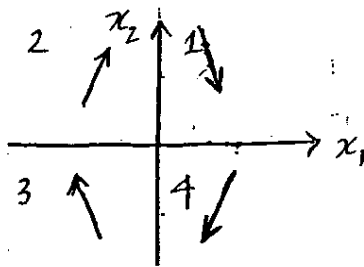


1 (a) An example is $H = (Q, X, \text{Init}, f, \text{Dom}, R)$.

with $Q = \{1, 2, 3, 4\}$, $X = \mathbb{R}^2$



$$f(1, \cdot) = \begin{bmatrix} 1 \\ -3 \end{bmatrix}$$

$$f(3, \cdot) = \begin{bmatrix} -1 \\ 3 \end{bmatrix}$$

$$f(2, \cdot) = \begin{bmatrix} 3 \\ 1 \end{bmatrix}$$

$$f(4, \cdot) = \begin{bmatrix} -3 \\ -1 \end{bmatrix}$$

Init (away from $x_1 = 0$ or $x_2 = 0$)

$$\text{Inv} = (1, \{x : x_1 > 0, x_2 > 0\}) \cup$$

$$(2, \{x : x_1 < 0, x_2 > 0\}) \cup$$

$$(3, \{x : x_1 < 0, x_2 < 0\}) \cup$$

$$(4, \{x : x_1 > 0, x_2 < 0\})$$

$$R(1, \{x : x_2 \leq 0\}) = (4, [x_1, x_2 := 0^-])$$

$$R(2, \{x : x_1 \geq 0\}) = (1, [x_1 := 0^+, x_2])$$

$$R(3, \{x : x_2 \geq 0\}) = (2, [x_1, x_2 := 0^+])$$

$$R(4, \{x : x_1 \leq 0\}) = (3, [x_1 := 0^-, x_2])$$

$$R = \emptyset \text{ otherwise}$$

Since $R(q, x) \neq \emptyset$ for all $(q, x) \in \text{Trans}$
 H is non-blocking. Since $|R(q, x)| \leq 1$
 for all (q, x) , H is deterministic.

- 1 (b) FIRST, NOTE that for every execution there exists i such that $x(\gamma_i) = (a, b)^T$ where either $a=0$ or $b=0$. Then $x(\gamma_i') = (c, d)^T$ where $(c, d) = (\frac{b}{3}, a)$ if $a=0$ and $(0, -\frac{a}{3})$ if $b=0$. It thus follows that H accepts a unique infinite execution for every initial state. The execution is zero because for all $t \in \tau$

$$\dot{W}(t) = \frac{d}{dt} (\|x_1(t)\| + \|x_2(t)\|) = -2$$

where $W(t) = \|x_1(t)\| + \|x_2(t)\|$ (the l_1 -norm)

so $W(t) = 0$ for some finite t and thus the origin is reached in finite time.

- 2 (a) An example is $H = (Q, X, \text{Init}, f, \text{Dom}, R)$

where $Q = \{q\}$, $X = \mathbb{R}^3$

$$f(q, \cdot) = [0, 0, 0]^T$$

$$\text{Init} = (q, (1, 0, 0)^T)$$

$$\text{Dom} = (q, (0, 0, 0)^T)$$

and

$$R(q, (x_1, x_2, x_3)) = \begin{cases} (q, ((x_1+x_2)/2, (x_1+x_2)/2, x_3)) & \text{if } x_1 > x_2 \\ (q, (x_1, (x_2+x_3)/2, (x_2+x_3)/2)) & \text{if } x_2 > x_3 \\ \emptyset & \text{otherwise} \end{cases}$$

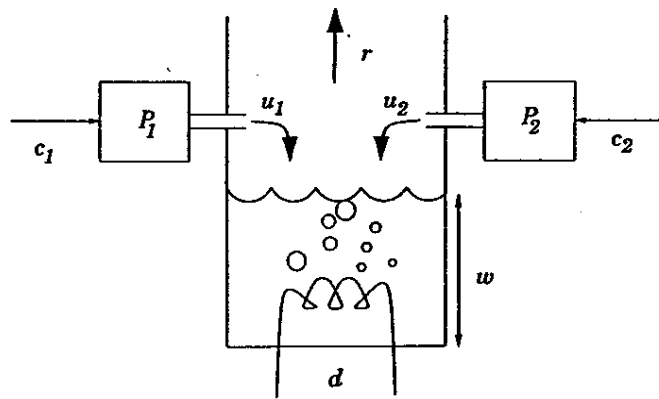


Figure 1: The Steam Boiler

Problem 3: Steam Boiler (Steam Boiler System, Figure 1)

The steam boiler consists of a tank containing water and a heating element that causes the water to boil and escape as steam. The water is replenished by two pumps which at time t pump water into the boiler at rates $u_1(t)$ and $u_2(t)$ respectively. At every time t , pump i can either be on ($u_i(t) = \bar{P}_i$) or off ($u_i(t) = 0$). There is a delay \bar{T}_i between the time pump i is ordered to switch on and the time q_i switches to P_i . There is no delay when the pumps are switched off. We will use three hybrid automata to describe this system, one for the boiler, $B = (Q_B, X_B, V_B, Y_B, Init_B, f_B, h_B, Inv_B, E_B, G_B, R_B)$, and one for each of the pumps, $P_i = (Q_i, X_i, V_i, Y_i, Init_i, f_i, h_i, Inv_i, E_i, G_i, R_i)$.

The boiler automaton is defined by:

- $Q_B = \{q_B\}, Q_B = \{BOILING\};$
- $X_B = \{w, r\}, X_B = \mathbb{R}^2;$
- $V_B = \{u_1, u_2, d\}, V_B = [0, \bar{P}_1] \times [0, \bar{P}_2] \times [-D_1, D_2],$ where $\bar{P}_1, \bar{P}_2, D_1, D_2 > 0;$
- $Y_B = \{y_1, y_2\}, Y_B = \mathbb{R}^2;$
- $Init = \{BOILING\} \times [0, W] \times [0, R],$ where $W, R > 0;$

you can ignore here for now

$$f(BOILING, w, r, u_1, u_2, d) = \begin{bmatrix} \dot{w} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} u_1 + u_2 - r \\ d \end{bmatrix}$$

$$h(BOILING, w, r) = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} w \\ r \end{bmatrix}$$

- $Inv(BOILING) = X_B \times V_B =: Dom$

edges • $E = \emptyset$

Notice that since $E = \emptyset$, G and R are trivial and need not be explicitly defined.

The automaton for pump i is defined by:

- $Q_i = \{q_i\}, Q_i = \{OFF, GOING_ON, ON\};$

- $X_i = \{T_i\}, X_i = R; \text{resets}$

"

- $V_i = \{c_i\}, V_i = \{0, 1\};$
- $Y_i = \{u_i\}, Y_i = [0, \bar{P}_i];$

- $Init = \{OFF\} \times \{0\},$

$$f(q_i, T_i, c_i) = \begin{cases} 1 & \text{if } q_i = GOING_ON \vee q_i = ON \\ 0 & \text{if } q_i = OFF \end{cases}$$

$$h(q_i, T_i) = \begin{cases} 0 & \text{if } q_i = OFF \vee q_i = GOING_ON \\ \bar{P}_i & \text{if } q_i = ON \end{cases}$$

$$Dom := Inv(q_i) = \begin{cases} X_i \times \{c_i = 0\} & \text{if } q_i = OFF \\ \{T_i \leq \bar{T}_i\} \times \{c_i = 1\} & \text{if } q_i = GOING_ON \\ X_i \times \{c_i = 1\} & \text{if } q_i = ON \end{cases}$$

edges

- $E = \{(OFF, GOING_ON), (GOING_ON, OFF), (GOING_ON, ON), (ON, OFF)\}$

guards

$$G(e) = \begin{cases} X_i \times \{c_i = 1\} & \text{if } e = (OFF, GOING_ON) \\ X_i \times \{c_i = 0\} & \text{if } e = (GOING_ON, OFF) \\ \{T_i \geq \bar{T}_i\} \times \{c_i = 1\} & \text{if } e = (GOING_ON, ON) \\ X_i \times \{c_i = 0\} & \text{if } e = (ON, OFF) \end{cases}$$

resets

$$R(e, T_i, c_i) = \begin{cases} \{0\} & \text{if } e = (OFF, GOING_ON) \\ \{0\} & \text{if } e = (GOING_ON, OFF) \\ \{T_i\} & \text{if } e = (GOING_ON, ON) \\ \{0\} & \text{if } e = (ON, OFF) \end{cases}$$

The controller synthesis for this model can be found in [4]. The continuous dynamics of the boiling process are summarized by the differential equations:

$$\begin{aligned}\dot{w} &= p_1 + p_2 - r \\ \dot{r} &= d\end{aligned}$$

The dynamics of pump i are summarized by the open hybrid automaton of Figure 2. Notice that p_i is both an output variable of the pump and an input variable of the boiling process. For a formal definition of the model (with slight differences in the notation) please refer to Lecture 8.

The composite automaton has 4 continuous, real valued state variables:

$$x = (w, r, T_1, T_2) \in \mathbb{R}^4$$

SOLUTION FROM
"Controllers for
Reachability
Specifications for
Hybrid Systems"
Lygeros, Tomlin, Sastry
Automatica 1999.

Figure 1: The Steam Boiler

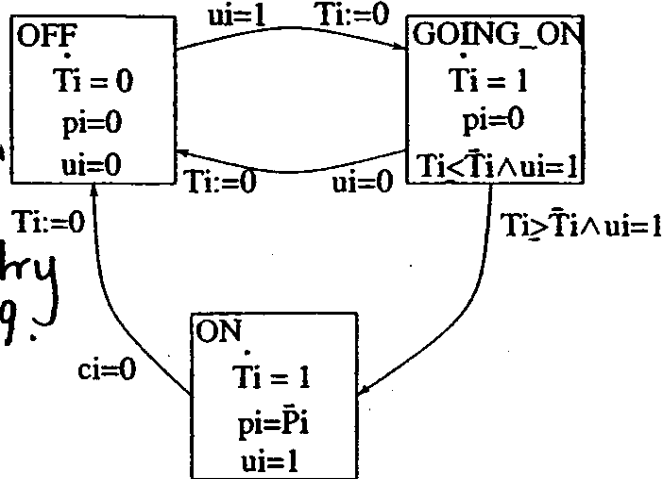


Figure 2: The pump hybrid automaton

9 discrete states:

$$q \in \{(OFF, OFF), (OFF, GOING_ON), \dots, (ON, ON)\}$$

2 discrete input variables:

$$(u_1, u_2) \in \{0, 1\} \times \{0, 1\} = U$$

and one continuous input variable:

$$d \in [-D_1, D_2] = D$$

As the notation suggests, u_1 and u_2 will play the role of controls and d will play the role of the disturbance. The additional requirement that $\tau \in [0, R]$ can be encoded by a state dependent input constraint:

$$\phi(q, x) = \begin{cases} U \times [0, D_2] & \text{if } \tau \leq 0 \\ U \times D & \text{if } \tau \in (0, R) \\ U \times [-D_1, 0] & \text{if } \tau \geq R \end{cases}$$

Proposition 3 If $\text{Init} \subseteq Q \times \mathbb{R} \times [0, R] \times \mathbb{R}^2$, then for all $\chi = (\tau, q, x, u_1, u_2, d)$ and for all $t \in \tau$, $\tau(t) \in [0, R]$.

Our goal is to design a controller that keeps the water level in a given range, $[M_1, M_2]$, with $0 \leq M_1 < M_2$. This requirement can easily be encoded by a safety property $(Q \cup X, \square F)$ with

$$F = Q \times [M_1, M_2] \times \mathbb{R}^3$$

We will try to achieve this goal by treating the situation as a game between (u_1, u_2) and d over the cost function J . Recall that this involves solving the equation:

$$J^*(q_0, x_0) = \max_g \min_d \left(\min_{\chi=(\tau, q, x, (u, d)) \in \mathcal{H}_g} J(\chi) \right)$$

Fortunately, for this example, the equation simplifies considerably.

First, notice that the steam boiler system is deterministic, in the sense that for each initial state and each input sequence consistent with ϕ the automaton accepts a unique execution. In this case, we can represent an execution more compactly by $((q_0, x_0), (u_1, u_2), d)$ with the interpretation that (u_1, u_2) and d represent the entire sequence for these variables. Moreover, if the memoryless controller we pick is single valued, this implies we need not worry about the innermost minimization.

Next notice that J can be encoded by means of two real valued cost functions:

$$J_1(x^0, u_1, u_2, d) = \inf_{t \geq 0} w(t) \quad \text{and} \quad J_2(x^0, u_1, u_2, d) = -\sup_{t \geq 0} w(t) \tag{3}$$

Clearly:

$$J = 1 \Leftrightarrow (J_1 \geq M_1) \wedge (J_2 \geq -M_2)$$

The problem of finding a solution to the game over the discrete cost function (known as *qualitative game* or *game of kind*) reduces to finding solutions to two real valued games (known as *quantitative games* or *games of degree*). Even though there is no obvious benefit to doing this, it allows us to use tools from continuous optimal control to address the problem.

Start with the game over J_1 . Guess a possible solution:

$$u_i^*(q, x) = 1 \text{ for all } (q, x), \quad \text{and} \quad d^*(q, x) = \begin{cases} D_2 & \text{if } \tau < R \\ 0 & \text{if } \tau = R \end{cases} \tag{4}$$

Notice that both players resort to a feedback strategy (a trivial one).

Lemma 1 (u_1^*, u_2^*, d^*) is globally a saddle solution for the game between (u_1, u_2) and d over J_1 .

Proof: See [4]. ■

A *saddle solution* is a solution to equation (1) for which the order in which the players make their decisions turns out to be unimportant. In other words, a solution for which for all (q, x) :

$$J_1^*(q, x) = \max_{(u_1, u_2)} \min_d J_1((q, x), (u_1, u_2), d) = \min_d \max_{(u_1, u_2)} J_1((q, x), (u_1, u_2), d)$$

Or, in other words, a solution for which for all $(q, x), u_1, u_2$ and d :

$$J_1((q, x), (u_1, u_2), d^*) \leq J_1((q, x), (u_1^*, u_2^*), d^*) \leq J_1((q, x), (u_1^*, u_2^*), d)$$

The last definition is usually somewhat easier to to work with. It is in fact used to prove the lemma.

The saddle cost:

$$J_1^*(q, x) = J_1((q, x), (u_1^*, u_2^*), d^*)$$

Can be computed in closed form. This allows us then to compute the set of states for which there exists a control that for all actions of the disturbance prevents draining. This set turns out to be of the form:

$$W_1^* = \{(q, x) : J_1^*(q, x) \geq M_1\} = \{(q, x) : w \geq \hat{w}(\tau, T_1, T_2)\}$$

Two level sets of this function are shown in Figure 3.

The expression for J^* also allows us to compute the least restrictive controller that renders the set W_1^* invariant. It turns out to be unique:

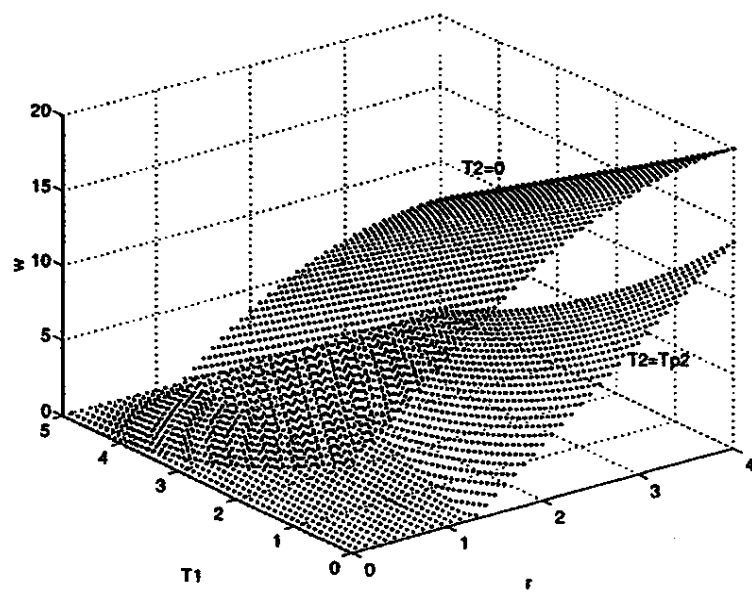


Figure 3: Lower limit on w to avoid draining

Lemma 2 The feedback controller g_1^1 given by:

$$\begin{aligned}
 &u_1 \in \{0, 1\} \text{ and } u_2 \in \{0, 1\} \text{ if } [w > \hat{w}(r, 0, 0)] \vee [w < \hat{w}(r, T_1, T_2)] \\
 &u_1 = 1 \text{ and } u_2 \in \{0, 1\} \text{ if } \hat{w}(r, 0, 0) \geq w > \hat{w}(r, T_1, 0) \\
 &u_1 \in \{0, 1\} \text{ and } u_2 = 1 \text{ if } \hat{w}(r, 0, 0) \geq w > \hat{w}(r, 0, T_2) \\
 &u_1 = 1 \text{ and } u_2 = 1 \text{ if } w = \hat{w}(r, T_1, T_2)
 \end{aligned}$$

is the unique, least restrictive, non-blocking, feedback controller that renders W_1^{1*} invariant.

Proof: See [4].

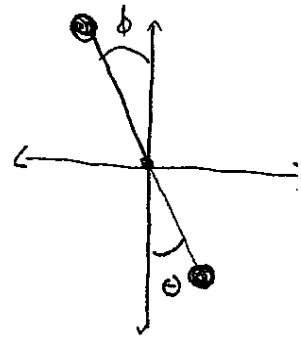
Note that the first term applies to states in the interior of the safe set ($w > \hat{w}(r, 0, 0)$) as well as all the states outside the safe set ($w < \hat{w}(r, T_1, T_2)$). The expression for \hat{w} (see [4]) suggests that \hat{w} is monotone in T_1 and T_2 . Therefore, the condition on the last case is enabled if and only if all other conditions fail. The two middle conditions may overlap, however. Therefore there is some nondeterminism in the choice of safe controls (some states may be safe with either one or the other pump on, but not neither).

References

- [1] J.-R. Abrial, E. Börger, and H. Langmaack, "The steam-boiler case study project, an introduction", in *Formal Methods for Industrial Applications: Specifying and Programming the Steam Boiler Control*, J.-R. Abrial, E. Börger, and H. Langmaack, Eds., number 1165 in LNCS. Springer Verlag, 1996.
- [2] J.-R. Abrial, "The steam-boiler control specification problem", in *Formal Methods for Industrial Applications: Specifying and Programming the Steam Boiler Control*, J.-R. Abrial, E. Börger, and H. Langmaack, Eds., number 1165 in LNCS. Springer Verlag, 1996.
- [3] Tomas A. Henzinger and Howard Wong-Toi, "Using HYTECH to synthesize control parameters for a steam boiler", in *Formal Methods for Industrial Applications: Specifying and Programming the Steam Boiler Control*, J.-R. Abrial, E. Börger, and H. Langmaack, Eds., number 1165 in LNCS, pp. 265–282. Springer Verlag, 1996.

3b

The pendulum dynamics are



$$\ddot{\phi} + F(\phi) - k \sin \phi + u(\phi, \dot{\phi}) = 0$$

$$\text{note: } \phi = \theta - \pi$$

The controller is designed with 3 modes

$$q_1 = \text{Balance: when } \frac{\phi^2}{\phi_{\max}^2} + \frac{\dot{\phi}^2}{\dot{\phi}_{\max}^2} \leq 1$$

$$u = (c_{11} + k) \phi + c_{12} \dot{\phi}$$

$$q_2 = \text{Pumping: when not in Balance and}$$

$$S = \frac{1}{2} \dot{\theta}^2 - k(1 - \cos(\theta)) < 0$$

$$u = -(c_2 + c_3) \dot{\theta}$$

$$q_3 = \text{Spinning: otherwise}$$

$$u = c_2 \dot{\theta}$$

The coefficients given are $c = 0.01$, $k = 10$, $u_{\max} = 4$

$$c_{11} = 0.4, c_{12} = 0.3, c_2 = 0.5, c_3 = 0.5$$

For systems such as the inverted pendulum, it is quite natural to design a hybrid controller to achieve the distinct goals of adding sufficient energy to the system and balancing the pendulum. The resulting control laws are simple and easy to understand intuitively. The issue of chatter between modes arises, however, and must be dealt with in any real implementation.

```
clear;

% Simulation Parameters
y0 = [0 -1];
t_max = 25;

% Call solver
[T,Y] = ode45('pendulum',[0 t_max], y0);

% Recalculate control inputs
for i=1:length(T)
    [u(i), q(i)] = controller(Y(i,:));
end

%Plot
figure(1);clf;
subplot(3,1,1);
plot(T,Y(:,1),T,Y(:,2),'r:')
title('Pendulum Trajectory')
legend('\theta', '\theta_{dot}')

subplot(3,1,2);
plot(T,u)
title('Control Inputs')

subplot(3,1,3);
plot(T,q)
title('Discrete Mode')
xlabel('Time')
```

```
function [dydt] = plant(T,Y)

c = 0.01;
k = 10;

% System Models, using both coordinate systems
[u, q] = controller(Y);
thetad(1) = Y(2);
thetad(2) = - c*Y(2) - k*sin(Y(1))- u;

% Derivative output for ode23 solver
dydt = thetad';
```

```
function [u_out,q_out] = controller(Y)

% This model has three modes: Pumping, Spinning and Balancing, with the only difference
% between the three being the control input.

%System Constants
c = 0.01;
k = 10;
umax = 4;
c11 = 0.4;
c12 = 0.3;
c2 = 0.5;
c3 = 0.5;

%Input states
theta = Y;
phi(1) = Y(1) - pi;
phi(2) = Y(2);

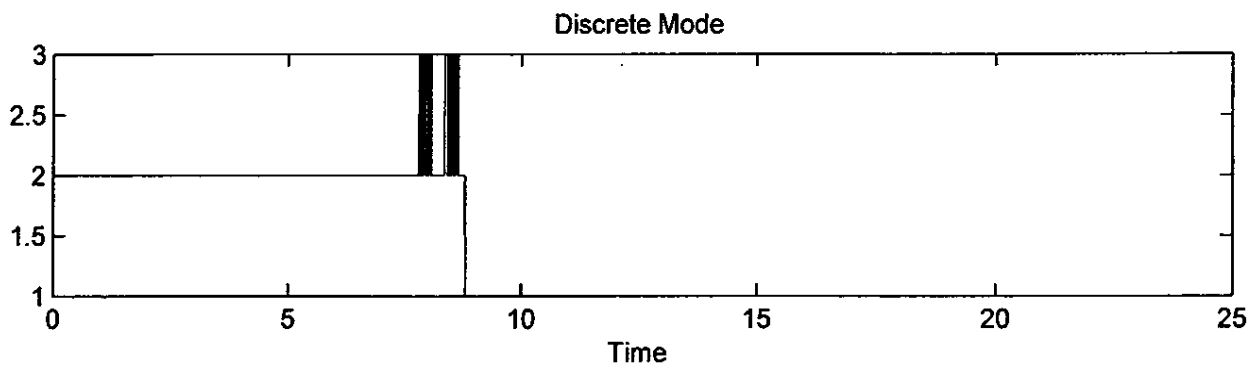
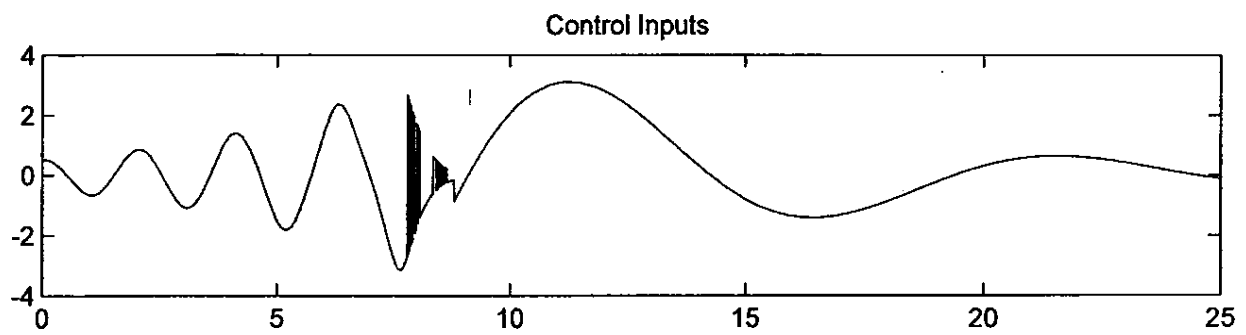
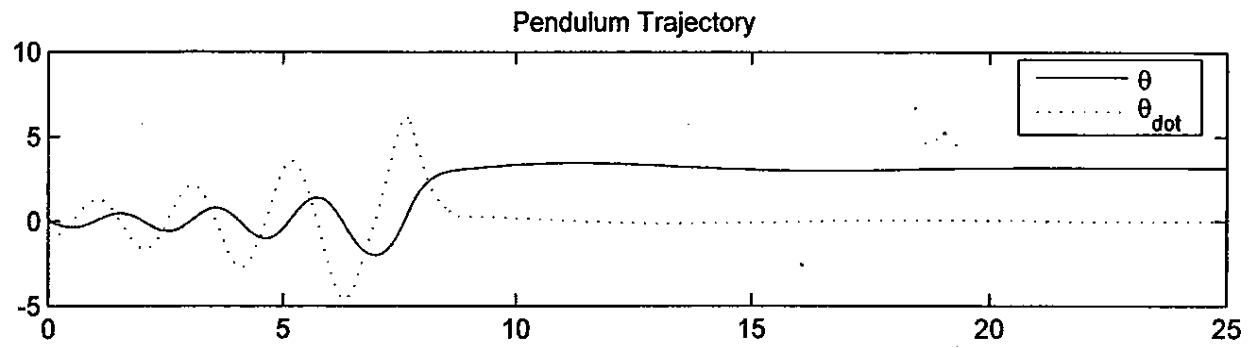
%Controller selection
phimax = 0.4;
phidmax = 0.3;

balance = phi(1)^2/phimax^2 + phi(2)^2/phidmax^2;
        = 1/2*theta(2)^2-k*(1+cos(theta(1)));

if balance <= 1
    % Balance Control
    u = (c11+k)*sin(phi(1)) + c12*phi(2);
    q = 1;
elseif s < 0
    %Pumping Control
    u = -(c+c3)*theta(2);
    q = 2;
else
    %Spinning Control
    u = c2*theta(2);
    q = 3;
end

if abs(u) > umax
    u = umax * sign(u);
end

u_out = u;
q_out = q;
```



this kind of Zeno behavior. The classical way of analyzing such systems is by introducing the notion of sliding modes [8, 18].

← Regularization → SOLUTION TO PROBLEM 4.
FROM "ON THE REGULARIZATION OF ZENO
HYBRID AUTOMATA"
JOHANSSON, EGERSTEDT, LYGEROS, SASTRY SCL 1999.

Regularization is a standard technique for dealing with differential equations whose solutions are not well defined. We propose a similar approach to extend Zeno executions beyond the Zeno time, primarily for the purpose of simulation. The formal treatment of how to regularize general Zeno hybrid automata is the topic of current research. Here we limit ourselves to specific regularizations of the water tank and bouncing ball automata introduced above. All regularizations are motivated by physical considerations of the underlying systems. For the water tank automaton, it is interesting to notice that different regularizations suggest different extensions of the executions. For the bouncing ball automaton, all extensions considered here are consistent with one another and physical intuition. The regularizations are only presented graphically in this section; see [10] for formal definitions.

Consider a non-blocking and deterministic hybrid automaton H and assume that for every $(q_0, x_0) \in \text{Init}$ the execution $\chi \in \mathcal{H}_{(q_0, x_0)}^\infty$ is Zeno. Regularization of H involves constructing a family of deterministic, non-blocking, and non-Zeno automata H_ϵ , parameterized by a real valued parameter, $\epsilon > 0$, and a continuous map, $\phi : Q_\epsilon \times X_\epsilon \rightarrow Q \times X$, relating the state of each H_ϵ to the state of H . Given an execution $\chi_\epsilon = (\tau_\epsilon, q_\epsilon, x_\epsilon)$, we use $\phi(\chi_\epsilon)$ as a shorthand notation for the collection (τ, q, x) with $\tau = \tau_\epsilon$, and $(q(t), x(t)) = \phi(q_\epsilon(t), x_\epsilon(t))$ for all $t \in \tau$. Note that in general $\phi(\chi_\epsilon)$ will not be an execution of H . However, the construction of the family H_ϵ should be such that H_ϵ tends to H as ϵ tends to 0, in the sense that if $(q_{\epsilon_0}, x_{\epsilon_0}) \in \text{Init}_{\epsilon_0}$, then $\phi(q_{\epsilon_0}, x_{\epsilon_0}) \in \text{Init}$, and if χ_{ϵ_0} is the execution of H_{ϵ_0} with initial condition $(q_{\epsilon_0}, x_{\epsilon_0})$, then $\phi(\chi_{\epsilon_0})$ converges to $\chi \in \mathcal{H}_{\phi(q_{\epsilon_0}, x_{\epsilon_0})}^\infty$ over all compact subintervals of $[\tau_0, \tau_\infty)$, where the convergence is taken in the Skorohod metric [4].³

³ Formally, we need to eliminate all "inert" transitions from τ , that is, replace all $[\tau_i, \tau'_i][\tau_{i+1}, \tau'_{i+1}]$ for which $\phi(q_\epsilon(\tau'_i), x_\epsilon(\tau'_i)) = \phi(q_\epsilon(\tau_{i+1}), x_\epsilon(\tau_{i+1}))$ by a single interval $[\tau_i, \tau'_{i+1}]$.

Water Tank Automaton

We first study temporal and spatial regularizations of the water tank automaton. Throughout, we assume that $\max\{v_1, v_2\} < w < v_1 + v_2$, so that WT is Zeno.

Physically, temporal regularization represents a situation where there is a delay, $\epsilon > 0$, between the time the inflow is commanded to switch from one tank to the other and the time the switch actually takes place. The temporal regularization of the water tank automaton, WT_ϵ^T , is shown in Figure 3. It is easy to show that WT_ϵ^T accepts a unique non-Zeno execution for each initial state. Overloading the notation somewhat, we can express the relation between the states of WT_ϵ^T and the states of WT through the map $\phi(q_i, (x_1, x_2, x_3)) = \phi(q'_i, (x_1, x_2, x_3)) = (q_i, (x_1, x_2))$, for $i = 1, 2$. If we set $r_1 = r_2 = 1$, $v_1 = 2$, $v_2 = 3$, and $w = 4$, and assume that initially $x_1(0) = x_2(0) = 2$ and $q(0) = q_1$, then $\tau_\infty = 2$. Figure 4 shows simulation results for WT_ϵ^T ; x_1 and x_2 are plotted as functions of time for two values of ϵ , 0.1 and 0.01. Note that as ϵ decreases, the execution of WT_ϵ^T converges over the interval $(\tau_0, \tau_\infty) = (0, 2)$ to the execution of WT , in the sense discussed above. For $t > \tau_\infty$, the continuous part of the execution of WT_ϵ^T tends to $(x_1(t), x_2(t)) = (1, 1 - (t - \tau_\infty))$.

The spatial regularization of the water tank automaton corresponds to a situation where the measurement of x_1 and x_2 is based on floats, which have to move a certain distance ϵ to register a change. It can be implemented by introducing a minimum deviation in the continuous state variables between the discrete transitions. The regularized automaton, WT_ϵ^S , is presented in Figure 5. Again one can show that WT_ϵ^S accepts a unique non-Zeno execution for each initial state. We can relate the state of WT_ϵ^S to the state of WT through $\phi(q_i, (x_1, x_2, x_3, x_4)) = (q_i, (x_1, x_2))$, for $i = 1, 2$. Figure 6 shows simulation results for WT_ϵ^S with $\epsilon = 0.1$ and 0.01 and the parameters given above. As for the temporal regularization, the execution of WT_ϵ^S converges to the execution of WT over the interval (τ_0, τ_∞) . For $t > \tau_\infty$, however, the execution converges to $x_1(t) = x_2(t) = -(t - \tau_\infty)/2 + 1$, which is different from the limit in the case of temporal regularization.

Bouncing Ball Automaton

Next, we consider temporal and dynamic regularizations of the bouncing ball automaton. Throughout we assume $c > 1$ so that BB is Zeno.

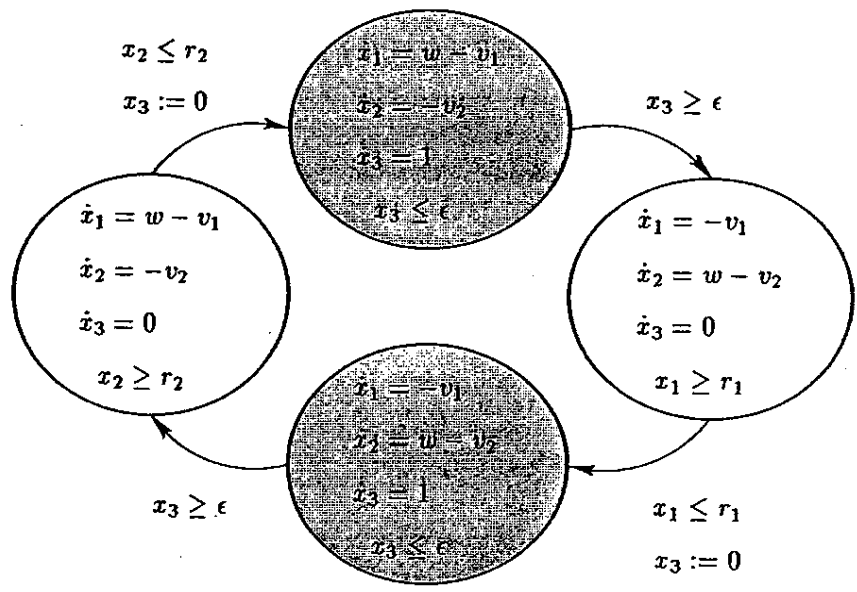


Fig. 3. Temporal regularization of the water tank automaton.

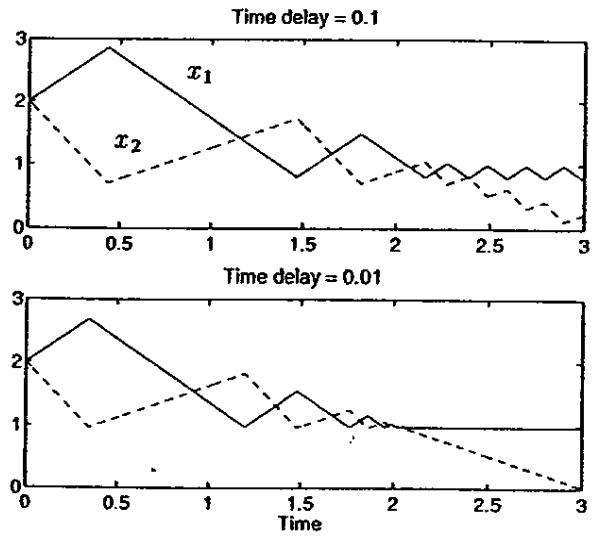


Fig. 4. Simulation of the temporally regularized water tank automaton.

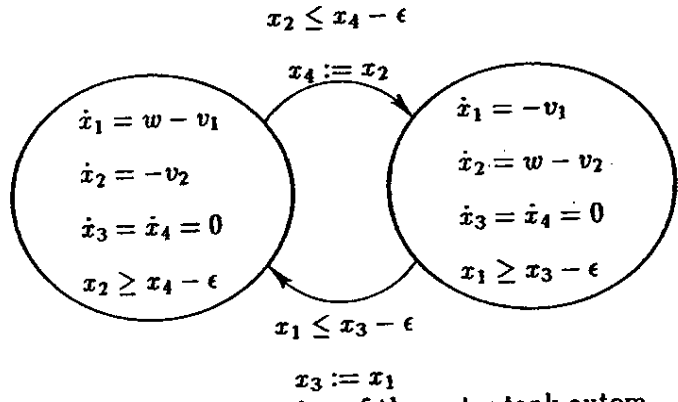


Fig. 5. Spatial regularization of the water tank automaton

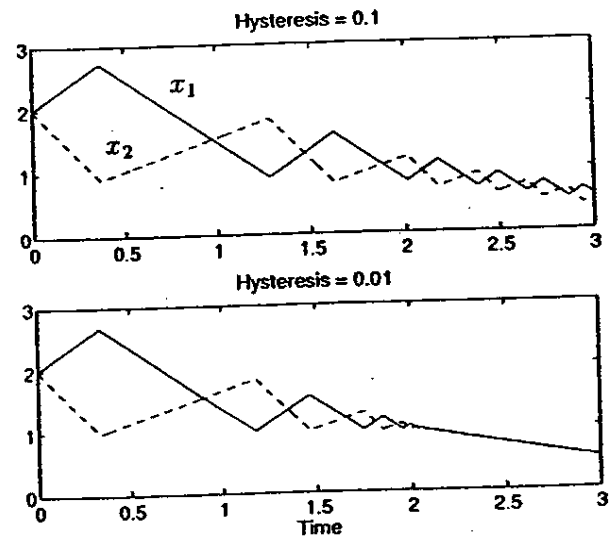


Fig. 6. Simulation of the spatially regularized water tank automaton