Compositionally Analyzing a Proportional-Integral Controller Family

Ashish Tiwari

***Abstract*— We define *always eventually region stability* and then formulate the *absolute always eventually region stability problem* as the problem of finding a class of plants that a generic proportional-integral (PI) controller can always eventually stabilize. We use real quantifier elimination methods to solve the absolute always eventually region stability problem. The class of plants found in our solution includes nonlinear and switched plant models. Our analysis reveals that any PI controller that satisfies two assumptions of the form:**
(a) the proportional gain K_p and the integral gain K_i satisfy an inequality, and
(b) the integral of the error term in the controller is saturated, then such a PI controller can be used to establish always eventually region stability for any plant that suitably responds to the control input. Such a result is useful for compositionally verifying a system consisting of a plant and a controller.

I. Introduction

Any complex cyber-physical system will necessarily contain one or more controllers. One of the most commonly used controller is a proportional-integral (PI) or a proportional-integral-derivative (PID) controller. In this paper, we use formal verification techniques to analyze an open PI controller shown in Figure 1.

Our investigation is motivated by two reasons. First, there is recent push in exploring the idea of building complex cyber-physical systems compositionally. The hope is that there would be a library of components from which one could pick components and connect them to build complex systems. The ability to analyze systems compositionally is crucial for making progress on this vision. Since controllers will necessarily be part of any such library, we need verification techniques that can analyze individual components, such as a controller. The second reason comes from the need to build scalable techniques for formally verifying complex systems. One way of achieving scalability is to perform verification compositionally. Hence, we need approaches for analyzing each component separately in such a way that the analysis results can be used later to verify the composed system.

The current practice in verification of complex cyber-physical systems is based on performing extensive (numerical) simulation and testing. However, simulation-based methods are incomplete – how do we know that the system has been tested enough? In the past few years, there has been an extensive push for extending formal verification approaches to also verify physical and cyber-physical systems. Broadly speaking, these techniques can be classified as follows:

This work funded in part by DARPA META under contract FA8650-10-C-7078 and by NSF under grants CSR-0917398 and SHF:CSR 1017483.

A. Tiwari is with the Computer Science Laboratory, SRI International, Menlo Park, CA 94025 ashish.tiwari@sri.com

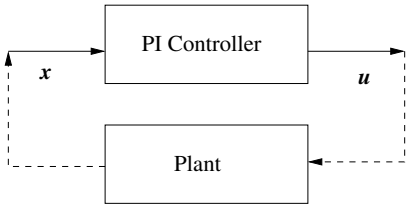


Fig. 1. Block diagram of a PI Controller and a plant system.

1) *reach-set methods* that compute the set of all reachable states of the system, either exactly [20], or approximately [7], [35], [19], [12], [17], [15]
2) *abstraction-based methods* that first abstract the system and then analyze the abstraction [33], [1], [8]
3) *certificate-based methods* that directly search for certificates of correctness (such as inductive invariants and Lyapunov functions) of systems [30], [25], [27], [23], [16], [31], [4], [24]

While all these techniques have had some success, the certificate-based methods are turning out to be particularly effective in proving deep properties of complex systems. Certificate-based methods work by fixing a template for the "certificate of correctness", and casting the verification problem as a problem of finding an appropriate instantiation of the template. This search is accomplished using numeric or symbolic solvers that reason about arithmetic constraints in the theory of reals. For example, for verifying stability, the user can provide a template for the Lyapunov function (say a quadratic Lyapunov function) and the solver can find a concrete quadratic function that proves stability of the given system. Most commonly used solvers include ones for sums-of-squares (SOS) programming [26] and real quantifier elimination. Numeric approaches, such as SOS programming, have two limitations: first, they may give incorrect answers, and second, they have their limitations when solving Boolean combination of constraints. We use symbolic solvers in the form of real quantifier elimination in this paper.

Motivated by the need to analyze systems compositionally, the question we answer in this paper is the following: what is the class of plant models that can be stabilized by a given PI controller. This problem is related to the so-called *absolute stability problem* [37], [22], but there are important differences that we will discuss later. We focus on systems of the form shown in Figure 1. We assume the PI controller is being used to drive the plant to a given setpoint. We wish to prove that the error – difference between the setpoint

and the observed state – always eventually falls below a given constant. We formalize this notion as *always eventually region stability*. The problem of finding the class of plant models can be formulated as the *absolute always eventually region stability problem*. Our approach for finding the class of plant models is based on using templates to describe the dynamics of all the possible plants, and then deriving conditions on the parameters in the plant model. Specifically, we find that PI controllers can guarantee always eventually region stability for nonlinear and switched plant models.

The outline of the paper is as follows. Section II describes the main technical result. Section III illustrates the main result using some concrete examples. The tools used for performing the real quantifier elimination are described in Section IV. Comparison to related work, especially to work on absolute stability, is presented in Section V.

II. PI Controller: Compositional Reasoning

A proportional-integral, or PI (respectively, proportional-integral-derivative or PID), controller is a generic controller that can be instantiated to give a feedback controller by fixing two (respectively, three) parameters. It is widely used in the design of industrial control systems.

Our goal is to formally verify a PI controller. A PI controller cannot be verified in isolation, and we need a model of the plant (process) that the PI controller is controlling to formally state and prove its safety and/or stability. Here, we assume that we do not have access to the plant model, and hence the goal is to verify the PI controller under *suitable generic assumptions* about the plant.

Consider a plant (process) that needs to be driven to some desired state. Suppose the state of the plant is given by the values of n real-valued variables. The state of the plant is, hence, a point in $\mathbb{R}^n$. Assume that we need to drive the plant to a state such that a given variable, say X, takes a particular value, say $x_{sp} \in \mathbb{R}$ (*sp* refers to the desired *setpoint*). If $x \in \mathbb{R}$ is the (estimate of the) current value of the variable X in the plant, then the *error*, $\texttt{err}$, is given by

$$\texttt{err} \quad = \quad x - x_{sp} \tag{1}$$

Let $\texttt{interr}$ denote the integral of this error, but saturated to stay within some range, say $[-1, 1]$; that is,

$$\frac{d\texttt{interr}}{dt} \quad = \quad \begin{cases} \texttt{err} & \text{if } \texttt{interr}^2 = 1 \ \wedge \\ & \quad \texttt{err} * \texttt{interr} < 0 \\ \texttt{err} & \text{if } \texttt{interr}^2 < 1 \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

The symbol $\wedge$ denotes conjunction (and). Note that the above dynamics for $\texttt{interr}$ guarantees that it stays in the range $[-1, 1]$. Saturation blocks that restrict a signal to a specific range, such as the one modeled above, are commonly used in engineering designs. A PI controller computes the control input for the plant as a weighted sum of the $\texttt{err}$ and $\texttt{interr}$.

$$u \quad = \quad K_p * \texttt{err} + K_i * \texttt{interr} \tag{3}$$

where the constants K_p and K_i are chosen by the control designer. Let us say we are given $K_p = 500$ and $K_i = 10$. We will later analyze the controller for symbolic K_p and K_i.

The description of the PI controller is contained in Equation 1, Equation 2, and Equation 3. However, to enable any analysis, we need to relate the control input u to the plant and the output x of the plant. In the spirit of *assume-guarantee* reasoning, we assume a very generic plant whose dynamics, when projected onto the variable X, can be abstracted to the following *nondeterministic* differential equation

$$\begin{array}{rcl} \dfrac{dx}{dt} & = & \beta - \alpha * u \\ \alpha & \in & [a, b] \\ \beta & \in & [a_1, b_1] \end{array} \tag{4}$$

where α and β can *vary* nondeterministically in the interval $[a, b]$ and $[a_1, b_1]$ respectively. In other words, $\frac{d\alpha}{dt}$ and $\frac{d\beta}{dt}$ are both unrestricted and can be anything as long as α, β remain in the specified range.

Note that the output x of the plant will, in general, depend on the internal state of the plant as well as the control input u. The dependence of x on the internal state of the plant is captured by β and the dependence of x on the control input is captured by α. That is why there are no restrictions on the dynamics of α and β since we want to allow *arbitrary* dynamics for α and β.

The closed loop control system is now described by the union of Equation (1), Equation (2), Equation (3), and Equation (4).

For such a closed loop system, we are interested here in a version of stability we call *always eventually region stability*.

Definition 1 (always eventually region stability): A system is *always eventually region stable (a.e.r.s)* with respect to a region $R \subseteq \mathbb{R}^n$ if, for any trajectory $\vec{s}(\vec{x_0}, t)$ starting from any initial state $\vec{x_0} \in \mathbb{R}^n$ of the system, it is the case that

$$\forall t \geq 0 : \forall \epsilon > 0 : \exists t' \geq t : \vec{s}(\vec{x_0}, t') \in R^\epsilon$$

where R^ϵ is the set of points that are at most ϵ distance away from R.

When the initial state $\vec{x_0}$ of the system can be arbitrary, then the above condition is equivalent to

$$\forall \vec{x_0} \in \mathbb{R}^n : \forall \epsilon > 0 : \exists t' \geq 0 : \vec{s}(\vec{x_0}, t') \in R^\epsilon$$

This is similar to global asymptotic stability, except that (a) we consider a region R here and not a point, and (b) we do not require that the trajectory remain inside R^ϵ at all times after t'. Such a weak notion of region stability is useful for two reasons: first, it naturally allows us to say that the error e always eventually falls below a given constant, say 1, and second, it helps in getting simpler criteria for the absolute always eventually region stability problem formalized below.

Definition 2 (Absolute a.e.r.s problem): Given an open system, such as the one specified in Equation (1), Equation (2), and Equation (3), the problem is to derive conditions involving only a, b, a_1, b_1 that will guarantee that the system is always eventually region stable with respect to a given

region, say $\texttt{err} \in [-1, 1]$, for every plant that satisfies Equation (4).

Solving the always eventually region stability problem

We use the *certificate-based verification* approach to solve the absolute a.e.r.s problem. The certificate here is a Lyapunov-like function. Specifically, we establish the existence of a *common* Lyapunov function that works for all possible plants that satisfy the (unknown) plant assumptions. We fix a form of the Lyapunov function, say,

$$V \quad = \quad c * \texttt{err}^2 + d * \texttt{interr}^2 \tag{5}$$

We have to simultaneously find the assumptions on the plant – namely, values a, b, a_1 and b_1 – and the certificate for the closed-loop system – namely, values c, d – such that a.e.r.s stability holds. We find these values by solving the exists-forall formula ϕ_0 in Equation 6 below. Before we present the formula, let us fix some definitions. Note that we have two modes in the closed-loop system based on whether `interr` is saturated or not.

1) In $\texttt{Mode}_1$, `interr` is not saturated. In this case, the dynamics are given by

$$\begin{aligned} d\texttt{interr}/dt &:= \texttt{err} \\ d\texttt{err}/dt &:= \beta - \alpha * (K_p * \texttt{err} + K_i * \texttt{interr}) \end{aligned}$$

with the mode invariant

$$\begin{aligned} \texttt{Inv}_1 \quad := \quad & (\texttt{interr}^2 \leq 1 \ \wedge \\ & (\texttt{interr}^2 < 1 \vee \texttt{err} * \texttt{interr} < 0) \ \wedge \\ & a \leq \alpha \leq b \ \wedge \ a_1 \leq \beta \leq b_1) \end{aligned}$$

2) In $\texttt{Mode}_1$, `interr` is saturated. In this case, the dynamics are given by

$$\begin{aligned} d\texttt{interr}/dt &:= 0 \\ d\texttt{err}/dt &:= \beta - \alpha * (K_p * \texttt{err} + K_i * \texttt{interr}) \end{aligned}$$

with the mode invariant

$$\begin{aligned} \texttt{Inv}_2 \quad := \quad & (\texttt{interr}^2 \geq 1 \ \wedge \ \texttt{err} * \texttt{interr} \geq 0 \ \wedge \\ & a \leq \alpha \leq b \ \wedge \ a_1 \leq \beta \leq b_1) \end{aligned}$$

Note that the mode invariants are determined by the saturation conditions of `interr` (see Equation (2)).

Let $\frac{dV}{dt}|_{\texttt{Mode}_i}$ denote the Lie derivative of V in $\texttt{Mode}_i$. It is obtained by symbolically differentiating the expression for V with respect to time in the two modes. Specifically, dV/dt in $\texttt{Mode}_i$ can be obtained by replacing $d\texttt{err}/dt$ and $d\texttt{interr}/dt$ in the following expression by their respective definitions in $\texttt{Mode}_i$:

$$\frac{dV}{dt} \quad := \quad 2 * c * \texttt{err} * \frac{d\texttt{err}}{dt} + 2 * d * \texttt{interr} * \frac{d\texttt{interr}}{dt}$$

Now, we are ready to write the formula that encodes the absolute a.e.r.s problem. The formula below states that there exist some values for a, b, a_1, b_1, c, d (parameters of the nondeterministic plant) such that V is a Lyapunov function that proves a.e.r.s stability of the system.

$$\begin{aligned} \phi_0 \quad &:= \quad \exists a, b, a_1, b_1, c, d : \phi_1 \\ \phi_1 \quad &:= \quad \forall \texttt{err}, \texttt{interr}, \alpha, \beta : \phi_2 \\ \phi_2 \quad &:= \quad b > a \ \wedge \ b_1 > a_1 \ \wedge \ c \geq 0 \ \wedge \ d \geq 0 \ \wedge \ \phi_3 \\ \phi_3 \quad &:= \quad (\texttt{err}^2 \geq 1 \ \wedge \ \texttt{Inv}_1 \Rightarrow \frac{dV}{dt}|_{\texttt{Mode}_1} < 0) \ \wedge \\ & \qquad (\texttt{err}^2 \geq 1 \ \wedge \ \texttt{Inv}_2 \Rightarrow \frac{dV}{dt}|_{\texttt{Mode}_2} < 0) \end{aligned} \tag{6}$$

Here $\Rightarrow$ denotes logical implication. Since we are interested in finding the plant parameters, a, b, a_1, b_1, we will find the constraints on a, b, a_1, b_1 such that if some concrete values for a, b, a_1, b_1 satisfy those constraints, then those values will make the Formula $\exists c, d : \phi_1$ true. This is precisely what quantifier elimination achieves.

The correctness of our approach relies on the following proposition.

Proposition 1: Let $\psi_1(a, b, a_1, b_1)$ be a formula containing only the variables a, b, a_1, b_1 that is obtained as a result of eliminating quantifiers from Formula $\exists c, d : \phi_1$, where ϕ_1 is defined in Equation 6; that is,

$$\psi_1(a, b, a_1, b_1) \quad \leftrightarrow \quad \exists c, d : \phi_1$$

Then, ψ_1 is a solution for the absolute a.e.r.s problem.

Proof: (Sketch) Suppose a, b, a_1, b_1 are concrete values that satisfy ψ_1. We do a proof by contradiction, and assume a.e.r.s stability does not hold for this choice of a, b, a_1, b_1. Then, starting from some initial state and for some plant that satisfies Equation 4, the system trajectory never reaches $\texttt{err}^2 \leq 1$. This means, $\texttt{err}^2 > 1$ always. By ϕ_3, $dV/dt < 0$ always, and hence `err` gets arbitrarily close to 0, which contradicts that $\texttt{err}^2 > 1$ always. ■

Analysis Results

We use symbolic tools that perform real quantifier elimination to eliminate quantifiers from Formula ϕ_1 shown above in Equation 6. Details of the specific tools and techniques used can be found in Section IV. Here we present the results obtained by analyzing Formula ϕ_1 above.

We eliminate all the $\forall$ quantified variables from Formula ϕ_1. The result is a formula containing only the remaining variables a, b, a_1, b_1, c, d that is equivalent to Formula ϕ_1. There are automatic tools that perform quantifier elimination. The following formula, slightly rewritten to fit margins, is returned by the tool.

$$\begin{aligned} \psi_1 \ := \ & a_1 \leq b_1 \ \wedge \ 0 < a \leq b \ \wedge \\ & ((10 * c * a \geq d \ \wedge \ \{c * a_1, c * b_1\} \leq |490 * c * a + d|) \vee \\ & (10 * c * a < d \ \wedge \ \{c * a_1, c * b_1\} \leq |510 * c * a - d|)) \end{aligned}$$

Here $|x|$ denotes the absolute value of x and $\{a, b\} \leq x$ denotes $a \leq x \ \wedge \ b \leq x$. Formula ψ_1 gives conditions on the parameters in the plant model, along with conditions on the coefficients c, d of the Lyapunov function. If we existentially quantify c, d in the above formula, and ask the quantifier

elimination tool to eliminate c, d from $\exists c, d : c \geq 0 \ \wedge \ d \geq 0 \ \wedge \ \psi_1$, then the tool returns the formula ψ_0 shown below.

$$\psi_0 \ := \ b_1 - 500 * a \leq 0 \ \wedge \ a_1 + 500 * a \geq 0$$

Formula ψ_0 is now a solution of the absolute a.e.r.s problem.

Theorem 1: If there is a constant $a > 0$ such that the dynamics of a plant can be shown to satisfy the following assumptions,

$$\begin{array}{rcl} \frac{dx}{dt} & = & \beta - \alpha * u \\ \alpha & \in & [a, +\infty], a > 0 \\ \beta & \in & [-500 * a, 500 * a] \end{array} \quad (7)$$

then, when this plant is composed with a PI controller

$$\begin{array}{rcl} u & = & 500 * \texttt{err} + 10 * \texttt{interr} \\ \texttt{err} & = & x - x_{sp} \\ \frac{d\texttt{interr}}{dt} & = & \texttt{err}, \ \texttt{interr} \text{ saturates to } [-1, 1] \end{array}$$

the resulting system is always eventually region stable with respect to the region $\texttt{err}^2 \leq 1$.

Note that the plant can be nonlinear, and can even have switched dynamics, and still satisfy the conditions of Theorem 1. This is because α, β can change *dynamically* – they need not even be continuous – as long as they satisfy the conditions in the theorem above. Thus, Theorem 1 is applicable to a very large class of plants.

We note here that if we remove the saturation module from the PI controller, then we are unable to prove a.e.r.s stability for such a class of nondeterministic plant models. (that is, where $d\alpha/dt$ and $d\beta/dt$ are unrestricted as long as α, β remain inside the given bounds). Without the saturation module, we can, of course, prove stability for many *fixed* plant models (that is, where $d\alpha/dt = d\beta/dt = 0$), but assuming $d\alpha/dt = d\beta/dt = 0$ significantly narrows the class of plants and hence it is not very useful for performing compositional reasoning on systems.

Generic PI Controller

We can now generalize from a particular value of K_p and K_i and instead use symbolic values for these gains. We can now ask the question *for what values of K_p and K_i does the same solution of the absolute a.e.r.s problem work?*

We solve this problem by formulating it as a quantifier elimination problem. In fact, we can find the desired conditions on K_p and K_i by eliminating the quantified variables from the following Formula ϕ_4:

$$\begin{array}{rcl} \phi_4 & := & \exists a, c, d : a > 0 \ \wedge \ c \geq 0 \ \wedge \ d \geq 0 \ \wedge \ \phi_5 \\ \phi_5 & := & \forall \texttt{err}, \texttt{interr}, \alpha, \beta : \phi_3 \end{array} \quad (8)$$

where ϕ_3 is as defined in Equation 6 except for two differences: first, $\texttt{Inv}_1$ and $\texttt{Inv}_2$ are now given by

$$\begin{array}{rcl} \texttt{Inv}_1 & := & \texttt{Inv}_0 \ \wedge \ \texttt{interr}^2 \leq 1 \ \wedge \\ & & \quad (\texttt{interr}^2 = 1 \Rightarrow \texttt{err} * \texttt{interr} < 0) \\ \texttt{Inv}_2 & := & \texttt{Inv}_0 \ \wedge \ \texttt{interr}^2 = 1 \ \wedge \ \texttt{err} * \texttt{interr} > 0 \\ \texttt{Inv}_0 & := & (a < \alpha \ \wedge \ -500 * a < \beta \ \wedge \ \beta < 500 * a) \end{array}$$

That is, we are considering only the class of plants for which the concrete PI controller worked. As a consequence, the variables a_1, b_1, b were present in Formula ϕ_1 in Equation 6, but they are no longer present in Formula ϕ_4 in Equation 8. The second difference between Formula ϕ_1 in Equation 6 and Formula ϕ_4 in Equation 8 is that K_p and K_i were replaced by their fixed values (500 and 10 respectively) in the former, whereas variables K_p and K_i are still present in the latter.

We use real quantifier elimination to first eliminate the universal quantifiers from Formula ϕ_5 in Equation 8. We get the following equivalent formula ψ_5 on the remaining variables.

$$\psi_5 \ := \ K_p \geq K_i \ \wedge \ c * a * (K_p - 500) \geq |c * a * K_i - d|$$

Now we eliminate the existential variables from Formula ϕ_4. We replace ϕ_5 in Formula ϕ_4 by ψ_5, and then eliminate a, c, d using real quantifier elimination tools. We get the following Formula ψ_4 as the answer.

$$\psi_4 \ := \ K_p \geq 500 \ \wedge \ K_p \geq K_i \ \wedge \ K_p + K_i \geq 500$$

Thus, our solution for the absolute a.e.r.s problem works for any PI controller that satisfies this condition. Note that our concrete choice $K_p = 500$ and $K_i = 10$, that we had used to obtain results of the previous section, satisfies this condition.

Theorem 2: Suppose K_p, K_i satisfy Formula ψ_4, and suppose $a > 0$, $b = +\infty$, $a_1 = -500 * a$ and $b_1 = 500 * a$ in Equation 4. Then, the feedback control system defined by Equation 1, Equation 2, Equation 3, and Equation 4 always eventually reaches a state where $\texttt{err}^2 \leq 1$.

It is folklore in control theory that PI controllers can stabilize "almost any plant". The above result is one possible formulation of this fact, which we have derived using advanced quantifier elimination techniques.

III. Experiments

We present simulation plots in this section to illustrate the results above.

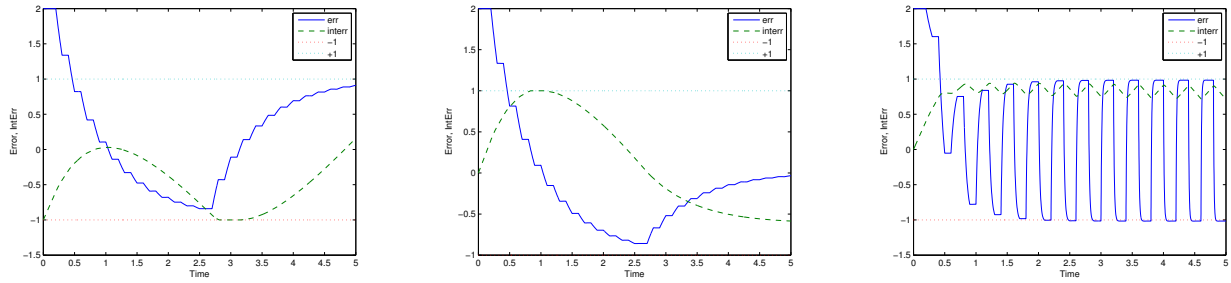
First consider a PI controller that has fixed parameters, namely, $K_p = 500$ and $K_i = 10$. We consider three different plant models. The first plant model switches between two different dynamics every 0.1 time units. We only give the value of the parameters α and β of the plant in Equation 4 here. Between time $t = 0$ to $t = 2.5$ units, the plant switches between the following modes.

$$\begin{array}{rcl} \texttt{Mode}_1 & : & \alpha = 10^{-5}, \beta = 5 * 10^{-3} \\ \texttt{Mode}_2 & : & \alpha = 5 * 10^{-3}, \beta = -2.5 \end{array}$$

We are assuming $d\alpha/dt = d\beta/dt = 0$ inside each mode. Between time $t = 2.5$ to $t = 5$ units, the plant switches between the following modes.

$$\begin{array}{rcl} \texttt{Mode}_3 & : & \alpha = 10^{-5}, \beta = -5 * 10^{-3} \\ \texttt{Mode}_4 & : & \alpha = 5 * 10^{-3}, \beta = 2.5 \end{array}$$

Note that the plant parameters satisfy the condition ψ_0, and hence Theorem 1 tells us that the error will always eventually



(a) (b) (c)

Fig. 2. Simulation of a closed loop control system where the plant is a switched system whose dynamics change every 0.1 time units.

approach the range $[-1, 1]$ for this case. Figure 2(a) presents a simulation run for this system when we start with $\texttt{err} = 2$ and $\texttt{interr} = -1$.

The second plant behaves like the first plant for the first 2.5 time units, and thereafter it switches between the following two modes after every 0.1 time units.

$$\begin{aligned} \texttt{Mode}_5 &: \alpha = 10^{-5}, \beta = -5 * 10^{-3} \\ \texttt{Mode}_6 &: \alpha = 5 * 10^{-3}, \beta = 5 * 10^{-3} \end{aligned}$$

A simulation of the same PI controller, when composed with this plant, is shown in Figure 2(b). Initially, $\texttt{err} = 2$ and $\texttt{interr} = 0$. Note again that the error err always eventually reaches the range $[-1, 1]$.

The third plant switches between the following two modes every 0.1 time units.

$$\begin{aligned} \texttt{Mode}_7 &: \alpha = 10^{-5}, \beta = 5 * 10^{-3} \\ \texttt{Mode}_8 &: \alpha = 5 * 10^{-3} * i, \beta = -2.5 * i * (-1)^{\texttt{floor}(i/2)} \end{aligned}$$

where i is a natural number that increments by one every 0.1 time units. A simulation of the behavior of the same PI controller with this plant is shown in Figure 2(c).

The second set of simulation plots consider the case when the PI controller parameters are not fixed, but they vary with time. Specifically, K_p is still fixed to 500, but K_i is set to $100 * t$, where t is the time. Since $t \in [0, 5]$ in our simulation, we always satisfy the condition that $K_i \leq K_p$. We use the same three plant models. Simulations with the varying parameter PI controller are shown in Figure 3. Note that the error err overshoots the $[-1, 1]$ range. But this is not an inconsistency for two reasons: first, overshoots are allowed in the definition of always eventually region stability, as long as the systems always returns to the desired region. Hence, overshooting is not a violation of always eventually region stability. Second, Theorem 2 says that any fixed PI controller can guarantee that error eventually gets inside the $[-1, 1]$ range, but no claim is made when dynamically using different PI controllers (from the set of controllers that satisfy constraint ψ_4). In particular, for a fixed controller, Theorem 2 says that there is a common Lyapunov function that works for all plants in the specified set of plants, but that Lyapunov function itself depends on K_p, K_i, and hence it changes as these parameters change.

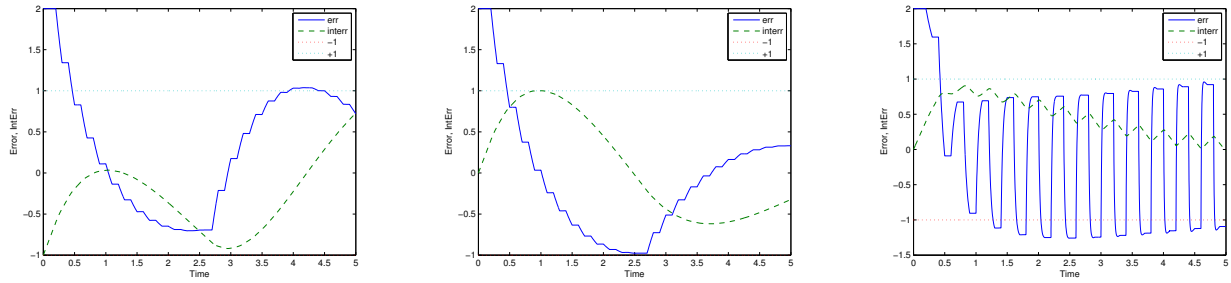
IV. Quantifier Elimination Tools

Quantifier elimination refers to the problem of finding a quantifier-free formula that is equivalent to a quantified formula. For example, $\exists x : y = x^2$ is equivalent to the quantifier-free formula $y \geq 0$ (in the theory of reals). In the theory of reals, for any given quantified formula, a quantifier-free equivalent formula always exists. In fact, it can even be computed algorithmically.

There are tools (some are freely available) that perform quantifier elimination in the theory of reals, such as qepcad [9], [10], [3], [11], [6], and the computer algebra systems redlog [14] and Mathematica. However, quantifier elimination is a computationally hard problem: the time complexity of real quantifier elimination is (both lower and upper) bounded by a double exponential function in the length of the input formula [38], [13]. As a result, when faced with problems that contains large number of variables (about 10) with large (about 3) degrees, quantifier elimination tools can fail to produce a result.

In our experiments, we used a combination approach as described in a recent paper [28]. Our approach can be briefly described as follows: (1) First, we use the virtual substitution method of redlog to eliminate quantified variables. Virtual substitution method [38], [21], [39] can eliminate variables whose degrees are bounded by 2, and hence it can fail to eliminate some variables. However, virtual substitution is an efficient method that scales to large formulas. (2) The output of virtual substitution is a large formula that is given to a simplifier called slfq [5] that simplifies the formula using a divide-and-conquer strategy. (3) The quantified variables that virtual substitution failed to eliminate are then eliminated from the simplified formula using a complete quantifier elimination procedure qepcad.

In all our quantifier elimination runs, the time taken by virtual substitution was negligible. In the first quantifier elimination exercise where Formula ψ_1 was generated, the simplifier slfq took 61.1 seconds of system time and made 8669 calls to qepcad. The step of eliminating c, d from ψ_1 was performed using a single call to qepcad, which took 100 milliseconds of system time. In the next quantifier elimination exercise where we generated ψ_5, virtual substitution took negligible (at most 1 minute of *real* time) time, whereas slfq needed 205.87 seconds of *system* time



(a) (b) (c)

Fig. 3. Simulation of a closed loop control system where the plant is a switched system whose dynamics change every 0.1 time units and the PI controller also dynamically changes its parameters.

and made 7779 calls to `qepcad`. Finally, the quantifier elimination problem for generating ψ_4 was performed by a single call to `qepcad` and it again took negligible time. (All computations described here have been carried out on Intel Xeon E5630 2.53GHz single-core processor (x86_64 arch) with 4G RAM running Ubuntu Linux 2.6.32-26.) The scripts containing quantifier elimination problems generated in the experiments are publicly available [32].

V. Related Work

Our results are analogous to the results on absolute stability and the Lur'e problem [37], [22]. In fact, in retrospect, now that we know the result of Theorem 1 and Theorem 2, we can see if we can prove global (exponential) stability of the feedback system using the known results. The known results are not applicable if $\beta \neq 0$. Hence, let us fix $\beta = 0$.

To compare, we first use our technique to analyze the case when $\beta = 0$. We applied our method to find the condition on a such that the feedback system with $b = \infty$ is always eventually region stable with respect to the region $\texttt{err} = 0$. Using a template for a *general quadratic* Lyapunov function, and for the case when $K_p = 500, K_i = 10$, our tools produced $a \geq 1/25000$.

Now, let us use the passivity theorem. We can express our feedback system in the form required for passivity theorem as follows:

$$\begin{aligned} \frac{d\vec{x}}{dt} &= A\vec{x} + B\vec{u} \\ y &= C\vec{x} + D\vec{u} \\ u &= -\Phi(y) \end{aligned}$$

In our case, we have $\vec{x}$ is a 2-d vector of `err` and `interr`, and

$$A = \begin{bmatrix} -a*K_p & -a*K_i \\ 1 & 0 \end{bmatrix} \quad B = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$
$$C = [K_p \;\; K_i] \quad D = 0$$

The function Φ is a nonlinear function belonging to the sector $[0, \infty)$. Note that the lower bound a was moved into the A matrix. (This is an equivalent way of capturing our condition in Equation 4 where $b = \infty$.)

Now, since $a > 0$, we have that A is Hurwitz, (A, B) is controllable, and (C, A) is observable (if $K_i \neq 0$). We can apply the passivity theorem (circle criterion) and conclude that the feedback system is globally exponentially stable *if* the transfer matrix $H(s) = C(sI - A)^{-1}B + D$ satisfies the condition $\inf_{\omega \in \mathbb{R}} \texttt{Re}(H(j\omega)) > 0$. However, this latter condition is false in general, but it holds when $a * K_p^2 \geq K_i$, which evaluates to $a \geq 1/25000$ for the case when $K_p = 500, K_i = 10$. Thus, we get the same result using passivity theorem as we did using our quantifier elimination procedure. However, our technique is an algorithmic procedure – it can be applied to any system – and it can handle systems on which passivity theorem may not be applicable (for example, when $\beta \neq 0$ or A is not Hurwitz). Also note that our results guarante $a.e.r.s$ stability and not global exponential stability. For instance, even when $0 < a < 1/25000$, we can prove $a.e.r.s$ stability, whereas passivity theorem does not give any useful information when $a < 1/25000$.

We now briefly mention other related work in the literature. Quantifier elimination has been applied to solve nonlinear control system design [18] before. However, most of this early work focused only on continuous dynamical systems (and not switched systems), and moreover, only on simple properties. Properties like safety and stability were not considered – partly because they cannot be captured as semi-algebraic sets in a sound and complete way. We give up completeness and restrict our search for certificates to only certain templates, which helps us reduce the analysis problem to a first order formula over reals.

The classical way of proving stability by finding Lyapunov functions is an instance of certificate-based verification. In this classical approach, the search for Lyapunov function of a particular form is reduced to solving of an $\exists\forall\phi$ formula, where ϕ is an *atomic* fact. Numerical techniques in the form of semidefinite programming exist for solving such sum-of-squares problems [23], [4]. These techniques can be extended to also handle the case when ϕ contains Boolean connectives, but not without losing significant efficiency. The work on barrier certificates [25] moves this overall approach from stability to safety. Apart from stability and safety, there is also work on solving other (region-of-attraction) problems using sum-of-squares programming [36]

There is plenty of work on certificate-based verification of hybrid systems [30], [27], [34], [16], but none of it has used

real quantifier elimination and instead relied on approximate methods to eliminate quantifiers. In a recent paper [29], we used `qepcad` to solve $\exists\forall$ formulas arising from certificate-based synthesis.

Recently, Anai [2] used a combination of numerical methods (sum-of-squares) and symbolic quantifier elimination methods to solve problems arising in control, and such an integration is left for future work here.

VI. Conclusions

We defined the notion of *always eventually region stability* and formulated the *absolute always eventually region stability problem*. We then used the certificate-based approach and reduced the absolute a.e.r.s problem to a first-order (quantified) formulas over the reals. We then used real quantifier elimination methods to eliminate quantifiers from the formula and get a solution for the absolute a.e.r.s problem. An important observation we make here is that the study of absolute stability and absolute always eventually region stability is important for analyzing systems compositionally.

Several avenues exist for future work. While we have framed and solved the absolute a.e.r.s problem for a one-dimensional target region, we believe the results generalize to higher dimensions and this is left for future work. Our result also suggests that it can be used to create abstractions of open systems and this needs to be explored further too.

References

[1] R. Alur, T. Dang, and F. Ivancic. Reachability analysis of hybrid systems via predicate abstraction. In *HSCC*, volume 2289 of *LNCS*. Springer, 2002.

[2] H. Anai. A symbolic-numeric approach to nonlinear dynamical system analysis, 2010. SIAM/MSRI workshop on hybrid method. for symb.-numeric comp.

[3] D. S. Arnon, G. E. Collins, and S. McCallum. Cylindrical algebraic decomposition I: The basic algorithm. *SIAM Journal on Computing*, 13(4):865–877, Nov. 1984.

[4] S. Boyd, L. El Ghaoui, E. Feron, and V. Balakrishnan. *Linear matrix inequalities in system and control theory*. SIAM, 1994. Volume 15 of *Studies in Applied Mathematics*.

[5] C. Brown. slfq: Simplifying large formulas with Qepcad, 2009. `www.usna.edu/Users/cs/qepcad/SLFQ/Home.html`.

[6] C. W. Brown. QEPCAD B: a program for computing with semi-algebraic sets using CADs. *ACM SIGSAM Bulletin*, 37(4):97–108, 2003.

[7] A. Chutinan and B. H. Krogh. Computing polyhedral approximations to flow pipes for dynamic systems. In *37th IEEE Conference on Decision and Control*, 1998.

[8] E. M. Clarke, A. Fehnker, Z. Han, B. H. Krogh, O. Stursberg, and M. Theobald. Verification of hybrid systems based on counterexample-guided abstraction refinement. In *9th Intl. TACAS*, volume 2619 of *LNCS*, pages 192–207. Springer, 2003.

[9] G. E. Collins. The SAC-1 polynomial system. Technical Report 115, Computer Science Department, University of Wisconsin, Madison, Wisconsin, 1968.

[10] G. E. Collins. Quantifier elimination for real closed fields by cylindrical algebraic decomposition – preliminary report. *ACM SIGSAM Bullitin*, 8(3):80–90, Aug. 1974.

[11] G. E. Collins and H. Hong. Partial cylindrical algebraic decomposition for quantifier elimination. *J. Symbolic Computation*, 12(3):299–328, Sept. 1991.

[12] T. Dang and O. Maler. Reachability analysis via face lifting. In *HSCC*, volume 1386 of *LNCS*, pages 96–109. Springer, 1998.

[13] J. H. Davenport and J. Heintz. Real quantifier elimination is doubly exponential. *J. of Symbolic Computation*, 5(1–2):29–35, Feb.–Apr. 1988.

[14] A. Dolzmann and T. Sturm. Redlog: Computer algebra meets computer logic. *ACM SIGSAM Bulletin*, 31(2):2–9, June 1997.

[15] G. Frehse. PHAVer: Algorithmic verification of hybrid systems past hytech. In *HSCC*, volume 3414 of *LNCS*, pages 258–273. Springer, 2005.

[16] S. Gulwani and A. Tiwari. Constraint-based approach for analysis of hybrid systems. In *Proc. 20th CAV*, volume 5123 of *LNCS*, pages 190–203. Springer, 2008.

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

[18] M. Jirstrand. Nonlinear control system design by quantifier elimination. *J. Symb. Comput.*, 24(2):137–152, 1997.

[19] A. B. Kurzhanski and P. Varaiya. On ellipsoidal techniques for reachability analysis. *Dynamics of Continuous, Discrete and Impulsive Systems Series B: Applications and Algorithms*, 9:347–367, 2002.

[20] G. Lafferriere, G. J. Pappas, and S. Yovine. Symbolic reachability computations for families of linear vector fields. *J. Symbolic Computation*, 32(3):231–253, 2001.

[21] R. Loos and V. Weispfenning. Applying linear quantifier elimination. *The Computer Journal*, 36(5):450–462, 1993.

[22] K. S. Narendra and J. H. Taylor. *Frequency domain criteria for absolute stability*. Academic Press, New York, 1973.

[23] P. A. Parrilo. SOS methods for semi-algebraic games and optimization. In *HSCC 2005*, volume 3414 of *LNCS*, page 54. Springer, 2005.

[24] A. Platzer. Differential dynamic logic for hybrid systems. *J. Autom. Reasoning*, 41(2):143–189, 2008.

[25] S. Prajna and A. Jadbabaie. Safety verification of hybrid systems using barrier certificates. In *Proc. 7th HSCC*, volume 2993 of *LNCS*, pages 477–492. Springer, 2004.

[26] S. Prajna, A. Papachristodoulou, and P. A. Parrilo. SOSTOOLS: Sum of Square Optimization Toolbox, 2002. `http://www.cds.caltech.edu/sostools`.

[27] S. Sankaranarayanan, H. Sipma, and Z. Manna. Constructing invariants for hybrid systems. In *Hybrid Systems: Computation and Control, HSCC 2004*, volume 2993 of *LNCS*, pages 539–554. Springer, 2004.

[28] T. Sturm and A. Tiwari. Verification and synthesis using real quantifier elimination, 2011. Submitted.

[29] A. Taly and A. Tiwari. Switching logic synthesis for reachability. In *Intl. Conf. on Embedded Software, EMSOFT*, 2010.

[30] A. Tiwari. Approximate reachability for linear systems. In *Proc. 6th HSCC*, volume 2623 of *LNCS*, pages 514–525. Springer, 2003.

[31] A. Tiwari. Generating box invariants. In *Proc. Hybrid Systems: Computation and Control*, LNCS 4981, pages 658–661. Springer, 2008.

[32] A. Tiwari. Certificate-based verification: Tools and benchmarks, 2011. `http://www.csl.sri.com/~tiwari/existsforall/`.

[33] A. Tiwari and G. Khanna. Series of abstractions for hybrid automata. In *HSCC*, volume 2289 of *LNCS*, pages 465–478. Springer, 2002.

[34] A. Tiwari and G. Khanna. Nonlinear Systems: Approximating reach sets. In *HSCC 2004*, volume 2993 of *LNCS*, pages 600–614. Springer, 2004.

[35] C. J. Tomlin, I. Mitchell, A. M. Bayen, and M. Oishi. Computational techniques for the verification of hybrid systems. *Proc. of the IEEE*, 91(7), 2003.

[36] U. Topku, A. Packard, P. Seiler, and T. J. Wheeler. Stability region analysis using simulation and sum-of-squares programming. In *Proc. American Control Conference*, 2007.

[37] M. Vidyasagar. *Nonlinear systems analysis*. SIAM, 2 edition, 1993.

[38] V. Weispfenning. The complexity of linear problems in fields. *J. of Symbolic Computation*, 5(1&2):3–27, 1988.

[39] V. Weispfenning. Quantifier elimination for real algebra—the quadratic case and beyond. *Applicable Algebra in Engineering Communication and Computing*, 8(2):85–101, Feb. 1997.