MODEL PREDICTIVE CONTROL

HYBRID MODELS FOR MPC

Alberto Bemporad

imt.lu/ab
Course structure

- Basic concepts of model predictive control (MPC) and linear MPC
- Linear time-varying and nonlinear MPC
- Quadratic programming (QP) and explicit MPC
  - Hybrid MPC
  - Stochastic MPC
  - Learning-based MPC

Course page:
http://cse.lab.imtlucca.it/~bemporad/mpc_course.html
HYBRID MODELS
HYBRID DYNAMICAL SYSTEMS

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

- Variables are **real-valued**
  \[x_c \in \mathbb{R}^{n_c}, \quad u_c \in \mathbb{R}^{m_c}\]
- Difference/differential equations
- **Linear inequality** constraints
TECHNOLOGICAL PUSH FOR STUDYING HYBRID SYSTEMS

- Technological push for studying hybrid systems
- Continuous dynamical system
- Discrete inputs
- Symbols
- Automaton / logic
- Interface
- Continuous dynamical system
- Continuous states
- Continuous inputs
- Embedded systems
- Networked control systems
- Cyber-physical systems

- >1995
- >2005
- >2010

"Model Predictive Control" - © 2023 A. Bemporad. All rights reserved.
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

\[
\begin{align*}
    x' &= Ax + Bu \\
    y &= Cx + Du
\end{align*}
\]

*linear hybrid systems*

\[
\begin{align*}
    x' &= f(x, u, t) \\
    y &= g(x, u, t)
\end{align*}
\]

"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)
\[ x(k+1) = A_{i(k)} x(k) + B_{i(k)} u(k) + f_{i(k)} \]
\[ y(k) = C_{i(k)} x(k) + D_{i(k)} u(k) + g_{i(k)} \]
\[ i(k) \quad \text{s.t.} \quad H_{i(k)} x(k) + J_{i(k)} u(k) \leq K_{i(k)} \]

- PWA systems can approximate nonlinear dynamics arbitrarily well (even discontinuous ones)
Discrete Hybrid Automaton (DHA)

(Torrisi, Bemporad, 2004)

Event Generator

\[ \delta_e = 1 \iff [Hx_c + Ku_c \leq W] \]

Switched Affine System

\[ x_c(k+1) = A_i x_c(k) + B_i u_c(k) + f_i \]

Mode Selector

\[ i = f_M(x_\ell, u_\ell, \delta_e) \]

\[ x_\ell \in \{0, 1\}^{n_\ell} = \text{binary state} \]

\[ u_\ell \in \{0, 1\}^{m_\ell} = \text{binary input} \]

\[ \delta_e \in \{0, 1\}^{n_e} = \text{event variable} \]

\[ x_c \in \mathbb{R}^{n_c} = \text{real-valued state} \]

\[ u_c \in \mathbb{R}^{m_c} = \text{real-valued input} \]

\[ i \in \{1, \ldots, s\} = \text{current mode} \]
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 variables are generated by linear threshold conditions over continuous

\[ \delta_e^i(k) = 1 \iff \left[ H^i x_c(k) + K^i u_c(k) \leq W^i \right] \]

\[ x_c \in \mathbb{R}^{n_c}, \quad u_c \in \mathbb{R}^{m_c} \]
\[ \delta_e \in \{0, 1\}^{n_e} \]

• Example: \[ \delta_e(k) = 1 \iff [x_c(k) \geq 0] \]
• The binary state of the finite state machine evolves according to a Boolean state update function $f_B : \{0, 1\}^{n_\ell+m_\ell+n_e} \rightarrow \{0, 1\}^{n_\ell}$:

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

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

• Example: $x_\ell(k+1) = \neg\delta_e(k) \lor (x_\ell(k) \land u_\ell(k))$
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
  
  $i(k) = f_M(x_\ell(k), u_\ell(k), \delta_e(k))$

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

- Example:
  
  $i(k) = \left[ \neg u_\ell(k) \lor x_\ell(k) \right] / u_\ell(k) \land x_\ell(k)$

  \[
  \begin{array}{c|c|c}
  u_\ell/x_\ell & 0 & 1 \\
  \hline
  0 & i = [\frac{1}{0}] & i = [\frac{1}{0}] \\
  1 & i = [0] & i = [\frac{1}{1}] \\
  \end{array}
  \]

  the system has 3 modes
Conversion of Logic Formulas to Linear Inequalities

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

- Key observation: \( X_1 \lor X_2 = \text{true} \)
  \[
  \delta_1 + \delta_2 \geq 1, \quad \delta_1, \delta_2 \in \{0, 1\}
  \]
- We want to impose the Boolean statement
  \[
  F(X_1, \ldots, X_n) = \text{true}
  \]
- Convert the formula to **Conjunctive Normal Form (CNF)**
  \[
  \bigwedge_{j=1}^{m} \left( \bigvee_{i \in P_j} X_i \bigvee_{i \in N_j} \bar{X}_i \right) = \text{true}, \quad P_j \cup N_j \subseteq \{1, \ldots, n\}
  \]
- Transform the CNF into the equivalent linear inequalities
  \[
  \begin{aligned}
  \sum_{i \in P_1} \delta_i + \sum_{i \in N_1} (1 - \delta_i) &\geq 1 \\
  \vdots & \quad \vdots \\
  \sum_{i \in P_m} \delta_i + \sum_{i \in N_m} (1 - \delta_i) &\geq 1
  \end{aligned}
  \]
  \[
  A\delta \leq b, \quad \delta \in \{0, 1\}^n
  \]
  \(\text{polyhedron}\)

Any logic proposition can be translated into integer linear inequalities
• Example:

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

• Convert Conjunctive Normal Form (CNF):

(see e.g. \texttt{http://formal.cs.utah.edu:8080/pbl/PBL.php} or just google “CNF + converter” ...)

\[ (X_3 \lor \neg X_1 \lor \neg X_2) \land (X_1 \lor \neg X_3) \land (X_2 \lor \neg X_3) \]

• Transform into inequalities:

\[
\begin{align*}
\delta_3 + (1 - \delta_1) + (1 - \delta_2) & \geq 1 \\
\delta_1 + (1 - \delta_3) & \geq 1 \\
\delta_2 + (1 - \delta_3) & \geq 1
\end{align*}
\]
• Consider the Boolean statement $F(X_1, \ldots, X_n) = \text{true}$ 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 $F$

(Mignone, Bemporad, Morari, 1999)

<table>
<thead>
<tr>
<th>$X_1$</th>
<th>$X_2$</th>
<th>...</th>
<th>$X_n$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>...</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>...</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>...</td>
<td>0</td>
</tr>
</tbody>
</table>

• 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:** \( F(X_1, X_2, X_3) = [X_3 \leftrightarrow X_1 \land X_2] \) (logic and)

<table>
<thead>
<tr>
<th>( X_1 )</th>
<th>( X_2 )</th>
<th>( X_3 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

\( T(F) : 0 \quad 1 \quad 0 \)

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

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

\[
\gg \ V=\text{struct('V',\[0 0 0;0 1 0;1 0 0;1 1 1\])};
\gg \ H=\text{cddmex('hull',V)};A=H.A,b=H.B
\]
• The polyhedron obtained via convex hull is the smallest one
• The one obtained via CNF may be larger. Example:

\[(X_1 \lor X_2) \land (X_1 \lor X_3) \land (X_2 \lor X_3) = \text{true}\]

• **Note:** no other example with 3 vars but

\[(X_1 \lor X_2) \land (X_1 \lor X_3) \land (X_2 \lor X_3) \land (\neg X_1 \lor \neg X_2 \lor \neg X_3) = \text{true}\]
• Consider the **if-and-only-if** condition

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

\[x_c \in \mathcal{X}, \quad \delta \in \{0, 1\}\]

• Assume \(\mathcal{X} \subset \mathbb{R}^{nc}\) 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) (Bemporad, 2022)

\[
a_+ = \max\{a, 0\}
\]
\[
a_- = a_+ - a = \max\{-a, 0\}
\]
\[
m < a_+ \ell - a_- u - b
\]
\[
M > a_+ u - a_- \ell - b
\]
• Consider the **if-then-else** condition

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

\[
x_c \in \mathcal{X} \\
\delta \in \{0, 1\} \\
z \in \mathbb{R}
\]

• Assume \( \mathcal{X} \subset \mathbb{R}^{nc} \) bounded. Let \( M_1, M_2 \) and \( m_1, m_2 \) such that \( \forall x_c \in \mathcal{X} \)

\[
M_1 > a'_1 x_c - b_1 > m_1 \\
M_2 > a'_2 x_c - b_2 > m_2
\]

• The if-then-else condition is equivalent to

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

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

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 eq. \( y_c(k) = C_i x_c(k) + D_i u_c(k) + g_i \) admits a similar transformation
$X_1 \lor X_2 = \text{TRUE} \quad \Rightarrow \quad \delta_1 + \delta_2 \geq 1, \quad \delta_1, \delta_2 \in \{0, 1\}$

Any logic statement $f(X) = \text{TRUE}$

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

(CNF)

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

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

**IF** $[\delta = 1]$ **THEN** $z = a_1^T x + b_1^T u + f_1$  
**ELSE** $z = a_2^T x + b_2^T u + f_2$

$$\begin{cases} 
(m_1 - M_2)(1 - \delta) + z \leq a_1 x + b_1 u + f_1 \\
(m_2 - M_1)(1 - \delta) - z \leq -a_1 x - b_1 u - f_1 \\
(m_2 - M_1)\delta + z \leq a_2 x + b_2 u + f_2 \\
(m_1 - M_2)\delta - z \leq -a_2 x - b_2 u - f_2 
\end{cases}$$
• By converting logic relations into mixed-integer linear inequalities, a DHA can be rewritten as the **Mixed Logical Dynamical (MLD)** system.

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

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

• The translation from DHA to MLD can be automatized, see e.g. the language **HYSDEL** (HYbrid Systems DEscription Language) (Torrisi, 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 system\(^1\):  \[ x(k + 1) = \begin{cases} 
0.8x(k) + u(k) & \text{if } x(k) \geq 0 \\
-0.8x(k) + u(k) & \text{if } x(k) < 0 
\end{cases} \]

- Introduce event variable \([\delta(k) = 1] \leftrightarrow [x(k) \geq 0]\) and use big-M technique:
  \[
  x(k) \geq m(1 - \delta(k)) \quad M = -m = 10 \\
x(k) \leq -\epsilon + (M + \epsilon)\delta(k) \quad \epsilon > 0 \text{ “small”}
  \]

- Since \(x(k + 1) = 1.6\delta(k)x(k) - 0.8x(k) + u(k)\), introduce the aux variable \(z(k) = \delta(k)x(k)\)
  \[
  z(k) \leq M\delta(k) \\
z(k) \geq m\delta(k) \\
z(k) \leq x(k) - m(1 - \delta(k)) \\
z(k) \geq x(k) - M(1 - \delta(k)) \\
\]

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

\(^1\)This is the nonlinear system \(x(k + 1) = 0.8|x(k)| + u(k)\)
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 */

Additional relations constraining system’s variables
How to model the bouncing ball as a discrete-time hybrid system?

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

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

$$\alpha \in [0, 1]$$
- **Case** \( y(k) > 0 \) (ball falling):

\[
\begin{align*}
    v(k) & \approx \frac{y(k) - y(k-1)}{T_s}, \\
    -g & \approx \frac{v(k) - v(k-1)}{T_s}.
\end{align*}
\]

\[
\begin{aligned}
    v(k+1) &= v(k) - T_s g, \\
    y(k+1) &= y(k) + T_s v(k+1) = y(k) + T_s v(k) - T_s^2 g.
\end{aligned}
\]

- **Case** \( y(k) \leq 0 \) (ground level):

\[
\begin{aligned}
    v(k+1) &= -(1 - \alpha) v(k), \\
    y(k+1) &= y(k-1) = y(k) - T_s v(k).
\end{aligned}
\]

- We need a binary variable \( \delta(k) = 1 \) if \( y(k) \leq 0 \).
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
>> 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
Two hybrid models $\Sigma_1, \Sigma_2$ 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 $y_1(k) \equiv y_2(k), \forall k = 0, 1, \ldots$
**EQUIVALENCE OF HYBRID MODELS**

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

  **Proof:** For a given combination \((x_\ell, u_\ell, \delta)\) 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

\[
\begin{align*}
x(k + 1) &= A_i(k)x(k) + B_i(k)u(k) + f_i(k) \\
y(k) &= C_i(k)x(k) + D_i(k)u(k) + g_i(k) \\
i(k) &= \text{such that } \left[ \begin{array}{c} x(k) \\ u(k) \end{array} \right] \in C_i(k)
\end{align*}
\]

with \( C_i = \{ [x \ u] : H_i x + J_i u \leq K_i \} \), and \( \bar{C}_i \cap \bar{C}_j = \emptyset, \forall i \neq j, i, j = 1, \ldots, s \)

\( \{C_i\} \) is a polyhedral partition of the set \( C \triangleq \bigcup_{i=1}^{s} C_i \)

- Introduce \( s \) binary variables \( \delta_i, i = 1, \ldots, s \) and the logic constraints

\[
\begin{align*}
\bigoplus_{i=1}^{s}[\delta_i = 1] &= \text{true} \\
\sum_{i=1}^{s} \delta_i &= 1
\end{align*}
\]

\[
H_i x + J_i u \leq K_i + M_i (1 - \delta_i)
\]

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

$$\begin{align*}
  x(k+1) &= \sum_{i=1}^{s} z_i(k) \\
  y(k) &= \sum_{i=1}^{s} w_i(k)
\end{align*}$$
• A PWA system with bounded states and inputs is equivalent to the disjunction

\[
\bigvee_{i=1}^{s} \begin{bmatrix} H_i x(k) + J_i u(k) & \leq K_i \\
 x(k+1) = A_i x(k) + B_i u(k) + f_i & 
\end{bmatrix} \quad \begin{bmatrix} x_{\ell b} \leq x(k) \leq x_{ub} \\
 u_{\ell b} \leq u(k) \leq u_{ub} 
\end{bmatrix}
\]

• Introduce \( s \) binary variables \( \delta_1(k), \ldots, \delta_s(k) \) subject to \( \sum_{i=1}^{s} \delta_i(k) = 1 \)

• Introduce the convex hull relaxation of the disjunction

\[
x(k) = \sum_{i=1}^{s} v_i(k), \quad x_{\ell b} \delta_i(k) \leq v_i(k) \leq x_{ub} \delta_i(k)
\]

\[
u(k) = \sum_{i=1}^{s} w_i(k), \quad u_{\ell b} \delta_i(k) \leq w_i(k) \leq u_{ub} \delta_i(k)
\]

and impose

\[
x(k+1) = \sum_{i=1}^{s} A_i v_i(k) + B_i w_i(k) + f_i \delta_i(k), \quad H_i v_i(k) + J_i w_i(k) \leq K_i \delta_i(k)
\]
• Only introduce $s$ binary variables $\delta_i, i = 1, \ldots, s$ and set:

\[
\begin{align*}
  m^x_i (1 - \delta_i(k)) & \leq x(k + 1) - A_i x(k) - B_i u(k) - f_i \leq M^x_i (1 - \delta_i(k)) & [\delta_i(k) = 1] \rightarrow [x(k + 1) = A_i x(k) + B_i u(k) + f_i] \\
  m^y_i (1 - \delta_i(k)) & \leq y(k) - C_i x(k) - D_i u(k) - g_i \leq M^y_i (1 - \delta_i(k)) & [\delta_i(k) = 1] \rightarrow [y(k) = C_i x(k) + D_i u(k) + g_i] \\
  H_i x(k) + J_i u(k) & \leq K_i + M_i (1 - \delta_i(k)) & [\delta_i(k) = 1] \rightarrow [H_i x(k) + J_i u(k) \leq K_i] \\
  \sum_{i=1}^{s} \delta_i(k) & \leq 1 \\
  \bigoplus_{i=1}^{s} [\delta_i(k) = 1] & = \text{true}
\end{align*}
\]

where $m^x_i, M^x_i, m^y_i, M^y_i$ are suitably defined upper and lower bounds.
>> 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
\[
\frac{dT_i}{dt} = -\alpha_i (T_i - T_{amb}) + k_i (u_{hot} - u_{cold})
\]
\[i = 1, 2\]

*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 [-50,50]; }
      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);

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

get the MLD model in MATLAB

simulate the MLD model
**Example: Room Temperature Control**

- MLD model of the room temperature system

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

- 2 continuous states (temperature \(T_1, T_2\))
- 1 continuous input (room temperature \(T_{\text{amb}}\))
- 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 binary variables
EXAMPLE: ROOM TEMPERATURE CONTROL

- PWA model of the room temperature system

\[
x(k + 1) = A_i(k) x(k) + B_i(k) u(k) + f_i(k)
\]
\[
y(k) = C_i(k) x(k) + D_i(k) u(k) + g_i(k)
\]

\[i(k) \text{ s.t. } H_i(k) x(k) + J_i(k) u(k) \leq K_i(k)\]

5 polyhedral regions
(partition does not depend on input)

2 continuous states \((T_1, T_2)\)
1 continuous input \((T_{amb})\)
• MLD and PWA models are equivalent, hence simulated states are the same
• Assume plant + controller can be modeled as DHA:
  - \textbf{plant} = approximated as PWA system (e.g.: nonlinear switched model)
  - \textbf{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 \textbf{closed-loop stability} analysis using \textbf{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)

  \[
  x(k + 1) = Ax(k) + B_1 u(k) + B_2 w(k) \\
  y(k) = Cx(k) + D_1 u(k) + D_2 w(k) \\
  v(k) = E_1 x(k) + E_2 u(k) + E_3 w(k) + E_4 \\
  0 \leq v(k) \perp w(k) \geq 0
  \]

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

  \[
  x(k + 1) = M_x(x(k), u(k), w(k)) \\
  y(k) = M_y(x(k), u(k), w(k)) \\
  0 \geq M_c(x(k), u(k), w(k))
  \]

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

Examples:
- mechanical systems, electrical circuits
- discrete-event system
  \( k \) = event counter
• Hysteresis between $x_{\text{min}} \leq x_c(k) \leq x_{\text{max}}$

• Introduce two binary variables

  $[\delta_{\text{min}}(k) = 1] \iff [x_c(k) \leq x_{\text{min}}]$
  $[\delta_{\text{max}}(k) = 1] \iff [x_c(k) \geq x_{\text{max}}]$

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

  $x_\ell(k + 1) = (x_\ell(k) \land \neg \delta_{\text{min}}(k)) \lor (\neg x_\ell(k) \land \delta_{\text{max}}(k))$
IDENTIFICATION OF HYBRID SYSTEMS
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

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

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

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

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 \( \{x(k), y(k)\}, k = 1, \ldots, N \) and number \( s \) of models, compute a **piecewise affine** (PWA) approximation \( y \approx f(x) \)

\[
v(k) = \begin{cases} 
F_1 z(k) + g_1 & \text{if } H_1 z(k) \leq K_1 \\
\vdots \\
F_s z(k) + g_s & \text{if } H_s z(k) \leq K_s 
\end{cases}
\]

\[
v(k) = \begin{bmatrix} x(k+1) \\ y(k) \end{bmatrix}, \quad z(k) = \begin{bmatrix} x(k) \\ u(k) \end{bmatrix}
\]

- 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 (**offline learning**)

- Possibly update model+partition as new data are available (**online learning**)

- Any **ML technique** can be applied that leads to PWA models, such as (leaky)ReLU-NNs, decision trees, softmax regression, KNN, ...

"Model Predictive Control" - © 2023 A. Bemporad. All rights reserved.
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)
- Piecewise affine regression and classification (PARC) (Bemporad, 2022)
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

\[
i(k) \leftarrow \text{arg min}_{i=1,\ldots,s} \left[ e_i(k)' \Lambda_e^{-1} e_i(k) + (x(k) - c_i)' R_i^{-1} (x(k) - c_i) \right]
\]

where \( \Lambda_e = \text{one-step prediction error of model } i \)
\( R_i = \text{proximity to centroid of cluster } i \)

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**

\[
\phi(x) = \max_{i=1,\ldots,s} \{ w_i' x - \gamma_i \}
\]
**PWA Regression Examples**

(Breschi, Piga, Bemporad, 2016)

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

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

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

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

<table>
<thead>
<tr>
<th>( N )</th>
<th>(offline) RLP</th>
<th>(Offline) RPSN</th>
<th>(Online) ASGD</th>
</tr>
</thead>
<tbody>
<tr>
<td>( N = 4000 )</td>
<td>96.0 %</td>
<td>96.2 %</td>
<td>86.7 %</td>
</tr>
<tr>
<td>( N = 20000 )</td>
<td>96.5 %</td>
<td>96.4 %</td>
<td>95.0 %</td>
</tr>
<tr>
<td>( N = 100000 )</td>
<td>99.0 %</td>
<td>98.9 %</td>
<td>96.7 %</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>( N )</th>
<th>(offline) RLP</th>
<th>(Offline) RPSN</th>
<th>(Online) ASGD</th>
</tr>
</thead>
<tbody>
<tr>
<td>( N = 4000 )</td>
<td>96.2 %</td>
<td>96.3 %</td>
<td>87.4 %</td>
</tr>
<tr>
<td>( N = 20000 )</td>
<td>96.9 %</td>
<td>96.8 %</td>
<td>95.2 %</td>
</tr>
<tr>
<td>( N = 100000 )</td>
<td>99.0 %</td>
<td>99.0 %</td>
<td>96.4 %</td>
</tr>
</tbody>
</table>

BFR on validation data, open-loop validation

- **CPU time for computing the partition**: (i7 2.40-GHz Intel core)

<table>
<thead>
<tr>
<th>( N )</th>
<th>(Offline) RLP</th>
<th>(Offline) RPSN</th>
<th>(Online) ASGD</th>
</tr>
</thead>
<tbody>
<tr>
<td>( N = 4000 )</td>
<td>0.308 s</td>
<td>0.016 s</td>
<td>0.013 s</td>
</tr>
<tr>
<td>( N = 20000 )</td>
<td>3.227 s</td>
<td>0.086 s</td>
<td>0.023 s</td>
</tr>
<tr>
<td>( N = 100000 )</td>
<td>112.435 s</td>
<td>0.365 s</td>
<td>0.067 s</td>
</tr>
</tbody>
</table>

RLP = Robust linear programming
(Bennett, Mangasarian, 1994)

RPSN = Piecewise-smooth Newton method
(Bemporad, Bernardini, Patrinos, 2015)

ASGD = Averaged stochastic gradient descent
(Bottou, 2012)
PWA REGRESSION EXAMPLES

(Breschi, Piga, Bemporad, 2016)

• Identification of linear parameter varying ARX model

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

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

• Quality of fit (BFR):

<table>
<thead>
<tr>
<th>PWA regression</th>
<th>(y_1)</th>
<th>(y_2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>87 %</td>
<td>84 %</td>
<td></td>
</tr>
<tr>
<td>parametric LPV*</td>
<td>80 %</td>
<td>70 %</td>
</tr>
</tbody>
</table>

* (Bamieh, Giarré, 2002)

• Validation data (open-loop):

"Model Predictive Control" - © 2023 A. Bemporad. All rights reserved.
Identification of hybrid systems with logic states

- 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)
• New **Piecewise Affine Regression and Classification (PARC)** algorithm

• **Training dataset:**
  - **feature vector** $z \in \mathbb{R}^n$ (categorical features **one-hot encoded** in $\{0, 1\}$)
  - **target vector** $v_c \in \mathbb{R}^{mc}$ (numeric), $v_{di} \in \{w_{di}^1, \ldots, w_{di}^{m_i}\}$ (categorical)

• **PARC iteratively** **clusters** training data in $K$ sets and **fits** linear predictors:
  1. fit $v_c = a_j z + b_j$ by **ridge regression** ($=\ell_2$-regularized least squares)
  2. fit $v_{di} = w_{di}^{h_*}$, $h_* = \arg\max\{a_{dih}^h z + b_{di}^h\}$ by **softmax regression**
  3. fit a convex **PWL** separation function by **softmax regression**
      \[
      \Phi(z) = \omega^j(z) z + \gamma^j(z), \quad j(z) = \min \left\{ \arg\max_{j=1,\ldots,K} \{\omega^j z + \gamma^j\} \right\}
      \]

• Data reassigned to clusters based on weighted fit/PWL separation criterion

• **PARC** is a **block-coordinate descent** algorithm $\Rightarrow$ (local) convergence ensured
• 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/](http://cse.lab.imtlucca.it/~bemporad/parc/)
• **Example:** moving cart and bumpers + heat transfer during bumps.

Spring and viscous forces are **nonlinear**.

• Categorical input $F \in \{-\bar{F}, 0, \bar{F}\}$ and categorical output $c \in \{\text{green}, \text{yellow}, \text{red}\}$

• Continuous-time system simulated for 2,000 s, sample time = 0.5 s (=4000 training samples)

• Feature vector $z_k = [y_k, \dot{y}_k, T_k, F_k]$

• Target vector $v_k = [y_{k+1}, \dot{y}_{k+1}, T_{k+1}, c_k]$

• Hybrid model learned by **PARC** ($K = 5$ regions)
• **Open-loop** simulation on 500 s **test** data:

![Continuous-time system](image1.png)

![Discrete-time PWA model](image2.png)

- **Model fit is good enough for MPC design purposes (see later ...)**