Chattering-Free Simulation for Hybrid Dynamical Systems
Semantics and Prototype Implementation

Ayman Aljarbouh
Hycomes Team, Département D4
Centre de Recherche INRIA
35042 Rennes Cedex, France
Email: ayman.aljarbouh@inria.fr

Yingfu Zeng
Department of Computer Science
Rice University
Houston, Texas 77005-1827
Email: yingfuzeng@gmail.com

Adam Duracz
Effective Modeling Group (EMG)
Halmstad University
Halmstad, Sweden
Email: adam.duracz@hh.se

Benoit Caillaud
Hycomes Team, Département D4
Centre de Recherche INRIA
35042 Rennes Cedex, France
Email: benoit.caillaud@inria.fr

Walid Taha
Effective Modeling Group (EMG)
Halmstad University
Halmstad, Sweden
Email: walid.taha@hh.se

***Abstract*—Chattering is a fundamental phenomenon that is unique to hybrid systems, due to the complex interaction between discrete dynamics (in the form of discrete transitions) and continuous dynamics (in the form of time). In practice, simulating chattering hybrid systems is challenging in that simulation effectively halts near the chattering time point, as an infinite number of discrete transitions would need to be simulated. In this paper, formal conditions are provided for when the simulated models of hybrid systems display chattering behavior, and methods are proposed for avoiding chattering "on the fly" in runtime. We utilize dynamical behavior analysis to derive conditions for detecting chattering without enumeration of modes. We also present a new iterative algorithm to allow for solutions to be carried past the chattering point, and we show by a prototypical implementation how to generate the equivalent chattering-free dynamics internally by the simulator in the main simulation loop. The concepts are illustrated with examples throughout the paper.**

***Keywords*-Hybrid systems; Numerical simulations; Chattering execution; Modeling and simulation; Model verification and completeness.**

I. Introduction

Hybrid systems are dynamical systems that incorporate both continuous and discrete dynamics. As such, they describe a large variety of applications and physical systems [2] [14]. In such systems, the continuous behavior is modeled by time-driven continuous variable dynamics (i.e. ODEs, DAEs) while the discontinuous behavior is modeled by event-driven discrete logic dynamics (i.e. `if-then-else`, `when-then-else`). The discrete dynamics could include instantaneous jumps in a continuous state variable, as well as switches to completely different dynamical modes. The rich structure of hybrid systems allows them to accurately predict the behavior of quite complex systems [1]. However, special attention must be devoted to hybrid system models, because chattering hybrid system models may arise from the modelling over-abstraction employed by control engineers in an attempt to derive models that are simpler to analyze and control. Chattering is a fundamental phenomenon that is unique to hybrid systems, due to the complex interaction between discrete dynamics (in the form of discrete transitions) and continuous dynamics (in the form of time). This is the case even if this is a result of the abstractions that yield hybrid models[4]. While chattering behavior can be attributed to such insufficient modeling of the complex dynamics of a system, it is present even in simple hybrid systems. Formally, an execution (or solution) of a hybrid system is called chattering if it undergoes an infinite number of discrete transitions in a finite amount of time. This happens when nearly-equal thresholds for the transition conditions of different modes are satisfied and the system starts to oscillate around them. The limit of the set of switching time points for a chattering execution is called the 'chattering time' and the points to which chattering solutions converge are called 'chattering equilibria'. The latter are fixed points of the discrete dynamics, that is, chattering behavior causes the system to infinitely move back and forth between modes in a discrete fashion.

A. Problem Statement

The numerical simulation of hybrid systems exhibiting chattering behavior is quite challenging as, in general, small step-sizes are required to maintain the simulation's numerical precision. From a practical perspective, an essential element of the numerical simulation of a hybrid dynamical system is the generation of discrete events from continuous variables that exceed thresholds. A zero-crossing function $\gamma(t,x)$ is used to identify the boundary at which the change takes place. The nature of zero-crossing detection and location is to compare the sign of the function value $\gamma(t, x)$ at the beginning and the end of each time integration step and, if it changes, declare a state event, and then bracket the interval of the time integration step (i.e. bisectional search) to locate the zero-crossing. During each iteration of the zero-crossing location, the zero-crossing function is evaluated twice: at the left and the right side of the reducing interval. After the event is bracketed by T_{left} and T_{right}, the solver first advances integration time from t_i to T_{left}. The solver is then reset before advancing to T_{right} and switching the mode. However, this approach may fail if the system exhibits a chattering execution. The zero crossing function $\gamma(t, x)$ is a function of the model state, but it does not contribute to its continuous dynamics $f(t, x)$. Therefore, the numerical integration can proceed without taking the dynamics of $\gamma(t, x)$ into account and, as these are faster than the dynamics $f(t, x)$, the chattering execution then causes the previous T_{right} to become the T_{left} of the next time step, and the integration will move with the minimum step-size allowed. For both non-adaptive and adaptive time stepping with event localization, the root finding to locate the exact time of occurrence of the chattering event breaks down the numerical integration, as the system converges fast to the point in time at which infinitely many discrete transitions need to be simulated. Since each discrete transition takes a non-zero and finite computation time, the simulation inevitably halts near the chattering time point. When the system is integrated without event localization, the significant frequency of the solution trajectory's oscillation around the switching surface -because of chattering- causes continuous integration to become dramatically and excessively slow. High computational costs are then required to terminate the simulation in a reasonable time.

B. Relationship with Previous Results

The problem of chattering behavior in hybrid systems has been investigated by means of different methods in control, especially in sliding mode control. A smooth sliding motion can be induced on the switching manifold on which the chattering occurs [3] [10] [8] [7] [5]. Filippov's Differential Inclusion approach [12] can be used in this case to define equivalent sliding dynamics on the switching manifold in question. The so-called 'equivalent control' approach proposed by Utkin [11] can also be used. However, the computation of the equivalent dynamics or equivalent control turns out to be difficult and highly challenging whenever the system chatters between more than two dynamics. This case of chattering arises naturally when the chattering behavior occurs in dynamical systems having multiple discontinuous control variables. Some researchers have proposed dealing with chattering in simulation by adding a small hysteresis to the event indicators to avoid chattering [9]. This approach has the following disadvantages: I) The size of the small value ϵ of the hysteresis is directly related to the event function $\gamma(t, x)$. That is, in order to determine the correct size of ϵ in the simulation environment, the nominal value of $\gamma(t, x)$ has to be reported by the modeler, which requires a manual manipulation by the user during the entire simulation process. II) Adding hysteresis to the event indicators does not guarantee an efficient treatment of the chattering behavior, as the physics in chattering hybrid systems cause the solution $x_\epsilon(\cdot)$ be a saw-toothed, or zigzag function, i.e., a function that oscillates around the switching surface, with peaks at $-\epsilon < 0$ and $+\epsilon > 0$, with $t_{i+1} - t_i = 2\epsilon$.

C. Summary of Contributions

In this paper, we present a novel iterative computational framework for addressing chattering behavior of hybrid dynamical systems in run-time without modes enumeration, without the need to add a small hysteresis to the event functions, and without having to solve stiff nonlinear equations for the computation of the chattering-free coefficients, in the case of chattering on switching intersection. A prototypical implementation of our proposed chattering-free semantics was developed, tested, and validated in the Acumen simulation tool. We provide, via this implementation, guidance for the development of a chattering-free version for hybrid systems simulation tools, providing a computational framework for an ideal manipulation of chattering behavior.

The outline of the paper is as follows: in Section II, we provide definitions of hybrid systems, their executions, and chattering executions. Two chattering examples and their simulations are also provided in this section. In Section III we present our chattering-free semantics for an ideal treatment of chattering. A prototypical implementation of our proposed chattering-free semantics is sketched in Section IV. Finally, the simulation results and conclusions of the work are given in Section V and Section VI respectively.

II. Preliminaries and Overview

In this section we provide a brief introduction to hybrid systems, their executions, the chattering execution, and the simulation of two chattering examples.

Definition 1: Hybrid Dynamical Systems
We consider a hybrid system $\mathcal{H}$ described as a tuple

$$\mathcal{H} = (Q, D, E, G, R, F) \tag{1}$$

where

- $Q = \{1, ..., M\} \subset \mathbb{N}$: a finite set of (control) locations that represent *control modes* of the hybrid system,
- $D = \{D_q\}_{q \in Q}$: a set of *invariants*, where D_q, a compact subset of $\mathbb{R}^n$, is a predicate that constrains the possible valuations for the continuous state variables x when the control of the hybrid system is in mode $q \in Q$,

- $E \subset Q \times Q$: a set of *discrete transitions*, which define the connection beween control modes by identifying the pairs (q, q'), where for each $e = (q, q') \in E$ we denote its source $s(e) = q$ and its target $t(e) = q'$,
- $G = \{G_e\}_{e \in E}$: a set of *guards*, where $G_e \subseteq D_{s(e)}$,
- $R = \{\phi_e\}_{e \in E}$: a set of *reset maps*, where for each $e = (q, q') \in E$, $R_e : G_e \subseteq D_{s(e)} \to D_{t(e)}$,
- $F = \{f_q\}_{q \in Q}$: a set of *vector fields*, where for each $q \in Q$, the predicate $f_q : D_q \to \mathbb{R}^n$ is Lipschitz on $\mathbb{R}^n$ and states through a differential equation ODE the possible continuous evolutions when the control of the hybrid system is in location $q \in Q$. The solution to the ODE is denoted by $x_i(t)$, where $x_i(t_0) = x_0$.

Definition 2: Hybrid System Execution
An execution χ of a hybrid system is a tuple

$$\chi = (\tau, \xi, \eta) \tag{2}$$

where

- $\tau = \{\tau_i\}_{i \in \mathbb{N}}$: a set of strictly increasing time instants represents discontinuity points (state events time instants).
- $\xi = \{\xi_i\}_{i \in \mathbb{N}}$: a set of initial conditions with $\xi_i \in D_q$.
- $\eta = \{\eta_i\}_{i \in \mathbb{N}}$ with $\eta_i \in E$: a sequence of discrete transitions.

The execution χ must satisfy the following conditions:

$$\xi_i = x_{s(\eta_i)}(\tau_i) \tag{3}$$

$$\tau_{i+1} = min\{t \geq \tau_i : x_{s(\eta_i)}(t) \in G_{\eta_i}\} \tag{4}$$

$$s(\eta_{i+1}) = t(\eta_i) \tag{5}$$

$$\xi_{i+1} = R_{\eta_i}(x_{s(\eta_i)}(\tau_{i+1})) \tag{6}$$

The conditions in (3) and (4) say that an event must occur at time τ_{i+1}. The condition in (5) says that the discrete evolution map must evolve in a way that is consistent with the discrete transitions. The condition in (6) says that the initial conditions must be in the image of the guards under the reset maps.

Definition 3: Chattering Execution
An *execution* χ of a hybrid system is chattering if there exist finite constants τ_∞ and C such that

$$\lim_{i \to \infty} \tau_i = \sum_{i=0}^{\infty} (\tau_{i+1} - \tau_i) = \tau_\infty \tag{7}$$

$$\forall i \geq C : \quad \tau_{i+1} - \tau_i = 0 \tag{8}$$

In the following we illustrate, via simple examples, the simulation of a chattering execution with and without event localization.

Example 1: Chattering on one switching manifold
Consider a mechanical stick-slip system of two blocks of masses m and M (Figure 1), where only the block of mass m is connected to a fixed support by a linear spring of stiffness k and is under the action of a sinusoidal external force u generated by an actuator P. We denote x_m and x_M to the position of the small mass m and the inertial mass M, and $\mathcal{F}$ to the tangential contact force on the frictional interface between the two masses. The friction between the inertial mass M and the ground is neglected. The origin of the displacements x_m and x_M is taken where the spring is unstretched. The external force u is modeled as a sine wave of frequency ω. The friction force between the two blocks is modeled phenomenologically, as a function of relative velocity between surfaces. Clearly, the system dynamics is discontinuous on a switching manifold Γ defined as the zeros of the scalar function $\gamma(t, x)$ representing the relative velocity between the two masses.

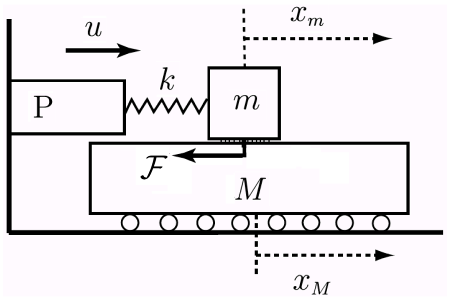


Figure 1. Schematic of the Stick-Slip system.

$$f(x) = \left\{ \begin{array}{l} \dot{x}_m = v_m \\ \dot{v}_m = \frac{1}{m}(u - kx_m - \mathcal{F}) \\ \dot{x}_M = v_M \\ \dot{v}_M = \frac{1}{M}\mathcal{F} \end{array} \right\} \tag{9}$$

$\mathcal{F}$ = 0 **init** $F_c \cdot sgn(\gamma_0)$ **reset** $[F_c$;-$F_c]$ **every** **up**$[\gamma$;-$\gamma]$;
$\gamma(t, x) = v_m(t) - v_M(t)$ **init** γ_0;

where F_c is the level of coulomb friction. The zero-crossing is described as an expression of the form **up(**γ**)** that becomes true when the sign of the event function $\gamma(t, x)$ switches from negative to positive during an execution, that is, **up(**γ**)**= $True$ if $\gamma(t_{i-1}, x_{i-1}) \leq 0 \wedge \gamma(t_i, x_i) > 0$ [13]. For a given simulation scenario, the trajectories initialized outside the switching surface $\Gamma = \{x \in \mathbb{R}^4 : \gamma(t, x) = 0\}$ reach the surface Γ in finite time and start to perform an infinite number of crossings on it. Hybrid systems simulation tools struggle even with this simple chattering example. In OpenModelica, the simulation of this example with adaptive time step and event localization, for the data set $m = M = 1[kg]$, $F_c = 0.4[N]$, $\omega = 0.055[rad/sec]$, $k = 1[N \cdot m^{-1}]$, $x_0 = [1 \ 1 \ 1 \ 0]^T$, terminates with a halt when the solution reaches the first chattering point. OpenModelica reports the following message: *Chattering detected around time 2.89522335659..2.89522338163 (100 state events in a row with a total time delta less than the step size 0.0012).*

OpenModelica Code:

```
model example1
parameter Real Fc=0.4, w=0.055,k=1,m=1,M=1;
parameter Real xm0=1, vm0=1, xM0=1, vM0=0;
Real xm,vm,xM,vM,F;
initial equation
xm = xm0; vm = vm0; xM = xM0; vM = vM0;
F = Fc * sign(vm - vM);
equation
when vm - vM <= 0 then
      F = -Fc;
elsewhen vm - vM >= 0 then
      F = Fc;
end when;
der(xm) = vm;
der(vm) = sin(w) - k * xm - F;
der(xM) = vM;
der(vM) = F;
end example1;
```

When simulating this example with a fixed time step without event localization, the solution trajectory exhibits a fast oscillation of control switching with high frequency components propagating around the switching surface Γ. Figure 2 shows the fixed time step simulation of Example 1 in the Acumen simulation tool for a fixed step size 0.0012 and data set $m = M = 1[kg]$, $F_c = 0.4[N]$, $\omega = 0.055[rad/sec]$, $k = 1[N \cdot m^{-1}]$, and $x_0 = [1\ 1\ 1\ 0]^T$.

Acumen Code:

```
model Main(simulator) =
initially
xm = 1, vm = 1, xM = 1, vM = 0,
xm'= 0, xM'= 0, vm'= 0, vM'= 0,
Fc = 0.4, w = 0.055, vr = 0,
k = 1, m = 1, M = 1, F = 0
always
xm' = vm,
vm' = (1/m) * (sin(w) - (k * xm) - F),
xM' = vM,
vM' = (1/M) * F,
if vm - vM >= 0 then F = Fc
elseif vm - vM <= 0 then F = -Fc noelse,
simulator.timeStep += 0.0012,
simulator.endTime += 10
```

Example 2. (Chattering on switching intersection)

Consider the simplest case of chattering on the intersection of two switching manifolds, Γ_1 and Γ_2, defined as the zeros of a set of functions $\gamma_1(t, x) = x_1(t)$ and $\gamma_2(t, x) = x_2(t)$ respectively.

$\dot{x}_1$ = 0 **init** -$sgn(\gamma_{10})$ **reset** [-1;1] **every up**[γ_1;-γ_1]
$\dot{x}_2$ = 0 **init** -$sgn(\gamma_{20})$ **reset** [-1;1] **every up**[γ_2;-γ_2]
γ_1 = x_1 **init** γ_{10}; γ_2 = x_2 **init** γ_{20}

In this example, the finite time convergence to the origin (0,0) is easy to establish, as the time intervals between two switches satisfy a geometric series and consequently have a finite sum. This system also has an infinity of spontaneous switches from the origin, that is, there is an infinity of trajectories which start with the initial data (0,0) and, except for the trivial solution that stays at the origin, they all cross the switching surfaces an infinity of times. Similarly to Example 1, with an adaptive time step and event localization, the simulation of this example gets stuck. In OpenModelica, the simulation terminates with the following error message: *Chattering detected around time 1.0000000001..1.0000000199 (100 state events in a row with a total time delta less than the step size 0.008).*

OpenModelica Code:

```
model example2
parameter Real x10 = 1, x20 = 2;
Real x1,x2, u1, u2;
initial equation
x1 = x10;
x2 = x20;
u1 = -sign(x1);
u2 = -sign(x2);
equation
when x1 <= 0 then
      u1 = 1;
elsewhen x1 => 0 then
      u1 = -1;
end when;
when x2 <= 0 then
      u2 = 1;
elsewhen x2 => 0 then
      u2 = -1;
end when;
der(x1) = u1;
der(x2) = u2;
end example2;
```

Figure 3 shows the fixed time step simulation of this example in Acumen for a step size of 0.0012 and initial conditions $x_1(0) = 1, x_2(0) = 2$. The smaller the step size the greater are the frequency components of the fast oscillation around the switching surfaces, and the higher is the time consumption of the simulation process.

Acumen Code:

```
model Main(simulator) =
initially x1 = 1, x2 = 2, x1' = 0, x2' = 0
always
if x1 => 0 then x1' = - 1
elseif x1 <= 0 then x1' = 1 noelse,
if x2 => 0 then x2' = - 1
elseif x2 <= 0 then x2' = 1 noelse,
simulator.timeStep += 0.0012,
simulator.endTime += 2
```

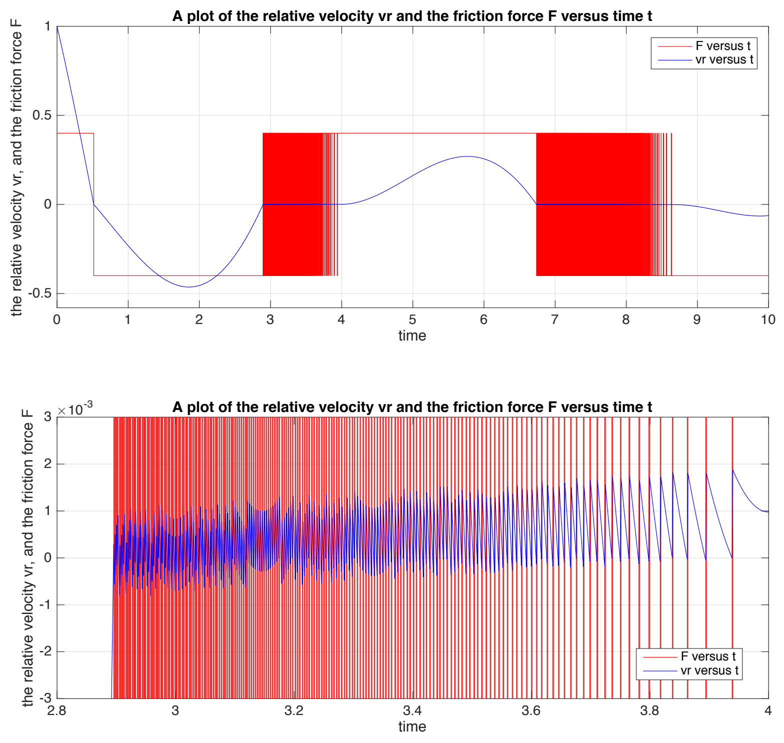


Figure 2. Chattering execution: Fixed time step simulation of Example 1 in Acumen without event localization. Up: time evolution of the event function and the control input with high chattering oscillation. Down: zoom on the chattering window around the switching surface $\Gamma = \{x \in \mathbb{R}^4 : v_m(t) - v_M(t) = 0\}$.

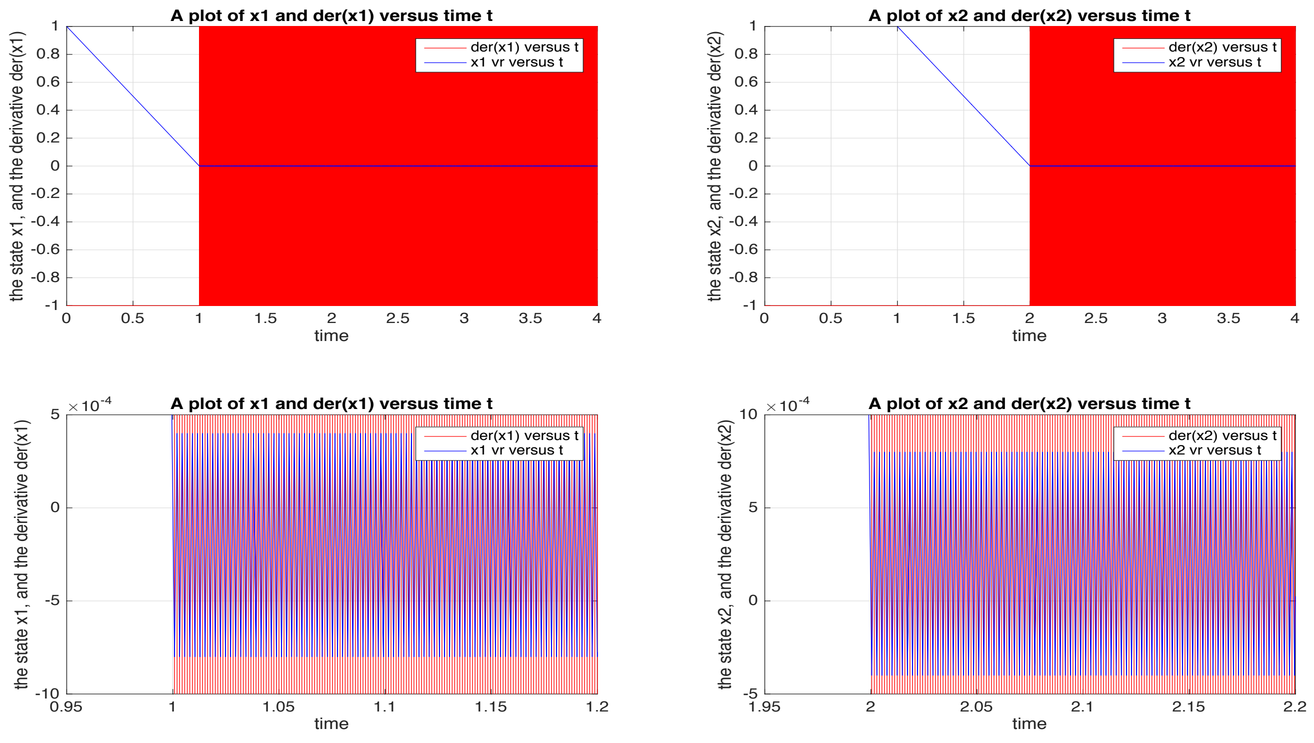


Figure 3. Chattering execution: Fixed time step simulation of Example 2 in Acumen without event localization. Up: time evolution of the states and their derivatives with high chattering oscillation. Down: zoom on the chattering windows around $\Gamma_1 = \{x \in \mathbb{R}^2 : x_1(t) = 0\}$ and the origin $\Delta = \{x \in \mathbb{R}^2 : x_1(t) = 0 \wedge x_2(t) = 0\}$.

As in physical hybrid systems there is no chattering, the simulator then should be able to correctly simulate the chattering-free behavior of the real physical hybrid system. This motivates the need for chattering-free simulation semantics to carry the execution (or trajectory) past the point at which chattering occurs.

III. Chattering-Free Semantics

We consider a hybrid system $\mathcal{H}$ with a finite set of discrete states $q \in Q$, where the entire state space $S \subseteq \mathbb{R}^n$ of the hybrid system is split into 2^p different invariants $D_q \subset \mathbb{R}^n$ by the intersection of p intersected switching manifolds $\Gamma_j \subset \mathbb{R}^{n-1}$, defined as the zero-set of scalar event functions $\gamma_j(t,x)$,

$$\Gamma_j = \{x \in \mathbb{R}^n : \ \gamma_j(t,x) = 0 \ ; \quad j = 1,2,...,p\} \tag{10}$$

The flow map vector field $f(t,x)$ of the hybrid system is discontinuous on all the switching manifolds Γ_j, that is for each Γ_j, the dynamics $f(t,x)$ is a triplet (X_j, Y_j, γ_j), where X_j and Y_j are $\mathcal{C}^r$ ($r \geq 1$ or $r = \infty$) vector fields and are extendable over a full neighborhood of the boundary Γ_j. We assume that 0 is a regular value of γ_j and thus Γ_j is a smooth $\mathbb{R}^{(n-1)}$ submanifold. Each discontinuity manifold Γ_j separates S into two open subsets S_{X_j} and S_{Y_j},

$$S_{X_j} = \{x \in \mathbb{R}^n : \gamma_j(t,x) < 0\} \tag{11}$$

$$S_{Y_j} = \{x \in \mathbb{R}^n : \gamma_j(t,x) > 0\} \tag{12}$$

The dynamics of $\mathcal{H}$ in S_{X_j} and S_{Y_j} are defined by the flows of X_j and Y_j respectively.

$$\dot{x} = f_j(t,x) = \left\{ \begin{array}{ll} X_j(t,x) & for \quad x \in S_{X_j} \\ Y_j(t,x) & for \quad x \in S_{Y_j} \end{array} \right\} \tag{13}$$

Each switching manifold Γ_j is divided into the crossing set

$$\Gamma_{j_c} = \{x \in \Gamma_j : \mathcal{L}_{X_j}\gamma_j(t,x) \cdot \mathcal{L}_{Y_j}\gamma_j(t,x) \geq 0\} \tag{14}$$

and the sliding set

$$\Gamma_{j_s} = \{x \in \Gamma_j : \mathcal{L}_{X_j}\gamma_j(t,x) \cdot \mathcal{L}_{Y_j}\gamma_j(t,x) < 0\} \tag{15}$$

where $\mathcal{L}_{X_j}$ (respectively $\mathcal{L}_{Y_j}$) denotes the Lie (or directional) derivative of γ_j with respect to the vector field X_j (respectively Y_j) at the discontinuity point $x \in \Gamma_j$, that is,

$$\mathcal{L}_{X_j}\gamma_j(t,x) = \left(\frac{\partial \gamma_j(t,x)}{\partial x}\right) \cdot X_j(x(t)) \tag{16}$$

$$\mathcal{L}_{Y_j}\gamma_j(t,x) = \left(\frac{\partial \gamma_j(t,x)}{\partial x}\right) \cdot Y_j(x(t)) \tag{17}$$

Note that the Lie derivative, $\mathcal{L}_{X_j}$ (respectively $\mathcal{L}_{Y_j}$), has a geometric interpretation here as the time derivative of $\gamma_j(t,x)$ along trajectories of the ODE X_j (respectively Y_j). If, at a point $x \in \Gamma_j$, the vector field X_j points toward Γ_j and Y_j points away from Γ_j, or vice versa, then $x \in \Gamma_{j_c}$. In the former case, where X_j points toward and Y_j points away from Γ_j, an orbit of $\mathcal{H}$ that arrives at $x \in \Gamma_j$ following the flow of X_j continues from $x \in \Gamma_j$ following the flow of Y_j. Thus the orbit is continuous and crosses from S_{X_j} to S_{Y_j}. Similarly, if Y_j points toward Γ_j while X_j points away, then the orbit crosses, in the same way, from S_{Y_j} to S_{X_j}. The sufficient condition for the hybrid system $\mathcal{H}$ to exhibit a chattering back and forth between the two domains S_{X_j} and S_{Y_j} at $x \in \Gamma_j$ requires that both vector fields X_j and Y_j point toward Γ_j (i.e. $x \in \Gamma_{j_s}$). That is, the gradient of continuous-time behavior in each one of two adjacent subsets S_{X_j} and S_{Y_j} is directed towards their common switching surface Γ_j. When, in either of the two adjacent subsets, an infinitesimal step causes a mode change. In the new subset, the gradient directs behavior to the previous subset and, after another infinitesimal step, a change to the previous subset occurs.

Lemma 1: Chattering detection on a single switching manifold
A chattering is detected in the integration time interval $[t_i, t_{i+1}]$ if the following three constraints are satisfied:

$$\gamma_j(t_i,x) \cdot \gamma_j(t_{i+1},x) < 0 \tag{18}$$

$$\gamma_j(t_{i+1}(\sigma),x) = 0; \quad \sigma \in \ (0,1) \tag{19}$$

$$\mathcal{L}_{X_j}\gamma_j(t_{i+1}(\sigma),x) \cdot \mathcal{L}_{Y_j}\gamma_j(t_{i+1}(\sigma),x) < 0 \tag{20}$$

Equations (18) and (19) indicate that a zero-crossing is detected and located in $[t_i, t_{i+1}]$, while equation (20) indicates that the switching point $(t_{i+1}(\sigma),x) \in \Gamma_j$ belongs to the sliding set Γ_{j_s}.

Lemma 2: Chattering detection on switching intersection
In the integration time interval $[t_i, t_{i+1}]$, a chattering is detected on the intersection of k switching manifolds Γ_j if equations (18) to (20) are satisfied for all $j = 1,2,\cdots,k \leq p$. In this case, the execution of the hybrid system $\mathcal{H}$ chatters back and forth between all the invariants D_q in the neighborhood of the intersection.

When a chattering execution is detected at $x \in \Gamma_j$ by the simulator, the dynamics at x should be defined by an equivalent chattering-free vector f_s which is the unique convex linear combination of $X_j(t,x)$ and $Y_j(t,x)$ that is tangent to Γ_j at x. An orbit that starts initially at $x \in \Gamma_j$ slides along the surface Γ_j. Specifically, the chattering-free vector field f_{js} is defined as

$$f_{js}(t,x) = \frac{1-\delta_j(\gamma_j(t,x))}{2} \cdot X_j(t,x) + \frac{1+\delta_j(\gamma_j(t,x))}{2} \cdot Y_j(t,x) \tag{21}$$

where

$$\delta_j(\gamma_j(t,x)) = \frac{\mathcal{L}_{X_j}\gamma_j(t,x) + \mathcal{L}_{Y_j}\gamma_j(t,x)}{\mathcal{L}_{X_j}\gamma_j(t,x)) - \mathcal{L}_{Y_j}\gamma_j(t,x)} \tag{22}$$

provided that $\mathcal{L}_{X_j}\gamma_j(t,x)) \neq \mathcal{L}_{Y_j}\gamma_j(t,x))$. In the case where $\mathcal{L}_{X_j}\gamma_j(t,x)) = \mathcal{L}_{Y_j}\gamma_j(t,x))$ for $x \in \Gamma_{j_s}$ (further implying that $\mathcal{L}_{X_j}\gamma_j(t,x)) = \mathcal{L}_{Y_j}\gamma_j(t,x)) = 0$) we use the higher order conditions to check whether we should keep sliding, or exit from sliding. For more details about the higher order conditions analysis for sliding we refer the reader to [6]. Points $x \in \Gamma_{j_s}$ are called tangency points of X_j (respectively Y_j) if $\mathcal{L}_{X_j}\gamma_j(t,x)) = 0$ (respectively $\mathcal{L}_{X_j}\gamma_j(t,x)) = 0$). The scalar product function $\mathcal{L}_{X_j}\gamma_j(t,x) \cdot \mathcal{L}_{Y_j}\gamma_j(t,x)$ changes sign at a tangency point and therefore such points are generically positioned at the boundary of the sliding set Γ_{j_s}. One of the most challenging issues to deal with is the computation of the chattering-free dynamics f_{js} when the chattering occurs on a switching intersection. In particular, in such cases, applying Filippov's classical theory requires solving stiff nonlinear equations for computing the chattering-free coefficients $\delta_j(\gamma_j(t,x))$. This is computationally time-consuming and does not guarantee a unique solution for $\delta_j(\gamma_j(t,x))$, if the solution chatters on the intersection of more than two switching manifolds. Another challenge is the enumeration of all the 2^p open regions (invariants) in the neighborhood of the switching intersection. This can be quite time consuming in simulation, when the system starts to exhibit a chattering on the intersection of sufficiently large number of intersected switching manifolds. One possible way to deal with these challenges is to compute the intersection equivalent chattering-free dynamics iteratively (Algorithm 1). The number of iterations that need to be performed by Algorithm 1 to compute the chattering-free dynamics is equal to the total number p of the intersected switching manifolds Γ_j on which the chattering occurs. The main benefit of the iterative approach in Algorithm 1 is that it allows us to eliminate chattering efficiently without any need to enumerate the modes, even when the chattering is occurring on an intersection $\Delta = \bigcap_j(\Gamma_j), j = 1, 2, ..., p$ of a large number p of intersected switching manifolds. Another benefit is that there is no need to solve stiff nonlinear equations for the computation of the chattering-free coefficients $\delta_j(g_j(t,x))$ in case of chattering on a switching intersection with $p > 1$.

```
Data: $f(t,x)$, $\gamma_j(t,x)$.
Result: $f_{\Delta_s}(t,x) = f_{js}(t,x)$
Initialization:
$j = 1$;
$f(x(t)) = f_j(x(t))$ (Equation (13));
while $j \le p$ do
    Compute $f_{js}(t,x)$ (Equations (21) and (22));
    Set $f_j(t,x) = f_{js}(t,x)$;
    $j = j + 1$;
    Repeat;
end
```

Algorithm 1: Iterative computation of the equivalent chattering-free dynamics.

Example 2 revisited:
To generate the intersection chattering-free dynamics $f_{\Delta_s}(t,x)$ on the intersection (the origin) $\Delta = \Gamma_1 \cap \Gamma_2$, Algorithm 1 performs two iterations:

- In **Iteration1**, the algorithm computes the equivalent chattering-free dynamics on Γ_1, given by (23).
- In **Iteration2**, the algorithm computes the equivalent chattering-free dynamics on the intersection Δ, given by (24).

$$f_{1s}(t,x) = \begin{bmatrix} 0 \\ \left\{ \begin{array}{rl} -1 & for \;\; x_2(t) > 0\} \\ 1 & for \;\; x_2(t) < 0\} \end{array} \right\} \end{bmatrix} \tag{23}$$

$$f_{\Delta s}(t,x) = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{24}$$

It's worth noting that Algorithm 1 applies not only to the case of chattering on switching intersections but also to chattering on a single switching manifold (i.e. chattering between two dynamics). In the latter case, Algorithm 1 generates the chattering-free dynamics within one iteration.

IV. Prototype Implementation

In this section we show via a prototypical implementation how the chattering-free semantics can be embedded in the main simulation loop of a simulation tool for hybrid systems. For the prototype implementation we have used Acumen, which provides a simulation environment for cyber physical systems and hybrid systems. It was developed as an extension of event-driven formalisms that have a similar flavor to synchronous languages. Acumen adds systems of equations or functions on dense time for the purpose of describing the physical environment surrounding the purely cyber controllers that are describable in synchronous formalisms. The semantics in Acumen is defined by a series of translations from a large source language into progressively smaller subsets of the language. The purpose of the core language is to serve as the minimal subset needed to express all the features of the full source language. For more details about the formal and operational semantics of the Acumen language, we refer the reader to [15][16]. Models of hybrid systems are simulated in Acumen by a fine interleaving of sequences that can consist of multiple discrete computations, followed by a single computation updating the values that should evolve continuously. Thus, simulating what is happening at any single instant in time consists of zero or more discrete steps followed by a single continuous step. The discrete steps capture sudden changes in the system's state (e.g. the impact of two objects), and consist of collecting and evaluating all active discrete actions (discrete assignments or structural actions) in the program until the whole system is stabilized. A system is stabilized when no more discrete steps are required. Once all discrete actions have taken place, the system moves on to perform all adjustments to the continuous state of the system. The continuous step performs all updates in parallel, meaning that all updates are based on the state that results after the sequence of discrete steps, rather than some later state that resulted from other continuous updates. In our chattering-free semantics, the solver algorithm has to decide, at the state event, whether the solution trajectory should cross the switching surface transversally or slide on it. The computation of the chattering-free solution is split into two phases: I) chattering detection, and II) chattering elimination.

The chattering detection phase starts once a state event is detected. The algorithm inspects whether the state event is a chattering event or not by checking, at the state event, whether or not the sufficient condition for chattering is satisfied. This implies analyzing the gradients of the continuous time behavior before and after the state event. The directional derivatives (Lie derivatives) of the system are computed and evaluated at the beginning and at the end of the completed integration step at which the state event has been detected. The nature of our chattering detection semantics is to compare the sign of the directional derivatives (normal projections) at the beginning and the end of the time integration step $[t_i, t_{i+1}]$ in which a state event is occurred, and, if it changes, declare a chattering event. We should then have access to the dynamics at t_i (i.e. in the previous domain before the event) and at t_{i+1} (i.e. in the next domain after the event). To achieve this, we use two stores s_0 (at the beginning of the time-step, i.e. t_i) and s_1 (at the end of the time step, i.e. t_{i+1}). The reference semantics of Acumen transforms each store by repeatedly calling a function `step()` that implements a state machine with four states (called step types): Initial, Discrete, FixedPoint, and Continuous. If the system is stabilized after the discrete steps, the chattering-free interpreter does the following:

a) Performs all adjustments to the continuous state of the systems and sets the result type to *FixedPoint*,
b) Stores the step output in store s,
c) Computes and evaluates the flow map vector field (dynamics) (i.e. $X_j(t,x)$ and $Y_j(t,x)$) from the store s_0 and s_1,
d) Compute and evaluate the event function $\gamma_j(t,x)$. The technique of computing the switching functions $\gamma_j(t,x)$ involves parsing the `if` statement (`if Expr then Action`). The branches of an `if` statement are scanned sequentially until a guard sequence `Expr` which evaluates to true is found. The found predicate `Expr` is then converted into a set of switching functions, by converting a Boolean relation $\bowtie$ of the form {`Expr`: = $a \bowtie b$} into a switching function $\gamma_j = a - b$, where $\bowtie \in \{<, \leq, >, \geq\}$. The sub-expressions a and b are usually made of constants r, computed variables l, and state variables x (all coming from a finite set $\mathcal{V}$), as well as of arithmetic operations ($\diamond \in \{+, -, \times, \div\}$).
e) The resulting switching functions $\gamma_j(t,x)$ and the dynamics vector $X_j(t,x)$ (respectively $Y_j(t,x)$) are then used to compute the directional derivatives $\mathcal{L}_{X_j}$ (respectively $\mathcal{L}_{Y_j}$).

Therefore, during the continuous integration of the system, for a given time step from t_i to t_{i+1}, we have two stores: s_0 at t_i and s_1 at t_{i+1}. At the end of each time step we use $\mathcal{L}_{X_j}$ and $\mathcal{L}_{Y_j}$ to check whether or not the execution is chattering by checking the conditions in Lemma 1, when a state event is detected on a single switching manifold; and the conditions in Lemma 2, when a state event is detected on a switching intersection $\Delta = \bigcap_{j=1}^{p}(\Gamma_j), p >= 2$. In the chattering elimination phase, Algorithm 1 is employed to compute, internally in the main simulation loop, the equivalent chattering-free dynamics, giving the dynamics before and after the state event (i.e. $X_j(t,x)$ and $Y_j(t,x)$) and the directional derivatives of the system onto the switching manifold (i.e. $\mathcal{L}_{X_j}$ and $\mathcal{L}_{X_j}$). Once the solution is at the final time of a simulation, i.e. the current time is equal to or greater than the end time, and the previous step was FixedPoint, then the simulation is terminated.

V. Simulation Results and Discussion

Figure 4 shows the chattering-free simulation in Acumen for the system in Example 1, with a fixed time step of size 0.0012 and simulation data set $m = M = 1[kg]$, $F_c = 0.4[N]$, $k = 1[N \cdot m^{-1}]$, and $x_0 = [x_{m0}\ v_{m0}\ x_{M0}\ v_{M0}]^T = [1\ \ 1\ \ 1\ \ 0]^T$. The external force u was modeled as a sine wave of frequency $\omega = 0.055[rad/sec]$. The first chattering event was detected at $t = 2.89$. During a simulation time of 10 seconds, two chattering windows were detected ($\mathbb{I}_1 = [2.89 \cdot \cdot 3.99]$, $\mathbb{I}_2 = [6.73 \cdot \cdot 8.69]$), and the system was integrated in these two chattering intervals with the chattering-free dynamics generated internally and iteratively (Algorithm 1) by the simulator. In Figure 5, Example 2 was simulated with the data set $x_1(0) = 1, x_2(0) = 2$ and a step size 0.0012. The first chattering event was detected at $t = 1$, where the solution trajectory started to chatter on the switching manifold $\Gamma_1 = \{x_1(t) = 0 \wedge x_2(t) > 0\}$. The chattering-free dynamics $(0, x_2')$ generated by Algorithm 1 allowed the solution to slide on the surface Γ_1. The solution trajectory converged to the origin ($x_1(t) = 0, x_2(t) = 0$) at $t = 2$, where the chattering was replaced by sliding on the intersection, by integrating the system with the chattering-free dynamics (0,0).

Other chattering case studies (Table 1) were simulated with different simulation scenarios for the purpose of evaluating the effeciency of our chattering-free implementation. In particular, a performance analysis and testing was done,

Table I
Summary of the case studies used in the performance analysis.

Case Study	The Chattering Model
1	$\dot{x}_1 = v_1$
	$\dot{v}_1 = \sin(\omega) - x_1 - (F_c \cdot \text{sign}(v_r))$
	$\dot{x}_2 = v_2$
	$\dot{v}_2 = F_c \cdot \text{sign}(v_r)$
	$v_r = v_1 - v_2$
	Initialization:
	$x_1(0) = 1,\ v_1(0) = 1,\ x_2(0) = 1,\ v_2(0) = 0$
	Parameters: ω=0.055, F_c=0.4
	Fixed Time Step: 0.0012
	Simulation Time: 10
2	$\dot{x}_1 = -\ \text{sign}\ (x_1)$
	$\dot{x}_2 = -\ \text{sign}\ (x_2)$
	Initialization: $x_1(0) = 1,\ x_2(0) = 2$
	Fixed Time Step: 0.0012
	Simulation Time: 4
3	$\dot{x}_1 = -3x_1 + x_2 - u$
	$\dot{x}_2 = -3x_1 + x_3 + u$
	$\dot{x}_3 = -x_1 - 0.25u$
	$u = \text{sign}(x_1)$
	Initialization: $x_1(0) = 0.5,\ x_2(0) = 3,\ x_3(0) = 0.1$
	Fixed Time Step: 0.0012
	Simulation Time: 10
4	$\dot{x} = -2 \cdot \text{sign}(y)$
	$\dot{\phi} = 0.5$
	$y = x - (0.3 \cdot \exp\ (\phi))$
	Initialization: $x(0) = 1,\ \phi(0) = 0$
	Fixed Time Step: 0.0012
	Simulation Time: 6
5	$\dot{x}_1 = x_2$
	$\dot{x}_2 = \sin(\phi) - x_1 - (F_c \cdot \text{sign}(x_2))$
	$\dot{\phi} = \omega$
	Initialization: $x_1(0) = 1,\ x_2(0) = 0,\ \phi(0) = 1$
	Parameters: ω = 0.4, F_c = 0.4
	Fixed Time Step: 0.0012
	Simulation Time: 20
6	$\dot{x}_1 = -2x_1 + x_3 - u$
	$\dot{x}_2 = -101x_1 + x_3 + 4u$
	$\dot{x}_3 = -100x_1 - u$
	$u = \text{sign}(x_1)$
	Initialization: $x_1(0) = 0.5,\ x_2(0) = -1,\ x_3(0) = 1$
	Fixed Time Step: 0.0012
	Simulation Time: 1

where the implementation performance was evaluated based on the following two criteria:

1) The time spent by the simulator to compute the chattring-free solution.
2) The precision and accuracy of the chattering-free simulation. Namely, whether or not:
    a) All the chattering events are captured.
    b) The solution enters the sliding mode instantly when the chattering occurs.
    c) The solution exits from sliding exactly at the tangency points on the boundary of the sliding set.
    d) The sliding dynamics generated by the solver are correct (based on the evaluation from (b) and (c)).

In Table 2, each case study is associated with its set of simulation and evaluation results.

Table II
Summary of results from the performance analysis and testing of our chattering-free implementation in Acumen.

Case St.	ACED?†	CTI‡[s]	TISE♭[s]	MST♮[s]
1	YES	$\mathbb{I}_1 = [2.89 \cdot\cdot 3.99]$	3.99	1.864
		$\mathbb{I}_2 = [6.74 \cdot\cdot 8.69]$	8.69	
2	YES	$\mathbb{I}_1 = [1.00 \cdot\cdot 4.00]$	-	0.480
		$\mathbb{I}_2 = [2.00 \cdot\cdot 4.00]$	-	
3	YES	$\mathbb{I}_1 = [2.65 \cdot\cdot 3.42]$	3.42	1.605
		$\mathbb{I}_2 = [8.22 \cdot\cdot 9.03]$	9.03	
4	YES	$\mathbb{I}_1 = [0.32 \cdot\cdot 5.18]$	5.18	0.765
5	YES	$\mathbb{I}_1 = [0.00 \cdot\cdot 3.74]$	3.74	3.196
		$\mathbb{I}_2 = [8.92 \cdot\cdot 11.0]$	11.0	
		$\mathbb{I}_3 = [16.5 \cdot\cdot 19.2]$	19.2	
6	YES	$\mathbb{I}_1 = [0.45 \cdot\cdot 1.00]$	-	0.164

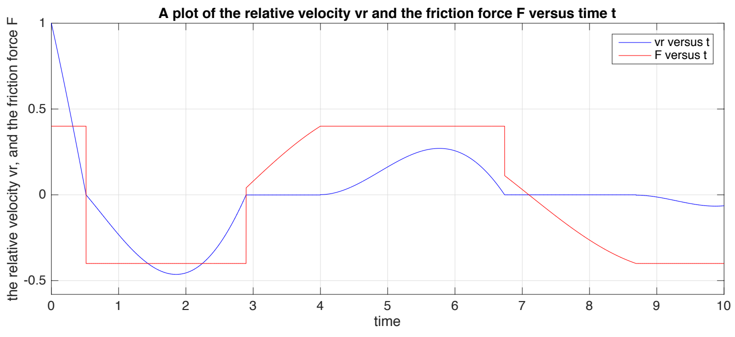
† ACED?: All Chattering Events are Detected?
‡ CTI: Chattering Time Intervals.
♭ TISE: Time Instants of the Smooth Exits.
♮ MST: Mean Simulation Time.

It's worth noting that the computational load in our proposed chattering-free semantics is not very considerable, as the chattering detection and elimination is done "on the fly", internally in the main simulation loop of the simulator. For the simulation scenario illustrated in Figure 4, the implemented chattering-free interpreter of Acumen has terminated the chattering-free simulation successfully within 1.864 seconds. A simulation of the same system with the same data set and simulation conditions (i.e. fixed step size = 0.0012) was terminated within 10.288 seconds when the chattering regularization was done manually in the model level by the user. Of course, this gap of 8.424 seconds - between manipulating the chattering internally and automatically by the solver and manipulating it manually by the user - is for a fixed integration step size of 0.0012. The smaller the step size, the larger this gap. Table 3 compares the time taken in the Acumen simulation tool to generate the chattering-free solution for each of the six case studies when: I) treating the chattering internally by the simulator in run-time, without enumeration of modes, and II) treating the chattering manually by the user (i.e. on the model level).

Table III
Mean Time of the Chattering-free Simulation: Manual manipulation by the user versus automatic detection and elimination by the solver

Case St.	Chattering-free Manually (User)	Chattering-free Automatically (Solver)
1	10.288 [s]	1.864 [s]
2	5.212 [s]	0.480 [s]
3	7.221 [s]	1.605 [s]
4	1.728 [s]	0.765 [s]
5	15.33 [s]	3.196 [s]
6	0.888 [s]	0.164 [s]

Another advantage of our technique is that a manual manipulation of chattering requires advanced knowledge of sliding mode control by the user. However, in our proposed technique there is no need at all for the user to have



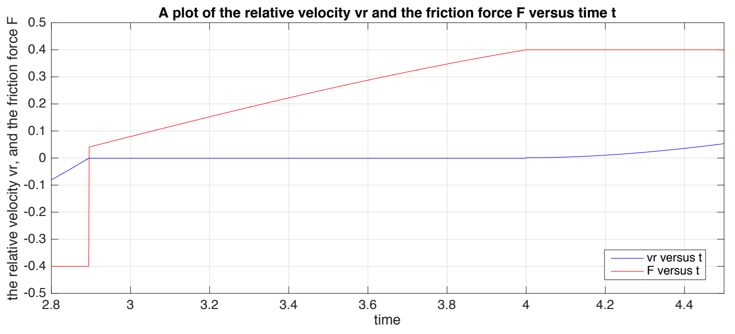


Figure 4. Chattering-free simulation of the system in Example 1 in Acumen. Up: time evolution of the event function and the control input. Down: zoom on the first window of sliding on the switching surface $\Gamma = \{x \in \mathbb{R}^4 : v_m(t) - v_M(t) = 0\}$.

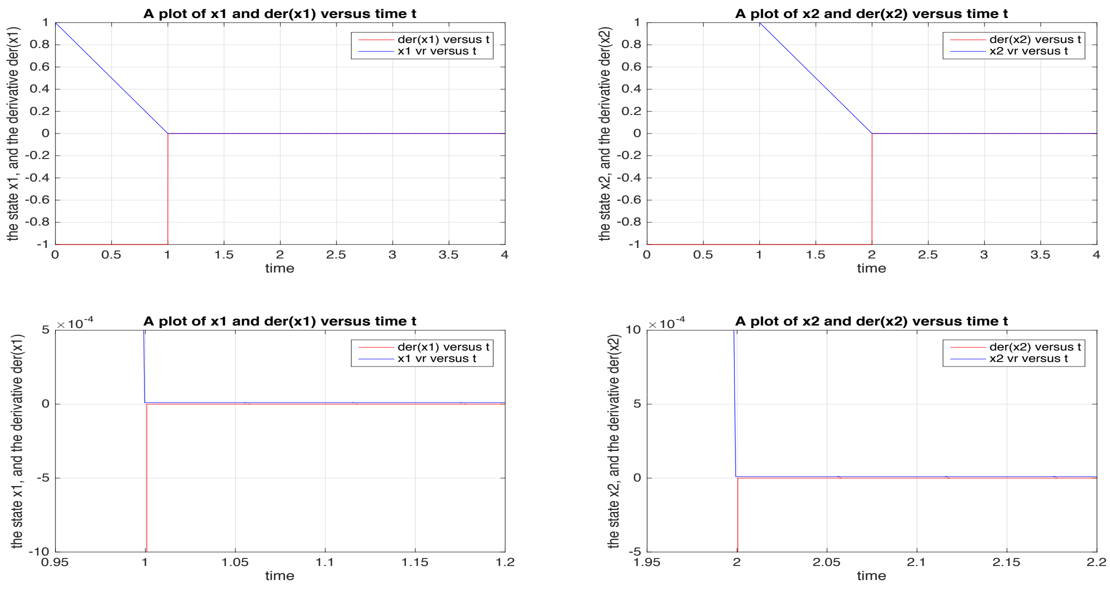


Figure 5. Chattering-free simulation of the system in Example 2 in Acumen. Up: time evolution of the states and their derivatives. Down: zoom on the sliding windows around $\Gamma_1 = \{x \in \mathbb{R}^2 : x_1(t) = 0\}$ and the origin $\Delta = \{x \in \mathbb{R}^2 : x_1(t) = 0 \wedge x_2(t) = 0\}$.

such knowledge, as it is the simulator interpreter who detects and regularizes the chattering execution. Furthermore, treating the chattering manually requires enumerating all the modes in the neighborhood of the chattering switching manifold/intersection. It also requires solving stiff nonlinear equations for the computation of chattering-free coefficients in equation (22), for which a unique solution is not guaranteed when the system chatters between more than four dynamics. That is, the manual manipulation of chattering is limited to cases of chattering between 2^p dynamics where $p \in \{1, 2\}$.

VI. Conclusions

In this paper we presented a novel approach, semantics, and prototype implementation of a chattering-free simulation technique for the reliable detection and elimination “on the fly” of chattering behavior when simulating chattering models of hybrid systems. The main benefit of the proposed approach is that the chattering detection and elimination is done internally in the simulation cycle of the simulator, without any manual manipulation on the model level (i.e. adding small hysterisis, chattering-free manually), and without any need to enumerate the modes in the neighborhood of the chattering switching manifold. Our chattering-free semantics robustly handles the case of chattering on a switching intersection without any need to solve stiff nonlinear equations for the computation of the chattering-free coefficients. Furthermore, this paper provides guidance for the development of chattering-free versions for hybrid systems simulation tools. Finally, the simulation results on a set of representative examples have demonstrated that our chattering-free algorithm is efficient and precise enough to correctly simulate the chattering-free behavior of the real physical hybrid system.

Acknowledgment

This work was supported by the Collège Doctoral International (CDI) of the European University of Brittany, Brittany Regional Council, ITEA2 MODRIO project, the US National Science Foundation award CPS-1136099, the Swedish Knowledge Foundation (KK), and the Center for Research on Embedded Systems (CERES).

References

[1] C. Cai, R. Goebel, R. Sanfelice, and A. Teel, Hybrid systems: limit sets and zero dynamics with a view toward output regulation, Springer-Verlag, 2008.

[2] J. Zhang, K. H. Johansson, J. Lygeros, and Sh. Sastry, Zeno hybrid systems, International Journal of Robust and Nonlinear Control, 11(05), pp.435-451, 2001.

[3] M. Biák, T. Hanus, and D. Janovská, Some applications of Filippov’s dynamical systems, Journal of Computational and Applied Mathematics, 254, pp.132–143, 2013.

[4] A. Aljarbouh, and B. Caillaud, Robust Simulation for Hybrid Systems: Chattering Path Avoidance, Linkoping Electronic Conference Proceedings, 119(018), pp.175-185, 2015.

[5] K. H. Johansson, A.E. Barabanov, and K.J. Astrom, Limit cycles with chattering in relay feedback systems, IEEE Transactions on Automatic Control, 47(9), pp.1414-1423, 2002.

[6] A. Aljarbouh, and B. Caillaud, On the Regularization of Chattering Executions in Real Time Simulation of Hybrid Systems, Baltic Young Scientists Conference Proceedings, https://hal.archives-ouvertes.fr/hal-01246853v2, pp.49-66, 2015.

[7] D. Weiss, T. Kupper, and H.A. Hosham, Invariant manifolds for nonsmooth systems with sliding mode, Mathematics and Computers in Simulation, 110, March 2014.

[8] M. di Bernardo, C. J. Budd, A. R. Champneys, Piotr Kowalczyk, A. B. Nordmark, G. O. Tost, and P. T. Piiroinen, Bifurcations in Nonsmooth Dynamical Systems, SIAM Review, 50(5), pp.629-701, 2008.

[9] T. Blochwitz, M. Otter, J. Akesson, M. Arnold, C. ClauB, H. Elmqvist, M. Friedrich, A. Junghanns, J. Mauss, D. Neumerkel, H. Olsson, and A. Viel, Functional Mockup Interface 2.0: The Standard for Tool independent Exchange of Simulation Model, In Proceedings of 9th International Modelica Conference, Munich, Germany, 076(17), pp.173-184, 2012.

[10] V. I. Utkin, Sliding Mode in Control and Optimization, Springer Berlin Heidelberg, ISBN: 978-3-642-06029-8, 2004.

[11] R. I. Leine, and H. Nijmeijer, Dynamics and Bifurcations of Non-Smooth Mechanical Systems, Springer Berlin Heidelberg, ISBN: 978-3-642-84379-2, 1992.

[12] A.F. Filippov, Differential Equations with Discontinuous Righthand Sides, Springer Netherlands, ISBN: 978-94-015-7793-9, 1988.

[13] P. Schrammel. Logico-Numerical Verification Methods for Discrete and Hybrid Systems, PhD dissertation, 2012.

[14] J. Lygeros, C. Tomlin, and Sh. Sastry. Hybrid Systems: Modeling, Analysis and Control, Lecture Notes on Hybrid Systems, 2008.

[15] W. Taha, et al., Acumen: An Open-source Testbed for Cyber-Physical Systems Research, In Proceedings of EAI International Conference on CYber physiCaL systems, iOt and sensors Networks, October 2015. http://mahilab.rice.edu/sites/mahilab.rice.edu/files/publications /cyclone15Taha.pdf.

[16] Y. Zeng, C. Rose, W. Taha, A. Duracz, K. Atkinson, R. Philippsen, R. Cartwright, M. O’Malley. Modeling Electromechanical Aspects of Cyber-Physical Systems, Journal of Software Engineering for Robotics, 1(1), September 2015, 123-126, ISSN: 2035-3928.