RECENT ADVANCES IN EMBEDDED MODEL PREDICTIVE CONTROL

Alberto Bemporad

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

www.odys.it
Use a dynamical model of the process to predict its future evolution and choose the "best" control action.
MODEL PREDICTIVE CONTROL (MPC)

- At time $t$: consider optimal control problem over a future horizon of $N$ steps

$$\min \sum_{k=0}^{N-1} \left\| W^y (y_k - r(t)) \right\|^2 + \left\| W^u (u_k - u^{\text{ref}}(t)) \right\|^2$$

s.t. $x_{k+1} = f(x_k, u_k, t)$

$y_k = g(x_k, u_k, t)$

constraints on $u_k, y_k$

$x_0 = x(t)$ \hspace{1cm} \text{feedback!}

optimization problem

- Solve problem w.r.t. \{ $u_0, \ldots, u_{N-1}$ \}

- Apply the first optimal move $u(t) = u_0^*$, throw the rest of the sequence away

- At time $t+1$: Get new measurements, repeat the optimization. And so on ...

\textbf{Used in process industries since the 80's}
Discrete Dynamic Optimization
Applied to On-Line Optimal Control

MARCHELL D. RAFAI and WILLIAM F. STEVENS
Northwestern University, Evanston, Illinois

A general method has been developed for controlling deterministic systems described by
linear or linearized dynamics. The discrete problem has been treated in detail. Step-by-step
optimal controls for a quadratic performance index have been derived. The method accom-
modates upper and lower limits on the components of the control vector.

A small binary distillation unit was considered as a typical application of the method.
The control vector was made up of feed rate, reflux ratio, and reboiler heat load. Control to
a desired state and about a load upset was effected.
Calculations were performed quite rapidly and only grew significantly with an increase in
the dimension of the control vector. Extension to much larger distillation units with the
same controls that seems practical.

The advent of high-speed computers has made possible the on-line digital control of many chemical engineering
processes. In on-line control a three-step procedure is ad-

1. Sense the current state.
2. Calculate a suitable control action.
3. Apply this control for a period of time known as the
sampling period.

The present study proposes a method for performing step 2. The technique developed is based on linearized
 dynamics. The strongly nonlinear binary distillation unit provides a suitable system for this study. While much has
been published recently (2, 3, 4) on modeling distillation,
title if anything has appeared on the optimal control of
such units.

In recent years, a good deal has been published by Kal-
mann, Lapinc, and others (4 to 7) on the control of linear
or linearized nonlinear systems by minimizing a quadratic
function of the states resulting from a sequence of control
actions. Their controls are always unconstrained, al-
though the introduction of a quadratic penalty function
limits this effect somewhat. The general constrained prob-
lem has been treated numerically (3) for a single control
variable. It was Wassinger (10, 11) who first chose to look
at the problem on a one-step-at-a-time basis rather than
considering a sequence of controls. However, he made no
attempt to solve completely the resulting quadratic pro-
gramming problem.

The approach taken in the present work is to set up the
problem on a one-step basis. This is quite compatible
with the on-line digital control scheme. The problem is
then shown to be a special case of the quadratic program-
ning problem and as such has a special solution. The
particulars concerning the theory underlying the solution
scheme and its implementation on a digital computer have
been presented (9). In addition, a derivation of the theo-
rems upon which the computational algorithm is based is
presented in the Appendix.

The authors wish to be very careful to point out that
optimal, as used herein, refers only to a single step of
control. Even for truly linear systems, the step-by-step
optimal control need not be overall optimal. A recent
work by Albanks and Pal (14) presents both the virtues and
defects of such an on-step method. In the present work, the
one-step approach is taken because it is amenable to
practical solution of the problem and is well suited to non-
linear situations where updating linearization is useful.

THE PROBLEM

The system under consideration is described by a set of
matrix differential equations:

\[ \dot{X}(t) = AX(t) + BM(t) + B(t) \]  \hspace{1cm} (1)
Automotive applications of MPC

Bemporad, Bernardini, Borrelli, Cimini, Di Cairano, Esen, Giorgetti, Graf-Plessen, Hrovat, Kolmanovsky, Levijoki, Ripaccioli, Trimboli, Tseng, Yanakiev, ... (2001-2016)

**Powertrain**
- direct-inj. engine control
- A/F ratio control
- magnetic actuators
- robotized gearbox
- power MGT in HEVs
- cabin heat control in HEVs
- electrical motors

**Vehicle dynamics**
- traction control
- active steering
- semiactive suspensions
- autonomous driving

Ford Motor Company
DENSO Automotive
Jaguar
FIAT
General Motors

ODYS
Advanced Controls & Optimization
MPC TOOLBOXES

- **MPC Toolbox** (The Mathworks, Inc.)
  (Bemporad, Ricker, Morari, 1998-present)
  - Part of Mathworks’ official toolbox distribution
  - Great for **education and research**

- **Hybrid Toolbox**
  (Bemporad, 2003-present)
  - Free download: [http://cse.lab.imtlucca.it/~bemporad/hybrid/toolbox/](http://cse.lab.imtlucca.it/~bemporad/hybrid/toolbox/)
  - Great for **research and education**

- **ODYS Toolbox**
  (Bemporad, Bernardini, 2013-present)
  - Provides flexible and customized MPC control **design** and **seamless integration** in production systems
  - Real-time code written in plain C
  - **Designed for production**
AEROSPACE APPLICATIONS OF MPC

- MPC capabilities explored in new space applications
- New MATLAB MPC Toolboxes developed (MPCTOOL and MPCSofT) (Bemporad, 2010) (Bemporad, 2012)

- Powered descent
- Cooperating UAVs
- Planetary rover

(Bemporad, Rocchi, 2011) (Pascucci, Bennani, Bemporad, 2016) (Krenn et. al., 2012)
Embedded Linear MPC and Quadratic Programming

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

\[
\begin{align*}
\min_z & \quad \frac{1}{2} z' H z + x'(t) F' z + \frac{1}{2} x'(t) Y x(t) \\
\text{s.t.} & \quad G z \leq W + S x(t)
\end{align*}
\]

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

ON MINIMIZING A CONVEX FUNCTION SUBJECT TO LINEAR INEQUALITIES

By E. M. L. Beale

Admiralty Research Laboratory, Teddington, Middlesex

SUMMARY

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

(Beale, 1955)

A rich set of good QP algorithms is available today, still more research is required to have an impact in real applications!
Requirements for production:

1. **Speed (throughput):** solve optimization problem within sampling interval
2. **Robustness** (e.g., with respect to numerical errors)
3. Be able to run on **limited hardware** (e.g., 150 MHz) with **little memory**
4. **Worst-case execution time** must be (tightly) estimated
5. **Code simple** enough to be validated/verified/certified (in general, it must be understandable by production engineers)
• Multivariable MPC controller

• Sampling frequency = 40 Hz (= 1 QP solved every 25 ms)

• Vehicle operating ~1 hr/day for ~360 days/year on average

• Controller may be run on 10 million vehicles

\[ \sim 520,000,000,000,000 \text{ QP/yr} \]

and none of them can fail!
Apply **fast gradient method** to dual QP:

\[
\begin{align*}
\min_z & \quad \frac{1}{2} z' H z + x' F' z \\
\text{s.t.} & \quad G z \leq W + S x \\
\beta_k & = \begin{cases} 0 & k = 0 \\ \frac{k-1}{k+2} & k > 0 \end{cases}
\end{align*}
\]

\[
\begin{align*}
w_k & = y_k + \beta_k(y_k - y_{k-1}) \\
z_k & = -K w_k - J x \\
s_k & = \frac{1}{L} G z_k - \frac{1}{L} (S x + W) \\
y_{k+1} & = \max\{y_k + s_k, 0\}
\end{align*}
\]

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

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

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

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

**dual function**

\[
\begin{align*}
K & = H^{-1} G' \\
J & = H^{-1} F'
\end{align*}
\]

**feasibility tol**

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

**optimality tol**

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

**dual function**
**Fast Gradient Projection for (Dual) QP**

- Main on-line operations involve only simple linear algebra

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

- Tight bounds on maximum number of iterations

- Can be used to warm-start other methods

- Currently extended to mixed-integer problems

---

(Patrinos, Bemporad, IEEE TAC, 2014)

(Naik, Bemporad, work in progress)
**Experiments with Embedded QP**

TMS320F28335 controlCARD
(Real-time Control Applications)
- 32-bit Floating Point (IEEE-754);
- 150MHz clock;
- 68KB Ram / 512KB Flash.

<table>
<thead>
<tr>
<th>var $\times$ constr.</th>
<th>GPAD</th>
<th>AS</th>
<th>ADMM</th>
<th>FBN</th>
</tr>
</thead>
<tbody>
<tr>
<td>4 $\times$ 16</td>
<td>332 $\mu$s (18)</td>
<td>120 $\mu$s (3)</td>
<td>1.42 ms (62)</td>
<td>208 $\mu$s (2)</td>
</tr>
<tr>
<td>8 $\times$ 24</td>
<td>1.1 ms (22)</td>
<td>446 $\mu$s (5)</td>
<td>4 ms (77)</td>
<td>396 $\mu$s (2)</td>
</tr>
<tr>
<td>12 $\times$ 32</td>
<td>2.59 ms (27)</td>
<td>1.19 ms (7)</td>
<td>8.25 ms (82)</td>
<td>652 $\mu$s (2)</td>
</tr>
</tbody>
</table>

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

* GPAD = Dual Accelerated Gradient Projection  
* FBN = Forward-Backwards Netwon (proximal method)  
* ADMM = Alternating Directions Method of Multipliers  

(Patrinos, Bemporad, 2014)  
(Patrinos, Guiggiani, Bemporad, 2014)  
(Boyd et al., 2010)
• **Fixed-point arithmetics** is very attractive for embedded control:

![Diagram of fixed-point representation](image)

- **Pros:**
  - Computations are fast and cheap
  - Hardware support in all platforms

- **Cons:**
  - Accumulation of quantization errors
  - Limited range (numerical overflow)
HARDWARE TESTS (FLOATING VS FIXED POINT)

- Gradient projection works in fixed-point arithmetics

\[
\max_i g_i(z_k) \leq \frac{2L\delta^2}{k+1} + L\nu\varepsilon_z^2 + 4D\varepsilon_\xi
\]

max constraint violation

expontially decreasing with number p of fractional bits

Gradient projection works in fixed-point arithmetics

**Table 1**: Fixed-point hardware implementation

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>10/20</td>
<td>22.9</td>
<td>226</td>
<td>15</td>
</tr>
<tr>
<td>20/40</td>
<td><strong>52.9</strong></td>
<td>867</td>
<td>17</td>
</tr>
<tr>
<td>40/80</td>
<td>544.9</td>
<td>3382</td>
<td>27</td>
</tr>
<tr>
<td>60/120</td>
<td>1519.8</td>
<td>7561</td>
<td>43</td>
</tr>
</tbody>
</table>

**Table 2**: Floating-point hardware implementation

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>10/20</td>
<td>88.6</td>
<td>974</td>
<td>16</td>
</tr>
<tr>
<td>20/40</td>
<td><strong>220.1</strong></td>
<td>3608</td>
<td>21</td>
</tr>
<tr>
<td>40/80</td>
<td>2240</td>
<td>13099</td>
<td>40</td>
</tr>
<tr>
<td>60/120</td>
<td>5816</td>
<td>30450</td>
<td>73</td>
</tr>
</tbody>
</table>

32-bit Atmel SAM3X8E

ARM Cortex-M3 processing unit

84 MHz, 512 KB of flash memory and 100 KB of RAM

fixed-point about 4x faster than floating-point
The **Least Squares (LS)** problem is probably the most studied problem in numerical linear algebra.

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

In MATLAB: \[ \gg v = A \backslash b \quad \% \text{(1 character !)} \]

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

\[ \min_v \| Av - b \|_2^2 \quad \text{s.t.} \quad v \geq 0 \]
\[ \min_v \|Av - b\|_2^2 \]
\[ \text{s.t. } v \geq 0 \]

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

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

- NNLS algorithm is very simple (**750 chars in Embedded MATLAB**), the key operation is to solve a **standard LS problem** at each iteration (via QR, LDL, or Cholesky factorization)
SOLVING QP’S VIA NONNEGATIVE LEAST SQUARES

• Use NNLS to solve strictly convex QP

\[
\min_{z} \frac{1}{2} z' Q z + c' z \\
\text{s.t.} \quad G z \leq g
\]

QP

\[
u \triangleq L z + L^{-T} c
\]

Q = L' L
\[
M = G L^{-1}
\]
\[
d = b + G Q^{-1} c
\]

least distance problem

\[
\min_{u} \frac{1}{2} \|u\|^2 \\
\text{s.t.} \quad M u \leq d
\]

QP problem infeasible

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

retrieve primal solution

\[
\min_{y} \frac{1}{2} \left\| \begin{bmatrix} M' \\ d' \\ 0 \\ 1 \end{bmatrix} y + \begin{bmatrix} 0 \\ 1 \end{bmatrix} \right\|_2^2 \\
\text{s.t.} \quad y \geq 0
\]

Nonnegative Least Squares

Least Distance Problem

(Bemporad, IEEE TAC, 2016)
SOLVING QP VIA NNLS: NUMERICAL RESULTS

- A rather **fast** and relatively **simple-to-code** QP solver

- Extended to solving mixed-integer QP’s

(Bemporad, IEEE TAC, 2016)

(Bemporad, NMPC 2015)
**Embedded MPC without Solving QP’s on Line**

- Dynamical model (based on data)
- Embedded model-based optimizer
- Optimization algorithm
  \[
  \min \frac{1}{2} z'Qz + c'z \\
  \text{s.t.} \quad Az \leq b
  \]

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

  **YES!**
The multiparametric solution of a strictly convex QP is **continuous** and **piecewise affine**

\[
\begin{align*}
    z^*(x) &= \text{arg min}_z \quad \frac{1}{2} z' Hz + x' F' z \\
    \text{s.t.} \quad Gz &\leq W + Sx
\end{align*}
\]

**Corollary:** The **linear MPC** control law is continuous & piecewise affine!

\[
z^* = \begin{bmatrix} u_0^* \\ u_1^* \\ \vdots \\ u_{N-1}^* \end{bmatrix}
\]

\[
u(x) = \begin{cases} 
    F_1 x + g_1 & \text{if } H_1 x \leq K_1 \\
    \vdots & \vdots \\
    F_M x + g_M & \text{if } H_M x \leq K_M 
\end{cases}
\]
A variety of mpQP solvers is available

Most computations are spent in operations on polyhedra (=critical regions)

- checking emptiness of polyhedra
- removal of redundant inequalities
- checking full-dimensionality of polyhedra

All such operations are usually done via linear programming (LP)
A polyhedron $P = \{ u \in \mathbb{R}^n : Au \leq b \}$ is nonempty iff

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

s.t. $v \geq 0, u$ free

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

### Key result:

<table>
<thead>
<tr>
<th>$m$</th>
<th>NNLS</th>
<th>LP</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0.0006</td>
<td>0.0046</td>
</tr>
<tr>
<td>4</td>
<td>0.0019</td>
<td>0.0103</td>
</tr>
<tr>
<td>6</td>
<td>0.0038</td>
<td>0.0193</td>
</tr>
<tr>
<td>8</td>
<td>0.0071</td>
<td>0.0340</td>
</tr>
<tr>
<td>10</td>
<td>0.0111</td>
<td>0.0554</td>
</tr>
<tr>
<td>12</td>
<td>0.0178</td>
<td>0.0955</td>
</tr>
<tr>
<td>14</td>
<td>0.0263</td>
<td>0.1426</td>
</tr>
<tr>
<td>16</td>
<td>0.0357</td>
<td>0.1959</td>
</tr>
</tbody>
</table>

- Random polyhedra of $\mathbb{R}^m$ with $10m$ inequalities
- NNLS = compiled Embedded MATLAB
- LP = compiled C code (GLPK)
- CPU time = seconds (this Mac)

### Many other polyhedral operations can be also tackled by NNLS
**NNLS FOR SOLVING MPQP PROBLEMS**

(Bemporad, IEEE TAC, 2015)

- New mpQP algorithm based on **NNLS + dual QP formulation** to compute active sets and deal with degeneracy

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

- Included in **MPC Toolbox 5.0 (≥R2014b)**

![The MathWorks](Bemporad, Morari, Ricker, 1998-2015)

<table>
<thead>
<tr>
<th>q</th>
<th>m</th>
<th>Hybrid Tbx</th>
<th>MPT</th>
<th>NNLS</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>2</td>
<td>0.0174</td>
<td>0.0256</td>
<td>0.0026</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>0.0203</td>
<td>0.0356</td>
<td>0.0038</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>0.0432</td>
<td>0.0559</td>
<td>0.0061</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>0.0650</td>
<td>0.0850</td>
<td>0.0097</td>
</tr>
<tr>
<td>4</td>
<td>6</td>
<td>0.0827</td>
<td>0.1105</td>
<td>0.0126</td>
</tr>
<tr>
<td>8</td>
<td>2</td>
<td>0.0347</td>
<td>0.0396</td>
<td>0.0050</td>
</tr>
<tr>
<td>8</td>
<td>3</td>
<td>0.0583</td>
<td>0.0680</td>
<td>0.0092</td>
</tr>
<tr>
<td>8</td>
<td>4</td>
<td>0.0916</td>
<td>0.0999</td>
<td>0.0140</td>
</tr>
<tr>
<td>8</td>
<td>5</td>
<td>0.1869</td>
<td>0.2147</td>
<td>0.0322</td>
</tr>
<tr>
<td>8</td>
<td>6</td>
<td>0.3177</td>
<td>0.3611</td>
<td>0.0586</td>
</tr>
<tr>
<td>12</td>
<td>2</td>
<td>0.0398</td>
<td>0.0387</td>
<td>0.0054</td>
</tr>
<tr>
<td>12</td>
<td>3</td>
<td>0.1121</td>
<td>0.1158</td>
<td>0.0191</td>
</tr>
<tr>
<td>12</td>
<td>4</td>
<td>0.2067</td>
<td>0.2001</td>
<td>0.0352</td>
</tr>
<tr>
<td>12</td>
<td>5</td>
<td>0.6180</td>
<td>0.6428</td>
<td>0.1151</td>
</tr>
<tr>
<td>12</td>
<td>6</td>
<td>1.2453</td>
<td>1.3601</td>
<td>0.2426</td>
</tr>
<tr>
<td>20</td>
<td>2</td>
<td>0.1029</td>
<td>0.0763</td>
<td>0.0152</td>
</tr>
<tr>
<td>20</td>
<td>3</td>
<td>0.3698</td>
<td>0.2905</td>
<td>0.0588</td>
</tr>
<tr>
<td>20</td>
<td>4</td>
<td>0.9069</td>
<td>0.7100</td>
<td>0.1617</td>
</tr>
<tr>
<td>20</td>
<td>5</td>
<td>2.2978</td>
<td>1.9761</td>
<td>0.4395</td>
</tr>
<tr>
<td>20</td>
<td>6</td>
<td>6.1220</td>
<td>6.2518</td>
<td>1.2853</td>
</tr>
</tbody>
</table>
HARDWARE (ASIC) IMPLEMENTATION OF EXPLICIT MPC

Technology: 90 nm, 9 metal-layer from Taiwan Semiconductor Manufacturing Company
Chip size: 1860x1860 μm²
Memory sizes: 24 KB (tree memory); 30 KB (parameter memory)
Power supply: 2.5V (ring); 1.2V (core)
Maximum frequency: 107.5 MHz (with two inputs)
Power consumption: 38.08 mW@107.5 MHz

\[ \frac{1}{107.5 \text{ MHz}} = 9.3 \text{ ns} \]

http://www.mobydic-project.eu/
Explicit MPC for idle speed - Experiments

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

Observer tuning as much important as tuning of MPC weights!

Load torque (power steering)

peak reduced by 50%

convergence 10s faster
The number of regions depends (exponentially) on the number of possible combinations of active constraints (weak dependence on the number of states and references).

Explicit MPC gets less attractive when the number of regions grows: too much memory required, too much time to locate state $x(t)$.

Fast on-line QP solvers may be preferable.

When is implicit preferable to explicit MPC?
• Consider a dual active-set QP solver
  (Goldfarb, Idnani, 1983)

\[
z^*(x) = \arg \min_z \quad \frac{1}{2}z'Hz + x'F'z
\]
\[
\text{s.t.} \quad Gz \leq W + Sx
\]

• 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!
• Examples (from MPC Toolbox):

<table>
<thead>
<tr>
<th></th>
<th>inv. pend.</th>
<th>DC motor</th>
<th>nonlin. demo</th>
<th>AFTI 16</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong># vars</strong></td>
<td>5</td>
<td>3</td>
<td>6</td>
<td>5</td>
</tr>
<tr>
<td><strong># constraints</strong></td>
<td>10</td>
<td>10</td>
<td>18</td>
<td>12</td>
</tr>
<tr>
<td><strong># params</strong></td>
<td>9</td>
<td>6</td>
<td>10</td>
<td>10</td>
</tr>
</tbody>
</table>

**Explicit MPC**

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong># regions</strong></td>
<td>87</td>
<td>67</td>
<td>215</td>
<td>417</td>
</tr>
<tr>
<td><strong>max flops</strong></td>
<td>3382</td>
<td>1689</td>
<td>9184</td>
<td>16434</td>
</tr>
<tr>
<td><strong>max memory</strong></td>
<td>55</td>
<td>30</td>
<td>297</td>
<td>430</td>
</tr>
</tbody>
</table>

**Implicit MPC**

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>max iters</strong></td>
<td>11</td>
<td>9</td>
<td>13</td>
<td>16</td>
</tr>
<tr>
<td><strong>max flops</strong></td>
<td>3809</td>
<td>2082</td>
<td>7747</td>
<td>7807</td>
</tr>
<tr>
<td><strong>sqrt</strong></td>
<td>27</td>
<td>9</td>
<td>37</td>
<td>33</td>
</tr>
<tr>
<td><strong>max memory</strong></td>
<td>15</td>
<td>13</td>
<td>20</td>
<td>16</td>
</tr>
</tbody>
</table>

explicit MPC is faster in the worst-case

online QP is faster in the worst-case

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

(Cimini, Bemporad, 2016)
MPC FOR POWER MANAGEMENT IN HEVS

Control problem: decide optimal generation of **mechanical power** (from engine) and **electrical power** (from battery) to satisfy **driver’s power request**.

What will the future power request from the driver be?

\[ P_{\text{req}}(w(t)) = \text{driver’s power request} \]

\[ P_{\text{req}}(k) = P_{\text{el}}(k) + P_{\text{mec}}(k) - P_{\text{br}}(k) \]
• Driver’s power request $w(k)$ modeled as a **stochastic** process (Markov chain)

• Real-world driving cycles (acquired on vehicle)

| Trace | Accel. Type | $||\Delta P||$ | Fuel cons. | $SoC$ gain/loss | Equiv. fuel cons. | impr. wrt FTMPc |
|-------|-------------|----------------|------------|----------------|-----------------|-----------------|
| #1    | smooth      | FTMPC 37.84kW  | 243g       | -0.05%         | 244g            | -               |
|       |             | $\rightarrow$ SMPCL 14.32kW | 244g | 0.90% | 225g | 8.04% |
|       |             | PMPC 14.08kW  | 223g       | -0.08%         | 224g            | 8.19%           |
| #2    | steep       | FTMPC 80.61kW  | 327g       | 0.11%          | 323g            | -               |
|       |             | $\rightarrow$ SMPCL 35.74kW | 320g | 1.16% | 287g | 11.34% |
|       |             | PMPC 30.67kW  | 287g       | 0.17%          | 282g            | 12.73%          |

**TABLE III** Percentage improvement of SMPCL strategy due to online learning of the Markov chain. PMPC that exploits full knowledge of the future power request. Also in this case, the advantages of PMPC and SMPCL are more evident in the driving profile with steeper accelerations (Trace 2), which is expected according to the power smoothing objective of the control strategy.

Finally, in Table III we provide an indication of the component of the SMPCL improvement that is exclusively due to online driver model learning. The reported percentage is the ratio of the difference between the equivalent fuel consumption of SMPCL and FTMPC and the difference between equivalent fuel consumption of SMPCL with ($\lambda = 0.01$) and without ($\lambda = 0$) online learning. In some cases, the benefits exclusively due to learning are small, because the initial Markov chain is already representative of the driving pattern, whereas in the case of more varied driving cycles the benefits of the learning algorithm are significant, indicating that overall learning is useful in driving conditions with complex patterns.
Can we implement MPC **without** even a **model** of the process? **YES!**
• Collect a set of observations \(\{u(k), y(k), p(k)\}, \ k = 1, \ldots, N\)

• Specify a desired closed-loop linear model \(\mathcal{M}\) from \(r\) to \(y\)

• Compute \(r_v(k) = \mathcal{M}^\# y(k)\) from pseudo-inverse model \(\mathcal{M}^\#\) of \(\mathcal{M}\)

• Identify linear (LPV) model from \(e_v = r_v - y\) (virtual tracking error) to \(u\)
• Design a linear MPC controller (reference governor) to generate command $r$

\[ r_o \xrightarrow{\text{MPC}} r \xrightarrow{K_p} y \]

Linear prediction model (totally known !)

(Piga, Formentin, Bemporad, 2016)

MPC can handle constraints on inputs and outputs, and improve closed-loop performance

(Bemporad, 1997)
• DC motor equations

\[
\begin{bmatrix}
\dot{\theta}(\tau) \\
\dot{\omega}(\tau) \\
\dot{I}(\tau)
\end{bmatrix} = \begin{bmatrix}
0 & \frac{1}{J} & 0 \\
0 & -\frac{K}{L} & \frac{K}{L} \\
0 & 0 & -\frac{R}{L}
\end{bmatrix} \begin{bmatrix}
\theta(\tau) \\
\omega(\tau) \\
I(\tau)
\end{bmatrix} + \begin{bmatrix}
0 & 1 & 0 \\
0 & \frac{mgl}{J} & 0 \\
0 & 0 & 0
\end{bmatrix} \begin{bmatrix}
\sin(\theta(\tau)) \\
\omega(\tau) \\
I(\tau)
\end{bmatrix} + \begin{bmatrix}
0 \\
0 \\
\frac{1}{L}
\end{bmatrix} V(\tau)
\]

\[y(\tau) = \begin{bmatrix}
1 & 0 & 0
\end{bmatrix} \begin{bmatrix}
\theta(\tau) \\
\omega(\tau) \\
I(\tau)
\end{bmatrix}
\]

• Desired closed-loop behavior \(M\) (=first-order low-pass filter):

\[x_M(k + 1) = 0.99 x_M(k) + 0.01 r(k)\]

\[\theta_M(k) = x_M(k)\]

• Chosen control structure for \(K_p\):

\[K'_p : u(k) = \sum_{i=1}^{4} a_i(\theta(k)) u(k - i) + \sum_{j=0}^{3} b_j(\theta(k)) e_l(k - j)\]

• MPC design w/ soft constraints on inputs, outputs and input increments
DATA-DRIVEN MPC SYNTHESIS OF CONTROLLERS - AN EXAMPLE

• Experimental results

No model of open-loop process identified to design the MPC controller!
CONCLUSIONS

• MPC can easily handle **multivariable control** problems with **constraints** in an **optimized** way, it’s **easy to design** and **reconfigure**, it handles **uncertainty**

• Long history of success in the **process industries** since the 80’s, now spreading in the automotive industry (and many others)

• MATLAB design/calibration tools and production-ready C-code are available

**Is MPC a mature technology for production in fast embedded applications?**

**YES.**

[http://cse.lab.imtlucca.it/~bemporad/publications](http://cse.lab.imtlucca.it/~bemporad/publications)