MODEL PREDICTIVE CONTROL

QUADRATIC PROGRAMMING AND EXPLICIT MPC

Alberto Bemporad

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

IMT SCHOOL FOR ADVANCED STUDIES LUCCA

COURSE STRUCTURE

- ✔ **Basic concepts** of model predictive control (MPC) and **linear MPC**
- ✔ **Linear time-varying** and **nonlinear** MPC
- **Quadratic programming** (QP) and **explicit MPC**
- **Hybrid** MPC
- **Stochastic** MPC
- **Learning-based** MPC

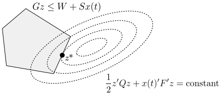
QUADRATIC PROGRAMMING (QP) SOLVERS FOR MPC

EMBEDDED LINEAR MPC AND QUADRATIC PROGRAMMING

- MPC based on linear models requires solving a **Quadratic Program (QP)**

$$\begin{array}{ll} \min\limits_{z} & \frac{1}{2}z'Qz + x'(t)F'z + \frac{1}{2}x'(t)Yx(t) \\ \text{s.t.} & Gz \leq W + Sx(t) \end{array}$$

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



ON MINIMIZING A CONVEX FUNCTION SUBJECT TO LINEAR INEQUALITIES

By E. M. L. BEALE

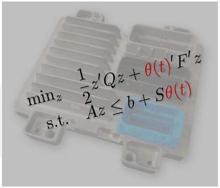
Admiralty Research Laboratory, Teddington, Middlesex

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)

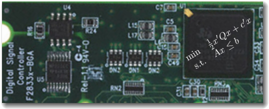
A rich set of good QP algorithms is available today



- Not all QP algorithms are suitable for **industrial embedded control**

MPC IN A PRODUCTION ENVIRONMENT

Key requirements for deploying MPC in production:



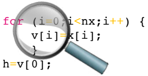
1. **speed (throughput)**
   - **worst-case** execution time less than sampling interval
   - also fast on **average** (to free the processor to execute other tasks)

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)

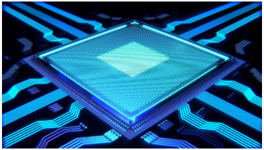


EMBEDDED SOLVERS IN INDUSTRIAL PRODUCTION

- 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

~520,000,000,000,000 QP/yr

and none of them should fail.

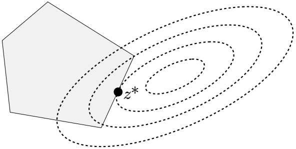


SOLUTION METHODS FOR QP

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

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

Quadratic Program (QP)



- **More material on QP solvers**:

  `http://cse.lab.imtlucca.it/~bemporad/optimization_course.html`

- Hybrid toolbox:

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

KKT OPTIMALITY CONDITIONS FOR QP

- Quadratic programming problem

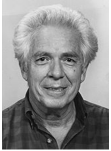
$$
\begin{aligned}
\min_z \quad & \frac{1}{2} z' Q z + x' F' z \\
\text{s.t.} \quad & Gz \leq W + Sx \\
& Ez = f
\end{aligned}
$$

- Karush-Kuhn-Tucker (KKT) conditions:

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

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

William Karush
(1917–1997)



Harold W. Kuhn
(1925–2014)

Albert W. Tucker
(1905–1995)

GRADIENT PROJECTION METHOD

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

- Optimization problem: $$\min_{z\in Z} f(z)$$ $f:\mathbb{R}^s\to\mathbb{R}$, $Z\subseteq\mathbb{R}^s$

- 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:

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

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

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

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

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

GRADIENT PROJECTION FOR BOX-CONSTRAINED QP

- Convex box-constrained QP

$$\begin{array}{ll} \min & \frac{1}{2}z'Qz + x'F'z \\ \text{s.t.} & \ell \leq z \leq u \end{array}$$

- Since $\|\nabla f(z_1) - \nabla f(z_2)\|_2 = \|Q(z_1 - z_2)\|_2 \leq \lambda_{\max}(Q)\|z_1 - z_2\|_2$ 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

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

- If $Q \succ 0$, we get **linear convergence**

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

DUAL GRADIENT PROJECTION FOR QP

- Consider the strictly convex QP and its dual

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

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

- Apply **proximal gradient method** to dual QP: (Combettes, Waijs, 2005)

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

- The primal solution is related to the dual solution by

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

- Convergence is slow: the initial error $f(z^0) - f(z^*)$ reduces as $1/k$

ACCELERATED GRADIENT PROJECTION METHOD

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

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

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

extrapolation step

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

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

- 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**:

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

FAST GRADIENT PROJECTION FOR (DUAL) QP

(Patrinos, Bemporad, 2014)

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

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

$$\begin{array}{rcl} K &=& Q^{-1}G' \\ J &=& Q^{-1}F \\ L &\geq& \lambda_{\max}(GQ^{-1}G') \end{array}$$

$$\beta_k = \max\{\tfrac{k-1}{k+2}, 0\}$$

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

```
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

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

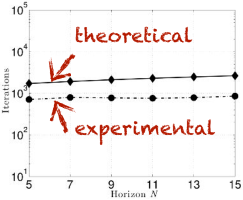
primal feasibility

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: $f(x^k) - f(x^*) \leq \dfrac{2L}{(k+2)^2}\|z_0 - z^*\|_2^2$



- Tight bounds on maximum number of iterations

- Can be useful to warm-start active-set methods (Bemporad, Paggi, 2015)

- 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 β_k (O'Donoghue, Candés , 2013)

- Restart conditions:

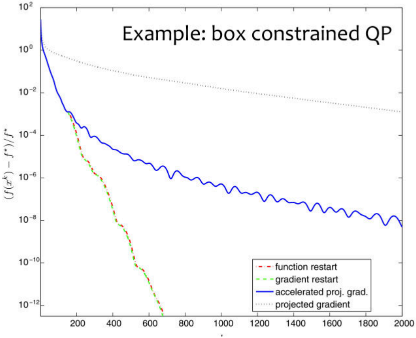
  – **function restart** whenever

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

  – **gradient restart** whenever

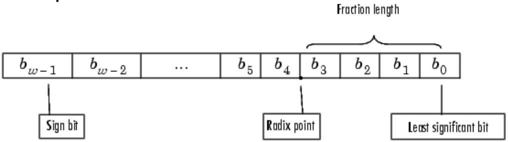
$$\nabla f(w^{k-1})'(y_k - y_{k-1}) > 0$$

  See also (Krupa, 2021)



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, Guiggiani, Bemporad, 2013)

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

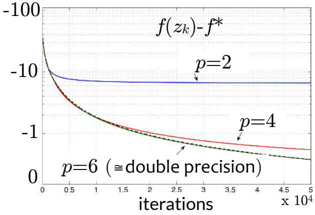
asymptotic feasibility

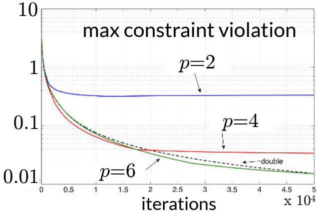
$$f(z_k)-f^\star \leq \frac{L}{2(k+1)}(\|y^\star\|^2+\|y_0\|^2)+\delta$$

asymptotic optimality

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

design guidelines (precision)



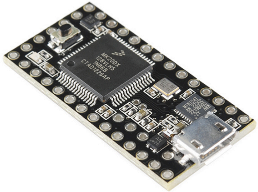


- Design guidelines for required #integer bits to avoid overflow also available

GRADIENT PROJECTION IN FIXED-POINT ARITHMETICS

- 32-bit Atmel SAM3X8E ARM Cortex-M3 processing unit (84 MHz, 512 kB of flash memory and 100 kB RAM)

  (Patrinos, Guiggiani, Bemporad, 2013)



fixed point

vars/constr.	CPU time (ms)	Code (kB)
10/20	22.9	15
20/40	52.9	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	220.1	21
40/80	2240	40
60/120	5816	73

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

- **Alternative**: formal certification of convergence and # iterations of the proximal-gradient method for box-constrained QP in fixed point arithmetics

  (Krupa, Inverso, Tribastone, Bemporad, 2024)

ADMM

(Gabay, Mercier, 1976) (Glowinski, Marrocco, 1975) (Douglas, Rachford, 1956) (Boyd et al., 2010)

- Alternating Directions Method of Multipliers for QP

$$\begin{array}{ll} \min & \frac{1}{2}z'Qz + c'z \\ \text{s.t.} & \ell \leq Az \leq u \end{array}$$

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

```
while k<maxiter
  k=k+1;
  z=-iM*(c+A'*(rho*(v-s)));
  Az=A*z;
  s=max(min(Az+v,u),ell);
  v=v+Az-s;
end
```

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

ρv = 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

(Stellato, Banjac, Goulart, Bemporad, Boyd, 2020)

- Robust "regularized" ADMM iterations:

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

- 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**

  http://osqp.org

  (Python interface: millions of downloads)
- Extended to solve **mixed-integer quadratic programming** problems

  (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)

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

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

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

$$\Rightarrow \quad \begin{array}{ll} \min & \frac{1}{2}y_s'(PHP)y_s + (Dx+W)'Py_s \\ \text{s.t.} & y_s \geq 0 \end{array}$$

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

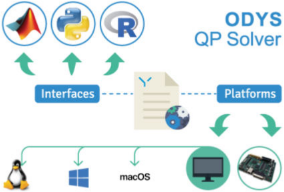
number of iterations (NS= Not Scaled, S=Scaled)

PQP = Projection-free parallel QP (Di Cairano, Brand, Bortoff, 2013)
DRSA = Accelerated Douglas-Rachford Splitting (Patrinos, Stella, Bemporad, 2014)
FBN = Forward-Backward Newton (Stella, Themelis, Patrinos, 2017)

ODYS QP SOLVER

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

$$\begin{array}{ll} \min\limits_{z} & \frac{1}{2}z'Qz + c'z \\ \text{s.t.} & b_\ell \leq Az \leq b_u \\ & \ell \leq z \leq u \\ & Ez = f \end{array}$$



- Implements a **proprietary** state-of-the-art method for QP
- Completely written in **ANSI-C**, it's **library-free** and **MISRA-C 2012** compliant
- **Fast**, **robust** (also in single precision), **low-memory** requirements
- **Optimized version for MPC** available ($\approx$ 50% faster)
- **Certifiable** execution time
- Licensed to several automotive OEMs and Tier-1 suppliers

odys.it/qp

MORE ON REQUIREMENTS FOR EMBEDDED OPTIMIZATION

- Need to distinguish between **offline** and **online** computations
  - **offline**: double precision (MATLAB/Python/Julia) on a PC (arbitrarily complex)
  - **online**: single-precision/fixed-point on a μ-processor (please simple!)
- One can often trade off **throughput vs. memory**
- Numerical robustness of online computations is important. The solver must perform well in **single precision** (or even fixed-point) arithmetics
- Limited linear algebra (online) (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$\div$50 variables, 10$\div$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)

$$\begin{array}{ll} \min_z & \|Az - b\|_2^2 \\ \text{s.t.} & z \geq 0 \end{array}$$

Bounded-Variable Least Squares (BVLS)

(Stark,Parker, 1995)

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

ACTIVE-SET METHOD FOR NNLS

(Lawson, Hanson, 1974)

- Active-set method to solve the NNLS problem $\min_{z\geq 0} \|Az-b\|_2^2,\ A\in\mathbb{R}^{m\times n}$

1. $\mathcal{P}\leftarrow\emptyset$, $z\leftarrow 0$;
2. $w\leftarrow A'(Az-b)$;
3. **if** $w\geq 0$ **or** $\mathcal{P}=\{1,\ldots,m\}$ **then go to** Step 10;
4. $i\leftarrow\arg\min_{i\in\{1,\ldots,m\}\setminus\mathcal{P}} w_i$, $\mathcal{P}\leftarrow\mathcal{P}\cup\{i\}$;
5. $y_{\mathcal{P}}\leftarrow\arg\min_{z_{\mathcal{P}}} \|((A')_{\mathcal{P}})'z_{\mathcal{P}}-b\|_2^2$,
   $y_{\{1,\ldots,m\}\setminus\mathcal{P}}\leftarrow 0$;
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', 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)

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

$$Q = Q' \succ 0$$

$$Q = L'L$$

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

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

- An LDP is equivalent to the **nonnegative least squares** (NNLS) problem

(Lawson, Hanson, 1974)

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

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

- If the residual $r^* = \left[\begin{smallmatrix} M' \\ d' \end{smallmatrix}\right] y^* + \left[\begin{smallmatrix} 0 \\ 1 \end{smallmatrix}\right] = 0$ then the original QP is infeasible. Otherwise set

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

RELATION BETWEEN NNLS PROBLEM AND DUAL QP

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

dual QP

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

can be unbounded if primal QP is infeasible

$(d = W + Dx)$

NNLS problem

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

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

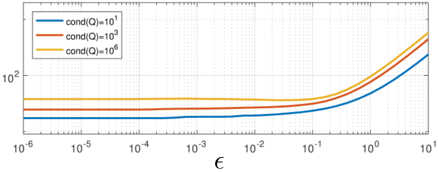
(Bemporad, 2018)

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

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

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

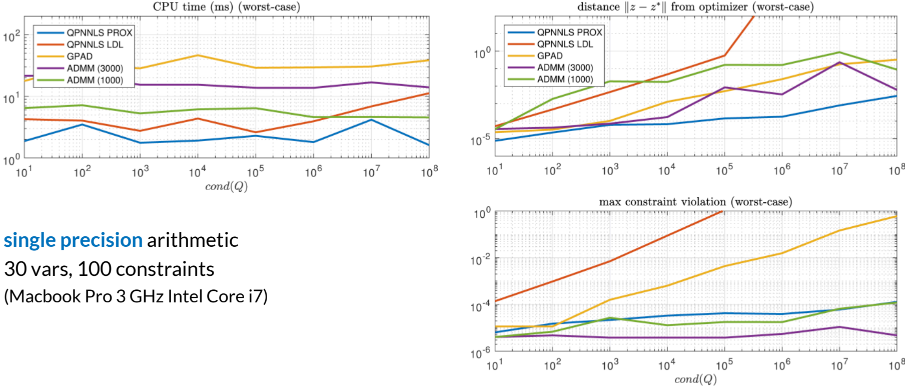
total number of active-set iterations as a function of ϵ



- 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 Karush-Kuhn-Tucker (KKT) optimality conditions for the convex QP

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

are

$$
\begin{array}{rclcl}
r_Q & = & Qx + c + E'y + A'z & = & 0 \\
r_E & = & Ex - f & = & 0 \\
r_A & = & Ax + s - b & = & 0 \\
r_S & = & [z_1 s_1 \ \ldots \ z_m s_m]' & = & 0 \\
z, s & \geq & 0 & &
\end{array}
$$

x = primal vars
y = dual vars (eq. constr.)
s = slacks (ineq. constr.)
z = dual vars (ineq. constr.)

- 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 $\mu \to 0$

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

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

- 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 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 (say >500 vars & constraints) IP well exploits sparse linear algebra
- Number of AS iterations can be **exactly certified** for given QP matrices (see later ...)

(see more in Nocedal-Wright's book, pp. 485 and 592-593)

LOSSLESS REDUCTION OF MPC PROBLEM VARIABLES

(Bemporad, Cimini, 2023)

- Linear (parameter-varying) MPC = **constrained least squares problem**

$$\begin{array}{rl} \min_z & \frac{1}{2}\|A(\theta)z - b(\theta)\|_2^2 \\ \text{s.t.} & C(\theta)z = e(\theta) \\ & G(\theta)z \leq g(\theta) \end{array} \qquad z = \begin{bmatrix} u_0 \\ x_1 \\ \vdots \\ u_{T-1} \\ x_T \end{bmatrix}, \quad \theta = \begin{bmatrix} x(t) \\ r(t) \\ \dots \end{bmatrix}$$

- **Standard condensing**: eliminate x_k, only keep u_k as optimization variables

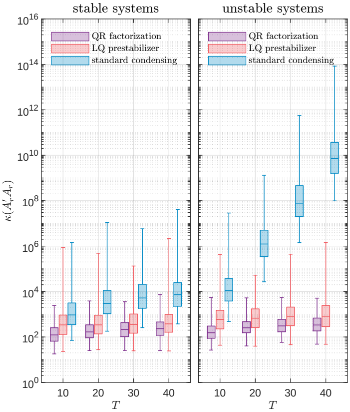
- **LQ prestabilizer**: apply $u_k = K_k x_k + v_k$ and condense

- **QR factorization**: $C' = [Q_1\ Q_2] \begin{bmatrix} R_1 \\ 0 \end{bmatrix}$

$z = Q_2(\theta)s + \bar{z}(\theta) \in \mathbb{R}^{T(n_x+n_u)}$

$s \in \mathbb{R}^{Tn_u}$ = new variables

➡ **numerically very robust**



LOSSY REDUCTION OF MPC PROBLEM VARIABLES

(Bemporad, Cimini, 2023)

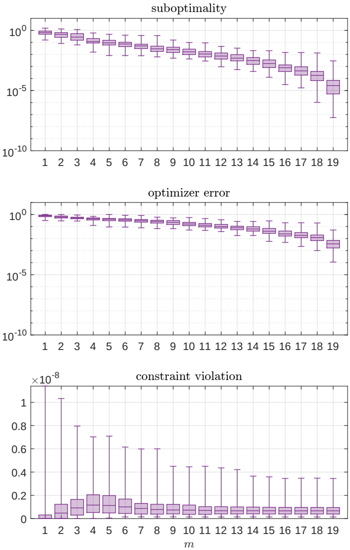
- **Control horizon** often used to reduce # vars
- **Idea**: **linear PCA** to reduce to $m < Tn_u$ vars
  - collect optimal solutions s_k^* for M given values of θ_k, remove mean $\bar{s}$
  - compute **Singular Value Decomposition**

$$S = \begin{bmatrix} s_1^* - \bar{s} & \ldots & s_M^* - \bar{s} \end{bmatrix}' = U\Sigma V'$$

  - keep only first m principal directions $\Phi = [V_1 \ \ldots \ V_m]$
  - new optimization vector $v \in \mathbb{R}^m$

$$s = \Phi v + \bar{s}$$

- Complexity / solution quality **tradeoff**



LOSSY REDUCTION OF MPC PROBLEM VARIABLES

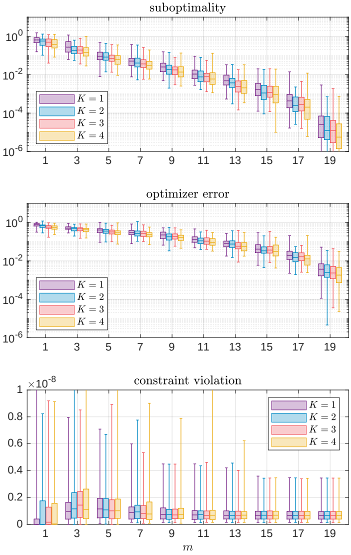
(Bemporad, Cimini, 2023)

- The optimal basis Φ is an average over all θ_k
- **K-SVD**: a new "K-means"-like algorithm to cluster $\theta_1, \ldots, \theta_M$ in K sets and get corresponding bases $\Phi_1,\ldots,\Phi_K$
- K-SVD converges in a finite number of steps to a local minimum of

$$\min_{J,\{\Phi_j,\phi_0^j\}_{j=1}^K} \sum_{i=1}^{M} \min_v \|s_i^* - \Phi_{J(i)}v - \phi_0^{J(i)}\|_2^2$$

$$\text{s.t.} \quad \phi_0^j = \frac{1}{M_j}\sum_{i\in I_j} s_i^*,\ j = 1,\ldots,K$$

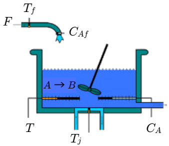
- K **neural one-to-all classifiers** trained to separate the resulting clusters

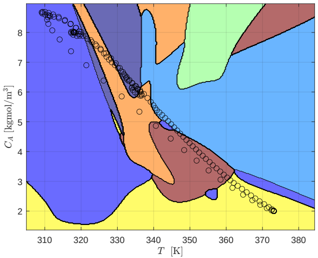


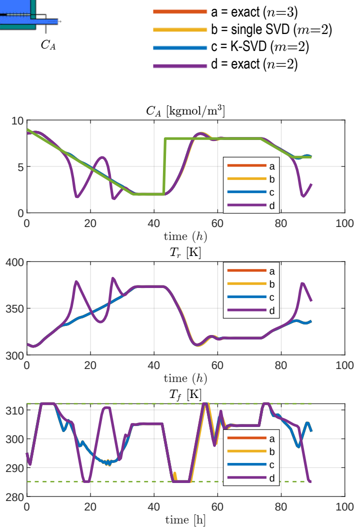
LOSSY REDUCTION OF MPC PROBLEM VARIABLES

(Bemporad, Cimini, 2023)

- Example: LPV-MPC control of CSTR
- M=10,000 samples, $K = 10$ clusters

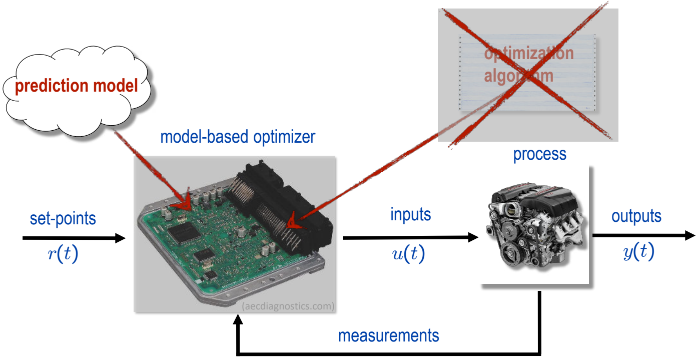


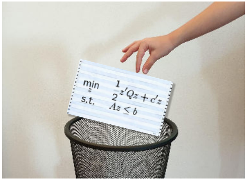




performance	value	MPC setting
$J_{\mathrm{exact},20}$	319.68	$N_u = 20 = T$, exact
$J_{\mathrm{exact},10}$	319.69	$N_u = 10$, exact
$J_{\mathrm{exact},4}$	319.93	$N_u = 4$, exact
$J_{\mathrm{exact},3}$	320.17	$N_u = 3$, exact
$J_{\mathrm{exact},2}$	567.86	$N_u = 2$, exact
J_{svd}	328.33	$N_u = 3$, $m = 2$ (single SVD)
J_{ksvd}	320.17	$N_u = 3$, $m = 2$ (K-SVD)

MPC WITHOUT ONLINE QP





- Can we implement constrained linear MPC **without an online QP solver** ?

YES !

ONLINE VS OFFLINE OPTIMIZATION

$$\begin{array}{ll} \min\limits_{z} & \frac{1}{2}z'Qz + x'(t)F'z + \frac{1}{2}x'(t)Yx(t) \\ \text{s.t.} & Gz \leq W + Sx(t) \end{array} \qquad z = \begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_{N-1} \end{bmatrix}$$

- **Online** 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)

- **Offline** 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

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

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** $V^*: X^* \to \mathbb{R}$ associating the optimal value $V^*(x)$ to each $x \in X^*$
- An **optimizer function** $z^*: X^* \to \mathbb{R}^s$ providing a solution for each $x \in X^*$

MULTIPARAMETRIC QUADRATIC PROGRAMMING (MPQP)

$$
\begin{array}{ll}
\min\limits_{z} & \frac{1}{2}z'Qz + x'F'z + \frac{1}{2}x'Yx \\
\text{s.t.} & Gz \leq W + Sx
\end{array}
\qquad
z = \begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_{N-1} \end{bmatrix}, \quad W \in \mathbb{R}^q
$$

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

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

$Q = Q' \succ 0$ 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

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

- 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 $z^*(x_0)$

$$\begin{array}{rcll} G_i z^*(x_0) & = & W_i + S_i x_0, & \forall i \in I(x_0) \\ G_i z^*(x_0) & < & W_i + S_i x_0, & \forall i \notin I(x_0) \end{array} \qquad I(x_0) \subseteq \{1, \ldots, q\}$$

- Consider now generic x, z, λ and the KKT optimality conditions for the QP

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

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

- Substitute into $\tilde{G}z - \tilde{W} - \tilde{S}x = 0$ and get

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

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

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

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

(Zafiriou, 1990)

MULTIPARAMETRIC QP ALGORITHM

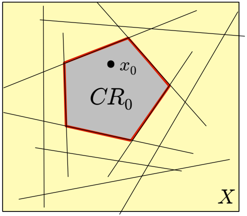
- Impose primal and dual feasibility

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

primal feasibility

dual feasibility

linear inequalities in x



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

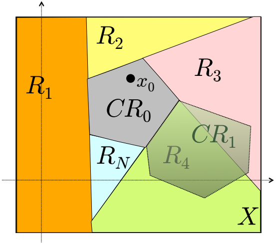
$$\begin{aligned} CR_0 &= \{x \in X : \hat{G}z(x) \leq \hat{W} + \hat{S}x,\ \tilde{\lambda}(x) \geq 0\} \\ &= \{x \in X : A_0 x \leq b_0\} \end{aligned}$$

- 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 infeasibile)

MULTIPARAMETRIC QP SOLVER #1

(Bemporad, Morari, Dua, Pistikopoulos, 2002)



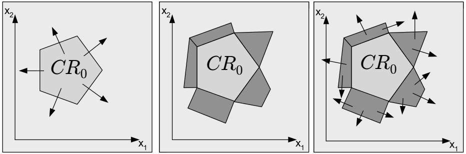
Method #1: Split X and proceed iteratively

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

- 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

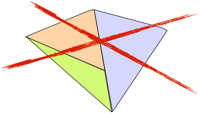
- **Method #2**: the active set of a neighboring region is obtained as follows:



(Tøndel, Johansen, Bemporad, 2003)

  - **add constraint** $\#i$ if the common facet comes from $\hat{G}^i(x) \leq \hat{W}^i + \hat{S}^i x$ (maintain feasibility of $z^*(x)$)
  - **remove constraint** $\#i$ if the common facet comes from $\tilde{\lambda}_j(x) \geq 0$ (maintain optimality of $z^*(x)$)

- **Method #3**: step out by ϵ outside each facet, solve QP, get new region, iterate (Baotic, 2002)
  The **facet-to-facet property** is required
  (Spjøtvold, Kerrigan, Jones, Tøndel, Johansen, 2006)



(Spjøtvold, 2008)

- **Method #4**: **implicit enumeration** of optimal active set combinations
  (Gupta, Bhartiya, Nataraj, 2011)

PROPERTIES OF MPQP SOLUTION

(Bemporad, Morari, Dua, Pistikopoulos, 2002)

Theorem

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

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

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

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

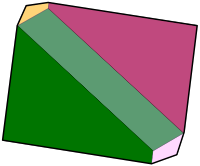
$$\begin{array}{rl} z^*(x) = \arg\min_z & \frac{1}{2} z'Qz + x'F'z \\ \text{s.t.} & Gz \leq W + Sx \end{array}$$

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

$$\begin{array}{rl} V^*(x) = \frac{1}{2} x'Yx + \min_z & \frac{1}{2} z'Qz + x'F'z \\ \text{s.t.} & Gz \leq W + Sx \end{array}$$

PROOF OF THEOREM

1) X^* is a convex polyhedron



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

- X^* is the **projection** of the convex polyhedron

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

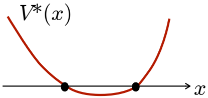
onto the x-space. Hence X^* is convex polyhedron

- Convexity can be also proved algebraically:
  - 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

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

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

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 $J(z,x) = \frac{1}{2}\left[\begin{smallmatrix} z \\ x \end{smallmatrix}\right]' \left[\begin{smallmatrix} Q & F \\ F' & Y \end{smallmatrix}\right] \left[\begin{smallmatrix} z \\ x \end{smallmatrix}\right]$

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

■

PROOF OF THEOREM

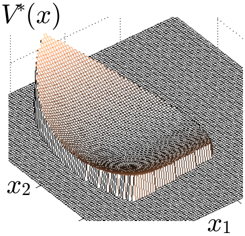
3) Continuity of z^* and V^* with respect to x

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

$\blacksquare$

MULTIPARAMETRIC CONVEX PROGRAMMING

$$\begin{array}{rl} \min_z & f(z,x) \\ \text{s.t.} & g_i(z,x) \leq 0,\ i=1,\ldots,p \\ & Ax+Bz+d=0 \end{array}$$



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

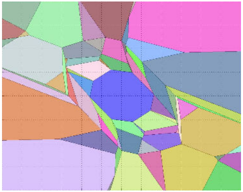
V^* and X^* may not be expressible analytically. Approximate solutions can be derived (Bemporad, Filippi, 2003)

EXPLICIT MPC SOLUTION

(Bemporad, Morari, Dua, Pistikopoulos, 2002)

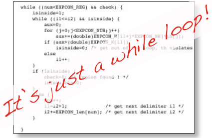
- **Corollary**: since the multiparametric solution

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

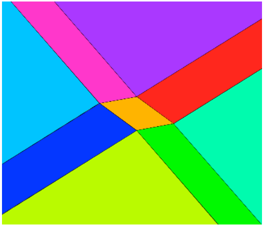


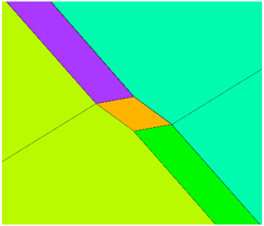
of a strictly convex QP is **continuous** and **piecewise affine**, the **linear MPC law is continuous & piecewise affine** too

$$z^* = \begin{bmatrix} \mathbf{u_0^*} \\ u_1^* \\ \vdots \\ u_{N-1}^* \end{bmatrix} \qquad u_0^*(x) = \begin{cases} F_1 x + g_1 & \text{if} \quad H_1 x \leq K_1 \\ \vdots & \quad\vdots \\ F_{n_r} x + g_{n_r} & \text{if} \quad H_{n_r} x \leq K_{n_r} \end{cases}$$



COMPLEXITY REDUCTION

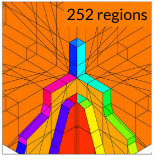


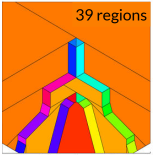


- We are interested only in the first components of the optimizer z^*

$$z^*(x) \triangleq \begin{bmatrix} \mathbf{u}_0^*(\mathbf{x})' & u_1^*(x)' & \dots & u_{N-1}^*(x)' \end{bmatrix}$$

- 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

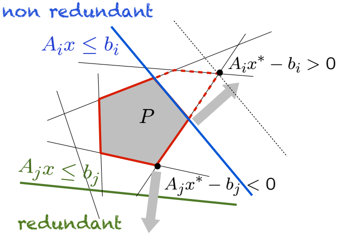
  (Geyer, Torrisi, Morari, 2008)

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, Bhartiya, Nataraj, 2011)

- Most computations are spent in removing redundant inequalities



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

$$\begin{array}{rl} 0 \ge \max_x & A_i x - b_i \\ \text{s.t.} & A_j x \le b_j,\ \forall j \ne i \end{array}$$

$\Rightarrow$ $A_i x \le b_i$ is redundant

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

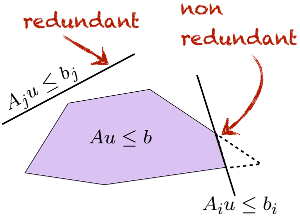
REMOVING REDUNDANT INEQUALITIES VIA NNLS

(Bemporad, 2015)

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

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

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



- Checking emptyness 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

n	q	NNLS	LP
2	20	0.0006	0.0046
4	40	0.0019	0.0103
10	100	0.0111	0.0554
16	160	0.0357	0.1959

random polyhedra of $\mathbb{R}^n$ with $q = 10n$ inequalities

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

CPU time = seconds (MacBook Pro 3.1 GHz i7)

MULTIPARAMETRIC QP BASED ON NNLS

(Bemporad, 2015)

- Other polyhedral operations can be solved also 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 CPU time wrt other existing methods:
  - **Hybrid Toolbox** (Bemporad, 2003)
  - **MPT Toolbox 2.6** (w/ default opts)
    (Kvasnica, Grieder, Baotic, 2004)
    (Herceg, 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:
$$\begin{cases} x(t+1) &= \left[\begin{smallmatrix}1 & 1\\ 0 & 1\end{smallmatrix}\right] x(t) + \left[\begin{smallmatrix}0\\ 1\end{smallmatrix}\right] u(t) \\ y(t) &= \left[\begin{smallmatrix}1 & 0\end{smallmatrix}\right] x(t)\end{cases}$$
$$-1 \leq u(t) \leq 1$$
- Objective:

$$\boxed{\min \sum_{k=0}^{\infty} y_k^2 + \frac{1}{100} u_k^2}$$

$$u_k = K x_k,\ \forall k \geq N_u,\ K = \text{LQR gain}$$

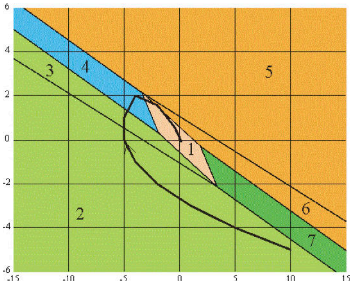
$$N_u = N = 2$$

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

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

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

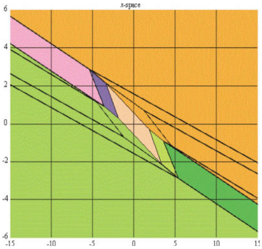
DOUBLE INTEGRATOR EXAMPLE - EXPLICIT SOLUTION



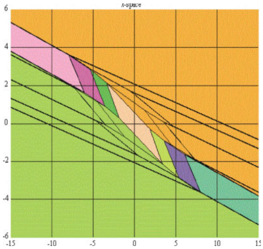
$N_u = 2$

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

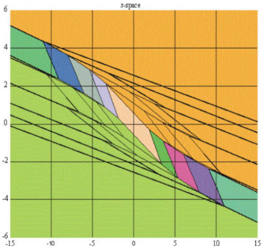
go to demo `linear/doubleintexp.m` (Hybrid Toolbox)



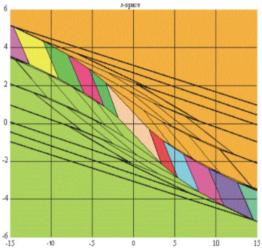
$N_u = 3$



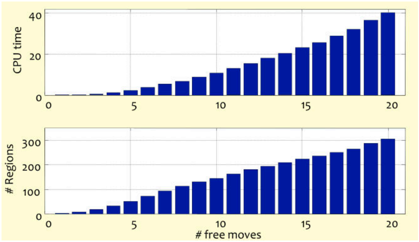
$N_u = 4$



$N_u = 5$



$N_u = 6$



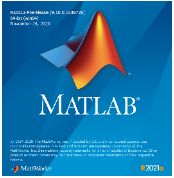
(is the number of regions finite for $N_u \to \infty$?)

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`

≈**10,000+ downloads**

≈**1.5 downloads/day**

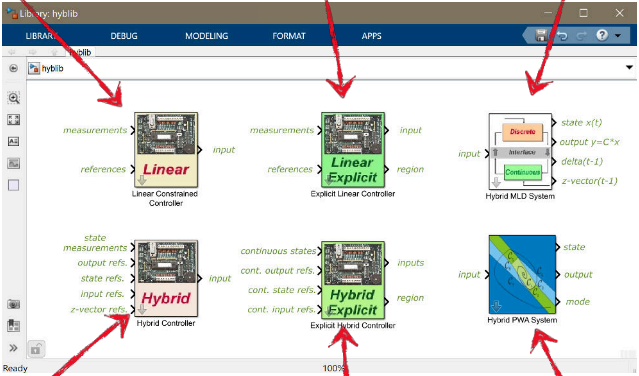
Initally supported by Ford Motor Company

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

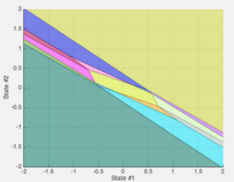
```
Ts=1; % sampling time
model=ss([1 1;0 1],[0;1],[0 1],0,Ts); % prediction model

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);
```


MPC TOOLBOX

(Bemporad, Morari, Ricker, $\geq$2014)

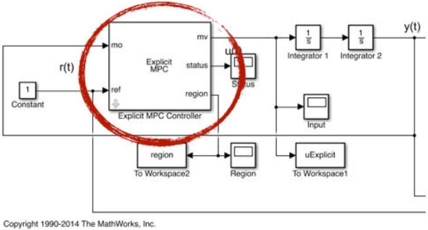
- **@explicitMPC** object The MathWorks

```
>> 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 online 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;
```



APPLICABILITY OF EXPLICIT MPC

- Consider the following general MPC formulation

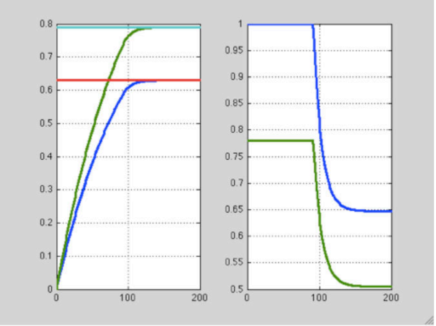
$$
\begin{array}{ll}
\min_{z} & \sum_{k=0}^{N-1} \frac{1}{2}(y_k - r(t+k))' S (y_k - r(t+k)) + \frac{1}{2}\Delta u_k' T \Delta u_k \\
& \quad +(u_k - u_r(t+k))' V (u_k - u_r(t+k))' + \rho_\epsilon \epsilon^2 \\
& \\
\text{subj. to} & x_{k+1} = A x_k + B u_k + B_v v(t+k),\ k = 0, \ldots, N-1 \\
& y_k = C x_k + D u_k + D_v v(t+k),\ k = 0, \ldots, N-1 \\
& u_{\min}(t+k) \leq u_k \leq u_{\max}(t+k),\ k = 0, \ldots, N-1 \\
& \Delta u_{\min}(t+k) \leq \Delta u_k \leq \Delta u_{\max}(t+k),\ k = 0, \ldots, N-1 \\
& y_{\min}(t+k) - \epsilon V_{\min} \leq y_k \leq y_{\max}(t+k) + \epsilon V_{\max},\ k = 1, \ldots, N \\
& x_0 = x(t)
\end{array}
$$

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

EXPLICIT MPC EXAMPLE - MIMO SYSTEM

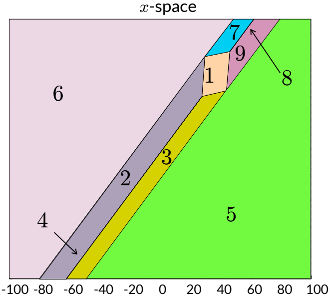
- Linear MIMO system $y(t) = \frac{10}{100s+1} \left[\begin{smallmatrix} 4 & -5 \\ -3 & 4 \end{smallmatrix}\right] u(t)$, sampled with $T_s = 1$ s
- Input constraints $\left[\begin{smallmatrix} -1 \\ -1 \end{smallmatrix}\right] \leq u(t) \leq \left[\begin{smallmatrix} 1 \\ 1 \end{smallmatrix}\right]$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

output $y(t)$, reference $r(t)$ input $u(t)$



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

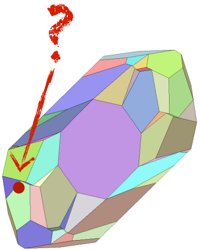
EXPLICIT MPC EXAMPLE - MIMO SYSTEM



Section of the polyhedral partition in the x-space obtained for $u(t-1) = \left[\begin{smallmatrix}0\\0\end{smallmatrix}\right]$ and $r(t) = \left[\begin{smallmatrix}0.63\\0.79\end{smallmatrix}\right]$

$$
\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} -0.1351 & 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 \leq \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} 0.5310 & -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 \leq \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 \leq \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 \leq \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 \leq \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 \leq \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 \leq \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 \leq \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 \leq \begin{bmatrix} 1.0000 \\ -1.0000 \end{bmatrix} \quad \text{(Region \#9)}
\end{cases}
$$

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)

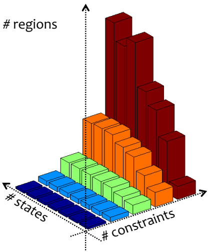
COMPLEXITY OF MULTIPARAMETRIC SOLUTIONS

- Number n_r of regions = # optimal combinations of active constraints:
  - mainly depends on the number q of constraints: $n_r \leq \sum_{h=0}^{q} \binom{q}{h} = 2^q$

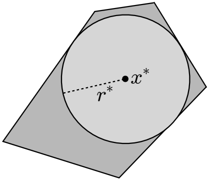
    (this is a worst-case estimate, most of the combinations are never optimal!)
  - also depends on the number s of free variables
  - weakly depends on the number n of parameters (states + references)

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 MPC - INTERPOLATION/EXTRAPOLATION 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

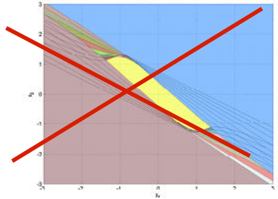
(Pannocchia, Rawlings, Wright, 2007) (Christophersen, Zeilinger, Jones, Morari, 2007)

(Alessio, Bemporad, 2008) (Jones, Morari, 2010) (Kvasnica, Fikar, 2010)

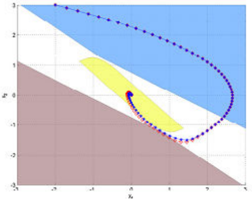
SUBOPTIMAL SOLUTIONS - INTERPOLATION METHODS

(Alessio, Bemporad, 2008)

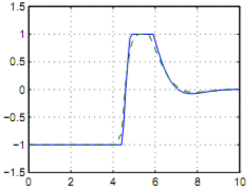
exact solution (99 regions)



L=3 most visited regions stored



input trajectory



- Weighted interpolation approach: set $\beta_i(x) = \max\{\max_j\{H_i^j x - K_i\}, 0\}$ and

$$u(x) = \begin{cases} F_i x + g_i & \text{if} \quad \beta_i(x) = 0 \quad \text{(x belongs to region $\#i$)} \\ \left(\sum_{i=1}^{L} \frac{1}{\beta_i(x)}\right)^{-1} \sum_{i=1}^{L} \frac{1}{\beta_i(x)}(F_i x + g_i) & \text{otherwise} \end{cases}$$

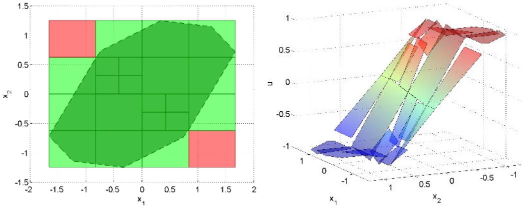
- Saturation of $u(x)$ might be enforced to ensure hard input constraints

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

- Approximate MPC laws by any **supervised learning method for regression**
  - Collect M samples (x_i, u_i) by solving MPC optimization problem for each x_i
  - Fit approximate mapping $\hat{u}(x)$ on the samples
  - Check performance / feasibility/ prove closed-loop stability (if possible)

- Possible function regression approaches:
  - **Lookup tables** (linear interpolation, inverse distance weighting, ...)
  - **Neural networks** (Parisini, Zoppoli, 1995) (Karg, Lucia, 2018)
  - **Hybrid system identification / PWA regression** (Breschi, Piga, Bemporad, 2016)
  - **Nonlinear systems identification** (Canale, Fagiano, Milanese, 2008)
  - **Decision trees**, **random forests**, **support vector machines**, **K-nearest neighbors**, ...

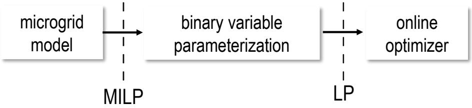
- Approach works for linear/nonlinear/stochastic/hybrid MPC

SEMI-EXPLICIT MPC

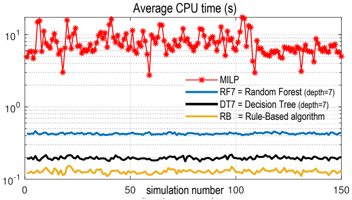
- **Semi-explicit MPC**: use **binary classification** to learn the optimal **active set** of a parametric QP for warm start (Klauco, Kalúz, Kvasnica, 2019)

- Learn optimal **binary variables** $\delta^*(x)$ of parametric MIQP/LP, then solve QP/LP online, or warm-start MIP solver (Masti, Bemporad, 2019)

- **Example**: hybrid MPC of for **microgrid optimization**



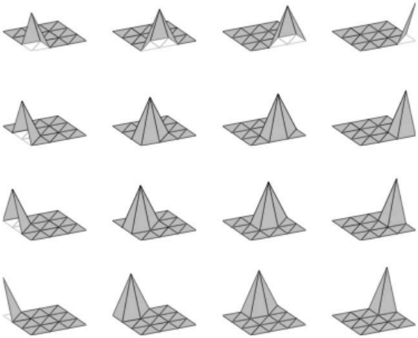
15,725 MILP solutions used to train $\delta^*(x)$



decision tree and **random forest** classifiers reduce CPU time by $\approx$ **96÷98%** with limited performance degradation (Masti, Pippia, Bemporad, De Schutter, 2020)

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 offline to approximate a given MPC law

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

SEVENTH FRAMEWORK PROGRAMME

- PWAS functions can be directly implemented on **FPGA** / **ASIC**

- Example: MPC of MIMO system $\begin{cases} x_{k+1} = \left[\begin{smallmatrix} 1.2 & 1 \\ 0 & 1.1 \end{smallmatrix}\right] x_k + \left[\begin{smallmatrix} 0 & 1 \\ 1 & 1 \end{smallmatrix}\right] u_k \\ y_k = \left[\begin{smallmatrix} 1 & 1 \\ 1 & 0 \end{smallmatrix}\right] x_k \end{cases}$

  with constraints on u and y ($N = 5$, $N_u = 5$)

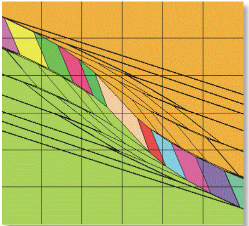
fit criterion	p	latency [ns] serial	latency [ns] parallel
L^2	7	170	31
	15	210	43
	31	238	45
	63	272	46
L^∞	7	170	31
	15	210	43
	31	238	45
	63	272	46

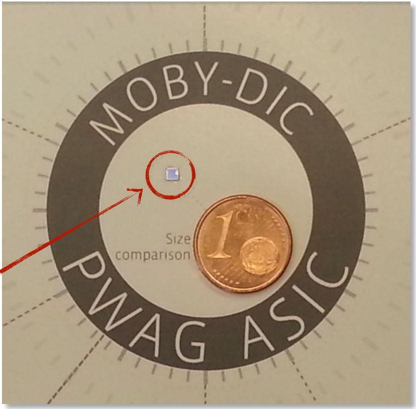
- latency = time to evaluate approximate MPC law on line (Xilinx Spartan 3 FPGA)
- p = # partitions per state dimension
- Exact explicit MPC (52 regions): 383 ns (avg), 486 ns (max)
- Function evaluation is **extremely fast** !

- 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 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) [redacted]
Power consumption: 38.08 mW@107.5MHz; [redacted]

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

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

EXAMPLE: AFTI-F16 AIRCRAFT

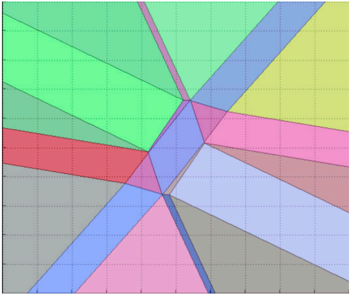
Linearized model:

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

- Open-loop unstable, poles are $-7.6636, -0.0075 \pm 0.0556j, 5.4530$
- Sampling time: $T_s = 0.05$ sec (+ ZOH)
- Constraints: $\pm 25^\circ$ on both angles

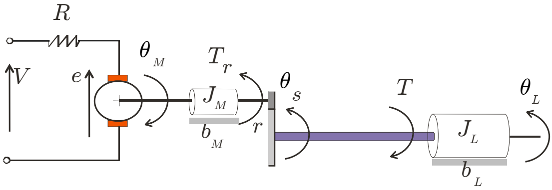
Explicit MPC: 8 parameters, 51 regions

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



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

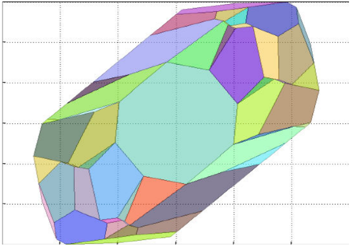
EXAMPLE: DC SERVOMOTOR



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

Explicit MPC: 7 parameters, 101 regions

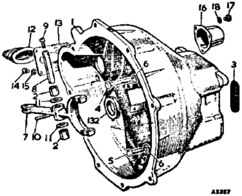
Section with $x_1 = x_4 = 0, u = 0, r = [0\ 0]'$



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 ($\omega_e > \omega_v$):

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

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

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

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

- Control objectives:
  - small friction losses
  - fast engagement
  - driver comfort

- Constraints:
  - clutch force
  - clutch force derivative
  - minimum engine speed

EXPLICIT MPC FOR DRY-CLUTCH ENGAGEMENT

- Linear model during slip + disturbance models

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

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

- Model sampled and converted to discrete time ($T_s = 10$ ms)

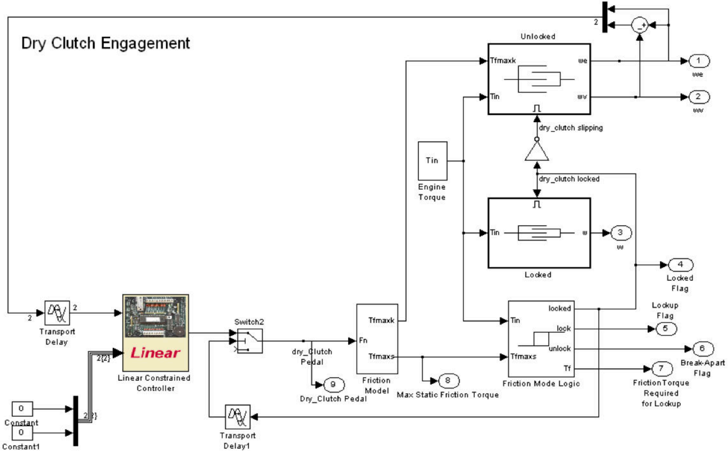
$$
\begin{bmatrix} x_1(t+1) \\ x_2(t+1) \\ T_{in}(t+1) \\ T_l(t+1) \\ u(t) \end{bmatrix} = \begin{bmatrix} 0.9985 & 0 & 0.0500 & 0 & -0.0049 \\ -0.0011 & 0.9996 & 0.0500 & 0.0129 & -0.0062 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \\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

- Control objective

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

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

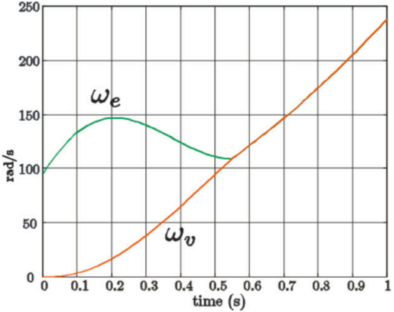
EXPLICIT MPC FOR DRY-CLUTCH ENGAGEMENT

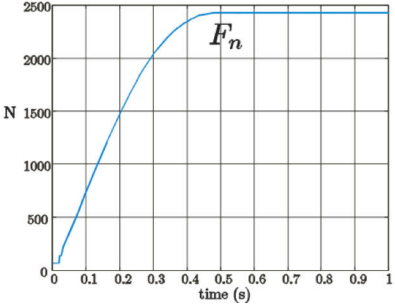


go to demo `dryclutch/dryclutch.m` (Hybrid Toolbox)

EXPLICIT MPC FOR DRY-CLUTCH ENGAGEMENT

- Simulation results





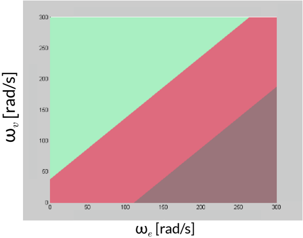
tuning	τ^*	$F_n(\tau^*)$ [N]	E_d [kJ]	$\dot{x}_2(\tau^*)$ $[rad/s^2]$
1	0.57	2333	6.676	230
2	0.76	2515	9.777	109
3	0.83	2572	10.838	62

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):

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



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

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

EXPLICIT MPC FOR IDLE SPEED CONTROL

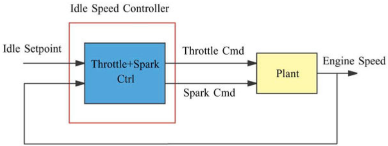
(Di Cairano, Yanakiev, Bemporad, Kolmanovsky, Hrovat, 2011)

- Main goal: **regulate engine speed at idle**

Ford pickup truck, V8 4.6L gasoline engine

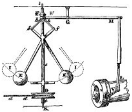
- Process model:
  - 1 output (engine speed)
  - 2 inputs (airflow, spark advance)
  - input delays

- Objectives and constraints
  - regulate engine speed at constant rpm
  - saturation limits on airflow and spark
  - lower bound on engine speed ≥ 450 rpm



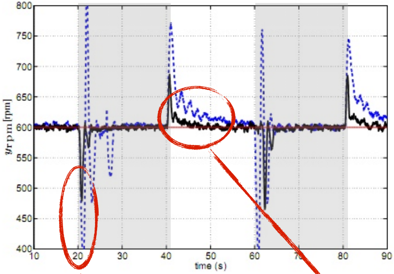
- Problem suitable for MPC design

  (Hrovat, 1996)



centrifugal governor
(Watt, 1788)

EXPLICIT MPC FOR IDLE SPEED CONTROL



Load torque (power steering)

(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 !

peak reduced by 50%

convergence 10s faster

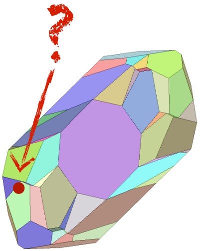
explicit MPC

baseline controller (linear)

set-point

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 online QP (=implicit MPC) ?

COMPLEXITY CERTIFICATION FOR ACTIVE-SET QP SOLVERS

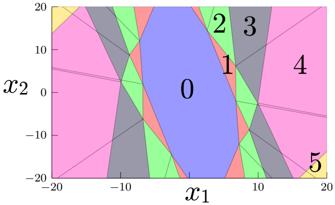
(Cimini, Bemporad, 2017)

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

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

- 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 online QP

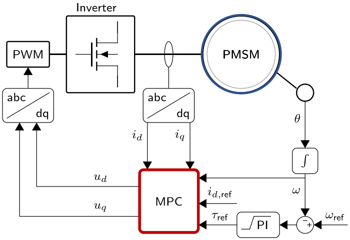
QP certification algorithm currently used in production

ODYS Advanced Controls & Optimization

MPC FOR TORQUE CONTROL OF PMSM

(Cimini, Bernardini, Levijoki, Bemporad, 2021)

- MPC of **Permanent Magnet Synchronous Motor**



- **Goal**: control motor torque

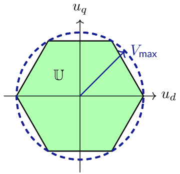
- Nonlinear isotropic PMSM model approximated by linear model @$\omega(t) = \omega_0$:

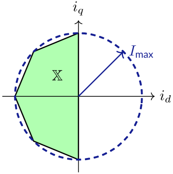
$$\dot{x} = \frac{d}{dt}\begin{bmatrix} i_d(t) \\ i_q(t) \end{bmatrix} = \begin{bmatrix} -\frac{R}{L} & \omega_0 \\ -\omega_0 & -\frac{R}{L} \end{bmatrix}\begin{bmatrix} i_d(t) \\ i_q(t) \end{bmatrix} + \begin{bmatrix} \frac{1}{L} & 0 \\ 0 & \frac{1}{L} \end{bmatrix}\begin{bmatrix} u_d(t) \\ u_q(t) \end{bmatrix} + \begin{bmatrix} 0 \\ -\frac{\lambda}{L} \end{bmatrix}\omega(t)$$

$$y(t) = \begin{bmatrix} i_d(t) \\ \tau(t) \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & K_t \end{bmatrix}\begin{bmatrix} i_d(t) \\ i_q(t) \end{bmatrix}$$

d = direct, q = quadrature

- Voltage/current constraints: (polyhedral approximation)



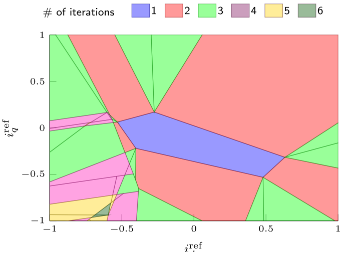


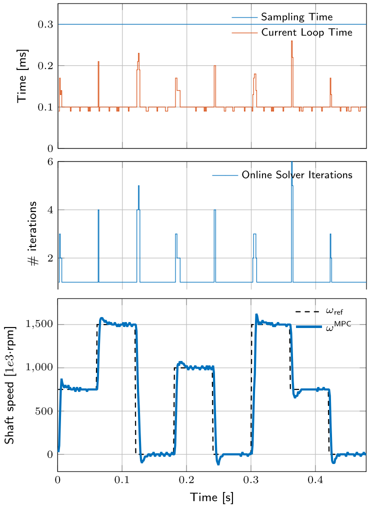
MPC FOR TORQUE CONTROL OF PMSM

- **Linear MPC** formulation, solved by **ODYS QP**

- Platform: TI F28335 Delfino 32-bit DSP
  150 MHz CPU, single precision

- **Complexity certification** algorithm guarantees
  2431 flops is the worst-case (=6 QP iters)





- Memory occupancy: 13 kB
  ($\leq$ single-access RAM block of 34 kB)

- Sampling time = 0.3 ms $\Rightarrow$
  real-time QP is 100% feasible

CERTIFICATION OF KR SOLVER

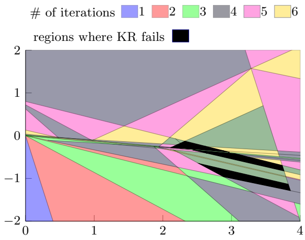
(Cimini, Bemporad, 2019)

- 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**

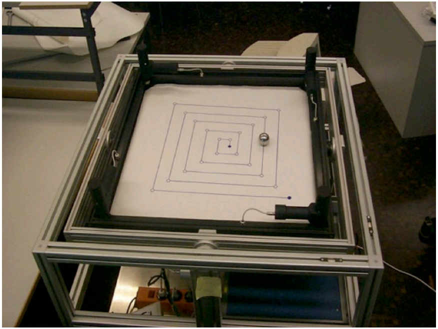
We can **exactly** map how many iterations KR takes to converge (or cycle)



	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



Target — **Ball & Plate**

inputs →

← measurements

BALL & PLATE: SPECIFICATIONS

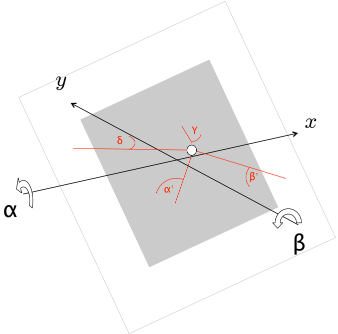
- Constraints:
  - plate angular position: ± 17 deg (soft)
  - ball position on plate: ± 30 cm (soft)
  - input voltage: ± 10 V (hard)

- platform: PC Pentium 166

- sample time: 30 ms

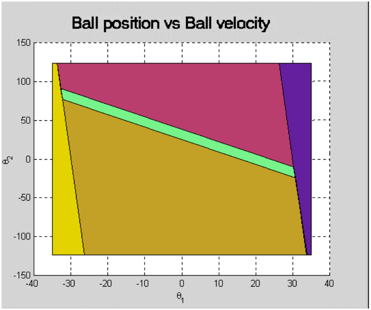
- prediction model: LTI model, 7×2 states

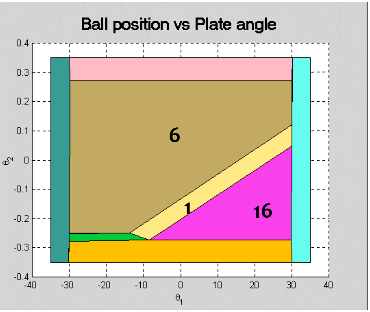
- MPC tuning:
  - prediction horizon $N = 50$
  - control horizon $N_u = 2$
  - weight on position error: $(W^y)^2 = 5$
  - weight on input rate: $W^{\Delta u} = 1$



BALL & PLATE: EXPLICIT MPC SOLUTION

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



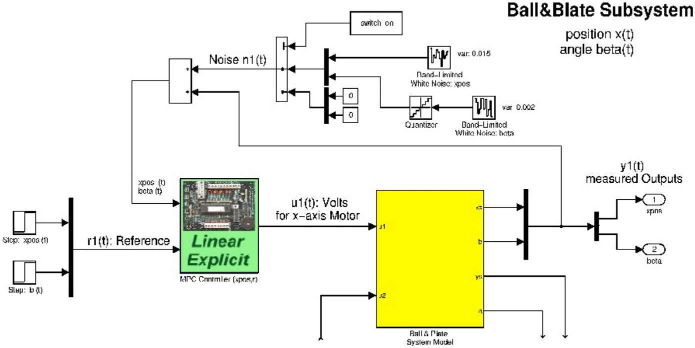


x-axis: sections at $\alpha_x = 0, \dot{\alpha}_x = 0, u_x = 0, r_x = 18, r_\alpha = 0$

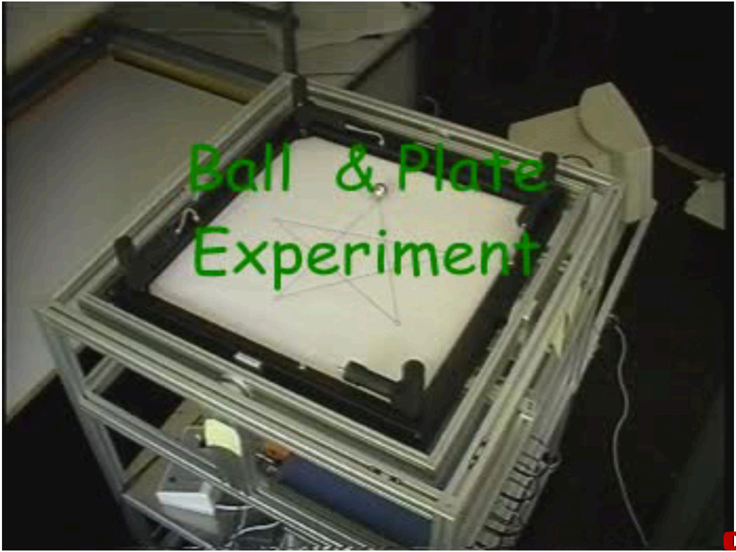
- Region #1: LQR controller
- Region #6: saturation at -10 V
- Region #16: saturation at +10 V

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

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

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

MULTIPARAMETRIC QP IN PORTFOLIO OPTIMIZATION

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

- **Markowitz portfolio optimization**:

$$\begin{array}{rl} \min_z & z'\Sigma z \\ \text{s.t.} & p'z \geq x \\ & \begin{bmatrix} 1 & 1 & \dots & 1 \end{bmatrix} z = 1 \\ & z \geq 0 \end{array}$$

z_i = fraction of total money invested in asset i

p_i = expected return of asset i

Σ_{ij} = covariance of assets i, j

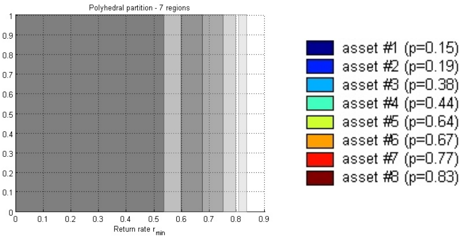
x = expected minimum return of portfolio

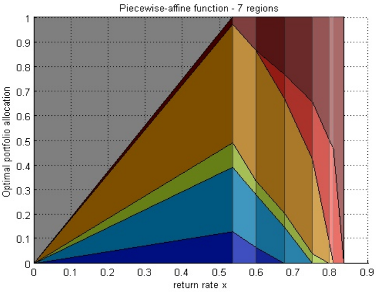
- Objective: minimize variance (=**risk**)
- Constraint: guarantee a minimum expected return x

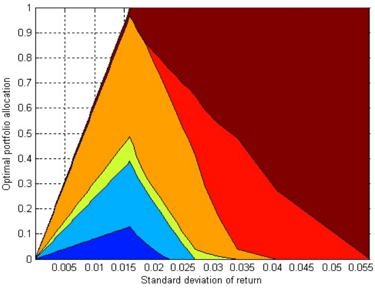
(Bemporad, NMPC plenary, 2008)

- Solution of mpQP problem

$$
\begin{array}{ll}
\min_z & z'\Sigma z \\
\text{s.t.} & p'z \geq \color{red}{x} \\
& \begin{bmatrix} 1 & 1 & \dots & 1 \end{bmatrix} z = 1 \\
& z \geq 0
\end{array}
$$







COMPARING DIFFERENT SOLUTION METHODS FOR MPC

Which solution method should we prefer for embedded MPC?

QP solver →	Active-Set	Interior-Point	ADMM	GPAD
CPU time (**small/medium** & **dense**)				
CPU time (**large** & **sparse**)				
Worst-case estimate of CPU time				
Numerical robustness (e.g., in single precision)				
Software complexity (linear algebra libraries)				

small-scale ≈ 20- variables, 50- constraints

large-size ≈ 500+ variables, 2500+ constraints