Model Predictive Control

Alberto Bemporad

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

Outline

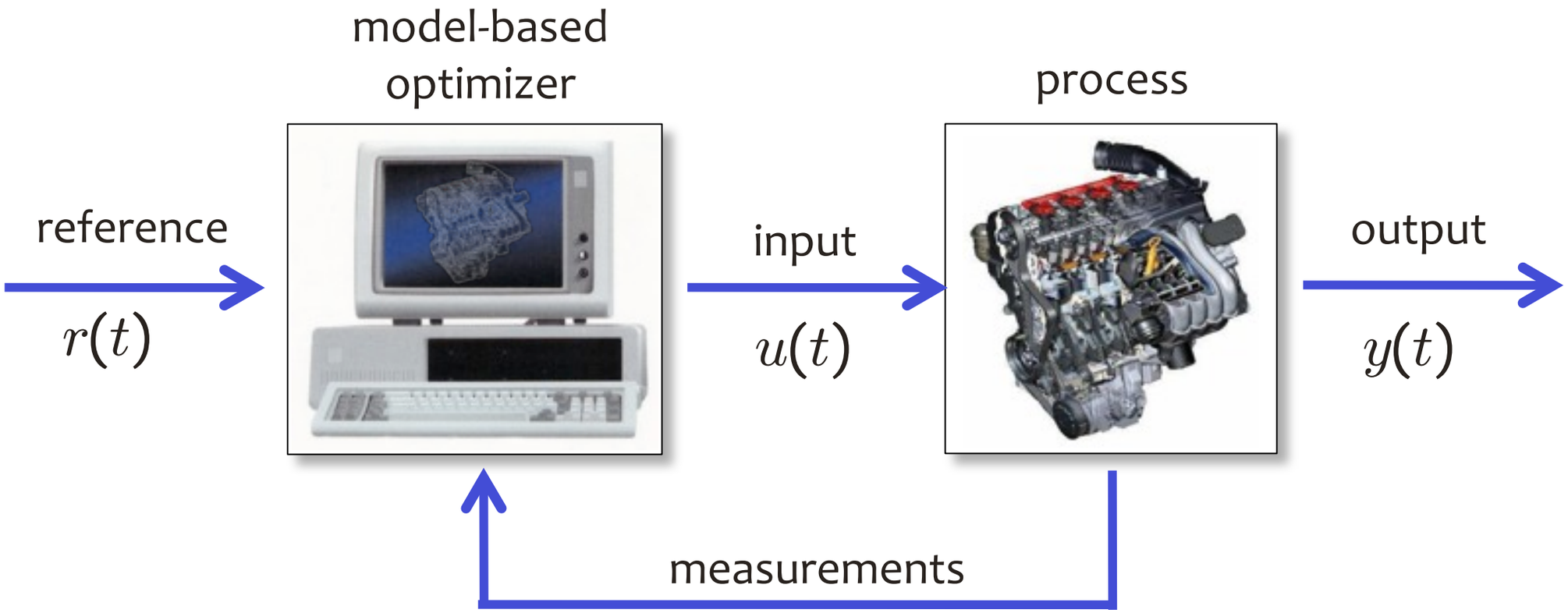
- Basic concepts of MPC
- Solution algorithms (quadratic optimization)
- Hybrid MPC
- Stochastic MPC
- Decentralized MPC

- More detailed lecture notes:

http://cse.lab.imtlucca.it/~bemporad/teaching/mpc/imt

Model Predictive Control: Basic Concepts

Model Predictive Control (MPC)



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

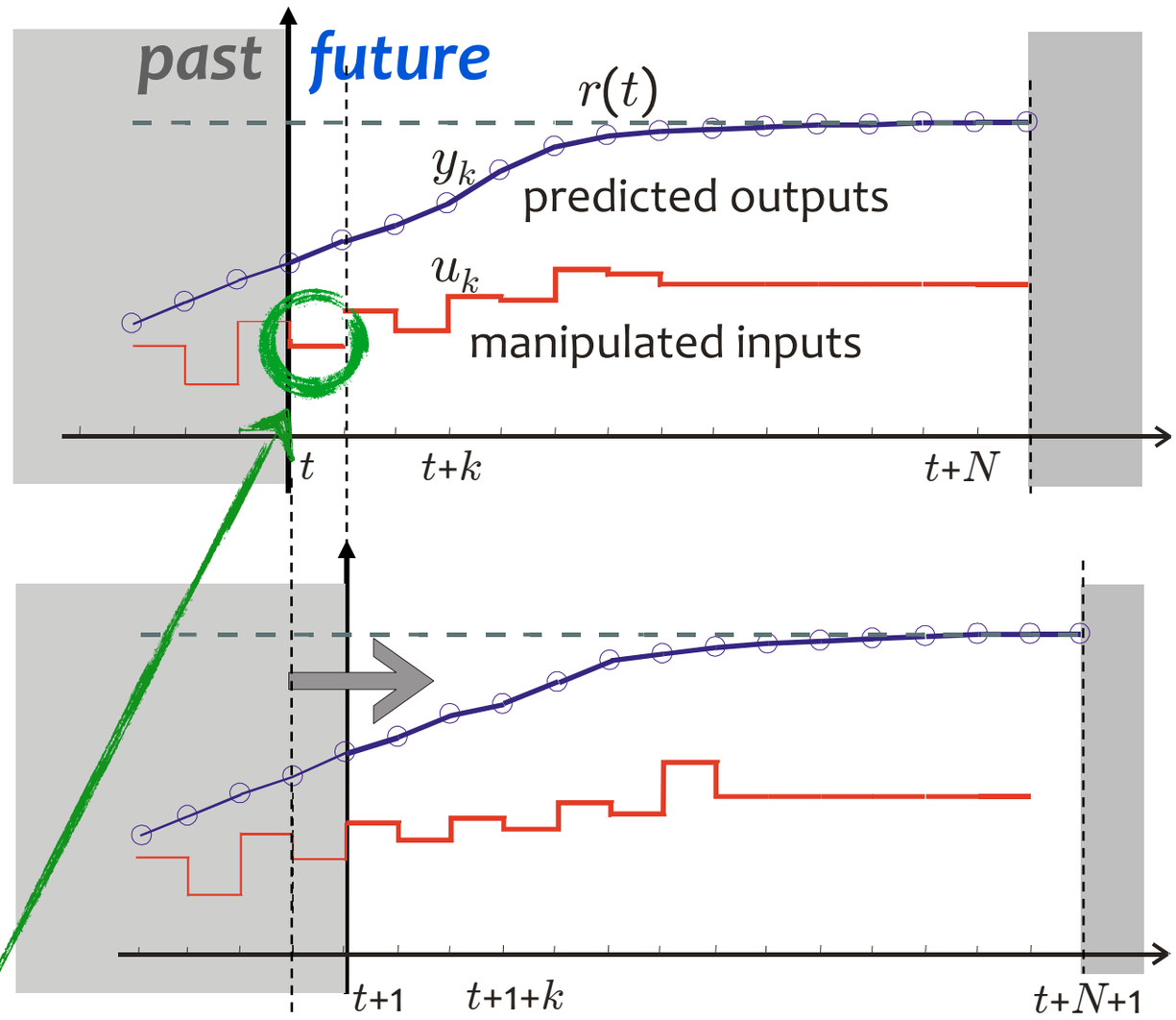
Receding horizon control

- **At time** t: solve an **optimal control** problem over a future horizon of N steps

penalty on tracking error

penalty on actuation effort

$$\begin{aligned}\min \quad & \sum_{k=0}^{N-1} \ell(y_k - r(t+k), u_k) \\ \text{s.t.} \quad & x_{k+1} = f(x_k, u_k) \\ & y_k = g(x_k, u_k) \\ & \text{constraints on } u_k, x_k, y_k \\ & x_0 = x(t)\end{aligned}$$



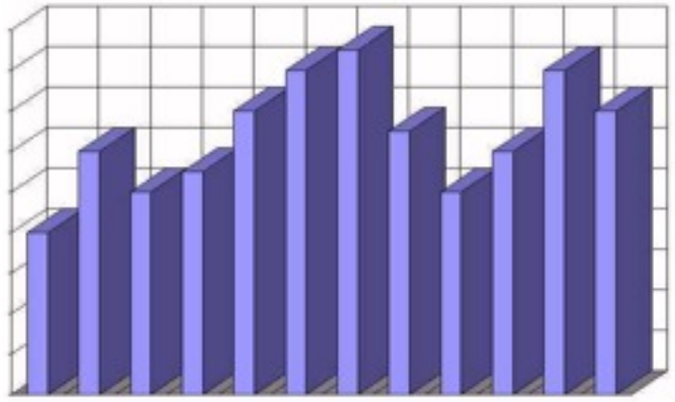
- Apply only the first optimal move $u^*(t)$, throw the rest of the sequence away
- **At time** $t+1$: Get new measurements, repeat the optimization. And so on …

MPC transforms open-loop optimal control into **feedback** control

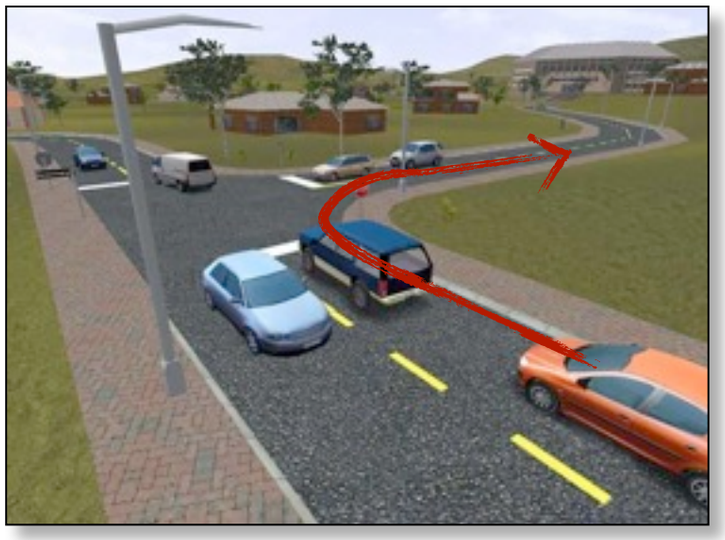
Receding horizon examples

- MPC is like **playing chess** !

- "Rolling horizon" policies are also used frequently **in finance**



- Drivers use "receding horizon" control too!



MPC of linear systems

linear model $\begin{cases} x_{k+1} = Ax_k + Bu_k \\ y_k = Cx_k \end{cases}$

$R = R' \succ 0$
$Q = Q' \succeq 0$
$P = P' \succeq 0$

performance index $\min_U x'_N P x_N + \sum_{k=0}^{N-1} x'_k Q x_k + u'_k R u_k$

$x_0 = x(t)$

$$\begin{aligned} \min_U \quad & \frac{1}{2} z' H z + x'(t) F' z + \frac{1}{2} x'(t) Y x(t) \\ \text{s.t.} \quad & Gz \leq W + Sx(t) \end{aligned}$$

$$z = \begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_{N-1} \end{bmatrix}$$

constraints $\begin{cases} u_{\min} \leq u_k \leq u_{\max} \\ y_{\min} \leq Cx_k \leq y_{\max} \end{cases}$

MPC problem maps to a **convex Quadratic Program (QP)**

Linear MPC algorithm

At each sampling time t:

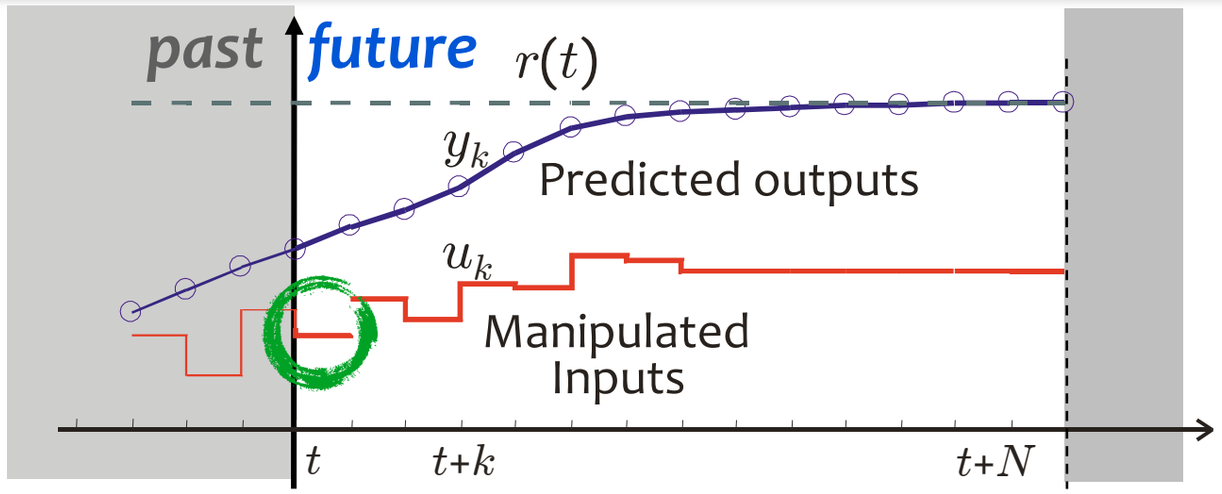
- Measure (or estimate) the current state $x(t)$

- Solve the QP problem

$$\min_z \quad \tfrac{1}{2} z'Hz + x'(t)Fz$$
$$\text{s.t.} \quad Gz \le W + Sx(t)$$

and let $z^* = \{u_0^*, \ldots, u_{N-1}^*\}$ be the solution

- Apply only $u(t) = u_0^*$ and discard the remaining optimal inputs



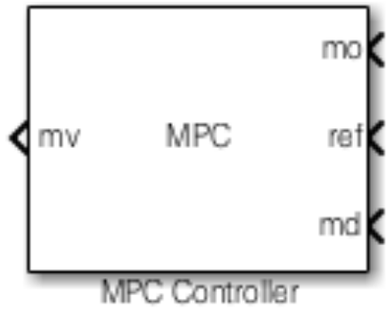
Routinely used in the process industries

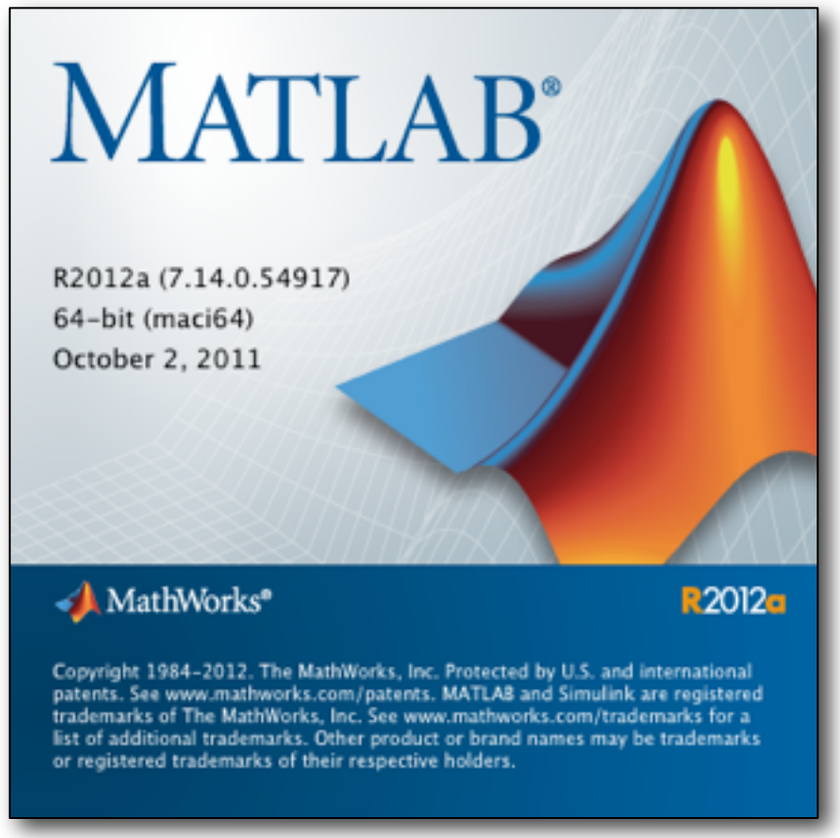
Honeywell aspentech ABB

Model Predictive Control Toolbox

(Bemporad, Ricker, Morari, 1998-2013)

- **MPC Toolbox 4.0** (The Mathworks, Inc.)
  - Object-oriented implementation (MPC object)
  - MPC Simulink Library
  - MPC Graphical User Interface
  - Code generation [RTW, xPC Target, dSpace, etc.]
  - Linked to OPC Toolbox, System ID Toolbox, ...





http://www.mathworks.com/products/mpc/

Basic convergence properties

Theorem. Consider the linear system

$$\begin{cases} x(t+1) &= Ax(t)+Bu(t) \\ y(t) &= Cx(t) \end{cases}$$

and the MPC control law based on optimizing

$$\begin{aligned} V^*(x(t)) = \min \quad & \sum_{k=0}^{N-1} x_k' Q x_k + u_k' R u_k \\ \text{s.t.} \quad & x_{k+1} = Ax_k + Bu_k \\ & u_{\min} \le u_k \le u_{\max} \\ & y_{\min} \le Cx_k \le y_{\max} \\ & x_N = 0 \end{aligned}$$

"terminal constraint"

with $R,Q>0$. If the optimization problem **is feasible at time** $t=0$ then

$$\lim_{t\to\infty} x(t) = 0$$
$$\lim_{t\to\infty} u(t) = 0$$

and the constraints are satisfied at all time $t\ge 0$, for all $R,Q>0$

(Keerthi and Gilbert, 1988) (Bemporad, Chisci, Mosca, 1994)

For general **stability result** see (Lazar, Heemels, Weiland, Bemporad, IEEE TAC, 2006)

Convergence proof

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

- Let U_t = optimal control sequence at time t, $U_t = [u_0^t \ \dots \ u_{N-1}^t]'$
- By construction $\bar{U}_{t+1} = [u_1^t \ \dots \ u_{N-1}^t \ 0]'$ is a feasible sequence at time $t+1$
- The cost of $\bar{U}_{t+1}$ is $V^*(x(t)) - x'(t)Qx(t) - u'(t)Ru(t) \geq V^*(x(t+1))$

- $V^*(x(t))$ is monotonically decreasing and ≥ 0, so $\exists \lim_{t\to\infty} V^*(x(t)) \triangleq V_\infty$
- Hence $0 \leq x'(t)Qx(t) + u'(t)Ru(t) \leq V^*(x(t)) - V^*(x(t+1)) \to 0$ for $t \to \infty$
- Since $R, Q > 0$, $\lim_{t\to\infty} x(t) = 0$, $\lim_{t\to\infty} u(t) = 0$ □

Global optimum is not needed to prove convergence !

MPC of linear systems - Tracking

- Tracking a reference $r(t)$:

$$\min_U \sum_{k=0}^{N-1} \frac{1}{2}(y_k - r(t))' S (y_k - r(t)) + \frac{1}{2}\Delta u_k' T \Delta u_k$$

$$\Delta u_k \triangleq u_k - u_{k-1}$$
$$S = S' \geq 0$$
$$T = T' > 0$$

- Integral action: include integral of tracking error $I_{k+1} = I_k + (y_k - r(t))$ in the prediction model and penalize also $\frac{1}{2} I_k' S_I I_k$ (Kwakernaak & Sivan, 1972)

Idea: $r(t) \equiv r$, $I_k \to \text{const} \Rightarrow y_k - r \to 0$

- Rejection of a measured disturbance $v(t)$:

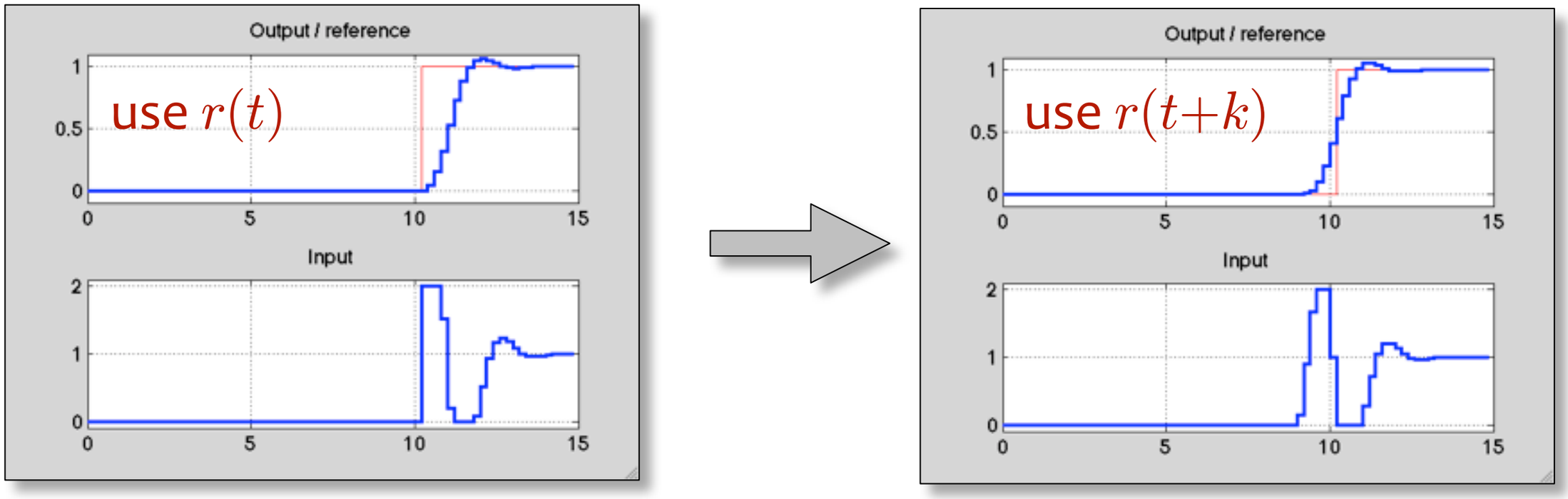
$$\begin{cases} x_{k+1} = Ax_k + Bu_k + B_v v(t) \\ y_k = Cx_k + D_v v(t) \end{cases}$$

All the above linear MPC problems map into a **convex QP**

MPC of linear systems - Preview

- Preview (a.k.a. "anticipative action"):

$$\min_U \sum_{k=0}^{N-1} \frac{1}{2}(y_k - r(t+k))' S (y_k - r(t+k)) + \frac{1}{2}\Delta u_k' T \Delta u_k$$



- Preview on measured disturbance

$$\begin{cases} x_{k+1} = Ax_k + Bu_k + Ev(t+k) \\ y_k = Cx_k + Fv(t+k) \end{cases}$$

both still map into a **convex QP**

MPC of linear systems - Improving robustness

- Extend the prediction model by adding constant unknown disturbances

$$\begin{cases} x_{k+1} &= Ax_k + Bu_k + B_d d_k \\ d_{k+1} &= d_k \\ y_k &= Cx_k + D_d d_k \end{cases}$$

- Use the extended model to design a state observer (e.g., Kalman filter) that estimates both the state $\hat{x}(t)$ and disturbance $d(t)$ from $y(t)$

- **Main idea:** observer makes $y(t) - (C\hat{x}(t) + D_d\hat{d}(t)) \to 0$ (estimation error)

  MPC makes $C\hat{x}(t) + D_d\hat{d}(t) \to r(t)$ (predicted tracking error)

  $\Rightarrow$ the combination makes $y(t) \to r(t)$ (actual tracking error)

- **Explanation:** in steady state, the term $D_d\hat{d}(t)$ compensates for model mismatch (=same effect of integral action, but at the observer level)

Linear time-varying MPC

- MPC can easily handle **linear time-varying (LTV)** problems

$$\begin{cases} x_{k+1} &= A_k(t,x(t))x_k + B_k(t,x(t))u_k + f_k(t,x(t)) \\ y_k &= C_k(t,x(t))x_k + D_k(t,x(t))u_k + g_k(t,x(t)) \end{cases}$$

$$E_k(t,x(t))x_k + F_k(t,x(t))u_k \le h_k(t,x(t)) \qquad k = 0,1,\dots,N-1$$

$$\min \sum_{k=0}^{N} \ell_k(y_k, u_k, r(t+k), t, x(t)) \qquad \ell_k = \text{quadratic function of } y_k, u_k$$

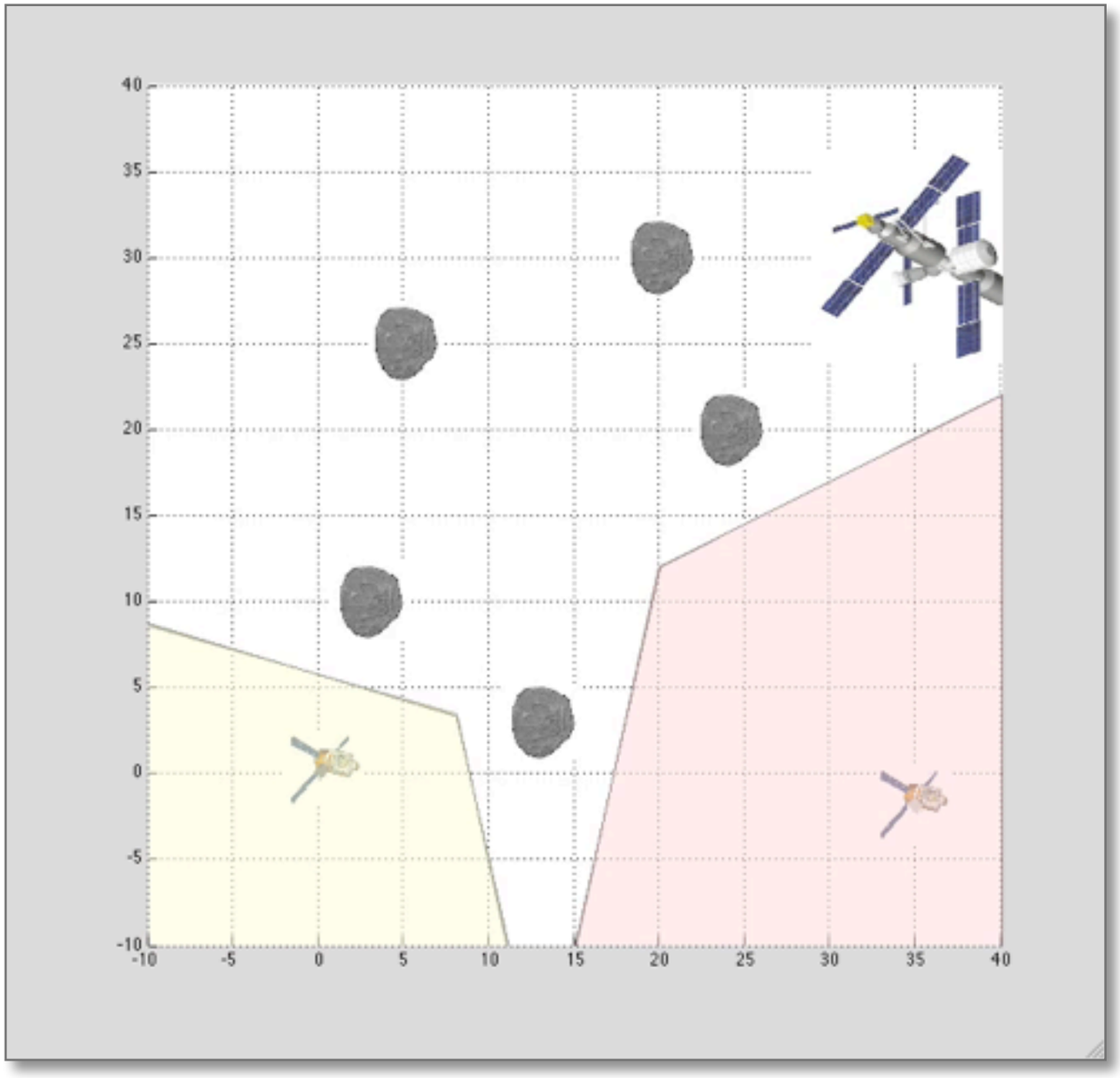
$$\begin{aligned} \min_z \quad & \frac{1}{2} z' H(t) z + F(t)' z + \alpha(t) \\ \text{s.t.} \quad & G(t) z \le W(t) \end{aligned}$$

LTV-MPC still leads to a convex QP

- Applications: time-varying systems (e.g.: aerospace), NL systems

Example: LTV-MPC for UAV navigation

- Goal: navigate two vehicles among obstacles to target position
- Dynamical model = 2nd-order transfer function
- Constrains are time-varying polyhedral approximations of the feasible space
- Each UAV solves its own LTV-MPC problem
- Previous optimal sequences are exchanged to improve cooperation



(Bemporad, Rocchi, 2011)

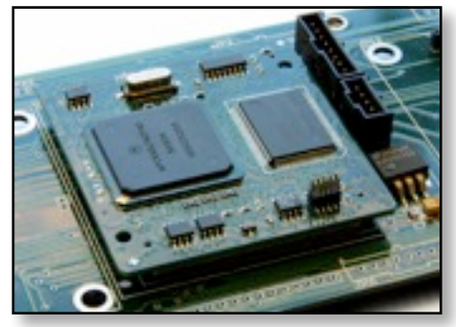
Computing the MPC control action

$$\begin{aligned} \min_{z} \quad & \tfrac{1}{2} z' H z + x'(t) F z \\ \text{s.t.} \quad & G z \leq W + S x(t) \end{aligned}$$

Requirements for Embedded MPC algorithms

- **Speed**: fast enough to provide a solution within short sampling intervals (such as 10-100 ms)

- Require **simple hardware** (microcontroller/FPGA) and little memory to store problem **data *and* code**



- **Code simple** enough to be **software-certifiable**

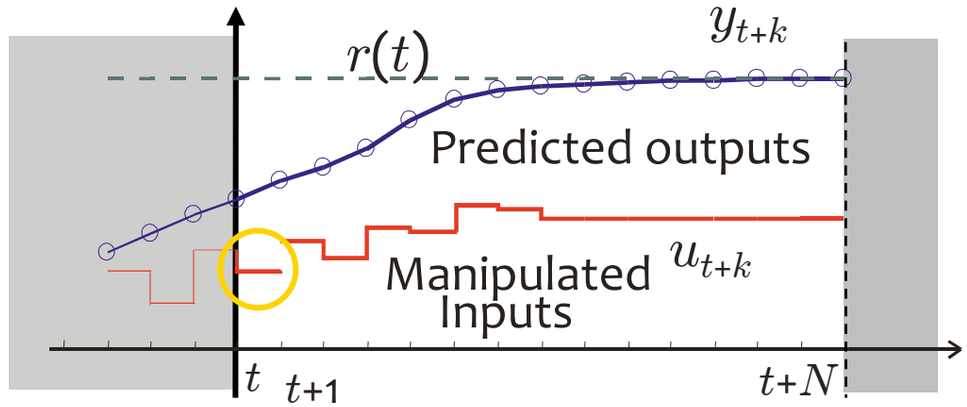
```
54    # Main iteration loop
55    while keepgoing & (nu<maxiter):
56
57        # Compute theta, beta
58        beta=th*(1/th0-1)
59
60        w=y+beta*(y-y0)
61        zhat=-(dot(MG,w)+gP)
62        if ~onlyzhat:
63            z=(1-th)*z+th*zhat
64        th0=th # Update th0
65        th2=th**2
66        th=(sqrt(th2**2+4.0*th2)-th2)/2.0
```

- Tightly estimate **worst-case execution time** to certify **hard real-time system** properties

Linear MPC - Unconstrained case

- Minimize quadratic function, no constraints

$$\min_z \; f(z) = \frac{1}{2} z' H z + x'(t) F z$$



- Solution: $\nabla f(z) = Hz + F'x(t) = 0$

$$z = \begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_{N-1} \end{bmatrix}$$

$$\longrightarrow \; z^* = -H^{-1} F' x(t)$$

$$\Longrightarrow \quad u(t) = -[I \; 0 \; \ldots \; 0] H^{-1} F x(t) \triangleq K x(t)$$

Unconstrained linear MPC = **linear state-feedback** !

MPC and Linear Quadratic Regulation (LQR)

- Special case: $f(z) = \min_z x'_N P x_N + \sum_{k=0}^{N-1} x'_k Q x_k + u'_k R u_k$

with matrix P solving the Algebraic Riccati Equation

$$P = A'PA - A'PB(B'PB + R)^{-1}B'PA + Q$$

Jacopo Francesco Riccati (1676 - 1754)

- **(unconstrained) MPC = LQR** (for any choice of the prediction horizon N)

Proof. Easily follows from Bellman's principle of optimality (dynamic programming): $x'_N P x_N$ = optimal "cost-to-go" from time N to ∞.

- Result extends to constrained LQR case

(Sznaier & Damborg, 1987) (Chmielewski & Manousiouthakis, 1996) (Scokaert & Rawlings, 1998)

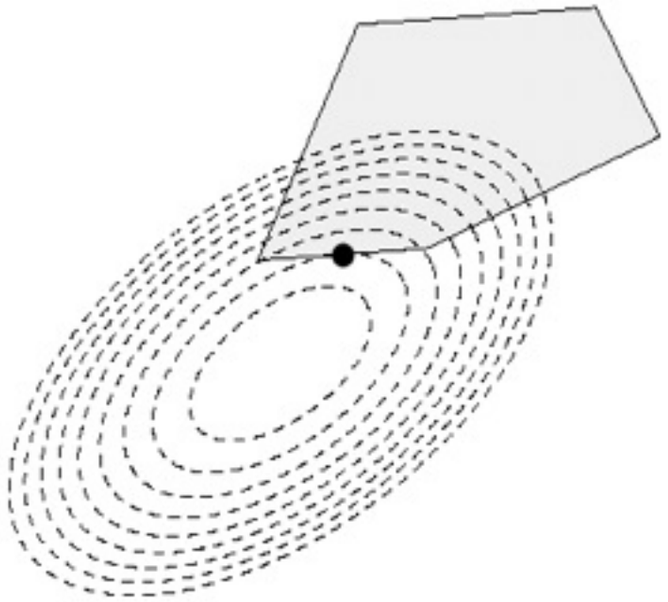
Solution methods for QP

Most used algorithms for solving QP problems:

- ***active set*** methods **(small/medium size)**
- ***interior point*** methods **(large scale)**
- ***gradient projection*** methods
- ***conjugate gradient*** methods
- ***augmented Lagrangian*** methods
  (and ***alternating direction method of multipliers***)

$$\begin{aligned} \min_z \quad & \tfrac{1}{2} z' H z + x' F z \\ \text{s.t.} \quad & Gz \le W + Sx \end{aligned}$$

Quadratic Program (QP)



A useful performance comparisons of many solvers can be found at

http://plato.asu.edu/guide.html

Gradient projection method

- Optimization problem: $\min_{z \in Z} f(z)$ $\quad f : \mathbb{R}^s \to \mathbb{R}$, $Z \subseteq \mathbb{R}^s$ (Goldstein, 1964) (Levitin & Poljak, 1965)
- Assume f is convex and has Lipschitz continuous gradient

$$\|\nabla f(z_1) - \nabla f(z_2)\| \leq L\|z_1 - z_2\| \qquad \forall z_1, z_2 \in Z$$

- Algorithm: given initial guess z_0, iterate

$$z_{k+1} = \mathcal{P}_Z\left(z_k - \frac{1}{L}\nabla f(z_k)\right)$$

where $\mathcal{P}_Z(z)$ = projection of z on Z and is "easy to compute"

- Example: $Z = \{z \in \mathbb{R}^s : \ z \geq 0\} \ \rightarrow \ \mathcal{P}_Z(z) = \max\{z, 0\}$
- Convergence rate: $f(z_{k+1}) - f^* \leq \frac{L}{2(k+1)}\|z_0 - z^*\|$

Gradient projection method for QP (very simple!)

- QP problem:

$$\begin{aligned} \min_z \quad & \frac{1}{2} z'Hz + x'Fz \\ \text{s.t.} \quad & Gz \le W + Sx \end{aligned}$$

$z =$ optimization vector

$x =$ parameter vector

- Apply gradient-projection to **dual** QP

$$\begin{aligned} \min_{y \ge 0} \quad & \frac{1}{2} y'My + (Dx + W)'y \end{aligned}$$

$$\begin{aligned} M &= GH^{-1}G' \\ D &= GH^{-1}F' + S \end{aligned}$$

L = max eigenvalue of M, or $L = \sqrt{\sum_{i,j=1}^{m} |M_{i,j}|^2}$ (Frobenius norm)

- Iterate $y_{k+1} = \max\{(I - \frac{1}{L}M)y_k - \frac{1}{L}(Dx + W), 0\}$ from $y_0=0$

  until convergence to optimal y^* (guaranteed if QP is feasible)

- Set $z^* = -H^{-1}(G'y^* + F'x)$

Fast gradient projection method

(Nesterov, 1983)

- Assume again f convex and ∇f Lipschitz continuous

- Algorithm: given initial guess $z_{-1} = z_0$, iterate

$$\begin{aligned} w_k &= z_k + \beta_k(z_k - z_{k-1}) \\ z_{k+1} &= \mathcal{P}_Z\left(w_k - \tfrac{1}{L}\nabla f(w_k)\right) \end{aligned}$$

$$\beta_k = \begin{cases} 0 & k = 0 \\ \frac{k-1}{k+2} & k > 0 \end{cases}$$

- Convergence rate:

$$f(z_{k+1}) - f^* \leq \frac{2L}{(k+2)^2}\|z_0 - z^*\|^2$$

Fast gradient projection for (dual) QP

(Patrinos, Bemporad, 2012)

- Take f = dual QP function. Iterations can be written as

$$\begin{aligned} w_k &= y_k + \beta_k(y_k - y_{k-1}) \\ z_k &= -Kw_k - Jx \\ s_k &= \tfrac{1}{L}Gz_k - \tfrac{1}{L}(Sx + W) \\ y_{k+1} &= \max\{y_k + s_k, 0\} \end{aligned}$$

$$\begin{aligned} K &= H^{-1}G' \\ J &= H^{-1}F' \end{aligned}$$

$$y_{-1} = y_0 = 0$$

- Termination criterion #1: **primal feasibility**

$$s_k^i \le \frac{1}{L}\epsilon_G, \ \ \forall i = 1, \dots, m$$

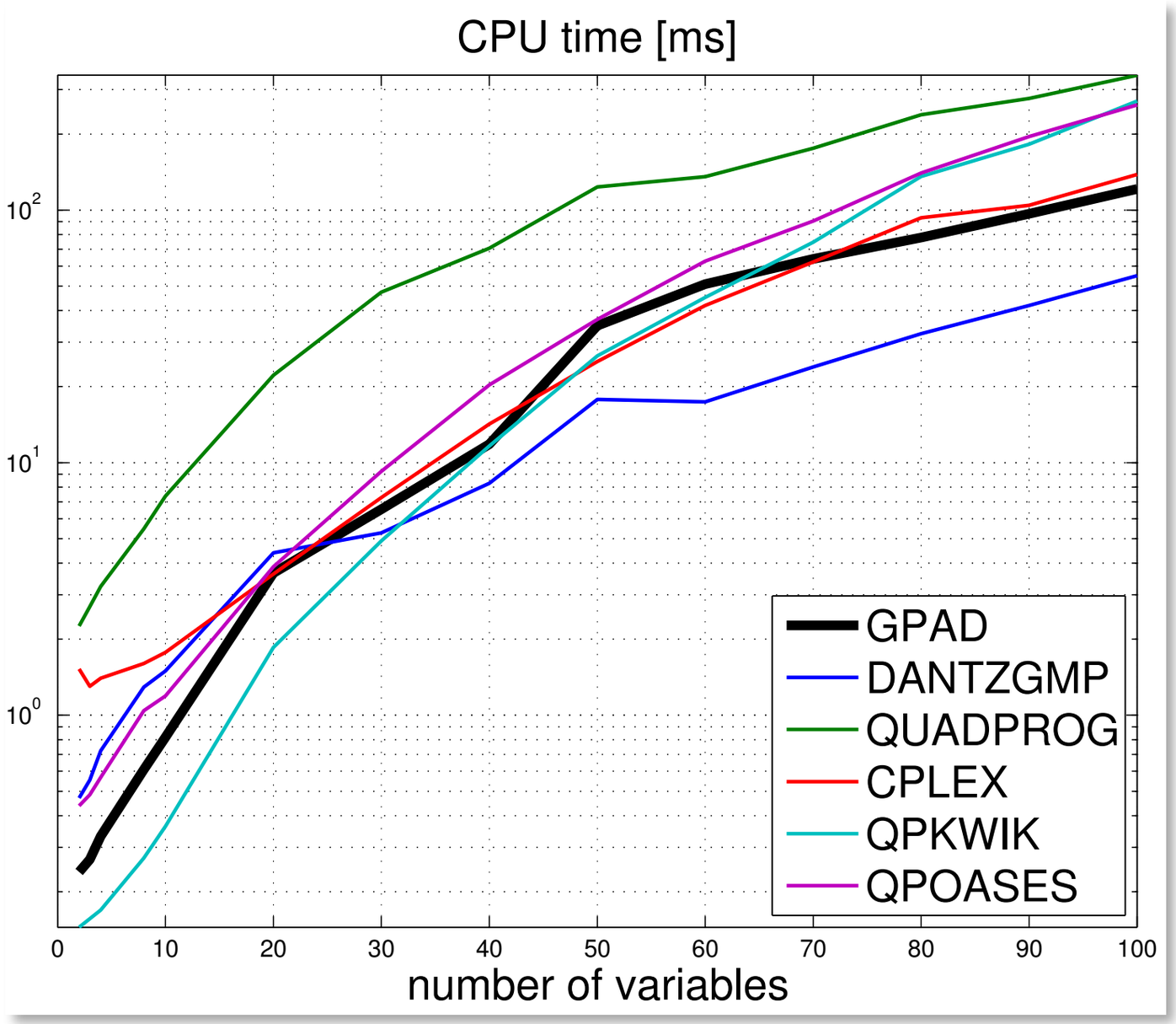
- Termination criterion #2: **primal optimality**

$$f(z_k) - f^* \le f(z_k) - \phi(w_k) = -w_k' s_k L \le \epsilon_V$$

dual function

$$-w_k' s_k \le \frac{1}{L}\epsilon_V$$

Numerical results



Average CPU time on random QP problems with n variables and $2n$ constraints

[standard algorithm with complexity $\mathrm{O}(n^2)$]

Fast gradient projection for (dual) QP

(Patrinos, Bemporad, 2012)

- Main on-line operations involve only **simple linear algebra**

- Few lines of **(Embedded) MATLAB** or **Python/Numpy** code !

- Operations can be easily **parallelized** (for example on **GPU**)

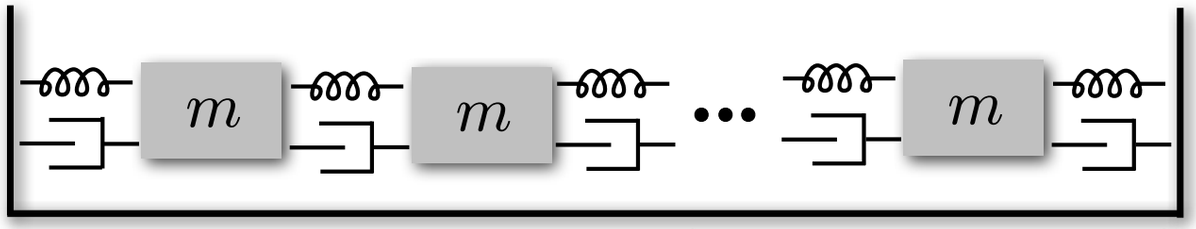
- Each iteration has complexity $\mathrm{O}(N^2)$ $[N = \text{prediction horizon}]$ (matrix-vector product)

- Using Riccati-like iterations, complexity gets down to $O(N)$

- Tight bounds exist on maximum number of iterations

```
while keepgoing && (i<maxiter),

    beta=(i-1)/(i-2).*(i>0);

    w=y+beta*(y-y0);
    z=-(iMG*w+iMc);
    s=GLz-bL;

    y0=y;

    % Check termination conditions
    if all(s<=epsGL),
        gapL=-w'*s;
        if gapL<=epsVL,
            return
        end
    end

    y=max(w+s,0);

    i=i+1;
end
```

Numerical results



(Wang, Boyd, 2008)

CPU time [ms]

M	N	GPAD		Gurobi 5.0	
		avg	max	avg	max
2	10	**0.40**	1.09	**4.48**	
2	30	0.76	3.22	5.64	9.77
4	10	1.41	2.63	5.61	7.49
6	30	8.35	15.52	15.22	23.57
8	20	8.65	14.95	16.42	18.36
15	10	11.00	16.62	21.52	22.95
20	20	39.01	82.32	82.71	134.05
30	30	**128.1**	218.13	**202.35**	465.80

number of iterations

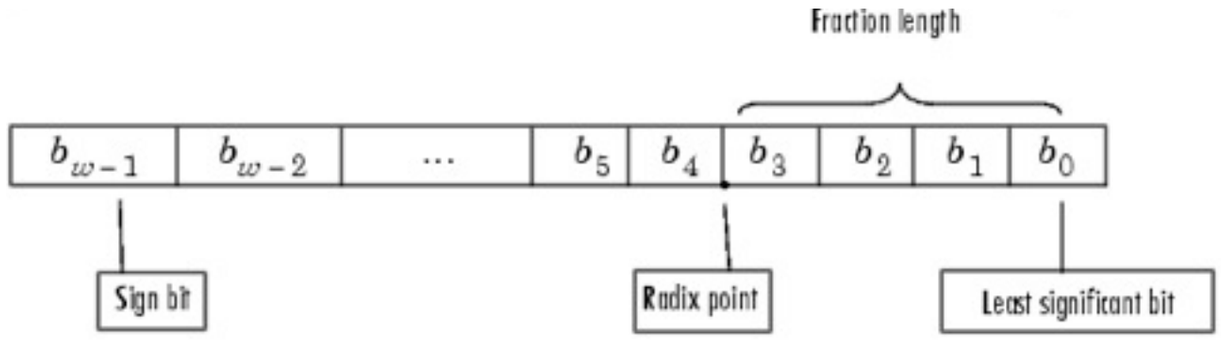
M	N	GPAD		Gurobi 5.0	
		avg	max	avg	max
2	10	19.90	60	5.42	7
2	30	20.40	100	5.50	7
4	10	41.80	80	6.05	8
6	30	52.20	100	6.57	8
8	20	52.20	90	6.51	8
15	10	54.00	80	6.56	7
20	20	61.60	140	6.85	8
30	30	**62.00**	100	**6.94**	8

QP **interior point** solver of `Gurobi 5.0`

tolerances: $\epsilon_V = \epsilon_g = 10^{-3}$ (results averaged on 100 random states p)

Solving QPs in fixed-point arithmetics

- Fixed-point arithmetics is very attractive for embedded control:
  - Computations are fast and cheap
  - Hardware support in all platforms



- Cons:
  - Accumulation of quantization errors
  - Limited range (numerical overflow)

- Dual gradient projection method for QP works very well in fixed-point !

(Patrinos, Guiggiani, Bemporad, ECC 2013)

Solving QPs in fixed-point arithmetics

(Patrinos, Guiggiani, Bemporad, 2013)

Result 1: Convergence to feasibility

$$\max_i g_i(z_k) \le \frac{\alpha^2}{\alpha-1}\frac{LD^2}{2(k+1)} + \delta_\alpha^g \qquad \alpha > 1 \ \text{(e.g., } \alpha = 2\text{)}$$

Result 2: Convergence to optimality

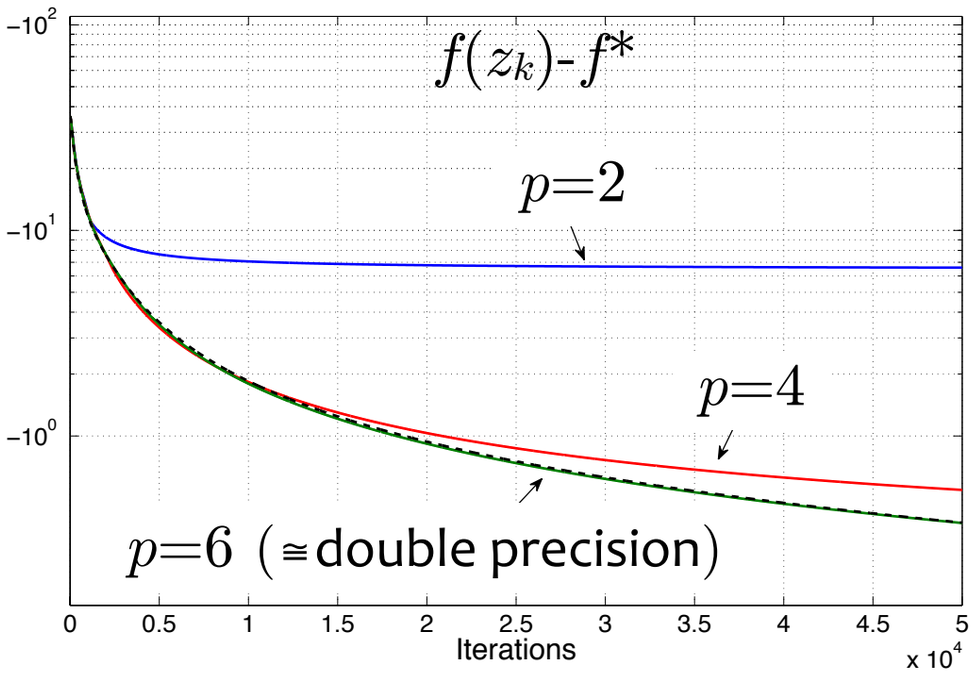
$$f(z_k) - f^\star \le \frac{L}{2(k+1)}(\|y^\star\|^2 + \|y_0\|^2) + \delta_\alpha$$

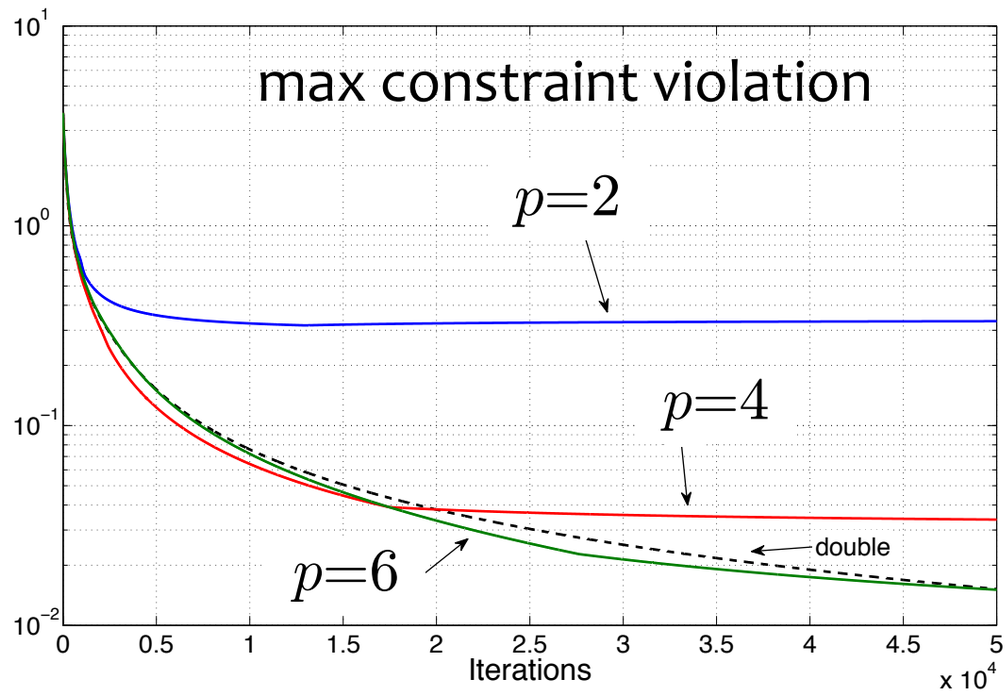
Result 3: Design guidelines

$$p \ge \log_2 \frac{m\sqrt{n}}{\sqrt{\frac{\varepsilon}{L_V} + \frac{n}{m}\left(\frac{2D}{L_V}\right)^2} - \sqrt{\frac{n}{m}}\frac{2D}{L_V}} - 1$$

fractional bits

desired solution "quality"





Feasibility of QP problem

$$
\begin{aligned}
\min_{z} \quad & \sum_{k=0}^{N-1} \frac{1}{2}(y_k - r(t)'S(y_k - r(t)) + \frac{1}{2}\Delta u_k' T \Delta u_k \\
\text{subj. to} \quad & u_{\min} \le u_k \le u_{\max}, \;\; k = 0, \ldots, N-1 \\
& \Delta u_{\min} \le \Delta u_k \le \Delta u_{\max}, \;\; k = 0, \ldots, N-1 \\
& y_{\min} \le y_k \le y_{\max}, \;\; k = 1, \ldots, N
\end{aligned}
$$

- **Feasibility**: Will the QP problem be feasible at all sampling instants t ?

- **Input constraints only:** no feasibility issues !

- **Hard output constraints:**
  - When $N<\infty$ there is no guarantee that the QP problem will remain feasible at all future time steps t
  - $N=\infty$ ⟹ infinite number of constraints !
  - Maximum output admissible set theory: $N<\infty$ is enough !

(Gilbert & Tan, 1991) (Kerrigan & Maciejowski, 2000) (Chmielewski & Manousiouthakis, 1996)

Soft constraints

- **Mail idea:** use **soft output constraints**

$$\min_z \quad \sum_{k=0}^{N-1} \frac{1}{2}(y_k - r(t)'S(y_k - r(t)) + \frac{1}{2}\Delta u_k' T \Delta u_k + \rho_\epsilon \epsilon^2$$

$$\text{subj. to} \quad u_{\min} \le u_k \le u_{\max}, \;\; k = 0, \ldots, N-1$$
$$\Delta u_{\min} \le \Delta u_k \le \Delta u_{\max}, \;\; k = 0, \ldots, N-1$$
$$y_{\min} - \epsilon V_{\min} \le y_k \le y_{\max} + \epsilon V_{\max}, \;\; k = 1, \ldots, N$$

ϵ = "panic" variable

$$z = [\Delta u'(0) \;\; \Delta u'(1) \;\; \ldots \;\; \Delta u'(N-1) \;\; \epsilon]'$$

$$\rho_\epsilon \gg W^y, W^{\Delta u}$$

$V_{\min}$, $V_{\max}$ = vectors with entries ≥ 0 (the larger the i-th entry of vector V, the relatively softer the corresponding constraint)

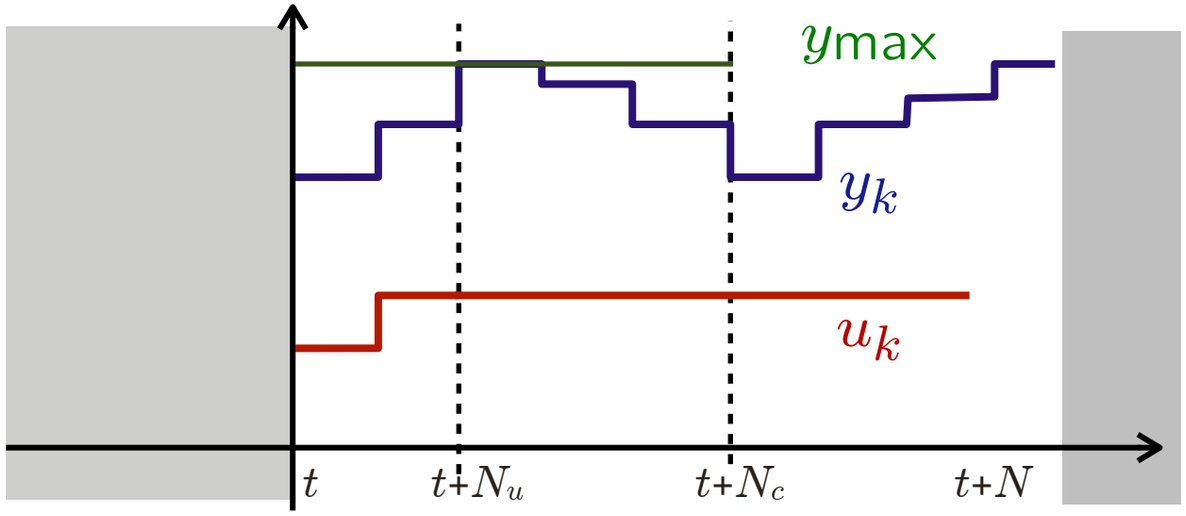
- QP problem always feasible, in spite of modeling errors, disturbances, wrong MPC setup (e.g., prediction horizon is too short)

Limit QP complexity: different prediction horizons

$$\min_z \quad \sum_{k=0}^{N-1} \frac{1}{2}(y_k - r(t)'S(y_k - r(t)) + \frac{1}{2}\Delta u_k' T \Delta u_k + \rho_\epsilon \epsilon^2$$

$$\text{subj. to} \quad u_{\min} \le u_k \le u_{\max}, \;\; k = 0, \dots, N-1$$
$$\Delta u_{\min} \le \Delta u_k \le \Delta u_{\max}, \;\; k = 0, \dots, N-1$$
$$\Delta u_k = 0, \;\; k = N_u, \dots, N-1$$
$$y_{\min} - \epsilon V_{\min} \le y_k \le y_{\max} + \epsilon V_{\max}, \;\; k = 1, \dots N_c,$$

- The **input horizon** N_u limits the number of free variables
  - Loss of performance
  - Decreased computation time (QP is smaller, less primal variables)



- The **output constraint horizon** N_c limits the number of constraints
  - Higher chances of violating output constraints
  - Decreased computation time (QP is smaller, less dual variables)

On-Line vs off-line optimization

$$\begin{aligned} \min_z \quad & \tfrac{1}{2} z' H z + x'(t) F z \\ \text{s.t.} \quad & G z \leq W + S x(t) \end{aligned}$$

- **On-line** optimization: given $x(t)$ solve the problem at each time step t (the control law $u{=}u(x)$ is **implicitly** defined by the QP solver)

  → Quadratic Program (QP)

- **Off-line** optimization: solve the QP **for all** $x(t)$ to find the control law $u{=}u(x)$ **explicitly**

  → multi-parametric Quadratic Program (mp-QP)

Explicit model predictive control

$$\begin{aligned} \min_z \quad & \tfrac{1}{2} z' H z + x'(t) F z \\ \text{s.t.} \quad & G z \le W + S x(t) \end{aligned}$$

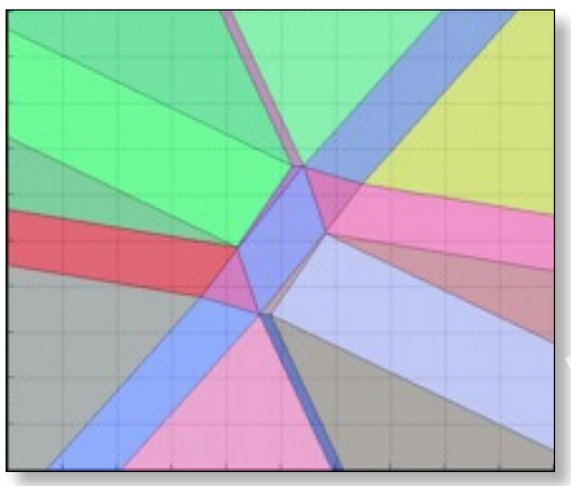
Idea: solve the QP **for all** $x(t)$ within a given range of $\mathbb{R}^n$ **off-line**

⇒ multi-parametric programming problem

Linear MPC is a **continuous and piecewise affine** control law !

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

(Bemporad, Morari, *et al.*, 2002)



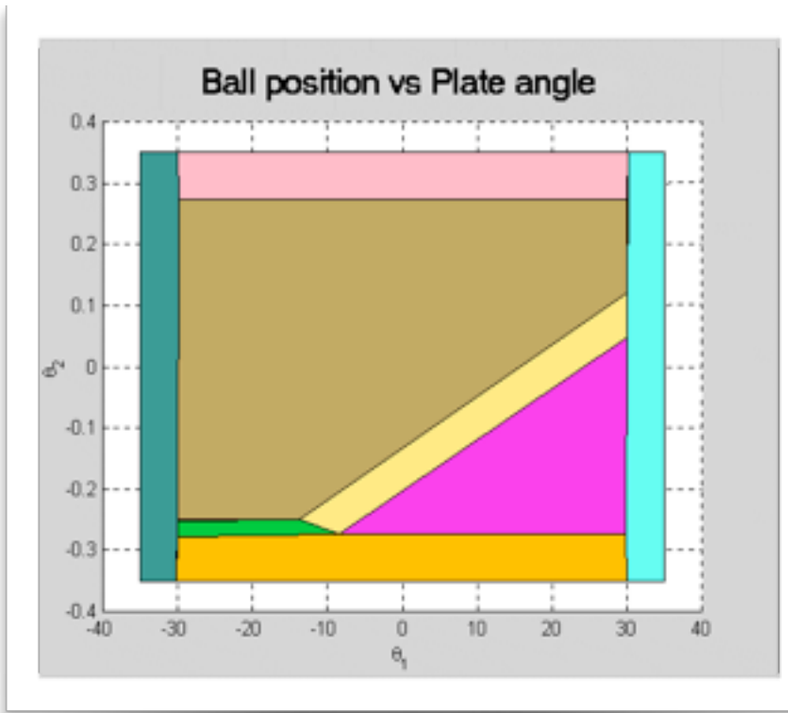
```
while ((num<EXPCON_REG) && check) {
    isinside=1;
    while ((i1<=i2) && isinside) {
        aux=0;
        for (j=0;j<EXPCON_NTH;j++)
            aux+=(double)EXPCON_H[i1+j*EXPCON_NTH][j];
        if (aux>(double)EXPCON_K[i1])
            isinside=0; /* get out of the loop, th violates
        else
            i1++;
    }
    if (isinside) {
        check=0; /* region found ! */
        infeas=0;
    }
    else {
        num++;
        i1=i2+1;              /* get next delimiter i1 */
        i2+=EXPCON_len[num]; /* get next delimiter i2 */
    }

}
```

It's just a while loop!

Explicit MPC of a ball & plate system

Two explicit MPC controllers

x-axis = 22 regions,
y-axis = 23 regions



sample time = 20ms

Explicit MPC for idle speed control

(Di Cairano, Yanakiev, Bemporad, Kolmanovsky, Hrovat, 2011)

- Ford pickup truck, V8 4.6L gasoline engine

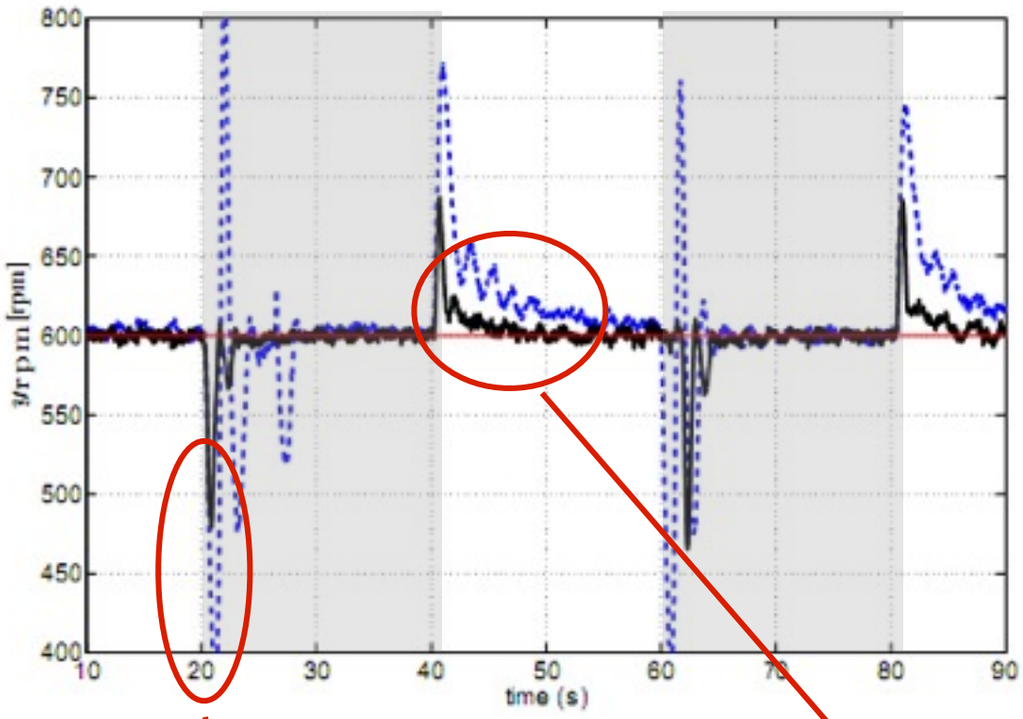
- Process:
  - *1 output* (engine speed) to regulate
  - *2 inputs* (airflow, spark advance)
  - input *delays*

- Objectives and specs:
  - **regulate engine speed** at constant rpm
  - **saturation limits** on airflow and spark
  - **lower bound** on engine speed ($\geq$450 rpm)

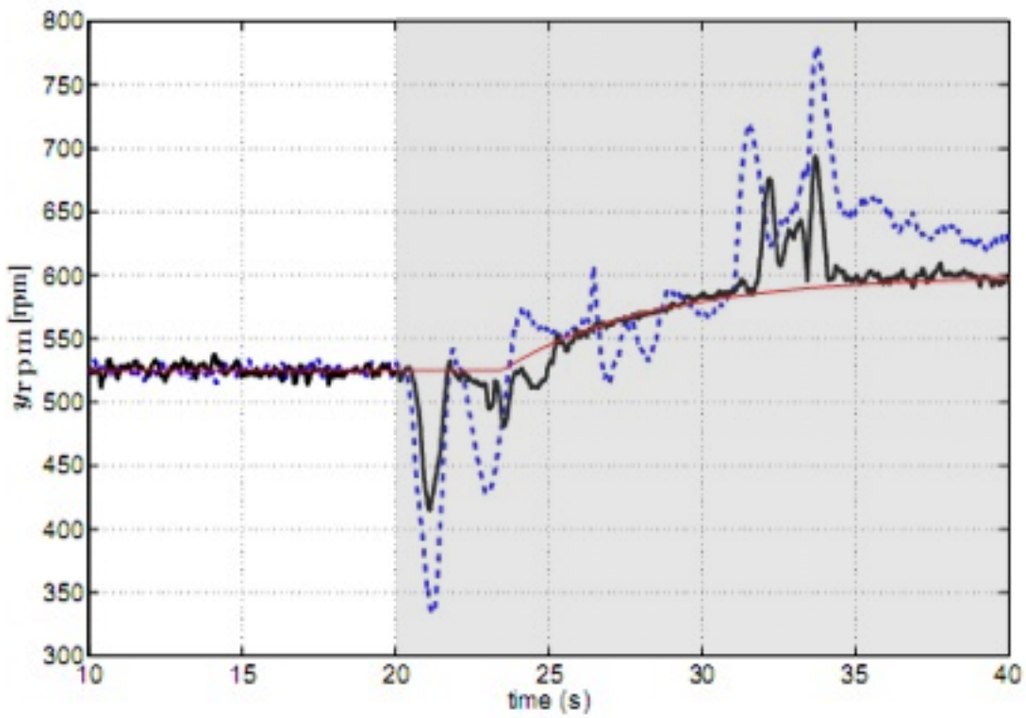
- Problem suitable for MPC design (Hrovat, 1996)

Explicit MPC for idle speed - Experiments

(Di Cairano, Yanakiev, Bemporad, Kolmanovsky, Hrovat, 2011)



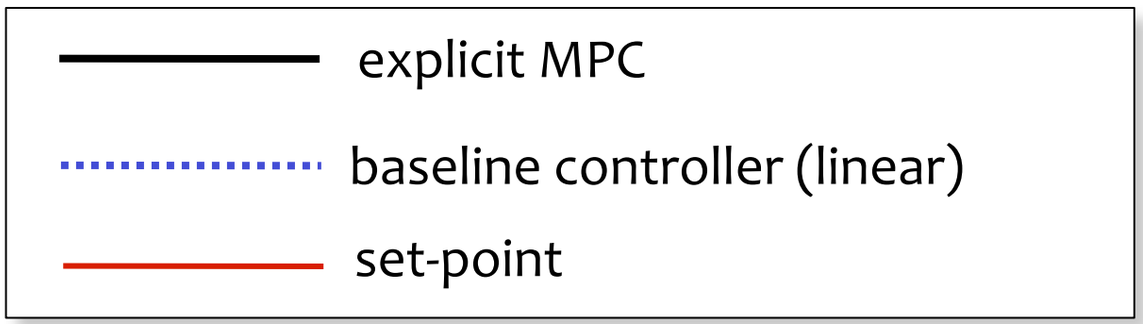
Load torque (power steering)



Power steering + air conditioning

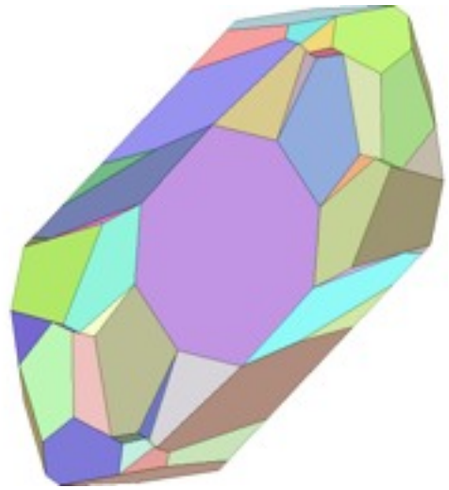
peak reduced by 50%

convergence 10s faster



Complexity of explicit MPC

- Number of regions depends on number of possible combinations of active constraints

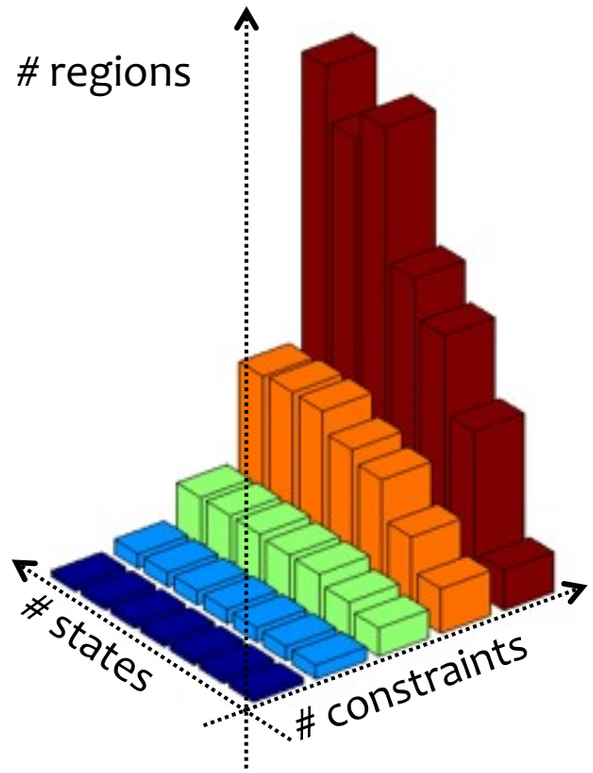


- Weak dependence on number of states and references

- On-line QP vs explicit MPC comparison:

$2N$	QP (ms) average	worst	explicit (ms) average	worst	regions	[storage kb]
4	1.1	1.5	0.005	0.1	25	16
8	1.3	1.9	0.023	1.1	175	78
20	2.5	2.6	0.038	3.3	1767	811
30	5.3	7.2	0.069	4.4	5162	2465
40	10.9	13.0	0.239	15.6	11519	5598

(Intel Centrino 1.4 GHz)



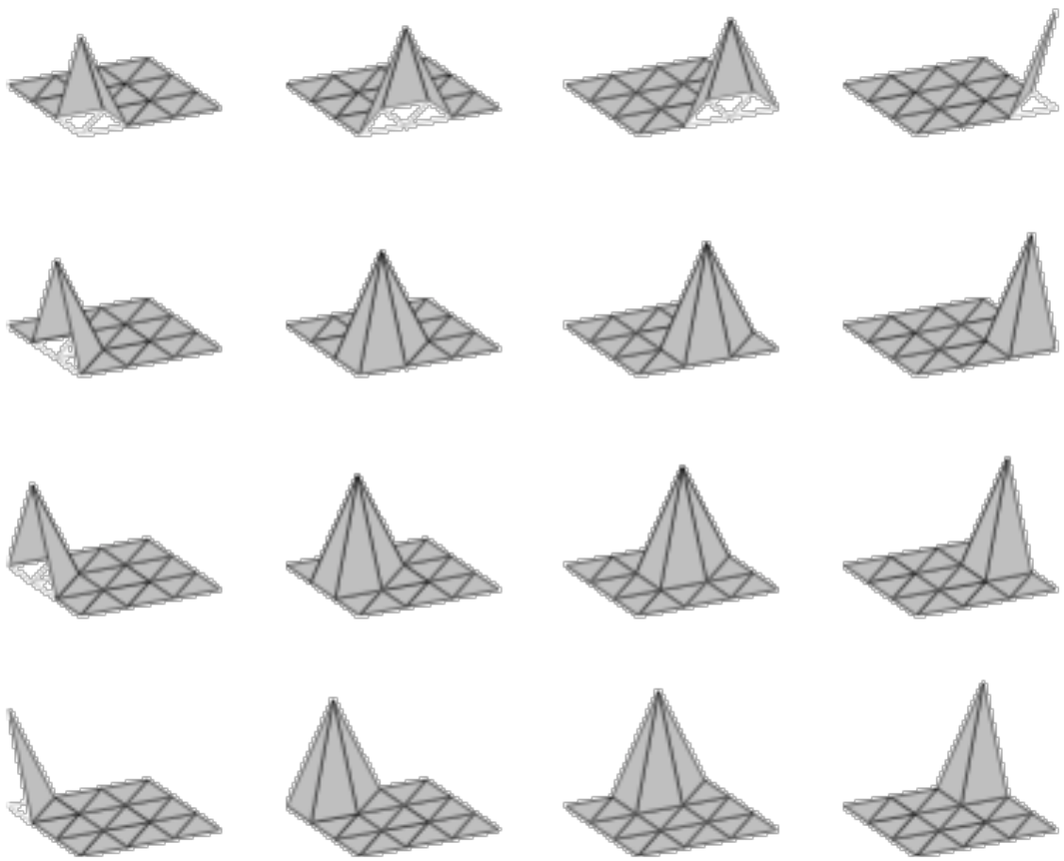
- Explicit approach typically limited to LTI-MPC problems with **6÷8 free control moves** and **8÷12 parameters** (states+references)

- Embedded QP methods are preferable otherwise

PWA approximation of MPC over simplices

- ***Approximate*** a given linear MPC controller by using **canonical PWA** functions over **simplicial partitions** (**PWAS**)

(Bemporad, Oliveri, Poggi, Storace , 2011)



(Julian, Desages, Agamennoni, 1999)

$$\hat{u}(x) = \sum_{k=1}^{N_v} w_k \, \phi_k(x) = w' \phi(x)$$

approximate MPC law

Weights w_k optimized **off-line**
to best approximate a given MPC law

http://www.mobydic-project.eu/

PWA approximation of MPC over simplices

- **Extremely cheap**: **PWAS functions can be directly implemented on FPGA**, or even **ASIC** (Application Specific Integrated Circuits)

- **Extremely fast** computations (**10-100 nanoseconds**)

fit criterion

Control	p	Latency (A - B) [ns]
L^2	7	170 - 31
	31	238 - 45
	63	272 - 46
L^∞	7	170 - 31
	31	238 - 45
	63	272 - 46

online time to compute MPC (Xilinx Spartan 3 FPGA)

Architecture A: mainly serial

Architecture B: fully parallel

 # partitions per dimension

Exact explicit MPC: 52 regions
383 ns (avg) - 486 ns (max)

MIMO system dynamics

$$x_{k+1} = \begin{bmatrix} 1.2 & 1 \\ 0 & 1.1 \end{bmatrix} x_k + \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix} u_k$$

$$y_k = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} x_k$$

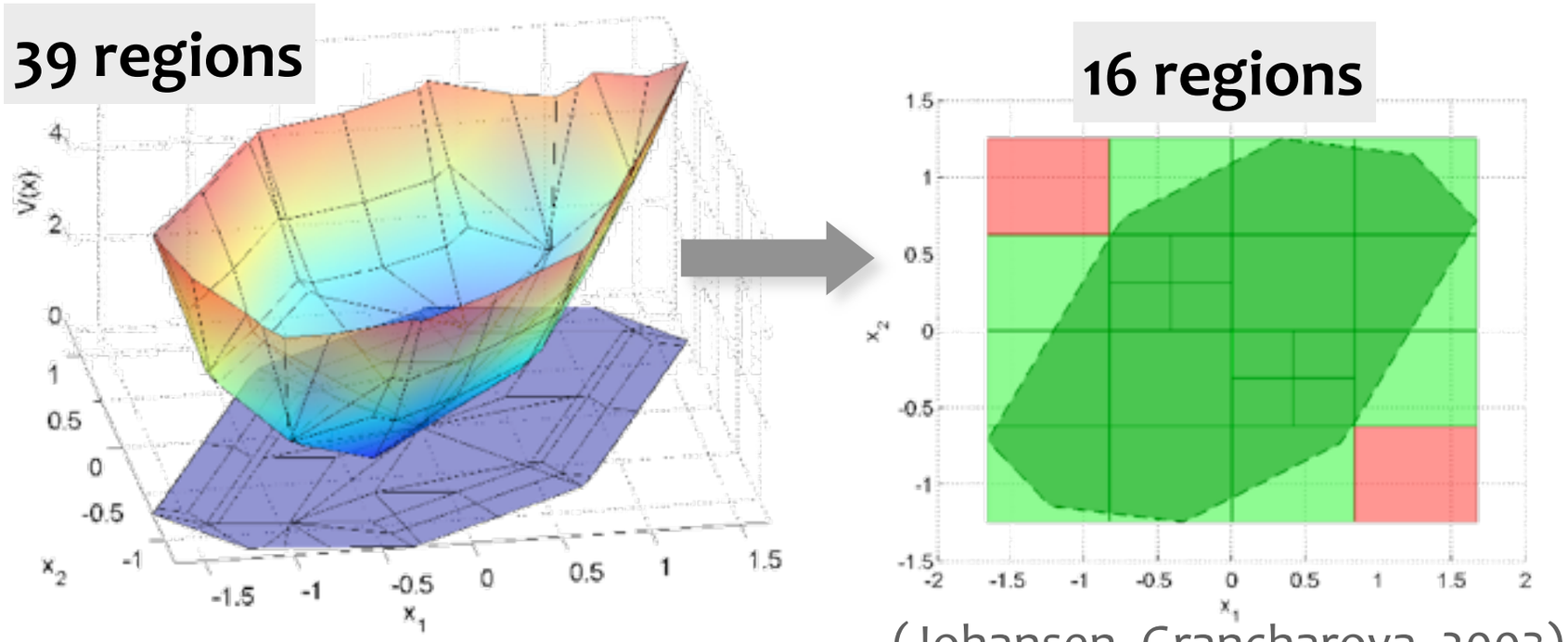
subject to saturation on u and y

- Closed-loop stability results are available (Bemporad, Oliveri, Poggi, Storace , 2011) (Rubagotti, Barcelli, Bemporad, 2012)

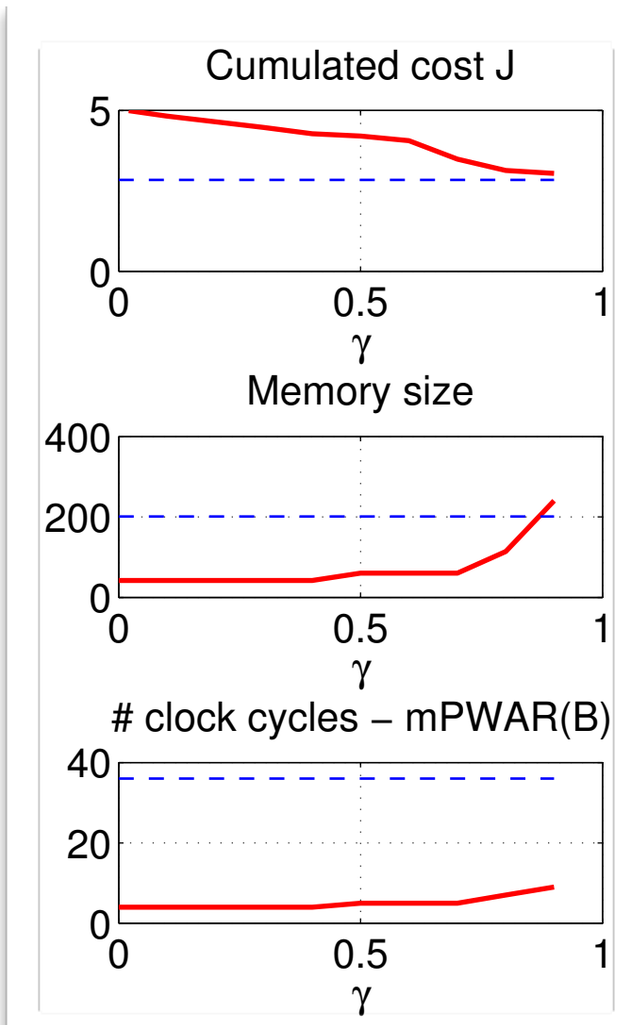
✖ Curse of dimensionality (wrt to state dimension)

PWA approximation of MPC over rectangles

- ***Approximate*** a given linear MPC controller by using **regular PWA** functions over **hyper-rectangular partitions** (**PWAS**) (Liang, Heemels, Bemporad, 2011)



(Johansen, Grancharova, 2003)



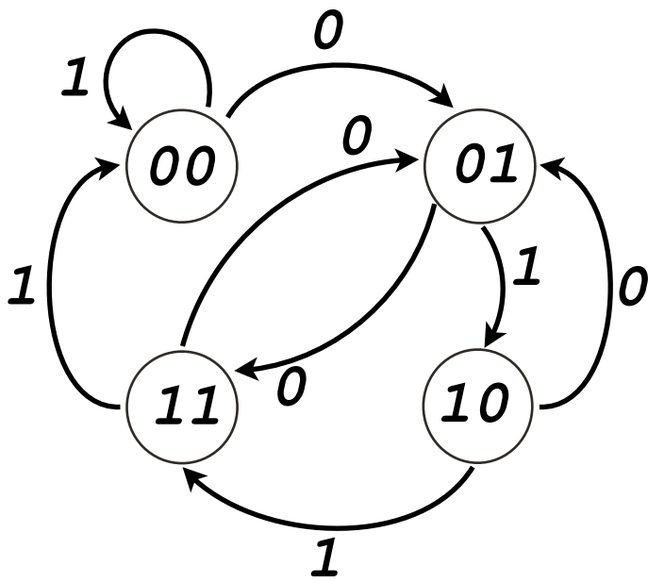
- Key idea: **use optimal PWA cost as a control Lyap. fnc**

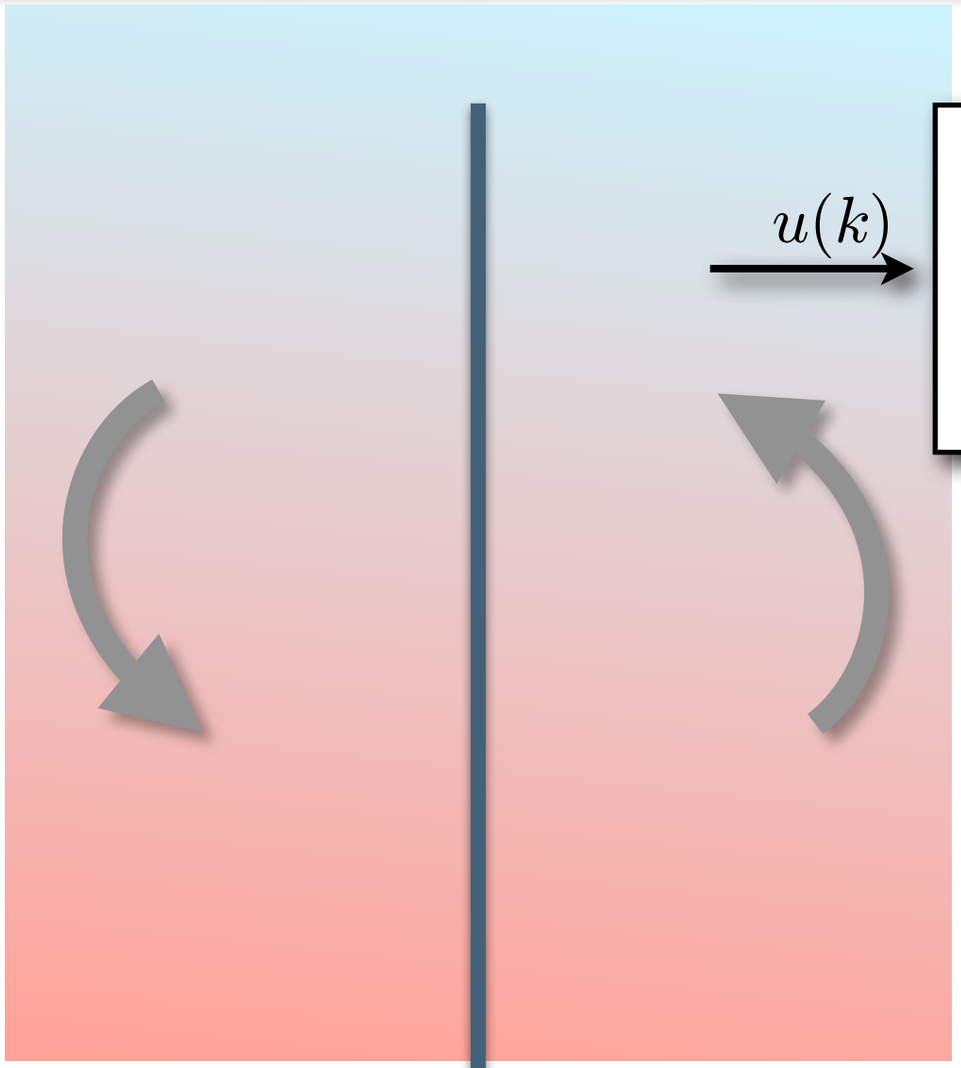
- Main features of suboptimal control law:

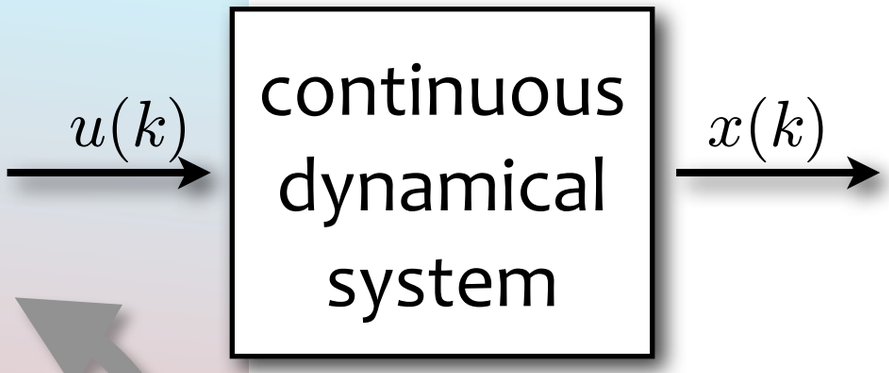
  - Can be computed (off-line) by solving a **linear program** (LP)
  - Closed-loop **stability** and degree of **performance loss** are guaranteed a priori
  - Fulfillment of **input and state constraints** is guaranteed a priori
  - Tradeoff between controller complexity and suboptimality is selectable

Hybrid MPC

Hybrid dynamical systems







hybrid dynamical system

- Variables are **discrete-valued**

$$x \in \{0,1\}^{n_b},\ u \in \{0,1\}^{m_b}$$

- Dynamics = **finite state machine**
- **Logic** constraints

- Variables are **real-valued**

$$x \in \mathbb{R}^{n_c},\ u \in \mathbb{R}^{m_c}$$

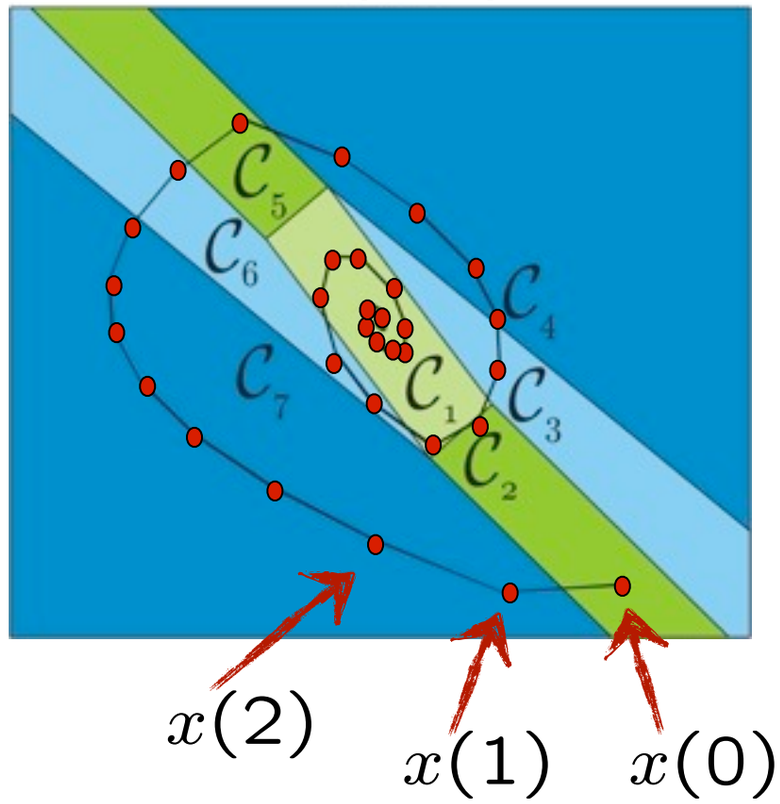
- **Difference/differential equations**
- **Linear inequality** constraints

Piecewise affine systems

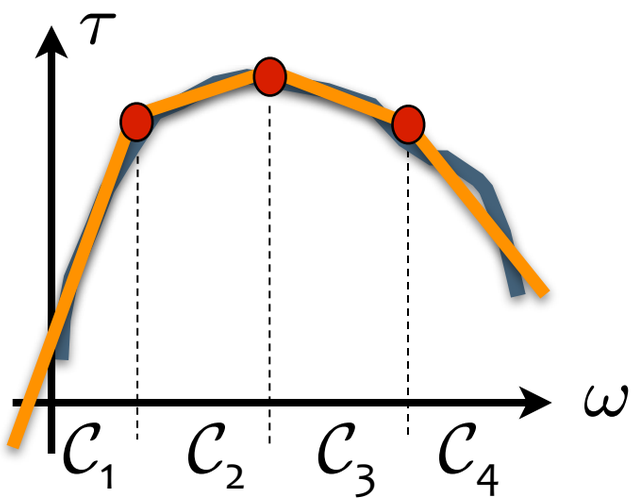
$$\begin{aligned} x(k+1) &= A_{i(k)}x(k) + B_{i(k)}u(k) + f_{i(k)} \\ y(k) &= C_{i(k)}x(k) + D_{i(k)}u(k) + g_{i(k)} \\ i(k) \text{ s.t. } & H_{i(k)}x(k) + J_{i(k)}u(k) \le K_{i(k)} \end{aligned}$$

$x \in \mathbb{R}^n,\ u \in \mathbb{R}^m,\ y \in \mathbb{R}^p,\ i \in \{1,\dots,s\}$

state+input space



Can approximate nonlinear and/or discontinuous dynamics arbitrarily well



Discrete Hybrid Automaton (DHA)

(Torrisi, Bemporad, 2004)

discrete

Event Generator

$$[\delta_e = 1] \leftrightarrow [Hx_c + Ku_c \leq W]$$

$\delta_e = 1$

$\delta_e = 0$

$\delta_e(k)$

$x_c(k)$

Finite State Machine

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

$u_\ell(k)$

discrete time counter

$u_c(k)$

Switched Affine System

$$x_c(k+1) = A_i x_c(k) + B_i u_c(k) + f_i$$

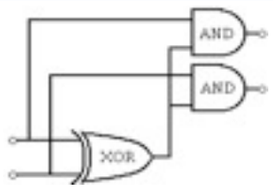
$$i = i(k)$$

$x_c(k)$

$x_\ell(k)$

mode $i(k)$

Mode Selector



$u_\ell(k)$

$\delta_e(k)$

$$i = f_{\mathsf{M}}(x_\ell, u_\ell, \delta_e)$$

continuous

$x_\ell \in \{0,1\}^{n_b}$ = binary state

$u_\ell \in \{0,1\}^{m_b}$ = binary input

$\delta_e \in \{0,1\}^{n_e}$ = event variable

$x_c \in \mathbb{R}^{n_c}$ = real-valued state

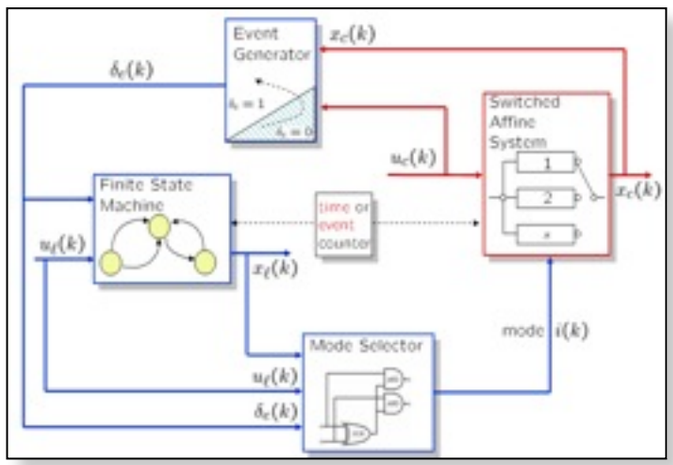
$u_c \in \mathbb{R}^{m_c}$ = real-valued input

$i \in \{1, \ldots, s\}$ = current mode

Mixed Logical Dynamical (MLD) systems

- Any **logic formula** involving Boolean variables can be translated into a set of **integer linear (in)equalities**

(Raman, Grossmann, 1991)



Example: δ_1 OR δ_2 = TRUE $\Leftrightarrow$ $\delta_1 + \delta_2 \geq 1$

HYSDEL modeling language

(Torrisi, Bemporad, 2004)

Mixed Logical Dynamical (MLD) system

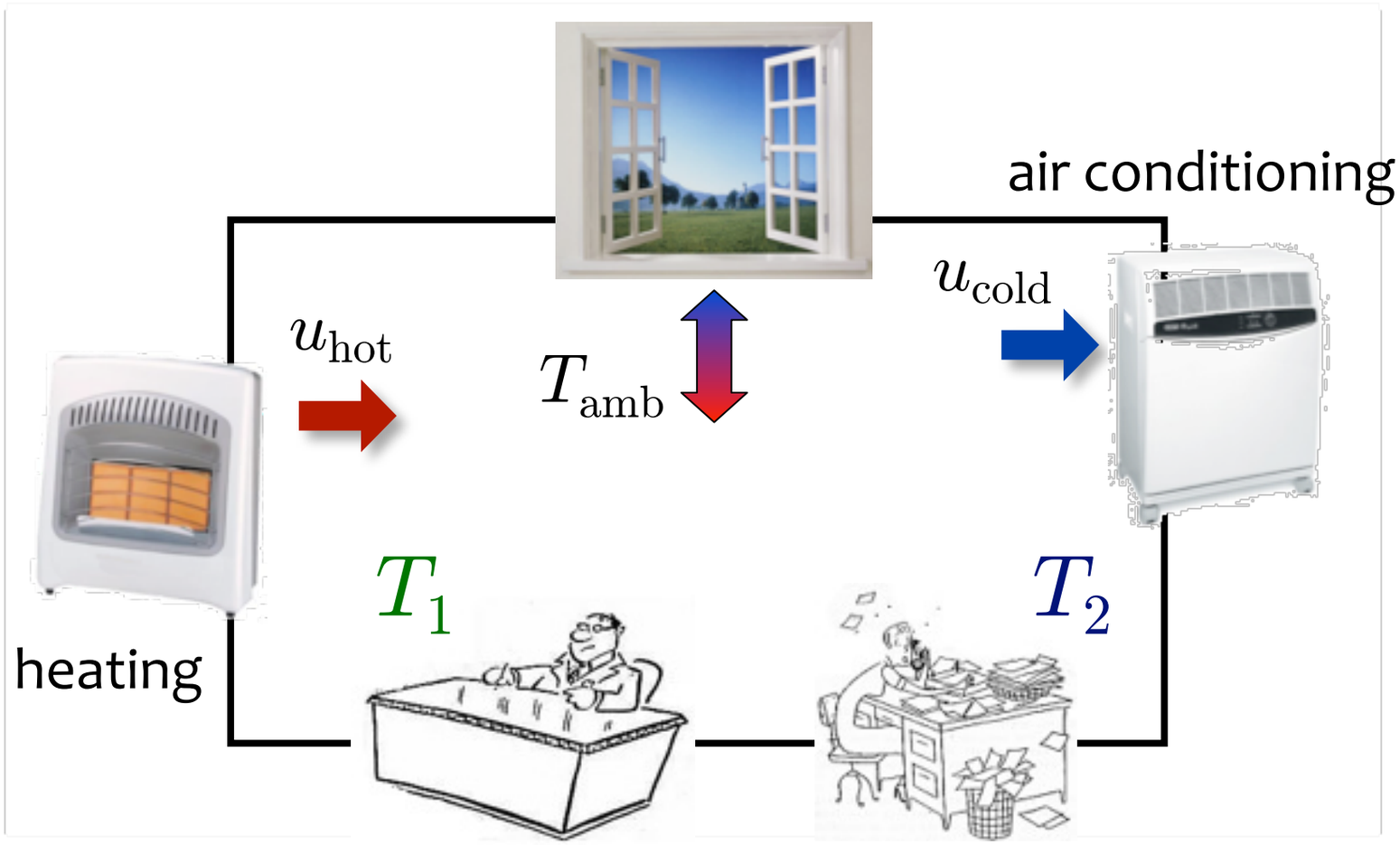
$$\begin{cases} x_{k+1} = Ax_k + B_1u_k + B_2\delta_k + B_3z_k + B_5 \\ y_k = Cx_k + D_1u_k + D_2\delta_k + D_3z_k + D_5 \\ E_2\delta_k + E_3z_k \leq E_4x_k + E_1u_k + E_5 \end{cases}$$

(Bemporad, Morari 1999)

- x,u,y,z,δ contain **mixed-integer** (=**real** and **binary**) components
- **MLD** systems and **PWA** systems are equivalent

(Bemporad, Ferrari-Trecate, Morari, 2000) (Heemels, De Schutter, Bemporad, 2001) (Bemporad, 2004)

Example: Room temperature control



discrete dynamics

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

continuous dynamics

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

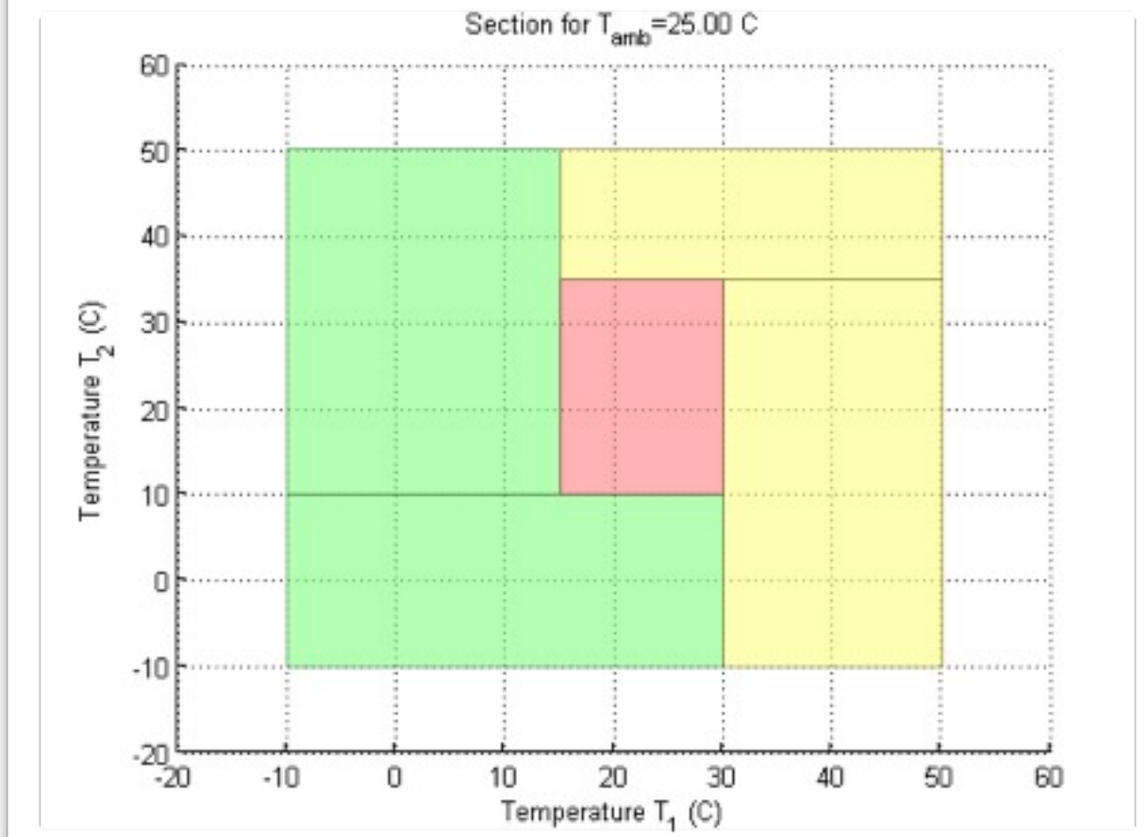
$$i = 1, 2$$

Example: Room temperature control

```
SYSTEM heatcool {

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

- Model described in **HYSDEL** and converted to **MLD** form (**20 mixed-integer inequalities**)

- Equivalent PWA model consists of **5 regions**



heater on | both off | A/C on

MPC of hybrid systems

hybrid MLD model

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

performance index

$$\min x_N'Px_N + \sum_{k=0}^{N-1} x_k'Qx_k + u_k'Ru_k$$

$$\begin{aligned} R &= R' \succ 0 \\ Q &= Q' \succeq 0 \\ P &= P' \succeq 0 \end{aligned}$$

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

$$\xi \in \mathbb{R}^c \times \{0,1\}^b$$

continuous and binary components

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

linear and logic constraints

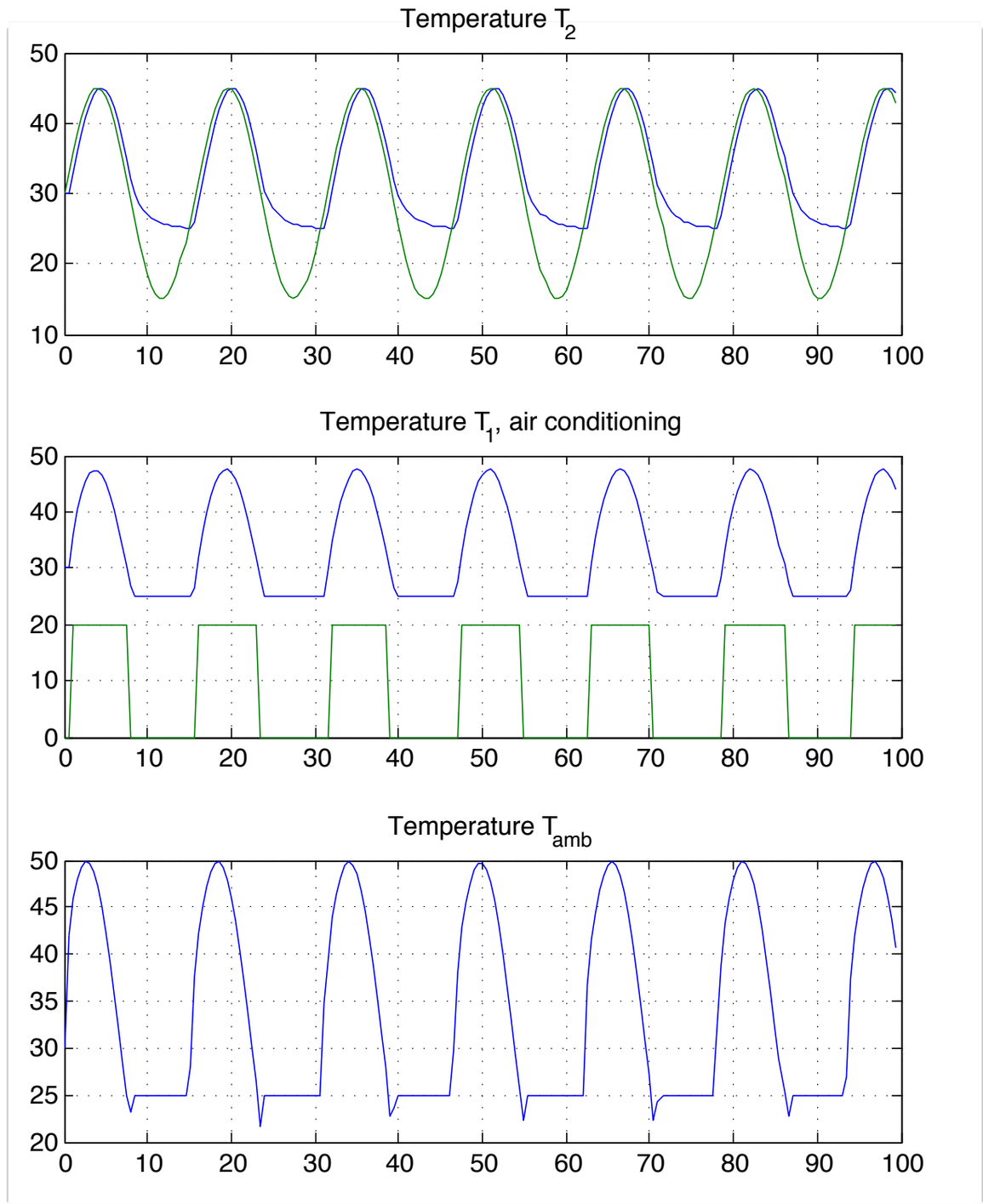
MPC implemented by solving a **Mixed-Integer Quadratic Program (MIQP)**

Alternative: Mixed-Integer Linear Program (**MILP**) formulation

Example: Room temperature control

- MPC problem formulation

$$\begin{aligned} \min \quad & \sum_{k=1}^{2} |x_2(k) - r| \\ \text{s.t.} \quad & x_1(k) \geq 25, \ \ k = 1,2 \\ & \text{hybrid dynamics} \end{aligned}$$



- MPC optimization problem (MILP)

```
20 optimization variables
   (8 continuous, 12 binary)

46 mixed-integer linear inequalities
```

CPU time = **1.3 ms** per time step

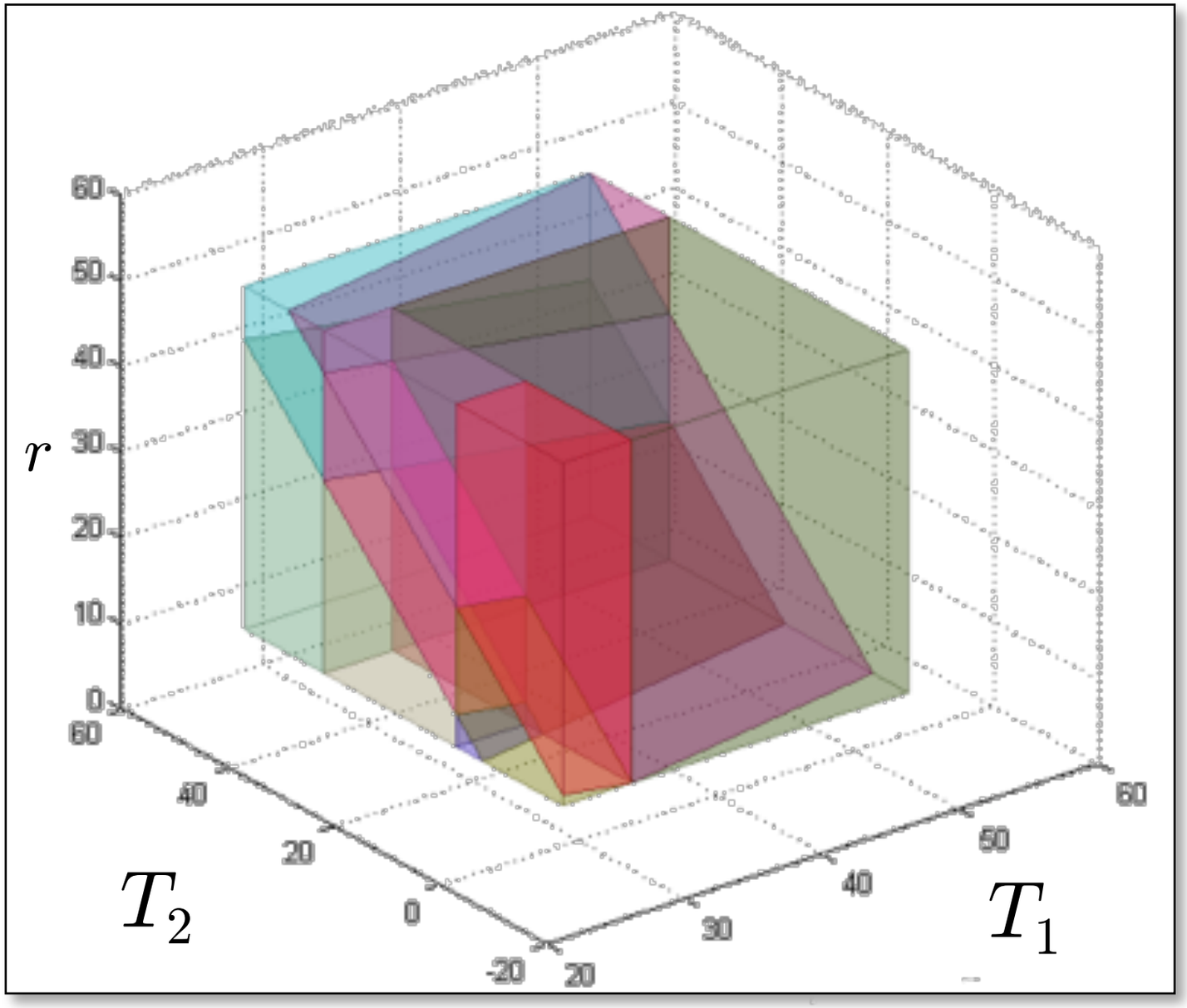
(IBM CPLEX 11.2, this Mac)

Example: Room temperature control

- Explicit solution

  (Borrelli, Baotic, Bemporad, Morari, 2005)
  (Mayne, Rakovic, 2002)
  (Alessio, Bemporad, 2006)

12 polyhedral regions
and linear gains



384 numbers to store
in memory

CPU time = **0.8 ms** (compiled C-code, this Mac)

Note: explicit form does not change the control law at all !

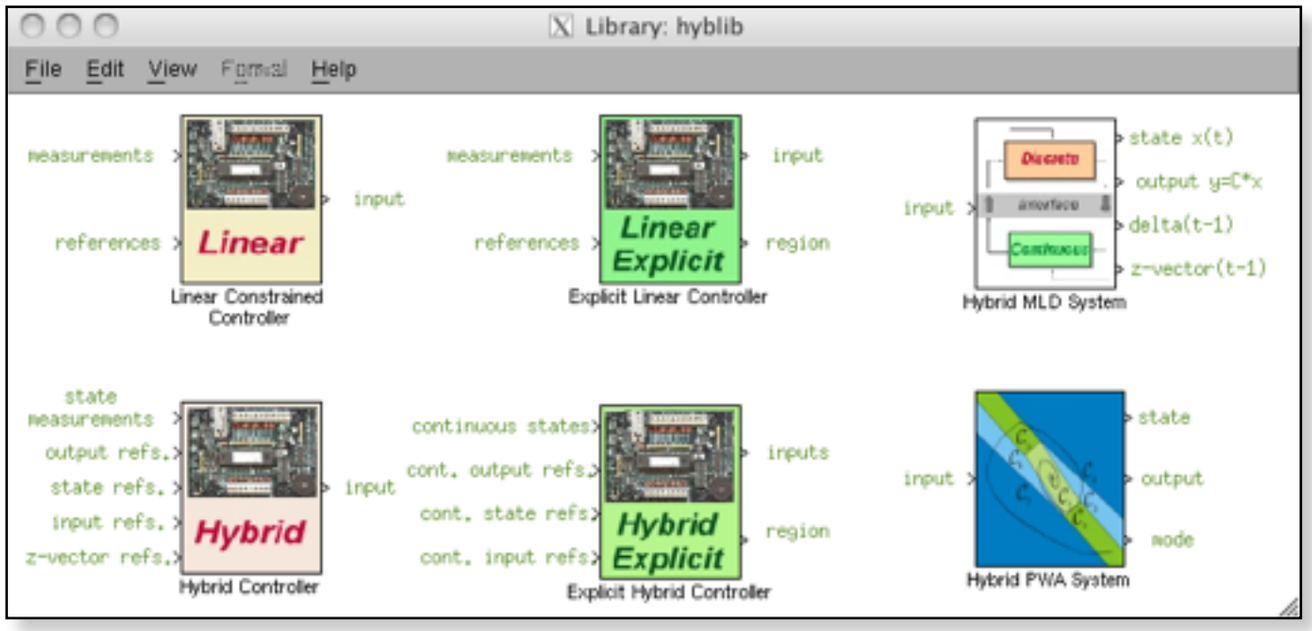
Hybrid Toolbox for MATLAB

(Bemporad, 2003-2013)

Features:

Supported by

- **Explicit** MPC control (via multi-parametric programming)
- Simulink library
- C-code generation



- **Hybrid** models: design, simulation, verification
- Control design for linear systems w/ constraints and hybrid systems
- Interfaces to several QP/LP and Mixed-Integer Programming solvers

5000+ download requests since October 2004

http://cse.lab.imtlucca.it/~bemporad/hybrid/toolbox/

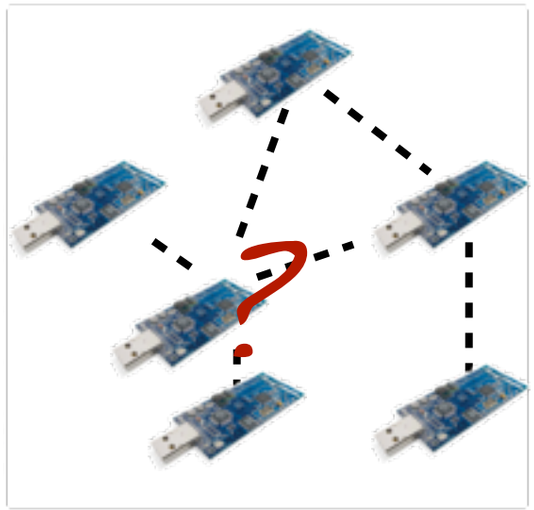
Stochastic MPC

Stochastic MPC

- In many control problems decisions must be taken under **uncertainty**


wind/solar power

prices


wireless networks

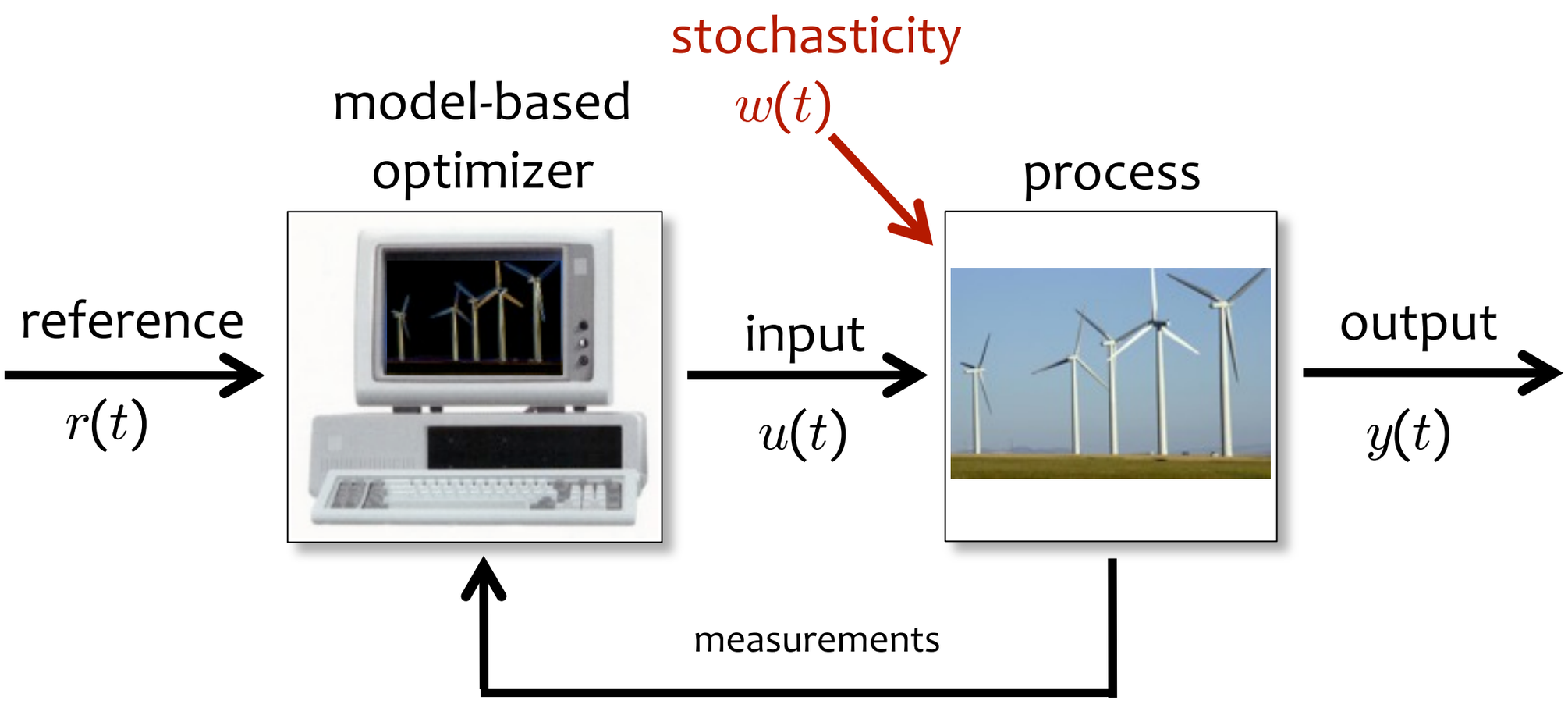

demand


human (inter)action

- **Robust** control approaches do not model uncertainty (only assume that is bounded) and pessimistically consider the worst case

- **Stochastic models** provide instead additional information about uncertainty

Stochastic Model Predictive Control (SMPC)



*Use a **stochastic** dynamical **model** of the process to **predict** its possible future evolutions and choose the "best" **control** action*

Stochastic Model Predictive Control

- At time t: solve a **stochastic optimal control** problem over a finite future horizon of N steps:

$$\begin{aligned}
\min_{u} \quad & E_w\left[\sum_{k=0}^{N-1} \ell(y_k - r(t+k), u_k)\right] \\
\text{s.t.} \quad & x_{k+1} = f(x_k, u_k, w_k) \\
& y_k = g(x_k, u_k, w_k) \\
& u_{\min} \le u_k \le u_{\max} \\
& y_{\min} \le y_k \le y_{\max}, \quad \forall w \\
& x_0 = x(t)
\end{aligned}$$

$x(t)$ = process state

$u(t)$ = manipulated vars

$y(t)$ = controlled output

$w(t)$ = **stochastic disturbances**

- Only apply the first optimal move $u^*(t)$, discard $u^*(t+1)$, $u^*(t+2)$, …

- At time $t+1$: Get new measurement $x(t+1)$, repeat the optimization. And so on …

Linear stochastic MPC w/ discrete disturbance

- Linear stochastic prediction model

$$x(t+1) = A(w(t))x(t) + B(w(t))u(t) + H(w(t))$$

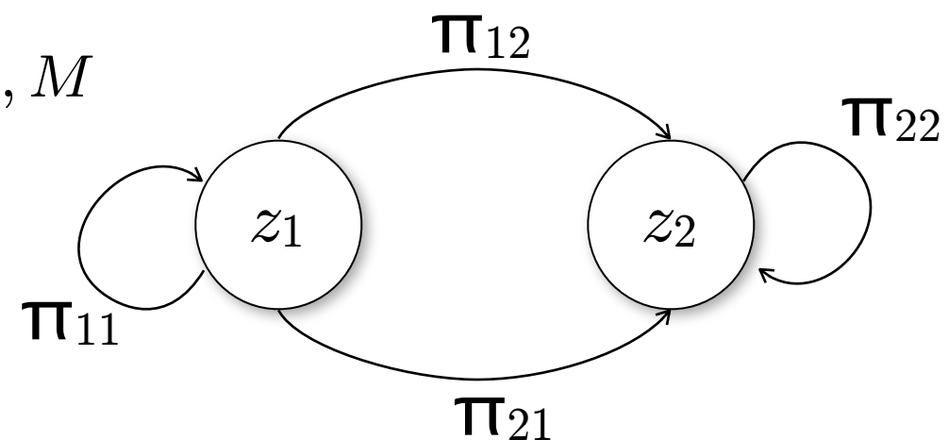
- Discrete disturbance

$$w(t) \in \{w_1, \ldots, w_s\} \qquad p_j(t) = \mathbf{Pr}[w(t) = w_j] \qquad \sum_{j=1}^{s} p_j(t) = 1, \ \forall t \geq 0$$

- Probabilities $p_j(t)$ can have their own dynamics. Example: Markov chain

$$\pi_{ih} = \mathbf{Pr}[z(t+1) = z_h \mid z(t) = z_i], \ \ i, h = 1, \ldots, M$$

$$p_j(t) = \begin{cases} e_{1j} & \text{if} \quad z(t) = z_1 \\ \vdots & \quad\ \vdots \\ e_{Mj} & \text{if} \quad z(t) = z_M \end{cases}$$



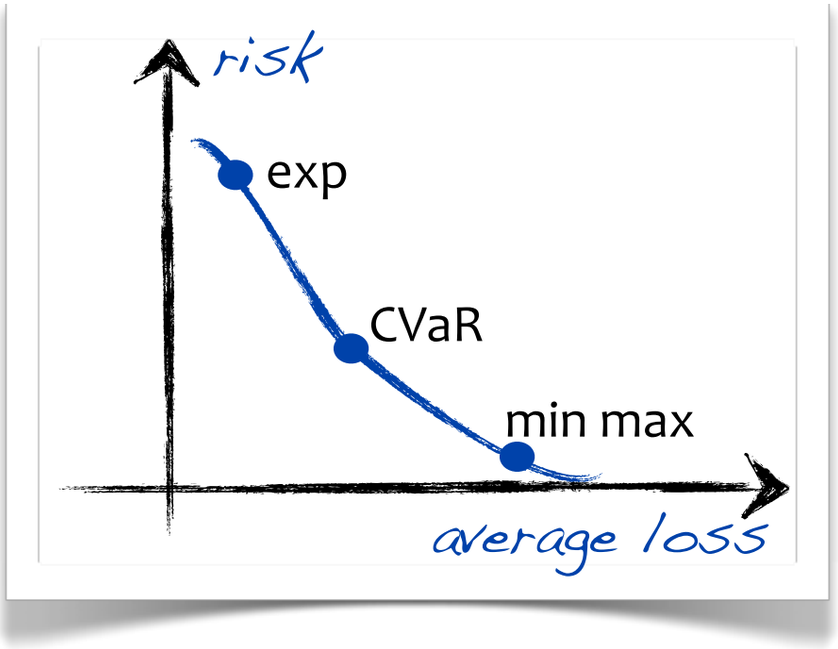
- Discrete distributions can be estimated from historical data (and adapted on-line)

Cost functions for SMPC to minimize

performance → $J(u,w) \triangleq \sum_{k=0}^{N-1} \ell(y_k - r(t+k), u_k)$

- **Expected performance**

$$\min_u E_w\left[J(u,w)\right]$$



- Tradeoff between **expected performance & risk**

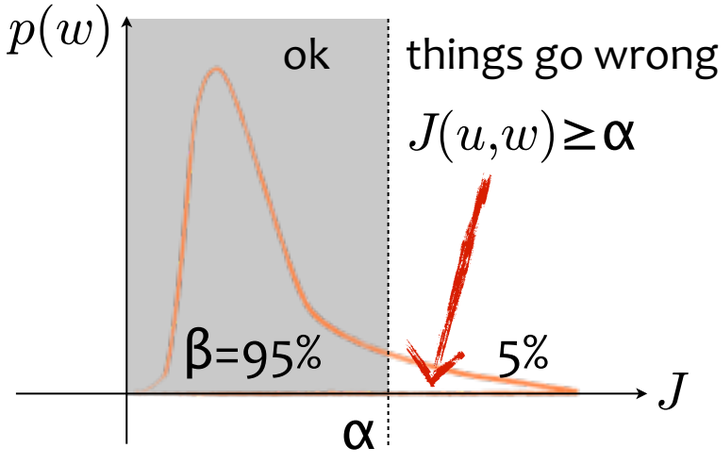
$$\min_u E_w\left[J(u,w)\right] + \rho\,\mathrm{Var}\left[J(u,w)\right]$$

- Conditional Value-at-Risk (**CVaR**)

$$\min_{u,\alpha}\left\{\alpha + \frac{1}{1-\beta}E[\max(J(u,w)-\alpha,0)]\right\}$$

(Rockafellar, Uryasev, 2000)

= minimize expected loss when things go wrong (convex if J convex !)



- **Min-max**

$$\min_u\{\max_w J(u,w)\}$$

=minimize worst case performance

Stochastic program

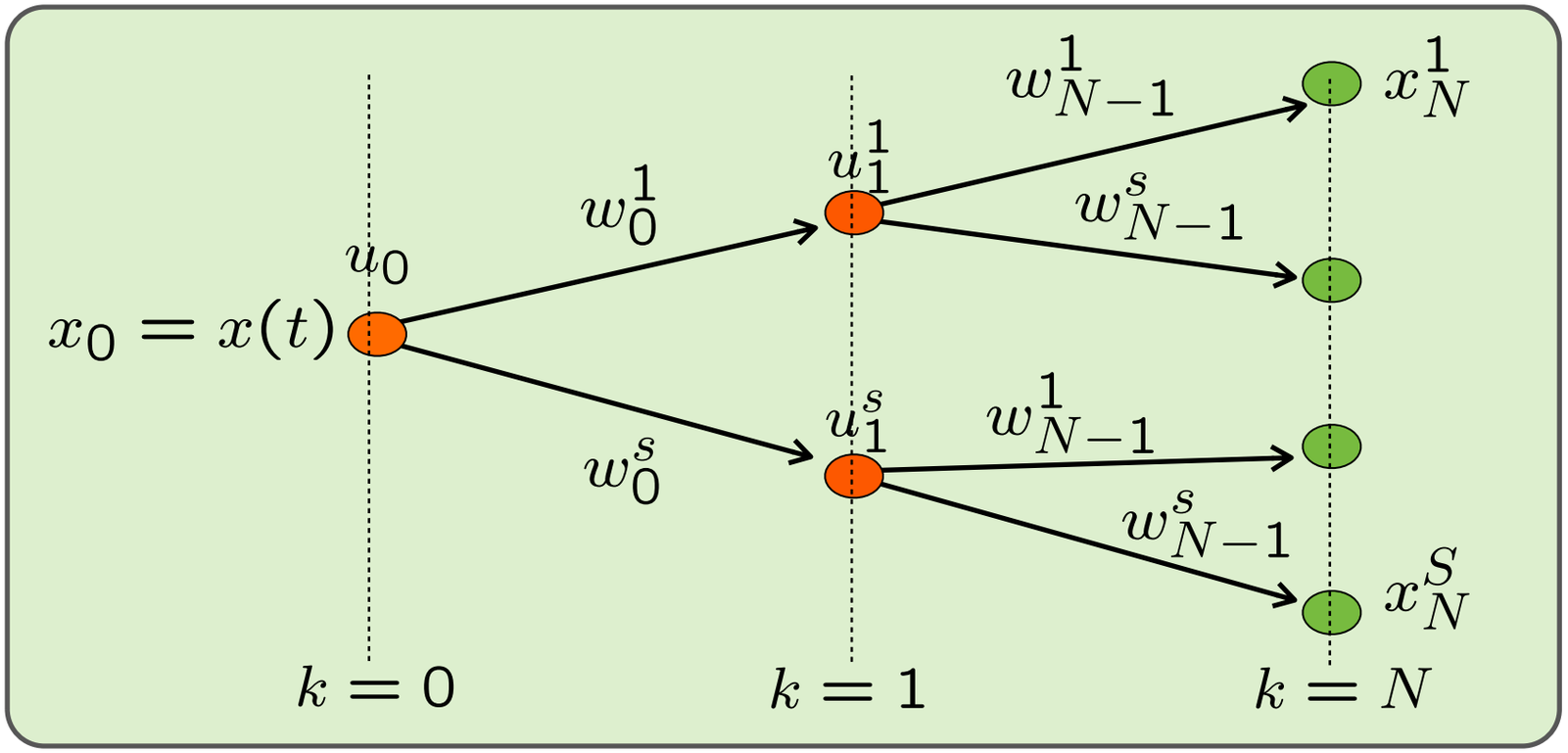
- Enumerate all possible scenarios $\{w_0^j, w_1^j, \ldots, w_{N-1}^j\},\ j = 1, \ldots, S$ $\qquad S = s^N$
- Each scenario has probability $p^j = \prod_{k=0}^{N-1} \mathbf{Pr}[w_k = w_k^j]$
- Each scenario has its own evolution $x_{k+1}^j = A(w_k^j)x_k^j + B(w_k^j)u_k^j$ (LTV system)
- Expectations become simple sums

$$\min E_w \left[x_N' P x_N + \sum_{k=0}^{N-1} x_k' Q x_k + u_k' R u_k \right]$$

$$\min \sum_{j=1}^{S} p^j \left((x_N^j)' P x_N^j + \sum_{k=0}^{N-1} (x_k^j)' Q x_k^j + (u_k^j)' R u_k^j \right)$$

This is again a quadratic function of the inputs

Scenario tree



- Scenario = path on the tree
- Number S of scenarios = number of leaf nodes

$$\min \ldots + p^j (x_k^j)' Q x_k^j + \ldots$$

$$y_{\min} \le y_k \le y_{\max}, \quad \forall w$$

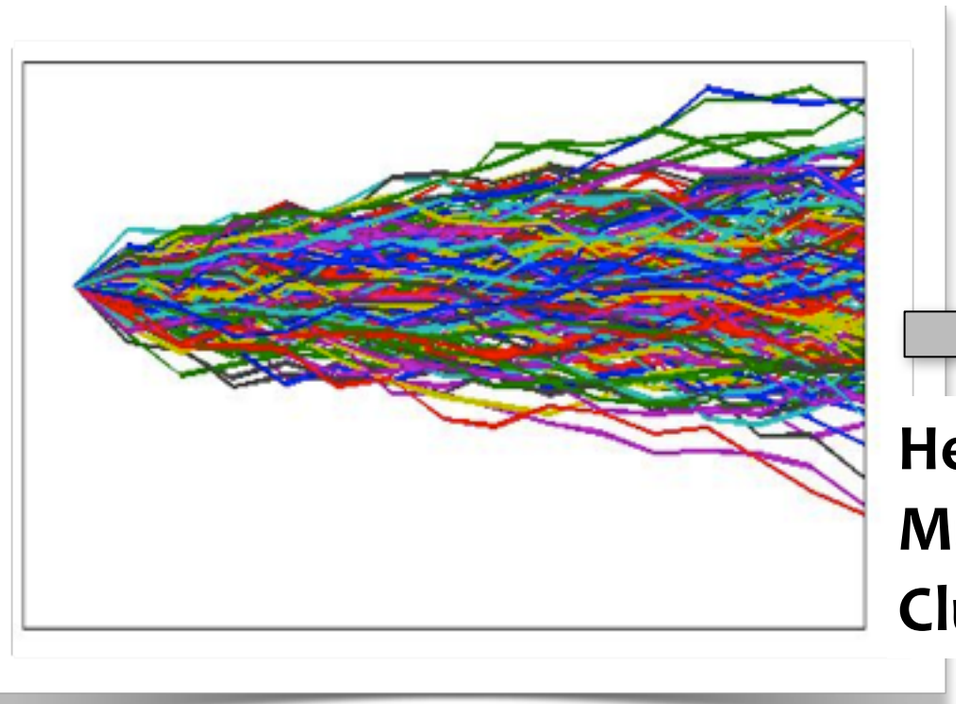
- Some paths can be removed if their probability is very small (at your own risk)
- **Causality constraint:** $u_k^j = u_k^h$ when scenarios j and h share the same node at prediction time k (for example: $u_0^j \equiv u_0^h$ at root node $k=0$)

Scenario tree generation from data

- Scenario trees can be generated by **clustering** sample paths

- Paths can be obtained by Monte Carlo simulation of (arbitrarily complex) models, or from historical data

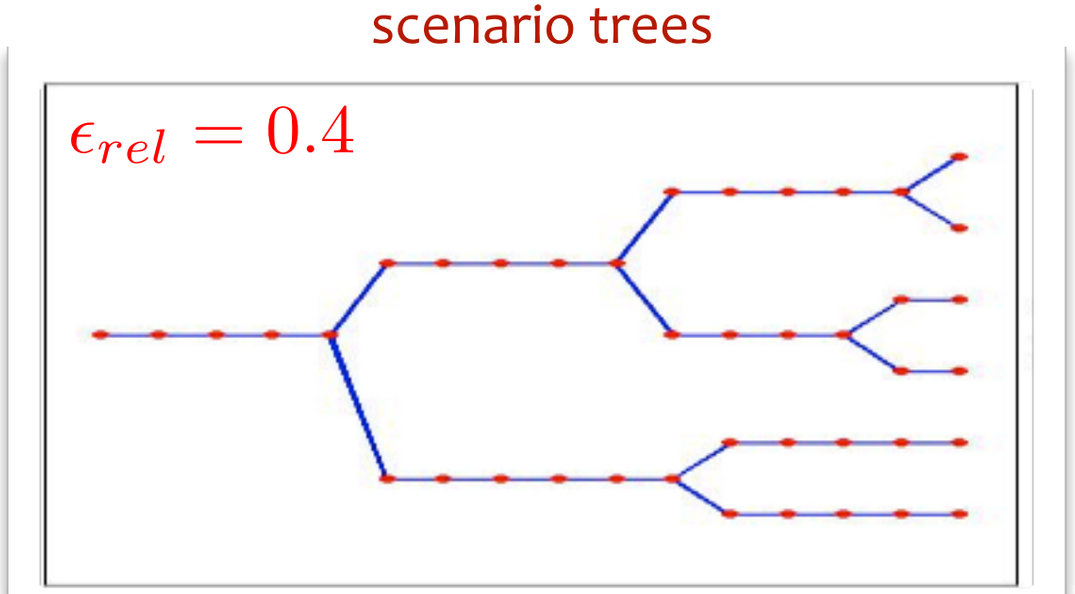
scenario "fan" (collection of sample paths)

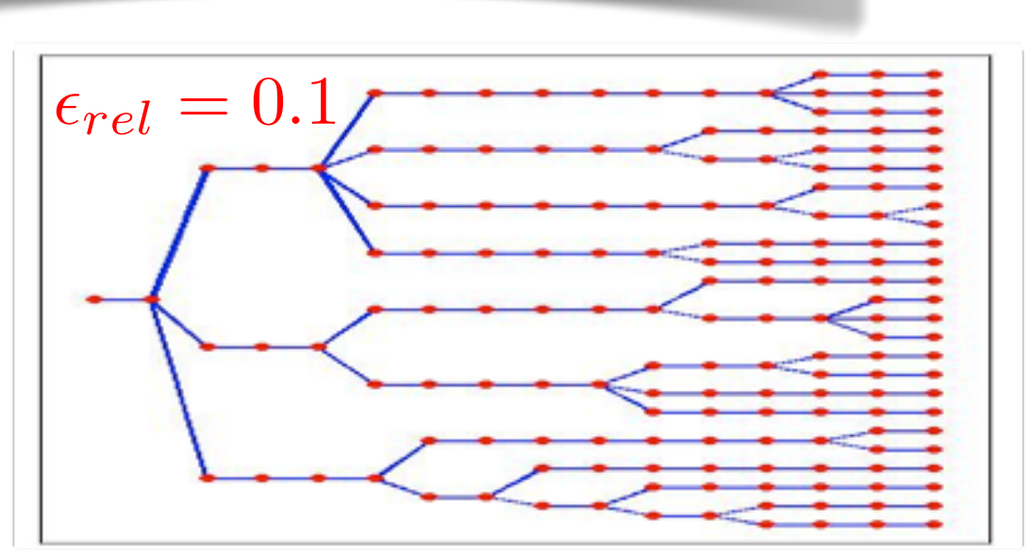


Heuristic Multilevel Clustering

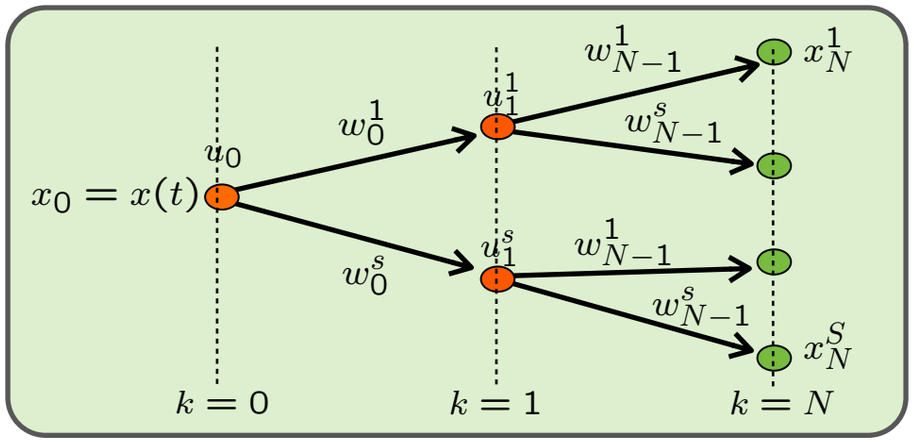
(Heitsch, Römisch, 2009)

scenario trees



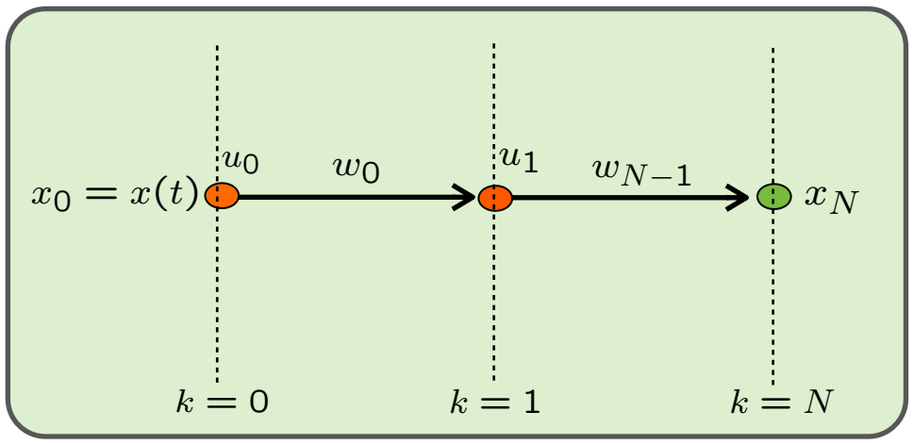


Scenario enumeration



scenario tree

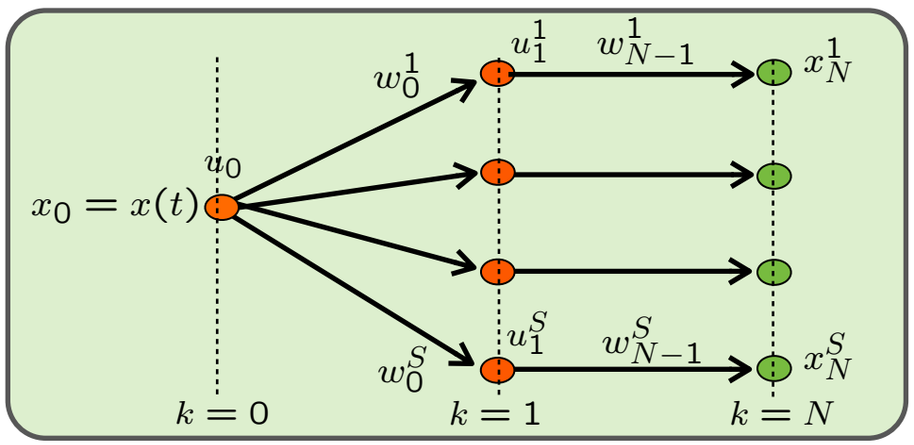
Causality is exploited: decision u_k only depends on past disturbance realizations $\{w_0, w_1, \ldots, w_{k-1}\}$



deterministic

Only a sequence of disturbances is considered

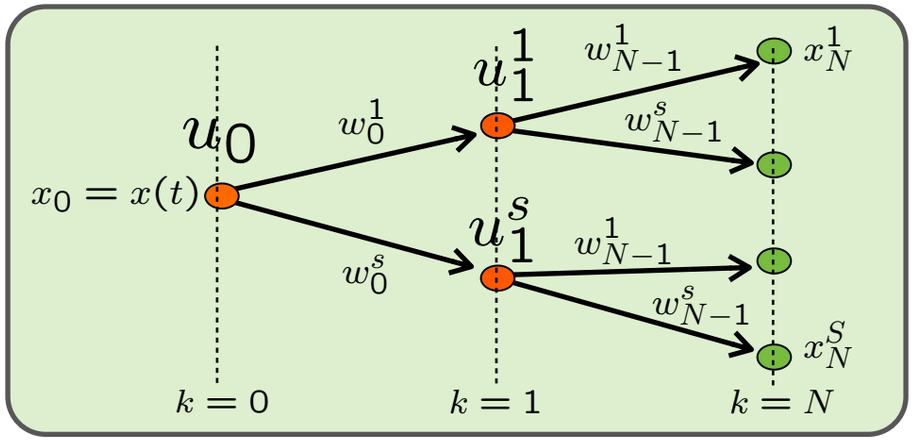
- frozen-time: $w_k \equiv w(t), \ \forall k$ (causal prediction)
- anticipative action: $w_k \equiv w(t+k)$ (non-causal)
- "expected" problem: $w_k = E[w(t+k)|t]$ (causal)



scenario "fan"

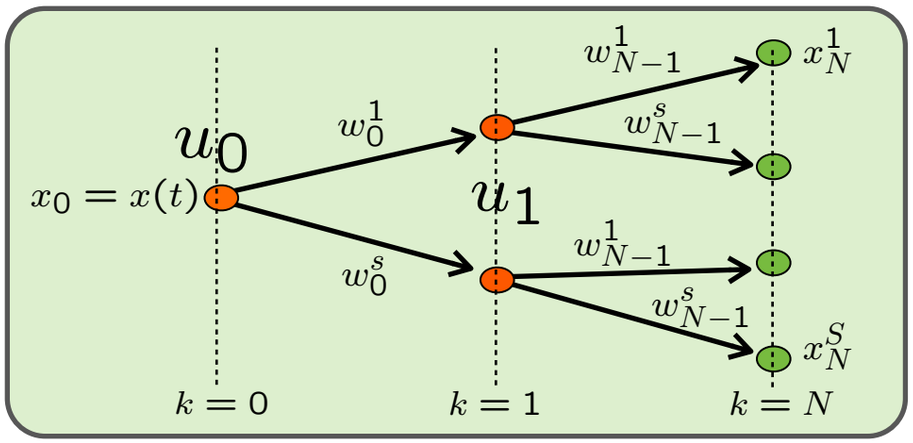
- generate a set of scenarios (Monte Carlo simulation)
- decision u_k also depends on future disturbance realizations $\{w_k, w_{k+1}, \ldots, w_{N-1}\}$

Open-loop vs. closed-loop prediction



closed-loop prediction

A proper move u is optimized to counteract each possible outcome of the disturbance w



open-loop prediction

Only a sequence of inputs $\{u_0, u_1, \ldots, u_{N-1}\}$ is optimized, the same u must be good for all possible disturbance w

- Intuitively: OL prediction is more conservative than CL in handling constraints
- OL problem = CL problem + additional constraints $u^j \equiv u, \ \forall j = 1, \ldots, S$ (=less degrees of freedom)

Scenario-based stochastic MPC

Existing literature on stochastic MPC

(Schwarme & Nikolaou, 1999) (Munoz de la Pena, Bemporad, Alamo, 2005) (Oldewurtel, Jones, Morari, 2008)
(Wendt & Wozny, 2000) (Couchman, Cannon, Kouvaritakis, 2006) (Ono, Williams, 2008)
(Batina, Stoorvogel, Weiland, 2002) (Primbs, 2007) (van Hessem & Bosgra 2002) (Bemporad, Di Cairano, 2005)
(Bernardini, Bemporad, 2012)

- Performance index $\min E_w \left[x'_N P x_N + \sum_{k=0}^{N-1} x'_k Q x_k + u'_k R u_k \right]$

- Goal: ensure **mean-square convergence** $\lim_{t\to\infty} E[x'(t)x(t)] = 0$ (for $H(w(t))=0$)

- The existence of a **stochastic Lyapunov function** $V(x) = x'Px$

$$E_{w(t)}[V(x(t+1)] - V(x(t)) \leq -x(t)'Lx(t), \quad \forall t \geq 0 \qquad L = L' > 0$$

(Morozan, 1983)

ensures mean-square stability

Linear stochastic stabilization

- Assume $w(t) \in \{w_1, \ldots, w_s\}$ and **constant** probability $p(t) \equiv p, \ \forall t$

- The **stochastic convergence** condition $E_{w(t)}[V(x(t+1)] - V(x(t)) \leq -x(t)'Lx(t)$ can be recast as the **LMI** condition

$$\begin{bmatrix} Q & Q & \sqrt{p_1}(A_1Q+B_1Y)' & \cdots & \sqrt{p_s}(A_sQ+B_sY)' \\ Q & W & 0 & \cdots & 0 \\ \sqrt{p_1}(A_1Q+B_1Y) & 0 & Q & & \\ \vdots & \vdots & & \ddots & \\ \sqrt{p_s}(A_sQ+B_sY) & 0 & & & Q \end{bmatrix} \succeq 0$$

$$Q = Q' > 0$$

$$W = W' > 0$$

(Bernardini, Bemporad, 2012)

- The Lyapunov function is $V(x) = x'Q^{-1}x$

- Mean-square stability guaranteed by linear feedback $u(k) = Kx(k), \ K = YQ^{-1}$
  $L = W^{-1}$

- A minimum decrease rate L can be imposed

Stabilizing SMPC

(Bernardini, Bemporad, 2012)

- Impose stochastic stability constraint in SMPC problem (=quadratic constraint w.r.t. u_0)

$$\begin{aligned}
\min_{u} \quad & E_w\left[\sum_{k=0}^{N-1} \ell(x_k,u_k)\right] \\
\text{s.t.} \quad & x_{k+1} = A(w_k)x_k + B(w_k)u_k \\
& E\left[V(A(w_0)x_0 + B(w_0)u_0)\right] \le x_0'(Q^{-1} - L)x_0 \\
& x_0 = x(t)
\end{aligned}$$

performance and stability are decoupled

- SMPC approach:
  1. Solve LMI problem off-line to find stochastic Lyapunov fcn $V(x) = x'Q^{-1}x$
  2. Optimize stochastic performance based on scenario tree

Theorem: The closed-loop system is as. stable in the mean-square sense

- SMPC can be generalized to handle **input and state constraints**

Note: *recursive feasibility* guaranteed by backup solution $u(k) = Kx(k)$

A few sample applications of SMPC

- **Financial engineering:** dynamic hedging of portfolios replicating synthetic options

  (Bemporad, Bellucci, Gabbriellini, 2009)
  (Bemporad, Gabbriellini, Puglia, Bellucci, 2010)
  (Bemporad,Puglia, Gabbriellini, 2011)

- **Energy systems:** power dispatch in smart grids, optimal bidding on electricity markets

  (Patrinos, Trimboli, Bemporad 2011)
  (Puglia, Bernardini, Bemporad 2011)

- **Automotive control:** energy management in HEVs, adaptive cruise control (human-machine interaction)

  (Bichi, Ripaccioli, Di Cairano, Bernardini, Bemporad, Kolmanovsky, CDC 2010)

- **Networked control:** improve robustness against communication imperfections

  (Bernardini, Donkers, Bemporad, Heemels, NECSYS 2010)

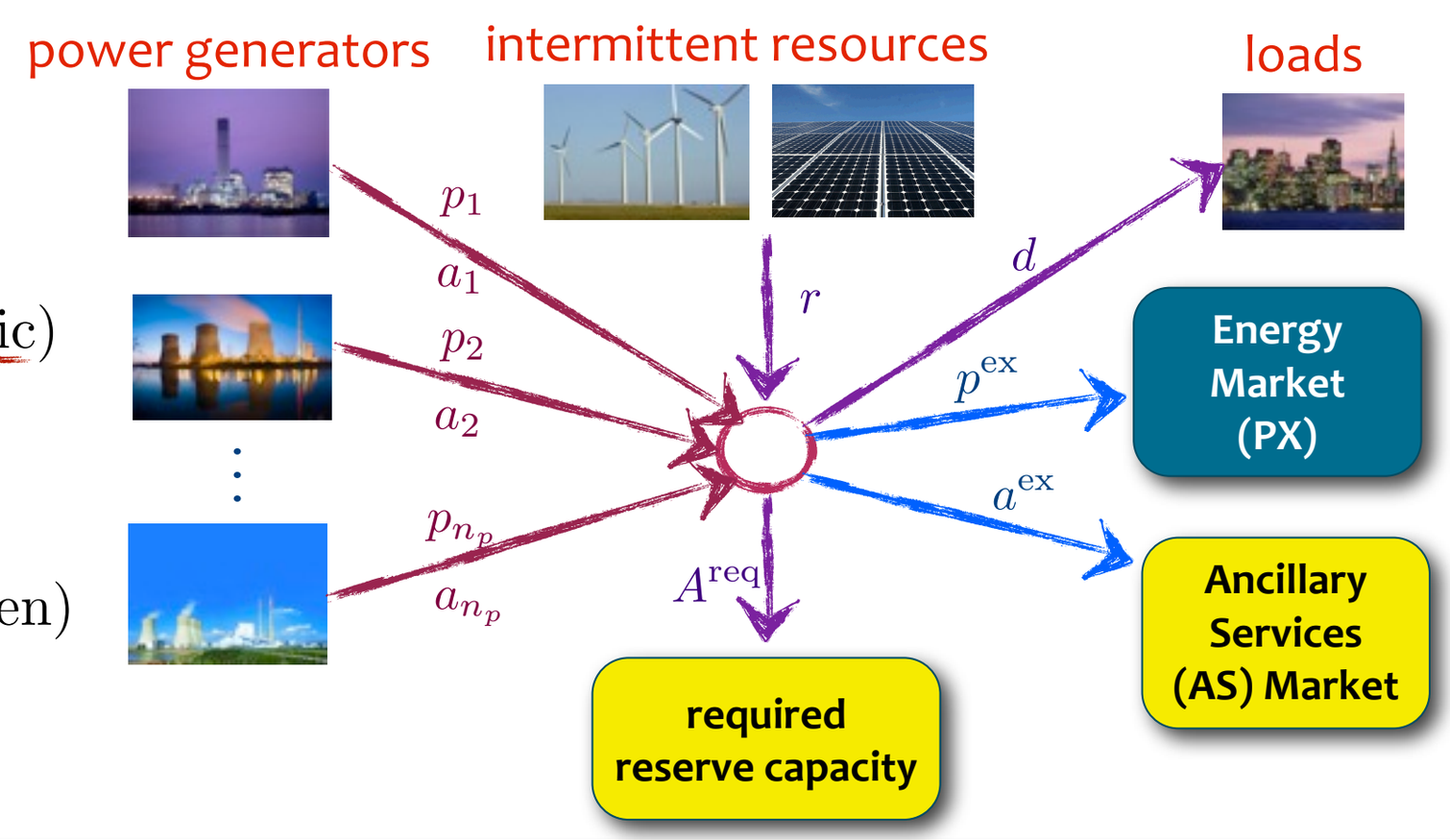
SMPC for real-time market-based power dispatch

(Patrinos, Trimboli, Bemporad, 2011)

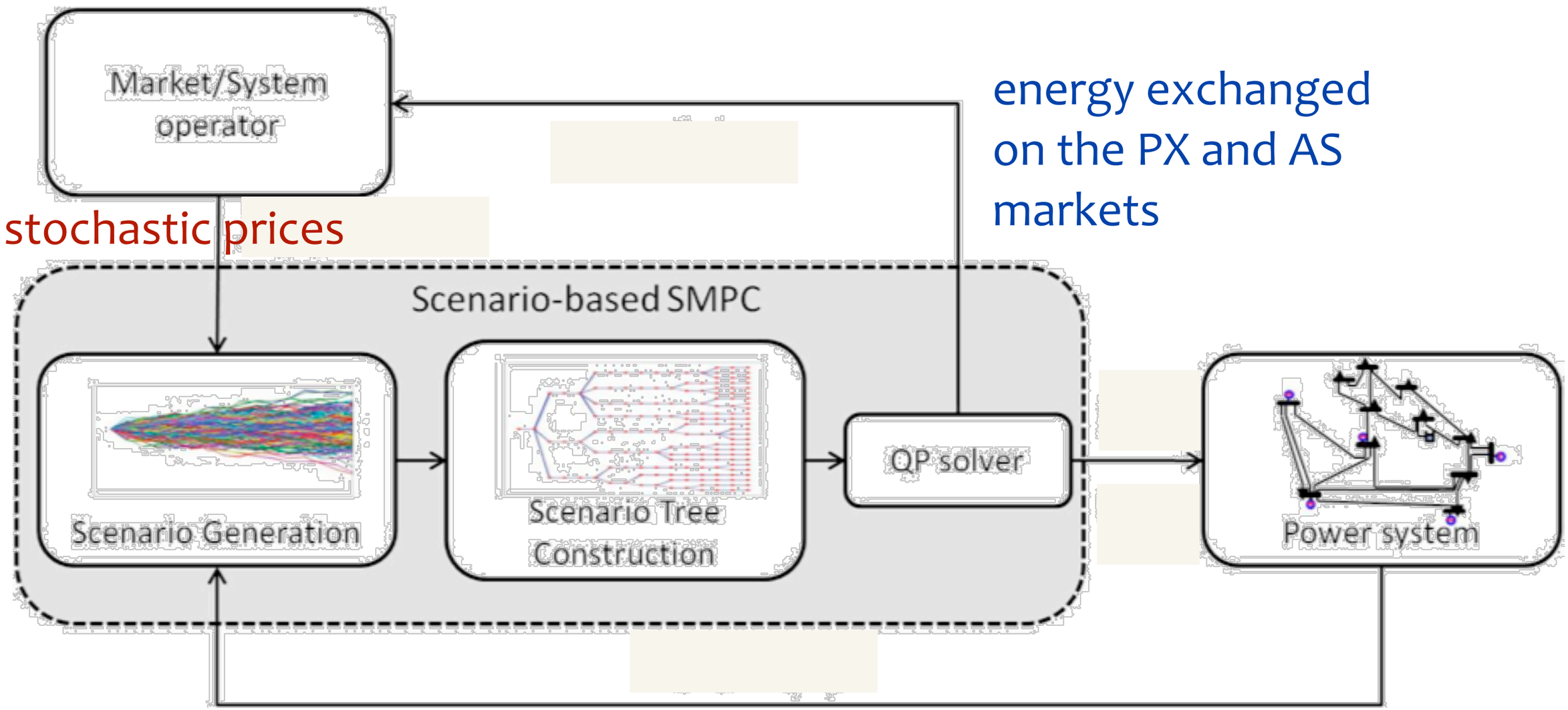
- We are a legal entity trading on the *energy (PX)* and *ancillary service (AS)* markets
- **Objective**: Minimize costs via efficient use of intermittent resources, and maximize profits by trading on electricity (PX, AS) markets
- **Constraints**: Grid capacity, rate limits, load balancing, AS balancing

p_i = power injection (decision)
a_i = reserve capacity (decision)
r = intermittent generation (stochastic)
d = demand (stochastic)
p^{ex} = power exchanged (decision)
p^{as} = power exchanged (decision)
A^{req} = required reserve for BRP (given)

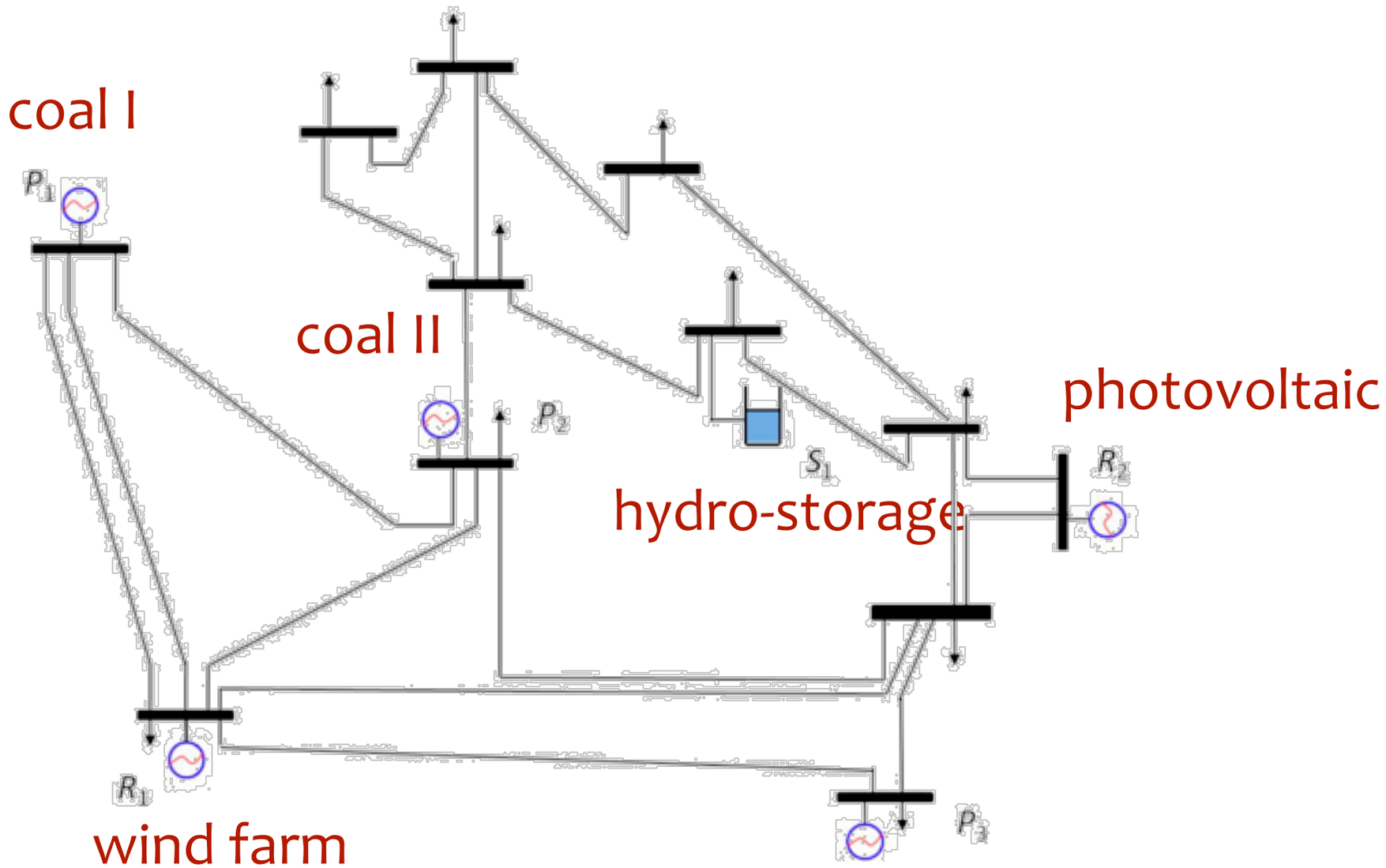
energy prices are also stochastic



SMPC architecture



SMPC for market-based optimal power dispatch



ENABLING THE FUTURE ENERGY SYSTEM

http://www.e-price-project.eu/

SEVENTH FRAMEWORK PROGRAMME

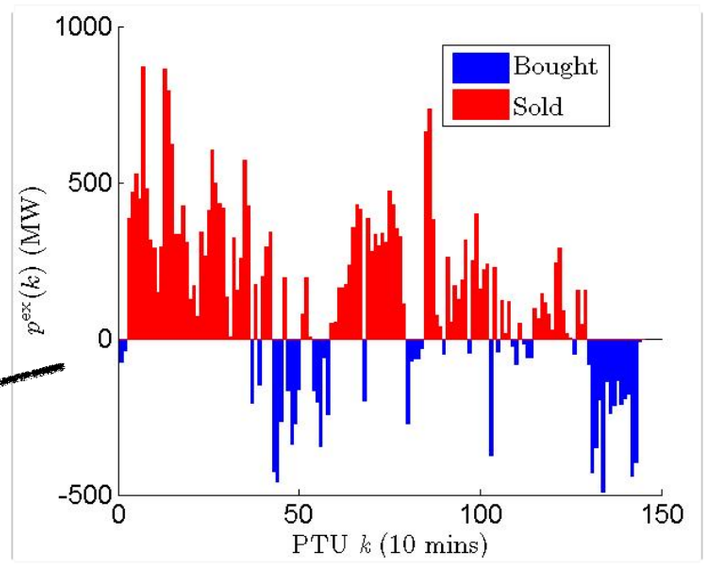
SMPC for market-based optimal power dispatch

Exact knowledge of future uncertainty

Deterministic: time-dependent expectations used for future uncertainty

Algorithm	Storage Cost	No Storage Cost	Avg # of nodes
Prescient-OC	6427979	6879741	
CE-MPC	9778750	9819518	
SSMPC ($e_{\text{rel}} = 0.1$)	7134582	7245962	350
SSMPC ($e_{\text{rel}} = 0.2$)	7144011	7249401	335
SSMPC ($e_{\text{rel}} = 0.3$)	7148494	7250207	172
SSMPC ($e_{\text{rel}} = 0.4$)	7179848	7264505	87
SSMPC ($e_{\text{rel}} = 0.5$)	7224912	7267497	50
SSMPC ($e_{\text{rel}} = 0.6$)	7239985	7277410	38
SSMPC ($e_{\text{rel}} = 0.7$)	7259491	7298023	31
SSMPC ($e_{\text{rel}} = 0.8$)	7255246	7312092	26
SSMPC ($e_{\text{rel}} = 0.9$)	7260424	7318643	22
SSMPC ($e_{\text{rel}} = 1.0$)	7260424	7318642	20

Stochastic formulation



power exchanged with grid

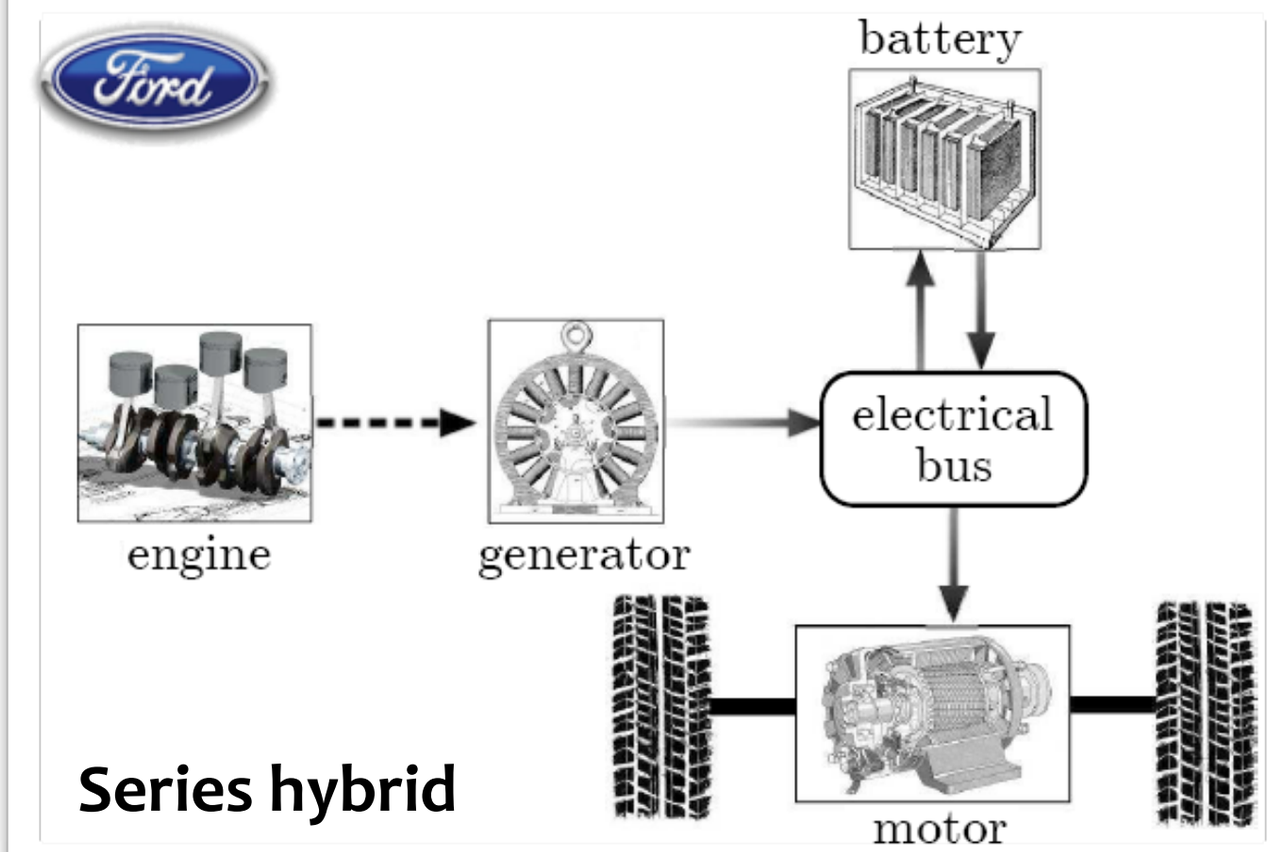
SMPC for hybrid electric vehicles (HEVs)

(Bichi, Ripaccioli, Di Cairano, Bernardini, Bemporad, Kolmanovsky, CDC'10)

Control problem:

Decide optimal generation of **mechanical power** (from engine) and **electrical power** (from battery) to satisfy **driver's power request**

What will the future power request from the driver be ?



Learning a stochastic model of the driver

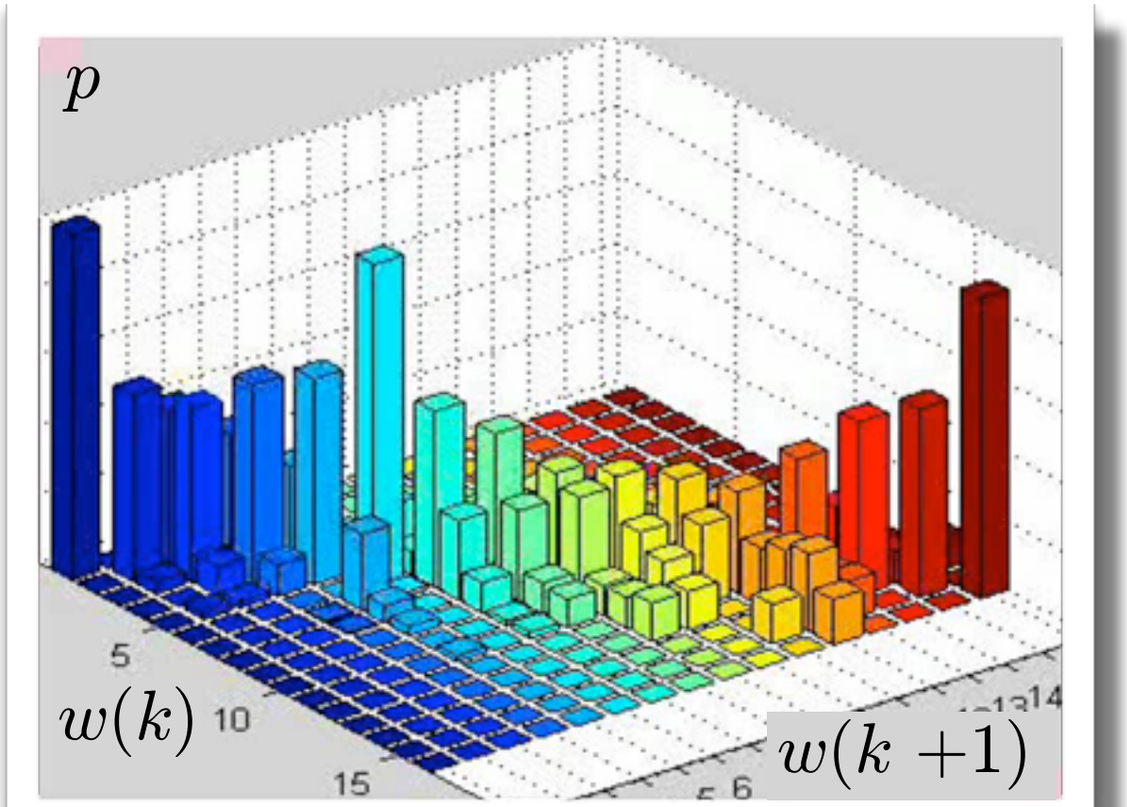
- Driver's action (power request) modeled by a **stochastic** process $w(k)$

- Good model for control purposes: $w(k)$ generated by a Markov chain

$$[T]_{ij} = \mathbf{P}[w(k+1) = w_j | w(k) = w_i]$$

Number of states in Markov chain determines the **trade-off** between complexity *and* accuracy

Transition probability matrix T is easily estimated from driver's data



Stochastic MPC results

Deterministic: future power request ≡ current one ≡ constant

Comparison on a standard driving cycle

	Fuel cons. [kg]	% Fuel improv.
FTMPC	0.281	–
SMPC (static)	0.243	13.5%
SMPC (adaptive)	0.199	29.2%
PMPC	0.197	29.9%

Stochastic formulation

Exact knowledge of future uncertainty

Decentralized and distributed MPC

Centralized vs decentralized/distributed control

Centralized control:

- Need a **global model** of the overall system (and its maintenance)
- Complexity **not scalable** with plant size
- Computation complexity may become prohibitive
- Control design hard to commission, start-up, and **maintain** (many tuning "knobs")
- High **risk**: a single controller is running the whole plant
- Good **theoretical** properties (e.g. closed-loop stability)

Centralized vs decentralized/distributed control

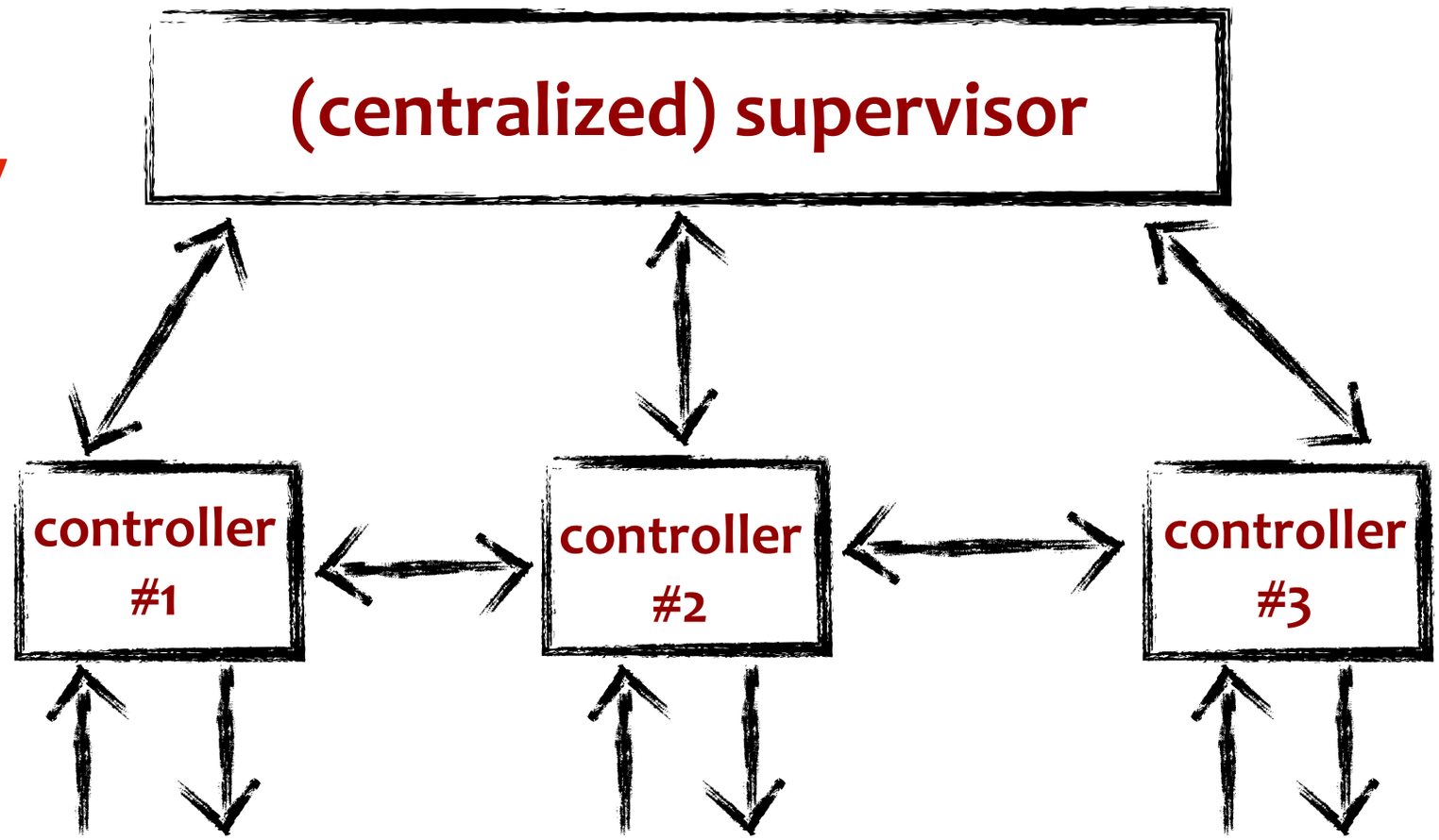
Decentralized control:

- **Local submodels** of system components are enough
- Computational tasks are **parallelized**, each task is simple
- Data gathering is simpler (**local measurements** used only locally)
- Commissioning, start-up, and **maintenance** more practical (controller updates do not require a whole plant shutdown)
- **Global properties** (stability, performance) harder to assess, especially in the presence of input/state constraints

⇒ Careful cooperation of controllers is needed to ensure global properties (such as stability and constraint fulfillment)

Typical decentralized approach

- Measure/estimate **local** states
- Compute control actions **locally**
- **Exchange** decisions with neighbors, possibly reiterate local computations
- Apply the current command input to local actuator(s)
- Possibly interact with upper level of decision making (**hierarchical control**)



large-scale process

Main issues: Global closed-loop stability ? Feasibility of global constraints ? Loss of performance w.r.t. centralized control ?

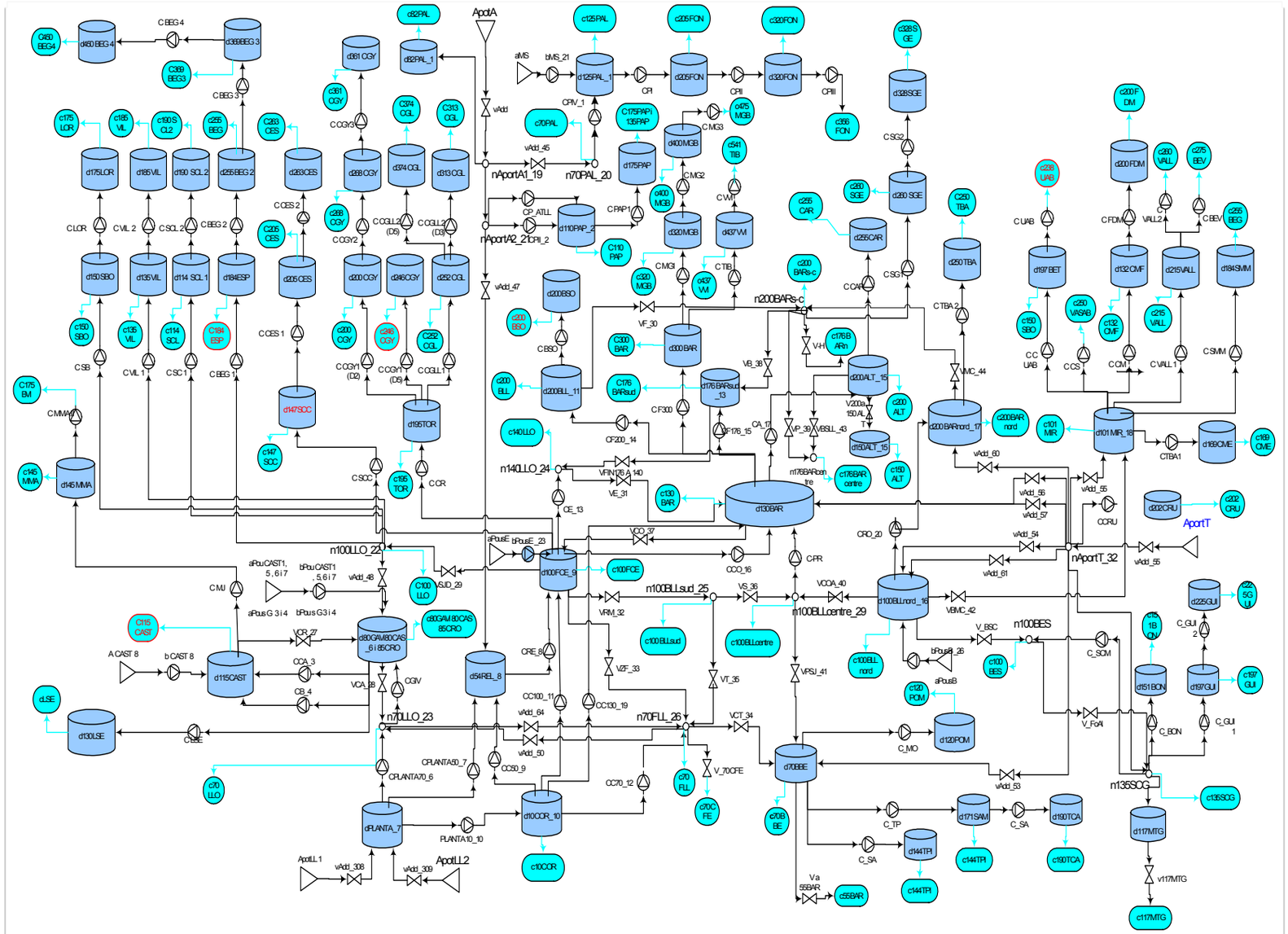
Decentralized/distributed MPC

submodels	constraints	intersampling iterations	broadcast prediction	state constraints	stability constraints	authors
coupled	local inputs	no	no	no	none	Alessio, Barcelli, Bemporad
coupled	local inputs	yes	no	no	none	Venkat, Rawlings, Wright
coupled	local inputs	yes	yes	no	none	Mercangöz, Doyle
decoupled	local inputs	no	yes	yes	compatibility	Dunbar, Murray
decoupled		no	yes	yes	none	Keviczy, Borrelli, Balas
coupled	local states	no	yes	yes	contractive	Jia, Krogh

Alternative approach: ***distribute the optimization*** problem associated with the centralized MPC formulation, instead of distributing the problem formulation

(see Mikael's lecture)

Distributed MPC of Barcelona water network



- General overview:

Municipalities supplied	23
Supply area	424 km^2
Population supplied	2.922.773
Average demand	7 m^3/s

- Network parameters:

Pipes length	4.645 km
Pressure floors	113
Sectors	218

- Facilities

Remote stations	98
Water storage tanks	81
Valves	64
Flow meters	92
Pumps / Pumping stations	180 / 84
Chlorine dosing devices	23
Chlorine analyzers	74

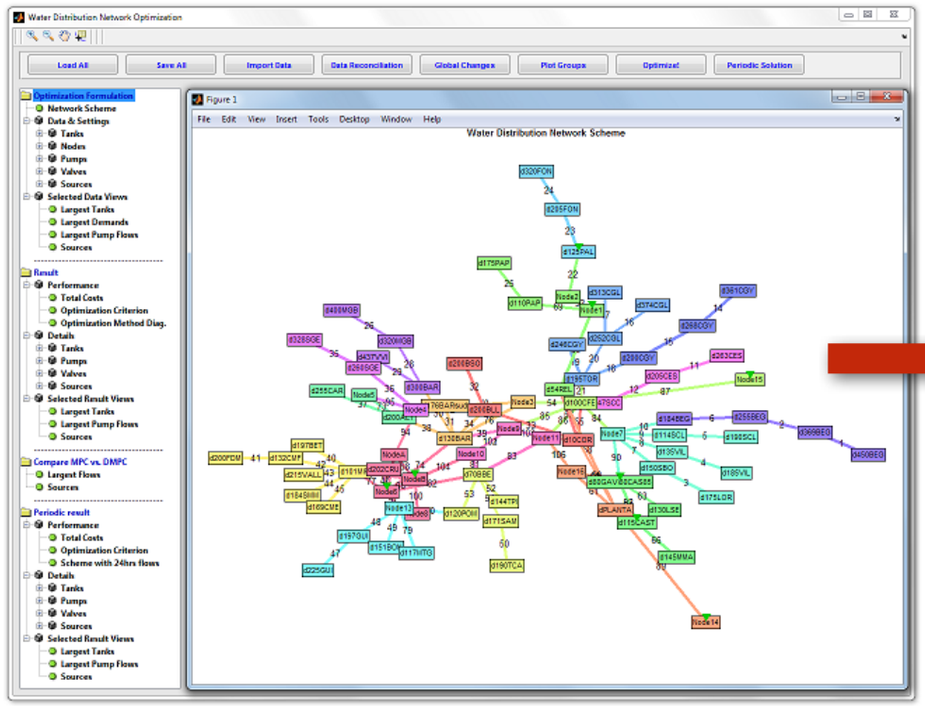
European FP7-ICT project WIDE
"DEcentralized and WIreless Control of Large-Scale Systems"

European FP7-ICT project EFFINET
"EFFIcient Integrated Real-time Monitoring and Control of Drinking Water NETworks"

DMPC of Barcelona water network

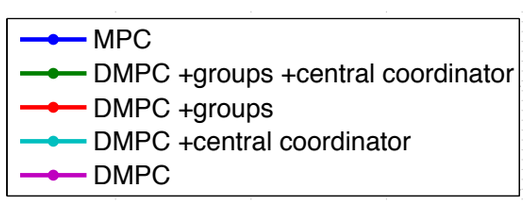
(Trnka, Pekar, Havlena, IFAC 2011)

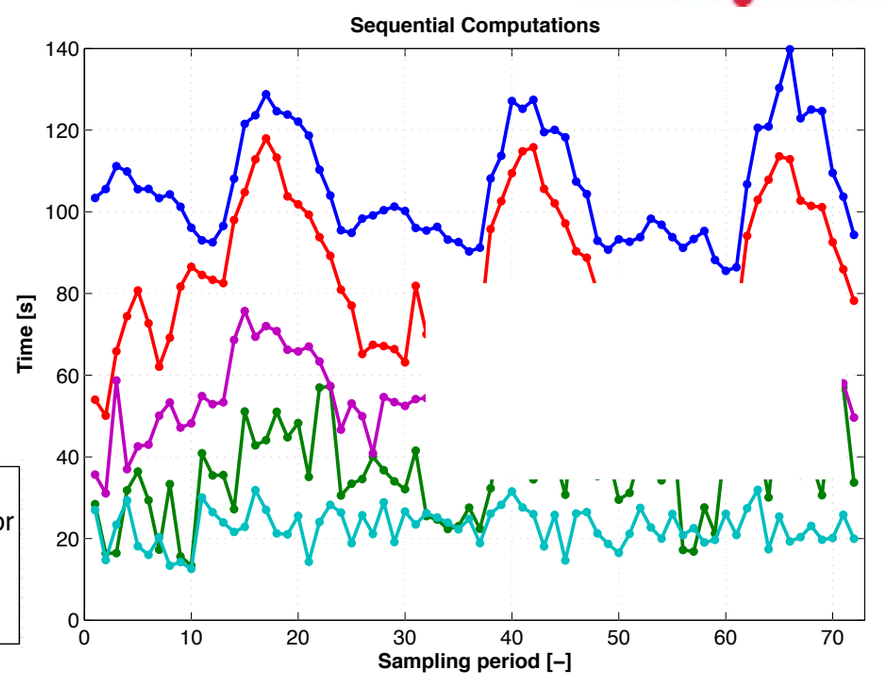
Honeywell



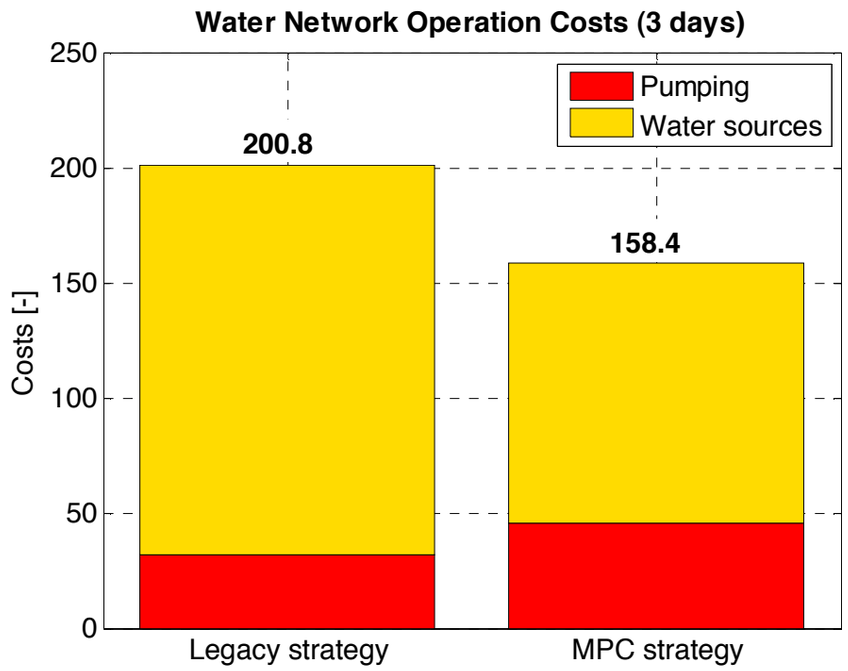
Network separation to groups

distributed MPC algorithm (optimal solution)



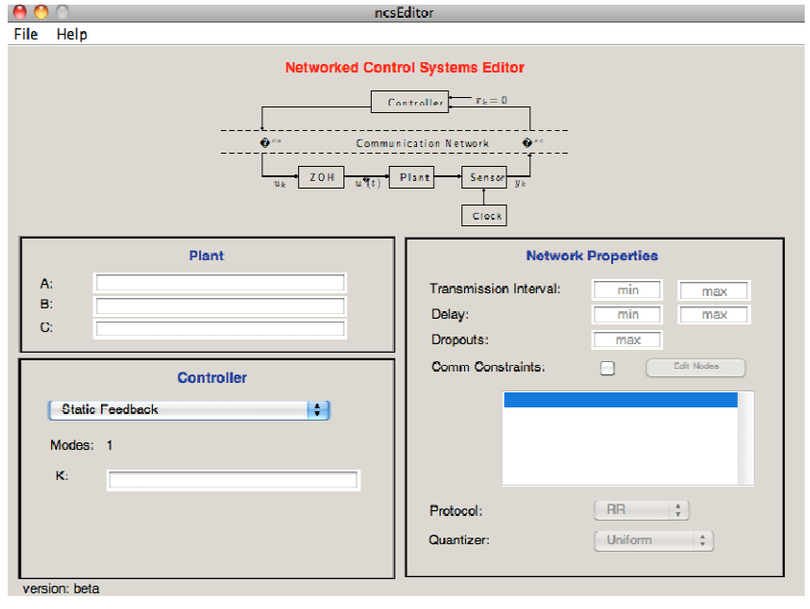


- Benefits evaluated on 3 days historic data set
- Benefits evaluation is complicated by water accumulation (different final accumulation for different control strategies)
- To get comparable data MPC was forced to fill tanks to the same final levels as in historical data (sub-optimal)
- ~20% direct cost savings (pumping and water sources)
- Indirect savings by smooth MV's operation -> leakage prevention by small pressure surges and reduced equipment tear & wear

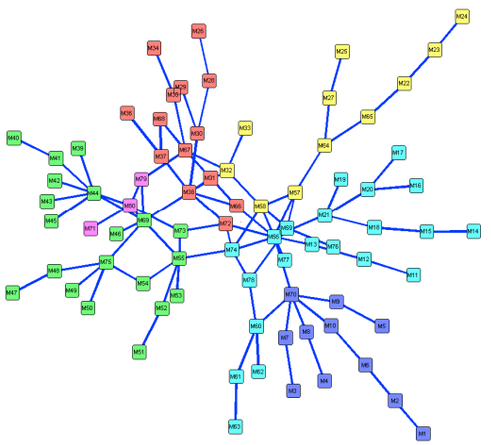


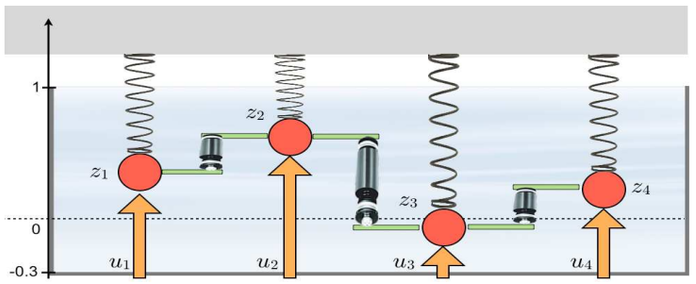
WIDE Toolbox for MATLAB

- **Networked control systems**: modeling, stability analysis, linear control synthesis
- **Model management** of large-scale systems (model reduction, create submodels, …)
- **Data acquisition** from physical WSN (Telos motes, E-Senza's nodes)



- **WSN simulation** (generate TrueTime code)
- **MPC control**: decentralized, hierarchical, network-aware

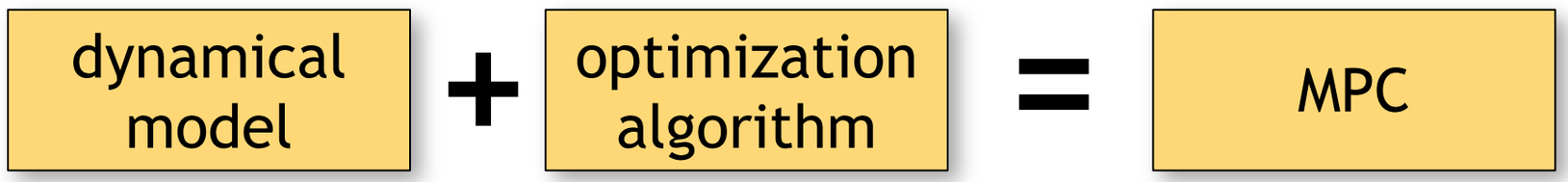




- Available for public download on September 1st 2011
  http://ist-wide.dii.unisi.it/

Conclusions and open research issues

- **MPC** is a very rich control methodology, with a variety of variants to handle different problems (large-scale, decentralized, networked, stochastic, …)



- Open research challenges include:
  - Large-scale (and huge-scale) MPC problems
  - Distributed optimization algorithms for MPC
  - Parallelization of MPC solution algorithms (such as on GPUs)
  - Low-power / low-cost / high-frequency embedded MPC (e.g., robust fixed-point algorithms for QP)
  - Applications (energy & smart grids, automotive, aerospace, finance, water networks, …)

The End

http://cse.lab.imtlucca.it/~bemporad/publications

Bibliography

Linear, explicit, and hybrid MPC

[1] D.Q. Mayne, J.B. Rawlings, C.V. Rao, and P.O.M. Scokaert. Constrained model predictive control: Stability and optimality. *Automatica*, 36(6):789–814, June 2000.

[2] A. Bemporad. Model-based predictive control design: New trends and tools. In *Proc. 45th IEEE Conf. on Decision and Control*, pages 6678–6683, San Diego, CA, 2006.

[3] A. Alessio and A. Bemporad. A survey on explicit model predictive control. In D.M. Raimondo L. Magni, F. Allgower, editor, *Nonlinear Model Predictive Control: Towards New Challenging Applications*, volume 384 of *Lecture Notes in Control and Information Sciences*, pages 345–369, Berlin Heidelberg, 2009. Springer-Verlag.

[4] A. Bemporad, M. Morari, V. Dua, and E.N. Pistikopoulos. The explicit linear quadratic regulator for constrained systems. *Automatica*, 38(1):3–20, 2002.

[5] A. Bemporad and M. Morari. Control of systems integrating logic, dynamics, and constraints. *Automatica*, 35(3):407–427, 1999.

[6] F.D. Torrisi and A. Bemporad, "HYSDEL — A tool for generating computational hybrid models," IEEE Trans. Contr. Systems Technology, vol. 12, no. 2, pp. 235–249, Mar. 2004.

[7] F. Borrelli, M. Baotić, A. Bemporad, and M. Morari. Dynamic programming for constrained optimal control of discrete-time linear hybrid systems. *Automatica*, 41(10):1709–1721, October 2005.

[8] M. Lazar, W.P.M.H. Heemels, S. Weiland, and A. Bemporad, "Stabilizing model predictive control of hybrid systems," IEEE Trans. Automatic Control, vol. 51, no. 11, pp. 1813–1818, 2006.

[9] D. Bernardini and A. Bemporad, "Stabilizing model predictive control of stochastic constrained linear systems," IEEE Trans. Automatic Control, vol. 57, no. 6, pp. 1468–1480, 2012.

[10] D. Bernardini and A. Bemporad, "Energy-aware robust model predictive control based on noisy wireless sensors," Automatica, vol. 48, pp. 36–44, 2012.

[11] A. Bemporad, A. Oliveri, T. Poggi, and M. Storace, "Ultra-fast stabilizing model predictive control via canonical piecewise affine approximations," IEEE Trans. Automatic Control, vol. 56, no. 12, pp. 2883–2897, 2011.

[12] P. Patrinos and A. Bemporad, "An accelerated dual gradient-projection algorithm for embedded linear model predictive control," IEEE Trans. Automatic Control, accepted for publication

[13] F. Borrelli, A. Bemporad, and M. Morari, *Predictive control for linear and hybrid systems*, Cambridge University Press, 2011, In press.

Bibliography

Decentralized & distributed MPC

[14] T. Keviczky, F. Borrelli, and G.J. Balas. Decentralized receding horizon control for large scale dynamically decoupled systems. *Automatica*, 42(12):2105–2115, 2006.

[15] W. Li and C.G. Cassandras. Stability properties of a receding horizon contoller for cooperating UAVs. In *Proc. 43th IEEE Conf. on Decision and Control*, pages 2905–2910, Paradise Island, Bahamas, 2004.

[16] E. Camponogara, D. Jia, B.H. Krogh, and S. Talukdar. Distributed model predictive control. *IEEE Control Systems Magazine*, pages 44–52, February 2002.

[17] A.N. Venkat, I.A. Hiskens, J.B. Rawlings, and S.J. Wright. Distributed MPC strategies with application to power system automatic generation control. *IEEE Transactions on Control Systems Technology*, 16(6):1192–1206, 2008.

[18] M. Mercangöz and F.J. Doyle III. Distributed model predictive control of an experimental four-tank system. *Journal of Process Control*, 17(3):297–308, 2007.

[19] L. Magni and R. Scattolini. Stabilizing decentralized model predictive control of nonlinear systems. *Automatica*, 42(7):1231–1236, 2006.

[20] A. Alessio, D. Barcelli, and A. Bemporad. Decentralized model predictive control of dynamically-coupled linear systems. J. Process Control, vol. 21, no. 5, pp. 705–714, June 2011.

[21] W.B. Dunbar and R.M. Murray. Distributed receding horizon control with application to multi-vehicle formation stabilization. *Automatica*, 42(4):549–558, 2006.

[22] P. Trnka, J. Pekar, V. Havlena, "Application of Distributed MPC to Barcelona Water Distribution Network", IFAC 2011.

[23] A. Richards and J.P. How. Decentralized model predictive control of cooperating UAVs. In *Proc. 43th IEEE Conf. on Decision and Control*, Paradise Island, Bahamas, 2004.

[24] A. Bemporad, D. Barcelli, "Decentralized model predictive control," in *Networked Control Systems*, A. Bemporad, W.P.M.H. Heemels, and M. Johansson, Eds., Lecture Notes in Control and Information Sciences, pp. 149–178. Springer-Verlag, Berlin Heidelberg, 2010.

[25] D. Barcelli, D. Bernardini, and A. Bemporad, "Synthesis of networked switching linear decentralized controllers," in Proc. 49th IEEE Conf. on Decision and Control, Atlanta, GA, USA, 2010, pp. 2480–2485

[26] R. Scattolini. Architectures for distributed and hierarchical model predictive control – a review. *Journal of Process Control*, 19:723–731, 2009.

Bibliography

Selected MPC applications

[27] A. Bemporad and C. Rocchi, "Decentralized hybrid model predictive control of a formation of unmanned aerial vehicles," in Proc. 18th IFAC World Congress, Milano, Italy, 2011

[28] A. Bemporad and C. Rocchi, "Decentralized linear time-varying model predictive control of a formation of unmanned aerial vehicles," in Proc. 50th IEEE Conf. on Decision and Control and European Control Conf., Orlando, FL, 2011, pp. 7488–7493.

[29] D. Bernardini, T. Donkers, A. Bemporad, and W.P.M.H. Heemels, "A model predictive control approach for stochastic networked control systems," in 2nd IFAC Workshop on Estimation and Control of Networked Systems (NecSys'10), Annecy, France, 2010

[30] A. Bemporad, L. Bellucci, and T. Gabbriellini, "Dynamic option hedging via stochastic model predictive control based on scenario simulation," Quantitative Finance, 2012, In press. DOI:10.1080/14697688.2011.649780.

[31] A. Bemporad, L. Puglia, and T. Gabbriellini, "A stochastic model predictive control approach to dynamic option hedging with transaction costs," in Proc. American Contr. Conf., San Francisco, CA, USA, 2011

[32] L. Puglia, P. Patrinos, D. Bernardini, and A. Bemporad, "Reliability and efficiency for market parties in power systems," in 10th International Conference on the European Energy Market (EEM13), Stockholm, Sweden, 2013.

[33] L. Puglia, A. Virag, A. Jokic, and A. Bemporad, "Double-sided ancillary services markets: Design and optimal bidding strategies," in 10th International Conference on the European Energy Market (EEM13), Stockholm, Sweden, 2013.

[34] L. Puglia, D. Bernardini, and A. Bemporad, "A multi-stage stochastic optimization approach to optimal bidding on energy markets," in Proc. 50th IEEE Conf. on Decision and Control and European Control Conf., Orlando, FL, 2011, pp. 1509–1514.

[35] P. Patrinos, D. Bernardini, A. Maffei, A. Jokic, and A. Bemporad, "Two-time-scale MPC for economically optimal real-time operation of balance responsible parties," in IFAC 8th Power Plant and Power Systems Control Symposium, Tolouse, France, 2012, pp. 741–746.

[36] P. Patrinos, S. Trimboli, and A. Bemporad, "Stochastic MPC for real-time market-based optimal power dispatch," in Proc. 50th IEEE Conf. on Decision and Control and European Control Conf., Orlando, FL, 2011, pp. 7111–7116.

[37] S. Di Cairano, D. Yanakiev, A. Bemporad, I.V. Kolmanovsky, and D. Hrovat, "Model predictive idle speed control: Design, analysis, and experimental evaluation," IEEE Trans. Contr. Systems Technology, vol. 20, no. 1, pp. 84–97, 2012.

Publications available for download at **http://cse.lab.imtlucca.it/~bemporad/publications**