MODEL PREDICTIVE CONTROL

HYBRID MPC

Alberto Bemporad

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

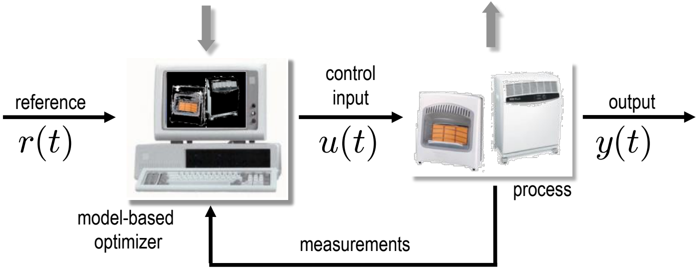
IMT SCHOOL FOR ADVANCED STUDIES LUCCA

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

HYBRID MPC

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



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

MIQP FORMULATION OF HYBRID MPC

(Bemporad, Morari, 1999)

- Finite-horizon optimal control problem (regulation)

$$
\begin{array}{ll}
\min & \displaystyle\sum_{k=0}^{N-1} y_k' Q y_k + u_k' R u_k \\
\text{s.t.} & \left\{\begin{array}{rcl}
x_{k+1} & = & A x_k + B_1 u_k + B_2 \delta_k + B_3 z_k + B_5 \\
y_k & = & C x_k + D_1 u_k + D_2 \delta_k + D_3 z_k + D_5 \\
E_2 \delta_k & + & E_3 z_k \leq E_4 x_k + E_1 u_k + E_5 \\
x_0 & = & x(t)
\end{array}\right.
\end{array}
$$

$Q = Q' \succ 0, R = R' \succ 0$

- Treat u_k, δ_k, z_k as free decision variables, $k = 0, \ldots, N-1$

- Predictions can be constructed **exactly as in the linear case**

$$
x_k = A^k x_0 + \sum_{j=0}^{k-1} A^j (B_1 u_{k-1-j} + B_2 \delta_{k-1-j} + B_3 z_{k-1-j} + B_5)
$$

MIQP FORMULATION OF HYBRID MPC

(Bemporad, Morari, 1999)

- After substituting x_k, y_k the resulting optimization problem becomes the following **Mixed-Integer Quadratic Programming (MIQP)** problem

$$\begin{array}{rl} \min_\xi & \frac{1}{2}\xi' H\xi + x'(t)F'\xi + \frac{1}{2}x'(t)Yx(t) \\ \text{s.t.} & G\xi \leq W + Sx(t) \end{array}$$

- The optimization vector $\xi = [u_0, \ldots, u_{N-1}, \delta_0, \ldots, \delta_{N-1}, z_0, \ldots, z_{N-1}]$ has **mixed real and binary** components

$$\begin{array}{l} u_k \in \mathbb{R}^{m_c} \times \{0,1\}^{m_b} \\ \delta_k \in \{0,1\}^{r_b} \\ z_k \in \mathbb{R}^{r_c} \end{array} \quad \Rightarrow \quad \xi \in \mathbb{R}^{N(m_c+r_c)} \times \{0,1\}^{N(m_b+r_b)}$$

HYBRID MPC FOR REFERENCE TRACKING

- Consider the more general set-point tracking problem

$$
\begin{array}{ll}
\min_{\xi} & \displaystyle\sum_{k=0}^{N-1} \|y_k - r\|_Q^2 + \|u_k - u_r\|_R^2 \\
& +\sigma\left(\|x_k - x_r\|_2^2 + \|\delta_k - \delta_r\|_2^2 + \|z_k - z_r\|_2^2\right) \\
\\
\text{s.t.} & \text{MLD model equations} \\
& x_0 = x(t) \\
& x_N = x_r
\end{array}
$$

with $\sigma > 0$ and $\|v\|_Q^2 = v'Qv$

- The equilibrium $(x_r, u_r, \delta_r, z_r)$ corresponding to r can be obtained by solving the following mixed-integer feasibility problem

$$
\begin{array}{rcl}
x_r & = & Ax_r + B_1 u_r + B_2\delta_r + B_3 z_r + B_5 \\
r & = & Cx_r + D_1 u_r + D_2\delta_r + D_3 z_r + D_5 \\
E_2\delta_r & + & E_3 z_r \leq E_4 x_r + E_1 u_r + E_5
\end{array}
$$

CLOSED-LOOP CONVERGENCE

(Bemporad, Morari, 1999)

- **Theorem.** Let $(x_r, u_r, \delta_r, z_r)$ be the equilibrium corresponding to r. Assume $x(0)$ such that the MIQP problem **is feasible at time** $t = 0$. Then $\forall Q, R \succ 0, \sigma > 0$ the hybrid MPC closed-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 $t \geq 0$.

- The proof easily follows from standard Lyapunov arguments (see next slide)

- **Lyapunov asymptotic stability** and **exponential stability** follows if proper terminal cost and constraints are imposed (Lazar, Heemels, Weiland, Bemporad, 2006)

CONVERGENCE PROOF

- **Main idea**: Use the value function $V^*(x(t))$ as a Lyapunov function
- Let $\xi_t = [u_0^t, \ldots, u_{N-1}^t, \delta_0^t, \ldots, \delta_{N-1}^t, z_0^t, \ldots, z_{N-1}^t]$ be the optimal sequence @t
- By construction @t+1 $\bar{\xi} = [u_1^t, \ldots, u_{N-1}^t, u_r, \delta_1^t, \ldots, \delta_{N-1}^t, \delta_r, z_0^t, \ldots, z_{N-1}^t, z_r]$ is feasible, as it satisfies all MLD constraints + terminal constraint $x_N = x_r$
- The cost of $\bar{\xi}$ is $V^*(x(t)) - \|y(t) - r\|_Q^2 - \|u(t) - u_r\|_R^2$

$$-\sigma \left(\|\delta(t) - \delta_r\|_2^2 + \|z(t) - z_r\|_2^2 + \|x(t) - x_r\|_2^2\right) \geq V^*(x(t+1))$$

- $V^*(x(t))$ is monotonically decreasing and ≥ 0, so $\exists \lim_{t\to\infty} V^*(x(t)) \in \mathbb{R}$
- Hence $\|y(t) - r\|_Q^2, \|u(t) - u_r\|_R^2, \|\delta(t) - \delta_r\|_2^2, \|z(t) - z_r\|_2^2, \|x(t) - x_r\|_2^2 \to 0$
- Since $R, Q \succ 0$, $\lim_{t\to\infty} y(t) = r$ and all other variables converge. $\square$

Global optimum is not needed to prove convergence !

MILP FORMULATION OF HYBRID MPC

(Bemporad, Borrelli, Morari, 2000)

- Finite-horizon optimal control problem using infinity norms

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

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

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

- Introduce additional variables $\epsilon_k^y, \epsilon_k^u, k = 0, \ldots, N-1$

$$\begin{cases} \epsilon_k^y \geq \|Qy_k\|_\infty \\ \epsilon_k^u \geq \|Ru_k\|_\infty \end{cases} \Rightarrow \begin{cases} \epsilon_k^y \geq \pm Q^i y_k \\ \epsilon_k^u \geq \pm R^i u_k \end{cases}$$

Q^i = ith row of Q

MILP FORMULATION OF HYBRID MPC

(Bemporad, Borrelli, Morari, 2000)

- After substituting x_k, y_k the resulting optimization problem becomes the following **Mixed-Integer Linear Programming (MILP)** problem

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

- $\xi = [u_0, \ldots, u_{N-1}, \delta_0, \ldots, \delta_{N-1}, z_0, \ldots, z_{N-1}, \epsilon_0^y, \epsilon_0^u, \ldots, \epsilon_{N-1}^y, \epsilon_{N-1}^u]$ is the optimization vector, with **mixed real and binary** components

$$\begin{aligned} u_k &\in \mathbb{R}^{m_c} \times \{0,1\}^{m_b} \\ \delta_k &\in \{0,1\}^{r_b} \\ z_k &\in \mathbb{R}^{r_c} \\ \epsilon_k^y, \epsilon_k^u &\in \mathbb{R} \end{aligned} \quad \Longrightarrow \quad \xi \in \mathbb{R}^{N(m_c+r_c+2)} \times \{0,1\}^{N(m_b+r_b)}$$

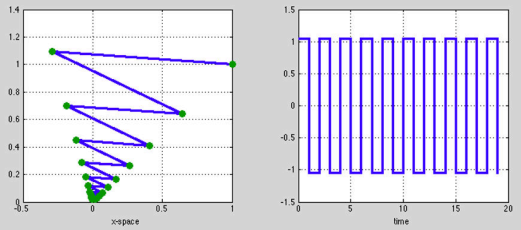
- Same approach applies to any **convex piecewise affine** stage cost

HYBRID MPC EXAMPLE

- PWA system:

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

- Open-loop simulation:



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

HYBRID MPC EXAMPLE

```
SYSTEM pwa {

INTERFACE {
    STATE { REAL x1 [-10,10];
            REAL x2 [-10,10]; }
    INPUT { REAL u [-1.1,1.1]; }
    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; }
}
```

$[\mathbf{sign}=1] \leftrightarrow [x_1 \geq 0]$

$x_1(t+1) = z_1(t)$

$x_2(t+1) = z_2(t) + u(t)$

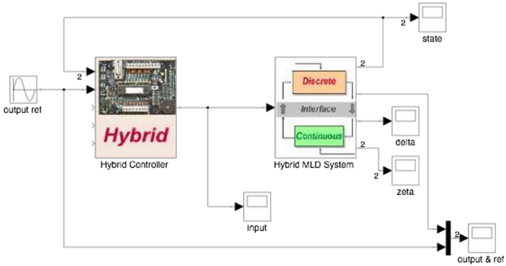
$y(t) = x_2(t)$

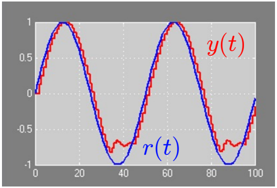
go to `demos/hybrid/bm99.hys`

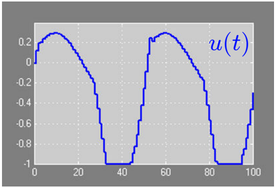
HYBRID MPC EXAMPLE

- Closed-loop MPC results:

$$\begin{array}{ll} \min & \sum_{k=1}^{2} |y_k - r(t)| \\ \text{s.t.} & -1 \leq u_k \leq 1,\ i = 0, 1 \end{array}$$







- Average CPU time to solve MILP: $\approx$ 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];
```

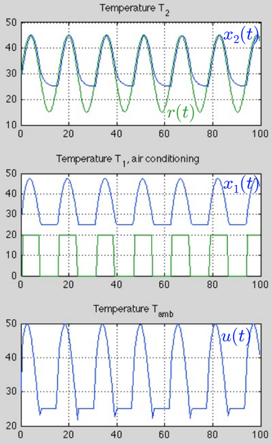
```
>> C=hybcon(S,Q,N,limits,refs);
```

```
>> C

Hybrid controller based on MLD model S <heatcoolmodel.hys>

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.
>>
```

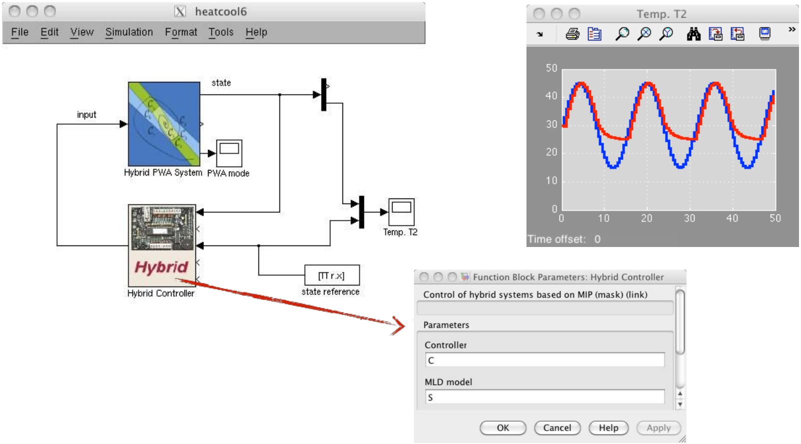
```
>> [XX,UU,DD,ZZ,TT]=sim(C,S,r,x0,Tstop);
```

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

$$\text{s.t.} \begin{cases} x_{1k} \geq 25, \; k = 1, 2 \\ \text{MLD model} \end{cases}$$



HYBRID MPC – TEMPERATURE CONTROL



- Average CPU time to solve MILP: $\approx$ 1 ms/step

  (Macbook Pro 3GHz Intel Core i7 using GLPK)

MIXED-INTEGER PROGRAMMING SOLVERS

- Binary constraints make Mixed-Integer Programming (MIP) a hard problem ($\mathcal{NP}$-complete)

- However, excellent general purpose **branch & bound** / **branch & cut** solvers available for MILP and MIQP (Gurobi, CPLEX, FICO Xpress, GLPK, CBC, ...)

  (more solvers/benchmarks: see `http://plato.la.asu.edu/bench.html`)

- MIQP approaches tailored to embedded hybrid MPC applications:

  - B&B + (dual) active set methods for QP

    (Leyffer, Fletcher, 1998) (Axehill, Hansson, 2006) (Bemporad, 2015) (Bemporad, Naik, 2018)

  - B&B + interior point methods: (Frick, Domahidi, Morari, 2015)
  - B&B + fast gradient projection: (Naik, Bemporad, 2017)
  - B&B + ADMM: (Stellato, Naik, Bemporad, Goulart, Boyd, 2018)

- No need to reach global optimum (see convergence proof), although performance may deteriorate

BRANCH & BOUND METHOD FOR MIQP

(Dakin, 1965)

- We want to solve the following MIQP

$$\begin{array}{rl} \min & V(z) \triangleq \frac{1}{2}z'Qz + c'z \\ \text{s.t.} & Az \leq b \\ & z_i \in \{0,1\},\ \forall i \in I \end{array} \qquad \begin{array}{l} z \in \mathbb{R}^n \\ Q = Q' \succeq 0 \\ I \subseteq \{1,\ldots,n\} \end{array}$$

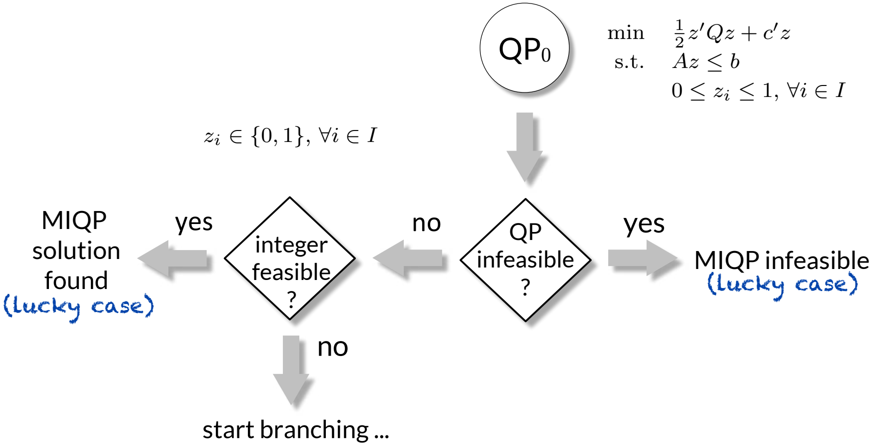
- **Branch & Bound (B&B)** is the simplest (and most popular) approach to solve the problem to optimality

- **Key idea**:
  - for each binary variable $z_i, i \in I$, either set $z_i = 0$, or $z_i = 1$, or $z_i \in [0,1]$
  - solve the corresponding **QP relaxation** of the MIQP problem
  - use QP result to decide the next combination of fixed/relaxed variables

BRANCH & BOUND METHOD FOR MIQP

QP relaxation

$$
\begin{array}{ll}
\min & \frac{1}{2}z'Qz + c'z \\
\text{s.t.} & Az \leq b \\
 & 0 \leq z_i \leq 1, \ \forall i \in I
\end{array}
$$

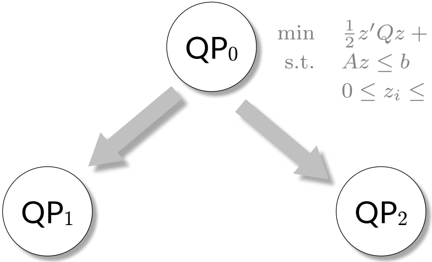


BRANCH & BOUND METHOD FOR MIQP

- **Branching rule**: pick the index i such that z_i is closest to $\frac{1}{2}$ (max fractional part)

  (Breu, Burdet, 1974)

- Solve two new QP relaxations



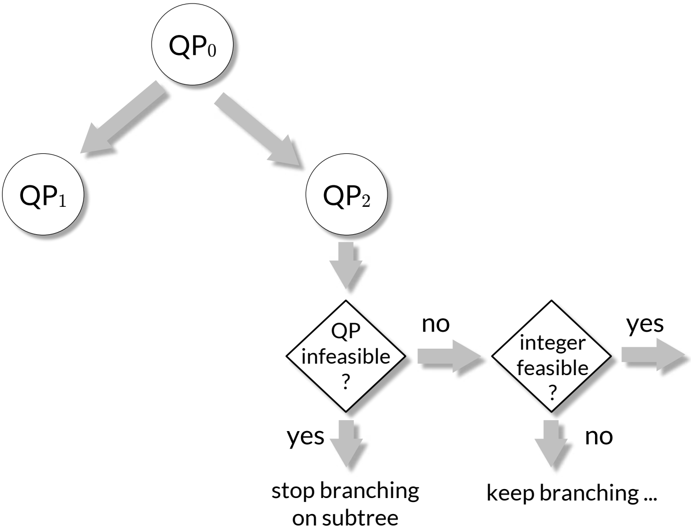
$$\begin{array}{ll} \min & \frac{1}{2}z'Qz + c'z \\ \text{s.t.} & Az \leq b \\ & 0 \leq z_i \leq 1,\ \forall i \in I \end{array}$$

$$\begin{array}{ll} \min & \frac{1}{2}z'Qz + c'z \\ \text{s.t.} & Az \leq b \\ & z_i = 0 \\ & 0 \leq z_j \leq 1,\ \forall j \in I,\ j \neq i \end{array}$$

$$\begin{array}{ll} \min & \frac{1}{2}z'Qz + c'z \\ \text{s.t.} & Az \leq b \\ & z_i = 1 \\ & 0 \leq z_j \leq 1,\ \forall j \in I,\ j \neq i \end{array}$$

- Possibly exploit **warm starting** from QP_0 when solving new relaxations QP_1 and QP_2

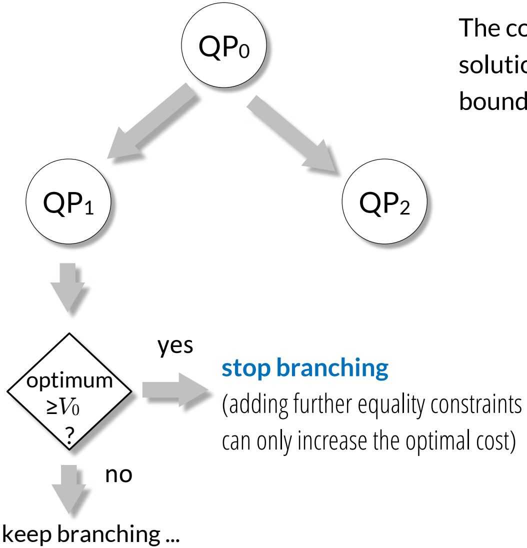
BRANCH & BOUND METHOD FOR MIQP



update upper bound $V_0 \geq V^*$ **on MIQP solution**

BRANCH & BOUND METHOD FOR MIQP

The cost V_0 of the best integer-feasible solution found so fare gives an upper bound $V_0 \geq V^*$ on MIQP solution



BRANCH & BOUND METHOD FOR MIQP

- While solving the QP relaxation, if the **dual cost** is available it gives a **lower bound** to the solution of the relaxed problem

- The QP solver can be stopped whenever the dual cost $\geq V_0$!

  This may save a lot of computations

- When no further branching is possible, either the MIQP problem is declared infeasible or an optimal solution z^* has been found

SOLVING MIQP VIA NNLS

(Bemporad, 2015)

- B&B method + QP solver based on **nonnegative least squares** applied to solving the MIQP

$$\begin{array}{ll} \min\limits_{z} & V(z) \triangleq \frac{1}{2} z' Q z + c' z \\ \text{s.t.} & \ell \leq Az \leq u \\ & Gz = g \\ & \bar{A}_i z \in \{\bar{\ell}_i, \bar{u}_i\},\ i = 1, \ldots, q \end{array} \qquad Q = Q' \succ 0$$

- Binary constraints on z are a special case: $\bar{\ell}_i = 0, \bar{u}_i = 1$, $\bar{A}_i = [0 \ldots 0\,1\,0 \ldots 0]$

- Warm starting from parent node exploited when solving new QP relaxation

- QP solver interrupted when dual cost larger than best known upper-bound

SOLVING MIQP VIA NNLS

- **Worst-case** CPU time (ms) 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 = # inequalities
q = # binary vars
(no equalities)

Compiled Embedded MATLAB code (QP solver) + MATLAB code (B&B)
CPU results measured on Macbook Pro 3GHz Intel Core i7

NNLS-LDL = recursive LDL' factorization used to solve least-square problems in QP solver

NNLS-QR = recursive QR factorization used instead (numerically more robust)

SOLVING MIQP VIA NNLS

- **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

N = prediction horizon

MIQP regularized to make Q strictly $\succ 0$ (solution difference is negligible)

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

SOLVING MIQP VIA NNLS AND PROXIMAL-POINT ITERATIONS

(Bemporad, Naik, 2018)

- **Robustified approach**: use **NNLS + proximal-point iterations** to solve QP relaxations (Bemporad, 2018)

$$
\begin{array}{rl}
z_{k+1} = \arg\min_z & \frac{1}{2}z'Qz + c'z + \frac{\epsilon}{2}\|z - z_k\|_2^2 \\
\text{s.t.} & \ell \leq Az \leq u \\
 & Gz = g
\end{array}
$$

- CPU time (ms) on **MIQP** coming from hybrid MPC (`bm99` demo):

For $N = 10$:
30 real vars
10 binary vars
160 inequalities

prox-NNLS* = warm start of binary vars exploited

N	prox-NNLS		prox-NNLS*		GUROBI		CPLEX	
	avg	max	avg	max	avg	max	avg	max
2	2.0	2.6	2.0	2.6	1.6	2.0	3.1	6.0
4	5.3	8.8	3.1	6.9	3.1	3.9	8.9	15.7
8	29.7	71.0	8.1	43.4	7.2	13.2	15.5	80.2
10	76.2	146.1	14.4	103.2	11.1	17.6	35.1	95.3
12	155.8	410.8	26.9	263.4	14.9	31.2	61.7	103.7
15	484.2	1242.3	61.7	766.9	25.9	109.8	89.9	181.1

CPU time measured on Intel Core i7-4700MQ CPU 2.40 GHz

FAST GRADIENT PROJECTION FOR MIQP

(Naik, Bemporad, 2017)

- Consider again the MIQP problem with Hessian $Q = Q' \succ 0$

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

$$\begin{aligned} w^k &= y^k + \beta_k (y^k - y^{k-1}) \\ z^k &= -Kw^k - Jx \\ s^k &= \ldots \\ y_i^{k+1} &= \max\left\{w_i^k + s_i^k, 0\right\},\ i \in I_{\mathrm{ineq}} \end{aligned}$$

- Use B&B and **fast gradient projection** to solve dual of QP relaxation

constraint is relaxed	$\bar{A}_i z \leq \bar{u}_i$	$\rightarrow$	$y_i^{k+1} = \max\left\{w_i^k + s_i^k, 0\right\}$	$(y_i \geq 0)$
constraint is fixed	$\bar{A}_i z = \bar{u}_i$	$\rightarrow$	$y_i^{k+1} = w_i^k + s_i^k$	$(y_i \lesseqgtr 0)$
constraint is ignored	$\bar{A}_i z = \bar{\ell}_i$	$\rightarrow$	$y_i^{k+1} = 0$	$(y_i = 0)$

FAST GRADIENT PROJECTION FOR MIQP

(Naik, Bemporad, 2017)

- **Same dual QP matrices** at each node, **preconditioning** computed only once
- **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

n = # variables
m = # inequality constraints
p = # binary constraints
q = # equality constraints

CPU time measured on Intel Core i7-4700MQ CPU 2.40 GHz

MIQP AND ADMM

- B&B + ADMM: solve QP relaxations via ADMM

  (Stellato, Naik, Bemporad, Goulart, Boyd, 2018)

$$\begin{array}{ll} \min & \frac{1}{2}x'Qx + c'x \\ \text{s.t.} & \ell \leq Ax \leq u \\ & A_i x \in \{\ell_i, u_i\},\ i \in I \end{array}$$

- Simpler **heuristic** approach: only perform one set of ADMM iterations

  (Takapoui, Moehle, Boyd, Bemporad, 2017)

quantization step

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

- Iterations converge to a (local) solution

- Similar heuristic idea also applicable to fast gradient methods (Naik, Bemporad, 2017)

(Takapoui, Moehle, Boyd, Bemporad, 2017)

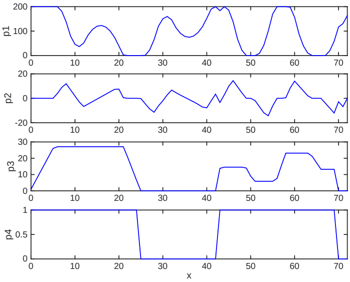
- **Example**: parallel hybrid electric vehicle control problem

engine power

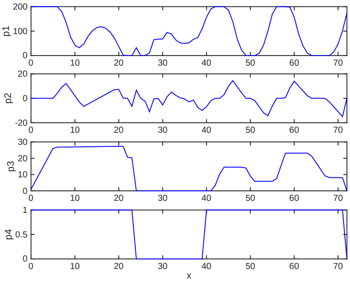
electrical power

energy stored
in battery

engine on/off



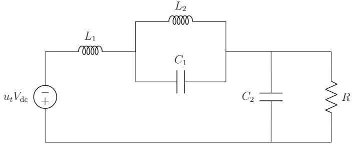
optimal solution



ADMM solution

(Takapoui, Moehle, Boyd, Bemporad, 2017)

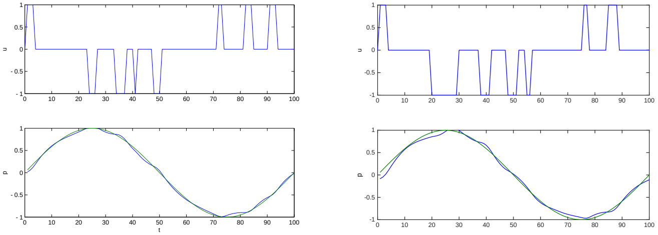
- **Example**: power converter control problem



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

input voltage sign u_t

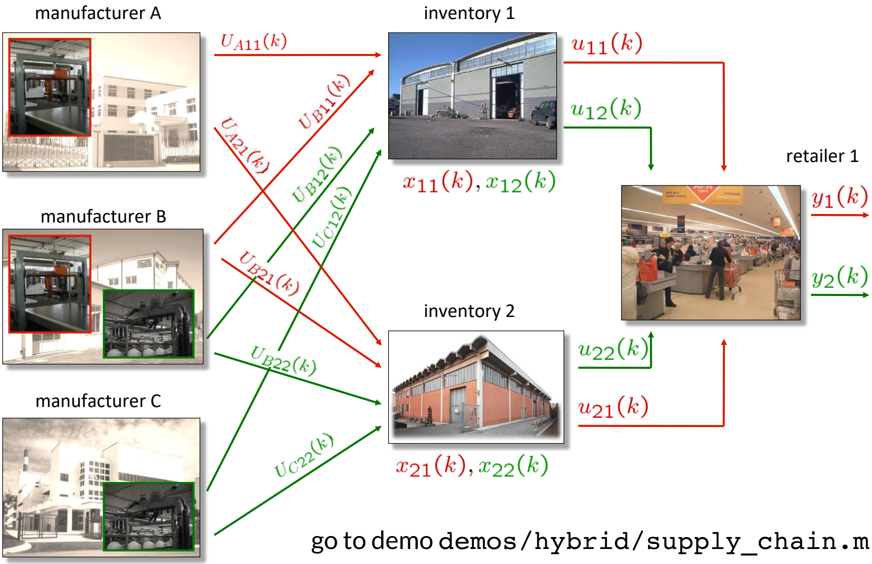
output voltage v_2



optimal solution

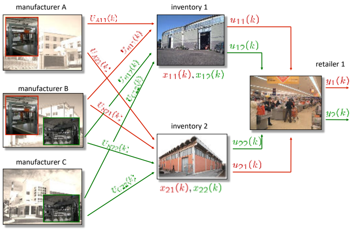
ADMM solution

A SIMPLE EXAMPLE IN SUPPLY CHAIN MANAGEMENT



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

SUPPLY CHAIN MANAGEMENT - SYSTEM VARIABLES

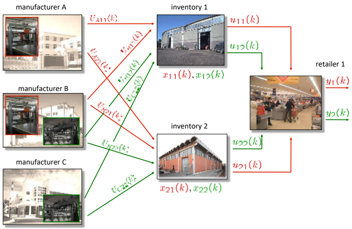


- **Continuous states**:
  $x_{ij}(k)$ = amount of j hold in inventory i
  at time k ($i = 1, 2, j = 1, 2$)

- **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



- Max capacity of inventory i:

$$0 \leq \sum_{j=1}^{2} x_{ij} \leq x_{Mi}$$

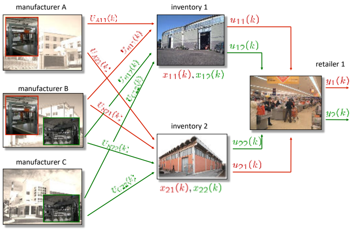
- Max transportation from inventories:

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

- A product can only be sent to one inventory:

  $U_{A11}(k)$ and $U_{A21}(k)$ cannot be both = 1
  $U_{B11}(k)$ and $U_{B21}(k)$ cannot be both = 1
  $U_{B12}(k)$ and $U_{B22}(k)$ cannot be both = 1
  $U_{C12}(k)$ and $U_{C22}(k)$ cannot be both = 1

- A manufacturer can only produce one type of product at one time:
  $[U_{B11}(k)$ or $U_{B21}(k) = 1]$, $[U_{B12}(k)$ or $U_{B22}(k) = 1]$ cannot be both true



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

- Level of inventories

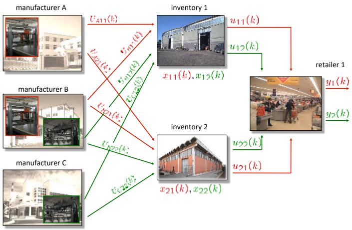
$$\begin{cases} x_{11}(k+1) &= x_{11}(k) + P_{A1}U_{A11}(k) + P_{B1}U_{B11}(k) - u_{11}(k) \\ x_{12}(k+1) &= x_{12}(k) + P_{B2}U_{B12}(k) + P_{C2}U_{C12}(k) - u_{12}(k) \\ x_{21}(k+1) &= x_{21}(k) + P_{A1}U_{A21}(k) + P_{B1}U_{B21}(k) - u_{21}(k) \\ x_{22}(k+1) &= x_{22}(k) + P_{B2}U_{B22}(k) + P_{C2}U_{C22}(k)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

- Retailer: all items requested from inventories are sold

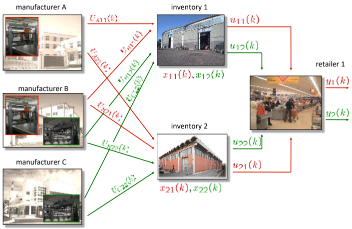
$$\begin{cases} y_1 &= u_{11} + u_{21} \\ y_2 &= u_{12} + u_{22} \end{cases}$$

SUPPLY CHAIN MANAGEMENT - HYSDEL CODE

```
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; }
} }
```



SUPPLY CHAIN MANAGEMENT - OBJECTIVES



- Meet customer demand as much as possible:

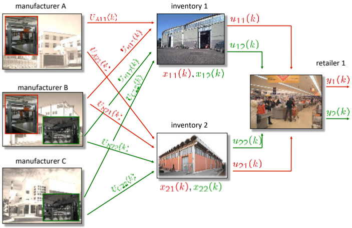
$$y_1 \approx r_1, \quad y_2 \approx r_2$$

- Minimize transportation costs

- Fulfill all constraints

$$
\begin{aligned}
\min \sum_{k=0}^{N-1} \quad & \overbrace{10(|y_{1,k} - r_1(t)| + |y_{2,k} - r_2(t)|}^{\text{penalty on demand tracking error}} + \\
& \overbrace{4(|u_{11,k}| + |u_{12,k}|)}^{\text{shipping cost from inv. 1 to market}} \qquad + \\
& \overbrace{2(|u_{21,k}| + |u_{22,k}|)}^{\text{shipping cost from inv. 2 to market}} \qquad + \\
& \overbrace{1(|U_{A11,k}| + |U_{A21,k}|)}^{\text{cost from A to inventories}} + \\
& \overbrace{4(|U_{B11,k}| + |U_{B12,k}| + U_{B21,k}| + |U_{B22,k}|)}^{\text{cost from B to inventories}} + \\
& \overbrace{10(|U_{C12,k}| + |U_{C22,k}|)}^{\text{cost from C to inventories}}
\end{aligned}
$$

```
>> 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)
```

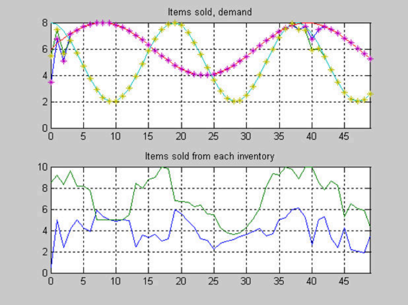


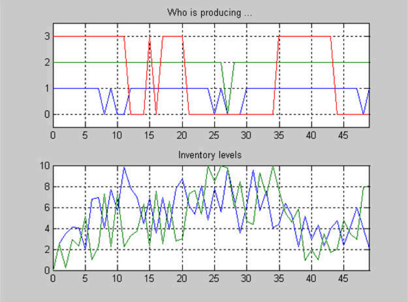

```
>> C=hybcon(S,Q,N,limits,refs);
```

```
 >> 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) (8 continuous, 12 binary)
176 mixed-integer linear inequalities
sampling time = 1, MILP solver = 'glpk'

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

```
>> 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)];
```

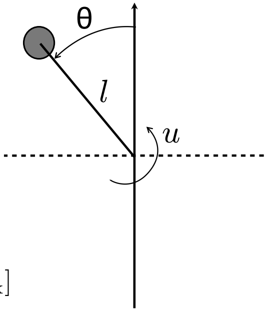
```
>> [XX,UU,DD,ZZ,TT]=sim(C,S,r,x0,Tstop);
```





CPU time: ≈ 13 ms/sample (GLPK) or 9 ms (CPLEX) on Macbook Pro 3GHz Intel Core i7

HYBRID MPC OF AN INVERTED PENDULUM



- **Goal**: swing the pendulum up

- **Non-convex** input constraint

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

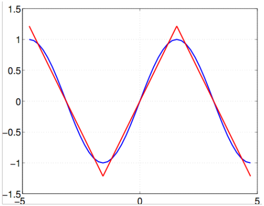
- **Nonlinear** dynamical model

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

INVERTED PENDULUM: NONLINEARITY

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

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



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

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

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

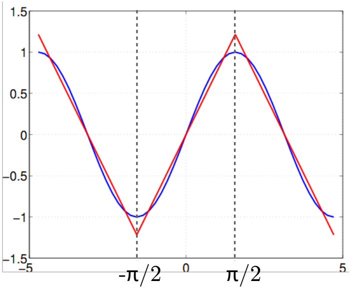
- Zeroing the derivative with respect to α gives $\alpha=\frac{24}{\pi^3}$

- Requiring $s=0$ for $\theta=\pi$ gives $\gamma=\frac{24}{\pi^2}$

INVERTED PENDULUM: NONLINEARITY

- Introduce the event variables

$$[\delta_3 = 1] \quad \leftrightarrow \quad [\theta \le -\tfrac{\pi}{2}]$$
$$[\delta_4 = 1] \quad \leftrightarrow \quad [\theta \ge \tfrac{\pi}{2}]$$



along with the logic constraint

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

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

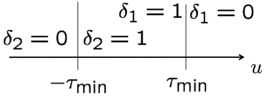
$$\tau_A = \begin{cases} u & \text{if } -\tau_{\min} \leq u \leq \tau_{\min} \\ 0 & \text{otherwise} \end{cases}$$

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

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

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

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

- Set $x \triangleq \left[\begin{smallmatrix}\theta \\ \dot{\theta}\end{smallmatrix}\right], y \triangleq \theta$ and transform into linear model

$$\begin{aligned}
\frac{d}{dt}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} &= \begin{bmatrix} 0 & 1 \\ \frac{g}{l}\alpha & -\frac{\beta}{l^2 M} \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ \frac{g}{l} & \frac{1}{l^2 M} \end{bmatrix}\begin{bmatrix} s_3 + s_4 \\ u - \tau_A \end{bmatrix} \\
y &= \begin{bmatrix} 1 & 0 \end{bmatrix} x
\end{aligned}$$

- Discretize in time with sample time $T_s = 50$ ms

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

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

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`

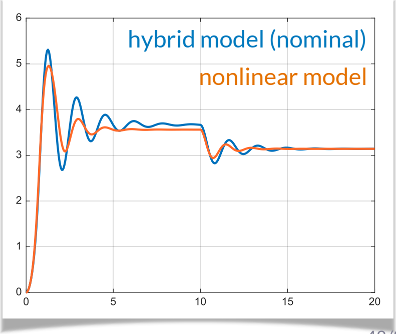
INVERTED PENDULUM: MODEL VALIDATION

- Open-loop simulation from initial condition $\theta(0) = 0, \dot{\theta}(0) = 0$
- Input torque excitation

$$u(t) = \begin{cases} 2\,\mathrm{Nm} & \text{if } 0 \leq t \leq 10\,\mathrm{s} \\ 0 & \text{otherwise} \end{cases}$$

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

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



INVERTED PENDULUM: MPC DESIGN

- MPC cost function

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

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

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

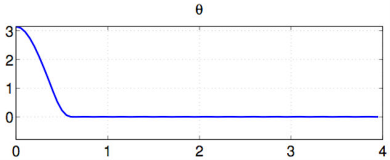
```
>> C

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.
>>
```

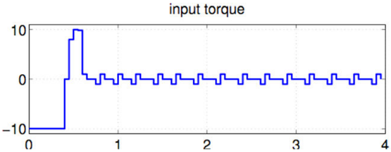
```
>> refs.y=1;
>> refs.u=1;
>> Q.y=1;
>> Q.y=0.01;
>> Q.rho=Inf;
>> Q.norm=Inf;
>> N=5;
>> limits.umin=-10;
>> limits.umax=10;
```

INVERTED PENDULUM: CLOSED-LOOP RESULTS

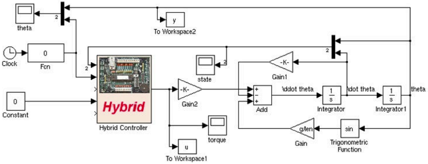
- Nominal simulation

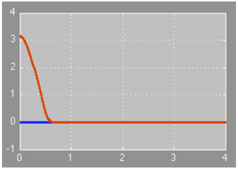
```
>> [X,U,D,Z,T,Y]=sim(C,S,r,x0,4);
```

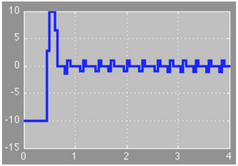




- Nonlinear simulation







CPU time:
51 ms per time step (GLPK)
22 ms per time step (CPLEX)
25 ms (GUROBI)
(Macbook Pro 3GHz Intel Core i7)

EXPLICIT HYBRID MPC

EXPLICIT HYBRID MPC (MLD FORMULATION)

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

$$\text{subject to} \quad \left\{ \begin{array}{rcl} x_{k+1} & = & Ax_k + B_1u_k + B_2\delta_k + B_3z_k + B_5 \\ y_k & = & Cx_k + D_1u_k + D_2\delta_k + D_3z_k + D_5 \\ E_2\delta_k + E_3z_k & \leq & E_4x_k + E_1u_k + E_5 \\ x_0 & = & x(t) \end{array} \right.$$

- **Online optimization**: solve the problem for a **given state** $x(t)$ as the **MILP**

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

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

  ➡ **multiparametric Mixed-Integer Linear Program (mp-MILP)**

MULTIPARAMETRIC MILP

- Consider the mp-MILP

$$\begin{array}{rl} \min_{\xi_c,\xi_d} & f_c'\xi_c + f_d'\xi_d \\ \text{s.t.} & G_c\xi_c + G_d\xi_d \leq W + Sx \end{array}$$

$$\xi_c \in \mathbb{R}^{n_c}$$
$$\xi_d \in \{0,1\}^{n_d}$$
$$x \in \mathbb{R}^m$$

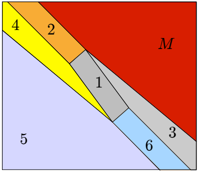
- A mp-MILP can be solved by alternating MILPs and mp-LPs

(Dua, Pistikopoulos, 1999)

- The multiparametric solution $\xi^*(x)$ is **PWA** (but possibly discontinuous)

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

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



(More generally, the parameter vector x includes states and reference signals)

EXPLICIT HYBRID MPC (PWA FORMULATION)

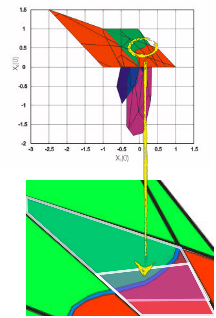
- Consider the MPC formulation using a PWA prediction model

$$
\begin{aligned}
\min_\xi J(\xi, x(t)) \quad &= \sum_{k=0}^{N-1} \|Qy_k\|_\infty + \|Ru_k\|_\infty \\
\text{subject to} \quad &\left\{\begin{array}{rcl}
x_{k+1} & = & A_{i(k)}x_k + B_{i(k)}u_k + f_{i(k)} \\
y_k & = & C_{i(k)}x_k + D_{i(k)}u_k + g_{i(k)} \\
 & & i(k) \text{ such that } H_{i(k)}x_k + W_{i(k)}u_k \leq K_{i(k)} \\
x_0 & = & x(t)
\end{array}\right.
\end{aligned}
$$

- **Method #1**: The explicit solution can be obtained by using a combination of **dynamic programming (DP)** and **mpLP** (Borrelli, Baotic, Bemporad, Morari, 2005)

- Clearly the explicit hybrid MPC law is again piecewise affine, as PWA systems $\equiv$ MLD systems

- **Method #2:** (Bemporad, Hybrid Toolbox, 2003)
  (Alessio, Bemporad, 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 of 1 / ∞-norms** or **convex PWA costs**: Compare value functions and **split regions**
  3b **Case of quadratic costs**: the partition may not be fully polyhedral, better **keep overlapping polyhedra** and compare online quadratic cost functions when overlaps are detected



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

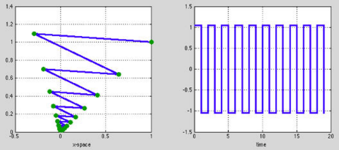
HYBRID MPC EXAMPLE - EXPLICIT VERSION

- PWA system:

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

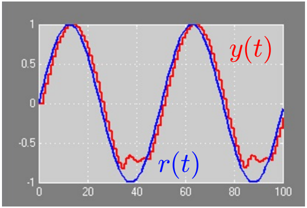
subject to $-1 \leq u(t) \leq 1$

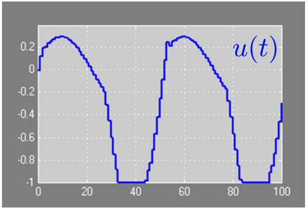
- MPC objective: $\min \sum_{k=1}^{2} |y_k - r(t)|$
- Open-loop behavior:



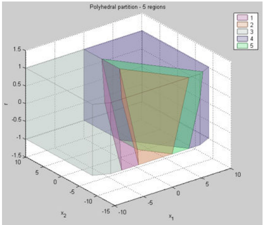
go to demo `demos/hybrid/bm99sim.m`

Closed-loop MPC

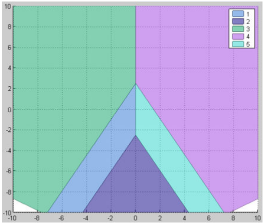




$$
u(x,r) = \begin{cases}
\begin{bmatrix} 0.6928 & -0.4 & 1 \end{bmatrix} \begin{bmatrix} x \\ r \end{bmatrix} & \text{if } \begin{bmatrix} 0.6928 & -0.4 & 1 \\ -0.4 & -0.6928 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \\ -0.6928 & 0.4 & -1 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ r \end{bmatrix}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



Section with $r = 0$



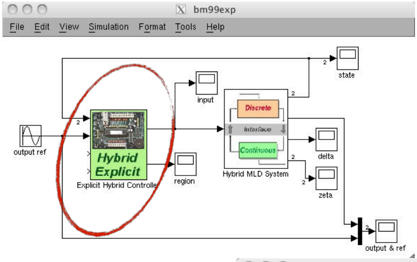
goto to `/demos/hybrid/bm99sim.m`

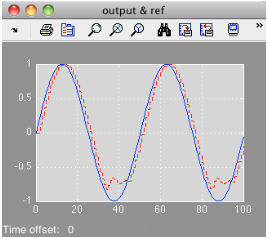
Offline CPU time = 1.51 s (Macbook Pro 3GHz Intel Core i7)

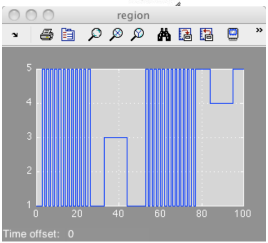
PWA law $\equiv$ MPC law !

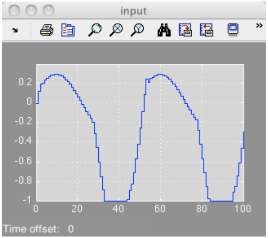
HYBRID MPC EXAMPLE – EXPLICIT VERSION

- Closed-loop explicit MPC



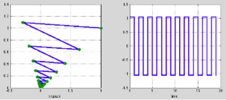






EXPLICIT PWA REGULATOR

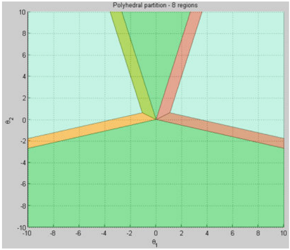
- MPC problem:



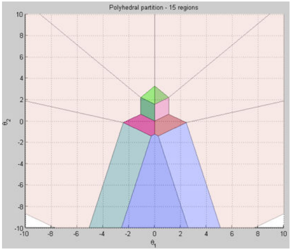
$$\begin{array}{ll} \min & 10\|x_N\|_\infty + \displaystyle\sum_{k=0}^{N-1} 10\|x_k\|_\infty + \|u_k\|_\infty \\ \text{s.t.} & \left\{\begin{array}{rcccl} -1 & \leq & u_k & \leq & 1,\ k=0,\ldots,N-1 \\ -10 & \leq & x_k & \leq & 10,\ k=1,\ldots,N \end{array}\right. \end{array}$$

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

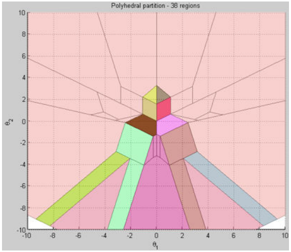
prediction horizon $N = 1$



prediction horizon $N = 2$



prediction horizon $N = 3$

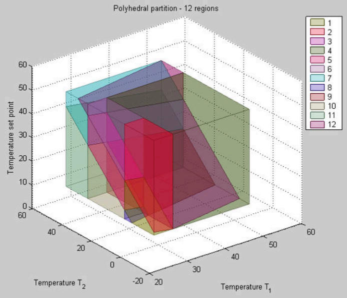


go to `demos/hybrid/bm99benchmark.m`

EXPLICIT HYBRID MPC – TEMPERATURE CONTROL

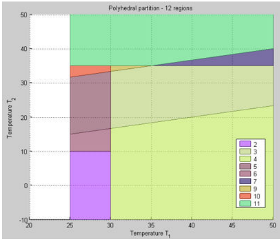
```
>> E=expcon(C,range,options);
```

```
>> 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

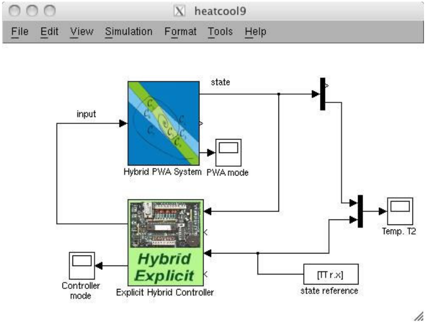
(T_1, T_2) section for $T_{\rm ref} = 30$

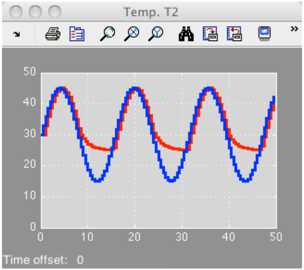


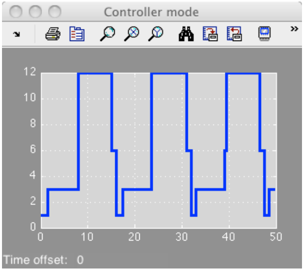
$$\min \quad \sum_{k=0}^{2} \|x_{2k} - r(t)\|_\infty$$

$$\text{s.t.} \quad \begin{cases} x_{1k} \geq 25, \ k = 1, 2 \\ \text{hybrid model} \end{cases}$$

EXPLICIT HYBRID MPC – TEMPERATURE CONTROL





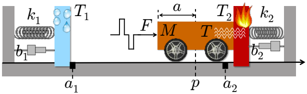


generated C-code

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,
```

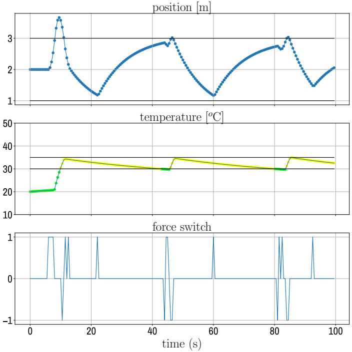
PARC - CART & BUMPERS EXAMPLE



- MPC problem with prediction horizon $N = 9$:

  (Bemporad, 2022)

$$\begin{aligned} \min_{F_0,\ldots,F_{N-1}} \quad & \sum_{k=0}^{N-1} |c_k - 1| + 0.25|F_k| \\ \text{s.t.} \quad & F_k \in \{-\bar{F}, 0, \bar{F}\} \\ & \text{PWA model equations} \end{aligned}$$



- MILP solution time: 0.37-1.9 s/step (CPLEX)

  (Intel Core i9-10885H CPU @2.40GHz)

- **Data-driven hybrid MPC** controller can keep temperature in yellow zone

- **Approximate explicit MPC**: fit a **decision tree** on 10,000 samples (accuracy: 99.7%). CPU time = 73÷88 μs. Closed-loop trajectories very similar.

IMPLEMENTATION ASPECTS OF HYBRID MPC

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

- **Small problems** (short horizon $N = 1, 2$, one or two inputs, 4-6 binary vars): explicit PWA control law is preferable
  - **CPU time** to evaluate the control law is shorter than by MIP
  - **control code** is simpler (no complex solver must be included in the control software!)
  - **more insight** in controller behavior

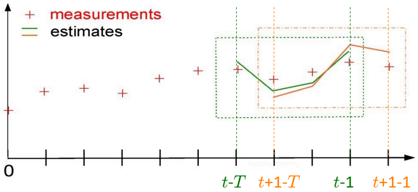
- **Medium/large problems** (longer horizon, many inputs and binary variables): online MIP is preferable

- **Further alternative**: collect MIP solutions and fit an **approximate explicit** form

MOVING HORIZON ESTIMATION AND FAULT DETECTION

(Bemporad, Mignone, Morari, 1999) (Ferrari-Trecate, Mignone, Morari, 2002)

- **Goal**: estimate the state of a hybrid system from past I/O measurements
- **Moving horizon estimation** based on MLD models solves the problem



MLD model augmented by

- state disturbance $\xi \in \mathbb{R}^n$
- output disturbance $\zeta \in \mathbb{R}^p$

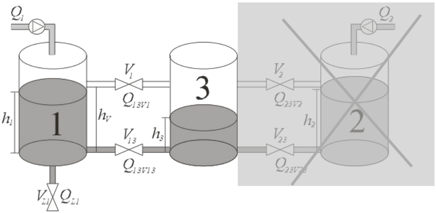
- At each time t get the estimate $\hat{x}(t)$ by solving the **MIQP**

$$\begin{array}{ll} \min_{\hat{x}(t-T|t)} & \displaystyle\sum_{k=0}^{T} \|\hat{y}(t-k|t) - y(t-k)\|_2^2 + \ldots \\ \text{s.t.} & \text{constraints on } \hat{x}(t-T+k|t), \hat{y}(t-T+k|t) \end{array}$$

- For **fault detection** also include unknown binary disturbances $\phi \in \{0,1\}^{n_f}$

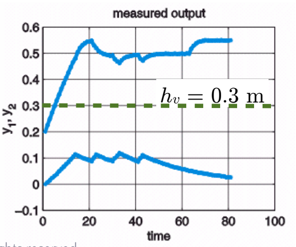
MHE EXAMPLE - THREE TANK SYSTEM

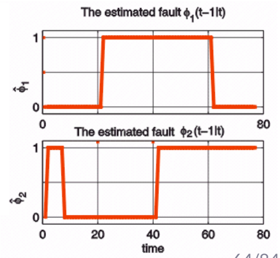
- Can only measure tank levels h_1, h_2
- The system has two faults:
  - ϕ_1: leak in tank 1 between 20 s $\leq t \leq$ 60 s
  - ϕ_2: valve V_1 blocked for $t \geq$ 40 s



(COSY benchmark problem)

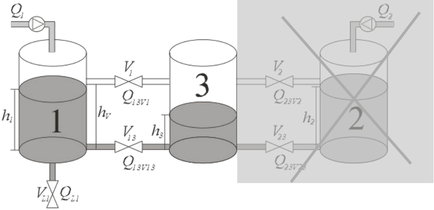
- Add logic constraint $[h_1 \leq h_v] \rightarrow \phi_2 = 0$





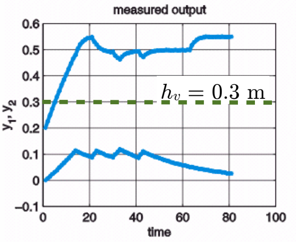
MHE EXAMPLE - THREE TANK SYSTEM

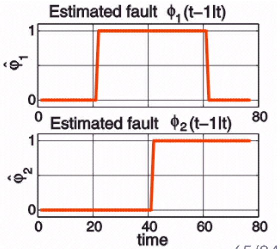
- Can only measure tank levels h_1, h_2
- The system has two faults:
  - ϕ_1: leak in tank 1 between 20 s $\leq t \leq$ 60 s
  - ϕ_2: valve V_1 blocked for $t \geq$ 40 s



(COSY benchmark problem)

- Add logic constraint $[h_1 \leq h_v] \rightarrow \phi_2 = 0$





A FEW (HYBRID) MPC TRICKS

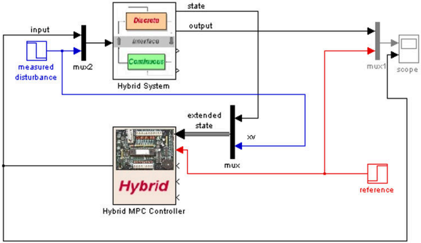
MEASURED DISTURBANCES

- A measured disturbance $v(t)$ enters the hybrid system
- Augment the hybrid prediction model with the constant state

$$\begin{array}{rcl} x_{k+1}^v & = & x_k^v \\ x_0^v & = & v(t) \end{array}$$

- HYSDEL model

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



- Same trick applies to linear MPC

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

REFERENCE TRACKING

- Hybrid MPC formulation for **reference tracking**

$$
\begin{array}{ll}
\min & \displaystyle\sum_{k=0}^{N-1} \|W^y(y_{k+1} - r(t))\|_2^2 + \|W^{\Delta u}\Delta u_k\|_2^2 \\
\text{s.t.} & \text{hybrid dynamics} \\
& \Delta u_k = u_k - u_{k-1},\ k = 0, \ldots, N-1,\ u_{-1} = u(t-1) \\
& u_{\min} \leq u_k \leq u_{\max},\ k = 0, \ldots, N-1 \\
& y_{\min} \leq y_k \leq y_{\max},\ k = 1, \ldots, N \\
& \Delta u_{\min} \leq \Delta u_k \leq \Delta u_{\max},\ k = 0, \ldots, N-1
\end{array}
$$

- The resulting optimization problem is the **MIQP**

$$
\begin{array}{ll}
\displaystyle\min_{\xi} & J(\xi, x(t)) = \frac{1}{2}\xi' H \xi + [x'(t)\, r'(t)\, u'(t-1)] F \xi \\
\text{s.t.} & G\xi \leq W + S \begin{bmatrix} x(t) \\ r(t) \\ u(t-1) \end{bmatrix}
\end{array}
\qquad
\xi = \begin{bmatrix} \Delta u_0 \\ \delta_0 \\ z_0 \\ \vdots \\ \Delta u_{N-1} \\ \delta_{N-1} \\ z_{N-1} \end{bmatrix}
$$

- Same trick as in linear MPC

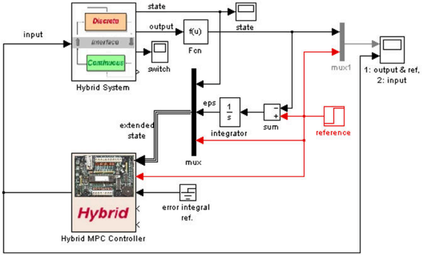
INTEGRAL ACTION

- Augment hybrid prediction model with **integrals of output tracking errors**

$$\epsilon_{k+1} = \epsilon_k + T_s(r(t) - y_k)$$

- Treat set point $r(t)$ as a measured disturbance (= constant state)
- Add weight on ϵ_k in cost function
- HYSDEL model:

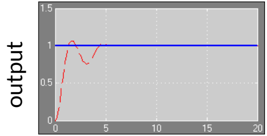
```
INTERFACE{
      STATE{
         REAL x          [-100,100];
         ...
         REAL epsilon    [-1e3, 1e3];
         REAL r          [0,     100]; }
      OUTPUT {
         REAL y; }
   ... }
IMPLEMENTATION{
      CONTINUOUS{
         epsilon=epsilon+Ts*(r-(c*x));
         r=r;
         ... 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



- Same trick applies to linear MPC

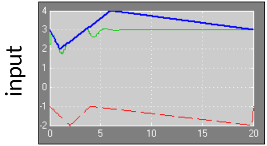
go to demo `demos/hybrid/hyb_integral_action.m`

TIME-VARYING CONSTRAINTS



- Consider the **time-varying constraint**

$$u(t) \leq u_{\max}(t)$$



- Augment the hybrid prediction model with the constant state

$$\begin{array}{rcl} x^u_{k+1} & = & x^u_k \\ x^u_0 & = & u_{\max}(t) \end{array}$$

and output $y^u_k = x^u(k) - u_k$, subject to the constraint $y^u_k \geq 0, k = 0, 1, \ldots, N$

- Same trick applies to linear MPC

go to demo `demos/linear/varbounds.m`

- **Alternative**: in HYSDEL simply impose `MUST {u <= xu;}`

REFERENCE/DISTURBANCE PREVIEW

(Bemporad, 2006)

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

- Augment the model with the following **buffer dynamics**

$$\begin{cases} x_{k+1}^{M-1} & = & x_k^{M-2} \\ x_{k+1}^{M-2} & = & x_k^{M-3} \\ & \vdots & \\ x_{k+1}^{1} & = & x_k^{0} \\ x_{k+1}^{0} & = & x_k^{0} \end{cases} \quad \text{with initial condition} \quad \begin{cases} x_0^{M-1} & = & v(t) \\ x_0^{M-2} & = & v(t+1) \\ \vdots & = & \vdots \\ x_0^{1} & = & v(t+M-2) \\ x_0^{0} & = & v(t+M-1) \end{cases}$$

- The predicted state x^{M-1} of the buffer is

$$\boxed{x_k^{M-1} = \begin{cases} v(t+k) & k = 0, \ldots, M-1 \\ v(t+M-1) & k = M, \ldots, N-1 \end{cases}}$$

- Preview of reference signal $r(t+k)$ can be dealt with in a similar way

- Same trick applies to linear MPC

DELAYS - METHOD #1

- Hybrid model with **delays**

$$\begin{array}{rcl} x(t+1) & = & Ax(t) + B_1 u(t-\tau) + B_2\delta(t) + B_3 z(t) + B_5 \\ E_2\delta(t) & + & E_3 z(t) \leq E_1 u(t-\tau) + E_4 x(t) + E_5 \end{array}$$

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

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

$$\begin{bmatrix} x(t+1) \\ x_\tau(t+1) \\ x_{\tau-1}(t+1) \\ \vdots \\ x_1(t+1) \end{bmatrix} = \begin{bmatrix} A & B_1 & 0 & 0 & \ldots & 0 \\ 0 & 0 & I_m & 0 & \ldots & 0 \\ 0 & 0 & 0 & I_m & \ldots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \ldots & 0 \end{bmatrix} \begin{bmatrix} x(t) \\ x_\tau(t) \\ x_{\tau-1}(t) \\ \vdots \\ x_1(t) \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ I_m \end{bmatrix} u(t) + \begin{bmatrix} B_2 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \delta(t) + \begin{bmatrix} B_3 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} z(t) + \begin{bmatrix} B_5 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

- Apply MPC to the extended MLD system
- Same trick as in linear MPC

DELAYS - METHOD #2

- **Delay-free** model:

$$\bar{x}(t) \triangleq x(t+\tau) \Rightarrow \begin{cases} \bar{x}(t+1) = A\bar{x}(t) + B_1 u(t) + B_2\bar{\delta}(t) + B_3\bar{z}(t) + B_5 \\ E_2\bar{\delta}(t) + E_3\bar{z}(t) \leq E_1 u(t) + E_4\bar{x}(t) + E_5 \end{cases}$$

- Design MPC for delay-free model, $u(t) = f_{\mathrm{MPC}}(\bar{x}(t))$

- Compute the predicted state

$$\bar{x}(t) = \hat{x}(t+\tau) = A^\tau x(t) + \sum_{j=1}^{\tau-1} A^j (B_1 \underbrace{u(t-1-j)}_{\text{past inputs!}} + B_2\bar{\delta}(t+j) + B_3\bar{z}(t+j) + B_5)$$

where $\bar{\delta}(t+j)$, $\bar{z}(t+j)$ are obtained from MLD inequalities or by simulation

- Compute the MPC control move $u(t) = f_{\mathrm{MPC}}(\hat{x}(t+\tau))$

CHOICE CONSTRAINTS

- **Logic constraint**: make one or more **choices** out of a set of alternatives:
  - make **at most one** choice: $\delta_1 + \delta_2 + \delta_3 \leq 1$
  - make **at least two** choices: $\delta_1 + \delta_2 + \delta_3 \geq 2$
  - **exclusive or** constraint: $\delta_1 + \delta_2 + \delta_3 = 1$
- More generally:

$\sum_{i=1}^{N} \delta_i \leq m$	choose **at most** m items out of N
$\sum_{i=1}^{N} \delta_i = m$	choose **exactly** m items out of N
$\sum_{i=1}^{N} \delta_i \geq m$	choose **at least** m items out of N

"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

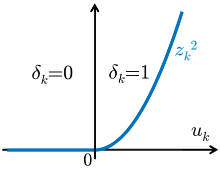
$$\sum_{i \in T} \delta_i - \sum_{i \in F} \delta_i \leq -1 + \sum_{i=1}^{n} \bar{\delta}_i \qquad \begin{array}{l} F = \{i : \bar{\delta}_i = 0\} \\ T = \{i : \bar{\delta}_i = 1\} \end{array}$$

or

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

ASYMMETRIC WEIGHTS

- **Asymmetric weight**: only weight a variable u_k if $u_k \geq 0$
- We can introduce a binary variable $[\delta_k = 1] \leftrightarrow [u_k \geq 0]$ and



$$z_k = \max\{u_k, 0\} = \begin{cases} u_k & \text{if } \delta_k = 1 \\ 0 & \text{otherwise} \end{cases}$$

then weight z_k instead of u_k

- **Better solution**: only introduce auxiliary variable z_k and optimize

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

- Similar approach when $\|\cdot\|_\infty$ or $\|\cdot\|_1$ are used as penalties
- 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 !

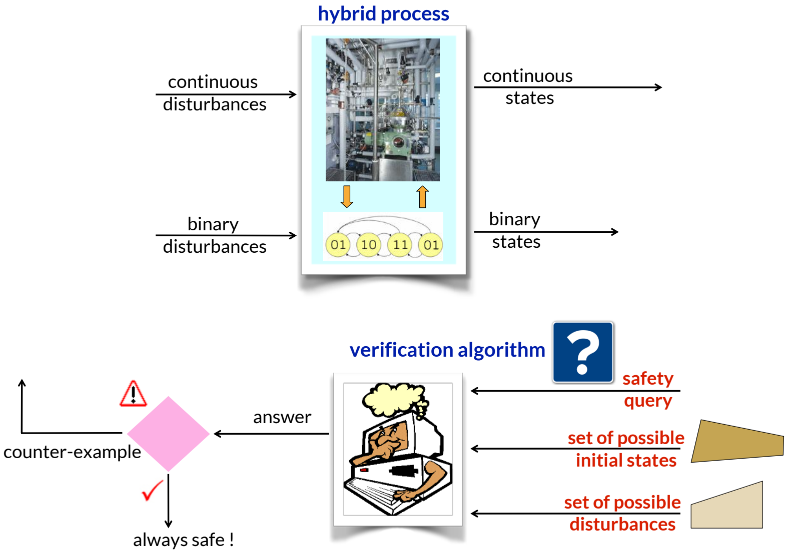
- Adding logical constraints usually helps

- Generally speaking

modeling is an art

VERIFICATION (REACHABILITY ANALYSIS)

HYBRID VERIFICATION PROBLEM



VERIFICATION ALGORITHM #1

- **Query**: Is the target set X_f reachable in N steps from some initial state $x_0 \in X_0$ for some input $u_0, \ldots, u_{N-1} \in U$?

- The query can be answered by solving the **mixed-integer feasibility test**

$$
\begin{array}{ll}
\min_\xi & 0 \\
\text{s.t.} & x_{k+1} = Ax_k + B_1 u_k + B_2 \delta_k + B_3 z_k + B_5 \\
& E_2 \delta_k + E_3 z_k \leq E_4 x_k + E_1 u_k + E_5 \\
& S_u u_k \leq T_u \quad (u_k \in U), \quad k = 0, 1, \ldots, N-1 \\
& S_0 x_0 \leq T_0 \quad (x_0 \in X_0) \\
& S_f x_N \leq T_f \quad (x_N \in X_f)
\end{array}
$$

with respect to $\xi = [x_0, \ldots, x_N, u_0, \ldots, u_{N-1}, \delta_0, \ldots, \delta_{N-1}, z_0, \ldots, z_{N-1}]$

- Other approaches:
  - Exploit structure and use polyhedral computation (Torrisi, 2003)
  - Use abstractions (LPs) + SAT solvers (Giorgetti, Pappas, Bemporad)

VERIFICATION EXAMPLE

- MLD model: room temperature control system
- Set of unsafe states:

$$X_f = \left\{ \left[\begin{smallmatrix} T_1 \\ T_2 \end{smallmatrix} \right] :\ 10 \leq T_1, T_2 \leq 15 \right\}$$

- Set of initial states:

$$X_0 = \left\{ \left[\begin{smallmatrix} T_1 \\ T_2 \end{smallmatrix} \right] :\ 35 \leq T_1, T_2 \leq 40 \right\}$$

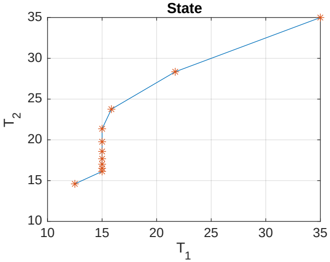
- Set of possible inputs:

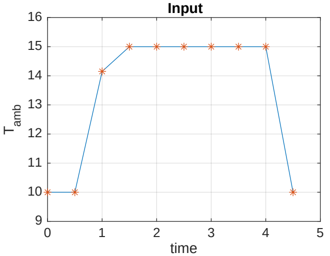
$$U = \{T_{\rm amb} :\ 10 \leq T_{\rm amb} \leq 30\}$$

- Time horizon: $N = 10$ steps

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

VERIFICATION EXAMPLE





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

```
>> umin=20;
>> reach(S,N,Xf,X0,umin,umax);
Hybrid Toolbox v.1.4.2 [February 2, 2020]
Elapsed time is 0.023282 seconds.

Xf is not reachable from X0

>>
```

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

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 $u_0, \ldots, u_{N-1} \in U$?

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

  - Add a new variable δ_k^f, with $[\delta_k^f = 1] \to [A_f x_{k+1} \leq b_f]$

$$\underbrace{\Longrightarrow}_{\text{big-M}} \quad A_f(Ax_k + B_1 u_k + B_2 \delta_k + B_3 z_k + B_5) \leq b_f + M(1 - \delta_k^f)$$

  - Add the constraint $\displaystyle\sum_{k=0}^{N-1} \delta_k^f \geq 1$ (i.e., $x_k \in X_f$ for at least one k)

  - Solve MILP feasibility test

- **Note**: the verification problem is a **bounded model-checking** problem with continuous and binary variables

A MORE COMPLEX VERIFICATION EXAMPLE

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

- Events:
$$\begin{array}{l} [\delta_1 = 1] \leftrightarrow [x_1 \leq 0] \\ [\delta_2 = 1] \leftrightarrow [x_2 \geq 1] \\ [\delta_3 = 1] \leftrightarrow [x_3 - x_2 \leq 1] \end{array}$$

- Switched dynamics

$$\begin{array}{rcl} x_1(k+1) & = & \begin{cases} 0.1x_1(k) + 0.5x_2(k) & \text{if } (\delta_1(k) \wedge \delta_2(k)) \vee x_4(k) \text{ true} \\ -0.3x_3(k) - x_1(k) + u_1(k) & \text{otherwise} \end{cases} \\ x_2(k+1) & = & \begin{cases} -0.8x_1(k) + 0.7x_3(k) - u_1(k) - u_2(k) & \text{if } \delta_3(k) \vee x_5(k) \text{ true} \\ -0.7x_1(k) - 2x_2(k) & \text{otherwise} \end{cases} \\ x_3(k+1) & = & \begin{cases} -0.1x_3(k) + u_2(k) & \text{if } (\delta_3(k) \wedge x_5(k)) \vee (\delta_1(k) \wedge x_4(k)) \text{ true} \\ x_3(k) - 0.5x_1(k) - 2u_1(k) & \text{otherwise} \end{cases} \end{array}$$

- Automaton

$$\begin{array}{rcl} x_4(k+1) & = & \delta_1(k) \wedge x_4(k) \\ x_5(k+1) & = & ((x_4(k) \vee x_5(k)) \wedge (\delta_1(k) \vee \delta_2(k)) \vee (\delta_3(k) \wedge u_3(k)) \end{array}$$

A MORE COMPLEX VERIFICATION EXAMPLE

- **Query**: Verify if it possible that, starting from the set X_0

$$X_0 = \{x : -0.1 \le x_1, x_3 \le 0.1,\ x_2 = 0.1,\ x_4, x_5 \in \{0,1\}\}$$

the state $x(k) \in X_f$

$$X_f = \{x : -1 \le x_1, x_3 \le 1,\ 0.5 \le x_2 \le 1,\ x_4, x_5 \in \{0,1\}\}$$

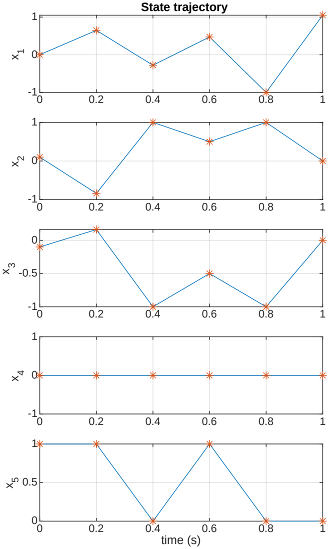
at some $k \le N$, $N = 5$, under the restriction that $\forall k \le N$

$$\begin{array}{l} x_3(k) + x_2(k) \le 0 \\ \delta_1(k) \vee \delta_2(k) \vee x_5(k) = \texttt{true} \\ \neg x_4(k) \vee x_5(k) = \texttt{true} \end{array}$$

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

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

A MORE COMPLEX VERIFICATION EXAMPLE




```
>> reachtest
Hybrid Toolbox v.1.4.2 [February 2, 2020]
Elapsed time is 0.038049 seconds.
>> reachtime

reachtime =


3
4


>>
```

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