COMPUTING THE LINEAR MPC CONTROL LAW: QP SOLVERS AND EXPLICIT MPC

Alberto Bemporad - “Model Predictive Control” course - Academic year 2016/17

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

$x_0 = x(t)$

$$V(x_0) = \tfrac{1}{2} x'_0 Y x_0 + \min_z \ \tfrac{1}{2} z' H z + x'_0 F' z$$

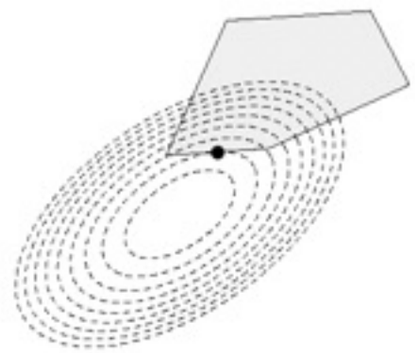
s.t. $Gz \le W + Sx_0$

(quadratic)

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

(linear)

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



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

MPC BASED ON CONVEX PIECEWISE AFFINE COSTS

(Propoi, 1963)
(Bemporad, Borrelli, Morari, 2003)

linear model

$$\begin{cases} x_{k+1} = Ax_k + Bu_k \\ y_k = Cx_k \\ x_0 = x(t) \end{cases}$$

performance index

$$\min_z \|Px_N\|_\infty + \sum_{k=0}^{N-1} \|Qx_k\|_\infty + \|Ru_k\|_\infty$$

$$\begin{aligned} \min_z \quad & \begin{bmatrix} 1 & \dots & 1 & 0 & \dots & 0 \end{bmatrix}' z \\ \text{subj. to} \quad & Gz \le W + Sx(t), \end{aligned}$$

$$z = \begin{bmatrix} \epsilon_0^u \\ \vdots \\ \epsilon_{N-1}^u \\ \epsilon_1^x \\ \vdots \\ \epsilon_N^x \\ \hline u_0 \\ u_1 \\ \vdots \\ u_{N-1} \end{bmatrix}$$

constraints

$$\epsilon_k^x \ge \|Qx_k\|_\infty, \quad \epsilon_k^u \ge \|Ru_k\|_\infty, \quad \epsilon_N^x \ge \|Px_N\|_\infty$$

$u_{\min} \le u_k \le u_{\max}$ (more generally: $u_k \in \mathcal{U}$, $\mathcal{U}$=polyhedron)

$y_{\min} \le y_k \le y_{\max}$ (more generally: $y_k \in \mathcal{Y}$, $\mathcal{Y}$=polyhedron)

MPC implemented by solving a **Linear Program (LP)**

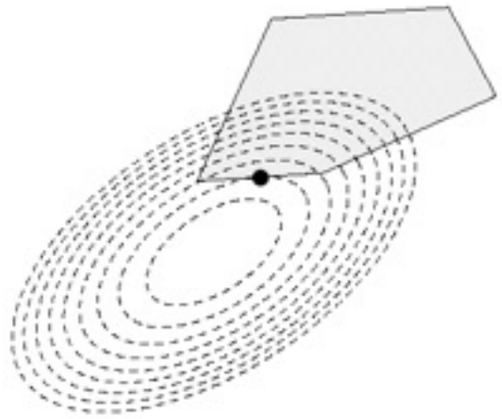
(holds for **any** sum of convex piecewise affine costs) (Schechter, 1987)

EMBEDDED LINEAR MPC AND QUADRATIC PROGRAMMING

- Linear MPC requires solving a **Quadratic Program (QP)**

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

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



ON MINIMIZING A CONVEX FUNCTION SUBJECT TO LINEAR INEQUALITIES

By E. M. L. BEALE

Admiralty Research Laboratory, Teddington, Middlesex

SUMMARY

THE minimization of a convex function of variables subject to linear inequalities is discussed briefly in general terms. Dantzig's Simplex Method is extended to yield finite algorithms for minimizing either a convex quadratic function or the sum of the t largest of a set of linear functions, and the solution of a generalization of the latter problem is indicated. In the last two sections a form of linear programming with random variables as coefficients is described, and shown to involve the minimization of a convex function.

(Beale, 1955)

A rich set of good QP algorithms is available today

Still a lot of research is going on to address **real-time requirements** …

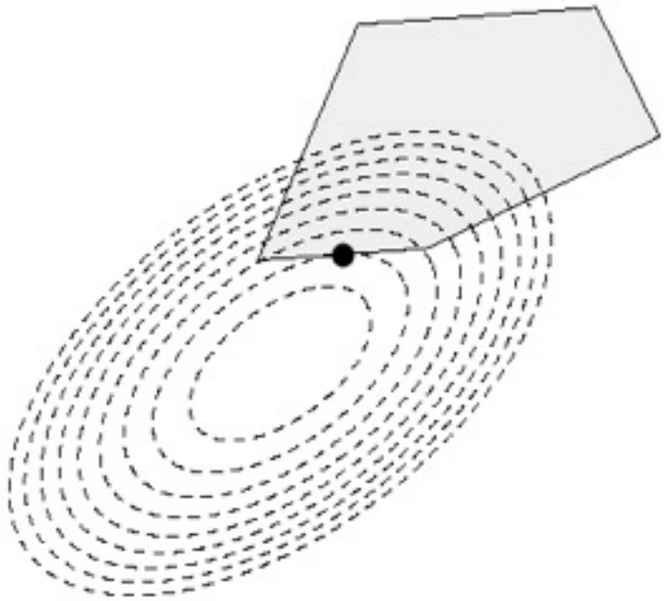
SOLUTION METHODS FOR QP

Most used algorithms for solving QP problems:

- ***active set*** methods (small/medium size)
- ***interior point*** methods (large scale)
- ***gradient projection*** methods
- ***alternating direction method of multipliers*** (**ADMM**)
- **…**

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

Quadratic Program (QP)



A useful performance comparisons of many solvers can be found at

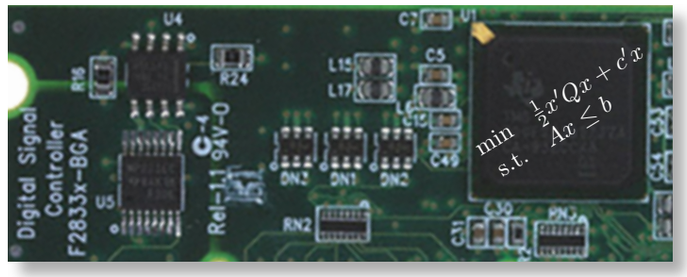
http://plato.asu.edu/guide.html

```
>>x=qpsol(Q,f,A,b,VLB,VUB,x0,solver)
```

(Hyb-Tbx)

MPC IN A PRODUCTION ENVIRONMENT

Key requirements for deployment in production:



1. **Speed (throughput)**

   a. Execution time must be less than sampling interval

   b. Also fast on average (to free the processor to execute other tasks)

2. Be able to run on **limited hardware** (e.g., 150 MHz) with **little memory**

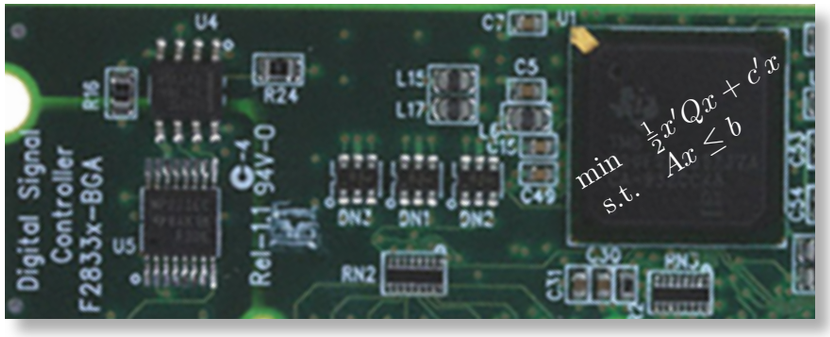
3. **Robustness** (e.g., with respect to numerical errors)

4. **Worst-case** execution time must be (tightly) **estimated**

5. **Code simple** enough to be validated/verified/certified
   (Library-free C code, easily understandable by production engineers)

MPC IN A PRODUCTION ENVIRONMENT

embedded model-based optimizer



Key requirements for production:

1. **Speed (throughput)**: solve optimization problem within sampling interval

2. **Robustness** (e.g., with respect to numerical errors)

3. Be able to run on **limited hardware** (e.g., 150 MHz) with **little memory**

4. **Worst-case execution time** must be (tightly) estimated

5. **Code simple** enough to be validated/verified/certified
   (Library-free C code. Must be understandable by production engineers)

KKT OPTIMALITY CONDITIONS

$$\begin{array}{ll} \min\limits_{z} & f(z) \\ \text{s.t.} & g_i(z) \le 0, \quad \forall i = 1,\dots,m \\ & h_j(z) = 0, \quad \forall j = 1,\dots,p \end{array}$$

Let z^* be a feasible solution, let $I = \{i : \ g_i(z^*) = 0\}$. Suppose f and g_i, h_j differentiable at z^*. Suppose $\nabla g_i(z^*)$, $\nabla h_j(z^*)$ linearly independent for $i \in I$.

Then, if z^* is optimal, there exist vectors of **Lagrange multipliers** $\lambda \in \mathbb{R}^m$, $\nu \in \mathbb{R}^p$ such that

$$\begin{aligned} \nabla f(z^*) + \sum_{i=1}^{m} \lambda_i \nabla g_i(z^*) + \sum_{j=1}^{p} \nu_j \nabla h_j(z^*) &= 0 \\ \lambda_i g_i(z^*) &= 0, \quad \forall i = 1,\dots,m \\ \lambda_i &\ge 0 \\ g_i(z^*) &\le 0 \\ h_j(z^*) &= 0, \quad \forall j = 1,\dots,p \end{aligned}$$

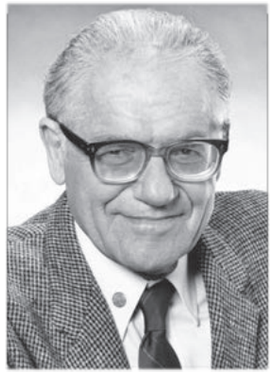
When f, g_i are convex functions and h_j are linear, the condition is also sufficient

KKT CONDITIONS FOR QP

$$\begin{array}{ll}\min_z & f(z) \triangleq \frac{1}{2} z'Hz + c'z \\ \text{s.t.} & Az \leq b\end{array}$$

$z \in \mathbb{R}^n,\ H \succeq 0 \in \mathbb{R}^{n \times n}$
$A \in \mathbb{R}^{m \times n}$

$$\begin{aligned}\nabla f(z) &= Hz + c \\ g_i(z) &= A_i' z - b_i \ (A_i' = i\text{th row of matrix } A) \\ \nabla g_i(z) &= A_i\end{aligned}$$



William Karush
(1917 - 1997)

Harold W. Kuhn
(1925 -)

Albert W. Tucker
(1905 - 1995)

$$\begin{aligned}Hz + c + A'\lambda &= 0 \\ \lambda_i(A_i' z - b_i) &= 0 \\ \lambda &\geq 0 \\ Az - b &\leq 0\end{aligned}$$

W. Karush, "Minima of functions of several variables with inequalities as side constraints", Master's thesis, Dept. of Mathematics, Univ. of Chicago, 1939

GRADIENT PROJECTION METHOD

- Optimization problem: $\min_{z \in Z} f(z)$ $\quad f : \mathbb{R}^s \to \mathbb{R}$, $Z \subseteq \mathbb{R}^s$ (Levitin & Poljak, 1965) (Goldstein, 1964)

- 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: $f(z_k) - f^* \leq \frac{L}{2k}\|z_0 - z^*\|$

GRADIENT PROJECTION METHOD FOR QP (VERY SIMPLE!)

- QP problem:

$$\begin{array}{ll}\min\limits_{z} & \frac{1}{2}z'Hz + x'F'z \\ \text{s.t.} & Gz \le W + Sx\end{array}$$

$z =$ optimization vector

$x =$ parameter vector

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

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

prepared off-line

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

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

FAST GRADIENT PROJECTION METHOD

(Nesterov, 1983)

- Optimization problem: $\min_{z \in Z} f(z)$

$$f : \mathbb{R}^s \to \mathbb{R}$$
$$Z \subseteq \mathbb{R}^s$$

- f convex and ∇f Lipschitz continuous with constant L

$$\|\nabla f(z_1) - \nabla f(z_2)\| \leq L\|z_1 - z_2\|$$

- Accelerated gradient projection iterations:

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

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

$z_{-1} = z_0$

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

FAST GRADIENT PROJECTION FOR (DUAL) QP

(Patrinos, Bemporad, IEEE TAC, 2014)

- Apply **fast gradient method** to **dual QP**:

$$\begin{array}{ll}\min\limits_{z} & \frac{1}{2}z'Hz + x'F'z \\ \text{s.t.} & Gz \le W + Sx\end{array} \quad \Rightarrow \quad \begin{array}{ll}\min\limits_{y\ge 0} & \frac{1}{2}y'My + (Dx+W)'y\end{array}$$

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

prepared off-line

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

- Iterations:

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

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

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

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

```
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
```

FAST GRADIENT PROJECTION FOR (DUAL) QP

(Patrinos, Bemporad, IEEE TAC, 2014)

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

feasibility tol

$$s_k^i \leq \frac{1}{L}\epsilon_G, \quad \forall i = 1, \ldots, m$$

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

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

dual function

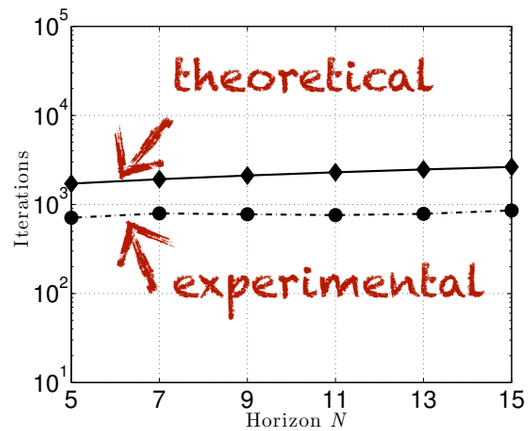
optimality tol

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

- **Convergence rate:**

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

- **Tight bounds** on maximum number of iterations



FAST GRADIENT PROJECTION FOR (DUAL) QP

(Patrinos, Bemporad, IEEE TAC, 2014)

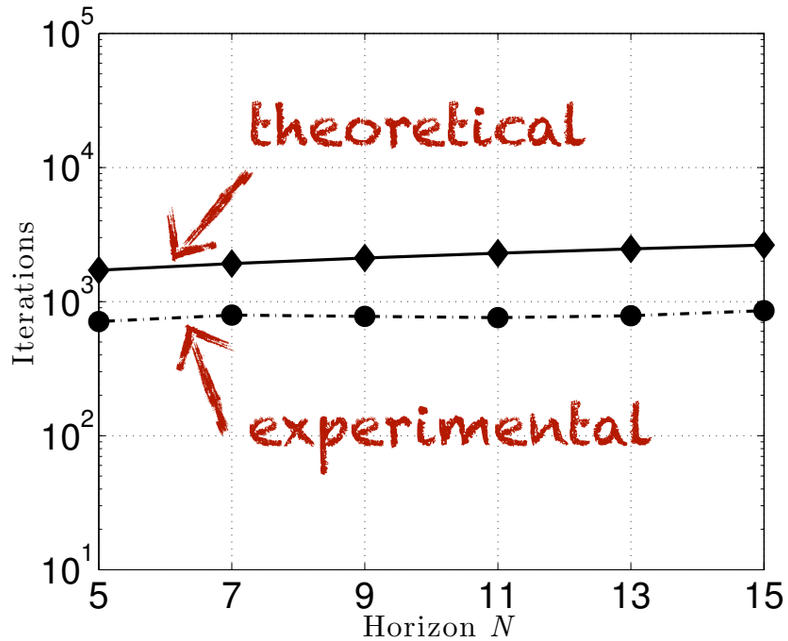
- Main on-line operations involve only **simple linear algebra**

- For MPC problems, using Riccati-like iterations complexity per iteration is $O(N)$ $[N = \text{prediction horizon}]$

- Convergence rate:

$$f(y_k) - f^* \leq \frac{2L}{(k+1)^2}\|y_0 - y^*\|^2$$

- Tight bounds on maximum number of iterations

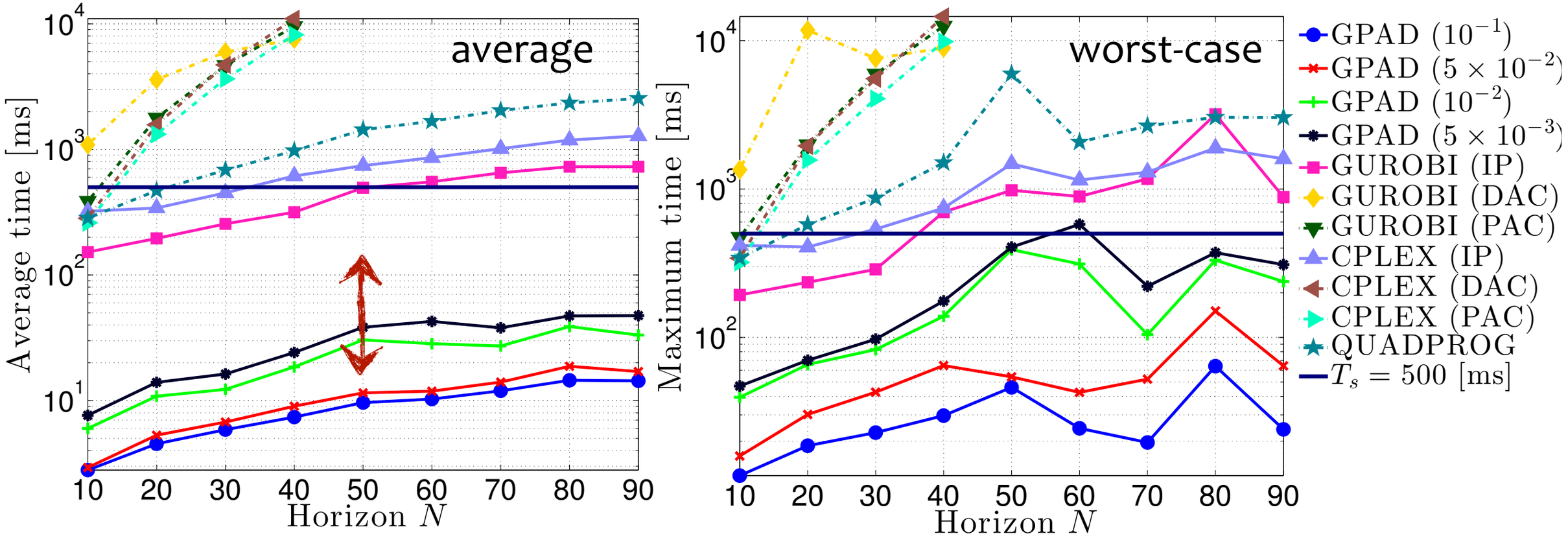


- Can be very useful to warm-start other methods !

- Currently extended to **mixed-integer** quadratic problems (Naik, Bemporad, 2016)

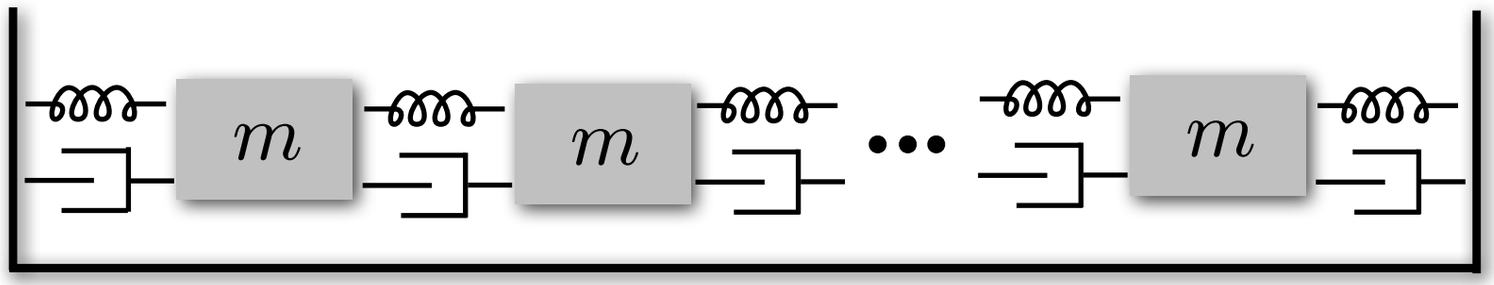
FAST GRADIENT PROJECTION FOR (DUAL) QP

- Good performance in MPC problems w.r.t. commercial solvers



Linear system:

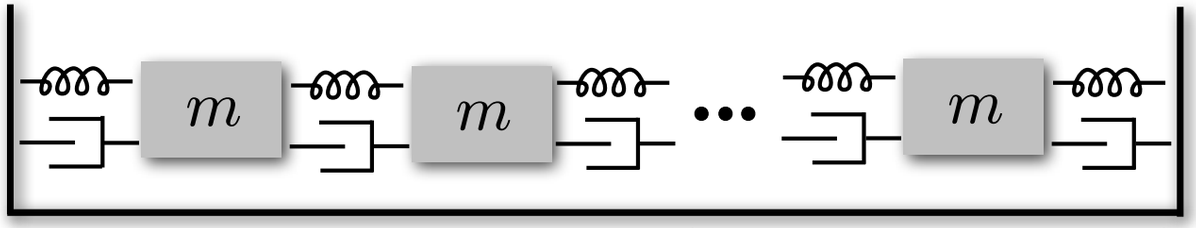
- $2M$ states, M-1 inputs
- state and input constraints
- terminal set



M=15 masses (Wang, Boyd, 2008)

- (Fast) gradient proj. can be very useful to warm-start other methods !

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)

RESTART IN FAST GRADIENT PROJECTION

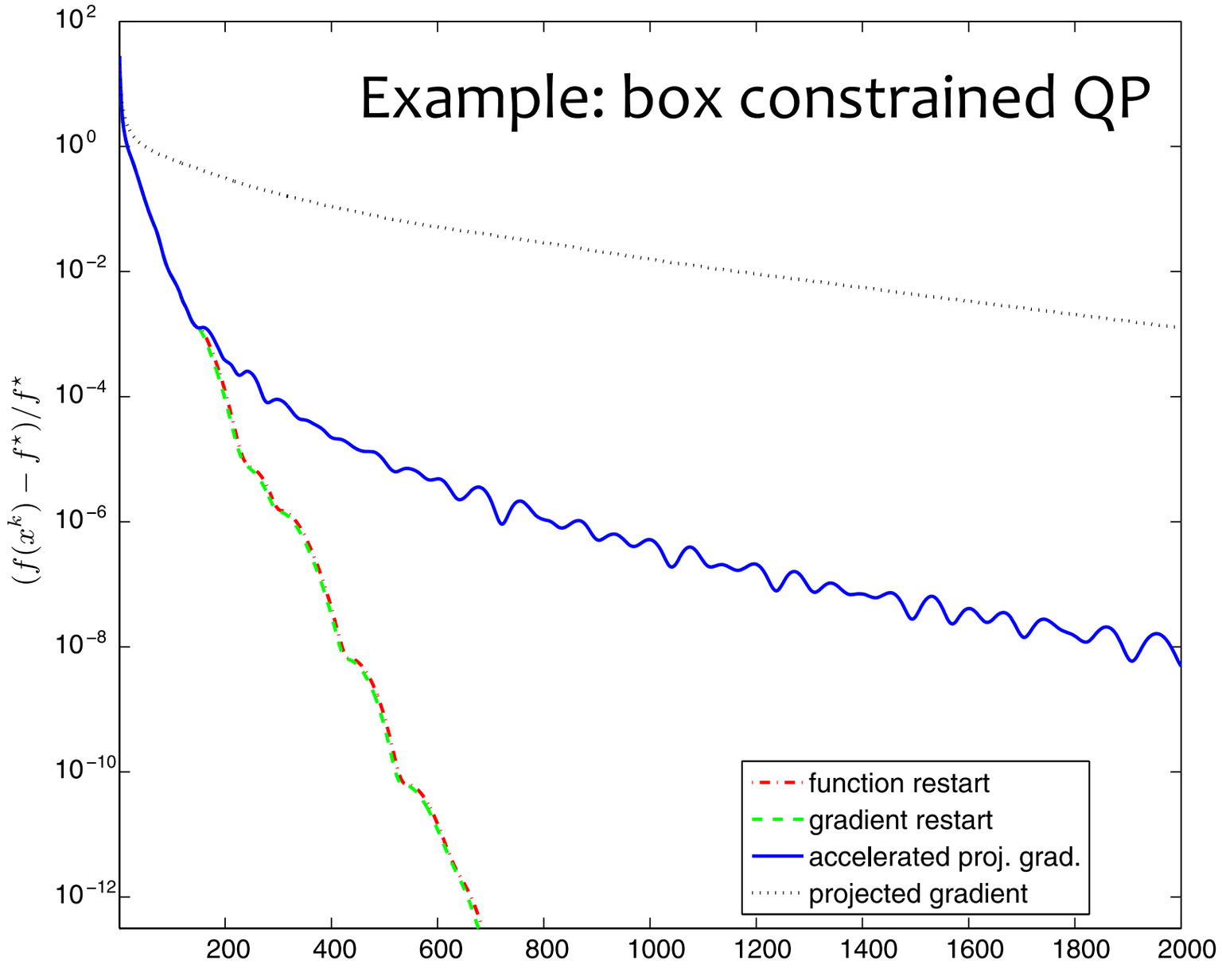
- Fast gradient projection methods can be sped up by adaptively restarting the sequence of coefficients β_k (O'Donoghue & Candès , 2013)

function restart: whenever

$$f(y_k) > f(y_{k-1})$$

gradient restart: whenever

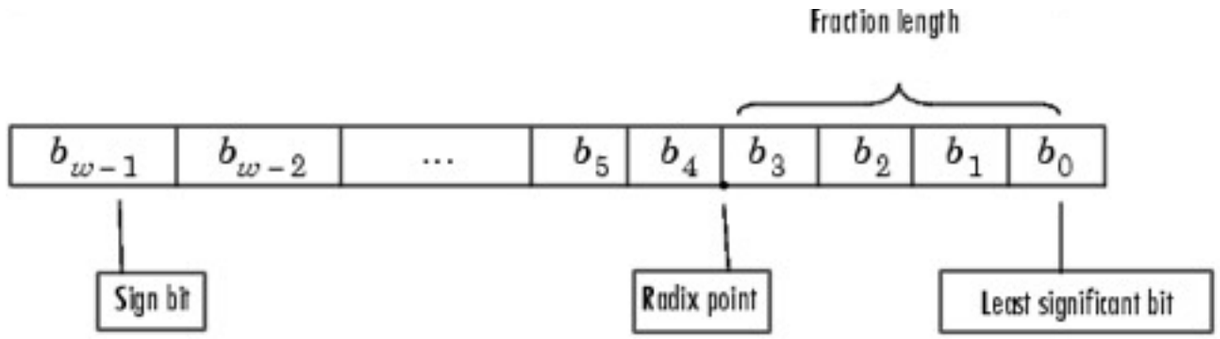
$$\nabla f(w'_{k-1})(y_k - y_{k-1}) > 0$$



SOLVING QPS IN FIXED-POINT ARITHMETICS

How about numerical robustness ?

- **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)

GRADIENT PROJECTION IN FIXED-POINT ARITHMETICS

(Patrinos, Guiggiani, Bemporad, 2013)

- Convergence to feasibility

$$\max_i g_i(z_k) \le \frac{2LD^2}{k+1} + L_v\epsilon_z^2 + 4D\epsilon_\xi$$

→ exponentially decreasing with p

- Convergence to optimality

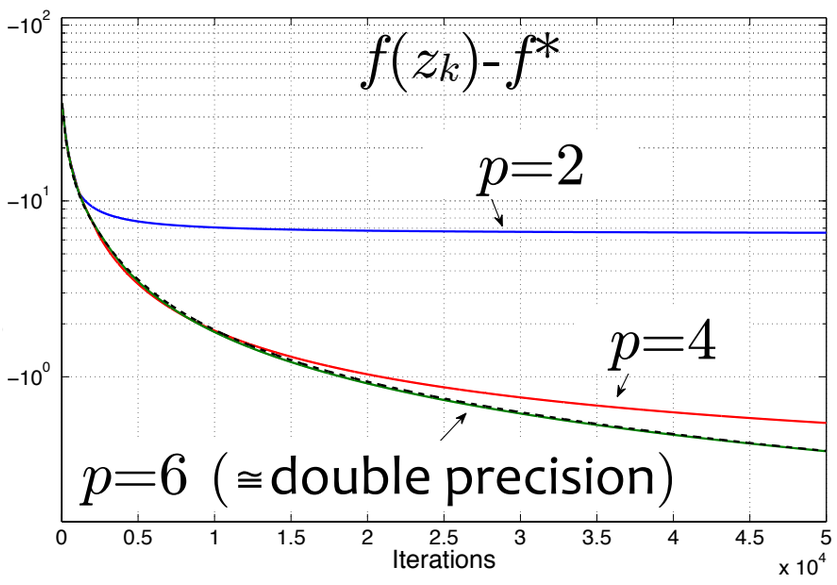
$$f(z_k) - f^\star \le \frac{L}{2(k+1)}(\|y^\star\|^2 + \|y_0\|^2) + \delta$$

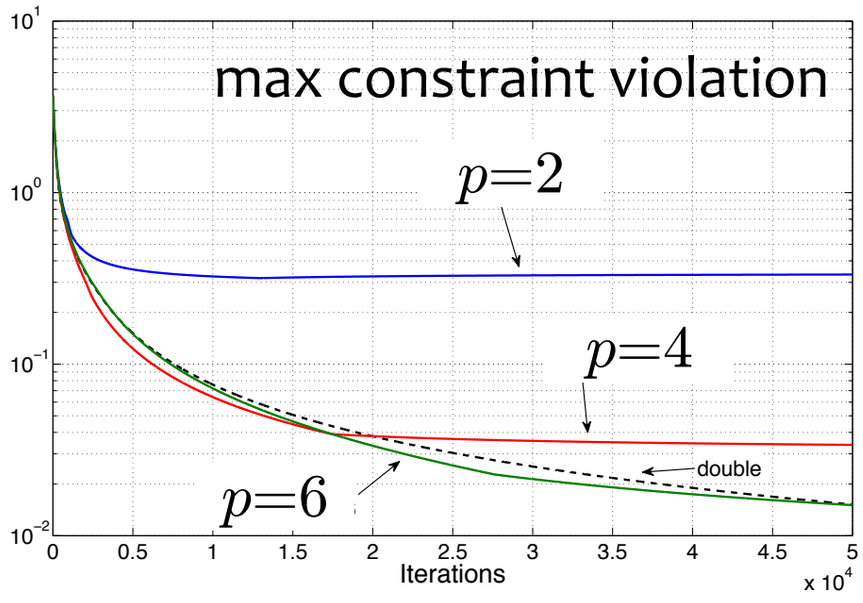
- Design guidelines (precision)

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

fractional bits

desired solution "quality"

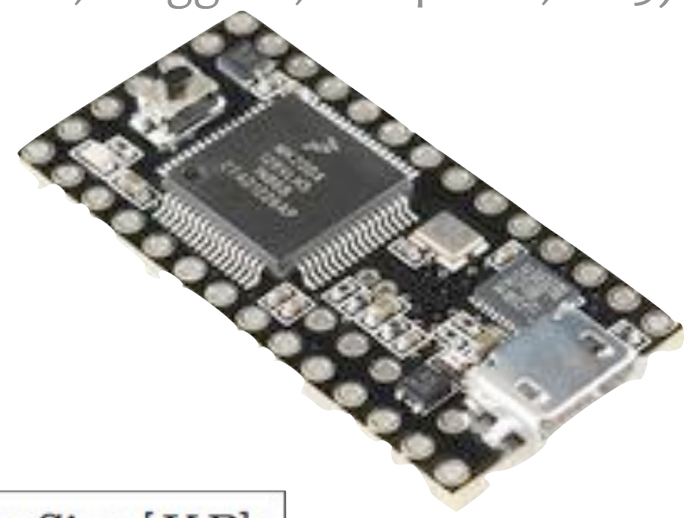




- Design guidelines for required **#integer bits** to avoid **overflow** also available

HARDWARE TESTS (FLOATING VS FIXED POINT)

(Patrinos, Guiggiani, Bemporad, 2013)



32-bit Atmel SAM3X8E **ARM Cortex-M3** processing unit:
84 MHz, 512 KB of flash memory and 100 KB of RAM.

Table 1
Fixed-point hardware implementation

Size [variables/constraints]	Time $[ms]$	Time per iteration $[\mu s]$	Code Size $[KB]$
10/20	22.9	226	15
20/40	52.9	867	17
40/80	544.9	3382	27
60/120	1519.8	7561	43

fixed point

Table 2
Floating-point hardware implementation

Size [variables/constraints]	Time $[ms]$	Time per iteration $[\mu s]$	Code Size $[KB]$
10/20	88.6	974	16
20/40	220.1	3608	21
40/80	2240	13099	40
60/120	5816	30450	73

floating point

fixed-point about 4x faster than floating-point

ADMM METHOD FOR QP

(Boyd et al., 2010)

- Alternating Directions Method of Multipliers (ADMM) for QP

$$\begin{array}{ll}\min & \frac{1}{2}x'Qx + q'x \\ \text{s.t.} & \ell \le Ax \le u\end{array}$$

- Separable form of QP:

$$\begin{array}{ll}\min & \frac{1}{2}x'Qx + q'x + g(z) \\ \text{s.t.} & Ax - z = 0\end{array} \qquad g(z) = \begin{cases} 0 & \text{if } \ell \le z \le u \\ +\infty & \text{otherwise}\end{cases}$$

- Scaled ADMM iterations:

$$\begin{aligned} x^{k+1} &= -(Q + \rho A^T A)^{-1}(\rho A^T(y^k - z^k) + q) \\ z^{k+1} &= \min\{\max\{Ax^{k+1} + y^k, \ell\}, u\} \\ y^{k+1} &= y^k + Ax^{k+1} - z^{k+1} \end{aligned}$$

"integral action"

(ρy = dual vector)

```
while i<maxiter,
    i=i+1;
    x=-iM*(c+A'*(rho*(u-z)));
    if i<maxiter,
        Ax=A*x;
        z=max(min(Ax+u,ub),lb);
        u=u+Ax-z;
    end
end
```

(9 lines EML code)

(~40 lines of C code)

REGULARIZED ADMM METHOD FOR QP

(Banjac, Stellato, Moehle, Goulart, Bemporad, Boyd, 2017)

- Scaled and regularized ADMM iterations:

$$\begin{aligned} x^{k+1} &= -(Q+\gamma A^TA+\epsilon I)^{-1}(q-\epsilon x_k+\gamma A^T(y^k-z^k)) \\ z^{k+1} &= \min\{\max\{Ax^{k+1}+y^k,\ell\},u\} \\ y^{k+1} &= y^k+Ax^{k+1}-z^{k+1} \end{aligned}$$

$Q \geq 0$

$\epsilon \geq 0$

ρy = dual vector

- Infeasibility detection:

$$\left\| \frac{y_k}{-u'\max\{y_k,0\}+l'\max\{-y_k,0\}} \right\|_\infty \leq \epsilon_I$$

- Unboundedness detection:

$$v_k = \frac{x_k}{-c'x_k},\ \|Qv_k\|_\infty \leq \epsilon_U,\ \begin{cases} A^iv_k \leq \epsilon_U \ \&\ u^i < +\infty \\ A^iv_k \geq -\epsilon_U \ \&\ \ell^i > -\infty \end{cases}$$

- Simple, fast, robust. Only needs to factorize $\begin{bmatrix} Q+\epsilon I & A' \\ A & -\gamma I \end{bmatrix}$ once

osQP solver https://github.com/oxfordcontrol/osqp

SCALING (OR PRECONDITIONING)

- **Preconditioning** can improve convergence rate of iterative algorithms (in particular first-order methods are very sensitive to scaling)

(Giselsson, Boyd, 2015)

primal QP

$$\begin{array}{ll} \min\limits_{z} & \frac{1}{2} z' H z + f' z \\ \text{s.t.} & Gz \leq W \end{array}$$

$$\begin{aligned} M &= GH^{-1}G' \\ d &= GH^{-1}f + W \end{aligned}$$

- **Dual scaling** (Jacobi scaling): (Bertsekas, 2009)

$$\begin{array}{ll} \min\limits_{y} & \frac{1}{2} y' M y + d' y \\ \text{s.t.} & y \geq 0 \end{array}$$

dual QP

scaling $y = P y_s$

$$P = \text{diag}\left(\frac{1}{\sqrt{M_{ii}}}\right)$$

$$\begin{array}{ll} \min\limits_{y_s} & \frac{1}{2} y_s' (PMP) y_s + d' P y_s \\ \text{s.t.} & y_s \geq 0 \end{array}$$

scaled dual QP

- Equivalent to just **scale constraints** in primal problem: $\frac{1}{\sqrt{M_{ii}}} G_i z \leq \frac{1}{\sqrt{M_{ii}}} W_i$

- Primal solution: $z^* = -H^{-1}((PG)' y_s^* + f)$

SCALING EXAMPLE

- AFTI-16 linearized model (4th order, unstable)
  - 2 inputs (elevator & flap angles)
  - 2 outputs (attack & pitch angles)

- MPC problem:
  - prediction horizon=10, control horizon=2 (-> 4 free optimization vars)
  - feasibility threshold = optimality threshold = 0.01

only input constraints				
AFTI	Max		Avg	
	NS	S	NS	S
GPAD	287	153	28	21
ADMM	1458	456	148	111
PQP	2396	2119	152	138
DRSA	441	278	57	32
FBN	6	7	3	2

input + output constraints				
AFTI2	Max		Avg	
	NS	S	NS	S
GPAD	1605	498	668	92
ADMM	3518	1756	741	208
PQP	631075	49472	16293	3581
DRSA	3142	901	473	160
FBN	118	32	7	4

number of iterations (NS= Not Scaled, S=Scaled)

PQP = Projection-free parallel QP (Di Cairano, Brand, Bortoff, 2013)
DRSA = Accelerated Douglas-Rachford Splitting (Patrinos, Stella, Bemporad, 2014)

ODYS' QP SOLVER

- General purpose QP solver
  designed for embedded control applications

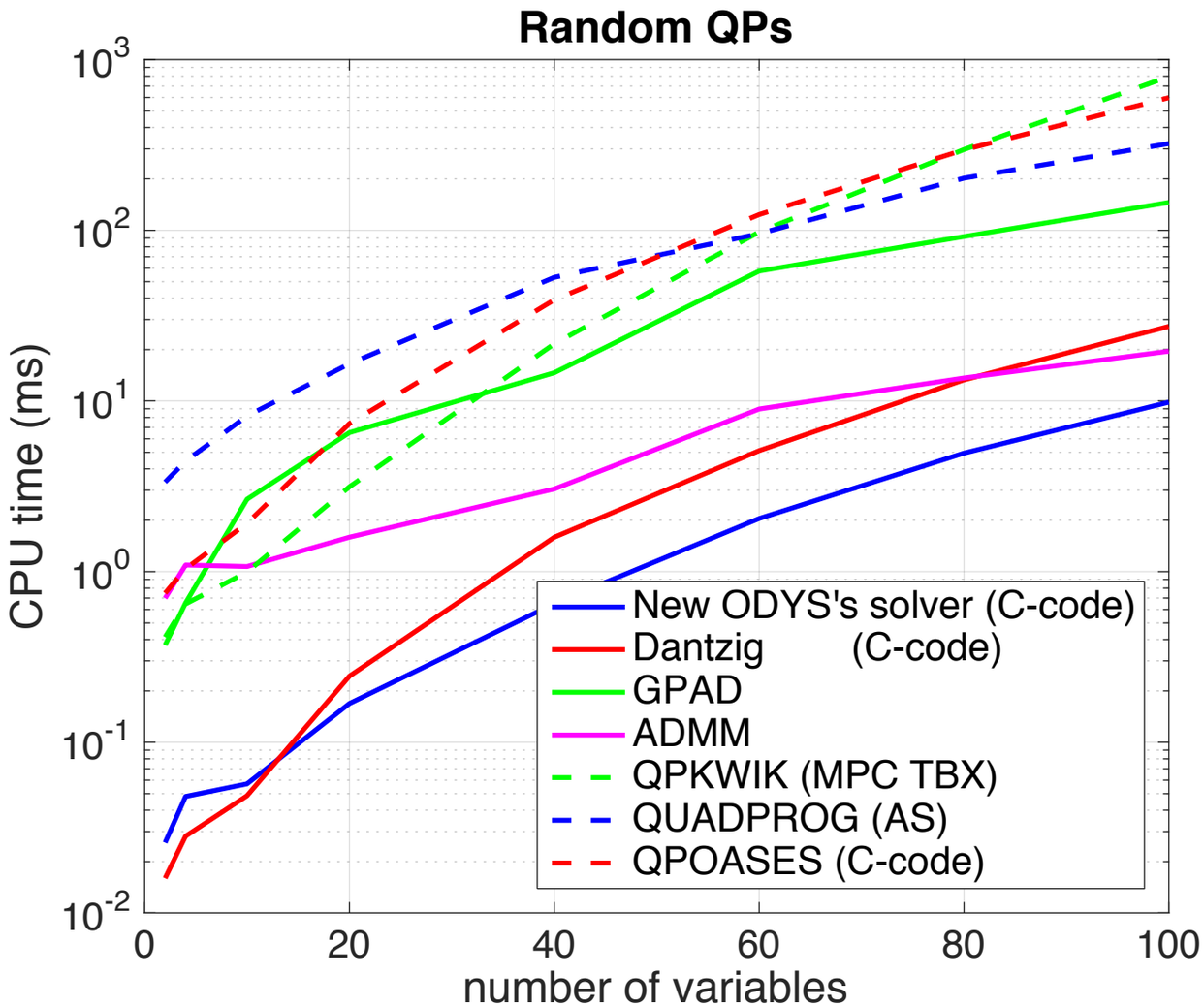
$$\begin{aligned} \min \quad & \frac{1}{2}x'Qx + c'x \\ \text{s.t.} \quad & Ax \leq b \\ & A_{\text{eq}}x = b_{\text{eq}} \end{aligned}$$

- Implements a state-of-the-art **active-set** method for QP
- Completely written in ANSI-C
- Emphasis on robustness (especially in single precision), speed of execution, low memory requirements
- No real need for preconditioning QP matrices

ODYS
Advanced Controls & Optimization

QP SOLVER COMPARISON - SPEED

- Random QP problems with n variables and $4n$ constraints:
- **Double precision** arithmetic (this Mac)

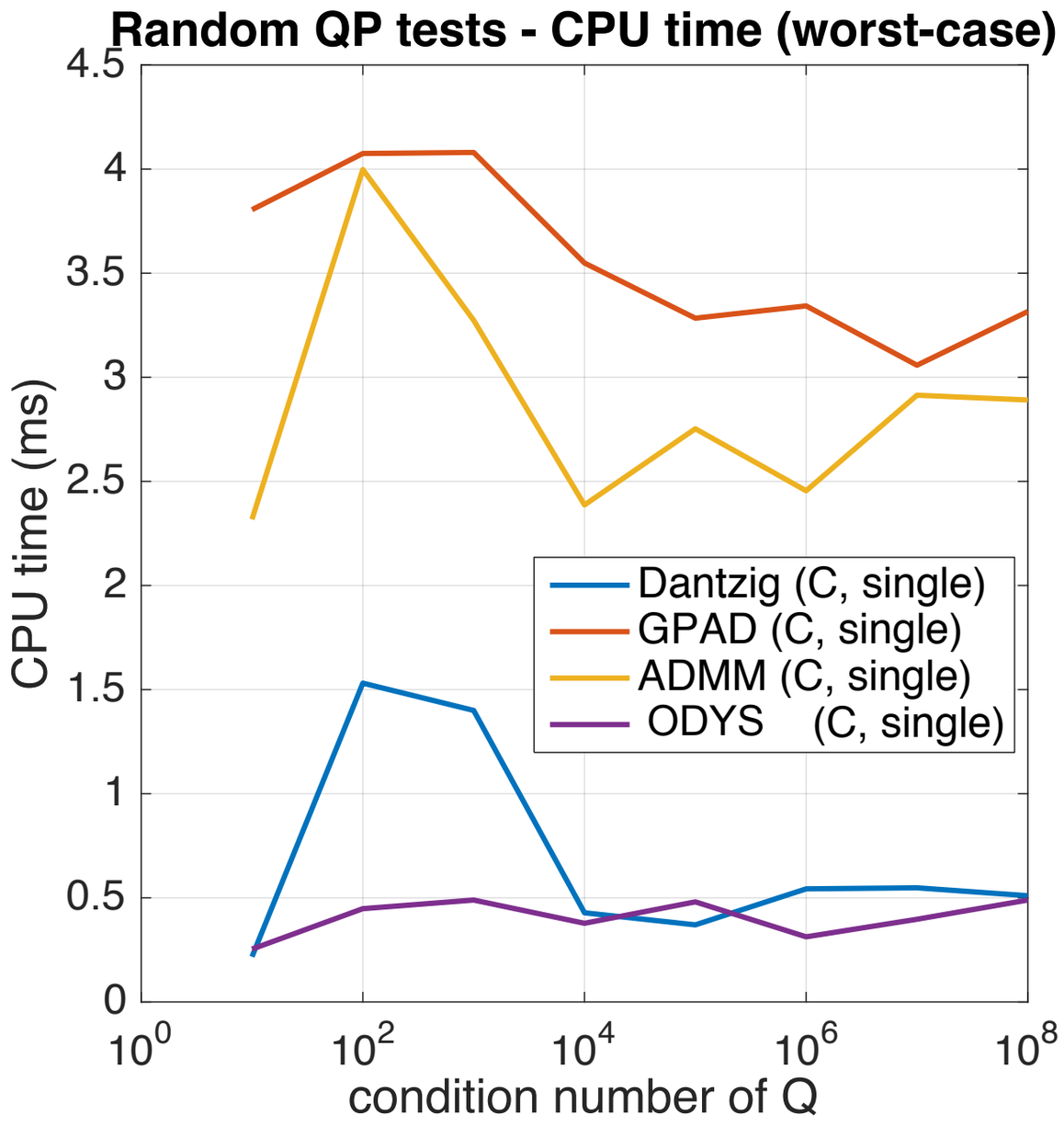


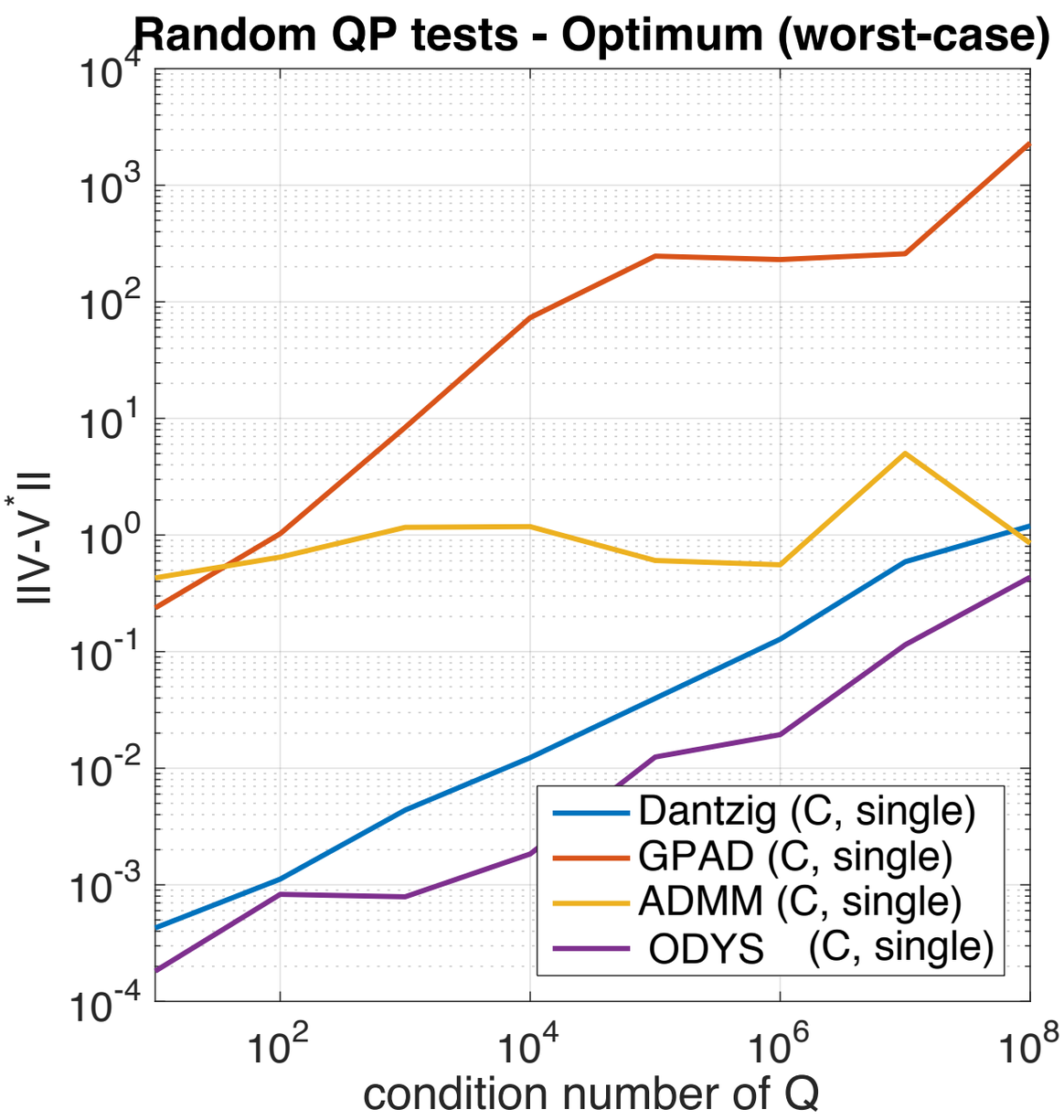
worst-case execution time

QP SOLVER COMPARISON - ROBUSTNESS

- Random QP problems with 20 variables and 100 constraints:

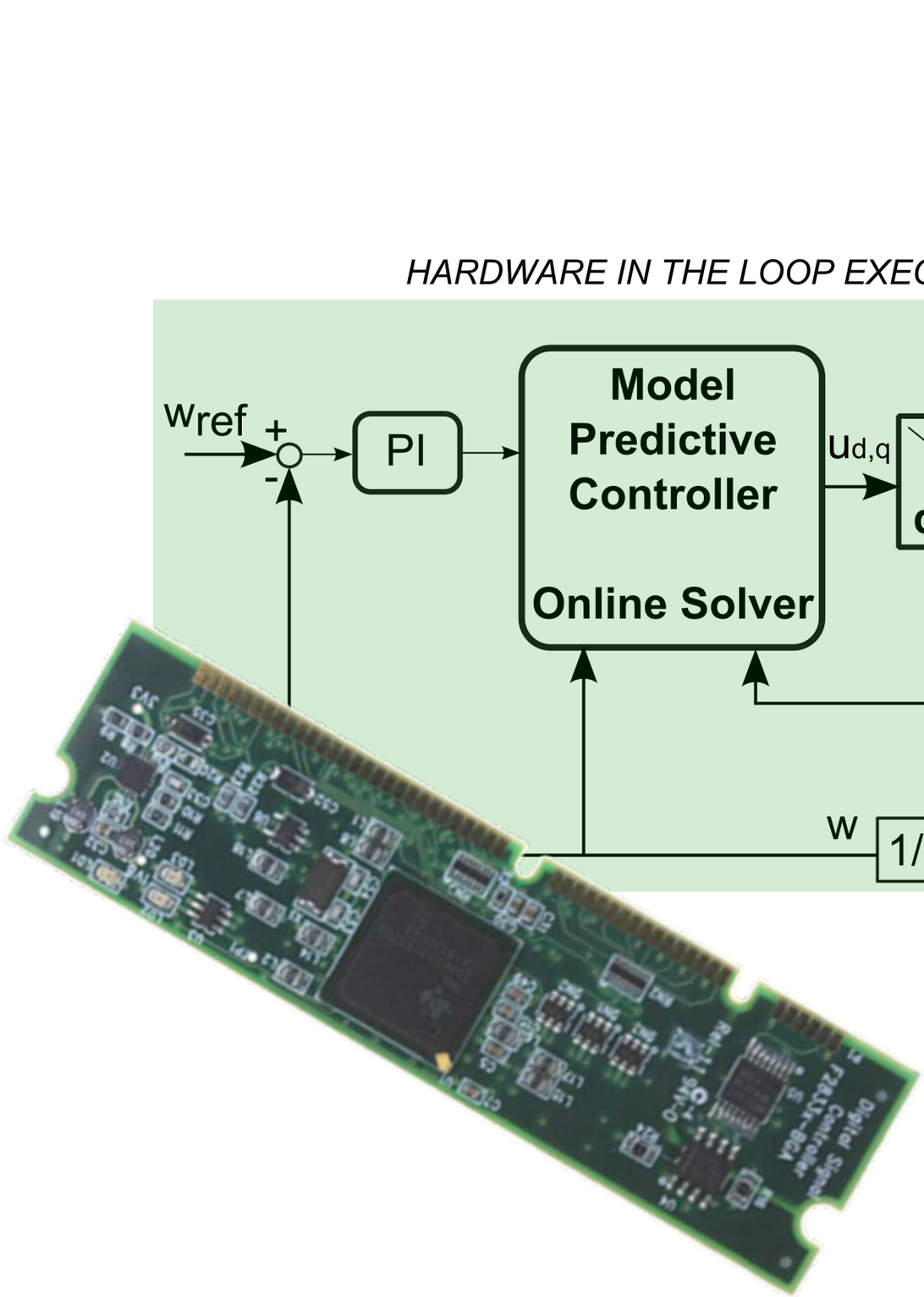
- **Single precision** arithmetic (this Mac)





HARDWARE-IN-THE-LOOP EXPERIMENTS (DSP)

- Control of a brushless DC motor



TMS320F28335 controlCARD
(Real-time Control Applications)

- 32-bit Floating Point (IEEE-754);
- 150MHz clock;
- 68KB Ram / 512KB Flash.

HARDWARE-IN-THE-LOOP EXPERIMENTS (DSP)

- Linear MPC setup:

$$\begin{cases} x(k+1) &= Ax(k) + B_u u(k) + B_v(k) \\ y(k) &= Cx(k) \end{cases}$$

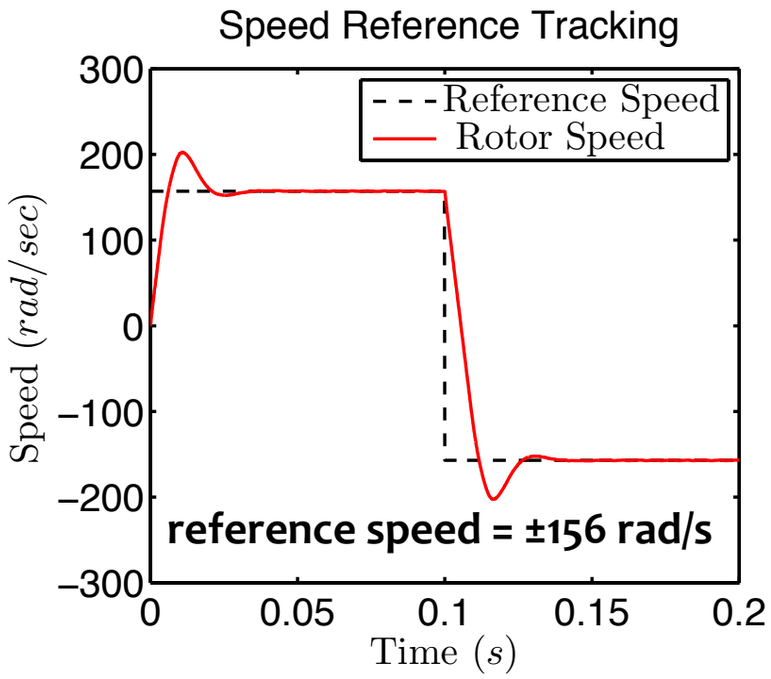
$$x = \begin{bmatrix} i_d \\ i_q \end{bmatrix} \qquad u = \begin{bmatrix} u_d \\ u_q \end{bmatrix}$$

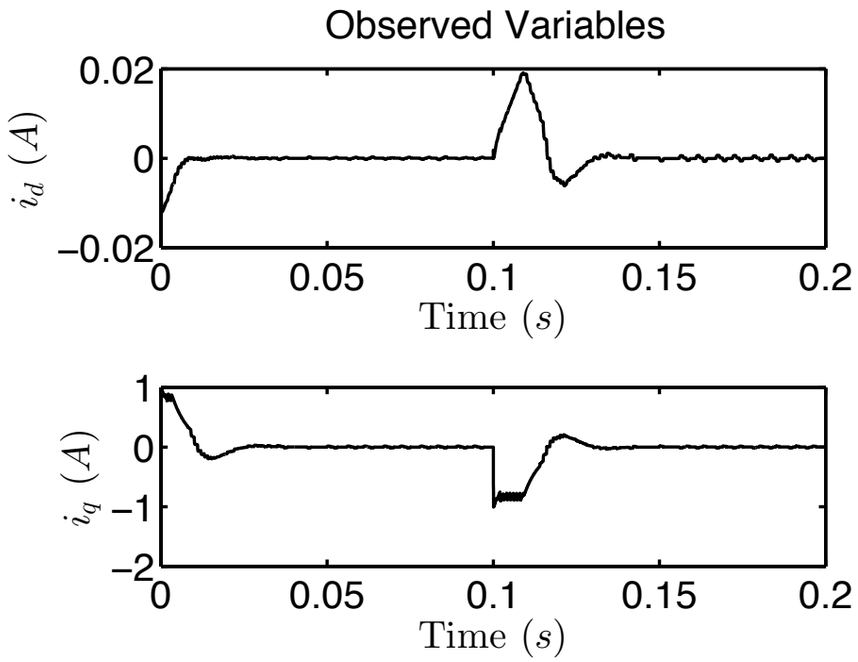
$$A = \begin{bmatrix} -\frac{R}{L} & \bar{\omega} \\ 0 & -\frac{R}{L} \end{bmatrix}, \; B_u = \begin{bmatrix} \frac{1}{L} & 0 \\ 0 & \frac{1}{L} \end{bmatrix}, \; B_v = \begin{bmatrix} 0 \\ -\omega^2 \frac{\lambda}{L} \end{bmatrix}, \; C = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

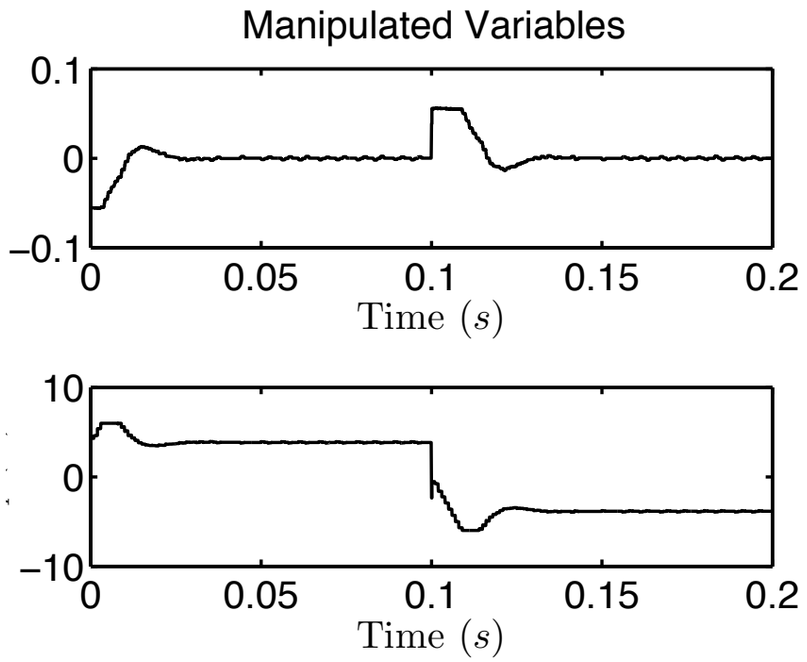
R = stator resistance [Ω]
L = stator inductance [H]
ω = rotor speed [rad/s]
λ = motor flux leakage [Wb]

prediction horizon = 4
control horizon = 2
box constraints on i_q and u_q

$$-\epsilon I_{\max} \leq i_d \leq \epsilon I_{\max}$$
$$V_{\max} \leq u_q \leq V_{\max}$$



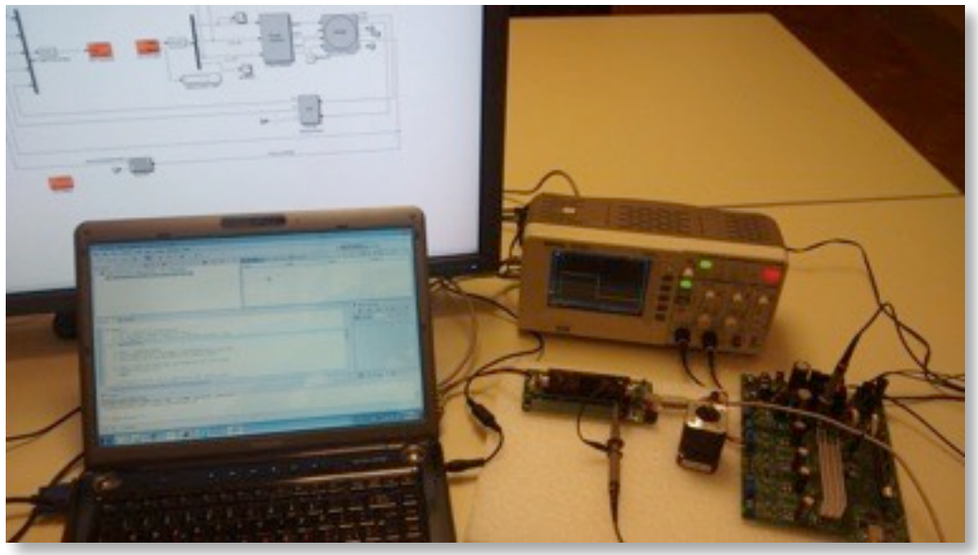




HARDWARE-IN-THE-LOOP EXPERIMENTS (DSP)

TMS320F28335 controlCARD
(Real-time Control Applications)

- 32-bit Floating Point (IEEE-754);
- 150MHz clock;
- 68KB Ram / 512KB Flash.



var × constr.	GPAD	ODYS (AS)	ADMM	FBN
4 × 16	332 μs (18)	120 μs (3)	1.42 ms (62)	208 μs (2)
8 × 24	1.1 ms (22)	446 μs (5)	4 ms (77)	396 μs (2)
12 × 32	2.59 ms (27)	1.19 ms (7)	8.25 ms (82)	652 μs (2)

- **Active set** (AS) methods are usually the best on small problems:
  - excellent quality solutions within few iterations
  - less sensitive to preconditioning (= behavior is more predictable)
  - no need for advanced linear algebra packages

* GPAD = Dual Accelerated Gradient Projection (Patrinos, Bemporad, 2014)
* FBN = Forward-Backwards Netwon (proximal method) (Patrinos, Guiggiani, Bemporad, 2014)
* ADMM = Alternating Directions Method of Multipliers (Boyd et al., 2010)

REQS FOR EMBEDDED OPTIMIZATION ALGORITHMS

- Distinguish between **off-line** and **on-line** computations
  - **off-line:** double precision (MATLAB/Python/Julia) on a PC (arbitrarily complex)
  - **on-line:** single-precision/fixed-point on a μ-processor (please simple!)
- Have degrees of freedom to trade off **throughput** vs. **memory**
- Typical problem size: **10÷50 variables, 10÷100 constraints**
- **Numerical robustness** (of on-line computations): must perform well in single precision (or even fixed-point) computations

REQS FOR EMBEDDED OPTIMIZATION ALGORITHMS

- **Limited linear algebra** operations (in on-line computations) for code simplicity (example: at most matrix-vector products)

- **Dense problems**, no need for sparse linear algebra

- **Suboptimal solutions are ok.** Optimality is of less concern than feasibility (the prediction model is approximate, the cost function usually comes from trial & error tuning)

CAN WE SOLVE QP'S USING LEAST SQUARES ?

The **Least Squares (LS)** problem is probably the most studied problem in numerical linear algebra


(Legendre, 1805)


(Gauss, <= 1809)

$$v = \arg\min \|Av - b\|_2^2$$

In MATLAB: >> **`v=A\b`** `% (1 character ! )`

- **Nonnegative Least Squares (NNLS):**

$$\begin{array}{ll} \min_v & \|Av - b\|_2^2 \\ \text{s.t.} & v \geq 0 \end{array}$$

- NNLS algorithm is very simple (**`750 chars in Embedded MATLAB`**)

(Lawson, Hanson, 1974)

ACTIVE-SET METHOD FOR NONNEGATIVE LEAST SQUARES

(Lawson, Hanson, 1974)

$$\begin{aligned} \min_v \quad & \|Av-b\|_2^2 \\ \text{s.t.} \quad & v \geq 0 \end{aligned}$$

NNLS algorithm: *While maintaining primal var v feasible, keep switching active set until dual var w is also feasible*

1) $\mathcal{P} \leftarrow \emptyset$, $v \leftarrow 0$;
2) $w \leftarrow A'(Av-b)$;
3) **if** $w \geq 0$ **or** $\mathcal{P} = \{1,\dots,m\}$ **then go to** Step 11;
4) $i \leftarrow \arg\min_{i \in \{1,\dots,m\}\setminus\mathcal{P}} w_i$, $\mathcal{P} \leftarrow \mathcal{P} \cup \{i\}$;
5) $y_{\mathcal{P}} \leftarrow \arg\min_{z_{\mathcal{P}}} \|((A')_{\mathcal{P}})' z_{\mathcal{P}} - b\|_2^2$, $y_{\{1,\dots,m\}\setminus\mathcal{P}} \leftarrow 0$;
6) **if** $y_{\mathcal{P}} \geq 0$ **then** $v \leftarrow y$ and **go to** Step 2;
7) $j \leftarrow \arg\min_{h \in \mathcal{P}:\ y_h \leq 0} \left\{ \frac{v_h}{v_h - y_h} \right\}$;
8) $v \leftarrow v + \frac{v_j}{v_j - y_j}(y - v)$;
9) $\mathcal{I} \leftarrow \{h \in \mathcal{P} : v_h = 0\}$, $\mathcal{P} \leftarrow \mathcal{P} \setminus \mathcal{I}$;
10) **go to** Step 5;
11) $v^* \leftarrow v$; **end**.

- NNLS algorithm is very simple (**`750 chars in Embedded MATLAB`**)
- The key operation is to solve a **standard LS problem** at each iteration (via QR, LDL, or Cholesky factorization)

SOLVING QP'S VIA NONNEGATIVE LEAST SQUARES

(Bemporad, IEEE TAC, 2016)

- Use NNLS to solve strictly convex QP

$$\begin{array}{ll}\min\limits_{z} & \frac{1}{2}z'Qz + c'z \\ \text{s.t.} & Gz \le g\end{array}$$

QP

$$u \triangleq Lz + L^{-T}c$$

complete the squares

$$\begin{aligned} Q &= L'L \\ M &= GL^{-1} \\ d &= b + GQ^{-1}c \end{aligned}$$

$$\begin{array}{ll}\min\limits_{u} & \frac{1}{2}\|u\|^2 \\ \text{s.t.} & Mu \le d\end{array}$$

Least Distance Problem

$$\begin{array}{ll}\min\limits_{y} & \frac{1}{2}\left\|\begin{bmatrix} M' \\ d' \end{bmatrix} y + \begin{bmatrix} 0 \\ 1 \end{bmatrix}\right\|_2^2 \\ \text{s.t.} & y \ge 0\end{array}$$

Nonnegative Least Squares

residual $= 0$?

yes

QP problem infeasible

no

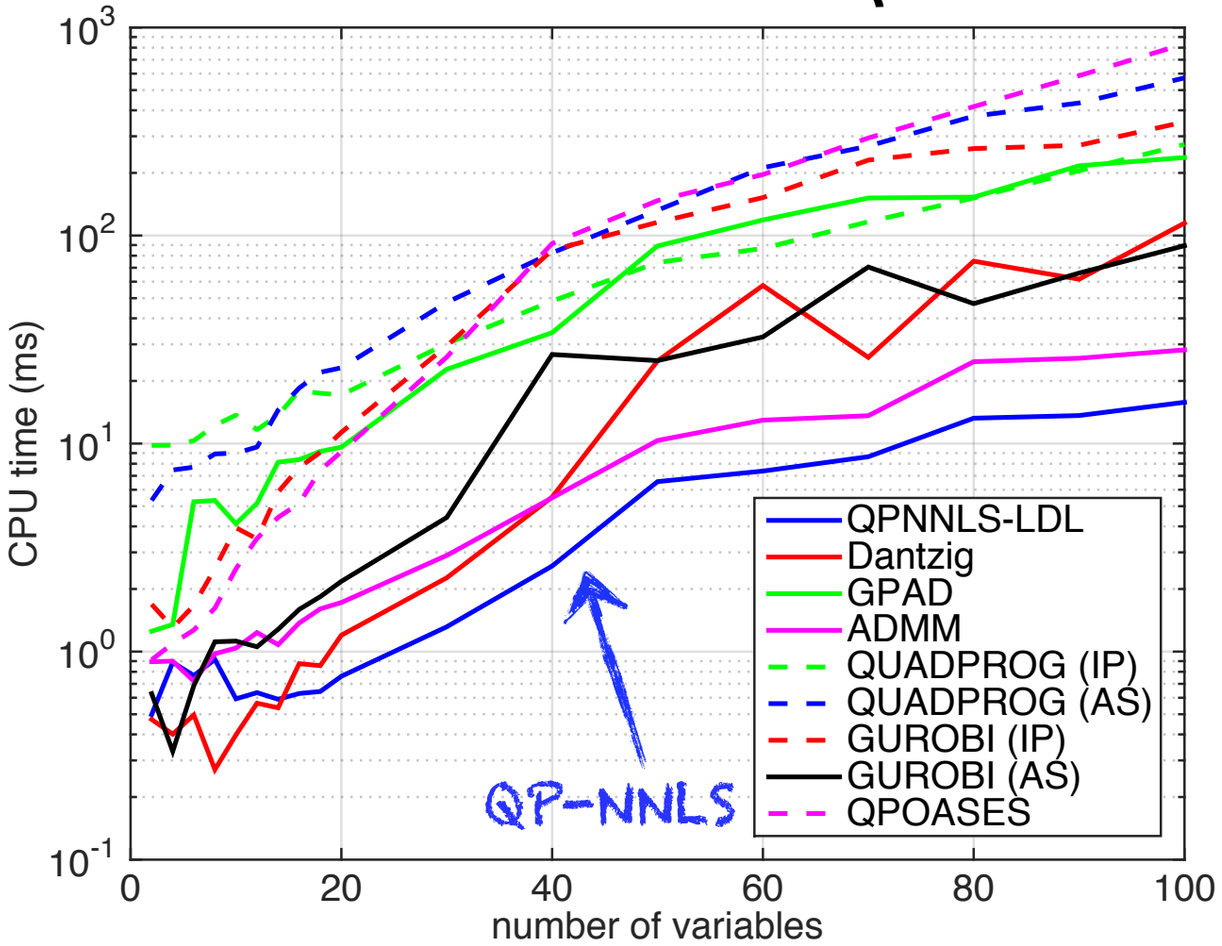
$$z^* = -\frac{1}{1 + d'y^*}L^{-1}M'y^* - Q^{-1}c$$

retrieve primal solution

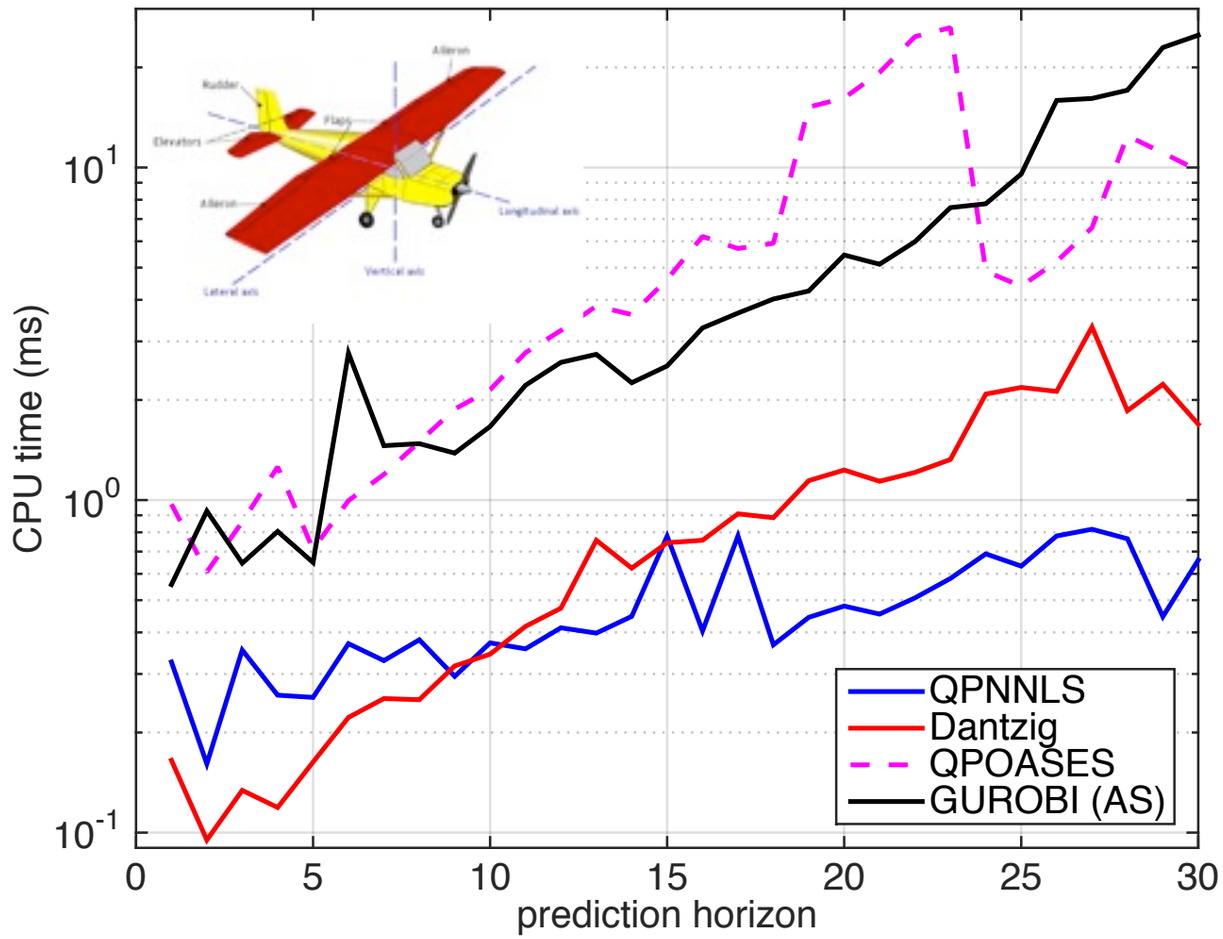
SOLVING QP VIA NNLS: NUMERICAL RESULTS

(Bemporad, IEEE TAC, 2016)

worst-case over 100 random QP instances



worst-case occurred during entire simulation*



* Step t=0 not considered for QPOASES not to penalize the benefits of the method with warm starting

- A rather **fast** and relatively **simple-to-code** QP solver
- Extended to solving mixed-integer QP's (Bemporad, NMPC 2015)
- Not very robust (e.g., in single precision arithmetics)

SOLVING QP VIA NNLS: ROBUST ALGORITHM

(Bemporad, 2016)

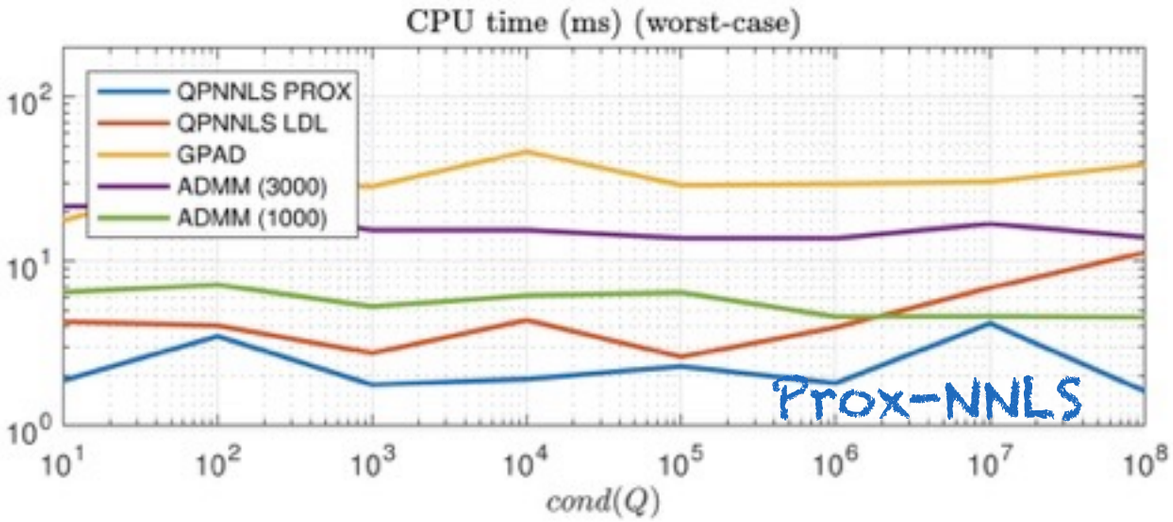
- **Key idea:** solve a sequence of regularized QP problems

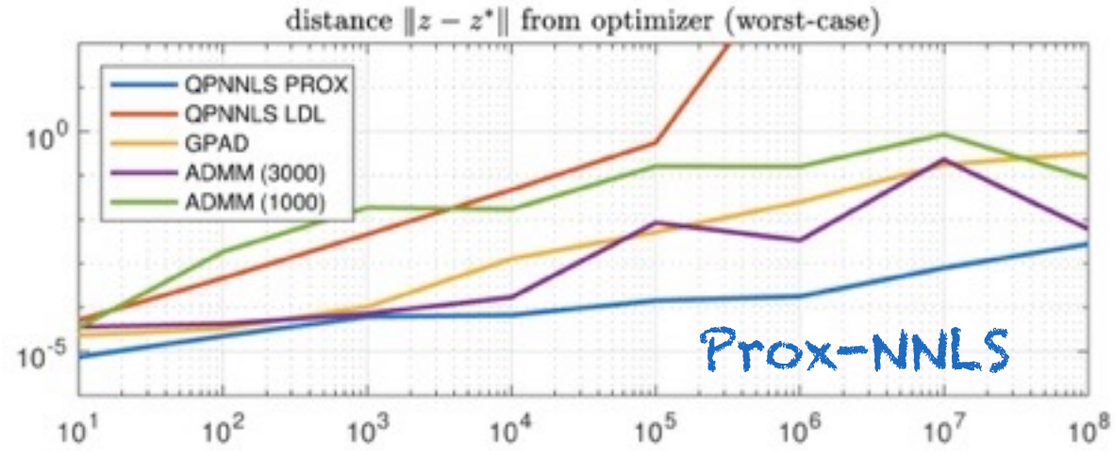
$$\begin{aligned} z_{k+1} = \arg\min_z \quad & \tfrac{1}{2} z'Qz + c'z + \tfrac{\epsilon}{2}\|z - z_k\|_2^2 \\ \text{s.t.} \quad & Az \le b \\ & Gx = g \end{aligned}$$

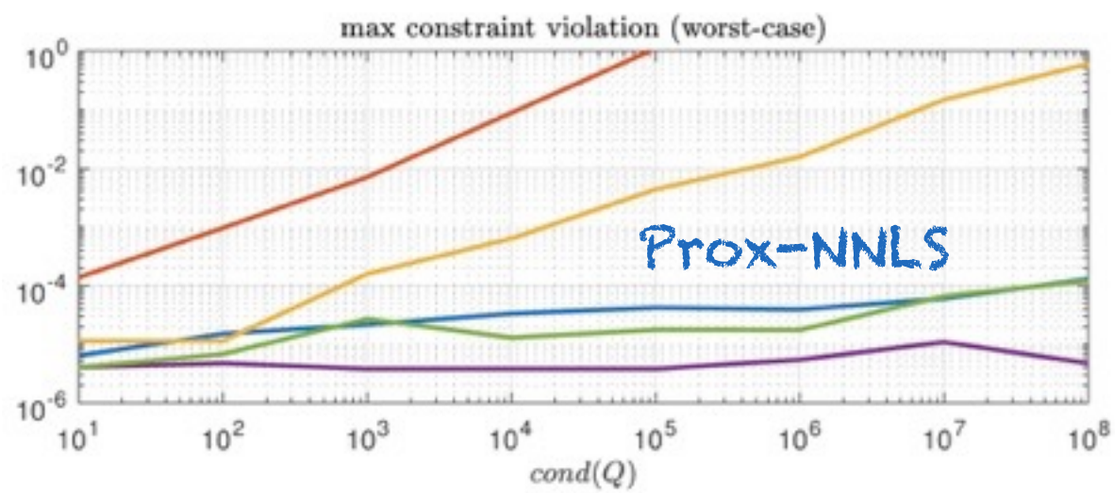
proximal-point algorithm iterations converge to the optimal solution !

- Main advantage: primal Hessian $Q + \epsilon I$ can be arbitrarily well conditioned !

(tradeoff robustness/CPU time)







single precision arithmetic,
30 vars, 100 constraints (this Mac)

EMBEDDED MPC WITHOUT SOLVING QP'S ON LINE

dynamical model (based on data)

optimization algorithm

$$\min_z \ \frac{1}{2} z'Qz + c'z \quad \text{s.t.} \quad Az \le b$$

embedded model-based optimizer

reference $r(t)$

input $u(t)$

process

output $y(t)$

measurements

- Can we implement MPC **without** an embedded **optimization solver** ?

YES !

ON-LINE VS OFF-LINE OPTIMIZATION

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

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

- **On-line** optimization: given $x(t)$ solve the problem at each time step t (the control law $u{=}u_0{}^*(x)$ 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 $u{=}u_0{}^*(x)$ **explicitly**

  → multi-parametric Quadratic Program (mp-QP)

MULTIPARAMETRIC PROGRAMMING PROBLEM

Given the optimization problem

$$\begin{aligned} \min_z \quad & f(z, x) \\ \text{s.t.} \quad & g(z, x) \leq 0 \end{aligned}$$

and a set X of parameters, determine:

- the **set of feasible parameters** X^* of all $x \in X$ for which the problem admits a solution (that is $g(z,x) \leq 0$ for some z)
- the **value function** $V^* : X^* \to \mathbb{R}$ associating the optimal value $V^*(x)$ to each x
- An **optimizer function** $z^* : X^* \to \mathbb{R}^s$

MULTIPARAMETRIC QUADRATIC PROGRAMMING

(Bemporad et al., 2002)

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

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

- ***Objective:*** solve the QP off line **for all** $x \in X$ to get the optimizer function z^*, and therefore the MPC control law $u(x) = [I\ 0\ \dots\ 0] z^*(x)$ **explicitly**

- Assumptions:

$$\begin{bmatrix} H & F \\ F' & Y \end{bmatrix} \succeq 0$$

always satisfied if mpQP comes from an MPC problem !

$$H = H' \succ 0$$

always satisfied if weight matrix $R > 0$

LINEARITY OF SOLUTION

(Bemporad, Morari, Dua, Pistikopoulos, 2002)

Fix a point $x_0 \in X$ in the parameter space

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

→ solve QP to find $z^*(x_0),\ \lambda^*(x_0)$

→ identify active constraints at $z^*(x_0)$

→ form matrices $\tilde{G},\ \tilde{W},\ \tilde{S}$ by collecting active constraints: $\tilde{G}z^*(x_0) - \tilde{W} - \tilde{S}x_0 = 0$

KKT optimality conditions:

$$\begin{array}{ll} (1)\ \ Hz + Fx + G'\lambda = 0 & (2)\ \ \tilde{G}z - \tilde{W} - \tilde{S}x = 0 \\ (3)\ \ \lambda_i(G^i z - W^i - S^i x) = 0 & (4)\ \ \hat{G}z \le \hat{W} + \hat{S}x \\ (5)\ \ \tilde{\lambda}_i \ge 0,\ \ \hat{\lambda}_i = 0 & \end{array}$$

$\hat{G}$=rows of G not in $\tilde{G}$ (inactive constraints)

From (1): $z = -H^{-1}(Fx + \tilde{G}'\tilde{\lambda})$

From (2):

$$\tilde{\lambda}(x) = -(\tilde{G}H^{-1}\tilde{G}')^{-1}(\tilde{W} + (\tilde{S} + \tilde{G}H^{-1}F)x)$$

$$z(x) = H^{-1}\left[\tilde{G}'(\tilde{G}H^{-1}\tilde{G}')^{-1}(\tilde{W} + (\tilde{S} + \tilde{G}H^{-1}F)x) - Fx\right]$$

→ In some neighborhood of x_0, λ and z are **explicit affine functions** of x

(Zafiriou, 1990)

MULTIPARAMETRIC QP ALGORITHM

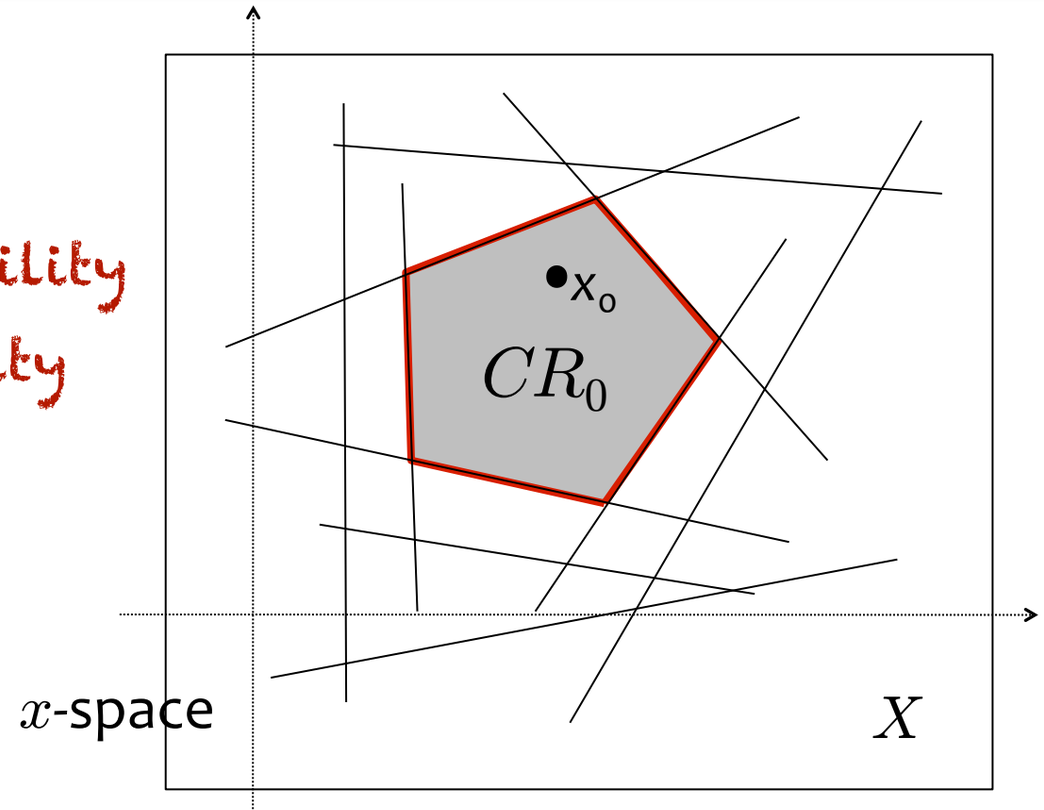
- Impose conditions (4) and (5):

$$\begin{array}{rcl} \hat{G}z(x) & \leq & \hat{W} + \hat{S}x \\ \tilde{\lambda}(x) & \geq & 0 \end{array}$$

← primal feasibility

← dual feasibility

→ linear inequalities in x !



- Remove redundant constraints (this requires solving LP's):

→ critical region CR_0

$$CR_0 = \{x \in X : \ \mathcal{A}x \leq \mathcal{B}\}$$

- CR_0 is the set of all and only parameters x for which $\tilde{G}$, $\tilde{W}$, $\tilde{S}$ is the optimal combination of active constraints at the optimizer

REMOVING REDUNDANT CONSTRAINTS VIA LP

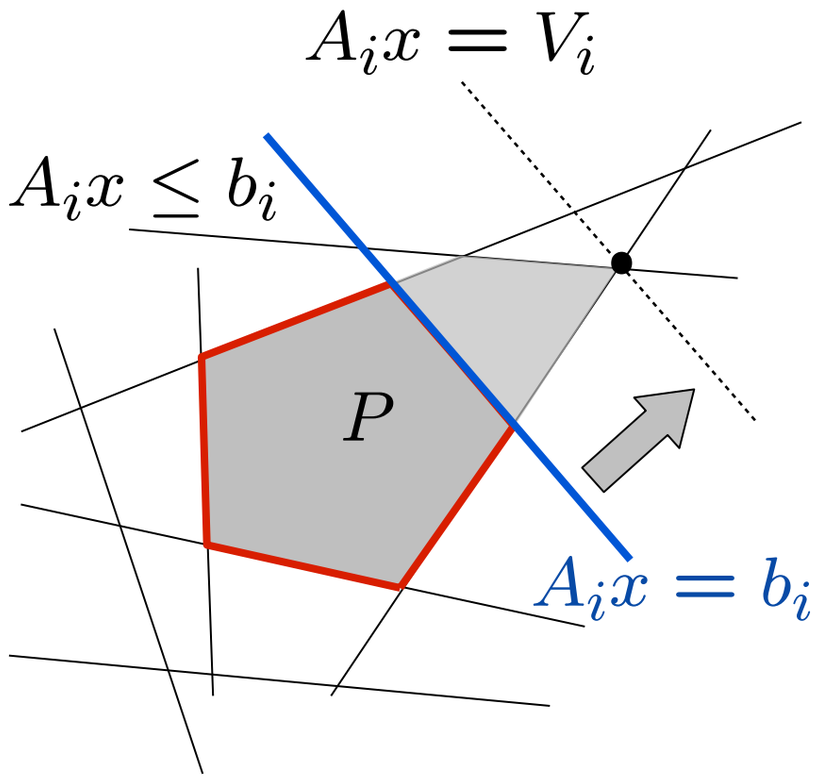
- A **minimal hyperplane representation** of a polyhedron P can be computed by solving a sequence of linear programs

- Let $P = \{x \in \mathbb{R}^n : \ A_i x \leq b_i, \ i = 1, \ldots, m\}$

- For each i=1,...,m solve the linear program

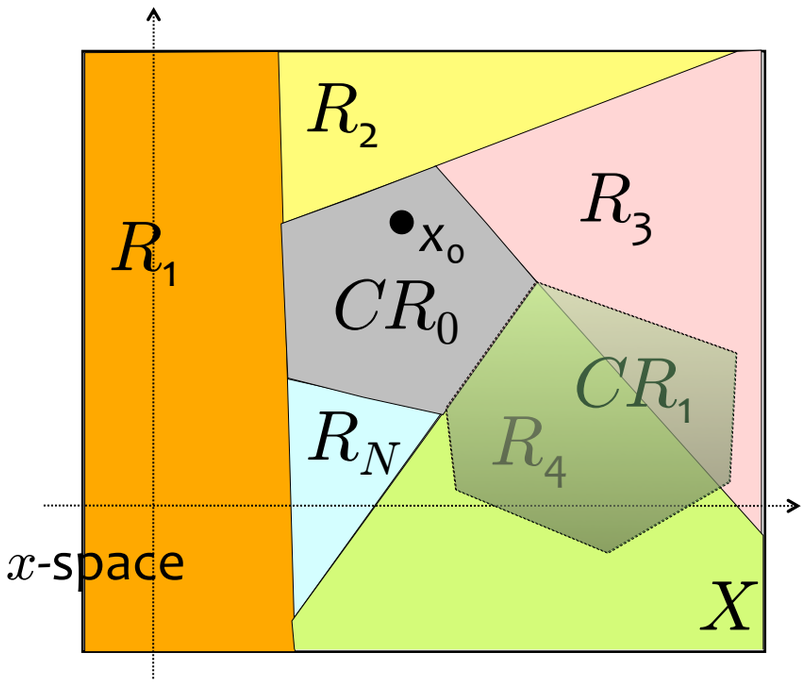
$$\begin{aligned} V_i = \max_x \ & A_i x \\ \text{s.t.} \ & A_j x \leq b_j, \ j = 1, \ldots, m, j \neq i \end{aligned}$$

$V_i \in \mathbb{R} \cup \{+\infty\}$



- If $V_i < b_i$ $(V_i \leq b_i)$ then constraint #i is redundant (weakly redundant)

MULTIPARAMETRIC QP SOLVER #1



Method #1: Split and proceed iteratively

(Bemporad, Morari, Dua, Pistikopoulos, 2002)

$$CR_0 = \{x \in X : \ \mathcal{A}x \leq \mathcal{B}\}$$

$$R_i = \{x \in X : \ \mathcal{A}^i x > \mathcal{B}^i, \ \mathcal{A}^j x \leq \mathcal{B}^j, \ \forall j < i\}$$

Note: while CR_i is characterizing a set of active constraints, R_i is not

1) Use the above splitting only as a search procedure, don't split the CR

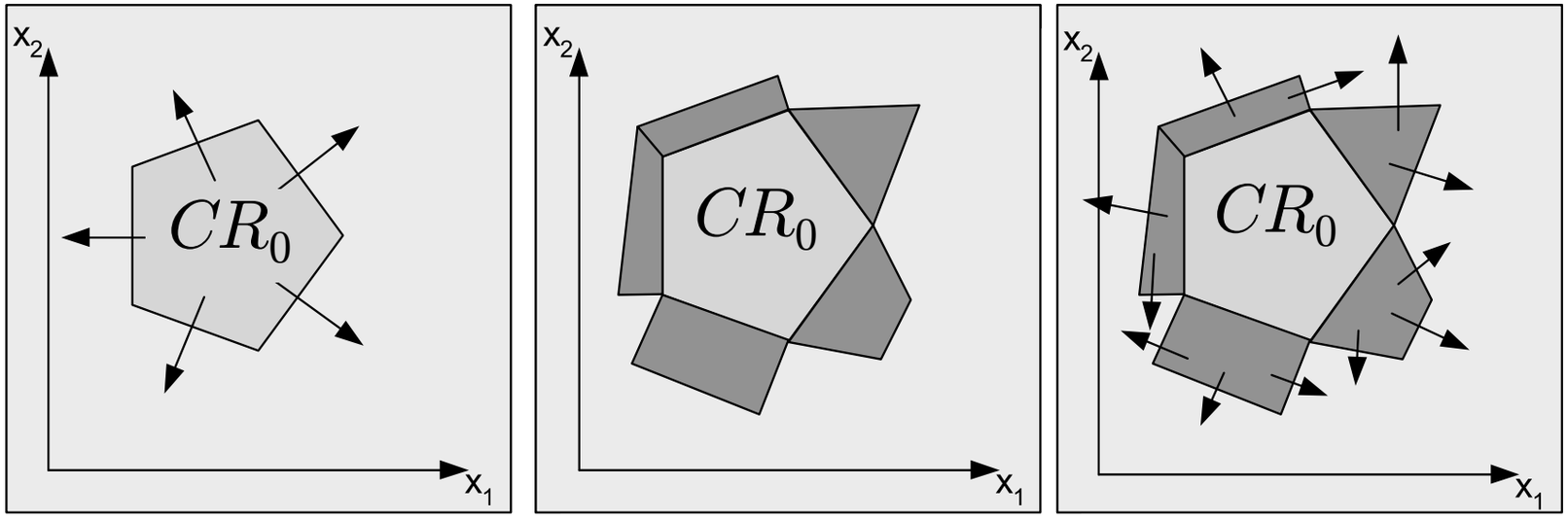
2) Remove duplicates of CR already found

After each recursion, one less combination of active constraints is available.

As the maximum total number of combinations is $\sum_{h=0}^{q} \binom{q}{h} = 2^q$, the procedure terminates after at most 2^q recursions (q=number of constraints)

MULTIPARAMETRIC QP SOLVER #2, #3

(Tøndel, Johansen, Bemporad, 2003)



Active set of neighboring region obtained by adding/removing active constraints based on knowledge of the type of crossed hyperplane in x-space

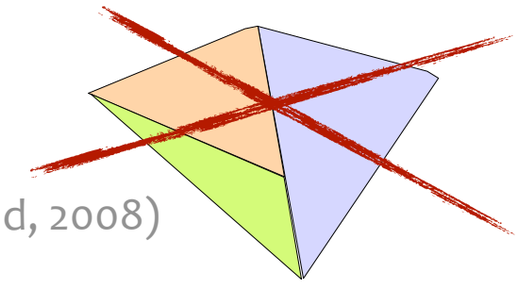
$$\begin{aligned} \widehat{G}^i z(x) &\leq \widehat{W}^i + \widehat{S}^i x \\ \tilde{\lambda}_j(x) &\geq 0 \end{aligned}$$

→ constraint #i **added** to active set (to maintain **feasibility** of solution)

→ constraint #j **withdrawn** from active set (to maintain **optimality** of solution)

Method #3: exploit the *facet-to-facet* property

(Spjøtvold, Kerrigan, Jones, Tøndel, Johansen, 2006)



(Spjøtvold, 2008)

Step out ϵ outside each facet, solve QP, get new region, iterate. (Baotic, 2002)

PROPERTIES OF MULTIPARAMETRIC QP

Theorem: Assume $H \succ 0$, $\begin{bmatrix} H & F \\ F' & Y \end{bmatrix} \succeq 0$ (always satisfied in QPs from MPC).

$$X^* = \{x \in \mathbb{R}^n : \ z^*(x) \text{ exists}\} \longrightarrow \text{polyhedron}$$

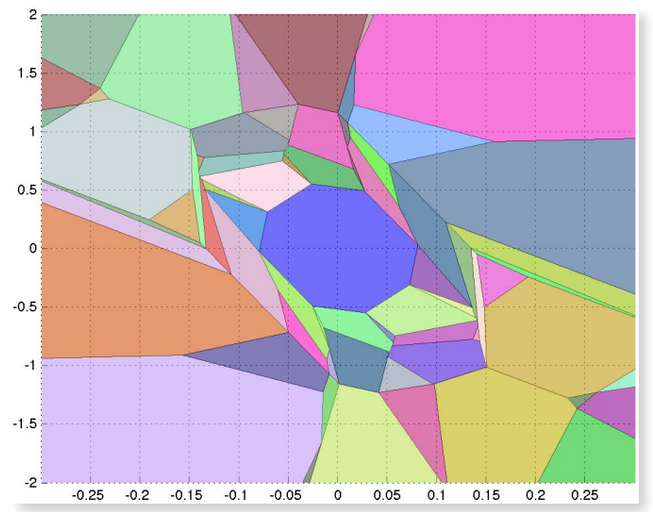
$$z^*(x) = \arg\min_z \ \tfrac{1}{2} z'Hz + x'F'z \quad \text{s.t.} \quad Gz \leq W + Sx \longrightarrow \text{continuous, piecewise affine}$$

$$V^*(x) = \tfrac{1}{2} x'Yx + \min_z \ \tfrac{1}{2} x'Hz + x'F'z \quad \text{s.t.} \quad Gz \leq W + Sx \longrightarrow \text{convex, continuous, piecewise quadratic, (even } C^1\text{, if no degeneracy)}$$

(Bemporad, Morari, Dua, Pistikopoulos, 2002)

Corollary: The linear MPC control law is continuous and piecewise affine

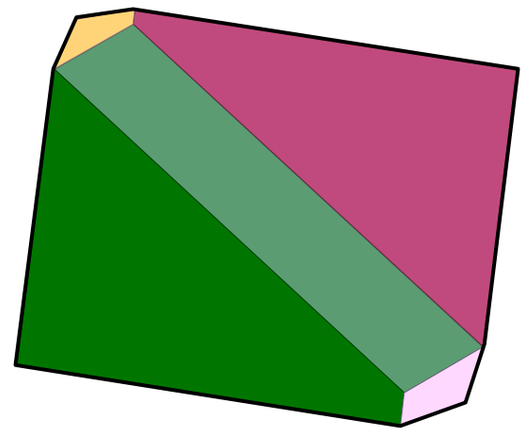
$$z^* = \begin{bmatrix} u_0^* \\ u_1 \\ \vdots \\ u_{N-1}^* \end{bmatrix} \qquad u(x) = \begin{cases} F_1 x + g_1 & \text{if} \quad H_1 x \leq K_1 \\ \vdots & \vdots \\ F_M x + g_M & \text{if} \quad H_M x \leq K_M \end{cases}$$



PROOF OF THEOREM

1) X^* is convex and polyhedral

$$X^* = \{x : \ \exists z \text{ such that } Gz \leq W + Sx\}$$



Let $x_\alpha = \alpha x_1 + (1-\alpha)x_2 \in X^*,\ x_1, x_2 \in X^*,\ \alpha \in [0,1]$

Let $z_\alpha \triangleq \alpha z^*(x_1) + (1-\alpha)z^*(x_2)$. Vector z_α satisfies the constraints

$$\begin{aligned} Gz_\alpha &= \alpha Gz^*(x_1) + (1-\alpha)Gz^*(x_2) \\ &\leq \alpha(W + Sx_1) + (1-\alpha)(W + Sx_2) = W + Sx_\alpha \end{aligned}$$

Therefore $x_\alpha \in X^*,\ \forall x_1, x_2 \in X^*,\ \forall \alpha \in [0,1]$

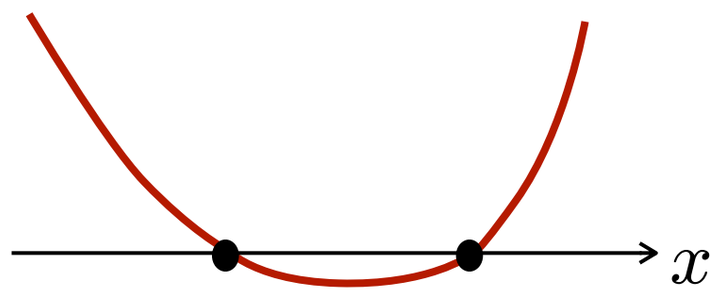
Note: X^* is the projection of a polyhedron onto the parameter space

$$X^* = \text{Proj}_x \left\{ \begin{bmatrix} z \\ x \end{bmatrix} : \ \begin{bmatrix} G & -S \end{bmatrix} \begin{bmatrix} z \\ x \end{bmatrix} \leq W \right\}$$

hence X^* is a polyhedron, and therefore convex.

PROOF OF THEOREM

2) V^* **is a convex function of** x



Since z_α satisfies the constraints

$$Gz_\alpha \leq W + Sx_\alpha$$

by optimality of $V^*(x_\alpha)$ and convexity of $J(z,x) = \frac{1}{2}\left[\begin{smallmatrix} z \\ x \end{smallmatrix}\right]' \left[\begin{smallmatrix} H & F \\ F' & Y \end{smallmatrix}\right] \left[\begin{smallmatrix} z \\ x \end{smallmatrix}\right]$

$$\begin{aligned} V^*(x_\alpha) &\leq J(z_\alpha, x_\alpha) = J(\alpha z^*(x_1) + (1-\alpha) z^*(x_2), \alpha x_1 + (1-\alpha) x_2) \\ &\leq \alpha J(z^*(x_1), x_1) + (1-\alpha) J(z^*(x_2), x_2) \\ &= \alpha V^*(x_1) + (1-\alpha) V^*(x_2) \end{aligned}$$

PROOF OF THEOREM

3) **Continuity of z^* and V^* with respect to x**

Let $z^*(x) = L_i x + M_i$ when $x \in CR_i$. z^* is linear and therefore continuous on the interior of each critical region CR_i.

Consider a parameter x on the boundary between two regions, $x \in CR_i \cap CR_j$. By construction, both $L_i x + M_i$ and $L_j x + M_j$ satisfy the optimality conditions.

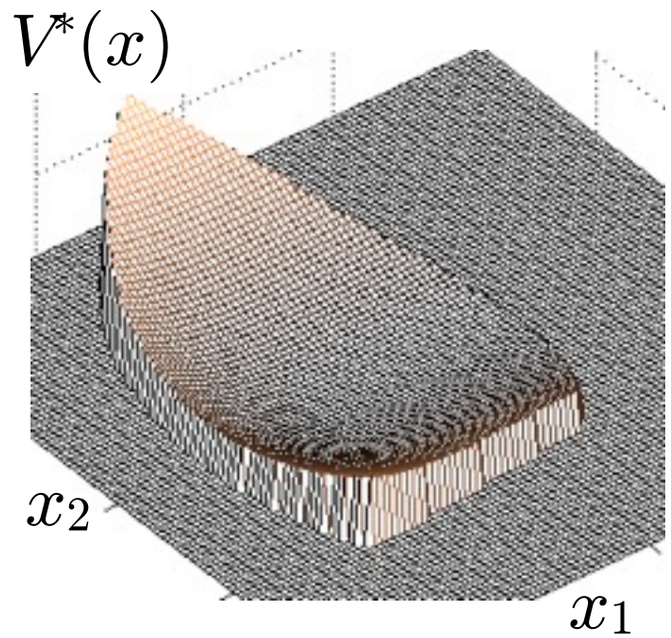
By strict convexity of the optimization problem ($H \succ 0$), the optimum is unique, so $L_i x + M_i = L_j x + M_j$, $\forall x \in CR_i \cap CR_j$.

This proves continuity of z^* across boundaries of critical regions.

As V^* is the composition of two continuous functions, $V^*(x) = J(z^*(x), x)$, it is also continuous. $\square$

MULTIPARAMETRIC CONVEX PROGRAMMING

$$\begin{aligned} \min_z \quad & f(z,x) \\ \text{s.t.} \quad & g_i(z,x) \le 0, \;\; i=1,\dots,p \\ & Az + Bz + d = 0 \end{aligned}$$



Lemma *Let f, g_i be jointly convex functions of (z,x) $(\forall i = 1,\dots,p)$. Then X^* is a convex set and $V^\star$ is a convex function of x.*

If f,g_i are also **continuous** in (z,x) then $V^*(x)$ is also continuous in x

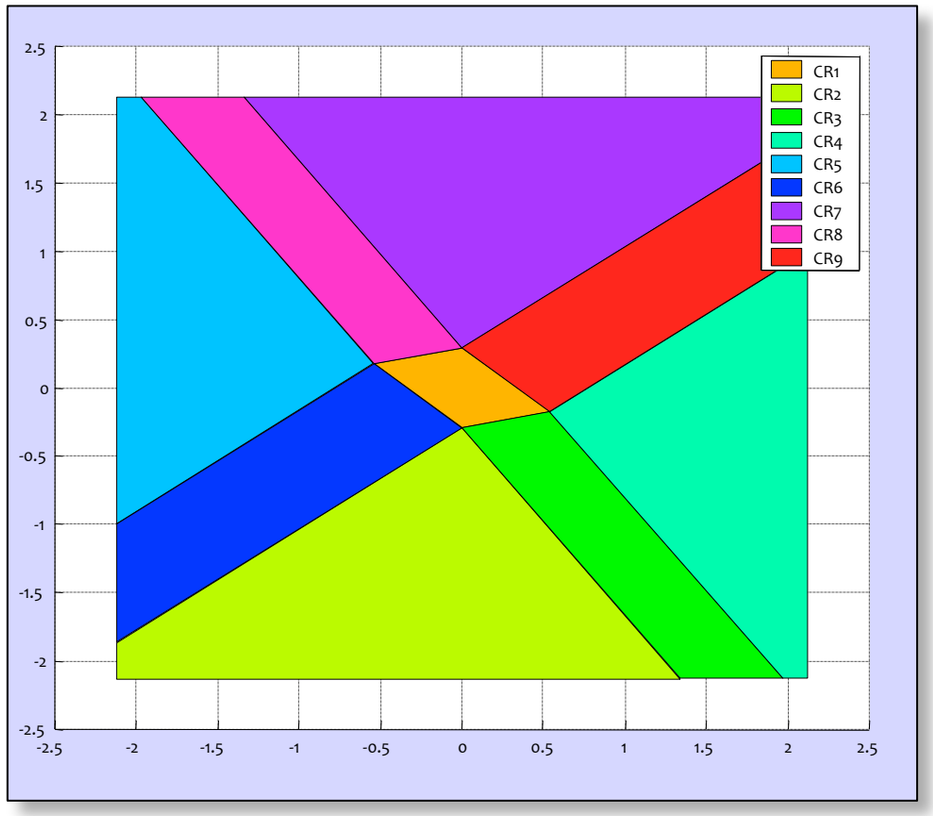
(Mangasarian, Rosen, 1964)

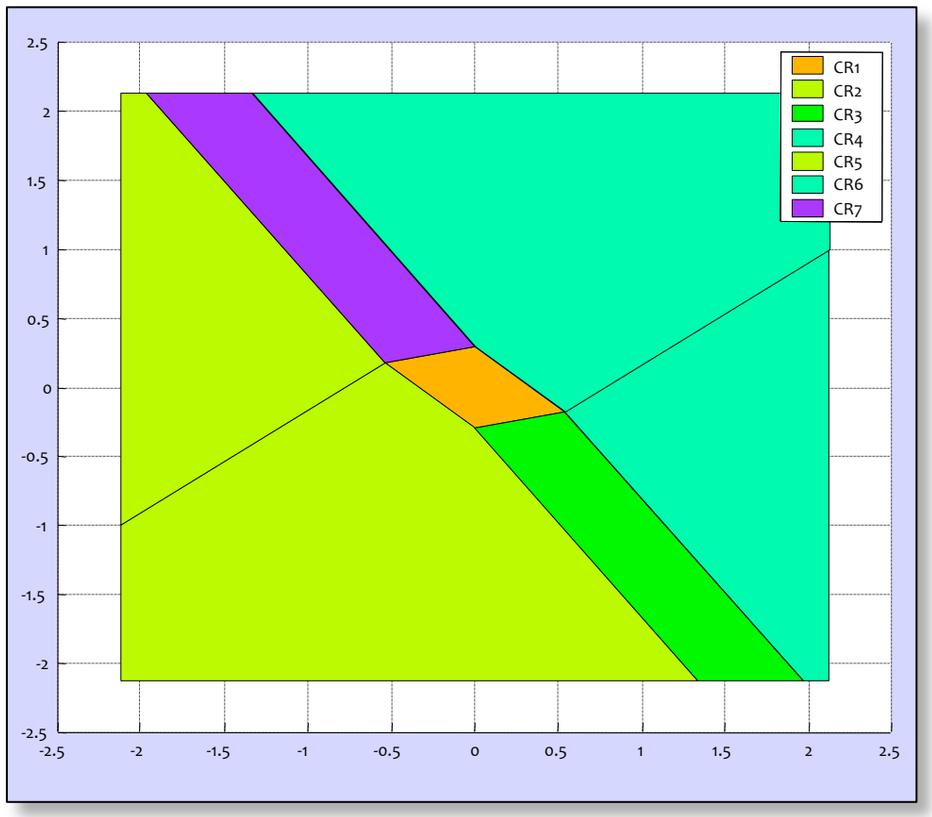
h, g convex and continuous in $(z,x) \Rightarrow V^*(x)$ convex and continuous

V^* and X^* may not be easy to express analytically. Approximate solutions possible

(Bemporad, Filippi, 2003)

COMPLEXITY REDUCTION





$$z(x) \triangleq \left[u'_0(x) \quad u'_1(x) \quad \dots \quad u'_{N-1}(x) \right]$$

Regions where the first component of the solution is the same can be joined (when their union is convex).

(Bemporad, Fukuda, Torrisi, *Computational Geometry*, 2001)

Optimal merging methods exist (Geyer, Torrisi, Morari, 2008)

A NEW MPQP ALGORITHM BASED ON NNLS

(Bemporad, IEEE TAC 2015)

- **Partially Nonnegative Least Squares (PNNLS):**

$$\begin{aligned}\min_{v,u} \quad & \|Av + Bu - c\|_2^2 \\ \text{s.t.} \quad & v \geq 0, \ \ u \text{ free}\end{aligned}$$

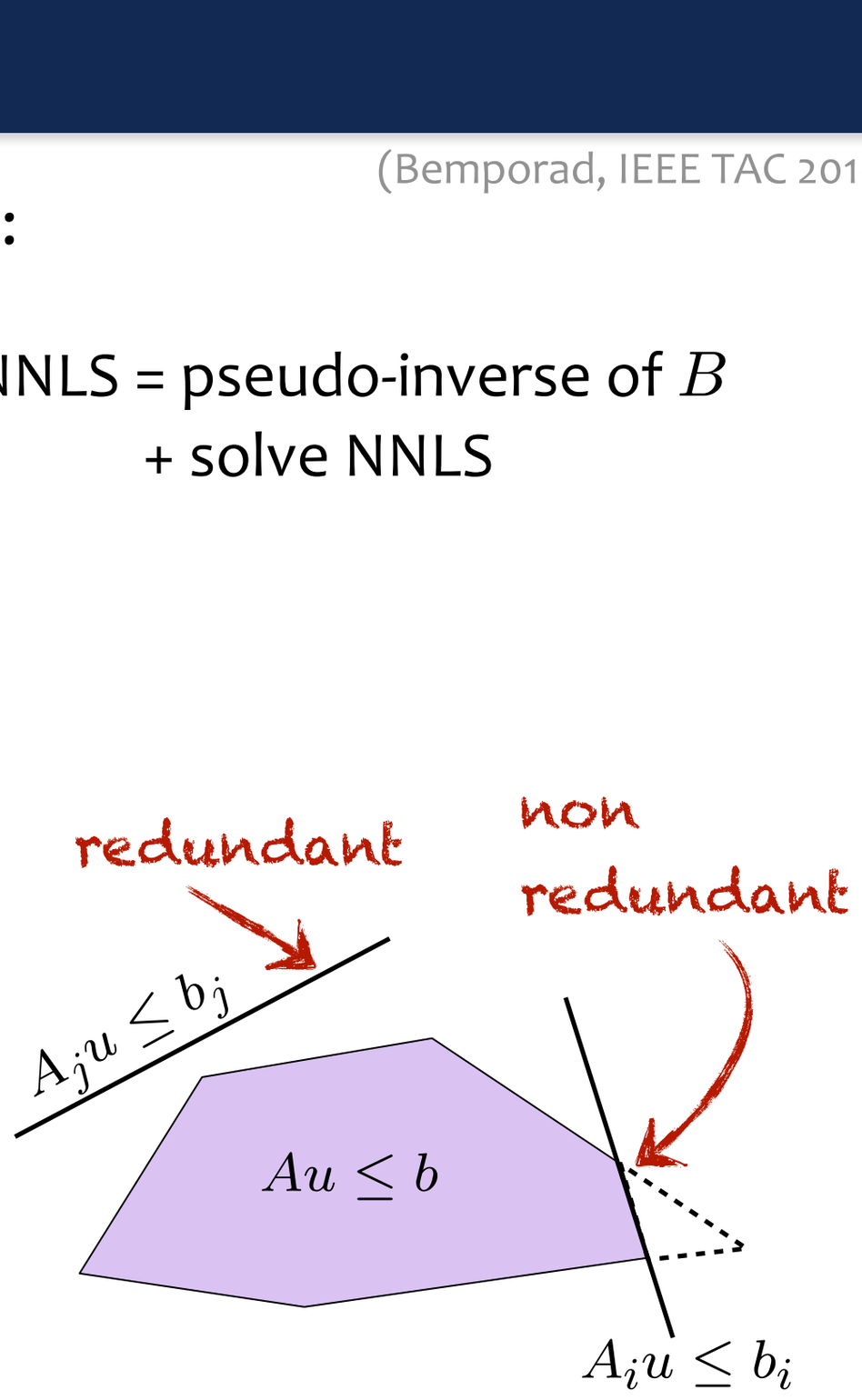
PNNLS = pseudo-inverse of B
+ solve NNLS

- Key result:

A polyhedron $P = \{u \in \mathbb{R}^n : \ \ Au \leq b\}$
is nonempty iff the PNNLS problem

$$\begin{aligned}(v^*, u^*) = \arg\min_{v,u} \quad & \|v + Au - b\|_2^2 \\ \text{s.t.} \quad & v \geq 0, u \text{ free}\end{aligned}$$

has zero residual $\|v^* + Au^* - b\|_2^2 = 0$

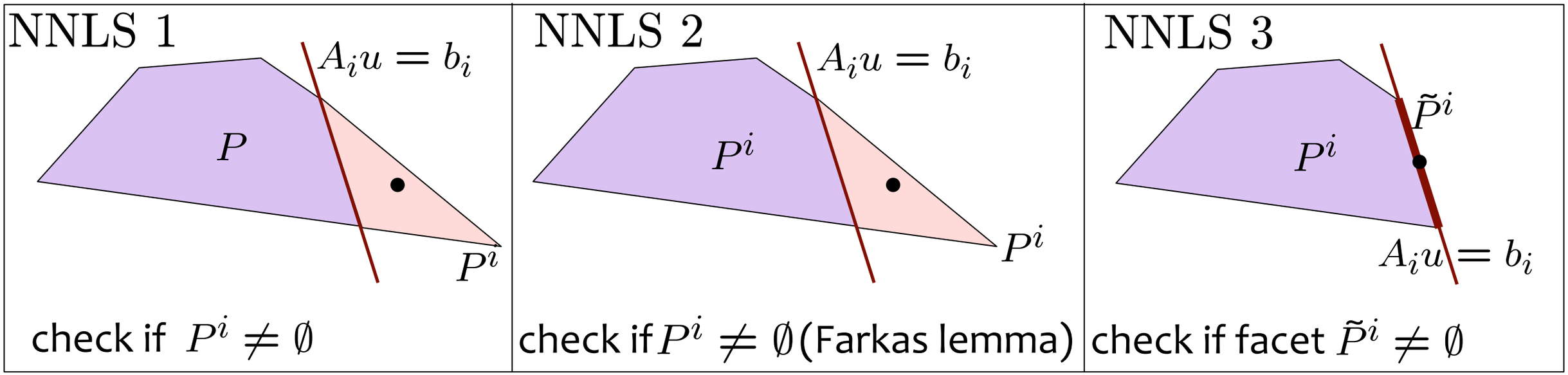


- Many other polyhedral operations can be also tackled by NNLS

POLYHEDRAL COMPUTATIONS BASED ON NNLS

(Bemporad, IEEE TAC 2015)

- Three possible methods to check inequality redundancy via NNLS:



- Numerical results

m	NNLS 1	NNLS 2	NNLS 3	LP
2	0.0051	0.0035	0.0006	0.0046
4	0.0351	0.0301	0.0019	0.0103
6	0.1297	0.1233	0.0038	0.0193
8	0.4046	0.3976	0.0071	0.0340
10	1.0185	1.0082	0.0111	0.0554
12	2.0665	2.1030	0.0178	0.0955
14	4.5279	4.5546	0.0263	0.1426
16	7.9159	7.9133	0.0357	0.1959

NNLS = compiled Embedded MATLAB
LP = compiled C code (GLPK)

CPU time = seconds (this Mac)

random polyhedra of $\mathbb{R}^m$ with $10m$ inequalities

- Other polyhedral operations can be also tackled by NNLS
  (full-dimensionality check, Chebychev radius, union, projection)

A NEW MPQP ALGORITHM BASED ON NNLS

- New mpQP algorithm based on **NNLS** + **dual QP formulation** to compute active sets and deal with degeneracy (Bemporad, IEEE TAC, 2015)

- Comparison with:
  - Hybrid Toolbox (Bemporad, 2003)
  - Multiparametric Toolbox 2.6 (with default opts) (Kvasnica, Grieder, Baotic, 2006)

- Included in MPC Toolbox 5.0 (R2014b)

The MathWorks (Bemporad, Morari, Ricker, 1998-2015)

q	m	Hybrid Tbx	MPT	NNLS
4	2	0.0174	0.0256	0.0026
4	3	0.0263	0.0356	0.0038
4	4	0.0432	0.0559	0.0061
4	5	0.0650	0.0850	0.0097
4	6	0.0827	0.1105	0.0126
8	2	0.0347	0.0396	0.0050
8	3	0.0583	0.0680	0.0092
8	4	0.0916	0.0999	0.0140
8	5	0.1869	0.2147	0.0322
8	6	0.3177	0.3611	0.0586
12	2	0.0398	0.0387	0.0054
12	3	0.1121	0.1158	0.0191
12	4	0.2067	0.2001	0.0352
12	5	0.6180	0.6428	0.1151
12	6	1.2453	1.3601	0.2426
20	2	0.1029	0.0763	0.0152
20	3	0.3698	0.2905	0.0588
20	4	0.9069	0.7100	0.1617
20	5	2.2978	1.9761	0.4395
20	6	6.1220	6.2518	1.2853

DOUBLE INTEGRATOR EXAMPLE

- System: $y(t) = \frac{1}{s^2}u(t)$

sampling + ZOH
T_s=1 s

$$\begin{aligned} x(t+1) &= \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} x(t) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(t) \\ y(t) &= \begin{bmatrix} 1 & 0 \end{bmatrix} x(t) \end{aligned}$$

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

- Control objective: $\min \sum_{k=0}^{\infty} y_k^2 + \frac{1}{100} u_k^2$

LQ gain

$$u_k = K_{LQ} x_k, \ \forall k \ge N_u$$

$$N_u = N = 2$$

$$\Rightarrow \min \left(\sum_{k=0}^{1} y_k^2 + \frac{1}{100} u_k^2 \right) + x_2' \begin{bmatrix} 2.1429 & 1.2246 \\ 1.2246 & 1.3996 \end{bmatrix} x_2$$

solution of algebraic Riccati equation

- Optimization problem

$$H = \begin{bmatrix} 0.8365 & 0.3603 \\ 0.3603 & 0.2059 \end{bmatrix}, \ F = \begin{bmatrix} 0.4624 & 1.2852 \\ 0.1682 & 0.5285 \end{bmatrix}$$

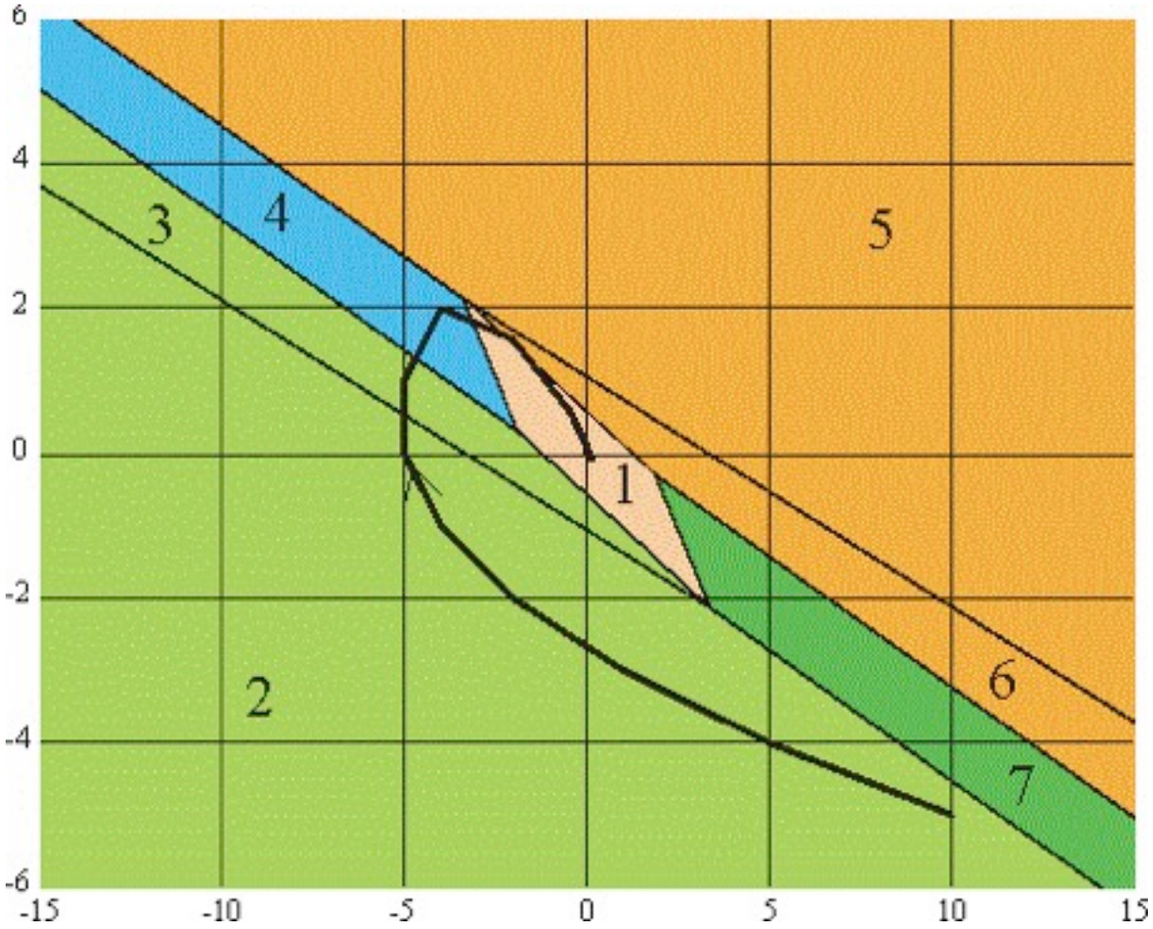
$$G = \begin{bmatrix} 1 & 0 \\ -1 & 0 \\ 0 & 1 \\ 0 & -1 \end{bmatrix}, \ W = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}, \ S = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}$$

(cost function is normalized by max svd(H))

DOUBLE INTEGRATOR EXAMPLE - MP-QP SOLUTION

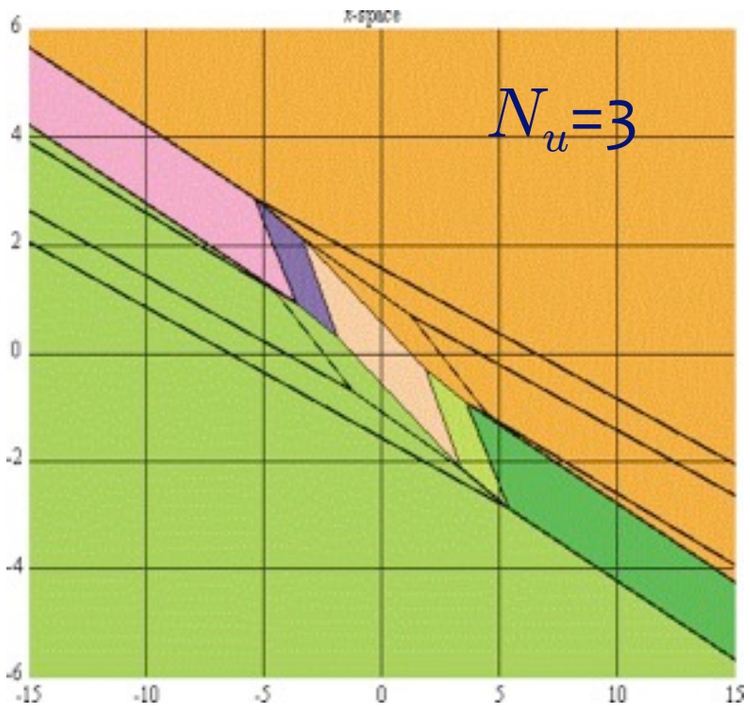
N_u=2

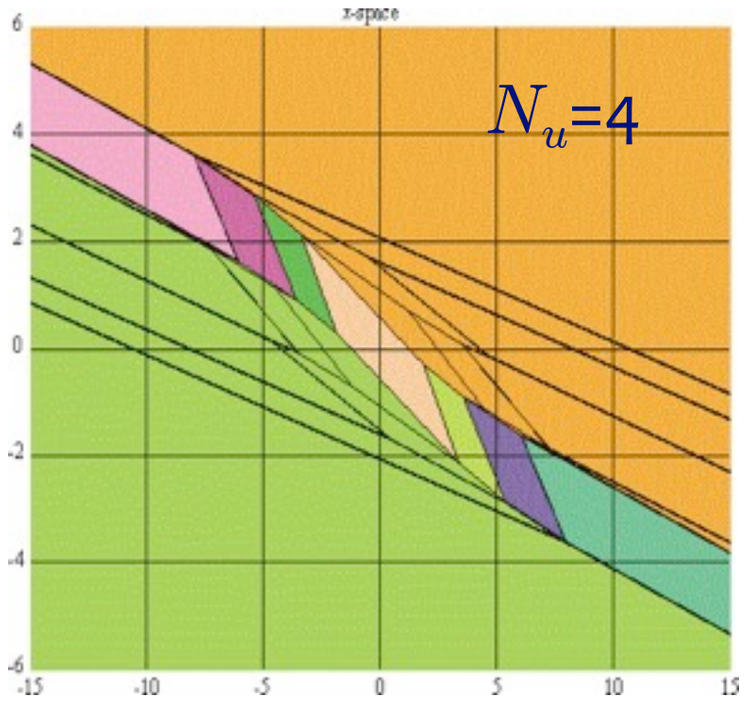
$$u(x) = \begin{cases} \begin{bmatrix} -0.8166 & -1.7499 \end{bmatrix} x & \text{if } \begin{bmatrix} -0.8166 & -1.7499 \\ 0.8166 & 1.7499 \\ 0.6124 & 0.4957 \\ -0.6124 & -0.4957 \end{bmatrix} x \le \begin{bmatrix} 1.0000 \\ 1.0000 \\ 1.0000 \\ 1.0000 \end{bmatrix} & \text{(Region \#1)} \\ 1.0000 & \text{if } \begin{bmatrix} 0.3864 & 1.0738 \\ 0.2970 & 0.9333 \end{bmatrix} x \le \begin{bmatrix} -1.0000 \\ -1.0000 \end{bmatrix} & \text{(Region \#2)} \\ 1.0000 & \text{if } \begin{bmatrix} 0.9712 & 2.6991 \\ -0.2970 & -0.9333 \\ 0.8166 & 1.7499 \end{bmatrix} x \le \begin{bmatrix} -1.0000 \\ 1.0000 \\ -1.0000 \end{bmatrix} & \text{(Region \#3)} \\ \begin{bmatrix} -0.5528 & -1.5364 \end{bmatrix} x + 0.4308 & \text{if } \begin{bmatrix} -0.9712 & -2.6991 \\ 0.3864 & 1.0738 \\ 0.6124 & 0.4957 \end{bmatrix} x \le \begin{bmatrix} 1.0000 \\ 1.0000 \\ -1.0000 \end{bmatrix} & \text{(Region \#4)} \\ -1.0000 & \text{if } \begin{bmatrix} -0.3864 & -1.0738 \\ -0.2970 & -0.9333 \end{bmatrix} x \le \begin{bmatrix} -1.0000 \\ -1.0000 \end{bmatrix} & \text{(Region \#5)} \\ -1.0000 & \text{if } \begin{bmatrix} -0.9712 & -2.6991 \\ 0.2970 & 0.9333 \\ -0.8166 & -1.7499 \end{bmatrix} x \le \begin{bmatrix} -1.0000 \\ 1.0000 \\ -1.0000 \end{bmatrix} & \text{(Region \#6)} \\ \begin{bmatrix} -0.5528 & -1.5364 \end{bmatrix} x - 0.4308 & \text{if } \begin{bmatrix} -0.3864 & -1.0738 \\ 0.9712 & 2.6991 \\ -0.6124 & -0.4957 \end{bmatrix} x \le \begin{bmatrix} 1.0000 \\ 1.0000 \\ -1.0000 \end{bmatrix} & \text{(Region \#7)} \end{cases}$$

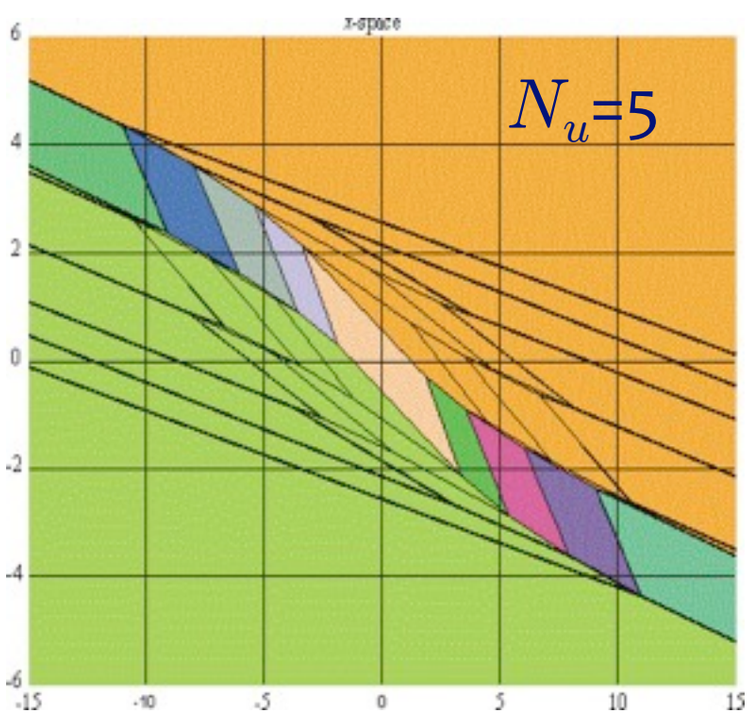


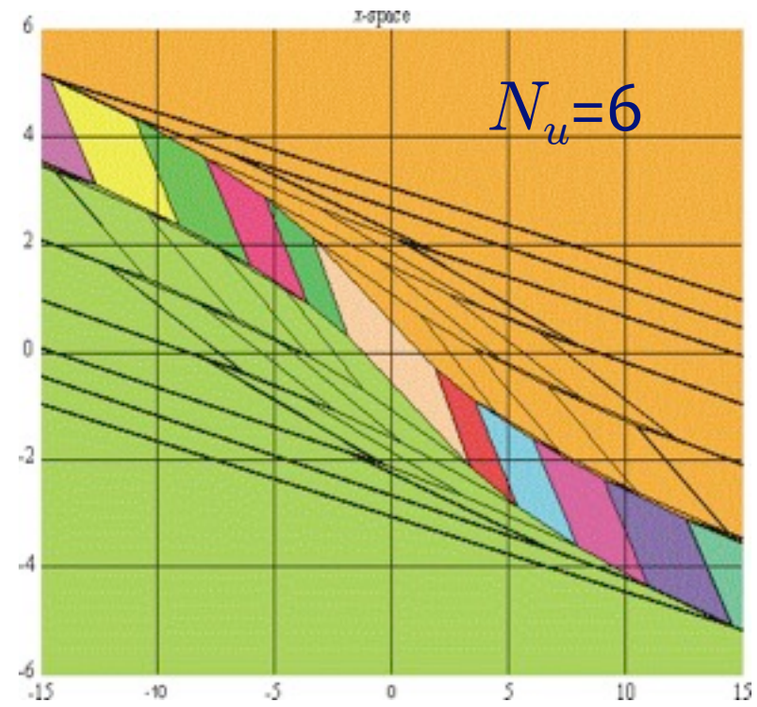
go to demo **/demos/linear/doubleintexp.m** (Hyb-Tbx)

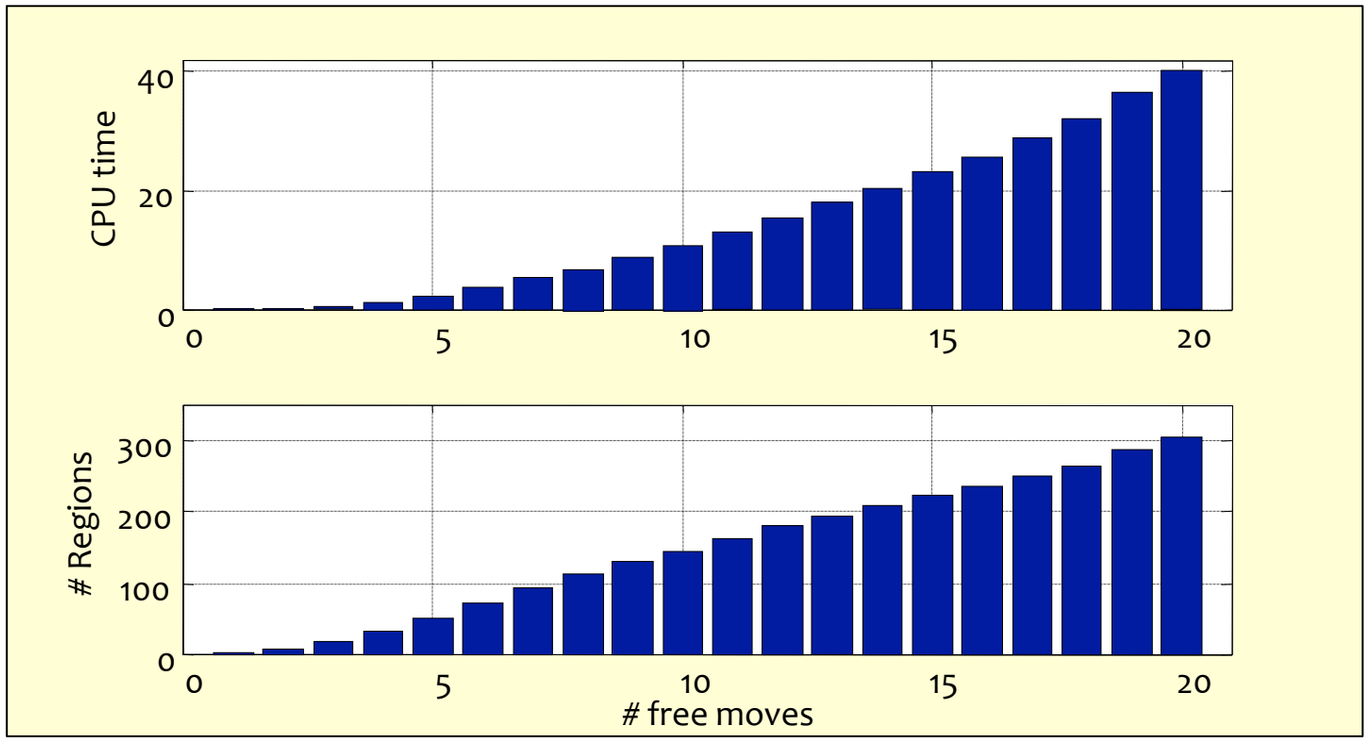
DOUBLE INTEGRATOR EXAMPLE - COMPLEXITY OF SOLUTION











(is the number of regions finite for $N_u \to \infty$?)

HYBRID TOOLBOX FOR MATLAB

(Bemporad, 2003-present)

Features:

- **Hybrid models**: design, simulation, verification
- **MPC control** design for linear systems w/ constraints and hybrid systems
- **Explicit linear and hybrid MPC** (multi-parametric programming)
- **Simulink** library
- **C-code** generation of explicit MPC controllers
- Interfaces to several QP/LP and Mixed-Integer Programming **solvers**
- **Polyhedral computation** functions

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

~6800 downloads
(since Jan 18, 2004)

~1.5 downloads/day

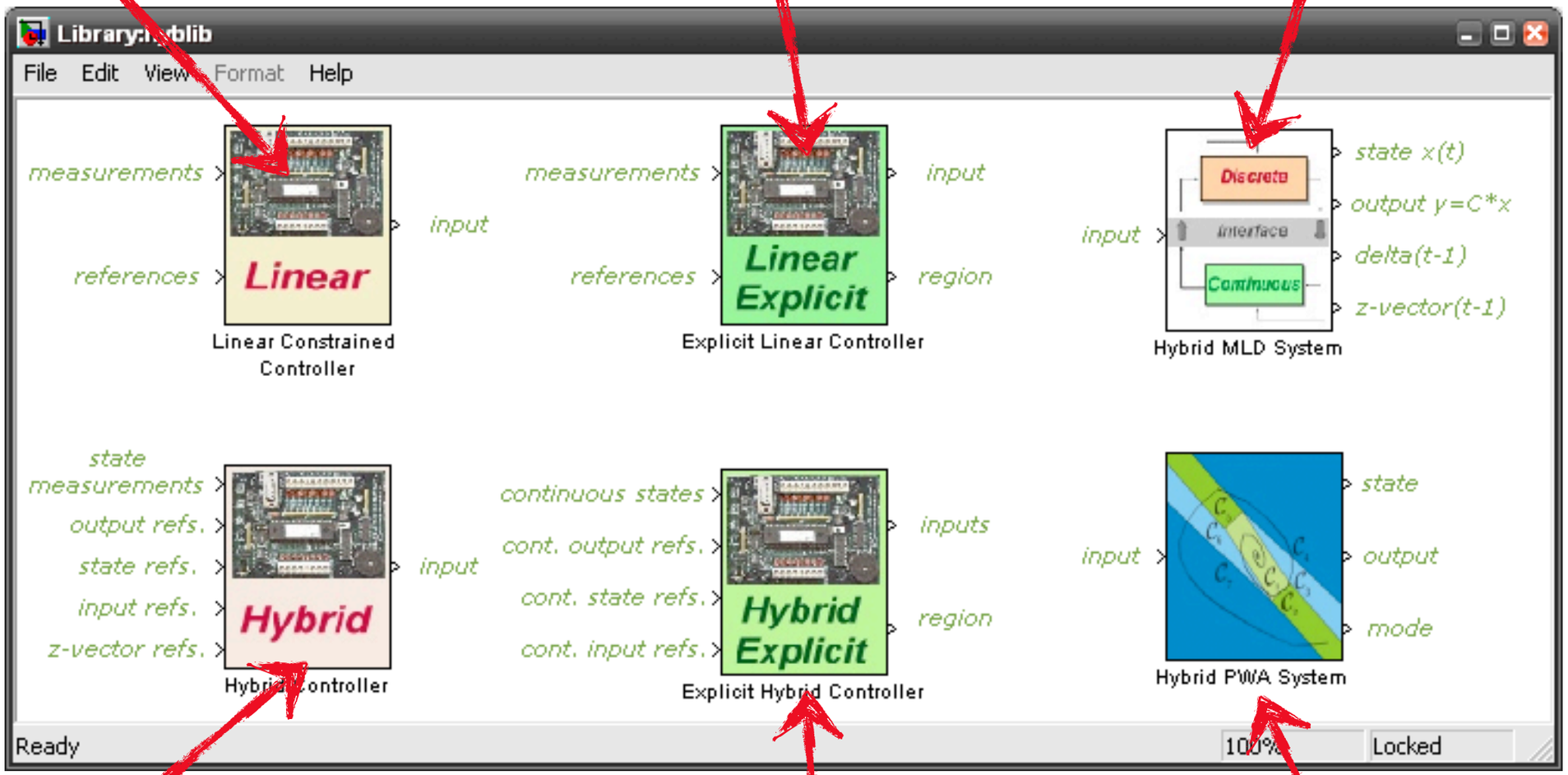
Initially supported by Ford

HYBRID TOOLBOX - SIMULINK LIBRARY

linear MPC based on on-line QP (M-code)

explicit linear MPC (C-code)

MLD dynamics (M-code)



hybrid MPC based on on-line MIP (M-code)

explicit hybrid MPC (C-code)

PWA dynamics (M-code)

DOUBLE INTEGRATOR EXAMPLE - HYBRID TOOLBOX

```
Ts=1; % sampling time
model=ss([1 1;0 1],[0;1],[0 1],0,Ts); % prediction model

limits.umin=-1; limits.umax=1; % input constraints

interval.Nu=2; % control horizon
interval.N=2;  % prediction horizon

weights.R=.1;
weights.Q=[1 0;0 0];
weights.P='lqr';  % terminal weight = Riccati matrix
weights.rho=+Inf; % hard constraints on outputs, if present

Cimp=lincon(model,'reg',weights,interval,limits); % MPC

range=struct('xmin',[-15 -15],'xmax',[15 15]);
Cexp=expcon(Cimp,range); % Explicit MPC

x0=[10,-.3]';
Tstop=40; %Simulation time
[X,U,T,Y,I]=sim(Cexp,model,[],x0,Tstop);
```

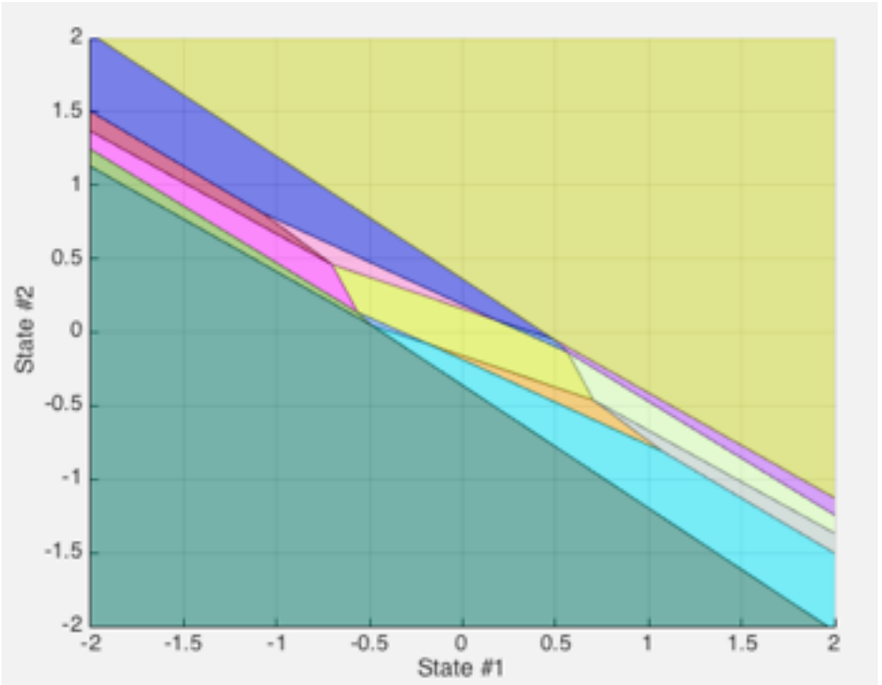
MPC TOOLBOX 5.0

(Bemporad, Morari, Ricker, 2014)

- New explicitMPC object

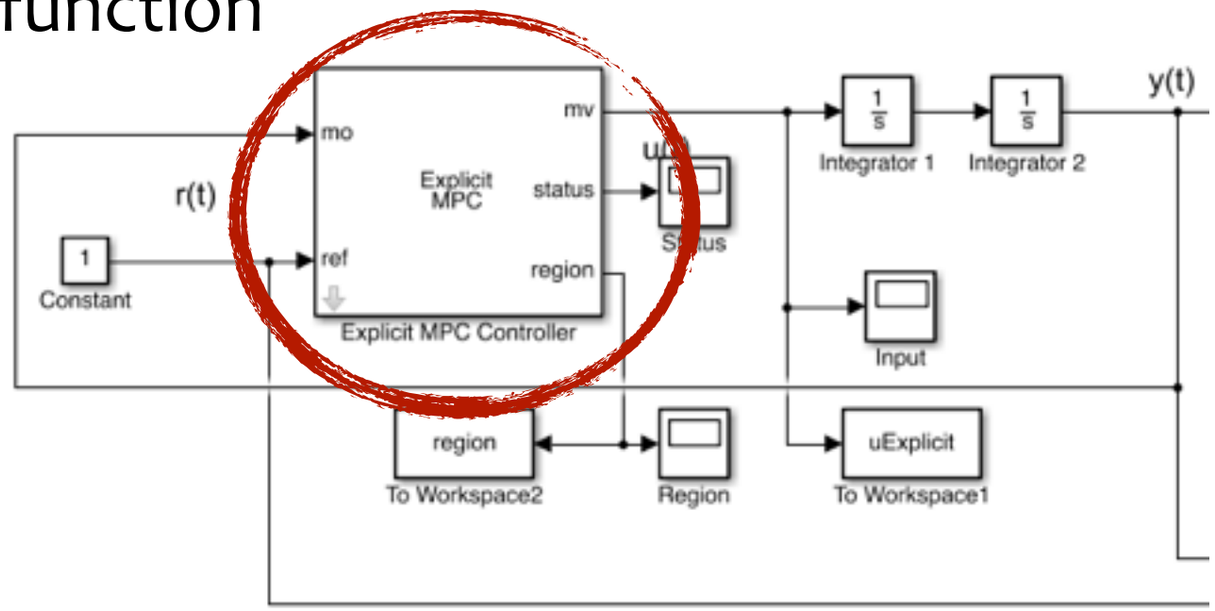
The MathWorks

```
>> mpcobj = mpc(plant, Ts, p, m);

>> empcobj = generateExplicitMPC(mpcobj, range);

>> empcobj2 = simplify(empcobj, 'exact')

>> [y2,t2,u2] = sim(empcobj,Tf,ref);

>> u = mpcmoveExplicit(empcobj,xmpc,y,ref);
```



- Very simple on-line PWA evaluation function

```
i=0; imin=0; vmin=Inf; flag=0;
while ~found && i<nr,
   i=i+1;
   v=max(pwafun(i).H*th-pwafun(i).K);
   if v<=0,
      found=true; flag=1;
   else
      if vmin>v,
         vmin=v; imin=i;
      end
   end
end
x=pwafun(imin).F*th+pwafun(imin).G;
```



Simulink block

APPLICABILITY OF EXPLICIT MPC APPROACH

- Consider the following general MPC formulation

$$
\begin{aligned}
\min_{z} \quad & \sum_{k=0}^{N-1} \frac{1}{2}(y_k - r(t+k))' S (y_k - r(t+k)) + \frac{1}{2}\Delta u_k' T \Delta u_k \\
& + (u_k - u_r(t+k))' V (u_k - u_r(t+k))' + \rho_\epsilon \epsilon^2 \\
\text{subj. to} \quad & x_{k+1} = Ax_k + Bu_k + B_v v(t+k), \;\; k = 0, \ldots, N-1 \\
& y_k = Cx_k + Du_k + D_v v(t+k), \;\; k = 0, \ldots, N-1 \\
& u_{\min}(t+k) \le u_k \le u_{\max}(t+k), \;\; k = 0, \ldots, N-1 \\
& \Delta u_{\min}(t+k) \le \Delta u_k \le \Delta u_{\max}(t+k), \;\; k = 0, \ldots, N-1 \\
& y_{\min}(t+k) - \epsilon V_{\min} \le y_k \le y_{\max}(t+k) + \epsilon V_{\max}, \;\; k = 1, \ldots, N \\
& x_0 = x(t)
\end{aligned}
$$

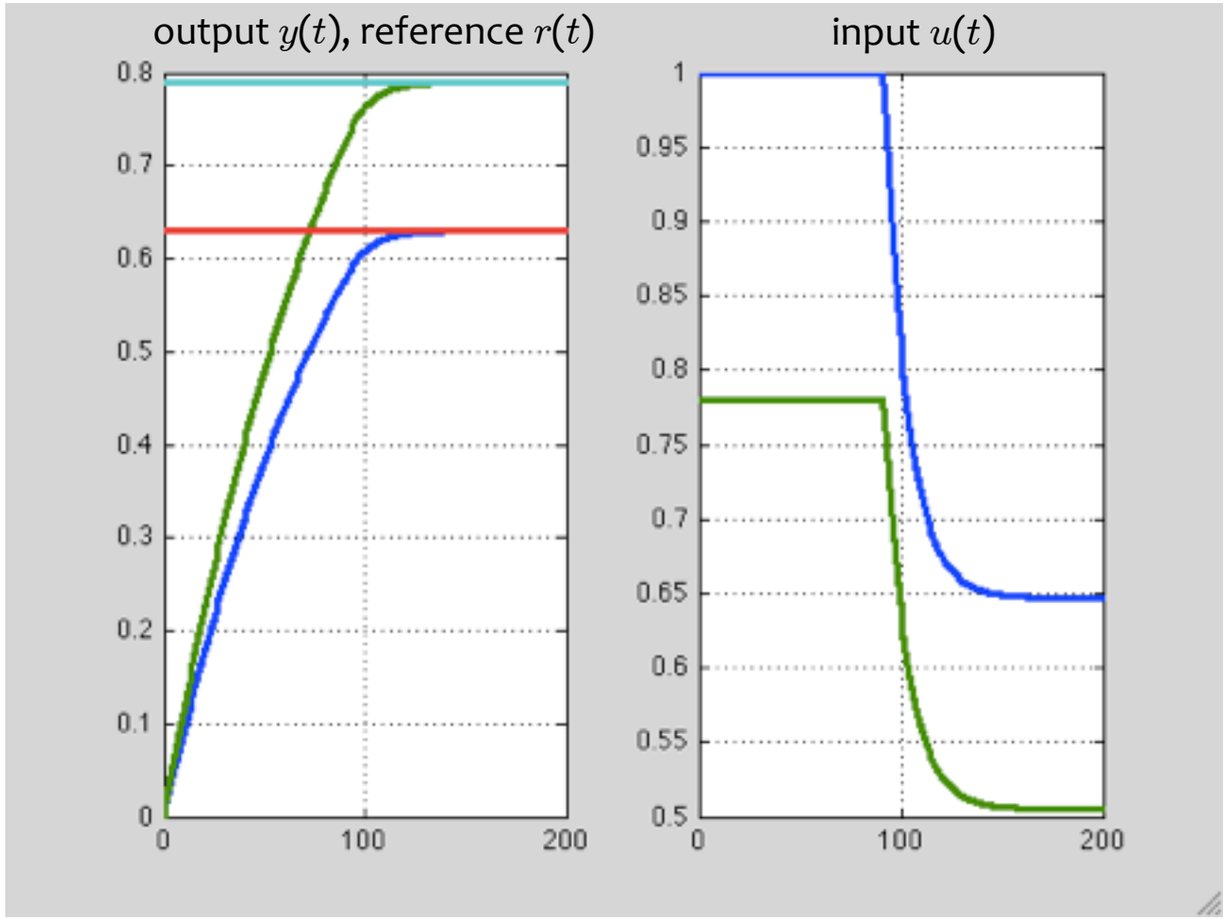
- Everything marked in red can be **time-varying** in explicit MPC

- Not applicable to time-varying problems (weights and/or system matrices)

REFERENCE TRACKING, MIMO SYSTEM

- System: $y(t) = \dfrac{10}{100s+1}\begin{bmatrix} 4 & -5 \\ -3 & 4 \end{bmatrix} u(t)$ sampling + ZOH (T_s=1 s)
- Constraints: $\begin{bmatrix} -1 \\ -1 \end{bmatrix} \le u(t) \le \begin{bmatrix} 1 \\ 1 \end{bmatrix}$
- Control objective: $\min \sum_{k=0}^{19} \|y_k - r(t)\|^2 + \frac{1}{10}\|\Delta u_k\|^2$ N=20

  $u_k \equiv u_0, \ \forall k \ge 1$ N_u=1



go to demo
`linear/mimo.m`

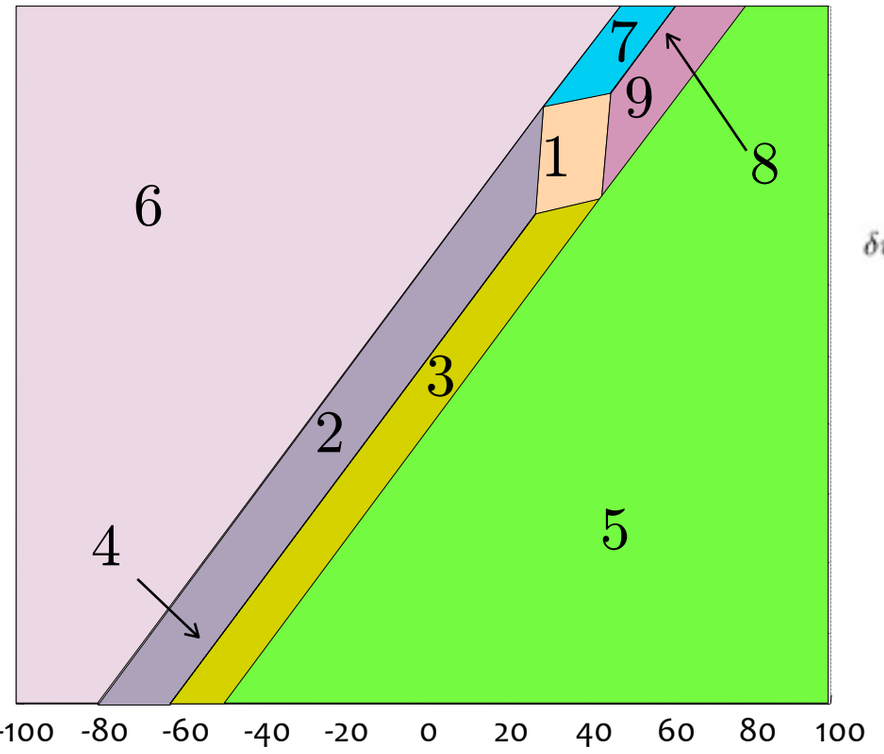
(Hyb-Tbx)

REFERENCE TRACKING, MIMO SYSTEM

MPC law

state-space partition

x-space



Polyhedral partition of the state-space for $u(t\text{-}1)$=[0 0]' and $r(t)$=[0.63 0.79]'

$$\delta u = \begin{cases}
\begin{bmatrix} -0.1251 & 0.0084 \\ 0.0168 & -0.0668 \end{bmatrix} x + \begin{bmatrix} -0.8535 & 0.1143 \\ 0.1143 & -0.9107 \end{bmatrix} u + \begin{bmatrix} 2.5583 & 3.1741 \\ 1.8949 & 2.5583 \end{bmatrix} r & \text{if } \begin{bmatrix} -0.1251 & 0.0084 \\ 0.0168 & -0.0668 \\ 0.1251 & -0.0084 \\ -0.0168 & 0.0668 \end{bmatrix} x + \begin{bmatrix} 0.1465 & 0.1143 \\ 0.1143 & 0.0893 \\ -0.1465 & -0.1143 \\ -0.1143 & -0.0893 \end{bmatrix} u + \begin{bmatrix} 2.5583 & 3.1741 \\ 1.8949 & 2.5583 \\ -2.5583 & -3.1741 \\ -1.8949 & -2.5583 \end{bmatrix} r \le \begin{bmatrix} 1.0000 \\ 1.0000 \\ 1.0000 \\ 1.0000 \end{bmatrix} \quad \text{(Region \#1)} \\
\begin{bmatrix} 0.0000 & 0.0000 \\ 0.1144 & -0.0733 \end{bmatrix} x + \begin{bmatrix} -1.0000 & 0.0000 \\ -0.0000 & -0.9999 \end{bmatrix} u + \begin{bmatrix} 0.0000 & 0.0000 \\ -0.1016 & 0.0813 \end{bmatrix} r + \begin{bmatrix} 1.0000 \\ 0.7804 \end{bmatrix} & \text{if } \begin{bmatrix} 0.5210 & -0.3337 \\ -0.0643 & 0.0412 \\ 0.1251 & -0.0084 \end{bmatrix} x + \begin{bmatrix} -0.0000 & 0.0005 \\ 0.0000 & -0.0001 \\ -0.1465 & -0.1143 \end{bmatrix} u + \begin{bmatrix} -0.4625 & 0.3700 \\ 0.0570 & -0.0456 \\ -2.5583 & -3.1741 \end{bmatrix} r \le \begin{bmatrix} 1.0000 \\ 1.0000 \\ -1.0000 \end{bmatrix} \quad \text{(Region \#2)} \\
\begin{bmatrix} -0.1466 & 0.0938 \\ 0.0000 & 0.0000 \end{bmatrix} x + \begin{bmatrix} -0.9998 & -0.0000 \\ 0.0000 & -1.0000 \end{bmatrix} u + \begin{bmatrix} 0.1333 & -0.0999 \\ -0.0000 & 0.0000 \end{bmatrix} r + \begin{bmatrix} 1.2798 \\ 1.0000 \end{bmatrix} & \text{if } \begin{bmatrix} -0.5239 & 0.3353 \\ 0.0643 & -0.0411 \\ -0.0168 & 0.0668 \end{bmatrix} x + \begin{bmatrix} 0.0007 & -0.0000 \\ -0.0001 & 0.0000 \\ -0.1143 & -0.0893 \end{bmatrix} u + \begin{bmatrix} 0.4763 & -0.3572 \\ -0.0584 & 0.0438 \\ -1.8949 & -2.5583 \end{bmatrix} r \le \begin{bmatrix} -1.0000 \\ 1.0000 \\ -1.0000 \end{bmatrix} \quad \text{(Region \#3)} \\
\begin{bmatrix} -1.0000 & 0.0000 \\ 0.0000 & -1.0000 \end{bmatrix} u + \begin{bmatrix} 1.0000 \\ 1.0000 \end{bmatrix} & \text{if } \begin{bmatrix} 0.5239 & -0.3353 \\ -0.5210 & 0.3337 \end{bmatrix} x + \begin{bmatrix} -0.0007 & 0.0000 \\ 0.0000 & -0.0005 \end{bmatrix} u + \begin{bmatrix} -0.4763 & 0.3572 \\ 0.4625 & -0.3700 \end{bmatrix} r \le \begin{bmatrix} 1.0000 \\ -1.0000 \end{bmatrix} \quad \text{(Region \#4)} \\
\begin{bmatrix} -1.0000 & 0.0000 \\ -0.0000 & -1.0000 \end{bmatrix} u + \begin{bmatrix} -1.0000 \\ 1.0000 \end{bmatrix} & \text{if } \begin{bmatrix} -0.0643 & 0.0412 \\ -0.0643 & 0.0411 \end{bmatrix} x + \begin{bmatrix} 0.0000 & -0.0001 \\ 0.0001 & -0.0000 \end{bmatrix} u + \begin{bmatrix} 0.0570 & -0.0456 \\ 0.0584 & -0.0438 \end{bmatrix} r \le \begin{bmatrix} -1.0000 \\ -1.0000 \end{bmatrix} \quad \text{(Region \#5)} \\
\begin{bmatrix} -1.0000 & 0.0000 \\ 0.0000 & -1.0000 \end{bmatrix} u + \begin{bmatrix} 1.0000 \\ -1.0000 \end{bmatrix} & \text{if } \begin{bmatrix} 0.0643 & -0.0411 \\ 0.0643 & -0.0412 \end{bmatrix} x + \begin{bmatrix} -0.0001 & 0.0000 \\ -0.0000 & 0.0001 \end{bmatrix} u + \begin{bmatrix} -0.0584 & 0.0438 \\ -0.0570 & 0.0456 \end{bmatrix} r \le \begin{bmatrix} -1.0000 \\ -1.0000 \end{bmatrix} \quad \text{(Region \#6)} \\
\begin{bmatrix} 0.0000 & 0.0000 \\ 0.1144 & -0.0733 \end{bmatrix} x + \begin{bmatrix} -1.0000 & 0.0000 \\ -0.0000 & -0.9999 \end{bmatrix} u + \begin{bmatrix} 0.0000 & 0.0000 \\ -0.1016 & 0.0813 \end{bmatrix} r + \begin{bmatrix} -1.0000 \\ -0.7804 \end{bmatrix} & \text{if } \begin{bmatrix} 0.0643 & -0.0412 \\ -0.5210 & 0.3337 \\ -0.1251 & 0.0084 \end{bmatrix} x + \begin{bmatrix} -0.0000 & 0.0001 \\ 0.0000 & -0.0005 \\ 0.1465 & 0.1143 \end{bmatrix} u + \begin{bmatrix} -0.0570 & 0.0456 \\ 0.4625 & -0.3700 \\ 2.5583 & 3.1741 \end{bmatrix} r \le \begin{bmatrix} 1.0000 \\ 1.0000 \\ -1.0000 \end{bmatrix} \quad \text{(Region \#7)} \\
\begin{bmatrix} -0.1466 & 0.0938 \\ 0.0000 & 0.0000 \end{bmatrix} x + \begin{bmatrix} -0.9998 & -0.0000 \\ 0.0000 & -1.0000 \end{bmatrix} u + \begin{bmatrix} 0.1333 & -0.0999 \\ -0.0000 & 0.0000 \end{bmatrix} r + \begin{bmatrix} -1.2798 \\ -1.0000 \end{bmatrix} & \text{if } \begin{bmatrix} -0.0643 & 0.0411 \\ 0.5239 & -0.3353 \\ 0.0168 & -0.0668 \end{bmatrix} x + \begin{bmatrix} 0.0001 & -0.0000 \\ -0.0007 & 0.0000 \\ 0.1143 & 0.0893 \end{bmatrix} u + \begin{bmatrix} 0.0584 & -0.0438 \\ -0.4763 & 0.3572 \\ 1.8949 & 2.5583 \end{bmatrix} r \le \begin{bmatrix} 1.0000 \\ -1.0000 \\ -1.0000 \end{bmatrix} \quad \text{(Region \#8)} \\
\begin{bmatrix} -1.0000 & 0.0000 \\ 0.0000 & -1.0000 \end{bmatrix} u + \begin{bmatrix} -1.0000 \\ -1.0000 \end{bmatrix} & \text{if } \begin{bmatrix} -0.5239 & 0.3353 \\ 0.5210 & -0.3337 \end{bmatrix} x + \begin{bmatrix} 0.0007 & -0.0000 \\ -0.0000 & 0.0005 \end{bmatrix} u + \begin{bmatrix} 0.4763 & -0.3572 \\ -0.4625 & 0.3700 \end{bmatrix} r \le \begin{bmatrix} 1.0000 \\ -1.0000 \end{bmatrix} \quad \text{(Region \#9)}
\end{cases}$$

COMPLEXITY - QP VS. EXPLICIT

$2N$	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)

Average time on 100 random 3D parameters (2N constraints)

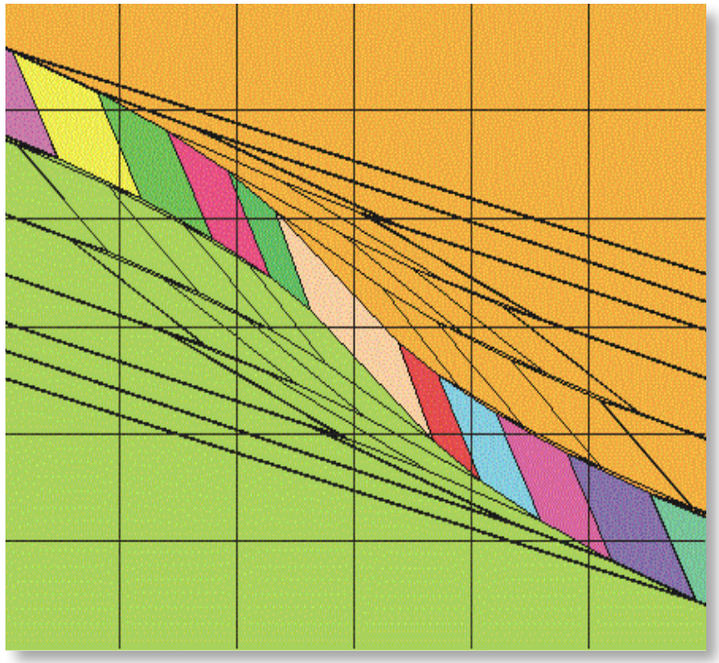
Worst-case time on 100 random 3D parameters (2N constraints)

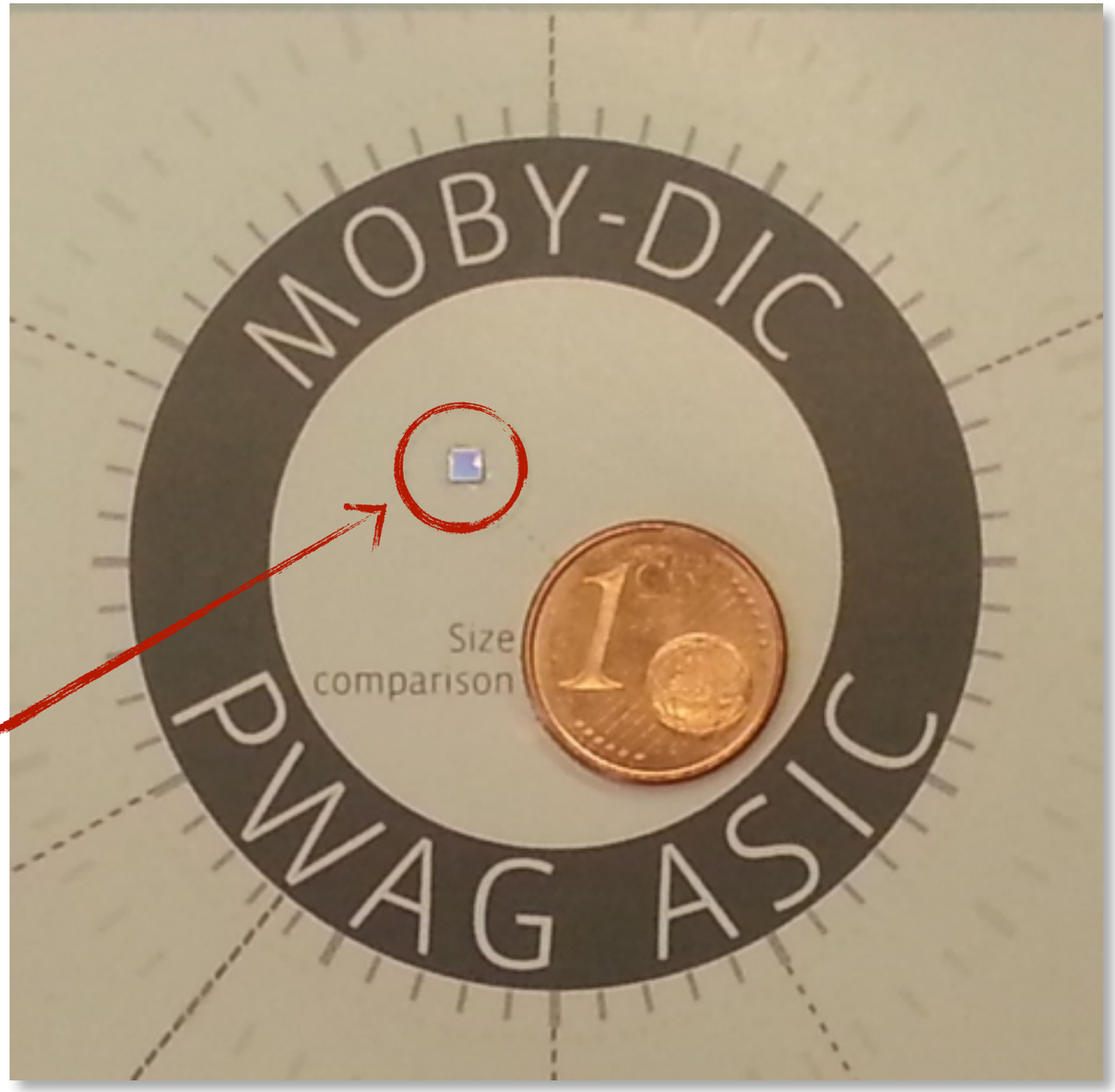
Explicit MPC typically practical for max 6-8 free control moves, 10-20 constraints, and 8-12 parameters (states+references)

⟹ Regions can be visited more efficiently than linear search

⟹ Number of regions can be usually reduced (e.g. suboptimal solutions)

HARDWARE (ASIC) IMPLEMENTATION OF EXPLICIT MPC





Technology: 90 nm, 9 metal-layer from Taiwan Semiconductor Manufacturing Company

Chip size: 1860x1860 µm²

Memory sizes: 24 KB (tree memory) ; 30 KB (parameter memory)

Power supply: 2.5V (ring); 1.2V (core)

Maximum frequency: 107.5 MHz (with two inputs)

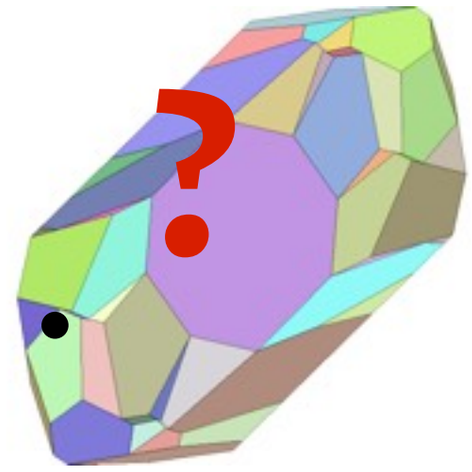
Power consumption: 38.08 mW@107.5MHz;

$$\frac{1}{107.5\ \text{MHz}} = 9.3\ \text{ns}$$

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

POINT LOCATION PROBLEM

In which region of the partition is $x(t)$?



- Store all regions and search linearly through them

- Exploit properties of mpLP solution to locate $x(t)$ from value function (also extended to mpQP) (Baotic, Borrelli, Bemporad, Morari 2008)

- Organize regions on a tree for logarithmic search (Tøndel, Johansen, Bemporad, 2003)

- For mpLP, recast as weighted nearest neighbour problem (logarithmic search) (Jones, Grieder, Rakovic, 2003)

- Exploit reachability analysis (Spjøtvold, Rakovic, Tøndel, Johansen, 2006) (Wang, Jones, Maciejowski, 2007)

- Use bounding boxes and trees (Christophersen, Kvasnica, Jones, Morari, 2007)

COMPLEXITY

- Number n_r of regions = number of combinations of active constraints at optimality

  - Mainly depends on number q of constraints: $n_r \leq \sum_{h=0}^{q} \binom{q}{h} = 2^q$

    (this is a worst-case estimate, most of the combinations are never optimal !)

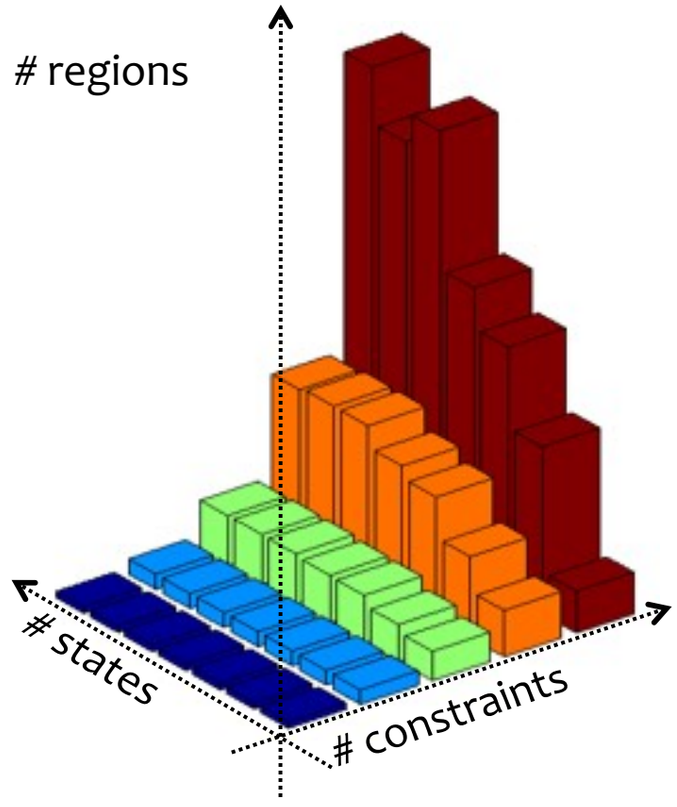
  - Also depends on #free variables

  - Weakly depends on #states

- Example

average on 20 random SISO systems (input saturation)

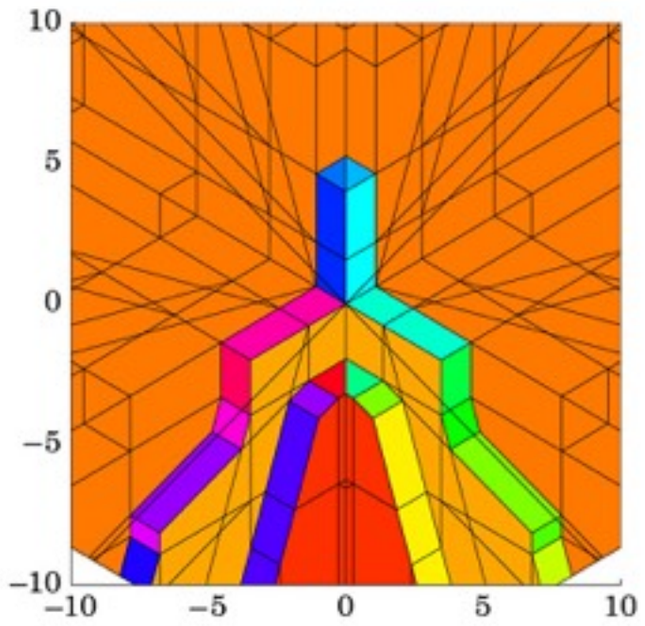
states\horizon	$N=1$	$N=2$	$N=3$	$N=4$	$N=5$
n=2	3	6.7	13.5	21.4	19.3
n=3	3	6.9	17	37.3	77
n=4	3	7	21.65	56	114.2
n=5	3	7	22	61.5	132.7
n=6	3	7	23.1	71.2	196.3
n=7	3	6.95	23.2	71.4	182.3
n=8	3	7	23	70.2	207.9



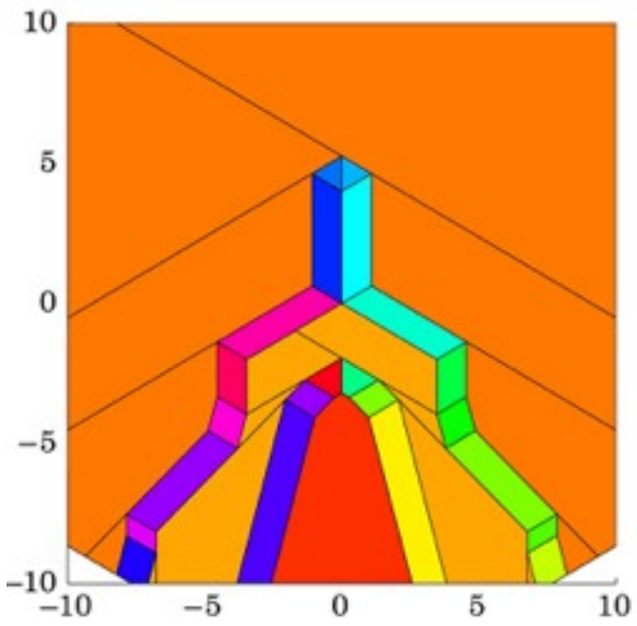
REGION REDUCTION AND APPROXIMATIONS

- Join regions more efficiently

  (Geyer, Torrisi, Morari, 2008)



(a) Original set of 252 polyhedra.



(c) Set of 39 polyhedra resulting from OCR.

- Change cost function (e.g. minimum time)

  (Grieder, Morari, 2003)

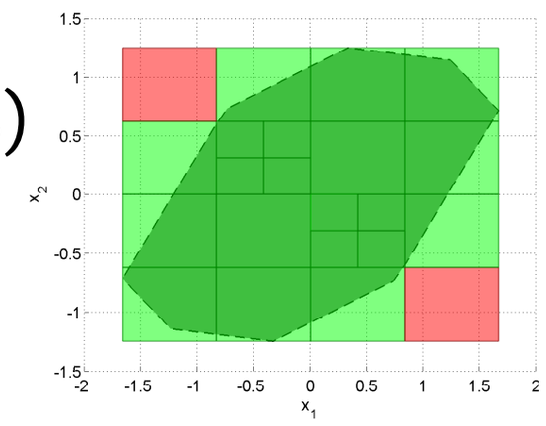
- Relax KKT conditions (suboptimal mpQP)

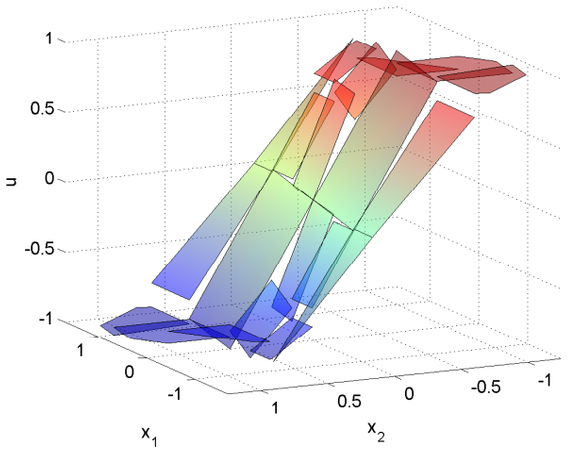
  (Bemporad, Filippi, 2003)

- Use orthogonal trees (suboptimal solutions)

  (Johansen, Grancharova, 2003)

  (Liang, Heemels, Bemporad, CDC 2011)



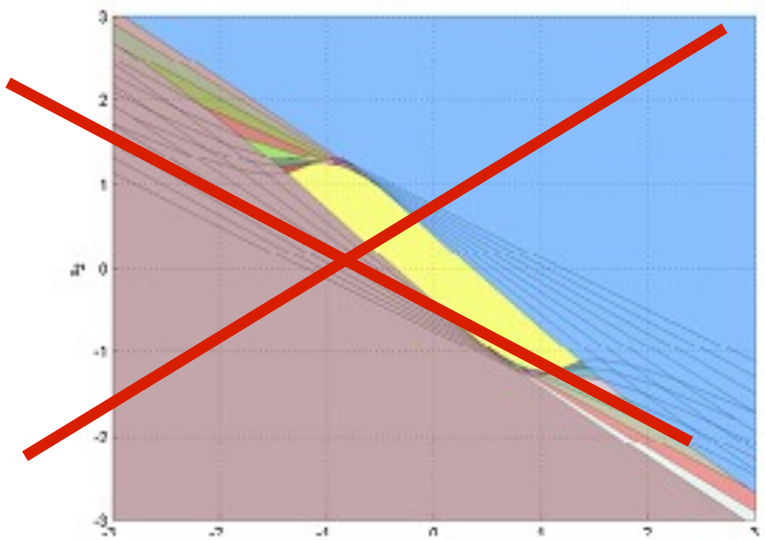


INTERPOLATION METHODS

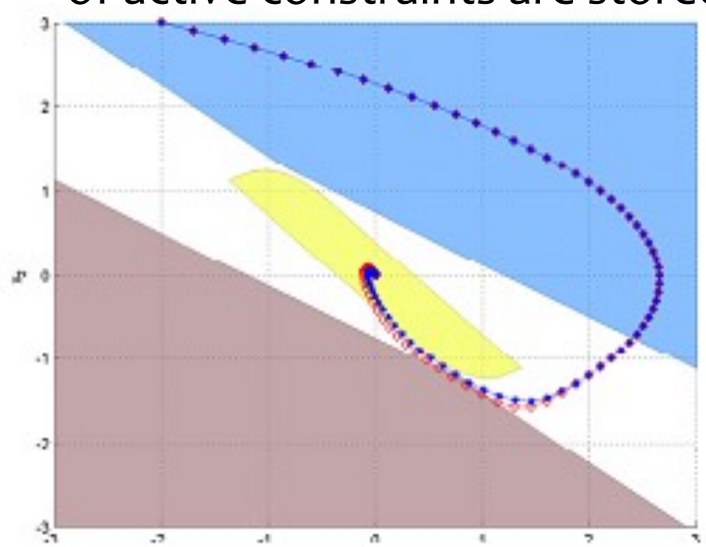
- Interpolate solution from reduced number of regions

(Pannocchia, Rawlings, Wright, 2007) (Christophersen, Zeilinger, Jones, Morari, 2007)

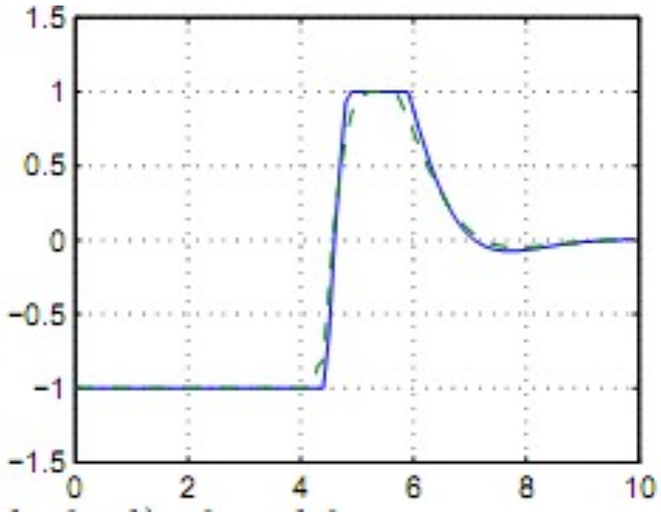
exact solution (99 regions)



3 most visited combinations of active constraints are stored



input trajectory



(Alessio, Bemporad, NMPC 2008)

$$\beta_i(x) = \max_j \{H_i^j x - K_i^j\}$$ max violation (how much x is outside region $H_i x \leq K_i$

$$\bar{u}(x) = \left(\sum_{i=1,\dots,L} \frac{1}{\beta_i(x)} \right)^{-1} \sum_{i=1,\dots,L} \frac{1}{\beta_i(x)} (F_i x + g_i)$$ or set $\bar{u}(x) = F_h x + g_h$

$$\beta_h(x) = \min_{i=1,\dots,L} \beta_i(x)$$

- Other approaches: (Jones, Morari, 2010) (Kvasnica, Fikar, 2010)

APPROXIMATE MPC SOLUTIONS

- Use gridding methods from *dynamic programming*

- Use any *function approximation* technique to get control law from *M* samples $u_i=u^*(x_i)$ of the exact MPC law, then check performance and prove stability

  - Lookup tables (linear interpolation, inverse distance weighting, ...)

  - Neural networks (Parisini, Zoppoli 1995)

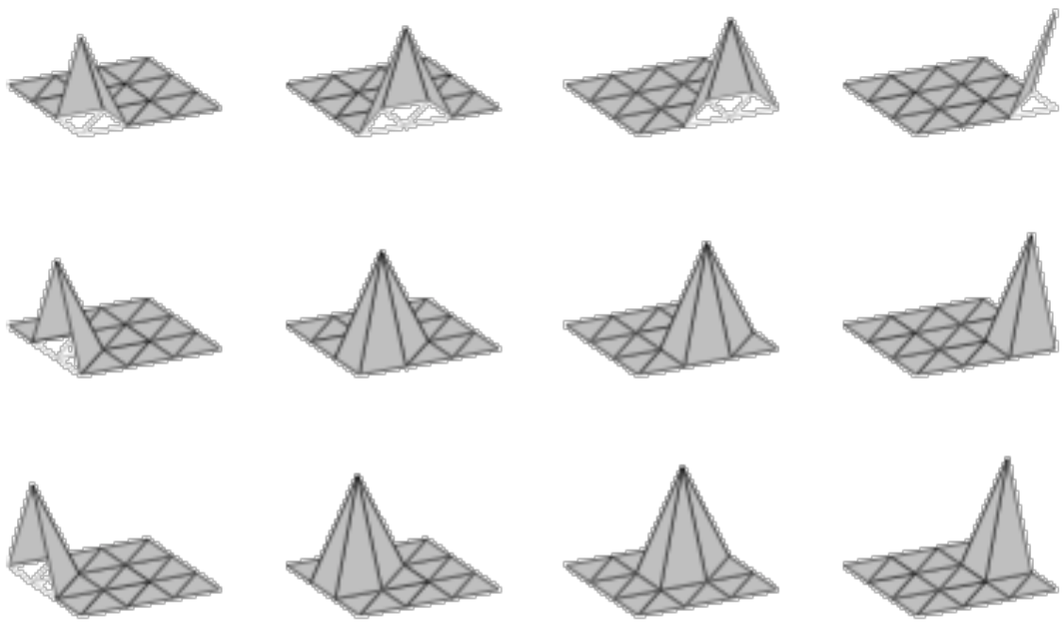
  - Hybrid (PWA) system identification (Breschi, Piga, Bemporad, 2016)

  - NL identification (Canale, Fagiano, Milanese NMPC'08)

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/

SEVENTH FRAMEWORK PROGRAMME

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)

EXAMPLE: AFTI-16

- Linearized model:

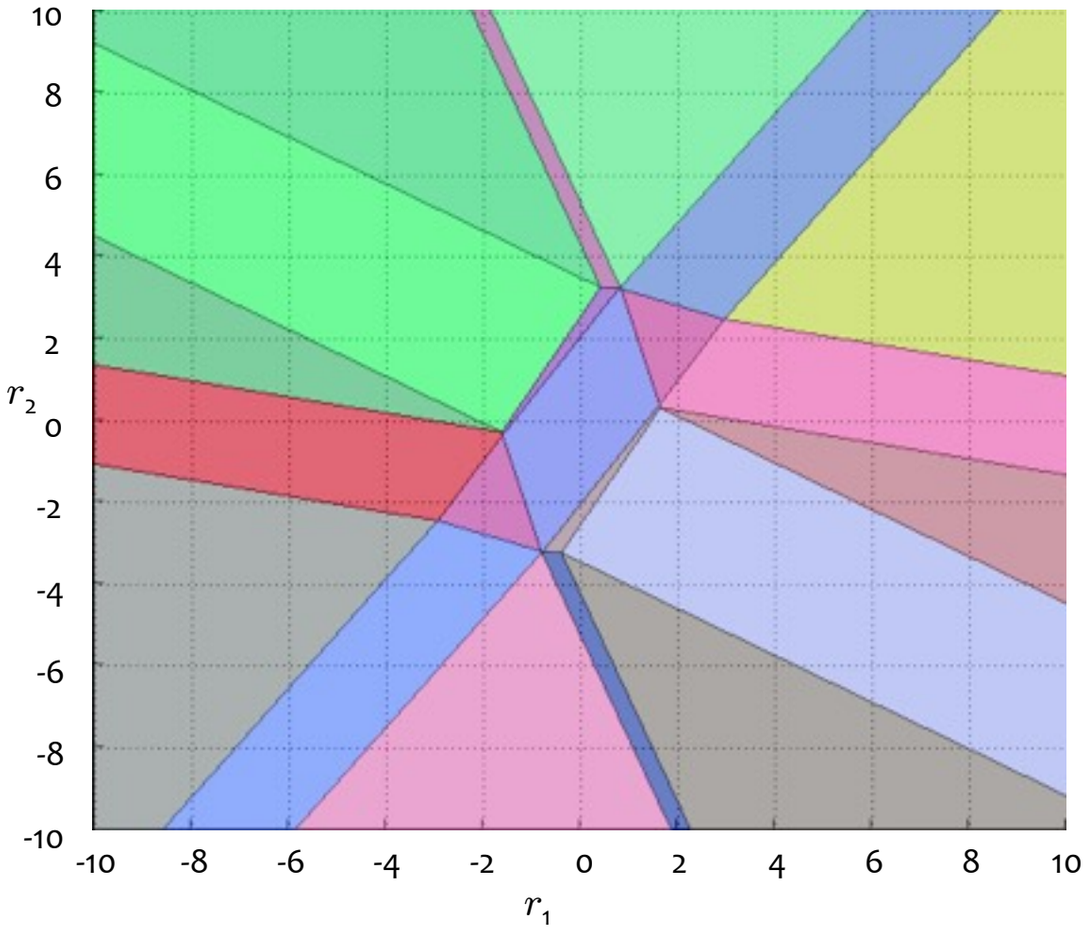
$$\begin{cases} \dot{x} = \begin{bmatrix} -.0151 & -60.5651 & 0 & -32.174 \\ -.0001 & -1.3411 & .9929 & 0 \\ .00018 & 43.2541 & -.86939 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} x + \\ \qquad \begin{bmatrix} -2.516 & -13.136 \\ -.1689 & -.2514 \\ -17.251 & -1.5766 \\ 0 & 0 \end{bmatrix} u \\ y = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} x, \end{cases}$$

- Inputs: elevator and flaperon angle
- Outputs: attack and pitch angle
- Sampling time: $T_s = .05$ s (+ zero-order hold)
- Constraints: max 25° on both angles
- Open-loop response: unstable
  (open-loop poles: $-7.6636, -0.0075 \pm 0.0556j, 5.4530$)

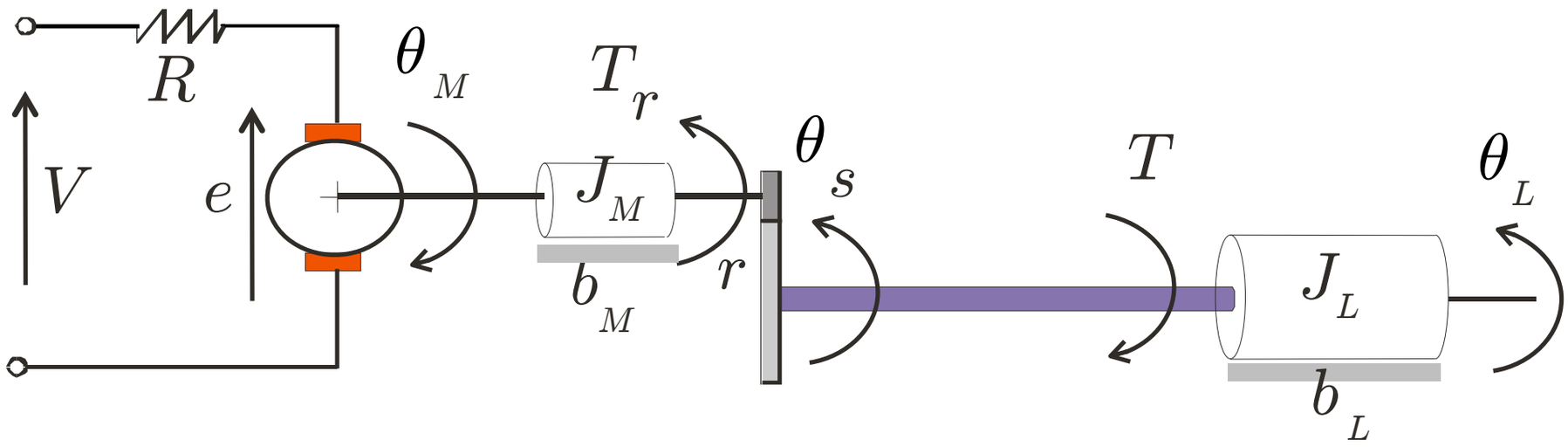
Explicit controller: 8 parameters, 51 regions

go to demo **`linear/afti16.m`** (Hyb-Tbx)

Section with x=[o o o o]', u=[o o]'



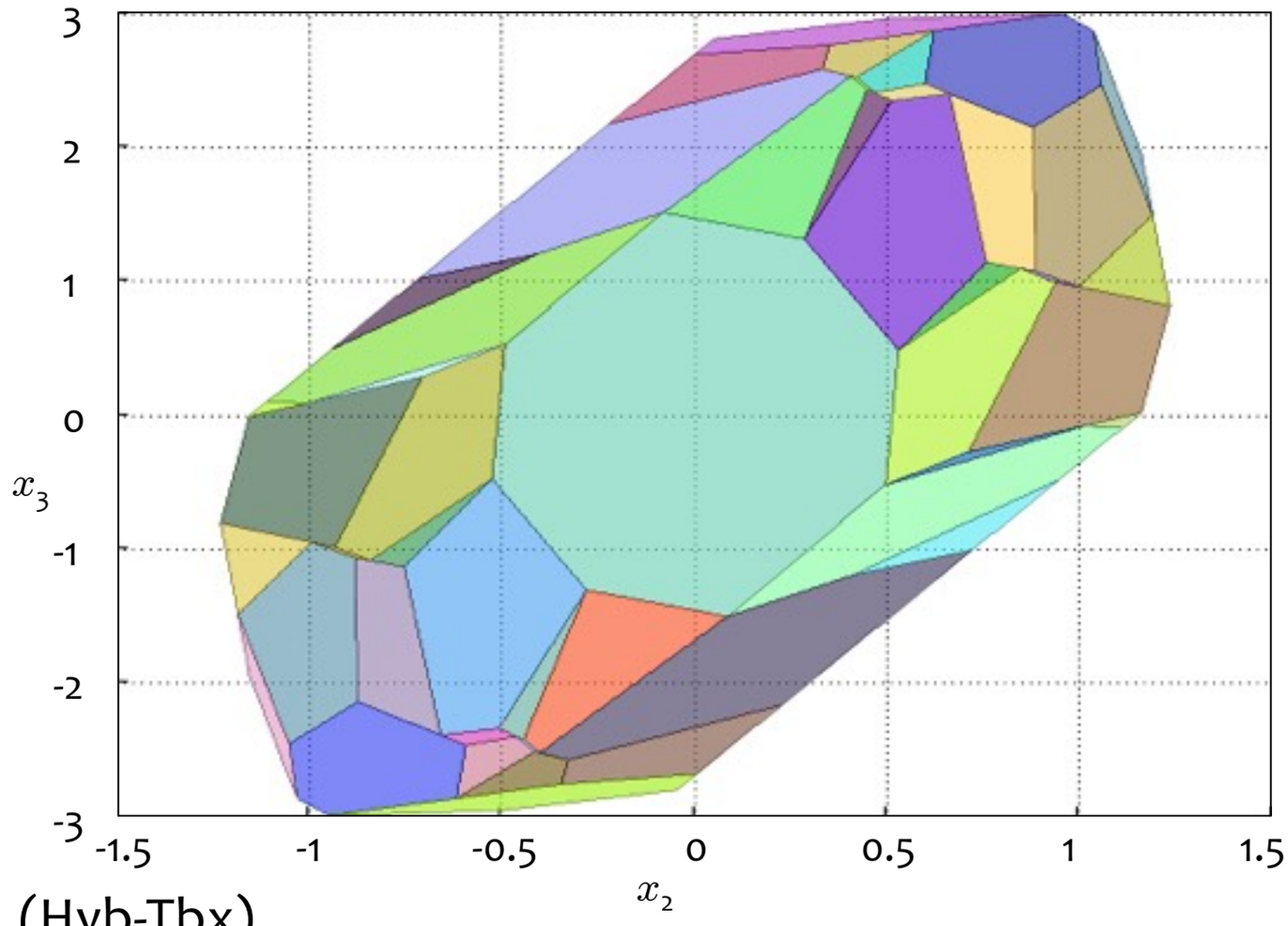
MPC OF A DC SERVOMOTOR



$$
\begin{aligned}
N &= 10 \\
N_u &= 2 \\
w_y &= \{1000, 0\} \\
w_{\delta u} &= .05 \\
u &\in [-220, 220] \text{ V} \\
y_2 &\in [-78.5398, 78.5398] \text{ Nm}
\end{aligned}
$$

Explicit controller:
7 parameters, 101 regions

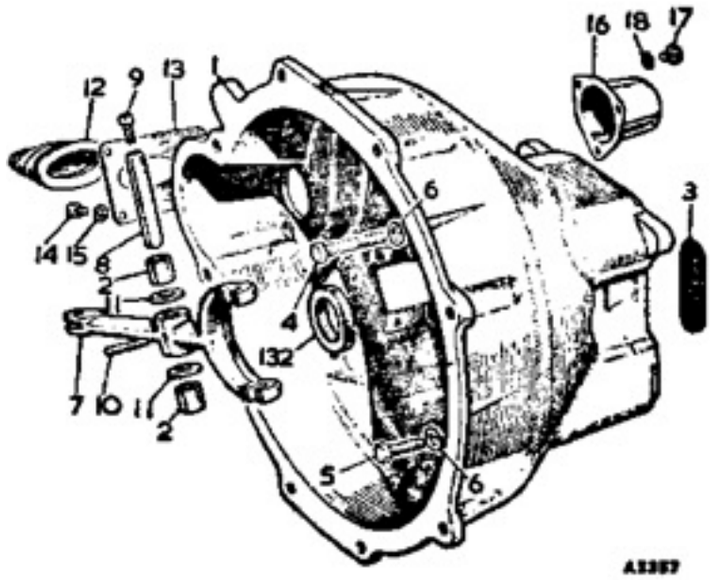
Section with x_1=x_4=0, r=[0 0]', u=0



go to demo **`linear/dcmotor.m`** (Hyb-Tbx)

DRY CLUTCH ENGAGEMENT

(Bemporad, Borrelli, Glielmo, Vasca, 2001)



- **Control objectives:**
  - small friction losses
  - fast engagement
  - driver comfort
- **Constraints:**
  - clutch force
  - clutch force derivative
  - minimum engine speed

Slip phase: $\omega_e > \omega_v$

$$\begin{aligned} I_e\dot{\omega}_e &= T_{in} - b_e\omega_e - T_{cl} \\ I_v\dot{\omega}_v &= T_{cl} - b_v\omega_v - T_l \end{aligned}$$

$$T_{cl} = kF_n\text{sign}(\omega_e - \omega_v)$$

Clutch engaged: $\omega_e = \omega_v = \omega$

$$(I_e + I_v)\dot{\omega} = T_{in} - (b_e + b_v)\omega - T_l$$

I_e	engine inertia
ω_e	crackshaft rotor speed
T_{in}	engine torque
b_e	crankshaft friction coefficient
T_{cl}	torque trasmitted by clutch
I_v	vehicle moment of inertia
ω_v	clutch disk rotor speed
b_v	clutch friction coefficient
T_l	equivalent load torque

LINEAR MPC DESIGN

- Linear model during slip

  + disturbance model

$$\begin{array}{rcl} x_1 & \triangleq & \omega_e \\ x_2 & \triangleq & \omega_e - \omega_v \\ u & \triangleq & F_n \end{array}$$

$$\begin{aligned} \dot{x}_1 &= -\frac{b_e}{I_e}x_1 + \frac{T_{in}}{I_e} - \frac{k}{I_e}u \\ \dot{x}_2 &= \left(-\frac{b_e}{I_e} + \frac{b_v}{I_v}\right)x_1 - \frac{b_v}{I_v}x_2 + \frac{T_{in}}{I_e} + \frac{T_\ell}{I_v} - k\left(\frac{1}{I_e} + \frac{1}{I_v}\right)u \\ \ddot{T}_{in} &= 0 \\ \dot{T}_\ell &= 0 \end{aligned}$$

- Sample time

  T=10 ms

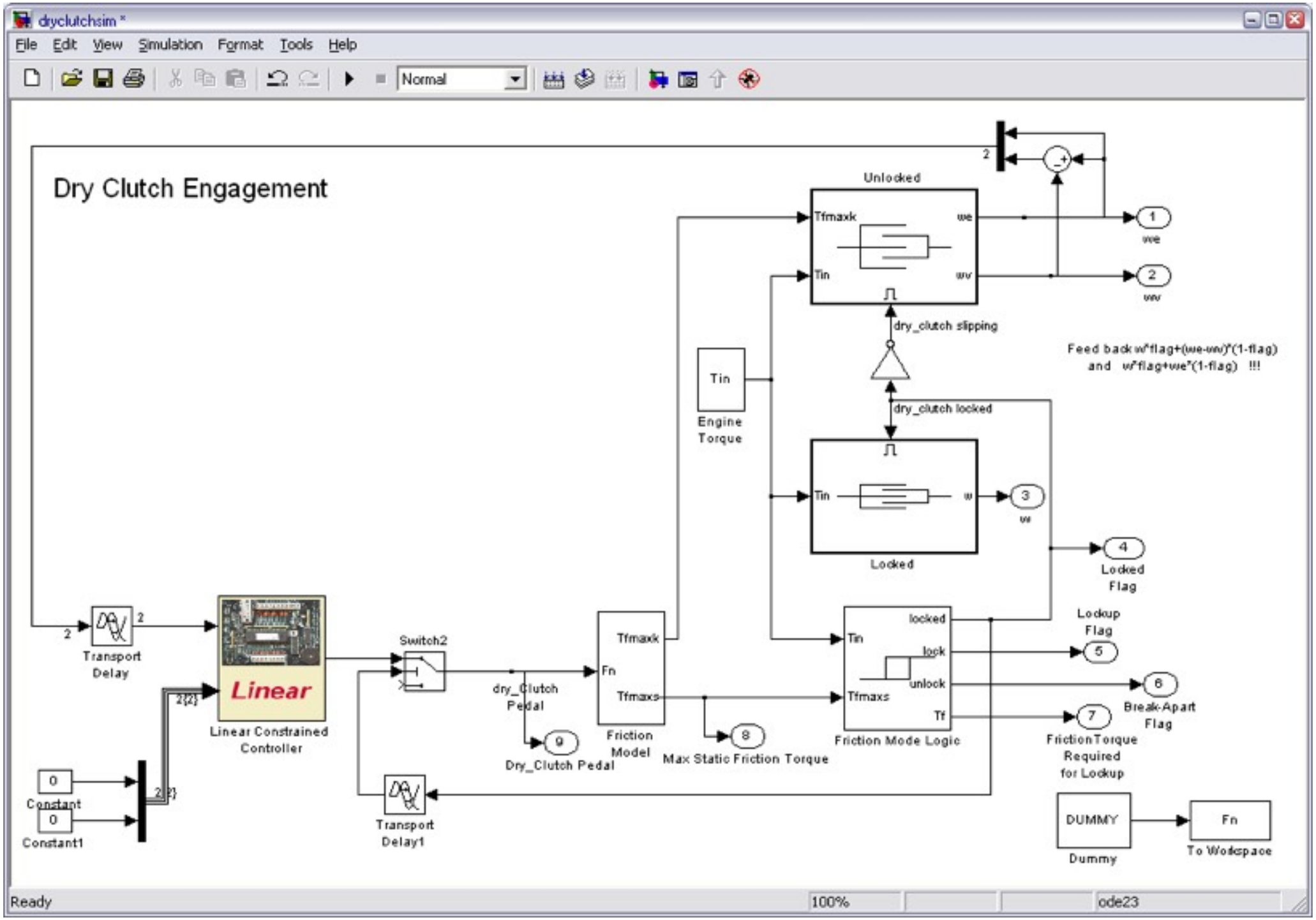
$$\begin{bmatrix} x_1(t+1) \\ x_2(t+1) \\ T_{in}(t+1) \\ \dot{T}_{in}(t+1) \\ T_l(t+1) \\ u(t) \end{bmatrix} = \begin{bmatrix} 0.9985 & 0 & 0.0500 & 0.0002 & 0 & -0.0049 \\ -0.0011 & 0.9996 & 0.0500 & 0.0002 & 0.0129 & -0.0062 \\ 0 & 0 & 1.0000 & 0.0100 & 0 & 0 \\ 0 & 0 & 0 & 1.0000 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1.0000 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1.0000 \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \\ T_{in}(t) \\ \dot{T}_{in}(t) \\ T_l(t) \\ u(t-1) \end{bmatrix} + \begin{bmatrix} -0.0049 \\ -0.0062 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \Delta u(t)$$

- Control objective:

$$\min_{\Delta u_0,\ldots,\Delta u_{Nu-1}} \sum_{k=0}^{N-1} Q x_{2,k}^2 + R\Delta u_k^2 + x'_N P x_N$$

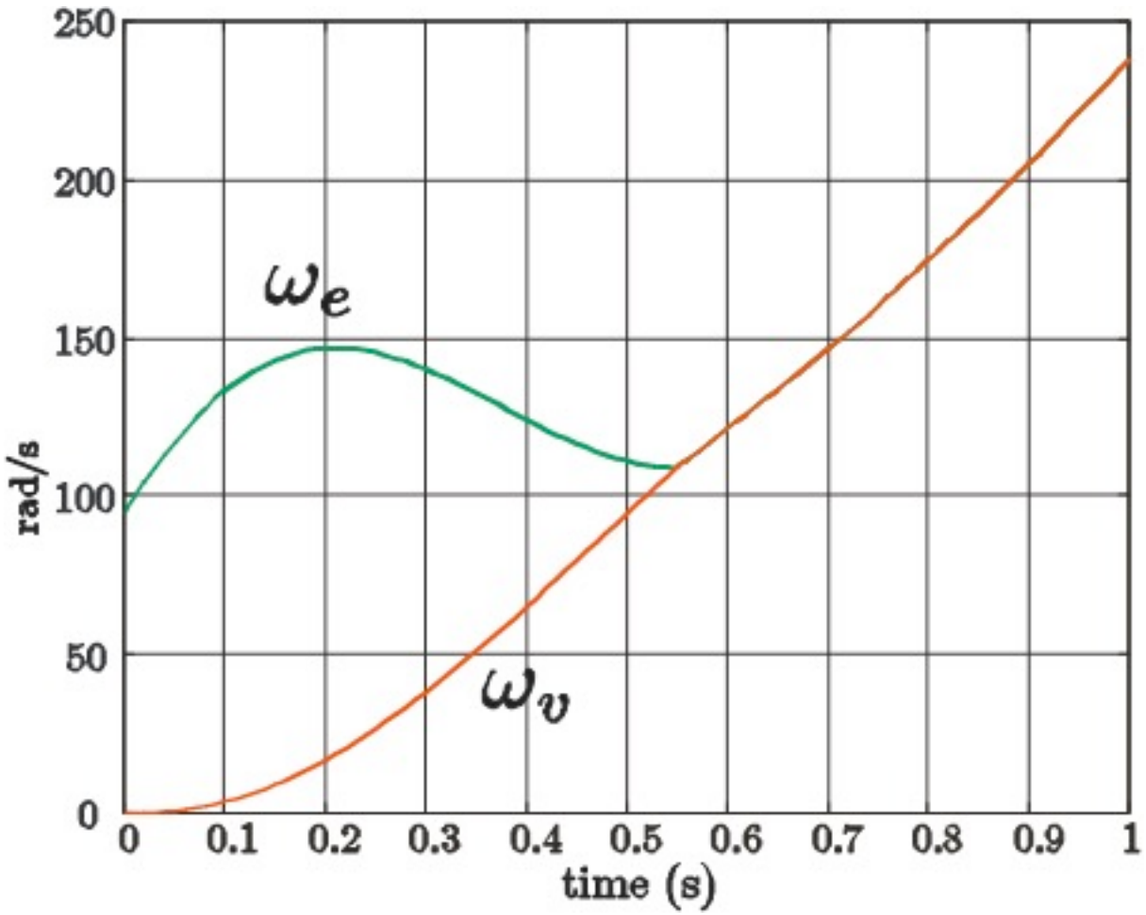
- Constraints: $0 \leq \Delta u \leq 80$ N, $0 \leq u \leq 5000$ N, $x_1 \geq 50$ rad/s, $x_2 \geq 0$

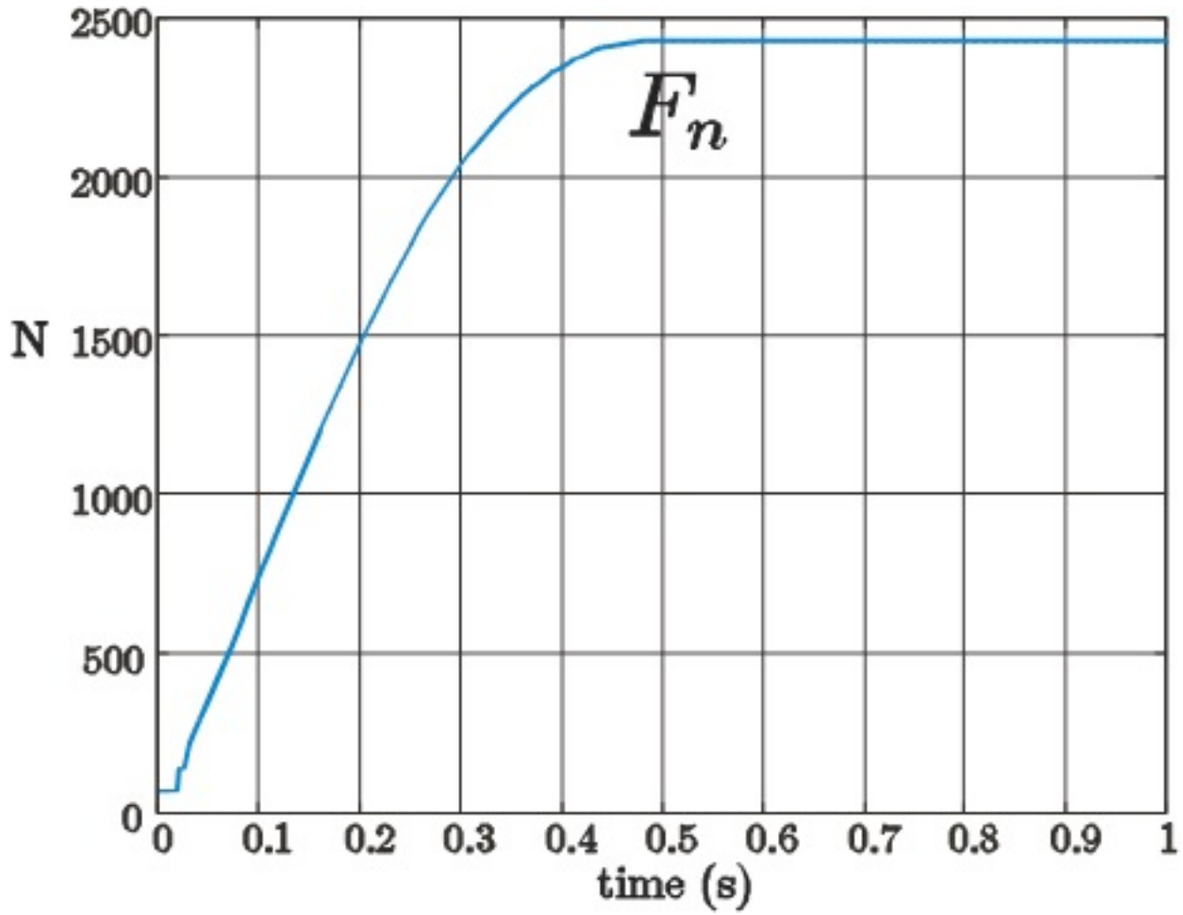
MPC TUNING



go to demo **/demos/dryclutch/dryclutch.m** (Hyb-Tbx)

MPC SIMULATION





Controller	τ^*	$F_n(\tau^*)$	E_d	$\dot{x}_2(\tau^*)$
1	0.57 s	2333 Nm	6676 J	230 rad/s^2
2	0.76 s	2515 Nm	9777 J	109 rad/s^2
3	0.83 s	2572 Nm	10838 J	62 rad/s^2

Choice:

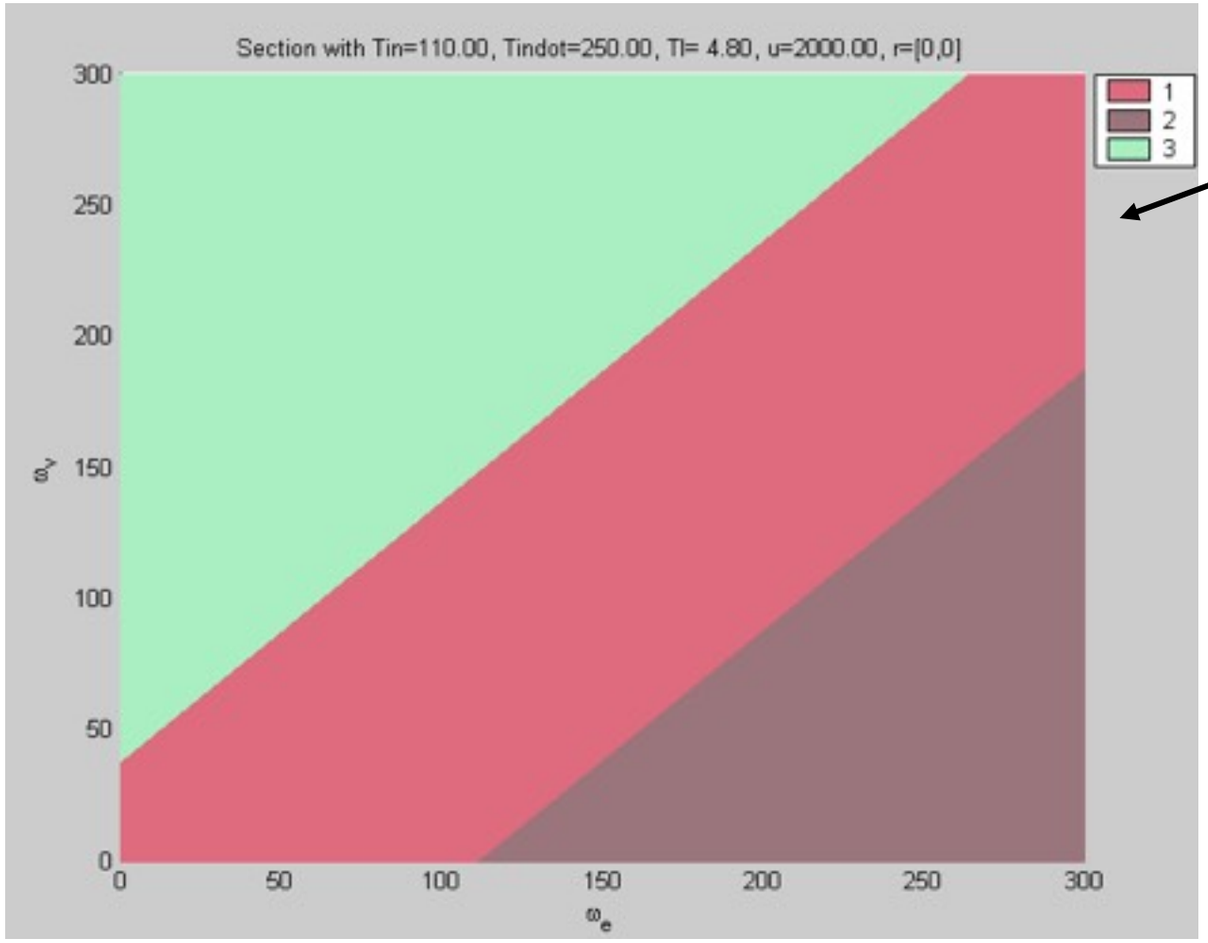
Q=2, R=1

N=10, N_u=1

EXPLICIT MPC CONTROLLER

$$\Delta u(\theta) = \begin{cases} [-0.003786\ 0.5396\ 0.1703\ 0.006058\ 0.04411]\,x + \\ -0.02101\,u + [0\ -0.5409]\,r & \text{if } \begin{bmatrix} -4.732e{-}005 & 0.006745 & 0.002129 & 7.572e{-}005 & 0.0005513 \\ 0.003786 & -0.5396 & -0.1703 & -0.006058 & -0.04411 \end{bmatrix} x + \\ & \begin{bmatrix} -0.0002626 \\ 0.02101 \end{bmatrix} u + \begin{bmatrix} 0 & -0.006762 \\ 0 & 0.5409 \end{bmatrix} r \le \begin{bmatrix} 1 \\ 0 \end{bmatrix} \\ & \text{(Region \#1)} \\ 80 & \text{if } [4.732e{-}005\ -0.006745\ -0.002129\ -7.572e{-}005\ -0.0005513]\,x + \\ & 0.0002626\,u + [0\ 0.006762]\,r \le [-1\ ] \\ & \text{(Region \#2)} \\ 0 & \text{if } [-0.003786\ 0.5396\ 0.1703\ 0.006058\ 0.04411]\,x + \\ & -0.02101\,u + [0\ -0.5409]\,r \le [0\ ] \\ & \text{(Region \#3)} \end{cases}$$

+ linear observer



ω_v rad/s

ω_e rad/s

T_{in}=110 Nm, $\dot{T}_{in}$=250 Nm/s
T_l=4.8 Nm, F_n=2000 Nm, r=[0 0]'

Alternative: explicit **hybrid** MPC

- Switching model (slipping mode, engaged mode)
- Design an explicit MPC controller for the switched system

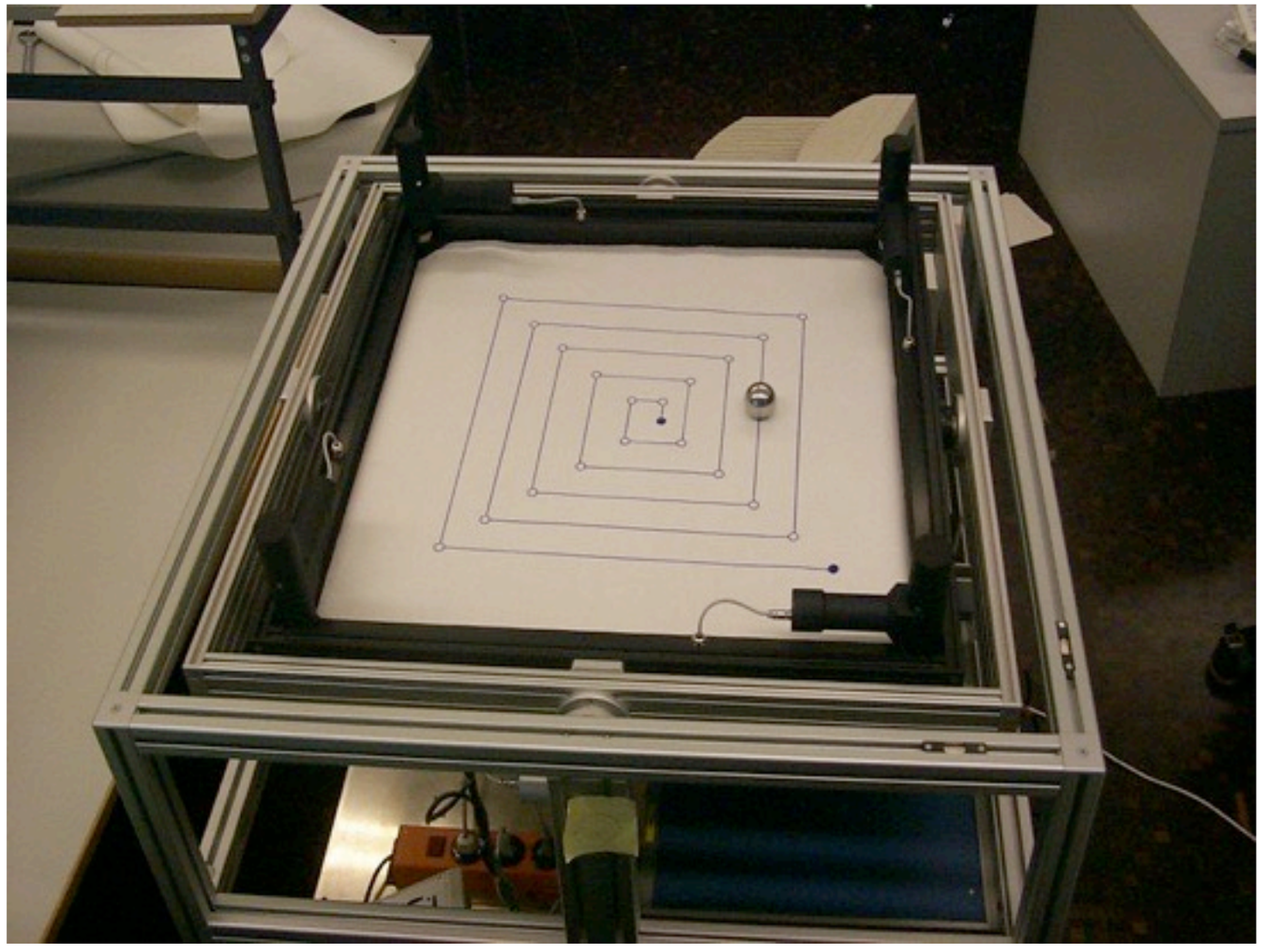
MPC REGULATION OF A BALL ON A PLATE

Task:

- Tune an MPC controller by simulation, using the ***MPC Simulink Toolbox***

- Get the ***explicit solution*** of the MPC controller.

- Validate the controller on ***experiments***.



BALL & PLATE: EXPERIMENTAL SETUP

Ball & Plate System

Host

TCP/IP

Code

Data

Ball & Plate

Target

u_1, u_2

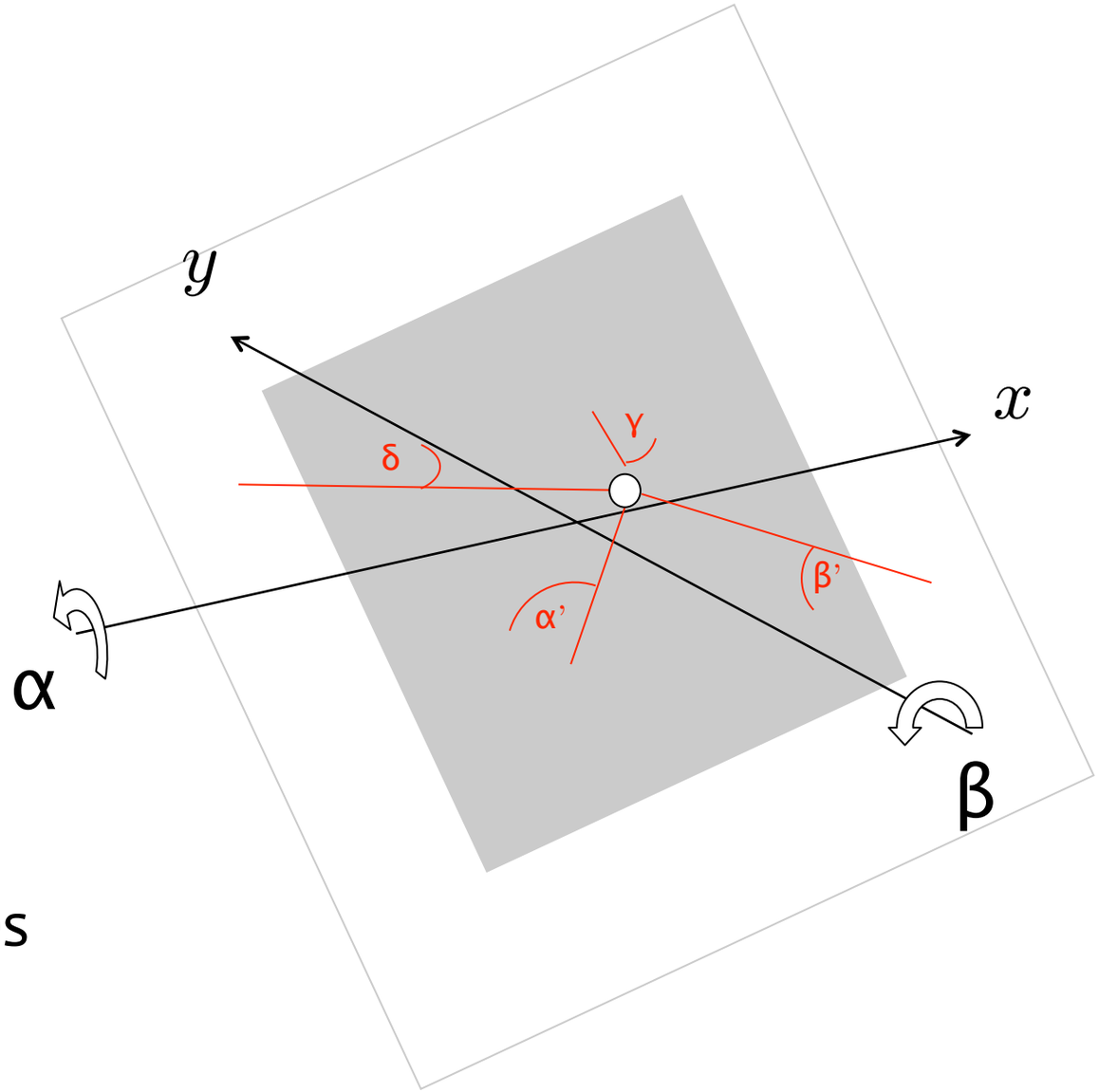
a,b

x, y

BALL & PLATE: MODEL AND SPECIFICATIONS

- Specifications:

Angle:	-17 deg … +17deg
Plate:	-30 cm …+30 cm
Input Voltage:	-10 V… +10 V
Computer:	PENTIUM166
Sampling Time:	30 ms



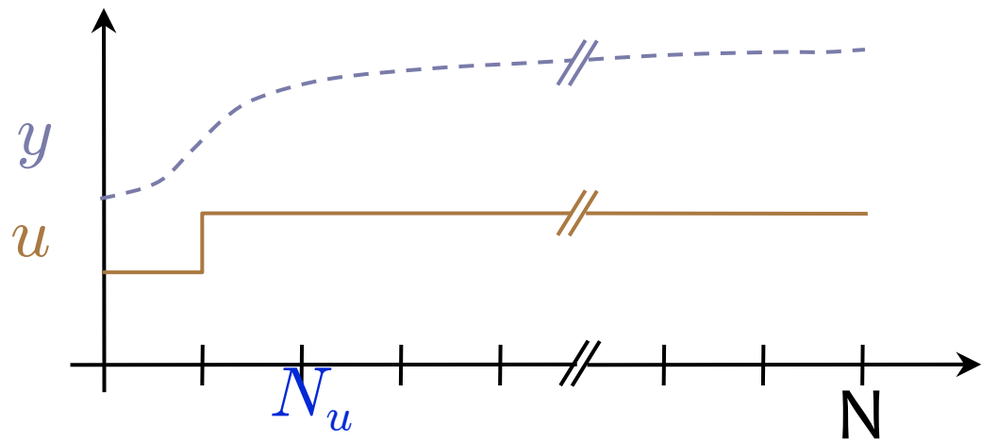
- Model: LTI 14 states
  Constraints on inputs and states

MPC TUNING

Sampling time: T_s = **30** ms

Prediction horizon: N = **50**

Free control moves: N_u = **2**



Output constraint horizon: **1** (*soft constraint*)

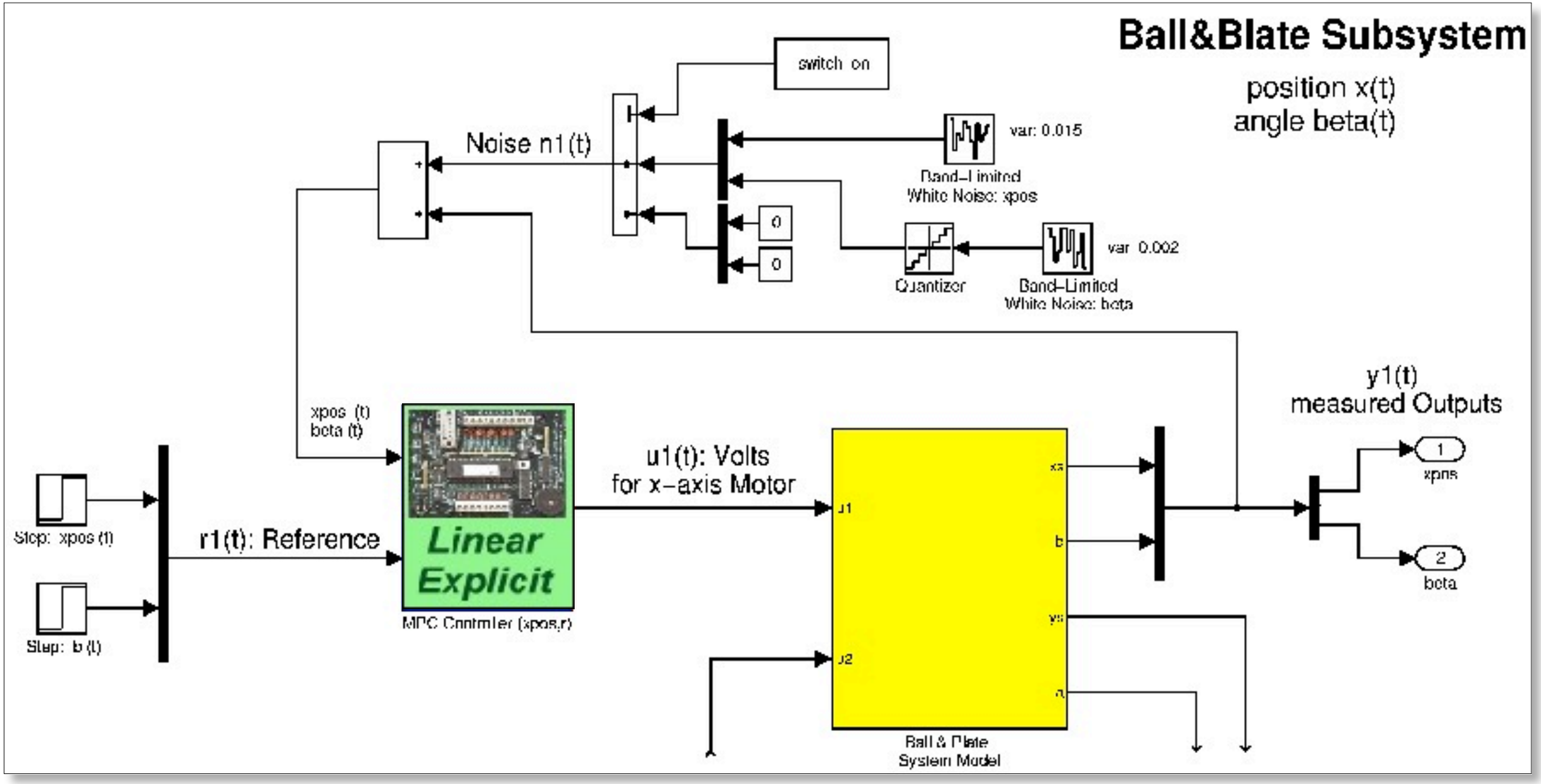
Input constraint horizon: **1** (*hard constraint*)

Weight on position error: **5**

Weight on input voltage changes: **1**

IMPLEMENTATION

- Solve mp-QP and implement explicit MPC

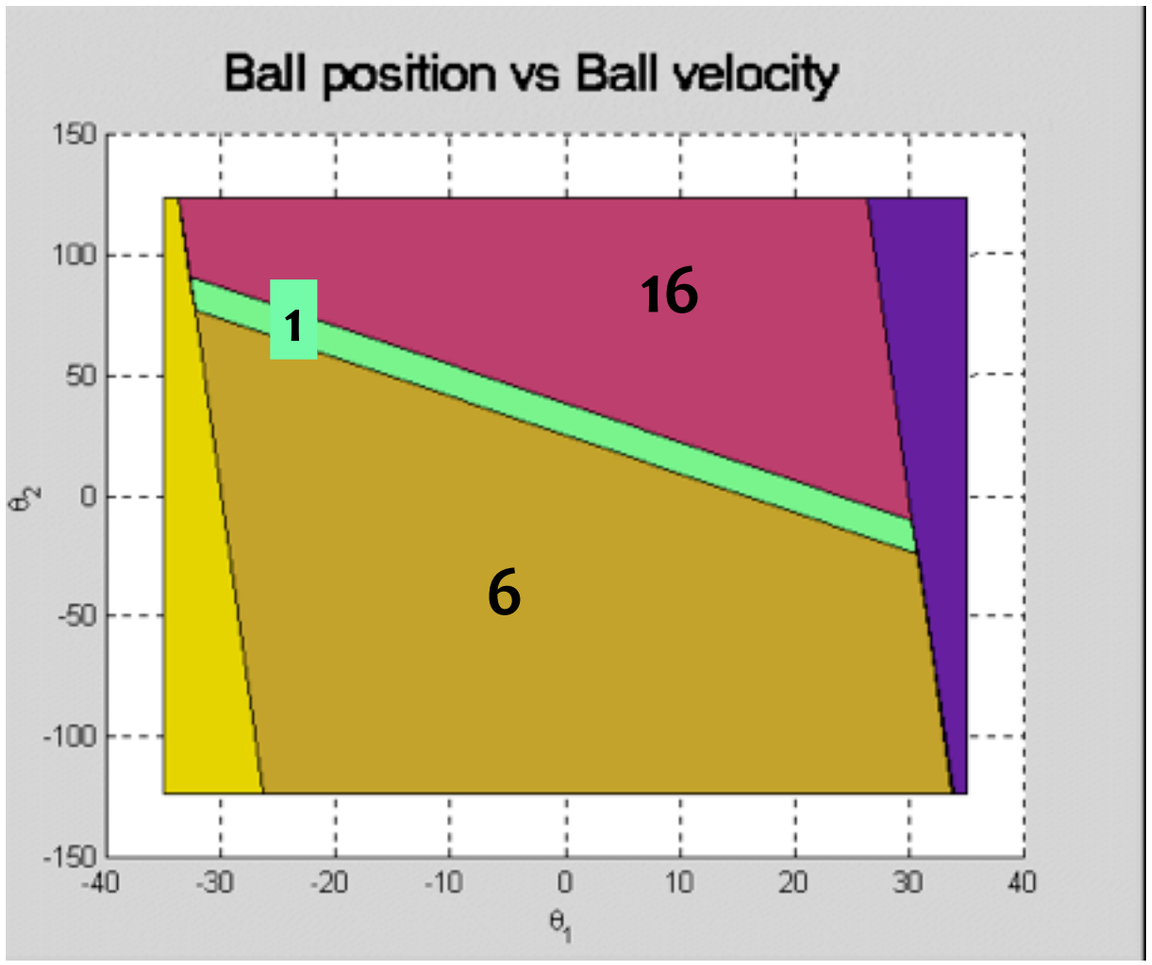


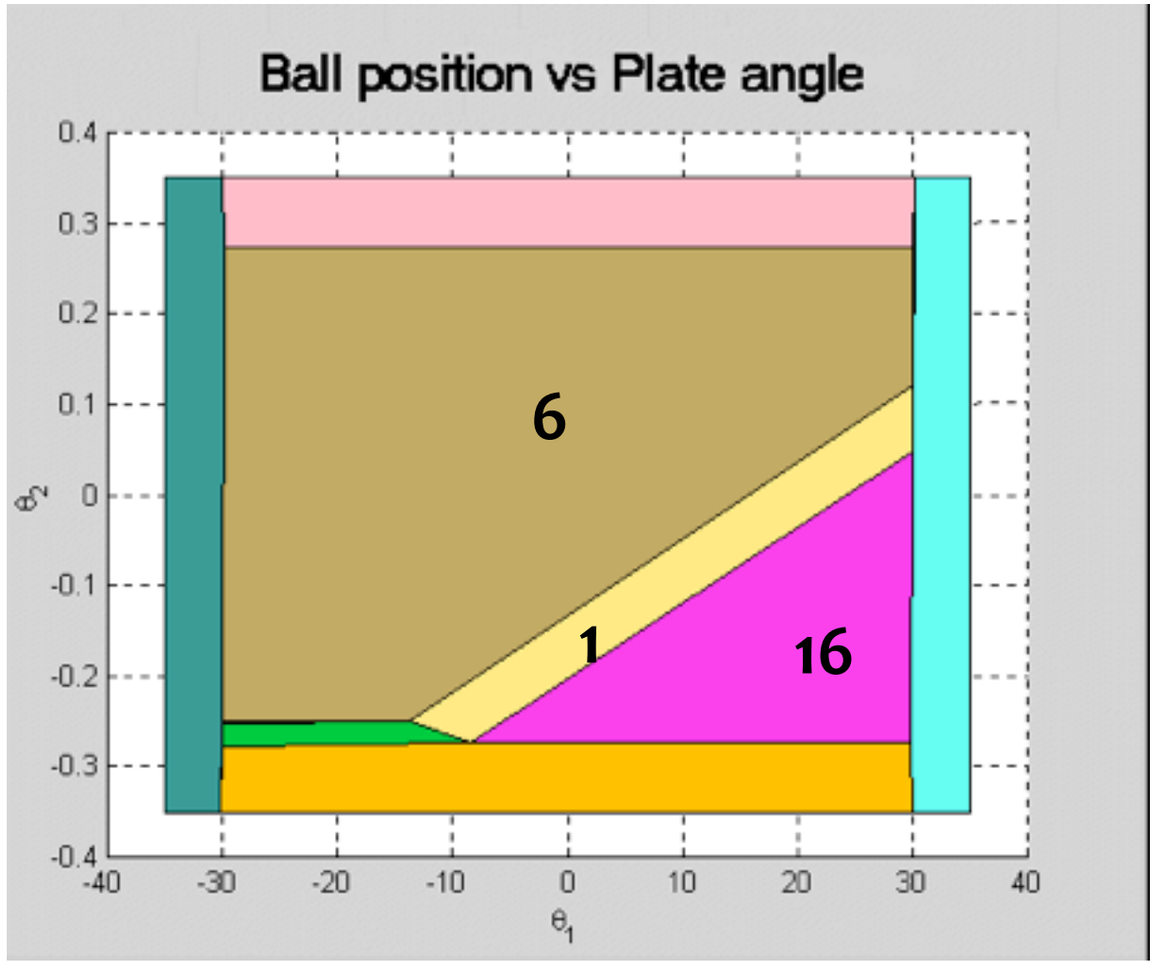
E.g: Real-Time Workshop + xPC Toolbox

EXPLICIT MPC SOLUTION

Controller: x: 22 regions, y: 23 regions

x-MPC: sections at $\alpha_x=0$, $\mathring{\alpha}_x=0$, $u_x=0$, $r_x=18$, $r_\alpha=0$



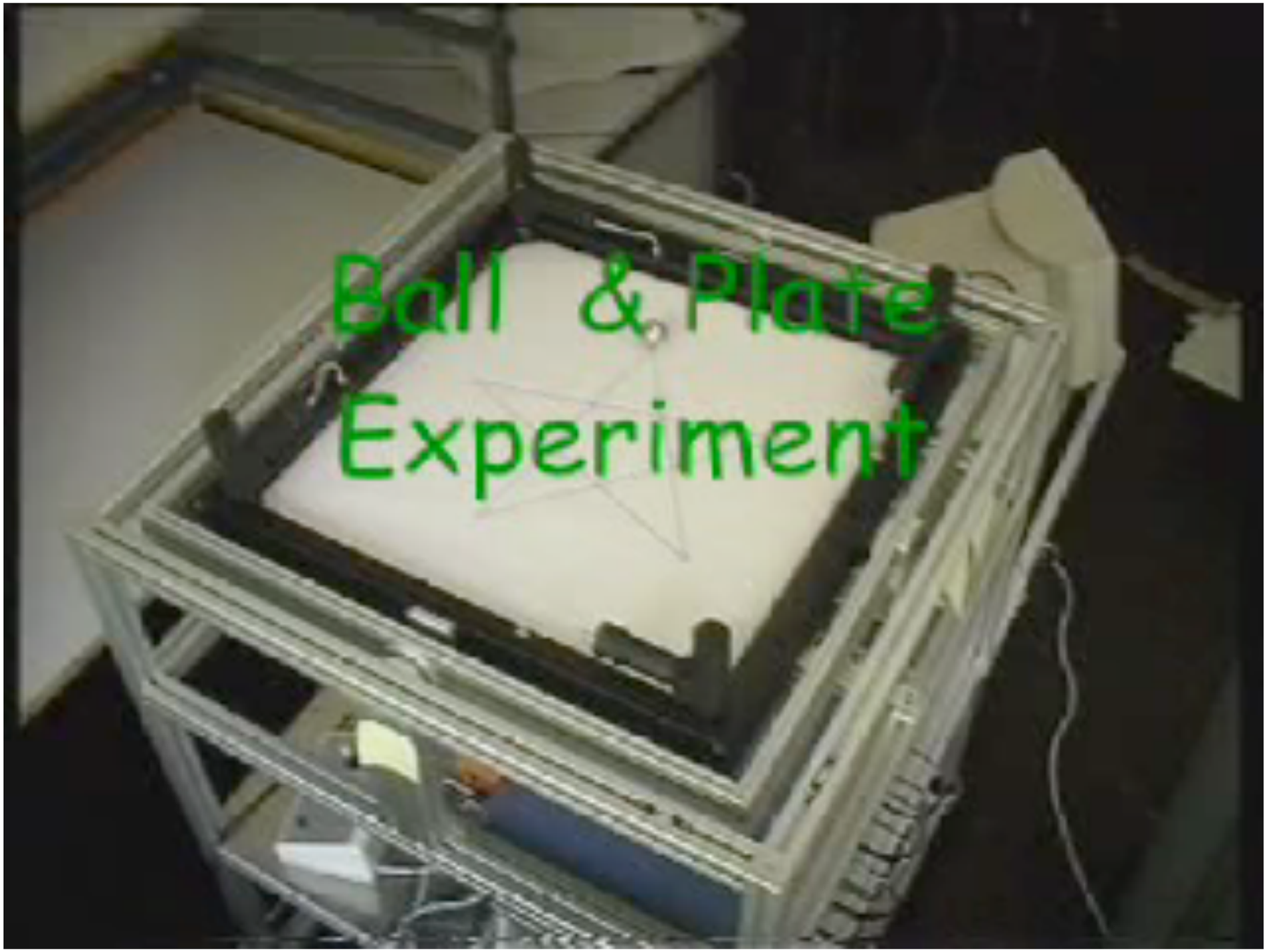


Region 1: LQR Controller (near equilibrium)

Region 6: Saturation at -10

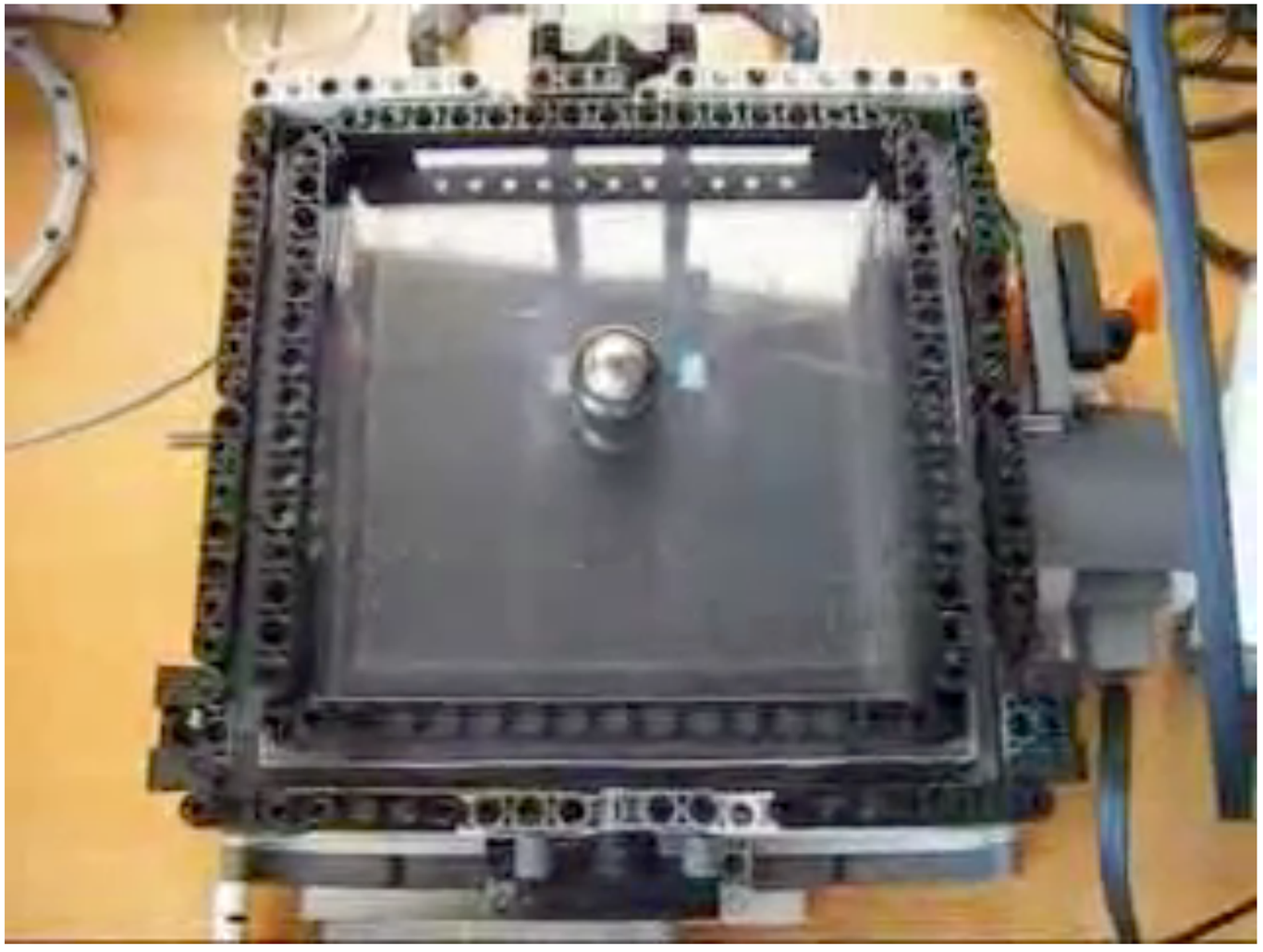
Region 16: Saturation at +10

BALL AND PLATE EXPERIMENT



BALL AND PLATE EXPERIMENT

Ball and plate experiment in LEGO, using explicit MPC and Hybrid Toolbox



(by Daniele Benedettelli, Univ. of Siena, July 2008)

- 20Hz sampling frequency
- camera used for position feedback
- explicit MPC coded using integer numbers

EXPLICIT MPC FOR IDLE SPEED CONTROL

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

- Ford pickup truck, V8 4.6L gasoline engine

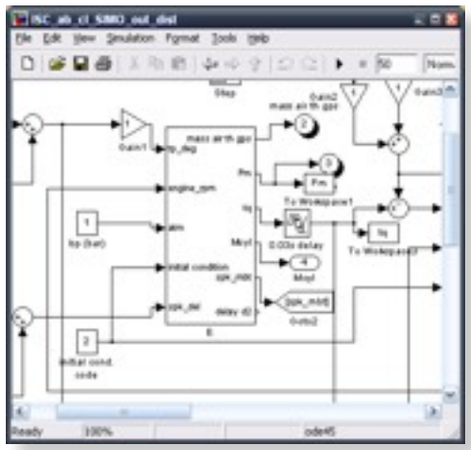
- Process:
  - *1 output* (engine speed) to regulate
  - *2 inputs* (airflow, spark advance)
  - input *delays*

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

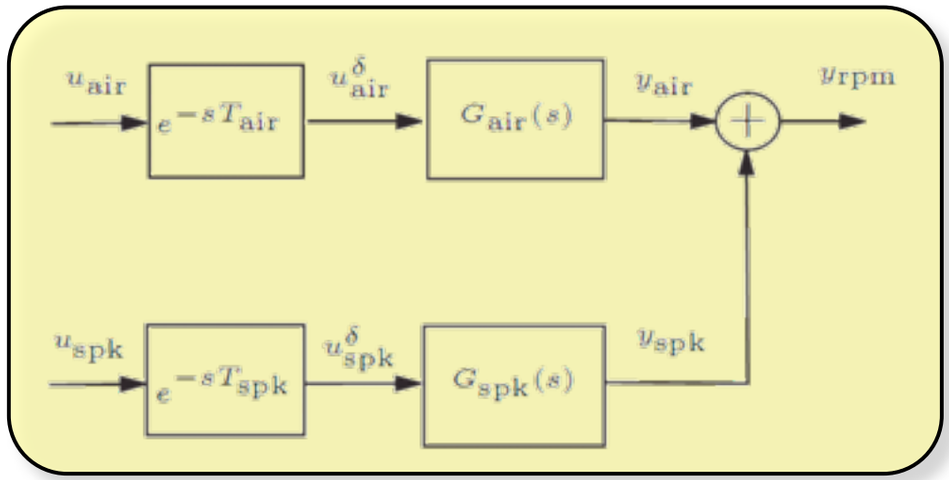
- Problem suitable for MPC design (Hrovat, 1996)

EXPLICIT MPC DESIGN FLOW

NL Simulink model



prediction model



control specs

add integral action

MPC setup (on-line QP)

closed-loop simulation

multi-parametric solver

```
#define EXPCON_NTH 2
#define EXPCON_NH 16
#define EXPCON_NF 5
#define EXPCON_NYM 2
static double EXPCON_F[]={
    -6.83553,-12.0296,0,0,-26.9062,-6
    -154.828};
```

PWA control law (C-code)

identification

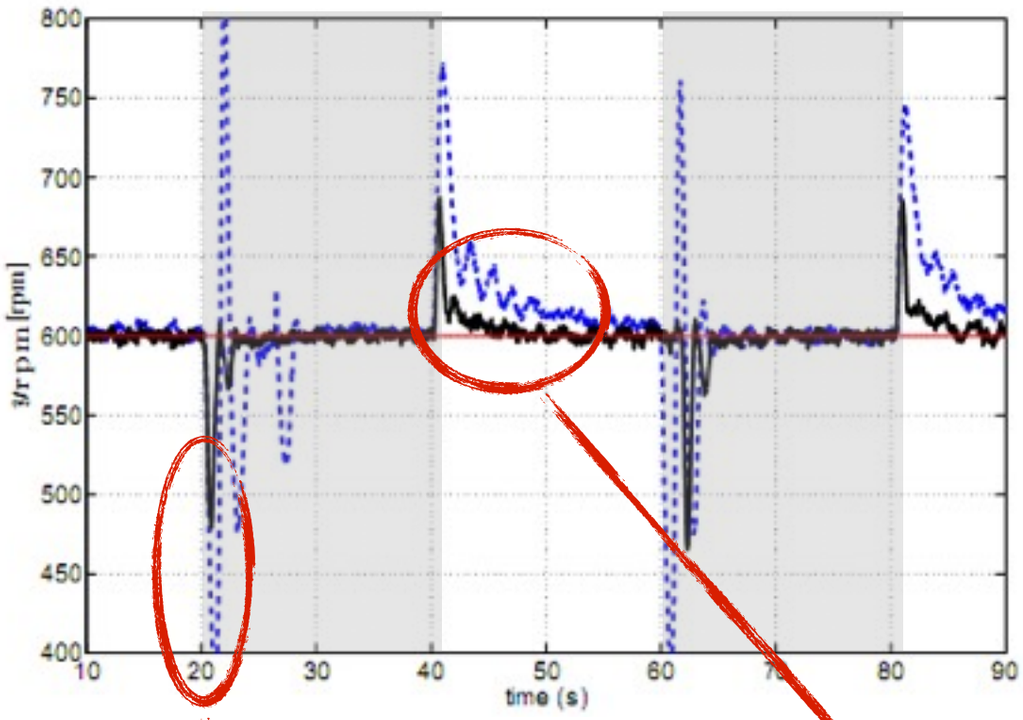
revise **weights** & **observer**

experiments

MATLAB tool: **Hybrid Toolbox**

EXPLICIT MPC FOR IDLE SPEED - EXPERIMENTS

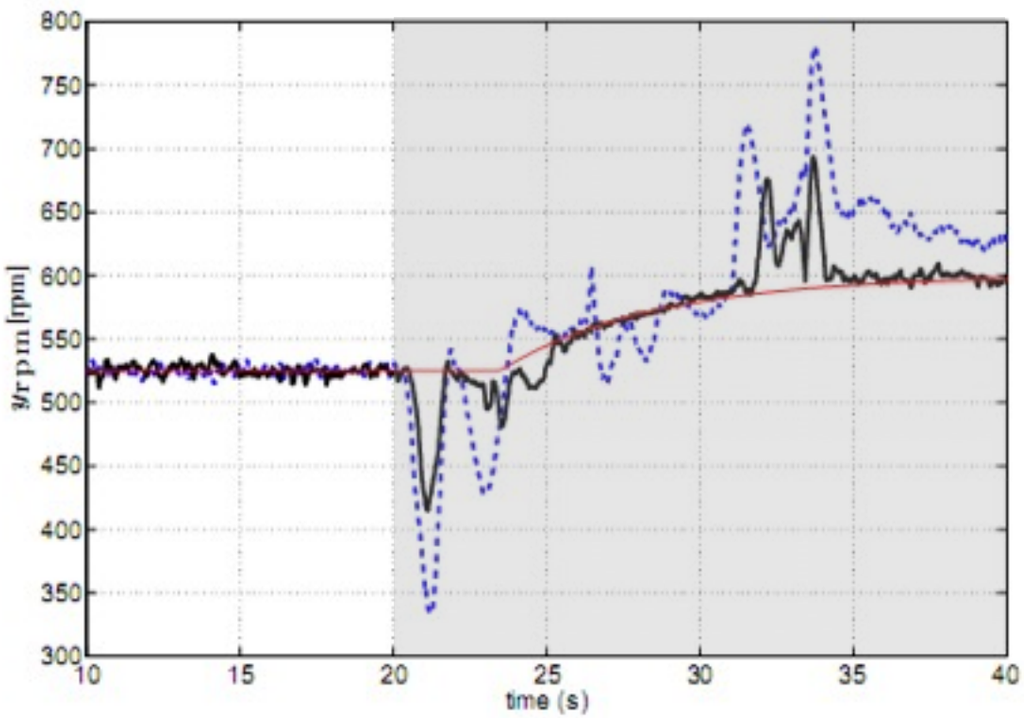
(Di Cairano, Yanakiev, Bemporad, Kolmanovsky, Hrovat, 2008)



Load torque (power steering)

peak reduced by 50%

convergence 10s faster



Power steering + air conditioning

- Sampling time = 30 ms
- Explicit MPC implemented in dSPACE MicroAutoBox rapid prototyping unit

explicit MPC

baseline controller (linear)

set-point

MPQP IN PORTFOLIO OPTIMIZATION

(Bemporad, *NMPC plenary*, 2008)
(Best, Grauer, *Manag. Science*, 1991)

Markowitz portfolio optimization

$$\begin{array}{ll} \min & z'\Sigma z \\ \text{s.t.} & p'z \geq x \\ & [1 \ \ldots \ 1]\, z = 1 \\ z \geq 0 & \end{array}$$

z_i = money invested in asset i

p_i = expected return of asset i

Σ_{ij} = covariance of assets i,j

x = expected minimum return of portfolio

Objective: minimize variance (=risk)

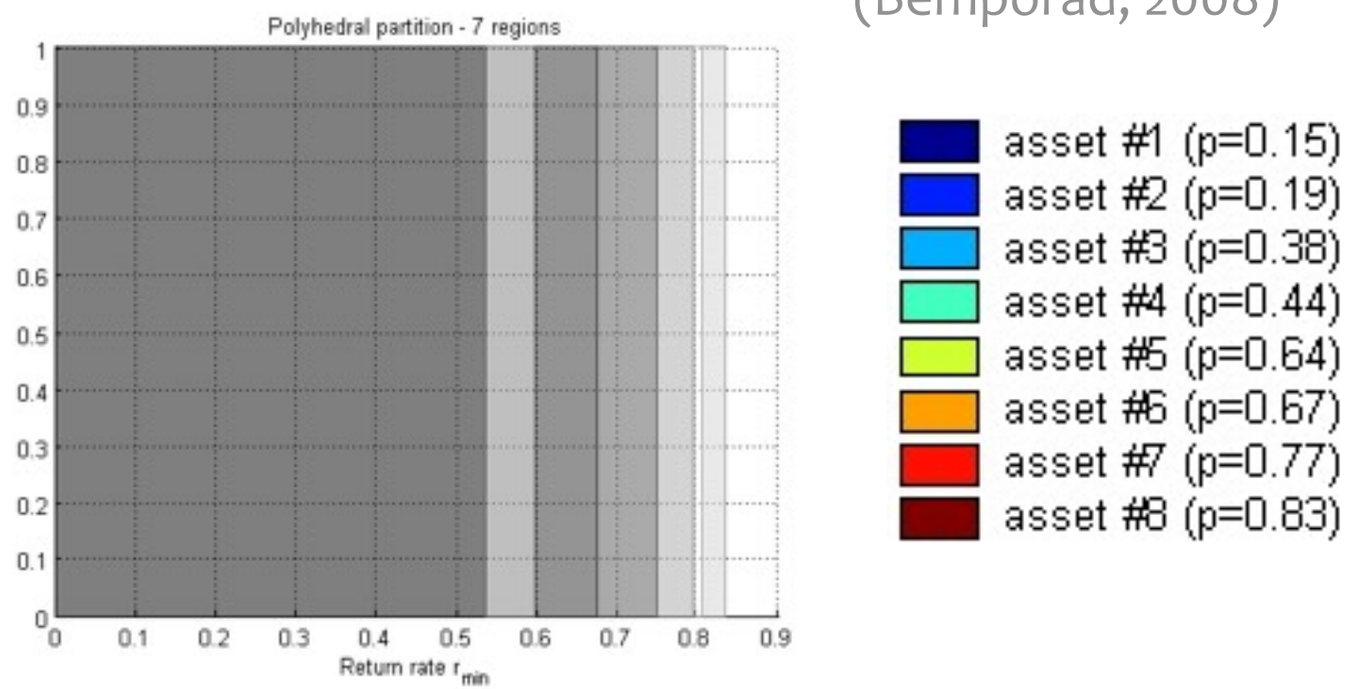
Constraint: guarantee a minimum expected return

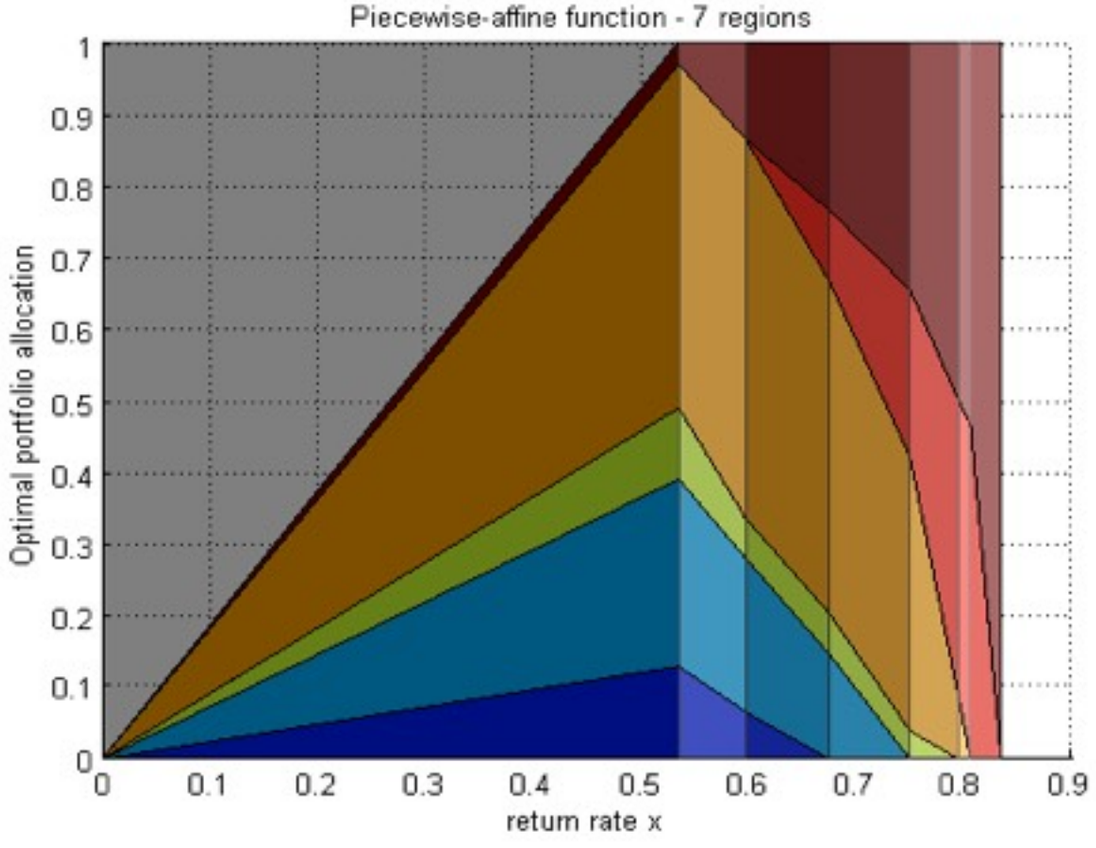
MPQP IN PORTFOLIO OPTIMIZATION

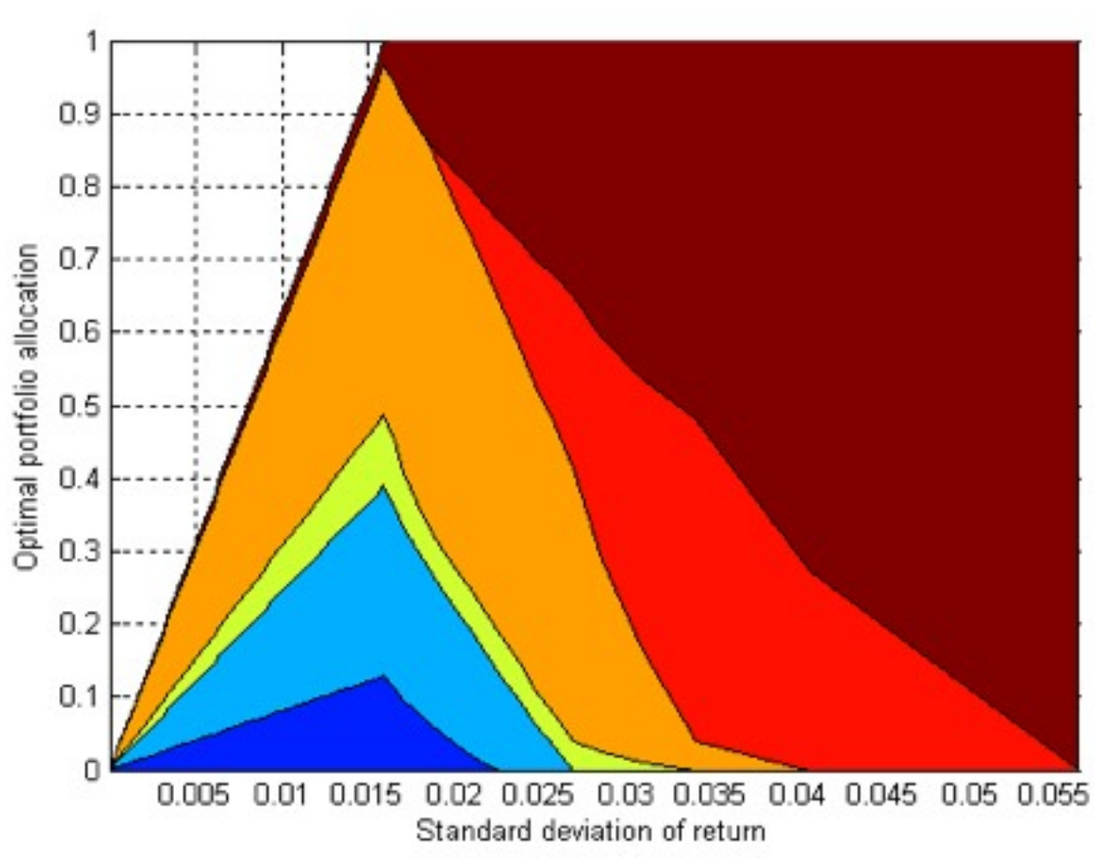
Multiparametric QP solution

$$\begin{aligned} \min \quad & z'\Sigma z \\ \text{s.t.} \quad & p'z \geq x \\ & [1 \ \dots \ 1]\, z = 1 \\ & z \geq 0 \end{aligned}$$

(Bemporad, 2008)







COMPARING DIFFERENT SOLUTION METHODS FOR MPC

Which method to prefer for embedded MPC ?

	Explicit MPC	Implicit (=on-line QP)					
		active set	interior point	gradient projection	proximal Newton	ADMM	PQP
speed (small probs)	🙂	🙂	😐	😐	🙂	😐	😐
speed (large probs)	☹️	☹️	🙂	😐	🙂	😐	☹️
worst-case estimates	🙂	🙂	☹️	😐	😐	☹️	☹️
numerical arithmetics	🙂	😐	😐	🙂	🙂	🙂	🙂
off-line computations	☹️	🙂	🙂	🙂	🙂	🙂	🙂
solution quality (feas/opt)	🙂	🙂	😐	😐	😐	😐	😐
data memory	☹️	🙂	🙂	🙂	🙂	🙂	🙂
control code	🙂	😐	😐	🙂	😐	🙂	🙂
best for problem size	small	medium	large	medium	medium	medium	small

very rough classification:

```
small  ≤   6 vars,   15 constraints,    8 states/references
medium ≃  20 vars,   80 constraints,   50 states/references
large  ≥ 500 vars, 3000 constraints, 2000 states/references
```

COMPLEXITY CERTIFICATION FOR ACTIVE SET QP SOLVERS

- Consider the dual active-set QP solver of Goldfarb-Idnani

(Goldfarb, Idnani, 1983)

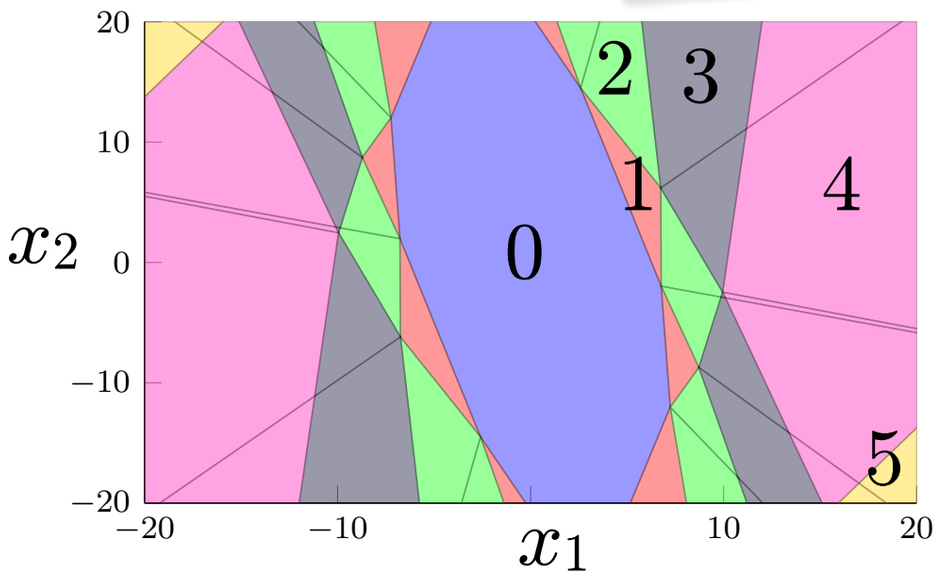
- What is the worst-case number of iterations over x to solve the QP

$$\begin{aligned} z^*(x) = \arg\min_z \quad & \tfrac{1}{2} z'Hz + x'F'z \\ \text{s.t.} \quad & Gz \le W + Sx \end{aligned}$$

- **Key result:**

The number of iterations to solve the QP is a piecewise constant function of the parameter x

(Cimini, Bemporad, IEEE TAC, 2017)



We can **exactly** quantify how many iterations (flops) the QP solver takes in the worst-case !

COMPLEXITY CERTIFICATION FOR ACTIVE SET QP SOLVERS

(Cimini, Bemporad, 2016, submitted)

- Examples (from MPC Toolbox):

	inv. pend.	DC motor	nonlin. demo	AFTI 16
# vars	5	3	6	5
# constraints	10	10	18	12
# params	9	6	10	10
Explicit MPC				
# regions	87	67	215	417
max flops	3382	1689	9184	16434
max memory (kb)	55	30	297	430
Implicit MPC				
max iters	11	9	13	16
max flops	3809	2082	7747	7807
sqrt	27	9	37	33
max memory (kb)	15	13	20	16

explicit MPC is faster in the worst-case

online QP is faster in the worst-case

- It is possible to combine explicit and on-line QP for best tradeoff

CONCLUDING REMARKS ABOUT MPC SOLVERS

- Embedded QP solvers for MPC must be **very simple** to code, **fast**, amenable for **low-precision** arithmetic, and with proved bounds on **real-time execution**

- A **"best QP algorithm" does not exist**, must have a library of methods, depending on memory/throughput/precision reqs.

- Research on QP solution methods started ~60 years ago …

(Beale, 1955)

ON MINIMIZING A CONVEX FUNCTION SUBJECT TO LINEAR INEQUALITIES

By E. M. L. BEALE

Admiralty Research Laboratory, Teddington, Middlesex

SUMMARY

THE minimization of a convex function of variables subject to linear inequalities is discussed briefly in general terms. Dantzig's Simplex Method is extended to yield finite algorithms for minimizing either a convex quadratic function or the sum of the t largest of a set of linear functions, and the solution of a generalization of the latter problem is indicated. In the last two sections a form of linear programming with random variables as coefficients is described, and shown to involve the minimization of a convex function.

More research on QP is still needed to have an impact in real applications !

THE ORIGINS OF (MULTI)PARAMETRIC PROGRAMMING

(MONO)-PARAMETRIC LP

THE COMPUTATIONAL ALGORITHM FOR THE
PARAMETRIC OBJECTIVE FUNCTION[1]

Saul Gass
U. S. Air Force[2]

and

Thomas Saaty
Melpar, Inc.[3]

(Gass and Saaty, 1955)

Let $\delta \le \lambda \le \phi$ be an arbitrary interval on the real line
For each λ in this interval, find a vector $x = (x_1, x_2, \ldots x_n)$ which minimizes

$$\sum_{j=1}^{n} (d_j + \lambda d'_j)\, x_j,$$

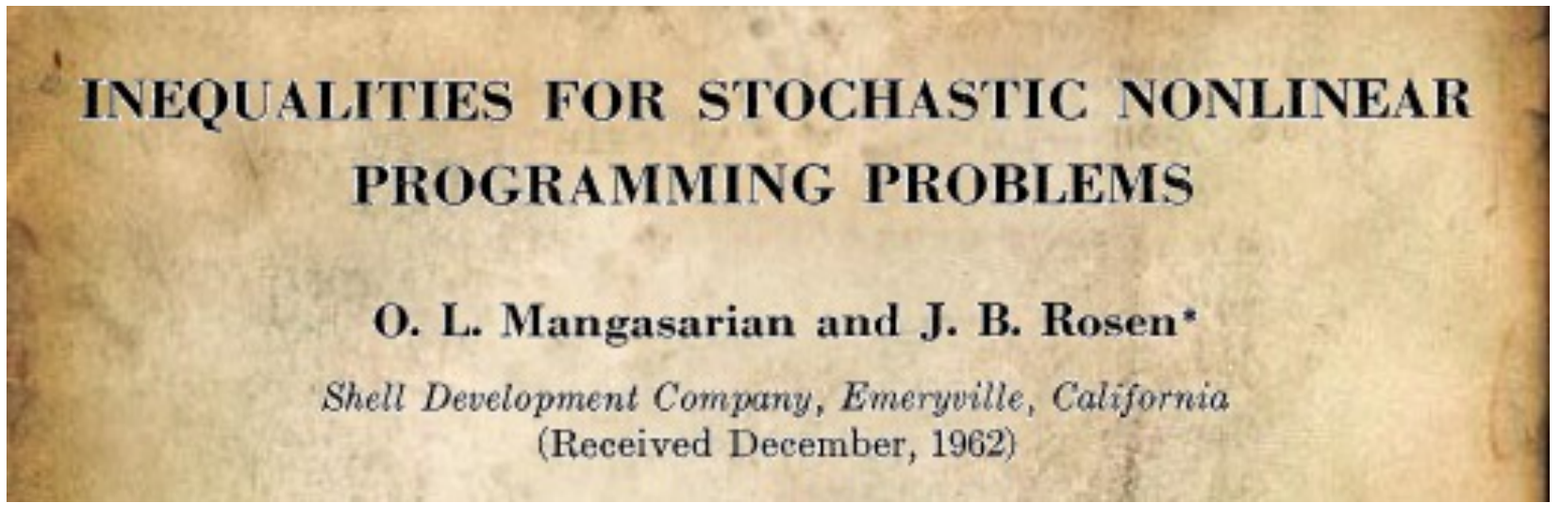
$$x_j \ge 0 \qquad j = 1, \ldots, n,$$

$$\sum_{j=1}^{n} a_{ij} x_j = a_{io} \qquad i = 1, \ldots, m,$$

$$\begin{array}{ll} \min\limits_{z} & (c_1 + x \cdot c_2)'z \\ \text{s.t.} & Gz = W \\ & z \ge 0 \end{array} \qquad x \in \mathbb{R}$$

Also extended to 2 parameters in 1955 by Gass and Saaty

MULTI-PARAMETRIC CONVEX PROGRAMMING



INEQUALITIES FOR STOCHASTIC NONLINEAR PROGRAMMING PROBLEMS

O. L. Mangasarian and J. B. Rosen*

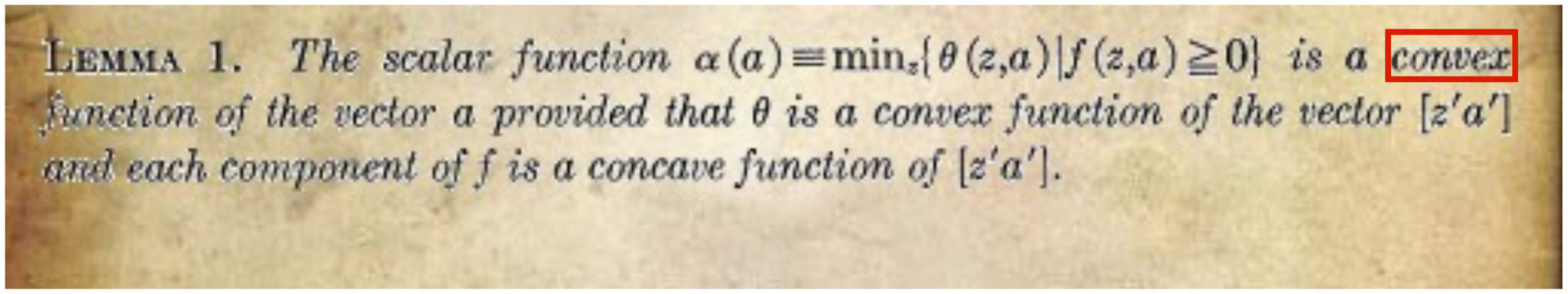
Shell Development Company, Emeryville, California

(Received December, 1962)

(Mangasarian and Rosen, 1964)

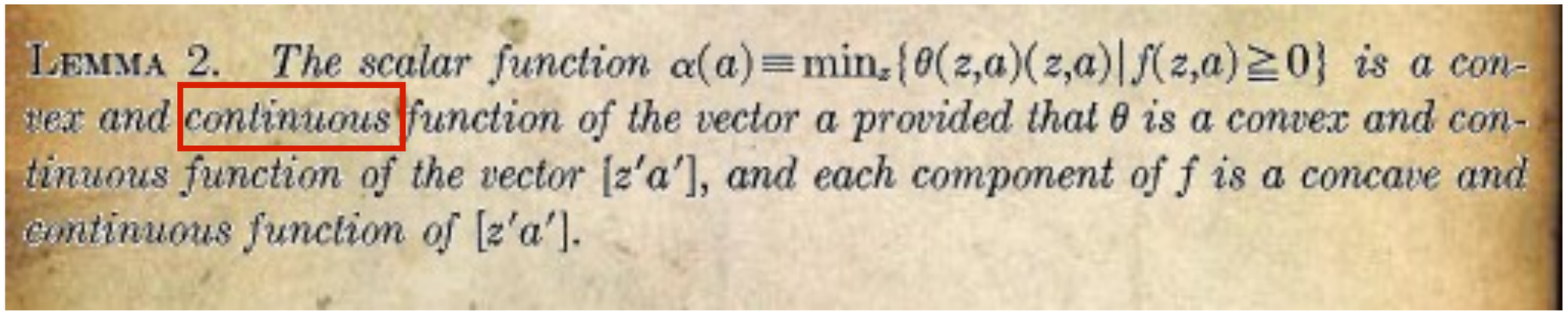
$$V^*(x) = \min_z \; h(z,x) \quad \text{s.t.} \quad g(z,x) \le 0$$

h, g **convex** in (z, x)



LEMMA 1. *The scalar function* $\alpha(a) \equiv \min_z\{\theta(z,a) | f(z,a) \geqq 0\}$ *is a convex function of the vector* a *provided that* θ *is a convex function of the vector* $[z'a']$ *and each component of* f *is a concave function of* $[z'a']$.

h, g convex in $(z, x) \Rightarrow V^*(x)$ convex



LEMMA 2. *The scalar function* $\alpha(a) \equiv \min_z\{\theta(z,a)(z,a) | f(z,a) \geqq 0\}$ *is a convex and continuous function of the vector* a *provided that* θ *is a convex and continuous function of the vector* $[z'a']$, *and each component of* f *is a concave and continuous function of* $[z'a']$.

h, g convex and continuous in $(z, x) \Rightarrow V^*(x)$ convex and continuous

MULTI-PARAMETRIC LP

MANAGEMENT SCIENCE
Vol. 18, No. 7, March, 1972
Printed in U.S.A.

MULTIPARAMETRIC LINEAR PROGRAMMING*

TOMAS GAL† AND JOSEF NEDOMA‡

The multiparametric linear programming (MLP) problem for the right-hand sides (RHS) is to maximize $z = c^T x$ subject to $Ax = b(\lambda), x \geqq 0$, where $b(\lambda)$ can be expressed in the form

$$b(\lambda) = b^* + F\lambda,$$

where F is a matrix of constant coefficients, and λ is a vector-parameter.

The multiparametric linear programming (MLP) problem for the prices or objective function coefficients (OFC) is to maximize $z = c^T(\nu)x$ subject to $Ax = b$, $x \geqq 0$, where $c(\nu)$ can be expressed in the form $c(\nu) = c^* + H\nu$, and where H is a matrix of constant coefficients, and ν a vector-parameter.

(Gal and Nedoma, 1972)

$$\begin{array}{ll} \min\limits_z & c'z \\ \text{s.t.} & Gz = W + Sx \\ & z \geq 0 \end{array}$$

$$\begin{array}{ll} \min\limits_z & (c_1 + x'c_2)'z \\ \text{s.t.} & Gz = W \\ & z \geq 0 \end{array}$$

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

MULTI-PARAMETRIC NLP - 1983

Introduction to
Sensitivity and Stability Analysis
in Nonlinear Programming

ANTHONY V. FIACCO
Operations Research Department
Institute for Management Science and Engineering
School of Engineering and Applied Science
The George Washington University
Washington, D.C.

$$\begin{aligned} \min_{z} \quad & h(z, x) \\ \text{s.t.} \quad & g(z, x) \leq 0 \end{aligned}$$

(Fiacco, 1983)

Very general treatment of multiparametric programming

MULTIPARAMETRIC PROGRAMMING ALGORITHMS

Problem	$z^*(x)$	$V^*(x)$	
mp-LP	continuous, PWA	convex (cont.), PWA	(Gal, Nedoma, 1972) (Gal 1995) (Borrelli, Bemporad, Morari, 2003)
mp-QP	continuous, PWA	convex (cont.) PWQ, C^1 (if no degen.)	(Bemporad, Morari, Dua, Pistikopoulos, 2002) (Tøndel, Bemporad, Johansen, 2003a) (Seron, De Doná, Goodwin, 2000) (Baotic, 2002)
mp-MILP	PWA	(nonconvex) PWA	(Acevedo, Pistikopoulos, 1997) (Dua, Pistikopoulos, 2000)
mp-LCP	continuous, PWA	[undefined]	(Jones, Morari, 2006) (Columbano, Fukuda, Jones, 2008)
mp-convex (mp-SDP)	PWA (approx.)	convex (approx.)	(Bemporad, Filippi. 2003)
mp-IP	PW constant	PWA	(Bemporad, 2003) (Crema, 1999)
mp-convex PWQ	PWA	convex PWQ	(Patrinos, Sarimveis, 2011)

Ways to handle *degeneracy* in mpQP/mpLP have been studied

(Tøndel, Bemporad, Johansen, 2003b) (Jones, Kerrigan, Maciejowski. 2007)

EXPLICIT MPC BASED ON LINEAR PROGRAMMING

ON-LINE VS. OFF-LINE OPTIMIZATION

$$V^*(x(t)) = \min_{\xi} \begin{bmatrix} 1 & \dots & 1 & 0 & \dots & 0 \end{bmatrix} \xi$$

$$\text{s.t.} \quad G z \leq W + S x(t)$$

$$\xi \triangleq [\epsilon^u_0 \ \dots \ \epsilon^u_{N-1} \ \epsilon^x_1 \ \dots \ \epsilon^x_N \ u'_0, \dots, u'_{N-1}]'$$

- **On-line** optimization: given $x(t)$ solve the problem at each time step t (the control law $u{=}u_0(x)$ is **implicitly** defined by the LP solver)

→ Linear Program (LP)

- **Off-line** optimization: solve the LP **for all** $x(t)$ to find the control law $u{=}u_0(x)$ **explicitly**

→ multi-parametric Linear Program (mp-LP)

MULTIPARAMETRIC LP

$$\begin{array}{ll}\min\limits_{\xi} & f'\xi \\ \text{s.t.} & G\xi \le W + Sx\end{array}$$

(Gal, Nedoma, 1972)

(Borrelli, Bemporad, Morari, 2003)

For a given parameter x_0:

- solve LP to find ξ_0^* (assume no degeneracy)

- identify active constraints $\tilde{G}\xi_0^* = \tilde{W} + \tilde{S}x,\ \tilde{G} \in \mathbb{R}^{n\times n}$ full rank

- from active constraints $\tilde{G}\xi^*(x) = \tilde{W} + \tilde{S}x$:

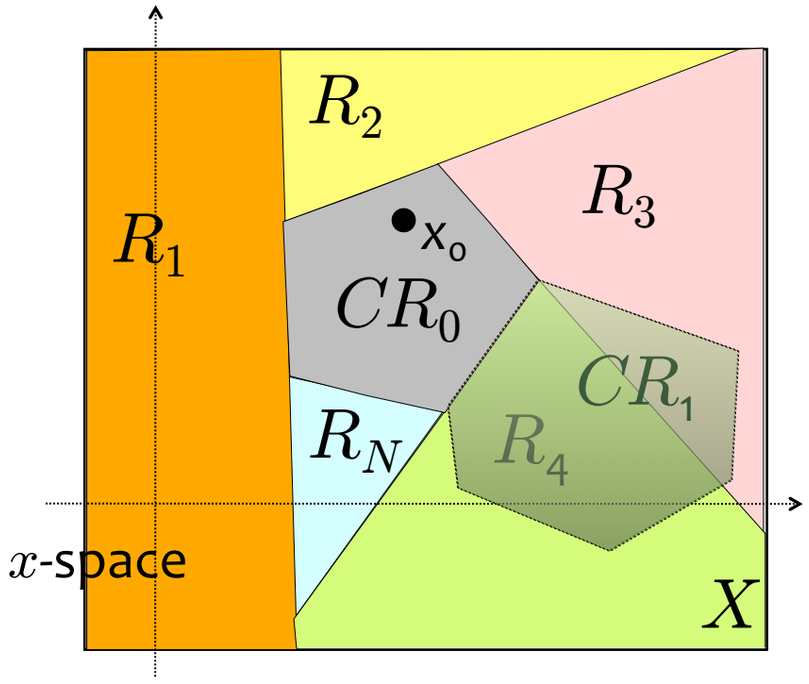
$$\xi^*(x) = \tilde{G}^{-1}(\tilde{W} + \tilde{S}x) \qquad \text{optimizer}$$

$$\hat{G}(\tilde{G}^{-1}\tilde{S})x + \hat{G}(\tilde{G}^{-1}W) \le \hat{W} + \hat{S}x \qquad \text{critical region } CR_0$$

$$V^*(x) = f'\xi^*(x) = f'\tilde{G}^{-1}(\tilde{W} + \tilde{S}x) \qquad \text{value function}$$

$$f + \tilde{G}'\tilde{\lambda} = 0 \ \Rightarrow\ \tilde{\lambda}(x) = -(\tilde{G}^{-1})'f \qquad \text{dual variables}$$

MULTIPARAMETRIC LP



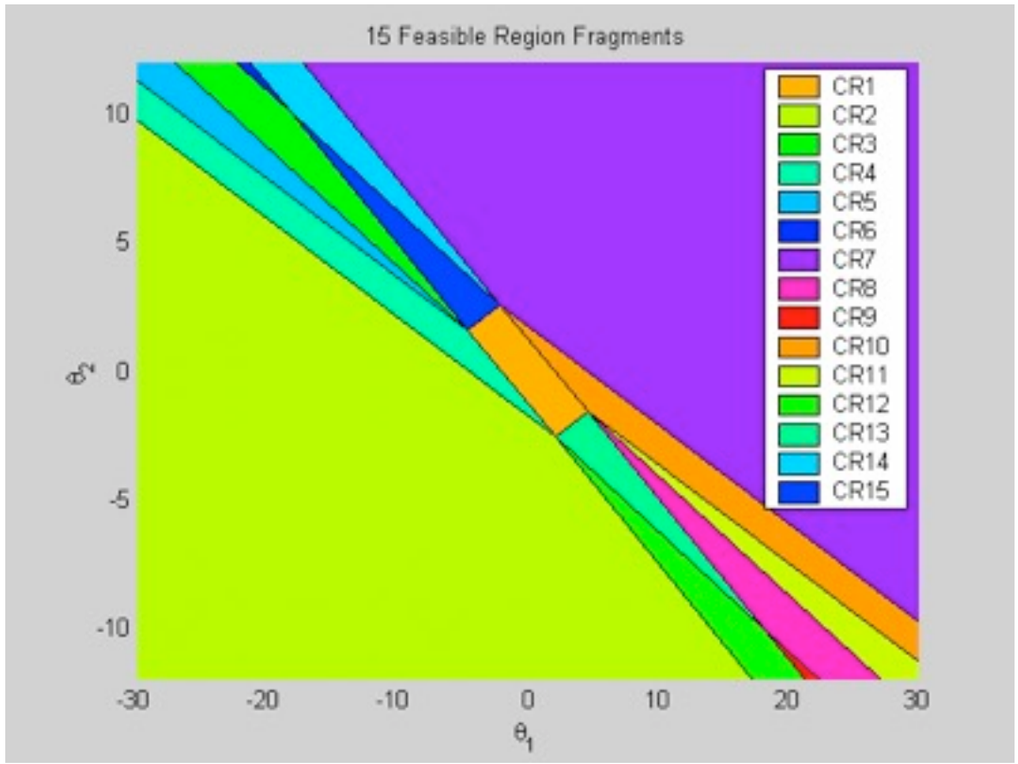
$$CR_0 = \{x \in X : \ \mathcal{A}x \leq \mathcal{B}\}$$

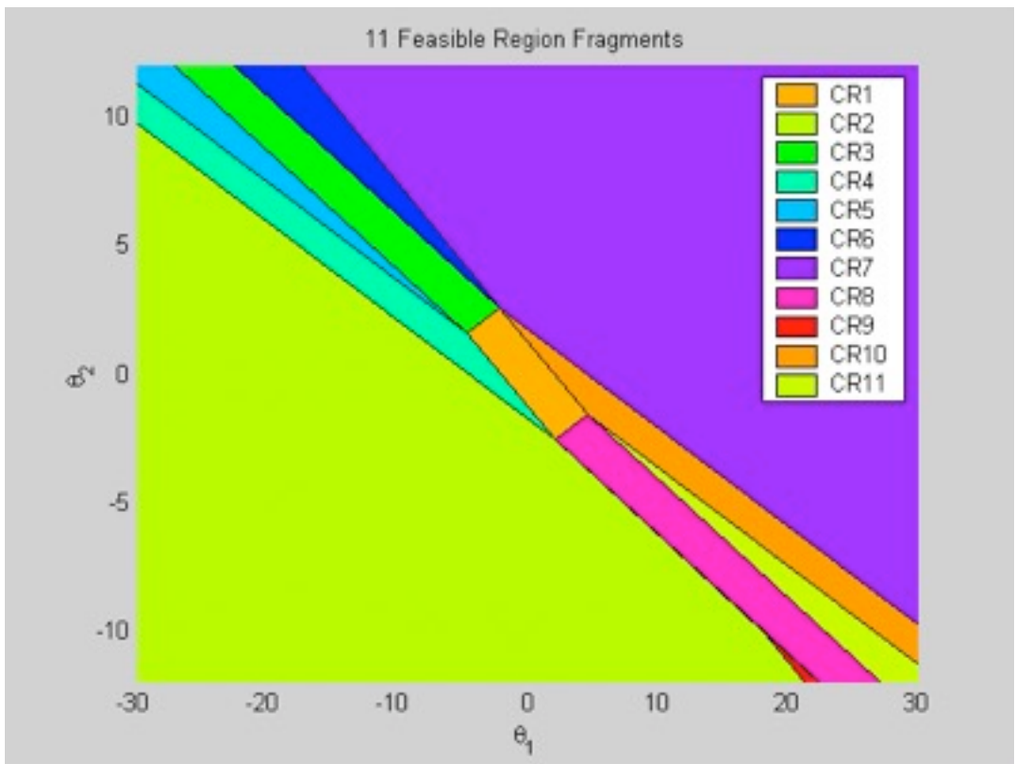
$$R_i = \{x \in X : \ \mathcal{A}^i x > \mathcal{B}^i, \ \mathcal{A}^j x \leq \mathcal{B}^j, \ \forall j < i\}$$

Note: while CR_i is characterizing a set of active constraints, R_i is not

1) Use the above splitting only as a search procedure, don't split the CR
2) Remove duplicates of CR already found

UNION OF REGIONS





$$z(x) \triangleq [\epsilon_0^u(x) \ \dots \ \epsilon_{N-1}^u(x) \ \epsilon_1^x(x) \ \dots \ \epsilon_N^x(x) \ u'_0(x) \ \dots \ u'_{N-1}(x)]'$$

Regions where the first component of the solution is the same can be joined (when their union is convex).

(Bemporad, Fukuda, Torrisi, *Computational Geometry*, 2001)

PROPERTIES OF MULTIPARAMETRIC LP

Theorem The set X^* of parameters for which the mpLP problem is feasible is a convex polyhedron. The functions $V^*(x)$ and $\xi^*(x)$ are piecewise affine and continuous over X^*, and $V^*(x)$ is also convex on X^*.

(Gal, Nedoma, 1972)
(Borrelli, Bemporad, Morari, 2003)

$$\begin{aligned}\xi^*(x) \;=\; & \arg\min f'\xi \\ & \text{s.t.} G\xi \le W + Sx\end{aligned}$$

piecewise affine, continuous
(if optimizer is always unique)

$$\begin{aligned}V^*(x) \;=\; & \min f'\xi \\ & \text{s.t.} G\xi \le W + Sx\end{aligned}$$

continuous, piecewise affine, convex

Corollary 1: The value function $V^*(x)$ is continuous piecewise affine

Corollary 2: The MPC controller is continuous piecewise affine !

TRIPLE INTEGRATOR EXAMPLE

- System:

$$y(t) = \frac{1}{s^3}u(t)$$

sampling + ZOH
T=0.1 s

$$\begin{aligned} x(t+1) &= \begin{bmatrix} 1.05 & 0.46 & 1.02 \\ 0 & 0.73 & -0.01 \\ 0 & 0.86 & 0.99 \end{bmatrix} x(t) + \begin{bmatrix} 0.15 \\ 0.86 \\ 0.45 \end{bmatrix} u(t) \\ y(t) &= \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} x(t) \end{aligned}$$

- Constraints:

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

- Control objective: minimize:

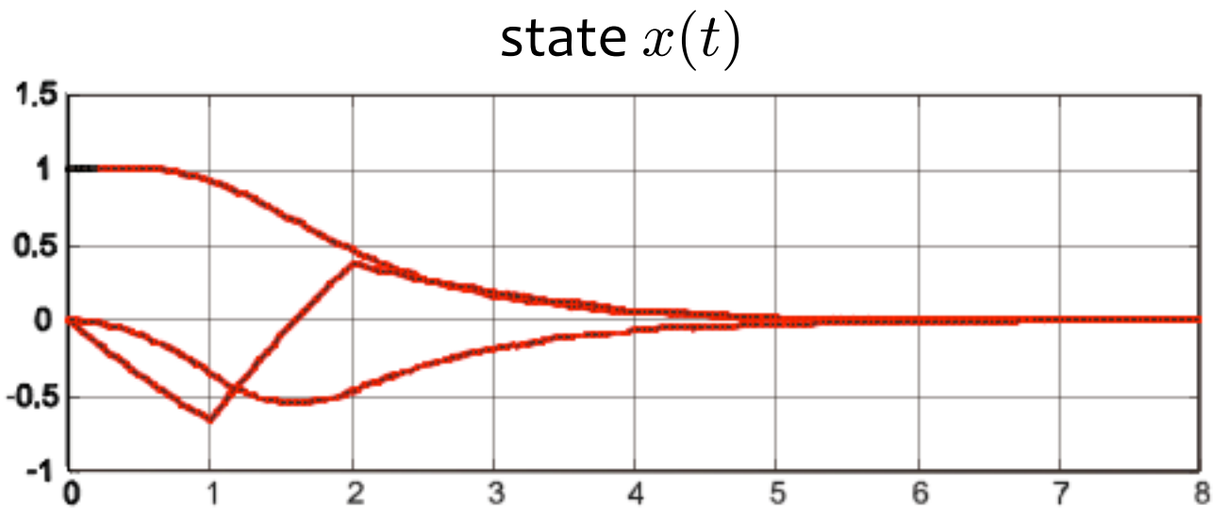
$$\sum_{k=0}^{2} |y(k)| + \left|\frac{1}{8000}u(k)\right|$$

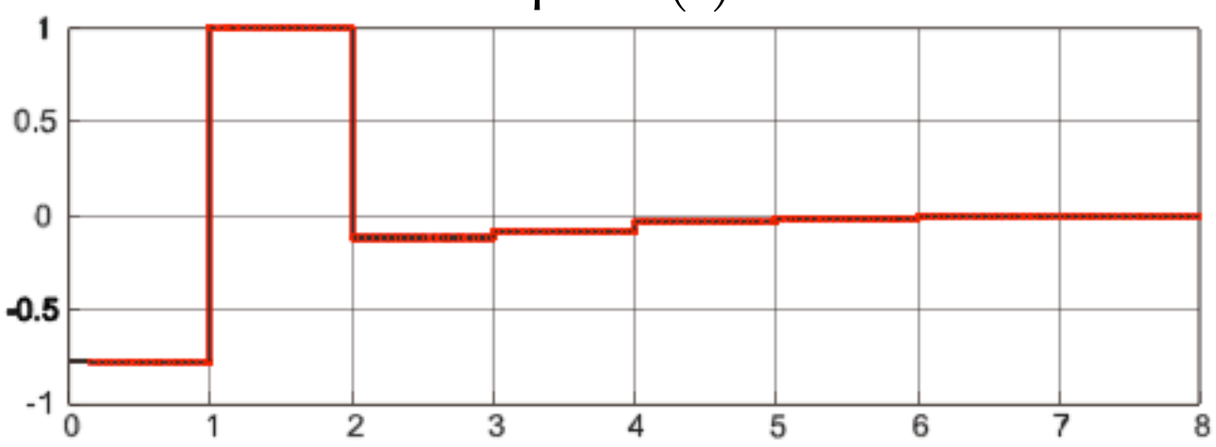
- LP problem:

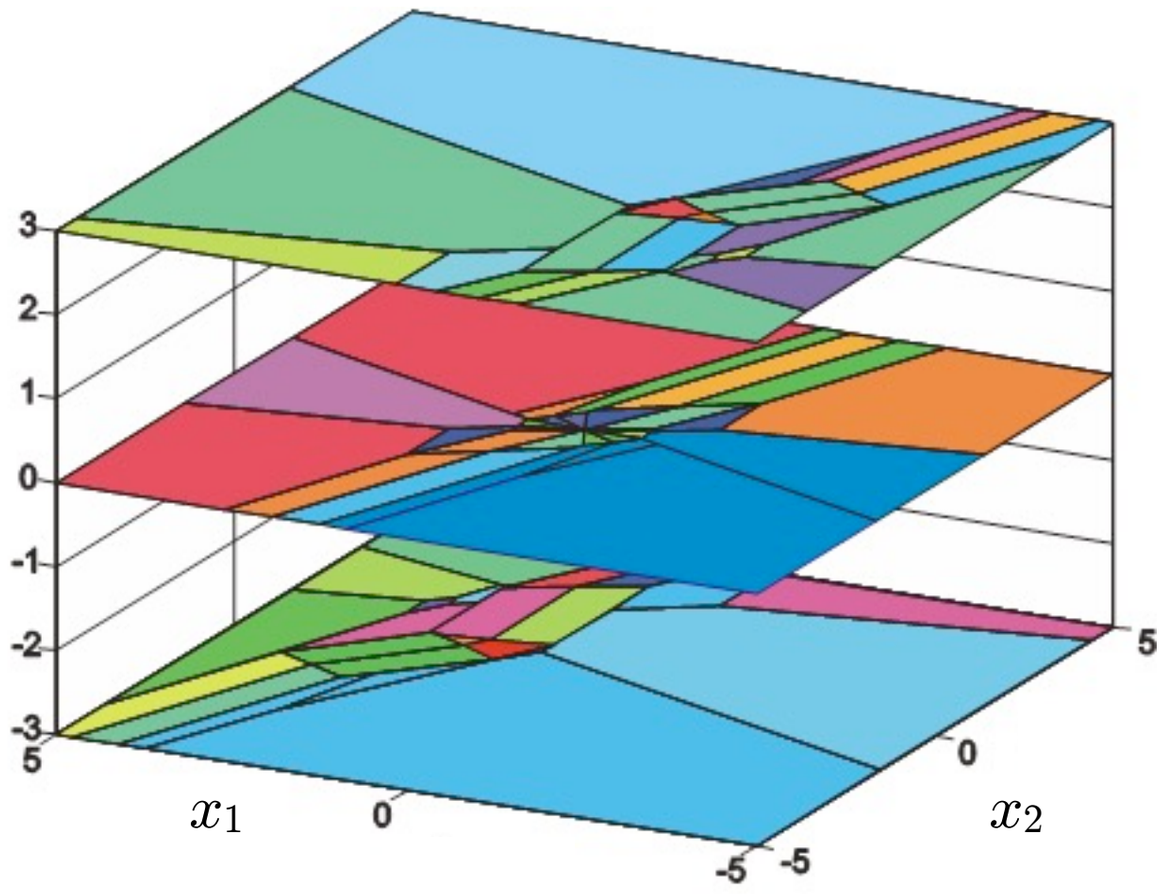
$$f = [1\ 1\ 1\ 1\ 0\ 0],$$

$$G = \begin{bmatrix} -1 & 0 & 0 & 0 & 1 & 0 \\ -1 & 0 & 0 & 0 & -1 & 0 \\ 0 & -1 & 0 & 0 & 0 & 1 \\ 0 & -1 & 0 & 0 & 0 & -1 \\ 0 & 0 & -1 & 0 & -1332.1 & 0 \\ 0 & 0 & -1 & 0 & -7319.1 & 0 \\ 0 & 0 & -1 & 0 & -3848.8 & 0 \\ 0 & 0 & -1 & 0 & 1332.1 & 0 \\ 0 & 0 & -1 & 0 & 7319.1 & 0 \\ 0 & 0 & -1 & 0 & 3848.8 & 0 \\ 0 & 0 & 0 & -1 & -8706.3 & -1332.1 \\ 0 & 0 & 0 & -1 & -5295.8 & -7319.1 \\ 0 & 0 & 0 & -1 & -10116 & -3848.8 \\ 0 & 0 & 0 & -1 & 8706.3 & 1332.1 \\ 0 & 0 & 0 & -1 & 5295.8 & 7319.1 \\ 0 & 0 & 0 & -1 & 10116 & 3848.8 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & -1 \end{bmatrix}, \quad W = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}, \quad S = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 8935.8 & 3915.4 & 8689.4 \\ 0 & 6227.3 & -146.38 \\ 0 & 7319.1 & 8423 \\ -8935.8 & -3915.4 & -8689.4 \\ 0 & -6227.3 & 146.38 \\ 0 & -7319.1 & -8423 \\ 9394 & 14467 & 17678 \\ 0 & 4436.2 & -252.3 \\ 0 & 12615 & 8220.7 \\ -9394 & -14467 & -17678 \\ 0 & -4436.2 & 252.3 \\ 0 & -12615 & -8220.7 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

EXPLICIT MPC CONTROLLER







# constraints	# variables	# critical regions
20	6	68

MAIN ISSUES IN LP-BASED MPC

- Idle control performance:

$$u^*(t) = 0 \quad x(t) \neq 0$$

– On-line optimization: Not detected

– Off-line optimization: Detected

- Multiple optima:

$$u^*(t) = \{u_1^*(t), \ldots, u_K^*(t)\}$$

– On-line optimization: Not detected

Arbitrary choice

– Off-line optimization Detected

Multiple explicit solution exploited

ROBUST (EXPLICIT) MPC

MULTIPARAMETRIC SOLUTIONS: MIN-MAX MPC

$$x_{k+1} = A(w_k)x_k + B(w_k)u_k + Ev_k$$ uncertain linear model

$$A(w) = A_0 + \sum_{i=1}^{q} A_i w_i, \quad B(w) = B_0 + \sum_{i=1}^{q} B_i w_i \qquad w, v \text{ belong to polytopes}$$

- open-loop prediction, ∞-norms: solved via mpLP $(A(w) \equiv A_0)$

  (Bemporad, Borrelli, Morari, 2003)

- closed-loop prediction, ∞-norms:

  - mpLP iterations (dynamic programming solution)

    (Bemporad, Borrelli, Morari, 2003)

  - mpLP solving single LP problem of Scokaert-Mayne

    (Kerrigan, Maciejowski, 2004)

- min-max MPC with quadratic costs

  (Ramirez, Camacho, 2006)

  (Munoz, Alamo, Ramirez, Camacho, 2007)

Explicit min-max MPC control law is piecewise affine

ROBUST OPTIMAL CONTROL PROBLEM

- Model

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

$x \in \mathbb{R}^n$
$u \in \mathbb{R}^m$

- Uncertainty

$$A(w) = A_0 + \sum_{i=1}^{q} A_i w_i, \quad B(w) = B_0 + \sum_{i=1}^{q} B_i w_i$$

$w(t) \in \mathcal{W}$ (bounded polyhedron)

$v(t) \in \mathcal{V}$ (bounded polyhedron)

Example: $\mathcal{W} = \{w : \ w_i \geq 0, \ \sum_{i=1}^{n_w} w_i = 1\}, A_0 = 0$

- Constraints:

$$Fx(t) + Gu(t) \leq f, \ \forall v(t) \in \mathcal{V}, \ \forall w(t) \in \mathcal{W}$$

Example: $u_{\min} \leq u(t) \leq u_{\max}, x_{\min} \leq x(t) \leq x_{\max}$

- Goal: find a controller that guarantees
  - **optimal worst-case** performance
  - **constraint** fulfillment **for all disturbances**

"OPEN-LOOP" WORST-CASE PERFORMANCE

Objective:

- MIN over all admissible control input sequences $\{u_0, \dots, u_{N-1}\}$
- MAX over all possible admissible disturbance sequences $\{v_0, \dots, v_{N-1}\}$, $\{w_0, \dots, w_{N-1}\}$ (=*worst-case scenario*)

$$J_{\max}(x_0, u_0, \dots, u_{N-1}) \triangleq \max_{\substack{v_0,\dots,v_{N-1} \\ w_0,\dots,w_{N-1}}} \left\{ \sum_{k=0}^{N-1} \|Qx_k\|_p + \|Ru_k\|_p + \|Px_N\|_p \right\}$$

$$\text{subj. to} \begin{cases} x_{k+1} = A(w_k)x_k + B(w_k)u_k + Ev_k \\ v_k \in \mathcal{V} \\ w_k \in \mathcal{W}, \\ k = 0, \dots, N-1 \end{cases}$$

Norm: $p = 1$, $p = \infty$

$$J_N^*(x_0) \triangleq \min_{u_0,\dots,u_{N-1}} J_{\max}(x_0, u_0, \dots, u_{N-1})$$

$$\text{subj. to} \left\{ \begin{array}{l} Fx_k + Gu_k \le f \\ x_{k+1} = A(w_k)x_k + B(w_k)u_k + Ev_k \\ x_N \in \mathcal{X}^f \\ k = 0, \dots, N-1 \end{array} \right\} \begin{array}{l} \forall v_k \in \mathcal{V}, w_k \in \mathcal{W} \\ \forall k = 0, \dots, N-1 \end{array}$$

"CLOSED-LOOP" WORST-CASE PERFORMANCE

Idea: Interleave min and max problems: (dynamic programming)

$$\min_{u_0}\left\{\max_{v_0,w_0}\left\{\dots\min_{u_{N-1}}\left\{\max_{v_{N-1},w_{N-1}} J_N\right\}\dots\right\}\right\}$$

- solve for j=N-1,…,0

parameters unknowns

$$J_{\max,j}(x_j,u_j) \triangleq \max_{v_j\in\mathcal{V},\ w_j\in\mathcal{W}} \{\|Qx_j\|_p + \|Ru_j\|_p + J^*_{j+1}(A(w_j)x_j + B(w_j)u_j + Ev_j)\}$$

$$J^*_j(x_j) \triangleq \min_{u_j} J_{\max,j}(x_j,u_j)$$

unknowns

$$\text{subj. to } \left\{\begin{array}{l} Fx_j + Gu_j \le f \\ A(w_j)x_j + B(w_j)u_j + Ev_j \in \mathcal{X}^{j+1} \end{array}\right\} \forall v_j\in\mathcal{V}, w_j\in\mathcal{W}$$

parameters

- where

$$\mathcal{X}^j = \left\{x\in\mathbb{R}^n |\ \exists u: \begin{array}{l} 1.\ Fx+Gu\le f \\ 2.\ A(w)x+B(w)u+Ev\in\mathcal{X}^{j+1}, \forall v\in\mathcal{V}, w\in\mathcal{W}\end{array}\right\}$$

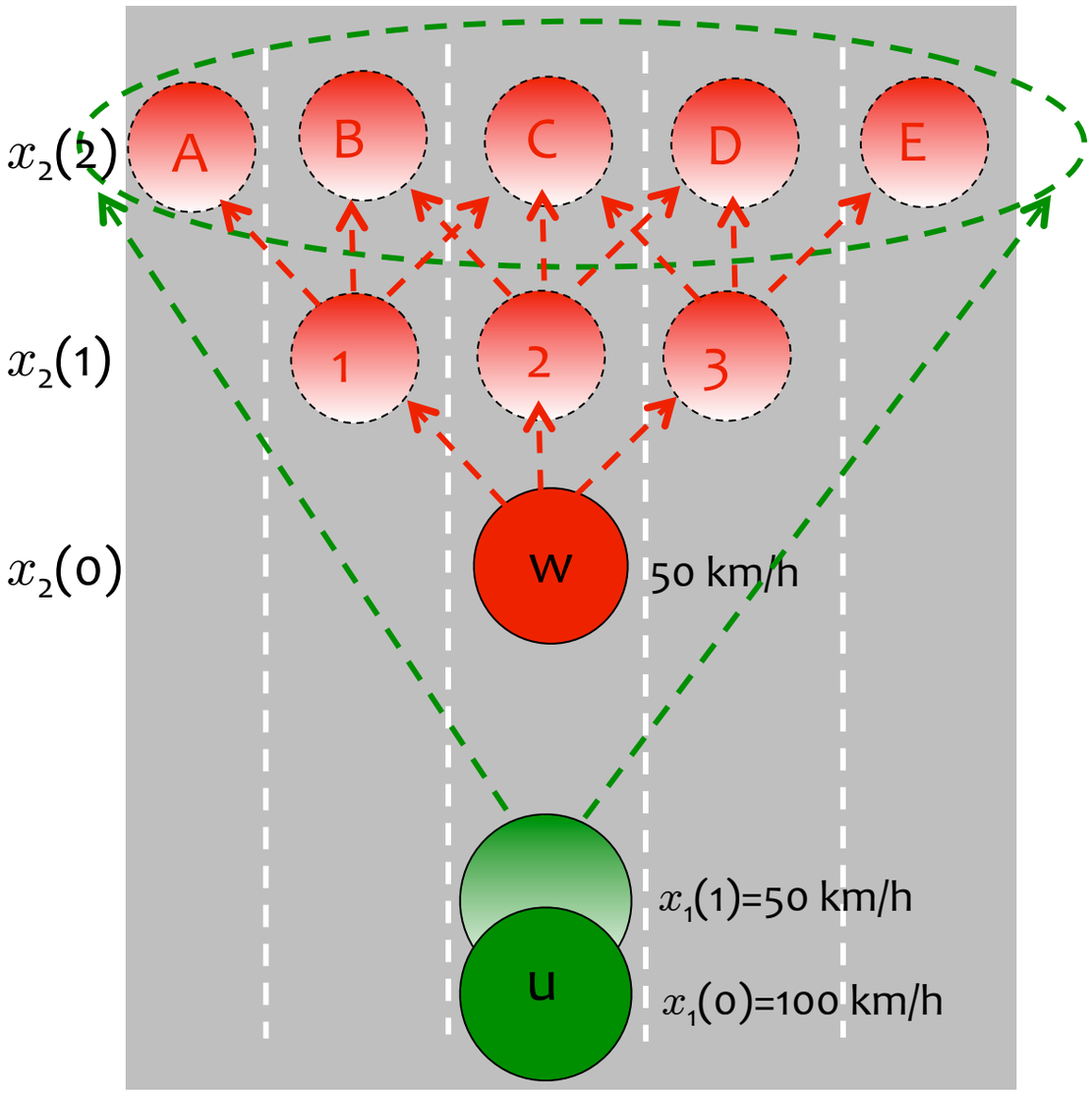
- with boundary condition

$$\begin{array}{rcl} J^*_N(x_N) & = & \|Px_N\|_p \\ \mathcal{X}^N & = & \mathcal{X}^f \end{array}$$

CL VS. OL: AN INTUITIVE EXAMPLE

$$\min_{u_0,\dots,u_{N-1}} \left\{ \max_{v_0,\dots,v_{N-1},w_0,\dots,w_{N-1}} J(\cdot) \right\}$$

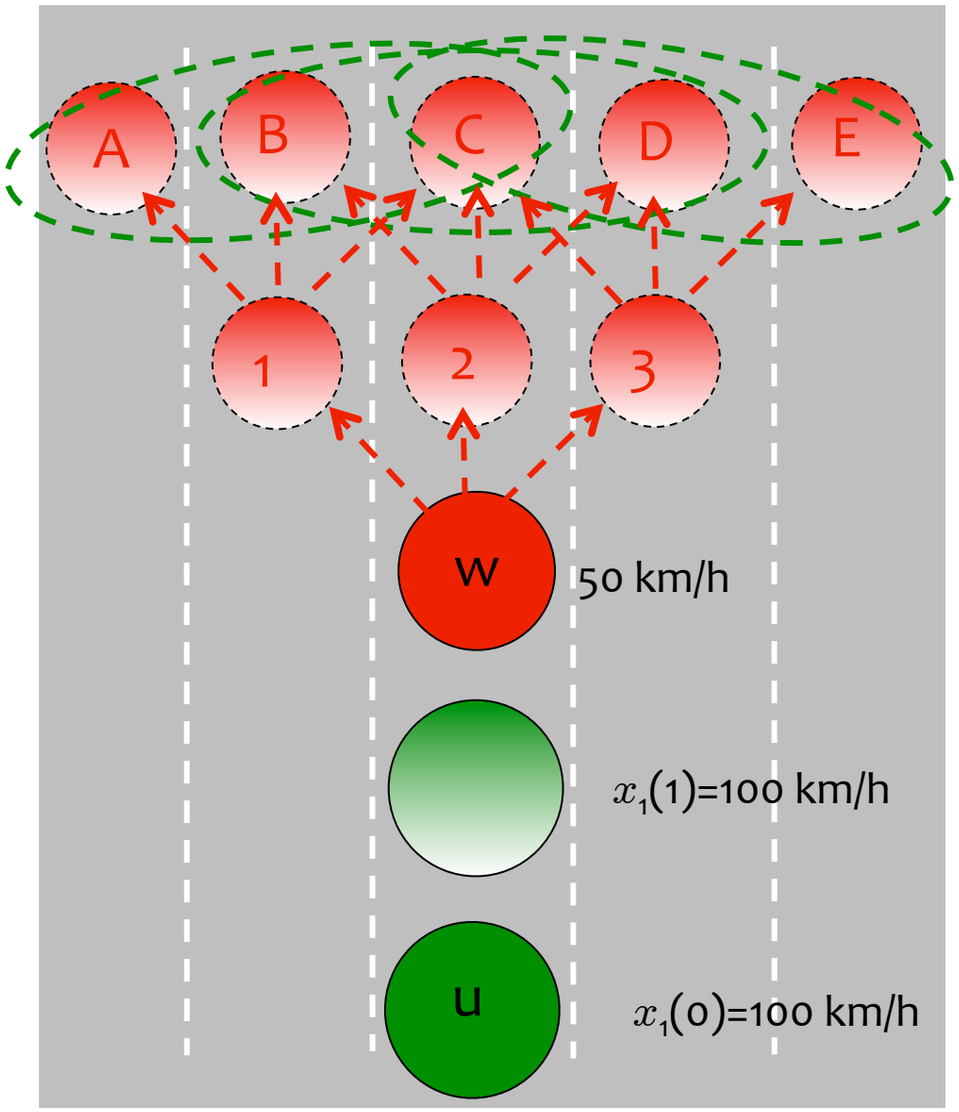
"Open-Loop" worst-case



better slow down: Car w can be anywhere at time k=2.

$$\min_{u_0} \left\{ \max_{v_0,w_0} \left\{ \dots \min_{u_{N-1}} \left\{ \max_{v_{N-1},w_{N-1}} J_N(\cdot) \right\} \dots \right\} \right\}$$

"Closed-Loop" worst-case



no need to slow down: Whatever w does, a feasible command u will be possible at time k=1

$u(1)$ will know $x_2(1)$, and hence $w(0)$

OPEN-LOOP VS. CLOSED-LOOP

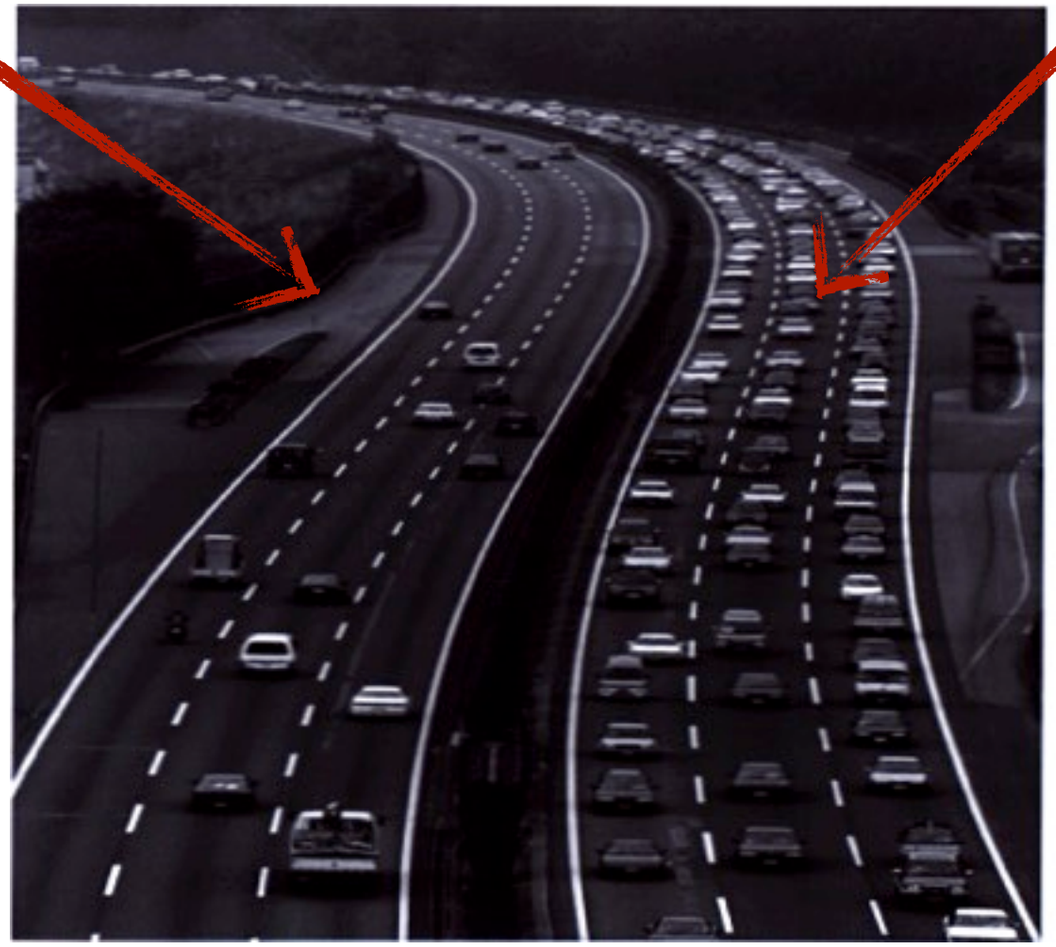
- **Open-Loop** Prediction Strategy
  - The whole disturbance sequence plays first:
    ⇒ more conservative, the benefits of future feedback are not exploited !
  - Consequences:
    — poor performance
    — Uncertainty grows over time ⇒ even possible infeasibility !

- **Closed-Loop** Prediction Strategy
  - Disturbance and input play one move at a time
  - $u^*(k)$ is a function of the current state $x(k)$
    ⇒ depends on previous **disturbances**

OPEN-LOOP VS. CLOSED-LOOP PREDICTION (EXAMPLE)

"Closed-loop" worst-case lane

"Open-loop" worst-case lane



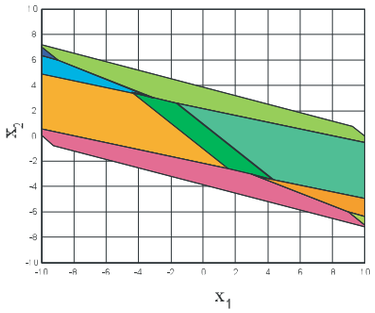
ROBUST EXPLICIT MPC - MAIN RESULTS

(Bemporad, Borrelli, Morari, IEEE TAC, 2003)

Structure

Theorem 1 *The solution to the Closed-Loop Constrained Robust Optimal Control problem is a* state-feedback, **continuous**, *and* **piecewise affine** *control law of the form*

$$u_k^*(x_k) = \begin{array}{l} F_i^k x_k + g_i^k, \ \text{if} \\ x_k \in \mathcal{X}_i^k \triangleq \{x: \ T_i^k x \leq S_i^k\}, \ i = 1, \ldots, s_k \end{array}$$



where $\{\mathcal{X}_i^k\}_{i=1}^{s_k}$ *is a partition of the set* $\mathcal{X}^k$ *of feasible states* x_k.

Solvability

Theorem 2 *The solution to the Closed-Loop Constrained Robust Optimal Control problem can be obtained by solving* N *multiparametric linear programs.*

Open-loop formulation:

Theorem 3 *The optimal sequence of inputs of the Open-Loop Constrained Robust Optimal Control problem (with* $A(w) = A$*) is a* **continuous** *and* **piecewise affine** *vector function of the initial state* x_0.

EXAMPLE

- System

$$x(t+1) = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} x(t) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(t) + \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} v(t)$$

- Uncertainty

$$|v_1|, |v_2| \leq 1.5$$

- Performance

$$\|Px_N\|_\infty + \sum_{k=0}^{N-1} (\|Qx_k\|_\infty + |Ru_k|)$$

$$N = 4,\ P = Q = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix},\ R = 1.8$$

- Constraints

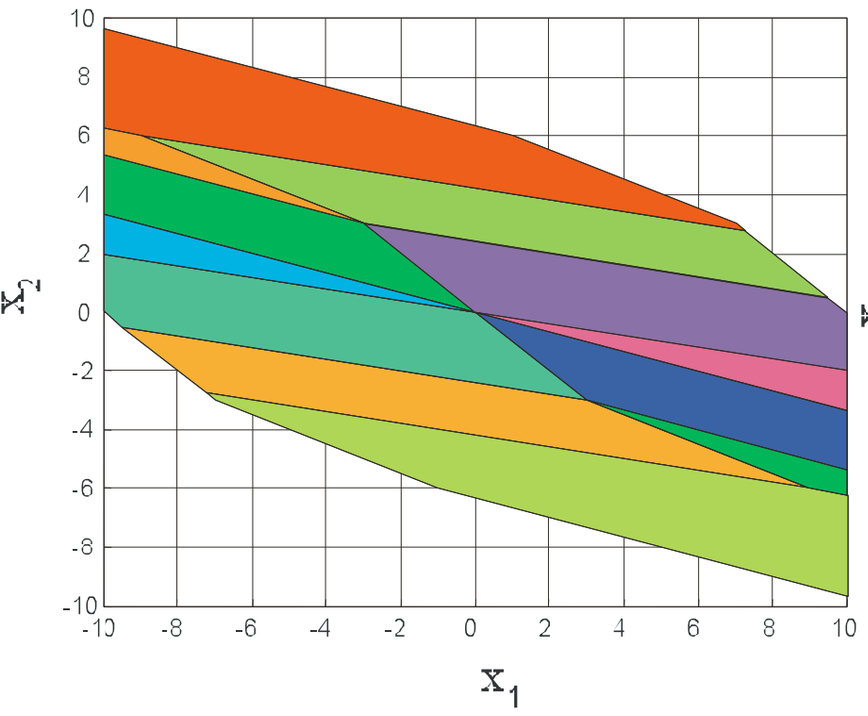
$$\begin{array}{l} \text{input constraints: } -3 \leq u_k \leq 3, \quad k = 0, \ldots, 3 \\ \text{state constraints: } -10 \leq x_k \leq 10, \quad k = 0, \ldots, 4 \end{array}$$

- Implementation: receding horizon (=robust MPC)

$$\begin{array}{rl} u(t) = & F_i^0 x(t) + g_i^0, \ \text{if} \\ & T_i^0 x(t) \leq S_i^0, \ i = 1, \ldots, s_0 \end{array}$$

MPC CONTROL LAWS

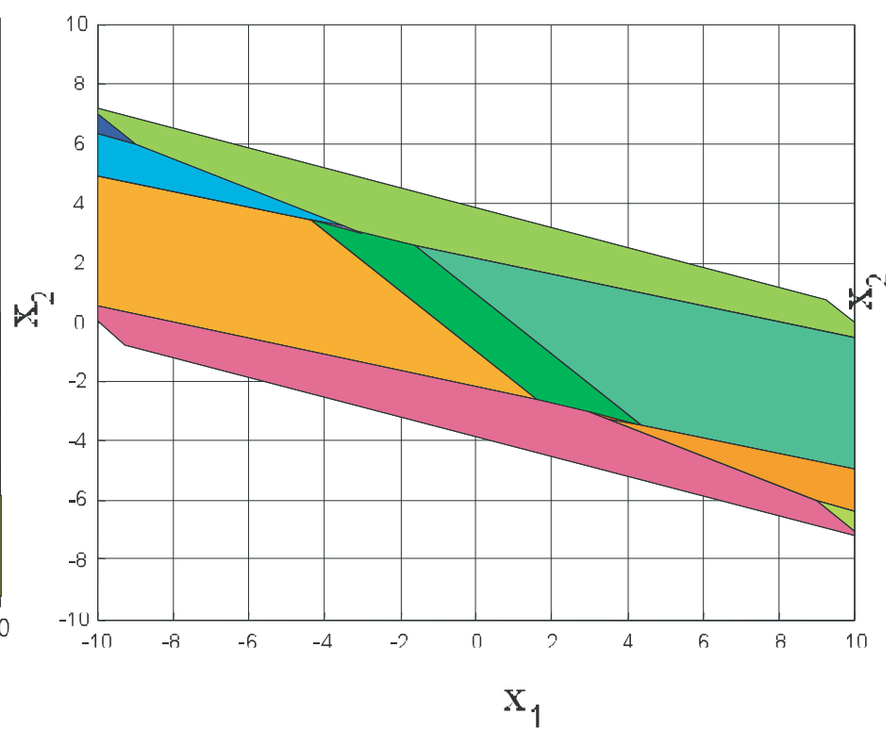
Nominal (v=0)



CPU time: 23 s

12 regions

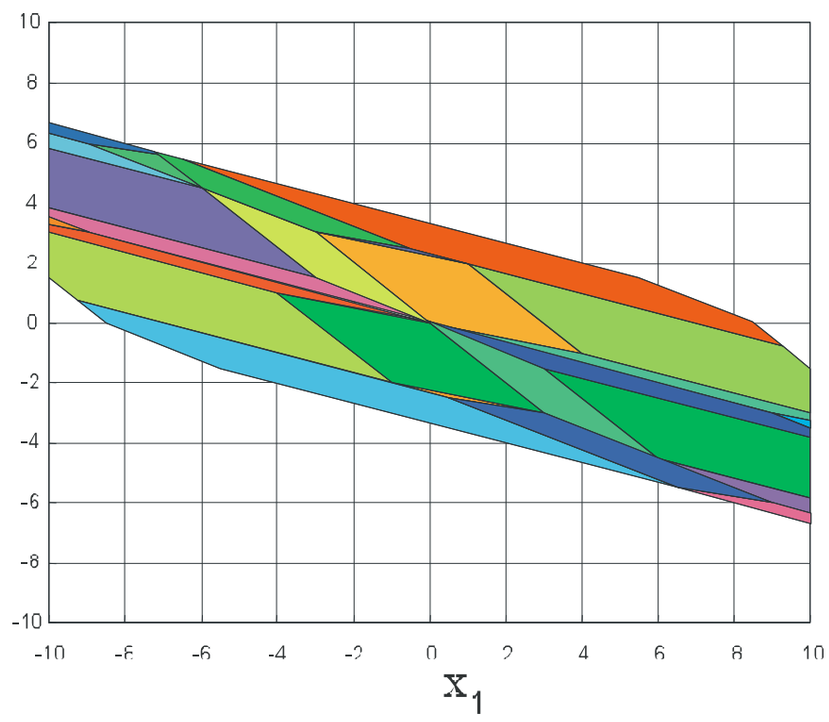
Closed-loop



CPU time: 53 s

12 regions

Open-loop



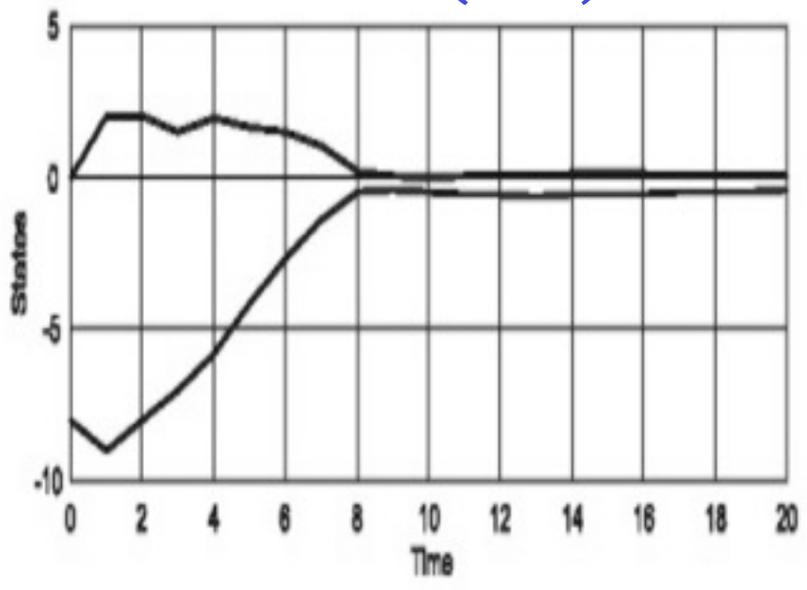
CPU time: 582 s

24 regions

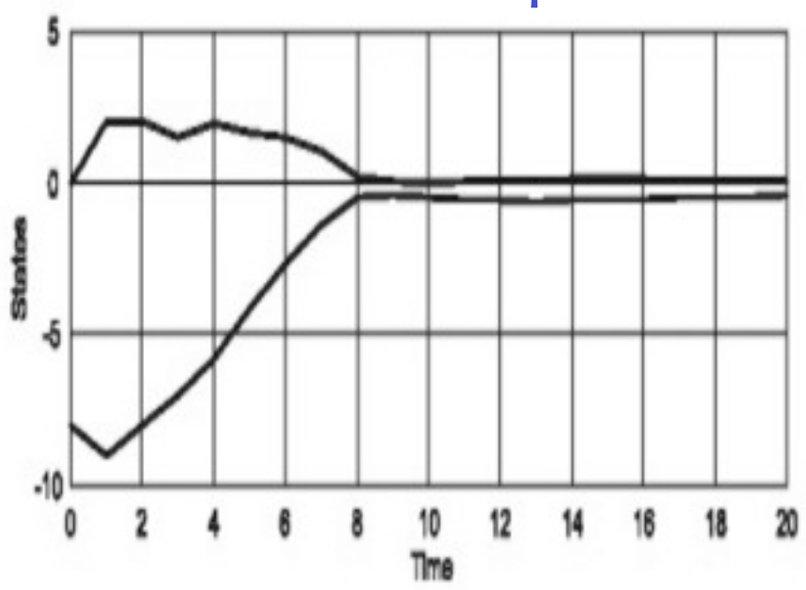
(CPU time: Matlab 5.3, P-III 800 — Off-line computations !)

TRAJECTORIES

Nominal (v=0)

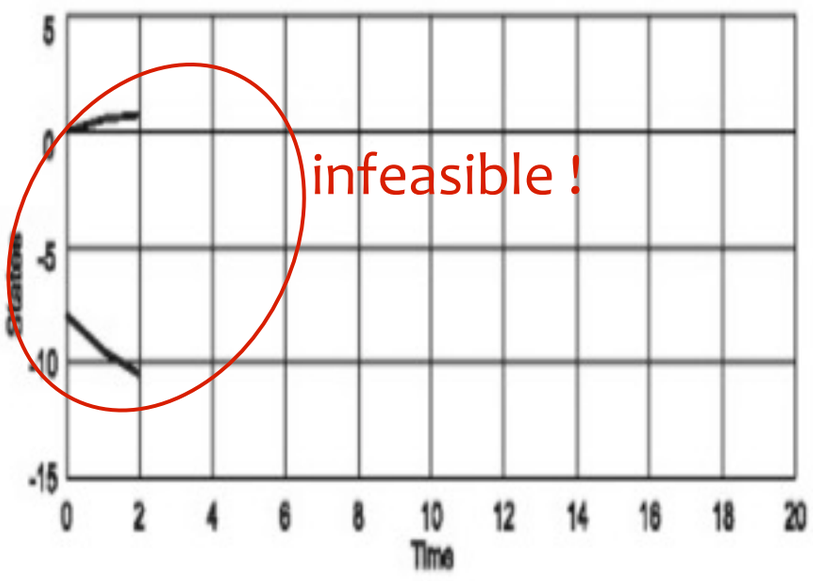


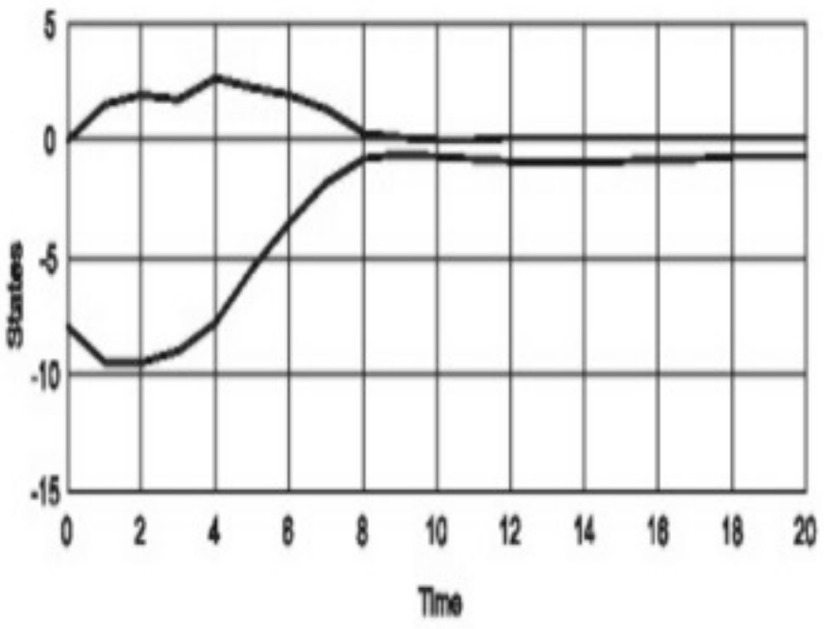
Closed-loop

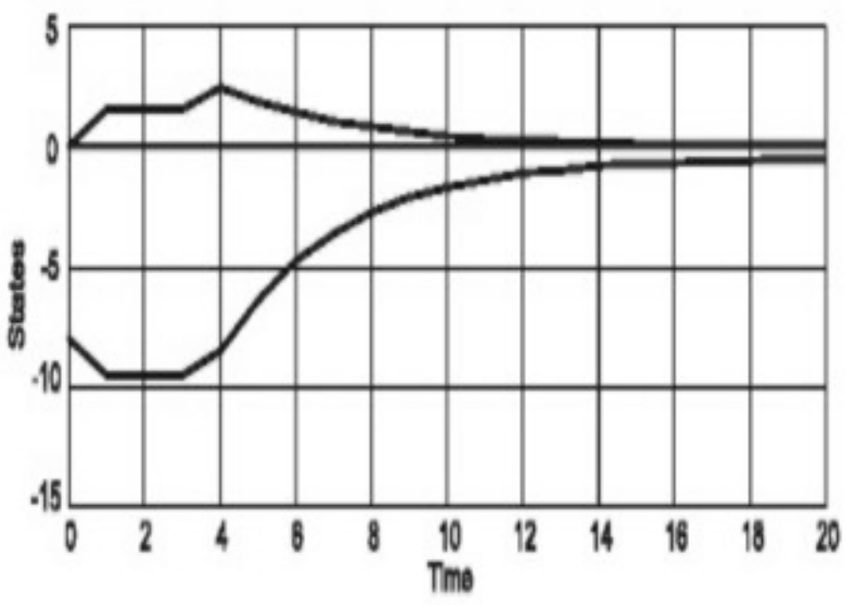


Open-loop

disturbance #1







disturbance #2