Formal Verification of an Automotive Engine Controller in Cutoff Mode

Tiziano Villa[§] Howard Wong-Toi[†] Andrea Balluchi[§] Jörg Preußig[♯]
Alberto L. Sangiovanni-Vincentelli[§‡] Yosinori Watanabe[¶]

Abstract

We describe formal verification of convergence and performance properties of an engine control algorithm being developed for Magneti-Marelli. We study the cutoff mode, where the driver releases the accelerator and the controller regulates fuel injection to minimize the oscillations while decelerating. The engine and its controller are modeled with hybrid automata and the sliding action of the hybrid controller is formally verified with the model checker HYTECH.

1 Introduction

1.1 Verification of embedded systems

Embedded systems are informally defined as a collection of programmable parts surrounded by ASICs and other standard components that interact continuously with an environment through sensors and actuators. Embedded systems often are used in life-critical situations, where reliability and safety are more important criteria than performance. The closed-loop system—which involves a discrete controller and its analog environment—has behaviors that are inherently hybrid, i.e., both discrete and continuous in nature. Therefore the correctness of such systems cannot be established without consideration of the analog real-world environment in which it operates. Prototyping is viable for simple embedded systems, but typically too expensive for complex ones. Hybrid system simulation, provided by such tools as Matlab and Xmath, is more feasible. However, while simulation runs can provide great insight into the behavior of the closed-loop system, as with all simulation methods, one is often left wondering if sufficient corner cases have been considered.

Therefore, formally verifying high-level designs of hybrid systems appears to be very desirable for safety-critical systems [1]. Given a mathematical model of a system, one uses automated methods to prove rigorously that its real-time requirements are met. Rather than simulating over some fraction of the possible inputs, all legal input sequences are considered. Our verification methodology consists of the following steps: (1) Identify the concurrent components in the system. (2) Model each component as a *hybrid automaton*, essentially a finite labeled transition system augmented with real-valued variables subject to differential inclusions. (3) Conservatively approximate every nonlinear hybrid automaton by a *linear hybrid automaton* [5]. Intuitively, this subclass requires the continuous dynamics to obey constant polyhedral differential inclusions (the flow conditions cannot depend on x and so dynamics of the form $\dot{x} = x$ are prohibited). (4) Analyze the collection of linear hybrid automata with the model-checking tool HYTECH which uses automated semi-decision procedures [6].

The application of this methodology to industrial design is not an easy task. The state of the art of industrial-strength formal verification techniques is limited to equivalence checking. Strong results are available for property checking on discrete systems but hybrid-system formal verification is still in its infancy. In this paper, we apply our methodology to a real-world complex embedded system design: a control strategy for a region of operation of a combustion engine [2, 3]. The controller was designed in response to a problem proposed by Magneti-Marelli, a major automotive electronics company. We formally verify convergence and performance requirements of the controller. A modified version of the controller presented here will be in production in vehicles this year.

1.2 Cutoff control problem

We consider control of the engine once the driver has released the accelerator pedal, thereby expressing a desire to have zero torque delivered by the engine. The control objective is to reach injection cutoff while minimizing acceleration oscillations, as these lead to passenger discomfort. However, if fuel injection is abruptly cut off, the vehicle may exhibit very undesirable acceleration oscillations. In order to minimize these oscillations, the controller makes intelligent decisions about when and how to cut off fuel injection.

We consider the original Magneti-Marelli specification of the car [2]. A revised model and an improved controller appear in [3]. The system consists of the engine, which includes the driveline and the cylinders, and its

§ PARADES, Via di S. Pantaleo 66, 00186 Roma, Italy. {villa, balluchi, alberto}@parades.rm.cnr.it

† Cadence Berkeley Labs, 2001 Addison St., Third Floor, Berkeley, CA 94704, USA. howard@cadence.com

♯ Dept. of Chemical Eng., Univ. of Dortmund, Dortmund, Germany, Joerg.Preussig@ast.chemietechnik.uni-dortmund.de. Supported by DFG Kondisk-Projekt Ko 1430/3.

‡ Dept. of Electrical Eng. and Computer Sciences, Univ. of California, Berkeley, CA 94720, USA. alberto@eecs.berkeley.edu

¶ Cadence European Labs, Via di S. Pantaleo 66, 00186 Roma, Italy. watanabe@cadence.com

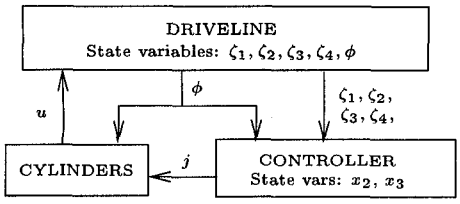

Figure 1: System of engine and controller.

controller, connected as shown in Figure 1. The car's driveline is described by a fifth-order linear system

$$\begin{bmatrix} \dot{\zeta} \\ \dot{\phi} \end{bmatrix} = A \begin{bmatrix} \zeta \\ \phi \end{bmatrix} + \boldsymbol{b}u \qquad (1)$$

where $\zeta = [\zeta_1, \zeta_2, \zeta_3, \zeta_4]^T \in \mathbb{R}^4$ and $\phi \in \mathbb{R}$. The state variable ζ_1 represents the engine block angle (in radians), ζ_2 represents the wheel revolution speed (in radians per second), ζ_3 represents the axle torsion angle (in radians), ζ_4 models the crankshaft revolution speed (in rotations per minute), and ϕ models the crankshaft angle (in degrees). The following parameters have been obtained by system identification of a car engine:

$$A = \begin{bmatrix} -34.7 & -6.345\ 10^{-2} & 12.62 & 8.112\ 10^{-4} & 0 \\ 19.21 & -0.6113 & 118.782 & 7.771\ 10^{-3} & 0 \\ 30.39 & -0.9444 & -11.05 & 1.229\ 10^{-2} & 0 \\ -8294 & 262.41 & -52209 & -3.843 & 0 \\ 0 & 0 & 0 & 6 & 0 \end{bmatrix}$$

and $\boldsymbol{b}^T = [1.52\ 10^{-3}, 8.41\ 10^{-3}, -1.331\ 10^{-3}, 47.83, 0]$. The engine has four cylinders, each of which cycles in lockstep through four phases in the following order: intake (I), compression (C), expansion (E), and exhaust (H). When fuel is injected, it enters the cylinder that is in its intake phase. The controller must make its decision on injection (modeled by the binary output variable j) at the beginning of the preceding exhaust phase. If fuel is injected into a cylinder, the cylinder produces torque in its next expansion phase. Thus the engine driveline does not react to a control decision until three phases later.

In previous work, a hybrid control strategy was designed, based on a switching-mode controller for a continuous abstraction that assumed no time delay in the effect of control decisions [2]. The idealized switching controller satisfies certain optimality and convergence properties. The hybrid control scheme mimics the idealized switching controller by making predictions of the state three phases in the future, and basing its decision on whether torque generation at the predicted state would bring the state closer to the switching surface or not. Performance and convergence properties, among others, were tested for the hybrid controller in a mixed Xmath/Ptolemy simulation environment [2]. It was also proven analytically that for a non-conservative approximation of the plant, for which the hybrid controller makes perfect predictions, the system state remains close to the switching surface during the final stages of being driven to the origin [2].

Here, we provide a formal verification that the hybrid controller, with the real plant model, satisfies two properties: convergence toward the origin along a portion of the switching surface, and quantifiably bounded acceleration. Our analysis indicates how well the hybrid controller behaves in light of its discrete decision points and its prediction errors. We consider contiguous sets of initial states (as opposed to the isolated initial points in space that simulation starts from). Our framework could also handle worst-case scenarios for errors in the identification matrix for the behavior of the engine, although this is not done here.

The next section contains a review of our modeling formalism—hybrid automata—and related analysis techniques. Hybrid automata for the engine cutoff system appear in Section 3. Section 4 describes the formal verification of the sliding action of the controller, for certain initial conditions. Analysis results are given. Finally, concluding remarks appear in Section 5.

2 Hybrid Automata

We review the basic notions of hybrid automata [1, 5].

2.1 Syntax

The components of a *hybrid automaton* $A = (X, V, E, \mathit{flow}, \mathit{inv}, \mathit{init}, \mathit{jump}, \Sigma, \mathit{syn})$ are:

Variables: A finite ordered set $X = \{x_1, \ldots, x_n\}$ of real-valued variables.

Control modes: A finite set V of control modes.

Control switches: A finite multiset E of control switches. A control switch (v_1, v_2) is a directed edge between a source mode v_1 and a target mode v_2.

Flow conditions: A labeling function *flow* that assigns a flow condition to each control mode. Flow conditions are predicates over the variables $X \cup \dot{X}$, where $\dot{X} = \{\dot{x}_1, .., \dot{x}_n\}$. The dotted variable $\dot{x}_i$ represents the first time derivative of the variable x_i. While the control of A is in mode v, the variables may evolve along any continuous piecewise-differentiable curve such that the flow condition $\mathit{flow}(v)$ is satisfied whenever the first derivatives exist.

Invariant conditions: A labeling function *inv* that assigns an invariant condition to each control mode. While the control of A is in mode v, the variables must satisfy the invariant condition $\mathit{inv}(v)$.

Initial conditions: A labeling function *init* that assigns an initial condition to each control mode. The control of A may start in mode v when $\mathit{init}(v)$ is true. In the graphical representation of automata, initial conditions appear as labels on incoming arrows without source modes, and initial conditions of the form *false* are not depicted.

Jump conditions: A labeling function *jump* that assigns a jump condition to each control switch. The jump condition is a predicate over the variables $X \cup X'$, where $X' = \{x'_1, ..., x'_n\}$. The unprimed symbol x refers to the value of the variable before the control switch, and the primed symbol x' refers to its

value after the switch. A jump condition relates the values of the variables before a control switch to the values after the control switch.

Events: A finite set Σ of events, and a labeling function *syn* that assigns an event to each control switch. Events allow the synchronization of jumps between concurrent hybrid automata.

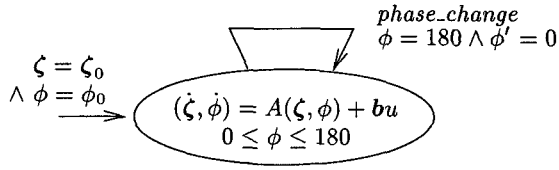

Figure 2: Hybrid automaton of driveline.

2.2 States and trajectories

A *state* of the hybrid automaton is a pair consisting of a control mode v and a vector $\boldsymbol{x}$ that represents a value for each variable, such that $\boldsymbol{x}$ satisfies the invariant for v. The state $(v, \boldsymbol{x})$ is *initial* if $\boldsymbol{x}$ satisfies the initial condition for v. Let $(v, \boldsymbol{x})$ and $(v', \boldsymbol{x}')$ be two states. Then $(v', \boldsymbol{x}')$ is a *jump successor* of $(v, \boldsymbol{x})$ if there is a control switch with source mode v and target mode v' such that the jump condition is satisfied. The state $(v', \boldsymbol{x}')$ is a *flow successor* of $(v, \boldsymbol{x})$ if $v = v'$ and there is a nonnegative real d and a continuous piecewise-differentiable function with domain $[0, d]$ leading from $\boldsymbol{x}$ to $\boldsymbol{x}'$, such that for all times $t \in [0, d]$ the invariant condition and flow condition are satisfied. A *trajectory* is a finite sequence of states such that (1) the first state of the sequence is an initial state, and (2) for each pair of consecutive states, the second is either a jump successor or a flow successor of the first. A state is *reachable* if it is the last state of some trajectory.

2.3 Parallel composition

Most systems involve several interacting components, each of which is modeled by a hybrid automaton. The system consists of the parallel composition of the component automata, which interact through both shared variables and synchronization via common event labels. The control modes of the parallel composition of two automata consist of the crossproduct of the components' modes. Control switches labeled with events that occur in the alphabet of both automata must occur simultaneously in both automata if at all. If the label of a control switch in one automaton does not appear in the alphabet of the other, then it may occur independently. Flow, invariant, initial, and jump conditions of product modes and switches are the conjunction of their counterparts for each component.

2.4 Verification

Our automated verification method relies on computing iterative fixpoints to determine the set of states that satisfy a certain property [1]. For instance, a safety requirement asserts that nothing bad will happen. These requirements can be specified via a set of "unsafe" states. The system satisfies the safety requirement iff no reachable states are unsafe. Verifying a safety requirement amounts to computing the set of reachable states. This computation can be performed by iterating a *Post* operator that returns for a given set of states the set of states that are either flow or jump successors.

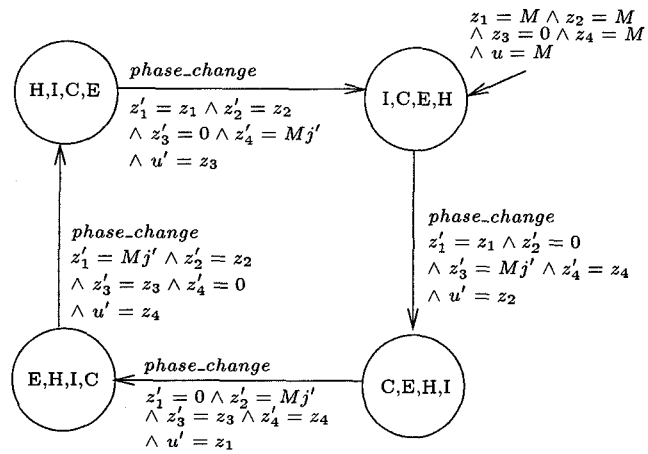

Figure 3: Hybrid automaton of cylinders.

2.5 Linear hybrid automata

In general, computing the *Post* operator exactly is not possible. However, it is computationally feasible for *linear hybrid automata*, in which the continuous dynamics are constant polyhedral differential inclusions, and the initial, invariant, and jump conditions are linear. Here the set of reachable states is linearly definable and effectively computable [1]. Termination of reachability analysis, however, is not guaranteed. The tool HyTech runs semi-decision procedures to determine whether a collection of processes modeled as linear hybrid automata satisfies its requirement specification [5]. It has been applied primarily to control-based applications that have nontrivial discrete dynamics combined with simple approximations of the continuous dynamics. However, for this engine-cutoff problem, a reasonably accurate representation of the continuous dynamics needs to be retained to faithfully model the system. The standard version of the tool encountered difficulties with internal arithmetic overflow in the manipulation of linear expressions over rational values. To overcome these problems, we use a conservatively approximate version of the tool that further restricts the input automata to be *rectangular*, i.e., all predicates define bounded rectangles in R^n [7].

3 Hybrid Automata for Engine Control

We provide a formal model for each of the three modules in the cutoff system—the cylinders, the engine driveline, and the controller—and for their interaction. The automata are depicted in Figures 2-4. They com-

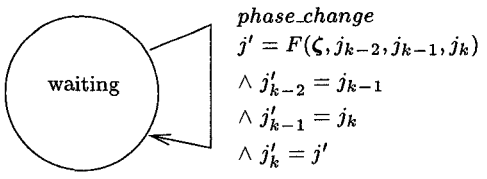

Figure 4: Hybrid automaton of controller.

municate via the shared variables ζ_1, ζ_2, ζ_3, ζ_4, u, and j, and the event label *phase_change*. The hybrid automaton of the driveline, shown in Figure 2, describes the dynamics of $\boldsymbol{\zeta}$ and ϕ. To simplify our analysis, we restrict ϕ, with no loss of generality, to the range $[0, 180]$. We thus model only the crankshaft angle offset, rather than the unbounded crankshaft angle itself. The hybrid automaton of the cylinders is shown in Figure 3. The cylinders are all in different phases. They change phase in lockstep, whenever the crankshaft angle reaches a multiple of 180 degrees. Thus the four cylinders in composition have only four possible configurations. At each phase change, the new value of j is read from the controller and stored in one of the four state variables z_1-z_4, which keep track of the last three previous choices of j. Three transitions later, the value of j multiplied by M is output as the control action to the driveline. The variables z_1, z_2, z_3, z_4, and j may be viewed as discrete variables with flow 0.

The controller automaton appears in Figure 4. It sets the value of j at each phase change, with the function F modeling the decision to inject fuel or not. The function F is defined over a transformed state space (over the variables x_1, x_2, x_3, and x_4) that helps isolate the fundamental modes related to acceleration oscillations. The variables $\boldsymbol{x}$ and $\boldsymbol{\zeta}$ are related via the transformation $\boldsymbol{x} = T\boldsymbol{\zeta}$, where T is

$$\begin{bmatrix} 4.146 & 6.544\ 10^{1} & 4.031 & 1.49\ 10^{-1} \\ 1.347\ 10^{3} & -4.116\ 10^{1} & 1.436\ 10^{3} & 5.48\ 10^{-1} \\ 1.536\ 10^{2} & 5.479\ 10^{1} & 1.072\ 10^{3} & -7.053\ 10^{-1} \\ 5.799\ 10^{2} & -2.8 & -1.558\ 10^{2} & 3.7\ 10^{-2} \end{bmatrix}.$$

The transformed state satisfies dynamics of the form

$$\begin{bmatrix} \dot{\boldsymbol{x}} \\ \dot{\phi} \end{bmatrix} = \hat{A} \begin{bmatrix} \boldsymbol{x} \\ \phi \end{bmatrix} + \hat{\boldsymbol{b}}\, u \qquad (2)$$

where

$$\hat{A} = \begin{bmatrix} -0.075719 & 0 & 0 & 0 & 0 \\ 0 & -3.3216 & -25.736 & 0 & 0 \\ 0 & 25.736 & -3.3216 & 0 & 0 \\ 0 & 0 & 0 & -43.491 & 0 \\ 5.99949 & 3.14631 & -5.10887 & -5.99997 & 0 \end{bmatrix}$$

and $\hat{\boldsymbol{b}}^T = [7.0748, 26.3824, -34.9729, 2.8608, 0]$. Powertrain oscillations are due to the pair of complex conjugate poles, which are related to the x_2 and x_3 components. Thus we concentrate on the x_2-x_3 subspace, where encirclements of the origin correspond to oscillations. Motion in the x_1 and x_4 coordinates is monotonic and does not produce any undesirable oscillating phenomena.

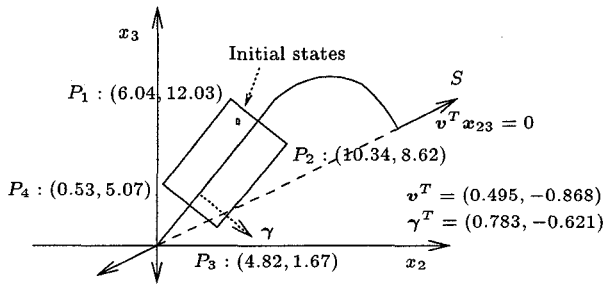

Figure 5: Region of interest for linearization.

The x_2-x_3 state, three phases into the future, is approximated on the assumption that each phase has fixed duration $T/2 = 0.00642271$. Let $\tilde{A}$ (respectively $\tilde{\boldsymbol{b}}$) denote the restriction of $\hat{A}$ (respectively $\hat{\boldsymbol{b}}$) to the x_2 and x_3 components. Integrating the dynamics for fixed input signal u, we obtain the approximation for one phase of $\hat{\boldsymbol{x}} = A'\boldsymbol{x} + \boldsymbol{b}'u$, where $A' = e^{\tilde{A}T/2}$, and $\boldsymbol{b}' = (e^{\tilde{A}T/2} - I)\tilde{\boldsymbol{b}}$. The assumption of a constant phase duration leads to error in the approximation, but it enables efficient implementation of the prediction, since a fixed A' and $\boldsymbol{b}'$ may be used. One may compute $\hat{\boldsymbol{x}}_0$ (the estimate for three future phases according to the stored control decisions, followed by a phase of free evolution) and $\hat{\boldsymbol{x}}_1$ (corresponding to the three stored control decisions, followed by a phase of forced evolution). The function F returns 0 if $|\sigma(\hat{\boldsymbol{x}}_0)| \leq |\sigma(\hat{\boldsymbol{x}}_1)|$, and 1 otherwise. The auxiliary function σ returns the (scaled) distance with sign of a state from a switching surface S, depicted in Figure 5, and formally defined in [2]. Intuitively, the line $\boldsymbol{v}^T\boldsymbol{x} = 0$, supporting in part the switching surface S away from the origin, is such that applying a bang-bang control $u = 0$ or $u = M$ with switchings on $\boldsymbol{v}^T\boldsymbol{x} = 0$ drives the state towards the origin following a trajectory along which the norm of the state decreases the most. Hence, oscillations are minimized. Once the state is close to the origin, the amount of torque available is large enough to enforce a sliding motion along a segment straight to the origin that completely cancels the oscillating behavior.

4 Verifying the Cutoff Controller

Our formal verification establishes a bound of 2.33 on the chattering amplitude (i.e., the maximal distance from the sliding line $\boldsymbol{\gamma}^T\boldsymbol{x} = 0$) of the sliding action due to the hybrid controller. From this we can infer bounds on the acceleration during the final stages of convergence. It has been proven analytically in [2] that under reasonable initial conditions, and the (nonconservative) assumption that the powertrain's phase durations were actually the predicted length $T/2$, the hybrid control strategy would maintain the state within 1.37 units of the switching line. Thus the hybrid control strategy is in some sense close to that of the idealized strategy. Note that the state may drift further than 1.37 units from the switching line since the variability

in the length of the phases (due to the dependence of $\dot{\phi}$ on the full state in $\mathbb{R}^4$) will result in inaccurate approximations. This inaccuracy can lead to incorrect predictions of whether injection will bring the state closer to the switching line or not. Therefore wrong decisions about injection can also result.

We also formally verify that the hybrid control strategy coupled with the real plant causes convergence toward the origin from initial states close to the sliding segment. Note that bounding the chattering amplitude does not of itself guarantee that the hybrid strategy inherits the convergence properties of the optimal sliding strategy [4]. Closeness to the switching line does not imply convergence: e.g., the state may repeatedly cross the switching line without progressing toward the origin. We prove that for a given initial condition—close to the sliding segment, but away from the origin—the state is close to the origin within a bounded number of phases. We take into account the variable phase durations by computing trajectories over a conservative range of durations, rather than just the predicted one (which is not conservative) or the actual one (which is harder to analyze). In particular, we consider only the perpendicular strip $P_1P_2P_3P_4$, depicted in Figure 5, which surrounds the switching segment. The outer bounds of the strip (the segments P_1P_4 and P_2P_3) are set at a distance twice that of the theoretical bounds proven for the non-conservative model, i.e., at 2.74 units. Our initial condition is $x_1 = 4672$, $x_2 \in [6.6, 6.9]$, $x_3 \in [9.9, 10.4]$, and $x_4 = 33$. We are interested in two problems: namely, (*Ampl*), find a bound on the distance from the state to the switching line, and (*Conv*), prove that the state leaves the strip via the P_3P_4 boundary within 30 phases. We prove *Conv* with the following three steps. First, we approximate the system dynamics using conservative piecewise-constant polyhedral differential inclusions. Second, from this approximation, we derive a conservative rectangular approximation. Third, we evaluate the rectangular approximation with HyTech. To prove *Ampl*, we repeat these steps, using the fact that *Conv* holds.

4.1 Conservative approximation

To prove *Conv* above, we need to consider only trajectories of duration up to 30 phases long. We show that all such trajectories leave the strip. To do so, we create a conservative approximation that includes all these trajectories, and possibly more. We then analyze this model using a form of reachability analysis, in which we iteratively compute overapproximations of the sets of states that are reachable in i phases via a trajectory yet to leave the strip.

The driveline dynamics depend on the state variables ζ, and thus do not fit HyTech's restriction to piecewise-constant differential inclusions. Some approximation is required. Rather than using many polyhedral differential inclusions to track the flow of the driveline, we instead model the effect of the flow over the duration of an entire phase. We view the system as a discrete-time system, where the time steps correspond not to integer times, but to the occurrence of phase changes. We model the states at the phase boundaries only, and express the relationship between states at the beginning of one phase to those at the beginning of the next phase via jumps. The jumps are then approximated using linear constraints as follows. First, for an arbitrary state $\boldsymbol{x}$, we compute a set $K(\boldsymbol{x})$ that contains all possible successor states of $\boldsymbol{x}$ after one phase (within the first 30 phases). Second, we define linear constraints $L(\boldsymbol{x})$ that enclose the set $K(\boldsymbol{x})$. Thus we capture the evolution of the driveline dynamics via jumps of the form $\boldsymbol{x} := L(\boldsymbol{x})$.

Bounds on the phase duration

The dynamics of the driveline depend on the input variable u. When $u = M$, the system is in *forced* mode, corresponding to torque being generated by the cylinder in its expansion phase; when $u = 0$, the system is in *free* mode, corresponding to no torque being generated. Given a state $\boldsymbol{x}$ and an evolution mode, we need a set $K(\boldsymbol{x})$ that contains all successor states reached from $\boldsymbol{x}$ after one phase of evolution of legal duration in the given mode. These states are of the form $\boldsymbol{x}(t) = e^{\tilde{A}t}\boldsymbol{x}(0) + \tilde{A}^{-1}(e^{\tilde{A}t} - I)\tilde{\boldsymbol{b}}u$, where t is a valid phase duration. To obtain $K(\boldsymbol{x})$, it suffices to determine an overapproximation of the range of legal phase durations. We do so by bounding $\dot{\phi} = \boldsymbol{c}_\phi^T\boldsymbol{x}$, where $\boldsymbol{c}_\phi^T = [5.99949, 3.14631, -5.10887, -5.99997]$. The extremal values of the sum of the x_2 and x_3 terms evaluated over the strip are a minimum of -42.438 at P_1 and a maximum of 6.6508 at P_3. We are given that the initial value of x_1 is 4672 and the initial value of x_4 is 33. Both variables are subject to negative exponential dynamics. We show how a lower bound of 4600 for x_1 can be justified. Assume that x_1 is no less than 4600. Then $\dot{\phi}$ must be no less than $5.99949 * 4600 - 240.44$, which implies that the maximal duration for a phase is no more than $0.00657962 > 180/27357.214$. This gives an upper bound on the time for 30 phases, which gives a lower bound of 4602.7 for the exponential decay of x_1. Since this value is greater than our assumed 4600, we can safely conclude that over 30 phases, the value of x_1 is always above 4600. Using the trivial lower bound of 0 for x_4, we then obtain a range of $[27357, 28036]$ for $\dot{\phi}$, which leads to a range of $[T_{min} = 0.0064203, T_{max} = 0.0065797]$ for the phase duration.

Linear bounds on successor states

We now compute linear bounds for x_2 and x_3 under free evolution. The analysis for forced evolution is similar, and omitted for space considerations. For $\dot{\boldsymbol{x}} = A\boldsymbol{x}$, it follows that $\boldsymbol{x}(t) = e^{At}\boldsymbol{x}(0)$. Let $a = -3.3216$ and $b = 25.736$. For our system, we have

$$A = \begin{bmatrix} a & -b \\ b & a \end{bmatrix} \text{ and } e^{At} = e^{at}\begin{bmatrix} \cos(bt) & -\sin(bt) \\ \sin(bt) & \cos(bt) \end{bmatrix}.$$

To minimize and maximize x_2 and x_3, we need to consider expressions of the form $e^{at}(\alpha\cos(bt) + \beta\sin(bt))$, whose derivative $e^{at}[\sin(bt)(-\alpha b + \beta a) + \cos(bt)(\alpha a + \beta b)]$ is zero when $bt = arctan[(\alpha a + \beta b)/(\alpha b - \beta a)]$. Note that from the above range of phase durations,

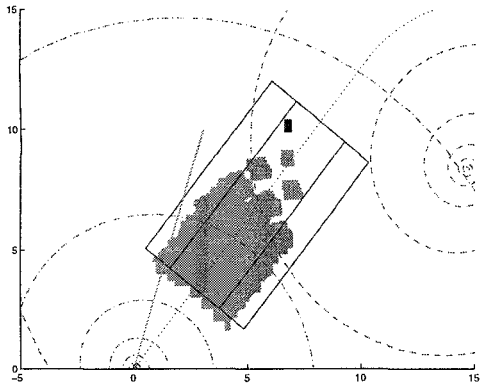

Figure 6: Superset of reachable states at phase changes.

we have that $\theta = bt$ lies in $[\theta_{min} = 0.16523, \theta_{max} = 0.16933]$.

For x_2' (the next state of x_2 at the phase change), we have $\alpha = x_2$ and $\beta = -x_3$. The fact that the derivative is negative for $\theta \in [\theta_{min}, \theta_{max}]$ implies that x_2' is minimized at T_{max} and maximized at T_{min}, thereby yielding linear (in x_2 and x_3) upper and lower bounds on x_2'.

For x_3', we have $\alpha = x_3$ and $\beta = x_2$. Examination of the derivative shows that it can be 0 in the θ range if $3.2594x_2 \leq x_3 \leq 3.3083x_2$. Let E denote the set of states satisfying the above inequalities. Within the strip, above E, the value of x_3' is maximized at T_{max} and minimized at T_{min} (by examination of the sign of the derivative); below E, it is maximized at T_{min} and minimized at T_{max}. Within E, we place conservative upper and lower bounds on $e^{at}f(t)$ (where $f(t) = x_3\cos(bt) + x_2\sin(bt)$), by independently minimizing and maximizing e^{at} and $f(t)$. It is easily seen that the extremal values of $f(t)$ are found at the endpoints T_{min} and T_{max}.

We derive a rectangular automaton using the linear bounds computed above. We omit details for lack of space. This further abstraction is necessary since the currently available tool is not numerically robust over automata with complex continuous dynamics.

4.2 Analysis results

First, HYTECH is used to verify property *Conv*. We cut the x_2-x_3 state space into 0.5×0.5 partition blocks in order to increase the accuracy of the rectangular approximations. Within each partition block, we take the convex hull of the reachable states after each phase, in order to reduce state-space explosion. The computation takes 70 seconds on a Sparc Ultra.

To prove *Ampl*, we perform reachability analysis to compute all states reachable within the strip. Because *Conv* holds, the phase durations lie in the interval $[T_{min}, T_{max}]$ as derived above. In Figure 6, the initial states are shown in black, and the gray region indicates a superset of the states reachable at phase changes. The dashed (resp. dash-dotted) line shows a sample trajectory under forced (resp. free) evolution. We verify a chattering amplitude of 2.33. The inner lines indicate the theoretical bounds for the hybrid controller with hypothetical perfect predictions. Thus our analysis bounds the additional distance (0.96 units) from the central line that is due to variable phase durations and possibly incorrect injection decisions resulting from stray state predictions. This computation takes 37 seconds on a Sparc Ultra.

5 Conclusions

While HYTECH was sufficient for this verification task, we had to battle against a number of shortcomings which must be addressed before hybrid systems verification becomes practical. We could not naturally express the system dynamics in the tool's input language, but had to approximate using constant polyhedral differential inclusions. These approximations may be made arbitrarily accurate, but only at a cost of severe state-explosion. Ideally, the methodology should allow a much wider class of continuous dynamics to be modeled directly. The tool could then use on-the-fly, conservative approximations to compute the reachable state sets. State sets need to be represented more efficiently, perhaps by exploiting information that allows parts of the state space to be stored abstractly.

References

[1] R. Alur, C. Courcoubetis, N. Halbwachs, T. Henzinger, P.-H. Ho, X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine. The algorithmic analysis of hybrid systems. *Theoretical Computer Science*, 138:3–34, 1995.

[2] A. Balluchi, M. D. Benedetto, C. Pinello, C. Rossi, and A. Sangiovanni-Vincentelli. Cut-off in engine control: a hybrid system approach. In *Proc. 36th IEEE Conference on Decision and Control*, 1997.

[3] A. Balluchi, M. D. Benedetto, C. Pinello, C. Rossi, and A. Sangiovanni-Vincentelli. Hybrid control for automotive engine management: the cut-off case. In *HSCC 98: Hybrid Systems—Computation and Control*, Lecture Notes in Computer Science 1386. Springer-Verlag, 1998.

[4] R. de Carlo, S. Zak, and G. Matthews. Variable structure control of nonlinear multivariable systems: a tutorial. *Proceedings of the IEEE*, 76:212–232, 1988.

[5] T. Henzinger, P.-H. Ho, and H. Wong-Toi. Algorithmic analysis of nonlinear hybrid systems. *IEEE Trans. on Automatic Control*, 43(4):540–554, 1998.

[6] T. A. Henzinger, P.-H. Ho, and H. Wong-Toi. HYTECH: A model checker for hybrid systems. *Software Tools for Technology Transfer*, 1(1):110–122, 1997.

[7] J. Preußig, S. Kowalewski, H. Wong-Toi, and T. A. Henzinger. An algorithm for the approximative analysis of rectangular automata. In *FTRTFT 98: Formal Techniques for Real-time and Fault-tolerant Systems*, Lecture Notes in Computer Science. Springer-Verlag, 1998.