

MODEL PREDICTIVE CONTROL

HYBRID MODELS FOR MPC

Alberto Bemporad

`imt.lu/ab`

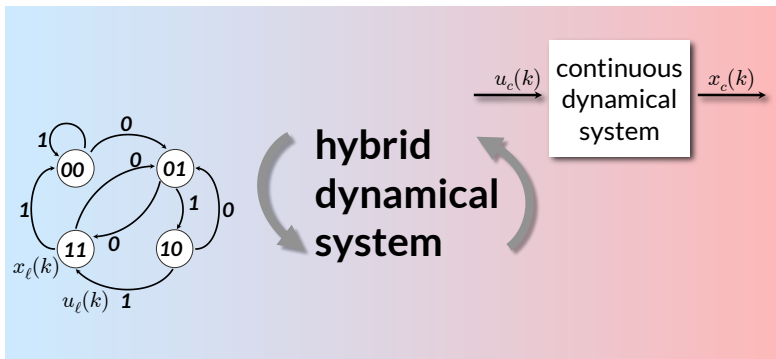
- ✓ Basic concepts of model predictive control (MPC) and linear MPC
- ✓ Linear time-varying and nonlinear MPC
- ✓ MPC computations: quadratic programming (QP), explicit MPC
 - Hybrid MPC
 - Stochastic MPC
 - Data-driven MPC

Course page:

http://cse.lab.imtlucca.it/~bemporad/mpc_course.html

HYBRID MODELS

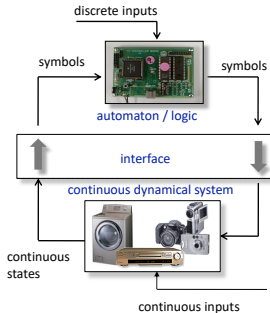
HYBRID DYNAMICAL SYSTEMS



- Variables are **binary-valued**
 $x_\ell \in \{0, 1\}^{n_\ell}$, $u_\ell \in \{0, 1\}^{m_\ell}$
- Dynamics = **finite state machine**
- **Logic constraints**

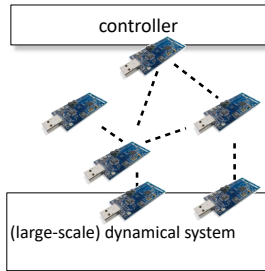
- Variables are **real-valued**
 $x_c \in \mathbb{R}^{n_c}$, $u_c \in \mathbb{R}^{m_c}$
- **Difference/differential equations**
- **Linear inequality** constraints

TECHNOLOGICAL PUSH FOR STUDYING HYBRID SYSTEMS



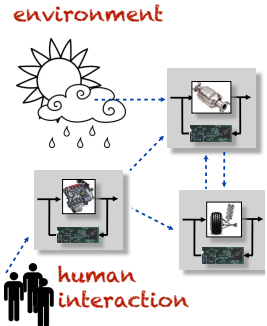
**embedded
systems**

>1995



**networked
control systems**

>2005



**cyber-physical
systems**

>2010

AN EXAMPLE OF "INTRINSICALLY HYBRID" SYSTEM

- Vehicle

continuous dynamical variables

(speed, torque, ...)

continuous commands

(brake & gas pedal)

+

discrete command

(R,N,1,2,3,4,5)



KEY REQUIREMENTS FOR HYBRID MODELS

- **Descriptive** enough to capture the behavior of the system
 - **continuous** dynamics (physical systems)
 - **logic** components (switches, automata)
 - **interconnection** between logic and dynamics
- **Simple** enough for solving analysis and synthesis problems

$$\begin{cases} x' &= Ax + Bu \\ y &= Cx + Du \end{cases}$$

←→
linear hybrid systems

$$\begin{cases} x' &= f(x, u, t) \\ y &= g(x, u, t) \end{cases}$$

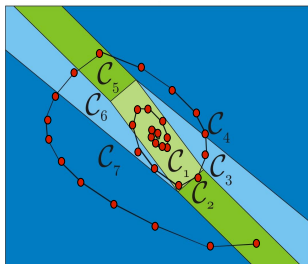
“Perfection is achieved not when there is nothing more to add,
but when there is nothing left to take away.”



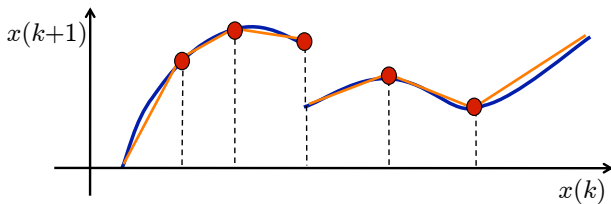
A. de Saint-Exupéry
(1900–1944)

PIECEWISE AFFINE SYSTEMS

$$\begin{aligned}x(k+1) &= A_{i(k)}x(k) + B_{i(k)}u(k) + f_{i(k)} \\y(k) &= C_{i(k)}x(k) + D_{i(k)}u(k) + g_{i(k)} \\i(k) \quad \text{s.t.} \quad &H_{i(k)}x(k) + J_{i(k)}u(k) \leq K_{i(k)}\end{aligned}$$

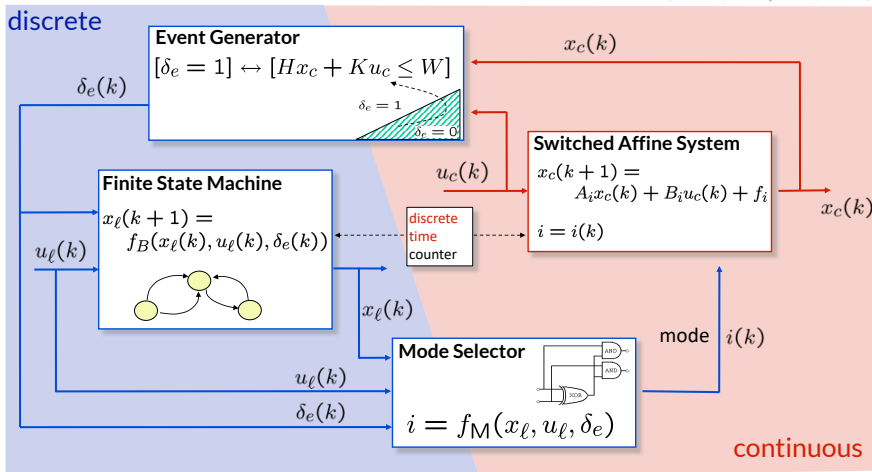


- PWA systems can approximate nonlinear dynamics arbitrarily well (even discontinuous ones)



DISCRETE HYBRID AUTOMATON (DHA)

(Torrì, Bemporad, 2004)

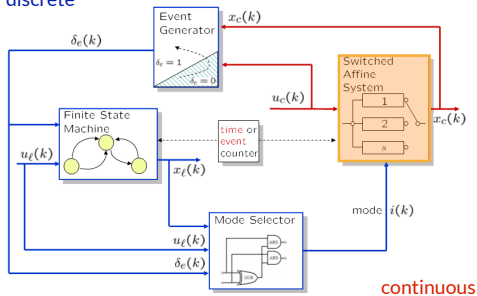


$x_\ell \in \{0, 1\}^{n_\ell}$ = **binary state**
 $u_\ell \in \{0, 1\}^{m_\ell}$ = **binary input**
 $\delta_e \in \{0, 1\}^{n_e}$ = **event variable**

$x_c \in \mathbb{R}^{n_c}$ = **real-valued state**
 $u_c \in \mathbb{R}^{m_c}$ = **real-valued input**
 $i \in \{1, \dots, s\}$ = **current mode**

SWITCHED AFFINE SYSTEM

discrete



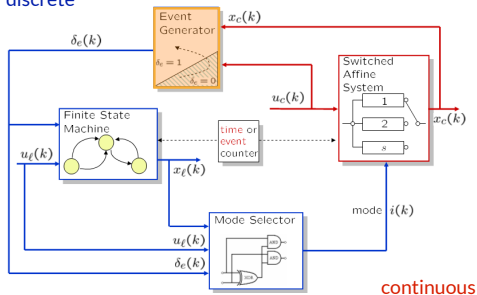
- The **affine dynamics** depend on the current mode $i(k)$:

$$x_c(k+1) = A_{i(k)}x_c(k) + B_{i(k)}u_c(k) + f_{i(k)}$$

$$x_c \in \mathbb{R}^{n_c}, u_c \in \mathbb{R}^{m_c}$$

EVENT GENERATOR

discrete



continuous

- Event variables** are generated by **linear threshold conditions** over continuous states, continuous inputs, and time:

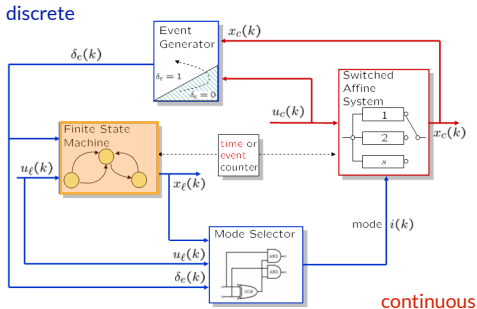
$$[\delta_e^i(k) = 1] \leftrightarrow [H^i x_c(k) + K^i u_c(k) \leq W^i]$$

$$x_c \in \mathbb{R}^{n_c}, \quad u_c \in \mathbb{R}^{m_c}$$

$$\delta_e \in \{0, 1\}^{n_e}$$

- Example:** $[\delta_e(k) = 1] \leftrightarrow [x_c(k) \geq 0]$

FINITE STATE MACHINE



- The binary state of the **finite state machine** evolves according to a Boolean state update function $f_B : \{0, 1\}^{n_\ell + m_\ell + n_e} \rightarrow \{0, 1\}^{n_\ell}$:

$$x_\ell(k+1) = f_B(x_\ell(k), u_\ell(k), \delta_e(k))$$

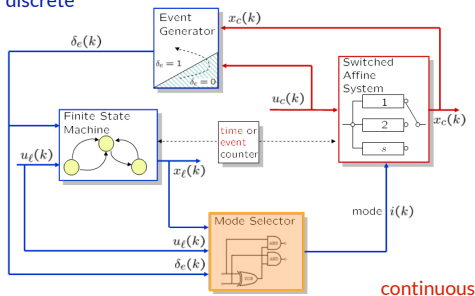
$$x_\ell \in \{0, 1\}^{n_\ell}, \quad u_\ell \in \{0, 1\}^{m_\ell}$$

$$\delta_e \in \{0, 1\}^{n_e}$$

- Example: $x_\ell(k+1) = \neg \delta_e(k) \vee (x_\ell(k) \wedge u_\ell(k))$

MODE SELECTOR

discrete



The mode selector can be seen as the output function of the discrete dynamics

- The active **mode** $i(k)$ is selected by a Boolean function of the current binary states, binary inputs, and event variables:

$$i(k) = f_M(x_\ell(k), u_\ell(k), \delta_e(k))$$

$$x_\ell \in \{0, 1\}^{n_\ell}, \quad u_\ell \in \{0, 1\}^{m_\ell}$$

$$\delta_e \in \{0, 1\}^{n_e}$$

- Example:

$$i(k) = \begin{bmatrix} \neg u_\ell(k) \vee x_\ell(k) \\ u_\ell(k) \wedge x_\ell(k) \end{bmatrix} \rightarrow \begin{array}{c|cc} u_\ell/x_\ell & 0 & 1 \\ \hline 0 & i = \begin{bmatrix} 1 \\ 0 \end{bmatrix} & i = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \\ 1 & i = \begin{bmatrix} 0 \\ 0 \end{bmatrix} & i = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \end{array}$$

the system has 3 modes

CONVERSION OF LOGIC FORMULAS TO LINEAR INEQUALITIES

(Glover, 1975) (Williams, 1977) (Hooker, 2000)

- Key observation: $X_1 \vee X_2 = \text{true}$ $\longrightarrow \delta_1 + \delta_2 \geq 1, \delta_1, \delta_2 \in \{0, 1\}$
- We want to impose the Boolean statement

$$F(X_1, \dots, X_n) = \text{true}$$

- Convert the formula to **Conjunctive Normal Form (CNF)**

$$\bigwedge_{j=1}^m \left(\bigvee_{i \in P_j} X_i \bigvee_{i \in N_j} \bar{X}_i \right) = \text{true}, \quad P_j \cup N_j \subseteq \{1, \dots, n\}$$

- Transform the CNF into the equivalent linear inequalities

$$\left\{ \begin{array}{ccc} \sum_{i \in P_1} \delta_i + \sum_{i \in N_1} (1 - \delta_i) & \geq & 1 \\ & \vdots & \\ \sum_{i \in P_m} \delta_i + \sum_{i \in N_m} (1 - \delta_i) & \geq & 1 \end{array} \right. \longrightarrow A\delta \leq b, \delta \in \{0, 1\}^n$$

polyhedron

Any logic proposition can be translated into integer linear inequalities

LOGIC \rightarrow INEQUALITIES: SYMBOLIC APPROACH

- Example:

$$F(X_1, X_2, X_3) = [X_3 \leftrightarrow X_1 \wedge X_2]$$

- Convert Conjunctive Normal Form (CNF):

(see e.g. <http://formal.cs.utah.edu:8080/pbl/PBL.php> or just google "CNF + converter" ...)

$$(X_3 \vee \neg X_1 \vee \neg X_2) \wedge (X_1 \vee \neg X_3) \wedge (X_2 \vee \neg X_3)$$

- Transform into inequalities:


$$\begin{cases} \delta_3 + (1 - \delta_1) + (1 - \delta_2) & \geq & 1 \\ \delta_1 + (1 - \delta_3) & \geq & 1 \\ \delta_2 + (1 - \delta_3) & \geq & 1 \end{cases}$$

LOGIC \rightarrow INEQUALITIES: GEOMETRIC APPROACH

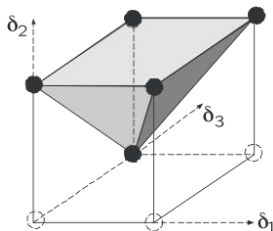
- Consider the Boolean statement $F(X_1, \dots, X_n) = \text{true}$ and collect the rows of the **truth table** $T(F)$ of F

The **convex hull** $P = \{\delta \in \mathbb{R}^n : A\delta \leq b\}$ of the **rows in** $T(F)$ is the smallest polytope equivalent to the Boolean statement F

	X_1	X_2	...	X_n
$T(F) :$	0	0	...	1
	0	1	...	1
	\vdots	\vdots	\vdots	\vdots
	1	1	...	0



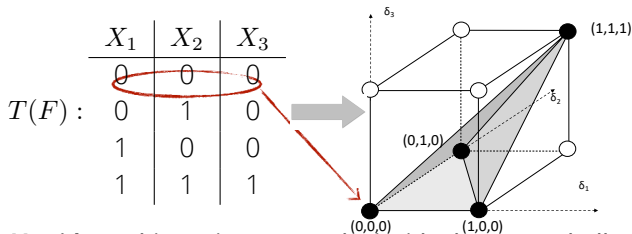
(Mignone, Bemporad, Morari, 1999)



- Convex hull packages: `cdd`, `lrs`, `qhull`, `chD`, `Hull`, `Porto`
CDDMEX package by K. Fukuda included in the Hybrid Toolbox

LOGIC \rightarrow INEQUALITIES: GEOMETRIC APPROACH

- Example: $F(X_1, X_2, X_3) = [X_3 \leftrightarrow X_1 \wedge X_2]$ (logic **and**)



- Key idea:** white points cannot be inside the convex hull of black points

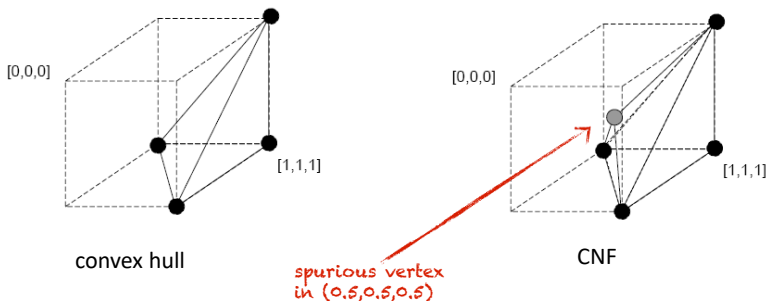
$$\text{conv} \left(\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \right) = \left\{ \delta \in \mathbb{R}^3 : \begin{array}{rcl} -\delta_1 + \delta_3 & \leq & 0 \\ -\delta_2 + \delta_3 & \leq & 0 \\ \delta_1 + \delta_2 - \delta_3 & \leq & 1 \end{array} \right\}$$

```
>> V=struct('V',[0 0 0;0 1 0;1 0 0;1 1 1]);
>> H=cddmex('hull',V);A=H.A,b=H.B
```

GEOMETRIC VS SYMBOLIC APPROACH

- The polyhedron obtained via convex hull is the smallest one
- The one obtained via CNF may be larger. Example:

$$(X_1 \vee X_2) \wedge (X_1 \vee X_3) \wedge (X_2 \vee X_3) = \text{true}$$



- **Note:** no other example with 3 vars but
 $(X_1 \vee X_2) \wedge (X_1 \vee X_3) \wedge (X_2 \vee X_3) \wedge (\neg X_1 \vee \neg X_2 \vee \neg X_3) = \text{true}$

BIG-M TECHNIQUE (IFF)

- Consider the **if-and-only-if** condition

$$[\delta = 1] \leftrightarrow [a'x_c - b \leq 0]$$

$$\begin{aligned} x_c &\in \mathcal{X} \\ \delta &\in \{0, 1\} \end{aligned}$$

- Assume $\mathcal{X} \subset \mathbb{R}^{n_c}$ bounded. Let M and m such that $\forall x_c \in \mathcal{X}$

$$M > a'x_c - b$$

$$m < a'x_c - b$$

- The if-and-only-if condition is equivalent to

$$\begin{cases} a'x_c - b \leq M(1 - \delta) \\ a'x_c - b > m\delta \end{cases}$$

- We can replace the second constraint with $a'x_c - b \geq \epsilon + (m - \epsilon)\delta$ to avoid strict inequalities, where $\epsilon > 0$ is a small number (e.g., the machine precision)

- If \mathcal{X} is a polyhedron, we can use linear programming

$$m < \min_{x_c \in \mathcal{X}} \{a'x_c\} - b$$

$$M > -\min_{x_c \in \mathcal{X}} \{-a'x_c\} - b$$

- If \mathcal{X} is a box, $\mathcal{X} = [\ell, u]$, we can use the following simpler method

(Lee, Kouvaritakis, 2000)

$$a_+ = \max\{a, 0\}$$

$$a_- = a_+ - a = \max\{-a, 0\}$$

$$m < a'_+ \ell - a'_- u - b$$

$$M > a'_+ u - a'_- \ell - b$$

BIG-M TECHNIQUE (IF-THEN-ELSE)

- Consider the **if-then-else** condition

$$z = \begin{cases} a'_1 x_c - b_1 & \text{if } \delta = 1 \\ a'_2 x_c - b_2 & \text{otherwise} \end{cases}$$

$$\begin{aligned} x_c &\in \mathcal{X} \\ \delta &\in \{0, 1\} \\ z &\in \mathbb{R} \end{aligned}$$

- Assume $\mathcal{X} \subset \mathbb{R}^{n_c}$ bounded. Let M_1, M_2 and m_1, m_2 such that $\forall x_c \in \mathcal{X}$

$$\begin{aligned} M_1 &> a'_1 x_c - b_1 > m_1 \\ M_2 &> a'_2 x_c - b_2 > m_2 \end{aligned}$$

- The if-then-else condition is equivalent to

$$\begin{cases} (m_1 - M_2)(1 - \delta) + z \leq a'_1 x_c - b_1 \\ (m_2 - M_1)(1 - \delta) - z \leq -(a'_1 x_c - b_1) \\ (m_2 - M_1)\delta + z \leq a'_2 x_c - b_2 \\ (m_1 - M_2)\delta - z \leq -(a'_2 x_c - b_2) \end{cases}$$

SWITCHED AFFINE SYSTEM

- The state-update equation of a SAS can be rewritten as

$$x_c(k+1) = \sum_{i=1}^s z_i(k) \quad z_i(k) \in \mathbb{R}^{n_c}$$

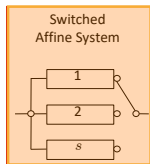
with

$$\begin{aligned} z_1(k) &= \begin{cases} A_1 x_c(k) + B_1 u_c(k) + f_1 & \text{if } \delta_1(k) = 1 \\ 0 & \text{otherwise} \end{cases} \\ \vdots \\ z_s(k) &= \begin{cases} A_s x_c(k) + B_s u_c(k) + f_s & \text{if } \delta_s(k) = 1 \\ 0 & \text{otherwise} \end{cases} \end{aligned}$$

and with $\delta_i(k) \in \{0, 1\}$ subject to the **exclusive or** condition

$$\sum_{i=1}^s \delta_i(k) = 1 \text{ or equivalently } \begin{cases} \sum_{i=1}^s \delta_i(k) & \geq 1 \\ \sum_{i=1}^s \delta_i(k) & \leq 1 \end{cases}$$

- Output eqs $y_c(k) = C_i x_c(k) + D_i u_c(k) + g_i$ admit similar transformation



TRANSFORMATION OF A DHA INTO LINEAR (INE)EQUALITIES

$$X_1 \vee X_2 = \text{TRUE}$$

$$\longrightarrow \delta_1 + \delta_2 \geq 1, \quad \delta_1, \delta_2 \in \{0, 1\}$$

Any logic statement

$$f(X) = \text{TRUE}$$

$$\bigwedge_{j=1}^m \left(\bigvee_{i \in P_j} X_i \vee \bigvee_{i \in N_j} \neg X_i \right) \quad (\text{CNF})$$

$N_j, P_j \subseteq \{1, \dots, n\}$

$$\begin{cases} 1 \leq \sum_{i \in P_1} \delta_i + \sum_{i \in N_1} (1 - \delta_i) \\ \vdots \\ 1 \leq \sum_{i \in P_m} \delta_i + \sum_{i \in N_m} (1 - \delta_i) \end{cases}$$

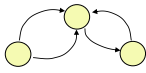
$$[\delta_e^i(k) = 1] \leftrightarrow [H^i x_c(k) \leq W^i]$$

$$\begin{cases} H^i x_c(k) - W^i \leq M^i (1 - \delta_e^i(k)) \\ H^i x_c(k) - W^i > m^i \delta_e^i(k) \end{cases}$$

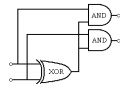
$$\begin{aligned} \text{IF } [\delta = 1] \text{ THEN } z &= a_1^T x + b_1^T u + f_1 \\ \text{ELSE } z &= a_2^T x + b_2^T u + f_2 \end{aligned}$$

$$\begin{cases} (m_1 - M_2)(1 - \delta) + z \leq a_1 x + b_1 u + f_1 \\ (m_2 - M_1)(1 - \delta) - z \leq -a_1 x - b_1 u - f_1 \\ (m_2 - M_1)\delta + z \leq a_2 x + b_2 u + f_2 \\ (m_1 - M_2)\delta - z \leq -a_2 x - b_2 u - f_2 \end{cases}$$

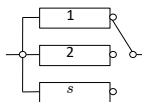
Finite State Machine



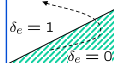
Mode Selector



Switched Affine System



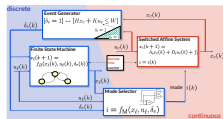
Event Generator



MIXED LOGICAL DYNAMICAL (MLD) SYSTEMS

(Bemporad, Morari, 1999)

- By converting logic relations into mixed-integer linear inequalities a DHA can be rewritten as the **Mixed Logical Dynamical (MLD)** system



$$\begin{cases} x(k+1) &= Ax(k) + B_1 u(k) + B_2 \delta(k) + B_3 z(k) + B_5 \\ y(k) &= Cx(k) + D_1 u(k) + D_2 \delta(k) + D_3 z(k) + D_5 \\ E_2 \delta(k) + E_3 z(k) &\leq E_4 x(k) + E_1 u(k) + E_5 \end{cases}$$

$$x \in \mathbb{R}^{n_c} \times \{0, 1\}^{n_b}, u \in \mathbb{R}^{m_c} \times \{0, 1\}^{m_b}$$
$$y \in \mathbb{R}^{p_c} \times \{0, 1\}^{p_b}, \delta \in \{0, 1\}^{r_b}, z \in \mathbb{R}^{r_c}$$

- The translation from DHA to MLD can be automatized, see e.g. the language **HYSDEL** (HYbrid Systems DEscription Language) (Torrìsi, Bemporad, 2004)
- MLD models allow solving MPC, verification, state estimation, and fault detection problems via **mixed-integer programming**

A SIMPLE EXAMPLE OF MLD SYSTEM

- PWA system¹:
$$x(k+1) = \begin{cases} 0.8x(k) + u(k) & \text{if } x(k) \geq 0 \\ -0.8x(k) + u(k) & \text{if } x(k) < 0 \end{cases}$$
- Introduce event variable $[\delta(k) = 1] \leftrightarrow [x(k) \geq 0]$ and use big-M technique:

$$\begin{aligned} &\longrightarrow \begin{aligned} x(k) &\geq m(1 - \delta(k)) \\ x(k) &\leq -\epsilon + (M + \epsilon)\delta(k) \end{aligned} \quad \begin{aligned} M &= -m = 10 \\ \epsilon &> 0 \text{ "small"} \end{aligned} \end{aligned}$$

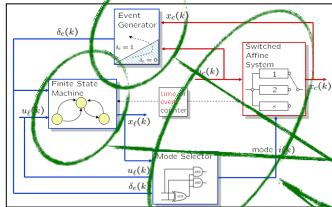
- Since $x(k+1) = 1.6\delta(k)x(k) - 0.8x(k) + u(k)$, introduce the aux variable

$$\begin{aligned} z(k) &= \delta(k)x(k) \longrightarrow \begin{aligned} z(k) &\leq M\delta(k) \\ z(k) &\geq m\delta(k) \\ z(k) &\leq x(k) - m(1 - \delta(k)) \\ z(k) &\geq x(k) - M(1 - \delta(k)) \end{aligned} \quad \boxed{\delta(k) \in \{0, 1\}} \end{aligned}$$

- Linear state update:
$$x(k+1) = -0.8x(k) + 1.6z(k) + u(k)$$

¹This is the nonlinear system $x(k+1) = 0.8|x(k)| + u(k)$

DHA AND HYSDEL MODELS



Additional relations
constraining system's
variables

```

SYSTEM name {
  INTERFACE {
    STATE {
      REAL xc [xmin,xmax];
      BOOL xl; }
    INPUT {
      REAL uc [umin,umax];
      BOOL ul; }
    PARAMETER {
      REAL param1 = 1;}
  } /* end of interface */

  IMPLEMENTATION {
    AUX { BOOL d;
          REAL z; }

    AUTOMATA { xl = xl & ~ul; }

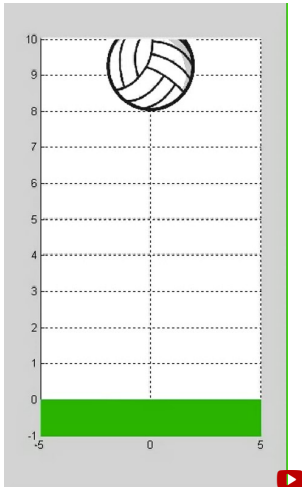
    AD { d = xc - 1 <= 0; }

    DA { z = { IF d THEN 2*xc ELSE -xc }; }

    CONTINUOUS {
      xc = z; }

    MUST {
      xc + uc <= 2;
      ~(xl & ul); }
  } /* end implementation */
} /* end system */
    
```

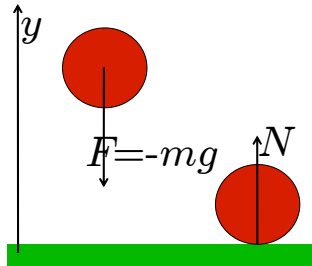
BOUNCING BALL EXAMPLE



$$\ddot{y} = -g$$

$$y \leq 0 \Rightarrow \dot{y}(t^+) = -(1 - \alpha)\dot{y}(t^-)$$

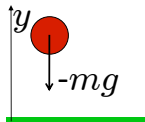
$$\alpha \in [0, 1]$$



How to model the bouncing ball as a discrete-time hybrid system ?

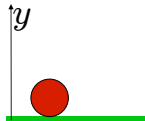
BOUNCING BALL — TIME DISCRETIZATION

- Case $y(k) > 0$ (ball falling):
$$v(k) \approx \frac{y(k) - y(k-1)}{T_s}$$
$$-g \approx \frac{v(k) - v(k-1)}{T_s}$$



$$\rightarrow \begin{cases} v(k+1) &= v(k) - T_s g \\ y(k+1) &= y(k) + T_s v(k+1) \\ &= y(k) + T_s v(k) - T_s^2 g \end{cases}$$

- Case $y(k) \leq 0$ (ground level):
$$v(k+1) = -(1 - \alpha)v(k)$$
$$y(k+1) = y(k) = y(k) - T_s v(k)$$



$$\rightarrow \begin{cases} v(k+1) &= -(1 - \alpha)v(k) \\ y(k+1) &= y(k) - T_s v(k) \end{cases}$$

- We need a binary variable $[\delta(k) = 1] \leftrightarrow [y(k) \leq 0]$

BOUNCING BALL - HYSDEL MODEL

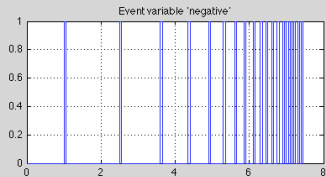
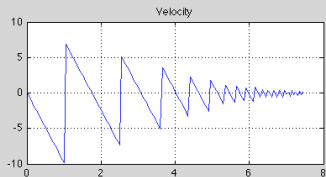
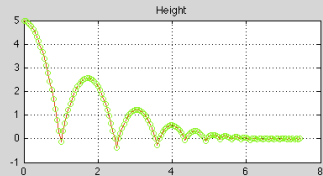
```
SYSTEM bouncing_ball {  
  INTERFACE {  
    /* Description of variables and constants */  
    STATE { REAL height [-10,10];  
            REAL velocity [-100,100];  }  
  
    PARAMETER {  
      REAL g;  
      REAL alpha; /* 0=elastic, 1=completely anelastic */  
      REAL Ts; }  
  }  
  IMPLEMENTATION {  
    AUX {  BOOL negative;  
           REAL hnext;  
           REAL vnext;  }  
  
    AD {    negative = height <= 0; }  
  
    DA {    hnext = { IF negative THEN height-Ts*velocity  
                     ELSE height+Ts*velocity-Ts*Ts*g};  
            vnext = { IF negative THEN -(1-alpha)*velocity  
                     ELSE velocity-Ts*g};  }  
  
    CONTINUOUS {  
      height  = hnext;  
      velocity = vnext;}  
  }  
}
```

go to `demo demos/hybrid/bball.m`

BOUNCING BALL - SIMULATION

```
>> Ts=0.05;  
>> g=9.8;  
>> alpha=0.3;  
  
>> S=mld('bouncing_ball',Ts);  
  
>> N=150;  
>> U=zeros(N,0);  
>> x0=[5 0]';  
  
>> [X,T,D]=sim(S,x0,U);
```

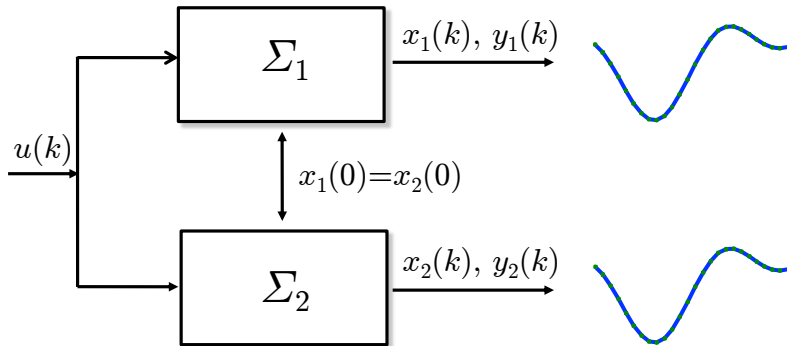
- **Note:** no **Zeno effect** in discrete time !



EQUIVALENCE OF HYBRID MODELS

EQUIVALENCE OF HYBRID MODELS

- Two hybrid models Σ_1, Σ_2 are **equivalent** if for all initial states $x_1(0) = x_2(0)$ and input excitations $u_1(k) \equiv u_2(k)$, the corresponding trajectories $x_1(k) \equiv x_2(k)$ and $y_1(k) \equiv y_2(k), \forall k = 0, 1, \dots$



EQUIVALENCE OF HYBRID MODELS

- **MLD** and **PWA** systems are equivalent (Bemporad, Ferrari-Trecate, Morari, 2000)

Proof: For a given combination (x_ℓ, u_ℓ, δ) of an MLD model, the state and output equation are linear and valid in a polyhedron.

Conversely, a PWA system can be modeled as MLD system (see next slide)

- Efficient **conversion algorithms** from MLD to PWA form exist
(Bemporad, 2004) (Geyer, Torrisi, Morari, 2003)
- Further equivalences exist with other classes of hybrid dynamical systems, such as **Linear Complementarity (LC)** systems (Heemels, De Schutter, Bemporad, 2001)

MODELING A PWA SYSTEM IN MLD FORM

- PWA system with bounded states and inputs and s regions

$$\begin{aligned}x(k+1) &= A_{i(k)}x(k) + B_{i(k)}u(k) + f_{i(k)} \\y(k) &= C_{i(k)}x(k) + D_{i(k)}u(k) + g_{i(k)} \\i(k) &= \text{such that } \begin{bmatrix} x(k) \\ u(k) \end{bmatrix} \in \mathcal{C}_{i(k)}\end{aligned}$$

with $\mathcal{C}_i = \{ \begin{bmatrix} x \\ u \end{bmatrix} : H_i x + J_i u \leq K_i \}$, and $\overset{\circ}{\mathcal{C}}_i \cap \overset{\circ}{\mathcal{C}}_j = \emptyset, \forall i \neq j, i, j = 1, \dots, s$
($\{\mathcal{C}_i\}$ is a **polyhedral partition** of the set $\mathcal{C} \triangleq \cup_{i=1}^s \mathcal{C}_i$)

- Introduce s binary variables $\delta_i, i = 1, \dots, s$ and the logic constraints

$$\begin{aligned}[\delta_i = 1] &\rightarrow [H_i x + J_i u \leq K_i] \\ \bigoplus_{i=1}^s [\delta_i = 1] &= \text{true} \end{aligned} \quad \longrightarrow \quad \begin{aligned} H_i x + J_i u &\leq K_i + M_i(1 - \delta_i) \\ \sum_{i=1}^s \delta_i &= 1 \end{aligned}$$

were the vector M_i of upper-bounds can be computed, e.g., via LP

MODELING A PWA SYSTEM IN MLD FORM

- Introduce auxiliary real vectors z_i, w_i defined by if-then-else rules

$$z_i = \begin{cases} A_i x + B_i u + f_i & \text{if } \delta_i = 1 \\ 0 & \text{otherwise} \end{cases} \quad w_i = \begin{cases} C_i x + D_i u + g_i & \text{if } \delta_i = 1 \\ 0 & \text{otherwise} \end{cases}$$

and convert the relations above into mixed-integer inequalities

- Finally, write the state update and output equations

$$\begin{cases} x(k+1) &= \sum_{i=1}^s z_i(k) \\ y(k) &= \sum_{i=1}^s w_i(k) \end{cases}$$

- A PWA system with bounded states and inputs is equivalent to the **disjunction**

$$\bigvee_{i=1}^s \left[\begin{array}{l} H_i x(k) + J_i u(k) \leq K_i \\ x(k+1) = A_i x(k) + B_i u(k) + f_i \end{array} \right] \quad \begin{array}{l} x_{lb} \leq x(k) \leq x_{ub} \\ u_{lb} \leq u(k) \leq u_{ub} \end{array}$$

- Introduce s binary variables $\delta_1(k), \dots, \delta_s(k)$ subject to $\sum_{i=1}^s \delta_i(k) = 1$
- Introduce the **convex hull relaxation** of the disjunction

$$\begin{aligned} x(k) &= \sum_{i=1}^s v_i(k), & x_{lb}\delta_i(k) &\leq v_i(k) \leq x_{ub}\delta_i(k) \\ u(k) &= \sum_{i=1}^s w_i(k), & u_{lb}\delta_i(k) &\leq w_i(k) \leq u_{ub}\delta_i(k) \end{aligned}$$

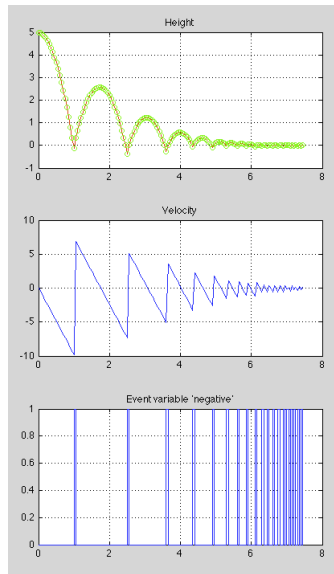
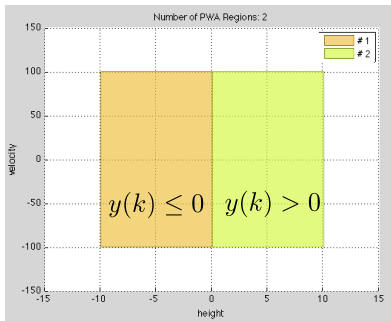
and impose

$$x(k+1) = \sum_{i=1}^s A_i v_i(k) + B_i w_i(k) + f_i \delta_i(k), \quad H_i v_i(k) + J_i w_i(k) \leq K_i \delta_i(k)$$

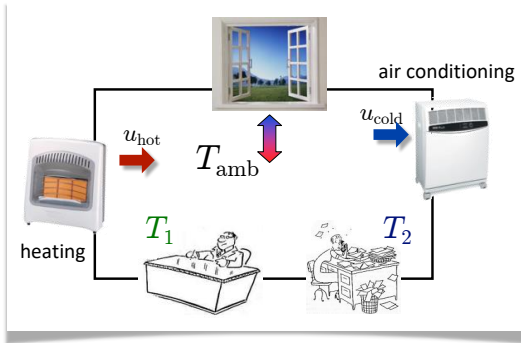
BOUNCING BALL - PWA EQUIVALENT

```
>> P=pwa(S);  
>> plot(P)  
  
>> [X,T,I]=sim(P,x0,U);
```

(Bemporad, 2004)



EXAMPLE: ROOM TEMPERATURE CONTROL



discrete dynamics

- #1 = cold \rightarrow heater = on
- #2 = cold \rightarrow heater = on **unless** #1 hot
- A/C activation has similar rules

continuous dynamics

$$\frac{dT_i}{dt} = -\alpha_i(T_i - T_{\text{amb}}) + k_i(u_{\text{hot}} - u_{\text{cold}})$$

$$i = 1, 2$$

go to `demo demos/hybrid/heatcool.m`

EXAMPLE: ROOM TEMPERATURE CONTROL

```
SYSTEM heatcool {  
  
  INTERFACE {  
    STATE { REAL T1 [-10,50];  
            REAL T2 [-10,50];  
          }  
    INPUT { REAL Tamb [-10,50];  
          }  
    PARAMETER {  
      REAL Ts, alpha1, alpha2, k1, k2;  
      REAL Thot1, Tcold1, Thot2, Tcold2, Uc, Uh;  
    }  
  }  
  IMPLEMENTATION {  
    AUX { REAL uhot, ucold;  
          BOOL hot1, hot2, cold1, cold2;  
        }  
    AD { hot1 = T1>=Thot1;  
        hot2 = T2>=Thot2;  
        cold1 = T1<=Tcold1;  
        cold2 = T2<=Tcold2;  
      }  
    DA { uhot = (IF cold1 | (cold2 & ~hot1) THEN Uh ELSE 0);  
        ucold = (IF hot1 | (hot2 & ~cold1) THEN Uc ELSE 0);  
      }  
    CONTINUOUS { T1 = T1+Ts*(-alpha1*(T1-Tamb)+k1*(uhot-ucold));  
                 T2 = T2+Ts*(-alpha2*(T2-Tamb)+k2*(uhot-ucold));  
      }  
  }  
}
```

```
>> S=mld('heatcoolmodel',Ts);
```

get the MLD model in MATLAB

```
>> [XX,TT]=sim(S,x0,U);
```

simulate the MLD model

EXAMPLE: ROOM TEMPERATURE CONTROL

- MLD model of the room temperature system

$$\left\{ \begin{array}{lcl} x(k+1) & = & Ax(k) + B_1u(k) + B_2\delta(k) + B_3z(k) + B_5 \\ y(k) & = & Cx(k) + D_1u(k) + D_2\delta(k) + D_3z(k) + D_5 \\ E_2\delta(k) + E_3z(k) & \leq & E_4x(k) + E_1u(k) + E_5 \end{array} \right.$$

- 2 continuous states (temperature T_1, T_2)
 - 1 continuous input (room temperature T_{amb})
 - 2 auxiliary continuous vars (power flows $u_{\text{hot}}, u_{\text{cold}}$)
 - 6 auxiliary binary vars (4 threshold events + 2 for the OR condition)
 - 20 mixed-integer inequalities
- In principle we have $2^6 = 64$ possible combinations of integer variables

EXAMPLE: ROOM TEMPERATURE CONTROL

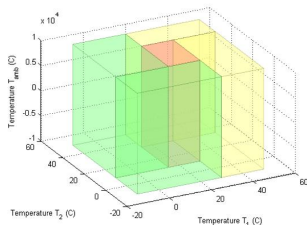
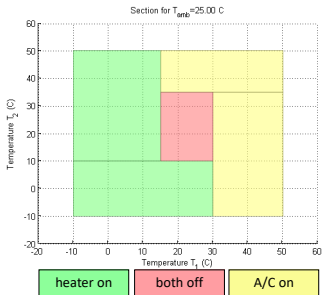
- PWA model of the room temperature system

$$x(k+1) = A_{i(k)}x(k) + B_{i(k)}u(k) + f_{i(k)}$$

$$y(k) = C_{i(k)}x(k) + D_{i(k)}u(k) + g_{i(k)}$$

$$i(k) \text{ s.t. } H_{i(k)}x(k) + J_{i(k)}u(k) \leq K_{i(k)}$$

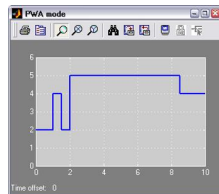
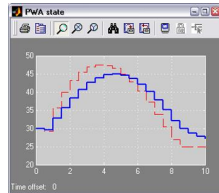
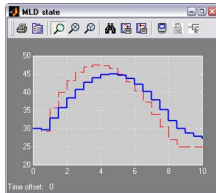
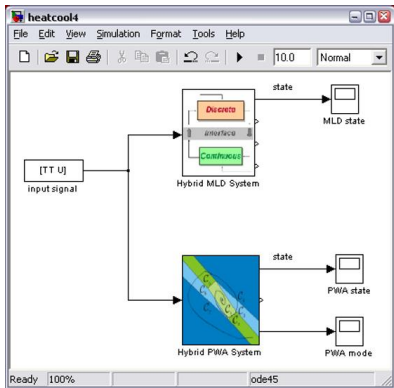
>> P=pwa (S) ;



5 polyhedral regions
(partition does not depend on input)

2 continuous states (T_1, T_2)
1 continuous input (T_{amb})

EXAMPLE: ROOM TEMPERATURE CONTROL



- MLD and PWA models are equivalent, hence simulated states are the same

USING PWA EQUIVALENCE FOR MODEL ANALYSIS

- Assume plant + controller can be modeled as DHA:
 - **plant** = approximated as PWA system (e.g.: nonlinear switched model)
 - **controller** = switched linear controller (e.g: combination of threshold conditions, logic, linear feedback laws, ...)
- Convert DHA to MLD form, then to PWA form
- The resulting closed-loop PWA model reveals how the closed-loop system behaves in different regions of the state-space
- Can analyze **closed-loop stability** analysis using **piecewise quadratic Lyapunov functions** (Johansson, Rantzer, 1998) (Mignone, Ferrari-Trecate, Morari, 2000)

OTHER EXISTING HYBRID MODELS

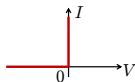
(Heemels, De Schutter, Bemporad, 2001)

- **Linear complementarity (LC)** systems (Heemels, 1999)

$$\begin{aligned}x(k+1) &= Ax(k) + B_1u(k) + B_2w(k) \\y(k) &= Cx(k) + D_1u(k) + D_2w(k) \\v(k) &= E_1x(k) + E_2u(k) + E_3w(k) + E_4 \\0 &\leq v(k) \perp w(k) \geq 0\end{aligned}$$

Examples:

mechanical systems,
electrical circuits



- **Min-max-plus-scaling (MMPS)** systems (De Schutter, Van den Boom, 2000)

$$\begin{aligned}x(k+1) &= M_x(x(k), u(k), w(k)) \\y(k) &= M_y(x(k), u(k), w(k)) \\0 &\geq M_c(x(k), u(k), w(k))\end{aligned}$$

Example:

discrete-event system

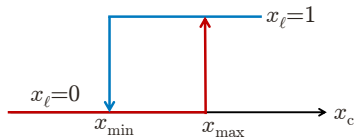
k = event counter

where M_{\square} are MMPS functions defined by the grammar

$$M := x_i | \alpha | \max(M_1, M_2) | \min(M_1, M_2) | M_1 + M_2 | \beta M_1$$

Example: $x(k+1) = 2 \max(x(k), 0) + \min(\frac{1}{2}u(k), 1)$

MODELING HYSTERESIS



- Hysteresis between $x_{\min} \leq x_c(k) \leq x_{\max}$

- Introduce two binary variables

$$[\delta_{\min}(k) = 1] \quad \leftrightarrow \quad [x_c(k) \leq x_{\min}]$$

$$[\delta_{\max}(k) = 1] \quad \leftrightarrow \quad [x_c(k) \geq x_{\max}]$$

- Introduce logic state $x_\ell \in \{0, 1\}$ with dynamics

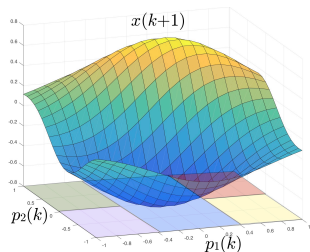
$$x_\ell(k+1) = (x_\ell(k) \wedge \neg \delta_{\min}(k)) \vee (\neg x_\ell(k) \wedge \delta_{\max}(k))$$

IDENTIFICATION OF HYBRID SYSTEMS

HYBRID SYSTEM IDENTIFICATION

- A hybrid model of the process may not be available from physical principles
- Therefore, a model must be either
 - **estimated from data** (model is unknown)
 - or **hybridized** (model is known but nonlinear)
- If one linear model is enough: easy problem (SYS-ID TBX) (Ljung, 1999)
- If switching sequence known: easy, just identify one linear model per mode
- If modes & dynamics must be identified simultaneously, we need **hybrid system identification** (or **piecewise affine regression**)

In industrial MPC most effort is spent in **identifying (multiple) linear prediction models** from data



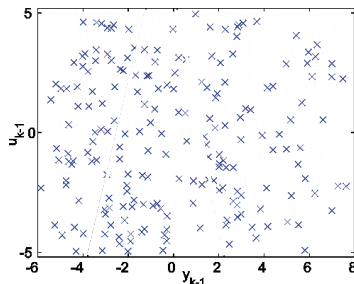
LEARNING PWA MODELS FROM DATA

Estimate from data **both** the **parameters** of the affine submodels and the **partition** of the PWA map

Example: Let the data be generated by the PWARX system

$$y_k = \begin{cases} \begin{bmatrix} -0.4 & 1 & 1.5 \end{bmatrix} \phi_k + \epsilon_k \\ \text{if } \begin{bmatrix} 4 & -1 & 10 \end{bmatrix} \phi_k < 0 \\ \begin{bmatrix} 0.5 & -1 & -0.5 \end{bmatrix} \phi_k + \epsilon_k \\ \text{if } \begin{bmatrix} -4 & 1 & 10 \\ 5 & 1 & -6 \end{bmatrix} \phi_k \leq 0 \\ \begin{bmatrix} -0.3 & 0.5 & -1.7 \end{bmatrix} \phi_k + \epsilon_k \\ \text{if } \begin{bmatrix} -5 & -1 & 6 \end{bmatrix} \phi_k < 0 \end{cases}$$

with $\phi_k = [y_{k-1} \ u_{k-1} \ 1]'$, $|u_k| \leq 5$, and $|\epsilon_k| \leq 0.1$

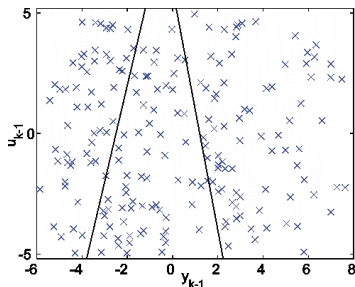


PWA IDENTIFICATION PROBLEM

Estimate from data **both** the **parameters** of the affine submodels and the **partition** of the PWA map

Example: Let the data be generated by the PWARX system

$$y_k = \begin{cases} \begin{bmatrix} -0.4 & 1 & 1.5 \end{bmatrix} \phi_k + \epsilon_k \\ \text{if } \begin{bmatrix} 4 & -1 & 10 \end{bmatrix} \phi_k < 0 \\ \begin{bmatrix} 0.5 & -1 & -0.5 \end{bmatrix} \phi_k + \epsilon_k \\ \text{if } \begin{bmatrix} -4 & 1 & 10 \\ 5 & 1 & -6 \end{bmatrix} \phi_k \leq 0 \\ \begin{bmatrix} -0.3 & 0.5 & -1.7 \end{bmatrix} \phi_k + \epsilon_k \\ \text{if } \begin{bmatrix} -5 & -1 & 6 \end{bmatrix} \phi_k < 0 \end{cases}$$



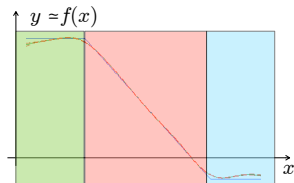
with $\phi_k = [y_{k-1} \ u_{k-1} \ 1]'$, $|u_k| \leq 5$, and $|\epsilon_k| \leq 0.1$

PWA REGRESSION PROBLEM

- **Problem:** Given input/output pairs $\{x(k), y(k)\}, k = 1, \dots, N$ and number s of models, compute a **piecewise affine** (PWA) approximation $y \approx f(x)$

$$f(x) = \begin{cases} F_1 x + g_1 & \text{if } H_1 x \leq K_1 \\ \vdots & \\ F_s x + g_s & \text{if } H_s x \leq K_s \end{cases}$$

- Need to learn **both** the parameters $\{F_i, g_i\}$ of the affine submodels **and** the partition $\{H_i, K_i\}$ of the PWA map from data (**off-line learning**)
- Possibly update model and partition as new data become available (**on-line learning**)



APPROACHES TO PWA IDENTIFICATION

- Mixed-integer linear or quadratic programming (Roll, Bemporad, Ljung, 2004)
- Partition of infeasible set of inequalities (Bemporad, Garulli, Paoletti, Vicino, 2005)
- K-means clustering in a feature space (Ferrari-Trecate, Muselli, Liberati, Morari, 2003)
- Bayesian approach (Juloski, Wieland, Heemels, 2004)
- Kernel-based approaches (Pillonetto, 2016)
- Hyperplane clustering in data space (Münz, Krebs, 2002)
- **Recursive multiple least squares & PWL separation** (Breschi, Piga, Bemporad, 2016)

PWA REGRESSION ALGORITHM

(Breschi, Piga, Bemporad, 2016)

1. Estimate models $\{F_i, g_i\}$ **recursively**. Let $e_i(k) = y(k) - F_i x(k) - g_i$ and only update model $i(k)$ such that

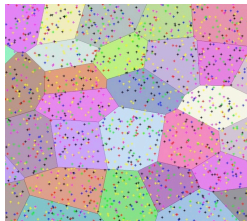
$$i(k) \leftarrow \arg \min_{i=1, \dots, s} \underbrace{e_i(k)' \Lambda_e^{-1} e_i(k)}_{\substack{\text{one-step prediction error} \\ \text{of model } \#i}} + \underbrace{(x(k) - c_i)' R_i^{-1} (x(k) - c_i)}_{\substack{\text{proximity to centroid} \\ \text{of cluster } \#i}}$$

using **recursive LS** and **inverse QR decomposition** (Alexander, Ghirnikar, 1993)

This also splits the data points $x(k)$ in **clusters** $C_i = \{x(k) : i(k) = i\}$

2. Compute a polyhedral partition $\{H_i, K_i\}$ of the regressor space via **multi-category linear separation**

$$\phi(x) = \max_{i=1, \dots, s} \{w_i' x - \gamma_i\}$$



- Identification of **piecewise-affine ARX** model

$$\begin{aligned} \begin{bmatrix} y_1(k) \\ y_2(k) \end{bmatrix} &= \begin{bmatrix} -0.83 & 0.20 \\ 0.30 & -0.52 \end{bmatrix} \begin{bmatrix} y_1(k-1) \\ y_2(k-1) \end{bmatrix} + \begin{bmatrix} -0.34 & 0.45 \\ -0.30 & 0.24 \end{bmatrix} \begin{bmatrix} u_1(k-1) \\ u_2(k-1) \end{bmatrix} \\ &+ \begin{bmatrix} 0.20 \\ 0.15 \end{bmatrix} + \max \left\{ \begin{bmatrix} 0.20 & -0.90 \\ 0.10 & -0.42 \end{bmatrix} \begin{bmatrix} y_1(k-1) \\ y_2(k-1) \end{bmatrix} \right. \\ &\left. + \begin{bmatrix} 0.42 & 0.20 \\ 0.50 & 0.64 \end{bmatrix} \begin{bmatrix} u_1(k-1) \\ u_2(k-1) \end{bmatrix} + \begin{bmatrix} 0.40 \\ 0.30 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \end{bmatrix} \right\} + e_o(k), \end{aligned}$$

- Quality of fit:** best fit rate (BFR) = $\max \left\{ 1 - \frac{\|y_{o,i} - \hat{y}_i\|_2}{\|y_{o,i} - \bar{y}_i\|_2}, 0 \right\}, i = 1, 2$

y_o = measured, \hat{y} = open-loop simulated, \bar{y} = sample mean of y_o

		$N = 4000$	$N = 20000$	$N = 100000$
y_1	(Off-line) RLP	96.0 %	96.5 %	99.0 %
	(Off-line) RPSN	96.2 %	96.4 %	98.9 %
	(On-line) ASGD	86.7 %	95.0 %	96.7 %
y_2	(Off-line) RLP	96.2 %	96.9 %	99.0 %
	(Off-line) RPSN	96.3 %	96.8 %	99.0 %
	(On-line) ASGD	87.4 %	95.2 %	96.4 %

BFR on validation data, open-loop validation

RLP = Robust linear programming

(Bennett, Mangasarian, 1994)

RPSN = Piecewise-smooth Newton method

(Bemporad, Bernardini, Patrino, 2015)

ASGD = Averaged stochastic gradient descent

(Bottou, 2012)

- CPU time for computing the partition:**

	$N = 4000$	$N = 20000$	$N = 100000$
(Off-line) RLP	0.308 s	3.227 s	112.435 s
(Off-line) RPSN	0.016 s	0.086 s	0.365 s
(On-line) ASGD	0.013 s	0.023 s	0.067 s

- Identification of **linear parameter varying ARX** model

$$\begin{bmatrix} y_1(k) \\ y_2(k) \end{bmatrix} = \begin{bmatrix} \bar{a}_{1,1}(p(k)) & \bar{a}_{1,2}(p(k)) \\ \bar{a}_{2,1}(p(k)) & \bar{a}_{2,2}(p(k)) \end{bmatrix} \begin{bmatrix} y_1(k-1) \\ y_2(k-1) \end{bmatrix} + \begin{bmatrix} \bar{b}_{1,1}(p(k)) & \bar{b}_{1,2}(p(k)) \\ \bar{b}_{2,1}(p(k)) & \bar{b}_{2,2}(p(k)) \end{bmatrix} \begin{bmatrix} u_1(k-1) \\ u_2(k-1) \end{bmatrix} + e_o(k)$$

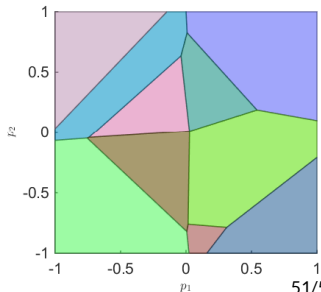
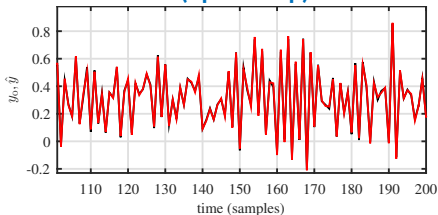
$\bar{a}(p)$ = PWA function of p
 $\bar{b}(p)$ has **quadratic** and **sin** terms

- Quality of fit (BFR):

	y_1	y_2
PWA regression	87 %	84 %
parametric LPV*	80 %	70 %

* (Bamieh, Giarré, 2002)

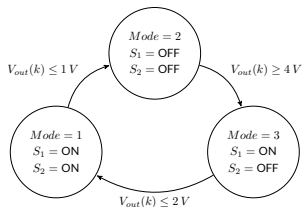
- Validation data (open-loop):



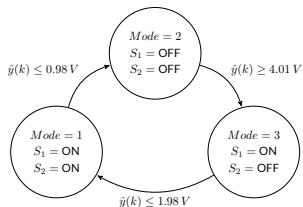
IDENTIFICATION OF HYBRID SYSTEMS WITH LOGIC STATES

(Breschi, Piga, Bemporad, 2016)

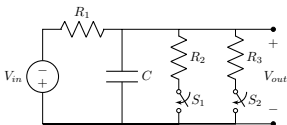
- Identification of a **hybrid model** with logic states



true system



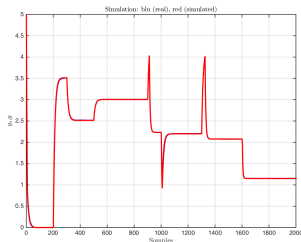
identified system



Quality of fit: BFR=96.64 % (validation)

CPU time for identification: 78 ms

(2000 samples, MacBook Pro 2.8 GHz)



PARC - PIECEWISE AFFINE REGRESSION AND CLASSIFICATION

- Hybrid SYS-ID = PWA regression & classification (**PARC**): (Bemporad, 2021)

feature vector: $z_k = \begin{bmatrix} x_c(k) \\ u_c(k) \\ x_\ell(k) \\ u_\ell(k) \end{bmatrix} \in \mathbb{R}^{n_c+m_c} \times \{0,1\}^{n_\ell+m_\ell}$

target vector: $v_k = \begin{bmatrix} x_c(k+1) \\ y_c(k) \\ x_\ell(k+1) \\ y_\ell(k) \end{bmatrix} \in \mathbb{R}^{n_c+p_c} \times \{0,1\}^{n_\ell+p_\ell}$

- Input:** dataset $\{z_k, v_k\}_{k=0}^{N-1}$ and desired number K of partitions.
The **PARC** algorithm finds:

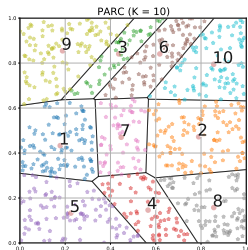
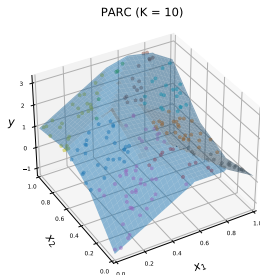
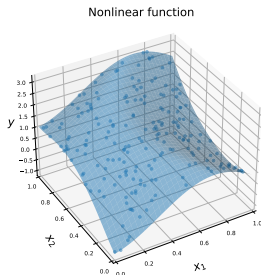
- $\begin{bmatrix} x_c(k+1) \\ y_c(k) \end{bmatrix} = a_j z_k + b_j$ by **ridge regression** (= L_2 -regularized least squares)
 $j = 1, \dots, K$
- $\begin{bmatrix} x_\ell(k+1) \\ y_\ell(k) \end{bmatrix}_i = \frac{1}{2}(1 + \text{sign}(c_j^i z_k + d_j^i))$ by **logistic regression**
 $j = 1, \dots, K$
- A PWL separation function $\phi(x) = \max_j \{\omega^j z + \gamma^j\}$ by **softmax regression**

- PARC uses **block-coordinate descent** to alternate between ridge, logistic, softmax regression and reassigning data to one of the K clusters

PARC - PIECEWISE AFFINE REGRESSION AND CLASSIFICATION

(Bemporad, 2021)

- Simple PWA regression example:
 - 1000 samples of $y = \sin(4x_1 - 5(x_2 - 0.5)^2) + 2x_2$ (use 80% for training)
 - Look for PWA approximation over $K = 10$ polyhedral regions



- Code download: 

<http://cse.lab.imtlucca.it/~bemporad/parc/>