LMI Methods in Optimal and Robust Control

Matthew M. Peet
Arizona State University

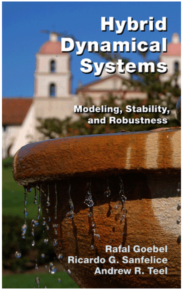
Lecture 19: Hybrid Systems

Hybrid Systems

Suggested Text 1: Switching in Systems and Controls
by Daniel Liberzon

Highly Recommend: One of the best texts in any field

Suggested Text 2:
Hybrid Dynamical Systems: Modeling, Stability, and Robustness
by R. Goebel; R. G. Sanfelice; A. R. Teel
Link: Chapter 1 Available Online Here



State-Dependent Switching

$$\dot{x}(t) = \begin{cases} f_i(x(t)) & x(t) \in X_i \end{cases}$$

Systems with Resets

$$\dot{x}(t) = \begin{cases} f(x(t)) & x(t) \notin G \end{cases}$$

and

$$\begin{cases} x_+ = g(x) & x \in G \end{cases}$$

Systems with Logical States

$$\dot{x}(t) = \begin{cases} f_i(x(t)) & \sigma(t) \in X_i \end{cases}$$

$$\dot{\sigma}(t) = \begin{cases} h(\sigma(t)) & x(t) \notin G \end{cases}$$

and

$$\begin{cases} \sigma_+ = g(\sigma) & x \in G \end{cases}$$

Discontinuous Control

Thermostat Control: The Hybrid Model

Control Logic:

```
> if u=1 and T>= 80 then
>    u=0
> elseif u=0 and T<= 70 then
>    u=1
> end
```

Temperature Dynamics:

$$\dot{T}(t) = c_w(T_e - T(t)) + c_q u(t)$$

- T_e is the external temperature.
- c_w is thermal resistance of the wall
- c_q is the capacity of the HVAC

Unicycle Dynamics:

$$\dot{x} = u_1 \cos\theta$$
$$\dot{y} = u_1 \cos\theta$$
$$\dot{\theta} = u_2$$

- x, y are the position.
- θ is the angle of the wheel.
- u_1 is the forward force.
- u_2 is the rotation rate.

Pose as

$$\begin{bmatrix}\dot{x}\\ \dot{y}\\ \dot{\theta}\end{bmatrix} = \begin{bmatrix}\cos\theta\\ \sin\theta\\ 0\end{bmatrix} u_1 + \begin{bmatrix}0\\ 0\\ 1\end{bmatrix} u_2 = \begin{bmatrix}\cos\theta & 0\\ \sin\theta & 0\\ 0 & 1\end{bmatrix} \begin{bmatrix}u_1\\ u_2\end{bmatrix}$$

Brockett's condition: if

$$\dot{x}(t) = G(x(t))u(t)$$

and rank$(G(0)) < n$ where $x \in \mathbb{R}^n$, then there is no asymptotically stabilizing continuous feedback control law

Discontinuous Control: The Brockett Integrator

Non-holonomic systems

Unicycle Dynamics:

$$\dot{x} = u_1 \cos\theta$$
$$\dot{y} = u_1 \cos\theta$$
$$\dot{\theta} = u_2$$

- x, y are the position.
- θ is the angle of the wheel.
- u_1 is the forward force.
- u_2 is the rotation rate.

Pose as
$$\begin{bmatrix}\dot{x}\\ \dot{y}\\ \dot{\theta}\end{bmatrix} = \begin{bmatrix}\cos\theta\\ \sin\theta\\ 0\end{bmatrix} u_1 + \begin{bmatrix}0\\ 0\\ 1\end{bmatrix} u_2 = \begin{bmatrix}\cos\theta & 0\\ \sin\theta & 0\\ 0 & 1\end{bmatrix}\begin{bmatrix}u_1\\ u_2\end{bmatrix}$$

Brockett's condition: if
$$\dot{x}(t) = G(x(t))u(t)$$
and rank$(G(0)) < n$ where $x \in \mathbb{R}^n$, then there is no asymptotically stabilizing continuous feedback control law

Spacecraft Attitude dynamics is another famous case:

- Three torques are required for existence of a continuous controller.
- Discontinuous control makes pointing problems hard.

Definition 1 (Hybrid System).

A hybrid system H is a tuple:

$$H = (Q, E, D, F, G, R)$$

where

- Q is a finite collection of discrete modes, states or indices.
- $E \subset Q \times Q$ is a collection of edges.
- $D = \{D_q\}_{q \in Q}$ is the collection of Domains associated with the discrete states, where for each $q \in Q$, $D_q \subseteq \mathbb{R}^n$.
- $F = \{f_q\}_{q \in Q}$ is the collection of vector fields associated with the discrete states, where for each $q \in Q$, $f_q : D_q \to \mathbb{R}^n$.
- $G = \{G_e\}_{e \in E}$ is a collection of guard sets, each associated with an edge. where for each $e = (q, q') \in E$, $G_e \subset D_q$
- $R = \{\phi_e\}_{e \in E}$ is a collection of Reset Maps, where for each $e = (q, q') \in E$, $\phi_e : G_e \to D_{q'}$.

What Are Hybrid Systems?

A Unified Definition of Hybrid Systems

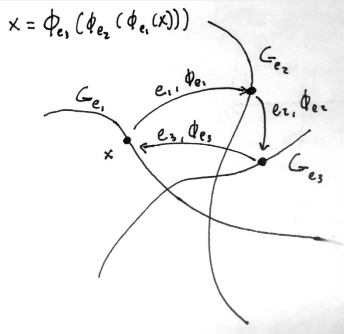
Definition 1 (Hybrid System).

A hybrid system H is a tuple:

$$H = (Q, E, D, F, G, R)$$

where

- Q is a finite collection of discrete modes, states or indices.
- $E \subset Q \times Q$ is a collection of edges.
- $D = \{D_q\}_{q \in Q}$ is the collection of Domains associated with the discrete states, where for each $q \in Q$, $D_q \subseteq \mathbb{R}^n$.
- $F = \{f_q\}_{q \in Q}$ is the collection of vector fields associated with the discrete states, where for each $q \in Q$, $f_q : D_q \to \mathbb{R}^n$.
- $G = \{G_e\}_{e \in E}$ is a collection of guard sets, each associated with an edge. where for each $e = (q, q') \in E$, $G_e \subset D_q$
- $R = \{\phi_e\}_{e \in E}$ is a collection of Reset Maps, where for each $e = (q, q') \in E$, $\phi_e : G_e \to D_{q'}$.



Note: Discrete-time systems are a bit tricky

- A hybrid system with no continuous evolution?
- Can we combine discrete and continuous dynamics?
- Where are the guard sets?

Thermostat Control: The Hybrid Model

Control Logic:

```
> if u=1 and T>= 80 then
>    u=0
> elseif u=0 and T<= 70 then
>    u=1
> end
```

(Thermostat Control) For heating, define the hybrid system H_T as:

$$H_T = (Q, E, D, F, G, R)$$

where

- $Q = \{1, 2\}$
- $E = \{e_1, e_2\}$, $e_1 = (1, 2)$,$e_2 = (2, 1)$
- $D := \{D_1, D_2\}$, $D_1 = [70, 80]$, $D_2 = [70, 80]$
- $G := \{G_1, G_2\}$, $G_{e_1} = \{T : T = 70\}$, $G_{e_2} = \{T : T = 80\}$
- $F = \{f_1, f_2\}$, $f_1 = c_w(T_e - T(t))$, $f_2 = c_w(T_e - T(t)) + c_q$.
- $R = \emptyset$ - No Reset Map.

Bouncing Ball: The Hybrid Model

$$\ddot{x}(t) = -g/m \qquad \begin{bmatrix}\dot{x}_1\\ \dot{x}_2\end{bmatrix} = \begin{bmatrix}0 & 1\\ 0 & 0\end{bmatrix}\begin{bmatrix}x_1\\ x_2\end{bmatrix} + \begin{bmatrix}0\\ -g/m\end{bmatrix}$$

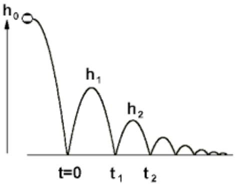
When the ball hit the floor, it bounces back up with coefficient of restitution c.

(Bouncing Ball) We define the hybrid system H_B as:

$$H_B = (Q, E, D, F, G, R)$$

where

- $Q = \{1\}$
- $E = \{e_1\}$, $e_1 = (1, 1)$
- $D := \{D_1\}$, $D_1 = [0, \infty]$
- $G := \{G_1\}$, $G_{e_1} = \{x : x_1 = 0, x_2 < 0\}$
- $F = \{f_1\}$, $f_1 = \begin{bmatrix}0 & 1\\ 0 & 0\end{bmatrix}\begin{bmatrix}x_1\\ x_2\end{bmatrix} + \begin{bmatrix}0\\ -g/m\end{bmatrix}$.
- $R = \{\phi_1\}$, $\phi_1\left(\begin{bmatrix}x_1\\ x_2\end{bmatrix}\right) = \begin{bmatrix}x_1\\ -cx_2\end{bmatrix}$.



State-Dependent Switching is typically defined by

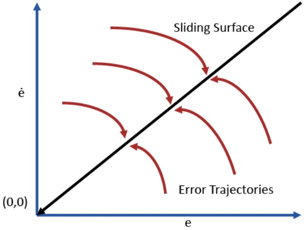
- A family of dynamical systems, one for each switching region
- A set of regions, defined by switching surfaces

$$\dot{x}(t) = \begin{cases} f_i(x(t)) & x(t) \in D_i \end{cases}$$

In this case, $H = (Q, \emptyset, D, F, \emptyset, \emptyset)$, $Q = \{i\}_{i=1}^{k}$, $D = \{D_i\}$, $F = \{f_i\}$.

Example:

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{cases} \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, & \text{if } x_1 > x_2 \\ \begin{bmatrix} -1 & 0 \\ 0 & -\lambda \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, & \text{otherwise.} \end{cases}$$



If $\lambda \in (-1, 1)$, the surface $x_1 = x_2$ is stable.

Note: State-Dependent Switching can also be defined by discrete-time dynamics

- But this is **Rare**.

State-Dependent Switching

General Form

State-Dependent Switching is typically defined by

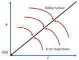
- A family of dynamical systems, one for each switching region
- A set of regions, defined by switching surfaces

$$\dot{x}(t) = \Big\{ f_i(x(t)) \quad x(t) \in D_i$$

In this case, $H = (Q, \emptyset, D, F, \emptyset, \emptyset)$, $Q = \{i\}_{i=1}^{k}$, $D = \{D_i\}$, $F = \{f_i\}$.

Example:

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{cases} \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, & \text{if } x_1 > x_2 \\ \begin{bmatrix} -1 & 0 \\ 0 & -\lambda \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, & \text{otherwise.} \end{cases}$$



If $\lambda \in (-1, 1)$, the surface $x_1 = x_2$ is stable.

Note: State-Dependent Switching can also be defined by discrete-time dynamics

- But this is **Rare**.

We probably should have defined E, G, and R.

- But this seems pedantic.

Several Operating Points:

Table 11.2 Parameter Values at the Seven Operating Points

Time (s)	t_1	t_2	t_3	t_4	t_5	t_6	t_7
$a_1(t)$	1.593	1.485	1.269	1.130	0.896	0.559	0.398
$a_1'(t)$	0.285	0.192	0.147	0.118	0.069	0.055	0.043
$a_2(t)$	260.559	266.415	196.737	137.385	129.201	66.338	51.003
$a_3(t)$	185.488	182.532	176.932	160.894	138.591	78.404	53.840
$a_4(t)$	1.506	1.295	1.169	1.130	1.061	0.599	0.421
$a_5(t)$	0.298	0.243	0.217	0.191	0.165	0.105	0.078
$b_1(t)$	1.655	1.502	1.269	1.130	0.896	0.559	0.398
$b_1'(t)$	0.295	0.195	0.147	0.118	0.069	0.055	0.043
$b_2(t)$	39.988	−24.627	−31.452	−41.425	−68.165	−21.448	−9.635
$b_3(t)$	159.974	170.532	182.030	184.093	154.608	89.853	59.587
$b_4(t)$	0.771	0.652	0.680	0.691	0.709	0.360	0.243
$b_5(t)$	0.254	0.191	0.188	0.182	0.162	0.102	0.072

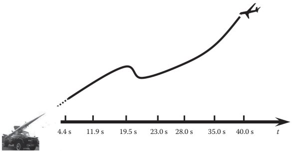
Dynamics:

$$\dot{x}(t) = Ax(t) + Bu(t)$$

$$u(t) = \begin{cases} K_1 x(t), & \text{if } |x(t)| \le 1 \\ K_2 x(t), & \text{if } |x(t)| \in [1,2] \\ K_3 x(t), & \text{otherwise.} \end{cases}$$

There can be a large array of gains.

In **Gain Scheduling**, the controller switches depending on operating point.



Often used to control nonlinear systems

- Each controller designed for linearized dynamics at a specific operating point.

State-Dependent Switching

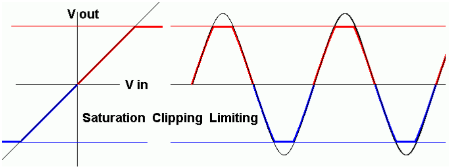
Input Saturation and Queueing

A common source of state-dependent Switching is *Input Saturation*

Input power is limited: $|u(t)| \le s$

$$\dot{x}(t) = Ax(t) + Bu(t)$$

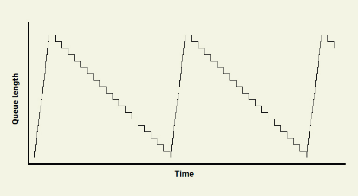
$$u(t) = \begin{cases} Kx(t) & |u(t)| \le s \\ \text{sign}(u(t))Ks & |u(t)| > s \end{cases}$$



Another source of switching in congestion control is due to *Queueing*

- Packets arrive at rate $u(t)$
- Packets are processed at constant rate c
- Router can't process packets if queue is empty!

$$\dot{x}(t) = \begin{cases} u(t) - c & x(t) \ge 0 \text{ OR } u(t) - c > 0 \\ 0 & \text{otherwise.} \end{cases}$$



State-Dependent Switching with Reset Maps

The General Form

Recall: $H = (Q, E, D, F, G, R)$. Now, we add in

Guard Sets: G_e

- A set of surfaces, typically the boundaries of D_i.
- Dynamics are continuous until we encounter a guard set

$$\dot{x}(t) = \begin{cases} f_i(x(t)) & \text{if } x(t) \in D_i \text{ and } x(t) \notin G_{\{i,j\}} \text{ for any } j \end{cases}$$

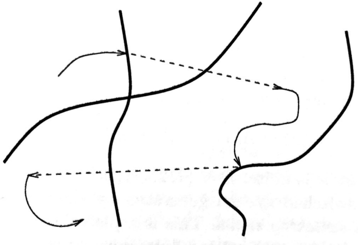
- For $e = \{i, j\}$, G_e are the points which transition the state from D_i to D_j

Reset Maps: ϕ_e

- If $x(t) \in D_i$ and $x(t) \in G_{i,j}$, we reset x to

$$x_+ = \phi_{\{i,j\}}(x)$$

- Thus $e = \{i, j\}$ implies $\phi_e(x) \in D_j$ for all $x \in D_i \cap G_e$



Recall the Bouncing Ball

What is a solution?

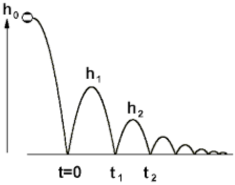
Dynamics are $\ddot{x} = -g$ until we hit the floor...

(Bouncing Ball) Define the hybrid system H_B as:

$$H_B = (Q, E, D, F, G, R)$$

where

- $Q = \{1\}$
- $E = \{(1, 1)\}$
- $D := \{x \in \mathbb{R}^2 : x_1 \geq 0\}$
- $G := \{x \in \mathbb{R}^2 : x_1 = 0,\ x_2 \leq 0\}$
- $F = \{\begin{bmatrix} x_2 \\ -g \end{bmatrix}\}$, i.e. $\dot{x}_1 = x_2$ and $\dot{x}_2 = -g$.
- $R = \phi(x) = [0, -cx_2]^T$. Here, $c < 1$ is the coefficient of restitution.



When will the ball stop bouncing???

Zeno Equilibria

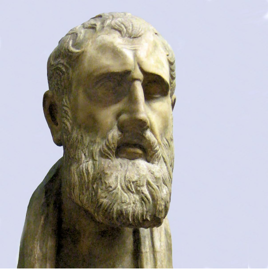
A **Zeno Equilibrium** is a point which is attractive, but is not an equilibrium ($f(x_e) \neq 0$).

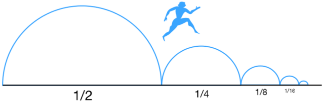
The Bouncing Ball vividly illustrates the concept of a Zeno Equilibrium.

- The floor is **NOT** an equilibrium! At $[x_1, x_2] = [0, 0]$

$$f(0) = \begin{bmatrix} 0 \\ -g \end{bmatrix}$$

- Yet clearly the floor is a stable point





Historical Note: Zeno of Elea (c. 490-430 BC) did not invent hybrid systems.

- Zeno's paradox rather illustrated the need for a concept of limit.
- Mostly irrelevant to Zeno equilibria

The concept also applies to switching systems without resets

- Sliding Mode control forces trajectories to a desired Manifold

Consider this simple example (Not Sliding Mode):

$$\ddot{x}(t) = \begin{cases} -c\dot{x} - u & x \geq 0 \ (D_1) \\ -c\dot{x} + u & x \leq 0 \ (D_2). \end{cases}$$

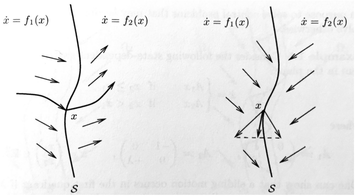


Figure: Illustration of Sliding Mode Control

$$f_1(x_1, x_2) = \begin{bmatrix} x_1 \\ -cx_2 - u \end{bmatrix}$$

$$f_2(x_1, x_2) = \begin{bmatrix} x_1 \\ -cx_2 + u \end{bmatrix}$$

The Origin is stable, but is not an equilibrium!

Define the Solution of a Hybrid System : An Execution

Definition 2 (Hybrid System Execution).

We say that the tuple

$$\chi = (I, T, p, C)$$

where

- $I \subseteq \mathbb{N}$ index the time intervals when the trajectory continuously evolves.
- $T = \{T_i\}_{i \in I}$ are the time intervals when the trajectory continuously evolves: $T_i = (\tau_i, \tau_{i+1}) \subset \mathbb{R}_+^n$ where $T_{i+1} = (\tau_{i+1}, \tau_{i+2})$.
- $p : I \to Q$ assigns each time interval to a discrete mode.
- $C = \{c_i\}_{i \in I}$ are the trajectories on each time interval $c_i \in \mathcal{C}[T_i]$.

is an *execution* of the hybrid system $H = F(Q, E, D, F, G, R)$ with initial condition (q_0, x_0) if

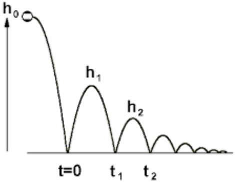
1. $c_1(0) = x_0$ and $p(1) = q_0$.
2. $\dot{c}_i(t) = f_{p(i)}(c_i(t))$ for $t \in T_i$ for every $i \in I$.
3. $c_i(t) \in D_{p(i)}$ for $t \in T_i$ for every $i \in I$.
4. $c_i(\tau_{i+1}) \in G_{(p(i),p(i+1))}$ for every $i \in I$. (End intervals on the Guard)
5. $c_{i+1}(T_{i+1}(1)) = \phi_{(p(i),p(i+1))}(c_i(T_i(2)))$ for every $i \in I$. (Start intervals with a reset)

Hybrid Execution: Example

Bouncing Ball

(Bouncing Ball) For an initial condition $x_0 = [0, v_0]$, the Hybrid Execution is

$$\chi_B = (I, T, p, C)$$



- $I = 1, \cdots, \infty$
- $T_i = [\tau_i, \tau_{i+1}]$ where $\tau_1 = 0$ and $\tau_{i+1} := \tau_i + \frac{2c^{i-1}v_0}{g}$
- $p_i = 1$
- $c_i(t) = c^{i-1}v_0(t - \tau_i) - \frac{1}{2}g(t - \tau_i)^2$

Zeno Execution: Formal Definition

Note that an execution does not require $\lim_{i\to\infty} \tau_i = \infty$, so the solution may not be defined for all time.

- An execution with infinite transitions in finite time is called Zeno.

Definition 3 (Zeno Execution).

We say an execution $\chi = (I, T, p, C)$ starting from (q_0, x_0) of a hybrid System $H = (Q, E, D, F, G, R)$ is Zeno if

1. $I = \mathbb{N}$
2. $\lim_{i\to\infty} \tau_i < \infty$

Question: is the bouncing ball a Zeno execution?

$$\tau_i = \sum_{i=1}^{i} \frac{2v_0}{g} c^{i-1}$$

Taking the limit:

$$\lim_{i\to\infty} \tau_i = \frac{2v_0}{g} c + \frac{2v_0}{g} \frac{1}{1-c} < \infty$$

So this is a zeno execution!

Zeno Equilibria: Formal Definition

Definition 4 (Zeno Equilibrium).

A set $z = \{z_q\}_{q \in Q}$ with $z_q \in D_q$ is a Zeno equilibrium of a Hybrid System $H = (Q, E, D, F, G, R)$ if it satisfies

1. For each edge $e = (q, q') \in E$, $z_q \in G_e$ and $\phi_e(z_q) = z_{q'}$.
2. $f_q(z_q) \neq 0$ for all $q \in Q$.

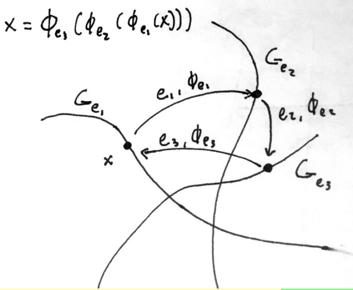
For any $z \in \{z_q\}_{q \in Q}$, where $\{z_q\}_{q \in Q}$ is a Zeno equilibrium of a cyclic hybrid system H_c,

$$\left(\phi_{i-1} \circ \cdots \circ \phi_0 \cdots \phi_i\right)(z) = z$$

For the **Bouncing Ball**,

$$z = \left\{ \begin{bmatrix} 0 \\ 0 \end{bmatrix} \right\}$$

is a Zeno Equilibrium.



Zeno Behaviour: Simulation

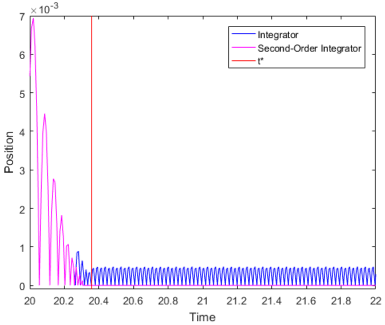
Zeno Executions are **Notoriously** hard to simulate accurately

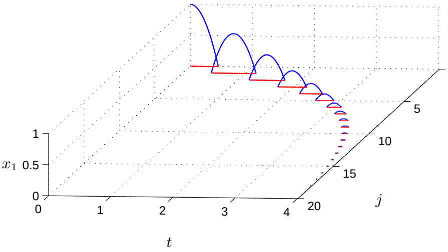
- Simulation relies on numerical integration
- But integration must stop when state encounters guard
- As intervals become smaller, this causes BIG problems

There are Specialized Software tools which handle this problem well.

- HyEQ is freely available and reliable
- Executions may still get stuck at Zeno points.

Link: HyEQ Hybrid System Simulator





Avoiding Zeno with Logical and Hysteresis Switching

Thermostat Control

A Thermostat uses **Memory** to avoid Zeno behaviour.

- The thermostat is *binary*.
  - It is either ON - $u = 1$
  - or OFF - $u = 0$
- Controls to set point, say $T = 75°$.
- But allows the temperature to vary in a Band $\pm 5°$.
  - Avoids *Chattering* associated with Zeno Executions

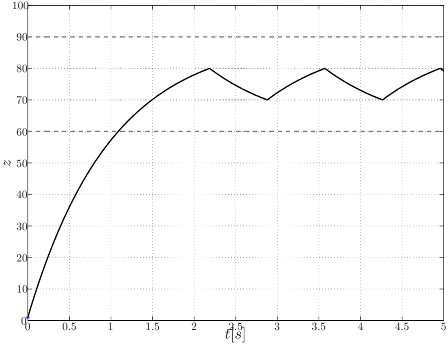
Control Logic:

```
> if u=1 and T>= 80 then
>    u=0
> elseif u=0 and T<= 70 then
>    u=1
> end
```

Temperature Dynamics:

$$\dot{T}(t) = c_w(T_e - T(t)) + c_q u(t)$$

- T_e is the external temperature.
- c_w is thermal resistance of the wall
- c_q is the capacity of the HVAC



Thermostat Control: The Hybrid Model

Control Logic:

```
> if u=1 and T>= 80 then
>    u=0
> elseif u=0 and T<= 70 then
>    u=1
> end
```

(Thermostat Control) For heating, define the hybrid system H_T as:

$$H_T = (Q, E, D, F, G, R)$$

where

- $Q = \{1, 2\}$
- $E = \{e_1, e_2\}$, $e_1 = (1, 2)$,$e_2 = (2, 1)$
- $D := \{D_1, D_2\}$, $D_1 = [70, 80]$, $D_2 = [70, 80]$
- $G := \{G_1, G_2\}$, $G_{e_1} = \{T : T = 70\}$, $G_{e_2} = \{T : T = 80\}$
- $F = \{f_1, f_2\}$, $f_1 = c_w(T_e - T(t))$, $f_2 = c_w(T_e - T(t)) + c_q$.
- $R = \emptyset$ - No Reset Map.

The Thermostat Model with heating AND cooling

(Thermostat Control) To include heating and cooling, redefine H_T as:

$$H_T = (Q, E, D, F, G, R)$$

where

- $Q = \{1, 2, 3\}$
- $E = \{e_1, e_2, e_3, e_4\}$, $e_1 = (1, 2)$, $e_2 = (2, 1)$, $e_3 = (1, 3)$, $e_4 = (3, 1)$,
- $D := \{D_1, D_2, D_3\}$, $D_1 = D_2 = D_3 = [70, 80]$.
- $G := \{G_1, G_2, G_3\}$, $G_1 = \{G_{e_1}, G_{e_3}\}$, $G_2 = \{G_{e_2}\}$, $G_3 = \{G_{e_4}\}$

$$G_{e_1} = \{T : T = 70, T > T_e\}, \quad G_{e_2} = \{T : T = 80\},$$

$$G_{e_3} = \{T : T = 80, T < T_e\}, \quad G_{e_4} = \{T : T = 70, \}$$

- $F = \{f_1, f_2\}$,

$$f_1(T) = c_w(T_e - T), \quad f_2(T) = c_w(T_e - T) + c_q, \quad f_3(T) = c_w(T_e - T) - c_c.$$

- $R = \emptyset$ - No Reset Map.

Question: How to verify executions don't leave the domain?