

# MODEL PREDICTIVE CONTROL

## QUADRATIC PROGRAMMING AND EXPLICIT MPC

Alberto Bemporad

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

# COURSE STRUCTURE

- ✓ Linear model predictive control (MPC)
- ✓ Linear time-varying and nonlinear MPC
  - MPC computations: quadratic programming (QP), explicit MPC
  - Hybrid MPC
  - Stochastic MPC
  - Data-driven MPC

## MATLAB Toolboxes:

- **MPC Toolbox** (linear/explicit/parameter-varying MPC)
- **Hybrid Toolbox** (explicit MPC, hybrid systems)

## Course page:

[http://cse.lab.imtlucca.it/~bemporad/mpc\\_course.html](http://cse.lab.imtlucca.it/~bemporad/mpc_course.html)

# PROS AND CONS OF MPC

- Extremely flexible control design approach:

- Prediction model: **multivariable**, w/**delays**, **time-varying**, **uncertain**, **stochastic**, **nonlinear**, ...
- Handles **constraints** on inputs and outputs
- Can exploit **preview** on future references and measured disturbances
- Tuning** similar to LQR (Linear Quadratic Regulator)
- Mature **offline design tools** and **real-time code** are available

- Price to pay:

- Requires a (simple) **model** (experiments + system identification, linearization, ... )
- Many **design/calibration knobs** (weights, horizon, constraints, ... )
- Requires **real-time computations** to solve the optimization problem

$$\begin{aligned} \min & \sum_{k=0}^{N-1} \|W^y(y_k - r(t))\|_2^2 \\ & + \|W^u(u_k - u_r(t))\|_2^2 \\ \text{s.t. } & x_{k+1} = f(x_k, u_k) \\ & y_k = g(x_k, u_k) \\ & h(u_k, x_k, y_k) \leq 0 \end{aligned}$$

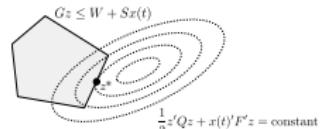
# QUADRATIC PROGRAMMING (QP) SOLVERS FOR MPC

# EMBEDDED LINEAR MPC AND QUADRATIC PROGRAMMING

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

$$\begin{array}{ll}\min_z & \frac{1}{2} z' Q z + x'(t) F' z + \frac{1}{2} x'(t) Y x(t) \\ \text{s.t.} & G z \leq W + S x(t)\end{array}$$

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



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

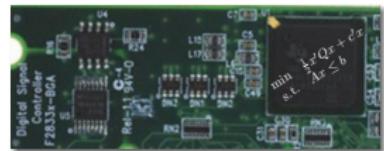
- Which QP algorithms are suitable for implementation in **embedded systems** ?

# MPC IN A PRODUCTION ENVIRONMENT

Key requirements for deploying MPC in production:

## 1. speed (throughput)

- worst-case execution time less than sampling interval
- also fast on average (to free the processor to execute other tasks)



## 2. limited memory and CPU power (e.g., 150 MHz / 50 kB)



## 3. numerical robustness (single precision arithmetic)



## 4. certification of worst-case execution time



## 5. code simple enough to be validated/verified/certified (library-free C code, easy to check by production engineers)

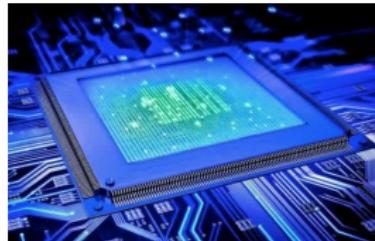


# EMBEDDED SOLVERS IN INDUSTRIAL PRODUCTION

- Multivariable MPC controller
- Sampling frequency = 40 Hz (= 1 QP solved every 25 ms)
- Vehicle operating  $\approx 1$  hr/day for  $\approx 360$  days/year on average
- Controller running on 10 million vehicles

$\sim \$20,000,000,000,000$  QP/yr

and none of them should fail.

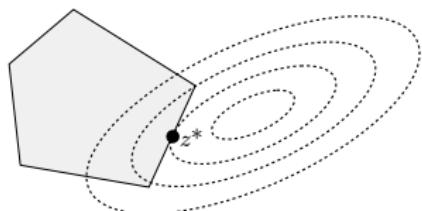


# SOLUTION METHODS FOR QP

- Most used algorithms for solving QP problems:
  - **active set** methods
  - **interior-point** methods
  - **gradient projection** methods
  - **alternating direction method of multipliers (ADMM)**
  - ...

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

Quadratic Program (QP)



- A useful performance comparisons of many solvers can be found at  
<http://plato.la.asu.edu/bench.html>
- Hybrid toolbox: `>> x=qpsol(Q,f,A,b,VLB,VUB,x0,solver)`

# KKT OPTIMALITY CONDITIONS FOR QP

- Quadratic programming problem

$$\begin{aligned} \min_z \quad & \frac{1}{2} z' Q z + x' F' z \\ \text{s.t.} \quad & G z \leq W + S x \\ & E z = f \end{aligned}$$



William Karush  
(1917–1997)

- Karush-Kuhn-Tucker (KKT) conditions:

$$\begin{aligned} & Qz + Fx + G'\lambda + E'\nu = 0 \\ & Ez = f \\ & Gz \leq W + Sx \\ & \lambda \geq 0 \\ & \lambda'(Gz - W - Sx) = 0 \end{aligned}$$



Harold W. Kuhn  
(1925–2014)



Albert W. Tucker  
(1905–1995)

- Necessary and sufficient conditions for optimality ( $Q \succeq 0$ )

# GRADIENT PROJECTION METHOD

(Levitin, Poljak, 1965) (Goldstein, 1964)

- Optimization problem:

$$\min_{z \in Z} f(z)$$

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

- $f$  convex and with Lipschitz continuous gradient

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

- Gradient projection algorithm:

$$z_{k+1} = \mathcal{P}_Z \left( z_k - \frac{1}{L} \nabla f(z_k) \right) \quad z_0 = \text{initial guess}$$

- $Z$  = convex set with an easy projection  $\mathcal{P}_Z(z) = \arg \min_{v \in Z} \|v - z\|_2^2$

Example:  $Z = \{z \in \mathbb{R}^s : z \geq 0\} \rightarrow \mathcal{P}_Z(z) = \max\{z, 0\}$

- Convergence rate:

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

- Special case of proximal gradient algorithm (or forward-backward splitting)

# GRADIENT PROJECTION FOR BOX-CONSTRAINED QP

- Convex box-constrained QP

$$\begin{aligned} \min \quad & \frac{1}{2} z' Q z + x' F' z \\ \text{s.t.} \quad & \ell \leq z \leq u \end{aligned}$$

- Since  $\|\nabla f(x) - \nabla f(y)\| = \|Q(x - y)\| \leq \lambda_{\max}(Q)\|x - y\|$  we can choose any  $L \geq \lambda_{\max}(Q)$

Examples:  $L = \lambda_{\max}(Q)$ ,  $L = \sqrt{\sum_{i,j=1}^m |Q_{i,j}|^2}$  (Frobenius norm)

- The **gradient projection method for box-constrained QP** is

$$z^{k+1} = \max\{\ell, \min\{u, z^k - \frac{1}{L}(Qz^k + Fx)\}\}$$

- If  $Q \succ 0$ , then  $f$  is strongly convex with parameter  $\lambda_{\min}(Q)$  and

$$\|z^k - z^*\|_2^2 \leq \left(1 - \frac{1}{\text{cond}(Q)}\right)^k \|z^0 - z^*\|_2^2 \quad \text{linear convergence}$$

# DUAL GRADIENT PROJECTION FOR QP

- Consider the strictly convex QP and its dual

$$\begin{array}{ll} \min & \frac{1}{2}z'Qz + x'F'z \\ \text{s.t.} & Gz \leq W + Sx \end{array} \quad \longrightarrow \quad \begin{array}{ll} \min & \frac{1}{2}y'Hy + (Dx + W)'y \\ \text{s.t.} & y \geq 0 \end{array}$$

with  $H = GQ^{-1}G'$ ,  $D = S + GQ^{-1}F$ . Take  $L \geq \frac{1}{\lambda_{\max}(H)}$

- By applying the proximal gradient method to the dual problem we get the **dual gradient projection method for QP**

$$y^{k+1} = \max\left\{y^k - \frac{1}{L}(Hy^k + Dx + W), 0\right\} \quad y_0 = 0$$

- The primal solution is related to the dual solution by

$$z^k = -Q^{-1}(Fx + G'y^k)$$

# ACCELERATED GRADIENT PROJECTION METHOD

(Nesterov, 1983) (Beck, Teboulle, 2008)

- The **accelerated (or fast) gradient projection** iterates the following

$$\begin{aligned}s^{k+1} &= z^k + \beta_k(z^k - z^{k-1}) \\ z^{k+1} &= \mathcal{P}_Z(s^{k+1} - \lambda_k \nabla f(s^{k+1}))\end{aligned}$$

*extrapolation step*

- Possible choices for  $\beta_k$  (with  $\beta_0 = 0$ ) are for example

$$\beta_k = \frac{k-1}{k+2}, \quad \beta_k = \frac{k}{k+3}, \quad \left\{ \begin{array}{lcl} \beta_k & = & \frac{\alpha_k}{\alpha_{k-1}} - \alpha_k \\ \alpha_{k+1} & = & \frac{1}{2}(\sqrt{\alpha_k^4 + 4\alpha_k^2} - \alpha_k^2) \\ \alpha_0 & = & \alpha_{-1} = 1 \end{array} \right.$$

- Thanks to adding the “momentum term”  $s^k$  the initial error  $f(z^0) - f(z^*)$  reduces as  $1/k^2$
- Fast gradient projection method for box-constrained QP:**

$$\begin{aligned}s^{k+1} &= z^k + \beta_k(z^k - z^{k-1}) \\ z^{k+1} &= \max\{\ell, \min\{u, s^{k+1} - \lambda(Qs^{k+1} + Fx)\}\}\end{aligned}$$

# FAST GRADIENT PROJECTION FOR (DUAL) QP

(Nesterov, 1983) (Patrinos, Bemporad, 2014)

- The **fast gradient method** is applied to solve the dual QP problem

$$\begin{array}{ll}\min_z & \frac{1}{2} z' Q z + x' F' z \\ \text{s.t.} & Gz \leq W + Sx\end{array}$$

$$\begin{array}{lcl}K & = & Q^{-1}G' \\ g & = & Q^{-1}Fx \\ L & \geq & \frac{1}{\lambda_{\max}(GQ^{-1}G')}\end{array}$$

$$\beta_k = \max\left\{\frac{k-1}{k+2}, 0\right\}$$

$$\begin{aligned}w^k &= y^k + \beta_k(y^k - y^{k-1}) \\ z^k &= -Kw^k - g \\ s^k &= \frac{1}{L}Gz^k - \frac{1}{L}(W + Sx) \\ y^{k+1} &= \max\{w^k + s^k, 0\}\end{aligned}$$

```
while k<maxiter
    beta=max((k-1)/(k+2),0);
    w=y+beta*(y-y0);
    z=-(iMG*w+iMC);
    s=GL*z-bL;
    y0=y;

% Termination
if all(s<=epsGL)
    gapL=-w'*s;
    if gapL<=epsVL
        return
    end
end

y=w+s;
k=k+1;
end
```

- Very **simple to code**

# FAST GRADIENT PROJECTION FOR (DUAL) QP

- **Termination criteria:** when the following two conditions are met

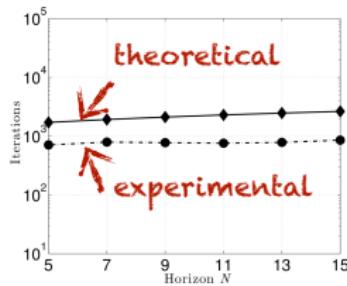
$$\begin{aligned}s_i^k &\leq \frac{1}{L}\epsilon_G, \quad i = 1, \dots, m \\ -(w^k)'s^k &\leq \frac{1}{L}\epsilon_f\end{aligned}$$

primal feasibility  
optimality

the solution  $z^k = -Kw^k - Jx$  satisfies  $G_i z^k - W_i - S_i x \leq \epsilon_G$  and, if  $w^k \geq 0$ ,

$$f(z^k) - f(z^*) \leq f(z^k) - \underbrace{q(w^k)}_{\text{dual fcn}} = -(w^k)'s^k L \leq \epsilon_f$$

- Convergence rate:  $f(x^k) - f(x^*) \leq \frac{2L}{(k+2)^2} \|z_0 - z^*\|_2^2$
- Tight bounds on maximum number of iterations
- Can be useful to warm-start active-set methods
- Extended to **mixed-integer quadratic programming (MIQP)** (Naik, Bemporad, 2017)



# RESTART IN FAST GRADIENT PROJECTION

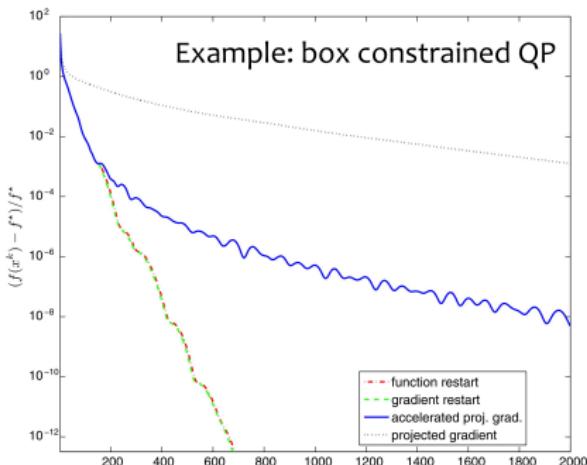
- Fast gradient projection methods can be sped up by adaptively restarting the sequence of coefficients  $\beta_k$  (O'Donoghue, Candés , 2013)
- Restart conditions:

- **function restart** whenever

$$f(y^k) > f(y^{k-1})$$

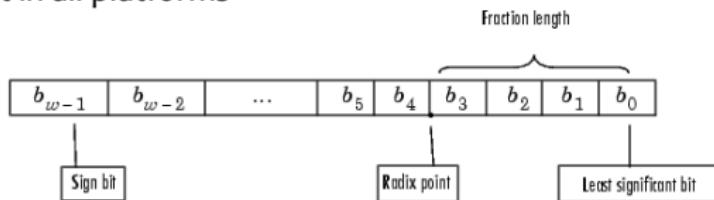
- **gradient restart** whenever

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



# 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



- Drawbacks of fixed-point arithmetics
  - Accumulation of quantization errors
  - Limited range of numbers (numerical overflow)

# GRADIENT PROJECTION IN FIXED-POINT ARITHMETICS

(Patrinos, Guggiani, Bemporad, 2013)

- Asymptotic feasibility

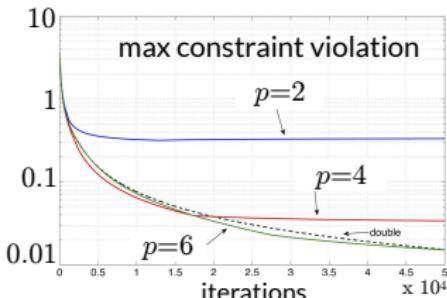
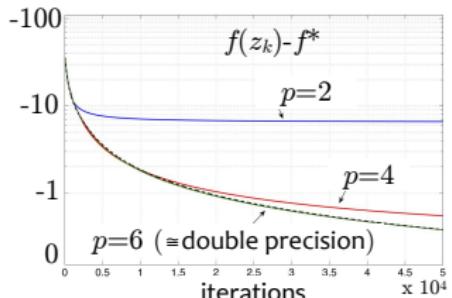
$$\max_i g_i(z_k) \leq \frac{2LD^2}{k+1} + L_v \epsilon_z^2 + 4D\epsilon_\xi$$

- Asymptotic optimality

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

- Design guidelines  
(precision)

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

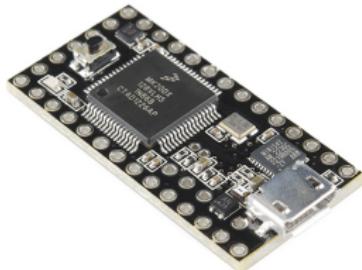


- Design guidelines for required #integer bits to avoid overflow also available

# 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 RAM)



**fixed point**

| vars/constr. | CPU time (ms) | Code (kB) |
|--------------|---------------|-----------|
| 10/20        | 22.9          | 15        |
| 20/40        | <b>52.9</b>   | 17        |
| 40/80        | 544.9         | 27        |
| 60/120       | 1519.8        | 43        |

**floating point**

| vars/constr. | CPU time (ms) | Code (kB) |
|--------------|---------------|-----------|
| 10/20        | 88.6          | 16        |
| 20/40        | <b>220.1</b>  | 21        |
| 40/80        | 2240          | 40        |
| 60/120       | 5816          | 73        |

Operations are about 4x faster in fixed point than in floating point

- Alternating Directions Method of Multipliers for QP

$$\begin{aligned} \min \quad & \frac{1}{2} z' Q z + c' z \\ \text{s.t.} \quad & \ell \leq A z \leq u \end{aligned}$$

$$\begin{aligned} z^{k+1} &= -(Q + \rho A' A)^{-1}(\rho A'(u^k - s^k) + c) \\ s^{k+1} &= \min\{\max\{A z^{k+1} + u^k, \ell\}, u\} \\ u^{k+1} &= u^k + A z^{k+1} - s^{k+1} \end{aligned}$$

```

while k<maxiter
  k=k+1;
  z=-1/M*(c+A'* (rho*(u-s)));
  Az=A*z;
  s=max(min(Az+u,ub),lb);
  u=u-Az-s;
end

```

(7 lines EML code)  
 (≈40 lines of C code)

$\rho u$  = dual vector

- Matrix  $(Q + \rho A' A)$  must be nonsingular
- The factorization of matrix  $(Q + \rho A' A)$  can be done at start and cached
- Very **simple to code**. Sensitive to matrix **scaling** (as gradient projection)
- Used in many applications (control, signal processing, machine learning)

# REGULARIZED ADMM FOR QUADRATIC PROGRAMMING

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

- Robust “regularized” ADMM iterations:

$$\begin{aligned} z^{k+1} &= -(Q + \rho A^T A + \epsilon I)^{-1}(c - \epsilon z_k + \rho A^T(u^k - z^k)) \\ s^{k+1} &= \min\{\max\{Az^{k+1} + y^k, \ell\}, u\} \\ u^{k+1} &= u^k + Az^{k+1} - s^{k+1} \end{aligned}$$

- Works for any  $Q \succeq 0$ ,  $A$ , and choice of  $\epsilon > 0$
- **Simple** to code, **fast**, and **robust**
- Only needs to factorize  $\begin{bmatrix} Q + \epsilon I & A' \\ A & -\frac{1}{\rho} I \end{bmatrix}$  once
- Implemented in free **osQP solver**  
**(Python interface:  $\approx 20,000$  downloads)**
- Extended to solve **mixed-integer quadratic programming** problems

<http://osqp.org>

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

# PRECONDITIONING (SCALING)

- First-order methods can be very sensitive to problem scaling
- Preconditioning required to improve convergence rate (Giselsson, Boyd, 2015)
- **Jacobi** scaling of dual problem: (Bertsekas, 2009)

$$\begin{array}{ll}\min_z & \frac{1}{2} z' Q z + x' F' z \\ \text{s.t.} & G z \leq W + S x\end{array}$$



$$\begin{array}{ll}\min & \frac{1}{2} y' H y + (D x + W)' y \\ \text{s.t.} & y \geq 0\end{array}$$

$$y = P y_s, \quad P = \text{diag} \left( \frac{1}{\sqrt{H_{ii}}} \right)$$



$$H = -GQ^{-1}G'$$

$$\begin{array}{ll}\min & \frac{1}{2} y_s' (P H P) y_s + (D x + W)' P y_s \\ \text{s.t.} & y_s \geq 0\end{array}$$

- Equivalent to **scale constraints** in primal problem:  $\frac{1}{\sqrt{H_{ii}}} G_i z \leq \frac{1}{\sqrt{H_{ii}}} W_i$
- Primal solution:  $z^* = -H^{-1}((PG)' y_s^* + f)$

# SCALING EXAMPLE

- AFTI-F16 example
- MPC setup:  $N = 10, N_u = 2$ .  
Feasibility & optimality thresholds =  $10^{-2}$



input constraints only

| AFTI | max  |      | average |     |
|------|------|------|---------|-----|
|      | 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    |       | average |      |
|-------|--------|-------|---------|------|
|       | 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)

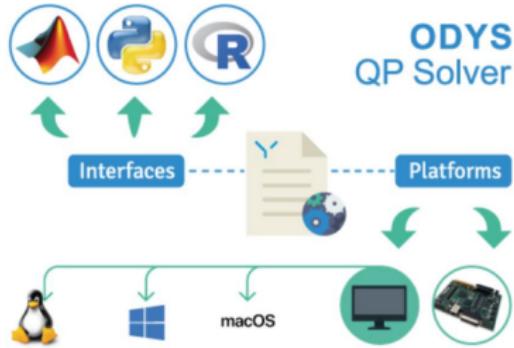
PQP = Projection-free parallel QP (Di Cairano, Brand, Bortoff, 2013)

DRSA = Accelerated Douglas-Rachford Splitting (Patrinos, Stella, Bemporad, 2014)

# ODYS QP SOLVER

- General purpose QP solver designed for **industrial production**

$$\begin{array}{ll}\min_z & \frac{1}{2} z'Qz + c'z \\ \text{s.t.} & Az \leq b \\ & Ez = f\end{array}$$

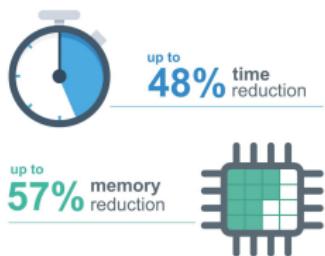


- Implements a **proprietary** state-of-the-art method for QP
- Completely written in **ANSI-C**
- Currently evaluated for production by some major OEMs

<http://odys.it/qp>

# ODYS QP SOLVER

- Emphasis on:
  - **robustness** (especially in **single precision**)
  - **speed** of execution
  - **low memory** requirements
- Compliant with **MISRA-C 2012** standards
- **Certifiable** execution time



| DOUBLE PRECISION             |                      |                      |                      |  |
|------------------------------|----------------------|----------------------|----------------------|--|
| # VARIABLES<br># CONSTRAINTS | 20<br>100            | 50<br>250            | 100<br>500           |  |
| Time [ms]                    | 0.12                 | 0.75                 | 5.47                 |  |
| Optimality                   | $0.9 \cdot 10^{-14}$ | $1.1 \cdot 10^{-14}$ | $2.0 \cdot 10^{-13}$ |  |
| Feasibility                  | $0.2 \cdot 10^{-13}$ | $0.6 \cdot 10^{-13}$ | $1.5 \cdot 10^{-13}$ |  |

Intel Core i7 2.5GHz

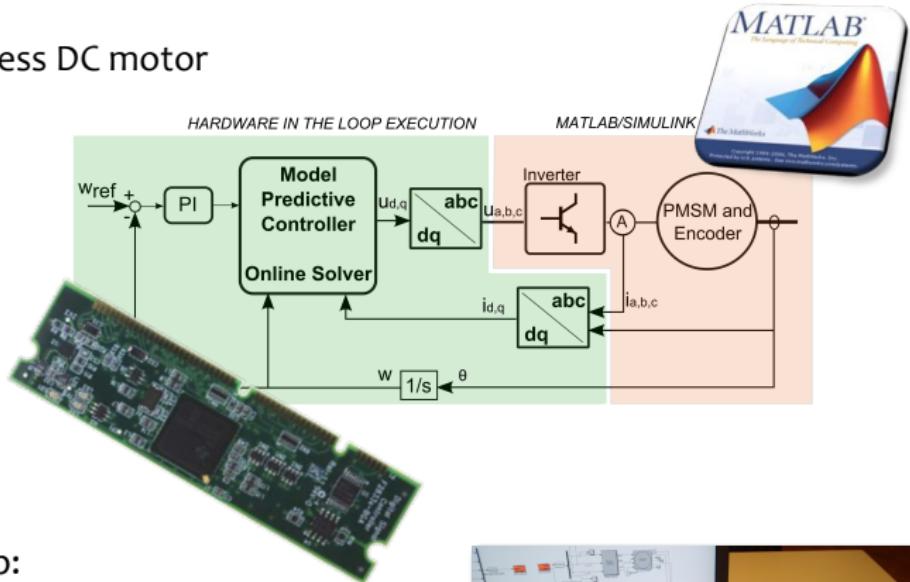
| SINGLE PRECISION             |                     |                     |                     |  |
|------------------------------|---------------------|---------------------|---------------------|--|
| # VARIABLES<br># CONSTRAINTS | 20<br>100           | 50<br>250           | 100<br>500          |  |
| Time [ms]                    | 0.10                | 0.71                | 5.20                |  |
| Optimality                   | $4.7 \cdot 10^{-7}$ | $1.2 \cdot 10^{-6}$ | $1.6 \cdot 10^{-5}$ |  |
| Feasibility                  | $1.5 \cdot 10^{-5}$ | $4.0 \cdot 10^{-5}$ | $7.8 \cdot 10^{-5}$ |  |

- Version **optimized for MPC** is available

<http://odys.it/qp>

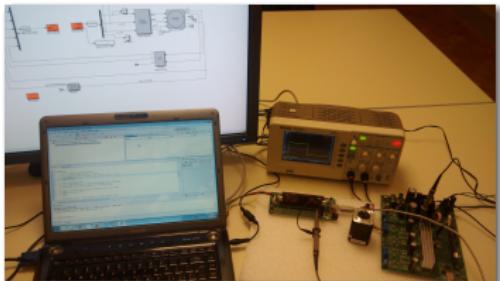
# HARDWARE-IN-THE-LOOP EXPERIMENTS

- Control of a brushless DC motor



- Experimental setup:

- PC with MATLAB/Simulink
- RS232 adapter
- TMS320F28335 DSP (**150 MHz**)
- 68 kB RAM, 512 kB flash



# HARDWARE-IN-THE-LOOP EXPERIMENTS

- Linear MPC setup ( $N = 4, N_u = 2$ )

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

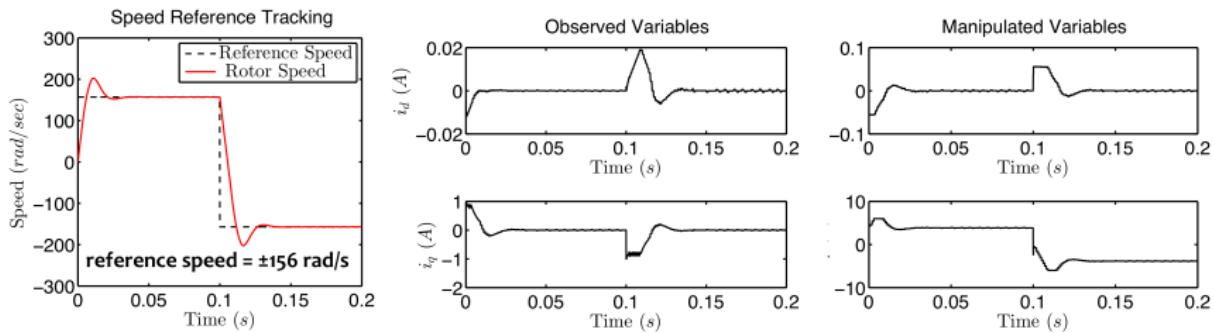
$x = \begin{bmatrix} i_d \\ i_q \end{bmatrix} \quad u = \begin{bmatrix} u_d \\ u_q \end{bmatrix}$

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

- $R$  = stator resistance [ $\Omega$ ]
- $L$  = stator inductance [ $H$ ]
- $\omega$  = rotor speed [rad/s]
- $\lambda$  = motor flux leakage [Wb]

input and output constraints

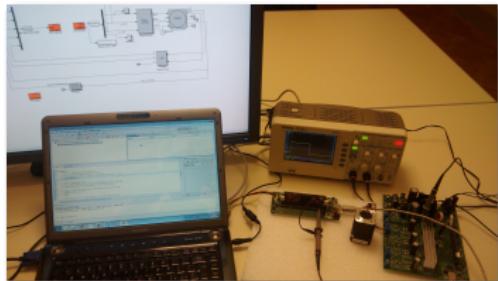
$$-\epsilon I_{\max} \leq i_d \leq \epsilon I_{\max}$$
$$V_{\max} \leq u_q \leq V_{\max}$$



# QP SOLVERS - AN EXPERIMENTAL COMPARISON

- **Experimental setup:**

- PC with MATLAB/Simulink
- RS232 adapter
- TMS320F28335 DSP (**150 MHz**)

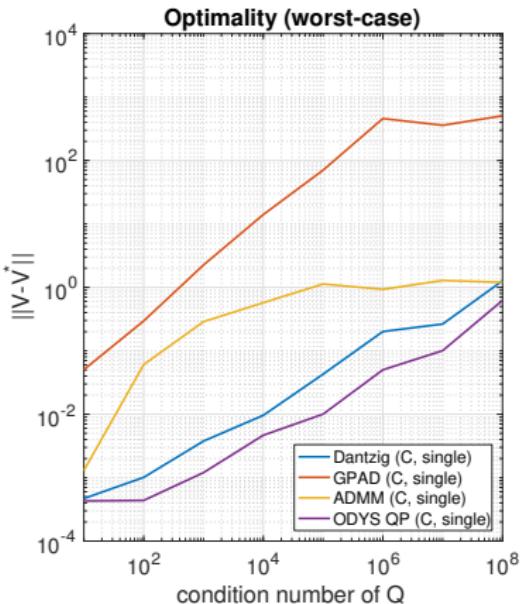
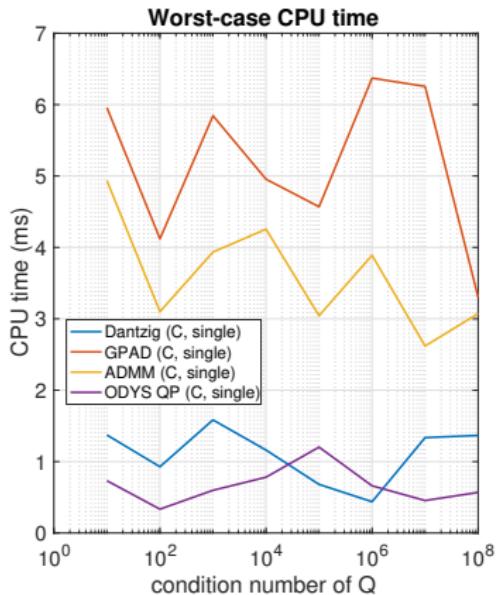


| vars x constr. | ODYS QP | GPAD    | ADMM   |
|----------------|---------|---------|--------|
| 4x 16          | 0.12 ms | 0.33 ms | 1.4 ms |
| 8x 24          | 0.44 ms | 1.1 ms  | 4 ms   |
| 12x32          | 1.2 ms  | 2.6 ms  | 8.2 ms |

- Active set (AS) methods are usually the best on small/medium problems:
  - excellent quality solutions within few iterations
  - behavior is more predictable (=less sensitive to preconditioning)
  - no linear algebra libraries required for matrix factorization/solving linear systems

# QP SOLVER COMPARISON - ROBUSTNESS

- Random QP problems with 20 vars, 100 constraints (worst-case over 100 QPs)
- **Single precision** arithmetic (MacBook Pro 3 GHz Intel Core i7)



# MORE ON REQUIREMENTS FOR EMBEDDED OPTIMIZATION

- Need to 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  $\mu$ -processor (please simple!)
- One can often trade off **throughput vs. memory**
- Numerical robustness of on-line computations is important. The solver must perform well in **single precision** (or even fixed-point) arithmetics
- Limited linear algebra (on-line) (e.g., matrix-vector products at most)
- **Dense problems**, no need for sparse linear algebra
- Suboptimal solutions may be ok. Feasibility more important than optimality
- Typical problem size: **10 ÷ 50 variables, 10 ÷ 100 constraints**

# CAN WE SOLVE QP'S USING LEAST SQUARES ?

The **least squares** (LS) problem is probably the most studied problem in numerical linear algebra



Adrien-Marie Legendre  
(1752–1833)

$$z^* = \arg \min \|Az - b\|_2^2$$



Carl Friedrich Gauss  
(1777–1855)

In MATLAB: `>> z=A\b` (one character !)

## Nonnegative Least Squares (NNLS)

(Lawson, Hanson, 1974)

$$\begin{aligned} \min_x \quad & \|Az - b\|_2^2 \\ \text{s.t.} \quad & z \geq 0 \end{aligned}$$

## Bounded-Variable Least Squares (BVLS)

(Stark, Parker, 1995)

$$\begin{aligned} \min_z \quad & \|Az - b\|_2^2 \\ \text{s.t.} \quad & \ell \leq z \leq u \end{aligned}$$

# ACTIVE-SET METHOD FOR NNLS

(Lawson, Hanson, 1974)

- Active-set method to solve the NNLS problem

$$\min_{z \geq 0} \|Az - b\|_2^2, \quad A \in \mathbb{R}^{m \times n}$$

1.  $\mathcal{P} \leftarrow \emptyset, z \leftarrow 0;$
2.  $w \leftarrow A'(Az - b);$
3. **if**  $w \geq 0$  **or**  $\mathcal{P} = \{1, \dots, m\}$  **then go to** Step 10;
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**  $z \leftarrow y$  and **go to** Step 2;
7.  $j \leftarrow \arg \min_{h \in \mathcal{P}: y_h \leq 0} \left\{ \frac{z_h}{z_h - y_h} \right\};$
8.  $z \leftarrow z + \frac{z_j}{z_j - y_j} (y - z);$
9.  $\mathcal{I} \leftarrow \{h \in \mathcal{P} : z_h = 0\}, \mathcal{P} \leftarrow \mathcal{P} \setminus \mathcal{I};$  **go to** Step 5;
10. **end.**

The algorithm maintains the primal vector  $z$  feasible and keeps switching the active set until the dual variable  $w$  is also feasible.

The key step 5 requires solving an unconstrained LS problem. An LDL<sup>T</sup>, Cholesky, or QR factorization of  $(A')_{\mathcal{P}}$  can be computed recursively

very simple to solve (750 chars in Embedded MATLAB)

# SOLVING QP'S VIA NONNEGATIVE LEAST SQUARES

(Bemporad, 2016)

- Complete the squares and transform QP to **least distance problem** (LDP)

$$\begin{array}{ll}\min_z & \frac{1}{2} z' Q z + c' z \\ \text{s.t.} & G z \leq g\end{array}$$

$$Q = Q' \succ 0$$

$$\begin{array}{c}Q = L'L \\ \xrightarrow{\hspace{1cm}} \\ u \triangleq Lz + L^{-T}c\end{array}$$

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

- An LDP can be solved by the NNLS (Lawson, Hanson, 1974)

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

$$\begin{array}{lll}M & = & GL^{-1} \\ d & = & b + GQ^{-1}c\end{array}$$

- If residual = 0 then QP is infeasible. Otherwise set

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

- Extended to solving mixed-integer QP's (Bemporad, NMPC, 2015)

# RELATION BETWEEN NNLS PROBLEM AND DUAL QP

- Note that the NNLS reformulation of the QP is **not** the dual QP:

dual QP

$$\begin{aligned}\min_{\lambda} \quad & \frac{1}{2} \lambda' (MM') \lambda + d' \lambda \\ \text{s.t.} \quad & \lambda \geq 0\end{aligned}$$

can be unbounded if primal QP  
is infeasible

$$(d = W + Dx)$$

NNLS problem

$$\begin{aligned}\min_y \quad & \frac{1}{2} y' (MM' + dd') y + d'y \\ \text{s.t.} \quad & y \geq 0\end{aligned}$$

always bounded by  $-\frac{1}{2}$   
(cost =  $-\frac{1}{2}$  if primal QP is  
infeasible)

- If primal QP is feasible:

$$\lambda^* = \frac{1}{1 + d'y^*} y^*$$

# ROBUST QP SOLVER BASED ON NNLS

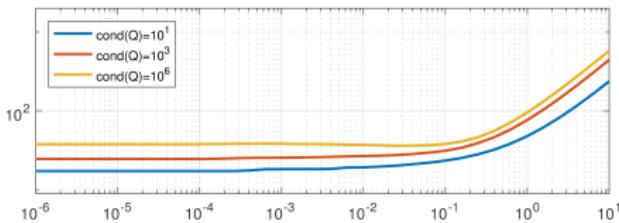
(Bemporad, 2018)

- QP solver based on NNLS is not very robust numerically
- **Key idea:** Solve a sequence of QP via NNLS within **proximal-point iterations**

$$\begin{aligned} z_{k+1} = \arg \min_z \quad & \frac{1}{2} z' Q z + c' z + \frac{\epsilon}{2} \|z - z_k\|_2^2 \\ \text{s.t.} \quad & A z \leq b \\ & G x = g \end{aligned}$$

- **Numerical robustness:**  $Q + \epsilon I$  can be arbitrarily well conditioned !
- Choice of  $\epsilon$  is not critical

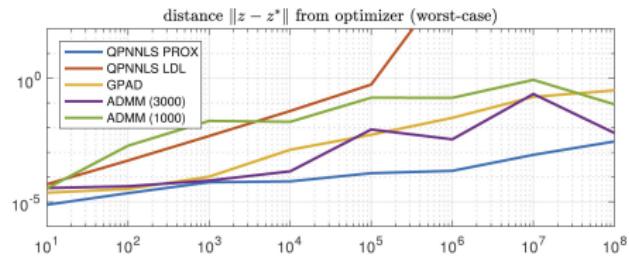
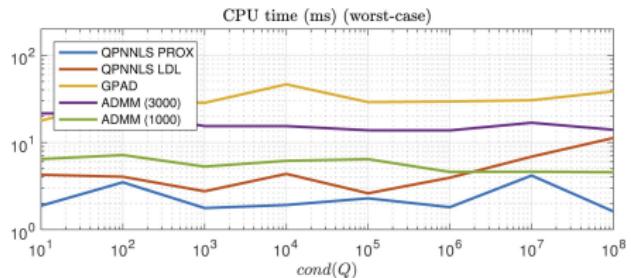
total number of active-set iterations  
as a function of  $\epsilon$



- Each QP is heavily warm-started and makes very few active-set changes
- Recursive  $LDL^T$  decompositions/rank-1 updates exploited for max efficiency

# SOLVING QP'S VIA NNLS AND PROXIMAL POINT ITERATIONS

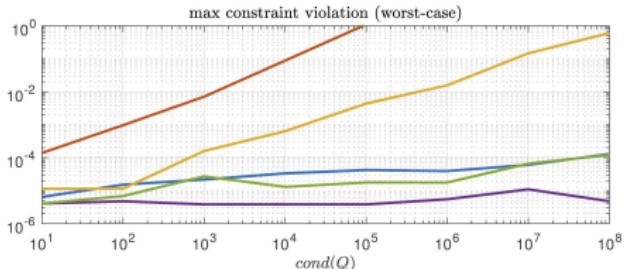
(Bemporad, 2018)



single precision arithmetic

30 vars, 100 constraints

(Macbook Pro 3 GHz Intel Core i7)



- Extended to solve **MIQP** problems (Naik, Bemporad, 2018)

# PRIMAL-DUAL INTERIOR-POINT METHOD FOR QP

(Nocedal, Wright, 2006) (Gondzio, Terlaki, 1994)

- The KKT conditions for the convex QP

$$\begin{aligned} \min_x \quad & \frac{1}{2} x' Q x + c' x \\ \text{s.t.} \quad & Ax \leq b \quad Q = Q' \succeq 0 \\ & Ex = f \end{aligned}$$

are

$$\begin{aligned} Qx + c + E'y + A'z &= 0 \\ Ex &= f \\ Ax + s &= b \\ z_i s_i &= 0 \\ z, s &\geq 0 \end{aligned}$$

- In a nutshell, **interior-point** methods use Newton's method with line search to solve the above nonlinear system of equations
- The complementary slackness constraint is replaced by  $z_i s_i = \mu$  and  $\mu \rightarrow 0$

# PRIMAL-DUAL INTERIOR-POINT METHOD FOR QP

(Nocedal, Wright, 2006) (Gondzio, Terlaki, 1994)

- Each interior-point iteration requires solving a linear system of the form

$$\begin{bmatrix} Q & E' & A' & 0 \\ E & 0 & 0 & 0 \\ A & 0 & 0 & I \\ 0 & 0 & S & Z \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta z \\ \Delta s \end{bmatrix} = \begin{bmatrix} -r_Q \\ -r_E \\ -r_A \\ -r_S \end{bmatrix}$$

$Z = \text{diag } z$   
 $S = \text{diag } s$

- We can eliminate some variables and solve instead

$$\begin{bmatrix} Q & E' & A'Z \\ E & 0 & 0 \\ A & 0 & -S \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta \tilde{z} \end{bmatrix} = \begin{bmatrix} -r_Q \\ -r_E \\ Z^{-1}r_S - r_A \end{bmatrix}$$

or even

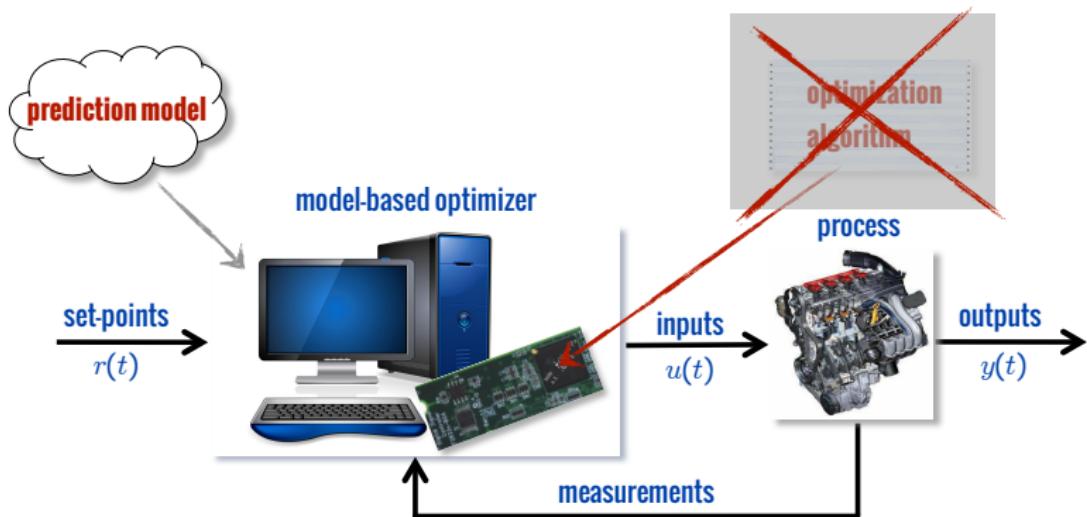
$$\begin{bmatrix} Q + A'ZS^{-1}A & E' \\ E & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} = \begin{bmatrix} -r_Q + A'S^{-1}(r_S - Zr_A) \\ -r_E \end{bmatrix}$$

- In MPC the **structure**  $x_{k+1} = Ax_k + Bu_k$  can be heavily exploited to factorize/solve the linear systems efficiently (Rao, Wright, Rawlings, 1998) (Wright, 2018)

# ACTIVE-SET VS INTERIOR-POINT METHODS

- Active-set (AS) is **simpler to implement** than interior-point (IP) method
- IP needs **advanced linear algebra** operations during iterations, AS does not
- Linear systems to solve in IP tend to become ill-conditioned at convergence
- AS converges in a **finite number of steps** (within machine precision),  
IP is an iterative method that only converges asymptotically
- IP provides good solutions within 10-15 IP iterations (**usually** ...).  
AS iterations increase when both vars and constraints increase
- IP faster than AS in QPs with say >500 vars & constraints. IP well exploits sparse linear algebra
- Number of AS iterations can be **exactly certified** for given QP matrices  
(see later ...)

# MPC WITHOUT ON-LINE QP



- Can we implement constrained linear MPC **without an on-line solver** ?

# ON-LINE VS OFF-LINE OPTIMIZATION

$$\begin{array}{ll}\min_z & \frac{1}{2} z' Q z + \textcolor{red}{x'(t) F' z} + \frac{1}{2} x'(t) Y x(t) \\ \text{s.t.} & G z \leq W + S \textcolor{red}{x(t)}\end{array}$$

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

- **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 Programming (QP)**

- **Off-line** optimization: solve the QP in advance for all  $x(t)$  in a given range to find the control law  $u = u_0^*(x)$  **explicitly**



**multi-parametric Quadratic Programming (mpQP)**

# MULTIPARAMETRIC PROGRAMMING PROBLEM

Given the optimization problem

$$\begin{array}{ll} \min_z & f(z, \textcolor{red}{x}) \\ \text{s.t.} & g(z, \textcolor{red}{x}) \leq 0 \end{array} \quad \begin{array}{l} \textcolor{red}{x} \in \mathbb{R}^n \\ z \in \mathbb{R}^s \end{array}$$

and a set  $X \subseteq \mathbb{R}^n$  of parameters  $x$  of interest, determine:

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

# MULTIPARAMETRIC QUADRATIC PROGRAMMING (MPQP)

$$\begin{array}{ll}\min_z & \frac{1}{2}z'Qz + \textcolor{red}{x'}F'z + \frac{1}{2}x'Yx \\ \text{s.t.} & Gz \leq W + S\textcolor{red}{x}\end{array}$$

$$z = \begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_{N-1} \end{bmatrix}, \quad W \in \mathbb{R}^q$$

- **Objective:** solve the QP off line **for all**  $x \in X$  to get the optimizer function  $z^*$ , and therefore the MPC control law  $u(x) = \begin{bmatrix} I & 0 & \dots & 0 \end{bmatrix} z^*(x)$  **explicitly**
- Assumptions:  $f(z, x) = \frac{1}{2} \begin{bmatrix} z \\ x \end{bmatrix}' \begin{bmatrix} Q & F \\ F' & Y \end{bmatrix} \begin{bmatrix} z \\ x \end{bmatrix}$  is jointly convex wrt  $\begin{bmatrix} z \\ x \end{bmatrix}$  and is strictly convex wrt  $z$

$$\begin{bmatrix} Q & F \\ F' & Y \end{bmatrix} \succeq 0 \quad \text{always satisfied if mpQP comes from an MPC problem}$$

$$Q = Q' \succ 0 \quad \text{always satisfied if weight matrix } R \succ 0$$

# LINEARITY OF MPQP SOLUTION

(Bemporad, Morari, Dua, Pistikopoulos, 2002)

- Fix a point  $x_0$  inside the parameter set  $X$

$$\begin{aligned} \min_z \quad & \frac{1}{2} z' Q z + \textcolor{red}{x'} F' z \\ \text{s.t.} \quad & G z \leq W + S \textcolor{red}{x} \end{aligned}$$

- Solve a QP and find  $z^*(x_0)$  and Lagrange multipliers  $\lambda^*(x_0)$
- Identify the set  $I(x_0)$  of indices of active constraints at  $z^*(x_0)$

$$G_i z^*(x_0) = W_i + S_i x_0, \quad \forall i \in I(x_0) \quad I(x_0) \subseteq \{1, \dots, q\}$$

$$G_i z^*(x_0) < W_i + S_i x_0, \quad \forall i \notin I(x_0)$$

- Consider now generic  $x, z, \lambda$  and the KKT optimality conditions for the QP

$$\begin{aligned} Qz + Fx + G'\lambda &= 0 \\ \lambda_i(G^i z - W^i - S^i x) &= 0, \quad \forall i = 1, \dots, q \\ \lambda &\geq 0 \\ Gz - W - Sx &\leq 0 \end{aligned}$$

# LINEARITY OF MPQP SOLUTION

- Impose the combination  $I(x_0)$  of active constraints:

$$\tilde{G}z - \tilde{W} - \tilde{S}x = 0, \quad \hat{G}z - \hat{W} - \hat{S}x < 0 \Rightarrow \hat{\lambda} = 0$$

where “~” means collecting the indices in  $I(x_0)$  and “^” the remaining ones

- From  $Qz + Fx + G'\lambda = 0$  we get  $z = -Q^{-1}(Fx + G'\lambda)$
- Substitute into  $\tilde{G}z - \tilde{W} - \tilde{S}x = 0$  and get

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

- Substitute  $\tilde{\lambda}$  and, assuming the rows of  $\tilde{G}$  linearly independent, get

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

In some neighborhood of  $x_0$ ,  $\lambda$  and  $z$  are explicit affine functions of  $x$

(Zafiriou, 1990)

# MULTIPARAMETRIC QP ALGORITHM

- Impose primal and dual feasibility

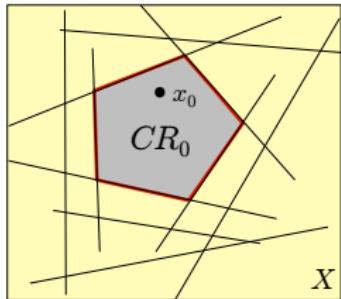
$$\hat{G}z(x) \leq \hat{W} + \hat{S}x$$

primal feasibility

$$\tilde{\lambda}(x) \geq 0$$

dual feasibility

Linear inequalities in  $x$



- Remove **redundant constraints** and get the **critical region**  $CR_0$

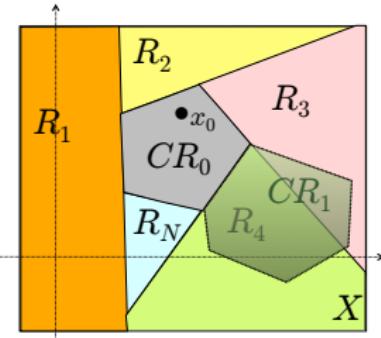
$$\begin{aligned} CR_0 &= \{x \in X : \hat{G}z(x) \leq \hat{W} + \hat{S}x, \tilde{\lambda}(x) \geq 0\} \\ &= \{x \in X : A_0x \leq b_0\} \end{aligned}$$

- For all parameters  $x$  in  $CR_0$ ,  $z(x), \tilde{\lambda}(x), \hat{\lambda}(x) \equiv 0$  is the optimal solution, as all KKT conditions are satisfied
- For any parameter  $x$  **outside**  $CR_0$ ,  $I(x_0)$  cannot be the optimal combination, as the KKT conditions cannot be satisfied ( $z(x)$  or  $\tilde{\lambda}(x)$  is infeasible)

# MULTIPARAMETRIC QP SOLVER #1

(Bemporad, Morari, Dua, Pistikopoulos, 2002)

**Method #1:** Split  $X$  and proceed iteratively

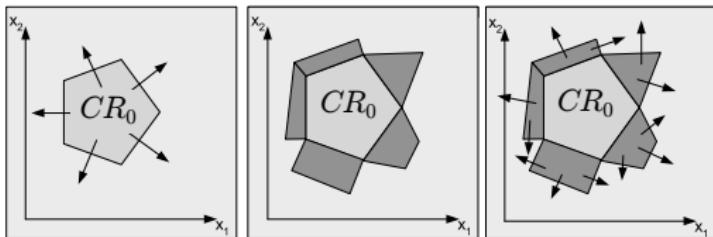


$$\begin{aligned} CR_0 &= \{x \in X : A_0 x \leq b_0\}, b_0 \in \mathbb{R}^{n_0} \\ R_i &= \{x \in X : A_{0,i} x > b_{0,i}, A_{0,j} x \leq b_{0,j}, \forall j < i\} \\ X &= CR_0 \cup R_1 \cup \dots \cup R_{n_0} \end{aligned}$$

- The above splitting is only used as a search procedure
- 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 stops after at most  $2^q$  recursions ( $q$ =number of constraints)
- Some  $CR$ 's can be found multiple times, duplicates must be removed

# MULTIPARAMETRIC QP SOLVERS #2, #3

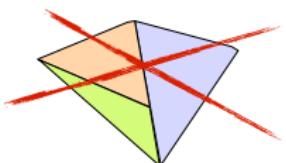
(Tøndel, Johansen, Bemporad, 2003)



- **Method #2:** the active set of a neighboring region is obtained as follows:
  - **add constraint**  $\#i$  if the common facet comes from  $\hat{G}^i(x) \leq \hat{W}^i + \hat{S}^i x$   
(this maintains feasibility of solution  $z^*(x)$ )
  - **remove constraint**  $\#i$  if the common facet comes from  $\tilde{\lambda}_j(x) \geq 0$   
(this maintains optimality of solution  $z^*(x)$ )
- **Method #3:** step out by  $\epsilon$  outside each facet, solve QP  
get new region, iterate. (Baotic, 2002)

This exploits the **facet-to-facet property**

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



(Spjøtvold, 2008)

# PROPERTIES OF MPQP SOLUTION

(Bemporad, Morari, Dua, Pistikopoulos, 2002)

## THEOREM

Assume  $H \succ 0$ ,  $\begin{bmatrix} H & F \\ F' & Y \end{bmatrix} \succeq 0$ . Then

- the **set of feasible parameters**  $X^*$  is a **polyhedron**

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

- the **optimizer function**  $z^* : X^* \rightarrow \mathbb{R}^s$  is **continuous and piecewise affine**

$$\begin{aligned} z^*(x) = \arg \min_z \quad & \frac{1}{2} z' Q z + x' F' z \\ \text{s.t.} \quad & G z \leq W + S x \end{aligned}$$

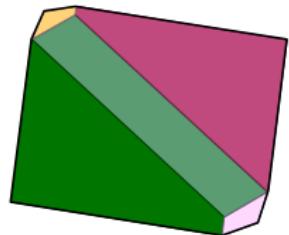
- the **value function**  $V^* : X^* \rightarrow \mathbb{R}$  is **convex, continuous, piecewise quadratic**, and (even  $\mathcal{C}^1$  if no degeneracy occurs)

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

# PROOF OF THEOREM

## 1) $X^*$ is convex and polyhedral

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



- **Geometric proof:**  $X^*$  is the **projection** of the polyhedron

$$\left\{ \begin{bmatrix} z \\ x \end{bmatrix} : \begin{bmatrix} G & -S \end{bmatrix} \begin{bmatrix} z \\ x \end{bmatrix} \leq W \right\}$$

onto the  $x$ -space. Hence  $X^*$  is a polyhedron and therefore convex

- **Algebraic proof:**

- 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  $z_\alpha$  satisfies the constraints

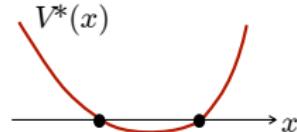
$$\begin{aligned} Gz_\alpha &= \alpha Gz^*(x_1) + (1 - \alpha)Gz^*(x_2) \\ &\leq \alpha(W + Sx_1) + (1 - \alpha)(W + Sx_2) = W + Sx_\alpha \end{aligned}$$

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

■

# PROOF OF THEOREM

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



- Since  $z_\alpha$  satisfies the constraints

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

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

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



# PROOF OF THEOREM

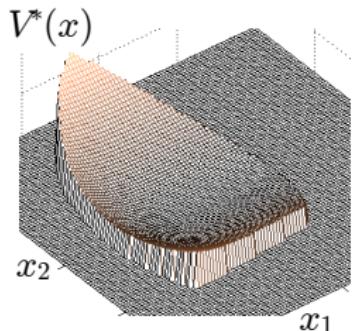
## 3) Continuity of $z^*$ and $V^*$ with respect to $x$

- Let  $z^*(x) = L_i x + M_i$  when  $x \in CR_i$
- $z^*$  is linear and therefore continuous on the interior of each  $CR_i$
- Take  $x$  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 KKT conditions
- By strict convexity of the optimization problem ( $Q \succ 0$ ), the optimizer 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  $z^*$  also across boundaries of critical regions
- As  $V^*$  is the composition of continuous functions,  
$$V^*(x) = \frac{1}{2} z^*(x)' Q z^*(x) + x' F' z^*(x) + \frac{1}{2} x' Y x,$$
 it is also continuous



# MULTIPARAMETRIC CONVEX PROGRAMMING

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



**General result** (Mangasarian, Rosen, 1964)

- If  $f, g_i$  are **jointly convex functions** of  $(z, x)$ , then  $X^*$  is a convex set and  $V^*$  is convex wrt  $x$
- If  $f, g_i$  are also **continuous** wrt  $(z, x)$  then  $V^*$  is also continuous wrt  $x$

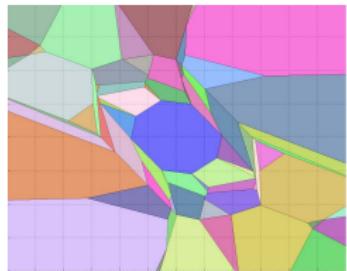
$V^*$  and  $X^*$  may not be expressible analytically. Approximate solutions can be derived (Bemporad, Filippi, 2003)

# EXPLICIT MPC SOLUTION

(Bemporad, Morari, Dua, Pistikopoulos, 2002)

- **Corollary:** since the multiparametric solution

$$\begin{aligned} z^*(x) = \arg \min_z & \quad \frac{1}{2} z' Q z + x' F' z \\ \text{s.t.} & \quad G z \leq W + S x \end{aligned}$$



of a strictly convex QP is **continuous** and **piecewise affine**,  
the **linear MPC law is continuous & piecewise affine too**

$$z^* = \begin{bmatrix} \mathbf{u}_0 \\ u_1 \\ \vdots \\ u_{N-1}^* \end{bmatrix}$$

$$u_0^*(x) = \begin{cases} F_1 x + g_1 & \text{if } H_1 x \leq K_1 \\ \vdots & \vdots \\ F_M x + g_M & \text{if } H_M x \leq K_M \end{cases}$$

```
while (!mocpEXPCDR_NEO) {  
    int i, l1, l2;  
    while ((l1+l2) < m_ocp.n) {  
        if (m_ocp.x[l1] > m_ocp.x[l2]) {  
            swap(m_ocp.x[l1], m_ocp.x[l2]);  
            swap(m_ocp.u[l1], m_ocp.u[l2]);  
            swap(m_ocp.y[l1], m_ocp.y[l2]);  
        }  
        i++;  
    }  
    if (m_ocp.y[i-1] > m_ocp.y[i]) {  
        if (m_ocp.y[i-1] > m_ocp.y[i-2]) {  
            l1 = i-2;  
            l2 = i-1;  
        } else {  
            l1 = i-1;  
            l2 = i;  
        }  
        if (m_ocp.y[l1] > m_ocp.y[l2]) {  
            swap(m_ocp.y[l1], m_ocp.y[l2]);  
            swap(m_ocp.u[l1], m_ocp.u[l2]);  
            swap(m_ocp.x[l1], m_ocp.x[l2]);  
        }  
    }  
}  
/* get next delimiter l1 */  
l2 = m_ocp.y.length(); /* get next delimiter l2 */  
}
```

If it's just a **while loop!**

# COMPLEXITY REDUCTION

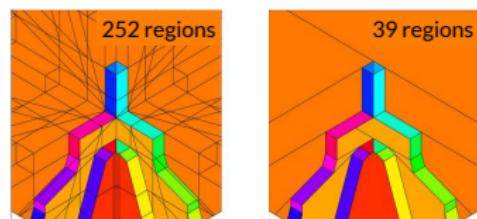


- We are interested only in the first components of the optimizer  $z^*$

$$z(x) \triangleq \begin{bmatrix} \mathbf{u}'_0(\mathbf{x}) & u'_1(x) & \dots & u'_{N-1}(x) \end{bmatrix}$$

- Regions where the first component of the solution is the same can be joined if their union is convex (Bemporad, Fukuda, Torrisi, 2001)
- Optimal merging methods exist

(Geyer, Torrisi, Morari, 2008)



# REMOVING REDUNDANT INEQUALITIES VIA LP

- A variety of multiparametric quadratic programming solvers is available

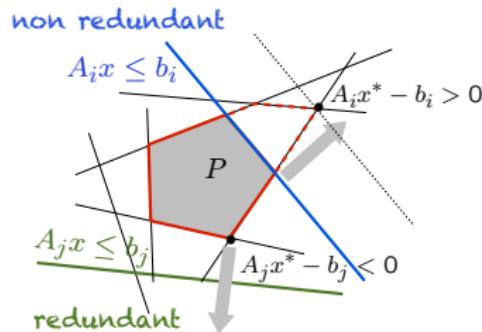
(Bemporad et al., 2002) (Baotic, 2002) (Tøndel, Johansen, Bemporad, 2003) (Jones, Morari, 2006)  
(Spjøtvold et al., 2006) (Patrinos, Sarimveis, 2010) (Gupta et al., 2011)

- Most computations are spent in removing redundant inequalities

- This is usually done by solving a linear program (LP) per facet:

$$\begin{aligned} 0 \geq \max_x \quad & A_i x - b_i \\ \text{s.t.} \quad & A_j x \leq b_j, \forall j \neq i \end{aligned} \quad \rightarrow \quad A_i x \leq b_i \text{ is redundant}$$

(if  $\max_x A_i x - b_i = 0$  the inequality is **weakly redundant**)



# REMOVING REDUNDANT INEQUALITIES VIA NNLS

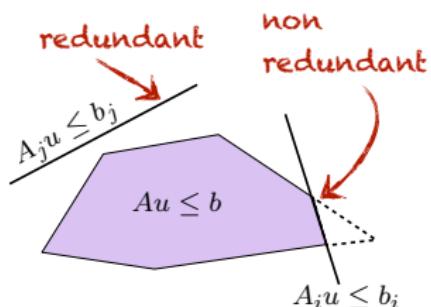
(Bemporad, 2015)

- **Key result:** A polyhedron  $P = \{u \in \mathbb{R}^n : Au \leq b\}$  is nonempty iff the partially NNLS problem

$$(v^*, u^*) = \arg \min_{v,u} \|v + Au - b\|_2^2$$

s.t.  $v \geq 0, u \text{ free}$

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



- Checking emptiness of facet  $P_i = \{x : A_i x = b_i, A_j x \leq b_j\}$  via NNLS is **about 10x faster** than by linear programming
- **Note:** very high precision typically required in computational geometry, so active-set methods are preferable to first-order and interior-point methods

# MULTIPARAMETRIC QP BASED ON NNLS

(Bemporad, 2015)

- Other polyhedral operations can be also solved by NNLS (check full dimension, Chebychev radius, union, projection)
- New mpQP algorithm based on NNLS + dual QP formulation to compute active sets and deal with QP degeneracy
- Comparable performance with other existing methods:

– **Hybrid Toolbox** (Bemporad, 2003)

– **MPT Toolbox 2.6** (w/ default opts)

(Kvasnica, Grieder, Baotic, 2004)

(Herczeg, Kvasnica, Jones, Morari, 2013)

| $q$ | $m$ | Hybrid Tbx | MPT    | NNLS   |
|-----|-----|------------|--------|--------|
| 4   | 2   | 0.0174     | 0.0256 | 0.0026 |
| 4   | 6   | 0.0827     | 0.1105 | 0.0126 |
| 12  | 2   | 0.0398     | 0.0387 | 0.0054 |
| 12  | 6   | 1.2453     | 1.3601 | 0.2426 |
| 20  | 2   | 0.1029     | 0.0763 | 0.0152 |
| 20  | 6   | 6.1220     | 6.2518 | 1.2853 |

- Included in **MPC Toolbox** since version R2014b

(Bemporad, Morari, Ricker, 1998-present)



# DOUBLE INTEGRATOR EXAMPLE

- Model and constraints: 
$$\begin{cases} x(t+1) = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} x(t) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(t) \\ y(t) = \begin{bmatrix} 1 & 0 \end{bmatrix} x(t) \\ -1 \leq u(t) \leq 1 \end{cases}$$

- Objective:

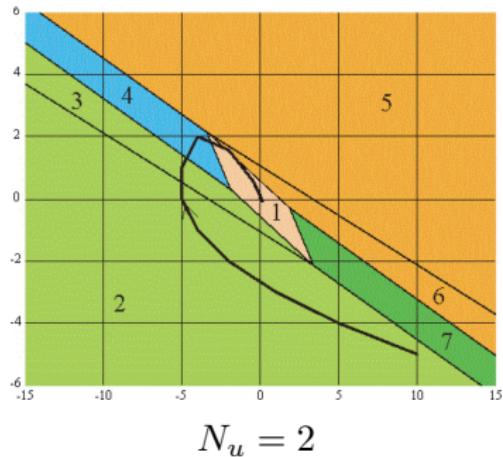
$$\min \sum_{k=0}^{\infty} y_k^2 + \frac{1}{100} u_k^2 \quad \left| \begin{array}{l} u_k = Kx_k, \forall k \geq N_u, K = \text{LQR gain} \\ N_u = N = 2 \end{array} \right.$$

$\xrightarrow{\hspace{1cm}}$   $\left( \sum_{k=0}^1 y_k^2 + \frac{1}{100} u_k^2 \right) + x_2' \underbrace{\begin{bmatrix} 2.1429 & 1.2246 \\ 1.2246 & 1.3996 \end{bmatrix}}_{\substack{\text{solution of algebraic} \\ \text{Riccati equation}}} x_2$

- QP matrices (cost function normalized by max singular value of  $H$ )

$$H = \begin{bmatrix} 0.8365 & 0.3603 \\ 0.3603 & 0.2059 \end{bmatrix}, F = \begin{bmatrix} 0.4624 & 1.2852 \\ 0.1682 & 0.5285 \end{bmatrix}$$
$$G = \begin{bmatrix} 1 & 0 \\ -1 & 0 \\ 0 & 1 \\ 0 & -1 \end{bmatrix}, W = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}, S = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}$$

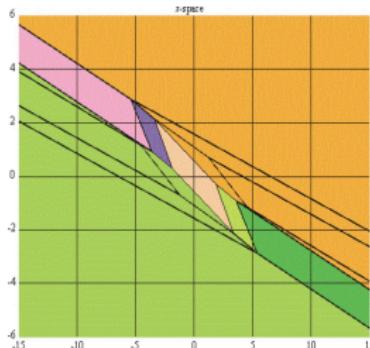
# DOUBLE INTEGRATOR EXAMPLE - EXPLICIT SOLUTION



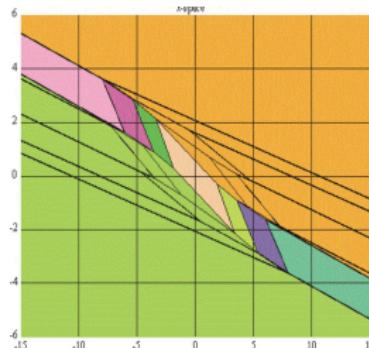
$$u(x) = \begin{cases} [-0.8166 \ -1.75] x & \text{if } \begin{bmatrix} -0.8166 & -1.75 \\ 0.8166 & 1.75 \\ 0.6124 & 0.4957 \\ -0.6124 & -0.4957 \end{bmatrix} x \leq \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \text{ (Region #1)} \\ 1 & \text{if } \begin{bmatrix} 0.3864 & 1.074 \\ 0.297 & 0.9333 \end{bmatrix} x \leq \begin{bmatrix} -1 \\ 1 \end{bmatrix} \text{ (Region #2)} \\ 1 & \text{if } \begin{bmatrix} -0.297 & -0.9333 \\ 0.8166 & 1.75 \\ 0.9712 & 2.699 \end{bmatrix} x \leq \begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix} \text{ (Region #3)} \\ [-0.5528 \ -1.536] x + 0.4308 & \text{if } \begin{bmatrix} -0.9712 & -2.699 \\ 0.3864 & 1.074 \\ 0.6124 & 0.4957 \end{bmatrix} x \leq \begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix} \text{ (Region #4)} \\ -1 & \text{if } \begin{bmatrix} -0.3864 & -1.074 \\ -0.297 & -0.9333 \end{bmatrix} x \leq \begin{bmatrix} -1 \\ -1 \end{bmatrix} \text{ (Region #5)} \\ -1 & \text{if } \begin{bmatrix} 0.297 & 0.9333 \\ -0.8166 & -1.75 \\ -0.9712 & -2.699 \end{bmatrix} x \leq \begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix} \text{ (Region #6)} \\ [-0.5528 \ -1.536] x - 0.4308 & \text{if } \begin{bmatrix} -0.3864 & -1.074 \\ 0.9712 & 2.699 \\ -0.6124 & -0.4957 \end{bmatrix} x \leq \begin{bmatrix} 1 \\ -1 \\ 1 \end{bmatrix} \text{ (Region #7)} \end{cases}$$

go to demo linear/doubleintexp.m (Hybrid Toolbox)

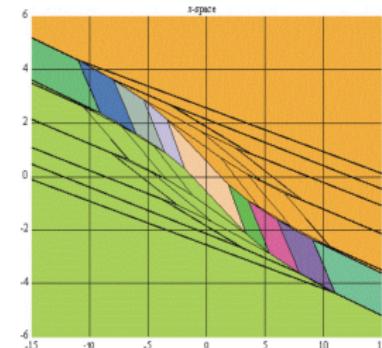
# DOUBLE INTEGRATOR EXAMPLE - EXPLICIT SOLUTION



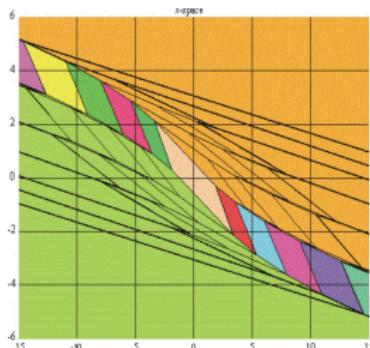
$N_u = 3$



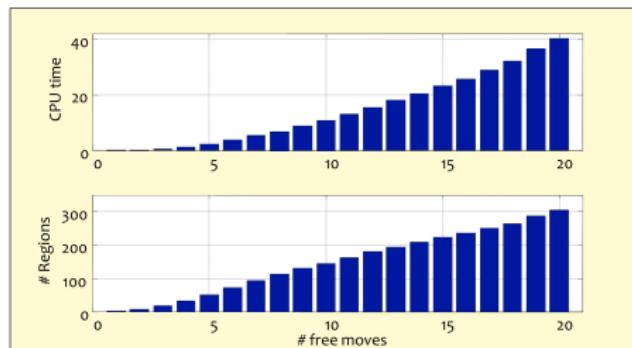
$N_u = 4$



$N_u = 5$



$N_u = 6$



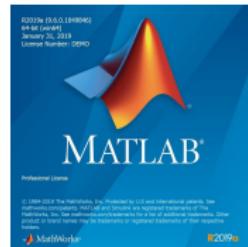
(is the number of regions finite for  $N_u \rightarrow \infty$ ?)

# HYBRID TOOLBOX FOR MATLAB

(Bemporad, 2003-today)

## Features:

- **Hybrid models**: design, simulation, verification
- **MPC** of 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>

7800 downloads

1.5 downloads/day

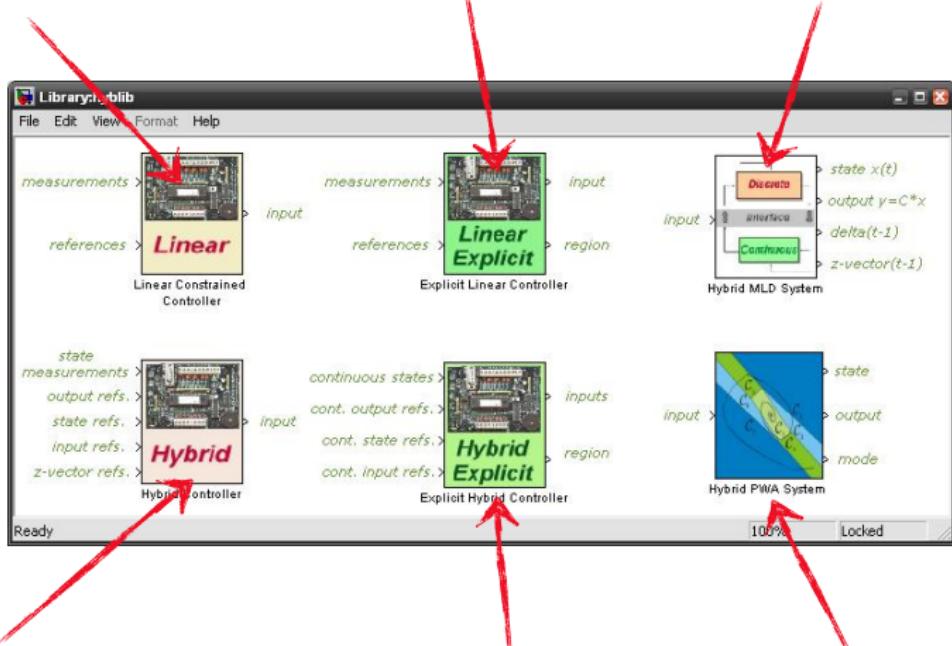
Initially supported by Ford Motor Company

# 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

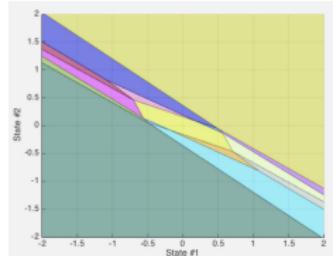
x0=[10,-.3]';
Tstop=40; % simulation time
[X,U,T,Y,I]=sim(Cexp,model,[],x0,Tstop);
```

- @**explicitMPC** object



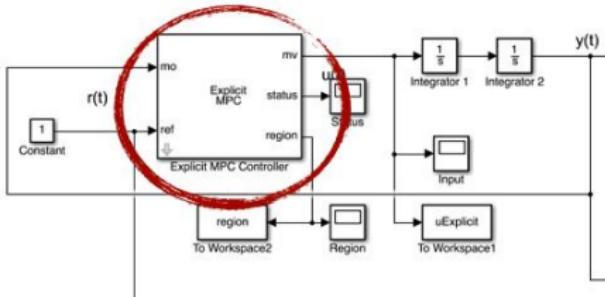
The MathWorks

```
>> 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 and robust 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;
```



Copyright 1990-2014 The MathWorks, Inc.

# APPLICABILITY OF EXPLICIT MPC

- Consider the following general MPC formulation

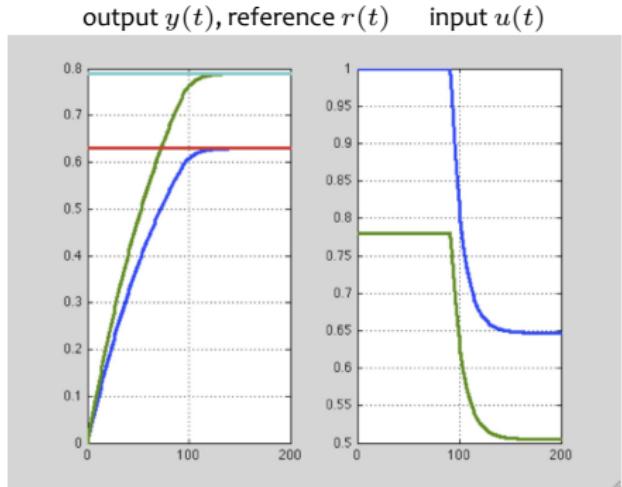
$$\min_z \quad \sum_{k=0}^{N-1} \frac{1}{2} (y_k - \mathbf{r}(t+k))' S (y_k - \mathbf{r}(t+k)) + \frac{1}{2} \Delta u_k' T \Delta u_k \\ + (u_k - \mathbf{u}_r(t+k))' V (u_k - \mathbf{u}_r(t+k))' + \rho_\epsilon \epsilon^2$$

subj. to  $x_{k+1} = Ax_k + Bu_k + B_v \mathbf{v}(t+k), k = 0, \dots, N-1$   
 $y_k = Cx_k + Du_k + D_v \mathbf{v}(t+k), k = 0, \dots, N-1$   
 $u_{\min}(t+k) \leq u_k \leq u_{\max}(t+k), k = 0, \dots, N-1$   
 $\Delta u_{\min}(t+k) \leq \Delta u_k \leq \Delta u_{\max}(t+k), k = 0, \dots, N-1$   
 $y_{\min}(t+k) - \epsilon V_{\min} \leq y_k \leq y_{\max}(t+k) + \epsilon V_{\max}, k = 1, \dots, N$   
 $x_0 = \mathbf{x}(t)$

- Everything marked in red can be time-varying in explicit MPC
- Not applicable to time-varying models and weights

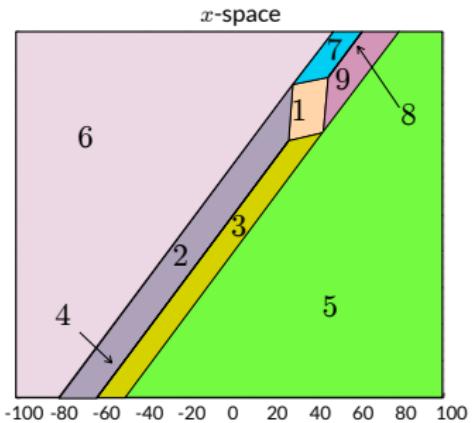
# EXPLICIT MPC EXAMPLE - MIMO SYSTEM

- Linear MIMO system  $y(t) = \frac{10}{100s+1} \begin{bmatrix} 4 & -5 \\ -3 & 4 \end{bmatrix} u(t)$ , sampled with  $T_s = 1$  s
- Input constraints  $\begin{bmatrix} -1 \\ -1 \end{bmatrix} \leq u(t) \leq \begin{bmatrix} 1 \\ 1 \end{bmatrix}$
- MPC tuning:  $N = 20$ ,  $N_u = 1$ , stage cost  $\|y_k - r(t)\|_2^2 + \frac{1}{10}\|\Delta u_k\|_2^2$



go to demo  
linear/mimo.m  
(Hybrid Toolbox)

# EXPLICIT MPC EXAMPLE - MIMO SYSTEM

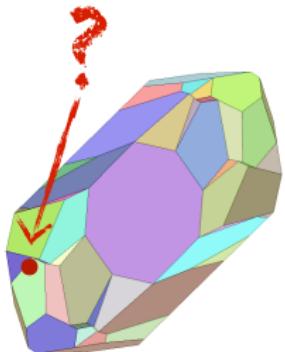


Polyhedral partition of the state-space for  $u(t-1) = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$  and  $r(t) = \begin{bmatrix} 0.63 \\ 0.79 \end{bmatrix}$

$$\delta u = \left\{ \begin{array}{ll} \begin{aligned} & [-0.1251 \quad 0.0084] x + [0.8535 \quad 0.1143] u \\ & + [0.0168 \quad -0.0668] r \\ & + [2.5583 \quad 3.1741] \end{aligned} & \text{if } \begin{aligned} & [-0.1251 \quad 0.0084] x + [0.1465 \quad 0.1143] u \\ & + [0.0168 \quad -0.0668] r \\ & + [2.5583 \quad 3.1741] \end{aligned} \\ & \leq \begin{bmatrix} 1.0000 \\ 1.0000 \\ -1.0000 \end{bmatrix} \end{array} \right. \quad (\text{Region } \#1) \\ \begin{array}{ll} \begin{aligned} & [0.0000 \quad 0.0000] x + [-1.0000 \quad 0.0000] u \\ & + [-0.0000 \quad 0.0000] r + [1.0000] \\ & + [0.1016 \quad 0.0813] \end{aligned} & \text{if } \begin{aligned} & [-0.0643 \quad 0.0412] x + [0.0000 \quad -0.0001] u \\ & + [0.1251 \quad -0.0668] r \\ & + [0.4623 \quad 0.3572] \end{aligned} \\ & \leq \begin{bmatrix} 1.0000 \\ -1.0000 \\ 1.0000 \end{bmatrix} \end{array} \quad (\text{Region } \#2) \\ \begin{array}{ll} \begin{aligned} & [-0.1466 \quad 0.0938] x + [-0.9998 \quad -0.0000] u \\ & + [0.0000 \quad 0.0000] r \\ & + [0.1333 \quad -0.0999] \end{aligned} & \text{if } \begin{aligned} & [-0.5239 \quad 0.3353] x + [0.0007 \quad -0.0000] u \\ & + [0.0168 \quad 0.0668] r \\ & + [-0.0584 \quad 0.0438] \end{aligned} \\ & \leq \begin{bmatrix} 1.0000 \\ -1.0000 \\ -1.0000 \end{bmatrix} \end{array} \quad (\text{Region } \#3) \\ \begin{array}{ll} \begin{aligned} & [-1.0000 \quad 0.0000] u + [1.0000] \\ & + [-0.0000 \quad -1.0000] \end{aligned} & \text{if } \begin{aligned} & [-0.5239 \quad -0.3353] x + [-0.0007 \quad 0.0000] u \\ & + [-0.4763 \quad -0.3572] r \end{aligned} \\ & \leq \begin{bmatrix} 1.0000 \\ -1.0000 \end{bmatrix} \end{array} \quad (\text{Region } \#4) \\ \begin{array}{ll} \begin{aligned} & [-1.0000 \quad 0.0000] u + [-1.0000] \\ & + [-0.0000 \quad -1.0000] \end{aligned} & \text{if } \begin{aligned} & [-0.0643 \quad 0.0412] x + [0.0000 \quad -0.0001] u \\ & + [0.0584 \quad -0.0456] r \end{aligned} \\ & \leq \begin{bmatrix} 1.0000 \\ -1.0000 \end{bmatrix} \end{array} \quad (\text{Region } \#5) \\ \begin{array}{ll} \begin{aligned} & [-1.0000 \quad 0.0000] u + [1.0000] \\ & + [-0.0000 \quad -1.0000] \end{aligned} & \text{if } \begin{aligned} & [0.0643 \quad -0.0412] x + [-0.0001 \quad 0.0000] u \\ & + [-0.0584 \quad 0.0438] r \end{aligned} \\ & \leq \begin{bmatrix} 1.0000 \\ -1.0000 \end{bmatrix} \end{array} \quad (\text{Region } \#6) \\ \begin{array}{ll} \begin{aligned} & [0.0000 \quad 0.0000] x + [-1.0000 \quad 0.0000] u \\ & + [0.1144 \quad -0.0733] r \\ & + [-0.1016 \quad 0.0813] \end{aligned} & \text{if } \begin{aligned} & [0.0643 \quad 0.0412] x + [0.0000 \quad 0.0000] u \\ & + [-0.1251 \quad 0.0084] r \\ & + [-0.0570 \quad 0.0456] \end{aligned} \\ & + [2.5583 \quad 3.1741] \end{aligned} & \leq \begin{bmatrix} 1.0000 \\ 1.0000 \\ -1.0000 \end{bmatrix} \end{array} \quad (\text{Region } \#7) \\ \begin{array}{ll} \begin{aligned} & [-0.1466 \quad 0.0938] x + [-0.9998 \quad -0.0000] u \\ & + [0.0000 \quad 0.0000] r \\ & + [0.1333 \quad -0.0999] \end{aligned} & \text{if } \begin{aligned} & [-0.0643 \quad 0.0411] x + [0.0001 \quad -0.0000] u \\ & + [0.0168 \quad -0.0668] r \\ & + [-0.4763 \quad 0.3572] \end{aligned} \\ & \leq \begin{bmatrix} 1.0000 \\ -1.0000 \\ 1.0000 \end{bmatrix} \end{array} \quad (\text{Region } \#8) \\ \begin{array}{ll} \begin{aligned} & [-1.0000 \quad 0.0000] u + [-1.0000] \\ & + [-0.0000 \quad -1.0000] \end{aligned} & \text{if } \begin{aligned} & [-0.5239 \quad 0.3353] x + [0.0007 \quad -0.0000] u \\ & + [0.4763 \quad 0.3572] r \end{aligned} \\ & \leq \begin{bmatrix} 1.0000 \\ -1.0000 \end{bmatrix} \end{array} \quad (\text{Region } \#9) \end{array}$$

# POINT LOCATION PROBLEM

Which is the region the current  $x(t)$  belongs to ?



## Approaches:

- 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 neighbor 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)

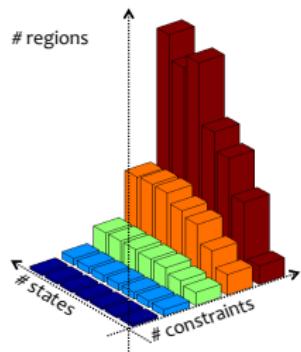
# COMPLEXITY OF MULTIPARAMETRIC SOLUTIONS

- Number  $n_r$  of regions = # optimal combinations of active constraints:

- mainly depends on the 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 the number  $s$  of free variables
- weakly depends on the number  $n$  of parameters (states + references)

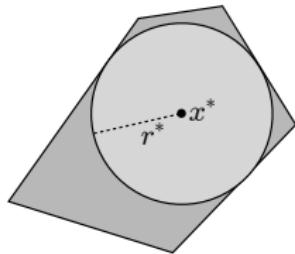
| states/horizon | $N = 1$ | $N = 2$ | $N = 3$ | $N = 4$ | $N = 5$ |
|----------------|---------|---------|---------|---------|---------|
| $n=2$          | 3       | 6.7     | 13.5    | 21.4    | 19.3    |
| $n=3$          | 3       | 6.9     | 17      | 37.3    | 77      |
| $n=4$          | 3       | 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   |

average on 20 random SISO systems w/ input saturation



# SUBOPTIMAL SOLUTIONS - INTERPOLATION METHODS

- Possible “holes” may appear in mpQP solution due to polyhedral computations, or after eliminating “flat” regions (=small Chebychev radius  $r^*$ )



- Safe PWA evaluation** function implemented in MPC Toolbox:  
set  $u(x) = F_i x + g_i$  with

$$i = \arg \min \beta_i(x), \quad \beta_i(x) = \max_j \{H_i^j x - K_i^j\}$$

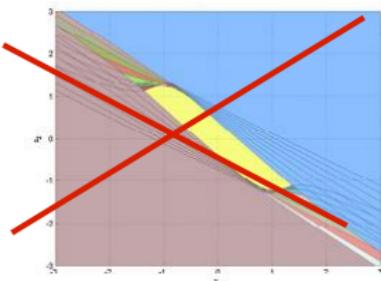
least violation

- Other approaches exist that enumerate offline the  $L$  most visited regions, store the corresponding gains, then interpolate or extrapolate online  
(Pannocchia, Rawlings, Wright, 2007) (Christophersen, Zeilinger, Jones, Morari, 2007)  
(Alessio, Bemporad, 2008) (Jones, Morari, 2010) (Kvasnica, Fikar, 2010)

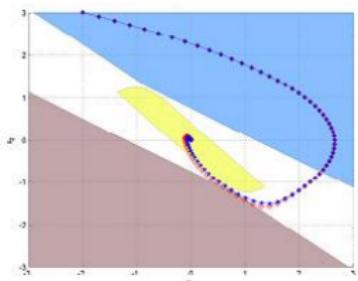
# SUBOPTIMAL SOLUTIONS - INTERPOLATION METHODS

(Alessio, Bemporad, 2008)

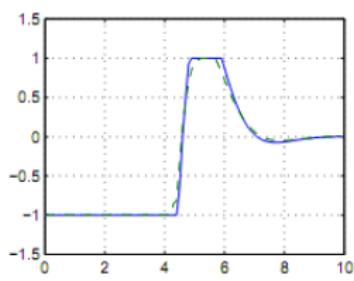
exact solution (99 regions)



$L=3$  most visited regions stored



input trajectory



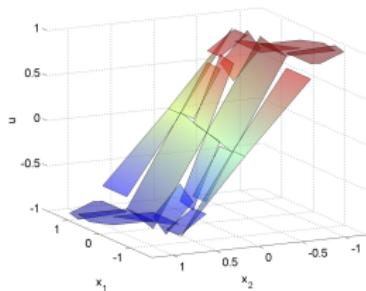
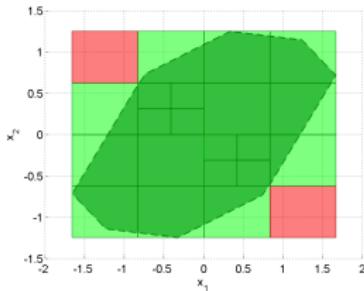
- Weighted interpolation approach

$$u(x) = \begin{cases} F_i x + g_i & \text{if } \beta_i(x) = 0 \quad (\beta(x) = \max_j \{H_i^j x - K_i\}) \\ \left( \sum_{i=1}^L \frac{1}{\beta_i(x)} \right)^{-1} \sum_{i=1}^L \frac{1}{\beta_i(x)} (F_i x + g_i) & \text{otherwise} \end{cases}$$

- Saturation of  $u(x)$  might be enforced to ensure hard input constraints

# SUBOPTIMAL SOLUTIONS - OTHER APPROACHES

- Relax KKT conditions (e.g., remove  $\tilde{\lambda}(x) \geq 0$ ) and solve mpQP suboptimally  
(Bemporad, Filippi, 2003)
- Change cost function (e.g., minimum time) (Grieder, Morari, 2003)
- Use **orthogonal trees** to approximate solution  
(Johansen, Grancharova, 2003) (Liang, Heemels, Bemporad, 2011)



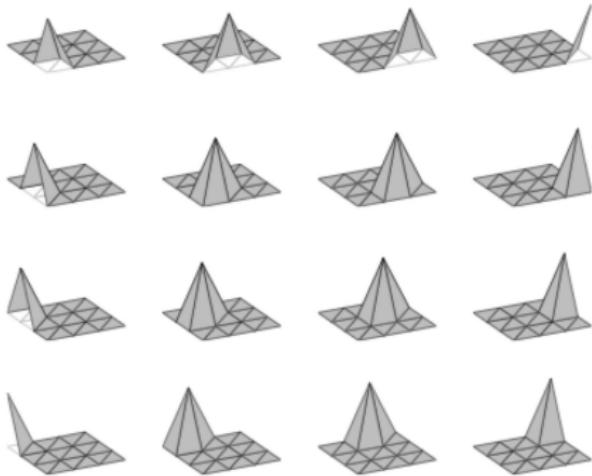
- Use gridding methods from **dynamic programming**

# SUBOPTIMAL SOLUTIONS - FUNCTION REGRESSION

- We can use any **function regression** technique to approximate explicit MPC
  - Collect  $M$  samples  $(x_i, u_i)$  by solving a QP (or evaluating the explicit solution) for each  $x_i$
  - Use function regression to approximate a mapping  $\hat{u}(x)$  that fits the samples
  - Check performance / prove closed-loop stability (if possible)
- Possible function regression approaches:
  - **Lookup tables** (linear interpolation, inverse distance weighting, ... )
  - **Neural networks** (Parisini, Zoppoli, 1995)
  - **Hybrid system identification / PWA regression** (Breschi, Piga, Bemporad, 2016)
  - **Nonlinear systems identification** (Canale, Fagiano, Milanese, 2008)

# SUBOPTIMAL SOLUTIONS - PWAS APPROXIMATION

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



(Julian, Desages, Agamennoni, 1999)

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

approximate MPC Law

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



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



# SUBOPTIMAL SOLUTIONS - PWAS APPROXIMATION



- PWAS functions can be directly implemented on **FPGA / ASIC**
- Example: MPC of MIMO system  $\begin{cases} x_{k+1} = \begin{bmatrix} 1.2 & 1 \\ 0 & 1.1 \end{bmatrix} x_k + \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix} u_k \\ y_k = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} x_k \end{cases}$  with constraints on  $u$  and  $y$  ( $N = 5$ ,  $N_u = 5$ )

| fit criterion | $p$ | latency [ns]<br>serial | latency [ns]<br>parallel |
|---------------|-----|------------------------|--------------------------|
| $L^2$         | 7   | 170                    | 31                       |
|               | 15  | 210                    | 43                       |
|               | 31  | 238                    | 45                       |
|               | 63  | 272                    | 46                       |
| $L^\infty$    | 7   | 170                    | 31                       |
|               | 15  | 210                    | 43                       |
|               | 31  | 238                    | 45                       |
|               | 63  | 272                    | 46                       |

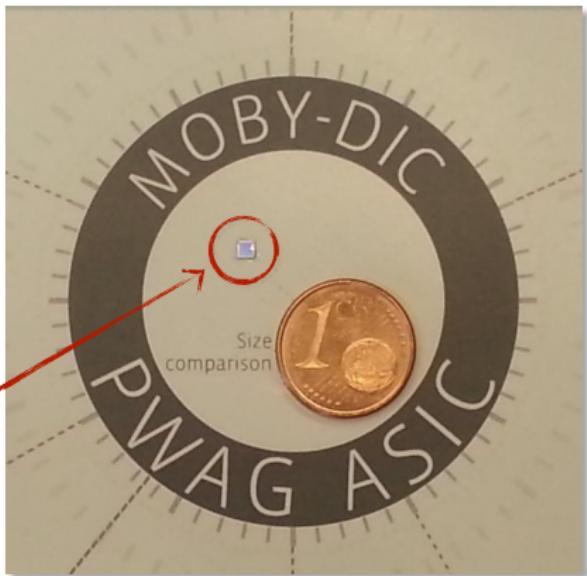
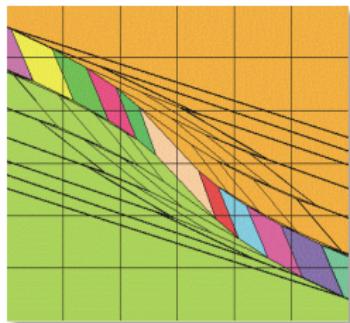
- latency = time to evaluate approximate MPC law on line (Xilinx Spartan 3 FPGA)
- $p = \#$  partitions per state dimension
- Exact explicit MPC (52 regions): 383 ns (avg), 486 ns (max)
- Function evaluation is **extremely fast !**

- Closed-loop stability can be proved (Bemporad, Oliveri, Poggi, Storace, 2011)

(Rubagotti, Barcelli, Bemporad, 2012)

- Main limitation: **curse of dimensionality** with respect to state dimension

# HARDWARE (ASIC) IMPLEMENTATION OF EXPLICIT MPC



Technology: 90 nm, 9 metal-layer from Taiwan Semiconductor Manufacturing Company  
Chip size: 1860x1860  $\mu\text{m}^2$   
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/>

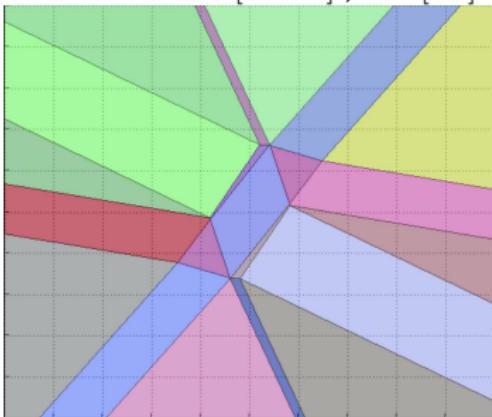
# EXAMPLE: AFTI-F16 AIRCRAFT

Linearized model:

$$\left\{ \begin{array}{lcl} \dot{x} & = & \begin{bmatrix} -0.0151 & -60.5651 & 0 & -32.174 \\ -0.0001 & -1.3411 & 0.9929 & 0 \\ 0.00018 & 43.2541 & -0.86939 & 0 \\ 0 & 0 & 1 & 0 \\ -2.516 & -13.136 & & \\ -0.1689 & -0.2514 & & \\ -17.251 & -1.5766 & & \\ 0 & 0 & & \end{bmatrix} x + \\ y & = & \begin{bmatrix} 0 & 1 & 0 & 0 \end{bmatrix} x \end{array} \right.$$



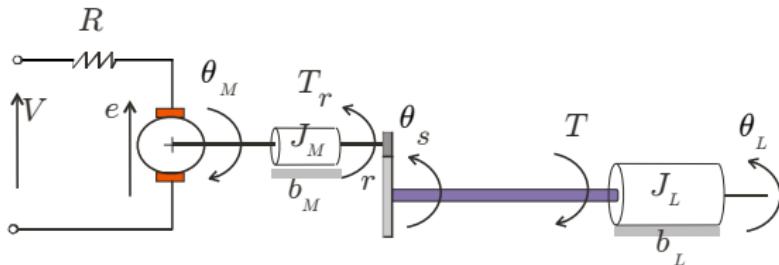
Section with  $x = [0 \ 0 \ 0 \ 0]', u = [0 \ 0]'$



Explicit MPC: 8 parameters, 51 regions

go to demo linear/afti16.m (Hybrid Toolbox)  
see also afti16.m (MPC Toolbox)

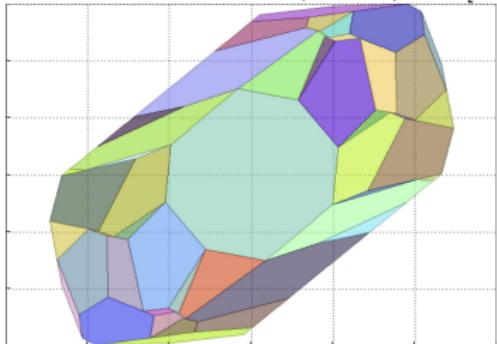
# EXAMPLE: DC SERVOMOTOR



- $N = 10, N_u = 2$
- $W^y = \begin{bmatrix} 0.1 & 0 \\ 0 & 0 \end{bmatrix}$
- $-220 \leq u \leq 220 \text{ V}$
- $-78.5398 \leq y_2 \leq 78.5398 \text{ Nm}$

Explicit MPC: 7 parameters, 101 regions

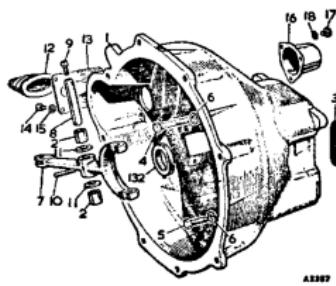
Section with  $x_1 = x_4 = 0, u = 0, r = [0\ 0]'$



go to demo `linear/dcmotor.m` (Hybrid Toolbox)  
see also `mpcmotor.m` (MPC Toolbox)

# EXPLICIT MPC FOR DRY-CLUTCH ENGAGEMENT

(Bemporad, Borrelli, Glielmo, Vasca, 2001)



- Model during slip phase ( $\omega_e > \omega_v$ ):

$$\begin{aligned} I_e \dot{\omega}_e &= T_{in} - b_e \omega_e - T_{cl} \\ I_v \omega_v &= T_{cl} - b_v \omega_v - T_l \\ T_{cl} &= k F_n \operatorname{sign}(\omega_e - \omega_v) \end{aligned}$$

- Model when clutch is engaged ( $\omega_e = \omega_v = \omega$ ):

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

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

|            |                                 |
|------------|---------------------------------|
| $I_e$      | engine inertia                  |
| $\omega_e$ | crankshaft rotor speed          |
| $T_{in}$   | engine torque                   |
| $b_e$      | crankshaft friction coefficient |
| $T_{cl}$   | torque transmitted by clutch    |
| $I_v$      | vehicle inertia                 |
| $\omega_v$ | clutch disk rotor speed         |
| $b_v$      | clutch viscous coefficient      |
| $T_l$      | equivalent load torque          |

# EXPLICIT MPC FOR DRY-CLUTCH ENGAGEMENT

- Linear model during slip + disturbance models

$$\begin{aligned}\dot{x}_1 &= -\frac{b_e}{I_e}x_1 + \frac{T_{in}}{I_e} - \frac{k}{I_e}u \\ \dot{x}_2 &= \left(-\frac{b_e}{I_e} + \frac{b_v}{I_v}\right)x_1 - \frac{b_v}{I_v}x_2 + \frac{T_{in}}{I_e} + \frac{T_l}{I_v} - \left(\frac{k}{I_e} + \frac{k}{I_v}\right)u \\ \ddot{T}_{in} &= 0 \quad \text{unknown ramp} \\ \dot{T}_l &= 0 \quad \text{unknown constant}\end{aligned}$$

$x_1 = \omega_e$   
 $x_2 = \omega_e - \omega_v$   
 $u = F_n$

- Model sampled and converted to discrete time ( $T_s = 10 \text{ ms}$ )

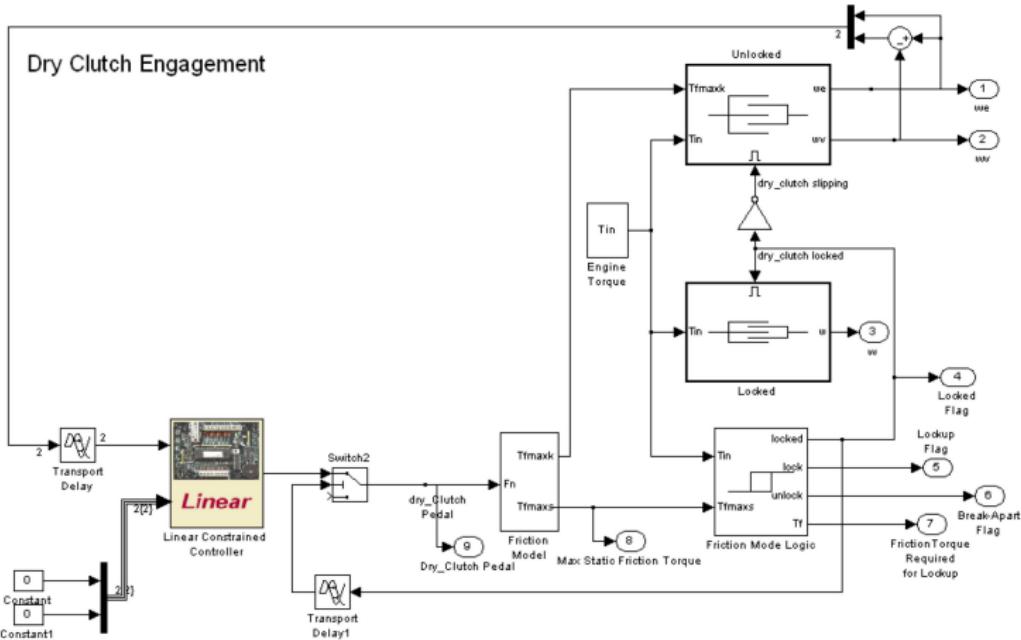
$$\begin{bmatrix} x_1(t+1) \\ x_2(t+1) \\ T_{in}(t+1) \\ T_l(t+1) \\ u(t) \end{bmatrix} = \begin{bmatrix} 0.9985 & 0 & 0.0500 & 0 & -0.0049 \\ -0.0011 & 0.9996 & 0.0500 & 0.0129 & -0.0062 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \\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

- Control objective

$$\min_{\Delta u_0, \dots, \Delta u_{N_u-1}} \sum_{k=0}^{N-1} Qx_{2,k}^2 + R\Delta u_k^2 + x_N'Px_N$$

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

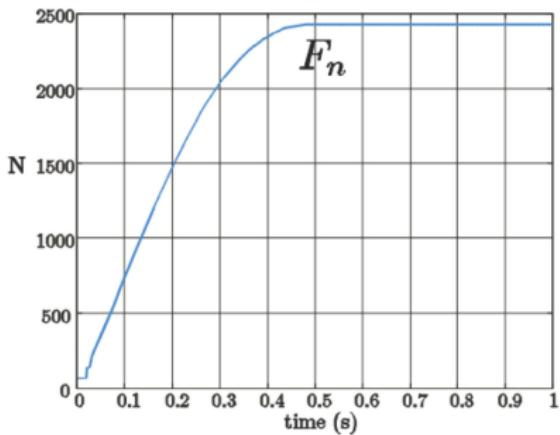
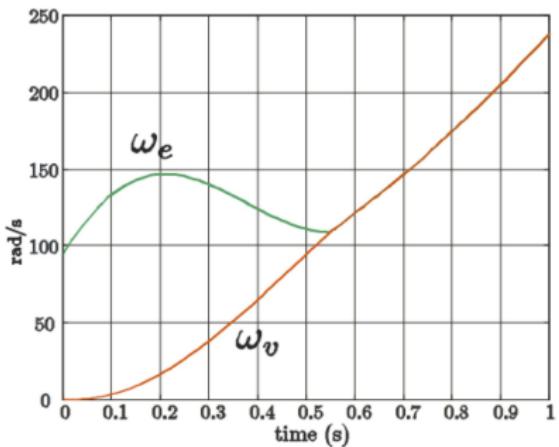
# EXPLICIT MPC FOR DRY-CLUTCH ENGAGEMENT



go to demo `dryclutch/dryclutch.m` (Hybrid Toolbox)

# EXPLICIT MPC FOR DRY-CLUTCH ENGAGEMENT

- Simulation results



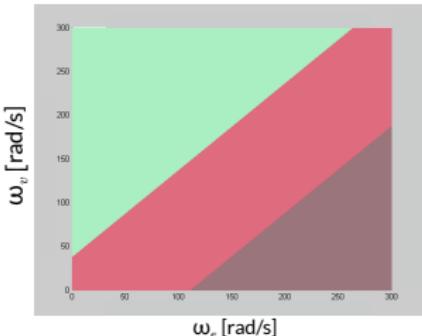
| tuning | $\tau^*$ | $F_n(\tau^*)$ [Nm] | $E_d$ [kJ] | $\dot{x}_2(\tau^*)$ [rad/s <sup>2</sup> ] |
|--------|----------|--------------------|------------|-------------------------------------------|
| 1      | 0.57     | 2333               | 6.676      | 230                                       |
| 2      | 0.76     | 2515               | 9.777      | 109                                       |
| 3      | 0.83     | 2572               | 10.838     | 62                                        |

MPC tuning #1 is chosen,  
corresponding to  $Q = 2$ ,  
 $R = 1$ ,  $N = 10$ ,  $N_u = 1$

# EXPLICIT MPC FOR DRY-CLUTCH ENGAGEMENT

- Explicit MPC law (+linear observer):

$$\Delta u = \begin{cases} \begin{aligned} & \left[ -0.003786 \ 0.5396 \ 0.1703 \ 0.006058 \ 0.04411 \right] x \quad \text{if} \\ & \left[ -0.02101 \right] u + \left[ 0 \ -0.5409 \right] r \end{aligned} & \begin{aligned} & \left[ -4.732e-05 \ 0.006745 \ 0.002129 \ 7.572e-05 \ 0.0005513 \right] x \\ & + \left[ 0.003786 \ -0.5396 \ -0.1703 \ -0.006058 \ -0.04411 \right] \\ & + \left[ -0.0002626 \right] u + \left[ 0 \ -0.006762 \right] r \leq \left[ \begin{smallmatrix} 1 \\ 0 \end{smallmatrix} \right] \end{aligned} \\ (Region \#1) \end{cases}$$
$$80 \quad \begin{aligned} & \text{if} \quad \left[ 4.732e-05 \ -0.006745 \ -0.002129 \ -7.572e-05 \ -0.0005513 \right] x \\ & + \left[ 0.0002626 \right] u + \left[ 0 \ 0.006762 \right] r \leq [-1] \\ (Region \#2) \end{aligned}$$
$$0 \quad \begin{aligned} & \text{if} \quad \left[ -0.003786 \ 0.5396 \ 0.1703 \ 0.006058 \ 0.04411 \right] x \\ & \left[ -0.02101 \right] u + \left[ 0 \ -0.5409 \right] r \leq [0] \\ (Region \#3) \end{aligned}$$



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

Alternative: use **explicit hybrid MPC**  
based on switching model  
(slipping mode, engaged mode)

# EXPLICIT MPC FOR IDLE SPEED CONTROL

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

- Main goal: regulate engine speed at idle

- Process model:

- 1 output (engine speed)
- 2 inputs (airflow, spark advance)
- input delays

- Objectives and constraints

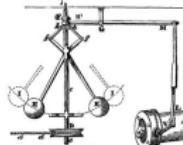
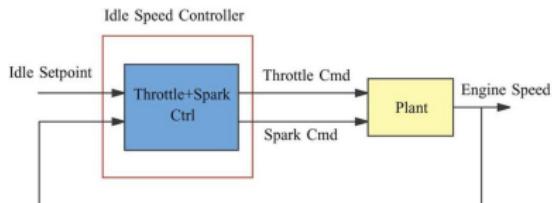
- regulate engine speed at constant rpm
- saturation limits on airflow and spark
- lower bound on engine speed  $\geq 450$  rpm

- Problem suitable for MPC design

(Hrovat, 1996)



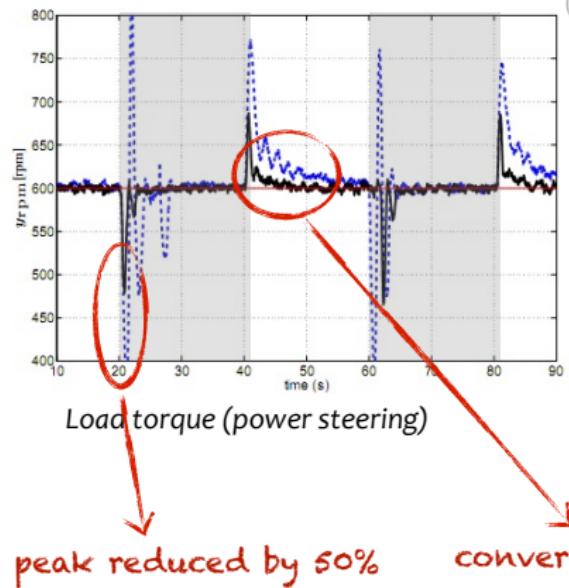
Ford pickup truck, V8 4.6L gasoline engine



centrifugal governor  
(Watt, 1788)

# EXPLICIT MPC FOR IDLE SPEED CONTROL

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

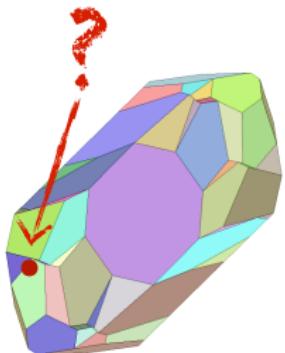


- Sampling time = 30 ms
- Explicit MPC implemented in dSPACE MicroAutoBox rapid prototyping unit
- **Observer tuning** as much important as tuning of MPC weights !



# COMPLEXITY OF MULTIPARAMETRIC SOLUTIONS

- The number of regions is (usually) exponential with the number of possible **combinations of active constraints**



- Too many regions make explicit MPC less attractive, due to **memory** (storage of polyhedra) and **throughput** requirements (time to locate  $x(t)$ )

When is explicit MPC preferable to on-line QP (=implicit MPC) ?

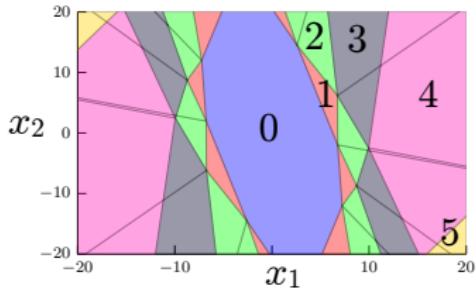
# COMPLEXITY CERTIFICATION FOR ACTIVE-SET QP SOLVERS

(Cimini, Bemporad, IEEE TAC, 2017)

- Consider a dual active-set QP method for solving the QP

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

- What is the worst-case number of iterations over  $x$  to solve the QP ?
- Key result:** The **number of iterations** to solve the QP is a **piecewise constant function** of the parameter  $x$



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, IEEE TAC, 2017)

- 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

online QP  
is faster

- Further improvements are possible by combining explicit and on-line QP

QP certification algorithm currently used in production

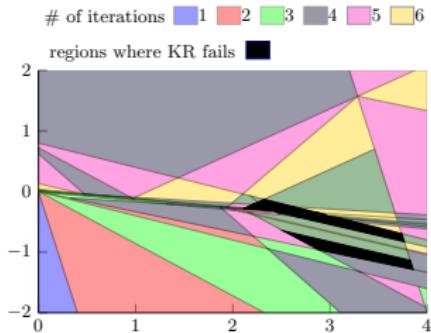


# CERTIFICATION OF KR SOLVER

(Cimini, Bemporad, 2019)

- The **KR algorithm** is a **simple** and **effective** solver for **box-constrained QP**.  
All violated/active constraints form the new active set at the next iteration  
(Kunisch, Rendl, 2003) (Hungerländer, Rendl, 2015)
- Assumptions for convergence are quite conservative, and indeed **KR can cycle**

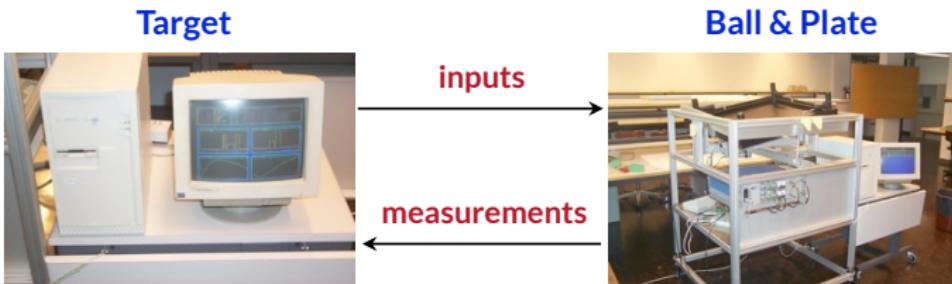
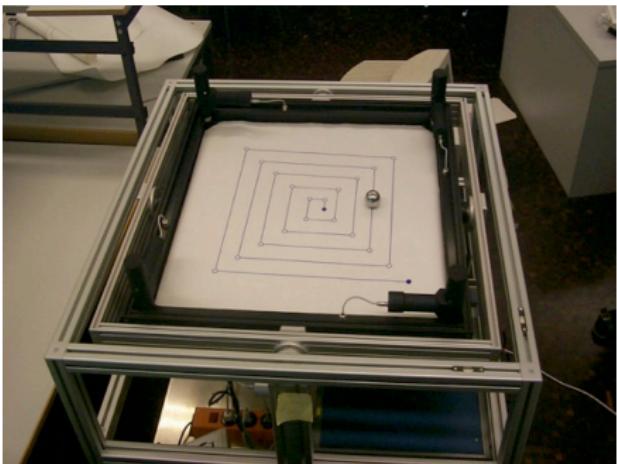
We can **exactly** map how many iterations KR takes to converge (or cycle)



|                  | Example 1 | Example 2 | Example 3 |
|------------------|-----------|-----------|-----------|
| Explicit MPC     |           |           |           |
| max flops        | 324       | 1830      | 5401      |
| max memory [kB]  | 3.97      | 15.9      | 89.69     |
| Dual active-set  |           |           |           |
| max flops + sqrt | 580 + 5   | 1922 + 13 | 3622 + 24 |
| max memory [kB]  | 8.21      | 8.63      | 8.90      |
| KR algorithm     |           |           |           |
| max flops        | 489       | 1454      | 2961      |
| max memory [kB]  | 3.19      | 3.39      | 3.51      |

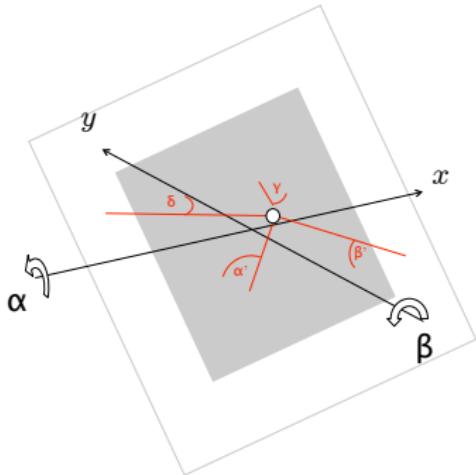
# MPC REGULATION OF A BALL ON A PLATE

- Goal: Regulate the position of a ball on a plate
- Experimental system @ETH Zurich
- Interface: Real-Time Workshop + xPC Toolbox



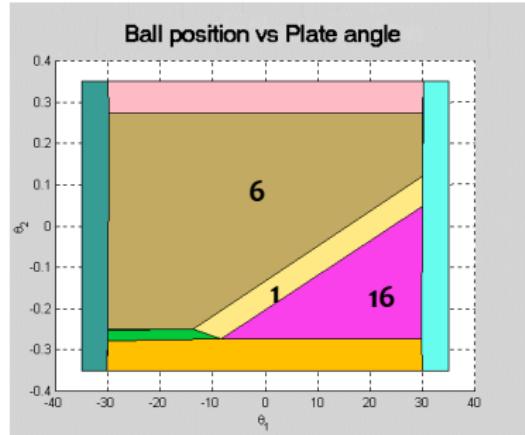
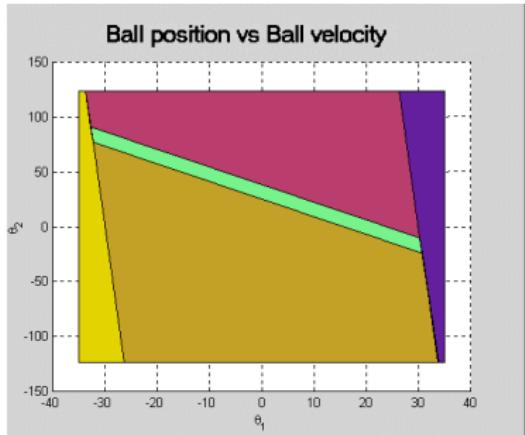
# BALL & PLATE: SPECIFICATIONS

- Constraints:
  - plate angular position:  $\pm 17$  deg (soft)
  - ball position on plate:  $\pm 30$  cm (soft)
  - input voltage:  $\pm 10$  V (hard)
- platform: PC Pentium 166
- sample time: 30 ms
- prediction model: LTI model,  $7 \times 2$  states
- MPC tuning:
  - prediction horizon  $N = 50$
  - control horizon  $N_u = 2$
  - weight on position error:  $(W^y)^2 = 5$
  - weight on input rate:  $W^{\Delta u} = 1$



# BALL & PLATE: EXPLICIT MPC SOLUTION

- PWA partitions: 22 regions ( $x$ -axis), 23 regions ( $y$ -axis)

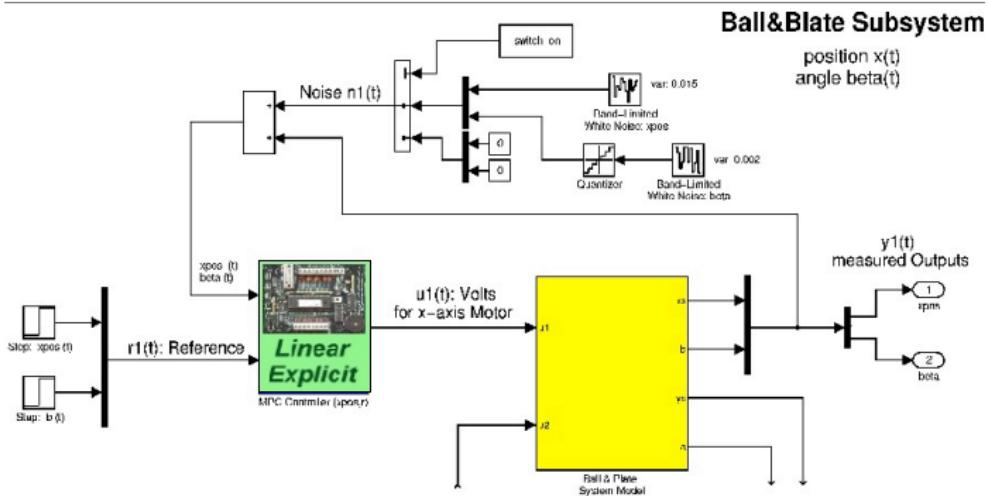


$x$ -axis: sections at  $\alpha_x = 0, \dot{\alpha}_x = 0, u_x = 0, r_x = 18, r_\alpha = 0$

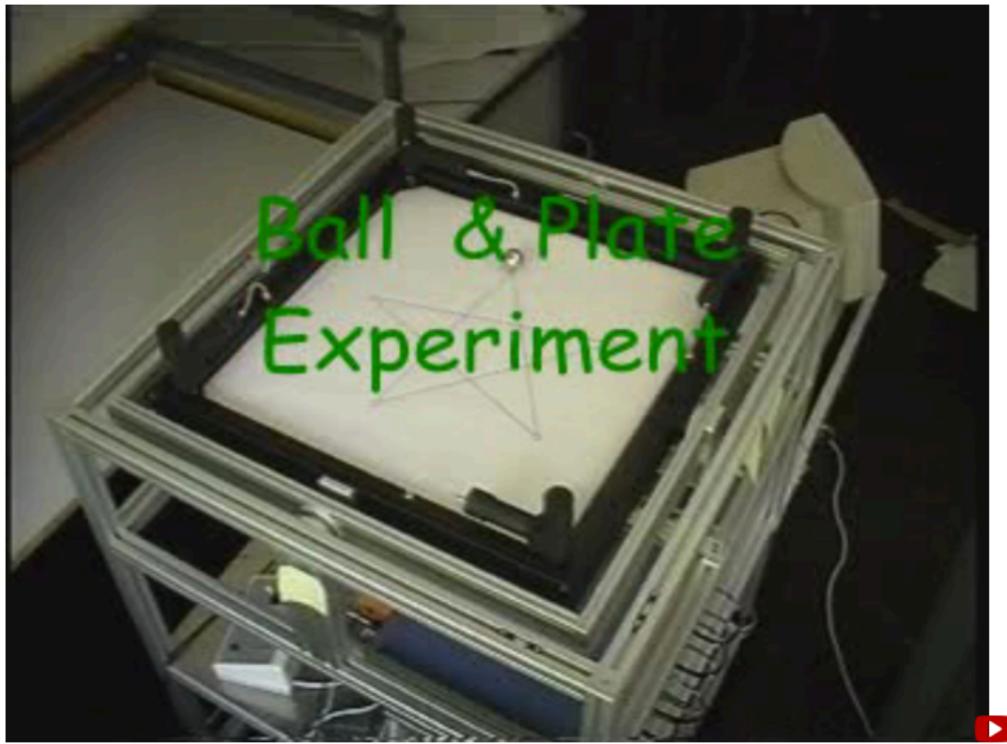
- Region #1: LQR controller
- Region #6: saturation at -10 V
- Region #16: saturation at +10 V

# BALL & PLATE: IMPLEMENTATION

- Solve mp-QP and implement explicit MPC

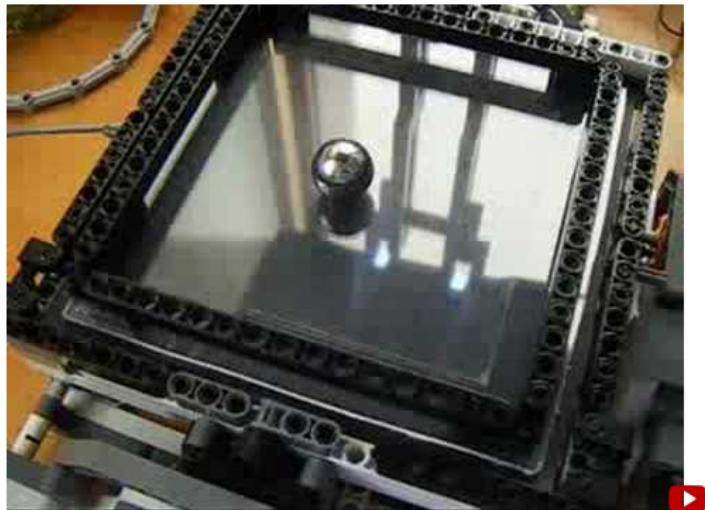


# BALL & PLATE: EXPERIMENTS



# BALL & PLATE: EXPERIMENTS

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



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

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

# MULTIPARAMETRIC QP IN PORTFOLIO OPTIMIZATION

(Markowitz, 1952) (Best, Grauer, *Management Science*, 1991)

- **Markowitz portfolio optimization:**

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

$z_i$  = money invested in asset  $i$

$p_i$  = expected return of asset  $i$

$\Sigma_{ij}$  = covariance of assets  $i, j$

$x$  = expected minimum return of portfolio

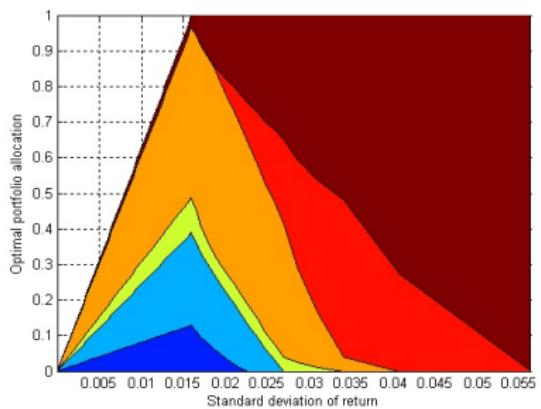
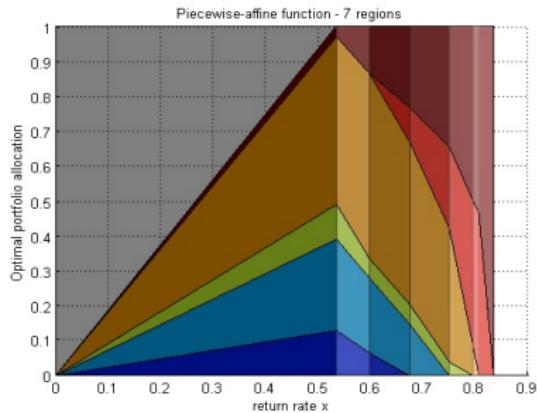
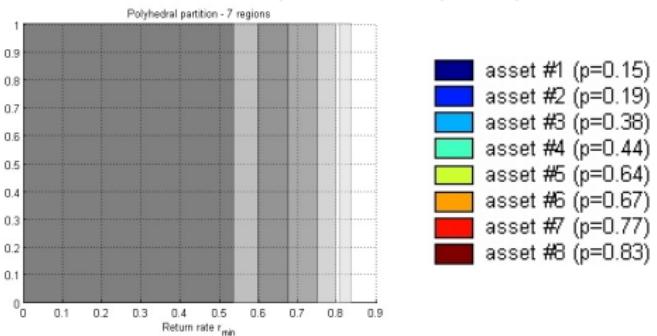
- Objective is to minimize variance (=risk)
- Constraints guarantee a minimum expected return

# MULTIPARAMETRIC QP IN PORTFOLIO OPTIMIZATION

(Bemporad, NMPC plenary, 2008)

- Solution of mpQP problem

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



# COMPARING DIFFERENT SOLUTION METHODS FOR MPC

Which method to prefer for embedded MPC ?

|                             | Explicit<br>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  $\leq$  6 vars, 15 constraints, 8 states/references

medium  $\approx$  20 vars, 80 constraints, 50 states/references

large  $\geq$  500 vars, 3000 constraints, 2000 states/references