Safety Verification of Pitch Controller for Model Helicopter*

Sayan Mitra[1†] Yong Wang[2‡] Nancy Lynch[1§]
Eric Feron[2¶]

[1]MIT Laboratory for Computer Science,

Cambridge, MA 02139, USA

{mitras,lynch}@theory.lcs.mit.edu

[2]MIT Laboratory for Information and Decision Systems,

Cambridge , MA 02139, USA

{y_wang, feron}@mit.edu

July 8, 2002

Abstract A supervisory controller for ensuring the safety along the pitch axis of a model helicopter system with three degrees of freedom is proposed and verified. Unsafe states are avoided by conservatively estimating the worst possible scenario that may arise under the user's controller and switching to a supervisory controller under appropriate conditions. The supervisory controller makes the system safe by performing recovery actions. It is observed that the limitations on the actuator magnitude and bandwidth, sensor inaccuracies and finite sampling rate constrain the user mode operations to a smaller region. The hybrid I/O automaton framework is used to model and verify the system components. The safety of the system is proved, assuming sufficiently small actuator delay and sampling period

*Funding for this research has been provided by AFRL contract F33615-01-C-1850
†
‡
§
¶

1 Introduction

Formal verification of hybrid systems is a hard problem. It has been shown that checking reachability for a simple subclass of hybrid automata is undecidable [3]. Algorithmic verification techniques have been developed for a several restricted subclasses of hybrid automata [1] but these subclasses are too weak to represent real-life hybrid system models. An alternative approach to verification has evolved based on the hybrid Input/Output automaton (HIOA) [8, 9, 7] model. The HIOA model is expressive enough to model real-life hybrid systems as it imposes very few restrictions on the evolution of the continuous variables. In this approach the properties of automata are proved by inductively reasoning about their executions, see [5, 12, 6] for related earlier works.

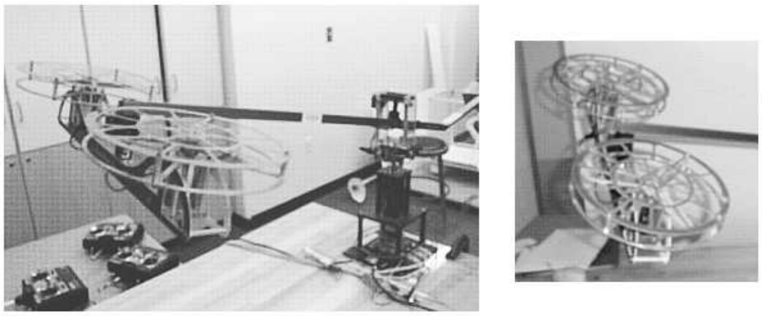

Figure 1: Helicopter model with three degrees of freedom.

This paper discusses the verification of a supervisory controller for the pitch axis of a model helicopter system using the HIOA framework. The helicopter system used here (Figure 1) is manufactured by Quanser [4]. It is driven by two rotors mounted at the two ends of its body and it is attached to an arm which is fixed at one end. So the helicopter can revolve about the fixed end of the arm and has three degrees of freedom. The rotor inputs are controlled by a joystick or through controller software designed by users. A controller can behave arbitrarily and may damage the helicopter by pitching it too high or too low. At MIT students experiment with different controllers for these systems and hence its safety is critical. To prevent the helicopter from reaching unsafe states, a supervisory controller is introduced which overrides the user's controller by predicting the worst possible scenario which may arise if the user is allowed to continue. Like actual flight control systems our helicopter-supervisor system presents saturations on the actuator bandwidth, limitation on sampling rate and sensor inaccuracies.

The main contribution of this paper is a supervisory controller which ensures safety along the pitch axis. The supervisory controller preserves safety by monitoring the state of the helicopter and performing automatic mode switches. The helicopter, sensors, actuator and the user and supervisory controllers are modeled as HIOA components. And the safety property is proved by induction on the length of an execution of the composed automaton.

This paper also illustrates a set of language constructs for specifying hybrid systems. As mentioned earlier, the algorithmic verification methods for hybrid systems apply to simple and restricted classes of hybrid automata, consequently the specification languages, like HyTech [2], developed for these methods are not adequately expressive for describing general hybrid systems. In this work, by adding extra structure to the HIOA model we are able to specify the continuous evolution of automata in terms of constrained state models while the discrete transitions are specified in the usual precondition-effect style. The details of the specification language can be found in [10].

The rest of the paper is organized as follows: in Section 2 the hybrid I/O automata model is reviewed and the restrictions we have introduced on the model are described. HIOA models of the helicopter, sensor, actuator and the user controller are presented in Section 3. In Section 4 the design of the supervisory controller is developed. Analysis and verification of the supervisory controller is presented in Section 5. Concluding remarks are made in Section 6.

2 Hybrid I/O Automata

In this section we briefly review the HIOA mathematical model, for a complete discussion of the model refer to [7]. Earlier versions of the model appeared in [8] and [9].

2.1 The Mathematical Model

A hybrid I/O automaton captures the hybrid behavior of a system in terms of discrete transitions and continuous evolution of its state variables. Let V be the set of variables of automaton $\mathcal{A}$. Each $v \in V$ is associated with a *(static) type* defining the set of values v can assume. A *valuation* $\mathbf{v}$ for V is a function that associates each variable $v \in V$ to a value in $type(v)$. A trajectory τ of V is defined as a mapping $\tau : J \rightarrow val(V)$ where J is a left closed interval of time. If J is a closed interval then τ is *closed* and its *limit time* is the supremum of the domain of τ, also written as $\tau.ltime$. Each variable $v \in V$ is also associated with a *dynamic type* (or *dtype*) which is the set of trajectories that v may follow. Dynamic types must satisfy the *time-shift, subinterval* and *pasting* closure properties described in [7]. A hybrid I/O automaton $\mathcal{A}$ is a tuple $(X, U, Y, Q, \Theta, H, I, O, \mathcal{D}, \mathcal{T})$ where

- X: set of *internal or state variables*, U : set of *input variables*, Y : set of *output variables*. The set of variables $V \triangleq U \cup Y \cup X$. The set of locally controlled variables $Z \triangleq X \cup Y$.

- H : set of *internal actions*, I : set of *input actions* , O : set of *output actions*. The set of actions $A \triangleq H \cup I \cup O$.
- $Q \subseteq val(X)$: a set of states
- $\Theta \subseteq Q$: non-empty set of *start states*.
- $\mathcal{D} \subseteq Q \times A \times Q$: set of *discrete transitions*. A transition $(\mathbf{x}, a, \mathbf{x}') \in \mathcal{D}$ is written in short as $\mathbf{x} \xrightarrow{a}_{\mathcal{A}} \mathbf{x}'$.
- $\mathcal{T}$: set of *trajectories* for V, such that for every trajectory τ in $\mathcal{T}$, and for every $t \in \tau.dom$, $\tau(t).X \in Q$. It is required that the set of trajectories $\mathcal{T}$ is closed under prefix, suffix, and concatenation. The first state $\tau(0).X$ of trajectory τ is written as $\tau.fstate$. Similarly if $\tau.dom$ is finite then $\tau.lstate = \tau(\tau.ltime).X$.

In addition, a hybrid I/O automaton also satisfies: (1) the input action enabling property, which prevents it from blocking any input action and (2) the input trajectory enabling property, which ensures that it is able to accept any trajectory of the input variables either by allowing time to progress for the entire length of the trajectory or by reacting with some internal action before that. A trajectory in which time does not progress is called a *point trajectory.*

An *execution* of a hybrid I/O automaton $\mathcal{A}$ is a finite or infinite sequence of actions and trajectories $\zeta = \tau_0, a_1, \tau_1, a_2 \ldots$,where (1) each $\tau_i \in \mathcal{T}$, (2) $\tau_0.fstate \in \Theta$ and (3) if τ_i is not the last trajectory in ζ then $\tau_i.lstate \xrightarrow{a_{i+1}} \tau_{i+1}.fstate$. An execution is *closed* if the sequence is finite and the domain of the final trajectory is a finite closed interval. The length of an execution is the number of elements in the sequence (actions plus trajectories).

2.2 Specification Language

We add the following extra structure to the model described in [7] and specify the set of trajectories $\mathcal{T}$ using constrained state models.

R1 The time domain is R. A variable v is discrete (continuous) if its dynamic type is the pasting closure of constant (continuous) functions from left closed intervals of time to *type*(v). For any set of variables S, we refer to the discrete and continuous subsets as S_d and S_a respectively.

R2 Locally controlled discrete variables remain constant over the trajectories, i.e., for all $\tau \in \mathcal{T}$, $\tau.lval\lceil Z_d = \tau.fval\lceil Z_d$.

For this restricted class of hybrid I/O automata, we can use the state model representation of the dynamics to specify the set of trajectories of an automaton. The following part explains our representation.
Let e be any expression involving the variables in V. For a given trajectory τ we use $\tau.e$ to denote the function with domain $\tau.dom$ that gives the value of the expression e at all times during trajectory τ. A trajectory τ satisfies the algebraic equation

$$v = e,$$

if for every $t \in \tau.dom$, $\tau \downarrow v(t) = \tau.e(t)$. An algebraic equation could be a nondeterministic choice. A trajectory τ satisfies the nondeterministic choice

$$v = \textbf{choose } [e_1, e_2],$$

if for every $t \in \tau.dom$, $\tau \downarrow v(t) \in [\tau.e_1(t), \tau.e_2(t)]$. Similarly, if the expression e is integrable when viewed as a function, then τ satisfies the differential equation

$$d(v) = e,$$

if for every $t \in \tau.dom$, $\tau \downarrow v(t) = \tau \downarrow v(0) + \int_0^t \tau.e(t')\, dt'$. A *state model* of $\mathcal{A}$ consists of $|Z_a|$ number of independent algebraic and/or differential equations with every variable v in Z_a appearing in the left hand side of exactly one of the equations as v or $d(v)$. The right hand sides of the equations are either constants or algebraic and trigonometric expressions involving the variables in V. A state model specifies the evolution of every variable v in Z_a from some initial valuation of the variables. A trajectory τ satisfies a state model E if at all times in $\tau.dom$, all the variables in Z_a satisfy the differential and algebraic equations in E with $\tau(0)$ defining the initial valuations.

The set of trajectories $\mathcal{T}$ is specified by the activities of $\mathcal{A}$. An *activity* α is a triple (P, P^+, E), where P and P^+ are sets of states and E is a state model. Activity α defines the set of trajectories that satisfy the following:

- τ satisfies the state model E.
- If $\tau \downarrow X(t) \in P^+$ for $t \in dom(\tau)$ then τ is closed and $t = \tau.ltime$.
- For all $t \in \tau.dom - \{0\}$, $\tau \downarrow X(t) \in P$.

In our specification language an activity $\alpha : (P, P^+, E)$ is written as:

activity α $\quad P$ **evolve** E **postcon** P^+.

If P^+ is not specified explicitly then it is assumed to be false, while P is assumed to be true.

3 Specification of System Components

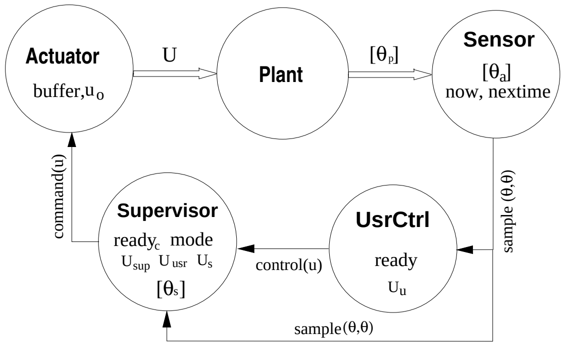

Figure 2: Components of Helicopter system. Continuous and discrete communication between components are shown by wide and thin arrows respectively. The internal variables are shown within the circles corresponding to each component.

In this section we present a HIOA model of the helicopter system, except for the supervisory controller, which is in Section 4; the interaction among the different components of the system are shown in Figure 2. A nonlinear dynamical model of the helicopter with three degrees of rotational freedom can be found in [11]. In this paper we consider the pitch dynamics, which are critical for safety. The roll and yaw effects are eliminated by making the initial conditions and the disturbances along these axes to be zero and giving identical input to the two rotors. The pitch dynamics is described by the following differential equation :

$$\ddot{\theta} + \Omega^2 \cos\theta = U(t), \tag{1}$$

where Ω is the rotational inertia and U is the net input for the pitch axis. The `Plant` automaton specifies the evolution of the pitch angle (θ_p^0) and velocity (θ_p^1) of the helicopter in terms of the input U. We define global types `RAD, RADPS` and `UTYPE` for the range of values for variables representing angle, angular velocity and actuator output respectively. The constant $\Theta(\dot{\Theta})$ corresponds to the largest absolute value of any variable representing angle (angular velocity). The state variables θ_p^0 and θ_p^1 are initialized to some value from the set $\mathbf{U}$, which is defined in equation (5). The activity `pitch_dynamics` describes the evolution of the locally controlled continuous variables. The state variables evolve according to equation (1) and the output variables θ_e^0 and θ_e^1 are simply copied from the internal variables.

```
type RAD = Real  suchthat (i : RAD, |i| ≤ Θ)
type RADPS = Real  suchthat (i : RADPS, |i| ≤ Θ̇)
type UTYPE = Real  suchthat (i : UTYPE |  U_min ≤ i ≤ U_max)

hybridautomaton Plant(Ω : Real )

variables
    input analog U : UTYPE,
    internal analog θ_p^0 : RAD, θ_p^1 : RADPS, initially (θ_p^0, θ_p^1) ∈ U,
    output analog θ_e^0 : RAD, θ_e^1 : RADPS

trajectories
    activity pitch_dynamics
    evolve d(θ_p^0) = θ_p^1;
           d(θ_p^1) = −Ω^2 cos θ_p^0 + U;
           θ_e^0 = θ_p^0; θ_e^1 = θ_p^1
```

Figure 3: HIOA specification of the plant

The `Plant` is said to be safe at a given state if the pitch angle θ_p^0 is within the allowed limits θ_{min} and θ_{max}. We define the set of safe states as:

$$\mathbf{S} \triangleq \{s \mid \theta_{min} \leq s.\theta_p^0 \leq \theta_{max}\} \tag{2}$$

The `Sensor` automaton models the sensing devices and the A/D conversion circuitry. It is parameterized by the sensor errors (ϵ_0 and ϵ_1) for pitch angle and velocity, and the sampling period (Δ). The input variables (θ_e^0, θ_e^1) are continuously copied into the internal state (θ_a^0, θ_a^1) of the automaton and the local time *now* advances at a constant rate. After every Δ interval of time an output action *sample*(θ_d^0, θ_d^1) is generated. The value of θ_d^0 (θ_d^1) is nondeterministically chosen to be within $\pm\epsilon_0$ ($\pm\epsilon_1$) of θ_a^0 (θ_a^1). This nondeterminism models the noise or the uncertainties in the sensing devices.

The specification of the `UsrCtrl` automaton is shown in Figure 5, it models an arbitrary user controller. This automaton reads the *sample* action as input and triggers an output *control*(u_d) action, which communicates the output of the user controller (U_u) to the supervisor. The output decision U_u of the user controller is modeled as a nondeterministic choice over the entire range of possible values. This captures our assumption that the user is capable of issuing arbitrarily bad inputs to the plant. The design of a safe supervisor for this particular model of `UsrCtrl` ensures that the system is safe for any given user because every user controller must implement this specification of `UsrCtrl`. On the other hand the supervisor designed to handle `UsrCtrl` may be too restrictive for user controllers that are better behaved than the above.

Next, we present the specification of the `Actuator` automaton, which models the actuator and the D/A converter. The automaton has two parameters: the effective actuator delay $\tau_{\mathtt{eff}}$ and the sampling period Δ.

hybridautomaton Sensor(ϵ_0,ϵ_1, Δ : Real)

actions
  output *sample* (θ_d^0 : RAD , θ_d^1: RADPS)

discrete transitions
  output *sample* (θ_d^0 , θ_d^1)
  pre $now = next\_time \wedge$
    $\theta_d^0 \in [\theta_a^0 - \epsilon_0, \theta_a^0 + \epsilon_0] \wedge$
    $\theta_d^1 \in [\theta_a^1 - \epsilon_1, \theta_a^1 + \epsilon_1]$
  eff $next\_time := now + \Delta$

trajectories
  activity *read*
  evolve d(now) =*1*;
    $\theta_a^0 = \theta_e^0$; $\theta_a^1 = \theta_e^1$;
  postcon $now = next\_time$

variables
  input analog θ_e^0 : RAD, θ_e^1 : RADPS,
  internal analog θ_a^0 : RAD, θ_a^1 : RADPS, now:Real ,
    initially all set to 0
  internal $next\_time$: Real $:= \Delta$

Figure 4: HIOA specification of the sensor and A/D conversion circuit

hybridautomaton UsrCtrl

actions
  input *sample* (θ_d^0 : RAD , θ_d^1 : RADPS),
  output *control* (u_d : UTYPE)

discrete transitions
  input *sample* (θ_d^0 , θ_d^1)
  eff $\theta_u^0 := \theta_d^0$; $\theta_u^1 = \theta_d^1$
    $U_u :=$ **choose** $[U_{min}, U_{max}]$;
    $ready :=$ true

variables
  internal θ_u^0: RAD , θ_u^1 : RADPS, U_u : UTYPE := 0,
    initially all set to 0
    $ready$: Bool := false

  output *control* (u_d)
  pre $(u_d = U_u) \wedge ready$
  eff $ready :=$ false

Figure 5: Specification of User's Controller

We have abstracted the actual delay $\tau_{\texttt{act}}$ in the actuators with its over approximation $\tau_{\texttt{eff}} = \lceil \frac{\tau_{\texttt{act}}}{\Delta} \rceil \Delta$. The delay in the actuator response is modeled as a FIFO queue called *buffer* of length $\frac{\tau_{\texttt{eff}}}{\Delta}$. The occurrence of *command(u_d)* appends u_d to *buffer* and removes the head element after copying its value to u_o.

4 Supervisory Controller

In this section we discuss the design principles for the supervisor and specify the supervisory controller. The supervisory controller has to ensure that the `plant` state stays in the safe region **S** defined in equation (2). A second requirement of the supervisor is to interfere as little as possible with the user's controller. In the next section we informally discuss the relevance of several different regions in the state space, their actual definitions appear in the following section.

```
hybridautomaton Actuator(τ_eff, Δ : Real )

actions
    input command ( u_d : UTYPE )

discrete transitions
    input command ( u_d )
    eff buffer + := (u_d, now)
        u_o := buffer.head[1];
        buffer := buffer.tail

trajectories
    activity d2a
    evolve U = u_o

variables
    input analog now : Real ,
    output analog U : UTYPE := 0,
    internal u_o : UTYPE := 0,
        buffer : sequence of UTYPE ×Real  of length τ_eff/Δ,
        initially all elements are (0,0)'s
```

Figure 6: Actuator and D/A conversion

4.1 Supervisor Strategy

The design of the supervisor is based on a simple principle: let the user drive the plant as long as the supervisor is guaranteed to be able to restore the plant to a safe state using maximum recovery force; otherwise block the user, perform recovery, and return control to the user. The key design issue is to decide the largest region in which the supervisor is guaranteed to restore the plant to a safe state, we call this the safe operating region for the user.

In a world where the plant state is instantaneously and correctly available to the supervisor and the actuator is without any delay and saturation, the safe operating region could be **S** itself. However, in reality this does not work. For example when the pitch angle is close to θ_{min}, a large downward velocity is unsafe because the helicopter cannot come to a halt instantaneously even with the maximum upward pull from the actuator. We define a *controllable* region **C**, such that a state s is in **C** if all reachable states from s are contained in **S**, assuming the actuator is turned on correctly at s and it responds instantaneously. The region **C** could be a safe operating region if the supervisor could change the output from the actuator instantaneously. As the actuator responds with a delay of $\tau_{\mathtt{eff}}$, issuing recovery commands at the boundary of **C** would be too late for the plant. So we introduce a smaller *recoverable region* **R** which is defined as the set of states from which all reachable states over a period of $\tau_{\mathtt{eff}}$ are within **C**. It turns out that even **R** is not suitable as the switching region because the supervisor is also limited by the sampling frequency (Δ) and the inaccuracy of the sensors. The supervisor receives updates about the state of the plant and issues *command* actions periodically with an interval Δ. So, at a given point of observation the supervisor has to decide for all future possibilities over the next Δ interval of time. Moreover, for a given state observed by

`Sensor` (θ_s^0, θ_s^1) the possible actual state of the plant is given by:

$$P(s) = \{s \mid \theta_s^0 - \epsilon_0 \leq s.\theta_p^0 \leq \theta_s^0 + \epsilon_0 \wedge \theta_s^1 - \epsilon_1 \leq s.\theta_p^1 \leq \theta_s^1 + \epsilon_1\}$$

Finally we define the *user control* region $\mathbf{U}$, at the boundary of which the supervisor takes over control from the user. An observed state s is in $\mathbf{U}$ if starting from any state in $P(s)$ all the reachable states over a Δ interval of time are in $\mathbf{R}$.

Switching back to the user's controller is delayed until the supervisor brings the state within a *inner* region $\mathbf{I}$. This asymmetry in the switching is introduced to prevent high frequency chattering between the user and the supervisory controller.

4.2 Regions of Control

In the previous section we informally described the intuitive meanings of the regions $\mathbf{C}$, $\mathbf{R}$, $\mathbf{U}$, and $\mathbf{I}$; now we present their precise definitions.

$$\begin{aligned}
\mathbf{C} &\triangleq \{s \mid \theta_{min} \leq s.\theta_p^0 \leq \theta_{max} \wedge \Gamma^-(s.\theta_p^0, 0) \leq s.\theta_p^1 \leq \Gamma^+(s.\theta_p^0, 0)\}, && (3)\\
\mathbf{R} &\triangleq \{s \mid \theta_{min} \leq s.\theta_p^0 \leq \theta_{max} \wedge \Gamma^-(s.\theta_p^0, \tau_{eff}) \leq s.\theta_p^1 \leq \Gamma^+(s.\theta_p^0, \tau_{eff})\}, && (4)\\
\mathbf{U} &\triangleq \{s \mid \theta_{min} + \epsilon_0 \leq s.\theta_s^0 \leq \theta_{max} - \epsilon_0 \ \wedge U^-(s.\theta_s^0) \leq s.\theta_s^1 \leq U^+(s.\theta_s^0)\}, && (5)\\
\mathbf{I} &\triangleq \{s \mid \theta_{min} + \epsilon_0 \leq s.\theta_s^0 \leq \theta_{max} - \epsilon_0 \ \wedge I^-(s.\theta_s^0) \leq s.\theta_s^1 \leq I^+(s.\theta_s^0)\}. && (6)
\end{aligned}$$

The regions are defined in terms of the functions Γ^+ and Γ^-, which are as follows:

$$\begin{aligned}
\Gamma^+(\theta, \mathcal{T}) &= -U_{mag}\mathcal{T} + \sqrt{2(\Omega^2 \cos\theta_{max} - U_{min})(\theta_{max} - \theta + \frac{1}{2}U_{mag}\mathcal{T}^2)}, && (7)\\
\Gamma^-(\theta, \mathcal{T}) &= U_{mag}\mathcal{T} - \sqrt{2(U_{max} - \Omega^2)(\theta - \theta_{min} + \frac{1}{2}U_{mag}\mathcal{T}^2)}, && (8)\\
U_{mag} &= U_{max} - U_{min} && (9)\\
U^+(\theta) &= -\epsilon_1 + \Gamma^+(\theta + \epsilon_0, \tau_{eff} + \Delta), && (10)\\
U^-(\theta) &= +\epsilon_1 + \Gamma^-(\theta - \epsilon_0, \tau_{eff} + \Delta). && (11)\\
I^+(\theta) &= -2\epsilon_1 + \Gamma^+(\theta + 2\epsilon_0, \tau_{eff} + \Delta), && (12)\\
I^-(\theta) &= +2\epsilon_1 + \Gamma^-(\theta - 2\epsilon_0, \tau_{eff} + \Delta). && (13)
\end{aligned}$$

From the definitions of Γ^+ and Γ^- the following properties can be shown,

Property 1 *Over the interval* $-\frac{\pi}{2} \leq \theta \leq \frac{\pi}{2}$ *the following properties hold :*

1. *$\Gamma^+(\theta, \mathcal{T})$ and $\Gamma^-(\theta, \mathcal{T})$ are monotonically decreasing with respect to θ.*
2. *$\Gamma^+(\theta, \mathcal{T})$ is monotonically decreasing with respect to $\mathcal{T}$. ($\mathcal{T} \geq 0$).*
3. *$\Gamma^-(\theta, \mathcal{T})$ is monotonically increasing with respect to $\mathcal{T}$. ($\mathcal{T} \geq 0$).*
4. *$\Gamma^+(\theta_{max}, \mathcal{T}) < 0$ and $\Gamma^-(\theta_{min}, \mathcal{T}) > 0$ for $\mathcal{T} > 0$.*

Using these properties, the following sequence of containments can be proved.

Lemma 1 $\mathbf{I} \subseteq \mathbf{U} \subseteq \mathbf{R} \subseteq \mathbf{C} \subseteq \mathbf{S}$

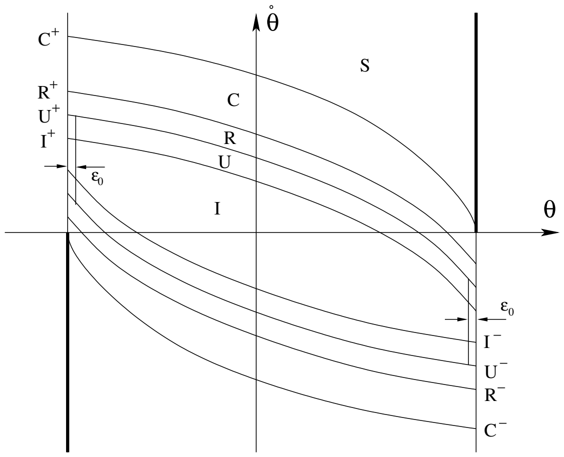

Figure 7: Regions in the statespace.

4.3 Supervisor Automaton

In Figure 8 the specification of the `supervisor` automaton is given. It receives copies the observed plant state into internal variables θ_s^0 and θ_s^1 when the *sample* action occurs. Based on this state information the tentative recovery U_{sup} is decided. When the *control* action takes place, the supervisor copies the user's command into another internal variable U_{usr} and decides the output command U_s and the *mode* for the next Δ interval based on (θ_s^0, θ_s^1) and the current *mode*. If *mode* is `usr` and the observed state is in $\mathbf{U}$ then *mode* remains unchanged and the output U_s is set to U_{usr}. If the present state is not in $\mathbf{U}$ then *mode* is changed to `sup` and the output U_s is set to U_{sup}. Similarly, if *mode* = `sup` then U_s is copied from U_{sup} and the mode changes only when (θ_s^0, θ_s^1) is in $\mathbf{I}$. The *control* action enables the output action *command* by setting the $ready_c$ flag.

hybridautomaton Supervisor

actions
input *sample* (θ_d^0: RAD θ_d^1:RADPS),
input *control* (u_d : UTYPE),
output *command* (u_d : UTYPE)

variables
internal θ_s^0 : RAD, θ_s^1 : RADPS, U_{sup}, U_{usr}, U_s : UTYPE,
initially all are 0
internal $ready_c$: Bool $:=$ false, $mode$: {sup, usr} $:=$ usr
internal analog rt : Real $:= 0$;

discrete transitions
input *sample* (θ_d^0, θ_d^1)
eff $\theta_s^0 := \theta_d^0$; $\theta_s^1 := \theta_d^1$;
if $\theta_s^1 \geq U^+(\theta_s^0)$ **then** $U_{sup} := U_{min}$
elseif $\theta_s^1 \leq U^-(\theta_s^0)$ **then** $U_{sup} := U_{max}$
else $U_{sup} := 0$ **fi** ;

output *command* (u_d)
pre $ready_c \wedge (u_d = U_s)$
eff $ready_c :=$ false

input *control* (u_d)
eff $U_{usr} := u_d$; $ready_c :=$ true
if $mode =$ usr **then**
if $(\theta_s^0, \theta_s^1) \in$ **U** **then** $U_s := U_{usr}$
else $U_s := U_{sup}$; $mode :=$ sup **fi**
elseif $mode =$ sup **then**
if $(\theta_s^0, \theta_s^1) \in$ **I** **then** $U_s := U_{usr}$; $mode :=$ usr
else $U_s := U_{sup}$ **fi fi**

trajectories
activity *timer*
evolve if $mode =$ sup
then $d(rt) = 1$ **else** $rt = 0$ **fi**
postcon $ready_c$

Figure 8: HIOA specification of supervisor automaton

5 Analysis of Helicopter System

In this section we verify the safety of the composition of all the components in the helicopter system. We name the composed automaton as $\mathcal{A}$. According to our definition of safety, the system is safe if all the reachable states of the $\mathcal{A}$ are contained within the safe region **S**. The proof of safety is based on induction over the length of the executions of $\mathcal{A}$. We begin this section by stating the assumptions on the different parameters in the model, then we state the key invariants of $\mathcal{A}$, and finally we review the proof of safety in the different modes.

The correctness and safety of the supervisory controller is based on the following assumptions:

1. $\theta_{min} < 0 < |\theta_{min}| < \theta_{max}$
2. $U_{max} > \Omega^2$
3. For any trajectory τ of $\mathcal{A}$, if $\tau.fstate\lceil\theta_s^1 > I^+(\tau.fstate\lceil\theta_s^0)$ then, $\tau.lstate\lceil\theta_s^1 > I^-(\tau.lstate\lceil\theta_s^0)$. And if $\tau.fstate\lceil\theta_s^1 < I^-(\tau.fstate\lceil\theta_s^0)$ then, $\tau.lstate\lceil\theta_s^1 < I^+(\tau.lstate\lceil\theta_s^0)$.

The first two are facts derived from the dimensions of the actual system. The third constraint is required to prevent the supervisor from holding control forever by jumping between the region above I^+ and the region below I^- over a single Δ interval of time. This condition translates to bounds on the values of τ_{act}, Δ, and U_{mag}.

5.1 Modes of Execution

From restriction **R2** we know that the discrete variables of $\mathcal{A}$ remain constant over all trajectories, therefore *mode* also remains constant over trajectories. A trajectory $\tau \in \mathcal{T}_{\mathcal{A}}$ is said to be in the supervisor mode if $\tau.fstate\lceil mode = \texttt{sup}$, otherwise τ is in the user mode.

Property 2 *For any discrete step π of automaton $\mathcal{A}$, if $s \xrightarrow{\pi} s'$ and s is in the region $\mathbf{Z}$ then s' is also in $\mathbf{Z}$, where $\mathbf{Z} \in \{\mathbf{C}, \mathbf{S}, \mathbf{R}\}$.*

Proof: The regions **R, C** and **S** are defined in terms of continuous variables θ_p^0 and θ_p^1 and from the specification of the `Plant` automaton we observe that for any discrete step $s \xrightarrow{\pi} s'$, $s.\theta_p^0 = s'.\theta_p^0$ and $s.\theta_p^1 = s'.\theta_p^1$.

Invariant 1 *In every reachable state s of $\mathcal{A}$, $0 \leq s.next\_time - s.now \leq \Delta$.*

Proof: This is proved by induction on the length of a closed execution of automaton $\mathcal{A}$. The base case holds trivially due to initialization. For the discrete part of the induction, we consider the *sample* action only, none of the other actions alter any of the variables in the invariant. Let $s \xrightarrow{\pi} s'$ with $\pi = sample$. Then $s.now = s.next\_time$, $s'.next\_time = s.next\_time + \Delta$, and $s'.now = s.now$. Therefore $s'.next\_time - s'.now = \Delta$.

For the continuous part of the induction, we consider a closed trajectory τ_i with limit time $k \geq 0$, let $\tau_i fstate = s$ and $\tau_i.lstate = s'$ and $s.next\_time - s.now = t \in [0, \Delta]$. Let us assume for the sake of contradiction that $k > t$. Then $\tau_i \downarrow now(t) = \tau_i \downarrow next\_time(t)$, and this satisfies the post-condition of activity *read*, therefore $t = \tau_i.ltime$. This contradicts our assumption, and therefore $k \leq t$. From activity *read* the $s'.next\_time - s'.now = t - k$. By induction hypothesis, $0 \leq t \leq \Delta$, therefore $0 \leq s'.next\_time - s'.now \leq \Delta$.

Corrollary 1 *The limit time of every trajectory of $\mathcal{A}$ is upper bounded by Δ.*

Proof: Immediate from Invariant 1

In the next lemma we show that any closed execution of $\mathcal{A}$ is a repetitive sequence of *sample*, *control*, and *command* actions followed by a trajectory over which Δ time passes.

Lemma 2 *Let ζ be a closed execution of $\mathcal{A}$, such that $\zeta = \tau_0 \ldots \alpha_i \tau_i \alpha_{i+1} \tau_{i+1} \alpha_{i+2} \tau_{i+2} \alpha_{i+3} \ldots$ and $\alpha_i = sample$ then the following hold:*

1. *τ_i is a point trajectory and α_{i+1} is a* control *action,*
2. *τ_{i+1} is a point trajectory and α_{i+2} is a* command *action,*
3. *$\tau_{i+2}.lstate\lceil now = \tau_{i+1}.lstate\lceil now + \Delta$ and α_{i+3} is a* sample *action.*

Proof: For part (1) note that occurrence of *sample* sets *ready* flag to true and this enables the postcondition of activity *timer* of UsrCtrl and the *control* action. Therefore α_{i+1} is a *control* action and τ_i is a point trajectory.

Part (2) is similar, *control* enables action *command* and the postcondition of activity *timer* by setting the $ready_c$ flag. Therefore α_{i+2} is a *command* action and τ_{i+1} is a point trajectory.

Let s be the $fstate$ of τ_{i+2}, k be the *limit time* of τ_{i+2}. From parts (1) and (2), $s.next\_time - s.now = \Delta$, and from Corollary 1, $0 \leq k \leq \Delta$. If $k < \Delta$ then $\tau_{i+2}.lstate$ does not satisfy the precondition of any action. Therefore k must be equal to Δ, and α_{i+3} is a *sample* action.

5.2 User Mode

We present the proof of safety of the system in the user mode. We define a set of regions $\mathbf{A}_t$ for $0 \leq t \leq \Delta$,

$$\mathbf{A}_t \triangleq \{s \mid \theta_{min} + \leq s.\theta_p^0 \leq \theta_{max} \wedge \Gamma^-(s.\theta_p^0, \tau_{eff} + t) \leq s.\theta_p^1 \leq \Gamma^+(s.\theta_p^0, \tau_{eff} + t)\}, \tag{14}$$

and we prove that the following properties hold.

Property 3 *The region* $\mathbf{A}_t$ *satisfies the following:*

1. $\mathbf{A}_0 = \mathbf{R}$,
2. $\mathbf{U} \subseteq \mathbf{A}_\Delta$,
3. *If* $0 \leq t \leq t' \leq \Delta$ *then* $A_{t'} \subseteq A_t$

Proof: Part 1 is proved by replacing $t = 0$ in equation (14).

We know, $\theta_s^0 - \epsilon_0 \leq \theta_p^0 \leq \theta_s^0 + \epsilon_0$ and $\theta_s^1 - \epsilon_1 \leq \theta_p^1 \leq \theta_s^1 + \epsilon_1$. Therefore we have:

$$\mathbf{A}_\Delta \quad = \quad \{s \mid \theta_{min} - \epsilon_0 \leq s.\theta_s^0 \leq \theta_{max} + \epsilon_0 \wedge \Gamma^-(s.\theta_p^0, \tau_{eff} + \Delta) - \epsilon_1 \leq s.\theta_s^1 \leq \Gamma^+(s.\theta_p^0, \tau_{eff} + \Delta) + \epsilon_1\}$$

From Property 1, $\theta_p^0 \geq \theta_s^0 - \epsilon_0 \Rightarrow \Gamma^-(\theta_p^0, y) \leq \Gamma^-(\theta_s^0 - \epsilon_0, y)$ and $\theta_p^0 \leq \theta_s^0 + \epsilon_0 \Rightarrow \Gamma^+(\theta_p^0, y) \geq \Gamma^+(\theta_s^0 + \epsilon_0, y)$. Therefore,

$$\mathbf{A}_\Delta \supseteq \quad \{s \mid \theta_{min} \leq s.\theta_p^0 \leq \theta_{max} \wedge \Gamma^-(s.\theta_s^0 - \epsilon_0, \tau_{eff} + \Delta) + \epsilon_1 \leq s.\theta_p^1 \leq \Gamma^+(s.\theta_s^0 + \epsilon_0, \tau_{eff} + \Delta) - \epsilon_1\}$$

Recognizing the right hand side to be $\mathbf{U}$ from equation (5) completes the proof of part 2.

For part 3, we observe that equation in (14) Γ^+ and Γ^- are monotonically decreasing and increasing respectively with respect to t. Therefore if $0 \leq t \leq t' \leq \Delta$ then $\mathbf{A}_{t'} \subseteq \mathbf{A}_t$.

The next lemma gives us a bound on the reachable states over a closed trajectory in the user mode.

Lemma 3 *For any closed trajectory* τ *of* $\mathcal{A}$, *if* $\tau.fstate \in \mathbf{A}_t$ *then* $\tau.lstate \in \mathbf{A}_{t - ltime(\tau)}$.

Proof: Let s be the $fstate$, s' be the $lstate$, and t' be the *limit lime* of τ. As s is in $\mathbf{A}_t$, $\theta_{min} \leq s.\theta_p^0 \leq \theta_{max}$

and $\Gamma^-(s.\theta_p^0,\ \tau_{eff}+t) \ \le\ s.\theta_p^1 \ \le\ \Gamma^+(s.\theta_p^0,\ \tau_{eff}+t)$. We conservatively approximate s' considering the maximum and the minimum bounds on the input U to the `Plant`. First considering the maximum positive output, $U = U_{max}$, we get the upper bound on the acceleration at any state s'' in τ. $d(s''.\theta_p^1) \le -\Omega^2 \cos\theta_{max} + U_{max}$. Integrating from t to t',

$$s'.\theta_p^1 \quad \le \quad (U_{max} - \Omega^2 \cos\theta_{max})t' + s.\theta_p^1 \tag{15}$$

$$s'.\theta_p^0 \quad \le \quad \frac{1}{2}(U_{max} - \Omega^2 \cos\theta_{max}){t'}^2 + \ s.\theta_p^1 t' + s.\theta_p^0 \tag{16}$$

Simplifying and using the definition of Γ^+ we get the bounds on $s'.\theta_p^0$ and $s'.\theta_p^1$.

$$s'.\theta_p^0 \quad \le \quad \theta_{max} \ and \tag{17}$$

$$s'.\theta_p^1 \quad \le \quad \Gamma^+(s'.\theta_p^0,\ \tau_{eff}+t-t') \tag{18}$$

Similarly considering maximum negative output, $U = U_{min}$, we get the lower bounds on $s'.\theta_s^0$ and $s'.\theta_s^1$.

$$s'.\theta_p^0 \quad \ge \quad \theta_{min} \quad and \tag{19}$$

$$s'.\theta_p^1 \quad \ge \quad \Gamma^-(s'.\theta_p^0,\ \tau_{eff}+t-t') \tag{20}$$

Combining equations (17) (18) (19) and (20) we have $s' \in \mathbf{A}_{t-t'}$.

Invariant 2 *For any reachable state s in the user mode, if $s.next\_time - s.now \neq 0$ then $s \in \mathbf{A}_{s.next\_time - s.now}$.*

Proof: The base case is true because $\tau_0.fstate.r = \Delta$ and $\tau_0.fstate \ \in \ \mathbf{U} \ \subseteq \ \mathbf{A}_\Delta$. For the discrete part of the induction we note that for a *sample* action $s \xrightarrow{\pi} s'$, to occur, $s.next\_time = s.now$ and the invariant holds vacuously. The *control* and *command* actions do not alter any of the variables involved in the invariant. For the continuous part, we consider a closed trajectory τ with $\tau.fstate = s$. Suppose $\tau.ltime = t'$ and $\tau.lstate = s'$, then $s'.now = s.now + t'$. From Lemma 3 we have $s \in \mathbf{A}_{s.next\_time - s.now} \Rightarrow s' \in \mathbf{A}_{s'.next\_time - s'.now}$.

Invariant 3 *All reachable states in the user mode are within* $\mathbf{R}$.

Proof: The base case is satisfied by the initialization of the variables. For a discrete transition $s \xrightarrow{\pi} s'$, according to Property 2, $s \in \mathbf{R}$ implies that s' is also in $\mathbf{R}$. Considering a closed trajectory τ in the user mode, with $\tau.fstate = s$ and $\tau.lstate = s'$. Let $s.r = s.next\_time - s.now$, there are two possible cases:

Case 1: If $s.r = 0$ then s satisfies the postcondition for activity *read* (Figure 4) and therefore τ is a point trajectory.

Case 2: If $0 < s.r \le \Delta$ then using Invariant 2, $s' \in \mathbf{A}_{s'.r}$. From Lemma 1 we know that, $0 \le s'.r \le \Delta$, therefore using Property 3, we have $s' \in \mathbf{R}$.

5.3 Supervisor Mode

In this section we prove the safety of the system in the supervisor mode. The *control* action decides the *mode* for the next Δ interval of time based on the current mode and the observed state. The switch from user to supervisor mode occurs when the observed state of the plant (θ_s^0, θ_s^1) is outside $\mathbf{U}$ and the reverse switch happens when (θ_s^0, θ_s^1) is in $\mathbf{I}$. From the code of `Supervisor` automaton it is clear that in any reachable state s, if $s.mode = \mathtt{sup}$ then $s.U_s = s.U_{sup}$, otherwise $s.U_s = s.U_{usr} = s.U_u$. The occurrence of a *command* action queues $(s.U_s, s.now)$ in *buffer*, removes the first element (u, t) from *buffer* and copies u into u_o. Since Δ time passes between the occurrence of consecutive *command* actions, a total of $\tau_{\mathtt{eff}}$ time elapses between the occurrence of a *command(u)* action and the corresponding u being copied to the output variable u_o. And once $u_o = u$, it remains unaltered for another Δ time. Formally we state this as the following claim:

Claim 1 *In any execution ζ of automaton $\mathcal{A}$, let s be a reachable state such that $s.U_s = q$. If ζ is longer than $s.now + \tau_{\mathtt{eff}}$, then for any state s' with $s.now + \tau_{\mathtt{eff}} \leq s'.now \leq s.now + \tau_{\mathtt{eff}} + \Delta$, $s'.U = q$.*

Proof: Since s is a reachable state with $s.U_s = q$, there exists a post-state s'' of a *command(q)* action in ζ such that, $s''.now = s.now$ and $s.buffer.\mathtt{last[1]} = q$. Let s''' be the post-state of the $(\frac{\tau_{\mathtt{eff}}}{\Delta} - 1)^{th}$ *command* action after *command(q)*. So $s'''.now = s''.now + \tau_{\mathtt{eff}} - \Delta$ and $s'''.buffer.\mathtt{head[1]} = q$. Let s' be any state after yet another *command* action after s''' then $s'.u_o = q$ and it remains unchanged until another *command* action occurs, i.e., for another Δ interval of time.

The above claim implies that the input U to the plant changes $\tau_{\mathtt{eff}}$ time after the corresponding command has been issued by the supervisor. In particular, when *mode* changes from `usr` to `sup`, $U_s = U_{sup}$, but for another $\tau_{\mathtt{eff}}$ time U continues to be equal to some U_{usr} issued prior to the mode switch. We use the stopwatch variable rt to split the supervisor mode into two phases (a) a *settling period* of constant length $\tau_{\mathtt{eff}}$ in which $rt \leq \tau_{\mathtt{eff}}$ and $U = U_{usr}$ and (b) a variable length *recovery period* during which $rt > \tau_{\mathtt{eff}}$ and $U = U_{sup}$. In the next two subsections we prove the safety in the settling and recovery periods.

5.4 Settling Period

We define a set of regions parameterized on t as follows:

$$\mathbf{B}_t \triangleq \{s \mid \theta_{min} \leq s.\theta_p^0 \leq \theta_{max} \ \wedge \ \Gamma^-(s.\theta_p^0,\ \tau_{\mathtt{eff}} - t) \leq s.\theta_p^1 \leq \Gamma^+(s.\theta_p^0,\ \tau_{\mathtt{eff}} - t)\} \tag{21}$$

where $0 \leq t \leq \tau_{\mathtt{eff}}$.

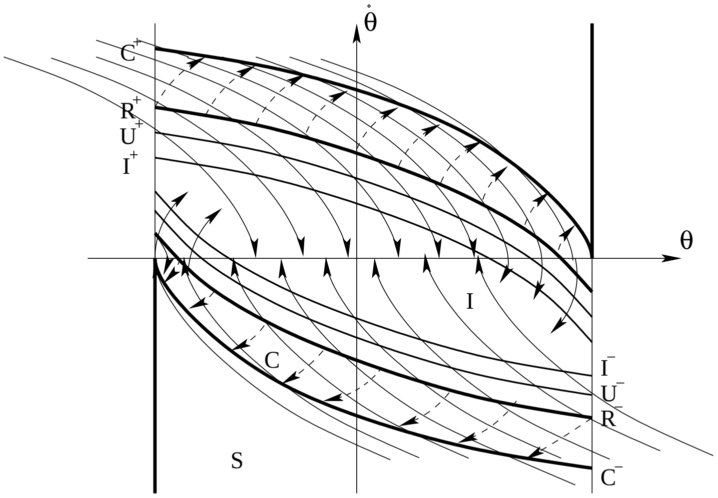

Figure 9: Trajectories in the settling (dashed lines) and recovery(solid lines) periods.

Property 4 *The region* $\mathbf{B}_t$ *satisfies the following:*

1. $\mathbf{B}_0 = \mathbf{R}$
2. $\mathbf{B}_{\tau_{\mathtt{eff}}} = \mathbf{C}$
3. *If* $0 \leq t \leq t' \leq \tau_{\mathtt{eff}}$ *then* $\mathbf{B}_t \subseteq \mathbf{B}_{t'}$

Proof: Parts 1 and 2 are proved by replacing $t = 0$ and $t = \tau_{\mathtt{eff}}$ in equation (21) and comparing with equations (4) and (3) respectively. Part 3 can be shown using Lemma 1, $t \leq t' \Rightarrow \mathbf{B}_t \subseteq \mathbf{B}_{t'}$.

Invariant 4 *Any reachable state* s *in the settling period is contained in the region* $\mathbf{B}_{s.rt}$.

Proof: It is sufficient to show that for any trajectory τ, if $s = \tau.fstate$, $s' = \tau.lstate$, and s and s' are in the settling period with s in $\mathbf{B}_{s.rt}$ then s' is in $\mathbf{B}_{s'.rt}$. Since $s \in \mathbf{B}_{s.rt}$, $\theta_{min} \leq s.\theta_p^0 \leq \theta_{max}$ and $\Gamma^-(s.\theta_p^0,\ \tau_{\mathtt{eff}} - s.rt) \leq s.\theta_p^1 \leq \Gamma^+(s.\theta_p^0,\ \tau_{\mathtt{eff}} - s.rt)$. Since both s and s' are in the settling period, for all states s'' with $s.rt \leq s''.rt \leq s.rt$, $s''.U = s''.U_{usr}$. Using the maximum positive value of U_{usr} and integrating from $\tau.ftime$ to $\tau.ltime$ we get the same bounds on $s'.\theta_p^1$ and $s'.\theta_p^0$ as in equations (15) and (16). Simplifying:

$$s'.\theta_p^0 \quad \leq \quad \theta_{max} \ and \tag{22}$$

$$s'.\theta_p^1 \quad \leq \quad \Gamma^+(s'.\theta_p^0,\ \tau_{\mathtt{eff}} - s.rt - t') \tag{23}$$

Where $t' = \tau.ltime$. Similarly considering maximum negative output, $U = U_{min}$, we get the lower bounds.

$$s'.\theta_p^0 \quad \geq \quad \theta_{min} \quad and \tag{24}$$

$$s'.\theta_p^1 \quad \geq \quad \Gamma^-(s'.\theta_p^0,\ \tau_{\texttt{eff}} - s.rt - t') \tag{25}$$

As $s'.rt = s'.rt + t'$ and $s'.rt \leq \tau_{\texttt{eff}}$, combining equations (22) (23) (24) and (25) we have $s' \in \mathbf{B}_{s'.rt}$.

Corrollary 2 *Any reachable state s in the settling period is in* $\mathbf{C}$.

Proof: From Invariant 4 using Property 4.

5.5 Recovery Period

In this section we prove safety in the recovery period and we give an upperbound on the recovery time. From the code of `Supervisor` we see that in any reachable state in the recovery period $U = U_{sup}$ and U_{sup} is either U_{max} or U_{min}. So the evolution of the continuous variables is completely deterministic. For a given state s in $\mathbf{C}$ with $s.rc \geq \tau_{\texttt{eff}}$, we show that all future states of s are contained in the set $\mathbf{C}$.

Lemma 4 *All reachable states over the recovery period are contained within* $\mathbf{C}$.

Proof: It is sufficient to show that for any closed trajectory τ if $\tau.fstate$ is reachable and is in $\mathbf{C}$ then $\tau.lstate$ is also in $\mathbf{C}$. Let $s = \tau.fstate$ and $s' = \tau.lstate$. First, if s is in $\mathbf{U}$, then we can make use of Lemma 3 to show that s' is in $\mathbf{R}$. Lemma 3 tells us that over any trajectory starting from $\mathbf{U}$, even the worst possible user input to the plant cannot take s' beyond $\mathbf{R}$. Therefore $s' \in \mathbf{C}$. Next, if s is not in $\mathbf{U}$ then, we show by Lyapunov theory that for all trajectories in the recovery period s' will be inside $\mathbf{C}$. We consider the four possible boundaries of $\mathbf{C}$ - $\mathbf{U}$.

Case 1: If $\theta_{min} \leq \theta \leq \theta_{max}$ and $\Gamma^+(\theta, \tau) \leq \dot{\theta} \leq \Gamma^+(\theta, 0)$, then $U = U_{sup} = U_{min}$, then we have

$$d(\theta_p^1) = -\Omega^2 \cos\theta_p^0 + U_{min}.$$

Suppose $\theta_p^1 \geq 0$, then we define

$$V_1(\theta_p^0, \theta_p^1) = \frac{1}{2}{\theta_p^1}^2 + \left(-U_{min} + \Omega^2 \cos\theta_{max}\right)\theta_p^0$$

$V_1 = K$ defines a family of curves for different values of K. When the $K = K_0 = \left(-U_{min} + \Omega^2 \cos\theta_{max}\right)\theta_{max}$ then the curve is the same as C^+. For $K < K_0$ then $V_1 = K$ represents curves under C^+.

$$\begin{aligned}
\frac{dV_1}{dt} &= \theta_p^1 d(\theta_p^1) + \left(-U_{min} + \Omega^2 \cos\theta_{max}\right)\theta_p^1 \\
&= \Omega^2\left(\cos\theta_{max} - \cos\theta_p^0\right)\theta_p^1.
\end{aligned}$$

Since by assumption $\theta_{min} \le \theta_p^0 \le \theta_{max}$ and $\theta_{max} \ge |\theta_{min}|$, we have $\frac{dV_1}{dt} \le 0$ for any $\theta_p^0 \in [\theta_{min}, \theta_{max}]$, where the equality holds if and only if $\theta_p^0 = \theta_{max}$ or $\theta_p^1 = 0$ which is not a trajectory of the system (in supervisor mode).

Now if $\theta_p^1 \le 0$, we define

$$V_2(\theta_p^0, \theta_p^1) = \frac{1}{2}{\theta_p^1}^2 + \left(-U_{min} + \Omega^2\right)\theta_p^0,$$

and we have

$$\frac{dV_2}{dt} = \Omega^2(1 - \cos\theta_p^0)\dot{\theta}_p^0 \le 0,$$

where the equality holds if and only if $\theta_p^0 = 0$ or $\theta_p^1 = 0$, which cannot be a state in a trajectory in the supervisor mode.

Case 2: $\theta_{min} \le \theta_p^0 \le \theta_{max}$ and $\Gamma^-(\theta_p^0, 0) \le \theta_p^1 \le \Gamma^-(\theta_p^0, \tau)$, and the actuator is executing the supervisor U_{max}, then we have

$$d(\theta_p^1) = -\Omega^2 - U_{max}.$$

Suppose $\theta_p^1 \le 0$, we define

$$V_3(\theta_p^0, \theta_p^1) = \frac{1}{2}{\theta_p^1}^2 - \left(U_{max} - \Omega^2\right)\theta_p^0,$$

then we have

$$\frac{dV_3}{dt} = \Omega^2\left(1 - \cos\theta_p^0\right)\theta_p^1.$$

So we have $\frac{dV_3}{dt} \le 0$ for any $\theta_p^0 \in [\theta_{min}, \theta_{max}]$, where the equality holds if and only if $\theta_p^0 = 0$ or $\theta_p^1 = 0$, which cannot be a state of any trajectory in the supervisor mode.

Now if $\theta_p^1 \ge 0$, we define

$$V_4(\theta_p^0, \theta_p^1) = \frac{1}{2}{\theta_p^1}^2 - \left(U_{max} + \Omega^2\cos\theta_{max}\right)\theta_p^0,$$

and we have

$$\frac{dV_4}{dt} = \Omega^2(\cos\theta_{max} - \cos\theta_p^0)\theta_p^1 \le 0,$$

where the equality holds if and only if $\theta_p^0 = \theta_{max}$ or $\theta_p^1 = 0$, and once again this is not possible in a trajectory in the supervisor mode.

Hence, we have shown that the trajectories flow from the boundary of $\mathbf{C}$ to its inside, and $\mathbf{C}$ is invariant with the supervisor controllers.

5.6 Upper Bound on Recovery Time

The recovery period from any state in $\mathbf{C}$ is finite, if the parameters τ_{eff}, U_{mag} and Δ satisfy the conditions stated at the beginning of this Section. Therefore the total duration for which the user loses control is also finite (= recovery period + $\tau_{\mathtt{eff}}$). We claim that the recovery time for any state in $\mathbf{C}$ is upper-bounded by the expression given below.

Claim 2 *For every $s \in \mathbf{C}$ the recovery period is upper-bounded by $\lceil \frac{T(s)}{\Delta} \rceil \Delta$. where*

$$
\begin{aligned}
T(s) \quad = \quad & \frac{1}{-U_{min} + \Omega^2 \cos\theta_{max}} \left(\epsilon_1 + \frac{1}{2} U_{mag}(\tau_{\mathtt{eff}} + \Delta) + s.\theta_p^1 + \frac{(s.\theta_p^1)^2}{2(2\epsilon_1 + U_{mag}(\tau_{\mathtt{eff}} + \Delta))} \right) \\
& - \frac{1}{2\epsilon_1 + U_{mag}(\tau_{\mathtt{eff}} + \Delta)} \left(\theta_{max} + \frac{1}{2} U_{mag}(\tau_{\mathtt{eff}} + \Delta)^2 - s.\theta_p^0 - 2\epsilon_0 \right).
\end{aligned}
$$

This upper bound can be proved by first finding the future state s', corresponding to a given state s, where the approximated parabolic trajectory intersects I^+ or I^-. Then the time elapsed in this execution fragment between s and s' gives the upper bound on the recovery time.

5.7 Safety

Theorem 1 *All reachable states of $\mathcal{A}$ are contained within $\mathbf{C}$.*

Proof: This is proved by induction over the length of an arbitrary execution ζ of $\mathcal{A}$. The base case is trivially satisfied and the discrete part of the induction is immediate from Property 2. For the continuous part, let $s = \tau.fstate$ and $s' = \tau.lstate$ for some closed trajectory τ. It is required to show that, $s \in \mathbf{C} \Rightarrow s' \in \mathbf{C}$. If $s.mode = \mathtt{usr}$ then $s'.mode = \mathtt{usr}$ and according to Invariant 3 $s' \in \mathbf{R} \subseteq \mathbf{C}$. Otherwise if $s.mode = \mathtt{sup}$ and $s.rc < \tau_{\mathtt{eff}}$ then using Corollary 2, $s' \in \mathbf{C}$, and if $s.rc \geq \tau_{\mathtt{eff}}$, then by Lemma 4 $s' \in \mathbf{C}$.

6 Conclusions

In this paper we have designed a supervisory controller for a model helicopter system and have verified its safety. The supervisor periodically observes the state of the plant and allows the user to be in control as long as the plant state is in a safe operating region. If the plant goes outside this region then the supervisor overrides the user's controller and performs the appropriate recovery action. The geometry of the safe operating region is determined by the dynamics of the plant, sensor inaccuracies, actuator bandwidth, and saturation. We observe that the limitations in the sensor and the actuator considerably shrinks the safe operating region for an arbitrary user controller.

In specifying the components of the system we have used the hybrid I/O automaton model of [7] with certain restrictions which allow us to specify the trajectories of the system using first order differential equations. For verification we have used the standard deductive techniques involving induction over the length of the executions of the automata, along with methods from continuous maths.

As an extension of this work we want to design and verify a class supervisory controllers which would utilize additional information about the user controllers and reduce unnecessary interference. We would also like to examine the scalability of the methods used here by studying models with three degrees of freedom.

References

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

[2] Thomas A. Henzinger, Pei-Hsin Ho, and Howard Wong-Toi. Hytech: A model checker for hybrid systems. In *Coputer Aided Verification (CAV '97)*, volume 1254 of *Lecture Notes in Computer Science*, pages 460–483, 1997.

[3] Thomas A. Henzinger, Peter W. Kopke, Anuj Puri, and Pravin Varaiya. What's decidable about hybrid automata? In *ACM Symposium on Theory of Computing*, pages 373–382, 1995.

[4] http://www.quanser.com/english/htm/about/fs_about_splash.htm.

[5] Carolos Livadas, John Lygeros, and Nancy A. Lynch. High-level modeling and analysis of TCAS. In *Proceedings of the 20th IEEE Real-Time Systems Symposium (RTSS'99),Phoenix, Arizona*, pages 115–125, December 1999.

[6] Nancy Lynch. A three-level analysis of a simple acceleration maneuver, with uncertainties. In *Proceedings of the Third AMAST Workshop on Real-Time Systems*, pages 1–22, Salt Lake City, Utah, March 1996. World Scientific Publishing Company.

[7] Nancy Lynch, Roberto Segala, and Frits Vaandraager. Hybrid I/O automata. Technical Report MIT-LCS-TR-827b, MIT Laboratory for Computer Science, Technical Report, Cambridge, MA 02139, February 2002. submitted for journal publication, theory.lcs.mit.edu/tds/papers/Lynch/HIOA-final.ps.

[8] Nancy Lynch, Roberto Segala, Frits Vaandrager, and H. B. Weinberg. Hybrid I/O automata. In T. Henzinger R. Alur and E. Sontag, editors, *Hybrid Systems III*, volume 1066 of *Lecture Notes in Computer Science*, New Brunswick, New Jersey, October 1995. Springer-Verlag.

[9] Nancy A. Lynch, Roberto Segala, and Frits W. Vaandrager. Hybrid I/O automata revisited. In M.D. Di Benedetto and A.L. Sangiovanni-Vincentelli, editors, *Proceedings Fourth International Workshop on Hybrid Systems: Computation and Control (HSCC'01),* Rome, Italy, volume 2034 of *lncs*. springer, March 2001.

[10] Sayan Mitra. HIOA - a specification language for hybrid Input/Output automata. Master's thesis, Department of Computer Science and Automation, IISc, Indian Institute of Science, Bangalore, 2001. http://theory.lcs.mit.edu/ mitras/research/mastersthesis.ps.gz.

[11] Yong Wang, S. Mitra, C. Livadas, N. Lynch, and E. Feron. Design of Supervisory Safety Control for 3DOF Helicopter using Hybrid I/O Automata, 2002. pre-print http://gewurtz.mit.edu/ishut/darpa_sec_mit/papers/quanser.ps.

[12] H. B. Weinberg, Nancy Lynch, and Norman Delisle. Verification of automated vehicle protection systems. In T. Henzinger R. Alur and E. Sontag, editors, *Hybrid Systems III: Verification and Control (DIMACS/SYCON Workshop on Verification and Control of Hybrid Systems)*, volume 1066 of *Lecture Notes in Computer Science*, pages 101–113, New Brunswick, New Jersey, October 1995. Springer-Verlag.