Characterizing Sliding Surfaces of Cyber-Physical Systems

Luc Jaulin\textsuperscript{a} and Fabrice Le Bars\textsuperscript{b}

Abstract

When implementing a non-continuous controller for a cyber-physical system, it may happen that the evolution function of the closed-loop system is not anymore piecewise continuous along the trajectory, mainly due to \texttt{if} statements inside the control algorithm. As a consequence, an unwanted chattering effect may occur. This behavior is often difficult to observe even in simulation. We propose here a set-membership method based on interval analysis to detect different types of discontinuities. One of them is the sliding surface where the state trajectory jumps indefinitely between two distinct behaviors. As an application, we consider the validation of a sailboat controller. We show that our approach is able to detect and explain some unwanted sliding effects that may be observed in rare and specific situations on our actual sailboat robots.

Keywords: sliding surface, interval analysis, sailboats

1 Introduction

Validating properties of cyber-physical systems \cite{16,29} is a difficult problem for which set membership techniques provide original and efficient solutions \cite{25,26}.

Different types of set-membership approaches exist for the validation. Some require the integration of nonlinear differential equations \cite{19,28,30}. Others are based on positive invariance approaches \cite{1,18}. For the numerical resolution some methods grid the state space \cite{7,27} which makes them computationally expensive. Lyapunov-based methods \cite{24}, level-set methods \cite{20}, or barrier functions \cite{4} are attractive since they do not perform any integration through time. Now, these methods generally require a parametric expression for candidate Lyapunov-like functions \cite{23}.

This paper considers the validation of the controller of a sailboat robot which is an illustrative example of what is a cyber-physical system. Due to the control strategy used, the robot is an hybrid system \cite{22} since it includes a physical system

\textsuperscript{a}Lab-STICC, ENSTA Bretagne, Brest, France, E-mail: lucjaulin@gmail.com

\textsuperscript{b}Lab-STICC, ENSTA Bretagne, Brest, France, E-mail: fabrice.lebars@ensta-bretagne.fr

DOI: 10.14232/actacyb.24.3.2020.9
(the sailboat) and an algorithm (the controller inside the computer of the robot). More precisely, it is a controlled switching system [10] due to some discrete state variables in the controller. The controller is an algorithm containing if statements and the validation requires approaches coming from invariance approaches [3], static analysis [12] and abstract interpretation [6].

To detect the discontinuities and Zeno effects, we propose in this paper to generate a set of equalities in the state space where undesirable switching phenomena could occur. The corresponding zone (called later sliding surface) may be stable and the system can be trapped inside without any possibility to escape. Characterizing these sliding zones will be done by using interval techniques [17, 21]. This characterization can be used for the validation of the controller or to correct it by eliminating the unwanted sliding surfaces.

The paper is organized as follows. Section 2 introduces the easy-boat model which is a simple sailboat with a controller. This model will be used to illustrate our approach. Section 3 provides the formalism and gives a list of three problems we want to solve. Section 4 shows how our approach can be used to validate the controller but also to detect and explain some unwanted sliding effects that occur on actual sailboat controllers. Section 5 concludes the paper and provides some perspectives.

2 Easy boat model

The easy-boat model is described by

\[ \dot{d} = \sin u \]

under the constraint

\[ \cos(\psi - u) + \cos \frac{\pi}{5} > 0. \]

It is a simple version of a sailboat following a line [13], where \( \psi \) is the angle of the wind, \( d \) is the algebraic distance to the line and \( u \) is the heading of the boat. This is illustrated by Figure 1, where \( s \) is the curvilinear abscissa.

![Figure 1: Easy-boat following the red line](image)
We want that, after some transient period, the distance $d$ becomes small ($|d| \leq 2$ for instance). The controller we propose is the following, where $q \in \{-1, 1\}$.

**Controller** in: $(d, \psi, q)$; out: $u$

1. if $d^2 - 1 > 0$ then $q := \text{sign}(d)$
2. if $\cos(\psi + \arctan(d)) + \cos(\frac{\pi}{4}) \leq 0$ or $(d^2 - 1 \leq 0 \text{ and } \cos(\psi + \cos(\frac{\pi}{4}) \leq 0)$
3. then $u := \pi + \psi - q \frac{\pi}{4}$.
4. else $u := -\arctan(d)$.

Figure 2 provides some simulations with $q = 1$ at time $t = 0$. We took different initial conditions to avoid the superposition of the curves, taking into account the fact that the behavior of the system does not depend on these initial values for $d$. When $q$ switches between $-1$ to 1, the trajectories are not differentiable.

![Figure 2: Simulation of the easy-boat model $(t, d)$ with respect to different wind angles $\psi$](image)

**Remark.** For a link to the sailboat, it is more interpretable to draw $d$ with respect to the curvilinear abscissa $s = \int t \cos u$ as in Figure 3. The boat has to follow the horizontal line, $(s, d)$ corresponds to the position of the boat and $u$ is the heading. The arrows represent different directions for the winds. As we can see on the figure, the boat never goes upwind: there always exists an angle between the heading and the wind greater than $\zeta = \frac{\pi}{5}$ where $\zeta$ is the angle defining the no-go zone. For the simulation, we added the state variable $s$ which satisfies $\dot{s} = \cos u$.
This section provides an abstraction of our sailboat robot in order to give useful definitions, theorems and proofs. The corresponding formalism will be applied in the next section on the sailboat validation problems.

**Definition.** Given \( Q^-, Q^+ \) two disjoint closed subsets of \( \mathbb{R}^n \), two smooth functions \( f_a, f_b : \mathbb{R}^n \times \{-1, 1\} \to \mathbb{R}^n \), we define the dynamical system

\[
S(A) : \begin{cases} 
\dot{x} &= f(x, q) \\
q &= -1 \\
&= +1 
\end{cases} 
\begin{array}{l}
\text{if } x \in A \\
as soon as } x \in Q^- \\
as soon as } x \in Q^+
\end{array}
\]

We assume that

- \( f_a, f_b \) are continuous and differentiable,
- \( A \) is a closed subset of \( \mathbb{R}^n \) that can be defined by inequalities linked by Boolean operators.

This definition is illustrated by the automaton of Figure 4 taking the conventions used for hybrid systems [2, 9]. The red arrows show transitions which may not be stable and which may generate the sliding phenomena that are studied in this paper.

This definition trivially extends to situations where we have more than two guard sets \( Q^-, Q^+ \) and more than two fields \( f_a, f_b \). An hybrid system which can be translated into the form (3) is said to be *expandable*.

**Remark.** In this paper, to avoid atypical situations, the closed sets are assumed to be topologically stable, i.e., they have the same boundary as their interior. For instance, a disk of \( \mathbb{R}^2 \) is topologically stable, but not the circle since its interior is empty. We will also assume that the closed sets can be defined as a finite composition (with unions and intersections) of sets of the form \( X = \{x \in \mathbb{R}^n | c(x) \leq 0\} \) where \( c \) is a smooth function.
Figure 4: Automaton representing our Cyber Physical System

Since $A$ is closed, the set $B$ is open and the boundaries $\partial A, \partial B$ of $A, B$ satisfy

$$\partial A = \partial B = A \cap \text{Clo}(B)$$  \hspace{1cm} (4)$$

where Clo$(B)$ denotes the smallest closed set which encloses $B$. This common boundary can be defined by an equality. Moreover the pair $(x, q)$ always satisfies the constraint

$$x \in Q^+ \Rightarrow q = 1$$
$$x \in Q^- \Rightarrow q = -1$$  \hspace{1cm} (5)$$

This formula can be denoted equivalently by $x \in Q^{-q}$, with the notation $Q^{-1} = Q^-$ and $Q^1 = Q^+$. The corresponding behavior is represented on Figure 5, where the blue arrows correspond to $f(x, -1)$ and the pink arrows to $f(x, 1)$.

In this paper, we consider three problems:

- the constraint satisfaction problem which checks that a given variable of the algorithm defining $f$ is inside a feasible domain.
- the positive invariance for a set defined by inequalities
- the characterization of the sliding surface.

3.1 Constraint satisfaction

We want to show the state of the the cyber-physical system never reaches a forbidden domain. This can often be expressed as showing that we never have

$$h(x, q) \leq 0,$$  \hspace{1cm} (6)$$

with

$$\begin{cases} h(x, q) = h_a(x, q) & \text{if } x \in A \\ h(x, q) = h_b(x, q) & \text{if } x \in B \end{cases}$$  \hspace{1cm} (7)$$

where $h_a, h_b$ are continuous.
Proposition 1. If the set

\[ H = \bigcup_{q \in \{-1, 1\}} \left( \{x | h_a(x, q) \leq 0\} \cap A \cap \overline{Q^q} \right) \cup \left( \{x | h_b(x, q) \leq 0\} \cap B \cap \overline{Q^q} \right) \]  

is empty then we cannot have \( h(x, q) \leq 0 \).

**Proof.** The proof is by contradiction. More precisely, we take \((x, q)\) such that \( h(x, q) \leq 0 \) and we show that \( x \in H \). Since \( B = A \), we should consider two cases \( x \in A \) and \( x \in B \).

Case 1: \( x \in A \). From Equation (7), \( h(x, q) = h_a(x, q) \) and thus

\[ x \in \{x | h_a(x, q) \leq 0\} \cap A. \]

Case 2: \( x \in B \). From Equation (7), \( h(x, q) = h_b(x, q) \) and thus

\[ x \in \{x | h_b(x, q) \leq 0\} \cap B. \]

Since from Equation (5), we always have \( x \in \overline{Q^q} \), in both cases, \( x \in H \). This is inconsistent with the fact that \( H = \emptyset \).

3.2 Capture set

Consider a function \( V : \mathbb{R}^n \to \mathbb{R} \). The set \( C = \{x | V(x) \leq 0\} \) is called a capture set (or a positive invariant set) if all trajectories \( x(t) \) that enter inside \( C \) stay inside forever. To check that \( C \) is a capture set, we recall the notion of Lie derivative of \( V \) with respect to the field \( f : \mathbb{R}^n \to \mathbb{R}^n \) as

\[ \mathcal{L}_f^V(x) = \frac{dV}{dx}(x) \cdot f(x). \]
We also define the Lie set as

$$L^V f = \{ x | L^V f (x) \leq 0 \}. \quad (10)$$

In our context, the field depends on $i \in \{a, b\}$ and $q$. We will write

$$L^V_i (x, q) = L^V_{i(q)} (x)$$

$$L^V_i (q) = L^V_{i(q)}$$

**Proposition 2.** Define the set

$$V = \bigcup_{q \in \{-1, 1\}} \left( \overline{L^V_a (q) \cap A \cap \mathbb{Q}^{-q}} \cup \left( \overline{L^V_b (q) \cap B \cap \mathbb{Q}^{-q}} \right) \right). \quad (12)$$

If $V \cap \mathcal{C} = \emptyset$ then $\mathcal{C}$ is a capture set.

**Proof.** The proof is by contradiction. Assume that $\mathcal{C}$ is not a capture set. There exists a trajectory leaving $V$ at a point $x$. Assume first that $x \in A$. Then, $L^V_a (x, q) \geq 0$ or equivalently, $x \in \overline{L^V_a (q)}$. Taking into account that from (5), $x \in \mathbb{Q}^{-q}$, we get that $x \in \overline{L^V_a (q)} \cap A \cap \mathbb{Q}^{-q}$. If now we assume that $x \in B$, we get $x \in \overline{L^V_b (q)} \cap B \cap \mathbb{Q}^{-q}$. $\blacksquare$

### 3.3 Sliding surface

The *sliding surface* $S (\mathcal{A})$ [8] for $S (\mathcal{A})$ (see Equation (3)) is defined as the largest subset of the boundary $\partial \mathcal{A}$ between $\mathcal{A}$ and $\mathcal{B} = \overline{\mathcal{A}}$ such that the system can stay inside for a non degenerated interval of time.

If $\mathcal{A}$ is defined by the inequality $c(x) \leq 0$, then $\mathcal{B}$ is defined by $c(x) > 0$ and the boundary by $c(x) = 0$. The sliding surface is

$$S (\mathcal{A}) = \partial \mathcal{A} \cap \{ x | \exists q, x \in \mathbb{Q}^{-q}, L^V_a (x, q) \geq 0 \land L^V_b (x, q) \leq 0 \}. \quad (13)$$

Figure 6 illustrates the principle of this proposition in the case where $\mathcal{A}$ is described by one inequality $c(x) \leq 0$ and with no discrete variable $q$. In this case

$$S (\mathcal{A}) = \partial \mathcal{A} \cap \{ x | L^V_a (x) \geq 0 \land L^V_b (x) \leq 0 \}. \quad (14)$$

The boundary $\partial \mathcal{A}$ of $\mathcal{A}$ is composed of four parts:

- $\partial \mathcal{A} \cap \overline{L^V_a (q)} \cap \overline{L^V_b (q)} \rightarrow$ magenta
- $\partial \mathcal{A} \cap \overline{L^V_a (q)} \cap \overline{L^V_b (q)} \rightarrow$ red
- $\partial \mathcal{A} \cap \overline{L^V_a (q)} \cap \overline{L^V_b (q)} \rightarrow$ yellow
- $\partial \mathcal{A} \cap \overline{L^V_a (q)} \cap \overline{L^V_b (q)} \rightarrow$ black

One trajectory (dotted line) $x(t)$ is also represented. Before the yellow arc, $c(x)$ is positive and decreases. When it crosses the yellow arc, $c(x) = 0$ for some isolated
time point $t_1$. Then $x(t)$ remains inside $A$ until it reaches the red arc. It slides in the red arc for some non-degenerated time interval. When $x(t)$ reaches the magenta arc, it leaves $A$.

**Proposition 3.** Consider two closed sets $A_1$ and $A_2$. As illustrated by Figure 7, we have

$$
\begin{align*}
(i) \quad S(A_1 \cap A_2) & = (S(A_1) \cap A_2) \cup (S(A_2) \cap A_1) \\
(ii) \quad S(A_1 \cup A_2) & = (S(A_1) \cap \text{clo}\overline{A_2}) \cup (S(A_2) \cap \text{clo}\overline{A_1})
\end{align*}
$$

**Proof.** Let us first prove (i). If $x \in S(A_1 \cap A_2)$, then $x$ belongs to the boundary $\partial(A_1 \cap A_2)$ of $A_1 \cap A_2$. Now, since $A_1, A_2$ are both closed, we have $\partial(A_1 \cap A_2) = (\partial A_1 \cap A_2) \cup (\partial A_2 \cap A_1)$. Thus, we have to consider two cases: (a) $x \in \partial A_1 \cap A_2$ and the system slides on $\partial A_1$ (i.e., $x \in S(A_1)$) or (b) $x \in \partial A_2 \cap A_1$ and the system slides on $\partial A_2$ (i.e., $x \in S(A_2)$). Considering the two cases, we get

$$
S(A_1 \cap A_2) = (\partial A_1 \cap A_2 \cap S(A_1)) \cup (\partial A_2 \cap A_1 \cap S(A_2))
$$

Let us now prove (ii). If $x \in S(A_1 \cup A_2)$, then $x$ belongs to the boundary $\partial(A_1 \cup A_2)$ of $A_1 \cup A_2$. Now, $\partial(A_1 \cup A_2) = (\partial A_1 \cap \text{clo}\overline{A_2}) \cup (\partial A_2 \cap \text{clo}\overline{A_1})$. Again, we have to consider two cases: (a) $x \in (\partial A_1 \cap \text{clo}\overline{A_2})$ and then $x \in S(A_1) \cap \text{clo}\overline{A_2}$ and (b) $x \in (\partial A_2 \cap \text{clo}\overline{A_1})$ then $x \in S(A_2) \cap \text{clo}\overline{A_1}$. $\blacksquare$

Proposition 3 can be used to compute the sliding surface of a set $A$ as soon as $A$ can be defined by inequalities connected by Boolean operators such as and, or, not. The proposition is illustrated by Figure 8 in the case where $A = A_1 \cup (A_2 \cap A_3)$ and $A_i = \{x|c_i(x) \leq 0\}$. The trajectory (green) slides twice, first on $\partial A_1$, then it slides on $\partial A_2$. The sliding surfaces are painted red.
Figure 7: Illustration of Proposition 3, the sliding surfaces are painted red

Figure 8: Sliding surfaces for $\mathbb{A} = \mathbb{A}_1 \cup (\mathbb{A}_2 \cap \mathbb{A}_3)$
4 Application to our easy-boat model

Taking into account the dynamic in (1), the controller given in Section 2, and setting \( x = (d, \psi) \), we obtain the following evolution function for the closed loop easy-boat model:

\[
\text{Function } f(x, q) = \begin{cases} 
\sin (\pi + x_2 - q \frac{\pi}{4}) & \text{if } \cos (x_2 + \tan x_1) + \cos \frac{\pi}{4} \leq 0 \\
\sin(-\tan x_1) & \text{otherwise}
\end{cases}
\]

Therefore, our easy-boat model can be described by the expandable form (3) by taking the following correspondences:

\[
x = (d, \psi) \\
f_a(x, q) = \left( \sin (\pi + x_2 - q \frac{\pi}{4}) \right) \\
f_b(x) = \left( \sin(-\tan x_1) \right) \\
A_1 = \{ x | \cos (x_2 + \tan x_1) + \cos \frac{\pi}{4} \leq 0 \} \\
A_2 = \{ x | x_1^2 - 1 \leq 0 \} \\
A_3 = \{ x | \cos x_2 + \cos \frac{\pi}{4} \leq 0 \} \\
A = A_1 \cup (A_2 \cap A_3) \\
Q^- = \{ x | x_1 + 1 \leq 0 \} \\
Q^+ = \{ x | 1 - x_1 \leq 0 \}
\]

We can now illustrate the resolution of the three problems treated at Section 3.

4.1 Constraint satisfaction

Using Proposition 1, we want to prove that the easyboat never goes upwind (see Equation (2)), i.e., we never have

\[
\cos (x_2 - u) + \cos \frac{\pi}{5} \leq 0
\]

where \( u \) is given by the controller (see Section 2)

\[
\begin{cases} 
u = \pi + x_2 - q \frac{\pi}{4} & \text{if } x \in A \\u = -\tan x_1 & \text{otherwise}
\end{cases}
\]

Thus, the no-go zone constraint can be expressed as

\[
h(x) = \cos (x_2 - u) + \cos \frac{\pi}{5} \leq 0
\]
with
\[
\begin{cases}
    h(x) = h_a(x) &= \cos\left(-\pi - q\frac{\pi}{4}\right) + \cos\frac{\pi}{5} & \text{if } x \in A \\
    &= \cos\frac{3\pi}{4} + \cos\frac{\pi}{5} \\
    h_b(x) &= \cos(x_2 + \text{atan} x_1) + \cos\frac{\pi}{5} & \text{otherwise}
\end{cases}
\] (21)

As required by (8), we compute the set
\[
H = \left(H_a(1) \cap \overline{Q} \cap \overline{A}\right) \cup \left(H_a(-1) \cap \overline{Q}^T \cap \overline{A}\right) \cup (H_b \cap \overline{B})
\] (22)

where
\[
H_a(q) = \{x|h_a(x, q) \leq 0\} \\
H_b = \{x|h_b(x) \leq 0\}
\] (23)

Using the interval based solver PyIbex\(^1\) we easily show that this set has no solution. From Proposition 1, we conclude that the forbidden constraint \(\cos(x_2 - u) + \cos\frac{\pi}{5} \leq 0\) is never reached.

### 4.2 Capture set

To show that the easyboat stays inside a corridor of radius 2, we take \(V(x) = x_1^2 - 4\). We have
\[
\begin{align*}
    \mathcal{L}_a^V(x, q) &= \frac{dV}{dx}(x) \cdot \bar{f}_a(x, q) = 2x_1 \cdot \sin\left(q\frac{\pi}{4} - x_2\right) \\
    \mathcal{L}_b^V(x) &= \frac{dV}{dx}(x) \cdot \bar{f}_b(x, q) = \frac{-2x_1}{\sqrt{x_1^2 + 1}}
\end{align*}
\] (24)

We compute the set
\[
V = \left(L_a^V(1) \cap \overline{Q} \cap \overline{A}\right) \cup \left(L_a^V(-1) \cap \overline{Q}^T \cap \overline{A}\right) \cup (L_b^V \cap \overline{B})
\] (25)

where
\[
L_a^V(q) = \{x|L_a^V(x, q) \leq 0\} \\
L_b^V = \{x|L_b^V(x) \leq 0\}
\] (26)

Since we need to compute with sets defined by non-linear inequalities that are connected with intersection, union, complementary operators, we decided to use separators [15] instead of contractors [5] (which do not allow the use of complementary operators).

We prove that set \(V \cap \overline{C}\) is empty using PyIbex. From Proposition 2, we conclude that \(C\) is a capture set.

\(^1\)http://benensta.github.io/pyIbex/
Figure 9 gives a superposition of the fields for $f_a(x, 1)$ (blue), $f_a(x, -1)$ (black) and $f_b(x)$ (red). Is also represented the capture set $C$ (green) and the set $V$ (red) which may not respect the constraint as soon as it is inside $C$. Since the wind is constant, the arrows are horizontal. Since we have $x \in Q^- \Rightarrow q = -1$, the blue arrow going left in the blue circle cannot be reached by a trajectory. From the figure, we can see that outside $C$, all fields are oriented toward the line $d = 0$ which is consistent with the results obtained in [14].

4.3 Sliding surface

Assume that for all $i$, $A_i$ is defined by the inequality $c_i(x) \leq 0$, $B$ by $c_i(x) > 0$ and the boundary $\partial A_i$ by $c_i(x) = 0$. From (13), the sliding surface for $A_i$ is

$$S(A_i) = \partial A_i \cap \bigcup_{q \in \{-1, 1\}} Q^{-q} \cap L_q^a(q) \cap L_q^b = \partial A_i \cap L_q^b \cap \left( L_q^a(1) \cap Q^- \cup L_q^a(-1) \cap Q^+ \right)$$

(27)

where

$$L_q^a(q) = \{x|L^a_{c_i}(x, q) \leq 0\}$$

$$L_q^b = \{x|L^b_{c_i}(x) \leq 0\}$$

(28)
Now, we have

$$\begin{align*}
L_{ca}(x, q) &= \frac{d}{dx}(x) \cdot f_a(x, q) = -\frac{\sin(\frac{q\pi}{4} - x_2 \cdot x_1)}{x_1^2 + 1} \\
L_{cb}(x) &= \frac{d}{dx}(x) \cdot f_b(x, q) = \frac{\sin(\frac{q\pi}{4} - x_2 \cdot x_1)}{\sqrt{x_1^2 + 1}} \\
L_{ca}(x, q) &= \frac{d}{dx}(x) \cdot f_a(x) = 2 \sin(\frac{q\pi}{4} - x_2 \cdot x_1) \\
L_{cb}(x) &= \frac{d}{dx}(x) \cdot f_b(x, q) = -\frac{2x_2}{\sqrt{x_1^2 + 1}} \\
L_{ca}(x, q) &= \frac{d}{dx}(x) \cdot f_a(x, q) = 0 \\
L_{cb}(x) &= \frac{d}{dx}(x) \cdot f_b(x, q) = 0
\end{align*}$$

(29)

The abstract syntax tree associated to the expression of the sliding surface $S$ is depicted on Figure 10. It can be generated automatically using the rules provided by Proposition 3. The complexity of the tree illustrates the advantage of using separator algebra for the characterization of the solution set.

The abstract syntax tree associated to the expression of the sliding surface $S$ is depicted on Figure 10. It can be generated automatically using the rules provided by Proposition 3. The complexity of the tree illustrates the advantage of using separator algebra for the characterization of the solution set.

We obtain Figure 11 where two horizontal segments appear. They correspond to a wind angle corresponding to $\pm \frac{3\pi}{4}$ as expected.

To have a deeper understanding, let us draw the trajectories associated to the simulations of Figure 2 (see also Figure 12). The red set, obtained with PyIbex, corresponds to $K = A_1 \cup (A_2 \cap A_3)$. We can see that most of trajectories cross the singularities at one time $t$. But the red stays on the sliding surface for time period that maybe long. Thus make the sailboat loosing a lot of time due to many unneeded maneuvers. The controller alternates indefinitely between two strategies:

$$\bar{\theta} := \bar{\varphi} \text{ and } \bar{\theta} := \pi + \psi - q\zeta.$$

Recall that this hesitation can be seen on simulations but also sometimes for short periods during real experiments with our actual sailboat.
Figure 10: Abstract syntax tree associated to the expression of $S$

Figure 11: Sliding surface (yellow)
5 Conclusion

In this paper, we have presented a new approach based on contractor/separator programming to compute the sliding surfaces of a cyber-physical system. If the state of the system is on this surface, it may hesitate indefinitely between two different strategies. As a result, the system may be trapped on this surface and the designed mission may fail. It is thus important to detect and compute the sliding surface in order to eliminate them by changing the controller.

Further researches we would like to address in the future are the following.

- Generalize the method to situations where we have more than two continuous evolution functions \( f_i, i \in \{a, b, \ldots\} \) and where \( q \) may take more than two values.
- Take into account quantifiers to consider different kinds of uncertainties [11].
- Build a tool able to cast automatically a physical system with a controller described by an algorithm with if-statements into the expandable form (3). This could be done, for instance, by obtaining a disjunctive normal form (BNF) of the controller. Or equivalently to replace all if-then-else in the controller by a single switch-case statement.
- Find a new controller for our sailboat, as efficient as the existing one, but without any sliding surface.
References


