Determination of Inner and Outer Bounds of Reachable Sets Through Subpavings

Francisco Rego, Elwin de Weerdt, Eddy van Oort,
Erik-Jan van Kampen, QiPing Chu, António M. Pascoal

May 30, 2018

1 Abstract

The computation of the reachable set of states of a given dynamic system is an important step to verify its safety during operation. There are different methods of computing reachable sets, namely interval integration, capture basin, methods involving the minimum time to reach function, and level set methods. This work deals with interval integration to compute subpavings to over or under approximate reachable sets.

An algorithm to over and under estimate sets through subpavings, which potentially reduces the computational load when the test function or the contractor is computationally heavy, is implemented and tested. This algorithm is used to compute inner and outer approximations of reachable sets. The test function and the contractors used in this work to obtain the subpavings involve guaranteed integration, provided either by the Euler method or by another guaranteed integration method.

The methods developed were applied to compute inner and outer approximations of reachable sets for the double integrator example. From the results it was observed that using contractors instead of test functions yields much tighter results. It was also confirmed that for a given minimum box size there is an optimum time step such that with a greater or smaller time step worse results are obtained.

2 Introduction

Computing reachable sets consists of determining which states can be reached by a particular system with known dynamics from a set of initial conditions (forwards reachable set), or from which initial conditions it is possible to reach a specified target set of states (backwards reachable set), as illustrated in Figure 1.

One method of estimating reachable sets is by simulating a single trajectory of the system at a time. However, this is a computationally expensive

method if we are dealing with systems with many different state values and input signals. Also, using only simulations, it is possible to miss important but isolated trajectories that lead to unsafe sets. To avoid those problems, there are methods of estimating reachable sets capturing the behavior of entire groups of trajectories at once. This concept of capturing the behavior of groups of trajectories is present in research work on Region of Attraction (RoA) analysis and reachability for systems with limited energy inputs, solving a Sum-of-Squares (SOS) relaxation [12]. Related work in [3] discusses the use of neural networks to approximate the solution of the Hamilton-Jacobi-Isaacs (HJI) PDE to obtain the evolution of reachable sets. A MATLAB level set toolbox tht approximates Hamilton-Jacobi-Isaacs PDEs through finite differences is described in Mitchell [9]. A method that aims to compute guaranteed bounds on the capture basin of a dynamical system using Interval Analysis is proposed in [7].

A solution to the problem of obtaining guaranteed bounds on reachable sets is involves the computation of reachable sets using wrappers, such as interval vectors, instead of points. Guaranteed integration using Interval Analysis is a widely developed field [11] which gives guaranteed solutions to Initial Value Problems (IVP). Interval analysis was applied also to several aerospace related topics such as finding trim-points for nonlinear aircraft models [14], pilot model identification [16], integer ambiguity resolution for aircraft attitude determination [15, 2], spacecraft re-entry optimization [4] and fuel optimization for constrained spacecraft formation rotations [1].

The paper is organized as follows: Section 3 gives a formal definition of reachable sets. Section 4 summarizes the theoretical background needed to compute approximations to reachable sets, Section 5 presents the proposed solution. Section 6 discusses the computational application of the method and Section 7 describes an application example. Conclusion and recommendations are drawn in Section 8.

3 Problem definition: computation of reachable sets.

In the present work we consider non-autonomous dynamical systems with control and disturbance inputs described by the set of equations

$$\begin{array}{c} \dot{x} = f(x(t), u(t), d(t)) \\ x(0) \in \mathcal{S}_0 \ (or\ x(t_f) \in \mathcal{T}_0),\ t \in [0, t_f] \end{array} \tag{1}$$

where $0 \leq t_f < \infty, x \in \mathbb{R}^n$ is the state, $u \in \mathbb{R}^m$ is the control input, $d \in \mathbb{R}^p$ is the disturbance input, $\mathcal{S}_0$ is an initial set of states, and $\mathcal{T}_0$ is a target set of states. The spaces of admissible control input and disturbance input trajectories are denoted as the spaces of piecewise continuous functions $\mathcal{U} = \{u(\bullet) \in PC^0 | u(t) \in \mathbb{U}, 0 \leq t \leq t_f\}$, and $\mathcal{D} = \{d(\bullet) \in PC^0 | d(t) \in \mathbb{D}, 0 \leq t \leq t_f\}$ respectively.

The control input is determined as a function of the disturbance through a strategy, which is a map $\gamma : \mathcal{D} \to \mathcal{U}$ specifying the control input as a function of

the disturbance signal. In this work we will consider non-anticipative strategies of the form

$$\gamma \in \{\beta : \mathcal{D} \to \mathcal{U} | d_1(r) = d_2(r) \forall r \in [0, s] \Rightarrow \beta[d_1](r) = \beta[d_2](r) \forall r \in [0, s]\}. \quad (2)$$

Formally, the forward and backwards reachable sets are defined as follows.

Definition 3.1 (Forwards Reachable Set)**.** *The forward reachable set $\mathcal{S}(\tau)$ of system (1) at time τ $(0 \leq \tau \leq t_f)$, from the initial set $\mathcal{S}_0$, is the set of all states $x(\tau)$ such that there exists a strategy $u(t) = \gamma[d](t) \in \mathcal{U}(\tau \leq t \leq t_f)$, for which $x(\tau)$ is reachable from some $x(0) \in \mathcal{S}_0$, for all disturbance inputs $d(t) \in \mathcal{D}(\tau \leq t \leq t_f)$, along a trajectory satisfying (1).*

Definition 3.2 (Backwards Reachable Set)**.** *The backward reachable set $\mathcal{T}(\tau)$ of system (1) at time $\tau(0 \leq \tau \leq t_f)$, from the target set $\mathcal{T}_0$, is the set of all states $x(\tau)$ such that there exists a strategy $u(t) = \gamma[d](t) \in \mathcal{U}(\tau \leq t \leq t_f)$, for which some $x(t_f) \in \mathcal{T}_0$ is reachable from $x(\tau)$, for all disturbance inputs $d(t) \in \mathcal{D}(\tau \leq t \leq t_f)$, along a trajectory satisfying (1).*

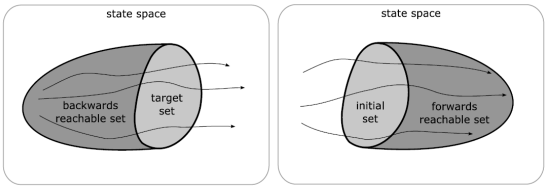

Figure 1: Backwards and forwards reachable sets.

4 The Approach adopted - Interval Analysis

4.1 Interval Analysis

Grid based methods for computing reachable sets do not provide guaranteed bounds, because it is possible to miss some important trajectory. One possible way to obtain guaranteed bounds is by using interval analysis, which deals with intervals instead of real numbers and extends some of the usual operations of real analysis into operations between intervals [10].

Two important characteristics of an interval $[x] = x \in \mathbb{R} | \underline{x} \leq x \leq \overline{x}$ are its width $w([x])$ and midpoint $m([x])$ defined by the following equations:

$$w([x]) = \overline{x} - \underline{x}, \quad (3)$$

$$m([x]) = \frac{\overline{x} + \underline{x}}{2}. \quad (4)$$

For interval vectors (also referred to as boxes) the width, midpoint, and volume are defined as follows:

$$w\left([x]\right) = \max_i w\left([x_i]\right), \tag{5}$$

$$m\left([x]\right) = \left(m\left([x_1]\right), \ldots, m\left([x_n]\right)\right)\right), \tag{6}$$

$$Vol\left([x]\right) = \prod_{(i=1)}^{n} w([x_i]). \tag{7}$$

Using interval analysis we can search for inclusion functions $[f]\left([x]\right)$ from intervals to intervals, such that if x is contained in $[x]$, $f(x)$ is contained in $[f]\left([x]\right)$. One property of an inclusion function is that it can be inclusion isotonic (that is, if $[x] \subseteq [y]$ then $[f]\left([x]\right) \subseteq [f]\left([y]\right)$).

Inclusion functions can have many forms such as interval extension, centered form, mean value form, slope form, and Taylor expansion [10]. The simplest form to have an inclusion function is to replace the operations in the expression of $f(x)$ by their equivalents on interval arithmetic. In this case we obtain an interval extension.

Interval arithmetic operations are an extension to intervals of the arithmetic operations between real numbers, namely sum, subtraction, multiplication, and division, and are computed as expressed in equations 8 to 11:

$$[a,b] + [c,d] = [a+c, b+d] \tag{8}$$

$$[a,b] - [c,d] = [a-c, b-d] \tag{9}$$

$$[a,b] * [c,d] = [\min(ab,ac,ad,bd), \max(ab,ac,ad,bd)] \tag{10}$$

$$\frac{[a,b]}{[c,d]} = [a,b] * [1/d, 1/c], \ if\ 0 \notin [c,d] \tag{11}$$

The rational interval functions are a subset of the interval valued functions defined according to definition 4.1.

Definition 4.1. *A rational interval function is an interval-valued function whose values are defined by a specific sequence of interval arithmetic operations.*

The next theorem holds for rational inclusion functions but it can be extended to various other functions well defined on interval analysis such as $sin\left([x]\right)$ or $ln\left([x]\right)$.

Theorem 4.2 ([10])**.** *If $[f]([x_1],[x_2],\ldots,[x_n])$ is a rational expression in the interval variables $[x_1]$, $[x_2]$,..., $[x_n]$, i.e., a finite combination of $[x_1]$, $[x_2]$,..., $[x_n]$ and a finite set of constant intervals with interval arithmetic operations, then*

$$[x_1]' \subset [x_1], [x_2]' \subset [x_2], \ldots, [x_n]' \subset [x_n] \tag{12}$$

implies

$$[f]([x_1]', [x_2]', \ldots, [x_n]') \subset [f]([x_1], [x_2], \ldots, [x_n]) \tag{13}$$

for every set of interval numbers $[x_1]$, $[x_2]$,..., $[x_n]$ *for which the interval arithmetic operations in* $[f]$ *are defined.*

From this theorem we can conclude that for smaller intervals where the inclusion function is evaluated $[x]$, the estimation on $f(x)$ provided by the inclusion function $[f]([x])$ is tighter than for large intervals.

For practical implementations of interval analysis there are several libraries, written in many different programming languages, that create a new interval variable type and define the basic functions on interval variables. Two of the most popular libraries that implement interval analysis are PROFIL\BIAS [6] for C++ and INTLAB for MATLAB [13].

When dealing with finite precision, there is a basic requirement that the result of a computation with finite precision encloses the real result. To satisfy this requirement, a library must make use of outward rounding, which consists of performing rounding towards $-\infty$ for the lower bound of an interval and rounding towards $+\infty$ for the upper bound.

4.2 Guaranteed Integration

It is possible to integrate interval functions in the same way we integrate real valued functions [10]. For an interval valued function $[f]([x])$ and $[x_i]$ a uniform subdivision of $[x] = [a, b]$, a definite sum S_N can be defined as

$$S_N([f]; [a, b]) = \sum_{i=1}^{N} ([x_i])\,(b - a)/N \tag{14}$$

We can also define the integral of $[f]([x])$ as follows:

$$\int_{[a,b]} [f](t)dt = \bigcap_{N=1}^{\infty} S_N([f]; [a, b]) \tag{15}$$

In practice, one way to compute an enclosure of the integral of an interval function is to use the above definition stopping at a finite N.

Using the above integrating method, it is possible to get an inclusion of the solution of an initial value problem, through Picard iterations. The homogeneous initial value problem, $\dot{x}(t) = f(x)$ and $x(0) = x_0$, is formulated as the integral equation:

$$x(t) = p(x)(t), \tag{16}$$

where

$$p(x)(t) = x_0 + \int_0^t f(x(s))\, ds \tag{17}$$

The interval function $[p]$ is a majorant of operator p if $p(x) \in [p]([x])$ for $x \in [x]$. The interval operator $[p]$ is inclusion isotonic if $[x] \subseteq [y]$ implies $[p]([x]) \subseteq [p]([y])$. If $[f]$ is an inclusion isotonic interval enclosure of f the following interval operator, the Picard iteration operator, is inclusion isotonic:

$$[p]([x])(t) = x_0 + \int_0^t [f]([x](s))\, ds \tag{18}$$

If $[x](t)$ is an inclusion function of the solution of an initial value problem then $[p]([x])(t)$ is also an inclusion function of the solution of an IVP and $[p]([x])(t) \subseteq [x](t)$. Therefore, a finite number of Picard iterations in 18 are desirable to obtain a solution to an initial value problem.

This method will be useful to compute the enclosures of the trajectories $\phi(t, x_0, u, d)$ on section 5.

4.3 Subpavings

A subpaving of a box $[x]_0 \subset \mathbb{R}^n$ is a union of non-overlapping subboxes of $[x]$ with non-zero width. Non-overlapping means that the intersection of two such subboxes must be empty or have zero volume, that is, they may only intersect at their boundaries. Subpavings are useful to provide over- ($\overline{\mathbb{X}}$) and under-approximations ($\underline{\mathbb{X}}$) of a set $\mathbb{X} \subseteq [x]$ such that $\underline{\mathbb{X}} \subseteq \mathbb{X} \subseteq \overline{\mathbb{X}}$. The theory associated with subpavings is thoroughly discussed in [5].

A subpaving of $[x]$ can be obtained through successive bisections and eliminations, and in this case it is called a regular subpaving. A bisection is a process of dividing a box into two smaller boxes of equal width. The left and right partitions, $L[x]$ and $R[x]$, are obtained as follows: let us consider the index j of the first component of $[x]$ of maximum width, that is, j is defined from (19).

$$j = \min\{i | w([x_i]) = w([x])\} \tag{19}$$

The boxes $L[x]$ and $R[x]$ are then defined as follows:

$$L[x] := \left([x_1], \ldots, \left[\underline{x_j}, m([x_j])\right], \ldots, [x_n]\right) \tag{20}$$

$$R[x] := ([x_1], \ldots, [m([x_j]), \overline{x_j}], \ldots, [x_n]) \tag{21}$$

To under- and over-approximate a set $\mathbb{X}$ by two subpavings $\underline{\mathbb{X}}$ and $\overline{\mathbb{X}}$, $\underline{\mathbb{X}} \subseteq \mathbb{X} \subseteq \overline{\mathbb{X}}$, an algorithm was developed in [5] named SIVIA (Set Inverter Via Interval Analysis). It utilizes an interval Boolean function $[test]([x])$ such that if the result is 1, then $[x] \subseteq \mathbb{X}$; if the result is 0, then $[x] \cap \mathbb{X} = \emptyset$, and the result is $[0, 1]$ if no conclusion can be drawn. Algorithm 4.3 presents the SIVIA algorithm in pseudo code.

In SIVIA, a natural choice for initializing $\underline{\mathbb{X}}$ and $\overline{\mathbb{X}}$ would be $\underline{\mathbb{X}} = \overline{\mathbb{X}} = \emptyset$. The choice of ϵ controls the precision of the enclosure, since the smallest boxes $[x]$ on the subpaving are the first boxes, obtained by successive bisections, for

Algorithm 1 SIVIA(in: test, $[x]$, ϵ; inout: $\underline{\mathbb{X}}$, $\overline{\mathbb{X}}$)

if $[\text{test}]([x]) = 1$ **then**
  $\underline{\mathbb{X}} \leftarrow \underline{\mathbb{X}} \cup [x]$
  $\overline{\mathbb{X}} \leftarrow \overline{\mathbb{X}} \cup [x]$
else if $w([x]) < \epsilon \wedge [\text{test}]([x]) \neq 0$ **then**
  $\overline{\mathbb{X}} \leftarrow \overline{\mathbb{X}} \cup [x]$
else if $[\text{test}]([x]) \neq 0$ **then**
  SIVIA(test, $L[x]$, ϵ, $\underline{\mathbb{X}}$, $\overline{\mathbb{X}}$)
  SIVIA(test, $R[x]$, ϵ, $\underline{\mathbb{X}}$, $\overline{\mathbb{X}}$)
end if

which $w([x]) < \epsilon$. A smaller ϵ yields a tighter enclosure but increases the computational time.

A possible way of improving the over-approximation is to use contractors instead of test functions. Contractors are functions $\forall [x], \mathcal{C}_{\mathbb{S}} : \mathbb{IR}^n \to \mathbb{IR}^n$ from interval vectors to interval vectors that satisfy the following conditions ($\mathbb{S}$ is a set associated with the contractor):

$$\forall [x], \mathcal{C}_{\mathbb{S}}([x]) \subseteq [x] \text{ (contractance)}, \tag{22}$$

$$\forall [x], [x] \cap \mathbb{S} \subseteq \mathcal{C}_{\mathbb{S}}([x]) \text{ (correctedness)}. \tag{23}$$

The SIVIA algorithm can be adapted for use with contractors, as in Algorithm 4.3, yielding a non-regular subpaving.

Algorithm 2 SIVIA(in: $\mathcal{C}_{\mathbb{X}}$, $[x]$, ϵ; inout: $\overline{\mathbb{X}}$)

$[x] \leftarrow \mathcal{C}_{\mathbb{X}}([x])$
if $w([x]) < \epsilon$ **then**
  $\overline{\mathbb{X}} \leftarrow \overline{\mathbb{X}} \cup [x]$
else
  SIVIA($\mathcal{C}_{\mathbb{X}}$, $L[x]$, ϵ, $\underline{\mathbb{X}}$, $\overline{\mathbb{X}}$)
  SIVIA($\mathcal{C}_{\mathbb{X}}$, $R[x]$, ϵ, $\underline{\mathbb{X}}$, $\overline{\mathbb{X}}$)
end if

5 The Proposed Solution

A procedure similar to that in [7] can be used to compute enclosures for reachable sets and is based on Theorem 5.1, illustrated in Figure 2.

Theorem 5.1. *Consider the box of states $[x] \in \mathbb{IR}^n$, the sample control input $u \in \mathcal{U}$, and the sample disturbance input $d \in \mathcal{D}$. If $\mathcal{T}^-(t)$ and $\mathcal{T}^+(t)$ are such that $\mathcal{T}^-(t) \subseteq \mathcal{T}(t) \subseteq \mathcal{T}^+(t)$ then*

$$[\phi](\tau, [x], u, [\mathbb{D}]) \subseteq \mathcal{T}^-(t) \Rightarrow [x] \subseteq \mathcal{T}(t - \tau) \tag{24}$$

$$[\phi](\tau,[x],[\mathbb{U}],d)\cap\mathcal{T}^{+}(t)=\emptyset\Rightarrow[x]\cap\mathcal{T}(t-\tau)=\emptyset. \tag{25}$$

Proof. Since $\phi\left(\tau,[x],\mathbb{U},d\right)\subseteq[\phi]\left(\tau,[x],\mathbb{U},d\right)$ and $\mathcal{T}(t)\subseteq\mathcal{T}^{+}(t)$, $[\phi]\left(\tau,[x],[\mathbb{U}],d\right)\cap\mathcal{T}^{+}(t)=\emptyset\Rightarrow\phi\left(\tau,[x],\mathbb{U},d\right)\cap\mathcal{T}(t)=\emptyset$.

From definition 3.2 there is at least one disturbance input such that, for all admissible control inputs, $x(t)\notin\mathcal{T}(t)$ from all states $x(t-\tau)\in[x]$.

Thus, for all states $x(t-\tau)\in[x]$, for every strategy $\gamma:\mathcal{D}\rightarrow\mathcal{U}$ there is at least one disturbance input $d(t)$ such that for a control input $\gamma[d](t),x(t_f)\notin\mathcal{T}_0$. This proves (25).

The proof of (24) is similar and is omitted here. □

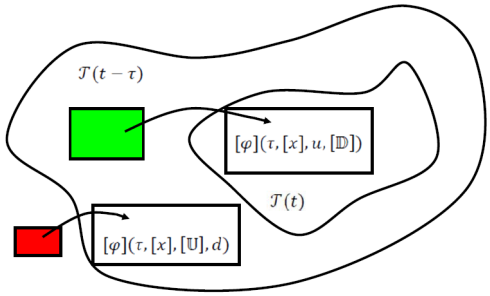

Figure 2: Illustration of Theorem 5.1

In Theorem 5.1 we consider a trajectory of the type $\phi(t,x_0,u,d)$ which is the solution $x(t)$ of the initial value problem $\dot{x}=f(x,u,d)$ for the initial vector x_0.

To get tighter bounds and to improve convergence, it is better to use contractors instead of test functions. For this purpose we can use Theorem 5.2, illustrated in Figure 3. In Theorem 5.2 we consider an inverse trajectory $\phi_{inv}(t_f,x_0,u,d)$ such that, if $\phi(t_f,x_0,u,d)$ is the solution of $\dot{x}=f(x,u,d)$, $\phi_{inv}(t_f,x_0,u,d)$ is the solution of $\dot{x}=-f(x,u_{inv},d_{inv})$, where $u_{inv}(t)=u(t_f-t)$ and $d_{inv}(t)=d(t_f-t)$. $\mathcal{T}^{c}(\tau)$ is the complementary set of $\mathcal{T}(\tau)$.

Theorem 5.2. *Consider the box of states $[x]\in\mathbb{IR}^n$, the sample control input $u\in\mathcal{U}$, and the sample disturbance input $d\in\mathcal{D}$. If $\mathcal{T}^{-}(t)$ and $\mathcal{T}^{+}(t)$ are such that $\mathcal{T}^{-}(t)\subseteq\mathcal{T}(t)\subseteq\mathcal{T}^{+}(t)$ then*

- *The interval function*

$$\mathcal{C}_{\mathcal{T}^{c}(t-\tau)}([x])=[x]\cap[\phi_{inv}]\left(\tau,\left[[\phi](\tau,[x],u,[\mathbb{D}])\cap\mathcal{T}^{-c}(t)\right],u,[\mathbb{D}]\right) \tag{26}$$

  is a contractor for the constraint $x\in\mathcal{T}^{c}(t-\tau)$.

- *The interval function*

$$\mathcal{C}_{\mathcal{T}(t-\tau)}([x])=[x]\cap[\phi_{inv}]\left(\tau,\left[[\phi](\tau,[x],[\mathbb{U}],d)\cap\mathcal{T}^{+}(t)\right],[\mathbb{U}],d\right) \tag{27}$$

  is a contractor for the constraint $x\in\mathcal{T}(t-\tau)$.

.

Proof. The contractance of $\mathcal{C}_{\mathcal{T}(t-\tau)}$ and $\mathcal{C}_{\mathcal{T}^c(t-\tau)}$ is proven by the fact that for any $[x], [y] \in \mathbb{IR}^n$, $[x] \cap [y] \subseteq [x]$.

$[[\phi]\,(\tau, [x], \mathbb{U}, d) \cap \mathcal{T}^+(t)] \supseteq \phi\,(\tau, [x], \mathbb{U}, d) \cap \mathcal{T}(t)$ is an enclosure of the states on $\mathcal{T}(t)$ the system can reach from the original box $[x]$ using as control some strategy $\gamma : \mathcal{D} \to \mathcal{U}$ and as disturbance input d. From definition 3.2 we have that $[[\phi]\,(\tau, [x], [\mathbb{U}], d) \cap \mathcal{T}^+(t)] \supseteq \phi\,(\tau, [x], \mathbb{U}, d) \cap \mathcal{T}(t) \supseteq \phi\,(\tau, [x] \cap \mathcal{T}(t-\tau), \mathbb{U}, d)$.

From the above relation it follows that $[\phi_{inv}]\,(\tau, [[\phi]\,(\tau, [x], [\mathbb{U}], d) \cap \mathcal{T}^+(t)]\,, [\mathbb{U}], d)$ contains $\phi_{inv}\,(\tau, [[\phi]\,(\tau, [x], [\mathbb{U}], d) \cap \mathcal{T}^+(t)]\,, \mathbb{U}, d)$ which contains $\phi_{inv}\,(\tau, \phi\,(\tau, [x] \cap \mathcal{T}(t-\tau), \mathbb{U}, d)\,, \mathbb{U}, d)$ which is in turn equal to $[x] \cap \mathcal{T}(t-\tau)$ and thus $\mathcal{C}_{\mathcal{T}(t-\tau)} \Leftarrow ([x]) \supseteq [x] \cap \mathcal{T}(t-\tau)$ proving the correctness of $\mathcal{C}_{\mathcal{T}(t-\tau)}$.

The proof of the correctness of $\mathcal{C}_{\mathcal{T}^c(t-\tau)}$ is similar and is omitted here. □

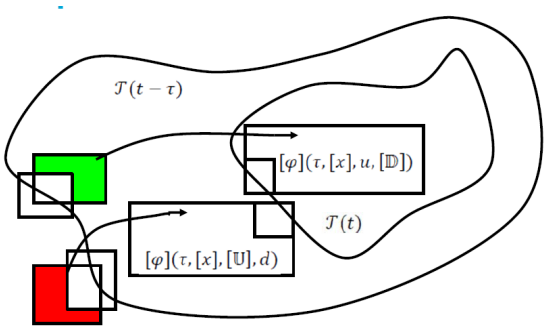

Figure 3: Illustration of Theorem 5.2

From Theorems 5.1 and 5.2, starting with a box $[x_0] \supset \mathcal{T}(t+\tau)$ and having a description of $\mathcal{T}^+(t)$ and $\mathcal{T}^-(t)$, we can obtain an inner or an outer bound for the reachable set $\mathcal{T}(t-\tau)$, through a branch and prune algorithm .

To obtain tighter enclosures, given the target set $\mathcal{T}_0$ we can compute inner and outer bounds for $\mathcal{T}(0)$ using a number of time steps N_{stp} and using $\mathcal{T}^-(t_f - i*t_f/Nstp)$ and $\mathcal{T}^+(t_f - i*t_f/N_{stp})$ as intermediate results, where i is an integer from 0 to $N_{stp} - 1$. Using intermediate time steps can help reduce the effects associated with the overestimation of $\phi(\tau, [x], \mathbb{U}, d)$, because this overestimation increases non-linearly with τ. On the other hand, with more time steps the accumulated error increases since at each time step we are overestimating the reachable set by, at most, the size of the smallest box on the subpaving, plus the highest overestimation of $\phi(\tau, [x], \mathbb{U}, d)$, and the error associated with Theorem 5.1 or 5.2.

6 Computational Application

6.1 Branch and Prune (B&P)

To implement the method described in the previous section, an algorithm to compute subpavings that enclose a given set was implemented and tested. The algorithm, named Branch and Prune, is described in Figure 13, in the appendix.

Figure 4 illustrates six iterations of the algorithm used to enclose a circle starting with a box $[x]_0 = ([-2,2],[-2,2])$ where the set $\mathcal{I}$ contains the sides of $[x]_0$.

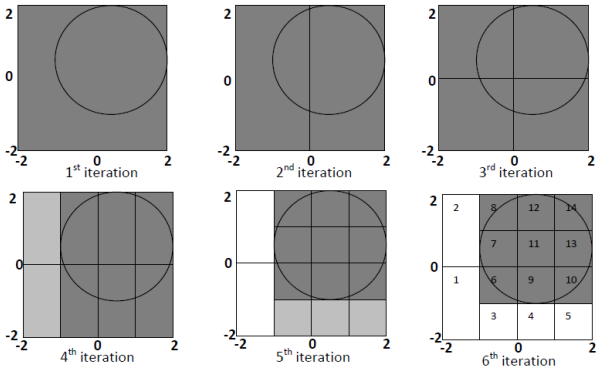

Figure 4: Result of the Branch and Prune algorithm at each iteration.

6.2 Branch and Contract (B&C)

In Figure 14 we describe an algorithm, based on the B&P algorithm described before, to compute subpavings using contractors. This algorithm will be named Branch and Contract (B&C).

For the B&C algorithm we need the function $exclude\left([x],[y]\right)$ that computes a subpaving of $cl\left([y]\setminus[x]\right)$, where $cl(\bullet)$ is the closure of a set. The function is defined as follows:

$$exclude\left([x],[y]\right) := \begin{cases} [y], & if\ [x]\cap[y]=\emptyset \\ \emptyset, & if\ [y]\subset[x] \\ \begin{array}{c}\{[z]\in\mathbb{IR}^n | \forall i\in\{1,\ldots,n\},\\ \left([z_i]=\left[\underline{y_i},\max\left(\underline{x_i},\underline{y_i}\right)\right]\right)\vee \\ \left([z_i]=\left[\max\left(\underline{x_i},\underline{y_i}\right),\min\left(\overline{x_i},\overline{y_i}\right)\right]\right)\vee \\ \left([z_i]=\left[\min\left(\overline{x_i},\overline{y_i}\right),\overline{y_i}\right]\right), \\ \neq[x]\cap[y], w([z])>0\}\end{array} & otherwise \end{cases} \tag{28}$$

In the Branch and Contract algorithm, we use a contractor $\mathcal{C}'_{\mathbb{T}}\left([x]\right)\subset[x]\cap\mathbb{T}$, which is the term $[x^N]$ of a sequence $[x^{k+1}]=\mathcal{C}_{\mathbb{T}}\left([x^k]\right)$, under some stopping

criteria. The stopping criteria considered included the definition of a constant sc, and the series stops at $N = sc$, or the series stops when $w\left([x^{N-1}]\right) - w\left([x^N]\right) < sc$. In the last case sc is a measure of the required precision.

Figure 5 illustrates six iterations of the B&C algorithm of the same problem of Figure 4.

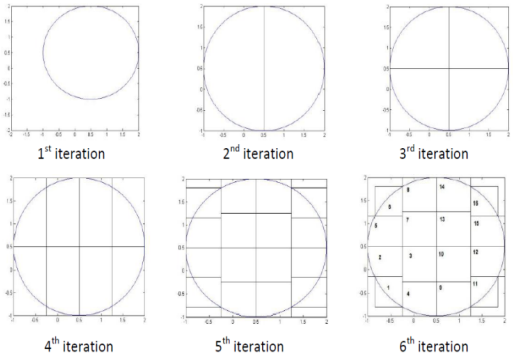

Figure 5: Result of the Branch and Contract algorithm at each iteration.

Figure 6 contains the eliminated boxes after the sixth iteration of Figure 5.

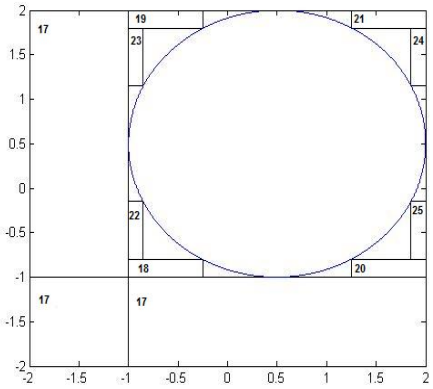

Figure 6: Excluded boxes at the 6^{th} iteration.

6.3 Applying the Euler Method

The computation of the guaranteed integration of the trajectories can be done using the Euler method [7]. This method computes an inclusion function of the trajectory $[\phi](t, [x], u)$ that determines the states reached at time t for initial conditions in the box $[x]$. The flow is evaluated in a box $[w]$, that includes

$[x]$, which is an expansion of $[v] = [[x](t_1) \cup [\hat{x}](t_2)]$ as is shown in Figure 7, where $[\hat{x}](t_2) = [x](t_1) + (t_2 - t_1)[f]([x](t_1), u)$. We then compute $[x](t_2) = [x](t_1) + (t_2 - t_1)[f]([w], u)$.

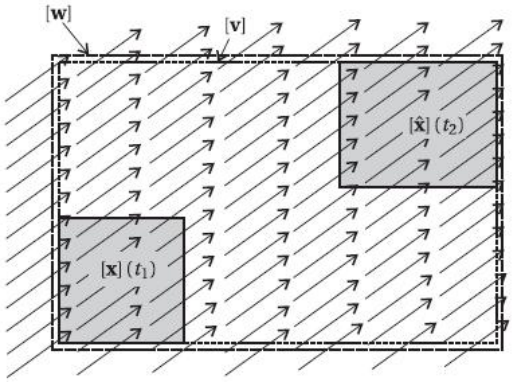

Figure 7: Interval Euler integration.

If $[x](t_2) \subset [w]$ the result is valid and $[\phi](t_2 - t_1, [x](t_1), u) = [x](t_2)$, otherwise it is not possible to reach a conclusion and we consider $[\phi](t_2 - t_1, [x](t_1), u) = \mathbb{R}^n$. The box $[w]$ is computed with the function $[w] = inflate([v], \alpha * w([v]) + \beta)$, where the function $inflate([x], \epsilon)$ is defined as:

$$inflate([x], \epsilon) := ([\inf([x_1]) - \epsilon, \sup([x_1]) + \epsilon], \ldots, [\inf([x_n]) - \epsilon, \sup([x_n]) + \epsilon]) \tag{29}$$

To reduce the overestimation it is preferable to compute the enclosure of the integration in $Nstp_{inteq}$ time steps. To this effect, naming the Euler method as $[\phi]_{step}$, the application of the Euler method to compute $[\phi](t, [x], \mathbb{U}, \mathbb{D})$ throughout the present work will be performed as Algorithm 6.3.

Algorithm 3 $[\phi]$(out: $[x]_f$ in: t_{end}, $[x]_0$, $\mathbb{U}$, $\mathbb{D}$, $N_{stp_{int}}$)

$t_{step} \leftarrow t_{end}/N_{stp_{int}}$; $[x] \leftarrow [x]_0$; $t \leftarrow 0$
while $t \neq t_{end}$ **do**
  $[x] \leftarrow [\phi]_{step}(t_{step}, [x], \mathbb{U}, \mathbb{D})$
  $t \leftarrow t + t_{step}$
end while
$[x]_f \leftarrow [x]$

6.4 Applying Interval Taylor Models

Besides boxes, a more flexible type of wrappers are Interval Taylor Models which consist of a polynomial, symbolic, part and an interval part.

To compute enclosures of solutions of initial value problems using Interval Taylor Models we used the VSPODE solver [8].

7 Application Example

7.1 Branch and Prune

The methods derived were tested for the double integrator system 30

$$\dot{x} = \begin{bmatrix} x_2 \\ u \end{bmatrix} = f(x, u), \tag{30}$$

where the control input is contained in $\mathbb{U} = [-1, 1]$. To obtain a general picture, the results of $cl\left(\mathcal{T}^{+c}(0) \cap [x]_0\right)$ and $cl\left(\mathcal{T}^{-}(0)\right)$, for $\mathcal{T}_0 = ([-0.2, 0.2], [-0.2, 0.2])$, for 10 time steps, a minimum box size of 0.005 and with $[x]_0 = ([-2, 2], [-2, 2])$ are displayed on Figure 8. The area of $\mathcal{T}^{+}(0)$ computed with the sum of the area $(Vol\left([x]\right))$ of all boxes of the subpaving describing $\mathcal{T}^{+}(0)$ is 2.4, and the area of $\mathcal{T}^{-c}(0) \cap [x]_0$ computed with the same method is 14.4.

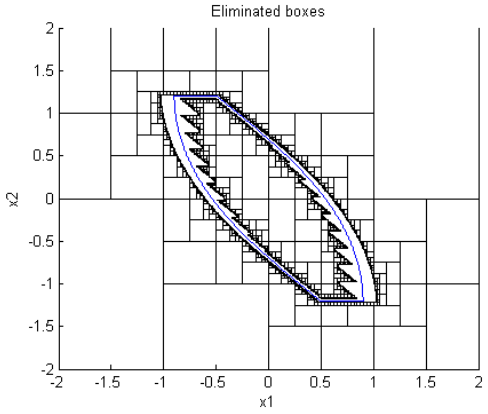

Figure 8: Inner and outer bound of reachable set for a double integrator system computed using the Branch and Prune algorithm with 10 time steps.

7.2 Branch and Contract

This section contains the results obtained when using the B&C algorithm. The results shown in this section are for $cl\left(\mathcal{T}^{+c}(0) \cap [x]_0\right)$ and $cl\left(\mathcal{T}^{-}(0)\right)$, with a target set $\mathcal{T}_0 = ([-0.2, 0.2], [-0.2, 0.2])$ and with a minimum box size of 0.005. The initial box, the root of the subpaving, is $[x]_0 = ([-2, 2], [-2, 2])$. The line in blue in the following figures is the analytical solution.

The computed set $\mathcal{T}^{+c}(0)$ obtained with one time step is displayed on Figure 9. The area of the computed set $\mathcal{T}^{+}(0)$ is 2.4.

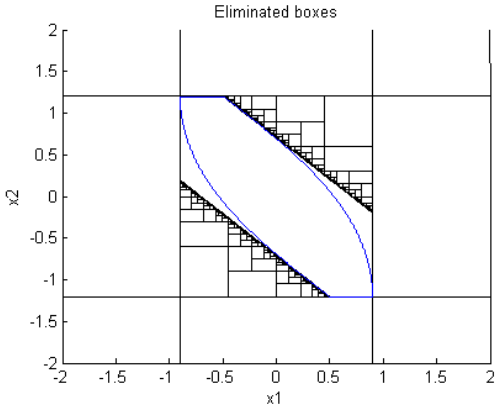

Figure 9: Outer bound of reachable set for a double integrator system computed using the Branch and Contract algorithm with 1 time step.

The sets $cl\left(\mathcal{T}^{+c}(0) \cap [x]_0\right)$ and $cl\left(\mathcal{T}^{-}(0)\right)$ computed with 10 time steps are in Figure 10. The area of $\mathcal{T}^{+}(0)$ computed with 10 time steps is 2.1 and the area of $\mathcal{T}^{-c}(0) \cap [x]_0$ is 14.2.

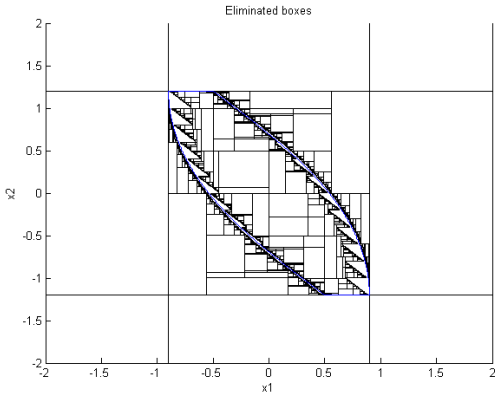

Figure 10: Inner and outer bound of reachable set for a double integrator system computed using the Branch and Contract algorithm with 10 time steps.

The results $cl\left(\mathcal{T}^{+c}(0) \cap [x]_0\right)$ and $cl\left(\mathcal{T}^{-}(0)\right)$ for 20 time steps are displayed in Figure 11. The area of $\mathcal{T}^{+}(0)$ is 2.2 and of $\mathcal{T}^{-c}(0) \cap [x]_0$ is 14.2.

7.3 VSPODE

The results of $cl\left(\mathcal{T}^{+c}(0) \cap [x]_0\right)$ and $cl\left(\mathcal{T}^{-}(0)\right)$ using VSPODE for state propagation, for 10 time steps with a minimum box size of 0.005 are displayed in

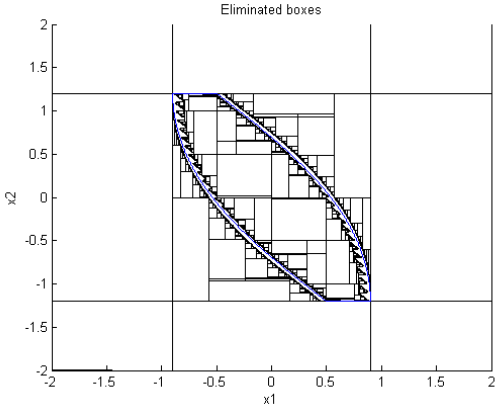

Figure 11: Inner and outer bound of reachable set for a double integrator system computed using the Branch and Contract algorithm with 20 time steps.

Figure 12. The target set is $\mathcal{T}_0 = ([-0.2, 0.2], [-0.2, 0.2])$ and the initial box is $[x]_0 = ([-2, 2], [-2, 2])$.

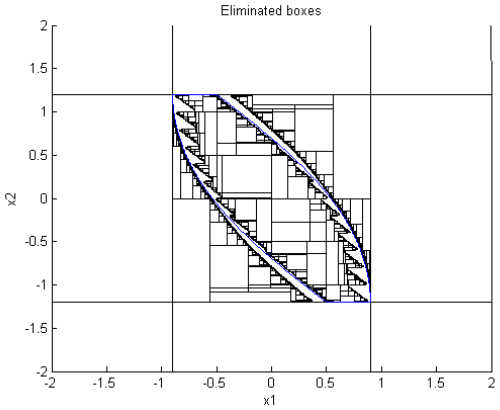

Figure 12: Inner and outer bound of reachable set for a double integrator system computed using the Branch and Contract algorithm with 10 time steps using the VSPODE integration program.

From the results of Figure 12 the computed area of $\mathcal{T}^+(0)$ is 2.2 and the area of $\mathcal{T}^{-c}(0) \cap [x]_0$ is 14.3.

8 Conclusions and Recommendations

8.1 Conclusions

It is concluded that the use of interval analysis, namely the construction of subpavings through set inversion and guaranteed integration, allows for the computation of compute inner and outer bounds of reachable sets defined by the union of non-overlapping boxes.

Two algorithms to construct subpavings (B&P and B&C) were implemented where only boxes in contact with already eliminated boxes or with a pre-defined set of boxes can be eliminated. The algorithm was tested and it was proven to work as expected; however, it suffers from the curse of dimensionality, the time to run grows exponentially with the dimension of the problem.

From the results obtained by using the method proposed to compute the reachable set of the double integrator system, some conclusions can be drawn. We observe that using a branch and contract algorithm yields tighter results than using a branch and prune algorithm. Increasing the number of time steps above a certain number may have a detrimental effect on the outer bound since at each time step we have a wrapping effect of the intermediate solution, but has a positive effect on the inner bound since we are considering more samples of the control input. The application of VSPODE was not successful, not yielding better results than Euler integration, which indicates that there is still work to be done either on how the program is applied or on choosing another integration program.

8.2 Recommendations

It is recommended that, in order to improve the effectiveness of the discussed methods, new guaranteed integration programs be devised and applied to the methods presented in this work, or that other available guaranteed integration programs be applied. Also, the application of VSPODE may be improved or possibly another program based on interval Taylor models may be applied, since they may have a great advantage when compared to other methods based only on interval vectors.

It was observed that programming on C++ instead of MATLAB improved the computational efficiency of the programs. Therefore, the applicability of the devised methods could improve significantly if all the components are programmed on C++.

Finally, instead of sampling the control input or the disturbance input, we can estimate the optimal control input or disturbance input as the optimal control input. The optimal control input is the input that maximizes the system function in the direction towards the reachable set at a particular state, and the optimal disturbance input is the input that minimizes the state function towards the reachable set.

9 Appendices

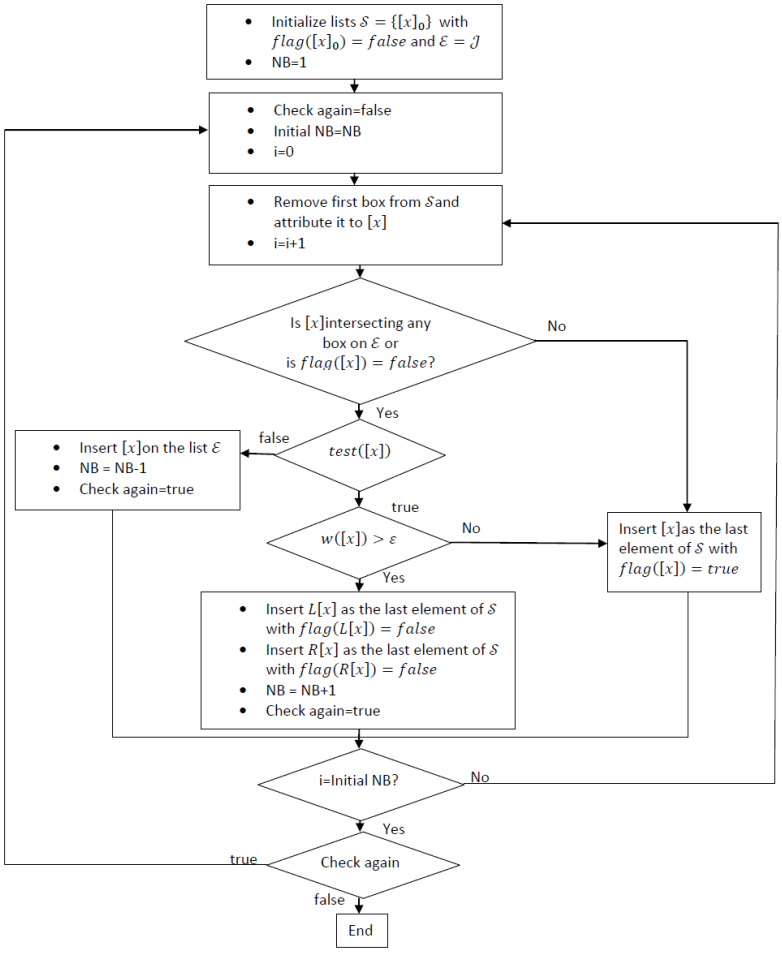

Figure 13: Flowchart of the Branch and Prune algorithm.

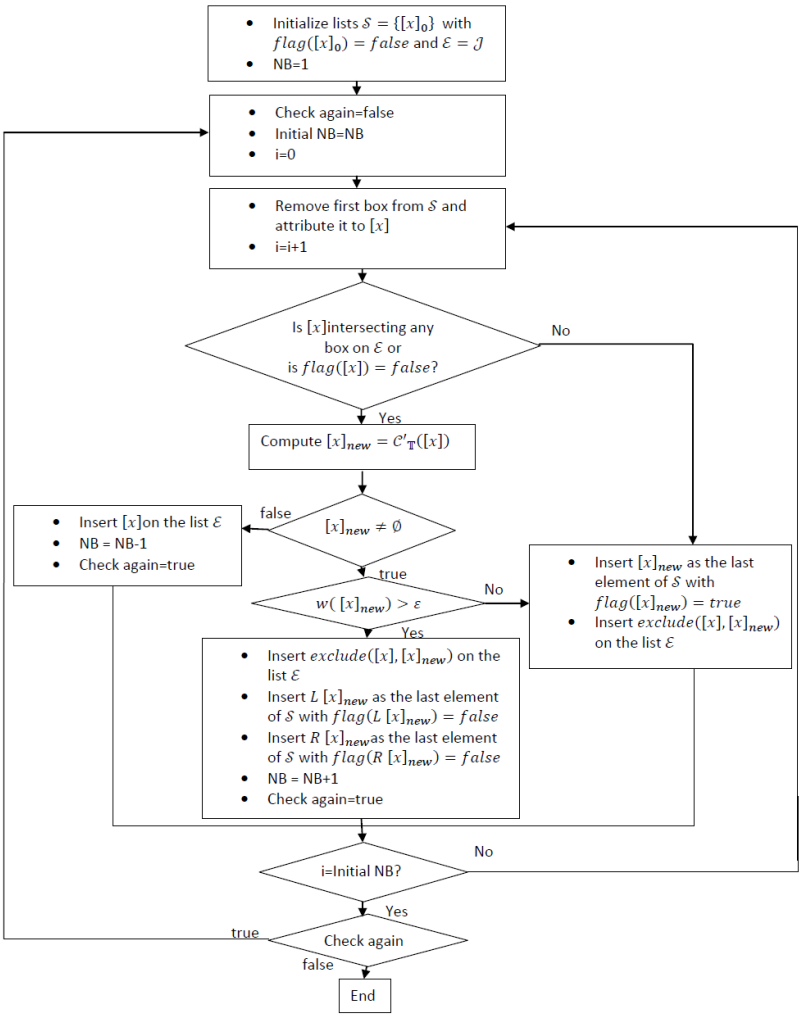

Figure 14: Flowchart of the Branch and Contract algorithm.

References

[1] de Weerdt, E., Chu, Q. & Mulder, J. (2009). *Global Fuel Optimization for Constrained Spacecraft Formation Rotations.* AIAA Guidance, Navigation, and Control Conference. Chicago.

[2] de Weerdt, E., van Kampen, E., Chu, Q. & Mulder, J. (2008). Integer Ambiguity Resolution Using Interval Analysis. ION Journal of Navigation , 293-307.

[3] Djeridane, B. & Lygeros, J. (2006). *Neural approximation of PDE solutions: An application to reachability computations.* 45th IEE Conference on Decision & Control, (pp. 3034-3039).

[4] Filipe, N., de Weerdt, E., van Kampen, E., Chu, Q. & Mulder, J. (2009). *Terminal Area Energy Management Trajectory Optimzation Using Interval Analysis.* AIAA Guidance, Navigation, and Control Conference. Chicago.

[5] Jaulin, L., Kiefer, M., Didrit, O. & Walter, E. (2001). *Applied Interval Analysis.* Springer.

[6] Knüppel, O. (1999). *PROFIL/BIAS V 2.0* report 99.1. TUHH.

[7] Lhommeau, M., Jaulin, L. & Hardouin, L. (2011). *Capture basin approximation using interval analysis.* International Journal of Adaptative Control and Signal Processing, 25, 264-272.

[8] Lin, Y. & Stadtherr, M. A. (2006). *Validated solution of ODEs with parametric uncertainties.* Computer Aided Chemical Engineering , 21, 167-172.

[9] Mitchell, I. (2002, August). *Application of level set methods to control and reachability problems in continuous and hybrid systems.* PhD dissertation, Stanford University.

[10] Moore, R. E., Kearfott, R. B. & Cloud, M. J. (2009). *Introduction to Interval Analysis.* SIAM.

[11] Neher, M., Jackson, K. R. & Nedialkov, N. S. (2007). *On Taylor Model Based Integration of ODEs.* SIAM Journal on Numerical Analysis, 45, 236-262.

[12] Packard, A., Topcu, U., Seiler, P. & Balas, G. (2009, June 9). *Quantitative Local Analysis of Nonlinear Systems Using Sum-of-Squares Decomposition, Main presentation.* St.Louis, Missouri, USA: 2009 American Control Conference.

[13] Rump, S. (1999). *INTLAB - INTerval LABoratory.* In T. Csendes, Developments in Reliable Computing (pp. 77-104). Dordrecht: Kluwer Academic Publishers.

[14] van Kampen, E., Chu, Q., Mulder, J. & van Emden, M. (2007). *Nonlinear Aircraft Trim Using Interval Analysis.* AIAA Guidance, Navigation, and Control Conference and Exhibit.

[15] van Kampen, E., de Weerdt, E., Chu, Q. & Mulder, J. (2009). *Aircraft Attitude Determination Using GPS and an Interval Integer Ambiguity Resolution Algorithm.* AIAA Guidance, Navigation, and Control Conference. Chicago.

[16] van Kampen, E., Zaal, P., de Weerdt, E., Chu, Q. & Mulder, J. (2010). *Optimization of Human Perception Modeling Using Interval Analysis.* Journal of Guidance, Control, and Dynamics , 33, 42-52.