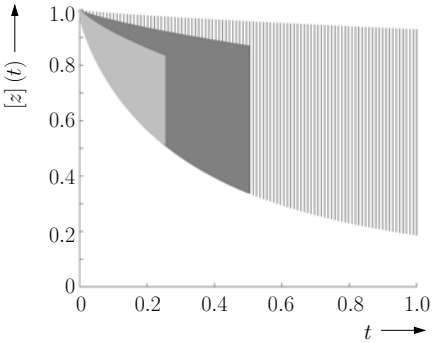
as well as for

Case b: $[\lambda] \in \{[-2.0001\ ;\ -0.0667], [-2.0001\ ;\ -0.0270], [-2.0001\ ;\ -0.0066]\}$.

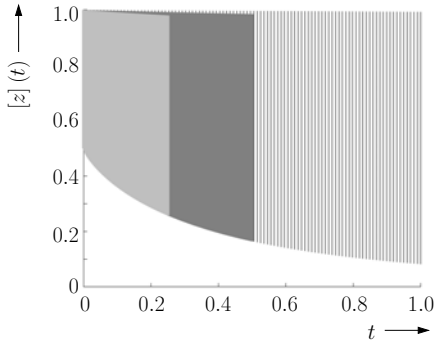
Considering again the **Case a**, a restart of the integration is now performed at the points $t_k \in \{0.25, 0.50, 0.75\}$. The resulting state enclosure, including a comparison with the bounds obtained for a single time window of length $T = 1$ are shown in Fig. 5c. Here, the contractor introduced in Theorem 9 has the form

$$[\tilde{\lambda}] := \frac{-p \cdot \Big([z_e]\,([\tau_k]) \cap [z_{\mathrm{ref}}]\,([\tau_k])\Big)^3 + [-1\ ;\ 1] \cdot \mu(t_{k-1})}{[z_e]\,([\tau_k]) \cap [z_{\mathrm{ref}}]\,([\tau_k])}\ , \tag{70}$$

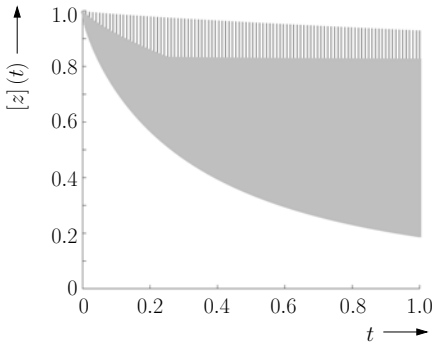
where $[z_{\mathrm{ref}}]\,([\tau_k])$ is the evaluation of the enclosure function obtained for the overall time window without any temporal discretization. It can be seen that the subdivision of the time window ($T = 1$) leads to a noticeable reduction of the computed



(a) State enclosures for **Case a**.



(b) State enclosures for **Case b**.



(c) State enclosures for **Case a**, including a restart of the integration at the points $t_k \in \{0.25, 0.50, 0.75\}$.

Figure 5: Guaranteed state enclosures for different integration horizons $T \in \{0.25, 0.50, 1.0\}$ and different levels of uncertainty.

interval widths. Future work will aim at the development of further contractor approaches, allowing both for a refinement of the bounds $\boldsymbol{\mu}$ and for incorporating simulations computed over long time windows as some kind of measured state enclosure as it would be done in the frame of an interval-based state observer synthesis. To perform a comparison with the parameter bounds listed above, the following results were obtained: $[\lambda]([\tau_1]) \in [-2.0001\ ;\ -0.5261]$, $[\lambda]([\tau_2]) \in [-3.3658\ ;\ -0.0121]$, $[\lambda]([\tau_3]) \in [-2.6236\ ;\ -0.0064]$, and $[\lambda]([\tau_4]) \in [-2.1951\ ;\ -0.0035]$ with $[\tau_1] = [0\ ;\ 0.25]$, $[\tau_2] = [0.25\ ;\ 0.50]$, $[\tau_3] = [0.50\ ;\ 0.75]$, and $[\tau_4] = [0.75\ ;\ 1.0]$.

7 Conclusions and Outlook on Future Work

In this paper, extensions of an interval-based exponential enclosure technique originally developed for integer-order sets of ordinary differential equations were presented to obtain a novel Mittag-Leffler function-based generalization valid also for explicit, continuous-time sets of fractional-order differential equations. This type of iteration was first discussed by the authors in [33,35], however, without accounting for the practically necessary extension towards the use of temporal subintervals. The corresponding time discretization scheme requires the quantification of truncation errors — which are caused by the infinite-horizon memory effects of fractional-order systems — and which do not exist for classical ordinary differential equations.

A first implementation of the novel routine for quantifying these error bounds, has been presented and interfaced for the first time with a contractor approach that further allows for reducing conservativeness of the obtained solution sets.

Future work will deal with a generalization of the iteration scheme to system models with an oscillatory behavior, for which it seems to be reasonable that complex-valued state enclosures are determined as it was already demonstrated for the case of the integer-order counterpart [36]. Moreover, possible strategies for determining optimal subdivision strategies of the investigated integration time horizons — with the aim of minimizing the computed interval diameters — will be investigated.

References

[1] Amairi, M., Aoun, M., Najar, S., and Abdelkrim, M.N. A constant enclosure method for validating existence and uniqueness of the solution of an initial value problem for a fractional differential equation. *Applied Mathematics and Computation*, 217(5):2162–2168, 2010. DOI: 10.1016/j.amc.2010.07.015.

[2] Andre, D., Meiler, M., Steiner, K., Wimmer, Ch., Soczka-Guth, T., and Sauer, D.U. Characterization of High-Power Lithium-Ion Batteries by Electrochemical Impedance Spectroscopy. I. Experimental Investigation. *Journal of Power Sources*, 196(12):5334–5341, 2011. DOI: 10.1016/j.jpowsour.2010.12.102.

[3] Bel Haj Frej, G., Malti, R., Aoun, M., and Raïssi, T. Fractional Interval Observers And Initialization Of Fractional Systems. *Communications in Nonlinear Science and Numerical Simulation*, 82:105030, 2020. DOI: 10.1016/j.cnsns.2019.105030.

[4] Chen, Y. Oustaloup-Recursive-Approximation for Fractional Order Differentiators. MATLAB Central File Exchange. www.mathworks.com/matlabcentral/fileexchange/3802-oustaloup-recursive-approximation-for-fractional-order-differentiators, accessed: Aug. 14, 2020.

[5] Craiem, D. and Magin, R. Fractional Order Models of Viscoelasticity as an Alternative in the Analysis of Red Blood Cell (RBC) Membrane Mechanics. *Physical Biology*, 7:13001, 03 2010. DOI: 10.1088/1478-3975/7/1/013001.

[6] Demirci, E. and Ozalp, N. A method for solving differential equations of fractional order. *Journal of Computational and Applied Mathematics*, 236(11):2754–2762, 2012. DOI: 10.1016/j.cam.2012.01.005.

[7] Dorjgotov, K., Ochiai, H., and Zunderiya, U. On Solutions of Linear Fractional Differential Equations and Systems Thereof. 2018. arXiv:1803.09063.

[8] Efimov, D., Raïssi, T., Chebotarev, S., and Zolghadri, A. Interval State Observer for Nonlinear Time Varying Systems. *Automatica*, 49(1):200–205, 2013. DOI: 10.1016/j.automatica.2012.07.004.

[9] El-Khazali, R., Batiha, I.M., and Momani, S. Approximation of fractional-order operators. In Agarwal, P., Baleanu, D., Chen, Y., Momani, S., and Machado, J.A.T., editors, *Fractional Calculus. ICFDA 2018. Springer Proceedings in Mathematics & Statistics, vol. 303*, pages 121–151, Singapore, 2019. Springer Singapore. DOI: 10.1007/978-981-15-0430-3_8.

[10] Garrappa, R. Numerical Evaluation of Two and Three Parameter Mittag-Leffler Functions. *SIAM Journal on Numerical Analysis*, 53(3):1350–1369, 2015. DOI: 10.1137/140971191.

[11] Ghosh, U., Sarkar, S., and Das, S. Solution of System of Linear Fractional Differential Equations with Modified Derivative of Jumarie Type. *American Journal of Mathematical Analysis*, 3(3):72–84, 2015. DOI: 10.12691/ajma-3-3-3.

[12] Gorenflo, R., Kilbas, A.A., Mainardi, F., and Rogosin, S.V. *Mittag-Leffler Functions, Related Topics and Applications*. Springer–Verlag, Berlin, Heidelberg, 2014. DOI: 10.1007/978-3-662-43930-2.

[13] Gorenflo, R., Loutchko, J., and Luchko, Y. Computation of the Mittag-Leffler Function and its Derivatives. *Fractional Calculus & Applied Analysis (FCAA)*, 5(4):491–518, 2002.

[14] Haubold, H.J., Mathai, A.M., and Saxena, R.K. Mittag-Leffler Functions and Their Applications. *Journal of Applied Mathematics*, 2011:51 pages, 2011. DOI: 10.1155/2011/298628.

[15] 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 *Proc. of the 21st IFAC World Congress*, Berlin, Germany, 2020.

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

[17] Kaczorek, T. *Positive 1D and 2D Systems*. Springer–Verlag, London, 2002. DOI: 10.1007/978-1-4471-0221-2.

[18] Kersten, J., Rauh, A., and Aschemann, H. State-Space Transformations of Uncertain Systems with Purely Real and Conjugate-Complex Eigenvalues into a Cooperative Form. In *Proc. of 23rd Intl. Conference on Methods and Models in Automation and Robotics*, Miedzyzdroje, Poland, 2018. DOI: 10.1109/MMAR.2018.8486085.

[19] Kersten, J., Rauh, A., and Aschemann, H. Application-Based Discussion of Verified Simulations of Interval Enclosure Techniques. In *Proc. of 24th Intl. Conference on Methods and Models in Automation and Robotics*, Miedzyzdroje, Poland, 2019. DOI: 10.1109/MMAR.2019.8864673.

[20] Kersten, J., Rauh, A., and Aschemann, H. Transformation of Uncertain Linear Fractional Order Differential Equations into a Cooperative Form. In *Proc. of the 8th IFAC Symposium on Mechatronic Systems (MECHATRONICS 2019) and the 11th IFAC Symposium on Nonlinear Control Systems (NOLCOS 2019)*, Vienna, Austria, 2019. DOI: 10.1016/j.ifacol.2019.12.035.

[21] Kersten, J., Rauh, A., and Aschemann, H. Application-Based Analysis of Transformations of Uncertain Dynamical Systems Into a Cooperative Form. *Reliable Computing*, 2020. Under review.

[22] Lyons, R., Vatsala, A.S., and Chiquet, R. Picard's Iterative Method for Caputo Fractional Differential Equations with Numerical Results. *Mathematics*, 5(4), 2017. DOI: 10.3390/math5040065.

[23] Malti, R. and Victor, S. CRONE Toolbox for System Identification Using Fractional Differentiation Models. In *Proc. of 17th IFAC Symposium on System Identification SYSID 2015*, volume 48, pages 769–774, 2015. DOI: 10.1016/j.ifacol.2015.12.223.

[24] Miller, K.S. and Samko, S.G. A Note on the Complete Monotonicity of the Generalized Mittag-Leffler Function. *Real Analysis Exchange*, 23(2):753–756, 1997-98. DOI: 10.2307/44153996.

[25] Müller, M. Über die Eindeutigkeit der Integrale eines Systems gewöhnlicher Differenzialgleichungen und die Konvergenz einer Gattung von Verfahren zur Approximation dieser Integrale. In *Sitzungsbericht Heidelberger Akademie der Wissenschaften*, 1927. In German.

[26] Nedialkov, N.S. Interval Tools for ODEs and DAEs. In *CD-Proc. of 12th GAMM-IMACS Intl. Symposium on Scientific Computing, Computer Arithmetic, and Validated Numerics SCAN 2006*, Duisburg, Germany, 2007. IEEE Computer Society. DOI: 10.1109/SCAN.2006.28.

[27] Oustaloup, A. *La Dérivation Non Entière: Théorie, Synthèse et Applications*. Hermès, Paris, 1995. In French.

[28] Oustaloup, A., Levron, F., Mathieu, B., and Nanot, F. M. Frequency-Band Complex Noninteger Differentiator: Characterization and Synthesis. *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, 47(1):25–39, 2000. DOI: 10.1109/81.817385.

[29] Papoulia, K., Panoskaltsis, V., Kurup, N., and Korovajchuk, I. Rheological Representation of Fractional Order Viscoelastic Material Models. *Rheologica Acta*, 49:381–400, 04 2010. DOI: 10.1007/s00397-010-0436-y.

[30] Podlubny, I. *Fractional Differential Equations: An Introduction to Fractional Derivatives, Fractional Differential Equations, to Methods of Their Solution and Some of Their Applications*. Mathematics in Science and Engineering. Academic Press, London, 1999. DOI: 10.1016/s0076-5392(99)x8001-5.

[31] 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.

[32] Raïssi, T., Efimov, D., and Zolghadri, A. Interval State Estimation for a Class of Nonlinear Systems. *IEEE Transactions on Automatic Control*, 57:260–265, 2012. DOI: 10.1109/TAC.2011.2164820.

[33] Rauh, A., Kersten, J., and Aschemann, H. Techniques for Verified Reachability Analysis of Quasi-Linear Continuous-Time Systems. In *Proc. of 24th Intl. Conference on Methods and Models in Automation and Robotics*, Miedzyzdroje, Poland, 2019. DOI: 10.1109/MMAR.2019.8864648.

[34] Rauh, A., Kersten, J., and Aschemann, H. Interval and Linear Matrix Inequality Techniques for Reliable Control of Linear Continuous-Time Cooperative Systems with Applications to Heat Transfer. *International Journal of Control*, pages 1–18, 2020. DOI: 10.1080/00207179.2019.1708966, Available online.

[35] Rauh, A., Kersten, J., and Aschemann, H. Interval-Based Verification Techniques for the Analysis of Uncertain Fractional-Order System Models. In *Proc. of the 18th European Control Conference ECC2020*, St. Petersburg, Russia, 2020.

[36] Rauh, A., Westphal, R., Aschemann, H., and Auer, E. Exponential Enclosure Techniques for Initial Value Problems with Multiple Conjugate Complex Eigenvalues. In Nehmeier, M., von Gudenberg, J. Wolff, and Tucker, W., editors, *Scientific Computing, Computer Arithmetic, and Validated Numerics*, pages 247–256, Cham, 2016. Springer International Publishing. DOI: 10.1007/978-3-319-31769-4_20.

[37] Smith, H.L. *Monotone Dynamical Systems: An Introduction to the Theory of Competitive and Cooperative Systems*, volume 41. Mathematical Surveys and Monographs, American Mathematical Soc., Providence, 1995.

[38] Wang, B., Liu, Z., Li, S., Moura, S., and Peng, H. State-of-Charge Estimation for Lithium-Ion Batteries Based on a Nonlinear Fractional Model. *IEEE Trans. on Control Systems Technology*, 25(1):3–11, 2017. DOI: 10.1109/TCST.2016.2557221.

[39] Zou, Ch., Zhang, L., Hu, X., Wang, Z., Wik, T., and Pecht, M. A Review of Fractional-Order Techniques Applied to Lithium-Ion Batteries, Lead-Acid Batteries, and Supercapacitors. *Journal of Power Sources*, 390:286–296, 2018. DOI: 10.1016/j.jpowsour.2018.04.033.

Acta Cybernetica **25** (2021) 49–68.

Confidence-based Contractor, Propagation and Potential Clouds for Differential Equations*

Julien Alexandre dit Sandretto[a]

Abstract

A novel interval contractor based on the confidence assigned to a random variable is proposed in this paper. It makes it possible to consider at the same time an interval in which the quantity is guaranteed to be, and a confidence level to reduce the pessimism induced by interval approach. This contractor consists in computing a confidence region. Using different confidence levels, a particular case of potential cloud can be computed. As application, we propose to compute the reachable set of an ordinary differential equation under the form of a set of confidence regions, with respect to confidence levels on initial value.

Keywords: interval analysis, confidence level, potential cloud, reachability for ODEs

1 Introduction

An interval (see [16] for more details) aims to bound all the values of an uncertain quantity, for example provided by a measurement device [11]. This approach is highly effective for every safety, verification or validation procedures because intervals are conservative. The major inconvenience is that intervals are sometimes too pessimistic, and lead to unexploitable results. Obviously, bounds can be set on a given measured datum (arbitrarily large), therefore the measurement can be guaranteed to be enclosed in an interval. However, a probability distribution can also be deduced from past observations, with more effort, and associated to the measurement device. We propose to exploit a probability distribution to reduce the pessimism of intervals.

In order to filter (or reduce) an interval with respect to a given information (such as a constraint, a measurement or any kind of information), contractors are mainly used [5]. A contractor is a function taking an interval as input and

*This work was supported by the "Chair Complex Systems Engineering - Ecole polytechnique, THALES, DGA, FX, Dassault Aviation, DCNS Research, ENSTA Paris, Télécom Paris, and Fondation ParisTech" and partially supported by DGA AID.

[a]ENSTA Paris, Institut Polytechnique de Paris, Palaiseau, France. E-mail: alexandre@ensta.fr, ORCID: https://orcid.org/0000-0002-6185-2480.

DOI: 10.14232/actacyb.285177

returning a smaller interval (included in the previous one). It can be seen as a filtering approach in the sense that a contractor reduces an interval without any solution loss. Contractors are often associated to a propagation procedure, at least a propagation loop, to communicate a contraction on a variable to the others through some constraints.

In this paper, a novel contractor is proposed to filter an interval following a confidence level given on the associated quantity. This confidence level is an input of the contractor, the "new" information, while the probability distribution of the considered variable is a characteristic of the associated random variable. To compute such contraction, the probability density function (or density for short) is taken into account. In this paper, we then focus on random continuous variable with a known (and analytical) density function, such as uniform distribution, normal distribution, beta distribution, etc.

Combining intervals and probability has been already proposed in numerous papers using techniques such as p-boxes [8, 21], fuzzy sets [7, 18], box-particles [1] and potential cloud [9, 18]. Some of these representations can be deduced from probability intervals [6]. The notion of cloud is interesting for us to represent a very substantial result to a problem such as computing the image of a function, a set inversion, or the solution of a constraint satisfaction problem (these problems have been solved with interval methods in [11]). Computing a solution of such a problem with several confidence levels provides different boxes which, gathered, provide a particular type of potential cloud.

We are particularly interested in Ordinary Differential Equations (ODEs) and validated methods to compute their reachable sets via validated simulation [3, 12, 17, 20]. In the case of Initial Value Problems (IVPs) with ODEs, the initial state is primordial. An uncertain initial state is generally bounded in a box (also in a zonotope [3] or a polytope [4]). As experimentation, we propose to consider in addition to this initial box some confidence levels, and we apply the presented approach. It allows us to describe the reachable set by a cloud. This more expressive result can then be used in various control problems, parameter identification, verification, etc.

This paper is organized as follows. The next section is dedicated to establishing the notation and recall the notion of confidence in probability theory. Section 3 presents the relationship we use between probabilities and interval analysis. Sections 4 and 5 contain the main results presented in this article: a confidence-based contractor and the propagation of a confidence contraction to a potential cloud. The last section concludes the article and gives some hint on future works.

2 Confidence Interval

In this section, some notions required for the definition of confidence interval are introduced. However, to clarify the concept of confidence interval as soon as possible, an informal explanation can be given:

A confidence interval is a set $\mathcal{S}$ for which the probability of the given random variable that lies in this set is equal to the given probability P.

Let us define X a random variable (also called random quantity, aleatory variable). A random variable takes different values, resulting from a random phenomenon. In this paper, we focus on continuous univariate distributions such that $X \in \mathbb{R}$. Such a variable implies a mapping (a function) between its possible values and a probability of appearance. It means that X is measurable.

Definition 1 (Continuous random real variable)**.** *A* **continuous random variable** *is a random variable whose cumulative distribution function is continuous everywhere [14].*

Considering a continuous random real variable $X : \Omega \mapsto \mathcal{A}$, X is a measurable function from a set of possible outcomes Ω to the measurable space $\mathcal{A}$.

The probability that X takes a value in a measurable subset $S \subset \mathcal{A}$ is given by:

$$\Pr[X \in S] = P(\{\omega \in \Omega | X(\omega) \in S\}),$$

where P is the probability measure equipped with Ω.

Considering continuous variable, X can take any numerical value in an interval following the distribution. The distribution is then fully characterized by a probability density function (see Figure 1).

Definition 2 (Probability density function)**.** *A random variable X with values in a measurable space (usually $\mathbb{R}^n$) has* **probability density function** *f_X , where f_X is a non-negative Lebesgue-integrable function, if:*

$$\Pr[a \leq X \leq b] = \int_a^b f_X(x)\, dx.$$

A property is fundamental in probability:

$$\Pr[-\infty < X < \infty] = \int_{-\infty}^{\infty} f_X(x)\, dx = \mathbf{1}.$$

Let us define $\hat{x}$ a single observed sample of the quantity X. In statistics, an observed datum allows to compute a confidence interval [19], that is to say an interval which may contain the actual value, with respect to a given confidence level. A formal definition can then be stated:

Definition 3 (Confidence interval)**.** *Let X be a random sample from a probability distribution f_X. A* **confidence interval** *with confidence level cc is an interval with endpoints a and b with the property:*

$$Pr(a < X < b) = \int_a^b f_X(x)\, dx = cc. \tag{1}$$

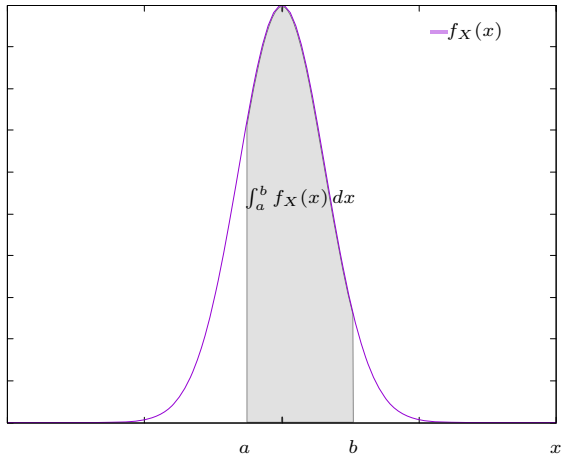


Figure 1: Probability density function of a variable X.

For example, considering a confidence level $CL = 95\%$, one can define the confidence interval $C_{95\%}$. This interval can be obtained by observation (statistical approach) or with the help of a known distribution (probability approach). A new measurement $\hat{x}$ coming from the (same) experiment will be in the associated confidence interval such that:

$$\hat{x} \in C_{95\%} \quad 95\% \text{ of the time.}$$

Figure 2 illustrates the concept of confidence interval and confidence level for a normal distribution.

In the particular case of symmetric distribution and regarding centered confidence intervals, they follow the inclusion property:

$$CL_1 < CL_2 \implies C_{CL_1} \subset C_{CL_2}$$

For example, $C_{90\%} \subset C_{95\%}$.

Remark 1. The extremal values for a confidence level have a particular meaning: 0% means that there is no chance that a future observation will follow the previous observations, while 100% means that it is sure that a future observation will follow the previous observations.

In the following, the distribution are considered symmetric. Therefore, we mainly focus on normal dstribution.

3 Interval Analysis and Probability

3.1 Introduction to intervals

The simplest and most common way to represent and manipulate sets of values is *interval arithmetic* (see [15]). An interval $[x_i] = [\underline{x_i}, \overline{x_i}]$ defines the set of reals x_i such that $\underline{x_i} \leq x_i \leq \overline{x_i}$. $\mathbb{IR}$ denotes the set of all intervals over reals. The size or the width of $[x_i]$ is denoted by $w([x_i]) = \overline{x_i} - \underline{x_i}$.

Interval arithmetic extends to $\mathbb{IR}$ elementary functions over $\mathbb{R}$. For instance, the interval sum, *i.e.*, $[x_1] + [x_2] = [\underline{x_1} + \underline{x_2}, \overline{x_1} + \overline{x_2}]$, encloses the image of the sum function over its arguments. An interval vector or a *box* $[\mathbf{x}] \in \mathbb{IR}^n$, is a Cartesian product of n intervals. The enclosing property basically defines what is called an *interval extension* or an *inclusion function*.

Definition 4 (Inclusion function)**.** *Consider a function* $f : \mathbb{R}^n \to \mathbb{R}^m$*, then* $[f] : \mathbb{IR}^n \to \mathbb{IR}^m$ *is said to be an extension of* f *to intervals if*

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

It is possible to define inclusion functions for all elementary functions such as $\times$, $\div$, sin, cos, exp, etc. The *natural* inclusion function is the simplest to obtain: all occurrences of the real variables are replaced by their interval counterpart and all arithmetic operations are evaluated using interval arithmetic. More sophisticated inclusion functions such as the centered form, or the Taylor inclusion function may also be used (see [11] for more details).

Combining the inclusion function and the rectangle rule, integral can be bounded following:

$$\int_a^b f(x)\, dx \in (b-a) \cdot [f]([a,b])$$

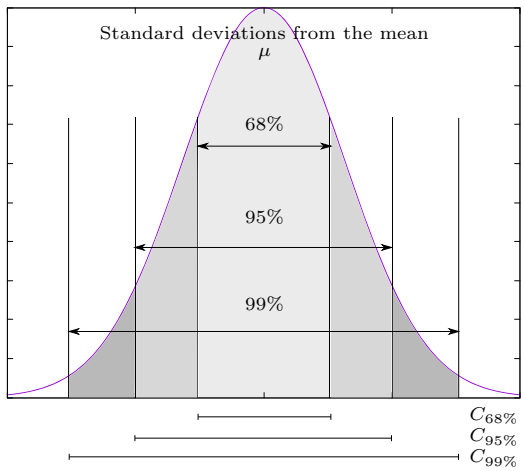


Figure 2: Confidence intervals and confidence levels.

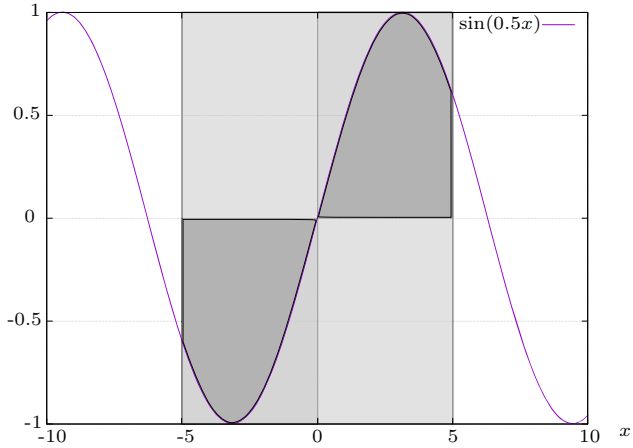


Figure 3: Computation of $\int_{-5}^{5} \sin(0.5x)\,dx$. With interval $[-5, 5]$ (in light grey), result is $[-10, 10]$ while with two intervals $([-5, 0], [0, 5])$ (in darker grey) we obtain $[-5, 5]$.

In order to obtain a better approximation, a discretization of the integral can be used, as shown in Figure 3:

$$\int_a^b f(x)\,dx = \sum_{i=1}^{n} \int_{k_i}^{k_{i+1}} f(x)\,dx \in \sum_{i=1}^{n} (k_{i+1} - k_i) \cdot [f]([k_i, k_{i+1}]), \tag{2}$$

with $k_1 = a$ and $k_{n+1} = b$.

Notations: In the following, we denote by $1.2[3, 4]$ the interval $[1.23, 1.24]$.

3.2 Intervals and probability

A random variable X with a probability density f_X is observed via a measurement device. We consider that the density is defined by μ, a mean or expectation of the distribution, and σ a standard deviation. An observed sample is denoted by $\hat{x}$, given with the device associated uncertainty $\pm m$. An interval containing the actual value can be defined as $[x] = [\hat{x} - m, \hat{x} + m]$ (bias can also be added). For example, we consider that X follows a normal distribution, such that

$$f_X(x|\mu, \sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

with $\mu = 1.0$ and $\sigma = 1.0$ (then the variance $\sigma^2 = 1.0$). Interval arithmetic and Equation (2) allow us to compute an enclosure of the integral of the density between 0 and 1 (*i.e.*, the probability $\Pr[0 \leq X \leq 1]$), as depicted in Figure 4. The computed result with $n = 100$ is $0.34[05578, 21275]$ (while a mathematical tool[1] using floating

[1]Matlab was used for this comparison.

numbers computes 0.341345). Using symmetry, the integral between 0 and 2, that is to say between $\mu - \sigma$ and $\mu + \sigma$, is included in $0.68[11, 42]$, which contains the theoretical confidence level (68.27%) for the confidence interval $[\mu - \sigma, \mu + \sigma]$.

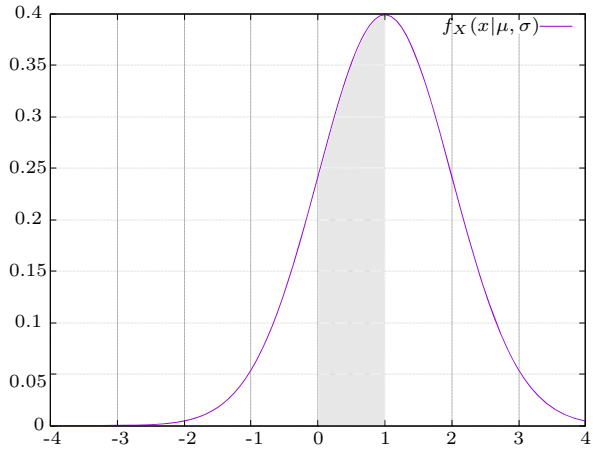


Figure 4: Interval computation of the integral of a probability density (for a normal distribution).

In theory, about 95% of the values lie within two standard deviations, that is to say in the interval $[\mu - 2\sigma, \mu + 2\sigma]$. With interval analysis, the obtained probability is $0.9[48, 61]$, with $n = 100$, while with $n = 1000$ it is reduced to $0.95[38, 51]$ (the approximation is better).

Remark 2. The extremal values for a confidence level have a particular meaning (see Remark 1). We define the associated confidence intervals such that $C_{0\%} = \emptyset$ and $C_{100\%} = [-\infty, \infty]$.

4 Confidence-based Contractor

The main idea of our contribution is that a measurement provides an interval (by considering the uncertainty of the measurement device) which is guaranteed to contain the actual quantity[2], but sometimes too pessimistic to be workable. In addition to an enclosure, the observed variable can be associated to a probability distribution. Our idea is to combine an interval provided by a measurement and the probability distribution by establishing a confidence level on the quantity.

4.1 A confidence-based contractor

A generic contractor Cr must satisfy two properties [11]:

- Contractance : $\forall [x] \in \mathbb{IR}, Cr([x]) \subset [x]$,

[2]Outliers are not considered in this paper.

- Correctness : $\forall [x] \in \mathbb{IR}, [x] \cap \mathbb{S} \subseteq Cr([x])$ (with $\mathbb{S}$ the solution set).

We propose a confidence-based contractor, denoted Cbc, defined as follows:

$$\begin{array}{rll} Cbc([x]|f_X, cc): & \mathbb{IR} & \mapsto \mathbb{IR} \\ & [x] & \rightarrow [x] \cap [y] \end{array}$$

with $[y]$ defined such that $Pr(x \in [y]) = \int_{[y]} f_X(x)\, dx = cc$ ($[y]$ is the confidence interval), cc being the confidence coefficient ($0 \leq cc \leq 1$). For example, one can use the parameter assignment $cc = 0.68$ for a confidence level of 68%.

Proposition 1. *The confidence-based contractor is a contractor.*

Proof. Two properties have to hold: contractance and correctness. The **contractance** is obvious because the confidence-based contractor uses the intersection operation, and $[x] \cap [y] \subseteq [x], \forall [y]$. **Correctness** is more complex to handle as a new type of correctness needs to be introduced: the confidence correctness. *Confidence correctness* means that if the confidence given on the quantity is well estimated for a sample, then the variable lies in the corresponding confidence interval with the associated probability and then the correctness holds. Therefore, if the correctness may not hold, the confidence correctness holds (all computations are conducted with validated interval arithmetic). For example, the 90%-correctness holds, except for 10% of the samples. □

Figure 5 illustrates the effect of the confidence-based contractor applied to the following example:

Example 1. Let X a random variable with a normal distribution, such that

$$f_X(x|\mu, \sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

with $\mu = 1.0$ and $\sigma = 1.0$. The quantity X is observed and one measurement is obtained: $[x] = [0.7, 2.1]$. A confidence level of 68.27% is given on X, that is to say that we are confident on the accuracy of the observations, so X stays close to its mean. Our method computes the contraction such that:

$$\begin{array}{rcl} Cbc([0.7, 2.1]|(1.0, 1.0), 0.6827) & = & [0.7, 2.1] \cap [0.0, 2.0] \\ & = & [0.7, 2.0] \end{array}$$

So the upper bound is reduced with respect to the confidence level. The pessimism induced by interval approach is thus limited.

As seen before, two special cases can be described:

- $\forall [x], Cbc([x]|f_X, 0) = \emptyset$ (annihilating element)
- $\forall [x], Cbc([x]|f_X, 1) = [x]$ (identity element)

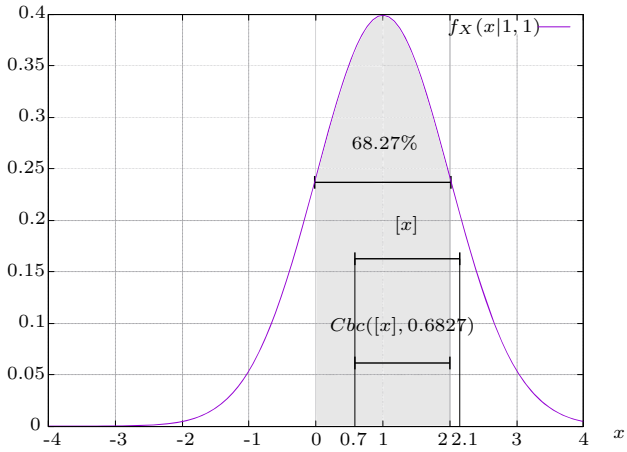


Figure 5: Illustration of confidence-based contraction.

For two different confidence coefficients cc_1 and cc_2 such that $cc_1 < cc_2$, the following order holds:

$$\forall [x], Cbc([x]|f_X, cc_1) \subset Cbc([x]|f_X, cc_2)$$

The contractor Cbc can be composed with other contractors or with itself. The order previously shown leads to two particularities:

- $\forall [x], Cbc(Cbc([x]|f_X, cc_2)|f_X, cc_1) = Cbc([x]|f_X, cc_1)$
- $\forall [x], Cbc(Cbc([x]|f_X, cc_1)|f_X, cc_2) = Cbc([x]|f_X, cc_1)$

That is to say that the lower confidence coefficient is primary. The standard operations on sets can be extended to this specific contractor:

- $(Cbc([x]|f_X, cc_1) \cap Cbc([x]|f_X, cc_2))([x]) = Cbc([x]|f_X, cc_1)$ (intersection)
- $(Cbc([x]|f_X, cc_1) \cup Cbc([x]|f_X, cc_2))([x]) = Cbc([x]|f_X, cc_2)$ (union)

Remark 3. As a confidence interval is enclosed by its support interval (which guarantee the enclosure of the quantity), outliers can be automatically detected and rejected. Our approach can then be able to produce *robust confidence intervals*.

4.2 Computation of confidence interval

The confidence-based contractor presented in this paper needs the computation of the confidence interval associated to a given confidence level. Three cases can be detailed:

- Case 1: a well known probability distribution and a particular confidence level with known confidence interval. For example, a normal distribution with a 95% confidence level gives a confidence interval $[\mu - 2\sigma, \mu + 2\sigma]$.

- Case 2: a probability distribution with a known inverse function, such as the inverse of error function for Gaussian density function (*i.e.* erf^{-1}).
- Case 3: the general symmetric case without any particular values.

We focus on the third case with the presentation of Algorithm 1. This algorithm follows a predictor-corrector based approach. This latter needs guesses a and b. For mean-centered confidence intervals, a and b have to be chosen such that $\mu = \frac{a+b}{2}$ (but it is not mandatory, in presence of bias for example). Two operations are also needed to implement this algorithm: Narrow and Widen. Widen is equivalent to an inflation of a well chosen percentage (*e.g.* 1%), while Narrow is a deflation of the same percentage.

We apply the proposed algorithm to Example 1 for different confidence levels. The results are gathered in Table 1 and plotted in Figure 6. Experiments are as follows: for different confidence levels cc_i from 10% to 99% – and with the particular value 68.27% – (first column of Table 1), we apply the contractor $Cbc([-5,5]|f_X, cc_i)$, with f_X from Example 1. It provides several contracted confidence intervals (second column of Table 1). Then, for all these confidence intervals, we compute the probability for the variable to be in the confidence interval with the method presented in Section 3.2 (third column of Table 1) for verification.

Remark 4. In the case of a non symmetric distribution, the maximal confidence interval can be computed to obtain a one-to-one application between a confidence coefficient cc and confidence interval $[a, b]$. The maximal confidence interval is defined by:

$$\max_{[a,b]} ||b - a||, \quad \Pr[a \leq X \leq b] = cc \tag{3}$$

5 Application to Reachability

As an application, we propose to compute the reachability of Ordinary Differential Equations (ODEs) from an interval initial value. We imagine that the initial value

Algorithm 1 Confidence interval computation

Require: A distribution f_X, a confidence coefficient cc, guesses for the bounds of confidence interval a and b
  Compute $[P_{a,b}] = \Pr[a \leq X \leq b] = \int_a^b f_X(x)\,dx$ with the interval method presented in Section 3
  while $cc \notin [P_{a,b}]$ **do**
    if $cc < \underline{[P_{a,b}]}$ **then**
      Narrow $[a, b]$
    else
      Widen $[a, b]$
    end if
    Compute $[P_{a,b}] = \Pr[a \leq X \leq b] = \int_a^b f_X(x)\,dx$
  end while

Table 1: Confidence intervals for different values of confidence level and computed probabilities.

Confidence	Interval	Probability
10%	$[0.8743, 1.1256]$	$[0.0999, 0.1000]$
20%	$[0.7466, 1.2533]$	$[0.1999, 0.2000]$
30%	$[0.6146, 1.3853]$	$[0.2999, 0.3000]$
40%	$[0.4756, 1.5243]$	$[0.3999, 0.4000]$
50%	$[0.3256, 1.6743]$	$[0.4998, 0.5000]$
60%	$[0.1581, 1.8418]$	$[0.5999, 0.6003]$
68.27%	$[0, 2]$	$[0.6824, 0.6830]$
70%	$[-0.0362, 2.0362]$	$[0.6995, 0.7002]$
80%	$[-0.2800, 2.2800]$	$[0.7988, 0.8000]$
90%	$[-0.6450, 2.6450]$	$[0.8990, 0.9010]$
95%	$[-0.9500, 2.9500]$	$[0.9475, 0.9501]$
99%	$[-1.5200, 3.5200]$	$[0.9863, 0.9901]$

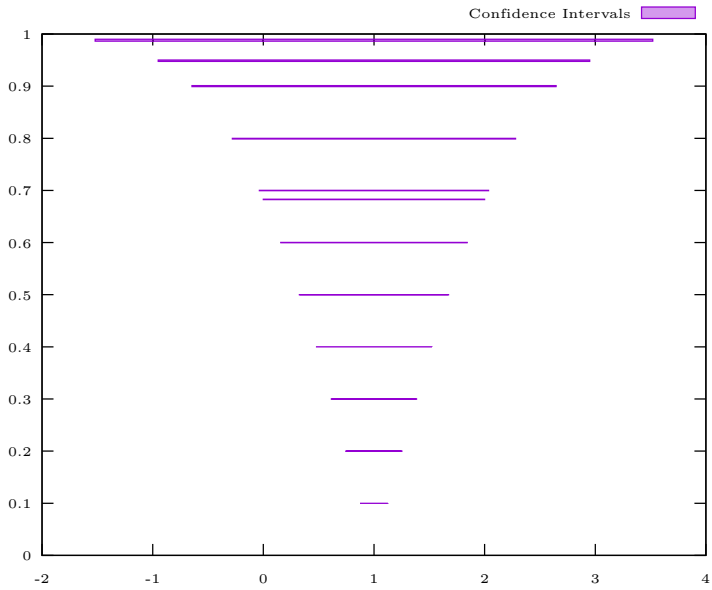


Figure 6: Confidence intervals for different values of confidence level with respect to computed probabilities.

is provided by a measurement and that different confidence levels can be considered. The proposed confidence-based contractor can then be used to reduce the initial interval. We solve the Initial Value Problem (IVP) for all the obtained intervals by the help of validated integration. The collection of computed reachable sets is

depicted through a kind of potential cloud that is useful for verification, control synthesis, or validation problems on ODEs. An approach consisting also in the propagation of probabilities through an initial value problem with ODE has been proposed in last decade [13]. If the goal is the same, how to add information coming from a probability knowledge to the reachability analysis, the technique is different. In [13], authors exploit Taylor models to reprensent uncertainties (mainly on parameters), and propagate them into an integration process to compute fuzzy trajectories (see [22] for fuzzy sets). Our contribution is mainly based on confidence level, this concept is not considered in [13]. Nevertheless, an example from this latter is studied in Section 5.3 to compare and discuss both approaches.

5.1 Integration and propagation

5.1.1 Validated simulation

When dealing with validated computation, mathematical representation of an IVP-ODE is as follows:

$$\begin{cases} \dot{\mathbf{y}}(t) = g(t, \mathbf{y}(t)) \\ \mathbf{y}(0) \in [\mathbf{y}_0] \subseteq \mathbb{R}^n. \end{cases} \tag{4}$$

We assume that $g : \mathbb{R} \times \mathbb{R}^n \to \mathbb{R}^n$ is continuous in t and globally Lipschitz in $\mathbf{y}$, so Equation (4) admits a unique solution.

The set (expressed as a box) $[\mathbf{y}_0]$ of initial conditions is used to model some (bounded) uncertainties. For a given initial condition $\mathbf{y}_0 \in [\mathbf{y}_0]$, the solution at time $t > 0$, when it exists, is denoted $\mathbf{y}(t; \mathbf{y}_0)$. The goal, for **validated numerical integration** methods, is then to compute the set of solutions of Equation (4), *i.e.*, the set of possible solutions at time t given the initial condition in the set of initial conditions $[\mathbf{y}_0]$:

$$\mathbf{y}(t; [\mathbf{y}_0]) = \{\mathbf{y}(t; \mathbf{y}_0) \mid \mathbf{y}_0 \in [\mathbf{y}_0]\}. \tag{5}$$

Validated numerical integration schemes, exploiting set-membership framework, aim at producing the solution of the IVP-ODE that is the set defined in Equation (5). It results in the computation of an outer approximation of $\mathbf{y}(t; [\mathbf{y}_0])$. The use of set-membership computation for the problem described above makes possible the design of an inclusion function for the computation of $[\mathbf{y}](t; [\mathbf{y}_0])$, which is an outer approximation of $\mathbf{y}(t; [\mathbf{y}_0])$ defined in Equation (5). To do so, a sequence of time instants $t_1, \ldots, t_n$ such that $t_1 < \cdots < t_n$ and a sequences of boxes $[\mathbf{y}_1], \ldots, [\mathbf{y}_n]$ such that $\mathbf{y}(t_{i+1}; [\mathbf{y}_i]) \subseteq [\mathbf{y}_{i+1}]$, $\forall i \in [0, n-1]$ are computed. From $[\mathbf{y}_i]$, computing the box $[\mathbf{y}_{i+1}]$ is a classical 2-step method (see [12]):

- *Phase 1:* compute an a priori enclosure $[\tilde{\mathbf{y}}_i]$ of the set $\{\mathbf{y}(t_k; \mathbf{y}_i) \mid t_k \in [t_i, t_{i+1}], \mathbf{y}_i \in [\mathbf{y}_i]\}$, such that $\mathbf{y}(t_k; [\mathbf{y}_i])$ is guaranteed to exist,
- *Phase 2:* compute a tight enclosure of the solution $[\mathbf{y}_{i+1}]$ at time t_{i+1}.

Two main approaches can be used to compute the tight enclosure in *Phase 2*. The first one, and the most used, is the Taylor method [15, 17]. The second one, more

recently studied, is the validated Runge-Kutta approach [3]. The **reachability** consists in computing the enclosure of the set of states at a specific instant τ as defined above by $[\mathbf{y}(\tau); [\mathbf{y}_0]]$.

5.1.2 Propagation

The procedure given in Section 5.1.1, applied to Equation (5), produces a reachable tube based on time discretization as depicted in Figure 7. Considering an initial value $[\mathbf{y}_0]^*$ such that $[\mathbf{y}_0]^* \subseteq [\mathbf{y}_0]$, a **propagation** procedure [2] can be performed to compute the solution of the IVP-ODE:

$$\begin{cases} \dot{\mathbf{y}}(t) = g(t, \mathbf{y}(t)) \\ \mathbf{y}(0) \in [\mathbf{y}_0]^* \subseteq \mathbb{R}^n. \end{cases} \tag{6}$$

by keeping in mind the fact that $\mathbf{y}(t; [\mathbf{y}_0]^*) \subseteq \mathbf{y}(t; [\mathbf{y}_0]), \forall t$. We propagate along the discretization the contraction of the initial state with the help of a Runge-Kutta based contractor as proposed in [3]. The resulting reachable tube is showed in Figure 7.

One simulation followed by several propagation is much faster than several simulations [2] due to the economy of the dynamic discretization re-computation and the conservation of the first phase results (which is the more time-consuming step).

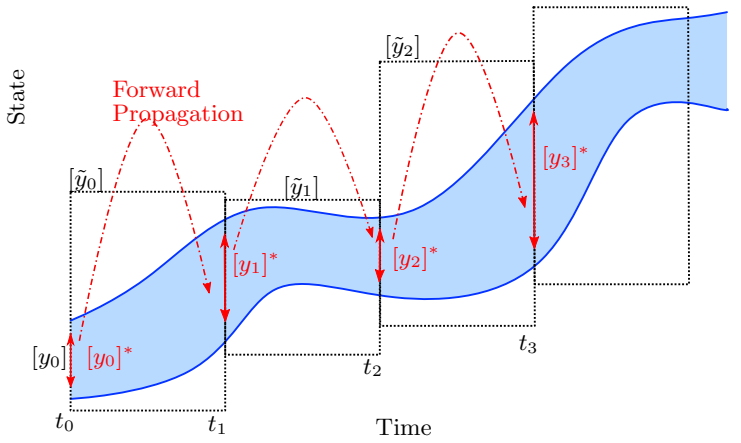


Figure 7: Continuous reachable tube (in blue) and propagation (in red) of a new initial condition $[y_0]^*$ (with $[y_0]^* \subset [y_0]$).

5.2 Potential clouds

The formalism of clouds has been proposed in [18] to handle uncertainties. With clouds, uncertainties are seen as safety constraints. The potential clouds can be exploited for high dimensional and non-formalised uncertainties, as in [10].

From [18], the formal definition of a cloud over a set $\mathcal{M}$ is a mapping $\mathbf{x}$ that associates with each $\xi \in \mathcal{M}$ a nonempty, closed and bounded interval $\mathbf{x}(\xi)$ such that

$$]0,1[\subseteq \bigcup_{\xi \in \mathcal{M}} \mathbf{x}(\xi) \subseteq [0,1]. \tag{7}$$

$\mathbf{x}(\xi) = [\underline{x}(\xi), \overline{x}(\xi)]$ is called the level of ξ in the cloud $\mathbf{x}$. A cloud and an α-cut are illustrated in Figure 8.

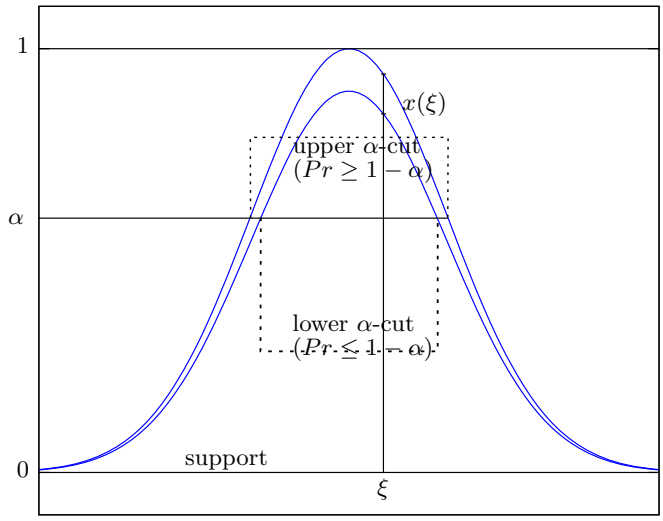


Figure 8: A cloud over $\mathbb{R}$ with an α-cut at $\alpha = 0.6$.

In the particular case where a cloud $\mathbf{x}$ is defined by a potential function $V : \mathcal{M} \mapsto \mathbb{R}$ (bounded below) such that

$$\mathbf{x}(\xi) := [Pr(V(x) > V(\xi)), Pr(V(x) \geq V(\xi))] \tag{8}$$

that can be written $\mathbf{x}(\xi) := [\underline{\alpha}(V(\xi)), \overline{\alpha}(V(\xi))]$ where α-cuts are level sets of V. In general, the probabilities are not known and then $\underline{\alpha}, \overline{\alpha} : \mathbb{R} \mapsto [0,1]$ are assumed (we call these functions potential level maps), and $\mathbf{x}$ is called a potential cloud.

Regarding the definition, determining a cloud is similar to compute a lower and an upper bounds of the confidence regions for different confidence levels (a discretization from 100% to 10% for example) with the help of Cumulative Distribution Functions (CDFs). Considering multivariate problems, a potential function is used to map a multivariate random variable to a univariate one. In the following, we consider only the upper bound of the confidence regions because our main interest concerns safety.

Proposition 2. *The collection of the reachable sets* $[\mathbf{y}(\tau; [\mathbf{y}_0]^i)], i = 1 \dots m$, *with* $[\mathbf{y}_0]^i = Cbc([\mathbf{y}_0] | f_X, cc_i)$ *is a special case of potential clouds applied to reachability.*

In order to illustrate this capability, a one-dimensional example is studied (multivariate problems can be considered by the help of a potential function).

Example 2. The following IVP is considered:

$$\begin{cases} \dot{y}(t) = y\ cos(y) \\ y(0) \in [0, 2] \end{cases} \tag{9}$$

The initial condition follows a normal distribution, such that

$$f_X(x|\mu, \sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

with $\mu = 1.0$ and $\sigma = 1.0$ (same as before, recalled for clarity). The system described by Equation (9) has to reach a goal given by the interval $[1.5, 1.6]$ at $t = 5$, *i.e.*, $[y(5; [y_0])] \subset [1.5, 1.6]$. Our objective is to prove that the system reaches its goal with respect to a confidence level given on the initial condition. Equation (9) is solved with validated simulation as described in [3]. The obtained reachable set at $t = 5$ is $[-276.986, 279.276]$. The goal is then unfulfilled. With successive tests from a confidence of 90% to 10% (from pessimistic to optimistic), the confidence-based contractor is applied followed by a forward propagation along the validated simulation (as presented in Section 5.1.2).

The reachable sets of problems described by:

$$\begin{cases} \dot{y}(t) = y\ cos(y) \\ y(0) \in [y_0]^i = Cbc([0, 2]|f_X, cc_i) \end{cases} \tag{10}$$

are gathered in Table 2.

Table 2: Confidence levels, contracted initial intervals and reachable sets.

Confidence	Initial	Final
90%	$[0, 2]$	$[-276.986, 279.276]$
80%	$[0, 2]$	$[-276.986, 279.276]$
70%	$[0, 2]$	$[-276.986, 279.276]$
60%	$[0.1581, 1.8418]$	$[-189.871, 192.408]$
50%	$[0.3256, 1.6743]$	$[1.56281, 1.57764]$
40%	$[0.4756, 1.5243]$	$[1.56871, 1.57205]$
30%	$[0.6146, 1.3853]$	$[1.56964, 1.57119]$
20%	$[0.7466, 1.2533]$	$[1.57004, 1.57082]$
10%	$[0.8743, 1.1256]$	$[1.57027, 1.57061]$

It shows that:

- Goal can be proved to be achieved after 50%, *i.e.*, we have one chance out of two that the initial state is in the contracted interval;

- Confidence contractor has effect from 60%;
- Contraction-propagation approach reduces computation time (106 seconds for ten simulations versus 87 seconds for one simulation and nine propagations).

The trajectories for 50% and 60% are given in Figure 9. A potential cloud composed by the reachable sets for different confidence levels is depicted in Figure 10.

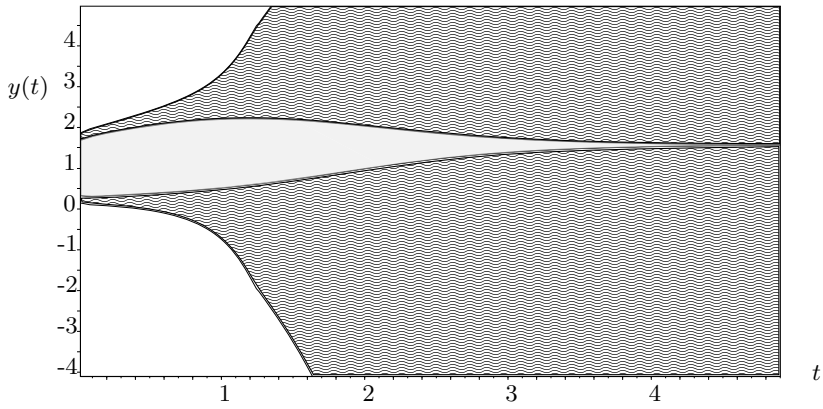


Figure 9: Validated trajectories for 60% (in grey waves) and for 50% (in light grey) confidence contraction.

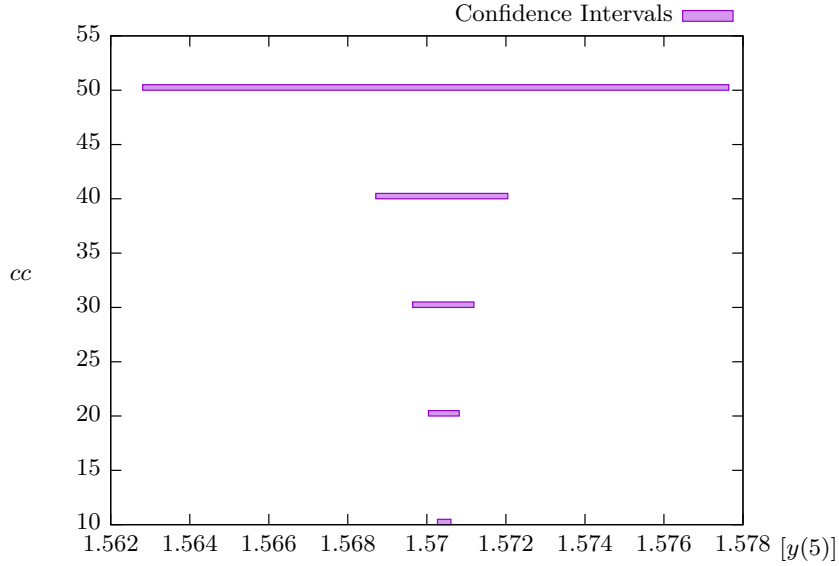


Figure 10: Reachable set for different confidence levels presented as a potential cloud.

5.3 Comparison with trapezoidal fuzzy numbers

In [13], authors propose several interesting examples. In particular, a two-state bioreactor model with uncertain parameters is studied. It consists in a well-mixed bioreactor in which biomass of a single organism is produced with respect to a single limiting substrate. The continuous dynamics is described by:

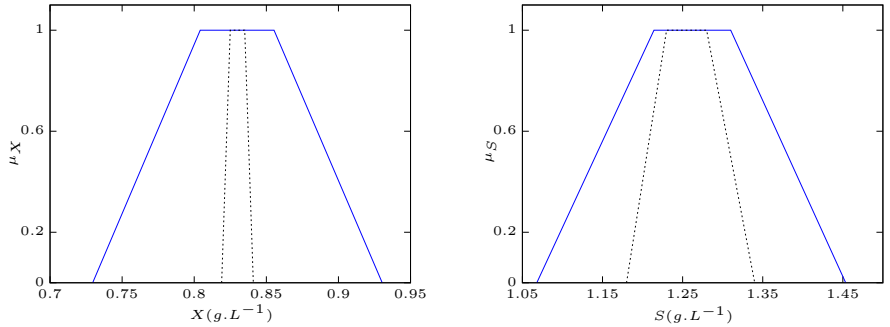
$$\begin{cases} \dot{X} = (\mu - aD)X \\ \dot{S} = D(S_f - S) - k\mu X, \end{cases} \tag{11}$$

where X and S are the concentrations of biomass and substrate. Here μ is a function of S describing the specific growth rate of biomass given by $\mu = \frac{\mu_{max} S}{K_s + S}$ (in the case of monod kinetics), D is the dilution rate, a the biomass washout fraction, k the inverse yield coefficient, and S_f the substrate feed concentration. The parameter values are: $a = 0.5$, $k = 10.53$, $S_f = 5.7$g/L and $\mu_{max} = 1.2\text{h}^{-1}$. The initial states are: $X(0) = 0.829$g/L and $S(0) = 0.8$g/L. The two parameters D and K_s are treated as uncertain and represented by symmetric trapezoidal fuzzy numbers in [13]. This kind of fuzzy number is described by its support and core intervals. For D, support and core are $[0.35, 0.37]\text{h}^{-1}$ and $[0.35667, 0.36333]\text{h}^{-1}$ (respectively), and for K_s they are $[6.8, 7.2]$g/L and $[6.93333, 7.06667]$g/L (respectively). We consider support and core intervals as confidence intervals (with a confidence level at 100% for the support and $\epsilon << 1\%$ for the core). Simulations are performed till $t = 8$h, and the final states are used to rebuild the trapezoidal fuzzy numbers as given in Figure 11.

Discussion: The example from [13] being different than the purpose treated in this paper, the capabilities of confidence based contractor are not really exploited. However, the comparison is interesting. First of all, it is important to notice that the reachability method used in [13] is more efficient than the one used here (VSPODE is dedicated to handle uncertain parameters), and thus the reachable tube seems thinner. However, the way that discretization with α-cuts is performed in [13] leads to consider as constant a parameter during all the integration process while our approach allows all the possible values at each instant. Based on this observation, the method proposed in [13] is probably more optimistic than the confidence based propagation, as depicted in Figure 11. To conclude this discussion, our method is not dedicated to consider correlation between parameters. Furthermore, this point is not clearly treated in [13].

5.4 Inverse problem

The inverse problem consisting in finding the confidence level such that a constraint on reachable set can be proved is interesting. For example, a requirement in term of confidence level on the position of a robot is directly connected to the quality of the measurement devices of the robot. Therefore, if a certain sensor quality is required by the system to validate a given property (*e.g.* safety), it is important to be able to bound the required confidence level.



(a) Biomass concentration

(b) Substrate concentration

Figure 11: Results for Example (11) at $t = 8$h (in blue: the concentrations w.r.t. confidence intervals for parameters (support / core); in dashed lines: results from [13]).

A contractor approach associated with a forward-backward propagation (the specific contractor programming approach presented in [2] is exploited) can be used to compute the initial condition $[y_0]$ such that $[y(5; [y_0])] \subset [1.5, 1.6]$. After this preliminary step, the confidence coefficient is computed with: $cc = Pr(x \in [y_0]) = \int_{[y_0]} f_X(x)\, dx$.
The algorithm used consists of four steps:

1. Validated simulation with $[y_0]$ till t_{end} to obtain $[y(t_{end}; [y_0])]$
2. Intersection of $[y(t_{end}; [y_0])]$ with the goal: $[y(t_{end})] = [y(t_{end}; [y_0])] \cap [y_{goal}]$
3. Backward propagation from t_{end} to $t = 0$ to obtain $[y(0; [y(t_{end})])]$
4. Computation of the confidence coefficient: $Pr(x \in [y(0; [y(t_{end})])])$

On the previous example, the backward propagation provides $[y_0] = [0.200625, 1.79937]$ which gives a confidence coefficient $cc = [0.576115, 0.576464]$, *i.e.*, a confidence level of around 57.62%.

6 Conclusion and future works

In this paper, a novel contractor based on confidence level is proposed. It aims to reduce the pessimism of the interval approach by considering the probability density of the variables. We showed on a simple running example that our method provides results corresponding to the theory on normal density. An application to the reachability of ordinary differential equations has been proposed. The confidence-based contractor has been associated to a validated integration method to compute reachable sets for different values of confidence level. A propagation procedure allows one to propagate the contraction on initial state to the reachable set. We

proposed to depict the different reachable sets under the form of a potential cloud. This method was tested on an example. Finally, the inverse problem consisting in computing the confidence coefficient such that a constraint on the reachable set is fulfilled has been solved.

As future work, even if no limits exist on the proposed approach in terms of problem size or considered distribution, we should apply it to more complex examples and different probability densities. Moreover, it could be interesting to exploit it in the field of robotics considering the distribution of the sensors in a control synthesis, safety verification or path planning.

References

[1] Abdallah, Fahed, Gning, Amadou, and Bonnifait, Philippe. Box particle filtering for nonlinear state estimation using interval analysis. *Automatica*, 44(3):807 – 815, 2008. DOI: 10.1016/j.automatica.2007.07.024.

[2] Alexandre dit Sandretto, Julien and Chapoutot, Alexandre. Contraction, propagation and bisection on a validated simulation of ODE. In *Small Workshop on Interval Methods*, 2016.

[3] Alexandre dit Sandretto, Julien and Chapoutot, Alexandre. Validated explicit and implicit Runge-Kutta methods. *Reliable Computing*, 22:79–103, 2016.

[4] Alexandre dit Sandretto, Julien and Wan, Jian. Reachability analysis of nonlinear odes using polytopic based validated runge-kutta. In Potapov, Igor and Reynier, Pierre-Alain, editors, *Reachability Problems*, pages 1–14. Springer International Publishing, 2018. DOI: 10.1007/978-3-030-00250-3_1.

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

[6] Destercke, Sébastien, Dubois, Didier, and Chojnacki, Eric. Transforming probability intervals into other uncertainty models. In *EUSFLAT 2007 proceedings*, volume 2, pages 367–373, 2007.

[7] Dubois, Didier, Kerre, Etienne, Mesiar, Radko, and Prade, Henri. *Fuzzy Interval Analysis*. In *Fundamentals of Fuzzy Sets*, pages 483–581. Springer US, Boston, MA, 2000. DOI: 10.1007/978-1-4615-4429-6_11.

[8] Ferson, Scott, Kreinovich, Vladik, Grinzburg, Lev, Myers, Davis, and Sentz, Kari. Constructing probability boxes and dempster-shafer structures. Technical report, Sandia National Lab.(SNL-NM), Albuquerque, NM (United States), 2015.

[9] Fuchs, Martin. Cloud based design optimization. In *IFSA/EUSFLAT Conf.*, pages 345–350, 2009.

[10] Fuchs, Martin and Neumaier, Arnold. Autonomous robust design optimisation with potential clouds. *Int. J. Reliability and Safety*, 3:23–34, 01 2009. DOI: 10.1504/IJRS.2009.026833.

[11] Jaulin, Luc, Kieffer, Michel, Didrit, Olivier, and Walter, Eric. *Applied Interval Analysis*. Springer, 2001. DOI: 10.1007/978-1-4471-0249-6.

[12] Lohner, Rudolf J. Enclosing the solutions of ordinary initial and boundary value problems. *Computer Arithmetic*, page 255–286, 1987.

[13] Măceş, D Andrei and Stadtherr, Mark A. Computing fuzzy trajectories for nonlinear dynamic systems. *Computers & chemical engineering*, 52:10–25, 2013. DOI: 10.1016/j.compchemeng.2012.11.008.

[14] Mendenhall, William, Beaver, Robert J, and Beaver, Barbara M. *Introduction to Probability and Statistics*. Cengage Learning, 2012.

[15] Moore, Ramon E. *Interval Analysis*. Series in Automatic Computation. Prentice Hall, 1966.

[16] Moore, Ramon E., Kearfott, R Baker, and Cloud, Michael J. *Introduction to Interval Analysis*. Siam, 2009. DOI: 10.1137/1.9780898717716.

[17] 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.

[18] Neumaier, Arnold. Clouds, fuzzy sets, and probability intervals. *Reliable computing*, 10(4):249–272, 2004. DOI: 10.1023/B:REOM.0000032114.08705.cd.

[19] Neyman, Jerzy. Outline of a theory of statistical estimation based on the classical theory of probability. *Phil. Trans. R. Soc. Lond. A*, 236(767):333–380, 1937. DOI: 10.1098/rsta.1937.0005.

[20] Rauh, A., Hofer, E. P., and Auer, E. Valencia-ivp: A comparison with other initial value problem solvers. In *12th GAMM - IMACS International Symposium on Scientific Computing, Computer Arithmetic and Validated Numerics (SCAN 2006)*, pages 36–36, 2006. DOI: 10.1109/SCAN.2006.47.

[21] Williamson, Robert C. and Downs, Tom. Probabilistic arithmetic. I. Numerical methods for calculating convolutions and dependency bounds. *International Journal of Approximate Reasoning*, 4(2):89 – 158, 1990. DOI: 10.1016/0888-613X(90)90022-T.

[22] Zadeh, Lotfi A. Fuzzy sets. *Information and control*, 8(3):338–353, 1965. DOI: 10.1016/S0019-9958(65)90241-X.

Acta Cybernetica **25** (2021) 69–84.

Identification of Multi-Faults in GNSS Signals using RSIVIA under Dual Constellation*

Shuchen Liu[ab], Jan-Jöran Gehrt[ac], Dirk Abel[ad], and René Zweigel[ae]

Abstract

This publication presents the development of integrity monitoring and fault detection and exclusion (FDE) of pseudorange measurements, which are used to aid a tightly-coupled navigation filter. This filter is based on an inertial measurement unit (IMU) and is aided by signals of the global navigation satellite system (GNSS). Particularly, the GNSS signals include global positioning system (GPS) and Galileo. By using GNSS signals, navigation systems suffer from signal interferences resulting in large pseudorange errors. Further, a higher number of satellites with dual-constellation increases the possibility that satellite observations contain multiple faults. In order to ensure integrity and accuracy of the filter solution, it is crucial to provide sufficient fault-free GNSS measurements for the navigation filter. For this purpose, a new hybrid strategy is applied, combining conventional receiver autonomous integrity monitoring (RAIM) and innovative robust set inversion via interval analysis (RSIVIA). To further improve the performance, as well as the computational efficiency of the algorithm, the estimated velocity and its variance from the navigation filter is used to reduce the size of the RSIVIA initial box. The designed approach is evaluated with recorded data from an extensive real-world measurement campaign, which has been carried out in GATE Berchtesgaden, Germany. In GATE, up to six Galileo satellites in orbit can be simulated. Further, the signals of simulated Galileo satellites can be manipulated to provide faulty GNSS measurements, such that the fault detection and identification (FDI) capability can be validated. The results show that the designed approach is able to identify the generated faulty GNSS observables correctly and improve the accuracy of the navigation solution. Compared with traditional RSIVIA, the designed new approach provides a more timely fault identification and is computationally more efficient.

*The development of current publication is part of the joint research project GALILEOnautic 2 (grant number 50NA1808), which is supported by the German Federal Ministry for Economic Affairs and Energy. Basis for the support is a decision by the German Bundestag.

[a]Institute of Automatic Control, RWTH Aachen University, Germany

[b]E-mail: s.liu@irt.rwth-aachen.de, ORCID: https://orcid.org/0000-0003-4685-491X

[c]E-mail: j.gehrt@irt.rwth-aachen.de, ORCID: https://orcid.org/0000-0003-4348-2110

[d]E-mail: d.abel@irt.rwth-aachen.de, ORCID: https://orcid.org/0000-0003-0286-3654

[e]E-mail: r.zweigel@irt.rwth-aachen.de, ORCID: https://orcid.org/0000-0003-2440-2138

DOI: 10.14232/actacyb.285315

Keywords: RSIVIA, RAIM, GNSS, fault detection and identification, Kalman filter

1 Introduction

As described in the market report from the European Global Navigation Satellite Systems Agency (GSA) [5], satellite-based navigation will substantially contribute to the future innovation of self-driving vehicles. In autonomous applications, especially in safety-critical scenarios, a false estimation of vehicle states can result in catastrophic accidents. Therefore, a reliable navigation solution with high integrity is required. To maintain the integrity of a satellite-based navigation system, the faulty GNSS observations caused by signal interferences and other possible reasons shall be detected, identified and excluded. Ever since the operation of open service of the newly developed EU satellite navigation system Galileo, the combination of GPS and Galileo provides more available satellites in view for the modern navigation systems. However, a higher number of satellites also increases the possibility that satellite observations contain a fault or even multi-faults. Therefore, identification of multi-faults becomes a crucial and challenging task to maintain the integrity of GNSS-based navigation systems.

The previous work [10] presents the development of a fault detection and exclusion (FDE) algorithm of GNSS measurements. The approach operates as an extension of a tightly-coupled navigation filter, which integrates the measurements from GNSS, an inertial measurement unit (IMU) and a Doppler velocity log (DVL) [6]. In [10], FDE bases on the receiver autonomous integrity monitoring (RAIM) approach with parity space [12], which is a pure statistical method. RAIM predicts pseudorange residuals, which are based on the estimated reference vehicle state using least square method, and uses the residuals to detect and identify pseudorange faults. Since RAIM is a pure statistic method and based on single fault assumption, it might not always be adequate, if multiple measurements are faulty. This can be observed in [11]. This work concentrates on multi-fault identification, when the conventional statistic based approach cannot certainly provide a correct identification solution.

In recent years, an alternative localization method, set inversion via interval analysis (SIVIA), is developed under such concern. SIVIA estimates a trust region of the state space fulfilling a predefined confidence level. The basic operations of interval analysis are introduced in [7] and applied to realize robot localization in [8]. Further, [13] shows an example of integrating velocity information from DVL to compute the guaranteed robot trajectories using interval analysis. With respect of GNSS application, robust SIVIA (RSIVIA) approach is applied for satellite positioning in [3] [4], which allows to estimate the trust region of antenna position with existing erroneous pseudorange measurements. Hereby, it is possible to identify outliers in the GNSS observations by checking the compatibility of each GNSS

measurement and the estimated trust region. The main drawback of this approach is its computational load, because RSIVIA begins with an initial guess of an arbitrary big box, bisects it into small boxes and operates on them separately and iteratively. In addition, RSIVIA usually runs at a lower rate (1 Hz in [4]) than GNSS observations (10 Hz in current navigation filter), such that the timely fault identification can not be guaranteed. This can result in corrupted state estimation, when there is an increasing pseudorange error [14].

The present publication proposes a FDE scheme to benefit the advantages of RAIM and RSIVIA and compensates the disadvantages of them, which is illustrated in Fig. 1. Once a new set of GNSS measurements is available, fault detection and identification (FDI) of RAIM are carried out iteratively. Eventually, a fault alert is generated from RAIM and used as a trigger for RSIVIA, if RAIM still detects a fault but is not capable to identify it. This reduces the computational load and enables a more timely fault identification. When RSIVIA is triggered for the further FDI task, it is executed in an iterative process, which refers to RSIVIA GNSS update in Fig. 1: it starts with the assumption that no fault exists in the observed measurement space. Whenever an empty trust region is returned, RSIVIA assumes one more fault existing in the measurements. This iterative process continues until a non-empty trust region is estimated. The rest of the faulty measurements are identified by checking the consistency of the measurements with the resulting trust region.

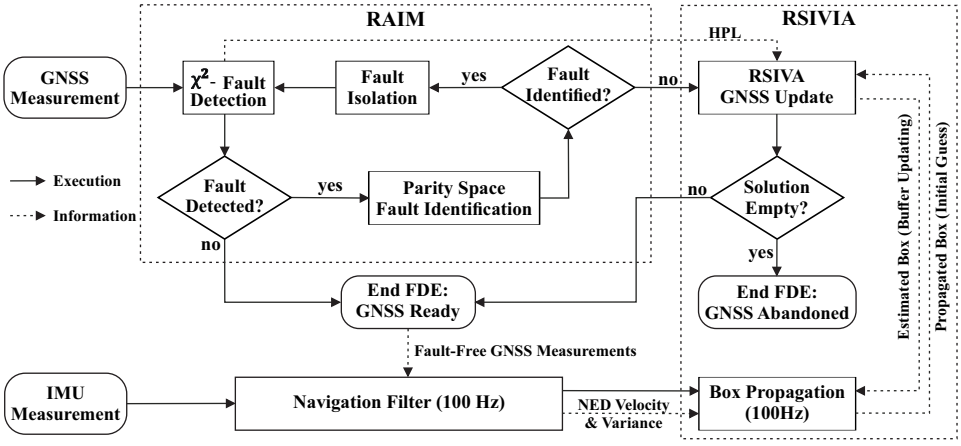


Figure 1: Scheme of GNSS FDE involving RAIM and RSIVIA

To reduce the computational load, RSIVIA is initialized with an arbitrary big box and the first trust region is estimated only with GNSS measurements. After that, the estimated velocity information and its variance from the navigation filter are used to propagate the trust region from last step. In this way, further RSIVIA steps start with the propagated trust box as initial guess, whose size is much smaller than an arbitrary big box.

Furthermore, output from the conventional RAIM is used to aid RSIVIA. Estimated horizontal protection level (HPL) is used for parameterizing minimal error bound, whose deterministic calculation is not given in [3] [4] for GNSS application.

The designed approach is evaluated in post-processing environment with the recorded data, with respect of correctness of FDI, accuracy improvement of the navigation solution and reduce of computational load. In order to reproduce the test scenario in a post-processing environment, all sensors and GNSS correction data are recorded in real-world tests at Galileo test and development environment (GATE) in Berchtesgaden, Germany. To validate the FDI functionality, the fault identification results are compared with report of generated faulty Galileo measurements from IFEN GmbH.

The paper is structured as follows: First, the state vector of navigation filter and RAIM are introduced briefly. Then, the method of RSIVIA using GNSS measurements is described as the basic. At the end of methodology part, the practical integration of velocity outputs from navigation filter and RAIM is given. In the experimental validation part, the measurement setup and target hardware are described, experimental results are evaluated and discussed afterwards. Finally, the last section draws the conclusion and provides an outlook for future developments of integrity monitoring within inertial navigation system.

2 Methodology

2.1 Navigation Filter

The basic concepts and equations of a tightly-coupled navigation filter are introduced in [6] [9]. This section concentrates on introducing the filter outputs. In total, 18 states are predicted within a strap-down algorithm using the measurements from 3D accelerometer and 3D gyroscope,

$$\boldsymbol{x} = [\boldsymbol{p}^e_{eb}\ \boldsymbol{v}^n_{eb}\ \boldsymbol{q}^n_b\ \boldsymbol{b}_a\ \boldsymbol{b}_g\ c_b\ c_d]^T. \tag{1}$$

The state vector $\boldsymbol{x}$ contains position $\boldsymbol{p}^e_{eb}$ of the IMU body-frame origin in Earth-Centered-Earth-Fixed (ECEF) coordinates (3×1) and the velocity of body-frame origin $\boldsymbol{v}^n_{eb}$ (3×1), which is in navigation frame North-East-Down (NED) coordinates with respect to ECEF frame. Furthermore, a quaternion $\boldsymbol{q}^n_b$ for alignment of body frame and NED frame (4×1), accelerometer bias $\boldsymbol{b}_a$ (3×1) and gyroscope bias $\boldsymbol{b}_g$ (3×1) are estimated. Additionally, a tightly coupled system needs to estimate receiver clock bias c_b and drift c_d for correction of pseudo- and deltaranges. The process and measurement model used for Kalman update is given in [6] [9]. It should be stressed that the velocity information $\boldsymbol{v}^n_{eb}$ is given in NED coordinates, which is one important reason for choosing NED coordinates as the operation navigation frame for RSIVIA. The other reason is that using NED coordinates makes it easier to distinguish between horizontal and vertical components. Focusing on horizontal components is important for most autonomous applications.

2.2 RAIM with Parity Space

The previous work [10] presents the development of a FDE extension based on RAIM. The necessary equations of residual-based RAIM for pseudorange fault detection are given in [1] [10]. Further, using parity space based RAIM for fault identification is introduced in [12] [10]. Hereby, only necessary theory is explained, helping to understand the integration of RAIM in Sec. 2.5.

In General, RAIM uses the pseudorange residuals to detect and identify faulty GNSS measurements. Pseudoranges are predicted, based on the estimated reference vehicle states. The pseudorange residuals are calculated as the difference between measured and predicted pseudoranges. According to statistics, with ν independent standard normal random variables, the sum of their squares satisfies chi-squared distribution with ν degree of freedom (DOF). Assuming that the pseudorange measurement noise satisfies the white mean Gaussian distribution with various standard deviation. After using pseudorange measurements to estimate the 4 unknowns by using least square approach, the normalized predicted residual of pseudoranges ν shall satisfy $N-4$ DOF chi-squared distribution. N is the number of available pasudorange measurements. Otherwise, RAIM shall declare that an error occurs. The fault identification is done iteratively with the help of parity space using Bayes Rule, assuming all satellites having the same prior probability of being faulty [12][10]. It should be noticed that RAIM also estimates HPL, which is a function of pseudorange variances, the geometric satellite constellation and the predefined parameters, i.e. false alarm rate and missed detection probability. The estimated HPL is used in Sec. 2.5 to parametrize the minimal error bound of RSIVIA.

2.3 RSIVIA with Pseudorange Measurements

2.3.1 Interval Analysis Basics

Interval analysis (also called interval computation) is the operation on intervals instead of algebraic operation on numbers, although the basic operators are the same as in algebraic operation: $+$, $-$, $\times$, $\div$, sin, tan, exp. The computation of intervals is defined in [7], as follows

$$[a]\Diamond[b] = [\{a\Diamond b \in \mathbb{R} | a \in [a], b \in [b]\}], \tag{2}$$

where $[a]$ and $[b]$ are intervals. Further, the high dimensional interval is defined as a box. $\Diamond$ can be any of the algebraic operations listed above. By applying interval analysis on satellite-based navigation, the pseudorange measurement equation is expressed as

$$[\rho_i] = \sqrt{([x^s_{n,i}] - [x^a_n])^2 + ([x^s_{e,i}] - [x^a_e])^2 + ([x^s_{d,i}] - [x^a_d])^2} + [c_b], \quad \forall i \in \{1, 2, ..., N\}, \tag{3}$$

where $\boldsymbol{x}^s = [x^s_{n,i}, x^s_{e,i}, x^s_{d,i}]^T$ is the i^{th} satellite position and $\boldsymbol{x}^a = [x^a_n, x^a_e, x^a_d, c_b]^T$ is the antenna position and receiver clock bias, both in NED frame. N is the number of available pseudorange measurements.

2.3.2 Measurement Bounding

In the context of satellite pseudorange-based navigation, the measurement vector is $\tilde{\boldsymbol{\rho}} = [\tilde{\rho}_1, \cdots, \tilde{\rho}_i, \cdots, \tilde{\rho}_N]$. Based on the measured pseudoranges $\tilde{\boldsymbol{\rho}}$, the interval of them $[\boldsymbol{\rho}]$ should be estimated. In [4], an approach is proposed, estimating the lower and upper bound of the measurements, tolerating faulty measurements. Allowing a certain number of faulty measurements to estimate the trust box is defined as q-relax, where q is the number of tolerated faulty measurements.

Consider a set of N available measurements with the condition of q-relax, it means that at least $N-q$ measurements are required to be fault free. It is assumed that the probability of a measurement being faulty satisfies binomial distribution. The probability of q-relax condition satisfied is calculated as

$$P(n_{ff} \geq N-q) = \sum_{k=N-q}^{N} P(n_{ff}=k) = \sum_{k=N-q}^{N} \frac{N!}{k!(N-k)!} p_{ff}^{k} (1-p_{ff})^{N-k}, \quad (4)$$

where the confidence level $P(n_{ff} \geq N-q)$ is predefined. Therefore, p_{ff} can be estimated as the only unknown in Eq. (4), which is the probability of each satellite being faulty free. The probability density function of a measurement noise is known as $f(e)$, which is practically assumed to be a white Gaussian distribution. In this way, for each measurement p_{ff} is calculated as

$$p_{ff} = P(\rho \in [\widetilde{\rho}+a, \widetilde{\rho}+b]) = \int_{a}^{b} f(e)de. \quad (5)$$

With p_{ff} estimated from Eq. (4), the lower and upper bound can be calculated by minimizing the width of the interval $[a, b]$.

2.3.3 The RSIVIA Process with GNSS Measurements

To estimate the trust box of the state vector $[\boldsymbol{x}^a]$, RSIVIA starts with the feasible initial guess $[\boldsymbol{x}_0^a]$, which allows to be arbitrarily big and guarantees the true solution of $\boldsymbol{x}^a$ inside it. RSIVIA attempts to reduce the size of the initial guess with a contractor $\mathcal{C}$. A contractor is an operator $\mathbb{IR}^n \rightarrow \mathbb{IR}^n$ associated to a constraint (in our case Eq. (3)), which returns a box $\mathcal{C}[\boldsymbol{x}] \subseteq [\boldsymbol{x}]$ without losing any vector consistent with the constraint [13]. If the size of the operated box cannot be further reduced by a contractor , it will be bisected into two small boxes and the contractor operation will be repeated for all small boxes remained. This process ends, until the width of all remained boxes is smaller than a predefined error bound ϵ. The detailed design of the RSIVIA process in GNSS applications is given in [3] [4], which includes forward and backward contractor using constraints given in Eq. (3). This RSIVIA operation is summarized as Line 4 in Alg. 1.

Alg. 1 gives the whole process of a bounding box update, when a new GNSS measurement is available. This process starts with a fault-free assumption ($q=0$) and attempts to estimate the trust region with an increasing q. This operation is summarized as GNSS update. The resulting box $[\boldsymbol{x}^a]$ is applied in Eq. (3) to predict

Algorithm 1 GNSS update with pseudorange measurements

Function gnss_update(in:$[\boldsymbol{x}_0^a], \boldsymbol{x}^s, \boldsymbol{\rho}, \epsilon$, out:$[\boldsymbol{x}^a], \boldsymbol{f}$)

Initialization: $[\boldsymbol{x}^a] \leftarrow \emptyset$, $q \leftarrow 0$, get number of satellites N
while ($[\boldsymbol{x}^a] = \emptyset$ & $N - q \geq 4$) **do**
  $([\boldsymbol{x}^s], [\boldsymbol{\rho}]) \leftarrow$ get_bounds$(\boldsymbol{x}^s, \boldsymbol{\rho}, q)$ *Sec. 2.3.2*
  $[\boldsymbol{x}^a] \leftarrow$ rsivia(in:$[\boldsymbol{x}_0^a], [\boldsymbol{x}^s], [\boldsymbol{\rho}], q, \epsilon$)
  if $[\boldsymbol{x}^a] = \emptyset$ **then**
    $q \leftarrow q + 1$
  else
    $\boldsymbol{f} \leftarrow$ check_consistency$([\boldsymbol{x}^a], [\boldsymbol{x}^s], [\boldsymbol{\rho}])$
  end if
end while
return $\boldsymbol{x}^a, \boldsymbol{f}$

the interval of each pseudorange, with the corresponding box of satellite position $[x^s_{n,i}, x^s_{e,i}, x^s_{d,i}]$. The faulty measurement is identified when the predicted pseudorange interval has no intersection with the measured one. This consistency check returns a fault vector $\boldsymbol{f}$ consisting of N elements, which are 0 or 1, representing whether this measurement is fault free.

Still, questions remain in this process, i.e. how should the initial guess $[\boldsymbol{x}_0^a]$ and the minimal acceptable error bound ϵ be chosen, considering both the correctness of fault identification and computational load. These will be answered in Sec. 2.4 and 2.5, respectively.

2.4 Integration of Velocity Information in RSIVIA

In [13], a frame is proposed for guaranteed integration of state equations. An example is given in [13], which uses DVL measurements and differential state constraints to estimate tubes of driven trajectories of an autonomous underwater vehicle (AUV). A tube is defined as an envelope, which encloses an uncertain trajectory. To estimate the tube, a differential tube contractor $\mathcal{C}_{\frac{d}{dt}}$ is applied, which consists of a forward contractor $\mathcal{C}^{\rightarrow}_{\frac{d}{dt}}$ and a backward contractor $\mathcal{C}^{\leftarrow}_{\frac{d}{dt}}$ [13].

Due to several reasons, only forward contraction $\mathcal{C}^{\rightarrow}_{\frac{d}{dt}}$ is applied in the current work, using the output velocity information from the navigation filter in Sec. 2.1. First, the propagated box using velocity information is not the output as in [13]. Instead, the propagated box is only used as the initial box $[\boldsymbol{x}_0^a]$ for GNSS update in Alg. 1. GNSS update still dominates the result of the trust box estimation. Second, in [13] backward contraction is necessary, because only the DVL velocity measurements are available. Without backward contraction, the size of estimated tubes can never be reduced, because it is the integration of the width of measurement error bound. Due to the usage of GNSS measurements, this is not the situation in current publication. In practice, the estimated velocity is with relative narrow variance, which enables an accurate propagation, without losing integrity

of the initial guess $[\boldsymbol{x}_0^a]$. This will be validated in Sec. 3.2.1. Further, the size of the boxes can be strongly reduced by the GNSS update. Considering the difficulty of operating on multi-rate navigation system with GNSS measurement time delay, only forward contractor with the following differential constraint is applied,

$$[\boldsymbol{x}^a](t+dt) = [\boldsymbol{x}_0^a](t+dt) \cap ([\boldsymbol{x}^a](t) + dt \cdot [\dot{\boldsymbol{x}}^a](t)). \tag{6}$$

As mentioned in Sec. 2.1, all the operations are carried out in NED coordinates, such that $\dot{\boldsymbol{x}}^a$ is already given in Eq. (1): $\dot{\boldsymbol{x}}^a = [\ \boldsymbol{v}_{eb}^n\ c_d]^T$. The bounding box $[\dot{\boldsymbol{x}}^a]$ is estimated with the output $\dot{\boldsymbol{x}}^a$ and its estimated covariance from the navigation filter. The detailed contractor design and discretization are given in [13].

In practice, GNSS measurement is updated with the rate of ca. 10 Hz and with a time delay of 50 $\sim$ 300 milliseconds. This time delay is a result of signal processing from the GNSS receiver, after satellite signals are received by the antenna. However, the navigation filter runs at 100 Hz, because a high-rate navigation solution is necessary for autonomous vehicles. Considering the unnegligible delay, a structure is proposed in Fig. 2, which is an example supposing GNSS time delay is 30 milliseconds.

Fig. 2 shows, when no GNSS update is available, the box at $k+1$ step is propagated using $[\boldsymbol{x}_k^a]$ and $[\dot{\boldsymbol{x}}_k^a]$ from last step. Without additional information of the initial box at $k+1$ step, the initial box $[\boldsymbol{x}_{0,k+1}^a]$ is taken as $[-\mathbf{inf}, \mathbf{inf}]$, such that the part after $\cap$ symbol in Eq. 6 dominates the calculation. Once a GNSS update is available, the time of GNSS measurement is estimated by comparing the current time and measured time delay. The propagated box stored in the buffer is located and used as the initial guess $[\boldsymbol{x}_0^a]$ for the GNSS update, which is described in Sec. 2.3.3. After GNSS update, all the boxes until the current epoch are propagated again and the corresponding buffer will be replaced.

Fig. 3 shows an one dimensional example, which illustrates the change of estimated upper and lower bound before and after the GNSS update. In this example, the GNSS measurement is received at $k+4$ epoch (current) with time delay of 3 epochs, such that the GNSS update is carried out in the past (at $k+1$ epoch). The bounds at $k+1$ epoch (black) are used as initial guess for GNSS update and narrower bounds (blue) are estimated. The blue bounds are propagated using the stored velocity information in memory until the current time ($k+4$ epoch).

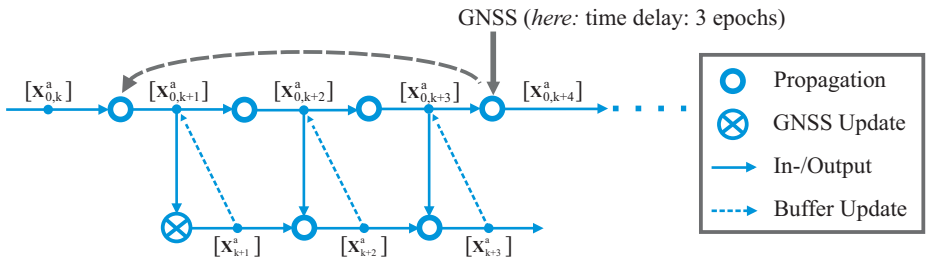


Figure 2: Propagation and GNSS update considering measurement delay

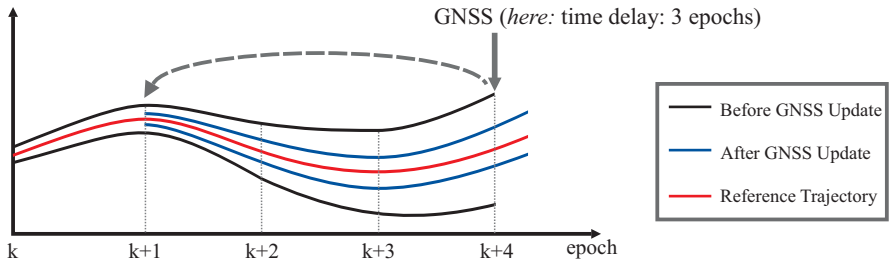


Figure 3: One dimensional (1D) example of integrating velocity information and resulting variance in GNSS update

2.5 Integration of RAIM

By integrating velocity information, RSIVIA does not start with an arbitrary box anymore, which reduces the computational load. This can be further improved by triggering the RSIVIA, when a GNSS update is necessary, instead of executing it by each new GNSS measurement. Whether a GNSS update is necessary, is decided by the RAIM fault detection result.

Algorithm 2 Integration velocity and RAIM information into RSIVIA at k^{th} step

Function estimate_box(in:$\boldsymbol{x}_k^s, \boldsymbol{\rho}_k, t_d, \boldsymbol{x}_{k-1}, \boldsymbol{P}_{k-1}$, out:$[\boldsymbol{x}_k^a], \boldsymbol{f}_k$)

Store $\boldsymbol{x}_{k-1}$ and $\boldsymbol{P}_{k-1}$ into buffer
if New GNSS measurement **then**
  $(f_a, \epsilon) \leftarrow \text{raim}(\boldsymbol{x}_k^s,\ \boldsymbol{\rho}_k,\ \boldsymbol{x}_{k-1})$ *Sec. 2.2*
  if $f_a = true \mid t_c - t_u > t_{u,max}$ **then**
    Estimate GNSS delay steps: $n_d \leftarrow t_d/T_0$
    $([\boldsymbol{x}_{k-n_d}^a], \boldsymbol{f}_k) \leftarrow \text{gnss\_update}([\boldsymbol{x}_{k-n_d}^a], \boldsymbol{x}_k^s, \boldsymbol{\rho}_k, \epsilon)$ *Sec. 2.3.3*
    for $i = k - n_d + 1$ to k **do**
      $([\boldsymbol{x}_i^a]) \leftarrow \mathcal{C}^{\rightarrow}_{\frac{d}{dt}}\ (\boldsymbol{x}_{i-1}, \boldsymbol{P}_{i-1}, [\boldsymbol{x}_{i-1}^a])$ *Sec. 2.4*
    end for
  else
    $([\boldsymbol{x}_k^a]) \leftarrow \mathcal{C}^{\rightarrow}_{\frac{d}{dt}}\ (\boldsymbol{x}_{k-1}, \boldsymbol{P}_{k-1}, [\boldsymbol{x}_{k-1}^a])$
  end if
else
  $([\boldsymbol{x}_k^a]) \leftarrow \mathcal{C}^{\rightarrow}_{\frac{d}{dt}}\ (\boldsymbol{x}_{k-1}, \boldsymbol{P}_{k-1}, [\boldsymbol{x}_{k-1}^a])$
end if
return $[\boldsymbol{x}_k^a]$, $\boldsymbol{f}_k$

Nevertheless, a situation should be avoided that no GNSS update is executed for a long duration, when no fault is detected by RAIM. Therefore, a parameter $t_{u,max}$ is introduced, which defines the maximum duration allowed between two

GNSS updates. The detailed implementation is given in Alg. 2, where t_c and t_u are the current time and time of last GNSS update, respectively. t_d is the GNSS measurement time delay. T_o is the navigation filter sample time.

Furthermore, Alg. 2 proposes a new method of parameterizing the minimum error bound ϵ of RSIVIA, which is a parameter used in Alg. 1. In [8], guaranteed minimum outlier number estimator (GOMNE) is applied, which proposes to reduce the error bound to half of the previous value ($\epsilon \leftarrow \epsilon/2$), when the q-relax increases. Still, the initial value of ϵ needed to be parametrized with this method. [2] discusses the choice of the error bound and the pseudorange quality by comparing different settings of ϵ. However, a deterministic calculation of the error bound ϵ is never given. By introducing the output of RAIM, both satellite constellation and measurement quality are considered (Sec. 2.2).

3 Experimental Validation

In this section, the experimental results of the proposed approach for GNSS integrity monitoring as well as pseudorange measurements FDE are given. The results are divided into two parts: first, the evaluation of the dynamic propagation using the velocity information; second, the validation of the FDE capability, the computational load and the accuracy improvement.

3.1 Measurement Setup and Test Scenario

The sensor data is recorded on a 900 MHz single core rapid control prototyping (RCP) unit, called MicroAutoBox II from dSPACE. GNSS signals are received and decoded by a Septentrio AstRx3 HDC receiver at a rate of 10 Hz. The communication between the receiver and the RCP unit is achieved via serial interface. For inertial measurement, the setup uses a LORD MicroStrain 3DM-GX4-25 industrial-class 9 DOF IMU-sensor, which is connected via serial interface and provides accelerations, angular rates and magnetometer measurements at a rate of 100 Hz. The receiver provides a pulse per second (PPS). Using the PPS, the communication and processing delays of the receiver are measured (see [8]). In order to reproduce the real-world test scenario in a post-processing environment, all sensors and GNSS correction data are recorded.

Fig. 4 shows the bird eye view of the driven trajectory. The experiment is carried out in the so-called "T-Cross" in Berchtesgaden, Germany, because it is the best test track for the visibility of all three base stations from GATE system. It should be noticed that the driven path in this experiment is in open area. Therefore, it is assumed that, except the generated feared events, the measurements from other satellites are fault free. However, the testing scenario is only reproducible by replay of the recorded data in post-processing environment, due to the changing environment in the reality, e.g. position of real satellites, ionosphere delay and troposphere delay. The post-processing environment runs on the MATLAB & Simulink platform on a laptop with an Intel Core i7-7700HQ CPU @ 2.80GHz.

Figure 4: Reference trajectory using a RTK capable GNSS receiver, ©2019 GeoBasis-DE/BKG (©2009), Google

During this drive, two feared events occur, which are range errors intentionally generated by GATE system. These errors are visualized in Fig. 5, which shows the pseudorange residuals of GPS and Galileo signals during this drive. The pseudorange residuals are calculated as the difference between measured and true pseudoranges. Here, true pseudoranges are estimated with the highly accurate RTK reference solution and satellite positions. Fig. 5 shows that from second 108 to 172 and from second 188 to 252, the pseudorange residuals from four Galileo satellites E10, E16, E17, E23 are extremely high. These are the two periods when feared events occur, which is verified by the experiment report from IFEN GmbH. These two periods are marked with the gray dashed lines in the following figures.

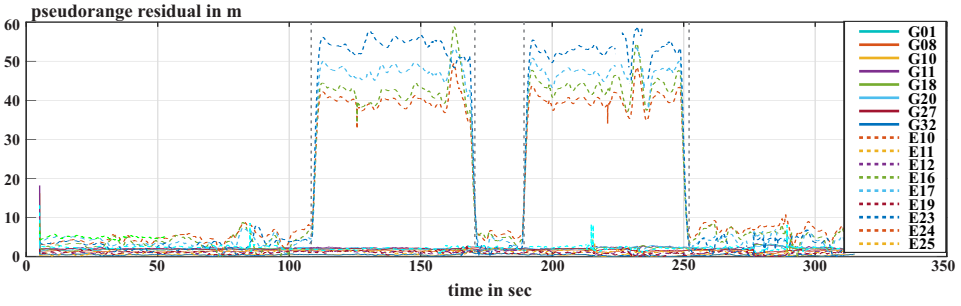


Figure 5: GNSS pseudorange residuals

3.2 Experimental Results

3.2.1 Velocity Integration without GNSS update

In this section, dynamic propagation using velocity information is validated. The box $[\boldsymbol{x}^a]$ is initialized when the first GNSS measurement is available. After that, the trust box is only propagated with the velocity information without further GNSS update. Various parameter settings and potential situations are evaluated and discussed. The three settings are:

- Setting 01: The trust box is propagated only with velocity information without considering their variance, which means the velocity error bound is zero.
- Setting 02: The trust box is propagated in the same way as Setting 01. The faulty GNSS measurements from E10, E16, E17, E23 are manually excluded.
- Setting 03: The trust box is propagated with velocity information and its variance. The error bound of velocity is estimated with the approach from Sec 2.3.2 without measurement relax using the estimated variance.

Experimental results are shown in Fig. 6. The three subfigures are the difference between propagated upper and lower bounds of antenna position in NED coordinates and their reference, respectively. It can be observed that with Setting 01 the lower and upper bounds are evenly distributed around the reference, before the first feared event. During the first feared event, the propagated bounds in north direction shift downwards, which makes the distance between the upper bound and the reference smaller. During the second feared event, the bounds drift further downwards and eventually cross the reference, which makes it an invalid propagation. The reason is that the huge pseudorange error results in faulty velocity estimation of the navigation filter. This can be verified with the Setting 02. After excluding the feared event manually, the velocity propagation is valid during the whole experiment. Introducing the estimated variance of the velocity estimation solves this problem, which is verified with Setting 03. The width of the trust box increases, when no GNSS update is carried out, because the width of the trust box is the integration of velocity error bounds.

It can also be observed that the width of the trust box is no larger than 100 meters, although there is no GNSS update in 300 seconds. This means, the confidential level of the velocity information can be set higher, if periodic GNSS updates are carried out. Because GNSS updates will reduce the size of trust box periodically, this will not introduce much extra computational load.

3.2.2 Identification Correctness and Accuracy Improvement

In [11], a situation is described in the experimental validation chapter, that RAIM with parity space fails to identify all faults in a very short period, while RSIVIA is capable to identify all faulty pseudoranges. The comparison between RAIM and RSIVIA under such condition is given in [11], and therefore, is not repeated in the present publication. In this section, the GNSS update as well as the integration

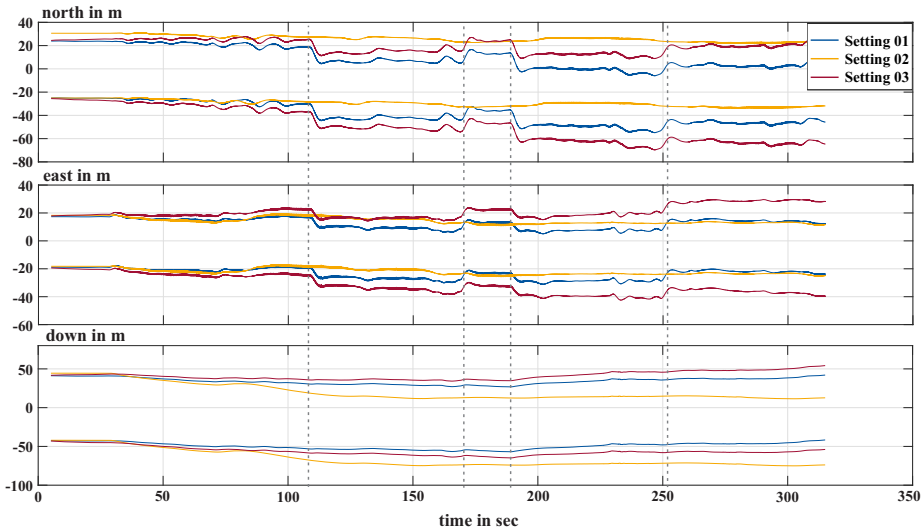


Figure 6: Validation of trust box propagation using velocity information without GNSS update: estimated upper and lower bounds minus the reference

of velocity information and RAIM are evaluated, with respect of identification correctness, the accuracy improvement and the computational load. To achieve the comparison of these, three settings are used in this section:

- Setting 04: GNSS update runs in 1 Hz. Each RSIVIA process starts with an initial guess of an arbitrary big box.
- Setting 05: GNSS update runs in 1 Hz. The trust box is propagated with velocity information (Sec. 2.4). RSIVIA starts with propagated trust box.
- Setting 06: The complete proposed approach in Sec. 2.5 is applied here. In case of no RAIM fault alert, the maximum duration without GNSS update $t_{u,max}$ is 60 seconds.

Fig. 7 shows the experimental results. The first subfigure shows the fault identification result from the three settings and the satellites availability during the test drive. As already shown in Fig. 5, the pseudorange error gradually increases at the beginning of each feared event. This type of pseudorange fault is introduced in [14] as most hazardous fault model for snapshot integrity monitoring, because the state estimation are corrupted before the fault is identified.

This can be verified by Fig. 7. When the first feared even starts (at 108.4 seconds), the erroneous pseudorange is first identified with Setting 06 (at 109.87 seconds), then with Setting 04 (at 110.20 seconds) and finally with Setting 05 (at 110.67 seconds). The reason of the identification delay with Setting 04 and 05 is, that the GNSS update runs in 1 Hz due to the high computational load, while

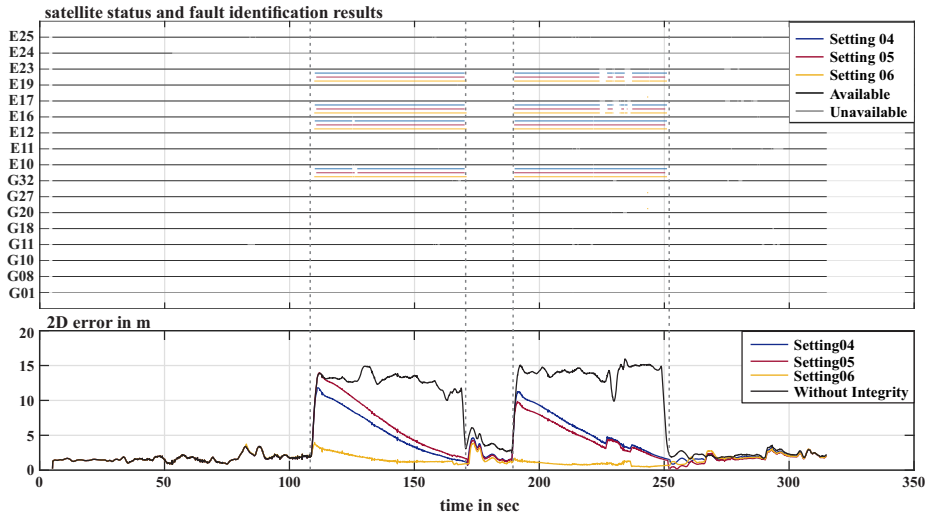


Figure 7: Validation of fault identification and accuracy improvement

the GNSS measurements are received in 10 Hz. Furthermore, the GNSS update is triggered timely by RAIM with Setting 06. Although the time difference of first fault identification among three Settings is tiny, it results in significant difference in state estimation.

The second subfigure shows the 2D error as a measure of state estimation quality. Without FDE, the 2D error remains between 10 meters to 16 meters during the feared events. With Setting 04 and 05, the 2D error increases up to 14 meters and 12 meters, respectively, and converges slowly towards the accuracy without feared events. With Setting 06, the 2D error increases slightly to 3.7 meters, because the fault is identified earlier, and converges quickly to 1 meter.

Finally, the computation time is evaluated. The post-processing takes 81.70 seconds without integrity monitoring, which is the baseline of computation time. Further, the post-processing takes 276.68, 171.51 and 121.89 seconds with Setting 04, 05 and 06, respectively. Considering that GNSS update runs at 1 Hz both with Setting 04 and 05, the number of GNSS updates is the same. By introducing the velocity information to reduce the size of the initial guess, the average computation time of each GNSS update is reduced by 53.94 %, which is very important for the future real-time implementation. In contrast, RAIM reduces the total computation time by reducing the number of GNSS updates, instead of reducing the average computation time. Therefore, RAIM may not improve the real-time computational performance very much. However, introducing RAIM provides a more timely fault identification, which improves the accuracy of navigation solution.

4 Conclusion

This publication presented the development of an integrity system as an extension of a tightly-coupled navigation filter within the joint-project GALILEOnautic 2. The main purpose of the integrity system is FDE of multi-faults in pseudorange measurements, such that a set of fault-free GNSS measurements can be fed into the navigation filter. In this work, a RSIVIA based FDE strategy is proposed, which involves RAIM with parity space and velocity estimation from the navigation filter. With respect to the experimental evaluation, an offline post-processing using data from GATE Berchtesgaden is carried out. In this experiment, multiple feared events are intentionally generated, which are correctly identified by the proposed approach. The measurement campaign evaluations visualize that this approach improves notably both the accuracy and robustness of the navigation filter and reduces significantly the computational load compared to the traditional RSIVIA.

In future works, a real time implementation of the designed approach is aimed. On this basis, an integrity monitoring system for all sensors integrated into the navigation system will be developed, which considers IMU, DVL and GNSS measurements. FDE will be performed on all measurements, which are used by the navigation filter, to enhance the navigation filter reliability.

References

[1] Brown, R.G. A baseline GPS RAIM scheme and a note on the equivalence of three RAIM methods. *Journal of The Insititute of Navigation*, 39(3):301–316, 1992. DOI: 10.1002/j.2161-4296.1992.tb02278.x.

[2] Dbouk, H. and Schön, S. Comparison of different bounding methods for providing GPS integrity information. In *2018 IEEE/ION Position Location and Navigation Symposium (PLANS)*, pages 355–366, Piscataway, NJ, 2018. IEEE. DOI: 10.1109/PLANS.2018.8373401.

[3] Drevelle, V. and Bonnifait, P. High integrity GNSS location zone characterization using interval analysis. *Proceedings of the 22nd International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS 2009)*, pages 2178–2187, 2009.

[4] Drevelle, V. and Bonnifait, P. A set-membership approach for high integrity height-aided satellite positioning. *GPS Solutions*, 15(4):357–368, 2011. DOI: 10.1007/s10291-010-0195-3.

[5] European Global Navigation Satellite System Agency. *GSA GNSS Market Report 2019*. Publications Office of the European Union, 2019.

[6] Gehrt, J.-J., Zweigel, R., Konrad, T., and Abel, D. DVL-aided navigation filter for maritime applications. *11th IFAC Conference on Control Applications in Marine Systems, Robotics and Vehicles (IFAC CAMS 2018)*, pages 418–423, 2018. DOI: 10.1016/j.ifacol.2018.09.451.

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

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

[9] Konrad, T., Breuer, M., Engelhardt, T., and Abel, D. State estimation for a multirotor using tight-coupling of GNSS and inertial navigation. *IFAC-PapersOnLine*, 50(1):11683–11688, 2017. DOI: 10.1016/j.ifacol.2017.08.1684.

[10] Liu, S., Gehrt, J.-J., Abel, D., and Zweigel, R. Dual-constellation aided high integrity and high accuracy navigation filter for maritime applications. *Proceeding of the 2019 International Technical Meeting of The Institute of Navigation (ION ITM)*, pages 762–774, 2019. DOI: 10.33012/2019.16723.

[11] Liu, S., Gehrt, J.-J., Abel, D., and Zweigel, R. Integrity of dual-constellation aided navigation filter in safety-critical maritime applications. *European journal of navigation*, 19(3):10–17, 2019.

[12] Pervan, B., Lawrence, D., Cohen, C., and Parkinson, B. Parity space methods for autonomous fault detection and exclusion using GPS carrier phase. *Position Location and Navigation Symposium*, pages 649–656, 1996. DOI: 10.1109/PLANS.1996.509141.

[13] 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.

[14] Tanil, C., Khanafseh, S., Joerger, M., and Pervan, B. Sequential integrity monitoring for kalman filter innovations-based detectors. In *Proceedings of the 31st International Technical Meeting of The Satellite Division of the Institute of Navigation (ION GNSS+ 2018)*, pages 2440–2455. Institute of Navigation, 2018. DOI: 10.33012/2018.15975.

Acta Cybernetica **25** (2021) 85–99.

Validated Trajectory Tracking using Flatness

Olivier Mullier[ab] and Julien Alexandre dit Sandretto[ac]

Abstract

The problem of a safe trajectory tracking is addressed in this paper. The method consists in using the results of a validated path planner: a set of safe trajectories. It produces the set of controls to apply to remain inside this set of planned trajectories while avoiding static obstacles. The computation is performed using the differential flatness property of many dynamical systems. The method is illustrated in the case of the Dubins car model.

Introduction

In the context of cyber-physical systems, the problem of validated trajectory tracking is addressed. It consists in driving a controlled differiential system from an initial state region to a given target region while avoiding collisions with static obstacles. The general method relies on two steps: *(i)*, a path planning provides one path from the initial state to the final state and, *(ii)*, a trajectory tracker computes the controls that will be given to the actual controlled system to follow this planned path. When the system is critical, it is mandatory to bring certainty on the non violation of the constraints on the system, even when uncertainties on the model and/or the control to be applied occur.

Related Work

A classical way for the path planning is to use the RRT algorithm [11] (Rapidly-exploring Random Tree) and its variants. In [21, 20, 2], it has been successfully adapted to the case of critical systems where guarantees on the result are mandatory and where uncertainties have to be considered. This adaptation of the RRT algorithm uses of a set-membership computation, interval analysis, to no longer produce a trajectory to track but a set of possible trajectories with the property that no static obstacle can collides with the system when in this set. The generic algorithm is then called the box[1] RRT algorithm [21]. The differential flatness

[a]ENSTA PARIS, 828 Boulevard des maréchaux, 91120 Palaiseau, France, E-mail: {lastname}@ensta.fr

[b]ORCID: https://orcid.org/0000-0003-1439-746X

[c]ORCID: https://orcid.org/0000-0002-6185-2480

[1]box is refering to the cartesian product of intervals, see Section 1.1.

DOI: 10.14232/actacyb.285729

of some controlled systems is well known and studied for the control of nonlinear systems (see, *e.g.* [13, 5, 6, 15]). Flatness has already been successfully applied to trajectory tracking in the case of discrete time systems [18].

Contribution

The work here considers that a previous computation of a set of validated path planner is performed and it results in a set of safe trajectories that avoid any static forbidden area. From this set, one particular path is chosen to be tracked, it is done with the computation of a cubic Hermite spline from the system position and the desired trajectory translated in the flat output space. The controls are then produced using an endogenous dynamic feedback using flatness.

This work is presented as follows. Section 1 recalls the notions necessary to the presentation of our work: interval analysis, validated numerical integration of controlled systems, box RRT algorithm and cubic Hermite spline computation. Section 2 contains the main results of our work that is the validated tracking of a trajectory guarranteed to avoid any static obstacle. It is then illustrated in Section 3 with the computation of the controls given to a robot to drive from a set of initial states to a target area using the Dubins car model. Conclusion and discussions ends our work presentation.

1 Preliminaries

This section is dedicated to the preliminary results our work is based on and introduces the notions that are used through this article. Some recalls on interval analysis and validated numerical integration scheme, differential flatness, trajectory planning and cubic Hermite spline are given.

1.1 Interval Analysis

Interval analysis [17] is a method designed to produce outer-approximation of the set of possible values for variables occurring in some computations in a sound manner. Hereafter, an interval is denoted $[x] = [\underline{x}, \overline{x}]$ with $\underline{x} \leqslant \overline{x}$ and the set of intervals is $\mathbb{IR} = \{[x] = [\underline{x}, \overline{x}] \mid \underline{x}, \overline{x} \in \mathbb{R},\ \underline{x} \leqslant \overline{x}\}$. The Cartesian product of intervals $[\mathbf{x}] \in \mathbb{IR}^n$ is a box (through this paper, vectors are represented in bold font). The main result of interval analysis is its fundamental theorem [16] stating that the evaluation of an expression using intervals leads to an outer-approximation of the resulting set of values for this expression whatever the values considered in the intervals.

For a given function $f : \mathbb{R}^n \to \mathbb{R}^m$ and a box $[\mathbf{x}] \subset \mathbb{R}^n$, an interval inclusion function also known as interval extension $[f]$ of f can be defined and an evaluation of $[f]$ over $[\mathbf{x}]$ gives a box $[\mathbf{y}]$ such that $(\forall \mathbf{x} \in [\mathbf{x}])(\exists \mathbf{y} \in [\mathbf{y}])(\mathbf{y} = f(\mathbf{x}))$ or equivalently, the box $[\mathbf{y}]$ contains $\{f(\mathbf{x}) | \mathbf{x} \in [\mathbf{x}]\}$ the range of f over $[x]$. Examples of interval extension are the natural interval extension where interval arithmetic is used to

replace all operations defining a function to its interval counterpart and the mean value extension where the function to be extended is first linearized on a point and a natural extension of the resulting function is applied. The interested reader can refer, for example, to [17, 10] and references therein for more details.

1.2 Validated Numerical Integration of Controlled Systems

In our work, we deal with controlled differential systems of the form

$$\begin{cases} \dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u}) \\ \mathbf{x}(0) \in [\mathbf{x_0}] \end{cases} \tag{1}$$

with $f : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ a smooth function, $\mathbf{x} \in \mathbb{R}^n$ the state vector of the system and $\mathbf{u} \in [\mathbf{u}] \subseteq \mathbb{R}^m$ the control vector. The problem to integrate the controlled system in Eq. (1) is to compute the value of the state vector at a time t, $\mathbf{x}(t, x_0, u)$ with $x_0 \in [x_0]$ and $\mathbf{u} \in [\mathbf{u}]$. By considering $\mathbf{u}$ as constant during the time the system is integrated, the problem in Equation (1) corresponds to solving the (ordinary) differential equation

$$\begin{cases} \dot{\mathbf{x}} = f_{\mathbf{u}}(\mathbf{x}) \\ \mathbf{x}(0) \in [\mathbf{x_0}] \end{cases} \tag{2}$$

with $f_{\mathbf{u}}$ being a function parameterized by the the control $\mathbf{u}$. The use of validated numerical integration on ODE for the problem in Equation (2) allows the design of an interval inclusion function of $\mathbf{x}(t, x_0, u)$. It provides the computation of $[\mathbf{x}](t; [\mathbf{x_0}], [\mathbf{u}])$ which stands for an outer approximation of the solution of the problem in Equation (1): $\{\mathbf{x}(t; \mathbf{x_0}, \mathbf{u}), \mathbf{x_0} \in [\mathbf{x_0}], \mathbf{u} \in [\mathbf{u}]\}$. It corresponds to the set of values that can take the state $\mathbf{x}$ at time t starting from any point $\mathbf{x}_0 \in [\mathbf{x_0}]$ for all controls $\mathbf{u} \in [\mathbf{u}]$ applied to the system. Any bounded uncertainty in the model can also be handled by representing it with a time constant parameter as done with the control $\mathbf{u}$ in Eq. (2).

To integrate the system until time t, a sequence of time instants $t_1, \ldots, t_n$ such that $t_1 < \cdots < t_n = t$ and a sequence of boxes $[\mathbf{x_1}], \ldots, [\mathbf{x_n}]$ such that $\mathbf{x}(t_{i+1}; [\mathbf{x_i}], [\mathbf{u}]) \subseteq [\mathbf{x_{i+1}}]$, $\forall i \in \{0, \ldots, n-1\}$ are computed with $[\mathbf{x}_i]$ an outer approximation of the set of the state vectors at time t_i. From the box $[\mathbf{x_i}]$, computing the box $[\mathbf{x_{i+1}}]$ is a classical 2-step method (see [14, 19]):

Phase 1 (Picard-Lindelof operator) compute an a priori enclosure $[\mathbf{x}_{i,i+1}]$ of the set

$$\{\mathbf{x}(t_k; \mathbf{x}_i, \mathbf{u}) \mid t_k \in [t_i, t_{i+1}], \mathbf{x}_i \in [\mathbf{x_i}], \mathbf{u} \in [\mathbf{u}]\} \subseteq [\mathbf{x}_{i,i+1}]$$

such that $\mathbf{x}(t_k; [\mathbf{x}_i], [\mathbf{u}])$ is guaranteed to exist and is unique,

Phase 2 compute an enclosure $[\mathbf{x}_{i+1}]$ of the solution at time t_{i+1}.

By repeating this process iteratively, we are able to produce an outer approximation of the system described in Equation (1) (see Figure 1). This scheme is used by the box RRT algorithm to compute a set of validated paths.

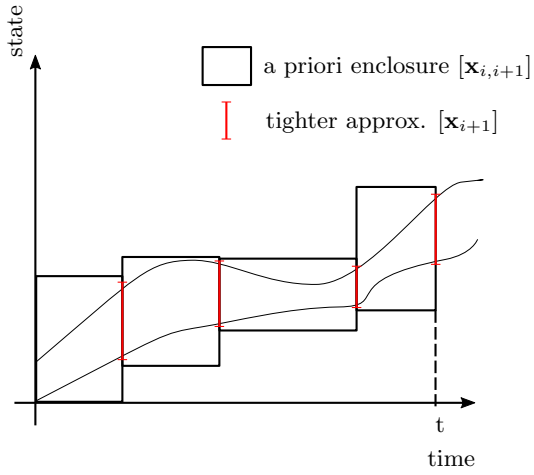


Figure 1: Illustration of the two-step method for the validated numerical integration of dynamical systems. $i = 1, ..., 5$.

1.3 The Box RRT Algorithm

The first part of the proposed method is to compute a set of validated trajectories. The box RRT algorithm and its improvements [2, 20] is a set of motion planner for robotic vehicles that guarantees to avoid static obstacles. It uses the data structures of the Rapidly-exploring Random Trees (RRT) to explore the state space. A tree of random subpaths is constructed until the goal is reached while any static obstacle is avoided. When the algorithm ends successfully, it provides, among other information, a set of boxes guaranteed to avoid any defined static obstacle. The result is in the form of a list of boxes $\{[x_1], \ldots, [x_N]\}$ all sorted by the time they are reached (see, for example, Figure 5 representing an example of the result of the algorithm described thereafter in Section 3). The box RRT algorithm provides also the control used during the seek of the set of trajectories but this information is not mandatory to the use of our method afterwards. Indeed, the box RRT can be used with a simplified model for the controlled dynamical system and then the controls $\mathbf{u}$ it uses are no longer relevant for the trajectory tracking part.

1.4 Differential Flatness

The differential flatness of dynamical systems is a structural property that a (possibly nonlinear) differential system can have. A system is differentially flat if there exists a set of independent variables (equal in number to the dimension of the control vector) referred to as flat outputs such that all states and controls of the system can be expressed in terms of those flat outputs and a finite number of their successive time derivatives. The mathematical definition of differential flatness is provided in Definition 1.

Definition 1 (Differential flatness [6])**.** *The controlled dynamical system*

$$\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u}) \tag{3}$$

with $\mathbf{x} \in \mathbb{R}^n$, $\mathbf{u} \in \mathbb{R}^m$ *is flat if there exist the maps* $h : \mathbb{R}^n \times (\mathbb{R}^m)^{r+1} \to \mathbb{R}^m$, $\varphi : (\mathbb{R}^m)^r \to \mathbb{R}^n$ *and* $\psi : (\mathbb{R}^m)^{r+1} \to \mathbb{R}^m$ *such that*

$$\begin{aligned} \mathbf{z} &= h\left(\mathbf{x}, \mathbf{u}, \dot{\mathbf{u}}, \ldots, \mathbf{u}^{(r)}\right) \\ \mathbf{x} &= \varphi\left(\mathbf{z}, \dot{\mathbf{z}}, \ldots, \mathbf{z}^{(r-1)}\right) \\ \mathbf{u} &= \psi\left(\mathbf{z}, \dot{\mathbf{z}}, \ldots, \mathbf{z}^{(r-1)}, \mathbf{z}^{(r)}\right) \end{aligned}$$

A differentially flat system can be caracterized when considering a subset of its state and control vectors and their associated derivatives. As recalled in the following, a trajectory can be tracked in a flat output space. An example of differentially flat system is given hereafter in Section 3.2.

1.5 Trajectory Tracking

The differential flatness makes possible to control a system by only using the flat output. The next definition recalls the endogenous dynamic feedback [7] in the case of a differentially flat system.

Definition 2 (Endogenous dynamic feedback using flatness)**.** *We consider the flat system described in Eq.* (3) *with a flat output* $\mathbf{z}$ *and a given trajectory* $\mathbf{z}_d : \mathbb{R} \to \mathbb{R}^m$*. Once the flat system is differentiated enough to get an equation of the type*

$$\mathbf{z}^{(p)} = \omega, \tag{4}$$

the system can be stabilized around a trajectory by stabilizing the flat output with ω *defined as a particular control:*

$$\omega = \mathbf{z}_d^{(p)} - k_0(\mathbf{z} - \mathbf{z}_d) - k_1(\dot{\mathbf{z}} - \dot{\mathbf{z}}_d) - \cdots - k_{p-1}(\mathbf{z}^{(p-1)} - \mathbf{z}_d^{(p-1)}) \tag{5}$$

with $\mathbf{z}^{(i)}$ *the* $i-th$ *time derivative of* $\mathbf{z}$ *and* $k_0, \ldots, k_{p-1}$ *such that the polynomial* $s^p = k_{p-1}s^{p-1} + \cdots + k_1 s + k_0$ *has only roots with negative real part. It results in the tracking error converging and stable.*

The goal is now to construct this function $\mathbf{z}_d(t)$ providing a trajectory in the flat output space. In our context, it is done using the cubic Hermite splines.

1.6 The Cubic Hermite Splines

In order to use endogenous dynamic feedback using flatness, we have to compute one particular trajectory $\mathbf{z}_d$ from the initial state to the target the robot will have to follow. An intermediary desired goal is required to be reached iteratively until we reach the goal. The cubic Hermite splines [8, 4] are one way to achieve this (see, *e.g.* [12] for a use in path planning and [9] for one in trajectory tracking).

Definition 3 (Cubic Hermite Spline)**.** *Let $\phi : \mathbb{R} \to \mathbb{R}^n$ be a diffentiable function, $t_0,\ t_1 \in \mathbb{R}$ and two positions $\mathbf{p_0} = \phi(t_0)$ and $\mathbf{p_1} = \phi(t_1)$ with the derivatives $\mathbf{m_0} = \dot{\phi}(t_0)$ and $\mathbf{m_1} = \dot{\phi}(t_1)$ respectively, the cubic Hermite spline is*

$$P(t) = p\left(\frac{t - t_0}{t_1 - t_0}\right) \tag{6}$$

with

$$p(t) = h_{00}(t)\mathbf{p_0} + h_{10}(t)\mathbf{m_0}(t_1 - t_0) + h_{01}(t)\mathbf{p_1} + h_{11}(t)\mathbf{m_1}(t_1 - t_0) \tag{7}$$

and

$$\begin{aligned} h_{00}(t) &= 2t^3 - 3t^2 + 1 \\ h_{10}(t) &= t^3 - 2t^2 + t \\ h_{01}(t) &= -2t^3 + 3t^2 \\ h_{11}(t) &= t^3 - t^2. \end{aligned}$$

The error between the spline $P(t)$ and the true trajectory $\phi(t)$ can be easily computed as follows: there exists τ such that

$$\phi(t) - P(t) = \frac{\phi^{(k)}(\tau)}{k!}\prod_i (t - t_i)^{k_i} \tag{8}$$

with k the number of points, k_i the number of known derivatives + 1. In our case, the number of points is 2 and the number of derivatives is 2.

This error can be approximated using interval analysis as follow: in the time interval $[t_0, t_1]$ where the spline is defined, The error can be outer approximated by the interval function $[E]\,(t, [t_0, t_1])$ such that:

$$\phi(t) - P(t) \in [E]\,(t, [t_0, t_1]) = \frac{\left[\ddot{\phi}\right]([t_0, t_1])}{2}(t - t_0)^3(t - t_1)^3,\ \forall t \in [t_0, t_1] \tag{9}$$

with $\left[\ddot{\phi}\right]$ an interval inclusion function of the second time derivative of ϕ. In practice, the model being provided, the second (as well as the first) derivative over time of ϕ is also available by symbolically derivating the system over time prior to the execution of the method. A guaranteed cubic Hermite spline $[P]\,(t)$ then consists in adding this box error:

$$[P]\,(t) = P(t) + [E]\,(t, [t_0, t_1])$$

2 Validated Trajectory Tracking using Flatness

This section is dedicated to the main results on validated trajectory tracking using flatness. We consider a controlled dynamical system of the form described in Equation (1).

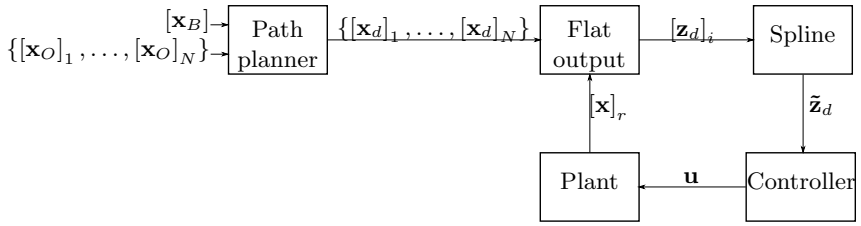


Figure 2: Block diagram representing the different components in our controller.

In this problem, the initial condition is assumed to be inside a set of values here defined as a box $[\mathbf{x}_0]$ and a goal $[\mathbf{x}_G]$ is also provided such that the goal is eventually reached at a bounded time $t_f \leqslant T$, as stated by the quantified proposition

$$(\exists u(t) : \mathbb{R} \to \mathbb{R}^m)(\exists t_f \in [0, T])(\mathbf{x}(t_f) \in [\mathbf{x}_G]). \tag{10}$$

A set of obstacles $[\mathbf{x}]_{O,1}, \ldots, [\mathbf{x}]_{O,N}$ is also defined and the trajectory $\mathbf{x}(t)$ verifying Eq. (10) must avoid them:

$$(\forall i = 1, \ldots, N)(\forall t \in [0, t_f])(\mathbf{x}(t) \notin [\mathbf{x}]_{O,i}). \tag{11}$$

The goal of trajectory tracking is then to produce the control function $u(t)$ such that the system follows a trajectory that respects the propositions in Equations (10) and (11) dealing with uncertainties maybe occurring in the model and the set of initial and final positions. A validated numerical integration scheme as described in Section 1.2 using interval analysis is used. We present here the method to produce the controls that allows a cyberphysical system to follow a path. The block diagram corresponding to the control strategy is given in Figure 2 and Algorithm 1 provides a sketch of the proposed method after the path planner provided its result.

Path planner The initial position of the robot $[\mathbf{x}_B]$, the goal $[\mathbf{x}_G]$ and the set of obstacles $\{[\mathbf{x}_O]_1, \ldots, [\mathbf{x}_O]_N\}$ are first given to the path planner which in return gives a set of guaranteed trajectories $\{[\mathbf{x}_d]_1, \ldots, [\mathbf{x}_d]_N\}$.

Flat output Using flatness, a box $[\mathbf{z}_d]_i$ in the flat output space is provided from the result of the path planner. This box $[\mathbf{z}_d]_i$ contains the current flat output of the robot $[\mathbf{z}_r]$ according to its current position $[\mathbf{x}_r]$.

Spline A cubic Hermite spline is then computed to return the next desired flat output $\tilde{\mathbf{z}}_d$ for the robot. It is illustrated in Fig. 3. A particular point $\tilde{\mathbf{z}}_{d;i+1} \in [\mathbf{z}_d]_{i+1}$ is chosen to be the desired flat output for the robot (we chose the midpoint of $[\mathbf{z}_d]_{i+1}$). The spline $P(t)$ is constructed from a point $\tilde{\mathbf{z}}_r$ in the flat output of the robot (we chose the midpoint of $[\mathbf{z}_r]$ as well) to the chosen point $\tilde{\mathbf{z}}_{d,i+1}$. The guarranteed spline $[P](t)$ is also computed and a verification on it is done to check is it remains in the union $[\mathbf{z}_d]_i \cup [\mathbf{z}_d]_{i+1}$. If the verification fails, a new endpoint for the spline is chosen, one inside the intersection $[\mathbf{z}_d]_i \cap [\mathbf{z}_d]_{i+1}$. An intermediary point $\tilde{\mathbf{z}}_d$ on the spline is chosen to be the next position to reach for the robot.

Input: $\{[\mathbf{z}_d]_i\}$: the result of the guaranteed path planner in the flat output space

while *the robot has not arrived* **do**

`// Flat output`

$[\mathbf{z}_r]$: the flat output of the robot from its current position $[\mathbf{x}_r]$

$[\mathbf{z}_d]_i$: box the robot is in $(([\mathbf{z}_r]) \subseteq [\mathbf{z}_d]_i)$

$\tilde{\mathbf{z}}_{d;i+1} = \text{mid}([\mathbf{z}_d]_{i+1})$

`// Spline`

$P(t)$: the Hermite spline between $\text{mid}([\mathbf{z}_r])$ and $\tilde{\mathbf{z}}_{d;i+1}$

$\tilde{\mathbf{z}}_d$: next reference point on the spline for the robot

`// Controller`

computation of the controls using dynamic feedback with flat output $\mathbf{z}_r$ and $\tilde{\mathbf{z}}_d$ for the robot to reach $\tilde{\mathbf{z}}_d$

`// Plant`

$[\mathbf{x}_r]$: new position of the robot after applying the controls

end

Algorithm 1: Sketch of the proposed method for the tracking of the trajectory provided by guaranteed path planner algorithm (sketch with correspondance with the block diagram in Fig. 2).

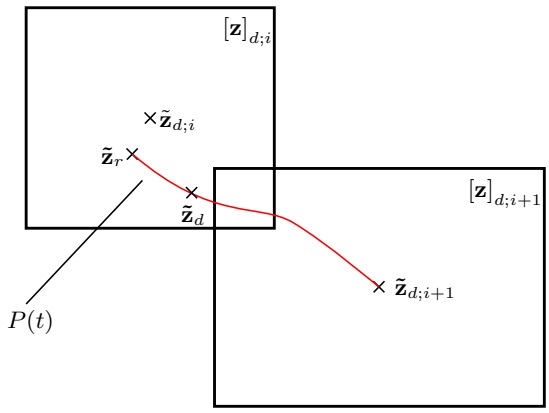


Figure 3: Computation of the next control from the computation of the cubic Hermite spline.

Controller All required information for the computation of a new control are provided: the spline provides the reference point $\tilde{\mathbf{z}}_d$ and the current flat output $[\mathbf{z}_r]$ of the robot is also known. Their time derivatives are directly known as well and a new control can be computed using the classical endogenous dynamic feedback using flatness as described in Definition 2. The control is then tested using a guaranteed numerical integration scheme to validate that the robot will not leave the set of guaranteed trajectories from the path planner in the same time horizon the controls will be applied to the robot. In the case the test fails, the time horizon and/or the choice on the reference point $\tilde{\mathbf{z}}_d$ should be modified. This change is not discussed in this article.

Plant The control is applied to the robot. It eventually provides the new position $[\mathbf{x}_r]$ of the robot and the process can start again until the goal $[\mathbf{x}_G]$ is reached ($[\mathbf{x}_r] \subseteq [\mathbf{x}_G]$).

The next section illustrates this method on a classical Dubins car model.

3 Experiments on the Dubins Car Model

In this section is given an example of the use of the method introduced in Section 2 on the Dubins car model.

3.1 Implementation

The experiment has been conducted using a C++ implementation of the trajectory tracking coupled with a C++ implementation of the box RRT algorithm. All the tests have been made by simulating the behaviour of the dynamical system in a guaranteed manner. The validated numerical integration part has been handled using the C++ library DynIbex[2] [1], a plugin of Ibex [3] on the Dubins car model, described in the following.

3.2 The Dubins Car Model

We consider the Dubins car model

$$\begin{cases} \dot{x} = v\cos\theta \\ \dot{y} = v\sin\theta \\ \dot{\theta} = u \end{cases} \tag{12}$$

with $(x, y)^T$ the position of the car and θ its angle to the coordinate system. The controls (u, v) are the angular and the longitudinal speeds of the vehicle respectively (see Figure 4 for an illustration of the Dubins car model).

It has been proved that this system is flat [13] and $z = (x, y)^T$ is a flat output. Indeed the state and the control variables can be written using x and y and their

[2]Available at: `https://perso.ensta-paris.fr/~chapoutot/dynibex/`

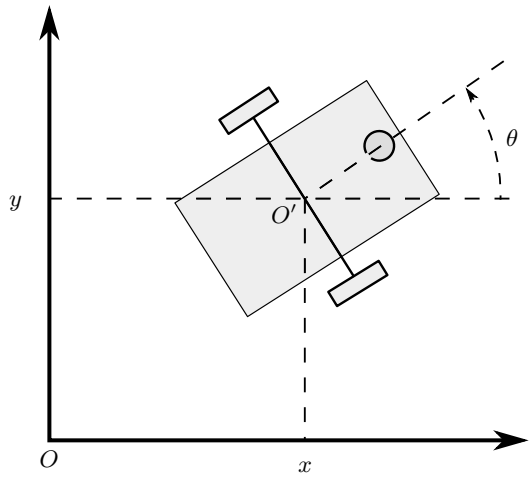


Figure 4: The Dubins car model.

derivatives:

$$\begin{cases} x = z_1 \\ y = z_2 \\ \theta = \tan^{-1}\left(\frac{\dot{y}}{\dot{x}}\right) = \tan^{-1}\left(\frac{\dot{z}_2}{\dot{z}_1}\right) \\ v = \sqrt{\dot{x}^2 + \dot{y}^2} = \sqrt{\dot{z}_1^2 + \dot{z}_2^2} \\ u = \frac{\ddot{y}\dot{x} - \ddot{x}\dot{y}}{\dot{x}^2 + \dot{y}^2} = \frac{\ddot{z}_2\dot{z}_1 - \ddot{z}_1\dot{z}_2}{\dot{z}_1^2 + \dot{z}_2^2}. \end{cases}$$

The set of available positions provided by the box RRT algorithm directly gives the flat output which is the position of the robot. In Figure 5 is given an example of the set of boxes returned by the box RRT algorithm. The black boxes represent the safe set of trajectories, the red ones are the obstacles, the blue one is the goal and the green dots are the centers of the black boxes (the safe set of trajectories). The set of trajectories given by the path planner (here the box RRT algoritm) is not optimal but it is not an issue for our method. We apply the controls produced by the method described in Section 2 and the results are shown in Figure 6. As in Figure 5 the result of the box RRT is still in black, the obstacle are in red and the goal is in blue. The green dots are the successive centers of the boxes containing the position of the robot after each application of the computed controls and the red dots are the reference points computed with the cubic Hermite splines. To make the figure more readable, only the centers of the position boxes of the robot are shown. At each step of the control to the goal, the box containing the robot $[\mathbf{x}_r]$ is checked to remain inside the black boxes of the box RRT algorithm as long as the guaranteed numerical simulation of the application of the control to the robot.

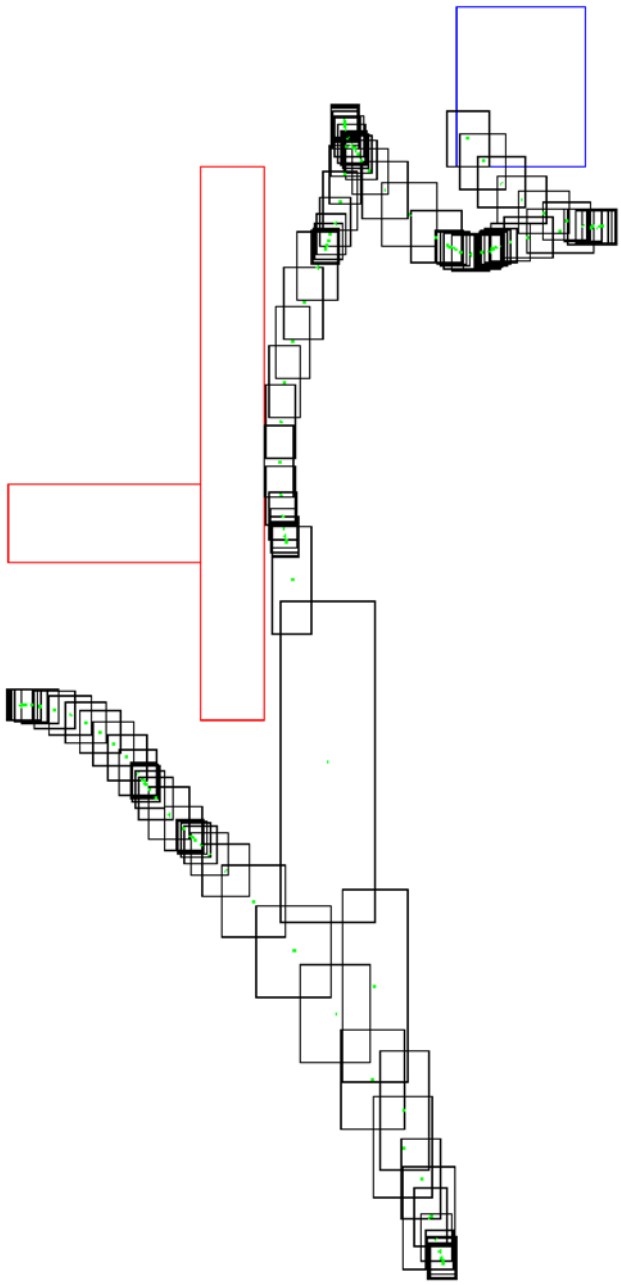


Figure 5: Example of the result provided by the box RRT algorithm. Red boxes: static obstacles ; blue: goal ; black: validated set of trajectories; green dots: center of the black boxes.

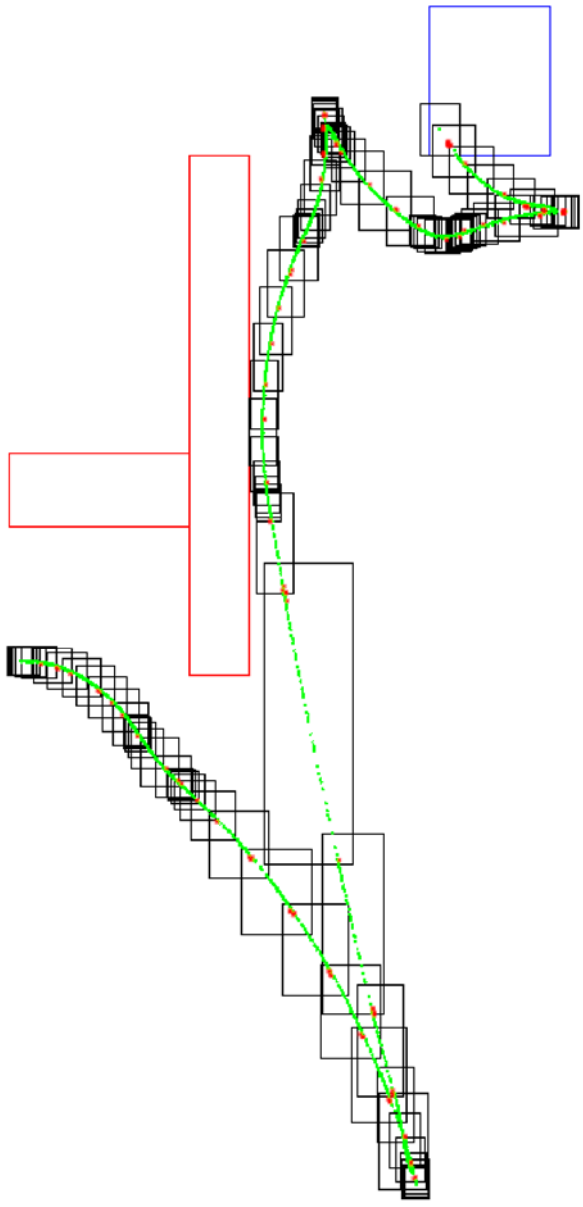


Figure 6: Example of the result of the controls computed to follow the set of trajectories from the box RRT algorithm (same as in Figure 5). Red boxes: static obstacles ; blue: goal ; black: validated set of trajectories; green dots: center of the robot position boxes. red dots (magnified): reference points from the cubic Hermite spline computation.

4 Conclusion

A validated trajectory tracking using flatness has been presented. From a guaranteed set of trajectory planner which provides an *a priori* set of guaranteed paths, our method uses flatness to perform the computation of the controls. The Intermediary reference points for the system are provided using guaranteed cubic Hermite splines. Eventually, the control to apply is proved to fulfill the requirements by simulating in a set-membership manner the use of the controls on the system. It allows to prove it remains inside the set of paths. This method has been illustrated with an example of a terrestrial robot with a flat system model. The next step will be to apply this method on a real robotic platform to validate the method in the context of an embedded system. An extra step is required to compute the control while the robot has not already reached its local goal provided by the cubic Hermite spline. Another improvement will be to take into account dynamical obstacles. It should be done by recalling the path planner each time a potential collision is detected. Eventually the method must be extended to the case of non flat systems.

Acknowledgment

This work has been partially supported by a DGA AID project.

References

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

[2] Alexandre dit Sandretto, Julien, Chapoutot, Alexandre, and Mullier, Olivier. Formal verification of robotic behaviors in presence of bounded uncertainties. In *2017 First IEEE International Conference on Robotic Computing (IRC)*, pages 81–88. IEEE, 2017. DOI: `10.1109/IRC.2017.17`.

[3] Chabert, Gilles et al. Ibex, an interval-based explorer. `http://www.ibex-lib.org/`, 2007.

[4] De Boor, Carl. *A practical guide to splines*, volume 27. Springer-Verlag New York, 1978. DOI: `10.2307/2006241`.

[5] Fliess, M, Lévine, J, Martin, Ph, Ollivier, F, and Rouchon, P. Controlling nonlinear systems by flatness. In *Systems and Control in the Twenty-first Century*, pages 137–154. Springer, 1997. DOI: `10.1007/978-1-4612-4120-1_7`.

[6] Fliess, Michel, Lévine, Jean, Martin, Philippe, and Rouchon, Pierre. Flatness and defect of non-linear systems: introductory theory and examples.

International journal of control, 61(6):1327–1361, 1995. DOI: 10.1080/00207179508921959.

[7] Fliess, Michel, Lévine, Jean, Martin, Philippe, and Rouchon, Pierre. A Lie–Bäcklund approach to equivalence and flatness of nonlinear systems. *IEEE Transactions on Automatic Control*, 44(5), 1999. DOI: 10.1109/9.763209.

[8] Hermite, Charles. Sur la théorie des fonctions elliptiques. *Comptes rendus de l'Académie des sciences*, 57:613, 1863.

[9] Hintzen, Niels T, Piet, Gerjan J, and Brunel, Thomas. Improved estimation of trawling tracks using cubic Hermite spline interpolation of position registration data. *Fisheries Research*, 101(1-2):108–115, 2010. DOI: 10.1016/j.fishres.2009.09.014.

[10] Jaulin, Luc, Kieffer, Michel, Didrit, Olivier, and Walter, Eric. Interval analysis. In *Applied interval analysis*, pages 11–43. Springer, 2001. DOI: 10.1137/1009099.

[11] LaValle, Steven M. Rapidly-exploring random trees: A new tool for path planning. Technical report, 1998.

[12] Lekkas, Anastasios M and Fossen, Thor I. Integral los path following for curved paths based on a monotone cubic Hermite spline parametrization. *IEEE Transactions on Control Systems Technology*, 22(6):2287–2301, 2014. DOI: 10.1109/TCST.2014.2306774.

[13] Levine, Jean. *Analysis and control of nonlinear systems: A flatness-based approach*. Springer Science & Business Media, 2009. DOI: 10.1007/978-3-642-00839-9.

[14] Lohner, Rudolf J. Computation of guaranteed enclosures for the solutions of ordinary initial and boundary value problems. In *Institute of mathematics and its applications conference series*, volume 39, pages 425–425. Oxford University Press, 1992.

[15] Ma, Dailiang, Xia, Yuanqing, Shen, Ganghui, Jia, Zhiqiang, and Li, Tianya. Flatness-based adaptive sliding mode tracking control for a quadrotor with disturbances. *Journal of the Franklin Institute*, 355(14):6300–6322, 2018. DOI: 10.1016/j.jfranklin.2018.06.018.

[16] Moore, R. E. *Interval Analysis*. Series in Automatic Computation. Prentice Hall, 1966. DOI: 10.1137/1009099.

[17] Moore, Ramon E, Kearfott, R Baker, and Cloud, Michael J. *Introduction to interval analysis*, volume 110. Siam, 2009. DOI: 10.1137/1.9780898717716.

[18] Mullier, Olivier and Courtial, Estelle. Set-membership computation of admissible controls for the trajectory tracking. *Reliable Computing*, 2017.

[19] Nedialkov, Nedialko S, Jackson, Kenneth R, and Corliss, George 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.

[20] Panchea, Adina, Chapoutot, Alexandre, and Filliat, David. Extended reliable robust motion planners. *56th IEEE Conference on Decision and Control*, Dec 2017. DOI: 10.1109/CDC.2017.8263805.

[21] Pepy, Romain, Kieffer, Michel, and Walter, Eric. Reliable robust path planner. In *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1655–1660. IEEE, 2008. DOI: 10.1109/IROS.2008.4650833.

Acta Cybernetica **25** (2021) 101–125.

Prospects on Solving an Optimal Control Problem with Bounded Uncertainties on Parameters using Interval Arithmetic

Etienne Bertin[abc], Elliot Brendel[ad], Bruno Hérissé[ae], Julien Alexandre dit Sandretto[bf], and Alexandre Chapoutot[bg]

Abstract

An interval method based on Pontryagin's Minimum Principle is proposed to enclose the solutions of an optimal control problem with embedded bounded uncertainties. This method is used to compute an enclosure of all optimal trajectories of the problem, as well as open loop and closed loop enclosures meant to validate an optimal guidance algorithm on a concrete system with inaccurate knowledge of the parameters. The differences in geometry of these enclosures are exposed, and showcased on a simple system. These enclosures can guarantee that a given optimal control problem will yield a satisfactory trajectory for any realization of the uncertainties. Contrarily, the probability of failure may not be eliminated and the problem might need to be adjusted.

Keywords: optimal control, Pontryagin's principle, interval arithmetic, bounded uncertainties, penalization

1 Introduction

Optimal control of aerospace systems is performed by modeling the considered system by dynamics depending on multiple uncertain parameters (for example, aerodynamic coefficients, mass,...). Usually, optimal control problems are solved for nominal values of these parameters, as in Figure 1. Then the robustness of the solution is demonstrated by dispersing the parameters around nominal values with

[a]DTIS, ONERA, Université Paris Saclay, F-91123 Palaiseau, France, E-mail: firstname.lastname@onera.fr

[b]U2IS, ENSTA Paris, Institut Polytechnique de Paris, Palaiseau, France. E-mail: firstname.lastname@ensta-paris.fr

[c]ORCID: https://orcid.org/0000-0003-4245-2964

[d]ORCID: https://orcid.org/0000-0002-0458-4993

[e]ORCID: https://orcid.org/0000-0002-9241-0810

[f]ORCID: https://orcid.org/0000-0002-6185-2480

[g]ORCID: https://orcid.org/0000-0002-7230-0710

DOI: 10.14232/actacyb.285798

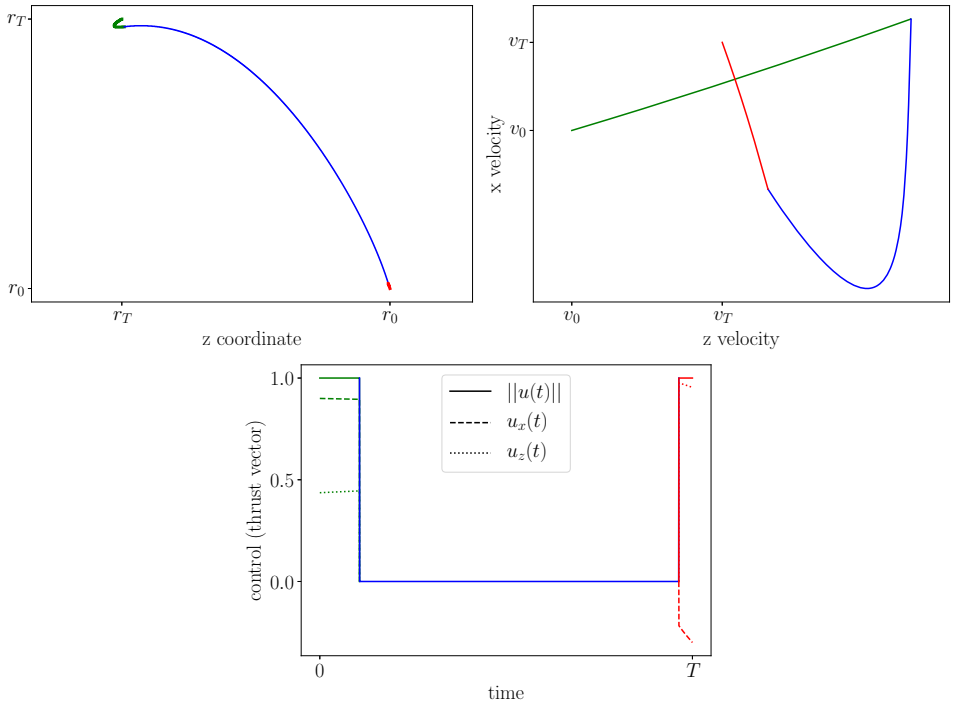


Figure 1: Optimal trajectory, velocity space trajectory and control for Goddard's re-entry problem in Cartesian Geo-centered coordinates[1]. In green the boostback burn, in thin blue the ballistic phase and in red the landing burn. Burn phases are characterized by a saturated control ($||u|| = 1$) and a dash in velocity space, while ballistic phases are characterized by no control ($||u|| = 0$). The model, the parameters and the initial and final positions are taken from [4]. More details on the model and boundary conditions are given in Section 5.

Monte Carlo simulations [5]. Another approach is to solve a chance constrained optimal control problem in order to guarantee a probability of success [7]. These methods do not exclude the eventuality of failure, which can be problematic on critical systems. In addition these problem-solving methods often introduce numerical approximations, which question the validity of the results.

Interval arithmetic has shown their ability to address several control problems [9], providing validated solutions while dealing with method uncertainties (numerical approximations) as well as with model uncertainties (unknown but bounded parameters). For instance the reachable set of a dynamic system can be over-approximated. This over-approximation can demonstrate that a system will remain

[1]Standard coordinates for rocket trajectory are altitude and latitude; Cartesian coordinates were chosen instead to allow some consistency with figures in Section 5

out of a critical region, thus guarantee its safety [2]. These methods can also be used to design robust optimal control. For instance, [11, 12] propose algorithms to compute a control that stirs a system to a desired state for any realization of a bounded noise while achieving the lowest upper bound on the cost.

Pontryagin's Maximum Principle [13] (PMP) provides necessary optimality conditions for the resolution of optimal control problems by transforming an optimal control problem into a root-finding problem. Derived methods have proven their efficiency and their precision compared to direct methods [13, 14], but their convergence strongly depends on their initialization and a prior knowledge of the solution structure is needed.

Our goal is to address an aerospace challenge highlighted by [3], which is to enclose the trajectory of a reusable launch vehicle subject to external perturbations. The trajectory during the landing burn is computed online so the control is not known beforehand. Computing an enclosure could guarantee that the vehicle lands safely on the landing platform.

To that end, the return version of Goddard's problem [5] is considered. It consists in performing the landing of the first stage of a rocket while minimizing its fuel consumption. The optimal trajectory of this system for nominal values of the parameters and the associated control are presented on Figure 1. Uncertainties on the parameters will be added into the model in the form of intervals and deterministic enclosures will be computed by combining interval arithmetic and the necessary optimality conditions given by the application of PMP. Although this method cannot be applied to Goddard's problem yet, this paper exposes our reasoning on simplified problems.

Firstly, Section 2 introduces the problem considered. Section 3 presents the interval and optimal control methods that will be later used. Section 4 proposes three enclosures that can be utilized to discuss the reliability of an optimal control problem with uncertainties. Section 5 showcases these enclosures on a simplified problem. Lastly, Section 6 proposes some uses of these enclosures and the remaining work to apply these methods on Goddard's problem.

Notations

Interval variables and interval vectors are always enclosed in brackets: $[x]$.
Any variable that is not enclosed in bracket is a real or vector variable.
Interval valued functions are enclosed in brackets: $[f] : [x] \mapsto [f]([x])$.
The bounds of an interval $[a]$ are noted $\underline{a}, \overline{a}$. Its middle point is noted $\text{mid}\,([a])$.
$(.)$ such as in $y(.)$ denotes a time function: $y(.) : t \in [0, T] \mapsto y(t) \in \mathbb{R}^n$.
$y(\tau)$ and y_τ are different, the later being a vector unrelated to function $y(.)$.
A hat as in $\hat{\xi}$ denotes a variable that is related to an estimation.

2 Control problem with uncertainties

In this section, the optimal control problem and the incorporation of uncertain parameters are presented. Then three enclosures meant to discuss the relevance of an optimal control problem will be introduced.

2.1 Optimal Control Problem (OCP)

The following non-interval optimal control problem is considered

$$\min_{u(.)\in\mathcal{U}} \int_0^T \ell(y(t),u(t))dt + \Psi(y(T)) \text{ s.t. } \begin{cases} \dot{y}(t) = f(y(t),u(t),\xi(t),t), \\ y(0) = y_0 \\ T \text{ is fixed.} \end{cases} \quad (1)$$

Interpretation : the system, a launcher for instance, is characterized by the following data:

- A state $y(t)$ e.g. position, velocity and mass of the launcher. The time function $y(.)$ is called a trajectory.
- An initial state y_0.
- Dynamics $\dot{y}(t) = f(y(t),u(t),\xi(t),t)$ e.g. the time derivative of position is velocity, the derivative of velocity is the forces divided by mass etc....
- A control input $u(.)$ e.g. the reactor thrust vector.
- Parameters $\xi(.)$ e.g. aerodynamic coefficients, maximum thrust....
- A continuous cost $\ell(y(t),u(t))$ e.g. fuel consumption.
- A final cost $\Psi(y(T))$ e.g. the distance between the final state and the target.

System (1) is considered controllable. Hence for any initial state y_0 and specific parameter function $\xi(.)$, an optimal control $u(.)$ can be found. As a consequence, a control can be defined implicitly as the solution of an OCP, as opposed to explicit methods such as time series or gain tables. For instance, the Model Predictive Control (MPC) approach defines the control as the solution of an optimal control problem; the controller solves this optimal control problem at each time step to determine the control to be applied. Some parts of a launch vehicle trajectory are also defined by an OCP: [3] states that the control during the landing burn is the optimal solution of a problem that is solved autonomously during the flight.

Bounded uncertainties on the initial state and the parameters will be added into this model. The initial state y_0 and parameter function $\xi(.)$ take their values in interval enclosures: $y_0 \in [y_0]$ and $\xi(t) \in [\xi], \forall t$, where $[\xi]$ and $[y_0]$ are intervals of the form $\left[\underline{\xi}, \overline{\xi}\right]$ or boxes (see definition in Section 3.2.1).

Since they are challenging to simulate with validated simulation, optimal control problems with varying time horizons and state dependant transitions are not

considered in this paper. However, it is worth noting that time dependent transitions, or switches, are possible (see Section 3.2.3). This assumption excludes Goddard's problem. The bang-bang transition from a zero control to a saturated control seen in Figure 1 is a state dependent transition and cannot be simulated with our method, unless a precise switch time is forced. Section 6.2 discusses the steps to be taken to apply this method to Goddard's problem.

Note also that the presented method requires additional regularity conditions which will be presented in Section 4.

2.2 Representation of the uncertainties in the OCP

Parameters are bounded functions of time: $\forall t \in [0,T], \xi(t) \in [\xi]$. It is also assumed that each function $\xi(.)$ is infinitely differentiable. This ensures that dynamics have all the needed mathematical properties to be simulated.

The time dependency establishes a general framework without invalidating the important assumptions of the optimality criterion used (see Section 3.1.2). It covers indeed more scenarios than the approach considering that the parameters are unknown constant values, which is an implicit assumption in [5] in which parameters are picked randomly at the start and stay constant during the simulations.

The most encompassing framework would consider that parameters are also functions of the state as a whole. However that would arise the need for more assumptions (bounded spatial derivatives...).

2.3 Contributions

The motivation is to assess whether a given OCP outputs an adequate trajectory for any realization of the initial state and parameter functions. A trajectory is characterized by two realizations of the parameter function: the actual parameter function $\xi(.)$ and the parameter function estimated by the system $\hat{\xi}(.)$. Both are taken amidst the same bound $[\xi]$ but they may differ as the actual parameters are often unknown in practice.

Hence, the following three enclosures of the solution of (1) are proposed:

- An *anticipative enclosure* containing trajectories whose control is computed with the perfect knowledge of the parameter function: $\hat{\xi}(.) = \xi(.)$.
- An *open loop*[2] *enclosure* containing trajectories whose control is computed once at the start with an inaccurate parameter function $\hat{\xi}(.) \neq \xi(.)$ and then followed blindly.
- A *closed loop*[3] *enclosure* containing trajectories whose control is computed several times online with accurate measurements of the state but an inaccurate parameter function $\hat{\xi}(.) \neq \xi(.)$.

[2]also known as feed forward
[3]also known as feedback

Measures of the state are considered accurate, contrarily to parameters that are unknown. The reason is that knowing parameter in advance is fundamentally impossible because it requires seeing into the future, while having accurate measurement is only a matter of hardware. Bounded errors on measurements will be considered in future works.

These three enclosure can be used to validate all solutions of an OCP and assess risks. Contrarily to the approach in [11], which optimizes a single control that works for all realizations of the bounded uncertainties, the proposed method does not output a control directly. Instead, it can guarantee that a given optimal control problem will produce a satisfactory control for all realizations of the uncertainties on the initial state and parameter function.

To take the example of launch vehicles [3], an OCP is solved right before the landing burn to compute a guidance trajectory. This OCP is designed to provide a trajectory to the landing platform with any possible dynamics and from any possible state the launcher may be in right before the landing burn. The proposed approach is to use a conservative enclosure of all possible dynamics and of all possible states before the landing burn to derive enclosures of all trajectories that are solutions of a given OCP with these initials states and dynamics. If every element in the enclosures satisfy the final constraint of finishing on the landing platform, then the OCP is guaranteed to output a satisfactory trajectory in practice and it may be used on an actual launcher.

This is only a necessary condition. Since the enclosures are conservative, the fact that the enclosure does not satisfy the constraints does not prove the existence of a realization of the uncertainties that will cause the launcher to violate the constraints. Nevertheless, on such critical systems, the possibility of failure is not tolerated and may justify changing the OCP.

3 Mathematical preliminaries

In this section, a numerical method for optimal control is presented as well as interval-based validation methods. Those methods will be combined in Section 4.

3.1 Numerical resolution of a control problem

As a first step, properties of dynamical systems will be recalled and the flow and resolvent notations will be introduced. Then these notations will be used to formalize the indirect method known as Pontryagin's Minimum Principle.

3.1.1 Flow and resolvent of an ordinary differential equation (ODE)

Consider an ODE in the form:

$$\begin{cases} \dot{x}(t) &= g(x, \xi(t), t) \\ x(t_0) &= x_0 \end{cases}$$

To simplify notations and since parameters only depend on time, $g(x, \xi(t), t)$ is denoted by $g(x, t)$ whenever doing so does not impede comprehension.

A solution can be approximated by numerical methods, such as Euler's method, or more efficient Runge-Kutta methods [6]. Integrating the dynamics between two boundary times τ and T can be seen as a flow function $\Phi_{\tau,T}$. The function $\Phi_{\tau,T}$ takes an initial state, simulates it from τ to T and returns the final state. If g is twice differentiable, the flow $\Phi_{\tau,T}$ has a spatial derivative in x_τ which is the resolvent $R_{\tau,T}(x_\tau)$ of the linearized system [8].

To illustrate these notations, consider two boundary times τ and T. Let $x(.)$ be the solution of the ODE:

$$\begin{cases} \dot{x}(t) &= g(x,t) \\ x(\tau) &= x_\tau \end{cases}$$

$\Phi_{\tau,T}(x_\tau)$ is the state of the solution at time T, $\Phi_{\tau,T}(x_\tau) = x(T)$.

If $z(.)$ is the solution of the same ODE with an initial perturbation δx:

$$\begin{cases} \dot{z}(t) &= g(z,t) \\ z(\tau) &= x_\tau + \delta x \end{cases}$$

then its final state will be $z(T) = x(T) + R_{\tau,T}(x_\tau) \cdot \delta x + o(||\delta x||)$. The resolvent of the linearized system enables a first order approximation of the final error caused by an initial perturbation, and as such, it is the spatial first derivative of the flow.

The resolvent can sometimes be computed analytically, notably for linear time invariant systems: if $\dot{x} = A \cdot x$ then $R_{\tau,T}(x_\tau) = \exp((T-\tau)A)$. If no analytic formula is available, $R_{\tau,T}(x_\tau)$ can be computed by integrating the following ODE:

$$\begin{cases} \dot{x}(t) &= g(x) \\ \dot{R}_{\tau,t}(x_\tau) &= \dfrac{\partial g}{\partial x}(x(t)) \cdot R_{\tau,t}(x_\tau) \\ x(\tau) &= x_\tau \\ R_{\tau,\tau}(x_\tau) &= I_n. \end{cases} \tag{2}$$

In the following sections, the spatial arguments will often be removed to simplify notations. Hence $R_{\tau,T}$ means $R_{\tau,T}(x_\tau)$.

3.1.2 Optimal control

An OCP is considered as in Section 2

$$\min_{u(.)\in\mathcal{U}} \int_\tau^T \ell(y(t), u(t))dt + \Psi(y(T)) \text{ s.t. } \begin{cases} \dot{y}(t) = f(y(t), u(t), \xi(t)), \forall t \in [\tau, T], \\ y(\tau) = y_\tau \\ T \text{ is fixed.} \end{cases}$$

This control can be computed by multiple methods that mainly fall in two categories:

- *direct methods*: the control is discretized, thus turning the infinite dimensional problem into a high dimensional Euclidean optimization problem.
- *indirect methods*: using a characterization of the optimal trajectory and control, the infinite dimensional optimization problem is transformed into a simpler problem.

The second approach is considered in this article.

To characterize the optimal solution, a new variable is added, the co-state $p(.)$ which is analog to the dual multiplier in constrained Euclidean optimization. $p(.)$ is a vector valued time function on the time range $[0, T]$ and the vectors $p(t)$ are the same dimension as the vectors $y(t)$.

The main result is the following theorem. This is a simplified version that is sufficient for the problem tackled. It does not explicit the domain of each function and does not address abnormal cases. A complete theorem can be found in [4, 13].

Pontryagin's Minimum Principle[4] **(PMP):** If $(y(.), u(.))$ are a normal optimum, then there exists a non trivial co-state $p(.)$ such that $(y(.), p(.), u(.))$ satisfy the following equations:

$$
\begin{aligned}
&H(y, p, u, t) = \ell(y, u) + p.f(y, u, \xi(t)) \\
&\begin{cases} \dot{p}(t) = -\dfrac{\partial H}{\partial y}(y(t), p(t), u(t), t) \\ p(T) = \dfrac{\partial \Psi}{\partial y}(y_T) \end{cases} \\
&u(t) \in \arg\min_u H(y(t), p(t), u, t) \forall t \in [\tau, T],
\end{aligned}
$$

where H is called the pre-Hamiltonian. The optimal control $u(.)$ is defined implicitly as minimizing the pre-Hamiltonian at every time.

In many control problems this implicit definition yields an explicit expression. That is, there is a function $\mu : y, p, t \to \mu(y, p, t)$ such that:

$$
u(t) \in \arg\min_u H(y, p, u, t) \iff u(t) = \mu(y, p, t). \tag{3}
$$

For instance, [4] shows that in Goddard's problem, the minimization condition implies that the control (i.e. the thrust vector) has the same orientation as a part of the co-state and that the magnitude of the control tends to be either saturated or zero based on a switching function that depends solely on the state and co-state (which leads to a bang-off-bang control as seen on Figure 1). Thus, the control in Goddard's problem is determined by the state and the co-state. In the double integrator problem defined in Section 5, the control minimizes the pre-Hamiltonian $H(y, p, u) = ||u||^2/2 + p_v \cdot u$, which yields the explicit expression $u(t) = \mu(y, p, t) = -p_v$.

[4] Also known as Pontryagin's Maximum Principle. There are multiple formalization of this principle in the literature.

Noting $g^y(y,p,t) = f(y,\mu(y,p,t),t)$ and $g^p(y,p,t) = -\partial H/\partial y(y,\mu(y,p,t),p,t)$. The equations of the PMP can be combined with those of Problem (1) to form the following two point boundary value problem:

$$\begin{cases} \dot{y}(t) &= g^y(y(t),p(t),t) \\ \dot{p}(t) &= g^p(y(t),p(t),t) \\ y(\tau) = y_\tau, & \quad p(\tau) \quad \text{is free} \\ y(T) \quad \text{is free}, & \quad p(T) = \dfrac{\partial \Psi}{\partial y}(y_T) \end{cases} \tag{4}$$

This is a problem of finding the value of $p(\tau)$ and $y(T)$ such that integrating from the state at time τ leads to the state at time T and vice versa. In many applications, it is enough to solve the Initial Value Problem (IVP), that is finding $p(\tau)$. By splitting the flow of Section 3.1.1, note $\phi^y_{\tau,T}(y_\tau,p_\tau)$ the flow function that returns the final state $y(T)$ and $\phi^p_{\tau,T}(y_\tau,p_\tau)$ the flow function that returns the final co-state $p(T)$. For a given y_τ, the IVP can be written as follow:

$$\text{find } p_\tau \text{ such that } C(y_\tau,p_\tau) = \phi^p_{\tau,T}(y_\tau,p_\tau) - \frac{\partial \Psi}{\partial y}\left(\phi^y_{\tau,T}(y_\tau,p_\tau)\right) = 0. \tag{5}$$

The IVP (5) is solved with a shooting method. First a guess of the initial co-state is taken. Then the system is *shot*, i.e. the ODE is integrated until the final time. Then the initial guess is corrected depending on how far the system landed from the target. This is repeated until a satisfactory co-state is found.

A shooting method can be complemented with a continuation method[5] when the problem is hard to initialize. See [4, 5] for the application of this method to a launcher system.

It is worth noting that the more general PMP states that if the dynamics do not depend on time, then the pre-Hamiltonian is stationary. This is not the case under our assumption since the dynamics depend on parameters that depend on time. It follows that the method used in [4, 5] to compute singular arc of the control, which is based on the stationarity of the pre-Hamiltonian, cannot be used under our assumption. Hence we currently have no method to enclose singular arcs, apart from a very crude over-approximation.

3.1.3 First derivative of the IVP

One way to solve the OCP is to apply a zero finding method to Equation (5). It can be useful to compute the derivatives of $\phi^y_{\tau,T}(y_\tau,p_\tau)$ and $\phi^p_{\tau,T}(y_\tau,p_\tau)$.

Let the variable $x(.) = (y(.),p(.))$. This variable is subject to an ODE $\dot{x} = g(x,t)$ as seen in System (4). It follows that if g is twice differentiable, a resolvent $R^x_{\tau,T}$

[5] also known as homotopic method

can be computed for this system using Equation (2). The resolvent $R^x_{\tau,T}$ is split in four square Matrices:

$$R^x_{\tau,T} = \begin{pmatrix} * & R^y_{\tau,T} \\ * & R^p_{\tau,T} \end{pmatrix}.$$

The two square matrices marked by $*$ are related to the final deviation caused by a variation of the initial state and are not useful here. $R^y_{\tau,T}$ (resp. $R^p_{\tau,T}$) is the final state (resp. co-state) deviation caused by a variation of the initial co-state and is the spatial derivative of $\phi^y_{\tau,T}(y_\tau, p_\tau)$ (resp. $\phi^p_{\tau,T}(y_\tau, p_\tau)$).

By splitting the boundary condition $R^x_{\tau,\tau} = I_d$ and taking the upper right and bottom right corner, boundary condition $R^y_{\tau,\tau} = 0$ and $R^p_{\tau,\tau} = I_d$ are obtained. Similarly, splitting the dynamic $\dot{R}_{\tau,t}(x_\tau) = \partial f/\partial x(x(t)) \cdot R_{\tau,t}(x_\tau)$ yields the dynamics of $R^y_{\tau,T}$ and $R^p_{\tau,T}$. Hence ODE (6) is deduced.

$$\begin{cases} \dot{y}(t) &= g^y(y(t), p(t), t) \\ \dot{p}(t) &= g^p(y(t), p(t), t) \\ \dot{R}^y_{\tau,t} &= \dfrac{\partial g^y}{\partial y}(y,p,t) \cdot R^y_{\tau,T} + \dfrac{\partial g^y}{\partial p}(y,p,t) \cdot R^p_{\tau,T} \\ \dot{R}^p_{\tau,t} &= \dfrac{\partial g^p}{\partial y}(y,p,t) \cdot R^y_{\tau,T} + \dfrac{\partial g^p}{\partial p}(y,p,t) \cdot R^p_{\tau,T} \\ y(\tau) &= y_\tau \\ p(\tau) &= p_\tau \\ R^y_{\tau,\tau} &= 0 \\ R^p_{\tau,\tau} &= I_d, \end{cases} \tag{6}$$

By simulating ODE (6), the value $C(y_\tau, p_\tau)$ and first derivative $\partial C/\partial p(y_\tau, p_\tau)$ of the IVP are obtained.

3.2 Validation methods

First, the general idea of interval arithmetic is recalled. Then two methods based on this arithmetic are presented: Krawczyk method that encloses the solution of a vector-valued equation and validated simulation that encloses the evolution of a dynamical system.

3.2.1 Set representation using interval arithmetic

An interval $[a]$ is a convex subset of $\mathbb{R}$ that contains all reals between a lower bound $\underline{a}$ and an upper bound $\overline{a}$. With $\underline{a}, \overline{a} \in \mathbb{R} \cup \{-\infty, +\infty\}$. Interval arithmetic can be used as an alternative to floating-point arithmetic to obtain an enclosure of the solution of a problem, rather than an approximation [10].

For each real valued function $f : a \mapsto f(a)$, one can create an inclusion function $[f] : [a] \mapsto [f]([a])$ following a set-membership principle: the result of the interval function $[f]$ is an interval that contains each possible value of f on $[a]$

$$[f]([a]) \supset \{f(a) | \forall a \in [a]\}.$$

This definition has an inclusion rather than an equality because the set on the left might not be an interval. Moreover, finding the minimal enclosure of the actual set is difficult. The tightness of the results depends on the effort put into their computations. For this reason, there may be multiple inclusion functions for a single real valued function.

Interval vectors, or boxes, are an axis-aligned rectangular set in a finite dimensional space. They are an inexpensive representation of a high dimensional set (compared to polytops) but may induce a wrapping effect during computations [9].

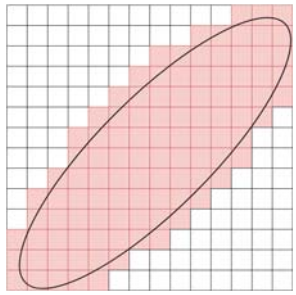


Figure 2: Outer tiling of a set. The ellipsoidal set is enclosed by the union of red boxes.

In lower dimension, an accurate enclosure of a set can be achieved with a paving of boxes, a set of mutually disjoint boxes which together cover the whole set. A simple paving is illustrated on Figure 2. A paving in which boxes are all the same shape and aligned can be called a tiling. Such a representation is potentially very precise, but the computational cost grows very fast in high dimension. Indeed, to double the precision, the number of tiles has to be doubled in each direction, which means 2^n as many tiles, where n is the dimension of the vector space. Hence the complexity of this representation is exponential in the dimension of the state, which makes it ill-suited for many practical cases.

3.2.2 Krawczyk contractor

Krawczyk's method is an interval based quasi-Newton algorithm. The formulation used is inspired by [9]. This method encloses the solutions of an equation $C(a) = 0$. It uses a **contractor** $[K]$, an operator which takes an input box and outputs a smaller box when possible. This contractor is built using $[\partial C/\partial a]([a])$, an enclosure of the first derivative of C on $[a]$ in the form of an interval matrix.

Let mid ([a]) the middle point of [a], [K] is defined as follows:

$$[K]([a]) = \text{mid}\,([a]) - M \cdot C(\text{mid}\,([a])) + \left(I_d - M \cdot \left[\frac{\partial C}{\partial a}\right]([a])\right) \cdot ([a] - \text{mid}\,([a])), \quad (7)$$

where M is an invertible real matrix, typically $M = \text{mid}\,([\partial C/\partial a]([a]))^{-1}$.

If $a \in [a]$ is such that $C(a) = 0$, then $a \in [K]([a])$. If $[K]([a])$ is contained in the interior of $[a]$ then there is a unique solution in $[a]$ or none.

To sum up, Krawczyk contractor is the function: $[a] \mapsto [a] \cap [K]([a])$, Krawczyk's method is made of the following steps: i) Initialize $[a]$ with the search area. ii) Repeat $[a] \leftarrow [a] \cap [K]([a])$ until convergence. Its output is a box containing all solution of $F(a) = 0$ in $[a]$.

3.2.3 Validated simulation

A set membership ODE is considered:

$$\begin{cases} \dot{x}(t) & \in [f](x,t) \\ x(t_0) & \in [x_0]. \end{cases}$$

Validated simulation encloses every solution of this system in a sequence of boxes (it encloses the solution for any realization of the initial state $x_0 \in [x_0]$ with any realization of the dynamics $f(x) \in [f(x)]$).

To that end, the time range $[0, T]$ is discretized in $(t_i)_{i \in 0..N}$, $t_0 = 0, t_N = T$. Starting with an enclosure $[y_i]$ of the systems at time t_i, an enclosure of the system on the whole time range $[t_i, t_{i+1}]$ (called a Picard box) is built. Then an enclosure of $y(t_{i+1})$ is computed. In [1], interval Runge-Kutta method are used coupled with inflating terms that enclose the truncation error of the method. Indeed, if the dynamics are sufficiently differentiable, the truncation error can be bounded by evaluating the Lagrange remainder of the difference between the Taylor series of the actual solution and the Taylor series of the Runge-Kutta approximation. For instance, if dynamics are four time differentiable, the truncation error of Runge-Kutta 4 may be enclosed.

The output of validated simulation resembles Figure 10, with the plain boxes being state boxes and the dashed boxes being Picard boxes.

Validated simulation is akin to simulating multiple systems between two common time stamps. A time switch is a time horizon shared by all systems, hence all systems can be simulated by doing a first simulation up until the switch and another simulation starting right after the switch. Contrarily, a variable time horizons or a state dependent transitions differs from one system to another. As a consequence, there is no shared time stamp to use as a duration for the simulation, which makes validated simulation challenging.

4 Computation of three informative enclosures

In this section, the main contribution is presented. An interval valued OCP solver is proposed and is used to build enclosures of the trajectory of a concrete system using the OCP as a controller.

4.1 Interval OCP solver based on the PMP

As per our assumption, there are multiple possible initial states and multiple possible dynamics which means infinitely many possible solutions. Hence the following set membership method to solve the OCP is proposed.

Assumption The presented method requires the dynamics g of the IVP (4) to be at least twice differentiable and to be $k+1$ times differentiable if a validated simulation method of order k is used. Indeed, Section 3.1.3 states that if g is twice differentiable, then a first derivative of the IVP (5) may be computed with ODE (6). As ODE (6) involves $\partial g/\partial x$, if g is $k+1$ times differentiable, then ODE (6) is k times differentiable and a method of order k may be used. This assumptions hold if f, μ and ℓ are infinitely differentiable, as will be the case in Section 5.

Method Based on the statements of Section 3.1.2, at a given time τ any couple (y_τ, p_τ) that satisfies the optimality condition $C(y_\tau, p_\tau) = 0$ is considered optimal. They will be referred as optimal (state, co-state) couples.

For a given enclosure of the state $[y_\tau]$, a Krawczyk operator of the function $p_\tau \to C(y_\tau, p_\tau)$ is built using Formula (7):

$$\begin{aligned} K([p_\tau]) &= \operatorname{mid}([p_\tau]) + M \cdot [C]([y_\tau], \operatorname{mid}([p_\tau])) \\ &\quad + \left(I_d - M \cdot \left[\frac{\partial C}{\partial p}\right]([y_\tau], [p_\tau])\right) \cdot ([p_\tau] - \operatorname{mid}([p_\tau])), \end{aligned} \tag{8}$$

where:

- $M = \operatorname{mid}\left(\left[\frac{\partial C}{\partial p}\right]([y_\tau], [p_\tau])\right)^{-1}$,
- $[C]([y_\tau], [p_\tau]_m) = [\phi]^p_{\tau,T}([y_\tau], [p_\tau]_m) - \left[\frac{\partial \Psi}{\partial y}\right]\left([\phi]^y_{\tau,T}([y_\tau], [p_\tau]_m)\right)$,
- $\left[\frac{\partial C}{\partial p}\right]([y_\tau], [p_\tau]) = \left[R^p_{\tau,T}\right]([y_\tau], [p_\tau]) - \left[\frac{\partial^2 \Psi}{\partial^2 y}\right]([y_\tau], [p_\tau]) \cdot \left[R^y_{\tau,T}\right]([y_\tau], [p_\tau])$.

The enclosure $\left[R^y_{\tau,T}\right]([y_\tau],[p_\tau])$ and $\left[R^p_{\tau,T}\right]([y_\tau],[p_\tau])$ of the resolvents can be computed with an analytic formula if it exists, or by integrating ODE (9), which is an enclosure of ODE (6).

$$\begin{cases} \dot{y}(t) & \in [g^y](y(t),p(t),[\xi]) \\ \dot{p}(t) & \in [g^p](y(t),p(t),[\xi]) \\ \dot{R}^y_{\tau,t} & \in \left[\frac{\partial g^y}{\partial y}\right](y,p,[\xi]) \cdot R^y_{\tau,T} + \left[\frac{\partial g^y}{\partial p}\right](y,p,[\xi]) \cdot R^p_{\tau,T} \\ \dot{R}^p_{\tau,t} & \in \left[\frac{\partial g^p}{\partial y}\right](y,p,[\xi]) \cdot R^y_{\tau,T} + \left[\frac{\partial g^p}{\partial p}\right](y,p,[\xi]) \cdot R^p_{\tau,T} \\ y(\tau) & \in [y_\tau] \\ p(\tau) & \in [p_\tau] \\ R^y_{\tau,\tau} & = 0 \\ R^p_{\tau,\tau} & = I_d, \end{cases} \tag{9}$$

The enclosure $[\phi]_{t_0,t_1}([y_0],\mathrm{mid}\,([p_0]))$ of the flow can be computed by integrating:

$$\begin{cases} \dot{y}(t) & \in [g^y](y(t),p(t),[\xi]) \\ \dot{p}(t) & \in [g^p](y(t),p(t),[\xi]) \\ y(\tau) & \in [y_\tau] \\ p(\tau) & = \mathrm{mid}\,([p_\tau])\,. \end{cases} \tag{10}$$

Algorithm 1 Computes $[K]([p_\tau])$ for given $[y_\tau]$ and $[p_\tau]$

Require: $[y_\tau]$ and $[p_\tau]$
Compute $\left[R^y_{\tau,T}\right][y_\tau],[p_\tau]$ and $\left[R^p_{\tau,T}\right]([y_\tau],[p_\tau])$
Compute $[\phi]_{\tau,T}([y_\tau],\mathrm{mid}\,(p_\tau))$
Compute $[K]([p_\tau])$ using Formula (8)

Our OCP solver is an implementation of Krawczyk's method. Algorithm 1 computes the Krawczyk operator, either with analytic formulae or by integrating System (9) and System (10). Algorithm 2 is a Krawczyk's method that outputs $\emptyset$ if there is no solution to $C((y_\tau,p_\tau)=0$. The convergence criterion is Hausdorff distance, which is a distance between sets particularly adapted to boxes [9].

Algorithm 2 needs to be initialized with a box $([p_\tau])_0$. This initial box depends on the type of enclosure built. As explained in Section 4.3, the anticipative, open-loop and closed-loop enclosures are computed using Algorithm 2 with different initialization.

Algorithm 2 Computes a thin enclosure $([p_\tau])_k$ of the optimal co-states corresponding to a given state enclosure $[y_\tau]$

Require: An initial searching area $([p_\tau])_0$ and $[y_\tau]$
  $([p_\tau])_k \leftarrow ([p_\tau])_0$
  Compute $[K](([p_\tau])_k)$ using algorithm 1
  while $D_{\text{Hausdorff}}(([p_\tau])_k, ([p_\tau])_k \cap [K](([p_\tau])_k)) >$ precision **do**
    if $0 \notin C(([y_\tau], [p_\tau]))$ **then**
      Return $\emptyset$
    else
      $([p_\tau])_k \leftarrow [K](([p_\tau])_k) \cap ([p_\tau])_k$
      Compute $[K]([p])$ using algorithm 1
    end if
  end while

4.2 Three enclosures to analyze the problem

For each realization of the initial condition parameters, there are an optimal control and a trajectory that are solution of the OCP. As a consequence, an OCP with interval uncertainties defines infinitely many controls and trajectories.

Similarly to many algorithms in the interval arithmetic literature that search a thin box containing all possible solutions of a problem, the anticipative enclosure contains all trajectories that are solution of an OCP.

Anticipative enclosure: For any parameter function $\xi(.) : [0, T] \to [\xi]$ and any initial state $y_0 \in [y_0]$, the anticipative enclosure contains the trajectory:

$$\begin{cases} \dot{y}(t) = f(y(t), u(t), \xi(t)) \\ y(0) = y_0, \end{cases}$$

with $u(.)$ the solution of the OCP:

$$\min_{u(.)\in\mathcal{U}} \int_0^T \ell(y(t), u(t))dt + \Psi(y(T)) \text{ s.t. } \begin{cases} \dot{y}(t) = f(y(t), u(t), \xi(t)) \\ y(0) = y_0. \end{cases}$$

$\xi(.)$ corresponds to both the actual realization of the parameters and the estimation made by the system. It is an ideal case in which the actual parameters are known with perfect accuracy. As such, this enclosure gives little information on the trajectory of a concrete system. In practice, parameter uncertainties will cause the system to deviate from its optimal trajectory and exit this enclosure.

The following two enclosures are proposed so as to inform on the behavior of the system in practical cases. They are meant to enclose the trajectory of a concrete system that computes its control by solving the OCP with an inaccurate model.

Open loop enclosure : It encloses a system which follows blindly an initial solution that was computed with inaccurate parameters. For any couple of parameters functions $\xi(.), \hat{\xi}(.) : [0,T] \to [\xi]$, and any initial state $y_0 \in [y_0]$, the open loop enclosure contains the trajectory:

$$\begin{cases} \dot{y}(t) = f(y(t), \hat{u}(t), \xi(t)) \\ y(0) = y_0, \end{cases}$$

with $\hat{u}(.)$ the solution of the OCP:

$$\min_{\hat{u}(.)\in\mathcal{U}} \int_0^T \ell(\hat{y}(t), \hat{u}(t))dt + \Psi(\hat{y}(T)) \text{ s.t. } \begin{cases} \dot{\hat{y}}(t) = f(\hat{y}(t), \hat{u}(t), \hat{\xi}(.)(t)) \\ \hat{y}(0) = y_0. \end{cases}$$

$\xi(.)$ corresponds to the actual realization of the parameters, which is unknown. Hence the system has an inaccurate estimation $\hat{\xi}(.)$. This is close to worst case analysis, as the worst case being the application of a control tailored for an extreme scenario to the opposite extreme scenario. This enclosure emphasizes the worst under or over-shooting possible.

It is possible to make a less pessimistic enclosure by taking into account the fact that the system can correct its trajectory using sensor data.

Closed loop enclosure : It encloses a system that uses a perfect measure of its state to recompute the solution of the OCP online (but with inaccurate parameters). Consider a list of recomputation time $(\tau_k)_{k\in 0..K}$, $0 = \tau_0 < \tau_1 .. < \tau_K = T$. For any couple of parameters functions $\xi(.), \hat{\xi}(.) : [0,T] \to [\xi]$, and any initial states $y_0 \in [y_0]$, the closed loop enclosure contains the trajectory of the piecewise-defined system:

$$\begin{cases} \dot{y}(t) = f(y(t), \hat{u}_k(t), \xi(t)), t), \forall t \in [\tau_k, \tau_{k+1}] \\ y(0) = y_0, \end{cases}$$

with $\hat{u}_k(.)$ the solution of the OCP:

$$\min_{\hat{u}(.)\in\mathcal{U}} \int_{\tau_k}^T \ell(\hat{y}(t), \hat{u}(t))dt + \Psi(\hat{y}(T)) \text{ such that } \begin{cases} \dot{\hat{y}}(t) = f(\hat{y}(t), \hat{u}(t), \hat{\xi}(t)) \\ \hat{y}(\tau_k) = y(\tau_k). \end{cases}$$

The overall control is made of pieces $\hat{u}_k(.)$ that are computed with an accurate measurement of the state $y(\tau_k)$. As in the open-loop enclosure, $\xi(.)$ corresponds to the actual realization of the parameters, which is unknown, and $\hat{\xi}(.)$ is an inaccurate estimation. This encloses the actual operation of a system with an optimal control regulator. The system does not have access to the value of the parameters, but it compensates using measures of its state.

In this paper, a finite set of recomputation time $(\tau_k)_{k\in 0..K}$ is considered. Under this assumption the closed loop enclosure is a sequence of open loop enclosures. Future works will investigate scenarios in which the control is recomputed at every time, which causes the closed loop to have a more unique geometry.

The computation of these enclosures is presented in Section 4.3.

4.3 The underlying geometry of these enclosures

A simpler system is considered to emphasize the geometry of the enclosures presented in Section 4.2. The following figures are depiction of the OCP:

$$\min_{u(.)\in\mathcal{U}} \int_\tau^T \frac{u^2}{2}dt + K\frac{(y(T)-y_T)^2}{2} \text{ s.t. } \begin{cases} \dot{y}(t) = \xi(t)u, \\ y(\tau) = y_\tau, \\ T \text{ is fixed.} \end{cases}$$

with $\forall t, \xi(t) \in [\xi]$, an uncertain parameter. As the state is one dimensional, state and co-state can be drawn on a single graph to showcase the important sets.

Open loop enclosure. As seen in Section 3.1.2, in the real case, the optimal control and trajectory are found by first solving the IVP (5) then integrating System (4).

To build the open-loop enclosure, an enclosure of the solution of the IVP is computed, then the evolution of System (4) is enclosed. If the initial state is such that $y_\tau \in [y_\tau]$, then the (state, co-state) couples can be anywhere in $[y_\tau] \times \mathbb{R}^n$, the orange vertical strip on Figure 3. The (state, co-state) couples that are solution of the IVP (5) are enclosed in the blue cone. The solutions of the OCP lie in the intersection (in red) of these two sets. If Algorithm 2 is initialized with a box $([p_\tau])_0$ that is large enough to contain this intersection, then it computes a box enclosure of this intersection.

Once $[p_\tau]$ has been computed, the evolution of System (4) is enclosed using validated simulation during a time range dt. This integration of System (4) is shown on Figure 4. This yields a set of possible state, co-state) couples at time $\tau + dt$, which in turn gives an enclosure of the possible states $[y_{\tau+dt}]$. By going on until the final time T, the open-loop enclosure is computed.

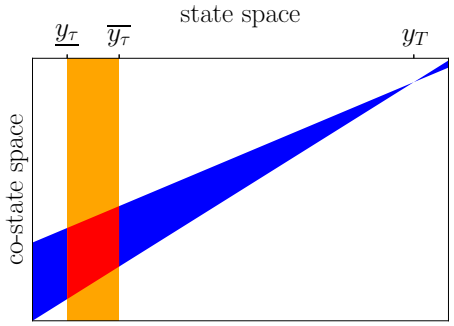


Figure 3: Geometric resolution of the OCP at time τ.

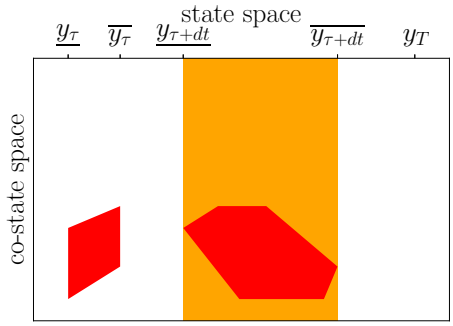


Figure 4: Simulation of the System (4) from time τ to time $\tau + dt$.

Closed loop enclosure. If the system recomputes its control at $\tau + dt$ using accurate measurements from the sensors, the OCP is solved again with an initial state in $[y_{\tau+dt}]$. This is illustrated in Figure 5, which is similar to Figure 4. An important point is that the red set of recomputed (state, co-state) couples is not contained in the purple set obtained by integrating System (4) from a prior step. This is due to the fact that the prior co-states were computed with the wrong parameters. Replacing them is a correction. Hence, these trajectories do not respect the equations of the PMP: applying a control that is optimal for a faulty model results in a non optimal trajectory.

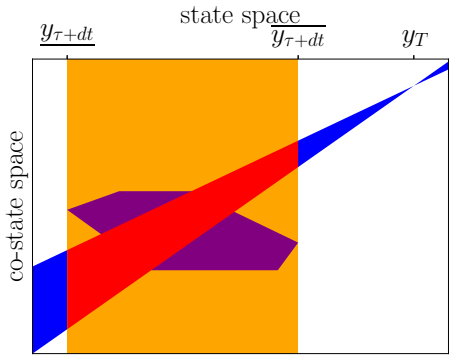


Figure 5: Closed-loop: recomputation of the optimal co-states at time $\tau + dt$.

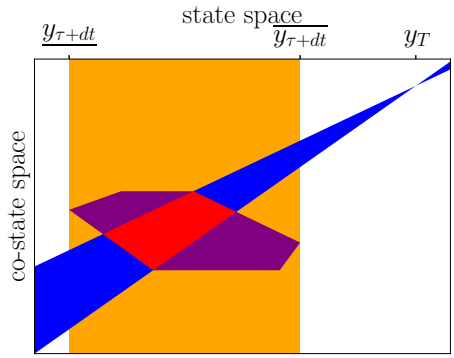


Figure 6: Anticipative: refining of the optimal couples at time $\tau + dt$.

Anticipative enclosure. To characterize the actual optimal trajectories, the following criterion is applied. If a trajectory is optimal from A to B and C is an intermediate state of that trajectory, then the trajectory is optimal from A to C and from C to B. If A is the state at time τ, B is the state at time T and C is the state at time $\tau + dt$, then its (state, co-state) couple at time $\tau + dt$ satisfies two properties.

- Optimality condition between A and C: the couple lies in the purple polyhedron on Figure 6, which is the set obtained by integrating System (4) from step τ.
- Optimality condition between C and B: the couple is in the blue cone on Figure 6, which is the set of solutions of the IVP (4) with the initial time $\tau + dt$.

By taking the intersection of these two sets, the set of optimal (state, co-state) couples is refined. This refinement can be done by initializing Algorithm 2 with $([p_{\tau+dt}])_0$ the projection of the purple polyhedron on co-state space.

5 Experimentations

The computation of the three enclosures is showcased on a double integrator with quadratic cost.

5.1 A naive implementation

Since Krawczyk's Algorithm 2 operates with boxes, the state representation has to be a box or a set of boxes. Representing the set of possible states by a box is computationally efficient but causes crippling wrapping effects. Instead, we propose a naive implementation that encloses the state at time t in a tiled bounding box, such as in Figure 2.

To compute an enclosure of state at time $t+dt$, for each tile that might contain a possible state, the OCP is solved (see Figure 7) and then the system is integrated. A first round of OCP resolution and validated integration is done to compute a bounding box of the state at time $t + dt$, then this bounding box is tiled uniformly and a second round finds which tiles might contain a solution.

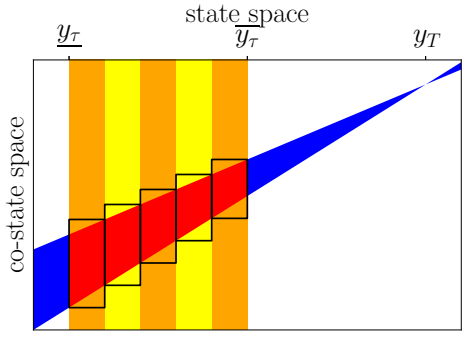


Figure 7: Paving the state enclosure to get a thinner approximation.

Each tile requires several calls to Algorithms 1 and 2 , and since the complexity of the tiling is exponential in the dimension, this naive method is not adapted to complicated systems.

5.2 Numerical results on a double integrator

Consider a double integrator in a uniform gravity field and with a reactor thrust. It is subject to a quadratic continuous cost and a quadratic penalization on the final state.

$$\min \int_0^T \frac{||u||^2}{2} dt + K_v \frac{||v(T)||^2}{2} + K_r \frac{||r(T) - r_T||^2}{2} \quad \text{such that} \quad \begin{cases} \dot{r}(t) = v \\ \dot{v}(t) = -\frac{C}{m} u - G e_2 \\ r(0) = r_0, \quad v(0) = v_0, \end{cases} \tag{11}$$

where G is the normalized gravity field, e_2 is the unit vector of the vertical axis, C is the maximum thrust and m is the mass of the system, which is assumed to be constant.

By using the method in Section 3.1.2, this OCP turns into the following two point boundary value problem:

$$\begin{cases} \dot{r}(t) = v \\ \dot{v}(t) = -p_v \left(\frac{C}{m}\right)^2 - Ge_2 \\ \dot{p_r}(t) = 0 \\ \dot{p_v}(t) = -p_r \end{cases} \tag{12}$$
$$r(0) = r_0, \qquad v(0) = v_0$$
$$p_r(T) - K_r r(T) = 0, \quad p_v(T) - K_v v(T) = 0.$$

The values used are inspired by the take-off problem in [4]. The parameter nominal values are the same as in [4]. The initial position of the system corresponds to the final position of the take-off mission. The initial velocity has been chosen to be coherent with a re-entry mission.

The solution of the nominal case is presented on Figure 8.

The initial position has an uncertainty of around 5 km, which is about 2% of the span of the trajectory. The initial speed has a relative uncertainty of 1.7% and the parameter ratio C/m has a relative uncertainty of 0.2%. However, the mission duration is greater than the optimal duration so as to emphasize the differences between the enclosures. The longer the mission, the more the system has to fight gravity, the more uncertainties are accumulated along the vertical axis.

Figures 9, 10 and 11 depict enclosures of the double integrator computed using Algorithm 2 coupled with a tiling of the state enclosure (as shown on Figure 7). The initialization of Algorithm 2 depends on the type of enclosure, as explained in Section 4.3.

Since System (12) is linear time invariant with a strictly triangular matrix, there are analytic formulae for the flow and the resolvents. We use these analytic formulae in Algorithm 1. We have also simulated Systems (9) and (10) with DynIbex [1]: a C++ library for validated simulation. The precision of the results of the simulations matches those of the analytic formulae. However, these simulations drastically increase computation time, which in turn significantly restrict the precision of the tiling. As a consequence, Figures 9, 10 and 11 were done with analytic formulae rather than validated simulation.

The anticipative enclosure on Figure 9 starts wide because of initial uncertainty but gets thinner as all the system converge to the target. As a result, the final box is very thin.

Contrarily, the open-loop enclosure on Figure 10 becomes wider over time and the final box is very big. Systems over-shoot or under-shoot the target because of their lack of correction.

Lastly, the closed loop enclosure on Figure 11 is somewhere in between the two other enclosures. These systems deviate from their optimal trajectory but correct

it, leading to a bulge on the middle and end of the trajectory that gets abruptly smaller at the end.

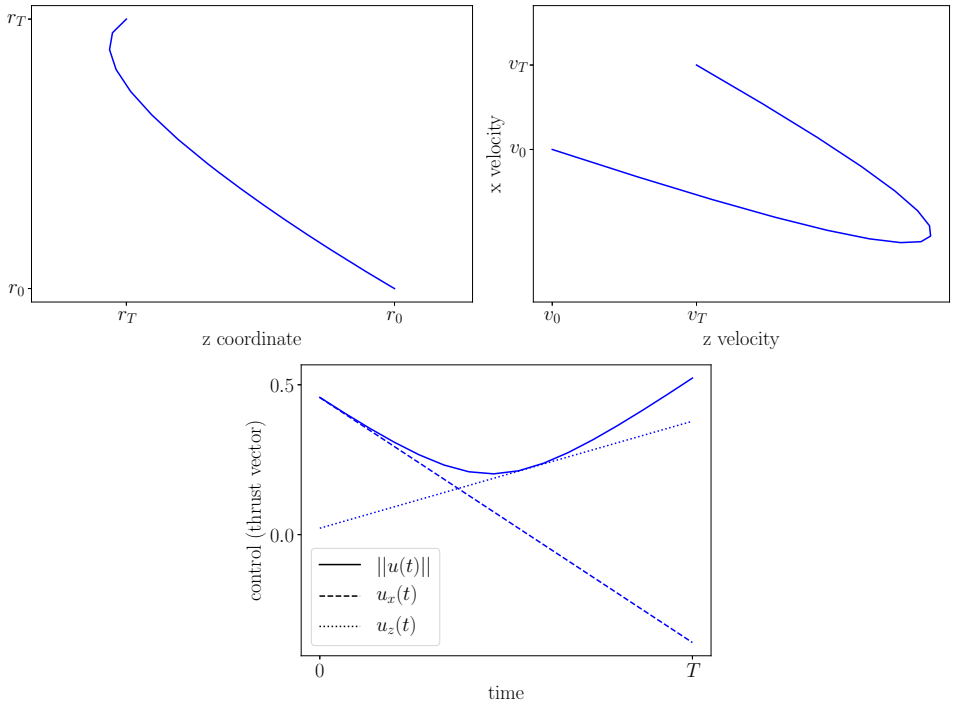


Figure 8: Optimal trajectory, velocity space trajectory and control for the a double integrator with quadratic cost. Unlike Goddard's problem on Figure 1, the control changes continuously.

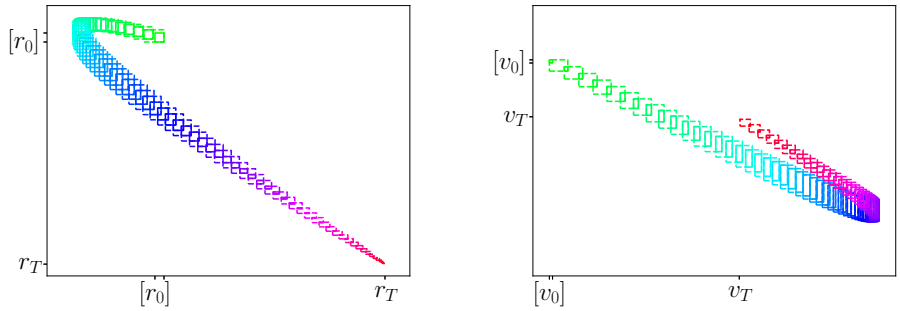


Figure 9: Anticipative enclosure of position and velocity.

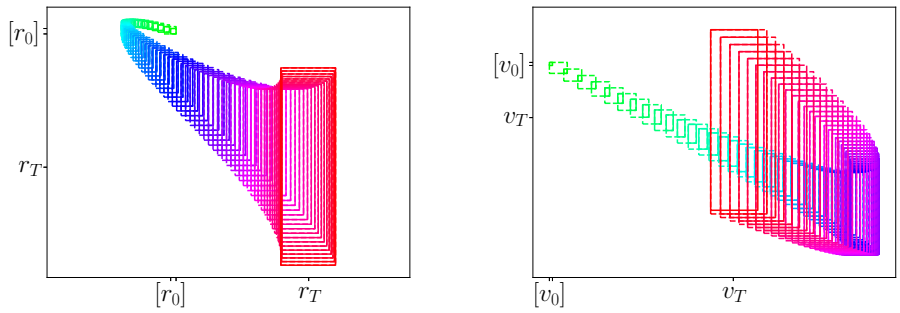


Figure 10: Open loop enclosure of position and velocity.

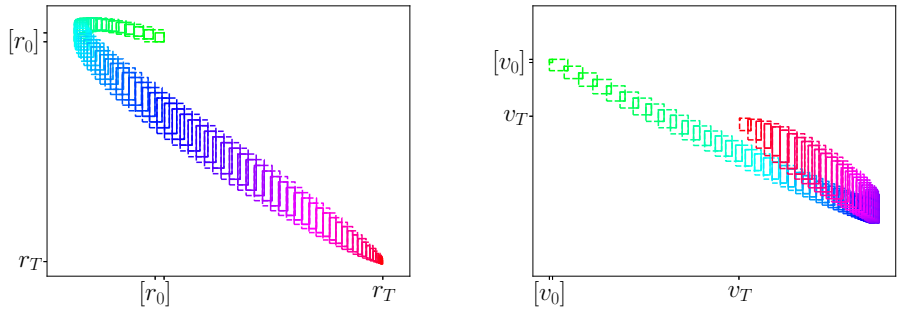


Figure 11: Closed loop enclosure of position and velocity.

6 Discussion

In this section, we propose some uses of the enclosures, then list what needs to be done to address more complex problems.

6.1 Risk assessment and indicating the relevance of an OCP

These enclosures are similar to those developed in [2]. As such, they can have the same application. Consider the example of final constraints: the final state must be in a safe zone. The final position must be in the recuperation platform and the final velocity must be sufficiently small for the system to land. A similar reasoning can be held for a state constraint, or other requirements of a mission.

If the final state box of the closed loop enclosure (Figure 11) is contained in the safe zone, then a system using this OCP in closed loop is guaranteed to satisfy the final constraint. Similarly, it can be guaranteed that applying the solution of the OCP in open loop will satisfy the mission (although this is clearly not the case on Figure 10). Both are strong assessment of robustness. The cost can be enclosed as

well, which gives a worst performance index.

When using a floating-point control, the anticipative enclosure will provide an upper bound to the distance to the actual optimal control. As a consequence, this provide a criterion to discuss the validity of a floating-point control. To minimize this criterion, the middle point of the control box of the anticipative enclosure can be used as a control.

These enclosures could be used to design the model or the concrete system. For instance, if the final enclosure of the velocity of the anticipative enclosure (Figure 9) is not entirely contained in the safe zone, that means that there might be a realization of the initial states and the parameters such that the system will crash by following the optimal trajectory defined by the OCP.

The existence of such a realization is not guaranteed as those enclosures are conservative, nevertheless, for critical systems, the mere possibility of the existence of a failure may require the OCP to be changed.

In Problem (11), this could mean increasing the penalization coefficient K_r and K_v, thus making the launch more costly but decreasing the uncertainty on the final state and maybe void the possibility of failure.

Contrarily, if the open-loop or closed-loop enclosures final boxes respect the final constraint by a significant margin, then the penalization could be decreased to achieve better performances on other criteria. Alternatively, the uncertainty could be altered. One could assess the maximum acceptable magnitude of the uncertainties and use it as specification for the creation of an engine or sensors. Lastly, if the open loop enclosure respects all the constraints, then it might not be necessary to elaborate a closed loop regulator.

Of course, these statements hold true as long as the model with uncertainties is accurate. If parameter uncertainties have been under-estimated or if the model has neglected a phenomenon, then there is no guarantee that the system will remain in the enclosures.

6.2 Generalizing to Goddard's problem

The double integrator with quadratic cost was considered because it is used to initialize a continuation method to solve Goddard's problem [4, 5].

In a similar manner to a continuation method, there are two steps to bridge the gap between the double integrator and Goddard's problem. First non linear systems have to be addressed. We need to simulate non linear problems and their resolvent, which is costly. Then state dependent transitions and variable time horizons will need to be modeled and enclosed. This means being able to enclose the evolution of a hybrid system, if possible in a manner that does not induce a significant over-approximation. This also makes the computation of the resolvent more challenging.

In addition to that, Goddard's representation of a spacecraft is in dimension seven. As a consequence, tiling may no longer be used to improve the precision of Algorithm 2. An entirely new method may be needed.

7 Conclusion

In this paper, a resolution tool was proposed for optimal control problems with embedded uncertainties. This resolution tool is used to compute three enclosures, an anticipative enclosure containing all possible solutions of the optimal control problem, and open-loop and closed loop enclosures that bounds the trajectory of a concrete system that uses this problem as a controller. These enclosures can show the deviation caused by the uncertainties, identify the critical zone that could be crossed or more generally assess the relevance of a given optimal control problem.

In later works, we will try to improve our method until it can reliably compute these enclosures for a problem as complex as Goddard's.

Acknowledgment

This work was partially supported by the "Chair Complex Systems Engineering – Ecole polytechnique, Thales, DGA, FX, Dassault Aviation, Naval Group Research, ENSTA Paris, Télécom Paris, and Fondation ParisTech"

References

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

[2] Althoff, M. *Reachability Analysis and its Application to the Safety Assessment of Autonomous Cars*. PhD thesis, Technische Universität München, 2010.

[3] Blackmore, L. Autonomous precision landing of space rockets. *The Bridge*, 4(46):15–20, 01 2016.

[4] Bonnans, F., Martinon, P., and Trélat, E. Singular arcs in the generalized goddard's problem. *Journal of Optimization Theory and Applications*, 139(2):439–461, 2008. DOI: 10.1007/s10957-008-9387-1.

[5] Brendel, E., Hérissé, B., and Bourgeois, E. Optimal guidance for Toss Back concepts of Reusable Launch Vehicles. In *EUCASS*, 2019.

[6] Butcher, J. C. Coefficients for the study of runge-kutta integration processes. *Journal of the Australian Mathematical Society*, 3(2):185–201, 1963. DOI: 10.1017/S1446788700027932.

[7] Caillau, J.-B., Cerf, M., Sassi, A., Trélat, E., and Zidani, H. Solving chance constrained optimal control problems in aerospace via kernel density estimation. *Optimal Control, Applications and Methods*, 39(5):1818–1832, 2018. DOI: 10.1002/oca.2445.

[8] Coron, J.M. and American Mathematical Society. *Control and Nonlinearity*. Mathematical surveys and monographs. American Mathematical Society, 2007.

[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 London, 2001. DOI: 10.1007/978-1-4471-0249-6.

[10] Moore, Ramon E. *Interval analysis*, volume 4. Prentice-Hall Englewood Cliffs, 1966.

[11] Rauh, A. and Hofer, E. P. Interval methods for optimal control. In Buttazzo, G. and Frediani, A., editors, *Variational Analysis and Aerospace Engineering*, chapter 22, pages 397–418. Springer New York, New York, NY, 2009. DOI: 10.1007/978-0-387-95857-6_22.

[12] Rauh, A., Minisini, J., and Hofer, E. P. Interval techniques for design of optimal and robust control strategies. In *12th GAMM - IMACS International Symposium on Scientific Computing, Computer Arithmetic and Validated Numerics (SCAN 2006), Duisburg,*, 2006. DOI: 10.1109/SCAN.2006.27.

[13] Trélat, E. *Contrôle optimal: théorie et applications*. Vuibert, 2005.

[14] Trélat, E. Optimal control and applications to aerospace: Some results and challenges. *Journal of Optimization Theory and Applications*, 154(3):713–758, 2012. DOI: 10.1007/s10957-012-0050-5.

Contents

ISSN 0324—721 X (Print)
ISSN 2676—993 X (Online)

Editor-in-Chief: Tibor Csendes