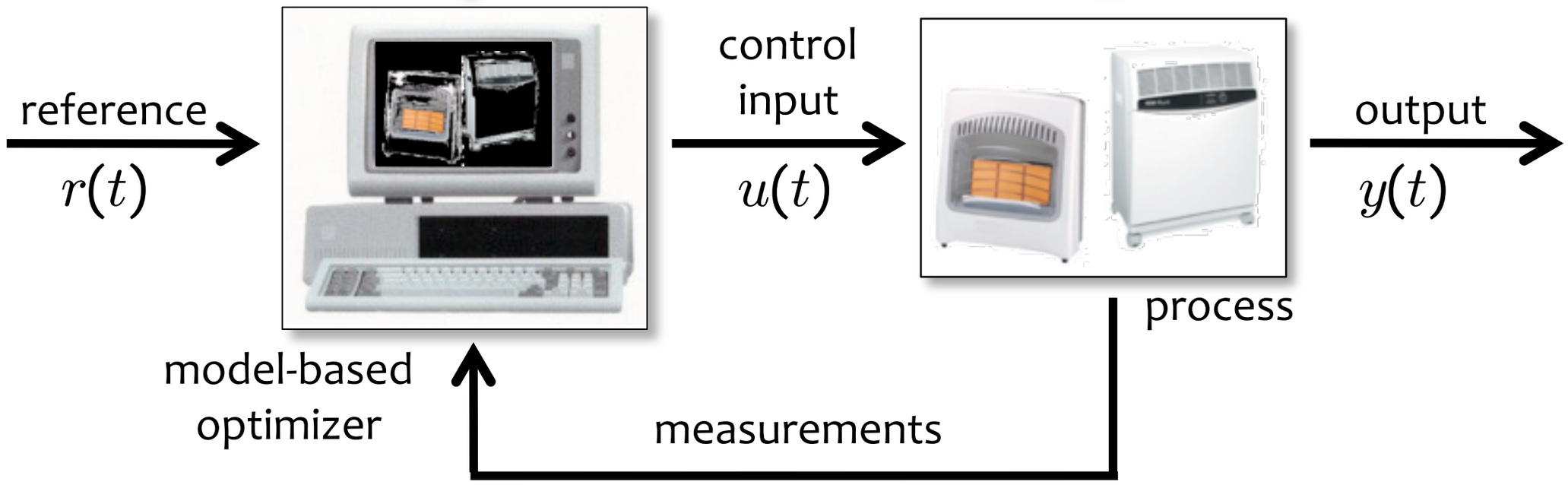
HYBRID SYSTEMS: MODEL PREDICTIVE CONTROL

Alberto Bemporad - “Model Predictive Control” course - Academic year 2017/18

MPC OF HYBRID SYSTEMS

$$\begin{cases} x_{k+1} = Ax_k + B_1u_k + B_2\delta_k + B_3z_k + B_5 \\ y_k = Cx_k + D_1u_k + D_2\delta_k + D_3z_k + D_5 \\ E_2\delta_k + E_3z_k \le E_4x_k + E_1u_k + E_5 \end{cases}$$



Use a hybrid dynamical model of the process to predict its future evolution and choose the "best" control action

MPC FOR HYBRID SYSTEMS

- At time t solve the finite-horizon optimal control problem w.r.t. $U \triangleq \begin{bmatrix} u_0 \\ \vdots \\ u_{N-1} \end{bmatrix}$

$$\min_{u_0,\dots,u_{N-1}} \quad \sum_{k=0}^{N-1} \|y_k - r\|_Q^2 + \|u_k - u_r\|_R^2$$
$$+\sigma\left(\|\delta_k - \delta_r\|^2 + \|z_k - z_r\|^2 + \|x_k - x_r\|^2\right)$$

subject to MLD equations

$$x_0 = x(t)$$

$$x_N = x_r$$

$Q, R \succ 0, \sigma > 0$

notation:

$\|v\|_Q^2 \triangleq v'Qv$

where the equilibrium condition $(x_r, u_r, \delta_r, z_r)$ is obtained by solving the following **mixed-integer feasibility problem**

$$\begin{cases} x_r = Ax_r + B_1u_r + B_2\delta_r + B_3z_r + B_5 \\ r = Cx_r + D_1u_r + D_2\delta_r + D_3z_r + D_5 \\ E_2\delta_r + E_3z_r \le E_4x_r + E_1u_r + E_5 \end{cases}$$

- Apply only $u(t)$=u_0* (discard the remaining optimal inputs)
- At time $t+1$: get new measurements, repeat optimization

MIQP FORMULATION OF MPC

(Bemporad, Morari, 1999)

$$\begin{aligned}\min_{\xi} \quad & \sum_{k=0}^{N-1} y_k' Q y_k + u_k' R u_k \\ \text{s.t.} \quad & \begin{cases} x_{k+1} = Ax_k + B_1 u_k + B_2 \delta_k + B_3 z_k + B_5 \\ y_k = Cx_k + D_1 u_k + D_2 \delta_k + D_3 z_k + D_5 \\ E_2 \delta_k + E_3 z_k \le E_4 x_k + E_1 u_k + E_5 \end{cases}\end{aligned}$$

- Optimization vector: $\xi = [u_0, \dots, u_{N-1}, \delta_0, \dots, \delta_{N-1}, z_0, \dots, z_{N-1}]$

$$\begin{aligned}\min_{\xi} \quad & \frac{1}{2}\xi' H \xi + x'(t) F' \xi + \frac{1}{2} x'(t) Y x(t) \\ \text{s.t.} \quad & G\xi \le W + Sx(t)\end{aligned}$$

Mixed Integer Quadratic Program (MIQP)

$u \in \mathbb{R}^{m_c} \times \{0,1\}^{m_b}$

$\delta \in \{0,1\}^{r_b}$

$z \in \mathbb{R}^{r_c}$

$\xi \in \mathbb{R}^{(m_c+r_c)N} \times \{0,1\}^{(m_b+r_b)N}$

vector ξ has both **real** and **binary** values

CLOSED-LOOP CONVERGENCE

Theorem Let $(x_r, u_r, \delta_r, z_r)$ be the equilibrium values corresponding to set point r. Assume $x(0)$ is such that the MPC problem is feasible at time $t=0$.

Then $\forall Q, R \succ 0,\ \forall \sigma > 0$ the closed-loop hybrid MPC loop **converges asymptotically**

$$\lim_{t\to\infty} y(t) = r \qquad \lim_{t\to\infty} x(t) = x_r$$
$$\lim_{t\to\infty} \delta(t) = \delta_r$$
$$\lim_{t\to\infty} u(t) = u_r \qquad \lim_{t\to\infty} z(t) = z_r$$

and all constraints are fulfilled at each time $t \geq 0$.

(Bemporad, Morari 1999)

Proof: Easily follows from standard Lyapunov arguments

Lyapunov asymptotic stability and **exponential stability** can be guaranteed by choosing a proper terminal cost and constraint set

(Lazar, Heemels, Weiland, Bemporad, 2006)

CONVERGENCE PROOF

Main idea: Use **value function** $V^*(x(t))$ as a **Lyapunov** function

- Let $\xi_t = [u_0^t, \dots, u_{N-1}^t, \delta_0^t, \dots, \delta_{N-1}^t, z_0^t, \dots, z_{N-1}^t]$ be the optimal sequence @t

- By construction $\bar{\xi}_{t+1} = [u_1^t, \dots, u_{N-1}^t, u_r, \delta_1^t, \dots, \delta_{N-1}^t, \delta_r, z_1^t, \dots, z_{N-1}^t, z_r]$ is feasible @$t+1$, as it satisfies all MLD constraints and terminal constraint

- The cost of $\bar{\xi}_{t+1}$ is $V^*(x(t)) - \|y(t) - r\|_Q^2 - \|u - u_r\|_R^2$
$$-\sigma \left(\|\delta(t) - \delta_r\|^2 + \|z(t) - z_r\|^2 + \|x(t) - x_r\|^2 \right) \geq V^*(x(t+1))$$

- $V^*(x(t))$ is monotonically decreasing and ≥ 0, so $\exists \lim_{t\to\infty} V^*(x(t)) \triangleq V_\infty$

- Hence $\|y(t) - r\|_Q^2$, $\|u(t) - u_r\|_R^2$ and $\|\delta(t) - \delta_r\|^2$, $\|z(t) - z_r\|^2$, $\|x(t) - x_r\|^2 \to 0$

- Since $R,Q>0$, $\lim_{t\to\infty} y(t) = r$, $\lim_{t\to\infty} u(t) = u_r$, and all other variables converge. □

Global optimum is not needed to prove convergence !

MILP FORMULATION OF MPC

(Bemporad, Borrelli, Morari, 2000)

$$\min_\xi \quad \sum_{k=0}^{N-1} \|Qy_k\|_\infty + \|Ru_k\|_\infty$$

$$\text{s.t.} \quad \begin{cases} x_{k+1} = Ax_k + B_1u_k + B_2\delta_k + B_3z_k + B_5 \\ y_k = Cx_k + D_1u_k + D_2\delta_k + D_3z_k + D_5 \\ E_2\delta_k + E_3z_k \le E_4x_k + E_1u_k + E_5 \end{cases}$$

$Q \in \mathbb{R}^{m_y \times n_y}$
$R \in \mathbb{R}^{m_u \times n_u}$

- Introduce slack variables: $\min_x |x|$ ⇒ $\min_{x,\epsilon} \ \epsilon$ s.t. $\epsilon \ge x, \ \epsilon \ge -x$

$$\begin{cases} \epsilon_k^y \ge \|Qy_k\|_\infty \\ \epsilon_k^u \ge \|Ru_k\|_\infty \end{cases} \Rightarrow \begin{cases} \epsilon_k^y \ge \pm Q^i y_k & i = 1,\dots,m_y \quad k = 0,\dots,N-1 \\ \epsilon_k^u \ge \pm R^i u_k & i = 1,\dots,m_u \quad k = 0,\dots,N-1 \end{cases}$$

Q^i = ith row of matrix Q

- Optimization vector:

$$\xi = [\epsilon_0^y, \dots, \epsilon_{N-1}^y, \epsilon_0^u, \dots, \epsilon_{N-1}^u, u_0, \dots, u_{N-1}, \delta_0, \dots, \delta_{N-1}, z_0, \dots, z_{N-1}]$$

⇒

$$\min_\xi \quad \sum_{k=0}^{N-1} \epsilon_k^y + \epsilon_k^u$$
$$\text{s.t.} \quad G\xi \le W + Sx(t)$$

Mixed Integer Linear Program (MILP)

vector ξ has both **real** and **binary** values

HYBRID MPC EXAMPLE

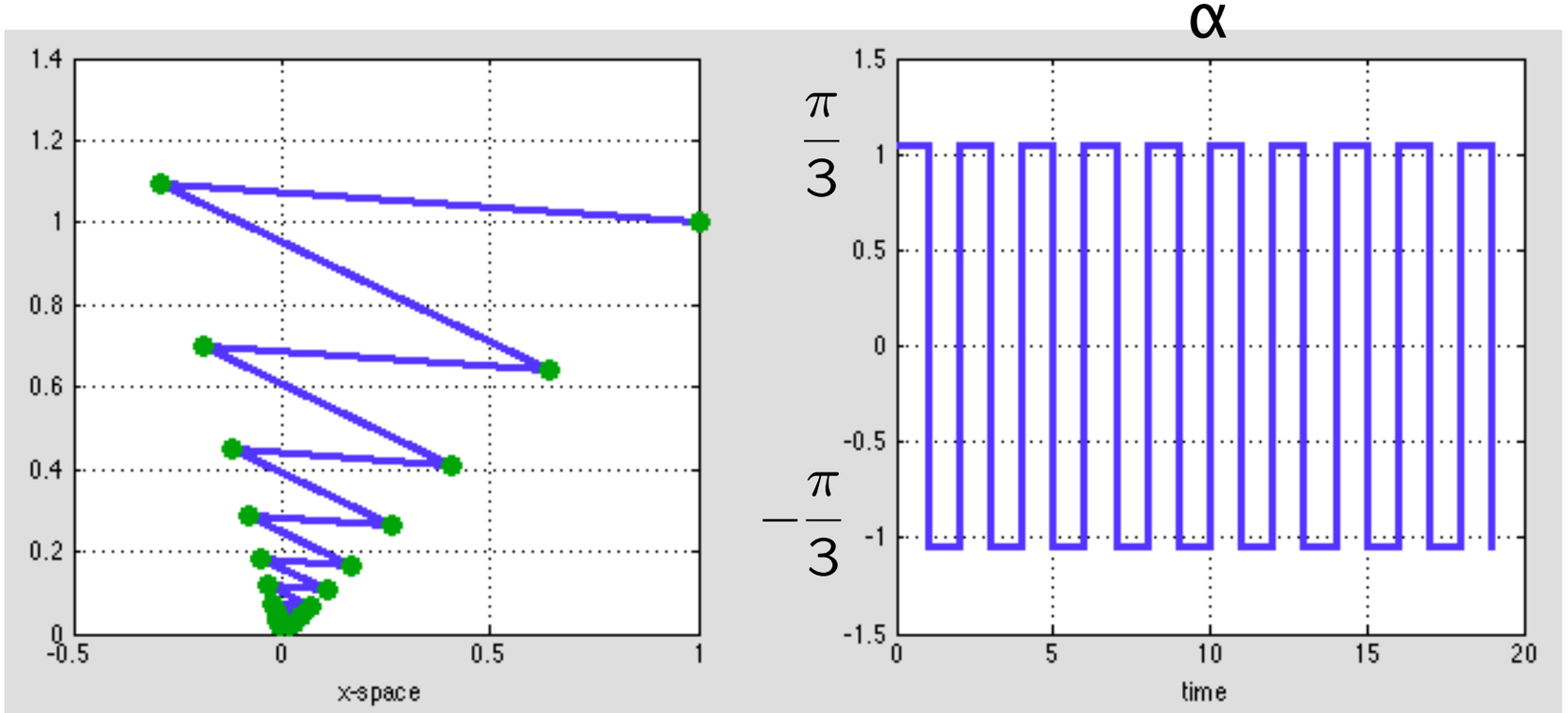
PWA system:

$$
\begin{aligned}
x(t+1) &= 0.8 \begin{bmatrix} \cos\alpha(t) & -\sin\alpha(t) \\ \sin\alpha(t) & \cos\alpha(t) \end{bmatrix} x(t) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(t) \\
y(t) &= x_2(t) \\
\alpha(t) &= \begin{cases} \frac{\pi}{3} & \text{if } x_1(t) > 0 \\ -\frac{\pi}{3} & \text{if } x_1(t) \le 0 \end{cases}
\end{aligned}
$$

constraints:

$$-1 \le u(t) \le 1$$

open-loop simulation



go to demo **/demos/hybrid/bm99sim.m**

HYBRID MPC EXAMPLE

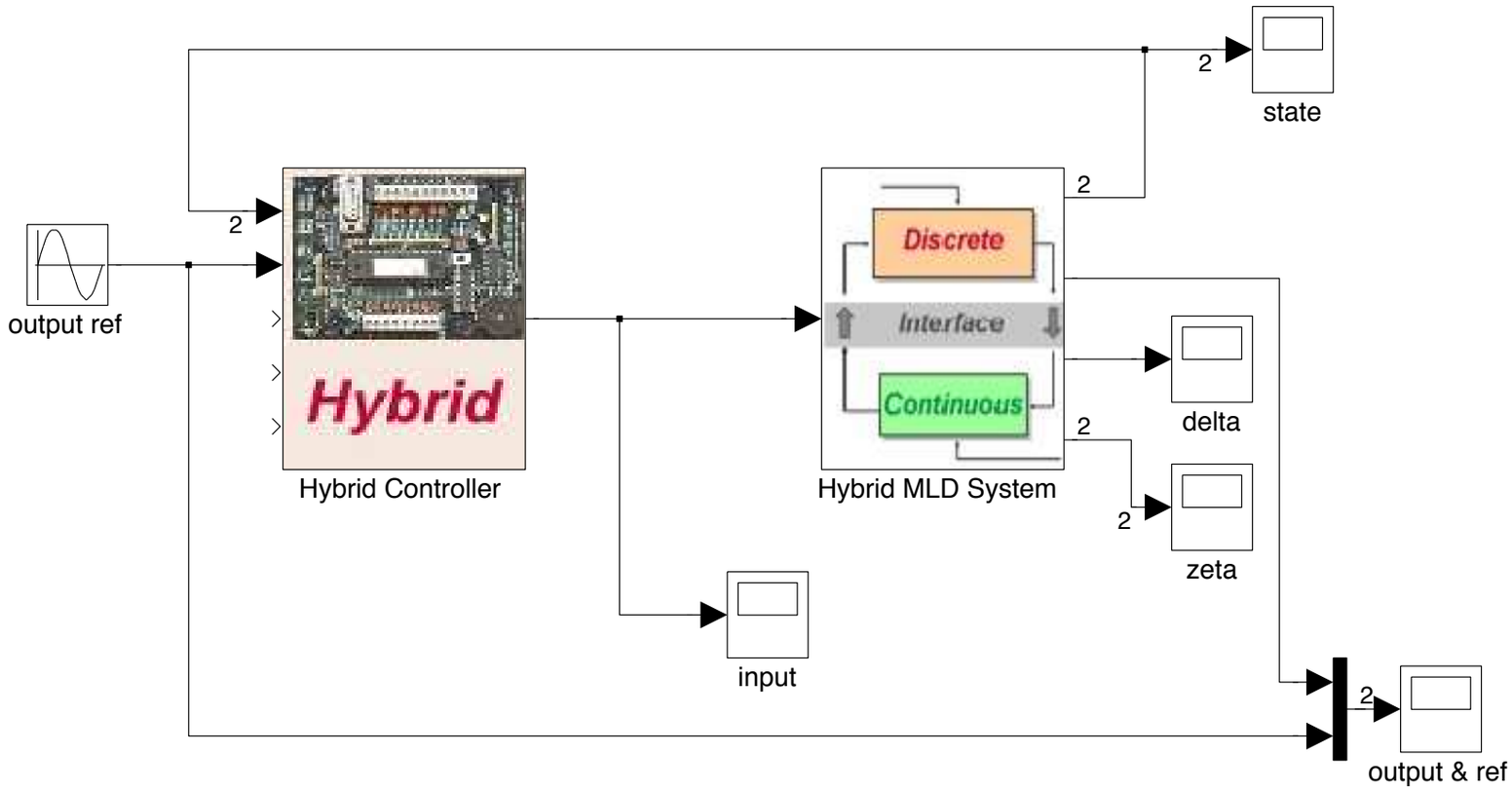
HYSDEL model

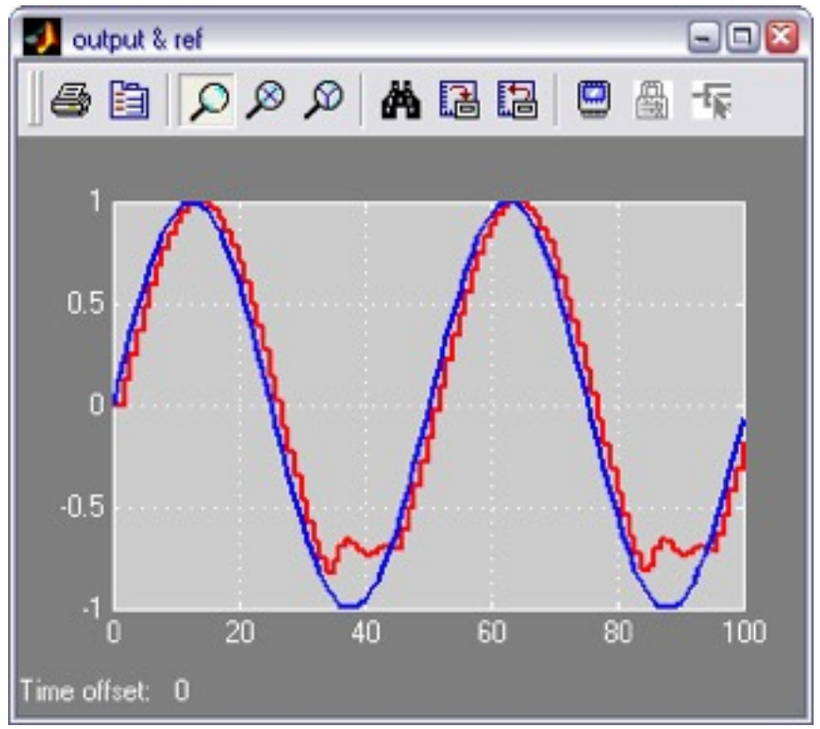
```
/* 2x2 PWA system  - Example from the paper
   A. Bemporad and M. Morari, ``Control of systems integrating logic, dynamics,
   and constraints,'' Automatica, vol. 35, no. 3, pp. 407-427, 1999.
   (C) 2003 by A. Bemporad, 2003 */

SYSTEM pwa {

INTERFACE {
            STATE { REAL x1 [-10,10];
                    REAL x2 [-10,10];}

            INPUT { REAL u [-1.1,1.1];}

            OUTPUT{ REAL y;}

            PARAMETER {
              REAL alpha = 1.0472; /* 60 deg in radiants */
              REAL C = cos(alpha);
              REAL S = sin(alpha);}
            }

IMPLEMENTATION {
            AUX { REAL z1,z2;
                  BOOL sign; }
            AD  { sign = x1<=0; }

            DA  { z1 = {IF sign THEN 0.8*(C*x1+S*x2)
                        ELSE 0.8*(C*x1-S*x2) };
                  z2 = {IF sign THEN 0.8*(-S*x1+C*x2)
                        ELSE 0.8*(S*x1+C*x2) };  }

            CONTINUOUS {x1 = z1;
                        x2 = z2+u; }

            OUTPUT { y = x2;  }
              }
}
```

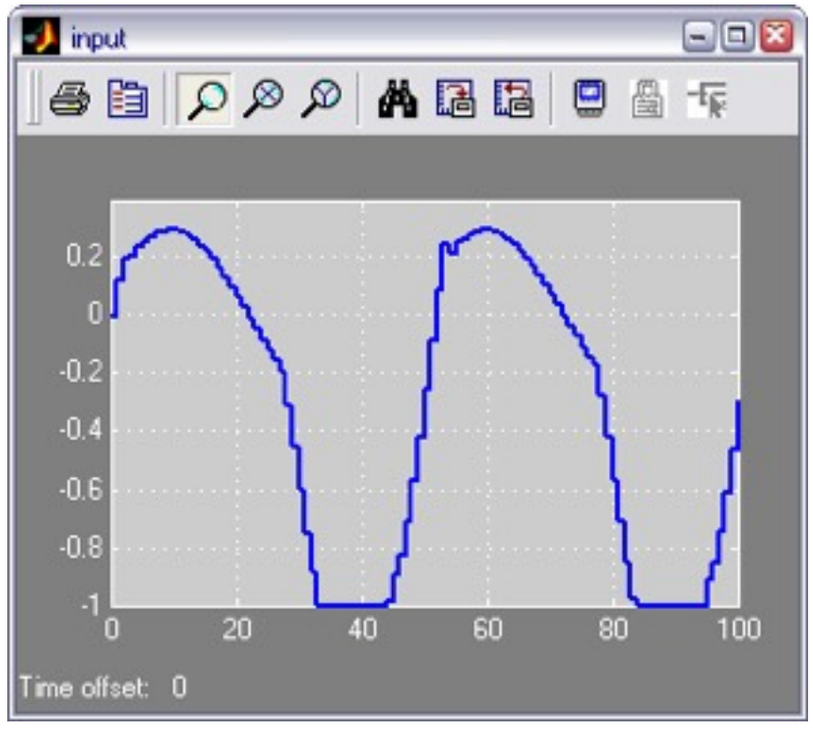
/demos/hybrid/bm99.hys

HYBRID MPC EXAMPLE

closed-loop:







performance index: $\min \sum_{k=1}^{2} |y_k - r(t)|$

Average CPU time to solve MILP: ~1 ms/step

(Macbook Pro 3GHz Intel Core I7 using GLPK)

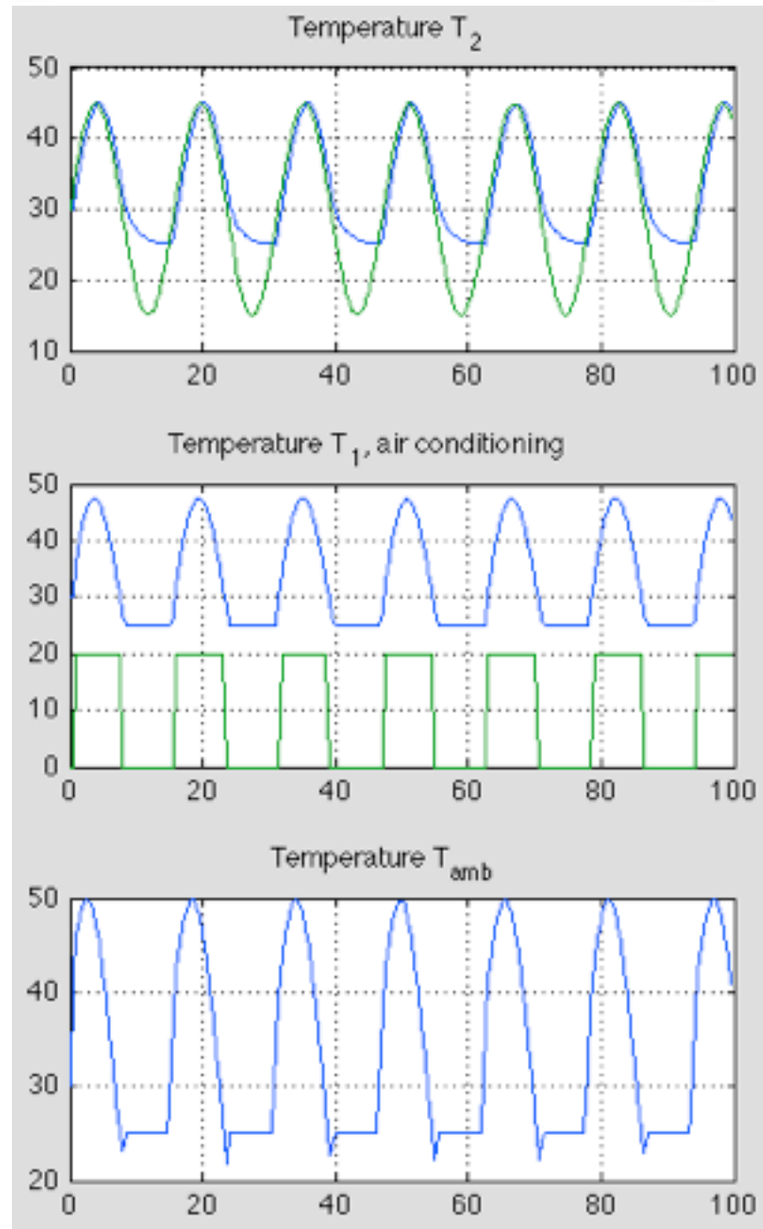
HYBRID MPC – TEMPERATURE CONTROL

```
>>refs.x=2;    % just weight state #2
>>Q.x=1;       % unit weight on state #2
>>Q.rho=Inf;   % hard constraints
>>Q.norm=Inf; % infinity norms
>>N=2;         % prediction horizon
>>limits.xmin=[25;-Inf];
```

$$\min \sum_{k=0}^{2} \|x_{2k} - r(t)\|_\infty$$
$$\text{s.t.} \begin{cases} x_{1k} \geq 25, \ k = 1,2 \\ \text{MLD model} \end{cases}$$

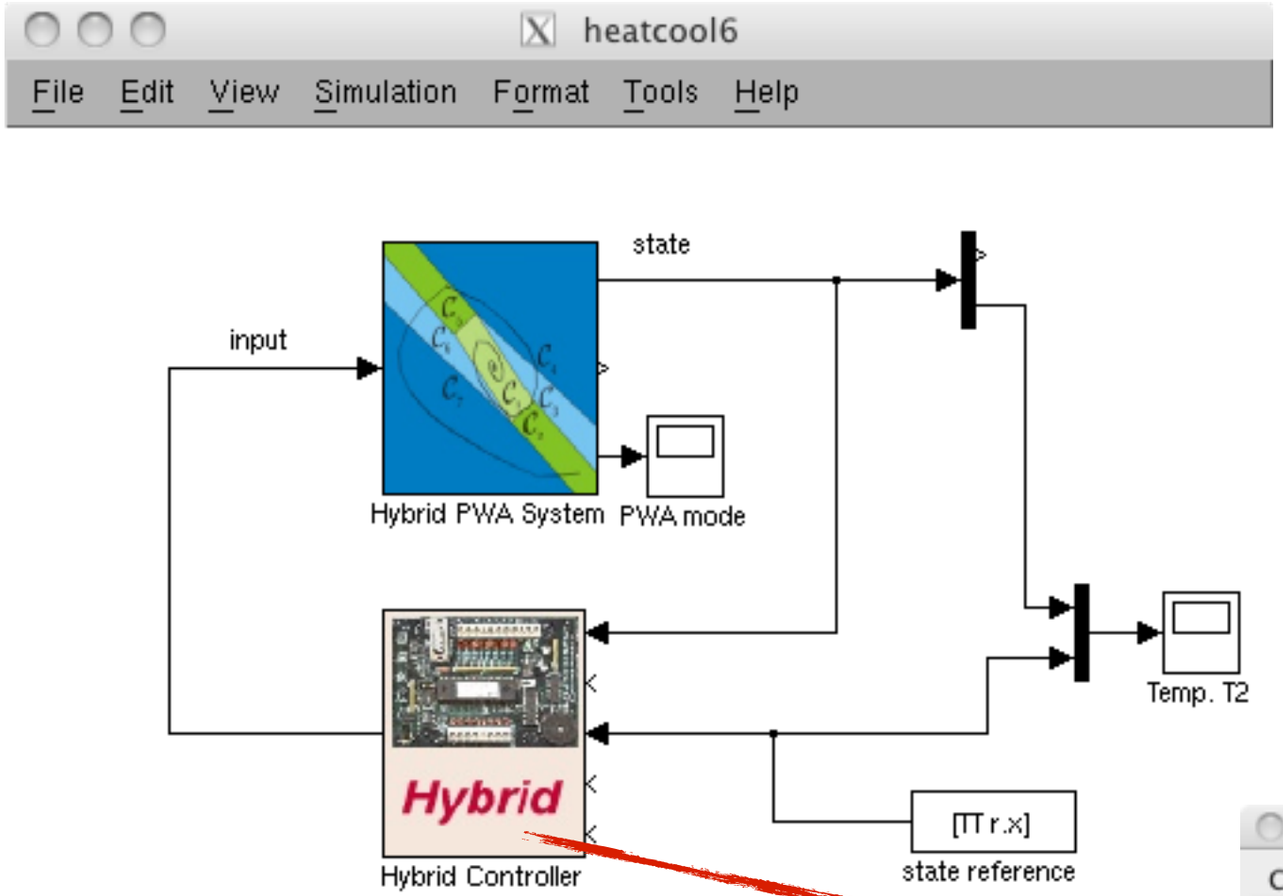
```
>>C=hybcon(S,Q,N,limits,refs);
```

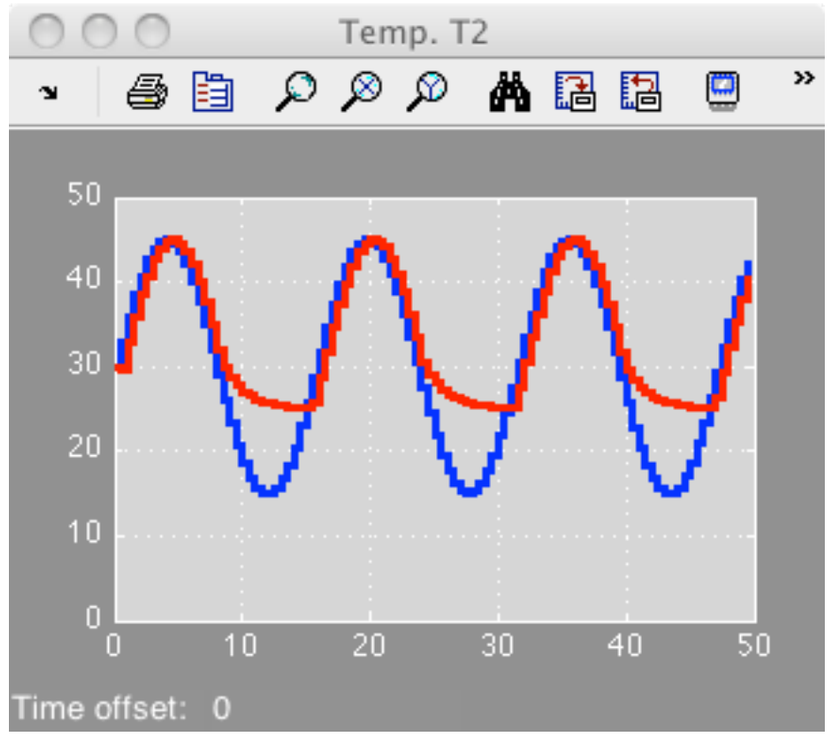
```
>> C

Hybrid controller based on MLD model S <heatcoolmodel.hys> [Inf-norm]

  2 state measurement(s)
  0 output reference(s)
  0 input reference(s)
  1 state reference(s)
  0 reference(s) on auxiliary continuous z-variables

 20 optimization variable(s) (8 continuous, 12 binary)
 46 mixed-integer linear inequalities
sampling time = 0.5, MILP solver = 'glpk'

Type "struct(C)" for more details.
>>
```

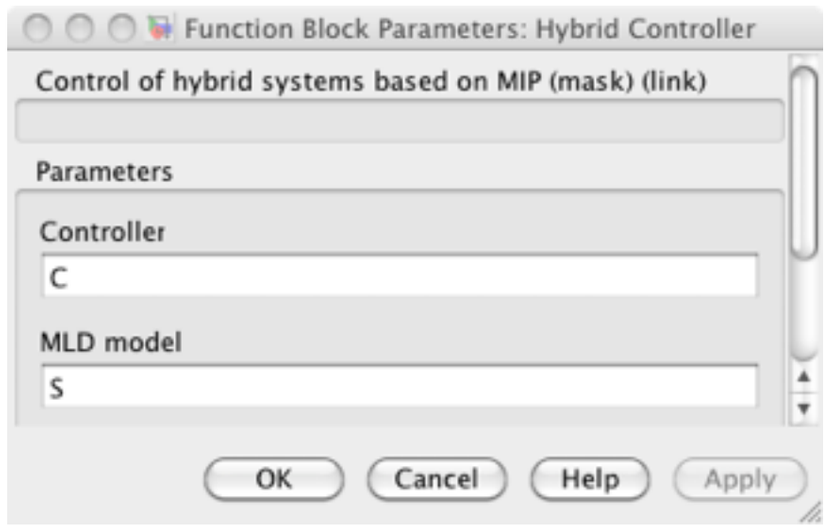



```
>>[XX,UU,DD,ZZ,TT]=sim(C,S,r,x0,Tstop);
```

HYBRID MPC – TEMPERATURE CONTROL







Average CPU time to solve MILP: ~1 ms/step

(Macbook Pro 3GHz Intel Core I7 using GLPK)

MIXED-INTEGER PROGRAM (MIP) SOLVERS

- Mixed-Integer Programming is *NP*-complete

BUT

- General purpose branch & bound / branch & cut solvers available for MILP and MIQP (CPLEX, GLPK, Xpress-MP, CBC, Gurobi, ...)

More solvers and benchmarks: http://plato.la.asu.edu/bench.html

- No need to reach global optimum (see proof of the theorem), although performance deteriorates

BRANCH & BOUND FOR MIQP (USING NNLS SOLVER FOR QP)

(Bemporad, NMPC, 2015)

- We consider a MIQP problem of the following form

$$\begin{aligned} \min_z \quad & V(z) \triangleq \frac{1}{2} z'Qz + c'z \\ \text{s.t.} \quad & \ell \le Az \le u \\ & Gz = g \\ & \bar{A}_i z \in \{\bar{\ell}_i, \bar{u}_i\}, \;\; i = 1, \ldots, q \end{aligned}$$

$$Q = Q' \succ 0$$

- Binary constraints on z are a special case:

$$\bar{\ell}_i = 0, \;\; \bar{u}_i = 1, \;\; \bar{A}_i = [0 \ldots 0 \;\; 1 \;\; 0 \ldots 0]$$

- QP algorithm based on NNLS is **extended** here to handle
  - **equality** constraints
  - **bilateral** constraints
  - **warm-starts**

  so to solve MIQP relaxations ($\bar{\ell}_i \le \bar{A}_i z \le \bar{u}_i$) very efficiently

BRANCH & BOUND FOR MIQP (USING NNLS SOLVER FOR QP)

- **Branch and bound** scheme:

QP relaxation

$$\begin{aligned}\min_z \quad & V(z) \triangleq \frac{1}{2} z'Qz + c'z \\ \text{s.t.} \quad & \ell \le Az \le u \\ & Gz = g \\ & \bar{\ell} \le \bar{A}z \le \bar{u}\end{aligned}$$

QP_0

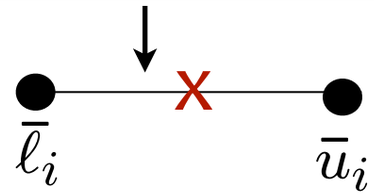
$$\bar{A}_i z \in \{\bar{\ell}_i, \bar{u}_i\}$$
$$\forall i = 1, \ldots, q$$

QP infeasible ?

- no → integer feasible ?
  - yes → MIQP solution found (lucky case)
  - no → start branching ... (typical case)
- yes → MIQP infeasible (lucky case)

BRANCH & BOUND FOR MIQP (USING NNLS SOLVER FOR QP)

- **Branching**: pick up index i such that $\bar{A}_i z$ is closest to $\frac{\bar{\ell}_i+\bar{u}_i}{2}$



- Solve two new QP problems:

QP_0

$$\begin{aligned}\min_z \quad & V(z) \triangleq \frac{1}{2}z'Qz + c'z \\ \text{s.t.} \quad & \ell \le Az \le u \\ & Gz = g \\ & \bar{\ell} \le \bar{A}z \le \bar{u}\end{aligned}$$

QP_1

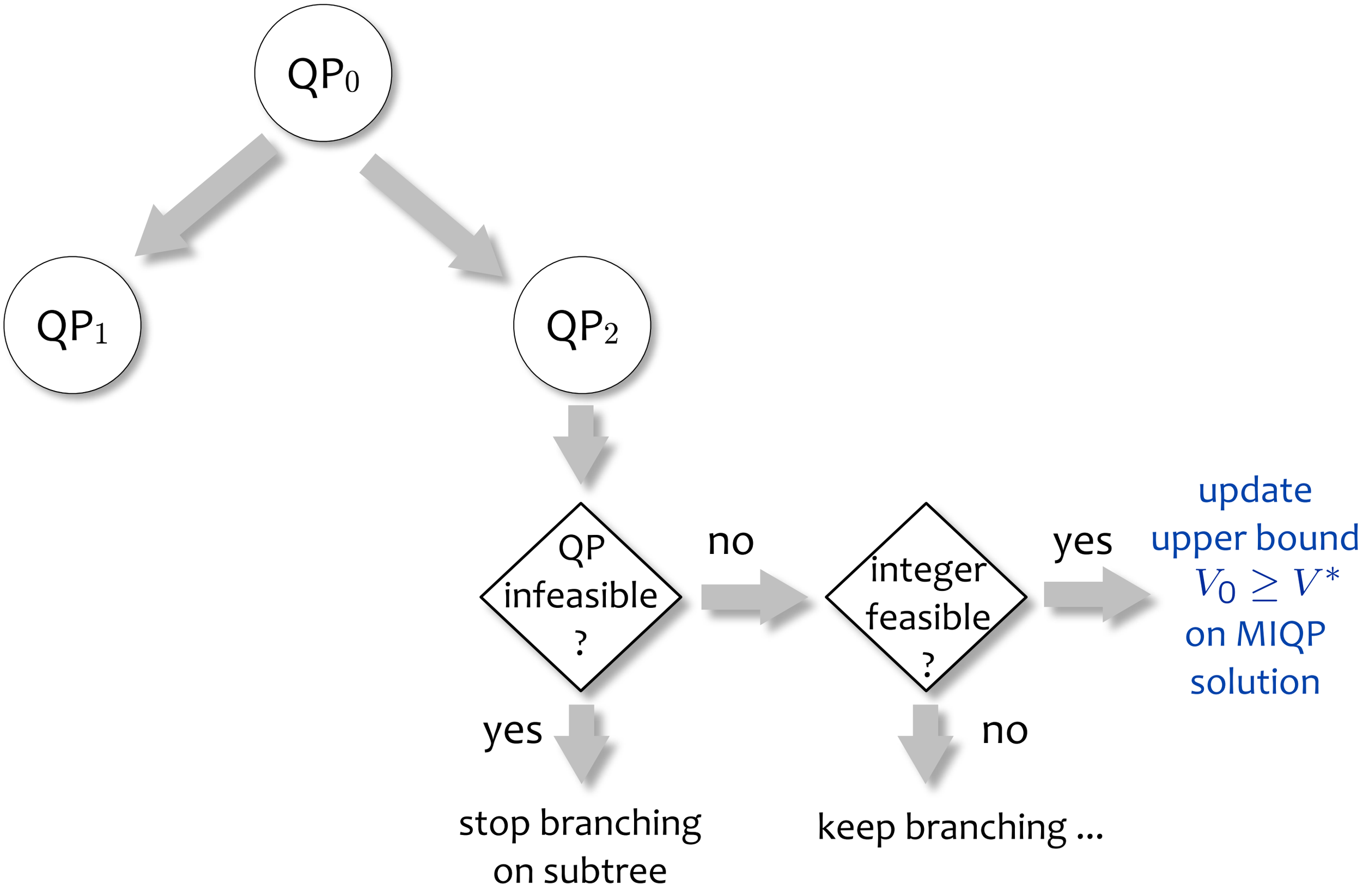
$$\begin{aligned}\min_z \quad & V(z) \triangleq \frac{1}{2}z'Qz + c'z \\ \text{s.t.} \quad & \ell \le Az \le u \\ & Gz = g \\ & \bar{A}_i z = \bar{\ell}_i \\ & \bar{\ell}_j \le \bar{A}_j z \le \bar{u}_j, \;\; j \ne i\end{aligned}$$

QP_2

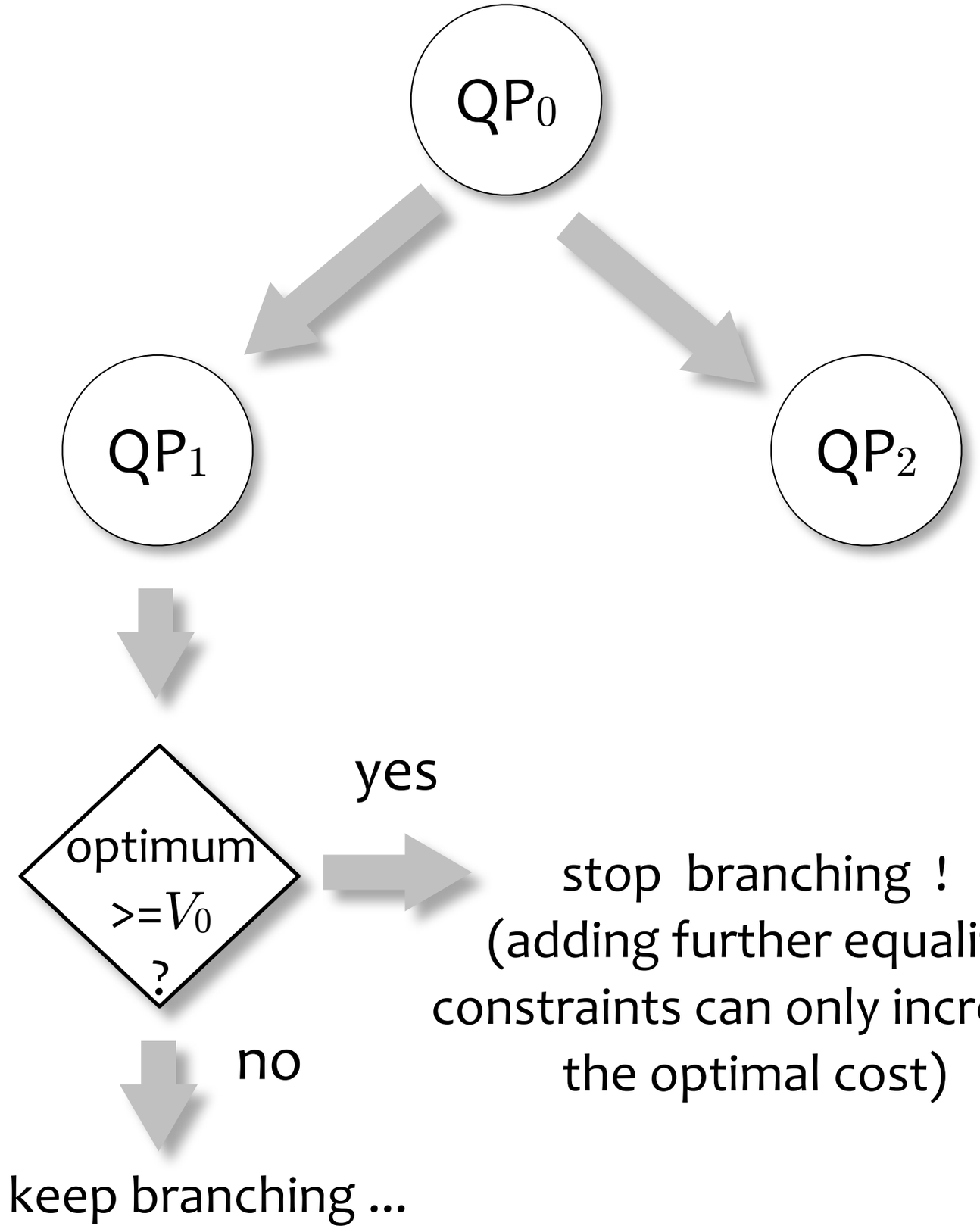
$$\begin{aligned}\min_z \quad & V(z) \triangleq \frac{1}{2}z'Qz + c'z \\ \text{s.t.} \quad & \ell \le Az \le u \\ & Gz = g \\ & \bar{A}_i z = \bar{u}_i \\ & \bar{\ell}_j \le \bar{A}_j z \le \bar{u}_j, \;\; j \ne i\end{aligned}$$

Warm start from previous solution of QP_0 heavily exploited in solving QP_1, QP_2 !

BRANCH & BOUND FOR MIQP (USING NNLS SOLVER FOR QP)



BRANCH & BOUND FOR MIQP (USING NNLS SOLVER FOR QP)



upper bound
$V_0 \geq V^*$
on MIQP
is available

BRANCH & BOUND FOR MIQP (USING NNLS SOLVER FOR QP)

- **Lower bound** V on optimal solution of QP relaxation are immediately available **during** the iterations of the QP algorithm (dual cost)

→ stop solving QP relaxation (and further branching) if $V \geq V_0$

This saves a lot of QP active set iterations !

- When no further branching is possible, either the MIQP problem is recognized infeasible or the optimal solution has been found

MIQP VIA NNLS: NUMERICAL RESULTS

(Bemporad, NMPC 2015)

- Worst-case CPU time on random **MIQP** problems:

n	m	q	NNLS_{LDL}	NNLS_{QR}	GUROBI	CPLEX
10	5	2	2.3	1.2	1.4	8.0
10	100	2	5.7	3.3	6.1	31.4
50	25	5	4.2	6.1	14.1	30.1
50	200	10	68.8	104.4	114.6	294.1
100	50	2	4.6	10.2	37.2	69.2
100	200	15	137.5	365.7	259.8	547.8
150	100	5	15.6	49.2	157.2	260.1
150	300	20	1174.4	3970.4	1296.1	2123.9

n = # variables, m = # inequality constraints, no equalities, q = # binary constraints

QP algorithm in compiled Embedded MATLAB code, B&B in interpreted MATLAB code.
CPU time measured on this Mac

NNLS_{LDL} = **recursive LDL factorization** used to solve least-square problems in QP solver
NNLS_{QR} = **recursive QR factorization** used instead (numerically more robust)

NUMERICAL RESULTS

- Worst-case CPU time (ms) on random **purely binary QP** problems:

n	m	q	NNLS_{LDL}	NNLS_{QR}	GUROBI	CPLEX
2	10	2	5.1	4.0	0.7	8.4
4	20	4	8.9	4.3	4.5	16.7
8	40	8	19.2	18.0	37.1	14.7
12	60	12	59.7	57.8	82.3	47.9
20	100	20	483.5	457.7	566.8	99.6
25	250	25	110.4	93.3	1054.4	169.4
30	150	30	1645.4	1415.8	2156.2	184.5

- Worst-case CPU time (ms) on a **hybrid MPC** problem (Bemporad, Morari, 1999)

N = prediction horizon

MIQP regularized to make Q strictly >0 (--> solution difference is negligible)

$n = 40,\ m = 160,\ q = 10$

N	NNLS_{LDL}	NNLS_{QR}	GUROBI	CPLEX
2	2.2	2.3	1.2	3.0
3	3.4	3.9	2.0	6.5
4	5.0	6.5	2.6	8.1
5	7.6	9.8	3.7	9.0
6	12.3	17.7	4.3	11.0
7	20.5	30.5	5.8	13.1
8	28.9	47.1	7.3	17.3
9	38.8	62.5	9.5	18.9
10	55.4	98.2	10.9	22.4

FAST GRADIENT PROJECTION FOR MIQP

(Naik, Bemporad, 2016)

- MIQP problem

special case:
binary constraints $z_i \in \{0,1\}$ →

$$\begin{aligned} \min_z \quad & V(z) \triangleq \frac{1}{2} z'Qz + c'z \\ \text{s.t.} \quad & \ell \le Az \le u \\ & A_{eq} z = b_{eq} \\ & \bar{A}_i z \in \{\bar{\ell}_i, \bar{u}_i\}, \;\; i = 1, \ldots, p \end{aligned}$$

- Use branch & bound, relax binary constraints to $\ell_i \le A_i z \le \bar{u}_i$

- Only projection changes from one QP relaxation to another:

constraint is relaxed	$\bar{A}_i z \le \bar{u}_i \;\rightarrow\; y^i_{k+1} = \max\left\{y^i_k + s^i_k, 0\right\}$	$y_i \geqq 0$
constraint is fixed	$\bar{A}_i z = \bar{u}_i \;\rightarrow\; y^i_{k+1} = y^i_k + s^i_k$	$y_i \lessgtr 0$
constraint is ignored	$\bar{A}_i z = \bar{\ell}_i \;\rightarrow\; y^{k+1}_i = 0$	$y_i = 0$

FAST GRADIENT PROJECTION FOR MIQP

(Naik, Bemporad, 2016)

- Same dual QP matrices, preconditioning only computed at root node
- Warm-start exploited, dual cost used to stop QP relaxations earlier
- Criterion based on Farkas lemma to detect QP infeasibility
- Numerical results (time in ms):

n	m	p	q	miqpGPAD	GUROBI
10	100	2	2	15.6	6.56
50	25	5	3	3.44	8.74
50	150	10	5	63.22	46.25
100	50	2	5	6.22	26.24
100	200	15	5	164.06	188.42
150	100	5	5	31.26	88.13
150	200	20	5	258.80	274.06
200	50	15	6	35.08	144.38

HEURISTIC ADMM METHOD FOR (SUBOPTIMAL) MIQP

(Takapoui, Moehle, Boyd, Bemporad, 2017)

- MIQP problem:

$$\begin{aligned} \min \quad & \tfrac{1}{2}x'Qx + q'x \\ \text{s.t.} \quad & \ell \le Ax \le u \\ & A_i x \in \{\ell_i, u_i\}, \;\; i \in I \end{aligned}$$

- ADMM iterations:

quantization →

$$\begin{aligned} x^{k+1} &= -(Q+\rho A^T A)^{-1}(\rho A^T(y^k - z^k) + q) \\ z^{k+1} &= \min\{\max\{Ax^{k+1} + y^k, \ell\}, u\} \\ z_i^{k+1} &= \begin{cases} \ell_i & \text{if } \; z_i^{k+1} < \frac{\ell_i + u_i}{2} \\ u_i & \text{if } \; z_i^{k+1} \ge \frac{\ell_i + u_i}{2}, \;\; i \in I \end{cases} \\ y^{k+1} &= y^k + Ax^{k+1} - z^{k+1} \end{aligned}$$

- Iterations converge to a (local) solution

- Similar idea also applicable to fast gradient methods (Naik, Bemporad, 2016)

ADMM METHOD FOR MIQP

(Takapoui, Moehle, Boyd, Bemporad, 2017)

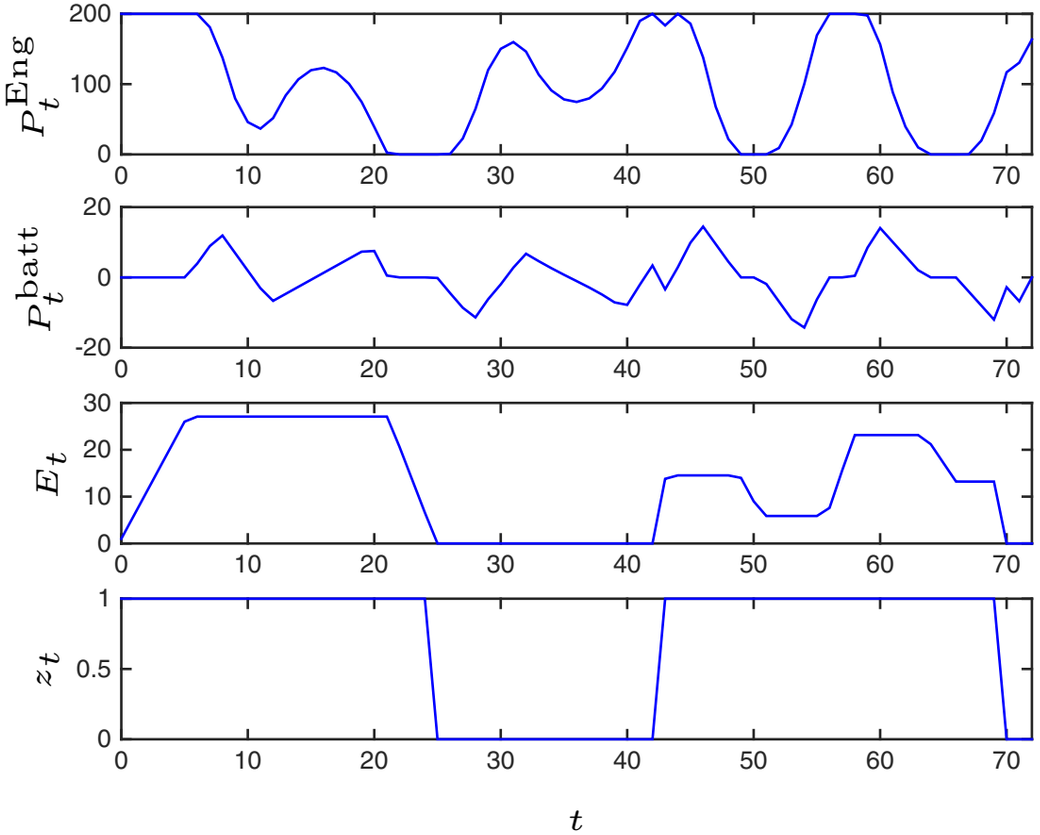
- Example: parallel Hybrid Electric Vehicle control problem

engine power

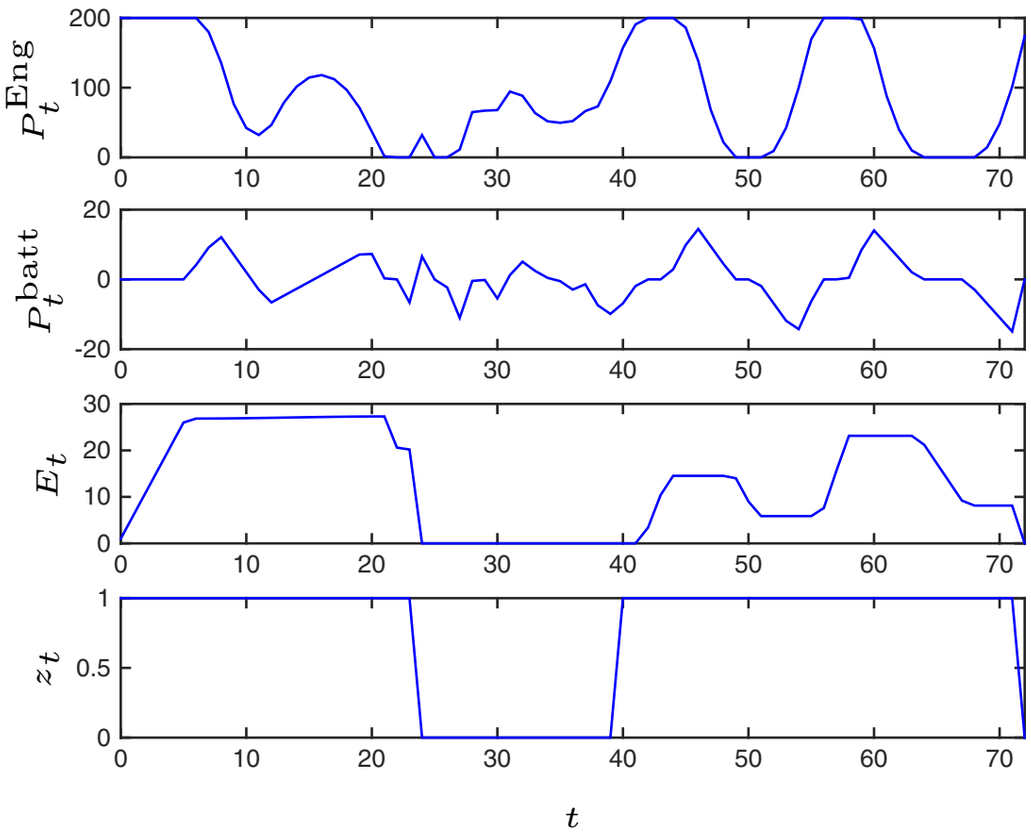
electrical power

energy stored in battery

engine on/off



optimal solution

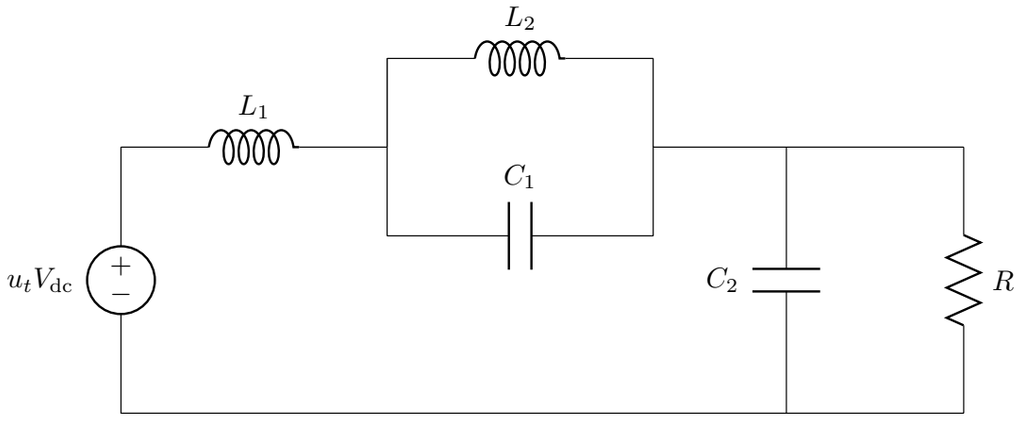


ADMM solution

ADMM METHOD FOR (SUBOPTIMAL) MIQP

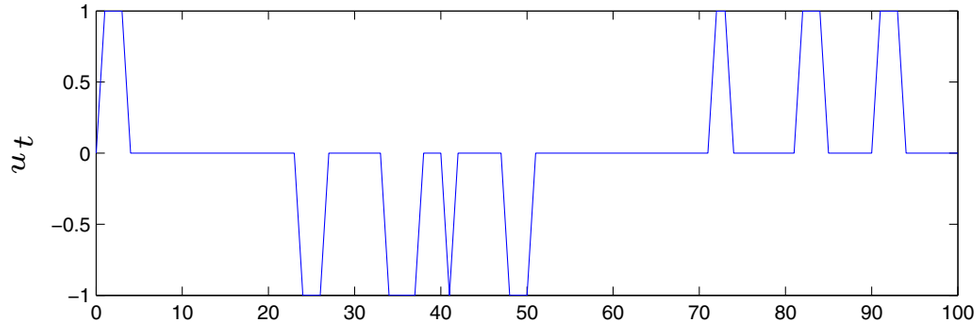
- Example: power converter control

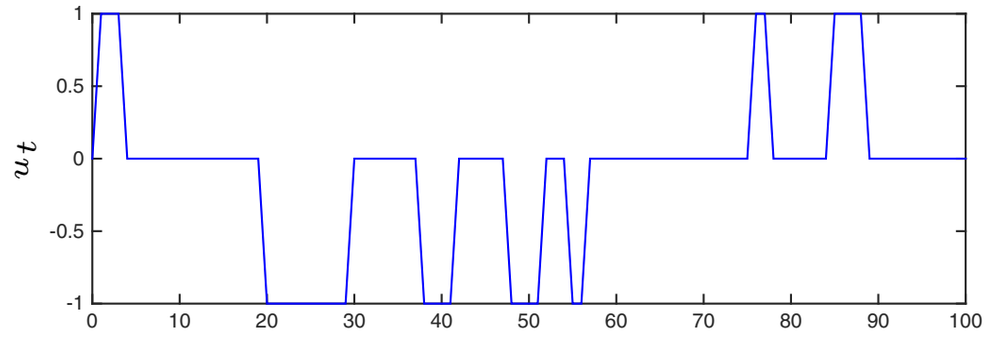
(Takapoui, Moehle, Boyd, Bemporad, 2017)



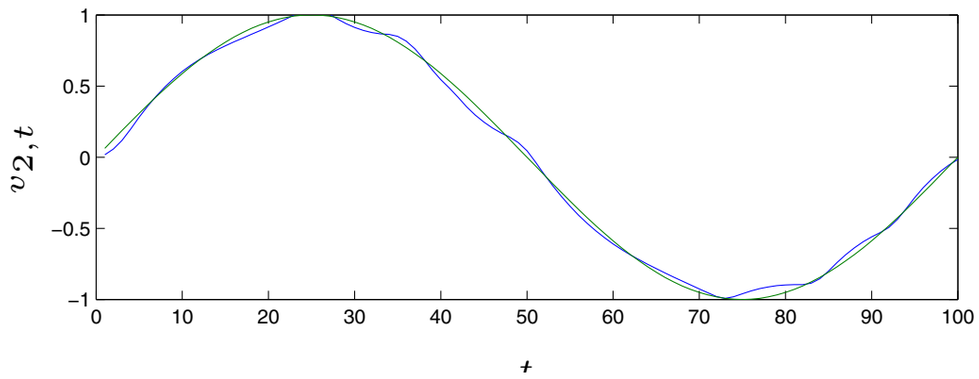
$$
\begin{aligned}
\text{minimize} \quad & \textstyle\sum_{t=0}^{T} (v_{2,t} - v_{\rm des})^2 + \lambda |u_t - u_{t-1}| \\
\text{subject to} \quad & \xi_{t+1} = G\xi_t + H u_t \\
& \xi_0 = \xi_T \\
& u_0 = u_T \\
& u_t \in \{-1, 0, 1\},
\end{aligned}
$$

input voltage sign u_t

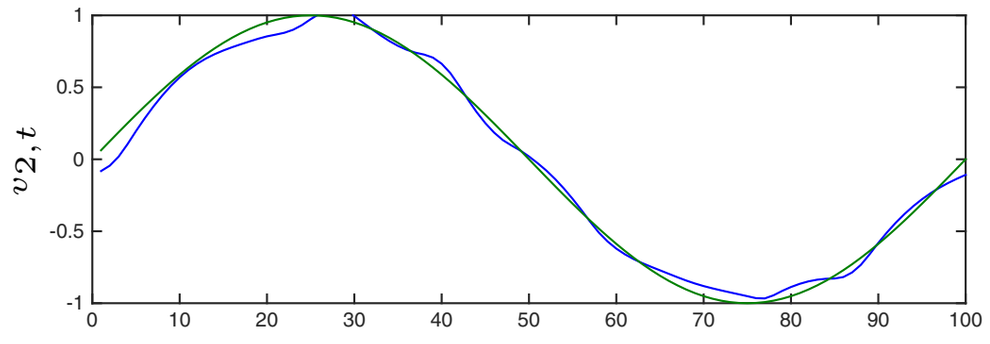




output voltage v_2



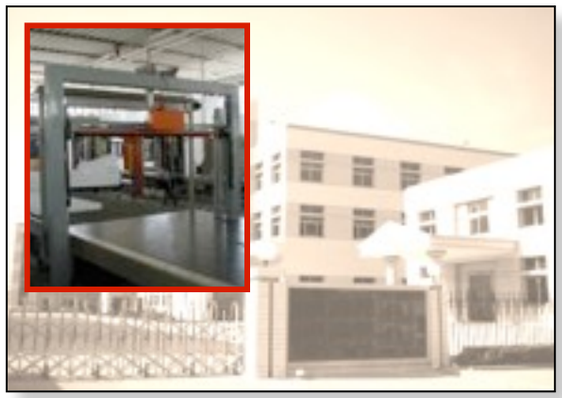
optimal solution



ADMM solution

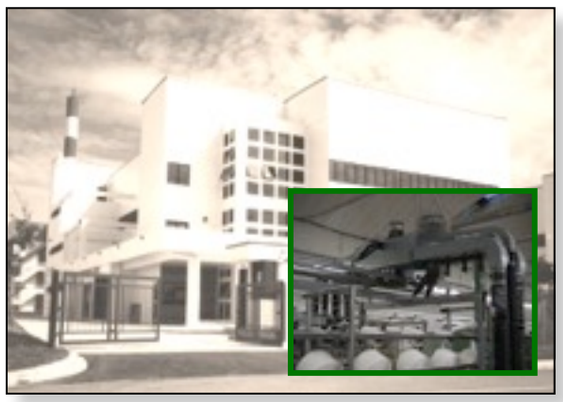
A SIMPLE EXAMPLE IN SUPPLY CHAIN MANAGEMENT

manufacturer A



manufacturer B

manufacturer C



$U_{A11}(k)$

$U_{A21}(k)$

$U_{B11}(k)$

$U_{B12}(k)$

$U_{B21}(k)$

$U_{B22}(k)$

$U_{C12}(k)$

$U_{C22}(k)$

inventory 1

$x_{11}(k), x_{12}(k)$

inventory 2

$x_{21}(k), x_{22}(k)$

$u_{11}(k)$

$u_{12}(k)$

$u_{22}(k)$

$u_{21}(k)$

retailer 1

$y_1(k)$

$y_2(k)$

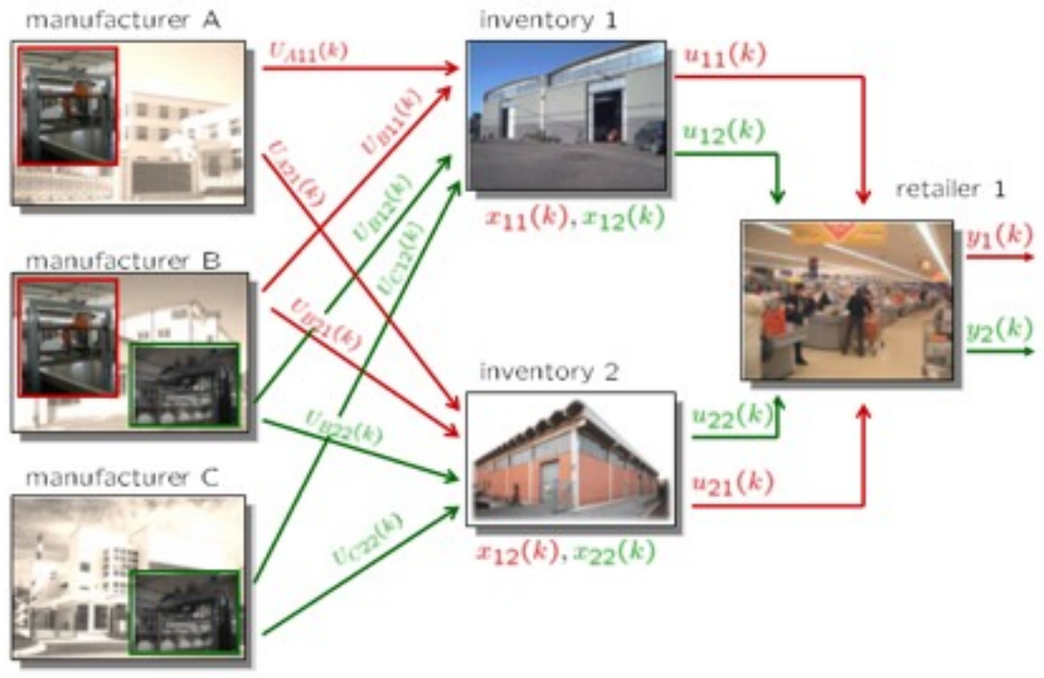
SYSTEM VARIABLES

- continuous states:

  $x_{ij}(k)$ = amount of j hold in inventory i at time k
  (i=1,2, j=1,2)

- continuous outputs:

  $y_j(k)$ = amount of j sold at time k
  (j=1,2)



- continuous inputs:

  $u_{ij}(k)$ = amount of j taken from inventory i at time k
  (i=1,2, j=1,2)

- binary inputs:

  $U_{Xij}(k)$ = 1 if manufacturer X produces and send j to inventory i at time k

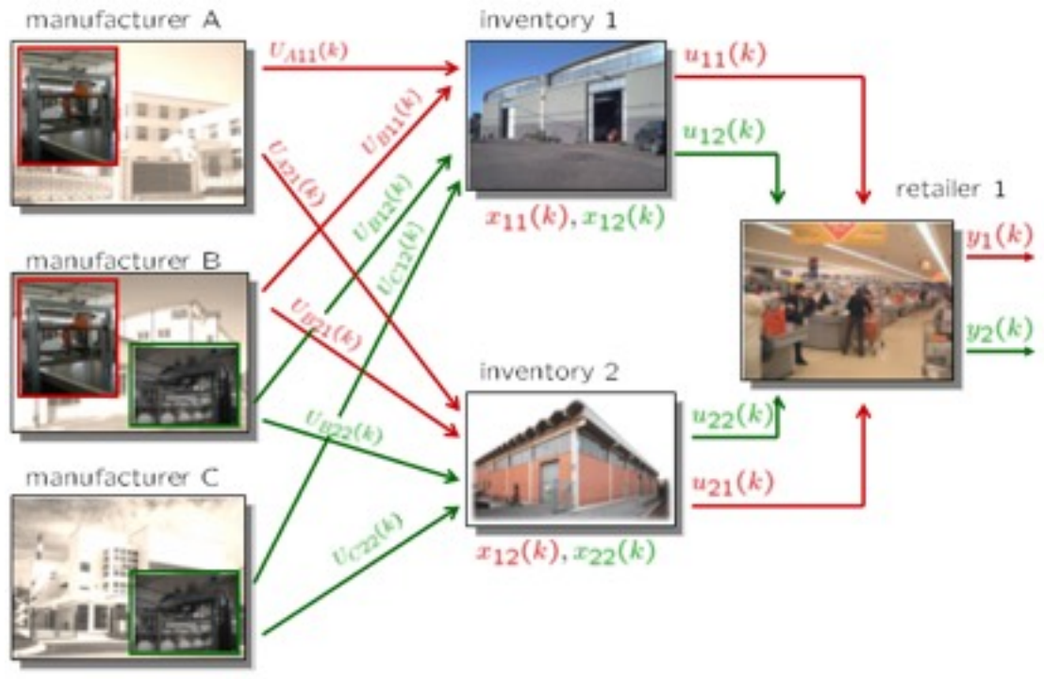
CONSTRAINTS

- Max capacity of inventory i:

$$0 \leq \sum_j x_{ij}(k) \leq x_{Mi}$$

Numerical values:

x_{M1}=10, x_{M2}=10



- Max transportation from inventories:

$$0 \leq u_{ij}(k) \leq u_M$$

- A product can only be sent to one inventory:

$UA_{11}(k)$ and $UA_{21}(k)$ cannot be both $=1$
$UB_{11}(k)$ and $UB_{21}(k)$ cannot be both $=1$
$UB_{12}(k)$ and $UB_{22}(k)$ cannot be both $=1$
$UC_{12}(k)$ and $UC_{22}(k)$ cannot be both $=1$

- A manufacturer can only produce one type of product at one time:

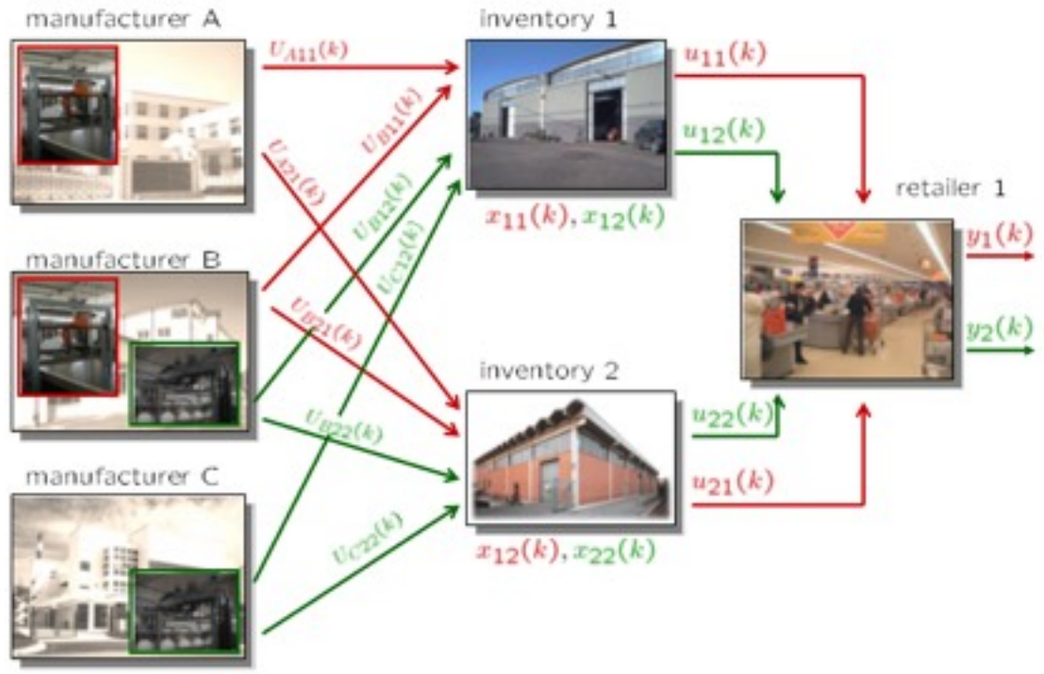
$[UB_{11}(k)$ or $UB_{21}(k)=1]$, $[UB_{12}(k)$ or $UB_{22}(k)=1]$ cannot be both true

DYNAMICS

P_{A1}, P_{B1}, P_{B2}, P_{C2} = amount of product of type 1 (2) produced by A (B,C) in one time interval

Numerical values:

P_{A1}=4, P_{B1}=6, P_{B2}=7, P_{C2}=3



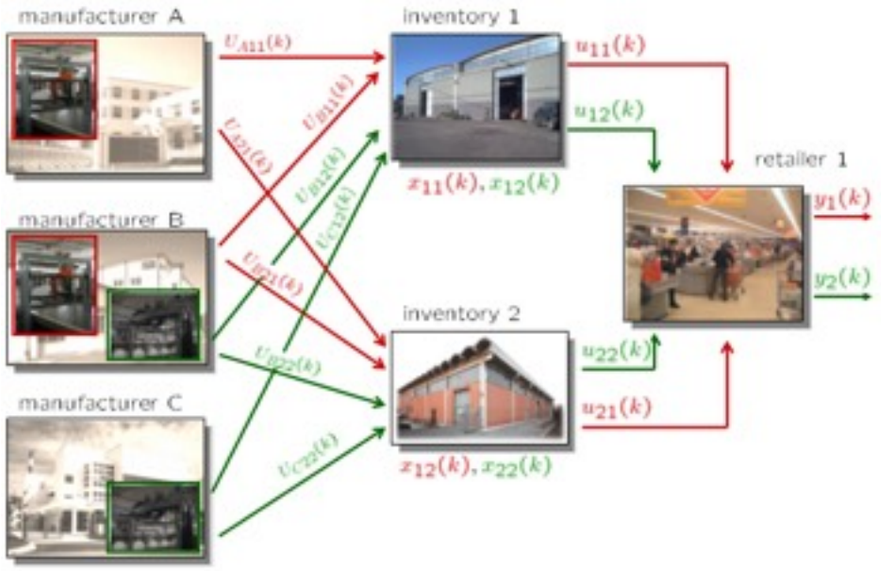
- Level of inventories:

$$\begin{cases} x_{11}(k+1) &= x_{11}(k) + P_{A1}U_{A11}(k) + P_{B1}U_{B11}(k) - u_{11}(k) \\ x_{12}(k+1) &= x_{12}(k) + P_{B2}U_{B12}(k) + P_{C2}U_{C12}(k) - u_{12}(k) \\ x_{21}(k+1) &= x_{21}(k) + P_{A1}U_{A21}(k) + P_{B1}U_{B21}(k) - u_{21}(k) \\ x_{22}(k+1) &= x_{22}(k) + P_{B2}U_{B22}(k) + P_{C2}U_{C22}(k) - u_{22}(k) \end{cases}$$

- Retailer: $\begin{cases} y_1 &= u_{11} + u_{21} \\ y_2 &= u_{12} + u_{22} \end{cases}$

HYBRID DYNAMICAL MODEL

```
SYSTEM supply_chain{
INTERFACE {
        STATE { REAL x11    [0,10];
                REAL x12    [0,10];
                REAL x21    [0,10];
                REAL x22    [0,10]; }

        INPUT { REAL u11 [0,10];
                REAL u12 [0,10];
                REAL u21 [0,10];
                REAL u22 [0,10];
                BOOL UA11,UA21,UB11,UB12,UB21,UB22,UC12,UC22; }

        OUTPUT {REAL y1,y2;}

        PARAMETER { REAL PA1,PB1,PB2,PC2,xM1,xM2;}
}
IMPLEMENTATION {

        AUX { REAL zA11, zB11, zB12, zC12, zA21, zB21, zB22, zC22;}


        DA { zA11 = {IF UA11 THEN PA1 ELSE 0};
             zB11 = {IF UB11 THEN PB1 ELSE 0};
             zB12 = {IF UB12 THEN PB2 ELSE 0};
             zC12 = {IF UC12 THEN PC2 ELSE 0};
             zA21 = {IF UA21 THEN PA1 ELSE 0};
             zB21 = {IF UB21 THEN PB1 ELSE 0};
             zB22 = {IF UB22 THEN PB2 ELSE 0};
             zC22 = {IF UC22 THEN PC2 ELSE 0}; }
```

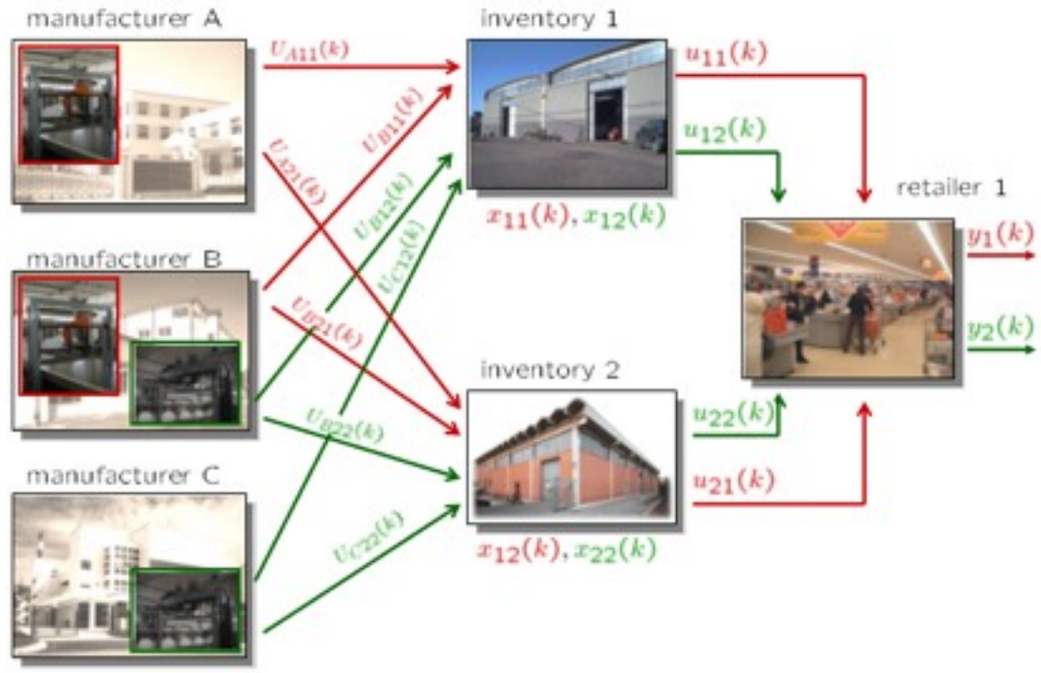



```
        CONTINUOUS {x11 = x11 + zA11 + zB11 - u11;
                    x12 = x12 + zB12 + zC12 - u12;
                    x21 = x21 + zA21 + zB21 - u21;
                    x22 = x22 + zB22 + zC22 - u22;  }

        OUTPUT {    y1 = u11 + u21;
                    y2 = u12 + u22; }

        MUST {  ~(UA11 & UA21);
            ~(UC12 & UC22);
            ~((UB11 | UB21) & (UB12 | UB22));
             ~(UB11 & UB21);
            ~(UB12 & UB22);
            x11+x12 <= xM1;
            x11+x12 >=0;
            x21+x22 <= xM2;
            x21+x22 >=0; }
} }
```

/demos/hybrid/supply_chain.m

OBJECTIVES

- Meet customer demand as much as possible: $y_1 \approx r_1, y_2 \approx r_2$



- Minimize transportation costs

- Fulfill all constraints

PERFORMANCE SPECS

penalty on demand tracking error

$$\min \sum_{k=0}^{N-1} 10(|y_{1,k} - r_1(t)| + |y_{2,k} - r_2(t)| +$$

cost for shipping from inv.#1 to market

$$4(|u_{11,k}| + |u_{12,k}|) +$$

$$2(|u_{21,k}| + |u_{22,k}|) +$$

cost for shipping from inv.#2 to market

$$1(|U_{A11,k}| + |U_{A21,k}|) +$$

$$4(|U_{B11,k}| + |U_{B12,k}| + U_{B21,k}| + |U_{B22,k}|) +$$

cost from A to inventories

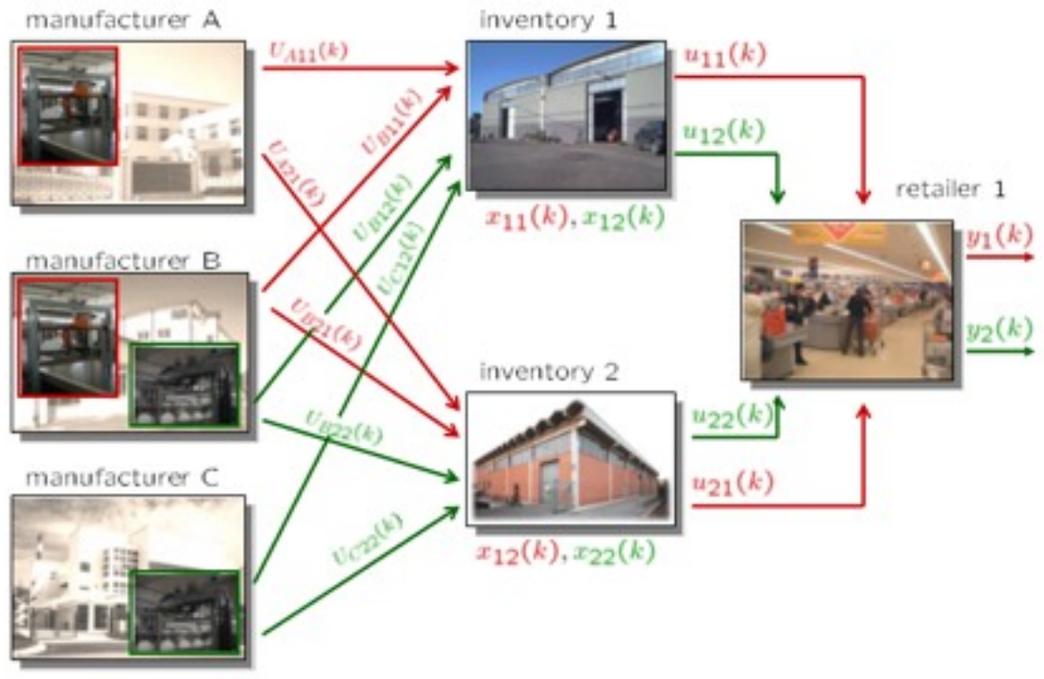
$$10(|U_{C12,k}| + |U_{C22,k}|)$$

cost from B to inventories

cost from C to inventories

SIMULATION SETUP

```
>>refs.y=[1 2];        % weights output2 #1,#2
>>Q.y=diag([10 10]);   % output weights
…
>>Q.norm=Inf;          % infinity norms
>>N=2;                 % optimization horizon
>>limits.umin=umin;    % constraints
>>limits.umax=umax;
>>limits.xmin=xmin;    % xij(k)>=0
>>limits.xmax=xmax;    % xij(k)<=xMi (redundant)
```



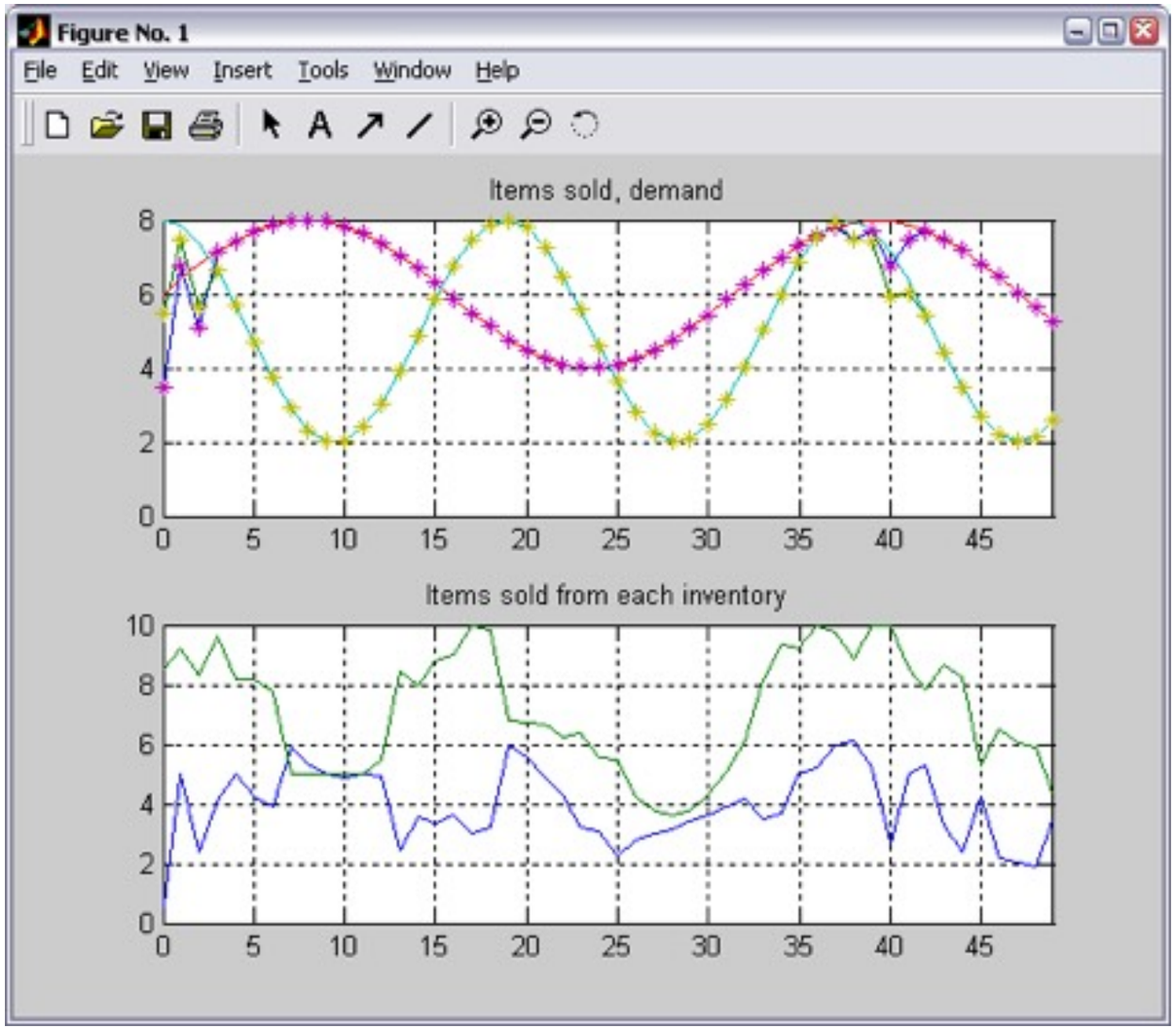

```
>>C=hybcon(S,Q,N,limits,refs);
```

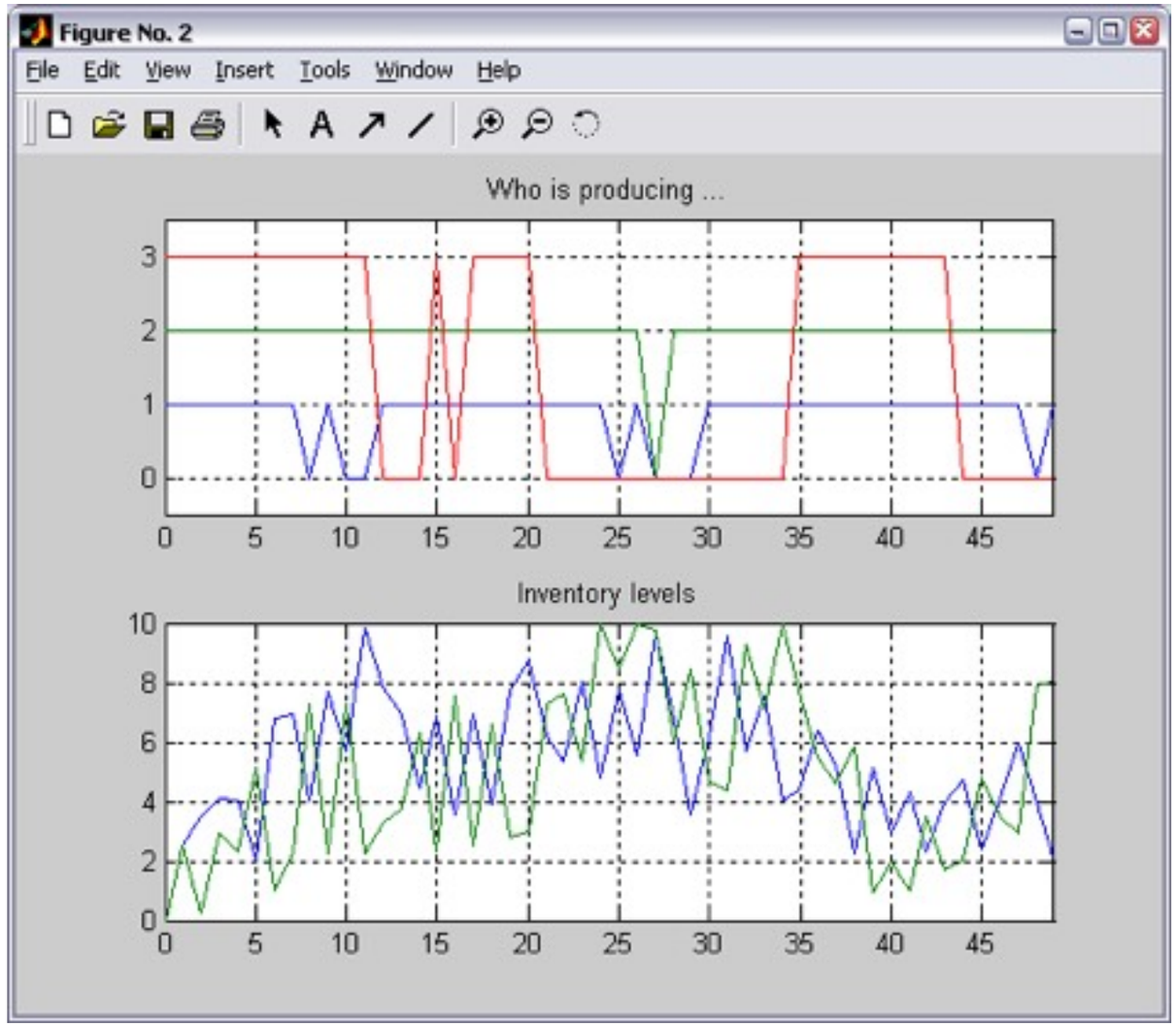
```
>> C

Hybrid controller based on MLD model S <supply_chain.hys> [Inf-norm]

  4 state measurement(s)
  2 output reference(s)
 12 input reference(s)
  0 state reference(s)
  0 reference(s) on auxiliary continuous z-variables

 44 optimization variable(s) (28 continuous, 16 binary)
176 mixed-integer linear inequalities
sampling time = 1, MILP solver = 'glpk'

Type "struct(C)" for more details.

>>
```

SIMULATION RESULTS

```
>>x0=[0;0;0;0];                          % Initial condition
>>r.y=[6+2*sin((0:Tstop-1)'/5)           % Reference trajectories
       5+3*cos((0:Tstop-1)'/3)];
```

```
>>[XX,UU,DD,ZZ,TT]=sim(C,S,r,x0,Tstop);
```

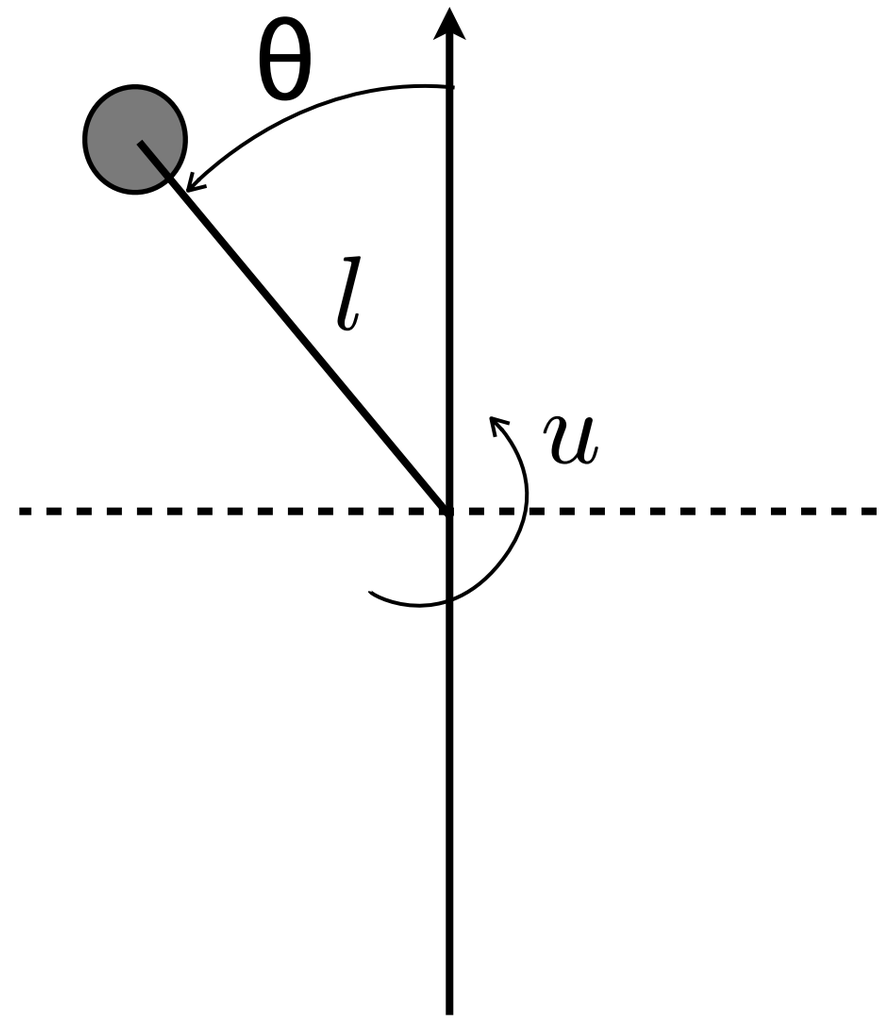




CPU time: ≈ 13 ms per time step using GLPK (9 ms using CPLEX) on this machine

HYBRID MPC OF AN INVERTED PENDULUM

- **Goal:** swing the pendulum up



- Non-convex input constraint

$$u \in [-\tau_{\max}, -\tau_{\min}] \cup \{0\} \cup [\tau_{\min}, \tau_{\max}]$$

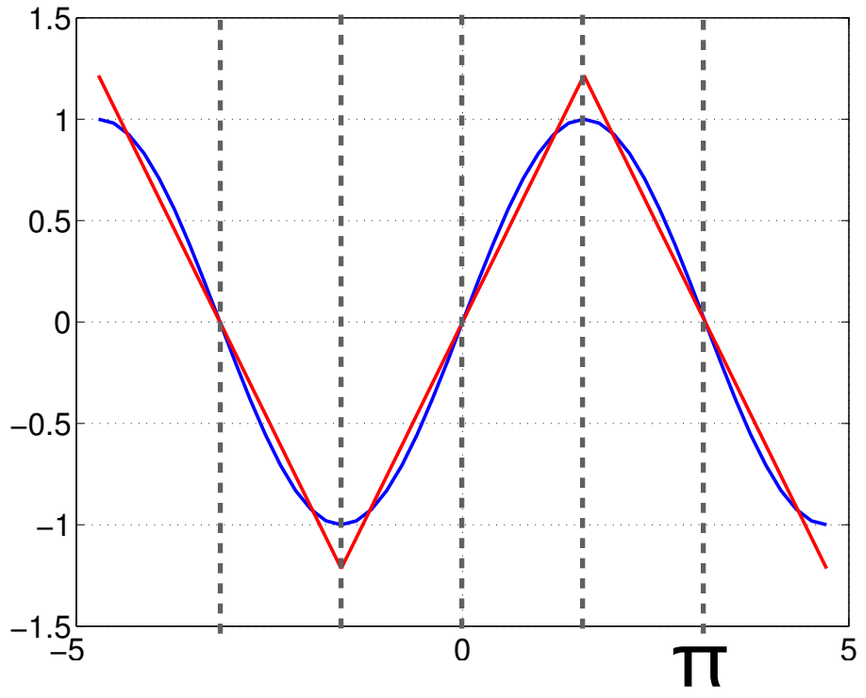
- Nonlinear dynamical model

$$l^2 M\ddot{\theta} = Mgl \sin\theta - \beta\dot{\theta} + u$$

INVERTED PENDULUM: NONLINEARITY

- Approximate $\sin(\theta)$ as the piecewise linear function

$$\sin\theta \approx s \triangleq \begin{cases} -\alpha\theta - \gamma & \text{if} \quad \theta \le -\frac{\pi}{2} \\ \alpha\theta & \text{if} \quad |\theta| \le \frac{\pi}{2} \\ -\alpha\theta + \gamma & \text{if} \quad \theta \ge \frac{\pi}{2} \end{cases}$$



- Get optimal values for α and γ by minimizing fit error

$$\min_{\alpha} \int_0^{\frac{\pi}{2}} (\alpha\theta - \sin(\theta))^2 d\theta$$

$$= \frac{\theta}{2} - \frac{1}{2}\cos\theta\sin\theta - 2\alpha\sin\theta + \frac{1}{3}\alpha^2\theta^3 + 2\alpha\theta\cos\theta \Big|_0^{\frac{\pi}{2}} = \frac{1}{24}\pi^3\alpha^2 - 2\alpha + \frac{\pi}{4}$$

- Zeroing the derivative with respect to α gives $\alpha = \frac{24}{\pi^3}$
- Requiring $s=0$ for $\theta=\pi$ gives $\gamma = \frac{24}{\pi^2}$

INVERTED PENDULUM: NONLINEARITY

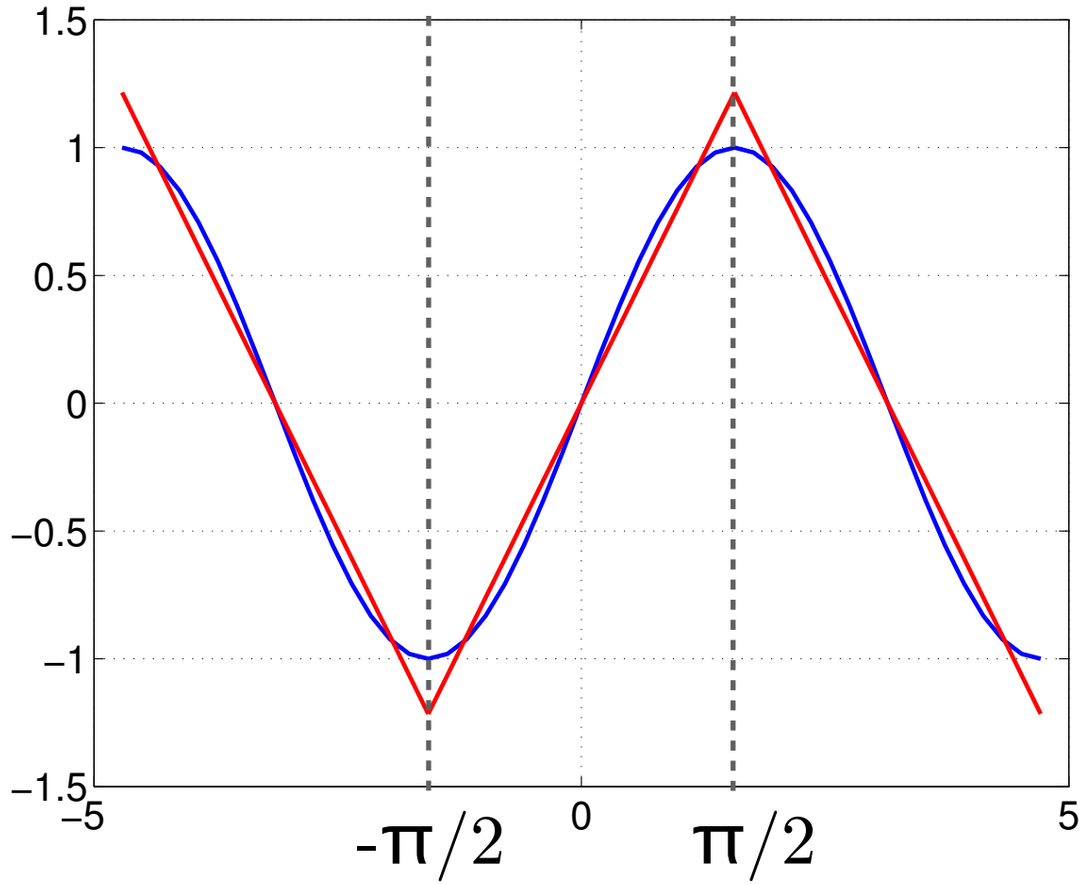
- Introduce the event variables

$$[\delta_3 = 1] \quad \leftrightarrow \quad [\theta \leq -\tfrac{\pi}{2}]$$

$$[\delta_4 = 1] \quad \leftrightarrow \quad [\theta \geq \tfrac{\pi}{2}]$$

along with the logic constraint

$$[\delta_4 = 1] \rightarrow [\delta_3 = 0]$$



- Set $s = \alpha\theta + s_3 + s_4$, with

$$s_3 = \begin{cases} -2\alpha\theta - \gamma & \text{if } \delta_3 = 1 \\ 0 & \text{otherwise} \end{cases}$$

$$s_4 = \begin{cases} -2\alpha\theta + \gamma & \text{if } \delta_4 = 1 \\ 0 & \text{otherwise} \end{cases}$$

INVERTED PENDULUM: NON-CONVEX CONSTRAINT

- To model the constraint $u \in [-\tau_{\max}, -\tau_{\min}] \cup \{0\} \cup [\tau_{\min}, \tau_{\max}]$ we introduce the auxiliary variable

$$\tau_A = \begin{cases} u & \text{if } -\tau_{\min} \le u \le \tau_{\min} \\ 0 & \text{otherwise} \end{cases}$$

and let $u - \tau_A$ be the torque acting on the pendulum, with

$$u \in [-\tau_{\max}, \tau_{\max}]$$

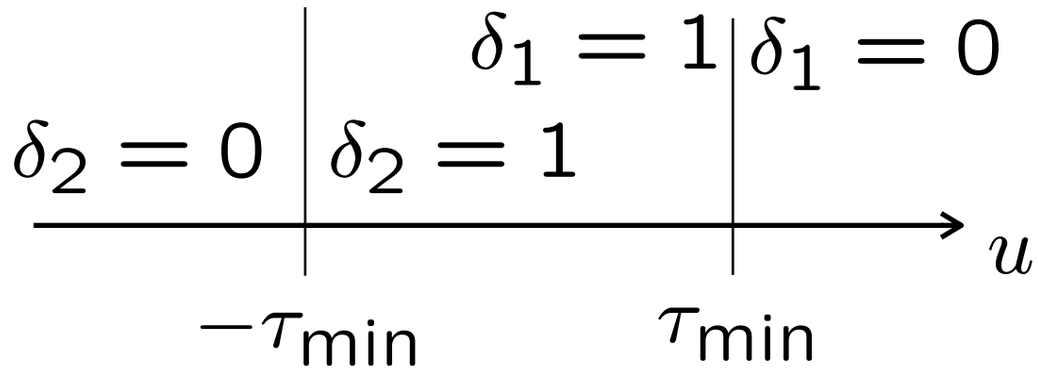
- As the input u has no effect on the dynamics for $u \in [-\tau_{\min}, \tau_{\min}]$, if u is penalized in the MPC cost function the numerical solver will not choose values in that range

INVERTED PENDULUM: NON-CONVEX CONSTRAINT

- Introduce new event variables

$$[\delta_1 = 1] \quad \leftrightarrow \quad [u \leq \tau_{\min}]$$

$$[\delta_2 = 1] \quad \leftrightarrow \quad [u \geq -\tau_{\min}]$$



along with the logic constraint $[\delta_1 = 0] \rightarrow [\delta_2 = 1]$

and set $\tau_A = \begin{cases} u & \text{if } [\delta_1 = 1] \wedge [\delta_2 = 1] \\ 0 & \text{otherwise} \end{cases}$

so that $u - \tau_A$ is zero in for $u \in [-\tau_{\min}, \tau_{\min}]$

INVERTED PENDULUM: DYNAMICS

- Transform into linear model

$$x \triangleq \begin{bmatrix} \theta \\ \dot{\theta} \end{bmatrix}, \; y \triangleq \theta$$

$$\begin{aligned} \frac{d}{dt}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} &= A_c \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + B_c \begin{bmatrix} s_3 + s_4 \\ u - \tau_A \end{bmatrix} \\ y &= \begin{bmatrix} 1 & 0 \end{bmatrix} x \end{aligned}$$

$$A_c \triangleq \begin{bmatrix} 0 & 1 \\ \frac{g}{l}\alpha & -\frac{\beta}{l^2 M} \end{bmatrix}, \; B_c \triangleq \begin{bmatrix} 0 & 0 \\ \frac{g}{l} & \frac{1}{l^2 M} \end{bmatrix}$$

- Discretize in time with sample time T_s=50 ms

$$\begin{aligned} \begin{bmatrix} x_1(k+1) \\ x_2(k+1) \end{bmatrix} &= A \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix} + B \begin{bmatrix} s_3(k) + s_4(k) \\ u(k) - \tau_A(k) \end{bmatrix} \\ y(k) &= \begin{bmatrix} 1 & 0 \end{bmatrix} x(k) \end{aligned}$$

$$A \triangleq e^{T_s A_c}, \; B \triangleq \int_0^{T_s} e^{t A_c} B_c dt$$

INVERTED PENDULUM: HYSDEL MODEL

```
/* Hybrid model of a pendulum

   (C) 2012 by A. Bemporad, April 2012 */

SYSTEM hyb_pendulum {

INTERFACE {
  STATE {
    REAL th     [-2*pi,2*pi];
    REAL thdot [-20,20];
                      }
  INPUT {
    REAL u [-11,11];
                      }
  OUTPUT{
    REAL y;
                      }
  PARAMETER {
    REAL tau_min,alpha,gamma;
    REAL a11,a12,a21,a22,b11,b12,b21,b22;
  }
}

IMPLEMENTATION {
   AUX {
     REAL tauA,s3,s4;
     BOOL d1,d2,d3,d4;
   }
   AD {
     d1 = u<=tau_min;
     d2 = u>=-tau_min;
     d3 = th <= -0.5*pi;
     d4 = th >= 0.5*pi;
   }
   DA  {
     tauA = {IF d1 & d2 THEN u ELSE 0};
     s3 = {IF d3 THEN -2*alpha*th-gamma ELSE 0};
     s4 = {IF d4 THEN -2*alpha*th+gamma ELSE 0};
   }

   CONTINUOUS {
     th    = a11*th+a12*thdot+b11*(s3+s4)+b12*(u-tauA);
     thdot = a21*th+a22*thdot+b21*(s3+s4)+b22*(u-tauA);
   }

   OUTPUT {
     y = th;
   }

   MUST {
     d4->~d3;
     ~d1->d2;
   }
}
}
```

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

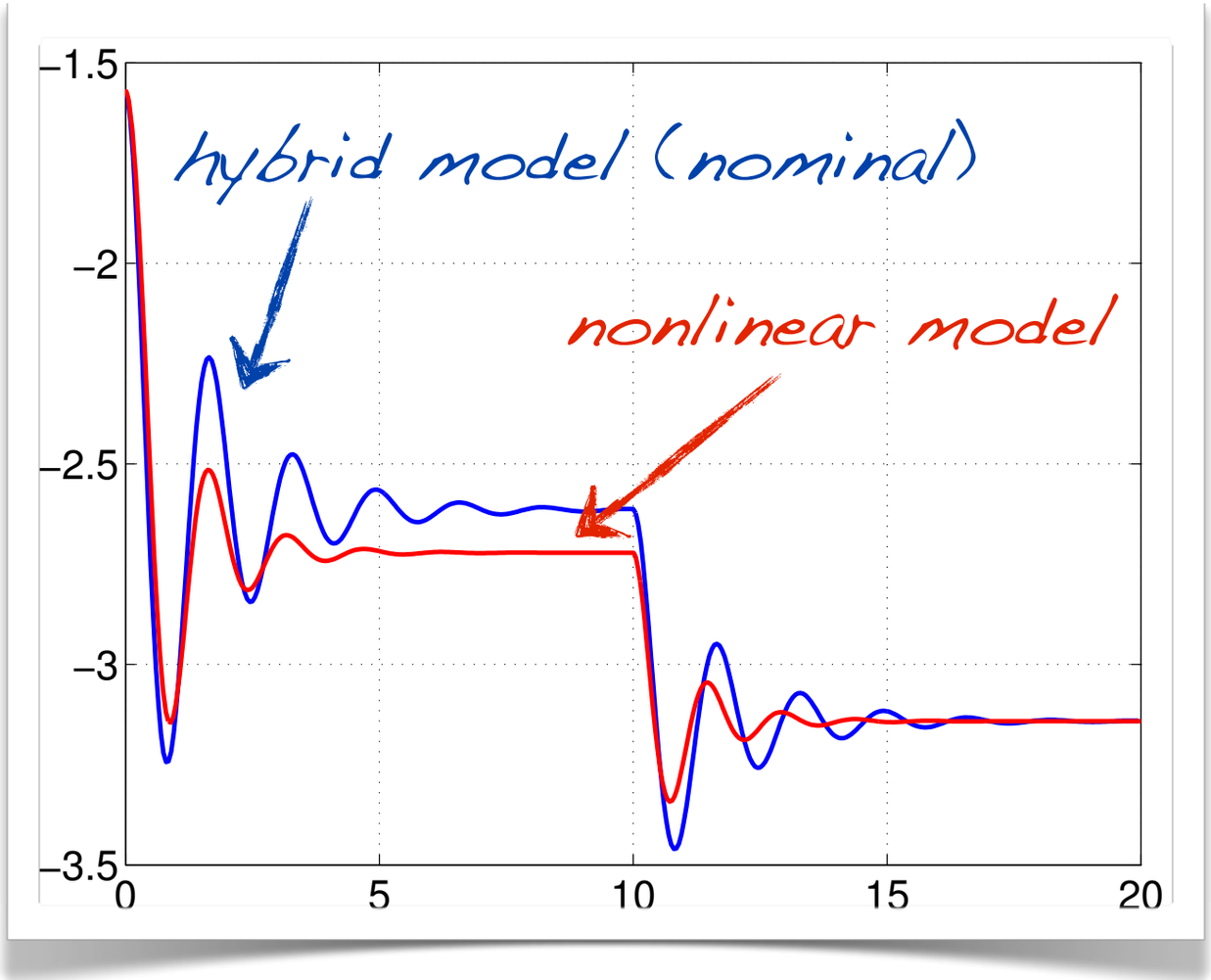
go to demo **/demos/hybrid/pendulum_init.m**

INVERTED PENDULUM: MODEL VALIDATION

- Open-loop simulation from initial condition $\theta(0) = -\frac{\pi}{2},\ \dot{\theta}(0) = 0$
- Input torque excitation $$u(t) = \begin{cases} 2 \text{ Nm} & \text{if } 0 \le t \le 10 \text{ s} \\ 0 & \text{otherwise} \end{cases}$$

```
>>u0=2;
>>U=[2*ones(200,1);zeros(200,1)];
>>x0=[-pi/2;0];
```

```
>>[X,T,D,Z,Y]=sim(S,x0,U);
```



INVERTED PENDULUM: MPC DESIGN

- MPC cost function $\sum_{k=0}^{4} |y_k - r(t))| + |0.01u_k|$

```
>>refs.y=1;
>>refs.u=1;
>>Q.y=1;
>>Q.y=0.01;
>>Q.rho=Inf;
>>Q.norm=Inf;
>>N=5;
```

- MPC constraints $u \in [-\tau_{\max}, \tau_{\max}]$

```
>>limits.umin=-10;
>>limits.umax=10;
```

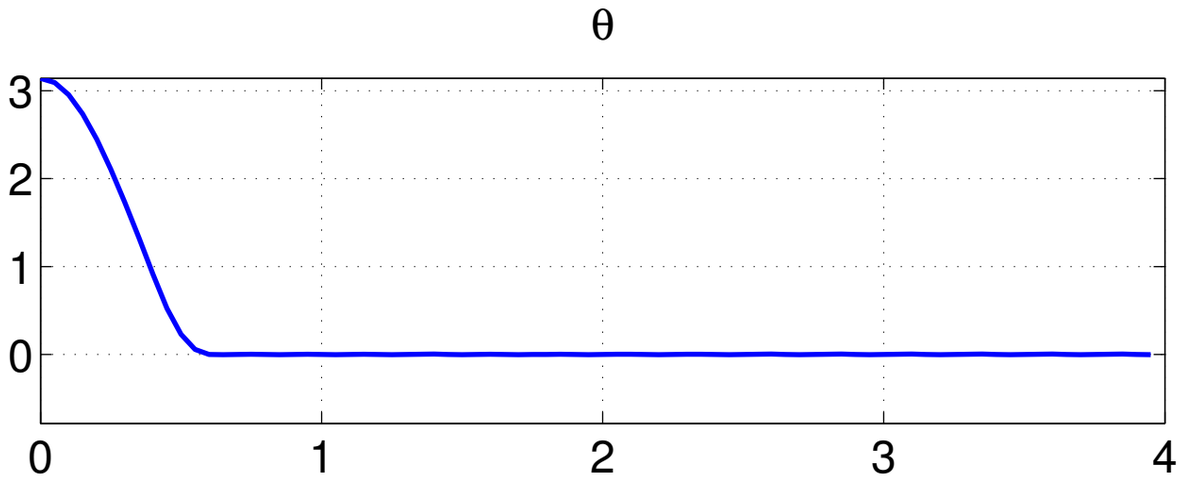
```
>>C=hybcon(S,Q,N,limits,refs);
```

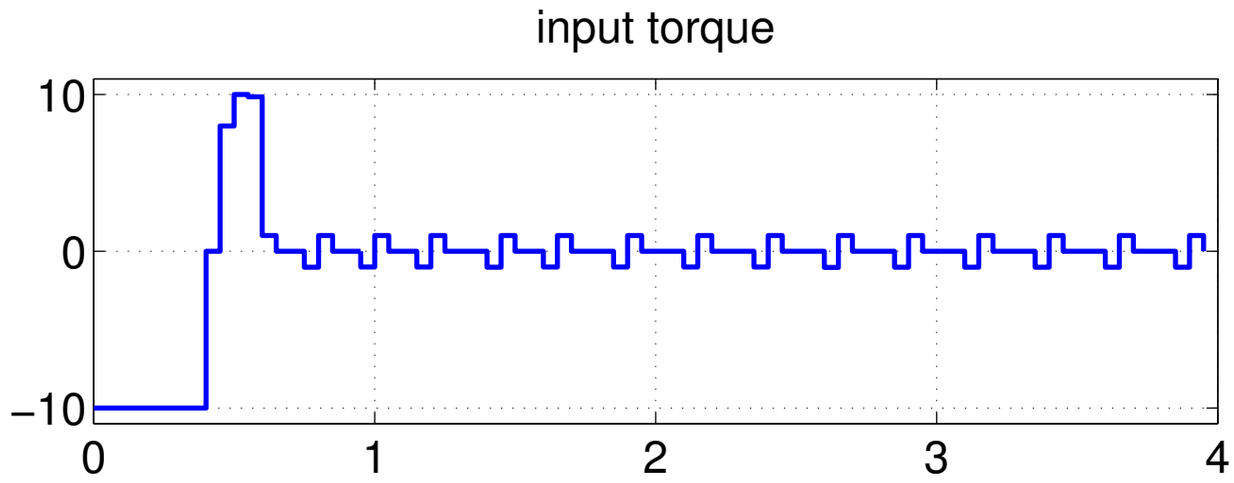
```
>> C

Hybrid controller based on MLD model S <pendulum.hys> [Inf-norm]

  2 state measurement(s)
  1 output reference(s)
  1 input reference(s)
  0 state reference(s)
  0 reference(s) on auxiliary continuous z-variables

 55 optimization variable(s) (30 continuous, 25 binary)
155 mixed-integer linear inequalities
sampling time = 0.05, MILP solver = 'gurobi'

Type "struct(C)" for more details.
>>
```

INVERTED PENDULUM: CLOSED-LOOP RESULTS

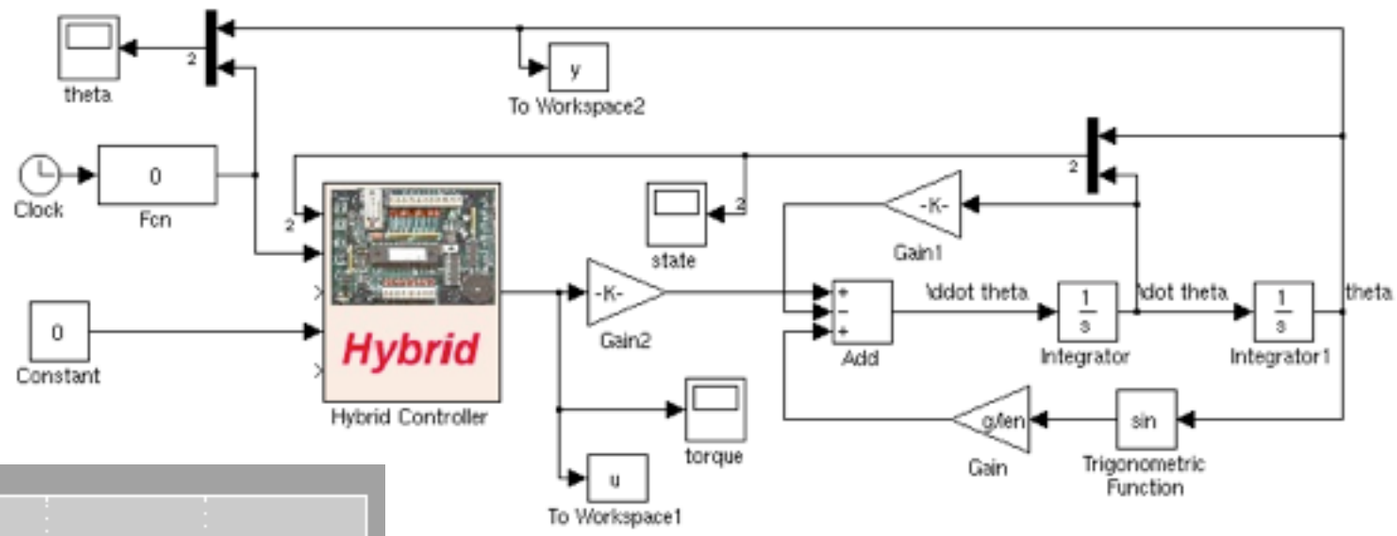
- Nominal simulation

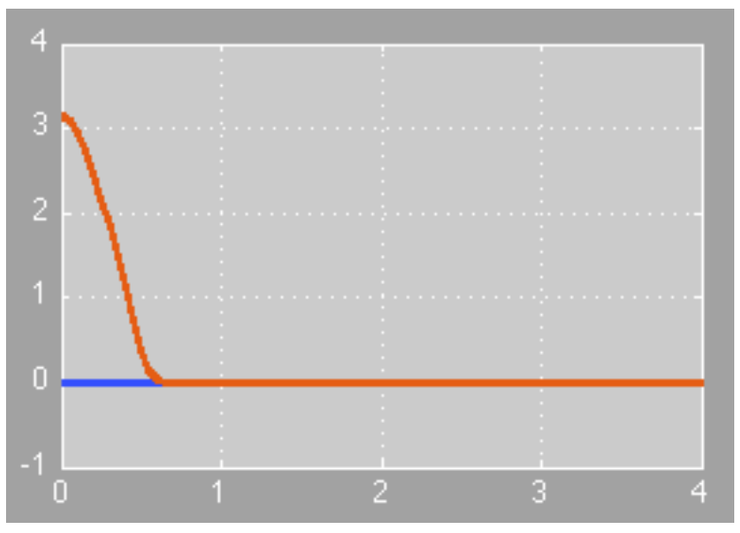
```
>>[X,U,D,Z,T,Y]=sim(C,S,r,x0,4);
```

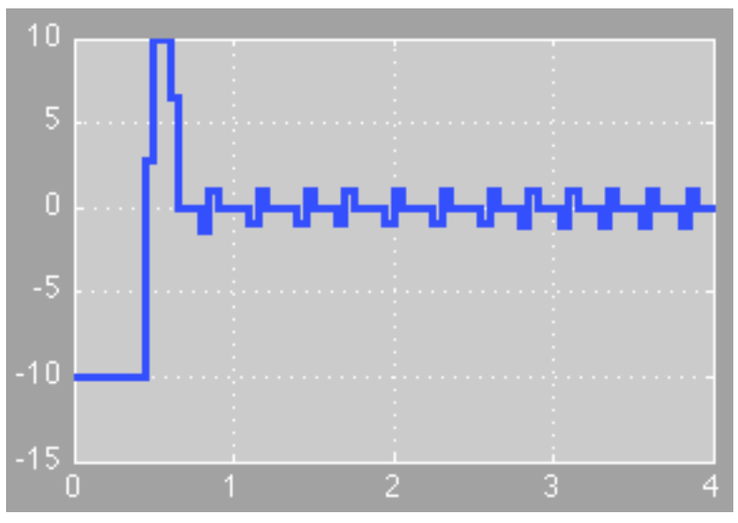




- Nonlinear simulation







CPU time:
51 ms per time step (GLPK)
22 ms per time step (CPLEX)
25 ms (GUROBI) on this machine

EXPLICIT HYBRID MPC

EXPLICIT HYBRID MPC (MLD FORMULATION)

$$\min_\xi J(\xi, x(t)) = \sum_{k=0}^{N-1} \|Qy_k\|_\infty + \|Ru_k\|_\infty$$

$$\text{subject to} \begin{cases} x_{k+1} = Ax_k + B_1u_k + B_2\delta_k + B_3z_k + B_5 \\ y_k = Cx_k + D_1u_k + D_2\delta_k + D_3z_k + D_5 \\ E_2\delta_k + E_3z_k \le E_4x_k + E_1u_k + E_5 \\ x_0 = x(t) \end{cases}$$

- On-line optimization: solve the problem for a given state $x(t)$

Mixed-Integer Linear Program (MILP)

- Off-line optimization: solve the MILP in advance **for all** $x(t)$

$$\begin{aligned} \min_\xi \quad & \sum_{k=0}^{N-1} \epsilon_k^y + \epsilon_k^u \\ \text{s.t.} \quad & G\xi \le W + Sx(t) \end{aligned}$$

multi-parametric Mixed Integer Linear Program (mp-MILP)

MULTIPARAMETRIC MILP

$$\begin{aligned} \min_{\xi_c,\xi_d} \quad & f_c'\xi_c + f_d'\xi_d \\ \text{s.t.} \quad & G_c\xi_c + G_d\xi_d \le W + Sx \end{aligned}$$

$\xi_c \in \mathbb{R}^n$

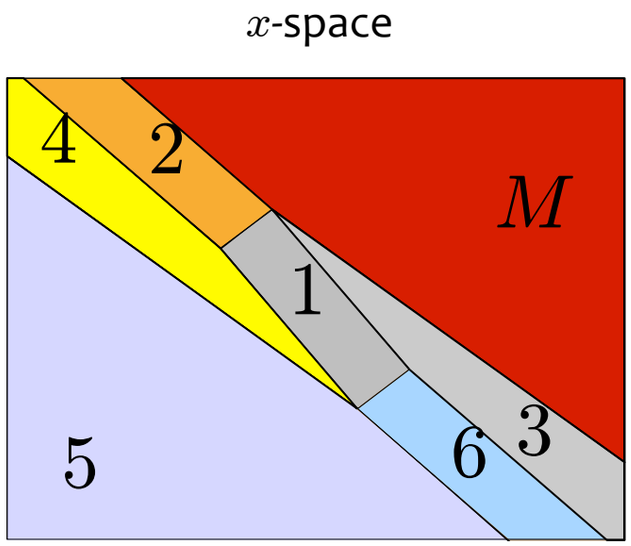
$\xi_d \in \{0,1\}^m$

- mp-MILP can be solved (by alternating MILPs and mp-LPs) (Dua, Pistikopoulos, 1999)

- **Theorem:** The multiparametric solution $\xi^*(x)$ is **piecewise affine** (but possibly discontinuous)

- **The MPC controller is piecewise affine in** $x=x(t)$

$$u(x) = \begin{cases} F_1x + g_1 & \text{if} \quad H_1x \le K_1 \\ \vdots & \vdots \\ F_Mx + g_M & \text{if} \quad H_Mx \le K_M \end{cases}$$



(more generally, the MPC controller is a PWA function of x and of the reference signals)

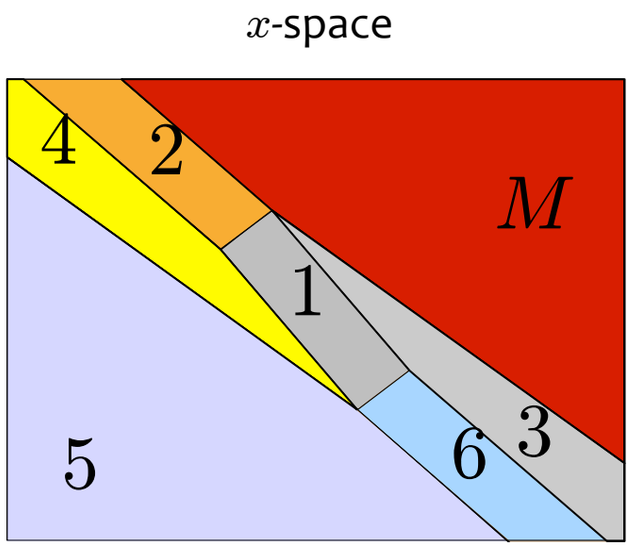
ALTERNATIVE: USE PWA FORMULATION

$$\min_\xi J(\xi, x(t)) = \sum_{k=0}^{N-1} \|Qy_k\|_\infty + \|Ru_k\|_\infty$$

$$\text{subject to } \begin{cases} x_{k+1} = A_{i(k)}x_k + B_{i(k)}u_k + f_{i(k)} \\ y_k = C_{i(k)}x_k + D_{i(k)}u_k + g_{i(k)} \\ i(k) \text{ such that } H_{i(k)}x_k + W_{i(k)}u_k \le K_{i(k)} \\ x_0 = x(t) \end{cases}$$

- **The MPC controller is piecewise affine in** $x=x(t)$

$$u(x) = \begin{cases} F_1 x + g_1 & \text{if } H_1 x \le K_1 \\ \vdots & \vdots \\ F_M x + g_M & \text{if } H_M x \le K_M \end{cases}$$



(more generally, the MPC controller is a PWA function of x and of the reference signals)

ALTERNATIVE: USE PWA FORMULATION

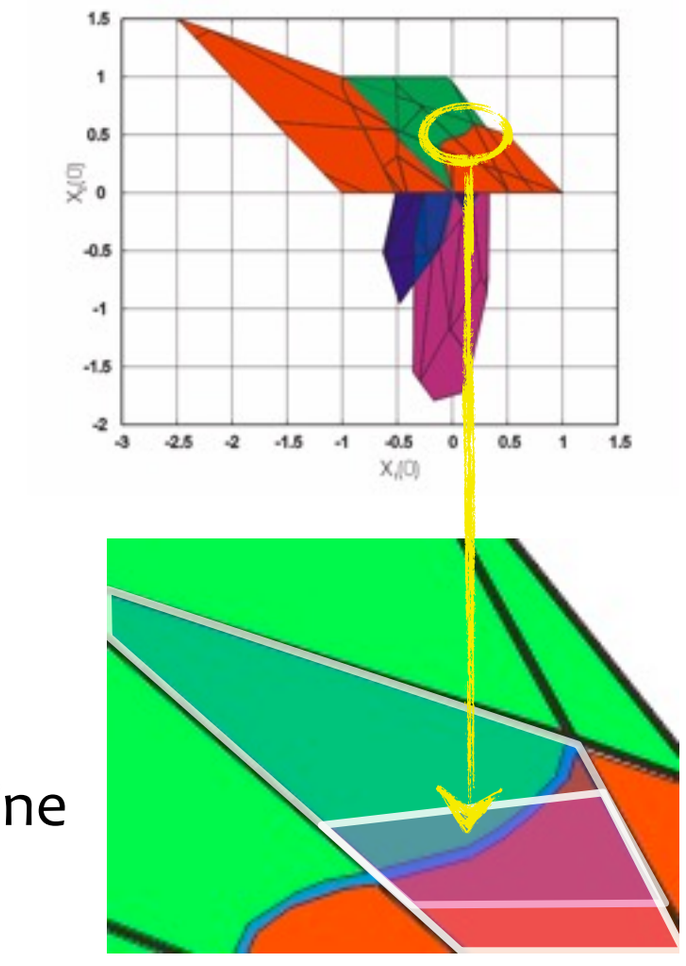
Method #1 (Borrelli, Baotic, Bemporad, Morari, *Automatica*, 2005)

Use a combination of DP (dynamic programming) and mpLP (1-norm, ∞-norm), or mpQP (quadratic forms)

Method #2 (Bemporad, *Hybrid Toolbox*, 2003) (Alessio, Bemporad, ADHS 2006) (Mayne, ECC 2001) (Mayne, Rakovic, 2002)

1 - Use backwards (=DP) reachability analysis for enumerating all feasible mode sequences $I=\{i(0),i(1),\ldots,i(N)\}$

2 - For each fixed sequence I, solve the explicit finite-time optimal control problem for the corresponding linear time-varying system (mpQP or mpLP)



3a - Case 1/∞-norm: Compare value functions and split regions

3b - Case quadratic costs: the partition may not be fully polyhedral

⇒ better keep overlapping polyhedra and compare on-line quadratic cost functions (when overlaps are detected)

Comparison of quadratic costs can be avoided by lifting the parameter space (Fuchs, Axehill, Morari, 2015)

HYBRID CONTROL EXAMPLES (REVISITED)

HYBRID CONTROL EXAMPLE

PWA system:

$$
\begin{aligned}
x(t+1) &= 0.8 \begin{bmatrix} \cos\alpha(t) & -\sin\alpha(t) \\ \sin\alpha(t) & \cos\alpha(t) \end{bmatrix} x(t) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(t) \\
y(t) &= x_2(t) \\
\alpha(t) &= \begin{cases} \frac{\pi}{3} & \text{if } x_1(t) \geq 0 \\ -\frac{\pi}{3} & \text{if } x_1(t) < 0 \end{cases}
\end{aligned}
$$

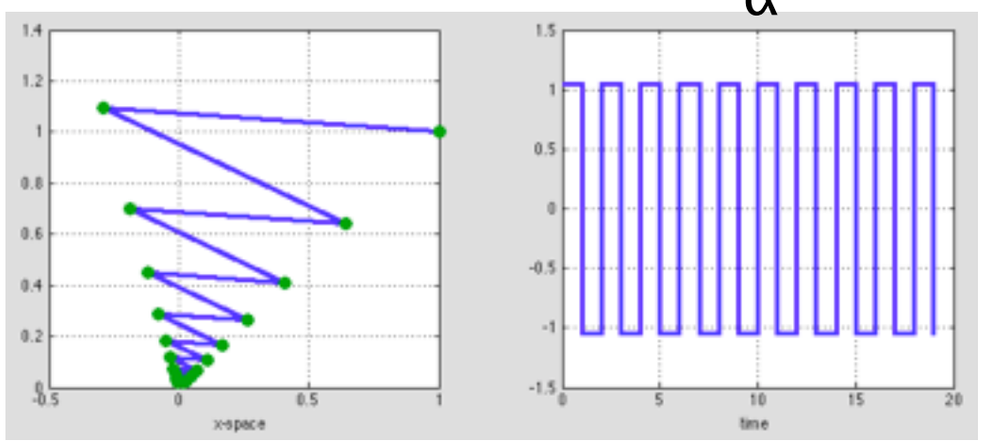
Constraints:

$$-1 \leq u(t) \leq 1$$
$$-10 \leq x(t) \leq 10$$

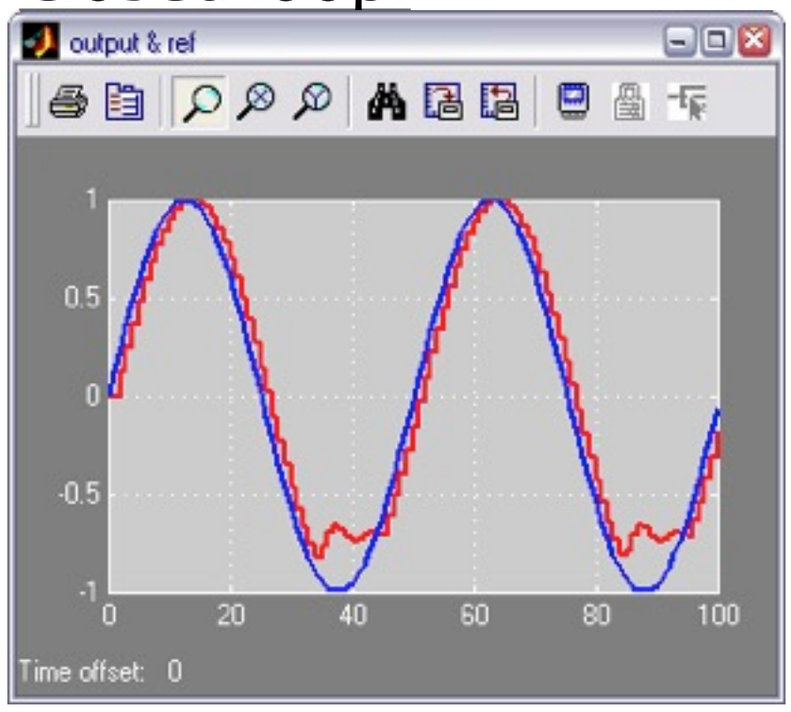
Objective:

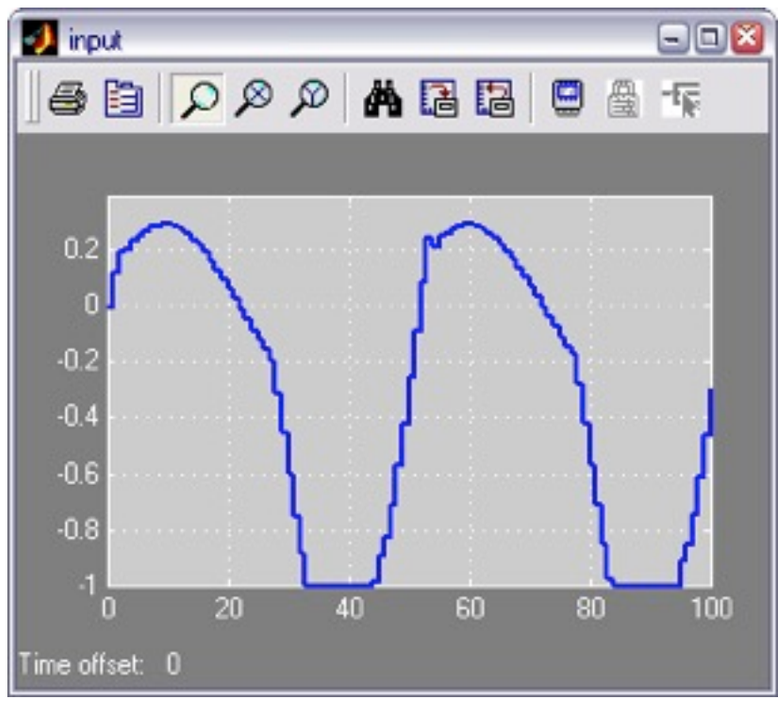
$$\min \sum_{k=1}^{2} |y_k - r(t)|$$

Open loop behavior:



Closed loop:

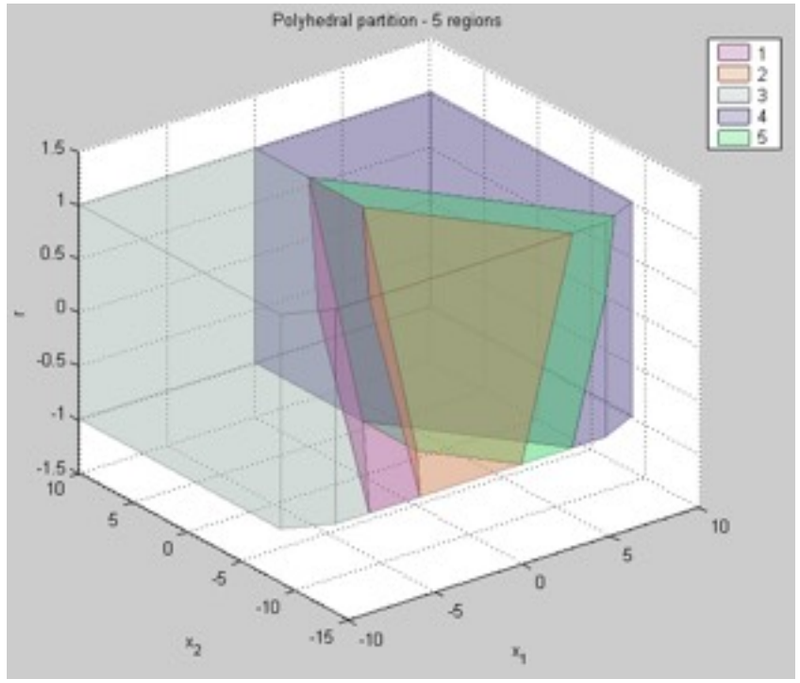




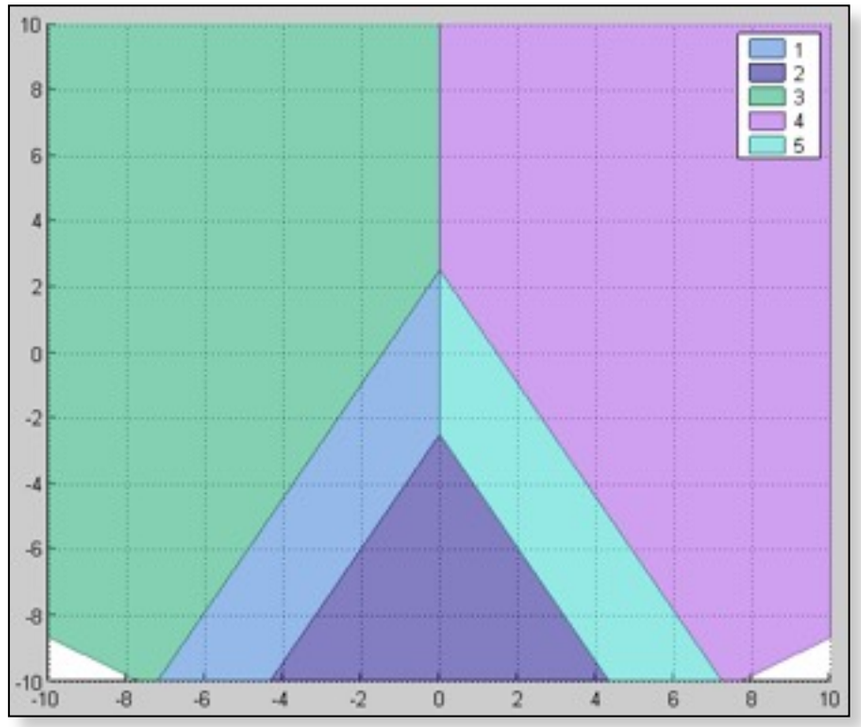
HybTbx: /**demos/hybrid/bm99sim.m**

EXPLICIT PWA CONTROLLER

$$u(x,r) = \begin{cases} \begin{bmatrix} 0.6928 & -0.4 & 1 \end{bmatrix} \begin{bmatrix} x \\ r \end{bmatrix} & \text{if } \begin{bmatrix} 0.6928 & -0.4 & 1 \\ -0.4 & -0.6928 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \\ -0.6928 & 0.4 & -1 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ r \end{bmatrix} \le \begin{bmatrix} 1 \\ 10 \\ 10 \\ 1 \\ 1 \\ 1 \\ 1e{-}006 \end{bmatrix} \\ \text{(Region \#1)} \\ 1 & \text{if } \begin{bmatrix} -0.6928 & 0.4 & -1 \\ 0.6928 & 0.4 & -1 \\ 0 & 0 & 1 \\ 0 & 0 & -1 \\ 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} x \\ r \end{bmatrix} \le \begin{bmatrix} -1 \\ -1 \\ 1 \\ 1 \\ 10 \end{bmatrix} \\ \text{(Region \#2)} \\ -1 & \text{if } \begin{bmatrix} -0.4 & -0.6928 & 0 \\ 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0.6928 & -0.4 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & -1 \\ -1 & 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ r \end{bmatrix} \le \begin{bmatrix} 10 \\ 10 \\ 1e{-}006 \\ 10 \\ -1 \\ 1 \\ 1 \\ 10 \end{bmatrix} \\ \text{(Region \#3)} \\ -1 & \text{if } \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0.4 & -0.6928 & 0 \\ 0 & -1 & 0 \\ -0.6928 & -0.4 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & -1 \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ r \end{bmatrix} \le \begin{bmatrix} 0 \\ 10 \\ 10 \\ 10 \\ -1 \\ 1 \\ 1 \\ 10 \end{bmatrix} \\ \text{(Region \#4)} \\ \begin{bmatrix} -0.6928 & -0.4 & 1 \end{bmatrix} \begin{bmatrix} x \\ r \end{bmatrix} & \text{if } \begin{bmatrix} -0.6928 & -0.4 & 1 \\ 0.4 & -0.6928 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \\ 0.6928 & 0.4 & -1 \\ 0 & 0 & 1 \\ -1 & 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ r \end{bmatrix} \le \begin{bmatrix} 1 \\ 10 \\ 10 \\ 1 \\ 1 \\ 1 \\ 0 \end{bmatrix} \\ \text{(Region \#5)} \end{cases}$$



Section with r=0



x_2

x_1

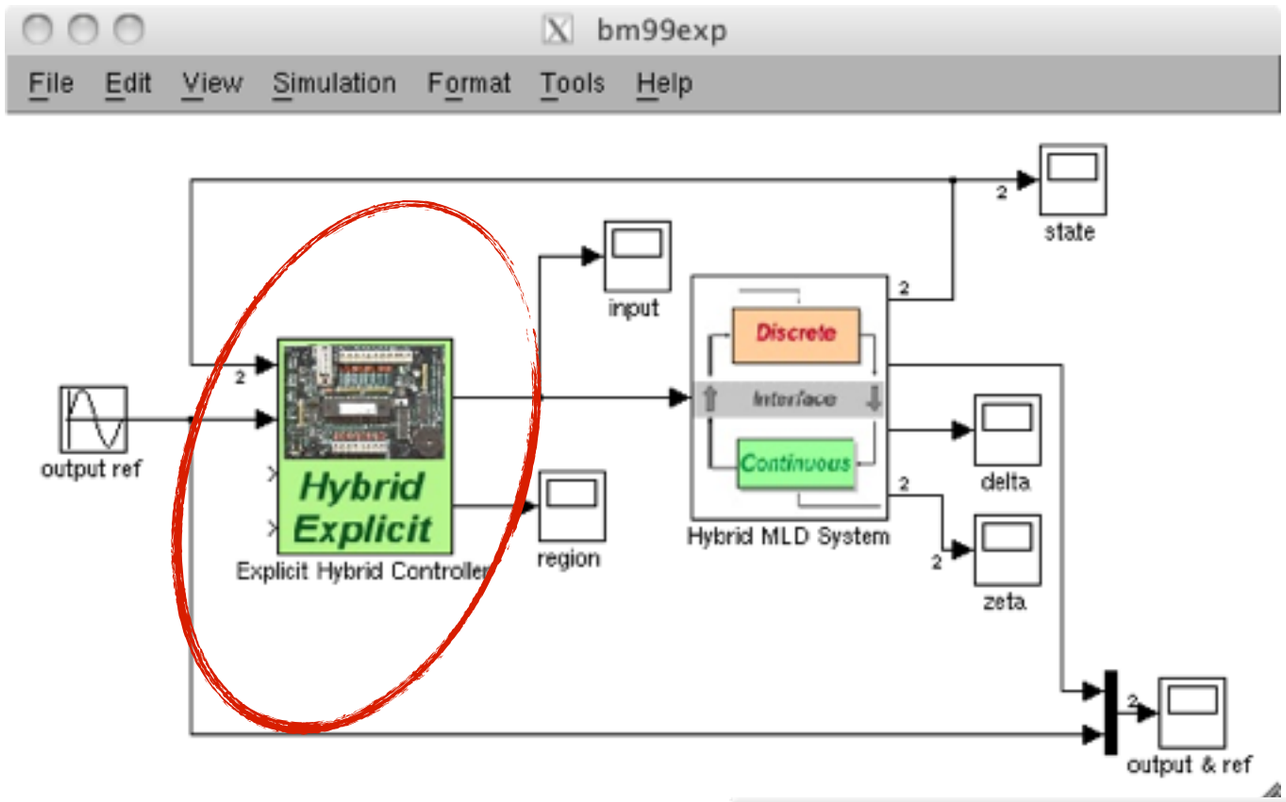
HybTbx: **/demos/hybrid/bm99sim.m**

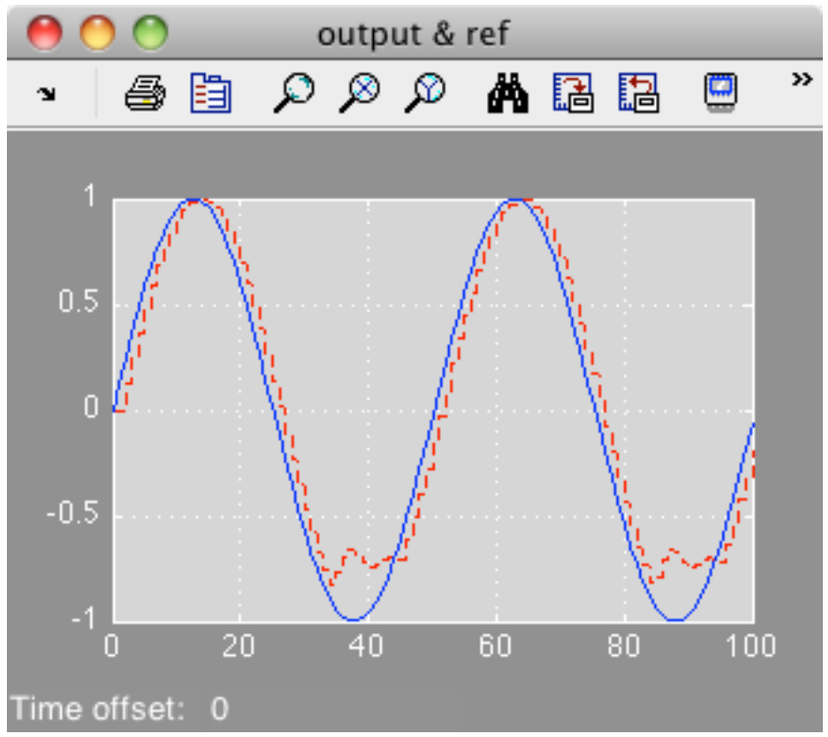
(CPU time = 1.51 s, this machine)

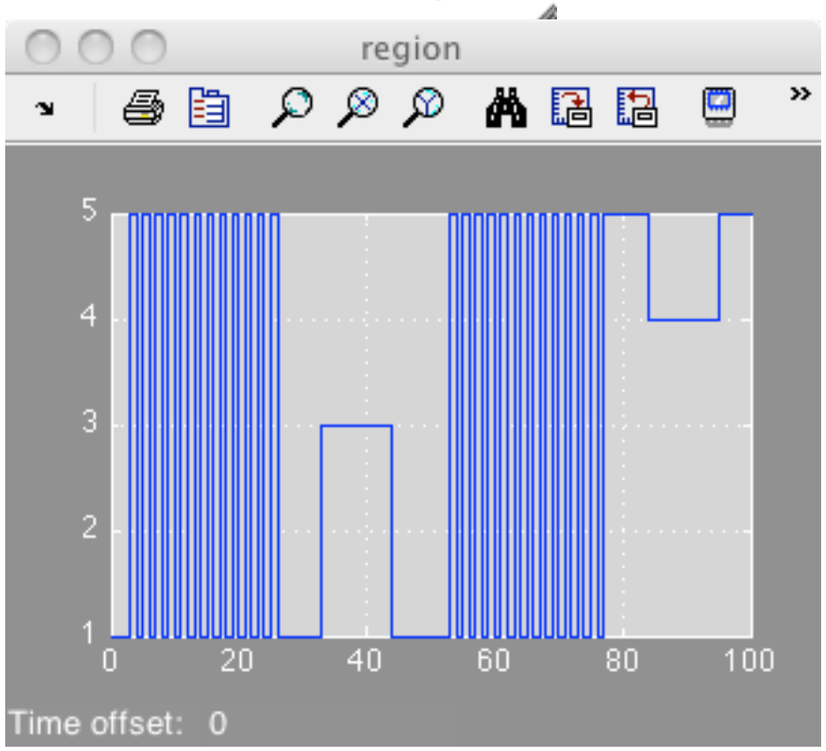
PWA law ≡ MPC law !

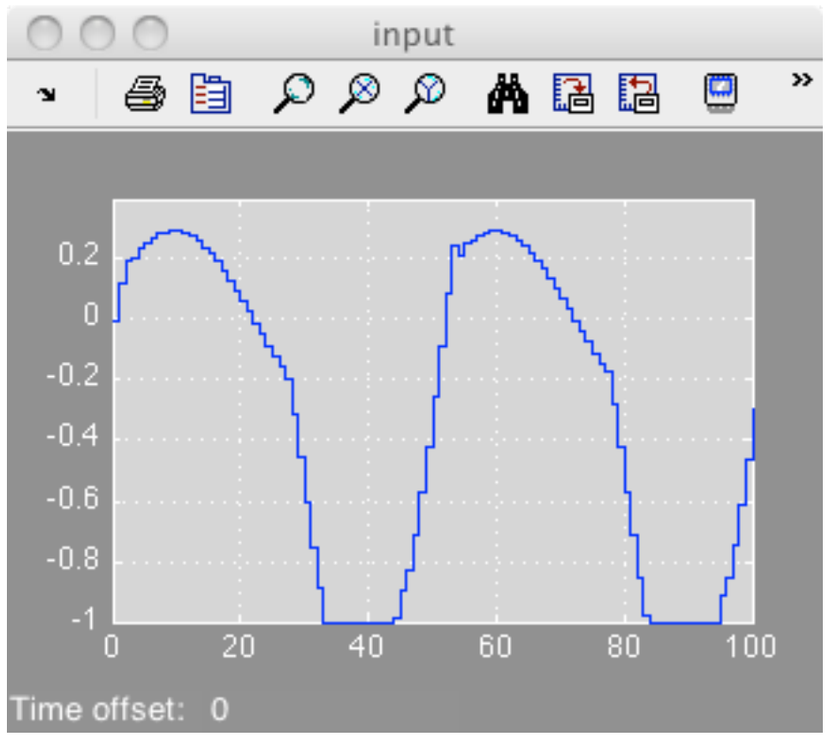
HYBRID CONTROL EXAMPLE

Closed loop:









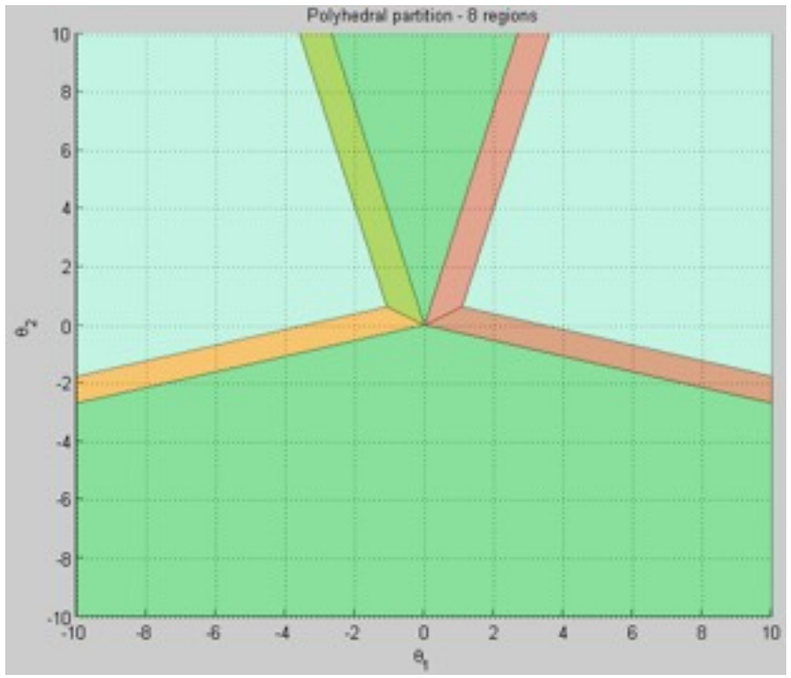
EXPLICIT PWA REGULATOR

MPC problem:

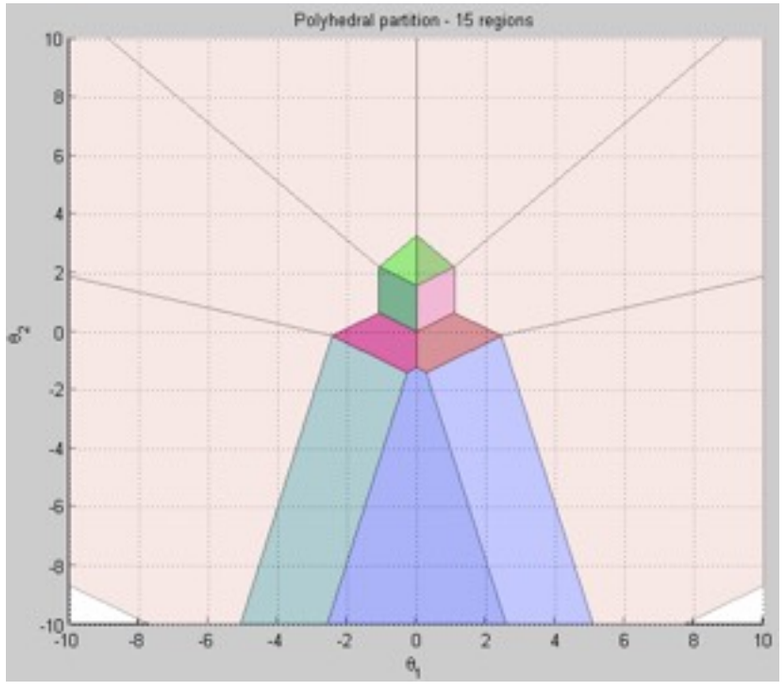
$$\begin{aligned} \min \quad & 10\|x_N\|_\infty + \sum_{k=0}^{N-1} 10\|x_k\|_\infty + \|u_k\|_\infty \\ \text{s.t.} \quad & \begin{cases} -1 \leq u_k \leq 1, \ k = 0, \ldots, N-1 \\ -10 \leq x_k \leq 10, \ k = 1, \ldots, N \end{cases} \end{aligned}$$

$$\left(\begin{array}{rcl} Q & = & \left[\begin{smallmatrix} 10 & 0 \\ 0 & 10 \end{smallmatrix}\right] \\ R & = & 1 \end{array} \right)$$

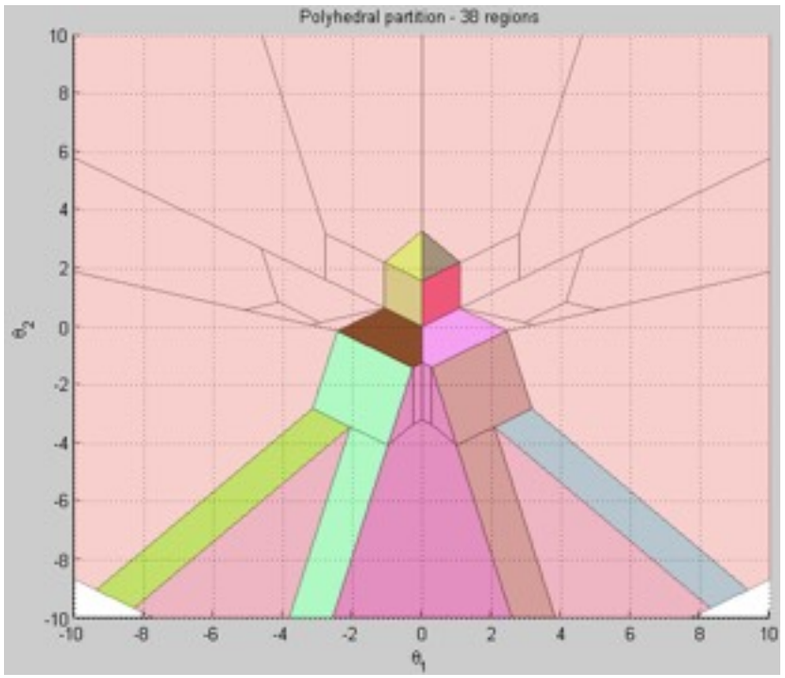
Prediction horizon N=1



Prediction horizon N=2



Prediction horizon N=3

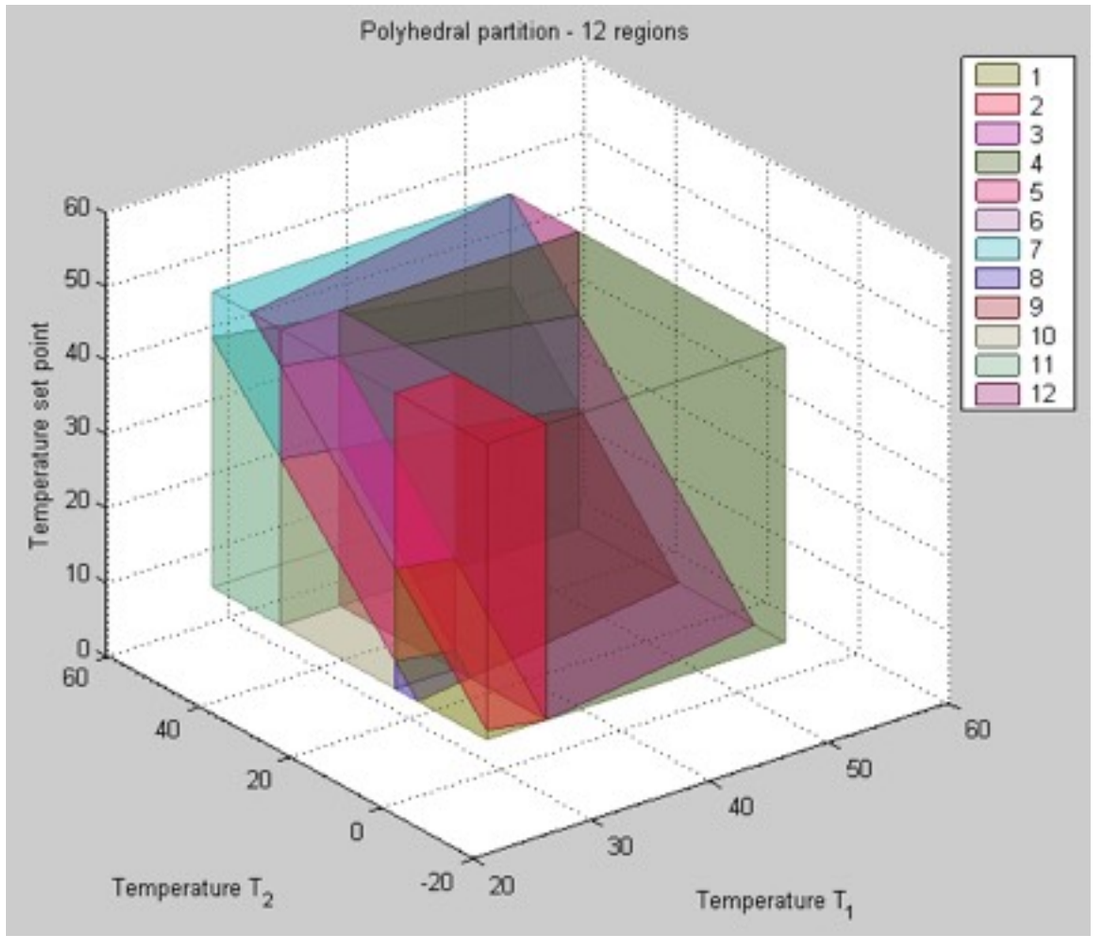


HybTbx: /**demos/hybrid/bm99benchmark.m**

EXPLICIT MPC – TEMPERATURE CONTROL

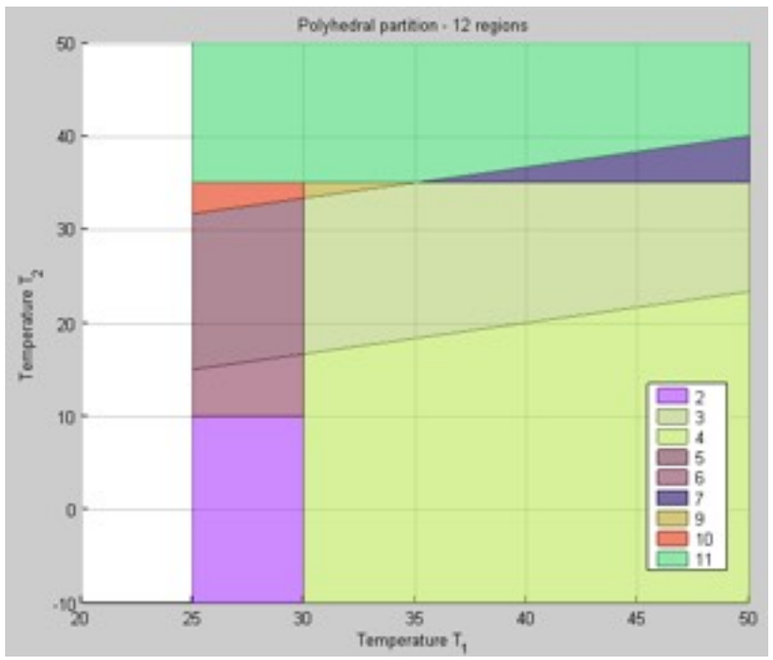
```
>>E=expcon(C,range,options);
```

```
>> E

Explicit controller (based on hybrid controller C)
  3 parameter(s)
  1 input(s)
 12 partition(s)
sampling time = 0.5

The controller is for hybrid systems (tracking)
This is a state-feedback controller.

Type "struct(E)" for more details.
>>
```



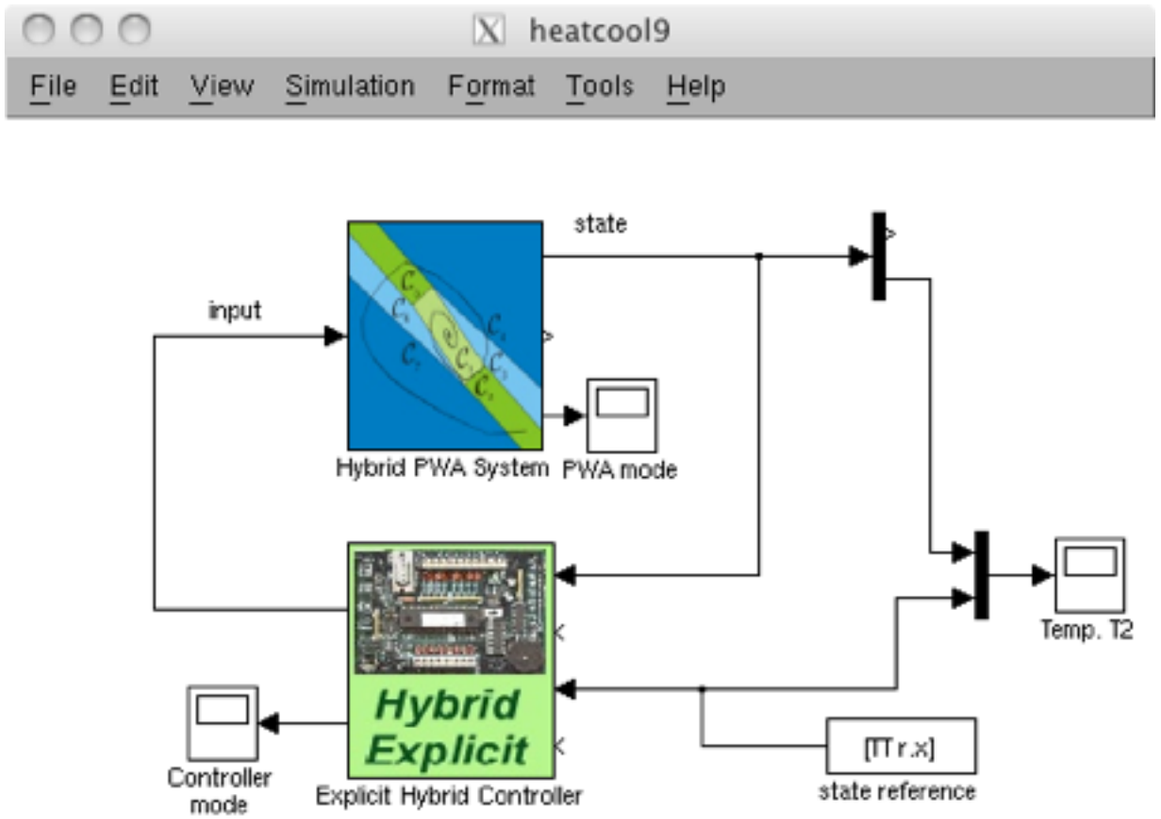
384 numbers to store in memory

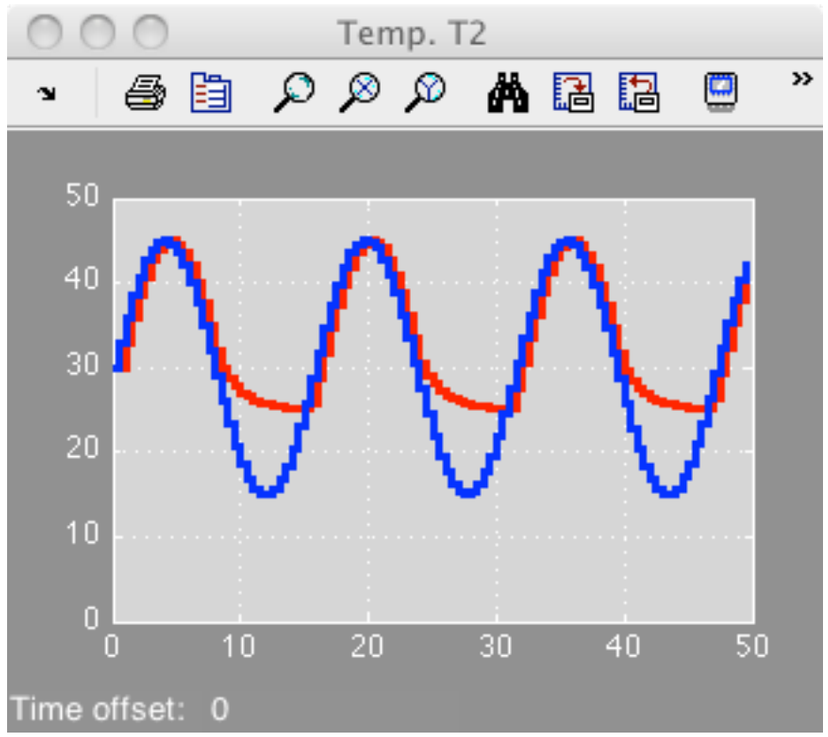
$$\min \sum_{k=0}^{2} \|x_{2k} - r(t)\|_\infty$$
$$\text{s.t.} \begin{cases} x_{1k} \geq 25, \ k = 1,2 \\ \text{hybrid model} \end{cases}$$



Section in the (T_1,T_2)-space for $T_{ref} = 30$

EXPLICIT MPC – TEMPERATURE CONTROL

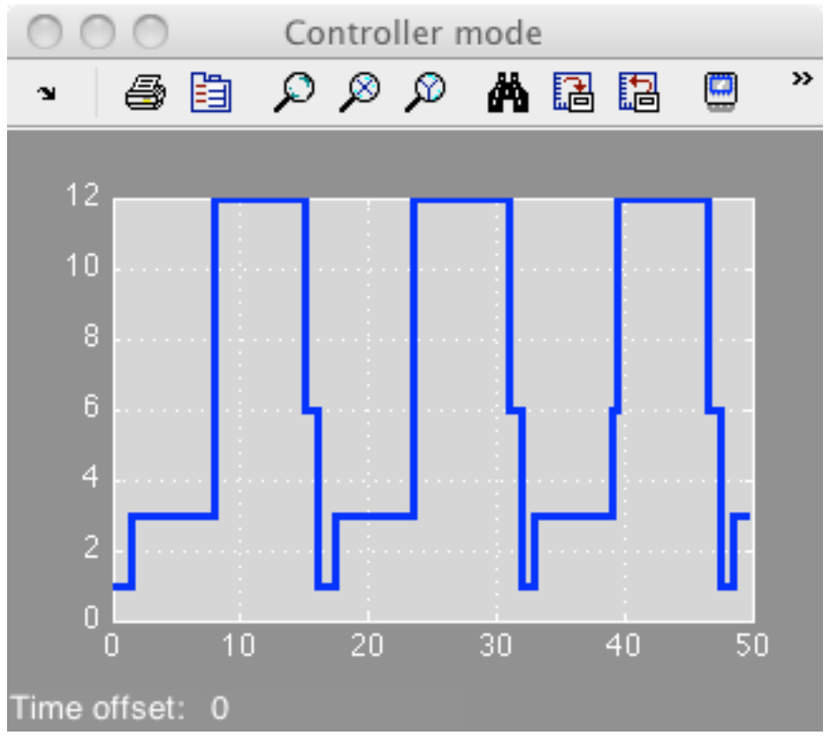




generated C-code

utils/expcon.h

```
#define EXPCON_REG 12
#define EXPCON_NTH 3
#define EXPCON_NYM 2
#define EXPCON_NH 72
#define EXPCON_NF 12
static double EXPCON_F[]={
    -1,0,0,0,-1,0,
    -1,-1,-1,-1,-1,-1,0,-3,-3,
    -3,0,-3,0,0,0,0,0,
    0,0,4,4,4,0,4,0,0,
    0,0,0,0);

static double EXPCON_G[]={
    101.6,1.6,1.6,-1.6,98.4001,0,100,51.6,
    101.6,51.6,48.4,50);

static double EXPCON_H[]={
    0,0,0,-0.00999999,0,-0.0333333,
    0.02,0.00999999,-0.02,0,0,-0.0333333,0.02,0.00999999,
    0,0,-0.02,0.02,0,-1,0.00999999,0,
```



IMPLEMENTATION ASPECTS OF HYBRID MPC

- **Alternatives**: (1) solve MIP on-line
  (2) evaluate a PWA function (explicit solution)

- **Small problems** (short horizon N=1,2, one or two inputs): explicit PWA control law preferable

  - time to evaluate the control law is shorter than MIP

  - control code is simpler (no complex solver must be included in the control software !)

  - more insight in controller's behavior

- **Medium/large problems** (longer horizon, many inputs and binary variables): MIP preferable

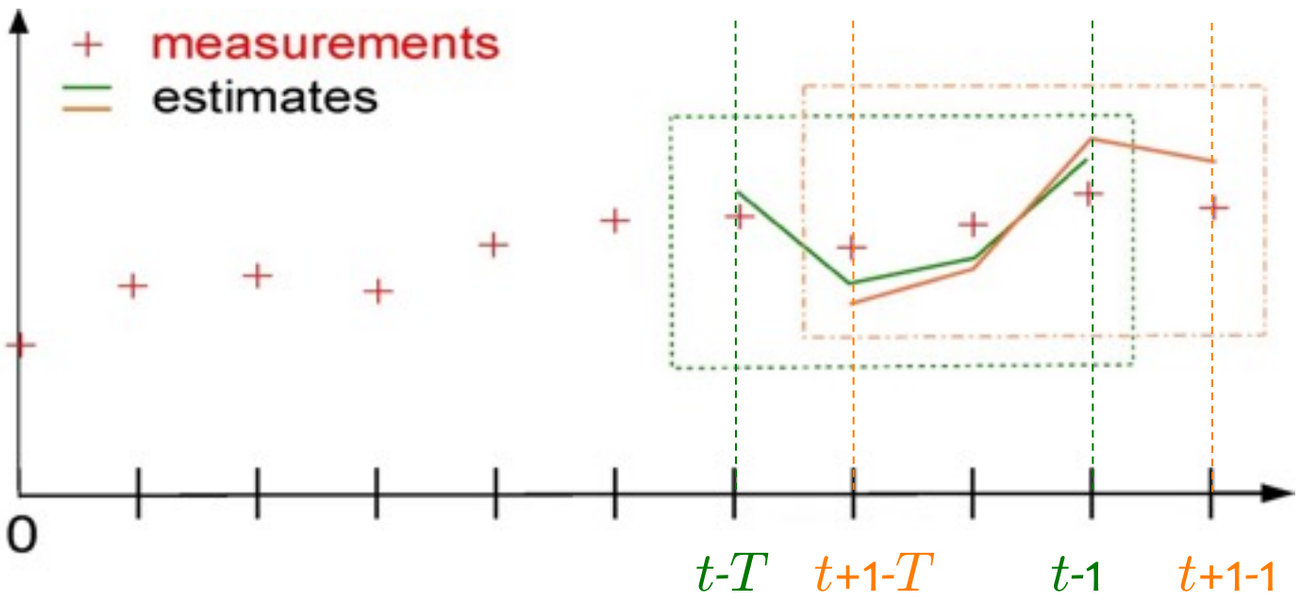
MOVING HORIZON ESTIMATION

FAULT DETECTION & ISOLATION

STATE ESTIMATION / FAULT DETECTION

- Problem: given past output measurements and inputs, estimate the current states and faults

- Solution: Use Moving Horizon Estimation for MLD systems (≈ dual of MPC)



Augment the MLD model with:

- Input disturbances $\xi \in \mathbb{R}^n$
- Output disturbances $\zeta \in \mathbb{R}^p$

At each time t solve the problem:

$$\min_{\hat{x}(t-T|t)} \sum_{k=0}^{T} \|\hat{y}(t-k|t) - y(t-k)\|_2^2 + \dots$$

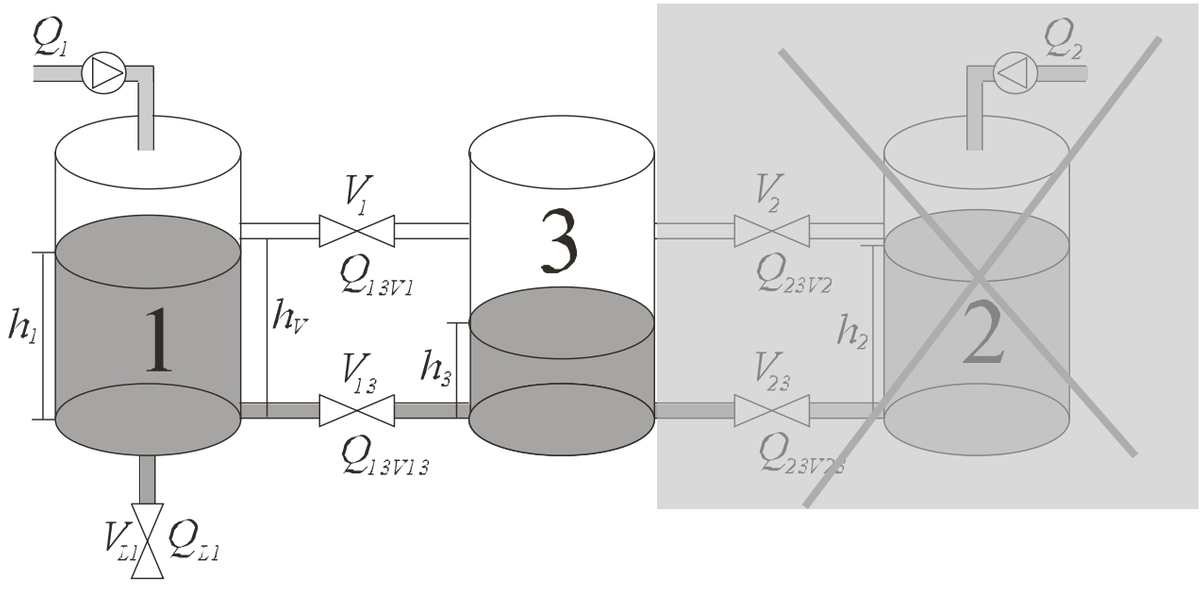
and get estimate $\hat{x}(t)$

⇒ MHE optimization = MIQP (Bemporad, Mignone, Morari, ACC 1999)

⇒ Convergence can be guaranteed (Ferrari-T., Mignone, Morari, 2002)

Fault detection: augment MLD with unknown **binary** disturbances $\phi \in \{0,1\}^{n_f}$

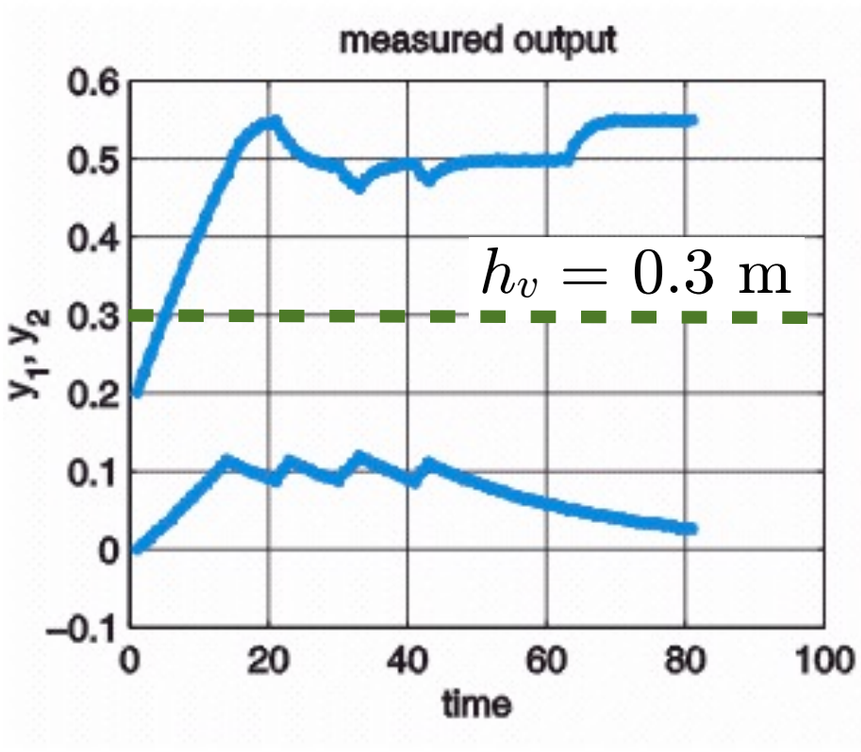
EXAMPLE: THREE TANK SYSTEM

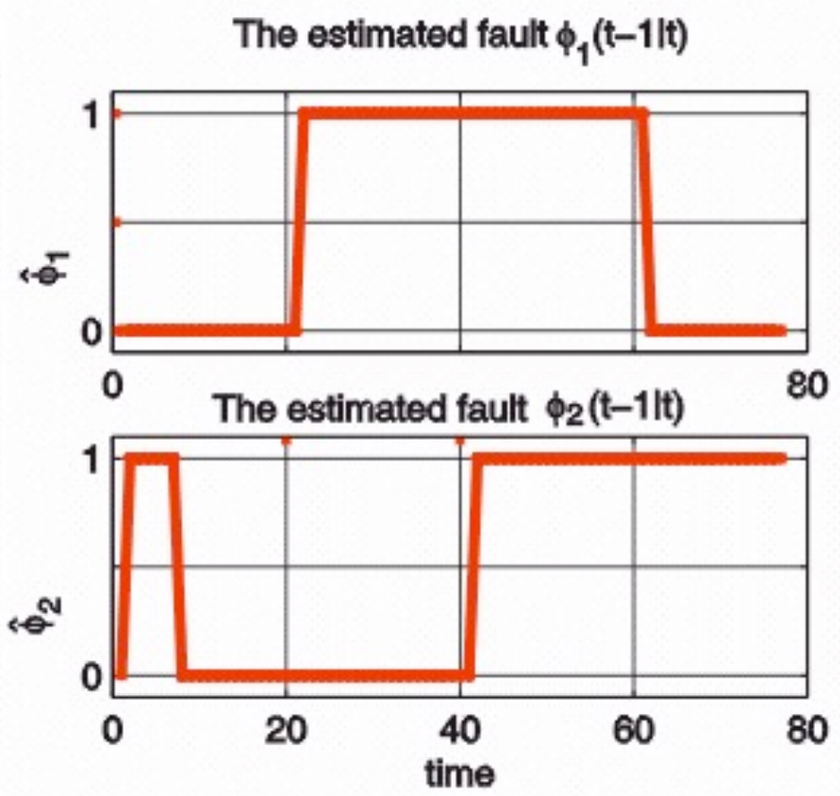


COSY Benchmark problem, ESF

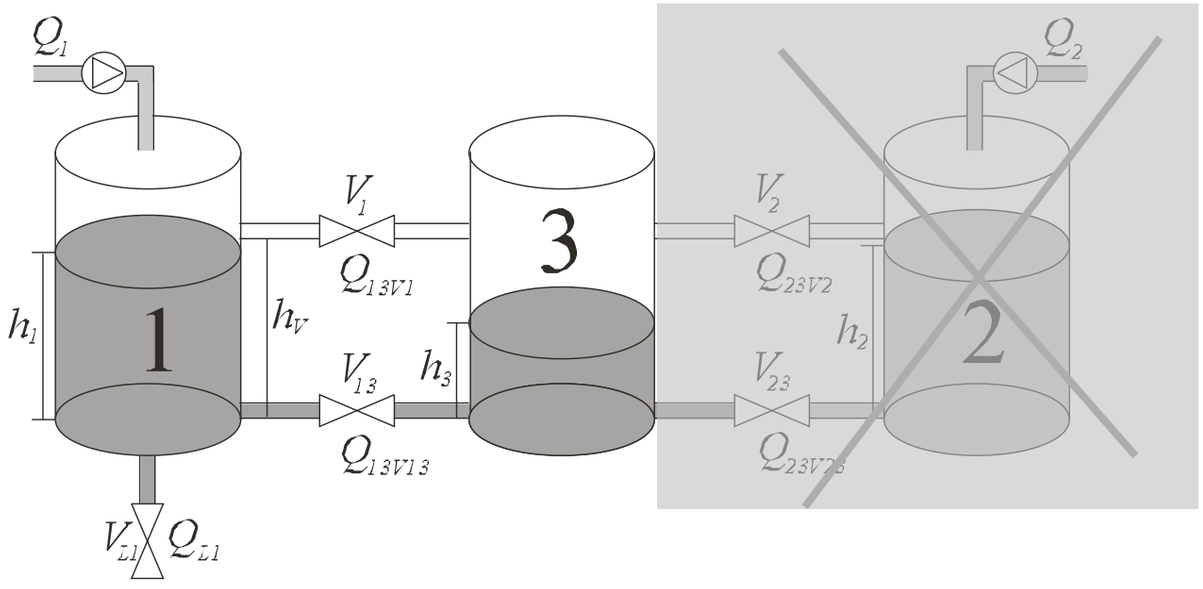
- ϕ_1 : leak in tank 1
  for $20\text{s} \le t \le 60\text{s}$

- ϕ_2 : valve V_1 blocked
  for $t \ge 40\text{s}$





EXAMPLE: THREE TANK SYSTEM



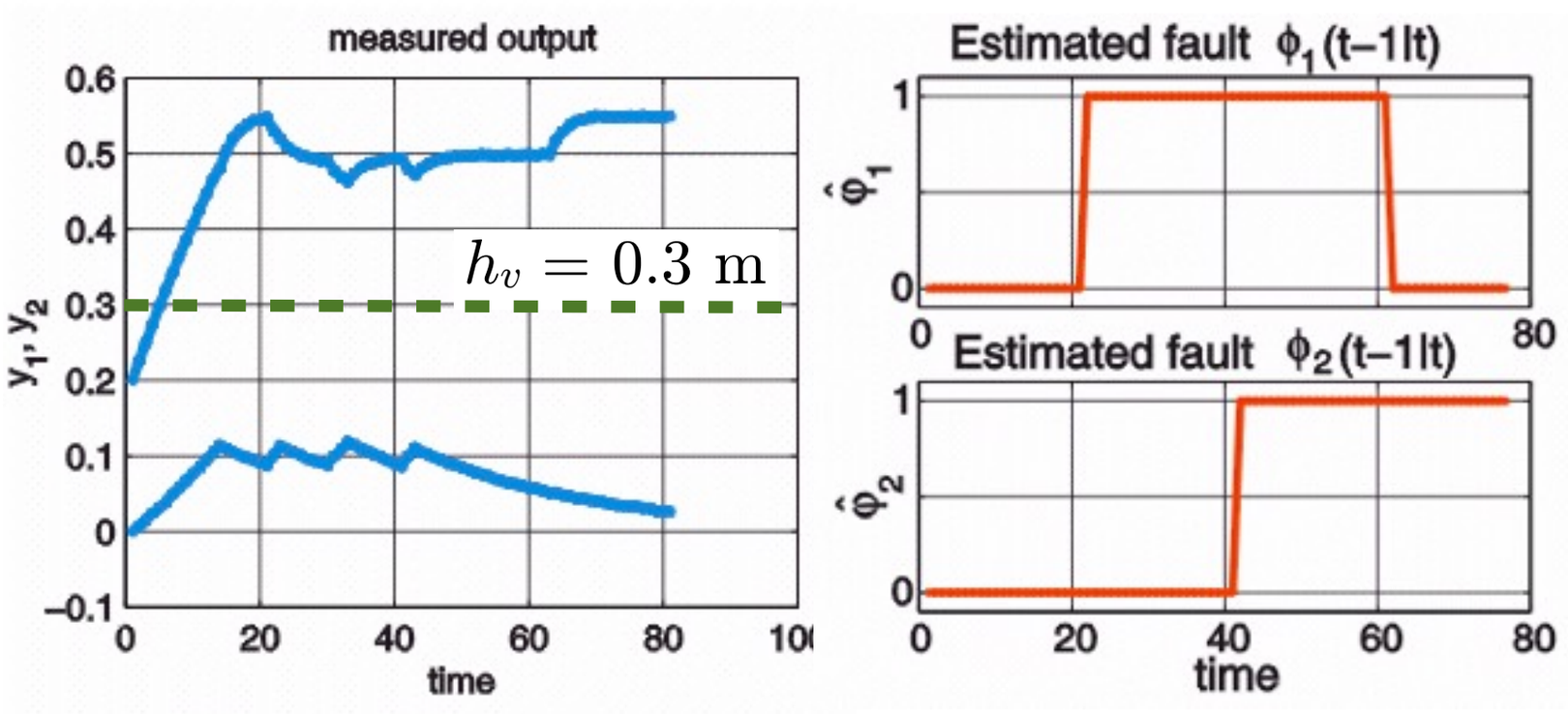
COSY Benchmark problem, ESF

- ϕ_1 : leak in tank 1
  for $20\text{s} \le t \le 60\text{s}$

- ϕ_2 : valve V_1 blocked
  for $t \ge 40\text{s}$

- Add logic constraint

$$[h_1 \le h_v] \Rightarrow \phi_2 = 0$$



A FEW HYBRID MPC TRICKS …

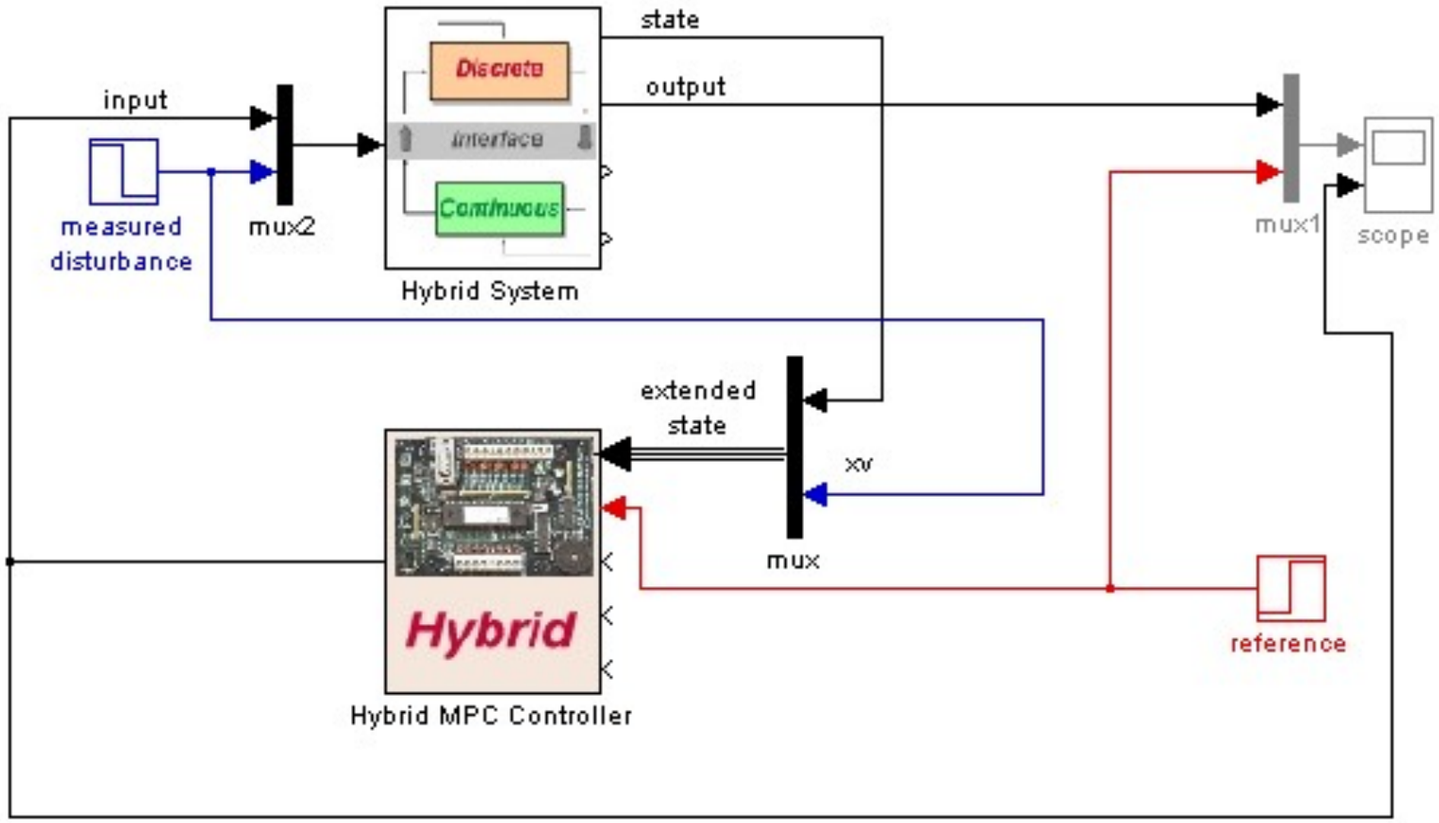
MEASURED DISTURBANCES

- Disturbance $v(t)$ can be measured at time t
- Augment the hybrid prediction model with a constant state

$$x_v(t+1) = x_v(t)$$

- In HYSDEL:

```
INTERFACE{
      STATE{
         REAL x      [-1e3, 1e3];
         REAL xv     [-1e3, 1e3];
      }
      ...
   }
IMPLEMENTATION{
      CONTINUOUS{
         x = A*x + B*u + Bv*xv
         xv= xv;
         ...
      }
}
```



/demos/hybrid/hyb_meas_dist.m

Note: same trick applies to linear MPC

HYBRID MPC - TRACKING

- Optimal control problem (quadratic performance index):

$$\begin{aligned}
\min_{\Delta U} \quad & \sum_{k=0}^{N-1} \|W^y(y_{k+1} - r(t))\|^2 + \|W^{\Delta u}\Delta u_k\|^2 \\
& [\Delta u_k \triangleq u_k - u_{k-1}],\ u_{-1} = u(t-1) \\
\text{subj. to} \quad & \text{hybrid dynamics} \\
& u_{\min} \le u_k \le u_{\max},\ k = 0,\dots,N-1 \\
& y_{\min} \le y_k \le y_{\max},\ k = 1,\dots,N \\
& \Delta u_{\min} \le \Delta u_k \le \Delta u_{\max},\ k = 0,\dots,N-1
\end{aligned}$$

- Optimization problem:

Mixed-Integer Quadratic Program (MIQP)

$$\begin{aligned}
\min_{\xi} \quad & J(\xi, x(t)) = \tfrac{1}{2}\xi' H \xi + [x'(t)\ r'(t)\ u'(t-1)]F\xi \\
\text{s.t.} \quad & G\xi \le W + S\begin{bmatrix} x(t) \\ r(t) \\ u(t-1) \end{bmatrix}
\end{aligned}$$

$$\xi = \begin{bmatrix} \Delta u_0 \\ \delta_0 \\ z_0 \\ \vdots \\ \Delta u_{N-1} \\ \Delta_{N-1} \\ z_{N-1} \end{bmatrix}$$

Note: same trick as in linear MPC

INTEGRAL ACTION IN HYBRID MPC

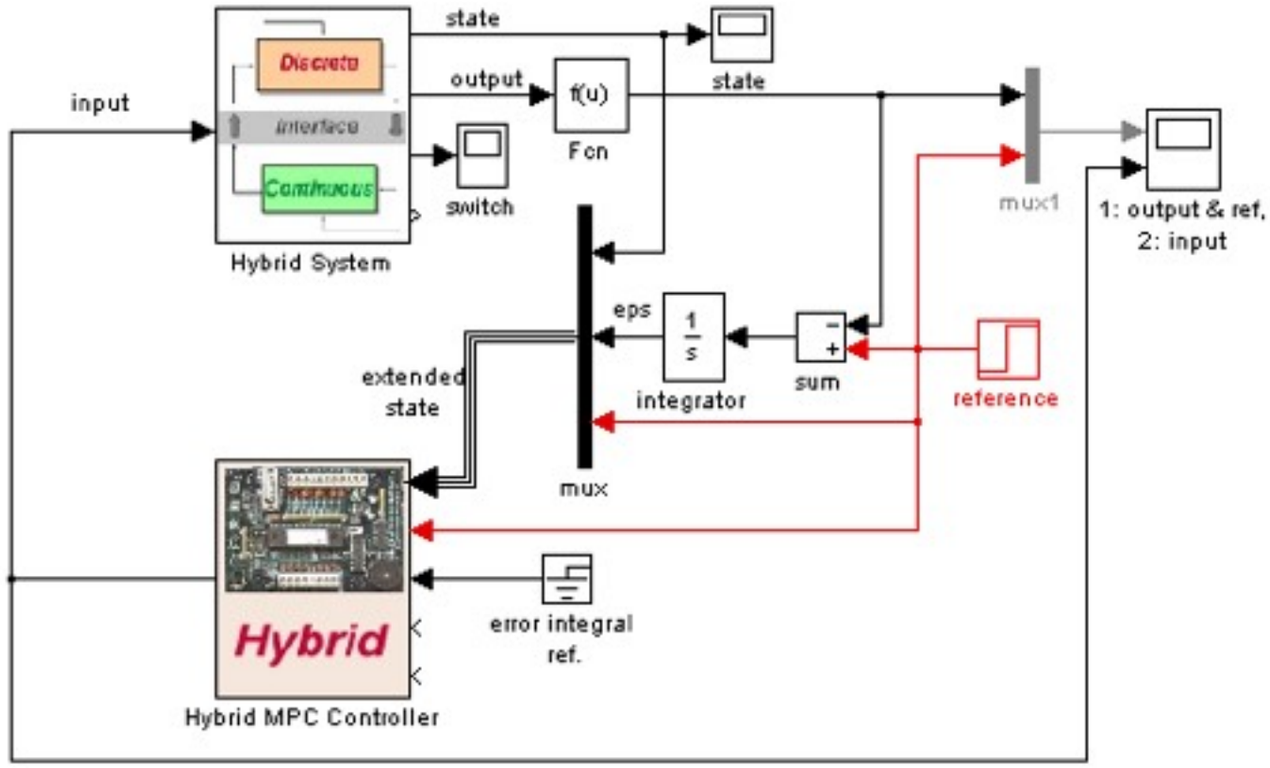
- Augment the hybrid prediction model with integrators of output errors as additional states:

$$\epsilon(k+1) = \epsilon(k) + T_s \cdot (r(k) - y(k))$$

T_s = sampling time

- Treat $r(k)$ as a measured disturbance (=additional constant state)
- Add weight on $\epsilon(k)$ in cost function to make $\epsilon(k) \longrightarrow 0$
- In HYSDEL:

```
INTERFACE{
      STATE{
         REAL x            [-100,100];
         ...
         REAL epsilon      [-1e3, 1e3];
         REAL r            [0,    100]; }
      OUTPUT {
         REAL y; }
      ... }
IMPLEMENTATION{
      CONTINUOUS{
         epsilon=epsilon+Ts*(r-(c*x));
         r=r;
         ...   }
      OUTPUT{
         y=c*x; }    }
```



/demos/hybrid/hyb_integral_action.m

Note: same trick applies to linear MPC

TIME-VARYING CONSTRAINTS

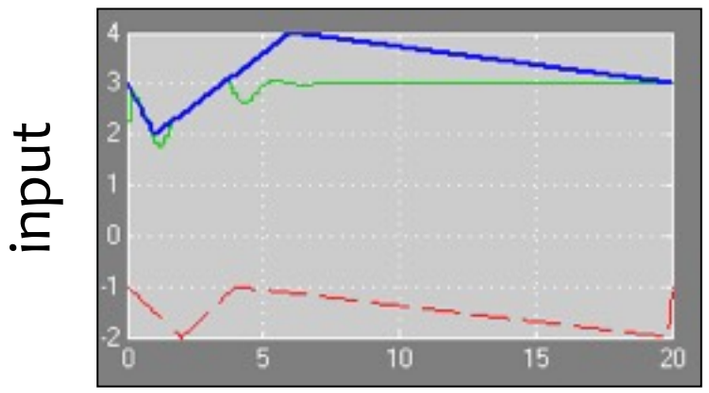
Problem: change upper (and/or lower) bounds on line $u(t) \leq u_{\max}(t)$

1. Add a constant state and a new output in the prediction model:

$$\begin{cases} x_u(k+1) = x_u(k) \\ y_u(k) = x_u(k) - u(k) \end{cases}$$

2. Add output constraint

$$y_u(k) \geq 0, \;\; k = 0, 1, \ldots, N$$



3. On-line implementation: feed the state $x_u(t) = u_{\max}(t)$ back to the controller

Note: same trick applies to linear MPC

/demos/linear/varbounds.m

Alternative (in HYSDEL): `MUST { u <= xu; }` (no need to add extra output)

REFERENCE/DISTURBANCE PREVIEW

Measured disturbance $v(t)$ is known M steps in advance

$$\begin{cases} x_{v,M-1}(k+1) = x_{v,M-2}(k) \\ x_{v,M-2}(k+1) = x_{v,M-3}(k) \\ \vdots \\ x_{v,1}(k+1) = x_{v,0}(k) \\ x_{v,0}(k+1) = x_{v,0}(k) \\ v(k) = x_{v,M-1}(k), \ k = 0,\dots,N-1 \end{cases}$$

initial condition

Note: same trick applies to linear MPC

$$\begin{cases} x_{v,M-1}(0) = v(t) \\ x_{v,M-2}(0) = v(t+1) \\ \vdots = \vdots \\ x_{v,1}(0) = v(t+M-2) \\ x_{v,0}(0) = v(t+M-1) \end{cases}$$

produces v= { $v(t)$,$v(t+1)$,…,$v(t+M-1)$, $v(t+M-1)$,… }

Preview of reference $r(t)$: similar.

DELAYS – METHOD 1

- Hybrid model w/ delays:

$$\begin{aligned} x(t+1) &= Ax(t) + B_1 u(t-\tau) + B_2 \delta(t) + B_3 z(t) \\ E_2\delta(t) &+ E_3 z(t) \le E_1 u(t-\tau) + E_4 x(t) + E_5 \end{aligned}$$

- Map delays to poles in $z{=}0$:

$$x_k(t) \triangleq u(t-k) \quad \Rightarrow \quad x_k(t+1) = x_{k-1}(t)$$

$$k = 1, \ldots, \tau$$

- Extend the state space of the MLD model:

$$\begin{bmatrix} x \\ x_\tau \\ x_{\tau-1} \\ \vdots \\ x_1 \end{bmatrix}(t+1) = \begin{bmatrix} A & B_1 & 0 & 0 & \ldots & 0 \\ 0 & 0 & I_m & 0 & \ldots & 0 \\ 0 & 0 & 0 & I_m & \ldots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \ldots & 0 \end{bmatrix} \begin{bmatrix} x \\ x_\tau \\ x_{\tau-1} \\ \vdots \\ x_1 \end{bmatrix}(t) + \begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ I_m \end{bmatrix} u(t) + \begin{bmatrix} B_2 \\ 0 \\ 0 \\ \vdots \\ I_m \end{bmatrix} \delta(t) + \begin{bmatrix} B_3 \\ 0 \\ 0 \\ \vdots \\ I_m \end{bmatrix} z(t)$$

- Apply MPC to the extended MLD system

Note: same trick as in linear MPC

DELAYS – METHOD 2

- Delay-free MLD model:

$$\begin{aligned} \bar{x}(t+1) &= A\bar{x}(t) + B_1 u(t) + B_2 \bar{\delta}(t) + B_3 \bar{z}(t) \\ E_2\bar{\delta}(t) &+ E_3\bar{z}(t) \leq E_1 u(t) + E_4 \bar{x}(t) + E_5 \end{aligned}$$

$$\bar{x}(t) \triangleq x(t+\tau),\ \bar{\delta}(t) \triangleq \delta(t+\tau),\ \bar{z}(t) \triangleq z(t+\tau)$$

- Design MPC for delay-free model:

$$u(t) = f_{\mathsf{MPC}}(\bar{x}(t))$$

- Compute the predicted state:

$$\bar{x}(t) = A^{\tau}x(t) + \sum_{j=0}^{\tau-1} A^{\tau-1-j}\left(B_1 u(t-1-j) + B_2\bar{\delta}(t+j) + B_3\bar{z}(t+j)\right)$$

where $\bar{\delta}(t+j), \bar{z}(t+j)$ are obtained from MLD ineq. (or HYSDEL model)

- Compute MPC action:

$$u(t) = f_{\mathsf{MPC}}(\bar{x}(t))$$

For better closed-loop performance the model used for predicting the future hybrid state $x(t+\tau)$ may be more accurate than MLD model !

CHOICE CONSTRAINTS

Sometimes, one has to make one or more choices among a certain set of alternatives. Examples:

$\delta_1 + \delta_2 + \delta_3 \leq 1$ = make at most one choice $\quad \delta_i \in \{0, 1\}$

$\delta_1 + \delta_2 + \delta_3 \leq 2$ = choose at most 2 items out of 3

$$\sum_{i=1}^{N} \delta_i \leq m \qquad \text{choose } \textbf{at most } m \text{ items out of } N$$

$$\sum_{i=1}^{N} \delta_i = m \qquad \text{choose } \textbf{exactly } m \text{ items out of } N$$

$$\sum_{i=1}^{N} \delta_i \geq m \qquad \text{choose } \textbf{at least } m \text{ items out of } N$$

"Exclusive or" constraint:

$\delta_1 + \delta_2 + \delta_3 = 1$ = make necessarily one (and only one) choice

"NO-GOOD" CONSTRAINTS

- Given a binary vector $\bar{\delta} \in \{0, 1\}^n$, we want to impose the constraint

$$\delta \neq \bar{\delta}$$

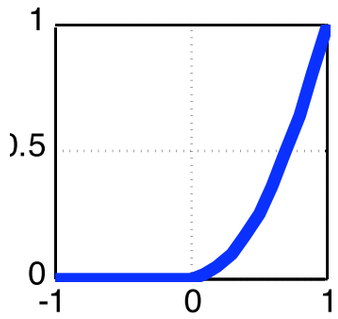
- This may be useful for example to extract different solutions from an MIP that has multiple optima

- The "no-good" condition can be expressed equivalently as

$$\sum_{i \in T} \delta_i - \sum_{i \in F} \delta_i \leq -1 + \sum_{i=1}^{n} \bar{\delta}_i \qquad \begin{aligned} F &= \{i : \bar{\delta}_i = 0\} \\ T &= \{i : \bar{\delta}_i = 1\} \end{aligned}$$

or $$\sum_{i=1}^{n} (2\bar{\delta}_i - 1)\delta_i \leq -1 + \sum_{i=1}^{n} \bar{\delta}_i$$

ASYMMETRIC WEIGHTS



- Say you want to weight a variable $u(k)$ only if $u(k) \geq 0$
- One way is to introduce a binary variable $[\delta=1] \leftrightarrow [u \geq 0]$, a continuous variable $z=u$ if $\delta=1$, $z=0$ otherwise, and weight z
- Better solution: avoid δ and set:
- In HYSDEL:

```
INTERFACE{
      INPUT{
         REAL u   [-100,100];
         REAL z   [-1, 1e3];
      ... }
IMPLEMENTATION{
      MUST{
           z >= u;
           z >= 0;
}    }
```

$$\begin{cases} \min & (\ldots) + \sum_{k=0}^{N-1} z_k^2 \\ \text{s.t.} & z_k \geq u_k \\ & z_k \geq 0 \end{cases}$$

- When ∞-norms are used, one can do the same trick:

(better way: if the MILP problem constructor can be accessed, avoid introducing z_u and just remove the constraint $\epsilon_u(k) \geq -[R]^i u(k)$ used to minimize $|Ru(k)|$, with constraint $\epsilon_u(k) \geq 0$)

$$\begin{cases} \min & (\ldots) + \sum_{k=0}^{N-1} |z_k| \\ \text{s.t.} & z_k \geq u_k \\ & z_k \geq 0 \end{cases}$$

Note: same trick applies to linear MPC

GENERAL REMARKS ABOUT MIP MODELING

The complexity of solving a mixed-integer program largely depends on the number of integer (binary) variables involved in the problem

Hence, when creating a hybrid model one has to

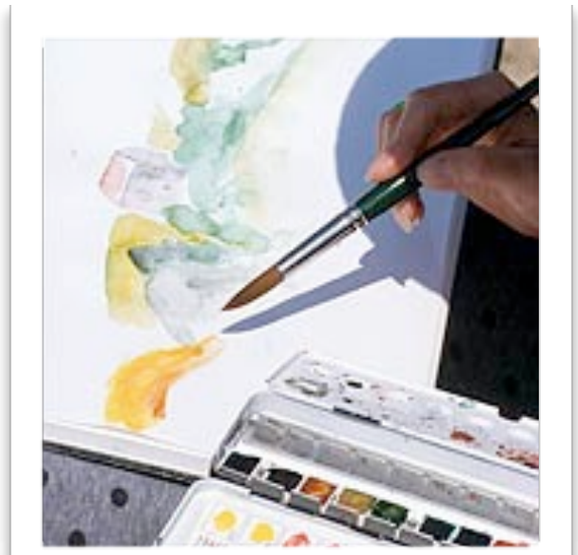
Be thrifty with binary variables !

Adding logical constraints usually helps ...

Generally speaking:

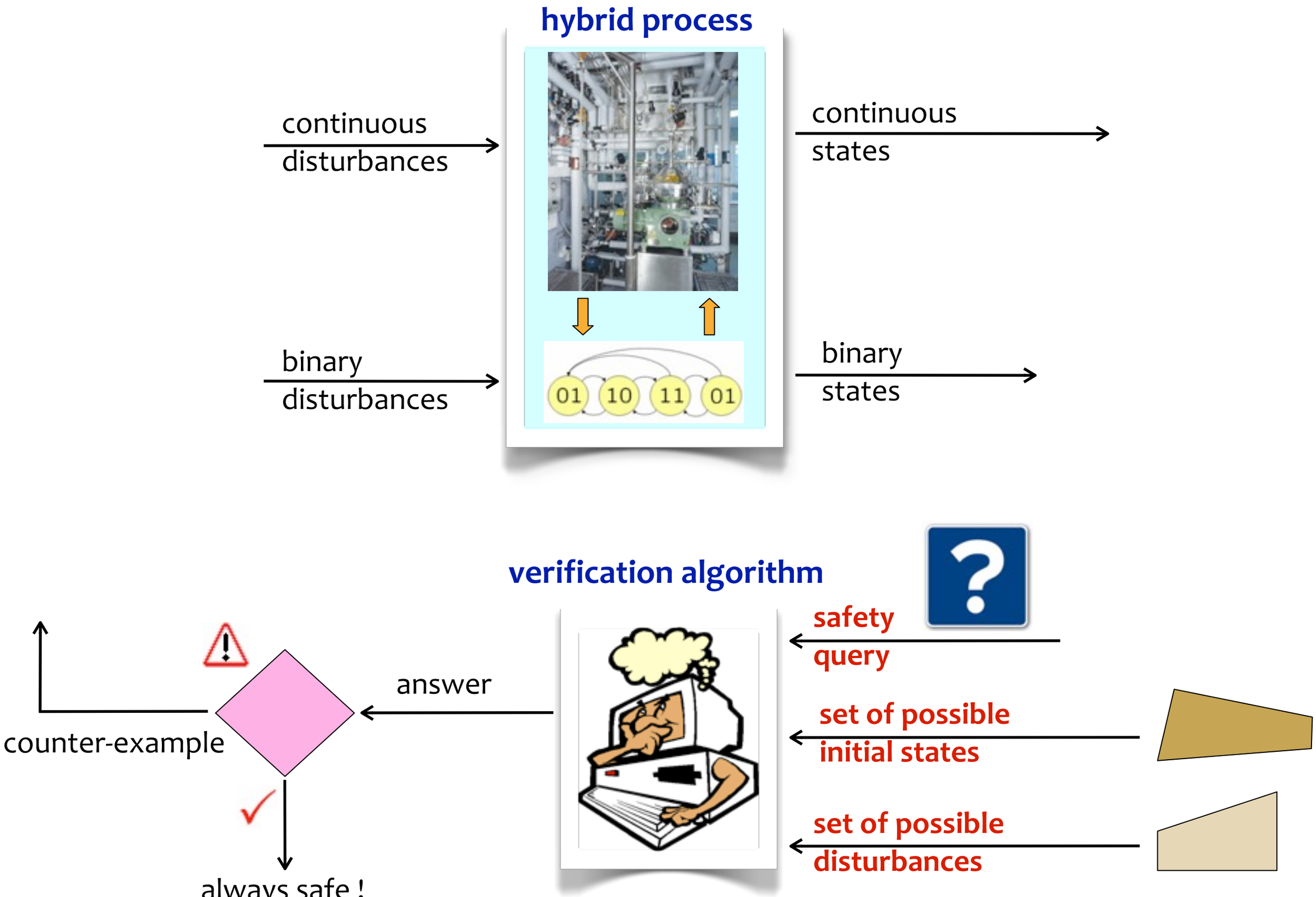
Modeling is an art

(a unifying general theory does not exist)



VERIFICATION OF SAFETY PROPERTIES (REACHABILITY ANALYSIS)

HYBRID VERIFICATION PROBLEM



VERIFICATION ALGORITHM #1

- **QUERY:** Is the target set X_f reachable after N steps from some initial state $x_0 \in X_0$ for some input profile $u \in U$?

- Computation: Solve the mixed-integer linear program (MILP)

$$\min \; 0$$

$$\text{s.t.} \begin{cases} x(k+1) = Ax(k) + B_1 u(k) + B_2 \delta(k) + B_3 z(k) \\ E_2\delta(k) + E_3 z(k) \le E_1 u(k) + E_4 x(k) + E_5 \\ S_u u(k) \le T_u \quad (u(k) \in U) \\ k = 0, 1, \ldots, N-1 \\ \\ S_0 x(0) \le T_0 \quad (x(0) \in X_0) \\ S_f x(N) \le T_f \quad (x(N) \in X_f) \end{cases}$$

with respect to $u(0)$, $\delta(0)$, $z(0)$, …, $u(N\text{-}1)$, $\delta(N\text{-}1)$, $z(N\text{-}1)$, $x(0)$

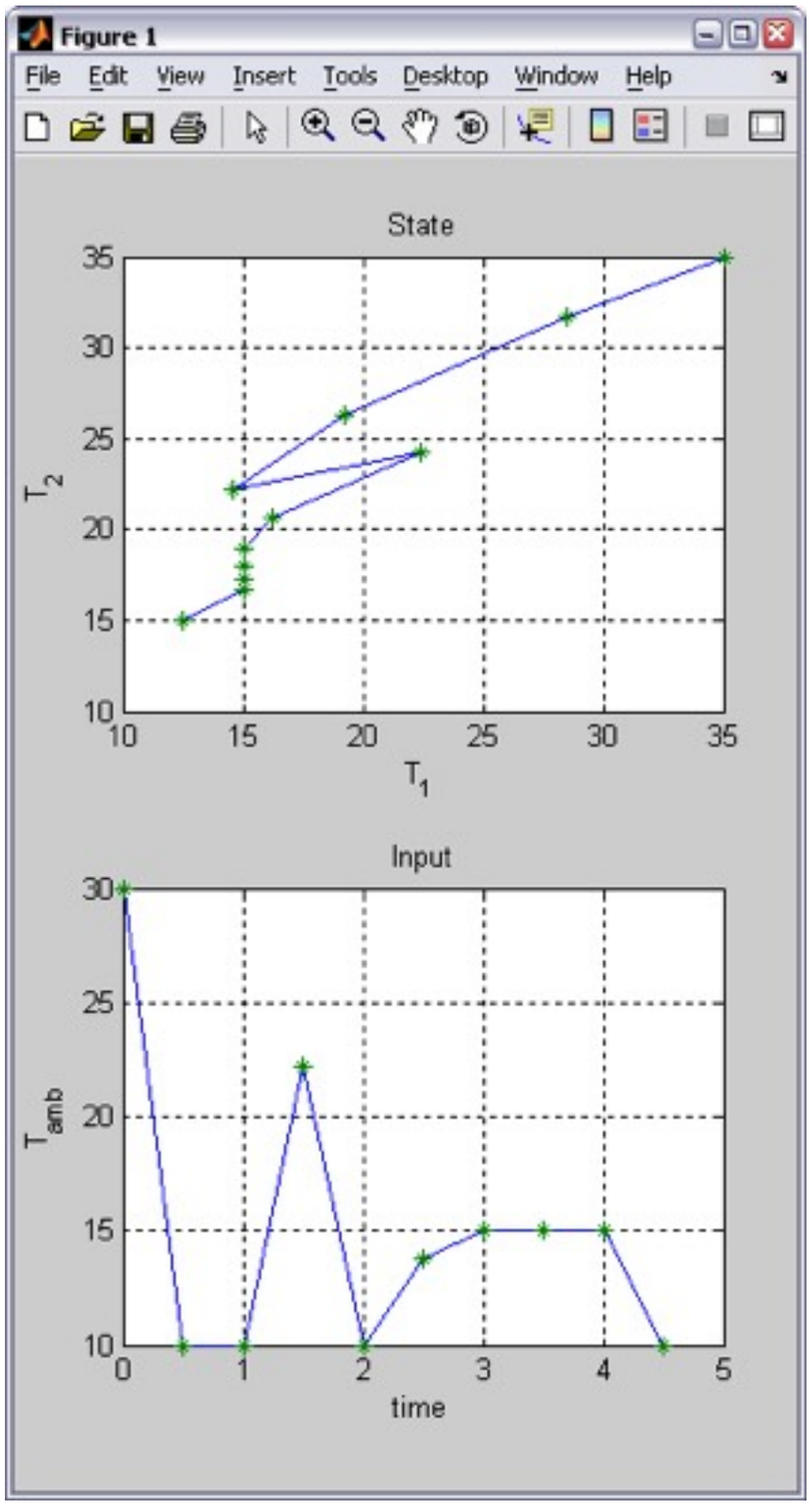
- **Alternative solutions:**
  - Exploit the special structure of the problem and use polyhedral computation. (Torrisi, 2003)
  - Use abstractions (LPs) + SAT solvers (Giorgetti, Pappas, Bemporad, 2005)

VERIFICATION EXAMPLE #1

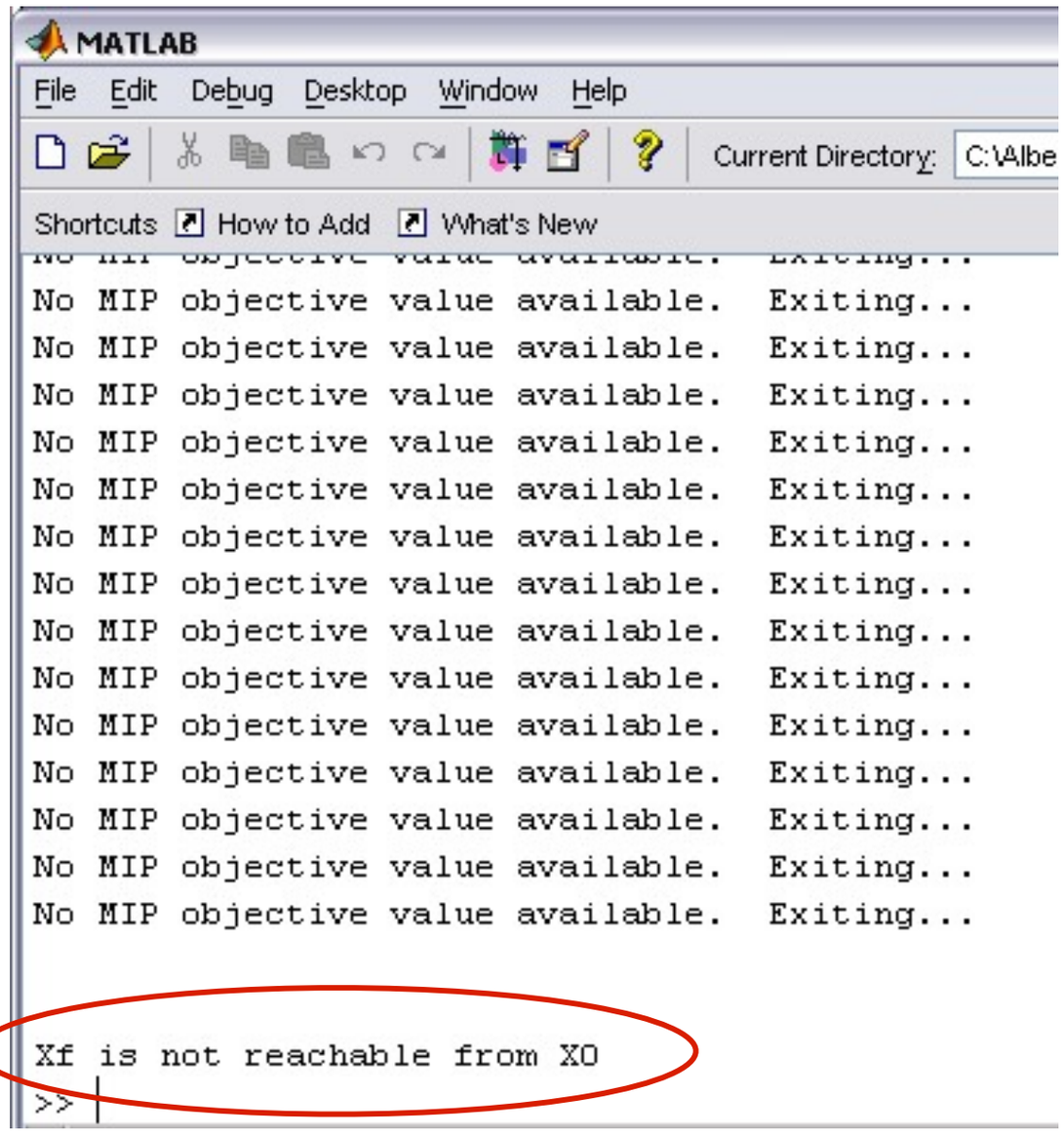
- MLD model: room temperature system
- $X_f = \{\begin{bmatrix} T_1 \\ T_2 \end{bmatrix} : 10 \leq T_1, T_2 \leq 15\}$ (set of unsafe states)
- $X_0 = \{\begin{bmatrix} T_1 \\ T_2 \end{bmatrix} : 35 \leq T_1, T_2 \leq 40\}$ (set of initial states)
- $U = \{T_{\text{amb}} : 10 \leq T_{\text{amb}} \leq 30\}$ (set of possible inputs)
- N=10 (time horizon)

```
>>[flag,x0,U]=reach(S,N,Xf,X0,umin,umax);
```

VERIFICATION EXAMPLE #1



$$U = \{T_{\text{amb}} : 10 \leq T_{\text{amb}} \leq 30\}$$



$$U = \{T_{\text{amb}} : \ 20 \leq T_{\text{amb}} \leq 30\}$$

VERIFICATION ALGORITHM #2

- **QUERY:** Is the target set X_f reachable **within** N steps from some initial state $x_0 \in X_0$ for some input profile $u \in U$?

- IDEA: Augment the MLD system to register the entrance of the target (unsafe) set $X_f = \{ x: A_f x \cdot b_f \}$:

  - Add a new variable $\delta_f(k)$, where $[\delta_f(k)=1] \rightarrow [x(k+1) \in X_f]$

  $\Longrightarrow$ big-M $\quad A_f(Ax(k)+B_1u(k)+B_2\delta(k)+B_3z(k)) \leq b_f+M(1-\delta_f(k))$

  - Add the constraint $\sum_{k=0}^{N-1} \delta_f(k) \geq 1$ ($x \in X_f$ for at least one k)

  - Solve MILP feasibility test

VERIFICATION EXAMPLE #3

- States/inputs:

$$x_1, x_2, x_3 \in \mathbb{R}, \;\; x_4, x_5 \in \{0, 1\}$$
$$u_1, u_2 \in \mathbb{R}, \;\; u_3 \in \{0, 1\}$$

- Events:

$$[\delta_1 = 1] \leftrightarrow [x_1 \leq 0]$$
$$[\delta_2 = 1] \leftrightarrow [x_2 \geq 1]$$
$$[\delta_3 = 1] \leftrightarrow [x_3 - x_2 \leq 1]$$

- Switched dynamics:

$$x_1(k+1) = \begin{cases} 0.1x_1 + 0.5x_2 & \text{if } \delta_1 \& \delta_2 | x_4 \\ -0.3x_3 - x_1 + u_1 & \text{otherwise} \end{cases}$$

$$x_2(k+1) = \begin{cases} -0.8x_1 + 0.7x_3 - u_1 - u_2 & \text{if } \delta_3 | x_5 \\ -0.3x_1 - 2x_2 & \text{otherwise} \end{cases}$$

$$x_3(k+1) = \begin{cases} -0.1x_3 + u_2 & \text{if } \delta_3 \& x_5 | x_4 \& \delta_1 \\ x_3 - 0.5x_1 - 2u_1 & \text{otherwise} \end{cases}$$

- Automaton:

$$x_4(k+1) = x_4 \& \delta_1$$
$$x_5(k+1) = ((x_4 | x_5) \& (\delta_1 | \delta_2)) | \delta_3 \& u_3$$

Query: Starting from X_0, is it possible that $x(k) \in X_f$ at some $k \leq N$, under the restriction that $x_3(k)+x_2(k) \leq 0$, $\delta_1(k) \mid \delta_2(k) \mid x_5(k)$ is true, and $\sim x_4(k) \mid x_5(k)$ is also true $\forall k \leq N$?

go to demo **`/demos/hybrid/reachtest.m`**

VERIFICATION EXAMPLE #3

- $X_f = \{ \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} : -1 \leq x_1, x_3 \leq 1,\ 0.5 \leq x_2 \leq 1,\ x_4, x_5 \in \{0,1\} \}$

  (set of unsafe states)

- $X_0 = \{ \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} : -0.1 \leq x_1, x_3 \leq 0.1,\ x_2 = 0.1,\ x_4, x_5 \in \{0,1\} \}$

  (set of initial states)

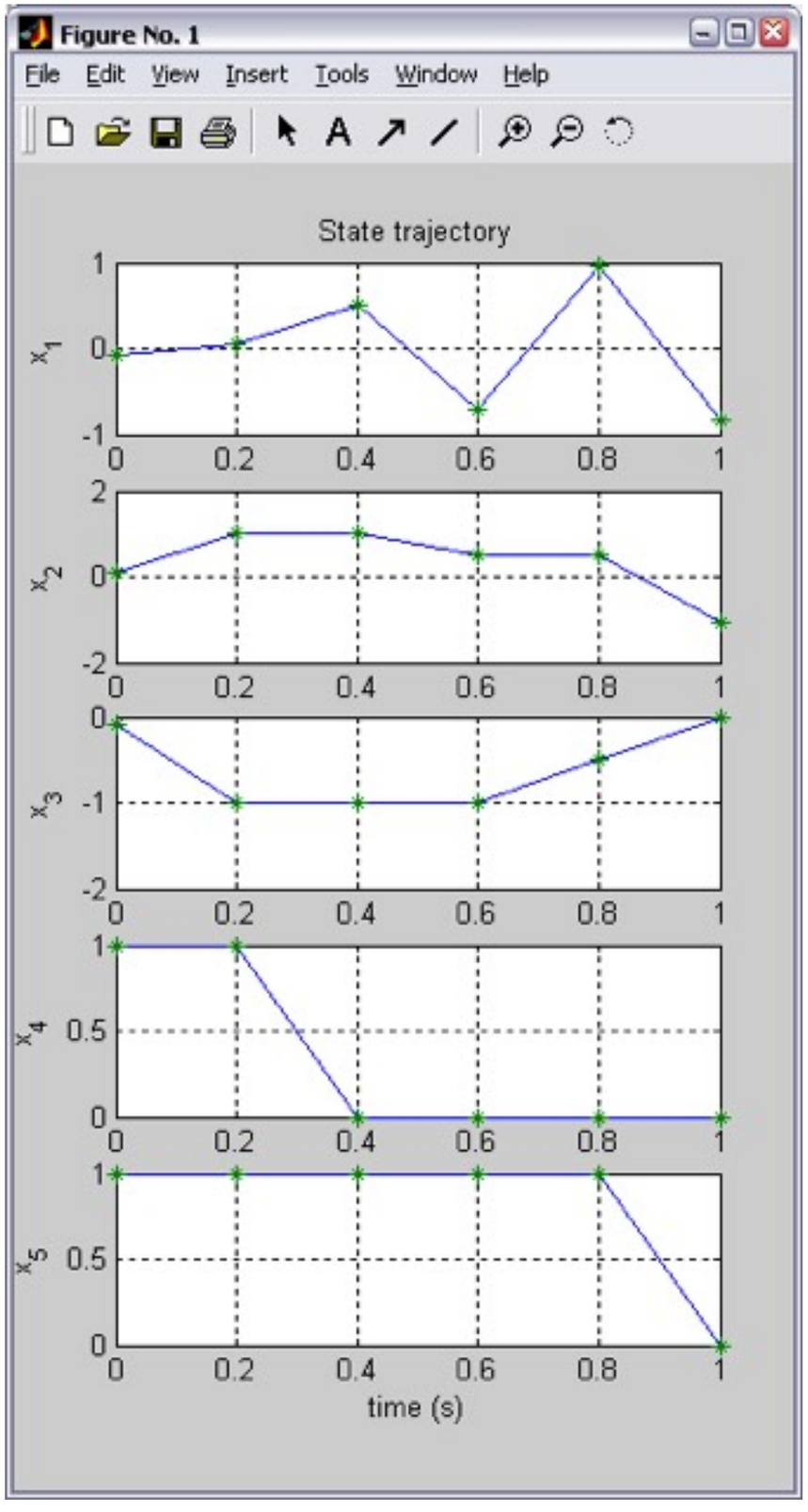
- $U = \{ \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} : -1 \leq u_1 \leq 1,\ -2 \leq u_2 \leq 2, u_3 \in \{0,1\} \}$

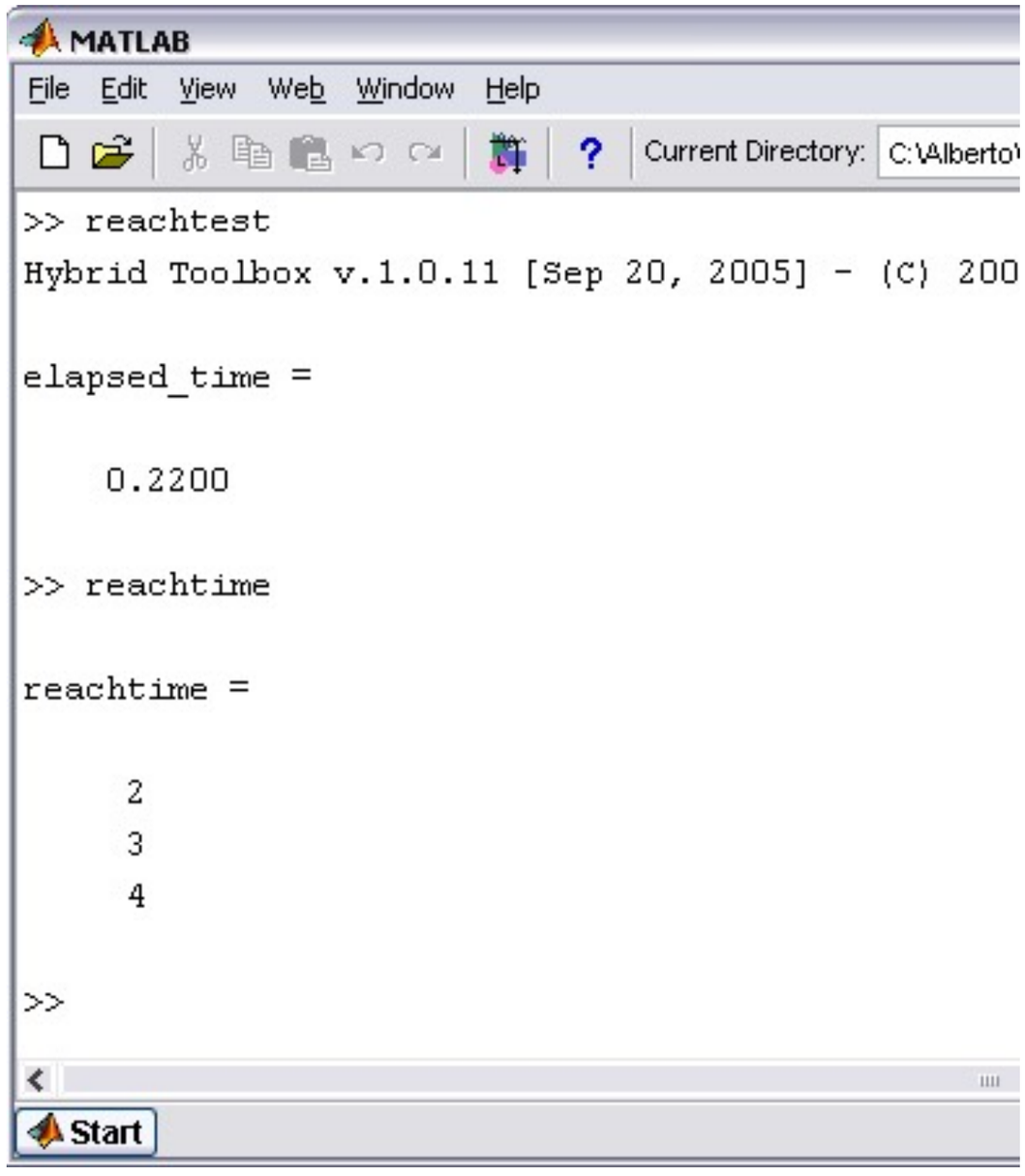
  (set of possible inputs)

- $N=5$ (time horizon)

```
>>[flag,x0,U,xf,X,T,D,Z,Y,reachtime]=reach(S,[1 N],Xf,X0);
```

VERIFICATION EXAMPLE #3





The set X_f is reached by $x(k)$ at times $k=2,3,4$