MODEL PREDICTIVE CONTROL FOR CYBER-PHYSICAL SYSTEMS

Alberto Bemporad

`imt.lu/ab`

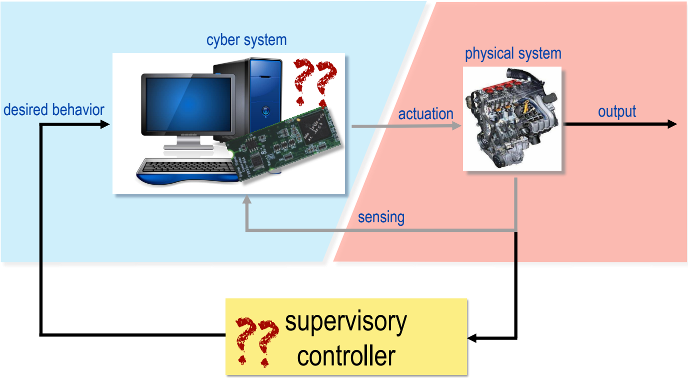
IMT SCHOOL FOR ADVANCED STUDIES LUCCA

OCPS

(extended slide set: `http://cse.lab.imtlucca.it/~bemporad/mpc_course.html`)

8th oCPS Fall School on Multidisciplinary CPS Design - October 30, 2019

CONTROL ISSUES IN CYBER-PHYSICAL SYSTEMS

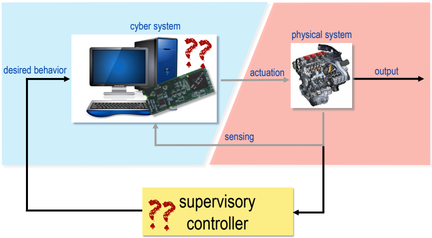


- What to embed inside the **cyber** system to make the **physical** system behave **autonomously** in a **robust**, **safe**, and **optimal** manner?

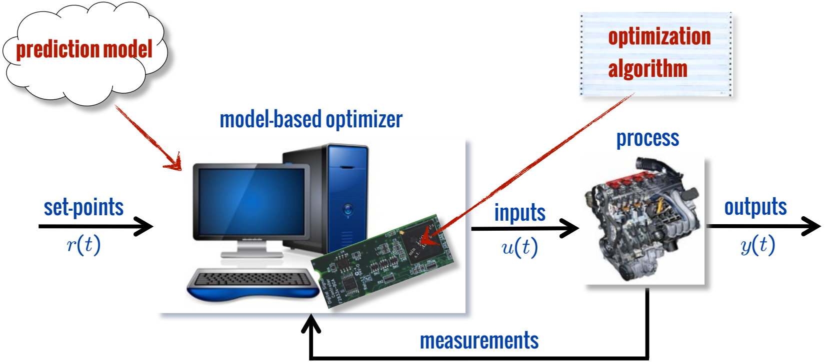
- How to synthesize a supervisory controller for a CPS ?

CONTENTS OF MY LECTURE

- Model predictive control **for** CPS's
- Embedded quadratic optimization algorithms (**inside** the CPS)
- Hybrid MPC = supervisory control **of** CPS's
- Data-driven controller synthesis for CPS's



MODEL PREDICTIVE CONTROL (MPC)

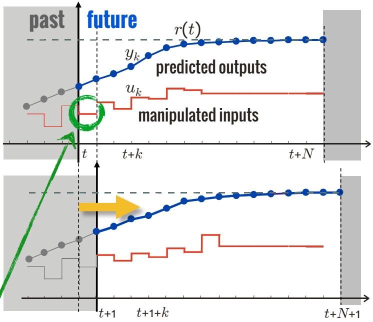


Use a dynamical **model** of the process to **predict** its future evolution and choose the "best" **control** action

MODEL PREDICTIVE CONTROL (MPC)

- **Goal**: find the best control sequence over a future horizon of N steps

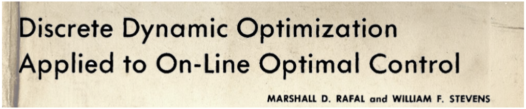
$$
\begin{array}{lll}
\min & \displaystyle\sum_{k=0}^{N-1} \|W^y(y_k - r(t))\|_2^2 + \|W^u(u_k - u_r(t))\|_2^2 & \\
\text{s.t.} & x_{k+1} = f(x_k, u_k) & \text{prediction model} \\
 & y_k = g(x_k) & \\
 & u_{\min} \le u_k \le u_{\max} & \text{constraints} \\
 & y_{\min} \le y_k \le y_{\max} & \\
 & x_0 = x(t) & \text{state feedback}
\end{array}
$$



➡ **numerical optimization problem**

- **At each time t:**
  - get new measurements to update the estimate of the current state $x(t)$
  - solve the optimization problem with respect to $\{u_0, \ldots, u_{N-1}\}$
  - apply only the first optimal move $u(t) = u_0^*$, discard the remaining samples

- The MPC concept for process control dates back to the 60's



Discrete Dynamic Optimization
Applied to On-Line Optimal Control

MARSHALL D. RAFAL and WILLIAM F. STEVENS

(Rafal, Stevens, AiChE Journal, 1968)

- MPC used in the process industries since the 80's

(Qin, Badgewell, 2003) (Bauer, Craig, 2008)

©SimulateLive.com

Today APC (advanced process control) = MPC

MPC IN INDUSTRY

(Samad, IEEE CS Magazine, 2017)

- Impact of advanced control technologies in industry

TABLE 1 A list of the survey results in order of industry impact as perceived by the committee members.

Rank and Technology	High-Impact Ratings	Low- or No-Impact Ratings
PID control	100%	0%
Model predictive control	78%	9%
System identification	61%	9%
Process data analytics	61%	17%
Soft sensing	52%	22%
Fault detection and identification	50%	18%
Decentralized and/or coordinated control	48%	30%
Intelligent control	35%	30%
Discrete-event systems	23%	32%
Nonlinear control	22%	35%
Adaptive control	17%	43%
Robust control	13%	43%
Hybrid dynamical systems	13%	43%

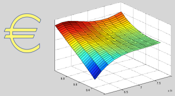
MPC IN INDUSTRY

(Samad, IFAC Newsletter, April 2019)

	Current Impact		Future Impact	
Control Technology	**% High**	**Low/No**	**High**	**Low/No**
PID control	91%	0%	78%	6%
System Identification	65%	5%	72%	5%
Estimation & filtering	64%	11%	63%	3%
Model-predictive control	62%	11%	85%	2%
Process data analytics	51%	15%	70%	8%
Fault detection & identification	48%	17%	8%	8%
Decentralized and/or coordinated control	29%	33%	54%	11%
Robust control	26%	35%	42%	23%
Intelligent control	24%	38%	5 9%	11%
Nonlinear control	21%	44%	42%	15%
Discrete-event systems	24%	45%	39%	27%
Adaptive control	18%	38%	44%	17%
Repetitive control	12%	74%	17%	51%
Other advanced control technology	11%	64%	25%	39%
Hybrid dynamical systems	11%	68%	33%	33%
Game theory	5%	76%	17%	52%

TYPICAL USE OF MPC

static optimizer

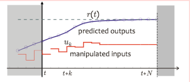


Steady-state optimization

- (economic) set-point optimization
- static I/O model

set-points

dynamic optimizer



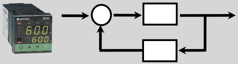
MPC

- coordinate multiple inputs
- performance optimization
- constraint handling

measurements

actuator set-points

regulators



low-level controllers

- fast-sampling
- single-loop

MPC OF AUTOMOTIVE SYSTEMS

(Bemporad, Bernardini, Borrelli, Cimini, Di Cairano, Esen, Giorgetti, Graf-Plessen, Hrovat, Kolmanovsky Levijoki, Livshiz, Long, Pattipati, Ripaccioli, Trimboli, Tseng, Verdejo, Yanakiev, ..., 2001-present)

Powertrain

engine control, magnetic actuators, robotized gearbox, power MGT in HEVs, cabin heat control, electrical motors

Vehicle dynamics

traction control, active steering, semiactive suspensions, autonomous driving

Ford Motor Company

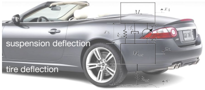
Jaguar

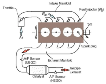
DENSO Automotive

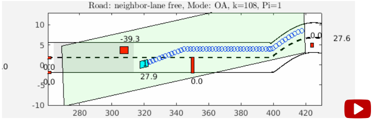
FCA

General Motors

ODYS



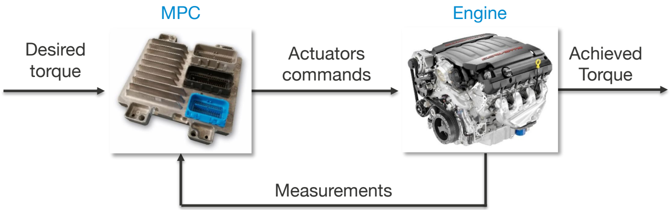




Most automotive OEMs are looking into MPC solutions today

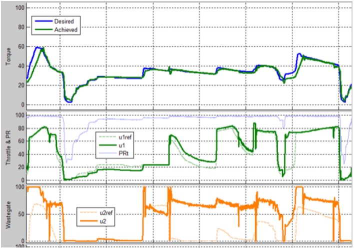
MPC OF GASOLINE TURBOCHARGED ENGINES

- Control **throttle, wastegate, intake & exhaust cams** to make **engine torque** track set-points, with max efficiency and satisfying **constraints**



numerical optimization problem solved in real-time on ECU

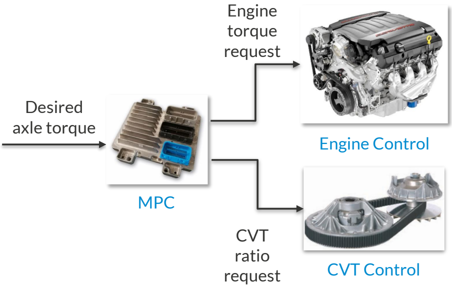
(Bemporad, Bernardini, Long, Verdejo, 2018)

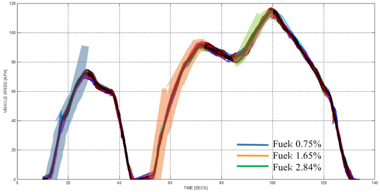


engine operating at low pressure (66 kPa)

SUPERVISORY MPC OF POWERTRAIN WITH CVT

- Coordinate **engine torque request** and continuously variable transmission (CVT) **ratio** to improve fuel economy and drivability
- Real-time MPC is able to take into account **coupled dynamics** and **constraints**, optimizing performance also during transients





US06 Double Hill driving cycle

(Bemporad, Bernardini, Livshiz, Pattipati, 2018)

MPC IN AERONAUTIC INDUSTRY

PRESS RELEASE

Pratt & Whitney's F135 Advanced Multi-Variable Control Team Receives UTC's Prestigious George Mead Award for Outstanding Engineering Accomplishment

EAST HARTFORD, CONN., THURSDAY, MAY 27, 2010

Pratt & Whitney engineers Louis Celiberti, Timothy Crowley, James Fuller and Cary Powell won the George Mead Award – United Technologies Corp.'s highest award for outstanding engineering achievement – for their pioneering work in developing the world's first advanced multi-variable control (AMVC) design for the only engine that powers the F-35 Lightning II flight test program. Pratt & Whitney is a United Technologies Corp. (NYSE:UTX) company.

The AMVC, which uses a proprietary model predictive control methodology, is the most technically advanced propulsion system control ever produced by the aerospace industry, demonstrating the highest pilot rating for flight performance and providing independent control of vertical thrust and pitch from five sources. This innovative and industry-leading advanced design is protected with five broad patents for Pratt & Whitney and UTC, and is the new standard for propulsion system control for Pratt & Whitney military and commercial engines.

Pratt & Whitney
A United Technologies Company

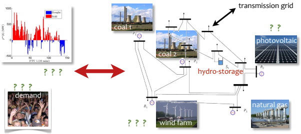
http://www.pw.utc.com/Press/Story/20100527-0100/2010

OTHER EXAMPLES OF MPC APPLICATIONS

- MPC for **smart electricity grids**

  Example: Dispatch power in smart distribution grids, trade energy on energy markets

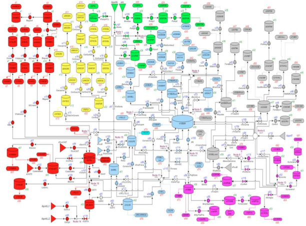
  (Patrinos, Trimboli, Bemporad, 2011)



- MPC of **drinking water networks**

  Example: save $\approx$5% energy costs in Barcelona's drinking water network w.r.t. current practice

  (Sampathirao, Sopasakis, Bemporad, Patrinos, 2017)



- MPC for **financial engineering**

  Example: dynamic portfolio optimization for option hedging

  (Bemporad, Bellucci, Gabbriellini, 2014)

€€€

All the above applications require **stochastic MPC** formulations

MODEL PREDICTIVE CONTROL - THE BASICS

LINEAR MPC

- Linear prediction model: $\begin{cases} x_{k+1} = Ax_k + Bu_k \\ y_k = Cx_k \end{cases}$

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

- Constraints to enforce:

$$\begin{cases} u_{\min} \leq u(t) \leq u_{\max} \\ y_{\min} \leq y(t) \leq y_{\max} \end{cases}$$

- Constrained optimal control problem (quadratic performance index):

$$\begin{aligned} \min_{z} \quad & x_N' P x_N + \sum_{k=0}^{N-1} x_k' Q x_k + u_k' R u_k \\ \text{s.t.} \quad & u_{\min} \leq u_k \leq u_{\max},\ k = 0, \ldots, N-1 \\ & y_{\min} \leq y_k \leq y_{\max},\ k = 1, \ldots, N \end{aligned}$$

$$\begin{aligned} R &= R' \succ 0 \\ Q &= Q' \succeq 0 \\ P &= P' \succeq 0 \end{aligned} \qquad z = \begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_{N-1} \end{bmatrix}$$

LINEAR MPC

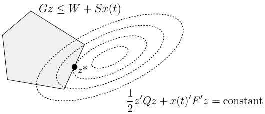
- Linear prediction model: $x_k = A^k x_0 + \sum_{i=0}^{k-1} A^i B u_{k-1-i}$
- Optimization problem (condensed form):

$$V(x_0) = \frac{1}{2}x_0' Y x_0 + \min_z \ \frac{1}{2}z' H z + x_0' F' z \quad \text{(quadratic objective)}$$

$$\text{s.t.} \quad Gz \leq W + S x_0 \quad \text{(linear constraints)}$$

convex Quadratic Program (QP)

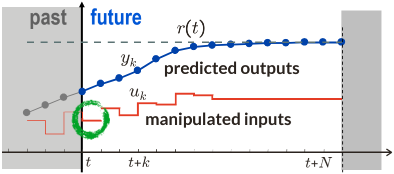
- $z = \begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_{N-1} \end{bmatrix} \in \mathbb{R}^{Nm}$ is the optimization vector



- $H = H' \succ 0$, and H, F, Y, G, W, S depend on weights Q, R, P upper and lower bounds $u_{\min}, u_{\max}, y_{\min}, y_{\max}$ and model matrices A, B, C.

LINEAR MPC ALGORITHM

@ each sampling step t:



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

- Get the solution $z^* = \begin{bmatrix} u_0^* \\ u_1^* \\ \vdots \\ u_{N-1}^* \end{bmatrix}$ of the QP

$$\left\{ \begin{array}{ll} \min\limits_z & \frac{1}{2} z' H z + \overbrace{x'(t) F' z}^{\text{feedback}} \\ \text{s.t.} & G z \leq W + S \underbrace{x(t)}_{\text{feedback}} \end{array} \right.$$

- Apply only $u(t) = u_0^*$, discarding the remaining optimal inputs $u_1^*, \ldots, u_{N-1}^*$

BASIC CONVERGENCE PROPERTIES

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

- **Theorem**: Let the MPC law be based on

$$
\begin{aligned}
V^*(x(t)) = \min \quad & \sum_{k=0}^{N-1} x_k' Q x_k + u_k' R u_k \\
\text{s.t.} \quad & x_{k+1} = A x_k + B u_k \\
& u_{\min} \leq u_k \leq u_{\max} \\
& y_{\min} \leq C x_k \leq y_{\max} \\
& x_N = 0 \qquad \leftarrow \text{“terminal constraint”}
\end{aligned}
$$

  with $R, Q \succ 0, u_{\min} < 0 < u_{\max}, y_{\min} < 0 < y_{\max}$.
  If the **optimization problem is feasible at time** $t = 0$ then

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

  and the constraints are satisfied at all time $t \geq 0$, for all $R, Q \succ 0$.

- Many more convergence and stability results exist (Mayne, 2014)

LINEAR MPC - TRACKING

- Optimal control problem (quadratic performance index):

$$
\begin{array}{ll}
\min\limits_{z} & \displaystyle\sum_{k=0}^{N-1} \|W^y(y_{k+1}-r(t))\|_2^2 + \|W^{\Delta u}\Delta u_k\|_2^2 \\
 & [\Delta u_k \triangleq u_k - u_{k-1}],\ u_{-1} = u(t-1) \\
 & \\
\text{s.t.} & u_{\min} \le u_k \le u_{\max},\ k = 0,\dots,N-1 \\
 & y_{\min} \le y_k \le y_{\max},\ k = 1,\dots,N \\
 & \Delta u_{\min} \le \Delta u_k \le \Delta u_{\max},\ k = 0,\dots,N-1
\end{array}
$$

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

weight W = diagonal matrix (more generally, Cholesky factor of $Q = W'W$)

$$
\begin{array}{ll}
\min\limits_{z} & J(z,x(t)) = \frac{1}{2}z'Hz + [x'(t)\, r'(t)\, u'(t-1)]F'z \\
\text{s.t.} & Gz \le W + S \begin{bmatrix} x(t) \\ r(t) \\ u(t-1) \end{bmatrix}
\end{array}
$$

convex Quadratic Program

- Add the extra penalty $\|W^u(u_k - u_{\text{ref}}(t))\|_2^2$ to track **input references**
- Constraints may depend on $r(t)$, such as $e_{\min} \le y_k - r(t) \le e_{\max}$

INTEGRAL ACTION AND Δu-FORMULATION

- In control systems, **integral action** occurs if the controller has a transfer-function from the output to the input of the form

$$u(t) = \frac{B(z)}{(z-1)A(z)}y(t), \qquad B(1) \neq 0$$

- One may think that the Δu-formulation of MPC provides integral action ...

  ... is it true ?

- **Example**: we want to regulate the output $y(t)$ to zero of the scalar system

$$\begin{array}{rcl} x(t+1) & = & \alpha x(t) + \beta u(t) \\ y(t) & = & x(t) \end{array}$$

INTEGRAL ACTION AND Δu-FORMULATION

- Design an unconstrained MPC controller with horizon $N = 1$

$$\begin{aligned}\Delta u(t) = \quad & \arg\min_{\Delta u_0} \Delta u_0^2 + \rho y_1^2 \\ \text{s.t.} \quad & \Delta u_0 = u_0 - u(t-1) \\ & y_1 = x_1 = \alpha x(t) + \beta(\Delta u_0 + u(t-1))\end{aligned}$$

- By substitution, we get

$$\begin{aligned}\Delta u(t) &= \arg\min_{\Delta u_0} \Delta u_0^2 + \rho(\alpha x(t) + \beta u(t-1) + \beta \Delta u_0)^2 \\ &= \arg\min_{\Delta u_0} (1+\rho\beta^2)\Delta u_0^2 + 2\beta\rho(\alpha x(t) + \beta u(t-1))\Delta u_0 \\ &= -\tfrac{\beta\rho\alpha}{1+\rho\beta^2} x(t) - \tfrac{\rho\beta^2}{1+\rho\beta^2} u(t-1)\end{aligned}$$

- Since $x(t) = y(t)$ and $u(t) = u(t-1) + \Delta u(t)$ we get the linear controller

$$u(t) = \frac{\frac{\rho\beta\alpha}{1+\rho\beta^2} z}{z - \frac{1}{1+\rho\beta^2}} y(t)$$

no pole in z=1

- Reason: MPC gives a feedback gain on both $x(t)$ and $u(t-1)$, not just on $x(t)$

OUTPUT INTEGRATORS AND OFFSET-FREE TRACKING

- Add constant unknown disturbances on measured outputs:

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

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

- Why we get offset-free tracking in steady-state (intuitively):
  - the observer makes $C\hat{x}(t) + \hat{d}(t) \to y(t)$ **(estimation error)**
  - the MPC controller makes $C\hat{x}(t) + \hat{d}(t) \to r(t)$ **(predicted tracking error)**
  - the combination of the two makes $y(t) \to r(t)$ **(actual tracking error)**

- In steady state, the term $\hat{d}(t)$ compensates for model mismatch

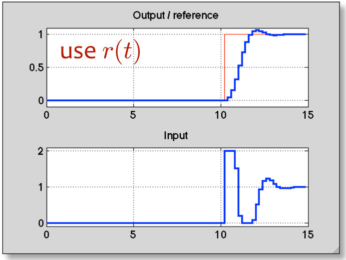
- See more on survey paper (Pannocchia, Gabiccini, Artoni, 2015)

ANTICIPATIVE ACTION (A.K.A. "PREVIEW")

$$\min_{\Delta U} \sum_{k=0}^{N-1} \|W^y(y_{k+1} - r_{k+1})\|_2^2 + \|W^{\Delta u}\Delta u(k)\|_2^2$$

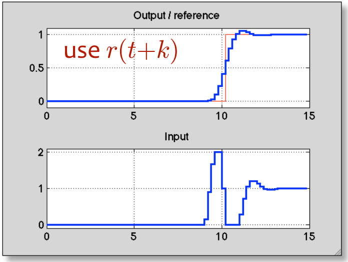
- Reference **not known** in advance (**causal**):

$$r_k \equiv r(t), \forall k = 0, \ldots, N-1$$



- Future refs (partially) **known** in advance (**anticipative action**):

$$r_k = r(t+k), \forall k = 0, \ldots, N-1$$



go to demo `mpcpreview.m` (MPC Toolbox)

- Same idea also applies for **preview of measured disturbances**

LTV-MPC

- **Linear Time-Varying (LTV)** model predictive control

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

- The model can change at each time t, even over the prediction horizon k
- The resulting optimization problem is still a QP

$$\begin{array}{ll} \min\limits_{z} & \frac{1}{2}z'H(t)z + \begin{bmatrix} x(t) \\ r(t) \\ u(t-1) \end{bmatrix}' F(t)'z \\ \text{s.t.} & G(t)z \leq W(t) + S(t)\begin{bmatrix} x(t) \\ r(t) \\ u(t-1) \end{bmatrix} \end{array}$$

- In LTV-MPC the QP matrices must be constructed online

LINEARIZING A NONLINEAR MODEL

- LTV models can be obtained by linearizing **nonlinear models**

$$\begin{cases} \frac{dx_c(t)}{dt} & = & f(x_c(t), u_c(t), p_c(t)) \\ y_c(t) & = & g(x_c(t), p_c(t)) \end{cases}$$

- At time t, consider **nominal trajectories**

$$U = \{\bar{u}_c(t), \bar{u}_c(t+T_s), \ldots, \bar{u}_c(t+(N-1)T_s)\}$$

(example: U = shifted previous optimal sequence or input ref. trajectory)

$$P = \{\bar{p}_c(t), \bar{p}_c(t+T_s), \ldots, \bar{p}_c(t+(N-1)T_s)\}$$

(no preview: $\bar{p}_c(t+k) \equiv \bar{p}_c(t)$)

- **Integrate** the model and get nominal state/output trajectories

$$\begin{array}{rcl} X & = & \{\bar{x}_c(t), \bar{x}_c(t+T_s), \ldots, \bar{x}_c(t+(N-1)T_s)\} \\ Y & = & \{\bar{y}_c(t), \bar{y}_c(t+T_s), \ldots, \bar{y}_c(t+(N-1)T_s)\} \end{array}$$

- Examples: $\bar{x}_c(t) =$ current state / equilibrium state / reference state

LINEARIZATION AND TIME-DISCRETIZATION

- Getting the discrete-time LTV model $A_k(t), B_k(t), C_k(t)$ requires to **linearize** and **discretize** in time the nonlinear continuous-time dynamical model

$$\frac{dx_c(t)}{dt} = f(x_c, u_c, p_c) \approx \underbrace{f(\bar{x}_c, \bar{u}_c, \bar{p}_c)}_{\frac{d\bar{x}_c}{dt}} + \underbrace{\left.\frac{\partial f}{\partial x_c}\right|_{\bar{x}_c, \bar{u}_c, \bar{p}_c}}_{\text{Jacobian matrix } A_c} (x_c - \bar{x}_c) + \underbrace{\left.\frac{\partial f}{\partial u_c}\right|_{\bar{x}_c, \bar{u}_c, \bar{p}_c}}_{\text{Jacobian matrix } B_c} (u_c - \bar{u}_c)$$

- Let $x = x_c - \bar{x}_c, u = u_c - \bar{u}_c$. We get the continuous-time linear system

$$\frac{dx}{dt} = A_c x + B_c u$$

- Similarly, from the output equation we get $y = y_c - \bar{y}_c \approx \underbrace{\left.\frac{\partial g}{\partial x_c}\right|_{\bar{x}_c, \bar{u}_c, \bar{p}_c}}_{\text{Jacobian matrix } C} x$

- Convert (A_c, B_c, C) to discrete-time model (A, B, C) (Euler method, exp. matrix, ...)

- **LTV-MPC**: @each time t simulate the NL model, get linearized models, build & solve QP

FROM LTV-MPC TO NONLINEAR MPC

(Gros, Zanon, Quirynen, Bemporad, Diehl, 2016)

- **NL-MPC**: We can solve a **sequence of LTV-MPC** problems at each time t

For $h = 0$ to $h_{\max} - 1$ do:

1. **Simulate** from $x(t)$ with inputs U_h and get state trajectory X_h
2. **Linearize** around (X_h, U_h) and **discretize** in time
3. Get U^*_{h+1} = **QP solution** of corresponding LTV-MPC problem
4. **Line search**: find optimal step size $\alpha_h \in (0, 1]$;
5. Set $U_{h+1} = (1 - \alpha_h)U_h + \alpha_h U^*_{h+1}$;

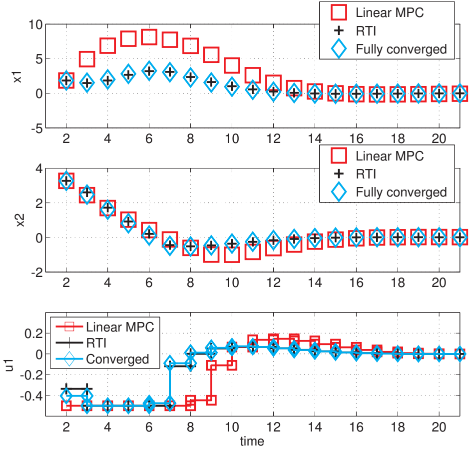
Return solution $U_{h_{\max}}$

- The above method is **Sequential Quadratic Programming** (SQP) applied to solve the full nonlinear MPC problem

- Special case: just solve one iteration with $\alpha = 1$ (a.k.a. **Real-Time Iteration**) (Diehl, Bock, Schloder, Findeisen, Nagy, Allgower, 2002) = LTV-MPC

(Gros, Zanon, Quirynen, Bemporad, Diehl, 2016)

- Example



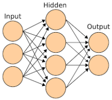
PREDICTION MODELS FOR MPC

- **Physics-based** nonlinear models

$$
\begin{aligned}
\dot{p}_1 &= k_1(W_c + W_{egr} - k_e p_1) + \frac{\dot{T}_1}{T_1} p_1 \\
\dot{p}_2 &= k_2(k_e p_1 - W_{egr} - W_t + W_f) + \frac{\dot{T}_2}{T_2} p_2 \\
\dot{P}_c &= \frac{1}{\tau}(P_c - \eta_m P_t)
\end{aligned}
$$

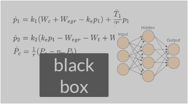
- Use **black-box system identification** algorithms to fit linear or nonlinear models to data

- Use **machine-learning** techniques to get nonlinear models (such neural networks) from data, with Jacobians



- A mix of the above (**gray-box** models)

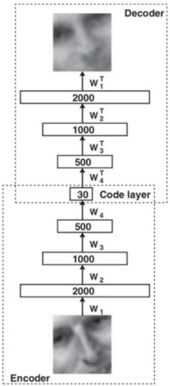
- **Note**: Computation complexity depends on chosen model, need to trade off **descriptiveness vs simplicity** of the model



LEARNING NONLINEAR MODELS FOR MPC

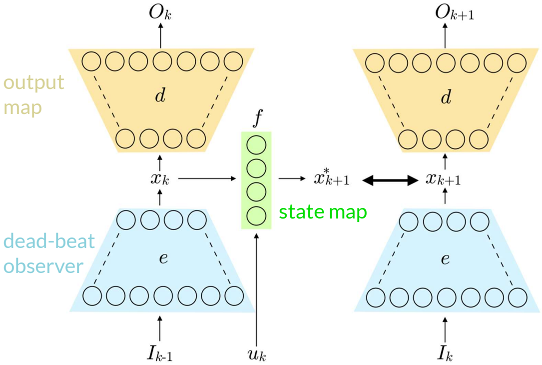
(Masti, Bemporad, 2018)

- **Idea**: use **autoencoders** and artificial neural networks to learn a **nonlinear state-space model** of **desired order** from input/output data



ANN with hourglass structure

(Hinton, Salakhutdinov, 2006)



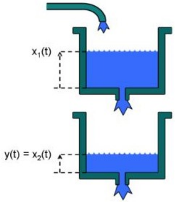
$$O_k = [y_k' \ \dots \ y_{k-m}']'$$

$$I_k = [y_k' \ \dots \ y_{k-n_a+1}' \ u_k' \ \dots \ u_{k-n_b+1}']'$$

LEARNING NONLINEAR MODELS FOR MPC - AN EXAMPLE

(Masti, Bemporad, 2018)

- System generating the data = nonlinear 2-tank benchmark

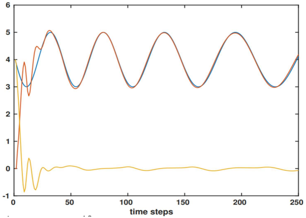


www.mathworks.com

$$\begin{cases} x_1(k+1) = x_1(k) - k_1\sqrt{x_1(k)} + k_2(u(k) + w(k)) \\ x_2(k+1) = x_2(k) + k_3\sqrt{x_1(k)} - k_4\sqrt{x_2(k)} \\ y(k) = x_2(k) + v(k) \end{cases}$$

Model is totally unknown to learning algorithm

- Artificial neural network (ANN): 3 hidden layers
  60 exponential linear unit (ELU) neurons
- For given number of model parameters,
  autoencoder approach is superior to NNARX
- **Jacobians** directly obtained from ANN structure
  for Kalman filtering & MPC problem construction



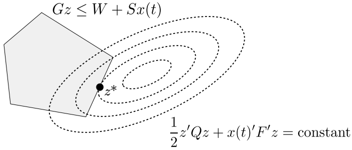
LTV-MPC results

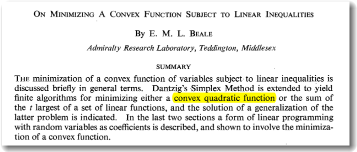
EMBEDDED QUADRATIC OPTIMIZATION FOR MPC

EMBEDDED MPC AND QUADRATIC PROGRAMMING

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

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





ON MINIMIZING A CONVEX FUNCTION SUBJECT TO LINEAR INEQUALITIES

By E. M. L. BEALE

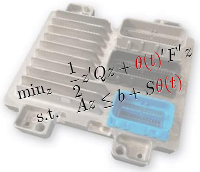
Admiralty Research Laboratory, Teddington, Middlesex

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)

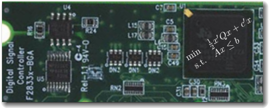
> A rich set of good QP algorithms is available today



- 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

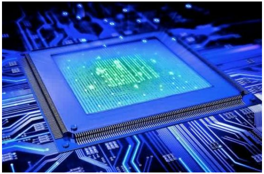
5. **code simple enough** to be validated/verified/certified
   (library-free C code, easy to check by production engineers)

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.



DUAL GRADIENT PROJECTION FOR QP

- Consider the strictly convex QP and its dual

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

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

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

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

- Primal solution: $z^k = -Q^{-1}(Fx + G'y^k)$
- Also works in fixed-point arithmetic (Patrinos, Guiggiani, Bemporad, 2015)
- Convergence is slow: the initial error $f(z^0) - f(z^*)$ reduces as $1/k$

FAST GRADIENT PROJECTION

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

- Solve (dual) QP by **fast gradient method**

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

$$\begin{array}{rcl} K & = & Q^{-1}G' \\ g & = & Q^{-1}Fx \\ L & \geq & \frac{1}{\lambda_{\max}(GQ^{-1}G')} \\ \beta_k & = & \max\{\frac{k-1}{k+2}, 0\} \end{array}$$

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

```
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**

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

  (Necoara, Nesterov, Glineur, 2018)

- Tight bounds on maximum number of iterations can be computed

ADMM

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

- Alternating Directions Method of Multipliers for QP

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

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

ρu = dual vector

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

(7 lines EML code)
($\approx$40 lines of C code)

- Very **simple to code**
- Sensitive to matrix **scaling** (as gradient projection)
- Used in many applications (control, signal processing, machine learning)

REGULARIZED ADMM FOR QUADRATIC PROGRAMMING

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

- Robust "regularized" ADMM iterations:

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

- Works for any $Q \succeq 0$, A, and choice of $\epsilon > 0$
- **Simple** to code, **fast**, and **robust**
- Only needs to factorize $\begin{bmatrix} Q+\epsilon I & A' \\ A & -\frac{1}{\rho}I \end{bmatrix}$ once
- Implemented in free osQP solver

  (Python interface: $\approx$ 800,000 downloads)

`http://osqp.org`

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)

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

(750 chars in Embedded MATLAB)

Bounded-Variable Least Squares (BVLS)

(Stark,Parker, 1995)

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

See Nilay's next talk

SOLVING QP'S VIA NONNEGATIVE LEAST SQUARES

(Bemporad, 2016)

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

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

$Q = Q' \succ 0$

$$Q = L'L \qquad u \triangleq Lz + L^{-T}c$$

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

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

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

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

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

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

ROBUST QP SOLVER BASED ON NNLS

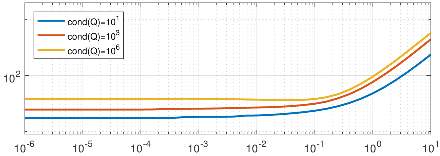
(Bemporad, 2018)

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

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

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

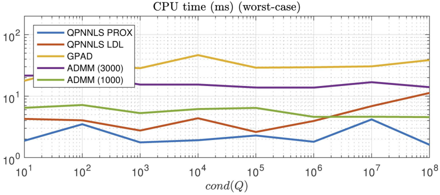
total number of active-set iterations as a function of ϵ

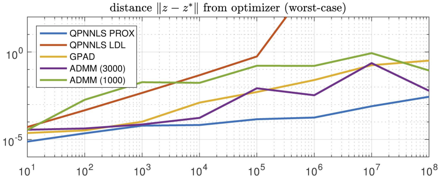


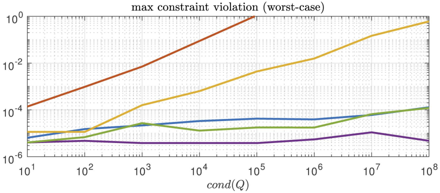
- Each QP is heavily warm-started and makes very few active-set changes
- Recursive LDL$^{\mathrm{T}}$ decompositions/rank-1 updates exploited for max efficiency

SOLVING QP'S VIA NNLS AND PROXIMAL POINT ITERATIONS

(Bemporad, 2018)





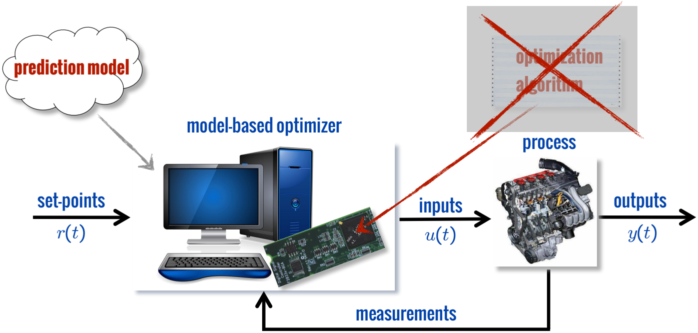


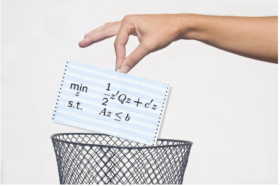
single precision arithmetic

30 vars, 100 constraints

(Macbook Pro 3 GHz Intel Core i7)

MPC WITHOUT ON-LINE QP





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

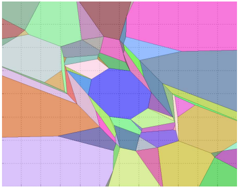
YES !

EXPLICIT MODEL PREDICTIVE CONTROL

- Continuous & piecewise affine solution of strictly convex multiparametric QP

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

(Bemporad, Morari, Dua, Pistikopoulos, 2002)



- Corollary: linear MPC is continuous & piecewise affine !

$$z^* = \begin{bmatrix} \mathbf{u_0} \\ u_1 \\ \vdots \\ u^*_{N-1} \end{bmatrix}$$

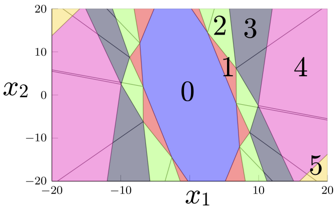
$$u_0^*(x) = \begin{cases} F_1 x + g_1 & \text{if} \quad H_1 x \leq K_1 \\ \vdots & \quad\vdots \\ F_M x + g_M & \text{if} \quad H_M x \leq K_M \end{cases}$$

- New mpQP solver based on NNLS available (Bemporad, 2015) and included in MPC Toolbox since R2014b (Bemporad, Morari, Ricker, 1998-today)

Is explicit MPC better than on-line QP (=implicit MPC) ?

COMPLEXITY CERTIFICATION FOR ACTIVE-SET QP SOLVERS

- **Result**: The **number of iterations** to solve the QP via a dual active-set method is a **piecewise constant function** of the parameter x



(Cimini, Bemporad, 2017)

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

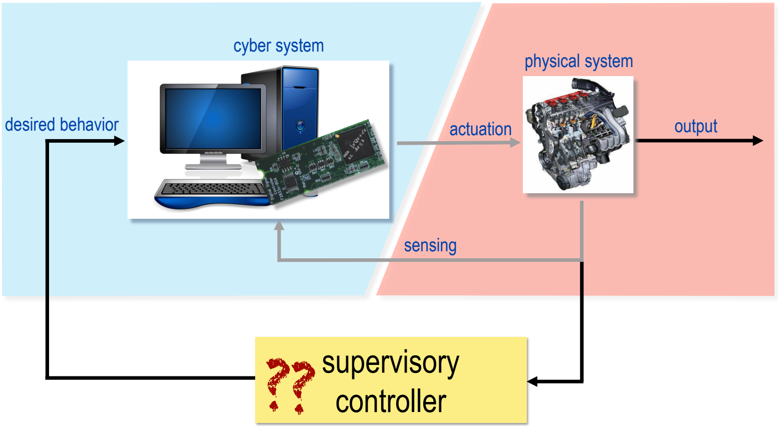
- Examples (from MPC Toolbox):

	inverted pendulum	DC motor	nonlinear demo	AFTI F16
Explicit MPC				
max flops	3382	1689	9184	16434
max memory (kB)	55	30	297	430
Implicit MPC				
max flops	3809	2082	7747	7807
sqrt	27	9	37	33
max memory (kB)	15	13	20	16

- QP certification algorithm currently used in industrial production projects

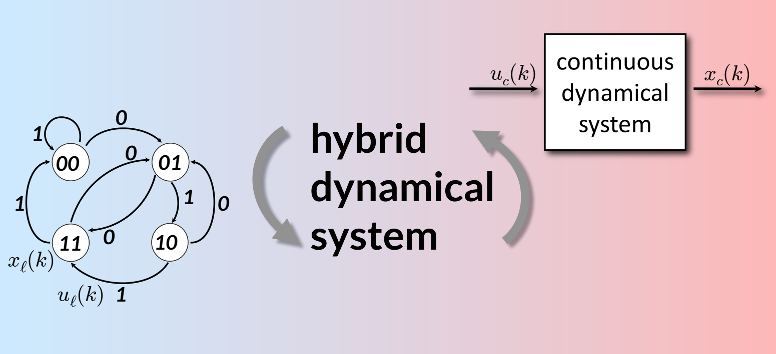
HYBRID MPC OF CYBER-PHYSICAL SYSTEMS

CONTROL OF CYBER-PHYSICAL SYSTEMS



- Can we use MPC to synthesize a supervisory controller of a CPS?

HYBRID DYNAMICAL SYSTEMS

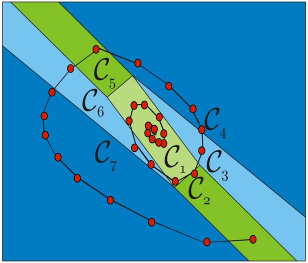


- Variables are **binary-valued**
  $x_\ell \in \{0,1\}^{n_\ell}$, $u_\ell \in \{0,1\}^{m_\ell}$
- Dynamics = **finite state machine**
- **Logic constraints**

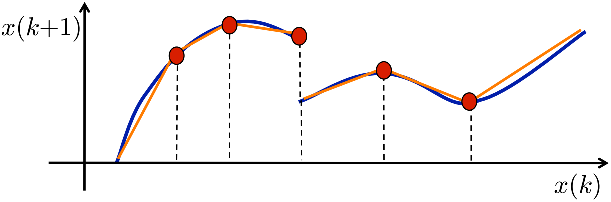
- Variables are **real-valued**
  $x_c \in \mathbb{R}^{n_c}$, $u_c \in \mathbb{R}^{m_c}$
- **Difference/differential equations**
- **Linear inequality** constraints

PIECEWISE AFFINE SYSTEMS

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

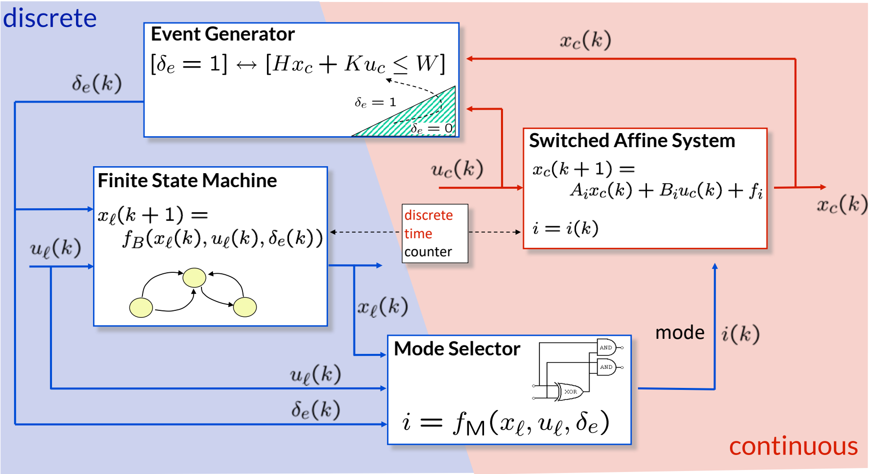


- PWA systems can approximate nonlinear dynamics arbitrarily well (even discontinuous ones)



DISCRETE HYBRID AUTOMATON (DHA)

(Torrisi, Bemporad, 2004)



$x_\ell \in \{0,1\}^{n_\ell}$	$=$	binary state	$x_c \in \mathbb{R}^{n_c}$	$=$	real-valued state
$u_\ell \in \{0,1\}^{m_\ell}$	$=$	binary input	$u_c \in \mathbb{R}^{m_c}$	$=$	real-valued input
$\delta_e \in \{0,1\}^{n_e}$	$=$	event variable	$i \in \{1,\ldots,s\}$	$=$	current mode

TRANSFORMATION OF A DHA INTO LINEAR (IN)EQUALITIES

$X_1 \vee X_2 = \text{TRUE} \quad \longrightarrow \quad \delta_1 + \delta_2 \geq 1, \qquad \delta_1, \delta_2 \in \{0, 1\}$

Any logic statement

$f(X) = \text{TRUE}$

$$\bigwedge_{j=1}^{m} \left(\vee_{i \in P_j} X_i \vee_{i \in N_j} \neg X_i \right) \quad \text{(CNF)}$$

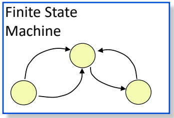
$N_j, P_j \subseteq \{1, \ldots, n\}$

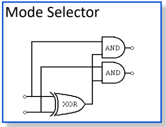
$$\longrightarrow \quad \begin{cases} 1 \leq \sum_{i \in P_1} \delta_i + \sum_{i \in N_1} (1 - \delta_i) \\ \vdots \\ 1 \leq \sum_{i \in P_m} \delta_i + \sum_{i \in N_m} (1 - \delta_i) \end{cases}$$

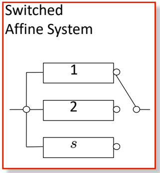
$$[\delta_e^i(k) = 1] \leftrightarrow [H^i x_c(k) \leq W^i] \quad \longrightarrow \quad \begin{cases} H^i x_c(k) - W^i \leq M^i (1 - \delta_e^i(k)) \\ H^i x_c(k) - W^i > m^i \delta_e^i(k) \end{cases}$$

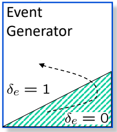
IF $[\delta = 1]$ THEN $z = a_1^T x + b_1^T u + f_1$
ELSE $z = a_2^T x + b_2^T u + f_2$

$$\longrightarrow \quad \begin{cases} (m_1 - M_2)(1 - \delta) + z & \leq & a_1 x + b_1 u + f_1 \\ (m_2 - M_1)(1 - \delta) - z & \leq & -a_1 x - b_1 u - f_1 \\ (m_2 - M_1)\delta + z & \leq & a_2 x + b_2 u + f_2 \\ (m_1 - M_2)\delta - z & \leq & -a_2 x - b_2 u - f_2 \end{cases}$$





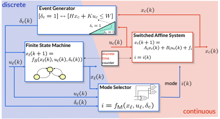




MIXED LOGICAL DYNAMICAL (MLD) SYSTEMS

(Bemporad, Morari, 1999)

- By converting logic relations into mixed-integer linear inequalities a DHA can be rewritten as the **Mixed Logical Dynamical (MLD)** system



$$\begin{cases} x(k+1) &= Ax(k) + B_1u(k) + B_2\delta(k) + B_3z(k) + B_5 \\ y(k) &= Cx(k) + D_1u(k) + D_2\delta(k) + D_3z(k) + D_5 \\ E_2\delta(k) &+ E_3z(k) \leq E_4x(k) + E_1u(k) + E_5 \end{cases}$$

$$x \in \mathbb{R}^{n_c} \times \{0,1\}^{n_b},\ u \in \mathbb{R}^{m_c} \times \{0,1\}^{m_b}$$
$$y \in \mathbb{R}^{p_c} \times \{0,1\}^{p_b},\ \delta \in \{0,1\}^{r_b},\ z \in \mathbb{R}^{r_c}$$

- The translation from DHA to MLD can be automatized, see e.g. the language **HYSDEL** (HYbrid Systems DEscription Language) (Torrisi, Bemporad, 2004)

- MLD models allow solving MPC, verification, state estimation, and fault detection problems via **mixed-integer programming**

HYBRID MPC

(Bemporad, Morari, 1999)

- Finite-horizon optimal control problem (regulation)

$$\min \sum_{k=0}^{N-1} y_k' Q y_k + u_k' R u_k$$

$$\text{s.t.} \quad \left\{ \begin{array}{rcl} x_{k+1} & = & Ax_k + B_1 u_k + B_2 \delta_k + B_3 z_k + B_5 \\ y_k & = & Cx_k + D_1 u_k + D_2 \delta_k + D_3 z_k + D_5 \\ E_2 \delta_k & + & E_3 z_k \leq E_4 x_k + E_1 u_k + E_5 \\ x_0 & = & x(t) \end{array} \right.$$

$Q = Q' \succ 0, R = R' \succ 0$

- Treat u_k, δ_k, z_k as free decision variables, $k = 0, \ldots, N-1$

- Predictions can be constructed **exactly as in the linear case**

$$x_k = A^k x_0 + \sum_{j=0}^{k-1} A^j (B_1 u_{k-1-j} + B_2 \delta_{k-1-j} + B_3 z_{k-1-j} + B_5)$$

MIQP FORMULATION OF HYBRID MPC

(Bemporad, Morari, 1999)

- After substituting x_k, y_k we get the **Mixed-Integer Quadratic Programming (MIQP)** problem

$$\begin{array}{rl} \min_\xi & \frac{1}{2}\xi' H\xi + x'(t)F'\xi + \frac{1}{2}x'(t)Yx(t) \\ \text{s.t.} & G\xi \leq W + Sx(t) \end{array}$$

- The optimization vector $\xi = [u_0, \ldots, u_{N-1}, \delta_0, \ldots, \delta_{N-1}, z_0, \ldots, z_{N-1}]$ has **mixed real and binary** components

- Hybrid modeling and MPC design available in **Hybrid Toolbox for MATLAB**

(Bemporad, 2003-today)

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

≈8000 downloads

≈1.5 downloads/day

CLOSED-LOOP CONVERGENCE

(Bemporad, Morari, 1999)

- **Theorem**. Let $(x_r, u_r, \delta_r, z_r)$ be the equilibrium corresponding to r. Assume $x(0)$ such that the MIQP problem **is feasible at time** $t = 0$. Then $\forall Q, R \succ 0, \sigma > 0$ the hybrid MPC closed-loop **converges asymptotically**

$$\lim_{t\to\infty} y(t) = r \qquad \lim_{t\to\infty} x(t) = x_r$$

$$\lim_{t\to\infty} \delta(t) = \delta_r$$

$$\lim_{t\to\infty} u(t) = u_r \qquad \lim_{t\to\infty} z(t) = z_r$$

  and **all constraints are fulfilled** at each time $t \geq 0$.

- The proof easily follows from standard Lyapunov arguments (see next slide)

- **Lyapunov asymptotic stability** and **exponential stability** follows if proper terminal cost and constraints are imposed (Lazar, Heemels, Weiland, Bemporad, 2006)

MIXED-INTEGER PROGRAMMING SOLVERS

- Binary constraints make Mixed-Integer Programming (MIP) a hard problem ($\mathcal{NP}$-complete)

- However, excellent general purpose **branch & bound** / **branch & cut** solvers available for MILP and MIQP (CPLEX, GLPK, Xpress-MP, CBC, Gurobi, ...)

- MIQP approaches tailored to embedded hybrid MPC applications:
  - B&B + (dual) active set methods for QP
    (Leyffer, Fletcher, 1998) (Axehill, Hansson, 2006) (Bemporad, 2015) (Bemporad, Naik, 2018)
  - B&B + interior point methods: (Frick, Domahidi, Morari, 2015)
  - B&B + fast gradient projection: (Naik, Bemporad, 2017)
  - B&B + ADMM: (Stellato, Naik, Bemporad, Goulart, Boyd, 2018)

- No need to reach global optimum (see convergence proof)

BRANCH & BOUND METHOD FOR MIQP

- We want to solve the following MIQP

$$\begin{array}{rl} \min & V(z) \triangleq \frac{1}{2}z'Qz + c'z \\ \text{s.t.} & Az \leq b \\ & z_i \in \{0,1\},\ \forall i \in I \end{array} \qquad \begin{array}{l} z \in \mathbb{R}^n \\ Q = Q' \succeq 0 \\ I \subseteq \{1,\ldots,n\} \end{array}$$

- **Branch & Bound (B&B)** is the simplest (and most popular) approach to solve the problem to optimality
- **Key idea** of B&B:
  - each binary variable z_i, $i \in I$, is either set to 0, or 1, or relaxed in $[0,1]$
  - solve the corresponding **QP relaxation** of the MIQP problem
  - use QP result to decide the next combination of fixed/relaxed variables, or to conclude that the optimal solution has been found, or that no solution exist

SOLVING MIQP VIA NNLS AND PROXIMAL-POINT ITERATIONS

(Bemporad, Naik, 2018)

- **Robustified approach**: use **NNLS + proximal-point iterations** to solve QP relaxations (Bemporad, 2018)

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

- CPU time (ms) on **MIQP** coming from hybrid MPC (`bm99` demo):

For $N = 10$:
30 real vars
10 binary vars
160 inequalities

prox-NNLS* = warm start of binary vars exploited

N	prox-NNLS		prox-NNLS*		GUROBI		CPLEX	
	avg	max	avg	max	avg	max	avg	max
2	2.0	2.6	2.0	2.6	1.6	2.0	3.1	6.0
4	5.3	8.8	3.1	6.9	3.1	3.9	8.9	15.7
8	29.7	71.0	8.1	43.4	7.2	13.2	15.5	80.2
10	76.2	146.1	14.4	103.2	11.1	17.6	35.1	95.3
12	155.8	410.8	26.9	263.4	14.9	31.2	61.7	103.7
15	484.2	1242.3	61.7	766.9	25.9	109.8	89.9	181.1

CPU time measured on Intel Core i7-4700MQ CPU 2.40 GHz

FAST GRADIENT PROJECTION FOR MIQP

(Naik, Bemporad, 2017)

- Consider again the MIQP problem with Hessian $Q = Q' \succ 0$

$$\begin{array}{rl} \min\limits_z & V(z) \triangleq \frac{1}{2} z'Qz + c'z \\ \text{s.t.} & \ell \leq Az \leq u \\ & Gz = g \\ & \bar{A}_i z \in \{\bar{\ell}_i, \bar{u}_i\},\ i = 1, \ldots, p \end{array}$$

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

- Use B&B and **fast gradient projection** to solve dual of QP relaxation

constraint is relaxed	$\bar{A}_i z \leq \bar{u}_i$	$\rightarrow$	$y_i^{k+1} = \max\left\{y_i^k + s_i^k, 0\right\}$	$(y_i \geq 0)$
constraint is fixed	$\bar{A}_i z = \bar{u}_i$	$\rightarrow$	$y_i^{k+1} = y_i^k + s_i^k$	$(y_i \lessgtr 0)$
constraint is ignored	$\bar{A}_i z = \bar{\ell}_i$	$\rightarrow$	$y_i^{k+1} = 0$	$(y_i = 0)$

FAST GRADIENT PROJECTION FOR MIQP

(Naik, Bemporad, 2017)

- **Same dual QP matrices** at each node, **preconditioning** computed only once
- **Warm-start** exploited, **dual cost** used to stop QP relaxations earlier
- Criterion based on Farkas lemma to detect **QP infeasibility**
- Numerical results (time in ms):

n	m	p	q	miqpGPAD	GUROBI
10	100	2	2	15.6	6.56
50	25	5	3	3.44	8.74
50	150	10	5	63.22	46.25
100	50	2	5	6.22	26.24
100	200	15	5	164.06	188.42
150	100	5	5	31.26	88.13
150	200	20	5	258.80	274.06
200	50	15	6	35.08	144.38

CPU time measured on Intel Core i7-4700MQ CPU 2.40 GHz

HEURISTIC ADMM METHOD FOR (SUBOPTIMAL) MIQP

(Takapoui, Moehle, Boyd, Bemporad, 2017)

- Consider again MIQP problem

$$
\begin{array}{ll}
\min & \frac{1}{2}x'Qx + q'x \\
\text{s.t.} & \ell \leq Ax \leq u \\
 & A_i x \in \{\ell_i, u_i\},\ i \in I
\end{array}
$$

- ADMM iterations:

quantization step

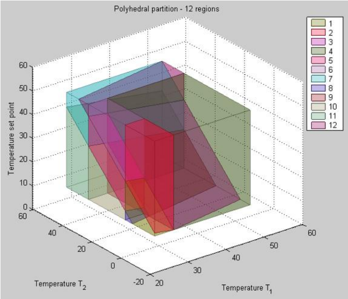
$$
\begin{array}{rcl}
x^{k+1} & = & -(Q + \rho A^T A)^{-1}(\rho A^T (y^k - z^k) + q) \\
z^{k+1} & = & \min\{\max\{Ax^{k+1} + y^k, \ell\}, u\} \\
z_i^{k+1} & = & \left\{\begin{array}{lll} \ell_i & \text{if} & z_i^{k+1} < \frac{\ell_i + u_i}{2} \\ u_i & \text{if} & z_i^{k+1} \geq \frac{\ell_i + u_i}{2} \end{array}\right., \ i \in I \\
y^{k+1} & = & y^k + Ax^{k+1} - z^{k+1}
\end{array}
$$

- Iterations converge to a (local) solution

- Similar idea also applicable to fast gradient methods (Naik, Bemporad, 2017)

EXPLICIT HYBRID MPC

- It is possible to write hybrid MPC laws in explicit form too !
- The explicit MPC law is still piecewise affine on polyhedra



(Bemporad, Borrelli, Morari, 2000)

(Mayne, ECC 2001)

(Mayne, Rakovic, 2002)

(Bemporad, Hybrid Toolbox, 2003)

(Borrelli, Baotic, Bemporad, Morari, 2005)

(Alessio, Bemporad, 2006)

- The control law may be discontinuous, polyhedra may overlap
- Comparison of quadratic costs can be avoided by lifting the parameter space

(Fuchs, Axehill, Morari, 2015)

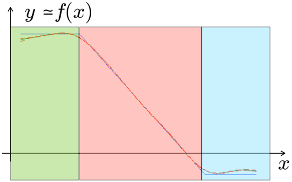
LEARNING PWA MODELS FROM DATA

- **Problem**: Given input/output pairs $\{x(k), y(k)\}, k = 1, \ldots, N$ and number s of models, learn a **piecewise affine** (PWA) model $y \approx f(x)$

$$f(x) = \begin{cases} F_1 x + g_1 & \text{if } H_1 x \leq K_1 \\ \vdots & \\ F_s x + g_s & \text{if } H_s x \leq K_s \end{cases}$$

- Need to learn **both** the parameters $\{F_i,\ g_i\}$ of the affine submodels **and** the partition $\{H_i,\ K_i\}$ of the PWA map from data (**off-line learning**)

- Possibly update model and partition as new data become available (**on-line learning**)



APPROACHES TO PWA SYSTEM IDENTIFICATION

- Mixed-integer linear or quadratic programming (Roll, Bemporad, Ljung, 2004)
- Partition of infeasible set of inequalities (Bemporad, Garulli, Paoletti, Vicino, 2005)
- K-means clustering in a feature space (Ferrari-Trecate, Muselli, Liberati, Morari, 2003)
- Bayesian approach (Juloski, Wieland, Heemels, 2004)
- Kernel-based approaches (Pillonetto, 2016)
- Hyperplane clustering in data space (Münz, Krebs, 2002)
- **Recursive multiple least squares & PWL separation** (Breschi, Piga, Bemporad, 2016)

PWA REGRESSION ALGORITHM

(Breschi, Piga, Bemporad, 2016)

1. Estimate models $\{F_i,\ g_i\}$ **recursively**. Let $e_i(k) = y(k) - F_i x(k) - g_i$ and only update model $i(k)$ such that

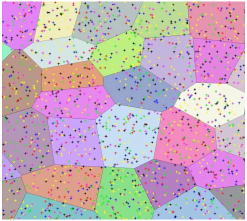
$$i(k) \leftarrow \arg\min_{i=1,\ldots,s} \underbrace{e_i(k)'\Lambda_e^{-1}e_i(k)}_{\text{one-step prediction error of model \#i}} + \underbrace{(x(k)-c_i)'\,R_i^{-1}(x(k)-c_i)}_{\text{proximity to centroid of cluster \#i}}$$

using **recursive LS** and **inverse QR decomposition** (Alexander, Ghirnikar, 1993)

This also splits the data points $x(k)$ in **clusters** $C_i = \{x(k):\ i(k) = i\}$

2. Compute a polyhedral partition $\{H_i,\ K_i\}$ of the regressor space via **multi-category linear separation**

$$\phi(x) = \max_{i=1,\ldots,s} \{w_i' x - \gamma_i\}$$



PWA REGRESSION EXAMPLES

(Breschi, Piga, Bemporad, 2016)

- Identification of **piecewise-affine ARX** model

$$\begin{bmatrix} y_1(k) \\ y_2(k) \end{bmatrix} = \begin{bmatrix} -0.83 & 0.20 \\ 0.30 & -0.52 \end{bmatrix} \begin{bmatrix} y_1(k-1) \\ y_2(k-1) \end{bmatrix} + \begin{bmatrix} -0.34 & 0.45 \\ -0.30 & 0.24 \end{bmatrix} \begin{bmatrix} u_1(k-1) \\ u_2(k-1) \end{bmatrix}$$
$$+ \begin{bmatrix} 0.20 \\ 0.15 \end{bmatrix} + \max \left\{ \begin{bmatrix} 0.20 & -0.90 \\ 0.10 & -0.42 \end{bmatrix} \begin{bmatrix} y_1(k-1) \\ y_2(k-1) \end{bmatrix} \right.$$
$$\left. + \begin{bmatrix} 0.42 & 0.20 \\ 0.50 & 0.64 \end{bmatrix} \begin{bmatrix} u_1(k-1) \\ u_2(k-1) \end{bmatrix} + \begin{bmatrix} 0.40 \\ 0.30 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \end{bmatrix} \right\} + e_o(k),$$

- **Quality of fit**: best fit rate (BFR) = $\max \left\{ 1 - \frac{\|y_{o,i} - \hat{y}_i\|_2}{\|y_{o,i} - \bar{y}_{o,i}\|_2}, 0 \right\}, i = 1, 2$

		$N = 4000$	$N = 20000$	$N = 100000$
y_1	(Off-line) RLP	96.0 %	96.5 %	99.0 %
	(Off-line) RPSN	96.2 %	96.4 %	98.9 %
	(On-line) ASGD	86.7 %	95.0 %	96.7 %
y_2	(Off-line) RLP	96.2 %	96.9 %	99.0 %
	(Off-line) RPSN	96.3 %	96.8 %	99.0 %
	(On-line) ASGD	87.4 %	95.2 %	96.4 %

RLP = Robust linear programming
(Bennett, Mangasarian, 1994)

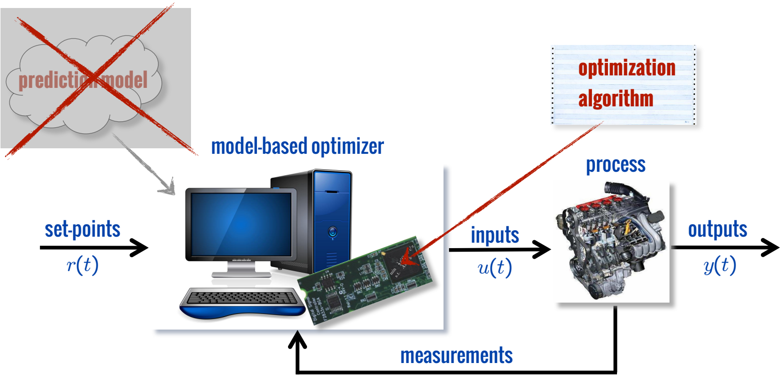
RPSN = Piecewise-smooth Newton method
(Bemporad, Bernardini, Patrinos, 2015)

ASGD = Averaged stochastic gradient descent
(Bottou, 2012)

- **CPU time for computing the partition**:

	$N = 4000$	$N = 20000$	$N = 100000$
(Off-line) RLP	0.308 s	3.227 s	112.435 s
(Off-line) RPSN	0.016 s	0.086 s	0.365 s
(On-line) ASGD	0.013 s	0.023 s	0.067 s

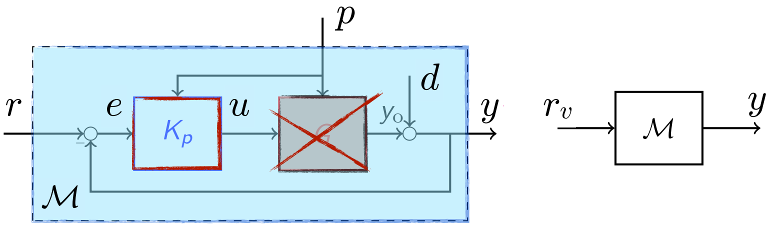
DATA-DRIVEN MPC



- Can we design an MPC controller **without** first identifying a model of the **open-loop process** ?

DATA-DRIVEN DIRECT CONTROLLER SYNTHESIS

(Campi, Lecchini, Savaresi, 2002) (Formentin et al., 2015)

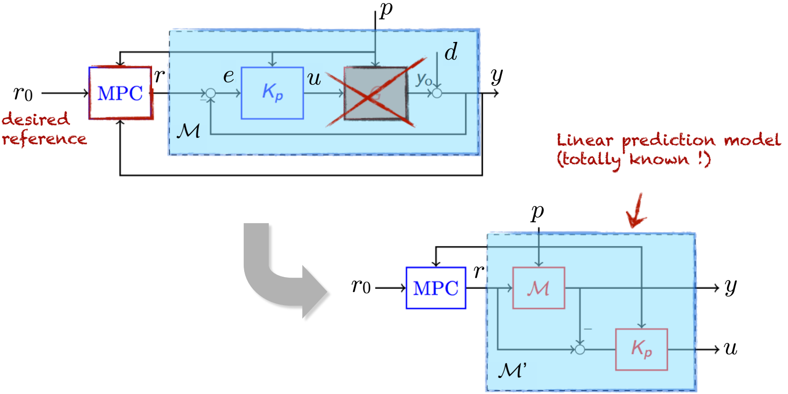


- Collect a set of **data** $\{u(t), y(t), p(t)\}, t = 1, \ldots, N$
- Specify a **desired closed-loop linear model** $\mathcal{M}$ from r to y
- Compute $r_v(t) = \mathcal{M}^{\#} y(t)$ from **pseudo-inverse model** $\mathcal{M}^{\#}$ of $\mathcal{M}$
- **Identify** linear (LPV) model K_p from $e_v = r_v - y$ (virtual tracking error) to u

DATA-DRIVEN MPC

- Design a linear MPC (**reference governor**) to generate the reference r

(Bemporad, Mosca, 1994) (Gilbert, Kolmanovsky, Tan, 1994)



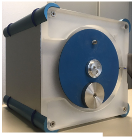
- MPC designed to handle input/output **constraints** and improve **performance**

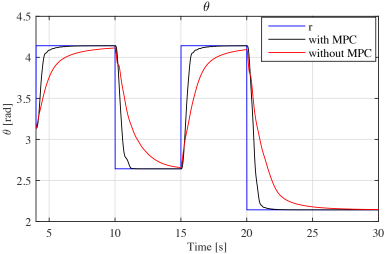
(Piga, Formentin, Bemporad, 2017)

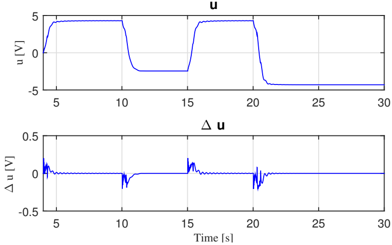
DATA-DRIVEN MPC - AN EXAMPLE

- Experimental results: MPC handles soft constraints on u, Δu and y

(motor equipment by courtesy of TU Delft)







desired tracking performance achieved

constraints on input increments satisfied

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

OPTIMAL DATA-DRIVEN MPC

(Selvi, Piga, Bemporad, 2018)

- Can we choose $\mathcal{M}$ from data so that K_p is an **optimal controller** ?

- **Idea**: parameterize desired closed-loop model $\mathcal{M}(\theta)$ and optimize

$$\min_{\theta} J(\theta) = \frac{1}{N}\sum_{t=0}^{N-1} \underbrace{W_y(r(t)-y_p(\theta,t))^2 + W_{\Delta u}\Delta u_p^2(\theta,t)}_{\text{performance index}} + \underbrace{W_{\text{fit}}(u(t)-u_v(\theta,t))^2}_{\text{identification error}}$$

- Evaluating $J(\theta)$ requires synthesizing $K_p(\theta)$ from data and simulating the nominal model and control law

$$y_p(\theta,t) = \mathcal{M}(\theta)r(t) \qquad u_p(\theta,t) = K_p(\theta)(r(t)-y_p(\theta,t))$$

$$\Delta u_p(\theta,t) = u_p(\theta,t) - u_p(\theta,t-1)$$

- Optimal θ obtained by solving a **(non-convex) nonlinear programming** problem

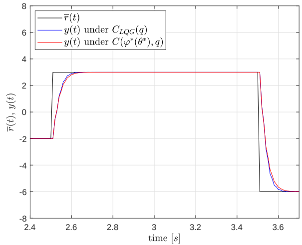
OPTIMAL DATA-DRIVEN MPC

(Selvi, Piga, Bemporad, 2018)

- **Results**: linear process

$$G(z) = \frac{z - 0.4}{z^2 + 0.15z - 0.325}$$

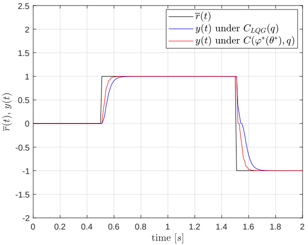
The data-driven controller is **only 1.3% worse** than model-based LQR



- **Results**: nonlinear (Wiener) process

$$\begin{aligned} y_L(t) &= G(z)u(t) \\ y(t) &= |y_L(t)| \arctan(y_L(t)) \end{aligned}$$

The data-driven controller is **24% better** than LQR based on identified open-loop model !



ONGOING RESEARCH ON LEARNING MPC FROM DATA

- **Goal**: learn MPC law from data that optimizes a given index

- **Q-learning**: learn Q-function defining the MPC law from data

  (Gros, Zanon, in press) (Zanon, Gros, Bemporad, 2019)

- **Policy gradient methods**: learn optimal policy coefficients directly from data using stochastic gradient descent (Ferrarotti, Bemporad, 2019)

- **Global optimization methods**: learn MPC parameters (weights, models, horizon, solver tolerances, ...) by optimizing observed closed-loop performance

  (Piga, Forgione, Formentin, Bemporad, 2019) (Forgione, Piga, Bemporad, in preparation)

- **Lessons learned** so far: if chosen model/policy structure does not include real plant/optimal policy
  - optimal policy **learned from data** can be better than **model-based** optimal policy
  - when open-loop model is used as a tuning parameter, **learned model** can be quite different from best **open-loop model** that can be identified from the same data

CONCLUSIONS

- MPC is a **universal control methodology** to provide autonomy to many CPS's:
  - different **models** (linear, nonlinear, hybrid, stochastic, ...)
  - **optimize** closed-loop performance subject to **constraints**
  - **widely applicable** to many industrial sectors

- **MPC research**:
  1. Linear, uncertain, explicit, hybrid, nonlinear MPC: **mature theory**
  2. Stochastic MPC, economic MPC: **still open issues**
  3. Embedded optimization methods for MPC: **still room for many new ideas**
  4. Data-driven MPC: **a lot of open issues**. There is a lot to "learn" from machine learning

- **MPC technology**: is MPC mature for widespread use in industrial applications ?

MPC IN AUTOMOTIVE PRODUCTION

The MPC developed by **General Motors** and **ODYS** for torque tracking in turbocharged gasoline engines **is in high-volume production since 2018**

- Multivariable system, **4 inputs, 4 outputs**. QP solved **in real time on ECU**
  (Bemporad, Bernardini, Long, Verdejo, 2018)

- Supervisory **MPC for powertrain control** also in production since 2018
  (Bemporad, Bernardini, Livshiz, Pattipati, 2018)

First known mass production of MPC in the automotive industry

http://www.odys.it/odys-and-gm-bring-online-mpc-to-production

ODYS Advanced Controls & Optimization