WORKSHOP ON

MODEL PREDICTIVE CONTROL

FROM THE BASICS TO REINFORCEMENT LEARNING

Alberto Bemporad alberto.bemporad@imtlucca.it

Mario Zanon mario.zanon@imtlucca.it

IMT SCHOOL FOR ADVANCED STUDIES LUCCA

CDC'19, Nice, France December 10, 2019

WORKSHOP PROGRAM

- Linear MPC: introduction and algorithms (AB)
- Nonlinear and economic MPC (MZ)
- Hybrid and stochastic MPC (AB)
- Reinforcement learning and MPC (AB+MZ)
- Concluding remarks (AB+MZ)

Workshop slides available at:

`http://dysco.imtlucca.it/mpc-cdc19`

SUPPLEMENTARY MATERIAL

- The contents presented in this workshop are an excerpt of **two PhD courses** held every year at IMT Lucca, Italy:
  - A. Bemporad - **Model Predictive Control**

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

    April 1-3, 6-7, 2020

  - M. Zanon - **Numerical Methods for Optimal Control**

    `https://mariozanon.wordpress.com/teaching/`

    May 25-29, 2020

- Registration is **free**, but compulsory

WORKSHOP PROGRAM

- Linear MPC: introduction and algorithms (AB)
- Nonlinear and economic MPC (MZ)
- Hybrid and stochastic MPC (AB)
- Reinforcement learning and MPC (AB+MZ)
- Concluding remarks (AB+MZ)

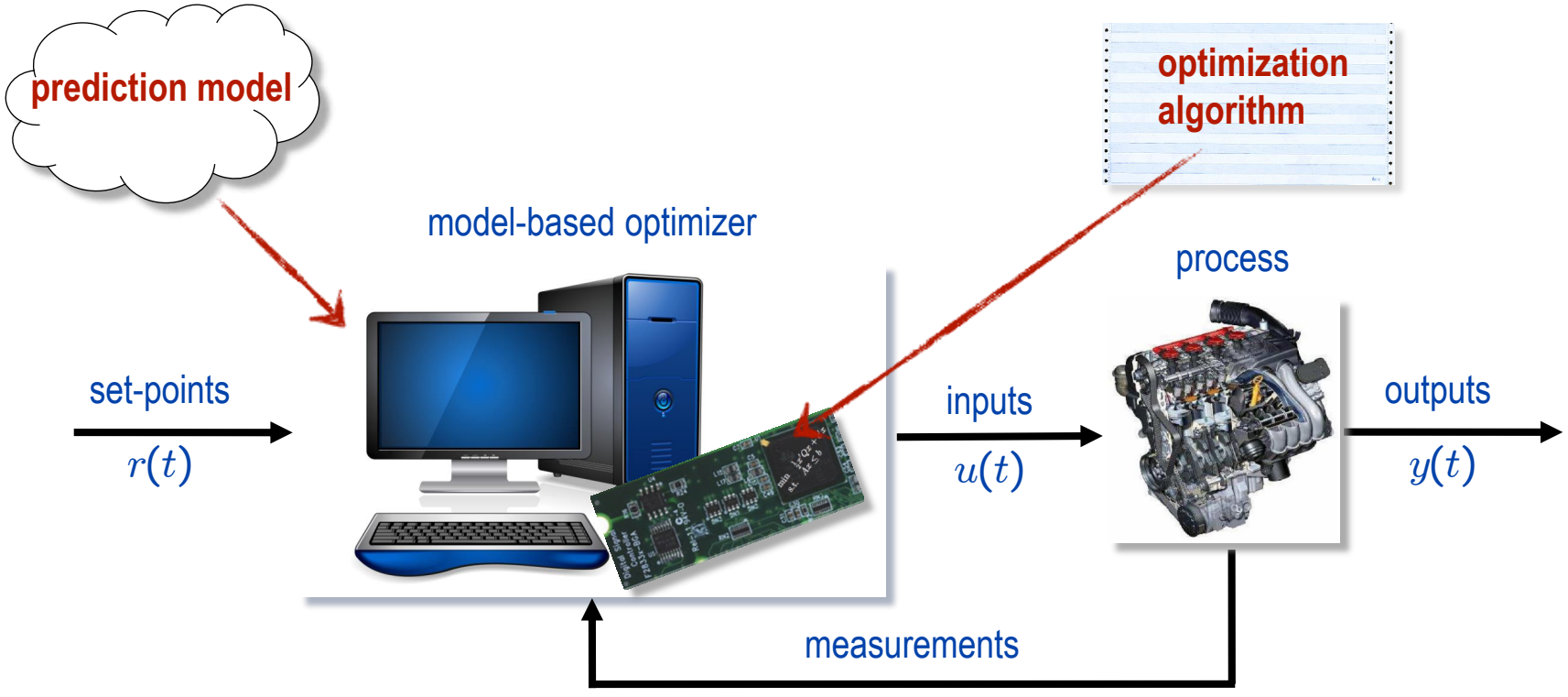
Supplementary material:

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

`https://mariozanon.wordpress.com/teaching/`

MODEL PREDICTIVE CONTROL: BASIC CONCEPTS

MODEL PREDICTIVE CONTROL (MPC)



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

simplified · likely · a good

MODEL PREDICTIVE CONTROL (MPC)

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

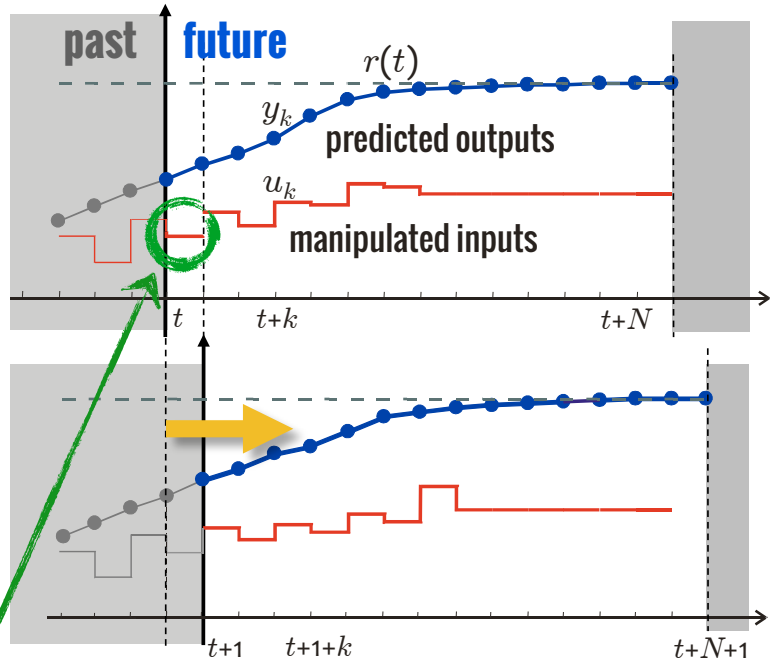
$$\min \sum_{k=0}^{N-1} \|W^y(y_k - r(t))\|_2^2 + \|W^u(u_k - u_r(t))\|_2^2$$

$$\text{s.t.} \quad x_{k+1} = f(x_k, u_k) \qquad \text{prediction model}$$
$$y_k = g(x_k)$$

$$u_{\min} \le u_k \le u_{\max} \qquad \text{constraints}$$
$$y_{\min} \le y_k \le y_{\max}$$

$$x_0 = x(t) \qquad \text{state feedback}$$

numerical optimization problem

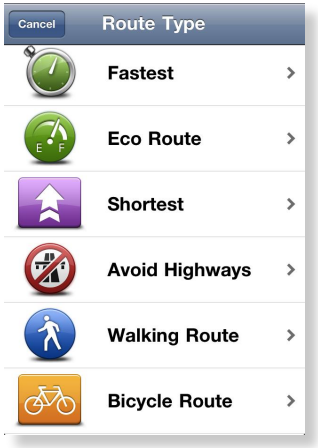


- **At each time** t:
  - get new measurements to update the estimate of the current state $x(t)$
  - solve the optimization problem with respect to $\{u_0, \ldots, u_{N-1}\}$
  - apply only the first optimal move $u(t) = u_0^*$, discard the remaining samples

DAILY-LIFE EXAMPLES OF MPC

- MPC is like playing chess !

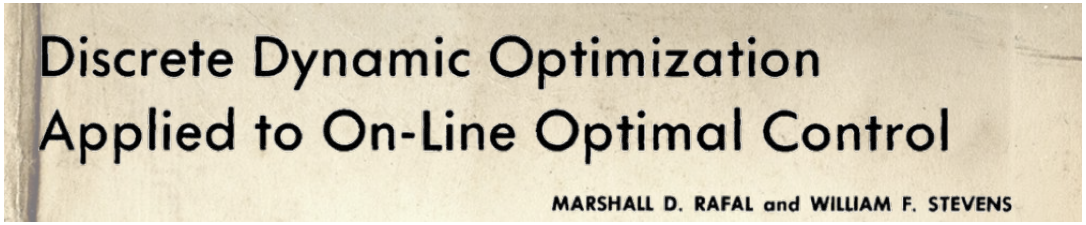
- On-line (event-based) re-planning used in GPS navigation



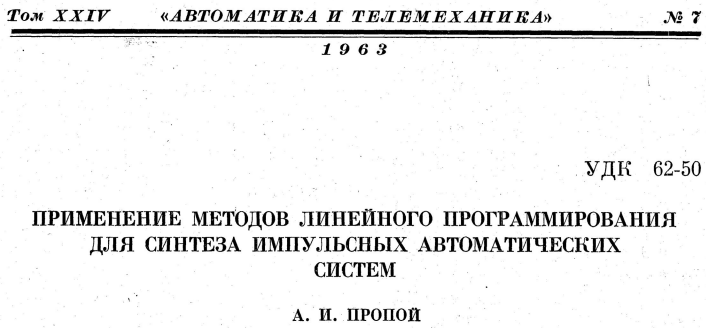
- You use MPC too when you drive !

MPC IN INDUSTRY

- The MPC concept dates back to the 60's


Discrete Dynamic Optimization
Applied to On-Line Optimal Control
MARSHALL D. RAFAL and WILLIAM F. STEVENS

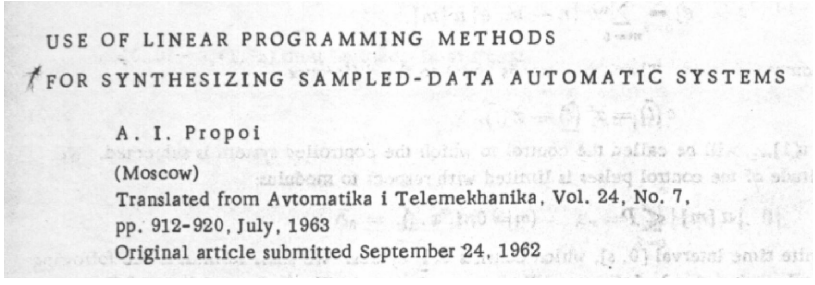
(Rafal, Stevens, AiChE Journal, 1968)


Том XXIV «АВТОМАТИКА И ТЕЛЕМЕХАНИКА» № 7
1963

УДК 62-50

ПРИМЕНЕНИЕ МЕТОДОВ ЛИНЕЙНОГО ПРОГРАММИРОВАНИЯ ДЛЯ СИНТЕЗА ИМПУЛЬСНЫХ АВТОМАТИЧЕСКИХ СИСТЕМ

А. И. ПРОПОЙ


USE OF LINEAR PROGRAMMING METHODS
FOR SYNTHESIZING SAMPLED-DATA AUTOMATIC SYSTEMS

A. I. Propoi
(Moscow)
Translated from Avtomatika i Telemekhanika, Vol. 24, No. 7,
pp. 912-920, July, 1963
Original article submitted September 24, 1962

(Propoi, 1963)

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

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

Today APC (advanced process control) = MPC


©SimulateLive.com

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.

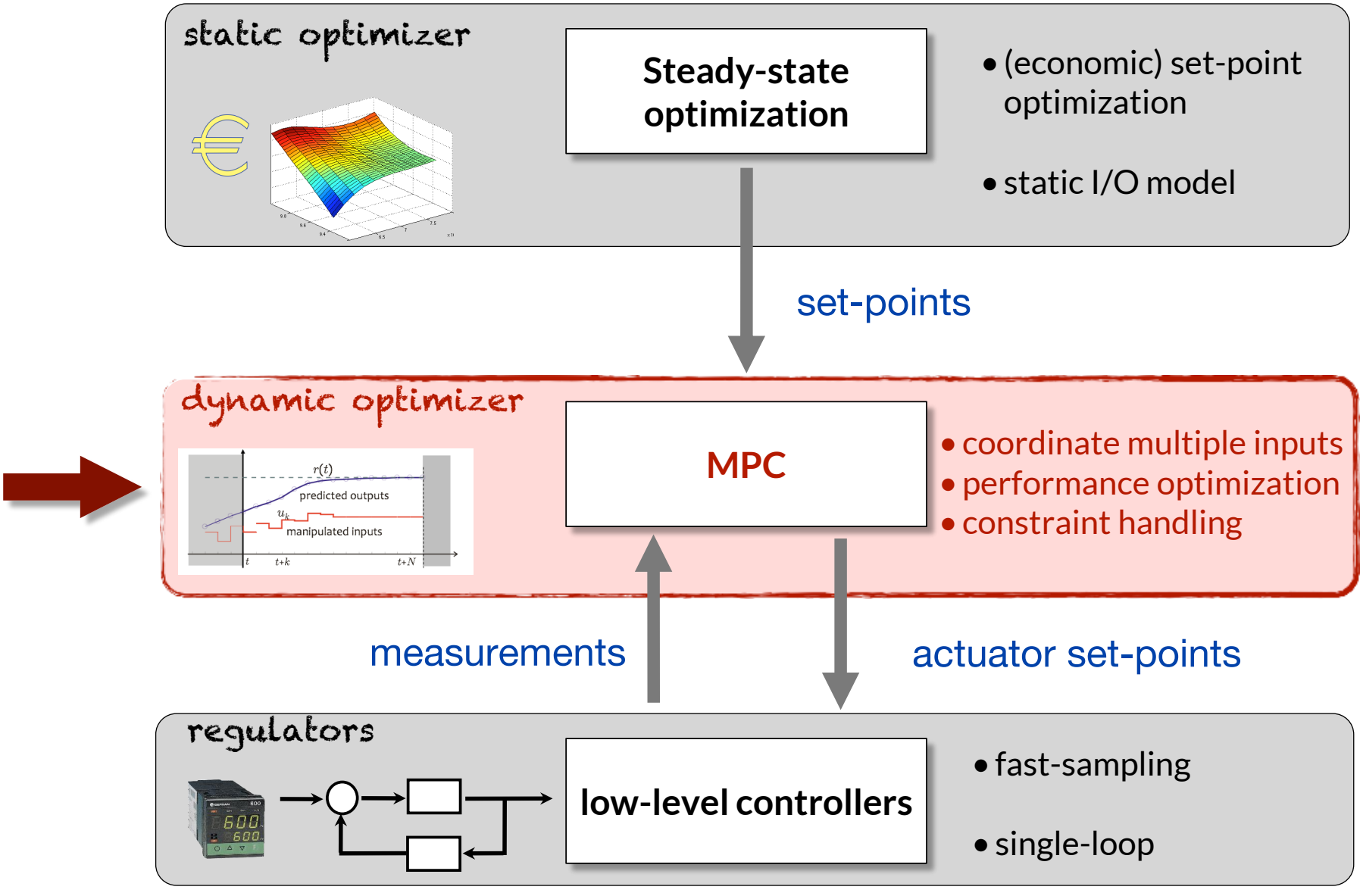
Rank and Technology	High-Impact Ratings	Low- or No-Impact Ratings
PID control	100%	0%
Model predictive control	78%	9%
System identification	61%	9%
Process data analytics	61%	17%
Soft sensing	52%	22%
Fault detection and identification	50%	18%
Decentralized and/or coordinated control	48%	30%
Intelligent 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



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

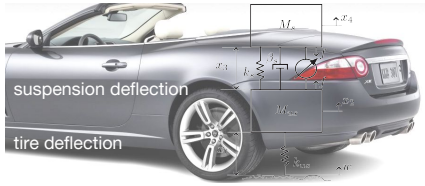
Jaguar

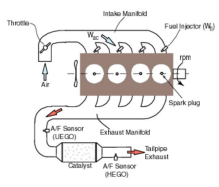
DENSO Automotive

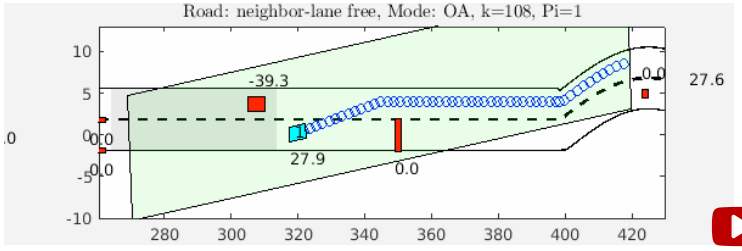
FCA

General Motors

ODYS



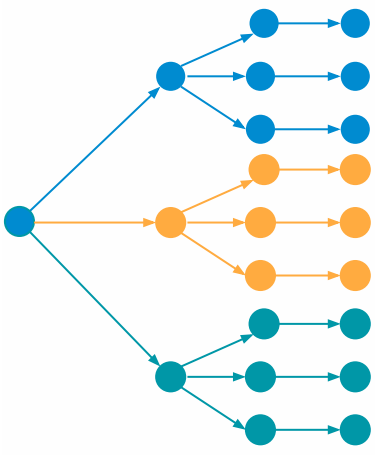


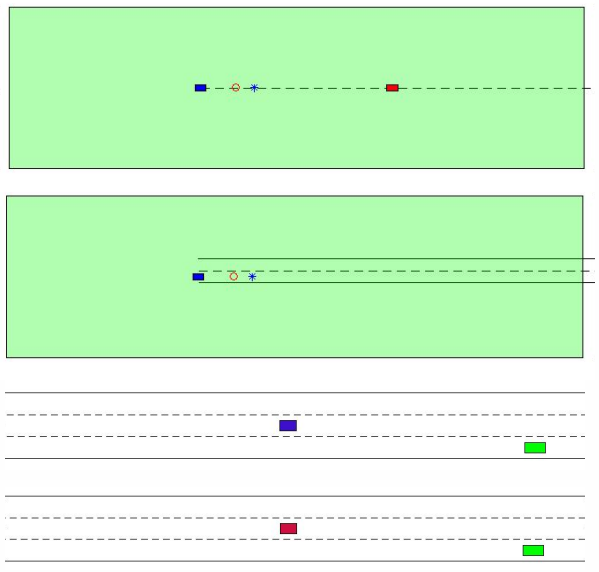


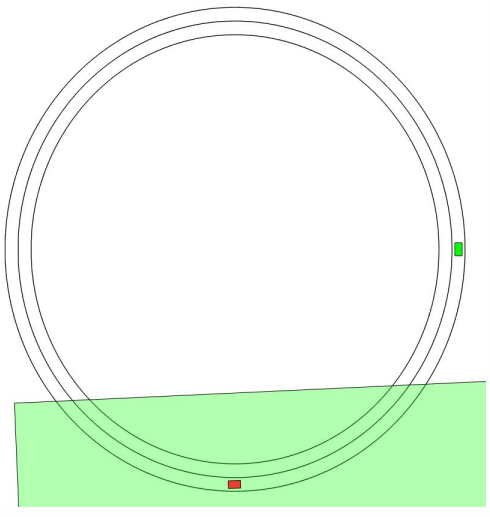
Most automotive OEMs are looking into MPC solutions today

MPC FOR AUTONOMOUS DRIVING

- Coordinate **torque request** and **steering** to achieve **safe and comfortable** autonomous driving with no collisions
- MPC combines **path planning**, **path tracking**, and **obstacle avoidance**
- **Stochastic prediction models** are used to account for uncertainty and driver's behavior

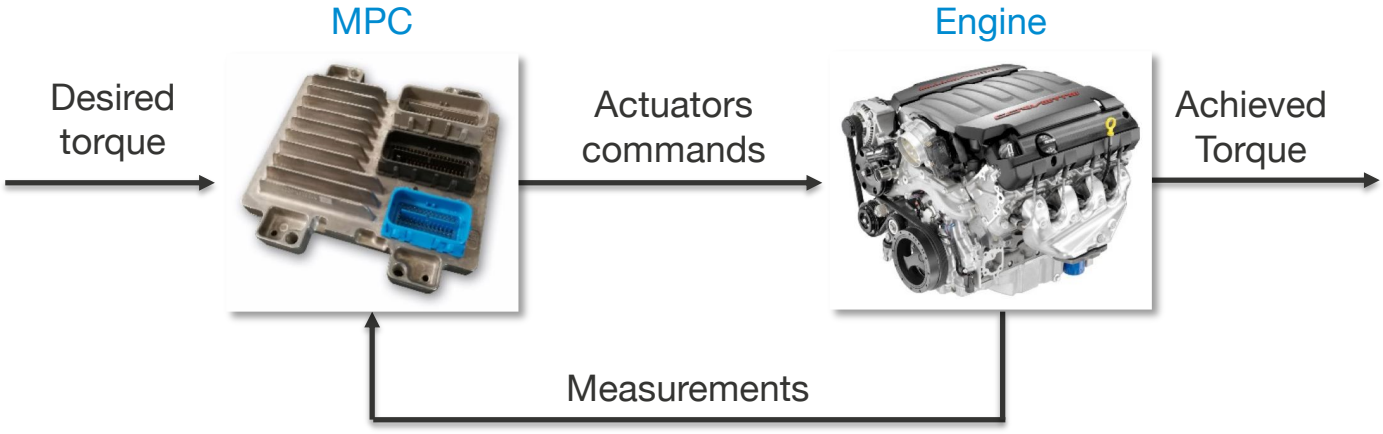






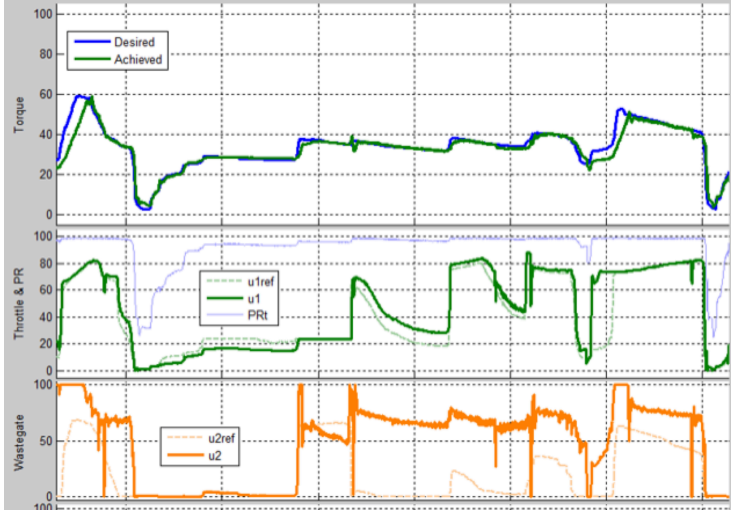
MPC OF GASOLINE TURBOCHARGED ENGINES

- 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

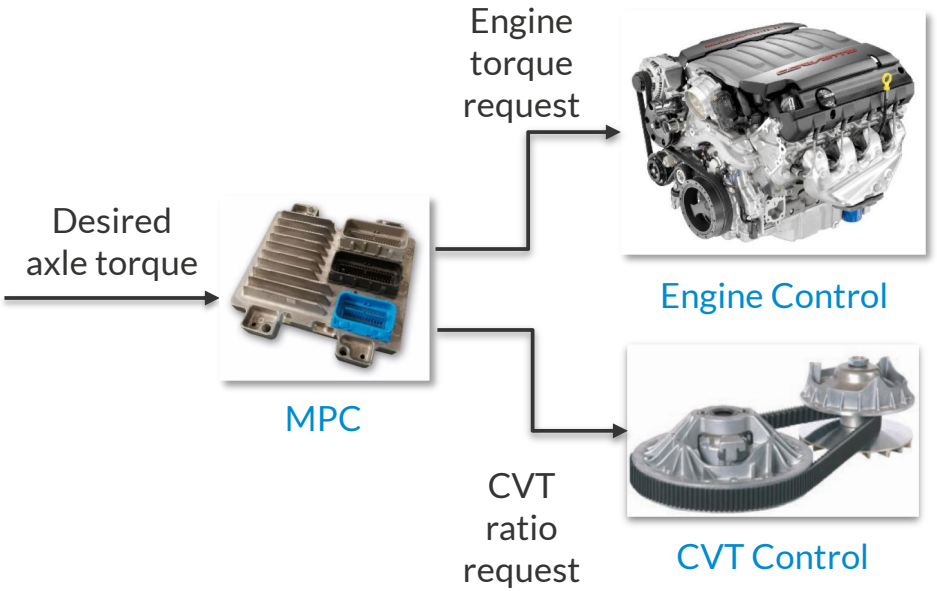
(Bemporad, Bernardini, Long, Verdejo, 2018)

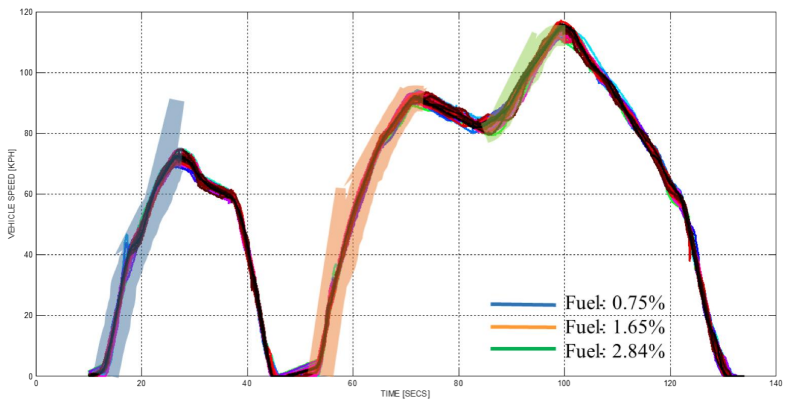


engine operating at low pressure (66 kPa)

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 IS IN AUTOMOTIVE PRODUCTION RIGHT NOW !

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)

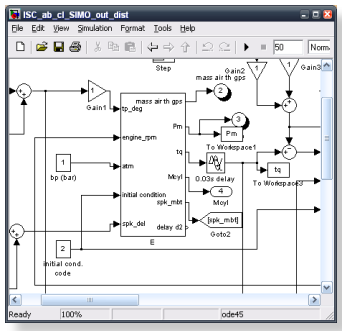
First known mass production of MPC in the automotive industry

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

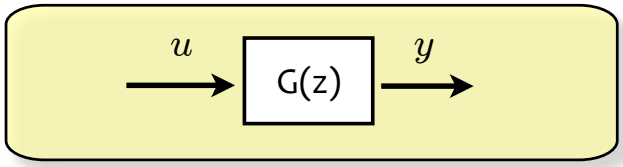
ODYS
Advanced Controls & Optimization

MPC DESIGN FLOW

High-fidelity simulation model



(simplified) control-oriented prediction model



performance index & constraints

MPC design

closed-loop simulation

physical modeling + parameter estimation

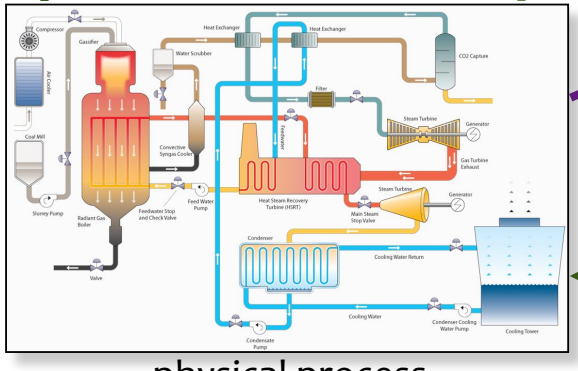
system identification

revise MPC setup

real-time code

```
/* z=A*x0; */
for (i=0;i<m;i++) {
    z[i]=A[i]*x[0];
    for (j=1;j<n;j++) {
        z[i]+=A[i+m*j]*x[j];
    }
}
```

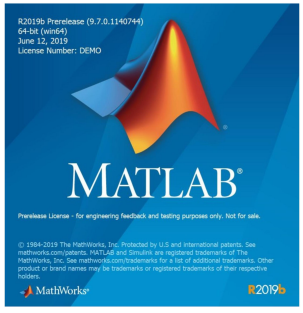
experiments



physical process

MPC TOOLBOXES

- **MPC Toolbox** (The Mathworks, Inc.): (Bemporad, Ricker, Morari, 1998-today)
  - Part of Mathworks' official toolbox distribution
  - All written in **MATLAB** code
  - Great for education and research



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

8000 downloads

1.5 downloads/day

- **ODYS NL-MPC Toolbox**: (ODYS Srl, 2013-present)
  - Very flexible MPC design and **seamless integration** in production
  - Real-time **MPC code** and **QP solver** written in **plain C**
  - Support for nonlinear models
  - Designed for **industrial production**

ODYS
Advanced Controls & Optimization

LINEAR MPC

LINEAR MPC - UNCONSTRAINED CASE

- Linear prediction model

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

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

Notation:

$x_0 = x(t)$

$x_k = x(t+k|t)$

$u_k = u(t+k|t)$

- Relation between input and states: $x_k = A^k x_0 + \sum_{j=0}^{k-1} A^j B u_{k-1-j}$
- Performance index

$$J(z, x_0) = x_N' P x_N + \sum_{k=0}^{N-1} x_k' Q x_k + u_k' R u_k$$

$$R = R' \succ 0 \qquad Q = Q' \succeq 0 \qquad P = P' \succeq 0$$

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

- **Goal**: find the sequence z^* that minimizes $J(z, x_0)$, i.e., that steers the state x to the origin optimally

COMPUTATION OF COST FUNCTION

$$J(z,x_0) = x_0'Qx_0 + \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{N-1} \\ x_N \end{bmatrix}' \overbrace{\begin{bmatrix} Q & 0 & 0 & \dots & 0 \\ 0 & Q & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & \dots & 0 & Q & 0 \\ 0 & 0 & \dots & 0 & P \end{bmatrix}}^{\bar{Q}} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{N-1} \\ x_N \end{bmatrix}$$

$$+ \begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_{N-1} \end{bmatrix}' \underbrace{\begin{bmatrix} R & 0 & \dots & 0 \\ 0 & R & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \dots & 0 & R \end{bmatrix}}_{\bar{R}} \begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_{N-1} \end{bmatrix}$$

$$\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix} = \underbrace{\begin{bmatrix} B & 0 & \dots & 0 \\ AB & B & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ A^{N-1}B & A^{N-2}B & \dots & B \end{bmatrix}}_{\bar{S}} \underbrace{\begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_{N-1} \end{bmatrix}}_{z} + \underbrace{\begin{bmatrix} A \\ A^2 \\ \vdots \\ A^N \end{bmatrix}}_{\bar{T}} x_0$$

$$J(z,x_0) = (\bar{S}z + \bar{T}x_0)'\bar{Q}(\bar{S}z + \bar{T}x_0) + z'\bar{R}z + x_0'Qx_0$$

$$= \frac{1}{2}z' \underbrace{2(\bar{R} + \bar{S}'\bar{Q}\bar{S})}_{H} z + x_0' \underbrace{2\bar{T}'\bar{Q}\bar{S}}_{F'} z + \frac{1}{2}x_0' \underbrace{2(Q + \bar{T}'\bar{Q}\bar{T})}_{Y} x_0$$

LINEAR MPC - UNCONSTRAINED CASE

$$J(z,x_0)=\frac{1}{2}z'Hz+x_0'F'z+\frac{1}{2}x_0'Yx_0 \qquad z=\begin{bmatrix}u_0\\u_1\\\vdots\\u_{N-1}\end{bmatrix}$$

condensed form of MPC

- The optimum is obtained by zeroing the gradient

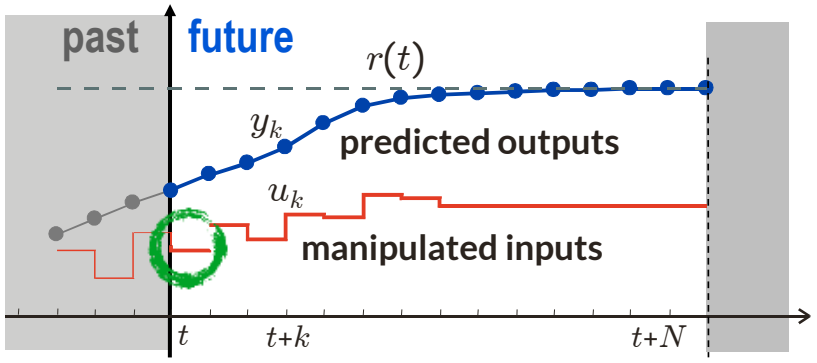
$$\nabla_z J(z,x_0)=Hz+Fx_0=0$$

and hence $z^*=\begin{bmatrix}u_0^*\\u_1^*\\\vdots\\u_{N-1}^*\end{bmatrix}=-H^{-1}Fx_0$ ("batch" solution)

- **Alternative #1**: find z^* via **dynamic programming** (Riccati iterations)
- **Alternative #2**: keep also $x_1,\dots,x_N$ as optimization variables and the equality constraints $x_{k+1}=Ax_k+Bu_k$ (**non-condensed form**)

UNCONSTRAINED LINEAR MPC ALGORITHM

@ each sampling step t:



- Minimize quadratic function, no constraints

$$\min_z \; f(z) = \frac{1}{2} z' H z + x'(t) F' z \qquad z = \begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_{N-1} \end{bmatrix}$$

- solution: $\nabla f(z) = Hz + Fx(t) = 0 \;\Rightarrow\; z^* = -H^{-1} F x(t)$

$$\Rightarrow \; u(t) = -\begin{bmatrix} I & 0 & \dots & 0 \end{bmatrix} H^{-1} F x(t) = K x(t)$$

Unconstrained linear MPC = linear state-feedback!

CONSTRAINED LINEAR MPC

- Linear prediction model: $\left\{ \begin{array}{rcl} x_{k+1} & = & Ax_k + Bu_k \\ y_k & = & Cx_k \end{array} \right.$ $\begin{array}{l} x \in \mathbb{R}^n \\ u \in \mathbb{R}^m \\ y \in \mathbb{R}^p \end{array}$

- Constraints to enforce:

$$\left\{ \begin{array}{l} u_{\min} \leq u(t) \leq u_{\max} \\ y_{\min} \leq y(t) \leq y_{\max} \end{array} \right.$$

- Constrained optimal control problem (quadratic performance index):

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

$$\begin{array}{rcll} R & = & R' \succ 0 & \\ Q & = & Q' \succeq 0 & z = \left[\begin{array}{c} u_0 \\ u_1 \\ \vdots \\ u_{N-1} \end{array} \right] \\ P & = & P' \succeq 0 & \end{array}$$

CONSTRAINED LINEAR MPC

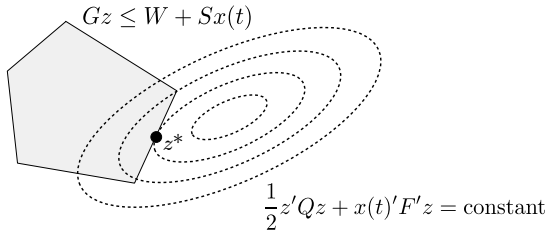
- Linear prediction model: $x_k = A^k x_0 + \sum_{i=0}^{k-1} A^i B u_{k-1-i}$
- Optimization problem (condensed form):

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

$$\text{s.t.} \quad Gz \le W + Sx_0 \quad \text{(linear constraints)}$$

convex Quadratic Program (QP)

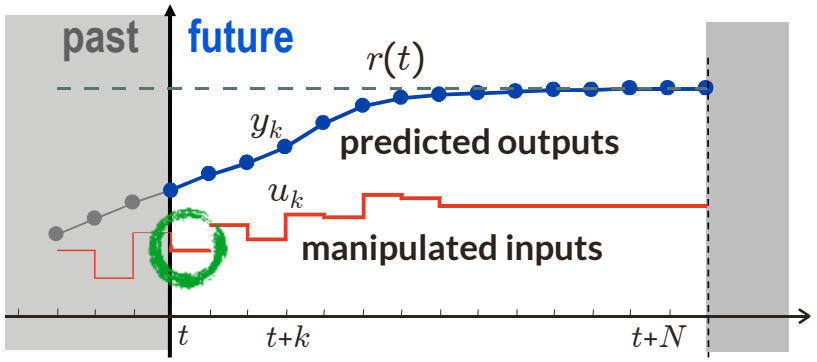
- $z = \begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_{N-1} \end{bmatrix} \in \mathbb{R}^{Nm}$ is the optimization vector



- $H = H' \succ 0$, 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

$$\begin{cases} \min\limits_{z} & \frac{1}{2} z' H z + \overbrace{x'(t) F' z}^{\text{feedback}} \\ \text{s.t.} & G z \le W + S \underbrace{x(t)}_{\text{feedback}} \end{cases}$$

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

LINEAR MPC - TRACKING

- Objective: make the output $y(t)$ track a reference signal $r(t)$
- Let us parameterize the problem using the **input increments**

$$\Delta u(t) = u(t) - u(t-1)$$

- As $u(t) = u(t-1) + \Delta u(t)$ we need to extend the system with a new state $x_u(t) = u(t-1)$

$$\begin{cases} x(t+1) &= Ax(t) + Bu(t-1) + B\Delta u(t) \\ x_u(t+1) &= x_u(t) + \Delta u(t) \end{cases}$$

$$\begin{cases} \begin{bmatrix} x(t+1) \\ x_u(t+1) \end{bmatrix} &= \begin{bmatrix} A & B \\ 0 & I \end{bmatrix} \begin{bmatrix} x(t) \\ x_u(t) \end{bmatrix} + \begin{bmatrix} B \\ I \end{bmatrix} \Delta u(t) \\ y(t) &= \begin{bmatrix} C & 0 \end{bmatrix} \begin{bmatrix} x(t) \\ x_u(t) \end{bmatrix} \end{cases}$$

- Again a linear system with states $x(t), x_u(t)$ and input $\Delta u(t)$

LINEAR MPC - TRACKING

- Optimal control problem (quadratic performance index):

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

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

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

$$\begin{aligned}
\min_{z} \quad & J(z, x(t)) = \tfrac{1}{2} z'Hz + [x'(t)\, r'(t)\, u'(t-1)]F'z \\
\text{s.t.} \quad & Gz \le W + S \begin{bmatrix} x(t) \\ r(t) \\ u(t-1) \end{bmatrix}
\end{aligned}$$

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

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

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

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

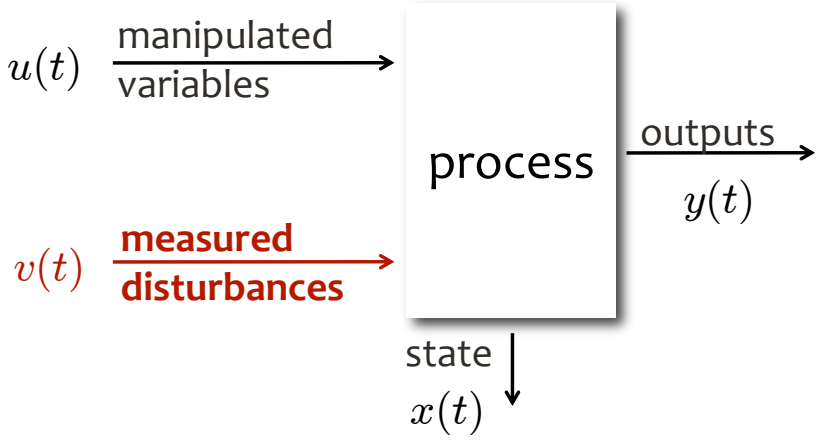
No pole in $z = 1$

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

MEASURED DISTURBANCES

- **Measured disturbance** $v(t)$ = input that is measured but not manipulated

$$\begin{cases} x_{k+1} &= Ax_k + Bu_k + B_v v(t) \\ y_k &= Cx_k + D_v v(t) \end{cases}$$



$$x_k = A^k x_0 + \sum_{j=0}^{k-1} A^j B u_{k-1-j} + A^j B_v v(t)$$

- Same performance index, same constraints. We still have a QP:

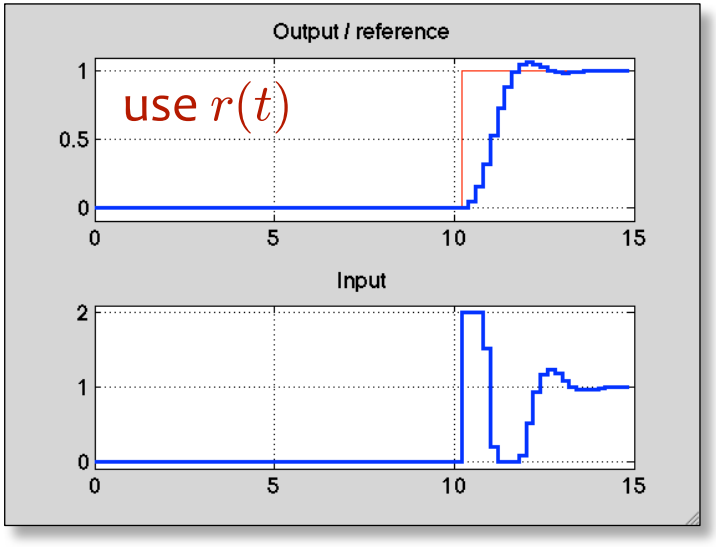
$$\begin{aligned} \min_z \quad & \tfrac{1}{2} z' H z + [x'(t)\, r'(t)\, u'(t-1)\, v'(t)] F' z \\ \text{s.t.} \quad & Gz \le W + S \begin{bmatrix} x(t) \\ r(t) \\ u(t-1) \\ v(t) \end{bmatrix} \end{aligned}$$

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

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

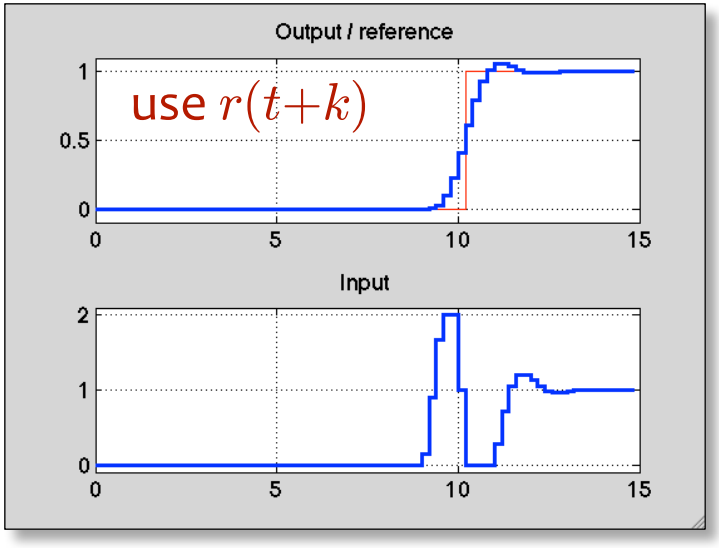
- Reference **not known** in advance (**causal**):

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



- 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**

SOFT CONSTRAINTS

- **Relax** output constraints to prevent QP infeasibility

$$\min_{z} \quad \sum_{k=0}^{N-1} \|W^y(y_{k+1} - r(t))\|_2^2 + \|W^{\Delta u}\Delta u_k\|_2^2 + \rho_\epsilon \epsilon^2$$

$$\text{subj. to} \quad x_{k+1} = Ax_k + Bu_k,\ k = 0, \ldots, N-1$$
$$u_{\min} \le u_k \le u_{\max},\ k = 0, \ldots, N-1$$
$$\Delta u_{\min} \le \Delta u_k \le \Delta u_{\max},\ k = 0, \ldots, N-1$$
$$y_{\min} - \epsilon V_{\min} \le y_k \le y_{\max} + \epsilon V_{\max},\ k = 1, \ldots, N$$

$$z = \begin{bmatrix} \Delta u(0) \\ \vdots \\ \Delta u(N-1) \\ \epsilon \end{bmatrix}$$

- ϵ = "panic" variable, with weight $\rho_\epsilon \gg W^y, W^{\Delta u}$
- $V_{\min}, V_{\max}$ = vectors with entries ≥ 0. The larger the i-th entry of vector V, the relatively softer the corresponding i-th constraint

- Infeasibility can be due to:
  - modeling errors
  - disturbances
  - wrong MPC setup (e.g., prediction horizon is too short)

MPC THEORY

- **After** the industrial success of MPC, a lot of research done:
  - **linear** MPC $\Rightarrow$ linear prediction model
  - **nonlinear** MPC $\Rightarrow$ nonlinear prediction model
  - **robust** MPC $\Rightarrow$ uncertain (linear) prediction model
  - **stochastic** MPC $\Rightarrow$ stochastic prediction model
  - **distributed/decentralized** MPC$\Rightarrow$ multiple MPCs cooperating together
  - **economic** MPC $\Rightarrow$ MPC based on arbitrary (economic) performance indices
  - **hybrid** MPC $\Rightarrow$ prediction model integrating logic and dynamics
  - **explicit** MPC $\Rightarrow$ off-line (exact/approximate) computation of MPC
  - **solvers** for MPC $\Rightarrow$ on-line numerical algorithms for solving MPC problems

- Main theoretical issues: **feasibility**, **stability**, solution **algorithms** (Mayne, 2014)

FEASIBILITY

$$\begin{aligned} \min_{z} \quad & \sum_{k=0}^{N-1} \|W^y(y_{k+1} - r(t))\|^2 + \|W^{\Delta u}\Delta u_k\|^2 \\ \text{subj. to} \quad & u_{\min} \le u_k \le u_{\max},\ k = 0, \dots, N-1 \\ & y_{\min} \le y_k \le y_{\max},\ k = 1, \dots, N \\ & \Delta u_{\min} \le \Delta u_k \le \Delta u_{\max},\ k = 0, \dots, N-1 \end{aligned}$$

QP problem

- **Feasibility**: Will the QP problem be feasible at all sampling instants t?
- **Input constraints** only: always feasible if $u/\Delta u$ constraints are consistent
- **Hard output constraints**:
  - When $N < \infty$ there is no guarantee that the QP problem will remain feasible at all t, even in the nominal case
  - **Maximum output admissible set** theory: $N < \infty$ is enough (Gutman, Ckwikel, 1987) (Gilbert, Tan, 1991) (Chmielewski, Manousiouthakis, 1996) (Kerrigan, Maciejowski, 2000)

BASIC CONVERGENCE PROPERTIES

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

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

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

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 $t \ge 0$, for all $R, Q \succ 0$.

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

CONVERGENCE PROOF

Proof: Main idea: use the **value function** $V^*(x(t))$ as a **Lyapunov-like function**

- Let z_t = optimal control sequence at time t, $z_t = [u_0^t \ \dots \ u_{N-1}^t]'$
- By construction $\bar{z}_{t+1} = [u_1^t \ \dots \ u_{N-1}^t \ 0]'$ is a feasible sequence at time $t+1$
- The cost of $\bar{z}_{t+1}$ is $V^*(x(t)) - x'(t)Qx(t) - u'(t)Ru(t) \geq V^*(x(t+1))$
- $V^*(x(t))$ is monotonically decreasing and ≥ 0, so $\exists \lim_{t\to\infty} V^*(x(t)) \triangleq V_\infty$
- Hence
  $0 \leq x'(t)Qx(t) + u'(t)Ru(t) \leq V^*(x(t)) - V^*(x(t+1)) \to 0$ for $t \to \infty$
- Since $R, Q \succ 0, \lim_{t\to\infty} x(t) = 0, \ \lim_{t\to\infty} u(t) = 0$ ■

Reaching the global optimum is not needed to prove convergence!

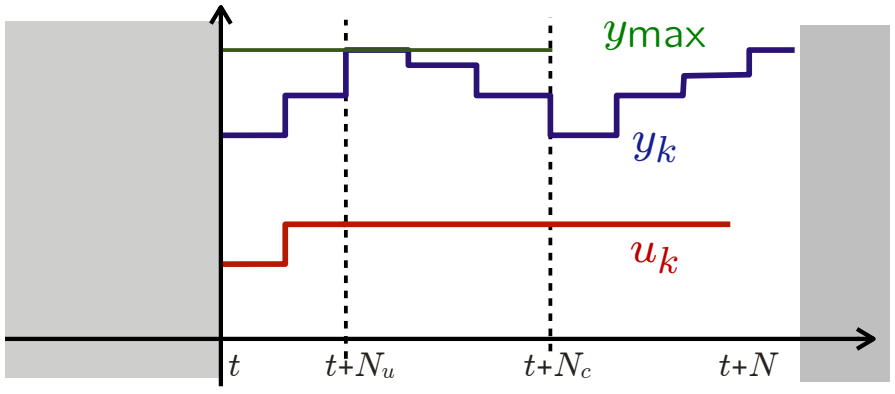
CONTROL AND CONSTRAINT HORIZONS

$$\min_{z} \quad \sum_{k=0}^{N-1} \|W^y(y_k - r(t))\|_2^2 + \|W^{\Delta u}\Delta u_k\|_2^2 + \rho_\epsilon \epsilon^2$$

$$\text{subj. to} \quad \begin{aligned} & u_{\min} \le u_k \le u_{\max},\ k = 0, \dots, N-1 \\ & \Delta u_{\min} \le \Delta u_k \le \Delta u_{\max},\ k = 0, \dots, N-1 \\ & \Delta u_k = 0,\ k = N_u, \dots, N-1 \\ & y_{\min} - \epsilon V_{\min} \le y_k \le y_{\max} + \epsilon V_{\max},\ k = 1, \dots, N_c \end{aligned}$$

- The **input horizon** N_u limits the number of free variables
  - Reduced performance
  - Reduced computation time

  typically $N_u = 1 \div 5$



- The **constraint horizon** N_c limits the number of constraints
  - Higher chance of violating output constraints
  - Reduced computation time

MPC AND LINEAR QUADRATIC REGULATION (LQR)

- Special case: $J(z,x_0) = x'_N P x_N + \sum_{k=0}^{N-1} x'_k Q x_k + u'_k R u_k$, $N_u = N$, with matrix P solving the Algebraic Riccati Equation

$$P = A'PA - A'PB(B'PB+R)^{-1}B'PA + Q$$

Jacopo Francesco Riccati

(1676–1754)

(unconstrained) **MPC = LQR** for any choice of the prediction horizon N

Proof: : Easily follows from Bellman's principle of optimality (dynamic programming): $x'_N P x_N$ = optimal "cost-to-go" from time N to ∞.

MPC AND CONSTRAINED LQR

- Consider again the constrained MPC law based on minimizing

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

- Choose matrix P and terminal gain K by solving the LQR problem

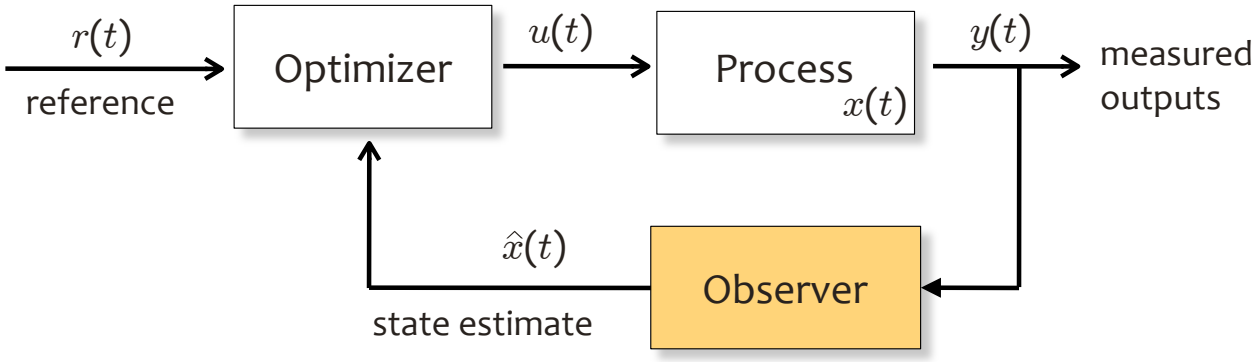
$$\begin{aligned}K &= -(R + B'PB)^{-1} B'PA \\ P &= (A + BK)'P(A + BK) + K'RK + Q\end{aligned}$$

- In a polyhedral region around the origin, **constrained MPC = constrained LQR** for any choice of the prediction and control horizons N, N_u

  (Sznaier, Damborg, 1987) (Chmielewski, Manousiouthakis, 1996) (Scokaert, Rawlings, 1998) (Bemporad, Morari, Dua Pistikopoulos, 2002)

- The larger the horizon N, the larger the region where MPC $\equiv$ constrained LQR

OBSERVER DESIGN FOR MPC



- Full state $x(t)$ of process may not be available, only outputs $y(t)$
- Even if $x(t)$ is available, noise should be filtered out
- Prediction and process models may be quite different
- The state $x(t)$ may not have any physical meaning
  (e.g., in case of model reduction or subspace identification)

We need to use a state observer

- Example: **Luenberger observer** $\hat{x}(t+1) = A\hat{x}(t) + Bu(t) + L(y(t) - C\hat{x}(t))$
- Most often, a **Kalman filter** based on the prediction model is used in practice

OUTPUT INTEGRATORS AND OFFSET-FREE TRACKING

- Add constant unknown disturbances on measured outputs:

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

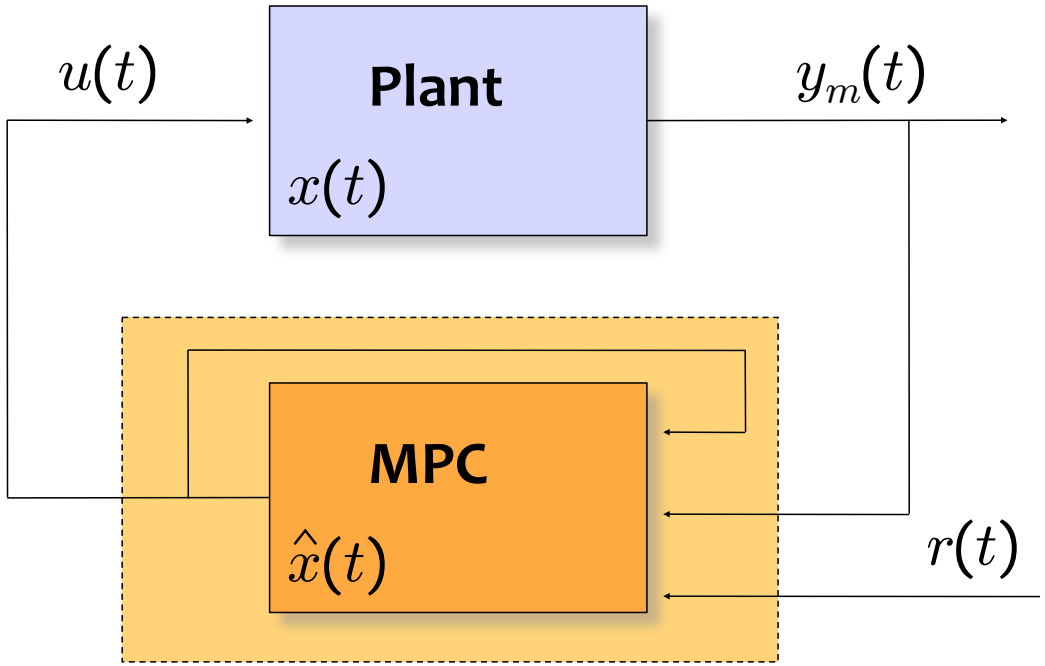
- Use the extended model to design a state observer (e.g., Kalman filter) that estimates both the state $\hat{x}(t)$ and disturbance $\hat{d}(t)$ from $y(t)$

- 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)

FREQUENCY ANALYSIS OF MPC (FOR SMALL SIGNALS)

- Unconstrained MPC gain + linear observer = linear dynamical system
- Closed-loop MPC analysis can be performed using standard frequency-domain tools (e.g., Bode plots for sensitivity analysis)




```
>> sys=ss(mpcobj)
>> sys=tf(mpcobj)
```

returns the LTI object of the MPC controller (when constraints are inactive)

CONTROLLER MATCHING

(Di Cairano, Bemporad, 2010)

- Given a desired linear controller $u = K_d x$, find a set of weights Q, R, P defining an MPC problem such that

$$-\begin{bmatrix} I & 0 & \dots & 0 \end{bmatrix} H^{-1} F = K_d$$

  i.e., the **MPC controller coincides with K_d** when the constraints are **inactive**

- Recall that the QP matrices are $H = 2(\bar{R} + \bar{S}'\bar{Q}\bar{S})$, $F = 2\bar{S}'\bar{Q}\bar{T}$, where

$$\bar{Q} = \begin{bmatrix} Q & 0 & 0 & \dots & 0 \\ 0 & Q & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & \dots & 0 & Q & 0 \\ 0 & 0 & \dots & 0 & P \end{bmatrix}, \; \bar{R} = \begin{bmatrix} R & 0 & \dots & 0 \\ 0 & R & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \dots & 0 & R \end{bmatrix}, \; \bar{S} = \begin{bmatrix} B & 0 & \dots & 0 \\ AB & B & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ A^{N-1}B & A^{N-2}B & \dots & B \end{bmatrix}, \; \bar{T} = \begin{bmatrix} A \\ A^2 \\ \vdots \\ A^N \end{bmatrix}$$

- The above inverse optimality problem can be cast as a convex problem

(Di Cairano, Bemporad, 2010)

CONTROLLER MATCHING - EXAMPLE

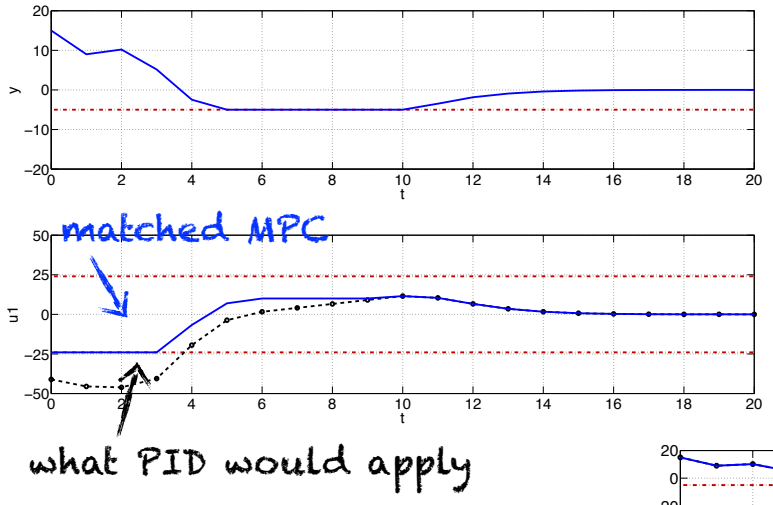
- Open-loop process: $y(t) = 1.8y(t-1) + 1.2y(t-2) + u(t-1)$
- Constraints: $-24 \leq u(t) \leq 24, y(t) \geq -5$
- Desired controller = PID with gains $K_I = 0.248, K_P = 0.752, K_D = 2.237$

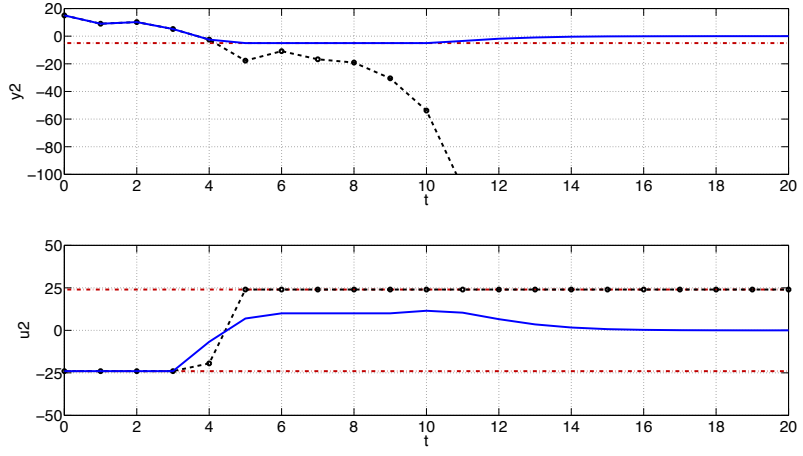
$$\begin{aligned} u(t) &= -\Big(K_I \mathcal{I}(t) + K_P y(t) + \tfrac{K_D}{T_s}(y(t) - y(t-1))\Big) \\ \mathcal{I}(t) &= \mathcal{I}(t-1) + T_s y(t) \end{aligned} \qquad x(t) = \begin{bmatrix} y(t-1) \\ y(t-2) \\ \mathcal{I}(t-1) \\ u(t-1) \end{bmatrix}$$

- Matching result (using inverse LQR):

$$Q^* = \begin{bmatrix} 6.401 & 0.064 & -0.001 & 0.020 \\ 0.064 & 6.605 & 0.006 & 0.080 \\ -0.001 & 0.006 & 6.647 & -0.020 \\ 0.019 & 0.080 & -0.020 & 6.378 \end{bmatrix}, \; R^* = 1, \; P^* = \begin{bmatrix} 422.7 & 241.7 & 50.39 & 201.4 \\ 241.7 & 151.0 & 32.13 & 120.4 \\ 50.39 & 32.13 & 19.85 & 26.75 \\ 201.4 & 120.4 & 26.75 & 106.6 \end{bmatrix}$$

CONTROLLER MATCHING - EXAMPLE





- Note: This is not trivially a saturation of a PID controller. In this case saturating the PID output leads to closed-loop instability!

A generalization of the inverse LQR design results is ongoing

(Zanon, Bemporad, in preparation)

LINEAR TIME-VARYING MODEL PREDICTIVE CONTROL

LPV MODELS

- **Linear Parameter-Varying (LPV)** model

$$\begin{cases} x_{k+1} &= A(p(t))x_k + B(p(t))u_k + B_v(p(t))v_k \\ y_k &= C(p(t))x_k + D_v(p(t))v_k \end{cases}$$

  that depends on a vector $p(t)$ of parameters

- The weights in the quadratic performance index can also be LPV
- The resulting optimization problem is still a QP

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

- The QP matrices must be constructed online, contrarily to the LTI case

LINEARIZING A NONLINEAR MODEL: LPV CASE

- An LPV model can be obtained by linearizing the **nonlinear model**

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

- $p_c \in \mathbb{R}^{n_p}$ = a vector of exogenous signals (e.g., ambient conditions)
- At time t, let $\bar{x}_c(t), \bar{u}_c(t), \bar{p}_c(t)$ be **nominal values**, that we assume constant in prediction, and linearize

$$\frac{d}{d\tau}(x_c(t+\tau) - \bar{x}_c(t)) = \frac{d}{d\tau}(x_c(t+\tau)) \simeq \underbrace{\left.\frac{\partial f}{\partial x}\right|_{\bar{x}_c(t),\bar{u}_c(t),\bar{p}_c(t)}}_{A_c(t)} (x_c(t+\tau) - \bar{x}_c(t)) +$$

$$\underbrace{\left.\frac{\partial f}{\partial u}\right|_{\bar{x}_c(t),\bar{u}_c(t),\bar{p}_c(t)}}_{B_c(t)} (u_c(t+\tau) - \bar{u}_c(t)) + \underbrace{f(\bar{x}_c(t), \bar{u}_c(t), \bar{p}_c(t))}_{B_{vc}(t)} \cdot 1$$

- Convert $(A_c, [B_c\ B_{vc}])$ to discrete-time and get prediction model $(A, [B\ B_v])$
- Same thing for the output equation to get matrices C and D_v

LTV MODELS

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

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

- At each time t the model can also change over the prediction horizon k
- The measured disturbance is embedded in the model
- The resulting optimization problem is still a QP

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

- As for LPV-MPC, the QP matrices must be constructed online

LINEARIZING A NONLINEAR MODEL: LTV CASE

- LPV/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

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

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

LINEARIZING A NONLINEAR MODEL: LTV CASE

- While integrating, also compute the **sensitivities**

$$
\begin{aligned}
A_k(t) &= \frac{\partial \bar{x}_c(t+(k+1)T_s)}{\partial \bar{x}_c(t+kT_s)} \\
B_k(t) &= \frac{\partial \bar{x}_c(t+(k+1)T_s)}{\partial \bar{u}_c(t+kT_s)} \\
C_k(t) &= \frac{\partial \bar{y}_c(t+kT_s)}{\partial \bar{x}_c(t+kT_s)}
\end{aligned}
$$

- Approximate the NL model as the LTV model

$$
\begin{cases}
\overbrace{x_c(k+1)-\bar{x}_c(k+1)}^{x_{k+1}} &= A_k(t)\overbrace{(x_c(k)-\bar{x}_c(k))}^{x_k}+B_k(t)\overbrace{(u_c(k)-\bar{u}_c(k))}^{u_k} \\
\underbrace{y_c(k)-\bar{y}_c(k)}_{y_k} &= C_k(t)\underbrace{(x_c(k)-\bar{x}_c(k))}_{x_k}
\end{cases}
$$

(the notation "(k)" is a shortcut for "$(t+kT_s)$")

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, we linearize the output equation and get

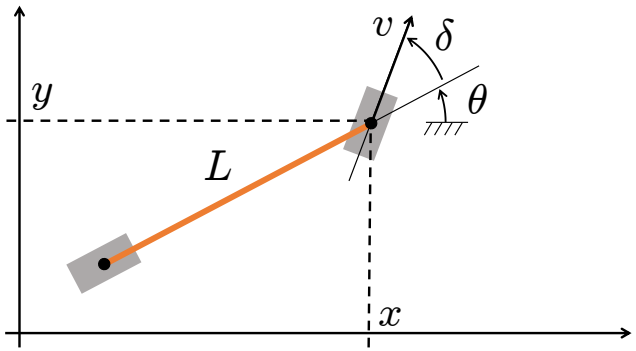
$$y = y_c - \bar{y}_c \approx \underbrace{\left.\frac{\partial g}{\partial x_c}\right|_{\bar{x}_c,\bar{u}_c,\bar{p}_c}}_{\text{Jacobian matrix } C} x$$

- The continuous-time linear system (A_c, B_c, C) can be converted to a discrete-time system (A, B, C) with sample time T_s

AUTONOMOUS DRIVING EXAMPLE

- **Goal**: Control **longitudinal acceleration** and **steering angle** of the vehicle simultaneously for **autonomous driving** with **obstacle avoidance**

- **Approach**: MPC based on a **bicycle-like kinematic model** of the vehicle in **Cartesian coordinates**



$$\begin{cases} \dot{x} &= v\cos(\theta+\delta) \\ \dot{y} &= v\sin(\theta+\delta) \\ \dot{\theta} &= \dfrac{v}{L}\sin(\delta) \end{cases}$$

(x, y)	Cartesian position of front wheel	v	velocity at front wheel
θ	vehicle orientation	δ	steering input
L	vehicle length = 4.5 m		

AUTONOMOUS DRIVING EXAMPLE

- Let $x_n, y_n, \theta_n, v_n, \delta_n$ nominal states/inputs satisfying

$$\begin{bmatrix} \dot{x}_n \\ \dot{y}_n \\ \dot{\theta}_n \end{bmatrix} = \begin{bmatrix} v_n \cos(\theta_n + \delta_n) \\ v_n \sin(\theta_n + \delta_n) \\ \frac{v_n}{L} \sin(\delta_n) \end{bmatrix}$$

feasible nominal trajectory

- Linearize the model around the nominal trajectory:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} \approx \begin{bmatrix} \dot{x}_n \\ \dot{y}_n \\ \dot{\theta}_n \end{bmatrix} + A_c \begin{bmatrix} x - x_n \\ y - y_n \\ \theta - \theta_n \end{bmatrix} + B_c \begin{bmatrix} v - v_n \\ \delta - \delta_n \end{bmatrix}$$

linearized model

where A_c, B_c are the **Jacobian matrices**

$$A_c = \begin{bmatrix} 0 & 0 & -v_n \sin(\theta_n + \delta_n) \\ 0 & 0 & v_n \cos(\theta_n + \delta_n) \\ 0 & 0 & 0 \end{bmatrix} \quad B_c = \begin{bmatrix} \cos(\theta_n + \delta_n) & -v_n \sin(\theta_n + \delta_n) \\ \sin(\theta_n + \delta_n) & v_n \cos(\theta_n + \delta_n) \\ \frac{1}{L} \sin(\delta_n) & \frac{v_n}{L} \cos(\delta_n) \end{bmatrix}$$

- Use first-order Euler method to discretize model:

$$A = I + T_s A_c, \quad B = T_s B_c, \quad T_s = 50\,\text{ms}$$

AUTONOMOUS DRIVING EXAMPLE

- Constraints on inputs and input variations $\Delta v_k = v_k - v_{k-1}, \Delta\delta_k = \delta_k - \delta_{k-1}$:

$$\begin{array}{lll} -20 \le v \le 70 & \text{km/h} & \text{velocity constraint} \\ -45 \le \delta \le 45 & \text{deg} & \text{steering angle} \\ -5 \le \Delta\delta \le 5 & \text{rad} & \text{steering angle rate} \end{array}$$

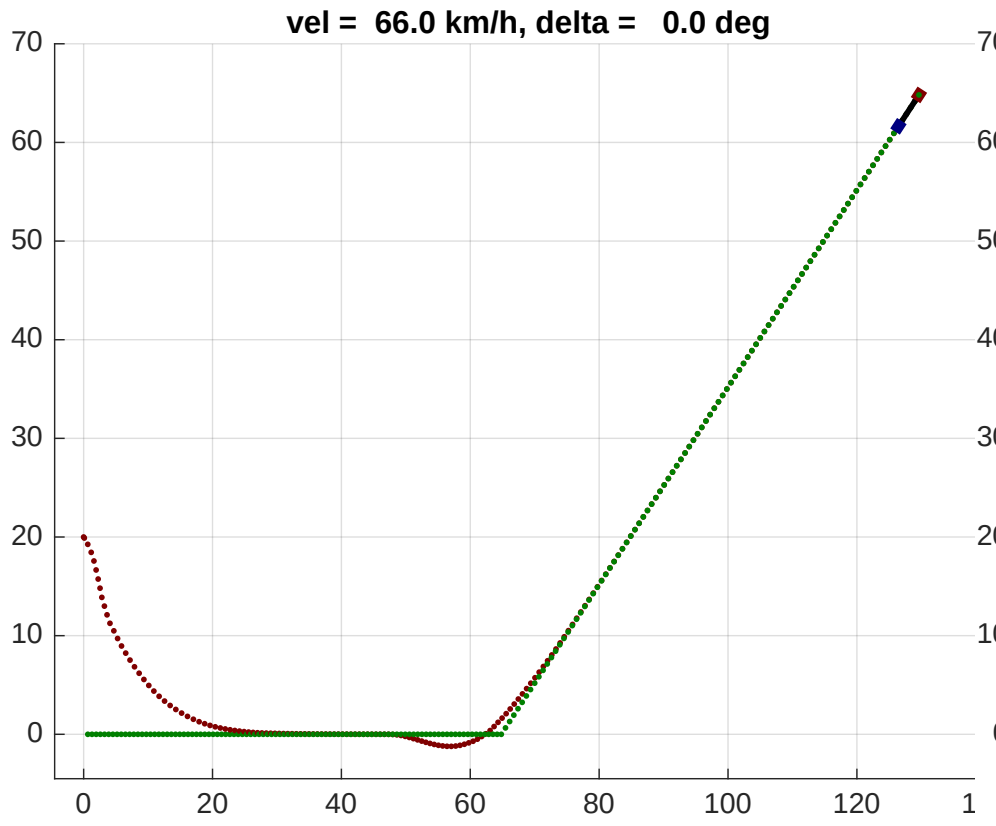
- Stage cost to minimize:

$$(x - x_{\text{ref}})^2 + (y - y_{\text{ref}})^2 + \Delta v^2 + \Delta\delta^2$$

- Prediction horizon: $N = 30$ (prediction distance = $NT_s v$, for example 25 m at 60 km/h)
- Control horizon: $N_u = 4$
- Preview on reference signals available

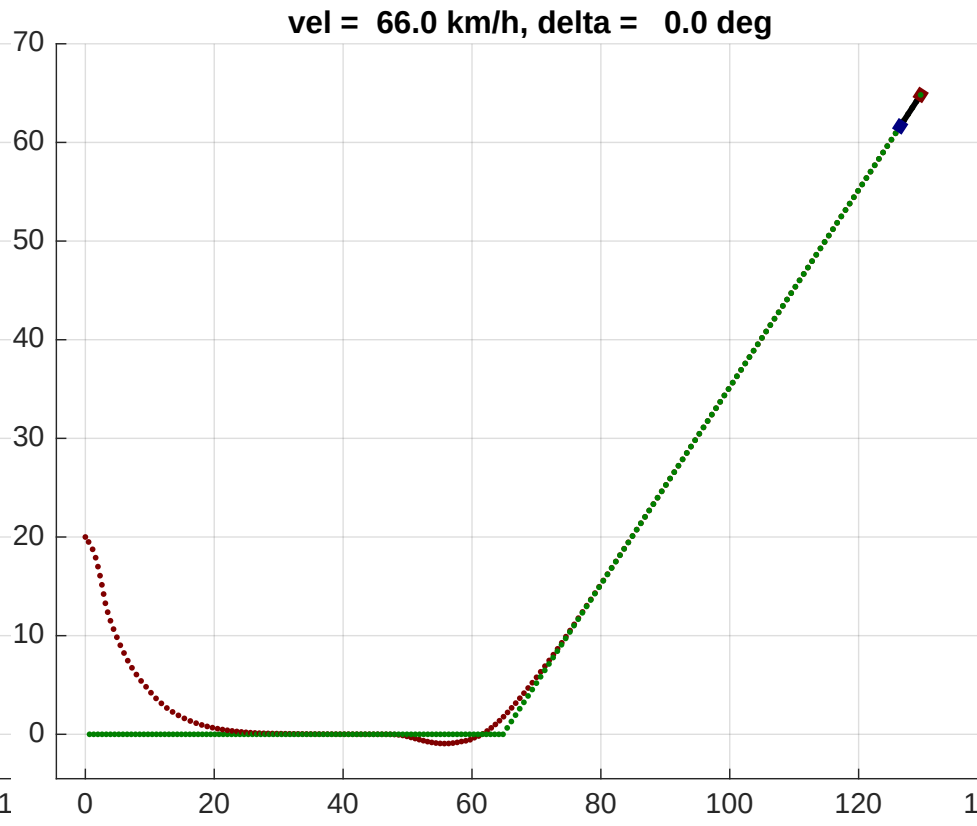
AUTONOMOUS DRIVING EXAMPLE

- Closed-loop simulation results



LPV-MPC

Model linearized @t

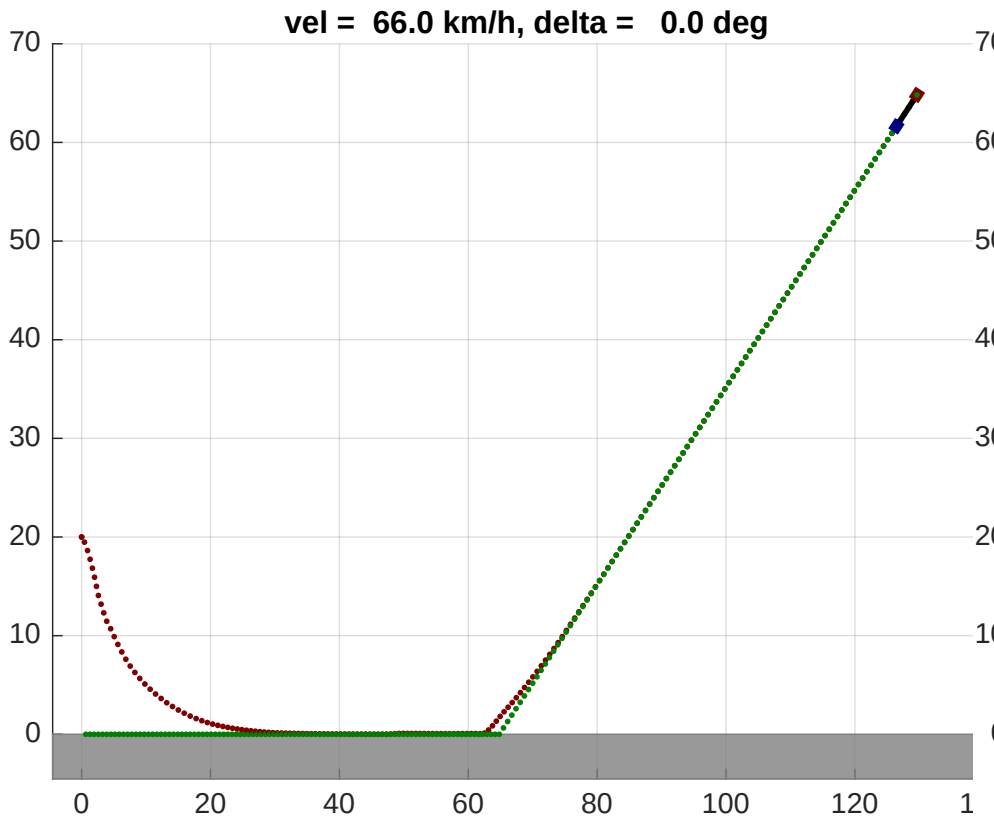


LTV-MPC

Model linearized @$t+k, k=0,\ldots,N-1$

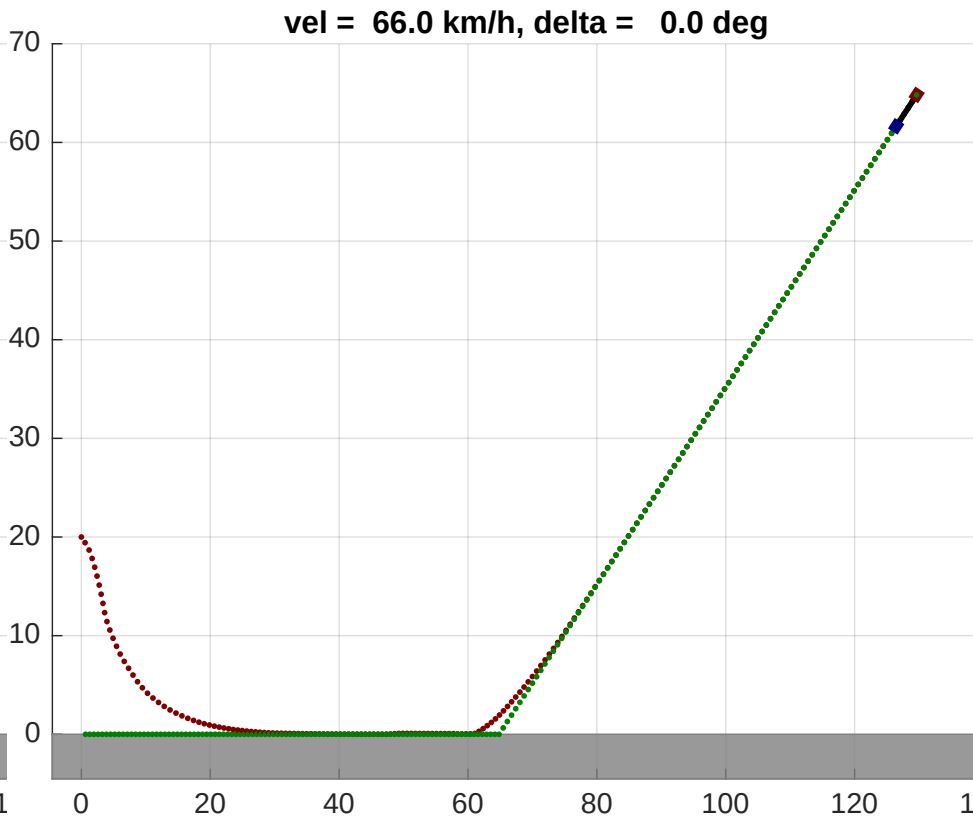
AUTONOMOUS DRIVING EXAMPLE

- Add position constraint $y \geq 0\,\mathrm{m}$



LPV-MPC

Model linearized @t



LTV-MPC

Model linearized @$t+k, k=0,\ldots,N-1$

FROM LTV-MPC TO NL-MPC

- **Key idea**: Solve a **sequence of LTV-MPC** problems at the same time t
- Given the current state $x(t)$ and reference $\{r(t+k), u_r(t+k)\}$, initial guess $U_0 = \{u_0^0, \ldots, u_{N-1}^0\}$ and corresponding state trajectory $X_0 = \{x_0^0, \ldots, x_N^0\}$
- A good initial guess U_0, X_0 is the previous (shifted) optimal solution
- At a generic iteration i, **linearize** the NL model around U_i, X_i:

$$\begin{cases} x_{k+1} &= f(x_k, u_k) \\ y_k &= g(x_k) \end{cases}$$

$$A_i = \frac{\partial f(x_0^i, u_0^i)}{\partial x},\ B_i = \frac{\partial f(x_0^i, u_0^i)}{\partial u},\ C_i = \frac{\partial g(x_0^i, u_0^i)}{\partial x}$$

NONLINEAR MPC

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

1. Simulate from $x(t)$ with inputs U_h, parameter sequence P and get state trajectory X_h and output trajectory Y_h
2. Linearize around (X_h, U_h, P) and discretize in time with sample time T_s
3. Let U^*_{h+1} be the solution of the QP problem corresponding to LTV-MPC
4. 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 method above is a **Gauss-Newton method** to solve the NL-MPC problem

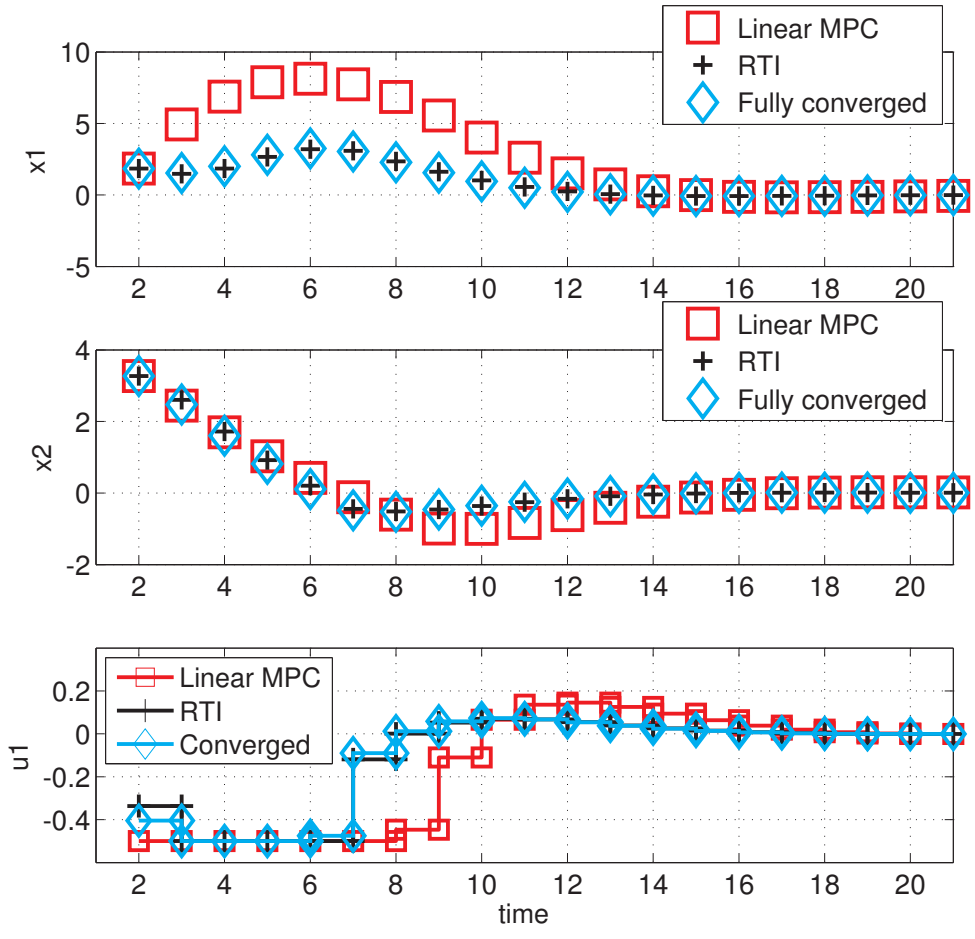
- Special case: just solve one iteration with $\alpha = 1$ (a.k.a. **Real-Time Iteration**)

  (Diehl, Bock, Schloder, Findeisen, Nagy, Allgower, 2002)

NONLINEAR MPC

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

- Example



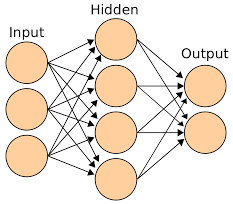
PREDICTION MODELS FOR MPC

- **Physics-based** nonlinear models

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

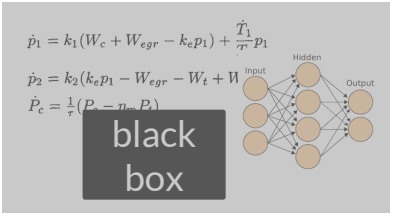
- Use **black-box system identification** algorithms to fit linear or nonlinear models to data

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



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

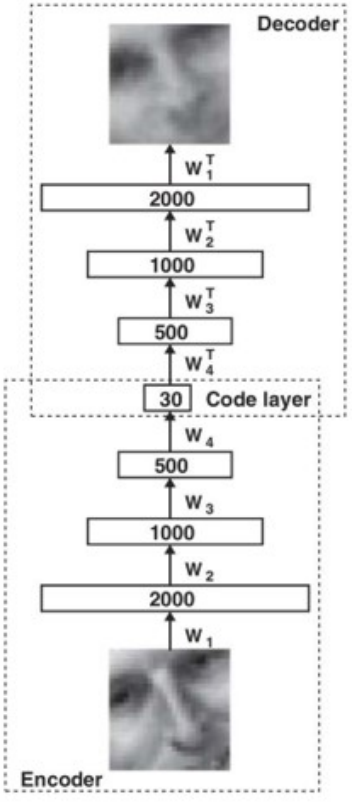
- **Note**: Computation complexity depends on chosen model, need to trade off **descriptiveness vs simplicity** of the model



LEARNING NONLINEAR MODELS FOR MPC

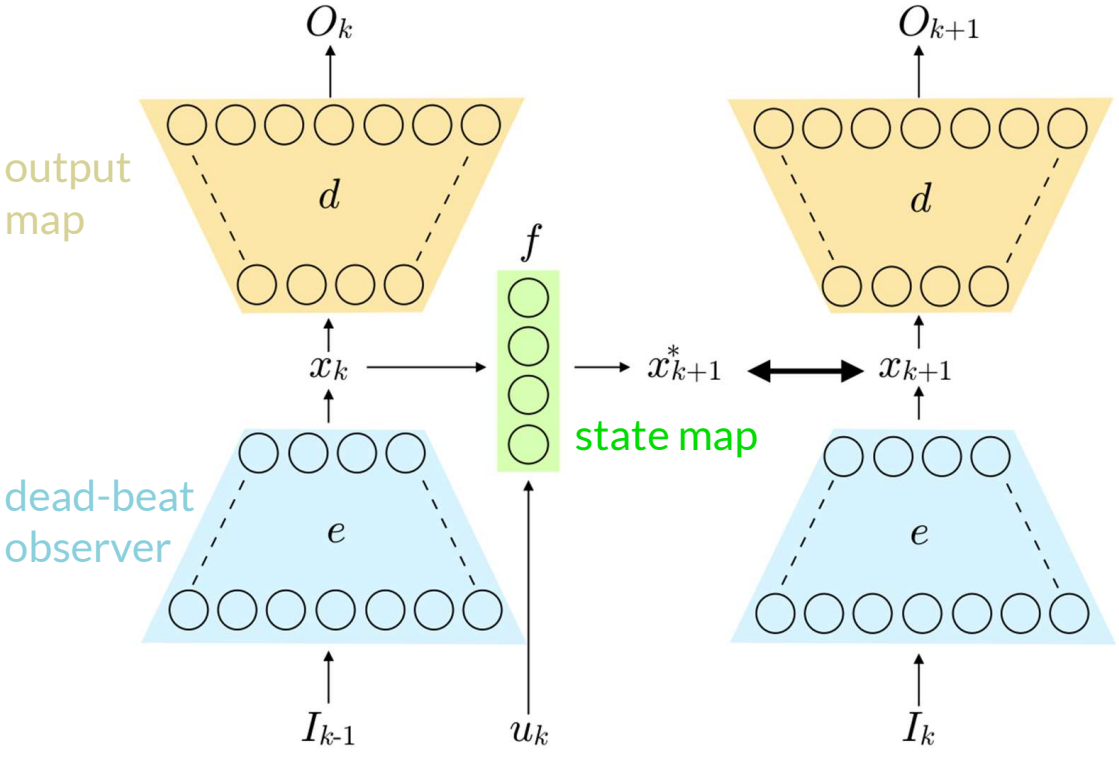
(Masti, Bemporad, 2018)

- **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)



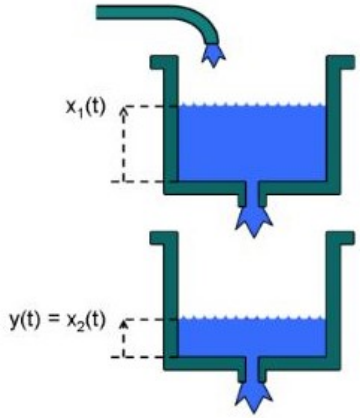
$$O_k = [y'_k \ \dots \ y'_{k-m}]'$$

$$I_k = [y'_k \ \dots \ y'_{k-n_a+1} \ u'_k \ \dots \ u'_{k-n_b+1}]'$$

LEARNING NONLINEAR MODELS FOR MPC - AN EXAMPLE

(Masti, Bemporad, 2018)

- System generating the data = nonlinear 2-tank benchmark

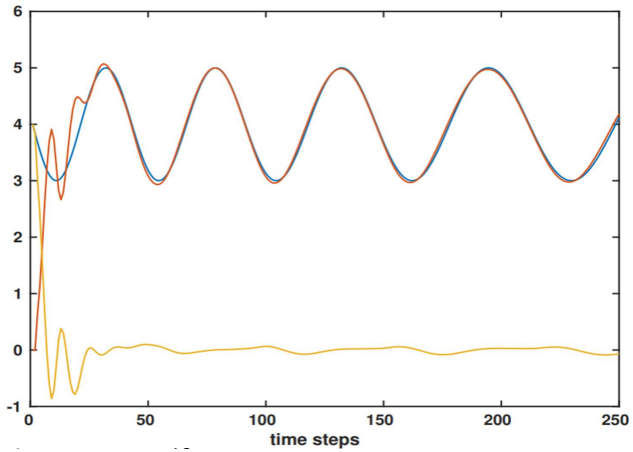


www.mathworks.com

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

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



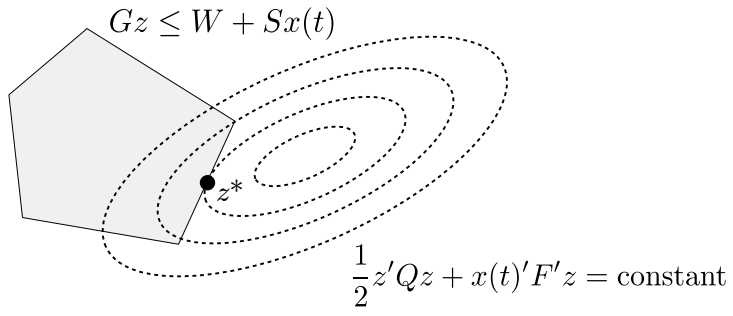
LTV-MPC results

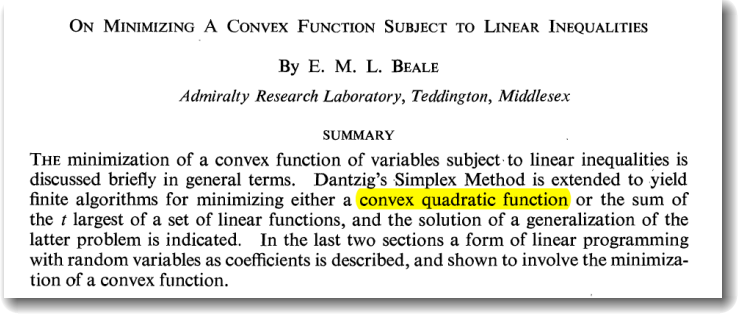
EMBEDDED QUADRATIC OPTIMIZATION FOR MPC

EMBEDDED MPC AND QUADRATIC PROGRAMMING

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

$$\begin{aligned} \min_{z} \quad & \frac{1}{2} z'Qz + x'(t)F'z \\ \text{s.t.} \quad & Gz \le W + Sx(t) \end{aligned} \qquad z = \begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_{N-1} \end{bmatrix}$$





ON MINIMIZING A CONVEX FUNCTION SUBJECT TO LINEAR INEQUALITIES

By E. M. L. BEALE

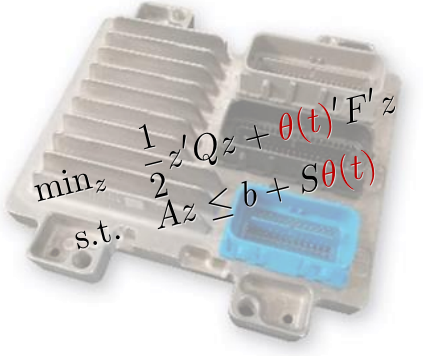
Admiralty Research Laboratory, Teddington, Middlesex

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)

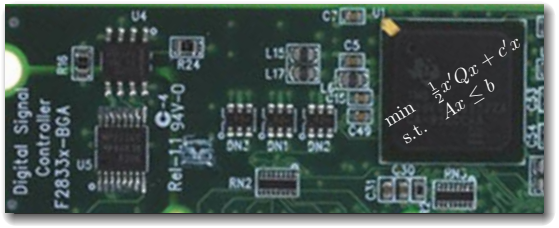
A rich set of good QP algorithms is available today



- 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

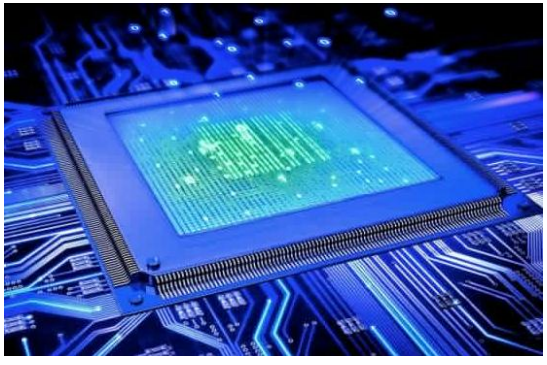
5. **code simple enough** to be validated/verified/certified
   (library-free C code, easy to check by production engineers)

EMBEDDED SOLVERS IN INDUSTRIAL PRODUCTION

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

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

and none of them should fail.

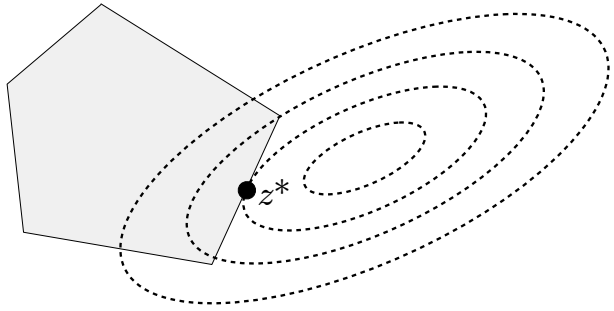


SOLUTION METHODS FOR QP

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

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

Quadratic Program (QP)



- A useful performance comparisons of many solvers can be found at `http://plato.la.asu.edu/bench.html`

- Hybrid toolbox:

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

KKT OPTIMALITY CONDITIONS FOR QP

- Quadratic programming problem

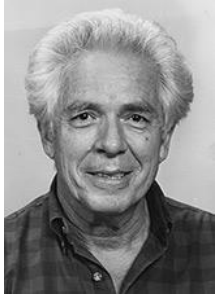
$$\begin{aligned} \min_z \quad & \frac{1}{2} z'Qz + x'F'z \\ \text{s.t.} \quad & Gz \le W + Sx \\ & Ez = f \end{aligned}$$

- Karush-Kuhn-Tucker (KKT) conditions:

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

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

William Karush
(1917–1997)



Harold W. Kuhn
(1925–2014)

Albert W. Tucker
(1905–1995)

DUAL GRADIENT PROJECTION FOR QP

- Consider the strictly convex QP and its dual

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

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

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

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

- 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**

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

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

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

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

  y0=y;


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

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

- Very **simple to code**
- Convergence rate: $f(x^k) - f(x^*) \le \dfrac{2L}{(k+2)^2}\|z_0 - z^*\|_2^2$

  (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

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

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

```
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)
(≈40 lines of C code)

ρu = dual vector

- 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:

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

- 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**

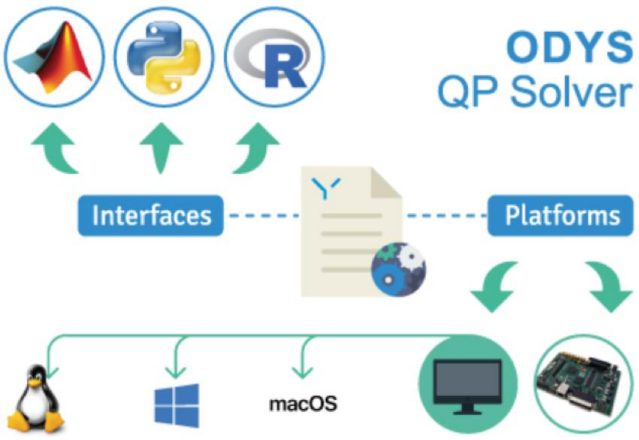
  (Python interface: ≈ 800,000 downloads)

`http://osqp.org`

ODYS QP SOLVER

- QP solver designed for **industrial production**, MPC-specific version available

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



- Implements a **proprietary** state-of-the-art method for QP
- Emphasis on **robustness** (especially in **single precision**), **speed** of execution, **low memory** requirements
- Written in **ANSI-C**, compliant with **MISRA-C 2012** standards
- Currently used for production by some major OEMs

http://odys.it/qp

CAN WE SOLVE QP'S USING LEAST SQUARES ?

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

Adrien-Marie Legendre

(1752–1833)

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

Carl Friedrich Gauss

(1777–1855)

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

Nonnegative Least Squares (NNLS)

(Lawson, Hanson, 1974)

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

(750 chars in Embedded MATLAB)

Bounded-Variable Least Squares (BVLS)

(Stark,Parker, 1995)

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

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)

$$\begin{aligned}\min_z \quad & \tfrac{1}{2} z'Qz + c'z \\ \text{s.t.} \quad & Gz \le g\end{aligned}$$

$$Q = Q' \succ 0$$

$$Q = L'L$$

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

$$\begin{aligned}\min_u \quad & \tfrac{1}{2}\|u\|^2 \\ \text{s.t.} \quad & Mu \le d\end{aligned}$$

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

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

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

- 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

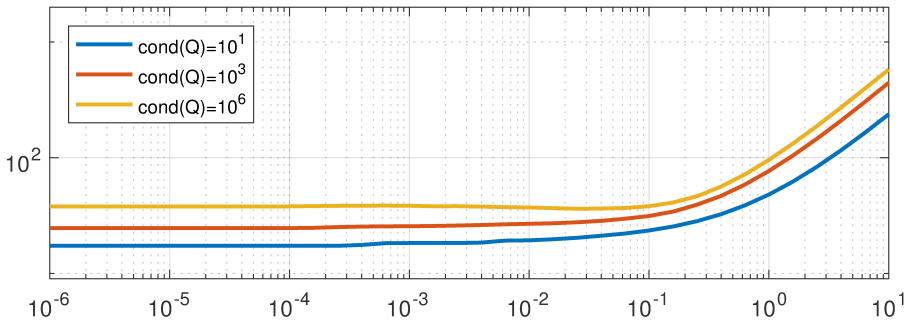
(Bemporad, 2018)

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

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

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

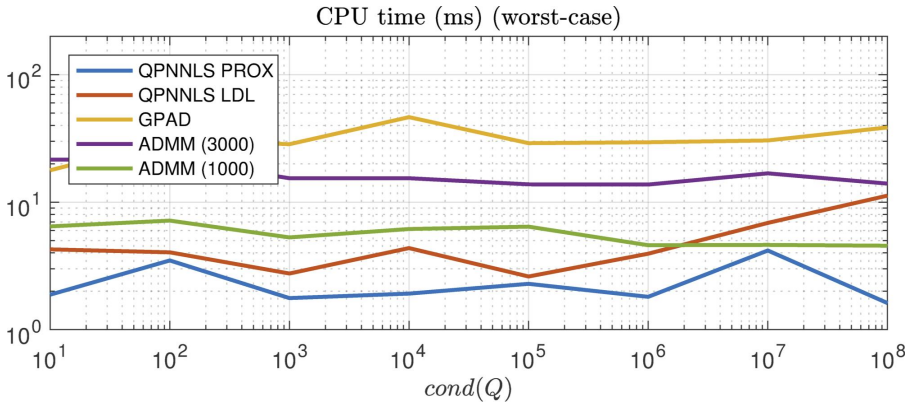
total number of active-set iterations as a function of ϵ

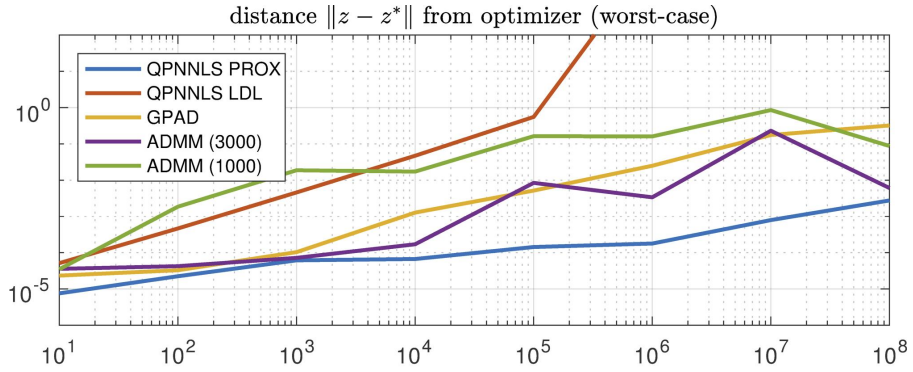


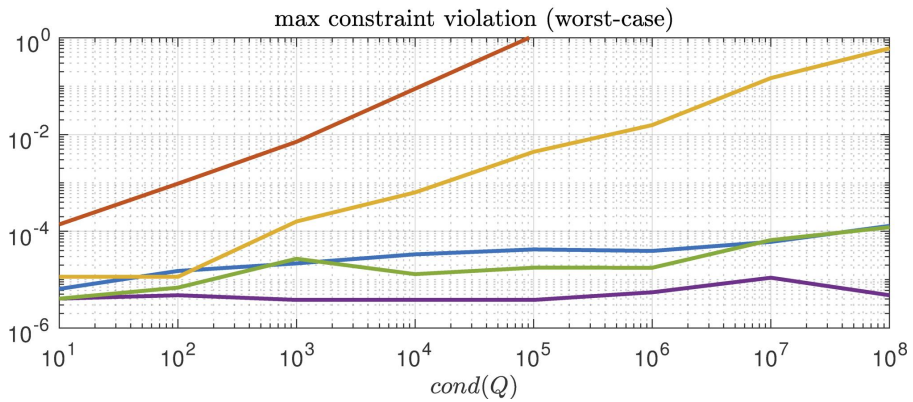
- Each QP is heavily warm-started and makes very few active-set changes
- Recursive LDL^T decompositions/rank-1 updates exploited for max efficiency

SOLVING QP'S VIA NNLS AND PROXIMAL POINT ITERATIONS

(Bemporad, 2018)







single precision arithmetic

30 vars, 100 constraints

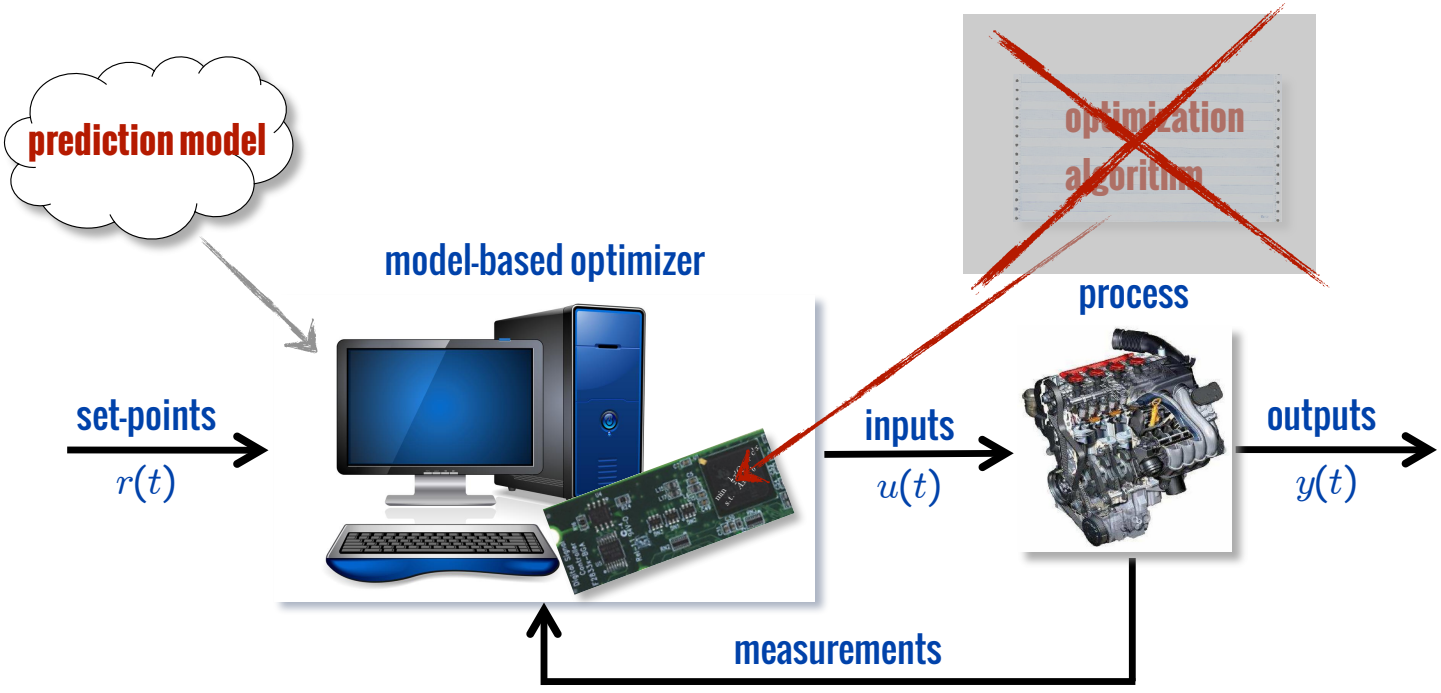
(Macbook Pro 3 GHz Intel Core i7)

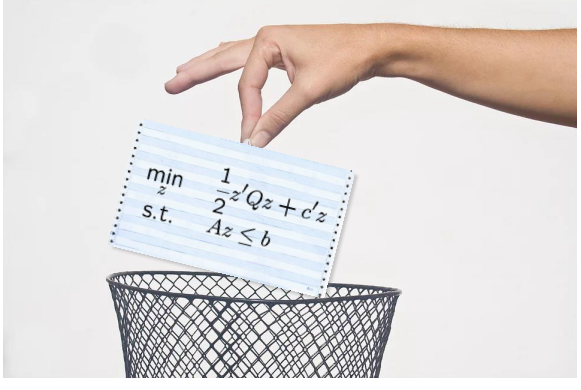
ACTIVE-SET VS INTERIOR-POINT METHODS

- IP needs **advanced linear algebra** operations during iterations, AS does not
- Linear systems to solve in IP tend to become ill-conditioned at convergence
- IP provides good solutions within 10-15 IP iterations (**usually** ...).
  AS iterations increase when both vars and constraints increase
- IP faster than AS in QPs with say >500 vars & constraints. IP well exploits sparse linear algebra
- Number of AS iterations can be **exactly certified** for given QP matrices (see later ...)

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

MPC WITHOUT ON-LINE QP





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

YES !

ON-LINE VS OFF-LINE OPTIMIZATION

$$\begin{aligned} \min_{z} \quad & \frac{1}{2} z'Qz + x'(t)F'z + \frac{1}{2} x'(t)Yx(t) \\ \text{s.t.} \quad & Gz \leq W + Sx(t) \end{aligned}$$

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

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

→ **Quadratic Programming (QP)**

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

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

PROPERTIES OF MPQP SOLUTION

(Bemporad, Morari, Dua, Pistikopoulos, 2002)

THEOREM

Assume $H \succ 0$, $\left[\begin{smallmatrix} H & F \\ F' & Y \end{smallmatrix}\right] \succeq 0$. *Then*

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

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

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

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

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

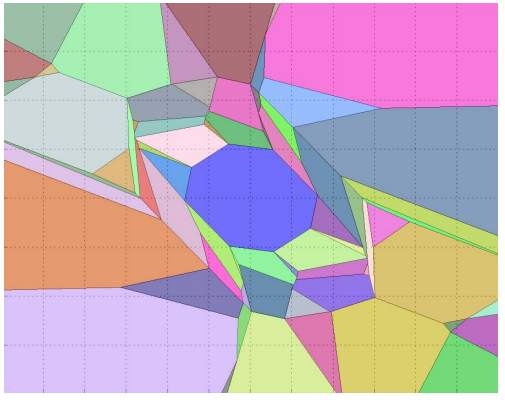
$$\begin{aligned} V^*(x) = \tfrac{1}{2} x'Qx + \min_z \quad & \tfrac{1}{2} x'Hz + x'F'z \\ \text{s.t.} \quad & Gz \le W + Sx \end{aligned}$$

EXPLICIT MPC SOLUTION

(Bemporad, Morari, Dua, Pistikopoulos, 2002)

- **Corollary**: since the multiparametric solution

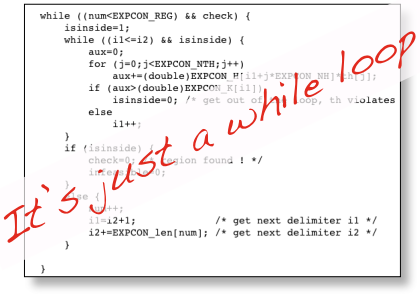
$$z^*(x) = \arg\min_z \quad \tfrac{1}{2} z'Qz + x'F'z$$
$$\text{s.t.} \quad Gz \le W + Sx$$



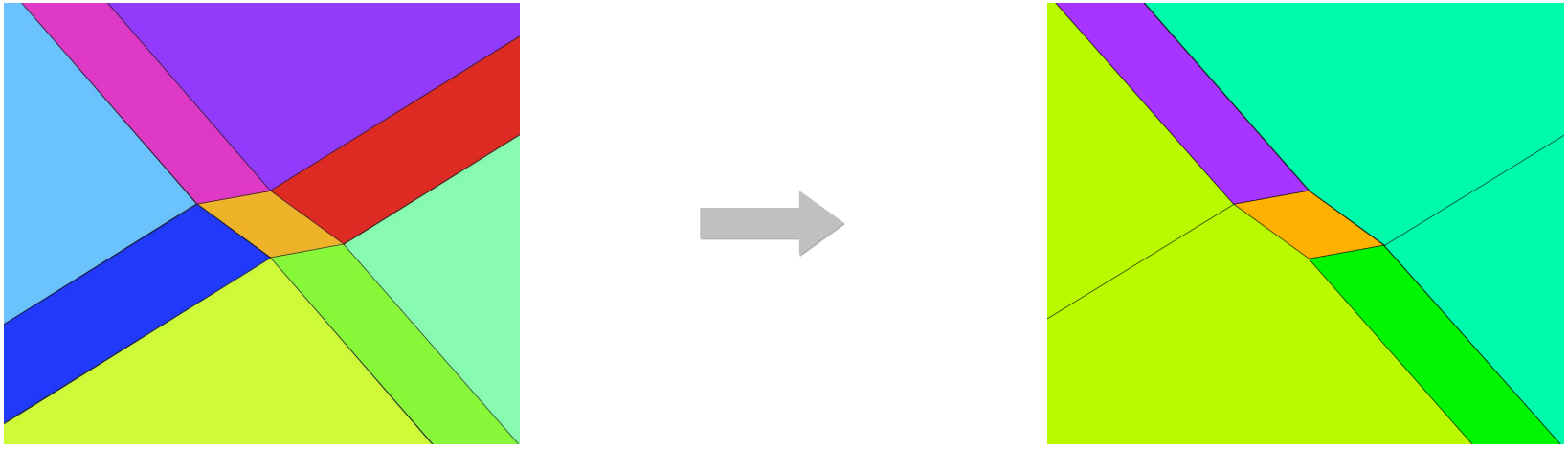
of a strictly convex QP is **continuous** and **piecewise affine**, the **linear MPC law is continuous & piecewise affine** too

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

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



COMPLEXITY REDUCTION

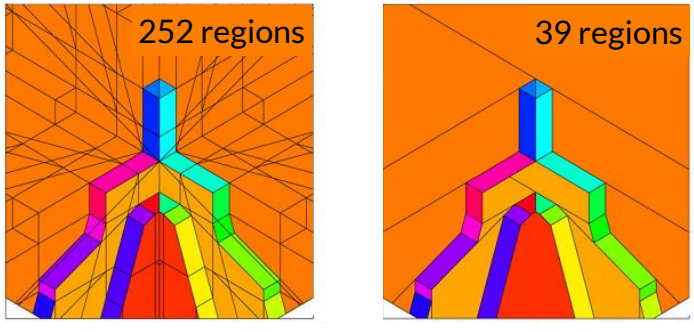


- We are interested only in the first components of the optimizer z^*

$$z(x) \triangleq \begin{bmatrix} \mathbf{u_0'(x)} & u_1'(x) & \dots & u_{N-1}'(x) \end{bmatrix}$$

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

  (Geyer, Torrisi, Morari, 2008)

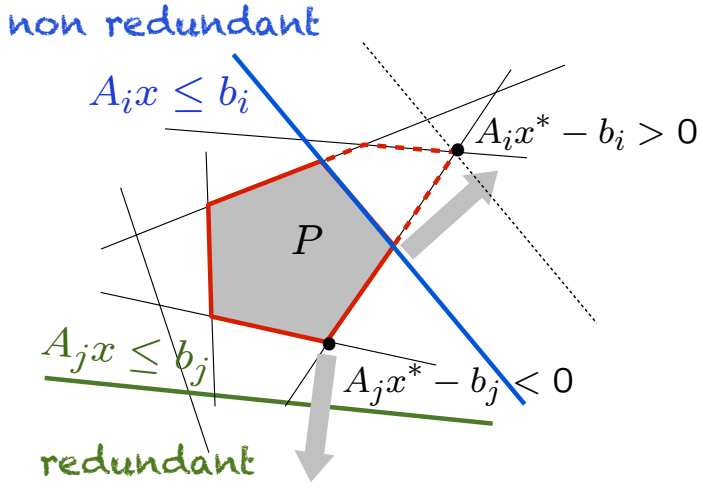


REMOVING REDUNDANT INEQUALITIES VIA LP

- A variety of multiparametric quadratic programming solvers is available

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

- Most computations are spent in removing redundant inequalities



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

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

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

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

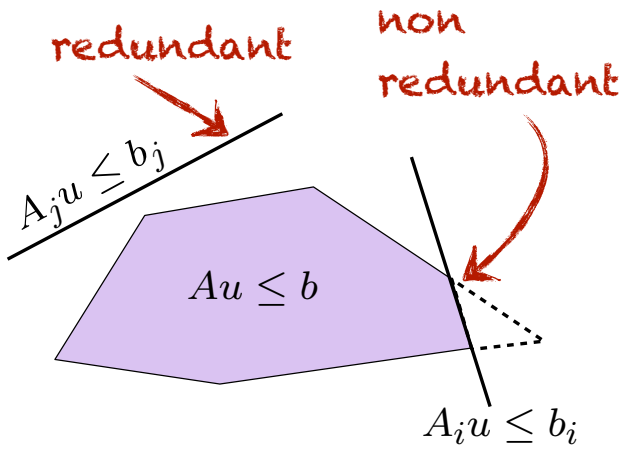
REMOVING REDUNDANT INEQUALITIES VIA NNLS

(Bemporad, 2015)

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

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

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

- **Note**: very high precision typically required in computational geometry, so active-set methods are preferable to first-order and interior-point methods

MULTIPARAMETRIC QP BASED ON NNLS

(Bemporad, 2015)

- Other polyhedral operations can be also solve by NNLS (check full dimension, Chebychev radius, union, projection)
- New mpQP algorithm based on NNLS + dual QP formulation to compute active sets and deal with QP degeneracy
- Comparable performance with other existing methods:
  - **Hybrid Toolbox** (Bemporad, 2003)
  - **MPT Toolbox 2.6** (w/ default opts)

    (Kvasnica, Grieder, Baotic, 2004)

    (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:

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

- Objective:

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

$$u_k = Kx_k,\ \forall k \ge N_u,\ K = \text{LQR gain}$$

$$N_u = N = 2$$

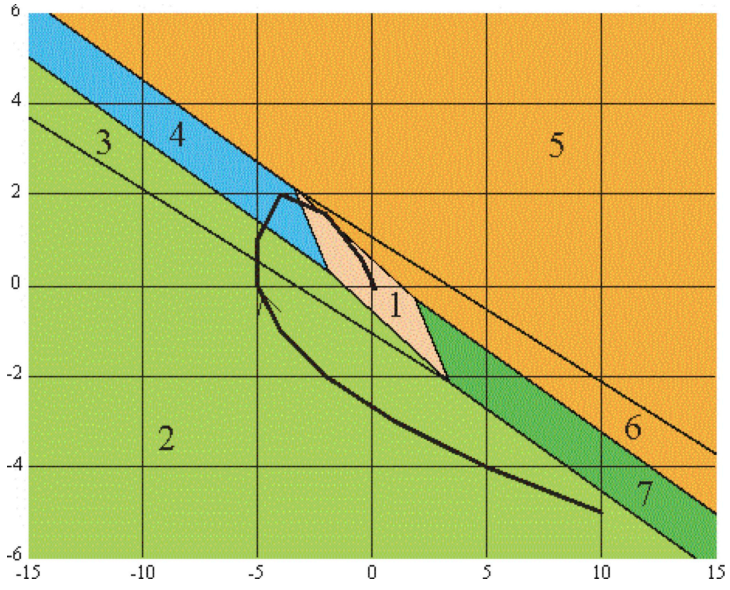
$$\left(\sum_{k=0}^{1} y_k^2 + \frac{1}{100} u_k^2\right) + x_2' \underbrace{\begin{bmatrix} 2.1429 & 1.2246 \\ 1.2246 & 1.3996 \end{bmatrix}}_{\text{solution of algebraic Riccati equation}} x_2$$

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

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

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

DOUBLE INTEGRATOR EXAMPLE - EXPLICIT SOLUTION

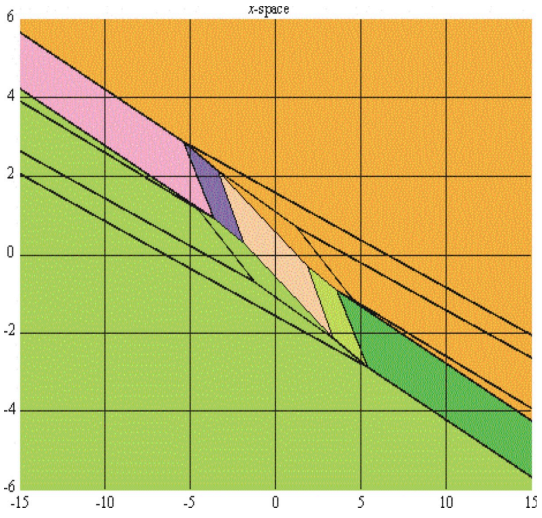


$$N_u = 2$$

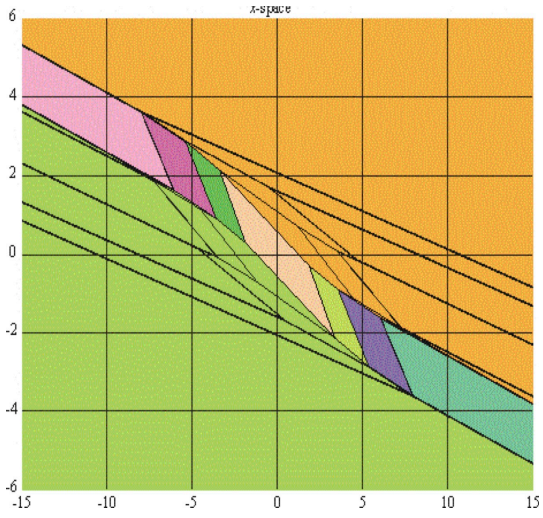
$$u(x) = \begin{cases} \begin{bmatrix} -0.8166 & -1.75 \end{bmatrix} x & \text{if } \begin{bmatrix} -0.8166 & -1.75 \\ 0.8166 & 1.75 \\ 0.6124 & 0.4957 \\ -0.6124 & -0.4957 \end{bmatrix} x \le \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} & \text{(Region \#1)} \\ 1 & \text{if } \begin{bmatrix} 0.3864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

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

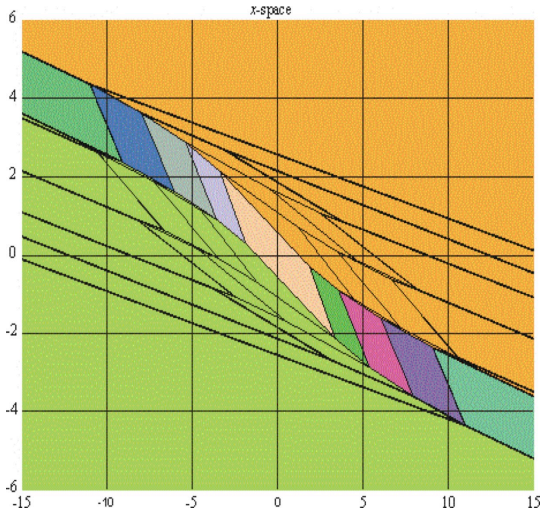
DOUBLE INTEGRATOR EXAMPLE - EXPLICIT SOLUTION



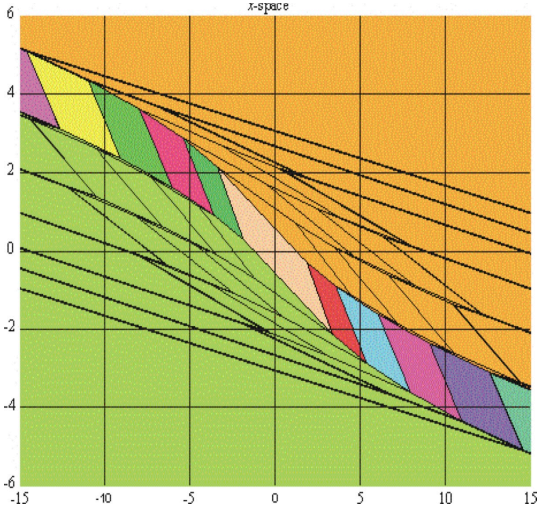
$N_u = 3$



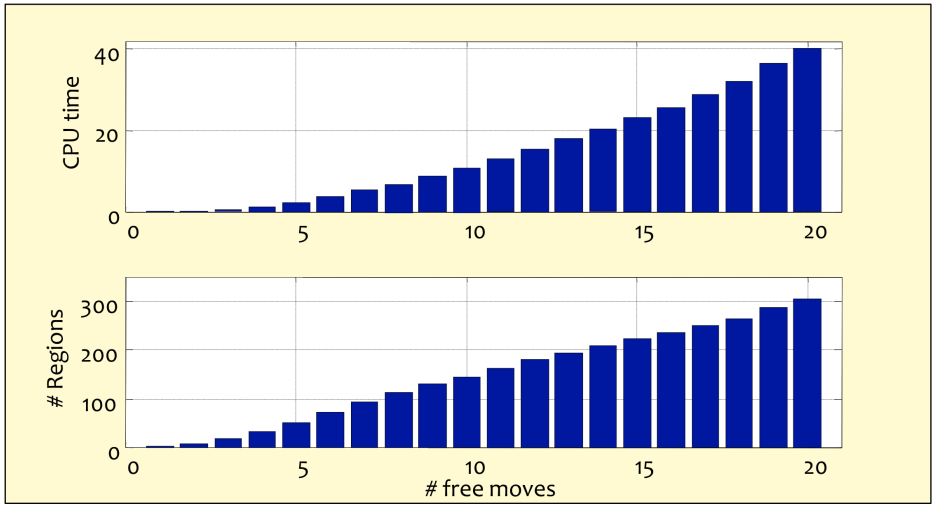
$N_u = 4$



$N_u = 5$



$N_u = 6$



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

APPLICABILITY OF EXPLICIT MPC

- Consider the following general MPC formulation

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

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

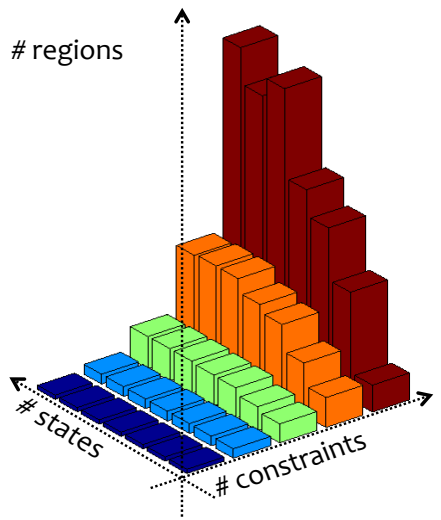
COMPLEXITY OF MULTIPARAMETRIC SOLUTIONS

- 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)

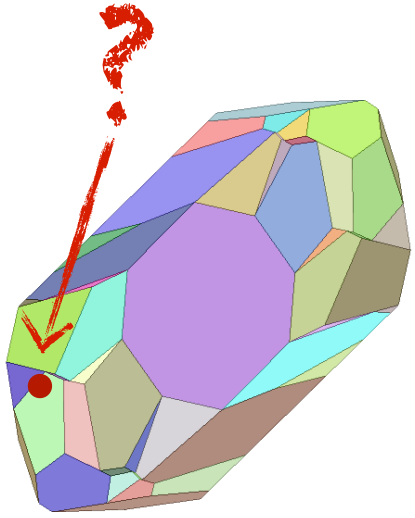
states/horizon	$N=1$	$N=2$	$N=3$	$N=4$	$N=5$
n=2	3	6.7	13.5	21.4	19.3
n=3	3	6.9	17	37.3	77
n=4	3	7	21.65	56	114.2
n=5	3	7	22	61.5	132.7
n=6	3	7	23.1	71.2	196.3
n=7	3	6.95	23.2	71.4	182.3
n=8	3	7	23	70.2	207.9

average on 20 random SISO systems w/ input saturation



COMPLEXITY OF MULTIPARAMETRIC SOLUTIONS

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



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

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

COMPLEXITY CERTIFICATION FOR ACTIVE-SET QP SOLVERS

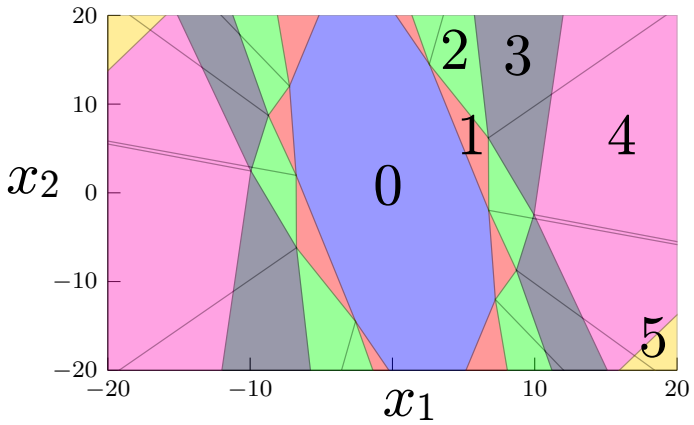
(Cimini, Bemporad, IEEE TAC, 2017)

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

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

- 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 !

COMPLEXITY CERTIFICATION FOR ACTIVE-SET QP SOLVERS

(Cimini, Bemporad, IEEE TAC, 2017)

- Examples (from MPC Toolbox):

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

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

QP certification algorithm currently used in production

ODYS Advanced Controls & Optimization

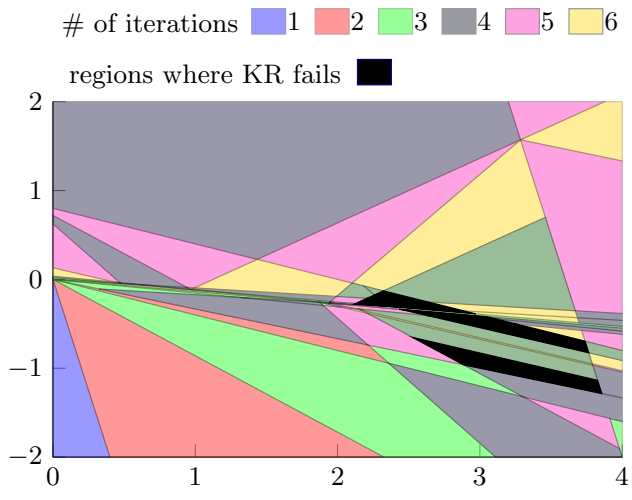
CERTIFICATION OF KR SOLVER

(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

WORKSHOP PROGRAM

- Linear MPC: introduction and algorithms (AB)
- **Nonlinear and economic MPC (MZ)**
- Hybrid and stochastic MPC (AB)
- Reinforcement learning and MPC (AB+MZ)
- Concluding remarks (AB+MZ)

Supplementary material:

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

`https://mariozanon.wordpress.com/teaching/`