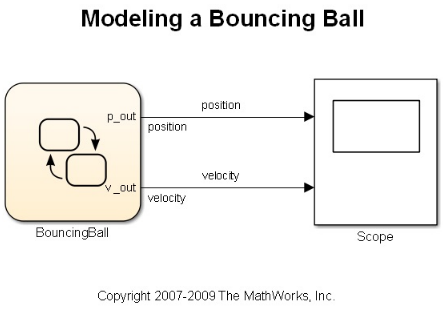
Contents

1 Bouncing Ball Example

The modeling and simulation of the bouncing ball example can be found in [1]. One can refer to step-by-step instructions for modeling a bouncing ball as a Stateflow chart [2].

The dynamics of a bouncing ball can be defined in terms of two continuous time variables, namely the position and the velocity of the ball, as shown in Fig. 1.

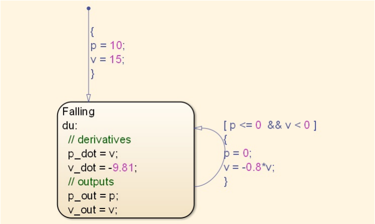

Modeling a Bouncing Ball

Copyright 2007-2009 The MathWorks, Inc.

Figure 1: Simulink diagram for the bouncing ball example

In order to model this in Stateflow, you first need to create a chart whose "Update Method" is "continuous" and which has "Enable zero-crossing detection" option checked. Once this is done, you can create local variables whose "Update Method" is "continuous". For this model, we create two local variables p and v whose update method is "continuous". When you create a "continuous" local variable, you can also define its time-derivative by using the notation *varname|_dot|* where varname is the name of the continuous variable. Thus the variable *p_dot* refers to the time derivative of the position variable p. You automatically get access to this variable once you declare p to have "continuous" update.

Using this convention, the bouncing ball can be modeled using the following Stateflow chart, as shown in Fig. 2.



```
{
p = 10;
v = 15;
}
```

```
Falling
du:
  // derivatives
  p_dot = v;
  v_dot = -9.81;
  // outputs
  p_out = p;
  v_out = v;
```

```
[ p <= 0  && v < 0 ]
{
p = 0;
v = -0.8*v;
}
```

Figure 2: Stateflow chart for the bouncing ball example

For most of the time, the ball is just freely falling under the influence of gravity. The dynamics in this "mode" are specified in the "during" section of the "Falling" state. The sudden resets in position and velocity which occurs when the ball bounces are modeled using a self-transition with a condition action which performs the state reset.

When this model is simulated, we get the familiar bouncing trajectory of the ball, as shown in Fig. 3.

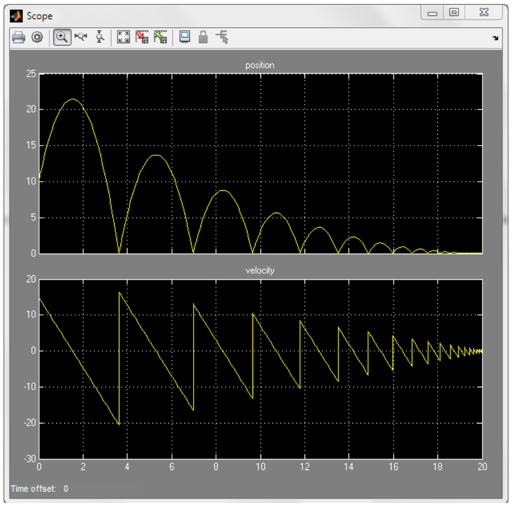


Figure 3: Simulation trajectories for the bouncing ball example

2 Thermostat Example

The simulation is built using Simulink and Stateflow in MATLAB. Fig. 4 shows the Simulink diagram for this example. The left block (named “Thermostat”) is a Stateflow block, which simulates the hybrid dynamics of the thermostat. It has two outputs, i.e. *x_output* and *q*. *x_output* represents the temperature and *q* represents the switching mode (0 as “OFF” and 1 as “ON”). The right block is a scope block, used to visualize the trajectory of temperature and switching modes.

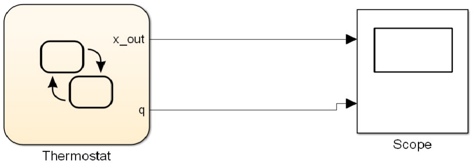
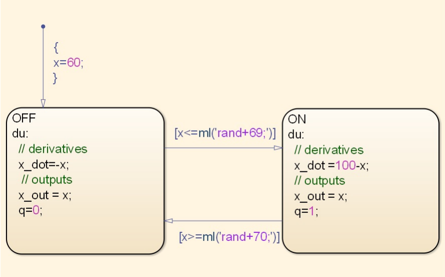


Figure 4: Simulink diagram for the thermostat example

You can explore the inside of “Thermostat” by double-clicking the Stateflow block. Fig. 5 shows how to model the thermostat example using the Stateflow chart. The chart is similar to the graphical representation of its hybrid automaton model. The two rectangular blocks represent two discrete modes, “OFF” and “ON”. In each mode, the temperature evolves according to a specified differential equation.



```
{
x=60;
}
```

```
OFF
du:
  // derivatives
  x_dot=-x;
   // outputs
  x_out = x;
  q=0;
```

```
[x<=ml('rand+69;')]
```

```
ON
du:
  // derivatives
  x_dot =100-x;
  // outputs
  x_out = x;
  q=1;
```

```
[x>=ml('rand+70;')]
```

Figure 5: Stateflow chart for the thermostat example

The initial mode (“OFF”) and temperature value (60) are specified by the arrow above the “OFF” block. The conditions of transition between “ON” and “OFF” are specified by the arrows between two modes. The condition *[x<=ml(’rand+69;’)]* guarantees that the furnace will be turned on if the temperature is below a random number between 69 and 70. Similarly, the condition *[x>=ml(’rand+70;’)]* guarantees that the furnace will be turned off if the temperature is above a random number between 70 and 71. Randomness is introduced because we know the hybrid automaton model for this example is non-deterministic. The trajectories of temperature and discrete modes should be different each time the model runs.

When this model is simulated, we can see the trajectories of temperature and switching mode from the scope block. Fig. 6 shows the trajectories when the model runs once. Fig. 7 shows the different trajectories when the model runs another time.

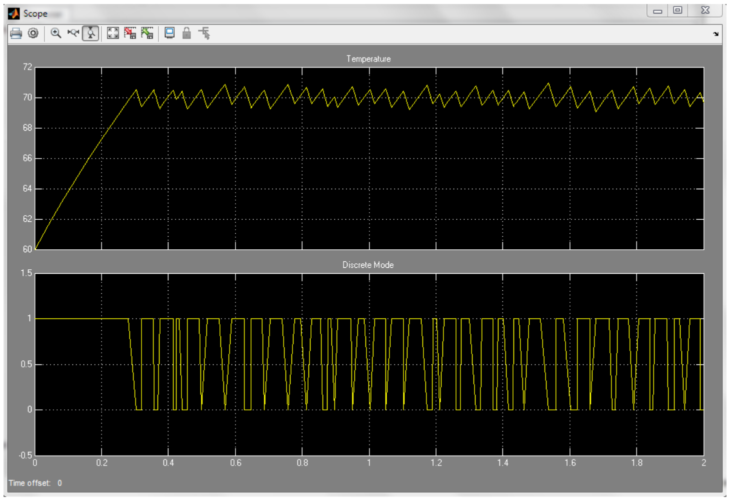


Figure 6: Simulation trajectories when the model runs once

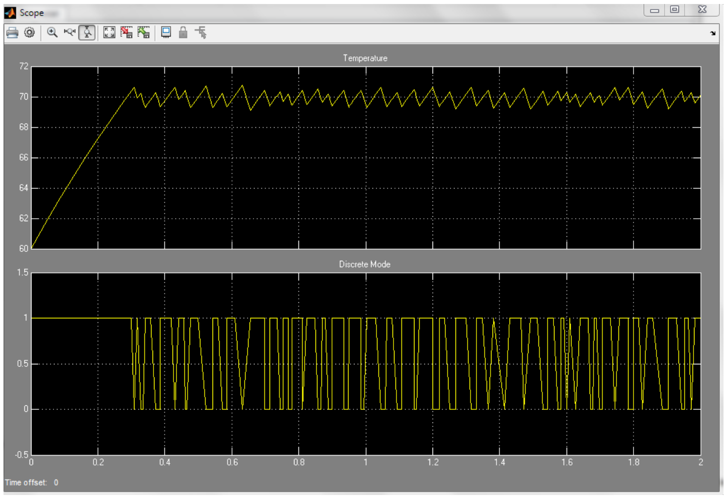


Figure 7: Simulation trajectories when the model runs another time

3 Water Tanks Example

Fig. 8 shows the Simulink diagram for this example. It consists of three blocks, which are the Stateflow, XY graph plot and scope blocks. The "two tanks" Stateflow block simulates the evolution of water levels of the two tanks. It has three outputs, *x1_output*, *x2_output* and q, representing the water level of tank 1, water level of tank 2 and switching mode respectively. The scope and XY graph blocks are used to visualize the trajectory of water levels and switching mode.

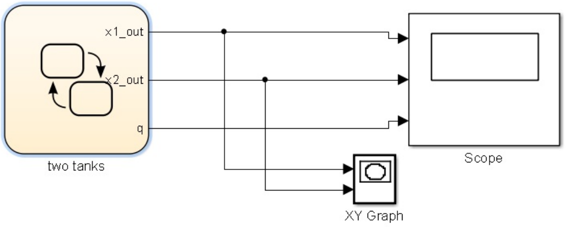
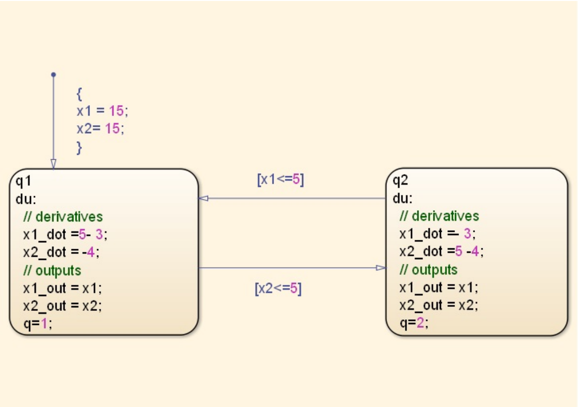


Figure 8: Simulink diagram for the water tanks example

The inside of "two tanks" State block is shown in Fig. 9. The chart is similar to the graphical representation of its hybrid automaton model. The two rectangular blocks represent two discrete modes, "the hose is switched to tank 1" and "the hose is switched to tank 2". In each mode, the water levels evolve according to the leaking and inflow rates, specified by the differential equations.

The initial water levels are specified by the arrow above the "q1" block. The water levels for both ranks are 15 at the beginning. The conditions of transition between "q1" and "q2" are specified by the arrows between two modes. The condition *[x2<=5]* means that the hose will be switched to tank 2 instantaneous whenever the water level of tank 2 falls down to 5. Similarly, the condition *[x1<=5]* guarantees that the hose will be switched to tank 1 instantaneous whenever the water level of tank 1 falls down to 5.

From Fig. 10 and 11, it can be seen that the water levels are always above 5, which verifies that the goal of keeping the water levels of the tanks above certain level is achieved. However, a drawback of this modeling is the occurrence of Zeno phenomenon. The switching between the two tanks becomes faster and faster, and the state trajectories will converge to the point $(5, 5)$, which cannot happen in practice. This is because there are infinite number of switches within a finite time interval. This phenomenon can be captured by the model and an error message is given, as in Fig. 12.

[H] 

```
{
x1 = 15;
x2= 15;
}
```

```
q1
du:
 // derivatives
 x1_dot =5- 3;
 x2_dot = -4;
 // outputs
 x1_out = x1;
 x2_out = x2;
 q=1;
```

```
[x1<=5]
```

```
[x2<=5]
```

```
q2
du:
 // derivatives
 x1_dot =- 3;
 x2_dot =5 -4;
 // outputs
 x1_out = x1;
 x2_out = x2;
 q=2;
```

Figure 9: Stateflow chart for the water tanks example

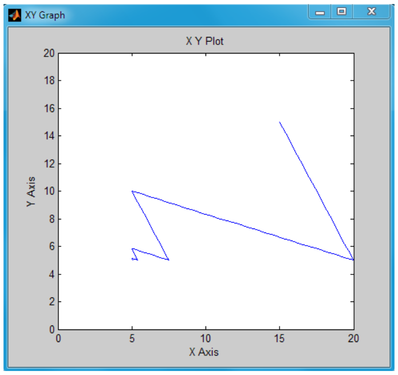


Figure 10: XY plot of the trajectory of water levels

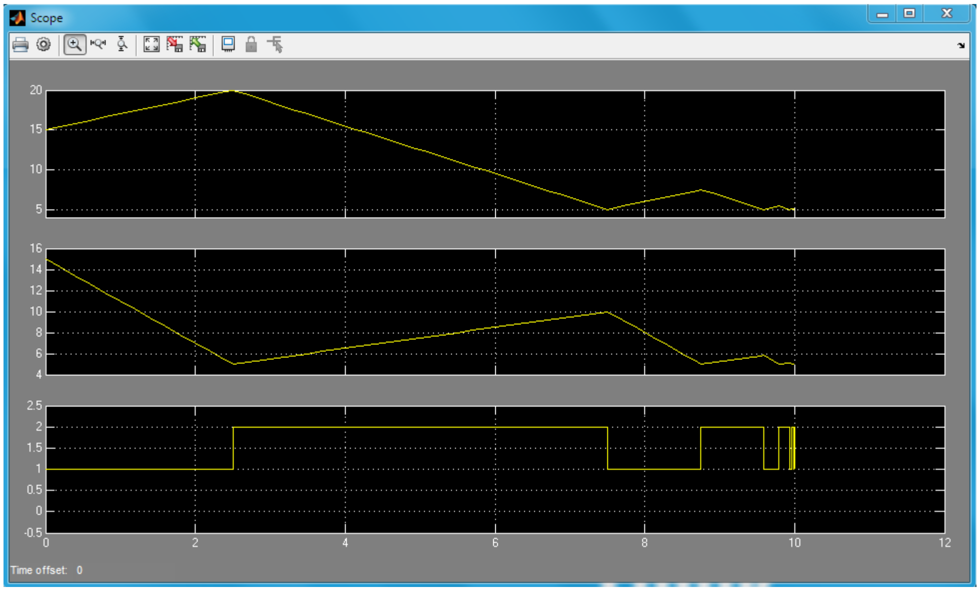
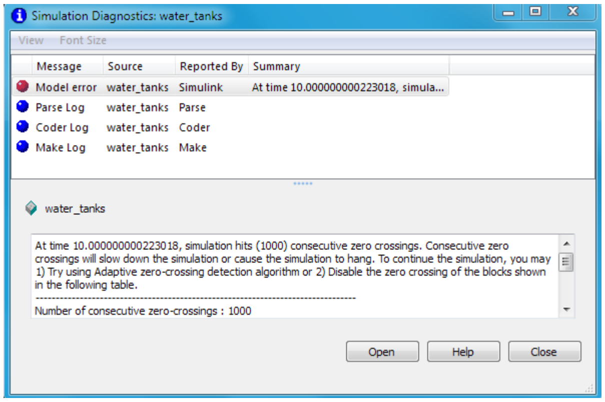


Figure 11: The trajectory of water levels and switching mode along time



Message	Source	Reported By	Summary
Model error	water_tanks	Simulink	At time 10.000000000223018, simula...
Parse Log	water_tanks	Parse	
Coder Log	water_tanks	Coder	
Make Log	water_tanks	Make	

water_tanks

```
At time 10.000000000223018, simulation hits (1000) consecutive zero crossings. Consecutive zero
crossings will slow down the simulation or cause the simulation to hang. To continue the simulation, you may
1) Try using Adaptive zero-crossing detection algorithm or 2) Disable the zero crossing of the blocks shown
in the following table.
--------------------------------------------------------------------------------
Number of consecutive zero-crossings : 1000
```

Figure 12: The error message due to Zeno behavior

4 Stable and Unstable Switched Systems

Fig. 13 shows the Simulink diagram for a switched system with two stable LTI subsystems

$$\dot{x} = A_1 x,\ \dot{x} = A_2 x$$

where $A_1 = \begin{bmatrix} -1 & -100 \\ 10 & -1 \end{bmatrix}$, $A_2 = \begin{bmatrix} -1 & 10 \\ -100 & -1 \end{bmatrix}$. Notice that both A_1 and $A_2$are are exponentially stable since all eigenvalues of A_1 and A_2 have negative real part. In addition, they have a common equilibrium point, which is the origin. The inside of "two stable systems" State block shows the dynamics of the two subsystems and the switching law. However, this simple hybrid system is unstable, since a divergent trajectory can be generated for an initial state even very close to the equilibrium point. See Fig. 14 for illustrations.

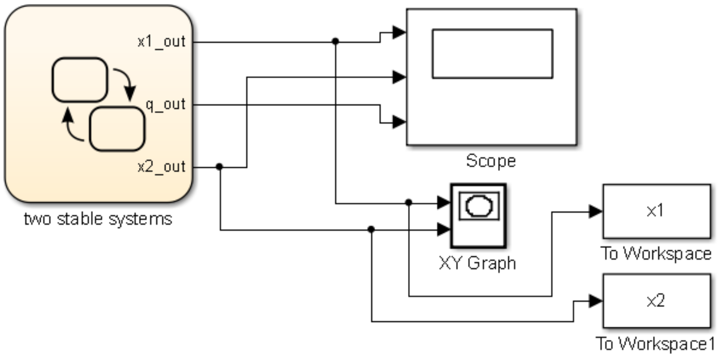


Figure 13: Simulink diagram for a switched system with two stable systems example

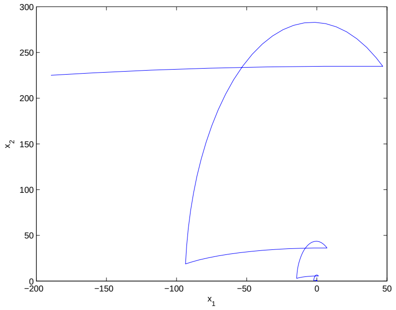


Figure 14: XY plot of the switched system with stable LTI subsystems

Fig. () shows the Simulink diagram for a switched system with two unstable

LTI subsystems

$$\dot{x} = A_1 x,\ \dot{x} = A_2 x$$

where $A_1 = \begin{bmatrix} 1 & -100 \\ 10 & 1 \end{bmatrix}$, $A_2 = \begin{bmatrix} 1 & 10 \\ -100 & 1 \end{bmatrix}$. Notice that both A_1 and A_2 are unstable since both have positive real part eigenvalues. However, the overall hybrid system is exponentially stable. This is illustrated in Fig. 16 as the state trajectory approaches the origin as time goes to infinity.

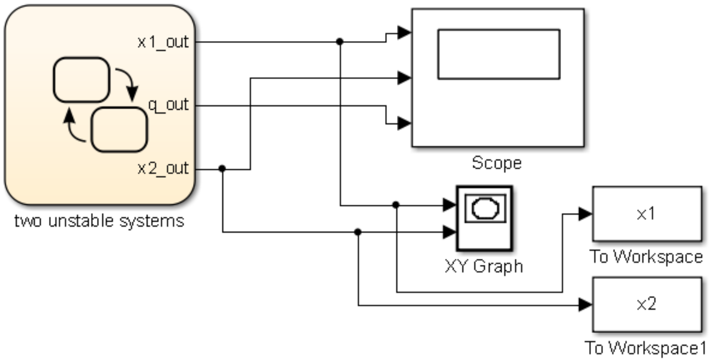


Figure 15: Simulink diagram for a switched system with two unstable systems example

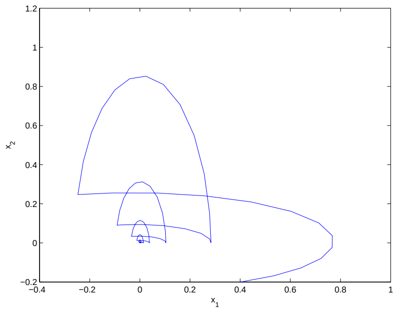


Figure 16: XY plot of the switched system with unstable LTI subsystems

5 Control of an Inverted Pendulum (can be put into Chapter 3)

This is a motivating example in [3], originally from [4]

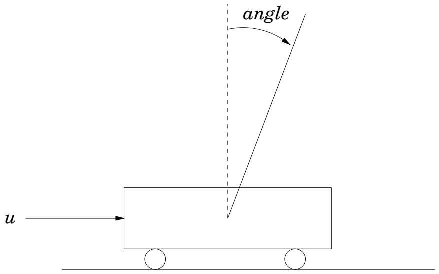


Figure 3: Inverted pendulum on a cart

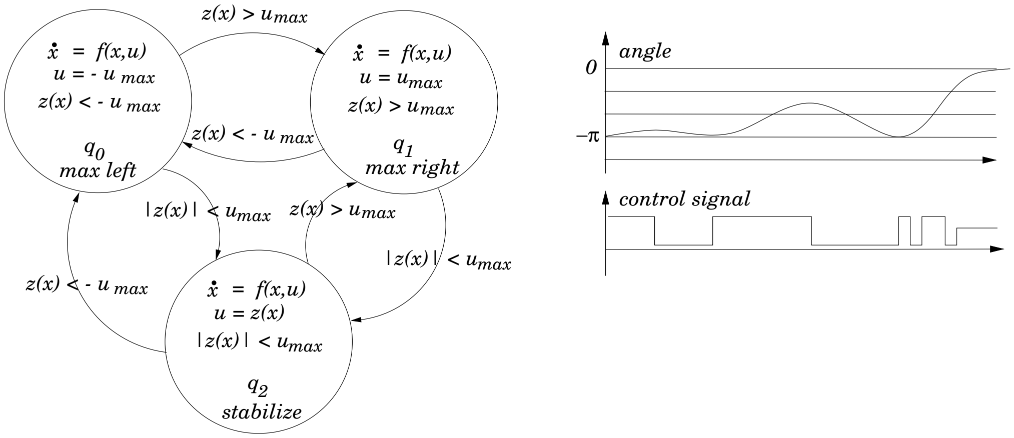


Figure 4: Hybrid control strategy and results for inverted pendulum

- regulate the inverted pendulum to an upright position using the linear acceleration of the pivot as control input u
- $f(x,u) = [x_2, g\sin x_1 - u\cos x_1]^T$, $z(x) = k[x_2^2/2 + g(\cos x_1 - 1)]\mathrm{sgn}(x_2 \sin x_1)$ where x_1 is the pendulum angle and x_2 is the angular velocity
- g is the acceleration due to gravity, $k > 0$

6 Piecewise Quadratic Lyapunov Function (can be put into 3.4.1)

This is an example in [5] for Piecewise Quadratic Lyapunov Function.

Theorem 9 (Piecewise Quadratic Lyapunov Function) *$H = (S, \mathrm{Init}, f, \mathrm{Dom}, R)$ with equilibrium $x_e = 0$. Assume that for all i:*

- $f(q_i, x) = A_i x, A_i \in \mathbb{R}^{n\times n}$
- $\mathrm{Dom} = \cup_i \{q_i\} \times \{x \in \mathbb{R}^n : E_{i1}x \geq 0, \ldots, E_{in}x \geq 0\}$
- $\mathrm{Init} \subseteq \mathrm{Dom}$
- *for all $x \in \mathbb{R}^n$*
  $$|R(q_i, x)| = \begin{cases} 1 & \text{if } (q_i, x) \in \partial\mathrm{Dom} \\ 0 & \text{otherwise} \end{cases} \tag{46}$$
  such that
  $$(q_k, x') \in R(q_i, x) \Rightarrow F_k x = F_i x, q_k \neq q_i, x' = x \tag{47}$$
  where $F_k, F_i \in \mathbb{R}^{n\times n}$.

Furthermore, assume that for all $\chi \in \mathcal{E}_H^\infty$, $\tau_\infty(\chi) = \infty$. Then, if there exists $U_i = U_i^T$, $W_i = W_i^T$, and $M = M^T$ such that $P_i = F_i^T M F_i$ satisfies:

$$\begin{aligned} A_i^T P_i + P_i A_i + E_i^T U_i E_i &< 0 \quad (48) \\ P_i - E_i^T W_i E_i &> 0 \quad (49) \end{aligned}$$

where U_i, W_i are non-negative, then $x_e = 0$ is asymptotically stable.

Example 3: Consider the hybrid automaton of Figure 11 with

$$A_1 = A_3 = \begin{bmatrix} -0.1 & 1 \\ -5 & -0.1 \end{bmatrix}, A_2 = A_4 = \begin{bmatrix} -0.1 & 5 \\ -1 & -0.1 \end{bmatrix} \tag{50}$$

Here, we may choose

$$E_1 = -E_3 = \begin{bmatrix} -1 & 1 \\ -1 & -1 \end{bmatrix}, E_2 = -E_4 = \begin{bmatrix} -1 & 1 \\ 1 & 1 \end{bmatrix} \tag{51}$$

and

$$F_i = \begin{bmatrix} E_i \\ I \end{bmatrix} \forall i \in \{1, 2, 3, 4\} \tag{52}$$

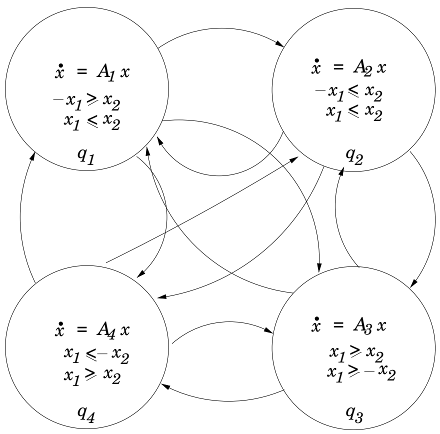


Figure 11: Example 3

The eigenvalues of A_i are $-1/10 \pm \sqrt{5}i$. The evolution of the continuous state is shown in Figure 12. We can use a Lyapunov function given by:

$$P_1 = P_3 = \begin{bmatrix} 5 & 0 \\ 0 & 1 \end{bmatrix}, P_2 = P_4 = \begin{bmatrix} 1 & 0 \\ 0 & 5 \end{bmatrix} \tag{53}$$

to prove asymptotic stability of the hybrid automaton.

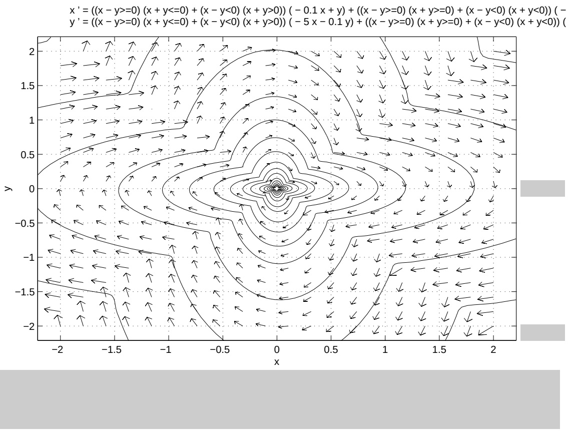


Figure 12: Example 3

7 Semi-automatic transmission [10] (can be put into Chapter 2)

The gear shift example describes a control design problem where both the continuous and the discrete controls need to be determined. Figure 6 shows a model of a car with a gear box having four gears. The longitudinal position of the car along the road is denoted by x_1 and its velocity by x_2 (lateral dynamics are ignored). The model has two control signals: the gear denoted $\texttt{gear} \in \{1, \ldots, 4\}$ and the throttle position denoted $u \in [u_{\min}, u_{\max}]$. These may both be considered as inputs to the system, while the position and the velocity are outputs. The gear shift is necessary because little power can be generated by the engine at very low or very high engine speed. The function α_i represents the efficiency of gear i.

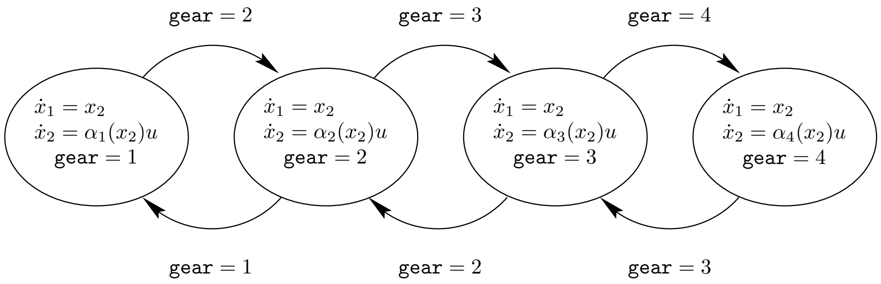
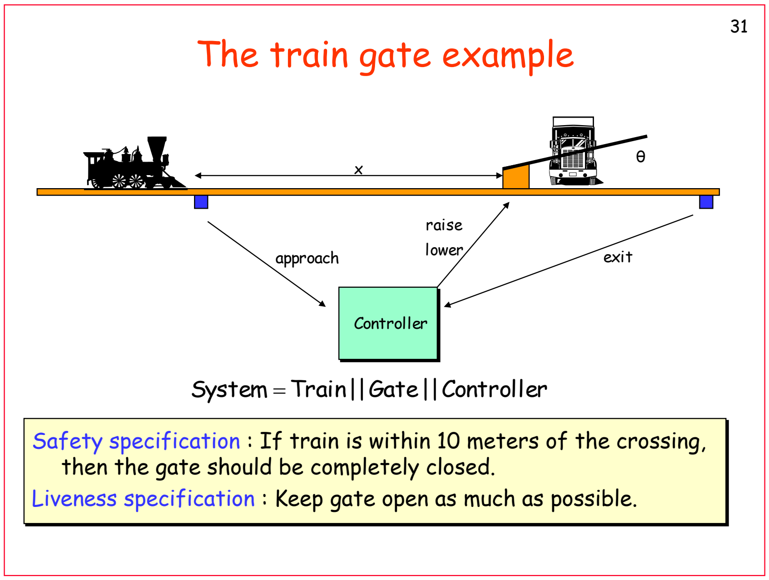


Figure 6: A hybrid system modeling a car with four gears.

Several interesting control problems can be posed for this simple car model, including the following: What is the optimal control strategy to drive from $x = (a, 0)$ to $(b, 0)$ in minimum time? The problem is non-trivial if the reasonable assumption is included that each gear shift takes a certain amount of time. The optimal controller, which can be modeled as a hybrid system, may be derived using the theory of optimal control of hybrid systems [EOLSS,6.43.28.5].

8 Train Gate Example [6, 7] (can be put into 5.4)

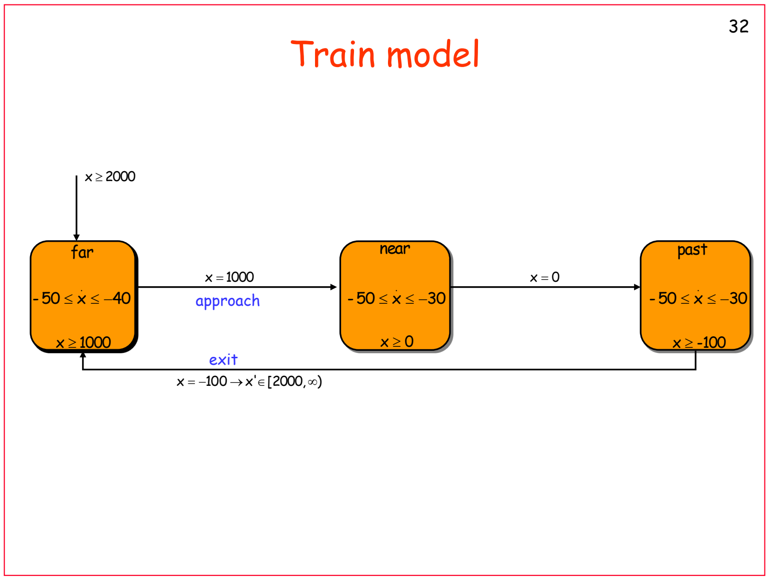


The train gate example

$\text{System} = \text{Train} \,||\, \text{Gate} \,||\, \text{Controller}$

Safety specification : If train is within 10 meters of the crossing, then the gate should be completely closed.

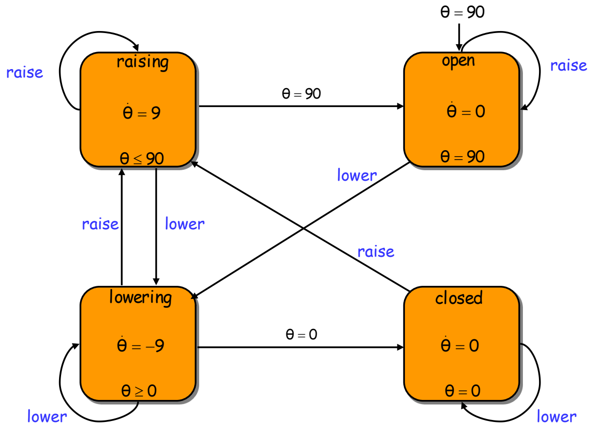
Liveness specification : Keep gate open as much as possible.



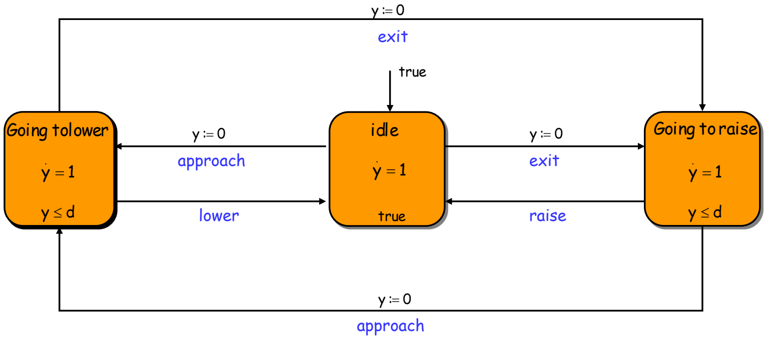
Train model

$x \ge 2000$
far
$-50 \le \dot{x} \le -40$
$x \ge 1000$
$x = 1000$
approach
near
$-50 \le \dot{x} \le -30$
$x \ge 0$
$x = 0$
past
$-50 \le \dot{x} \le -30$
$x \ge -100$
exit
$x = -100 \to x' \in [2000, \infty)$

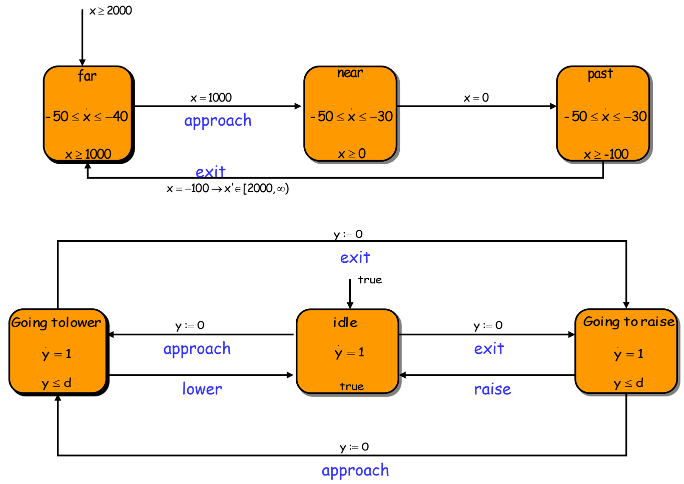
Gate model



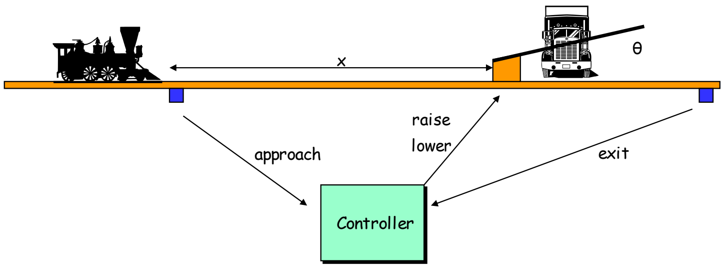
Controller model



Synchronized transitions



Verifying the controller

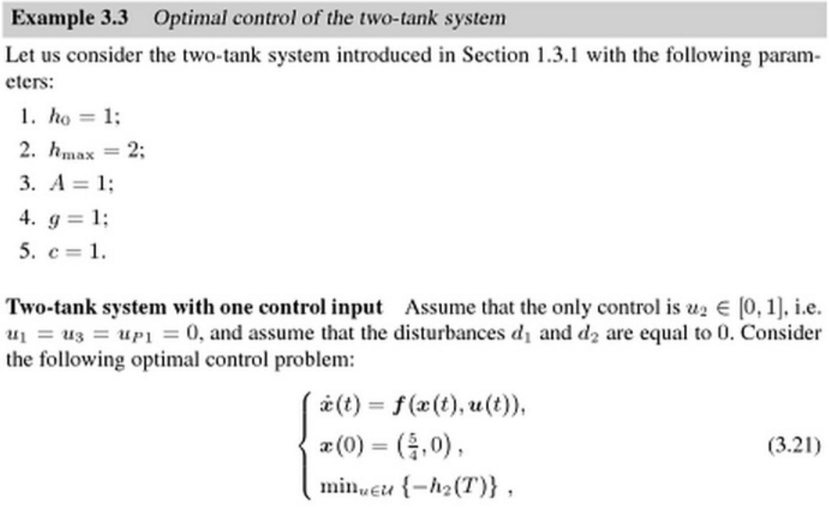


System = Train || Gate || Controller

Safety specification : Can we avoid the set $\theta > 0 \wedge (-10 \leq x \leq 10)$?

Parametric verification : YES if $d \leq \frac{49}{5}$

9 Optimization control of two tank system [8] (can be put into 4.2)



Example 3.3 *Optimal control of the two-tank system*

Let us consider the two-tank system introduced in Section 1.3.1 with the following parameters:

1. $h_0 = 1$;
2. $h_{\max} = 2$;
3. $A = 1$;
4. $g = 1$;
5. $c = 1$.

Two-tank system with one control input Assume that the only control is $u_2 \in [0, 1]$, i.e. $u_1 = u_3 = u_{P1} = 0$, and assume that the disturbances d_1 and d_2 are equal to 0. Consider the following optimal control problem:

$$\begin{cases} \dot{\boldsymbol{x}}(t) = \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}(t)), \\ \boldsymbol{x}(0) = \left(\frac{5}{4}, 0\right), \\ \min_{u \in \mathcal{U}} \left\{-h_2(T)\right\} , \end{cases} \tag{3.21}$$

with $\boldsymbol{x}(t) = (h_1(t), h_2(t))$ and

$$\boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}(t)) = \begin{pmatrix} -\sqrt{2\,(h_1(t) - h_2(t))}\, u_2(t) \\ \sqrt{2\,(h_1(t) - h_2(t))}\, u_2(t) \end{pmatrix} . \tag{3.22}$$

We show that the trajectory with control $u_2(t) \equiv 1$ satisfies the hybrid maximum principle. First of all, note that the trajectory associated to this control function is given by

$$\begin{cases} h_1(t) = \frac{5}{8} + \frac{1}{4}\left(\sqrt{\frac{5}{2}} - 2t\right)^2, \\ h_2(t) = \frac{5}{8} - \frac{1}{4}\left(\sqrt{\frac{5}{2}} - 2t\right)^2, \end{cases} \tag{3.23}$$

if $0 \leq t \leq \frac{1}{2}\sqrt{\frac{5}{2}}$, while

$$\begin{cases} h_1(t) = \frac{5}{8}, \\ h_2(t) = \frac{5}{8} \end{cases} \tag{3.24}$$

if $t \geq \frac{1}{2}\sqrt{\frac{5}{2}}$. Moreover, we have

$$q(t) = 2 \ \text{ if } \ 0 < t < \frac{1}{2\sqrt{2}}\left(\sqrt{5} - \sqrt{3}\right)$$
$$q(t) = 1 \ \text{ if } \ \frac{1}{2\sqrt{2}}\left(\sqrt{5} - \sqrt{3}\right) < t < \frac{1}{2}\sqrt{\frac{5}{2}},$$

i.e. the switching time t_1 is $\frac{1}{2\sqrt{2}}\left(\sqrt{5} - \sqrt{3}\right)$. Without loss of generality we put $T = \frac{1}{2}\sqrt{\frac{5}{2}}$.

If $\boldsymbol{\lambda}(t) = (\lambda_1(t), \lambda_2(t))$, then the transversality condition implies

$$-\boldsymbol{\lambda}(T) + (0, \lambda_0) = 0.$$

Moreover we have

$$\begin{cases} \dot{\lambda}_1(t) = \lambda_1(t)\frac{1}{\sqrt{2(h_1(t) - h_2(t))}} - \lambda_2(t)\frac{1}{\sqrt{2(h_1(t) - h_2(t))}}, \\ \dot{\lambda}_2(t) = -\lambda_1(t)\frac{1}{\sqrt{2(h_1(t) - h_2(t))}} + \lambda_2(t)\frac{1}{\sqrt{2(h_1(t) - h_2(t))}}, \end{cases}$$

and so

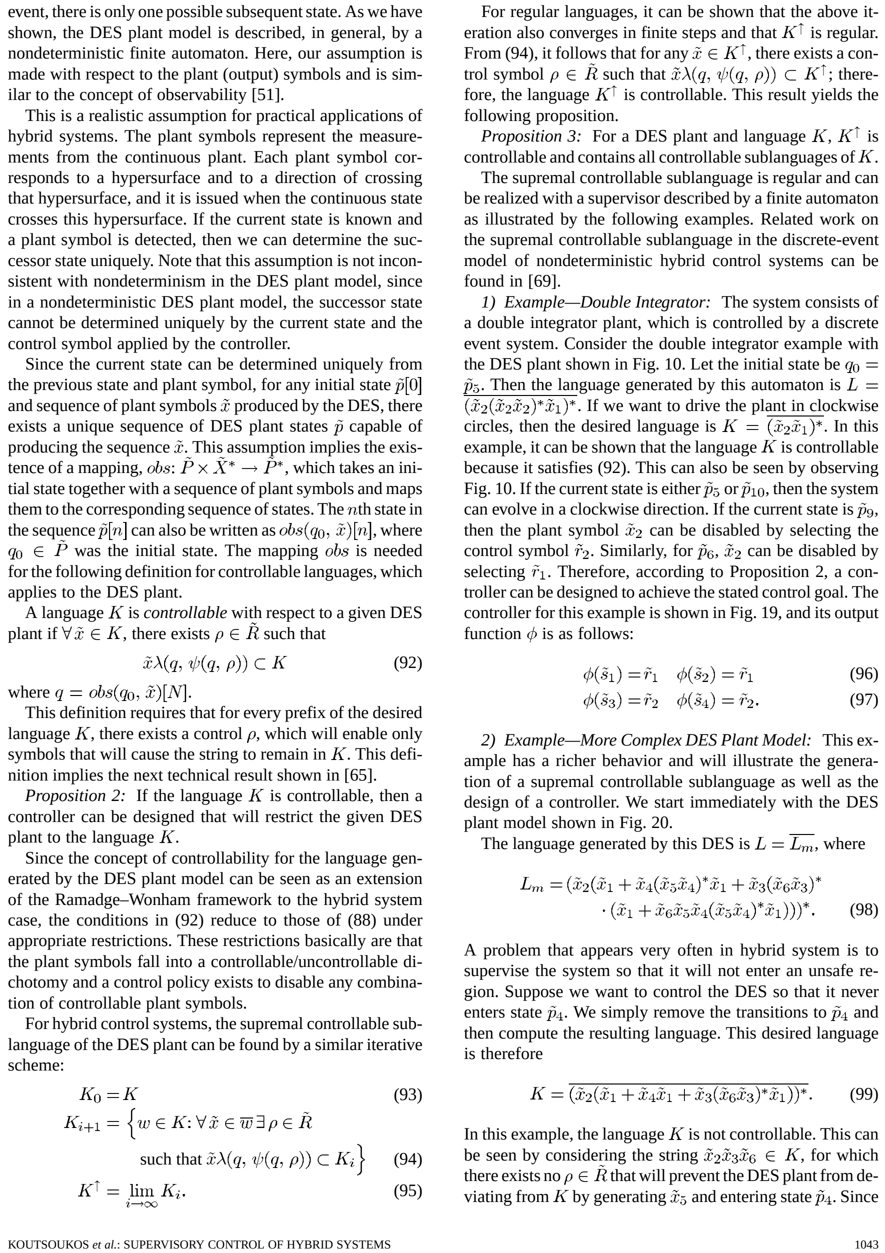
$$\begin{cases} \lambda_1(t) = \frac{\lambda_0}{2}\left(1 - \exp\left(\int_T^t \frac{2}{\sqrt{2(h_1(s) - h_2(s))}}\, ds\right)\right), \\ \lambda_2(t) = \frac{\lambda_0}{2}\left(1 + \exp\left(\int_T^t \frac{2}{\sqrt{2(h_1(s) - h_2(s))}}\, ds\right)\right). \end{cases}$$

Note that the nontriviality condition implies $\lambda_0 > 0$ and the function $\lambda_1(t) - \lambda_2(t) < 0$ for every t. We have

$$\begin{aligned} \tilde{H}(t) &= \sup\left\{(\lambda_2(t) - \lambda_1(t))\sqrt{2(h_1(t) - h_2(t))}\, u_2 \ : u_2 \in [0, 1]\right\} \\ &= (\lambda_2(t) - \lambda_1(t))\sqrt{2(h_1(t) - h_2(t))}; \end{aligned}$$

hence the Hamiltonian maximization property implies that the control u_2 is equal to 1.

10 Hybrid supervisory control[9] (4 examples in total, can be put into 6.3)



event, there is only one possible subsequent state. As we have shown, the DES plant model is described, in general, by a nondeterministic finite automaton. Here, our assumption is made with respect to the plant (output) symbols and is similar to the concept of observability [51].

This is a realistic assumption for practical applications of hybrid systems. The plant symbols represent the measurements from the continuous plant. Each plant symbol corresponds to a hypersurface and to a direction of crossing that hypersurface, and it is issued when the continuous state crosses this hypersurface. If the current state is known and a plant symbol is detected, then we can determine the successor state uniquely. Note that this assumption is not inconsistent with nondeterminism in the DES plant model, since in a nondeterministic DES plant model, the successor state cannot be determined uniquely by the current state and the control symbol applied by the controller.

Since the current state can be determined uniquely from the previous state and plant symbol, for any initial state $\tilde{p}[0]$ and sequence of plant symbols $\tilde{x}$ produced by the DES, there exists a unique sequence of DES plant states $\tilde{p}$ capable of producing the sequence $\tilde{x}$. This assumption implies the existence of a mapping, $obs\colon \tilde{P} \times \tilde{X}^* \rightarrow \tilde{P}^*$, which takes an initial state together with a sequence of plant symbols and maps them to the corresponding sequence of states. The nth state in the sequence $\tilde{p}[n]$ can also be written as $obs(q_0, \tilde{x})[n]$, where $q_0 \in \tilde{P}$ was the initial state. The mapping obs is needed for the following definition for controllable languages, which applies to the DES plant.

A language K is *controllable* with respect to a given DES plant if $\forall \tilde{x} \in K$, there exists $\rho \in \tilde{R}$ such that

$$\tilde{x}\lambda(q, \psi(q, \rho)) \subset K \tag{92}$$

where $q = obs(q_0, \tilde{x})[N]$.

This definition requires that for every prefix of the desired language K, there exists a control ρ, which will enable only symbols that will cause the string to remain in K. This definition implies the next technical result shown in [65].

Proposition 2: If the language K is controllable, then a controller can be designed that will restrict the given DES plant to the language K.

Since the concept of controllability for the language generated by the DES plant model can be seen as an extension of the Ramadge–Wonham framework to the hybrid system case, the conditions in (92) reduce to those of (88) under appropriate restrictions. These restrictions basically are that the plant symbols fall into a controllable/uncontrollable dichotomy and a control policy exists to disable any combination of controllable plant symbols.

For hybrid control systems, the supremal controllable sublanguage of the DES plant can be found by a similar iterative scheme:

$$K_0 = K \tag{93}$$

$$K_{i+1} = \Big\{ w \in K \colon \forall \tilde{x} \in \overline{w}\, \exists\, \rho \in \tilde{R} \text{ such that } \tilde{x}\lambda(q, \psi(q, \rho)) \subset K_i \Big\} \tag{94}$$

$$K^{\uparrow} = \lim_{i \to \infty} K_i. \tag{95}$$

For regular languages, it can be shown that the above iteration also converges in finite steps and that $K^{\uparrow}$ is regular. From (94), it follows that for any $\tilde{x} \in K^{\uparrow}$, there exists a control symbol $\rho \in \tilde{R}$ such that $\tilde{x}\lambda(q, \psi(q, \rho)) \subset K^{\uparrow}$; therefore, the language $K^{\uparrow}$ is controllable. This result yields the following proposition.

Proposition 3: For a DES plant and language K, $K^{\uparrow}$ is controllable and contains all controllable sublanguages of K.

The supremal controllable sublanguage is regular and can be realized with a supervisor described by a finite automaton as illustrated by the following examples. Related work on the supremal controllable sublanguage in the discrete-event model of nondeterministic hybrid control systems can be found in [69].

1) Example—Double Integrator: The system consists of a double integrator plant, which is controlled by a discrete event system. Consider the double integrator example with the DES plant shown in Fig. 10. Let the initial state be $q_0 = \tilde{p}_5$. Then the language generated by this automaton is $L = \overline{(\tilde{x}_2(\tilde{x}_2\tilde{x}_2)^*\tilde{x}_1)^*}$. If we want to drive the plant in clockwise circles, then the desired language is $K = \overline{(\tilde{x}_2\tilde{x}_1)^*}$. In this example, it can be shown that the language K is controllable because it satisfies (92). This can also be seen by observing Fig. 10. If the current state is either $\tilde{p}_5$ or $\tilde{p}_{10}$, then the system can evolve in a clockwise direction. If the current state is $\tilde{p}_9$, then the plant symbol $\tilde{x}_2$ can be disabled by selecting the control symbol $\tilde{r}_2$. Similarly, for $\tilde{p}_6$, $\tilde{x}_2$ can be disabled by selecting $\tilde{r}_1$. Therefore, according to Proposition 2, a controller can be designed to achieve the stated control goal. The controller for this example is shown in Fig. 19, and its output function ϕ is as follows:

$$\phi(\tilde{s}_1) = \tilde{r}_1 \quad \phi(\tilde{s}_2) = \tilde{r}_1 \tag{96}$$

$$\phi(\tilde{s}_3) = \tilde{r}_2 \quad \phi(\tilde{s}_4) = \tilde{r}_2. \tag{97}$$

2) Example—More Complex DES Plant Model: This example has a richer behavior and will illustrate the generation of a supremal controllable sublanguage as well as the design of a controller. We start immediately with the DES plant model shown in Fig. 20.

The language generated by this DES is $L = \overline{L_m}$, where

$$L_m = (\tilde{x}_2(\tilde{x}_1 + \tilde{x}_4(\tilde{x}_5\tilde{x}_4)^*\tilde{x}_1 + \tilde{x}_3(\tilde{x}_6\tilde{x}_3)^* \cdot (\tilde{x}_1 + \tilde{x}_6\tilde{x}_5\tilde{x}_4(\tilde{x}_5\tilde{x}_4)^*\tilde{x}_1)))^*. \tag{98}$$

A problem that appears very often in hybrid system is to supervise the system so that it will not enter an unsafe region. Suppose we want to control the DES so that it never enters state $\tilde{p}_4$. We simply remove the transitions to $\tilde{p}_4$ and then compute the resulting language. This desired language is therefore

$$K = \overline{(\tilde{x}_2(\tilde{x}_1 + \tilde{x}_4\tilde{x}_1 + \tilde{x}_3(\tilde{x}_6\tilde{x}_3)^*\tilde{x}_1))^*}. \tag{99}$$

In this example, the language K is not controllable. This can be seen by considering the string $\tilde{x}_2\tilde{x}_3\tilde{x}_6 \in K$, for which there exists no $\rho \in \tilde{R}$ that will prevent the DES plant from deviating from K by generating $\tilde{x}_5$ and entering state $\tilde{p}_4$. Since

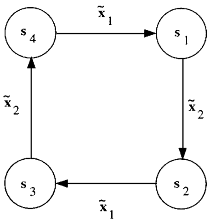
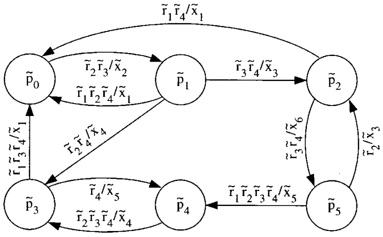


Fig. 19. Controller for the double integrator.



$\tilde{r}_1\tilde{r}_4/\tilde{x}_1$
$\tilde{p}_0$
$\tilde{r}_2\tilde{r}_3/\tilde{x}_2$
$\tilde{r}_1\tilde{r}_2\tilde{r}_4/\tilde{x}_1$
$\tilde{p}_1$
$\tilde{r}_3\tilde{r}_4/\tilde{x}_3$
$\tilde{p}_2$
$\tilde{r}_1\tilde{r}_3\tilde{r}_4/\tilde{x}_1$
$\tilde{r}_2\tilde{r}_4/\tilde{x}_4$
$\tilde{r}_3\tilde{r}_4/\tilde{x}_6$
$\tilde{r}_2/\tilde{x}_3$
$\tilde{p}_3$
$\tilde{r}_4/\tilde{x}_5$
$\tilde{r}_2\tilde{r}_3\tilde{r}_4/\tilde{x}_4$
$\tilde{p}_4$
$\tilde{r}_1\tilde{r}_2\tilde{r}_3\tilde{r}_4/\tilde{x}_5$
$\tilde{p}_5$

Fig. 20. DES plant model.

K is not controllable, we find the supremal controllable sublanguage of K as defined in (95). The supremal controllable sublanguage is

$$K^{\uparrow} = K_1 = \overline{(\tilde{x}_2(\tilde{x}_1 + \tilde{x}_4\tilde{x}_1 + \tilde{x}_3\tilde{x}_1))^*}. \tag{100}$$

Obtaining a DES controller once the supremal controllable sublanguage has been found is straightforward. The controller is a DES whose language is given by $K^{\uparrow}$. Since the language $K^{\uparrow}$ is regular, the supervisor is implemented by a finite automaton that generates the language $K^{\uparrow}$. Details regarding the equivalence between finite automata and regular languages can be found in [26]. The output of the controller in each state $\phi(\tilde{s})$ is the controller symbol, which enables only transitions that are found in the controller. The existence of such a controller symbol is guaranteed by the fact that $K^{\uparrow}$ is controllable. For this example, the controller is shown in Fig. 21 and its output function ϕ is as follows:

$$\phi(\tilde{s}_1) = \tilde{r}_2 \quad \phi(\tilde{s}_2) = \tilde{r}_4 \tag{101}$$

$$\phi(\tilde{s}_3) = \tilde{r}_1 \quad \phi(\tilde{s}_4) = \tilde{r}_1. \tag{102}$$

3) Example—Distillation Column: This example uses the model of a two-product distillation column with a single feed. A complete description of the nonlinear model can be found in [44]. Here, a condensed description is given to show the source of the DES plant model and provide insight into the physical meaning of the states and events.

Fig. 22 shows the distillation column. F represents the feed flow into the column, B is the flow of bottom product out of the column, x_B is the mole fraction of the light compound in the bottom product, D is the flow of distillate out of the column, and y_D is the mole fraction of light compound in the distillate. The boilup flow is denoted by V and the reflux flow by L. All units are in kmol's and minutes. The column can be controlled by setting the feed, boilup, and reflux. In general, the goal is to have a high level of light compound in the distillate ($y_D \to 1$) and a low level of light compound in the bottom product ($x_B \to 0$).

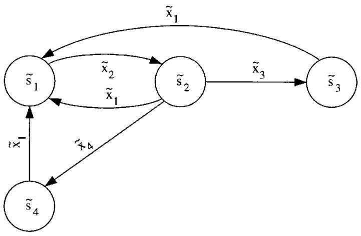


Fig. 21. DES controller.

There are 40 trays stacked vertically in the column. The state consists of the mole fractions of light compound in the liquid of each tray. The states evolve according to the following equations:

$$\begin{aligned}
2\dot{x}_1 &= (L+F_L)x_2 - Vy_1 - Bx_1\\
2\dot{x}_i &= (L+F_L)x_{i+1} + Vy_i - (L+F_L)x_i - Vy_i\\
2\dot{x}_{21} &= Lx_{22} + Vy_{20} - (L+F_L)x_{21} - Vy_{21} + F_L * x_F\\
2\dot{x}_{22} &= Lx_{23} + Vy_{21} - Lx_{22} - (V+F_V)y_{22} + F_V * y_F\\
2\dot{x}_j &= Lx_{j+1} + (V+F_V)y_j - Lx_j - (V+F_V)y_j\\
2\dot{x}_{41} &= (V+F_V)y_{40} - Lx_{41} - Dx_{41}
\end{aligned}$$

where $2 \leq i \leq 20$ and $23 \leq j \leq 40$. Trays 21 and 22 are special because they are below and above the feed location. Tray 41 is actually the condenser. The quantities y_i are the mole fractions of light compound in the vapor, given by

$$y_i = \frac{\alpha x_i}{1 + (\alpha - 1)x_i}$$

where $\alpha = 1.5$ is relative volatility. Other quantities of interest are

$$F_L = 0.6F \quad F_V = F - F_L \quad x_F = (0.5F - F_V y_F)/F_L$$

and the outputs are

$$\begin{aligned}
D &= V + F_V - L \quad B = L + F_L - V\\
y_D &= y_{41} \quad x_B = x_1.
\end{aligned}$$

To obtain a hybrid control system, appropriate control policies and plant symbols must be chosen. Their selection is based on our knowledge of the control goals and the design constraints, and it will determine the interface. Let the control policies be

$$\mathbf{r}(t) = \begin{bmatrix} L \\ V \\ F \end{bmatrix} \in \left\{ \begin{bmatrix} 9.5 \\ 10 \\ 1 \end{bmatrix}, \begin{bmatrix} 10 \\ 10 \\ 0.1 \end{bmatrix}, \begin{bmatrix} 9.5 \\ 10 \\ 2 \end{bmatrix}, \begin{bmatrix} 10 \\ 10 \\ 2 \end{bmatrix} \right\}.$$

These input values correspond to $\tilde{r}_1$, $\tilde{r}_2$, $\tilde{r}_3$, and $\tilde{r}_4$. Next, plant symbols are defined based on events as follows:

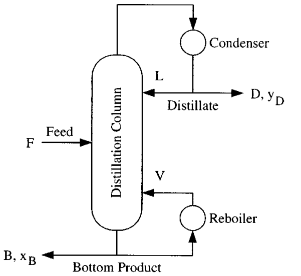


Fig. 22. Distillation column.

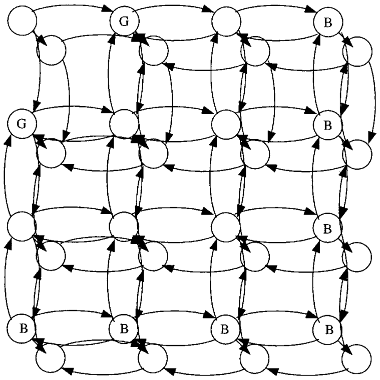


Fig. 23. DES plant for the distillation column.

$\tilde{x}_1$	$B+D$ falls below 2;
$\tilde{x}_2$	$B+D$ exceeds 2;
$\tilde{x}_3$	x_B falls below 0.13;
$\tilde{x}_4$	x_B exceeds 0.13;
$\tilde{x}_5$	x_B falls below 0.12;
$\tilde{x}_6$	x_B exceeds 0.12;
$\tilde{x}_7$	x_B falls below 0.08;
$\tilde{x}_8$	x_B exceeds 0.08;
$\tilde{x}_9$	y_D falls below 0.84;
$\tilde{x}_{10}$	y_D exceeds 0.84;
$\tilde{x}_{11}$	y_D falls below 0.85;
$\tilde{x}_{12}$	y_D exceeds 0.85;
$\tilde{x}_{13}$	y_D falls below 0.95;
$\tilde{x}_{14}$	y_D exceeds 0.95.

We would like to keep x_B below 0.13, y_D above 0.95, and the feed at 2. These conditions correspond to increased production of high-purity products. Simulations reveal that given the available controls and events, this is not possible; that is, even if the initial state is in this region, no available control policy will cause it to remain there. It is possible to drive the system close to this point, however. Specifically, our control goal shall be twofold: first, to drive the system near the ideal point, and second, to avoid having a high feed rate (2 kmol/min) when the system is not near the ideal point.

The distillation column is an example of a rather complex hybrid system. The generator was designed to recognize 14 different plant events. This leads to 32 distinct regions in the state space, and therefore, there are 32 DES plant states. Fig. 23 shows the DES plant model. The two states labeled "G" correspond to the desired operating regions of the system. This DES plant model was extracted by automating the testing process and implementing it on a computer.

A controller was obtained by automating the procedure for finding the supremal controllable sublanguage. The controller is shown in Fig. 24. This controller drives the plant from the initial state to a loop containing the two good states. Note that in this figure, the states of the controller have been labeled with the controller symbol that is generated by that state.

4) Example—Robotic Manufacturing System: An example of a free floating robotic vehicle with two articulated arms is presented. The robotic arms shown in Fig. 25 are required to obtain components from a *parts bin* and move these components to *work areas* where assembly operations are to be performed. The tasks of fetching the workpiece, transporting it to the work area, and then returning to the parts bin to fetch another workpiece are performed repeatedly. The introduction of a shared resource generates a *mutual exclusion* constraint on the system. This example is particularly interesting because of the free-floating base, which makes the dynamics quite challenging. Similar problems arise in control and coordination of modern complex engineering applications such as autonomous vehicles and multibatch chemical processes. The robotic manufacturing example described here has been used in [37] to illustrate various concepts in hybrid system theory. A simplified version of the system without the free rotating table has been used in [29] and [28] to illustrate regulatory control of hybrid systems based on discrete abstractions.

The motions of the arms are described by the following ordinary differential equations:

$$\ddot{\theta}_1 = -\dot{\theta}_1 + k(\theta_1 + \theta_b - r_1) \tag{103}$$

$$\ddot{\theta}_2 = -\dot{\theta}_2 + k(\theta_2 + \theta_b - r_2) \tag{104}$$

where θ_1 and θ_2 are the angular positions of arm 1 and arm 2 with respect to the body axis of the robot. For this example, the control law is a proportional feedback law with gain k and with reference inputs r_1 and r_2. These reference inputs represent commands that direct the arm to move to the parts bin or work area. The movement of the arms will induce a body rotation so that the total angular momentum of the system is conserved. Let $\overline{\theta}_1 = \theta_1 + \theta_b$ and $\overline{\theta}_2 = \theta_2 + \theta_b$ denote the inertial angles of the robot arms 1 and 2, respectively. The body angle θ_b with respect to the inertial frame must satisfy

$$J_b\dot{\theta}_b + J_a\dot{\overline{\theta}}_1 + J_a\dot{\overline{\theta}}_2 = 0 \tag{105}$$

where J_b and J_a are the moments of inertia for the body and arms, respectively.

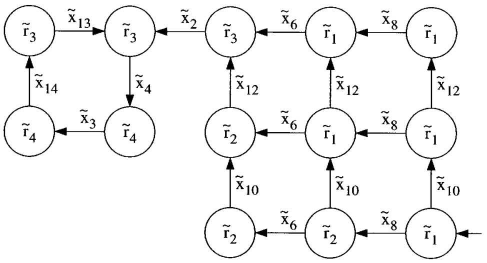


Fig. 24. Sample controller for distillation column.

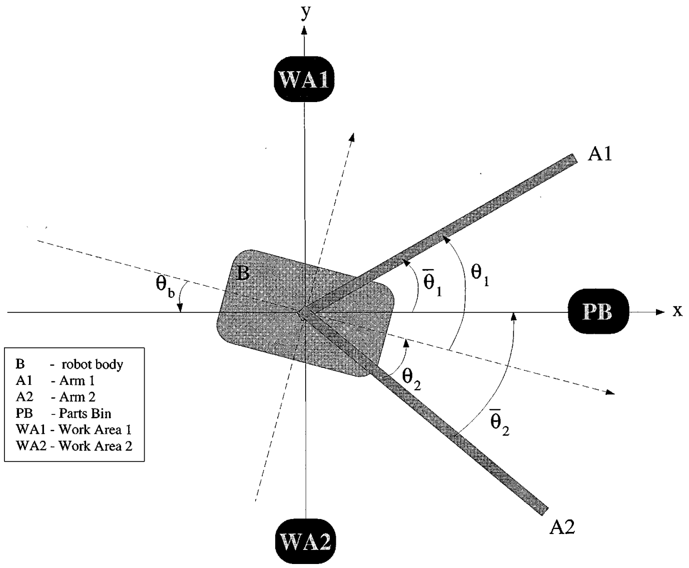


Fig. 25. Robotic manufacturing system on a free rotating platform.

The available control policies for the ith robotic arm are defined as follows:

$\tilde{r}_{i1}$	drive arm i to parts bin;
$\tilde{r}_{i2}$	drive arm i to work area;
$\tilde{r}_{i3}$	stop arm i.

Note that continuous controllers that guarantee that each command signal is executed in a suitable manner may be necessary. As discussed in Section II, it is assumed that these continuous controllers are included in the description of the plant. Next, plant symbols are defined based on events as follows:

$\tilde{x}_1$	Arm 1 approaches the parts bin;
$\tilde{x}_2$	Arm 1 enters the parts bin;
$\tilde{x}_3$	Arm 1 exits the parts bin;
$\tilde{x}_4$	Arm 1 leaves the parts bin;
$\tilde{x}_5$	Arm 2 approaches the parts bin;
$\tilde{x}_6$	Arm 2 enters the parts bin;
$\tilde{x}_7$	Arm 2 exits the parts bin;
$\tilde{x}_8$	Arm 2 leaves the parts bin.

The generator was designed to recognize eight different plant events. This leads to nine different regions in the state space, and therefore, the DES plant model has nine states as shown in Fig. 26.

We want to control the robotic manufacturing system so that it never enters the critical section. Therefore, the control requirement for the DES plant is that it never enters state $\tilde{p}_8$.

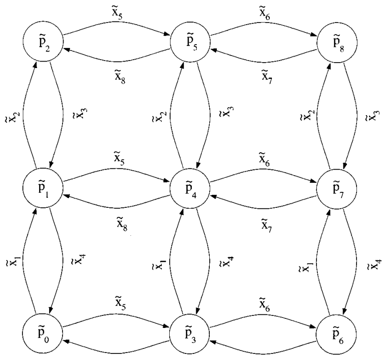


Fig. 26. DES plant model for the free-floating robotic system.

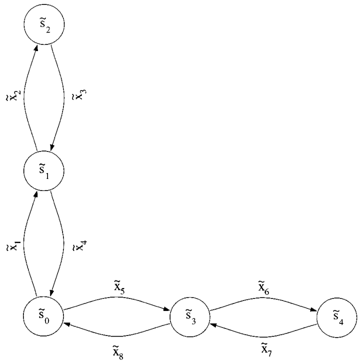


Fig. 27. DES controller for the free-floating robotic system.

The controller shown in Fig. 27 was obtained based on the supremal controllable sublanguage and does not allow the robotic arms to enter the critical section at the same time.

VI. Conclusions

In this paper, the supervisory control of hybrid systems has been introduced and discussed at length. Discrete abstractions that are represented by a DES plant model have been used to approximate the continuous plant. In general, the abstracting DES models are nondeterministic. Properties of the DES plant model to be a valid representation of the continuous plant have been presented. The emphasis has been put on the design of the interface between the continuous plant and the discrete event controller. A methodology to design the partition of the continuous state space based on the natural invariants of the plant has been briefly outlined. The robustness problem of the discrete transitions subject to small variations of the continuous system has also been addressed. Note that robustness to parameter variation is still an open issue in supervisory control of hybrid systems. An alternative methodology to the usual quantization technique of digital control based on the interface of hybrid control systems has been presented. The types of problems that have been addressed are those with control specifications that can be described by formal languages accepted by the DES plant model. The supervisory control problem for hybrid systems has been formulated, and algorithms for supervisory design based on the controllability of the specification language have been presented. Although the approach in this paper was based on a continuous-time model of the plant, similar results have been obtained using discrete-time systems [65], [59]. It should be noted that our coverage is primarily of a tutorial nature, and so many technical details have been just briefly outlined or simply omitted; the reader should consult the references for further details.

In this paper, we focused on the case when finite automata are used to describe both the plant and the controller. Hybrid control approaches based on Petri nets have been reported in the literature (see, e.g., the survey paper [8]). A similar approach to the one described in this paper using Petri nets, which may be computationally more efficient for large concurrent systems, has been reported in [23] and [31]. This approach addresses a particular class of supervisory control problems described by convex constraints on the marking of the Petri nets [43].

References

[1] R. Alur, C. Courcoubetis, N. Halbwachs, T. Henzinger, P.-H. Ho, X. Nicollin, A. Oliveiro, J. Sifakis, and S. Yovine, "The algorithmic analysis of hybrid systems," *Theoret. Comput. Sci.*, vol. 138, pp. 3–34, 1995.
[2] R. Alur, T. Henzinger, and E. Sontag, Eds., *Hybrid Systems III, Verification and Control*. Berlin, Germany: Springer-Verlag, 1996, vol. 1066, Lecture Notes in Computer Science.
[3] P. Antsaklis, "Defining intelligent control," *IEEE Contr. Syst.*, pp. 4–5, 58–66, June 1994.
[4] ——, "Intelligent control," in *Encyclopedia of Electrical and Electronics Engineering*. New York: Wiley, 1997.
[5] P. Antsaklis, W. Kohn, M. Lemmon, A. Nerode, and S. Sastry, Eds., *Hybrid Systems V*. Berlin, Germany: Springer-Verlag, 1999, vol. 1567, Lecture Notes in Computer Science.
[6] P. Antsaklis, W. Kohn, A. Nerode, and S. Sastry, Eds., *Hybrid Systems II*. Berlin, Germany: Springer-Verlag, 1995, vol. 999, Lecture Notes in Computer Science.
[7] ——, *Hybrid Systems IV*. Berlin, Germany: Springer-Verlag, 1997, vol. 1273, Lecture Notes in Computer Science.
[8] P. Antsaklis and X. Koutsoukos, "On hybrid control of complex systems: A survey," in *3rd Int. Conf. ADMP'98, Automation of Mixed Processes: Dynamic Hybrid Systems*, Reims, France, Mar. 1998, pp. 1–8.
[9] P. Antsaklis, X. Koutsoukos, and J. Zaytoon, "On hybrid control of complex systems: A survey," *Eur. J. Automat.*, vol. 32, no. 9–10, pp. 1023–1045, 1998.
[10] P. Antsaklis and M. Lemmon, "Introduction to the special issue on hybrid systems," *J. Discrete Event Dyn. Syst.*, vol. 8, p. 10, June 1998. (Special Issue on Hybrid Control Systems).
[11] P. Antsaklis and A. Nerode, "Hybrid control systems: An introductory discussion to the special issue," *IEEE Trans. Automat. Contr.*, vol. 43, pp. 457–460, Apr. 1998. (Special Issue on Hybrid Control Systems).
[12] P. Antsaklis and K. Passino, Eds., *An Introduction to Intelligent and Autonomous Control*. Norwell, MA: Kluwer, 1993.

References

[1] http://www.mathworks.com/help/stateflow/examples/modeling-a-bouncing-ball.html.

[2] http://www.mathworks.com/help/stateflow/ug/modeling-a-bouncing-ball-in-continuous-time.html.

[3] Claire J. Tomlin, Lecture Notes 1 in Hybrid Systems: Modeling, Analysis, and Control .

[4] K. J. Åström and K. Furuta, "Swinging up a pendulum by energy control," *Automatica*, vol. 36, no. 2, pp. 287–295, 2000.

[5] Claire J. Tomlin, Lecture Notes 6 in Hybrid Systems: Modeling, Analysis, and Control .

[6] Georgios Fainekos, ASU, CSE591: Theoretical aspects of Cyber-Physical Systems (Spring 2010) .

[7] T. A. Henzinger, *The theory of hybrid automata.* Springer, 2000.

[8] J. Lunze and F. Lamnabhi-Lagarrigue, *Handbook of hybrid systems control: theory, tools, applications.* Cambridge University Press, 2009.

[9] X. D. Koutsoukos, P. J. Antsaklis, J. A. Stiver, and M. D. Lemmon, "Supervisory control of hybrid systems," *Proceedings of the IEEE*, vol. 88, no. 7, pp. 1026–1049, 2000.

[10] Karl Henrik Johansson, John Lygeros, Shankar Sastry, Modeling of Hybrid Systems.