Quantitative Model Checking for a Controller Design

YoungMin Kwon[1] and Eunhee Kim[2]

[1] Department of Computer Science, The State University of New York, Korea
119 Songdo Moonhwa-Ro, Yeonsu-Gu, Incheon, Korea 21985,
youngmin.kwon@sunykorea.ac.kr
[2] 2e Consulting Corporation
1710 KnK Digital Tower, 220 Yeongsin-Ro, Yeongdeungpo-gu, Seoul, Korea 07288,
keh@2e.co.kr

Abstract. A controller design method based on a quantitative model checking technique is proposed. Controllers have been designed to shape the closed-loop system's responses to meet certain requirements. We use *Linear Temporal Logic for Control* (LTLC) [15] to formally describe complex requirements and design a controller to meet the requirements with guidance from model checking results. The technique can help design a controller robust against errors systematically hampering the controller's efforts. To demonstrate the usefulness of the proposed technique, we exercised several controller design examples with different types of errors.

1 Introduction

In a Cyber-Physical System (CPS), computers not only perform computations on their memory, but interact with the physical world as well. With the advances in sensor and actuator technologies, CPS can be found in many places such as autonomous cars, smart home appliances, automated farms, and so on. Automatic controllers, right at the border between cyber systems and physical systems, are interfacing the two systems. As cyber systems are becoming more and more complex, it is crucial to be able to formally specify certain requirements about the physical systems and design controllers to guarantee them.

Automatic controllers have been designed to meet certain global properties of a system. For instance, Nyquist plots are drawn for the stability of a system, Bode plots are used to shape overall frequency responses [6]. Even though controllers are designed with some margin of errors, these design practices do not guarantee the non-existence of events that can violate certain properties. In other words, the system cannot guarantee the requirements when events are systematically hampering the system. Formally describing such requirements and designing a controller to meet the requirements will be crucial especially for mission critical or safety critical systems.

We propose a quantitative model checking technique to specify the requirements and to design a controller. Requirements about a closed-loop system are written in a quantitative temporal logic called *Linear Temporal Logic for Control* (LTLC) [15, 14, 1] and controller parameters are searched while guided by the model checking results. In this paper, we designed a Pole-placement controller whose pole locations are chosen by

model checking results. Furthermore, we considered two types of measurement errors: simply bounded errors and an effect of buffering in sensors.

LTLC is expressive enough to specify nontrivial properties about the system. LTLC has the logical and temporal operators of *Linear Temporal Logic* (LTL) that make the specification concise and easy to understand [11, 5]. Its atomic propositions are linear (in)equalities about the input, output, and state variables of a hybrid system. Regarding the expressiveness of LTLC, an (in)equality specifies a hyperplane or a half space in the state space; their conjunctions can express a polytope in the space; and the disjunctions of the polytopes can form non-convex regions. Using the temporal operators, one can specify how such regions should change over time in the state space.

There have been model checking techniques for *infinite* state space. One of the first approaches is Timed Automata where clocks with a constant rate comprise the continuous state space [2]. A more expressive model is hybrid automata, where properties on the trajectories of continuous variables governed by differential equations can be specified [9], but its model checker implementations such as HyTech and UPPAAL handle simplified dynamics like clocks [10, 17]. HySAT and BACH are reachability checkers for hybrid automata [7, 4]. Unlike model checkers, reachability checkers do not check whether a system satisfies a given specification usually written in a temporal logic, but check whether certain states can be reached.

In this paper, the LTLC model checking technique is introduced to check if a controller design can satisfy requirements. Another direction of utilizing the technique is to embed the model checking algorithm into the controller to generate a control input directly [16]. In particular, a goal is described in LTLC and its negation is model-checked within the controller. Because a counterexample contains a control input that can be applied to the system to achieve the original goal, LTLC model checking technique can be employed in the feedback control loop [15]. Model checking techniques have been used in the automatic control systems to find control strategies to satisfy high-level goals [22, 12, 13, 8, 18]. They are based on finite abstraction of state spaces, where a state space is partitioned into finite polytopes and a transition system is built on the equivalent sets of states. A control strategy is computed such that the transition system can satisfy the goals. One of the advantages the LTLC based technique has over the finite abstraction based approaches is that it does not need to pre-partition the whole state space which usually takes a long time.

2 Hybrid System Model

A hybrid system is a mixed system with a continuous system described by differential equations and a discrete system governed by discrete events. For example, changing the transmission gears of a car is a discrete event and the responses of the car at a certain gear position follow a differential equation. We will design a controller for a *Linear Time Invariant* (LTI) system. Furthermore, to capture the effect of system dynamics changes, a hybrid system model is introduced. In this section, we formally define an LTI system, a hybrid system, and a computation path.

A *Linear System* is a system where the superposition principle holds. A *Linear Time Invariant System* is a linear system whose dynamics do not change over time. In

this paper, we consider a discrete-time LTI system that can be obtained by periodically sampling its continuous-time counterpart.

Definition 1. *A discrete-time* Linear Time Invariant (LTI) system[3] *is a seven tuple* $L = \langle U, Y, X, A, B, C, D\rangle$*, where* $U = \{u_1, \ldots, u_{nu}\}$*,* $Y = \{y_1, \ldots, y_{ny}\}$*,* $X = \{x_1, \ldots, x_{nx}\}$ *are the set of input, output, and state variables respectively,* $A \in \mathbb{R}^{nx \times nx}$*,* $B \in \mathbb{R}^{nx \times nu}$*,* $C \in \mathbb{R}^{ny \times nx}$*, and* $D \in \mathbb{R}^{ny \times nu}$ *are system matrices.* □

The relation between the input, output, and state variables of an LTI system satisfy the dynamics equations below.

$$\mathbf{x}(t+1) = A \cdot \mathbf{x}(t) + B \cdot \mathbf{u}(t), \quad \mathbf{y}(t) \ = \ C \cdot \mathbf{x}(t) + D \cdot \mathbf{u}(t), \tag{1}$$

where $\mathbf{u} : \mathbb{N} \to \mathbb{R}^{nu}$, $\mathbf{y} : \mathbb{N} \to \mathbb{R}^{ny}$, and $\mathbf{x} : \mathbb{N} \to \mathbb{R}^{nx}$ are trajectories of the input, output, and state variables respectively. That is, $\mathbf{u}(t)_i = u_i$ at time t for $i = 1, \ldots, nu$, $\mathbf{y}(t)_i = y_i$ at time t for $i = 1, \ldots, ny$, and $\mathbf{x}(t)_i = x_i$ at time t for $i = 1, \ldots, nx$.

A hybrid system is a graph of LTI systems, called *modes*, such that the system can change its dynamics from the set of modes.

Definition 2. *A* Hybrid system *is a quintuple* $H = \langle U, Y, X, M, E\rangle$*, where* $U = \{u_1, \ldots, u_{nu}\}$*,* $Y = \{y_1, \ldots, y_{ny}\}$*,* $X = \{x_1, \ldots, x_{nx}\}$ *are the set of input, output, and state variables respectively,* $M = \{m_1, \ldots, m_{nm}\}$ *is a set of LTI systems called* modes*, and* $E = \{e_1, \ldots, e_{ne}\}$ *is a set of labeled directed edges between the modes.* □

All modes of a hybrid system share the same set of input, output, and state variables. Therefore, a mode m_i of Definition 2 is in the form $m_i = \langle U, Y, X, A_i, B_i, C_i, D_i\rangle$ for $i = 1, \ldots, nm$. A labeled directed edge $e = (m_i, m_j, \psi)$ defines a mode-switch condition from m_i to m_j, where the label ψ is a propositional formula defining its enabling condition. When more than one edge are enabled at a time, the mode transition can nondeterministically occur along any of the enabled edges. For simplicity, we write $m_i \xrightarrow{\psi} m_j$ for (m_i, m_j, ψ) and $m_i \longrightarrow m_j$ for $(m_i, m_j, \mathtt{T})$.

The syntax of the propositional formula[4] ψ is

$$\begin{aligned} \psi &::= \mathtt{T} \mid \mathtt{F} \mid AP \mid (\psi) \mid \neg\psi \mid \psi \wedge \psi \mid \psi \vee \psi \mid \psi \to \psi \mid \psi \leftrightarrow \psi, \\ AP &::= c_1 \cdot v_1 + \cdots + c_n \cdot v_n \bowtie d, \end{aligned}$$

where $c_i, d \in \mathbb{R}$ for $i = 1, \ldots, n$, $v_i \in U \cup Y \cup X$ for $i = 1, \ldots, n$, and $\bowtie \in \{\leq, <, =, \neq, >, \geq\}$. The operators of ψ have their usual meanings and the truth value of AP depends on the system state. Let a state of a hybrid system $s \in M \times \mathbb{R}^{nu} \times \mathbb{R}^{ny} \times \mathbb{R}^{nx}$ be $s = (m, \mathbf{u}(t), \mathbf{y}(t), \mathbf{x}(t))$ and let an assignment function $\theta_s : U \cup Y \cup X \to \mathbb{R}$ be $\theta_s(v) = \mathbf{u}(t)_i$ if $v = u_i$ for some $i \in [1, nu]$, $\theta_s(v) = \mathbf{y}(t)_i$ if $v = y_i$ for some $i \in [1, ny]$, and $\theta_s(v) = \mathbf{x}(t)_i$ if $v = x_i$ for some $i \in [1, nx]$. Then, $c_1 \cdot v_1 + \cdots + c_n \cdot v_n \bowtie d$ is true in s iff $c_1 \cdot \theta_s(v_1) + \cdots + c_n \cdot \theta_s(v_n) \bowtie d$.

[3] We use the term LTI systems for both continuous-time LTI systems and discrete-time LTI systems when the usage is clear from the context.

[4] We use the term propositional formula because the semantics of LTLC is interpreted over each computation path where all the variables are assigned with a value.

Let $\mu : \mathbb{N} \to M$ be a function that returns the mode at time t and by overloading the notations let $A : \mathbb{N} \to \mathbb{R}^{nx \times nx}$, $B : \mathbb{N} \to \mathbb{R}^{nx \times nu}$, $C : \mathbb{N} \to \mathbb{R}^{ny \times nx}$, and $D : \mathbb{N} \to \mathbb{R}^{ny \times nu}$ be the functions that return the system matrices A, B, C and D of the mode $\mu(t)$. Then the system variables satisfy the dynamics equations below.

$$\mathbf{x}(t+1) = A(t) \cdot \mathbf{x}(t) + B(t) \cdot \mathbf{u}(t), \quad \mathbf{y}(t) \ = \ C(t) \cdot \mathbf{x}(t) + D(t) \cdot \mathbf{u}(t). \tag{2}$$

Solving $\mathbf{x}(t)$ and $\mathbf{y}(t)$ in Equation (2) in terms of $\mathbf{x}(0)$, $\mathbf{u}(\tau)$, $A(\tau)$, $B(\tau)$, $C(\tau)$ and $D(\tau)$ for $0 \le \tau \le t$,

$$\begin{aligned} \mathbf{x}(t) &= \left(\prod_{i=0}^{t} A(i) \right) \cdot \mathbf{x}(0) + \sum_{i=0}^{t-1} \left(\prod_{j=i+1}^{t-1} A(j) \right) \cdot B(i) \cdot \mathbf{u}(i), \\ \mathbf{y}(t) &= C(t) \cdot \mathbf{x}(t) + D(t) \cdot \mathbf{u}(t), \end{aligned} \tag{3}$$

where $\prod_{i=0}^{t} A_i = A_t \cdot A_{t-1} \cdots A_1 \cdot A_0$.

Definition 3. *A* computation path *of a hybrid system $H = \langle U, Y, X, M, E \rangle$ is a function $\pi : \mathbb{N} \to M \times \mathbb{R}^{nu} \times \mathbb{R}^{ny} \times \mathbb{R}^{nx}$ such that $\pi(t) = (\mu(t), \mathbf{u}(t), \mathbf{y}(t), \mathbf{x}(t))$. The input, output, and state trajectories $\mathbf{u}$, $\mathbf{y}$, and $\mathbf{x}$ satisfy Equation (2), and for all $t \ge 0$, there is a ψ such that $\mu(t) \xrightarrow{\psi} \mu(t+1) \in E$ and ψ is true in $\pi(t)$.* □

3 Specifications on Hybrid System Models

Given a hybrid system, we need a way to write some desirable or undesirable properties about the system. We developed a quantitative temporal logic called *Linear Temporal Logic for Control* (LTLC) [15] to formally specify those properties. In this section, we describe the syntax and semantics of LTLC.

LTLC has the logical and temporal operators of LTL. However, to specify properties of the infinite states of hybrid systems, its atomic propositions are linear (in)equalities about the system states.

Definition 4. *The syntax of LTLC is*

$$\begin{aligned} \phi ::= {} & \mathtt{T} \mid \mathtt{F} \mid AP \mid (\phi) \mid \neg\phi \mid \phi \wedge \varphi \mid \phi \vee \varphi \mid \phi \to \varphi \mid \phi \leftrightarrow \varphi \mid \\ & \mathtt{X}\,\phi \mid \Diamond\,\phi \mid \Box\,\phi \mid \phi\,\mathtt{U}\,\varphi \mid \phi\,\mathtt{R}\,\varphi, \\ AP ::= {} & c_1 \cdot v_1 + \cdots + c_n \cdot v_n \bowtie d \mid H@m, \end{aligned}$$

where $c_i, d \in \mathbb{R}$ for $i = 1, \ldots, n$, $v_i \in U \cup Y \cup X$ for $i = 1, \ldots, n$, $\bowtie \in \{\le, <, =, \ne, >, \ge\}$, and m is a mode of a hybrid system H. □

An implicit meaning of LTLC formulas is as follows. An *atomic proposition* $c_1 \cdot v_1 + \cdots + c_n \cdot v_n \bowtie d$ at time t is true iff the (in)equality is true when the variables are assigned with their corresponding values at the state $\pi(t)$. That is, $c_1 \cdot v_1 + \cdots + c_n \cdot v_n \bowtie d \Leftrightarrow c_1 \cdot \theta_{\pi(t)}(v_1) + \cdots + c_n \cdot \theta_{\pi(t)}(v_n) \bowtie d$. $H@m$ is true at time t iff the system is in mode m at the state $\pi(t)$. In other words, $H@m \Leftrightarrow \mu(t)$ is m.

$$
\begin{array}{l}
\pi, t \models \mathtt{T}, \\
\pi, t \not\models \mathtt{F}, \\
\pi, t \models c_1 \cdot v_1 + \cdots + c_n \cdot v_n \bowtie d \\
\qquad\qquad \Leftrightarrow \sum_{i=1}^{n} c_i \cdot \theta_{\pi(t)}(v_i) \bowtie d, \\
\pi, t \models H@m \Leftrightarrow \mu(t) = m, \\
\pi, t \models \neg\, \phi \quad \Leftrightarrow \pi, t \not\models \phi, \\
\pi, t \models \phi \wedge \varphi \Leftrightarrow \pi, t \models \phi \text{ and } \pi, t \models \varphi, \\
\pi, t \models \phi \vee \varphi \Leftrightarrow \pi, t \models \phi \text{ or } \pi, t \models \varphi, \\
\pi, t \models \mathtt{X}\, \phi \quad \Leftrightarrow \pi, t+1 \models \phi, \\
\pi, t \models \phi\, \mathtt{U}\, \varphi \Leftrightarrow \pi, i \models \varphi \text{ for some } i \geq t \text{ and} \\
\qquad\qquad\qquad \pi, j \models \phi \text{ for all } t \leq j < i, \\
\pi, t \models \phi\, \mathtt{R}\, \varphi \Leftrightarrow \pi, t \models \varphi \text{ and for } i > t,\ \pi, i \models \varphi \\
\qquad\qquad\qquad \text{if } \pi, j \not\models \phi \text{ for all } t \leq j < i.
\end{array}
$$

(a)

$$
\begin{array}{l}
\pi, t \models_b \mathtt{T}, \\
\pi, t \not\models_b \mathtt{F}, \\
\pi, t \models_b c_1 \cdot v_1 + \cdots + c_n \cdot v_n \bowtie d \\
\qquad\qquad \Leftrightarrow \sum_{i=1}^{n} c_i \cdot \theta_{\pi(t)}(v_i) \bowtie d, \\
\pi, t \models_b H@m \Leftrightarrow \mu(t) = m, \\
\pi, t \models_b \neg\, \phi \quad \Leftrightarrow \pi, t \not\models_b \phi, \\
\pi, t \models_b \phi \wedge \varphi \Leftrightarrow \pi, t \models_b \phi \text{ and } \pi, t \models_b \varphi, \\
\pi, t \models_b \phi \vee \varphi \Leftrightarrow \pi, t \models_b \phi \text{ or } \pi, t \models_b \varphi, \\
\pi, t \models_b \mathtt{X}\, \phi \quad \Leftrightarrow t < b \text{ and } \pi, t+1 \models_b \phi, \\
\pi, t \models_b \phi\, \mathtt{U}\, \varphi \Leftrightarrow \pi, i \models_b \varphi \text{ for some } t \leq i \leq b \text{ and} \\
\qquad\qquad\qquad \pi, j \models_b \phi \text{ for all } t \leq j < i, \\
\pi, t \models_b \phi\, \mathtt{R}\, \varphi \Leftrightarrow \pi, i \models_b \phi \text{ for some } t \leq i \leq b \text{ and} \\
\qquad\qquad\qquad \pi, j \models_b \varphi \text{ for all } t \leq j \leq i.
\end{array}
$$

(b)

Fig. 1. (a) ternary satisfaction relation $\models$ and (b) bounded ternary satisfaction relation $\models_b$.

The *logical operators* have their usual meanings. $\neg\phi$ is true at t iff ϕ is false at t; $\phi \wedge \varphi$ is true at t iff both ϕ and φ are true at t; $\phi \vee \varphi$ is true at t iff ϕ is true at t or φ is true at t. $\phi \rightarrow \varphi$ is equivalent to $\neg\phi \vee \varphi$ and $\phi \leftrightarrow \varphi$ is equivalent to $(\phi \rightarrow \varphi) \wedge (\varphi \rightarrow \phi)$.

The *temporal operators* have the following meanings. $\mathtt{X}\,\phi$ is true at t iff ϕ is true at $t+1$; $\Box\,\phi$ is true at t iff ϕ is always true from t; $\Diamond\,\phi$ is true at t iff ϕ eventually becomes true at some $t' \geq t$. $\phi\,\mathtt{U}\,\varphi$ is true at t iff ϕ is true until φ eventually becomes true. To be more specific, $\phi\,\mathtt{U}\,\varphi$ is true iff there is a time $t' \geq t$ when φ becomes true and ϕ is true for all $\tau \in [t, t')$. $\phi\,\mathtt{R}\,\varphi$ is equivalent to $\varphi\,\mathtt{U}\,(\phi \wedge \varphi)$ except that ϕ is not required to hold eventually. Between $\mathtt{U}$ and $\mathtt{R}$ operators, the following equivalence holds $\neg(\neg\phi\,\mathtt{U}\,\neg\varphi) \equiv \phi\,\mathtt{R}\,\varphi$.

Formally, the semantics of LTLC can be defined by the ternary satisfaction relation $\models \subset \Pi \times \mathbb{N} \times \Phi$ and the binary satisfaction relation $\models \subset \mathcal{H} \times \Phi$, where Π is the set of computation paths, Φ is the set of LTLC formulas, and $\mathcal{H}$ is the set of hybrid system models. For simplicity, we write $\pi, t \models \phi$ for $(\pi, t, \phi) \in \models$ and $H \models \phi$ for $(H, \phi) \in \models$.

Definition 5. *The* ternary satisfaction relation $\models \subset \Pi \times \mathbb{N} \times \Phi$ *is defined in Figure 1 (a). Using the ternary relation, the* binary satisfaction relation $\models \subset \mathcal{H} \times \Phi$ *is*

$$H \models \phi \Leftrightarrow \phi, 0 \models \phi \text{ for all computation paths } \pi \text{ of } H.$$

□

The missing operators have the following equivalence relations. $\Box\,\phi \equiv \mathtt{F}\,\mathtt{R}\,\phi$, and $\Diamond\,\phi \equiv \mathtt{T}\,\mathtt{U}\,\phi$. In addition, $\phi\,\mathtt{U}\,\varphi \equiv \varphi \vee (\,\phi \wedge \mathtt{X}\,(\phi\,\mathtt{U}\,\varphi)\,)$ and $\phi\,\mathtt{R}\,\varphi \equiv (\phi \wedge \varphi) \vee (\,\varphi \wedge \mathtt{X}\,(\phi\,\mathtt{R}\,\varphi)\,)$ are commonly used equivalence relations in the model checking process.

The satisfaction relation $\models$ of Definition 5 is about computation paths of the infinite length. However, in practice, it is difficult to track system trajectories unboundedly. Hence, we developed a bounded model checking algorithm, where infinite paths ending

with a loop before the bound or paths whose satisfiability can be decided before the bound are considered [3].

To define the bounded satisfaction relation $\models_b$, let us define a helper function $\tau_{s,p} : \mathbb{N} \to [0, s + p)$ first. $\tau_{s,p}$ maps a time step to an index into a loop-ending computation path such that $\tau_{s,p}(t) = t$ if $t < s + p$; otherwise, $\tau_{s,p}(t) = s + (t - s) \bmod p$. A simple example of this helper is: $\tau_{1,2}(t) = 0, 1, 2, 1, 2, \ldots$ when $t = 0, 1, 2, 3, 4, \ldots$. A computation path ending with a loop of a period p starting from time s satisfies $\pi(t) = \pi(\tau_{s,p}(t))$ for $t \geq 0$.

Definition 6. *The* ternary bounded satisfaction relation $\models_b \subset \Pi \times \mathbb{N} \times \Phi$ *is defined in Figure 1 (b). Using the ternary relation, the* bounded binary satisfaction relation $\models_b \subset \mathcal{H} \times \Phi$ *is*

$$H \models_b \phi \Leftrightarrow \begin{cases} \pi, 0 \models \phi & \text{if } \pi(t) = \pi(\tau_{s,p}(t)) \text{ for } t \geq 0 \text{ and } s + p \leq b \\ \pi, 0 \models_b \phi & \text{otherwise.} \end{cases}$$

□

Finally, to bring some intuitions about LTLC model checking algorithm, let us explain an example. LTLC model checking algorithm converts model checking problems to a series of *Linear Programming* (LP) [19] problems.

Example 1. Let $H = \langle \{u\}, \{y\}, \{x\}, \{M, N\}, \{M \to N, N \to N\} \rangle$ be a hybrid system, where $M = \langle \{u\}, \{y\}, \{x\}, 1, 2, 5, 0 \rangle$ and $N = \langle \{u\}, \{y\}, \{x\}, 3, -1, 7, 0 \rangle$ are its modes; a specification φ be $\varphi = \neg\phi$, where $\phi = (H@M \wedge y \leq 1) \wedge \mathbf{X}\,(y \leq 1) \wedge \mathbf{X}\,\mathbf{X}\,(y \leq 1)$; and the model checking problem be $H \models_2 \varphi$.

In the model checking process, we are looking for a counterexample that would violate the specification φ. Hence, we are searching for a computation path π, such that $\pi, 0 \models_2 \phi$. Based on the ternary satisfaction relation of Figure 1 (b), π should satisfy

$$\pi, 0 \models_2 H@M \text{ and } \pi, 0 \models_2 y \leq 1 \text{ and } \pi, 1 \models_2 y \leq 1 \text{ and } \pi, 2 \models_2 y \leq 1.$$

Let $\mu : \mathbb{N} \to \{M, N\}$ be a mode function, $\mathbf{u} : \mathbb{N} \to \mathbb{R}$, $\mathbf{y} : \mathbb{N} \to \mathbb{R}$, and $\mathbf{x} : \mathbb{N} \to \mathbb{R}$ be the input, output, and state trajectories of H. Observe that depending on the mode $\mu(t)$ at time t, $\mathbf{u}$, $\mathbf{y}$, and $\mathbf{x}$ conform either to the dynamics equations of M or to those of N. The variable assignment at time t is $\theta_{\pi(t)}(u) = \mathbf{u}(t)$, $\theta_{\pi(t)}(y) = \mathbf{y}(t)$, and $\theta_{\pi(t)}(x) = \mathbf{x}(t)$.

Let us check the conditions about the modes first. Based on the edges of H, μ can be either $MNNN\cdots$ or $NNN\cdots$. Because $H@M$ in ϕ is enforced at $t = 0$, μ should be $MNNN\cdots$.

The other inequality conditions can be written as

$$\mathbf{y}(0) \leq 1 \text{ and } \mathbf{y}(1) \leq 1 \text{ and } \mathbf{y}(2) \leq 1.$$

We rewrite the conditions in term of $\mathbf{x}(0)$ and $\mathbf{u}$. Let a vector of variables $\mathbf{v}$ be $\mathbf{v} = [\mathbf{x}(0), \mathbf{u}(0), \mathbf{u}(1)]^T$, then because $\mu = MNNN\cdots$, the following holds.

$$\begin{aligned} \mathbf{y}(0) &= 5 \cdot \mathbf{x}(0) &&= [5, 0, 0] \cdot \mathbf{v}, \\ \mathbf{y}(1) &= 7 \cdot (\mathbf{x}(0) + 2 \cdot \mathbf{u}(0)) &&= [7, 14, 0] \cdot \mathbf{v}, \\ \mathbf{y}(2) &= 7 \cdot (3 \cdot (\mathbf{x}(0) + 2 \cdot \mathbf{u}(0)) - \mathbf{u}(1)) &&= [21, 42, -7] \cdot \mathbf{v}. \end{aligned}$$

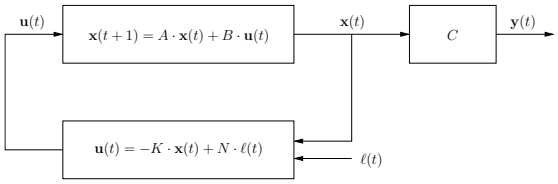


Fig. 2. Pole-placement control: $\mathbf{u}(t) = -K \cdot \mathbf{x}(t) + N \cdot \ell(t)$ is the control law.

Now, the model checking problem can be converted to a feasibility checking problem as below.

$$\pi, 0 \models_2 \phi \Leftrightarrow \{\mathbf{v} : [5,0,0] \cdot \mathbf{v} \leq 1,\ [7,14,0] \cdot \mathbf{v} \leq 1,\ [21,42,-7] \cdot \mathbf{v} \leq 1\} \neq \emptyset.$$

The feasibility of the set of inequality constraints can be checked by solving an LP problem [19]. There are feasible solutions that satisfy all inequalities. For example $\mathbf{v} = 0$ is a feasible solution. Hence, any computation path with $\mu(0) = N$, $\mu(t) = M$ for $t \geq 1$, $\mathbf{x}(0) = 0$, $\mathbf{u}(0) = 0$, and $\mathbf{u}(1) = 0$ violates the original specification φ. □

4 Pole-placement Control

Automatic control is a process of shaping the system responses to a desirable one and making the system output follow a *reference value*. In a feedback control system the output of a system is fed back to a controller such that a *control input* to the system can be generated based on the value. The whole system is called a *closed-loop system*.

The *transfer function* of a system is a mapping from the Laplace transform of an input to the Laplace transform of the output [6]. The roots of the numerator of the transfer function are called *zeros* and the roots of the denominator of the transfer function are called *poles*. The responses of the system are determined by the locations of the poles and zeros called *pole-zero constellation*.

Pole-placement control is a state space control method that shapes the closed-loop system responses by placing the poles of the system at predetermined locations [6]. Figure 2 shows a block diagram of a closed-loop system using the Pole-placement control. The target system, the upper-left-side block of Figure 2, has the dynamics equations of Equation (1). Let the *control law* be a feedback of a linear combination of the state variables.

$$\mathbf{u}(t) = -K \cdot \mathbf{x}(t),$$

where $K \in \mathbb{R}^{nu \times nx}$ is a design parameter that we need to decide. Substituting this equation into Equation (1), the characteristic equation of the closed-loop system is

$$\det(p \cdot I - A + B \cdot K) = 0.$$

The poles of the closed-loop system are the roots of this polynomial and K can be computed once the desired locations of the poles are decided.

The control law above decides the transient responses of the system, but we still need to make the system follow a reference value. Considering the reference input $\ell(t)$ the control law has the following form.

$$\mathbf{u}(t) = -K \cdot \mathbf{x}(t) + N \cdot \ell(t), \tag{4}$$

where the value of $N \in \mathbb{R}^{nu \times ny}$ can be computed from A, B, C, D, and K matrices [6]. Equation (4) is the control law of Pole-placement control, represented by the lower block of Figure 2.

In this paper, we employed the LTLC model checking technique to decide the locations of the poles. Specifically, the desirable responses of the closed-loop system were described in LTLC and the locations of the poles were iteratively searched while guided by the model checking results.

Model checking an uncontrolled system is straightforward: in a model checking problem $H \models \phi$, the system dynamics of Equation (2) can be easily described in H. However, model checking a closed-loop system is a little trickier unless the dynamics of the closed-loop system are computed separately. To avoid this additional step and to make the proposed method readily applicable to many situations, we embedded the control law in the specification ϕ. Particularly, the control law was brought in as an always enforced equality constraint between the input $\mathbf{u}(t)$ of Equation (2) and the RHS of the control law of Equation (4). To restrict the computation paths to only those that respect the control law, we added ϕ_c below to a specification as a precondition.

$$\phi_c = \Box\, (\mathbf{u} = -K \cdot \mathbf{x} + N \cdot \ell).$$

Observe that the equality sign between $\mathbf{u}$ and $-K \cdot \mathbf{x} + N \cdot \ell$ is not the assignment but the equality constraint.

5 Controller Design Guided by LTLC Model Checking

To demonstrate the proposed method, we design a helicopter velocity control system [6]. We will express properties such as the settling time, the maximum overshoot, etc in LTLC and evaluate the effects of measurement errors through model checking. Guided by model checking results, we will design a Pole-placement controller and make it robust against the errors.

5.1 Pole-placement Controller Design

Figure 3 shows a third-order continuous-time dynamics model of the longitudinal motion of a helicopter. By sampling the continuous-time model at $T = 100$ *msec* period, we acquired its discrete-time dynamics equations. Let $w(t)$ be a wall clock and $\ell(t)$ be a constant input of one that will be used for a clock duration and for a reference input value. Let $\mathbf{x}(t)$ be $[q(t), p(t), v(t), w(t)]^T$ and $\mathbf{u}(t)$ be $[r(t), \ell(t)]^T$, then the discrete-time model obtained by the *Zero Order Hold* (ZOH) method [20] with 100 *msec* sampling interval is

$$\mathbf{x}(t+1) = A \cdot \mathbf{x}(t) + B \cdot \mathbf{u}(t), \tag{5}$$

$$\frac{d}{dt}\tilde{\mathbf{x}}(t) = F \cdot \tilde{\mathbf{x}}(t) + G \cdot \tilde{r}(t),$$
$$\tilde{\mathbf{x}}(t) = [q(t)\ p(t)\ v(t)]^T,$$
$$F = \begin{bmatrix} -0.4 & 0 & -0.01 \\ 1 & 0 & 0 \\ -1.4 & 9.8 & -0.02 \end{bmatrix},\ G = [6.3\ 0\ 9.8]^T,$$

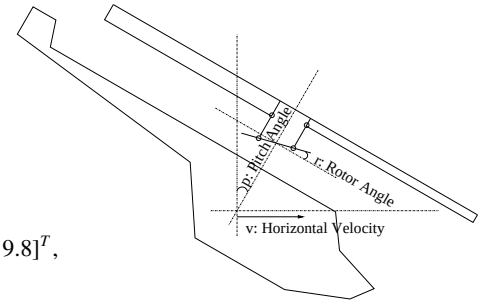


Fig. 3. A helicopter model. In the dynamics equations, q and p are the pitch rate and the pitch angle of the fuselage respectively, v is the horizontal velocity, and r is the rotor angle.

where the matrices A and B are in the mode n of Figure 4.

The hybrid system model with this single mode is

$$H = \langle U, Y, X, \{n\}, \{n \longrightarrow n\}\rangle,\ \ n = \langle U, Y, X, A, B, 0, 0\rangle,$$

where $U = \{r, \ell\}$ is the set of input variables, $Y = \{\}$ is the empty set for the output variables, $X = \{q, p, v, w\}$ is the set of state variables, and n is the LTI system model.

From Equation (4), the control law is in the form

$$r(t) = N \cdot \ell(t) - Kq \cdot q(t) - Kp \cdot p(t) - Kv \cdot v(t).$$

Suppose that we want to place the poles of the closed-loop system at 0.7, $0.7 \cdot e^{\frac{\pi}{12}i}$ and $0.7 \cdot e^{-\frac{\pi}{12}i}$, then $N = 0.6833$, $Kq = 0.1644$, $Kp = 5.2374$, and $Kv = 0.6793$.

This control law can be described in LTLC as

$$\phi'_c = \Box\,(\phi_c),\ \text{where}\ \phi_c = (r = N \cdot \ell - Kq \cdot q - Kp \cdot p - Kv \cdot v),$$
$$\phi'_\ell = \Box\,(\phi_\ell),\ \text{where}\ \phi_\ell = (\ell = 1).$$

The desired properties about the closed-loop system can be specified in LTLC as well. (1) we want to limit the maximum *overshoot* to 0.1 m/sec when the reference input is 1 m/sec. Because the output variable for the velocity is v, using the *always* operator, this condition can be written as below.

$$\phi'_o = \Box\,(\phi_o),\ \text{where}\ \phi_o\ =\ (v \leq 1.1).$$

(2) we want to keep the maximum *pitch-angle* not larger than 0.13 radian. Similarly to the previous condition, we can make an always formula about the output variable p.

$$\phi'_a = \Box\,(\phi_a),\ \text{where}\ \phi_a\ =\ (p \leq 0.13).$$

(3) with regard to the *settling-time*, we want to maintain the velocity of helicopter within the interval $[0.9, 1.1]$ m/sec from 1.5 *sec* onwards. In general, the time step t used in LTL model checking processes is not exposed to the specification. However, we can define a clock like $w(t+1) = w(t) + T \cdot \ell(t)$ and explicitly use it in the specification.

To specify that this settling-time condition is enforced only after 1.5 *sec*, we used the precondition $w(t) \geq 1.5$ in the always formula as below.

$$\phi'_s = \Box\,(\phi_s), \text{ where } \phi_s = (w \geq 1.5 \rightarrow (0.9 \leq v \wedge v \leq 1.1)).$$

(4) finally, the initial condition that the system is in the relaxed state can be expressed as below.

$$\phi_i = (q = 0 \wedge p = 0 \wedge v = 0 \wedge w = 0).$$

Combining them, the overall specification ϕ can be constructed as below.

$$\phi = (\phi_i \wedge \Box\,\phi_c \wedge \Box\,\phi_\ell) \rightarrow (\Box\,\phi_o \wedge \Box\,\phi_a \wedge \Box\,\phi_s).$$

One of the practical concerns is that we will need to check the specification over and over again while searching for the location of poles. Hence, any speed up in the model checking process will greatly improve the whole design process. An immediate improvement is to combine the *always* operators together as below so that the Büchi automaton $\mathcal{B}_\phi$ for the specification ϕ has a smaller number of nodes.

$$\phi = (\phi_i \wedge \Box\,(\phi_c \wedge \phi_\ell)) \rightarrow \Box\,(\phi_o \wedge \phi_a \wedge \phi_s).$$

A more significant improvement came from removing the *always* operators all together. Checking the bounded satisfaction relation $\models_b$ of Figure 1 (b) is usually faster than its unbounded counterpart $\models$ of Figure 1 (a).[5] Observe that, based on Figure 1 (b), *always* formula does not satisfy $\models_b$. If we relax the validation to a finite horizon, then we can reformulate the specification that can be checked by the efficient $\models_b$.

It might be easier to understand the formula to build by examining its counterexample. All counterexamples violate any of ϕ_o, ϕ_a, or ϕ_s conditions at some time. On the other hand, the computation paths that do not satisfy ϕ_c and ϕ_ℓ before any of ϕ_o, ϕ_a, or ϕ_s are violated cannot be a counterexample. Likewise, computation paths that do not satisfy the precondition ϕ_i initially should be disregarded as well. To recap, a counterexample is a computation path that satisfies ϕ_i initially and satisfies ϕ_c and ϕ_ℓ until any of ϕ_o, ϕ_a, or ϕ_s are violated. Hence, the reformulated specification to check is

$$\phi = \phi_i \rightarrow \neg(\ (\phi_c \wedge \phi_\ell)\,\mathtt{U}\,\neg(\phi_o \wedge \phi_a \wedge \phi_s)\ ).$$

Note that counterexamples satisfy $\phi_i \wedge (\ (\phi_c \wedge \phi_\ell)\,\mathtt{U}\,\neg(\phi_o \wedge \phi_a \wedge \phi_s)\ )$.

For a bound of 18, the model checking problem we want to examine is

$$H \models_{18} \phi.$$

Figure 4 shows an LTLC-Checker description for the model checking problem. In the description, contents from the # mark to the end of the line are comments. The description has two main sections: `model` section for a hybrid model definition and

[5] To check the ternary satisfaction relation $\models$, LTLC model checker has to find accepting runs ending with a loop and examine the feasibility of the runs generated by the Cartesian product of the accepting runs and computation paths ending with a loop of all possible periods.

```
model:
 const
  ### controller params & err bound
  Kq=0.1644,Kp=5.2374,Kv=0.6793,N=0.6833
  EB=0.0005;
 var
  ### pitch rate, pitch angle, velocity
  q: state, p: state, v: state,
  ### errors in q, p, v
  eq: input, ep: input, ev: input,
  ### measured q, p, v
  mq: output, mp: output, mv: output,
  ### rotor angle, constant one
  r: input, l: input,
  ### wall-clock, last rotor angle
  w: state, rr: state;
 mode
  ### sampled at 100 msec interval
  n = {
   q = 0.9608*q-0.0005*p-0.0010*v+0.6171*r,
   p = 0.0980*q+1.0000*p         +0.0311*r,
   v =-0.0888*q+0.9790*p+0.9981*v+0.9457*r,
   mq = q, mp = p, mv = v, rr = r,
   w = w + 0.1*l  },
  ### sampled at 90 msec interval
  s = {
   q = 0.9608*q-0.0005*p-0.0010*v+0.6171*r,
   p = 0.0980*q+1.0000*p         +0.0311*r,
   v =-0.0888*q+0.9790*p+0.9981*v+0.9457*r,
   mq= 1.0040*q         +0.0001*v-0.0631*rr,
   mp=-0.0100*q+1.0000*p         +0.0003*rr,
   mv= 0.0145*q-0.0980*p+1.0002*v-0.0985*rr,
   rr= r,
   w = w + 0.1*l };
 edge
  Enn = n -> n, Ens = n -> s,
  Esn = s -> n, Ess = s -> s;
 system
### ex1, ex2
  sys = ( {n}, {Enn} );
### ex3
# sys = ( {n, s}, {Enn, Ens, Esn, Ess} );


specification:
 condition
  init = (q=0 /\ p=0 /\ v=0 /\ w=0 /\ rr=0),
  overshoot = (v <= 1.1),
  settling_time = (w >= 1.5 ->
          (0.9 <= v /\ v <= 1.1)),
  pitch_angle = (p <= 0.13),
  ebound = ( -EB < eq /\ eq < EB /\
             -EB < ep /\ ep < EB /\
             -EB < ev /\ ev  < EB ),
  precond = (ctrl_law /\ l = 1 /\ ebound),
### ex1
  ctrl_law = (r = N*l - Kq*q  - Kp*p  - Kv*v),
### ex2
# ctrl_law = (r = N*l - Kq*q  - Kp*p  - Kv*v
#                     - Kq*eq - Kp*ep - Kv*ev),
### ex3
# ctrl_law = (r = N*l - Kq*mq - Kp*mp - Kv*mv),
  spec = (init -> ~(precond U ~(overshoot /\
                                settling_time /\
                                pitch_angle)));

check
  sys |= spec in 18;
```

Fig. 4. LTLC description for the hybrid system model and the specification.

`specification` section for describing a model checking problem. Input, output, and state variables are defined after `var` tag with their corresponding types suffixed. Using the variables, LTI systems and edges are defined after `mode` and `edge` tags respectively. A hybrid system comprising the modes and edges is defined after `system` tag. Regarding the checker description for the operators of LTLC, the logical operators $\wedge$, $\vee$, $\neg$, $\rightarrow$, and $\leftrightarrow$ are `/\`, `\/`, `~`, `->`, and `<->` respectively. The temporal operators `X`, `U`, `R`, $\Diamond$, and $\Box$ are `X`, `U`, `R`, `<>`, and `[]` respectively. Finally, `sys |= spec in 18` after `check` tag describes the model checking problem $H \models_{18} \phi$.

To simplify the example, we restricted the locations of poles to $\{r, r{\cdot}e^{\frac{\pi}{12}i}, r{\cdot}e^{-\frac{\pi}{12}i}\}$ and looked for the range of r that would make the closed-loop system satisfy the requirements. While model checking the description of Figure 4 with the controller parameters Kq, Kp, Kv, and N computed from the locations of poles, we found that if r is in the range of $0.700 \leq r \leq 0.716$, the controller satisfies the requirements. The responses of a controller when $r = 0.715$ are in Figure 5 (a) and Figure 5 (b) although their trajectories are disturbed by errors.

5.2 Controller Design with Measurement Errors

Practically all measurements about a system have errors coming from various sources like device errors, environmental noise, and so on. Here, we design a controller that can satisfy the requirements despite the presence of measurement errors. We do not assume that the errors are governed by any dynamics equations or they have any statistical behaviors. Instead, they can take any values in a range and trying to drive the closed-loop system to violate the requirements. One can regard the model checking process a two player game: the errors are trying to attack the system to violate the requirements, whereas the controller is trying to make the system comply with them.

We consider a case where additive errors are disturbing some or all of state variables directly. The measurement errors do not alter the underlying system state, but they feed incorrect state information to the controller and have effects on the computation of the next control input. The control law is reformulated as below.

$$\mathbf{u}(t) = -K \cdot (\mathbf{x}(t) + \mathbf{e}(t)) + N \cdot \ell(t),$$

where $\mathbf{e} : \mathbb{N} \to \mathbb{R}^{nx}$ is the measurement error function. The state trajectory of the closed-loop system deviates from the ideal one from the next step onward.

For the helicopter example, let $eq(t)$, $ep(t)$, and $ev(t)$ be the measurement errors in the pitch rate, pitch angle, and velocity. Including these errors, the control law is rewritten as below.

$$r(t) = N \cdot \ell(t) - Kq \cdot (q(t) + eq(t)) - Kp \cdot (p(t) + ep(t)) - Kv \cdot (v(t) + ev(t)).$$

To describe the new control law in LTLC, we added three input variables eq, ep, and ev respectively for the errors in the pitch rate, pitch angle, and velocity. These variables are added to the state variables q, p, and v in the control law and disturb the computation of the next control input value. The control law in LTLC is

$$\phi_c' = \Box\,(\phi_c), \text{ where } \phi_c \;=\; (r = N \cdot \ell - Kq \cdot (q + eq) - Kp \cdot (p + ep) - Kv \cdot (v + ev)).$$

Observe that the additive measurement errors do not touch the system dynamics directly, but disturb the control law and change the state trajectories of the closed-loop system. During the LTLC model checking process, the checker will try to find error functions $eq(t)$, $ep(t)$, and $ev(t)$ that can violate the requirements.

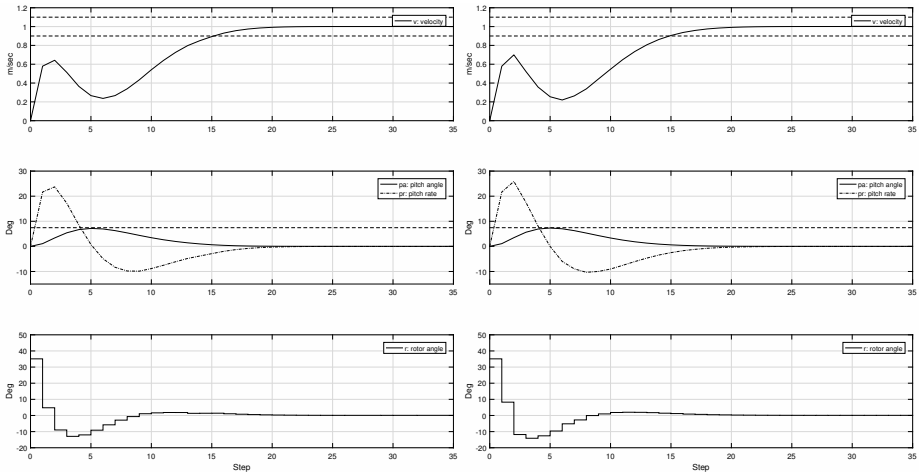
For these measurement errors, we assume that they are always within $(\underline{e}, \bar{e})$ range, where $\underline{e} = -0.0005$ and $\bar{e} = 0.0005$. This *error bound* condition can be expressed in LTLC as follows

$$\phi_e' = \Box\,(\phi_e), \text{ where } \phi_e \;=\; \underline{e} < eq \wedge eq < \bar{e} \;\wedge\; \underline{e} < ep \wedge ep < \bar{e} \;\wedge\; \underline{e} < ev \wedge ev < \bar{e}.$$

With the reformulated control law ϕ_c and the error bound condition ϕ_e, the combined formula ϕ to check is as below.

$$\phi = \phi_i \rightarrow \neg(\,(\phi_c \wedge \phi_\ell \wedge \phi_e)\,\mathsf{U}\,\neg(\phi_o \wedge \phi_a \wedge \phi_s)\,).$$

When the locations of the poles are $\{r,\ r \cdot e^{\frac{\pi}{12}i},\ r \cdot e^{-\frac{\pi}{12}i}\}$, the closed-loop system satisfies the requirements if $0.703 \leq r \leq 0.707$.



(a) (b)

Fig. 5. Helicopter responses of the Pole-placement control. (a) System responses with measurement errors of Section 5.2 and (b) System responses with mode switches of Section 5.3 in every step.

Figure 5 (a) shows the responses of the system when $r = 0.715$. Because r is not in the safe range of $[0.703, 0.707]$, the model checker reported a sequence of measurement errors $eq(t)$, $ep(t)$, and $ev(t)$ in the counterexample. Figure 5 (a) is plotted when these error values were added to the states in the control law. Observe that at time 15, the velocity is slightly below 0.9 and the requirement is violated.

5.3 Controller Design with Nondeterministic Dynamics Changes

Unexpected system dynamics change poses a nontrivial challenge to a controller design. Some examples of such systems are: an airplane flying at different altitudes shows different aerodynamics, a car changes its dynamics after shifting the transmission gear, the ADME process of a drug shows different drug kinetics when the drug molecules outnumber the enzymes, and so on. It is desirable or even critical to design a controller that can maintain its required properties regardless of the system dynamics changes. In this section, we will demonstrate how LTLC model checking can help design a controller that is robust against the dynamics changes.

Some smart sensors perform measurements at a specific interval and keep the results in a buffer. On data request, they return the buffered value without making a new measurement. Suppose that the sensors of the helicopter example sample 10 times during a 100 *msec* discretization interval. Then, depending on when the last measurement is buffered and when the data request is made, the data can be as much as 10 *msec* old. Furthermore, if the data requests and the data measurements are closely aligned, the readings can be either without any delay or 10 *msec* stale. To the controller, this buffering effect may look like a system dynamics change. We model this dynamics change by

a hybrid system with two modes: one for the normal sampling and one for the 10 *msec* hasty sampling.

Discrete-time system dynamics are obtained from their continuous-time counterparts by the ZOH sampling. Let $\tilde{\mathbf{u}} : \mathbb{R} \to \mathbb{R}^{nu}$ and $\tilde{\mathbf{x}} : \mathbb{R} \to \mathbb{R}^{nx}$ be continuous-time trajectories for the input and state variables respectively, $\mathbf{u}' : \mathbb{N} \to \mathbb{R}^{nu}$ and $\mathbf{x}' : \mathbb{N} \to \mathbb{R}^{nx}$ be their discrete-time counterparts, and let the continuous-time dynamics equations of an LTI system be

$$\frac{d}{dt}\tilde{\mathbf{x}}(t) = F \cdot \tilde{\mathbf{x}}(t) + G \cdot \tilde{\mathbf{u}}(t).$$

Let δ be a buffering delay, T be the discretization interval, and $T' = T - \delta$, then the discrete-time model has the following dynamics equations.

$$\begin{aligned}\tilde{\mathbf{x}}((t+1)\cdot T') &= e^{F\cdot T'} \cdot \tilde{\mathbf{x}}(t \cdot T') + \left(e^{F\cdot T'} \cdot \int_0^{T'} e^{-F\cdot\tau} d\tau \cdot G\right) \cdot \tilde{\mathbf{u}}(t \cdot T') \\ &= A' \cdot \tilde{\mathbf{x}}(t \cdot T') + B' \cdot \tilde{\mathbf{u}}(t \cdot T'), \\ \mathbf{x}'(t+1) &= A' \cdot \mathbf{x}'(t) + B' \cdot \mathbf{u}'(t).\end{aligned}$$

However, because the actual state trajectory is still governed by Equation (5), and the buffered measurement is from the actual state $\mathbf{x}(t)$,

$$\mathbf{x}'(t+1) = A' \cdot \mathbf{x}(t) + B' \cdot \mathbf{u}(t). \tag{6}$$

That is, we need both $\mathbf{x}'$ and $\mathbf{x}$ state vectors in the model checking. Because doubling the state space significantly slows down the model checking process (it quadruples A matrix and doubles B and C matrices), we reformulated Equation (6) to make $\mathbf{x}'$ an output vector as below.

$$\begin{aligned}\begin{bmatrix}\mathbf{x}'(t+1)\\ \mathbf{u}(t)\end{bmatrix} &= \begin{bmatrix}A' & B'\\ \mathbf{0} & \mathbf{1}\end{bmatrix} \cdot \begin{bmatrix}\mathbf{x}(t)\\ \mathbf{u}(t)\end{bmatrix} = \begin{bmatrix}A' & B'\\ \mathbf{0} & \mathbf{1}\end{bmatrix} \cdot \begin{bmatrix}A & B\\ \mathbf{0} & \mathbf{1}\end{bmatrix}^{-1} \cdot \begin{bmatrix}A & B\\ \mathbf{0} & \mathbf{1}\end{bmatrix} \cdot \begin{bmatrix}\mathbf{x}(t)\\ \mathbf{u}(t)\end{bmatrix} \\ &= \begin{bmatrix}A' & B'\\ \mathbf{0} & \mathbf{1}\end{bmatrix} \cdot \begin{bmatrix}A & B\\ \mathbf{0} & \mathbf{1}\end{bmatrix}^{-1} \cdot \begin{bmatrix}\mathbf{x}(t+1)\\ \mathbf{u}(t)\end{bmatrix} = C' \cdot \begin{bmatrix}\mathbf{x}(t+1)\\ \mathbf{u}(t)\end{bmatrix},\end{aligned}$$

where $\mathbf{0} \in \mathbb{R}^{2\times 3}$ is a matrix of zeros and $\mathbf{1} \in \mathbb{R}^{2\times 2}$ is the identity matrix. C' exist if the system is *controllable*. Let an extended state vector be $\mathbf{x}(t) = [q(t), p(t), v(t), w(t), r'(t)]$, and an output vector be $\mathbf{y}(t) = [q'(t), p'(t), v'(t)]$, where $r'(t)$ is the previous value of $r(t)$, and $q'(t)$, $p'(t)$, and $v'(t)$ are the measured pitch rate, pitch angle, and velocity respectively. Then, the dynamics equations are

$$\mathbf{x}(t+1) = A \cdot \mathbf{x}(t) + B \cdot \mathbf{u}(t), \;\; \mathbf{y}(t) \;=\; C \cdot \mathbf{x}(t), \tag{7}$$

where A and B are extended from Equation (5) to accommodate the new state variable r' with the dynamics $r'(t+1) = r(t)$, and C is the first 3 rows of C'. The elements of C matrix are in the mode s of Figure 4.

A hybrid system model can be built as below using the two modes (a normal mode n and a hasty mode s).

$$\begin{aligned}H &= \langle U, Y, X, \{n, s\}, \{n \longrightarrow n, n \longrightarrow s, s \longrightarrow n, s \longrightarrow s\}\rangle, \\ n &= \langle U, Y, X, A, B, I, 0\rangle, \;\; s = \langle U, Y, X, A, B, C, 0\rangle,\end{aligned}$$

where $U = \{r, \ell\}$, $Y = \{q', p', v'\}$, and $X = \{q, p, v, w, r'\}$.

The LTLC specification for the control law is

$$\phi_c' = \Box\,(\phi_c), \text{ where } \phi_c \;=\; (r = N \cdot \ell - Kq \cdot q' - Kp \cdot p' - Kv \cdot v').$$

Using the new control law ϕ_c, the LTLC formula to check is

$$\phi = \phi_i \rightarrow \neg(\,(\phi_c \wedge \phi_\ell)\,\mathtt{U}\,\neg(\phi_o \wedge \phi_a \wedge \phi_s)\,).$$

Like the previous examples, when the locations of the poles are $\{r, r{\cdot}e^{\frac{\pi}{12}i}, r{\cdot}e^{-\frac{\pi}{12}i}\}$, the closed-loop system satisfies the requirements if $0.700 \leq r \leq 0.715$. Figure 5 (b) shows the trajectories of input, output, and state variables when $r = 0.715$ and the system mode switches between n and s at every step. The figure shows that the closed-loop system satisfies the requirements despite the mode switches.

6 Acknowledgement

This work was supported by MSIP, Korea under the ITCCP program (IITP-2015-R0346-15-1007) and by KEIT under the GATC program (10077300).

7 Conclusion

Using the LTLC model checking technique, required properties about a control system can be formally specified and the controller can be designed to satisfy the requirements. We proposed a controller design method guided by LTLC model checking results and showed that the resulting closed-loop system is robust against errors not just stochastically disturbing the system but also systematically hampering it as well.

One of the missing elements in the proposed technique is the state observer. Instead of adding measurement noise directly to the state variables, we are working on extending the model checking technique to incorporate state observers like a Kalman filter [21]. We are expecting a more efficient controller design using the probability distributions of errors.

With the advances in CPS and IoT technologies, computer systems will interact more tightly with physical systems. Because the behaviors of physical systems are commonly modeled by differential equations, a natural way to interface cyber systems and physical systems is a logic that can handle quantitative values. We believe many quantitative techniques, including the proposed LTLC model checking technique, will produce many fruitful results.

References

1. LTLC-Checker: https://sites.google.com/site/youngminkwon.
2. R. Alur and D. L. Dill. A theory of timed automata. In *Theoretical Computer Science*, volume 126, pages 183–235. Elsevier Science Publishers Ltd., April 1994.

3. A. Biere, A. Cimatti, E. Clarke, and Y. Zhu. Symbolic model checking without BDDs. In *Tools and Algorithms for the Construction and Analysis of Systems*, volume 1579, pages 193–207, 1999.
4. L. Bu, Y. Li, L. Wang, and X. Li. BACH: Bounded reachability checker for linear hybrid automata. In *Formal Methods in Computer Aided Design*, pages 65–68. IEEE Computer Society, 2008.
5. E. Clarke, O. Grumberg, and D. Peled. *Model Checking*. MIT Press, 2000.
6. G. F. Franklin, J. D. Powell, and A. Emami-Naeini. *Feedback Control of Dynamic Systems*. Addison Wesley, 3rd edition, 1994.
7. M. Fränzle and C. Herde. HySAT: An efficient proof engine for bounded model checking of hybrid systems. In *Formal Methods in System Design*, volume 30, pages 179–198, 2007.
8. A. Girard, G. Pola, and P. Tabuada. Approximately bisimilar symbolic models for incrementally stable switched systems. In *Transactions on Automatic Control*, volume 55, pages 116–126. IEEE, 2010.
9. T. A. Henzinger. The theory of hybrid automata. In *Annual Symposium on Logic in Computer Science*, pages 278–292. IEEE Computer Society, 1996.
10. T. A. Henzinger, P.-H. Ho, and H. Wong-Toi. HyTech: A model checker for hybrid systems. In *Computer Aided Verification*, pages 460–463. Springer, 1997.
11. G. J. Holzmann. The model checker spin. In *IEEE Transactions on Software Engineering*, volume 23, pages 279–295, May 1997.
12. M. Kloetzer and C. Belta. A fully automated framework for control of linear systems from temporal logic specifications. In *Transactions on Automatic Control*, volume 53, pages 287–297. IEEE, 2008.
13. H. Kress-Gazit, G. E. Fainekos, and G. J. Pappas. Temporal-logic-based reactive mission and motion planning. In *Transactions on Robotics*, volume 25, pages 1370–1381. IEEE, 2009.
14. Y. Kwon and G. Agha. LTLC: Linear temporal logic for control. In *Hybrid Systems: Computation and Control*, volume LNCS 4981, pages 316–329. Springer Verlag, Berlin Heidelberg, 2008.
15. Y. Kwon and E. Kim. Bounded model checking of hybrid systems for control. In *IEEE Transactions on Automatic Control*, volume 60, pages 2961–2976. IEEE, 2015.
16. Y. Kwon, E. Kim, S. Jeong, and A. Lee. Quantitative model checking for a smart grid pricing. In *International Conference on the Quantitative Evaluation of Systems (QEST)*, pages 55–71. IEEE, 2017.
17. K. G. Larsen, P. Pettersson, and W. Yi. UPPAAL in a nutshell. In *Int. Journal on Software Tools for Technology Transfer*, volume 1, pages 134–152, Oct 1997.
18. J. Liu, N. Ozay, U. Topcu, and R. M. Murray. Synthesis of reactive switching protocols from temporal logic specifications. In *Transactions on Automatic Control*. IEEE, 2013.
19. D. G. Luenberger. *Linear and Nonlinear Programming*. Addison Wesley, 2nd edition, 1989.
20. A. V. Oppenheim and A. S. Willsky. *Signals and Systems*. Prentice-Hall, 1983.
21. H. Start and J. W. Woods. *Probability and Random Processes with Applications to Signal Processing*. Prentice-Hall, 3rd edition, 2002.
22. P. Tabuada and G. J. Pappas. Linear time logic control of discrete-time linear systems. In *Transactions on Automatic Control*, volume 51, pages 1862–1877. IEEE, 2006.