MODEL PREDICTIVE CONTROL: A RISING TECHNOLOGY IN THE AUTOMOTIVE INDUSTRY

Alberto Bemporad

http://imt.lu/ab

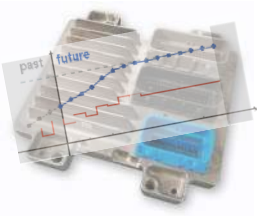
IMT SCHOOL FOR ADVANCED STUDIES LUCCA

ODYS Advanced Controls & Optimization

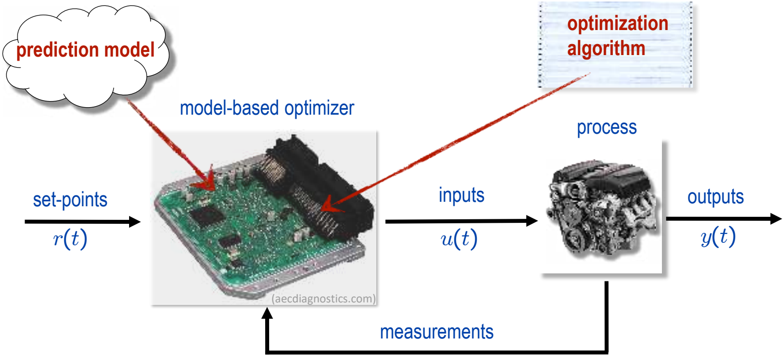
CCTA 2020

OUTLINE

- **Model Predictive Control** (MPC) (in a nutshell)
- MPC in the **automotive industry**
- **Embedded quadratic programming** (QP) solvers for MPC
- **Calibration** of embedded MPC controllers
- Trends in MPC technology (**MPC and machine learning**)



MODEL PREDICTIVE CONTROL (MPC)



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

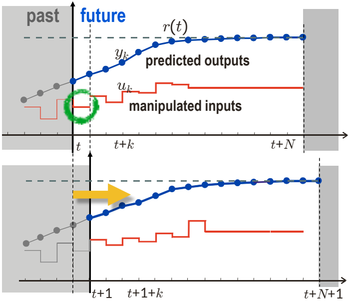
MODEL PREDICTIVE CONTROL (MPC)

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

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

$$\begin{aligned}
\text{s.t.}\quad & x_{k+1} = f(x_k, u_k) && \text{prediction model}\\
& y_k = g(x_k) \\
& u_{\min} \leq u_k \leq u_{\max} && \text{constraints}\\
& y_{\min} \leq y_k \leq y_{\max} \\
& x_0 = x(t) && \text{state feedback}
\end{aligned}$$

numerical optimization problem



1. **estimate** current state $x(t)$
2. **optimize** wrt $\{u_0, \ldots, u_{N-1}\}$
3. only **apply** optimal u_0 as input $u(t)$

- If prediction model is **linear** then optimization is a **Quadratic Program (QP)**

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)

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

OUTLINE

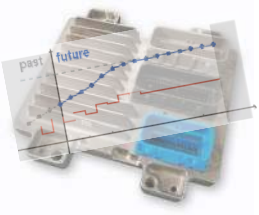
- **Model Predictive Control** (MPC) (in a nutshell)

➔ MPC in the **automotive industry**

- **Embedded quadratic programming** (QP) solvers for MPC

- **Calibration** of embedded MPC controllers

- Trends in MPC technology (**MPC and machine learning**)



AUTOMOTIVE APPLICATIONS OF MPC

(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

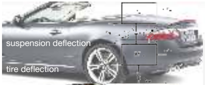
Ford Motor Company

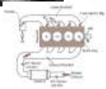
Jaguar

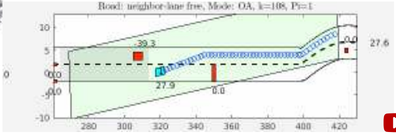
DENSO Automotive

FCA

General Motors

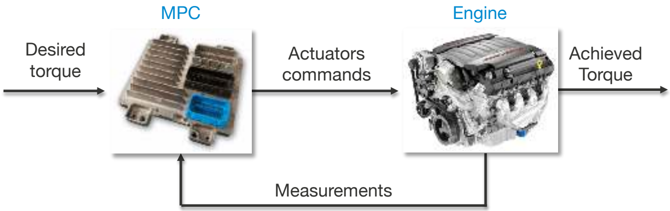






Most automotive OEMs are looking into MPC solutions today

- Optimize **engine actuators** (throttle, wastegate, intake/exhaust cams) to make **engine torque** track set-points, maximizing efficiency and satisfying **constraints**

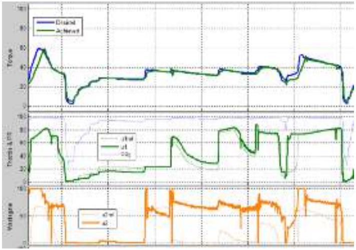


numerical optimization problem solved in real-time on ECU

General Motors

ODYS
Advanced Controls & Optimization

(Bemporad, Bernardini, Long, Verdejo, 2018)



engine operating at low pressure (66 kPa)

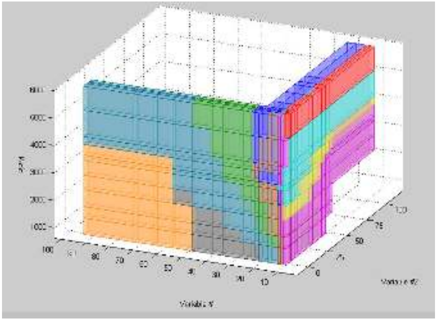
ENGINE CONTROL – MULTIPLE LINEAR MODELS

- **Multitude** of **linear** prediction models derived to cover entire operating envelope of the engine

- Models calibrated to engine data

- Each MPC paired to unique prediction model and Kalman filter

- Number and scheduling of models is important



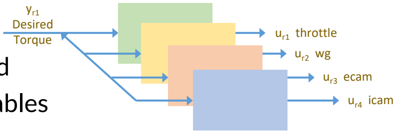
(courtesy of J. Verdejo, GM)

ENGINE CONTROL — STEADY-STATE INPUTS

- Good **steady-state input references** $u_r(t)$ are very important for fuel efficiency

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

- **Optimal** (most fuel-efficient) steady-state actuator positions computed for each desired torque level and **stored in ROM** as look-up tables



(courtesy of J. Verdejo, GM)

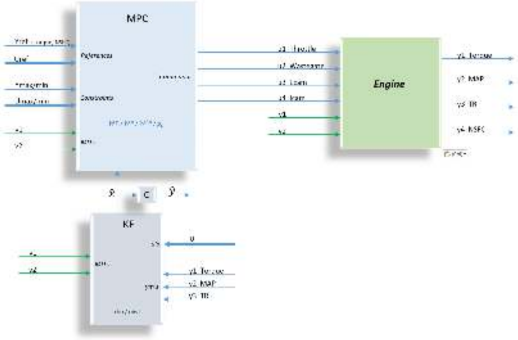
- MPC will closely follow u_r as a **suggestion** during steady-state

- MPC will deviate from u_r during transients and due to aging/changes in environmental conditions

ENGINE CONTROL - CONTROLLER STRUCTURE

- State estimated by Kalman Filter from output measurements $y(t)$

- Signals fed to MPC in real-time:
  - state estimate
  - **references** $r(t), u_r(t)$
  - **constraints**
  - **measured disturbances**



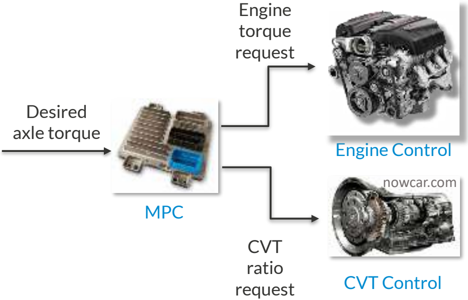
- Main tuning parameters: $W_y, W_u, W_{\Delta u}, \rho$

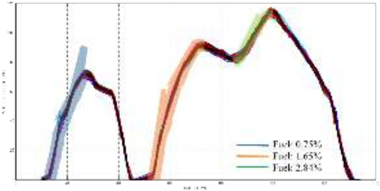
- Optimal input trajectory returned by **QP solver** in real-time

- Framework easily generalizable to other control problems

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 AUTOMOTIVE PRODUCTION SINCE 2018

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

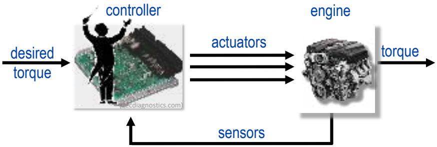
http://www.odys.it/odys-and-gm-bring-online-mpc-to-production

ODYS
Advanced Controls & Optimization

DO WE REALLY NEED ADVANCED CONTROL ?

- Increasingly demanding **requirements** (emissions/consumption, passenger safety and comfort, …)

- Better control performance only achieved by better **coordination** of actuators:

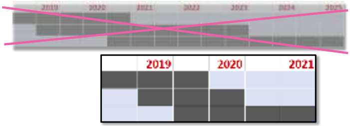


  – **increasing number** of actuators (e.g., due to electrification)

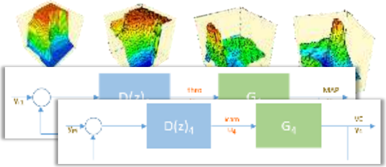
  – take into account **limited range** of actuators

  – resilience in case of some **actuator failure**

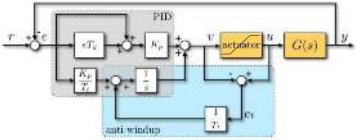
- **Shorter development time** for control solution (market competition, changing legislation)



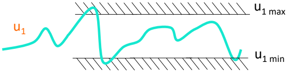
LIMITATIONS OF CLASSICAL CONTROL



- Classical approach:
  - many **single PID loops**
  - **anti-windup** for actuator saturation
  - many **lookup tables**



- Long design & calibration time due to:
  - **complexity** of anti-windup due to **interactions**
  - difficulty to recover from **actuator failure**
  - design space increases **exponentially** (e.g.: 5 inputs, 10 values each $\rightarrow 10^5$ entries)
  - hard to **coordinate** multiple actuators optimally
  - design difficult to port to a different vehicle model



Modern vehicles need advanced controls

KEY CHALLENGES IN MPC DESIGN FOR PRODUCTION

- **Online optimization**

  - Need fast & reliable embedded **optimization solvers**
  - Can we avoid real-time optimization ?

- **Modeling**

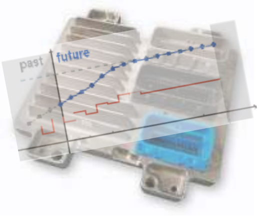
  - Getting the **prediction model** is usually the largest design effort
  - Can we learn good prediction models from data?

- **Calibration** (=reinforcement learning) of MPC

  - Can we automate MPC calibration based on observed performance?

OUTLINE

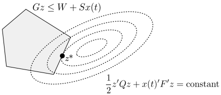
- **Model Predictive Control** (MPC) (in a nutshell)
- MPC in the **automotive industry**
- ➔ **Embedded quadratic programming** (QP) solvers for MPC
- **Calibration** of embedded MPC controllers
- Trends in MPC technology (**MPC and machine learning**)



EMBEDDED LINEAR MPC AND QUADRATIC PROGRAMMING

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

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



ON MINIMIZING A CONVEX FUNCTION SUBJECT TO LINEAR INEQUALITIES

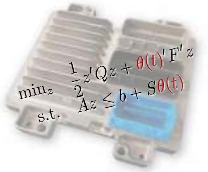
By E. M. L. BEALE

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**

Key requirements for deploying MPC in production:

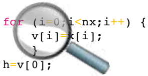
1. **speed (throughput)**
   - **worst-case** execution time less than sampling interval
   - also fast on **average** (to free the processor to execute other tasks)

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.

REGULARIZED ADMM FOR QUADRATIC PROGRAMMING

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

- "Regularized" Alternating Direction Method of Multipliers (**ADMM**):

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

- Works for any $Q \succeq 0$, A, and choice of $\epsilon > 0$ [constraints: $\ell \leq Az \leq u$]
- **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 the free **osQP solver**

  (Python interface: ≈ 1,700,000 downloads)

`http://osqp.org`

- Extended to solve **mixed-integer quadratic programming** problems

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

SOLVING QP'S VIA NONNEGATIVE LEAST SQUARES

(Bemporad, 2016)

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

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

$Q = Q' \succ 0$

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

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

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

(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 the original QP is infeasible. Otherwise set

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

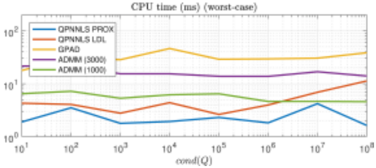
ROBUST QP SOLVER BASED ON NNLS

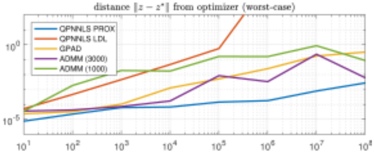
(Bemporad, 2018)

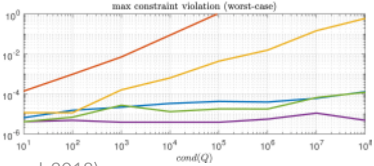
- Solve QP via NNLS within **proximal-point iterations**

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

- **Numerical robustness**: $Q + \epsilon I$ can be arbitrarily well conditioned !



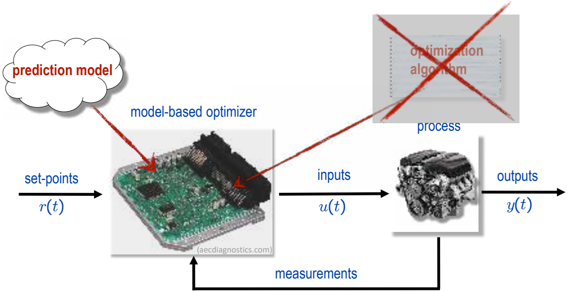




single precision arithmetic
30 vars, 100 constraints
(Macbook Pro 3 GHz Intel Core i7)

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

MPC WITHOUT ON-LINE QP



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

- If model / constraints are linear, and model / constraints / quadratic cost are **time-invariant**:

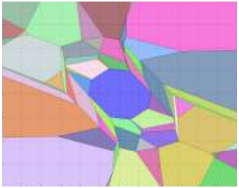
YES !

EXPLICIT MODEL PREDICTIVE CONTROL

- Continuous & piecewise affine solution of strictly convex multiparametric QP

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

(Bemporad, Morari, Dua, Pistikopoulos, 2002)



- Corollary: linear MPC is continuous & piecewise affine !

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

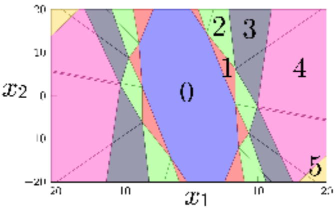
$$u_0^*(x) = \begin{cases} F_1 x + g_1 & \text{if} \quad H_1 x \leq K_1 \\ \vdots & \quad\;\;\vdots \\ F_M x + g_M & \text{if} \quad H_M x \leq K_M \end{cases}$$

- 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 !

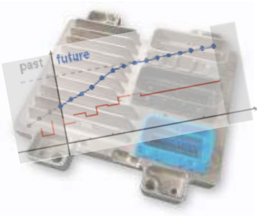
- Examples (from MPC Toolbox):

	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

OUTLINE

- **Model Predictive Control** (MPC) (in a nutshell)
- MPC in the **automotive industry**
- **Embedded quadratic programming** (QP) solvers for MPC
- ➔ **Calibration** of embedded MPC controllers
- Trends in MPC technology (**MPC and machine learning**)



MPC CALIBRATION PROBLEM

- Controller depends on a vector x of parameters

- Parameters can be many things:
  - MPC weights, prediction model coefficients, horizons
  - Entries of covariance matrices in Kalman filter
  - Tolerances used in numerical solvers
  - ...
- Define a **performance index** f over a closed-loop simulation or real experiment. For example:

$$f(x) = \sum_{t=0}^{T} \|y(t) - r(t)\|^2$$

(tracking quality)

- **Auto-tuning** = find the best combination of parameters by solving the **global optimization problem**

$$\min_x f(x)$$

GLOBAL OPTIMIZATION ALGORITHMS FOR AUTO-TUNING

What is a good optimization algorithm to solve $\min f(x)$?

- The algorithm should not require the gradient ∇f of $f(x)$ (**derivative-free** or **black-box optimization**)
- The algorithm should not get stuck on local minima (**global optimization**)
- The algorithm should make the **fewest evaluations** of the cost function f (which is expensive to evaluate)

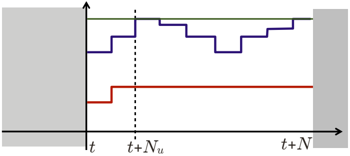
AUTO-TUNING - GLOBAL OPTIMIZATION ALGORITHMS

- Several derivative-free global optimization algorithms exist: (Rios, Sahidinis, 2013)
  - Lipschitzian-based partitioning techniques:
    - **DIRECT** (DIvide in RECTangles) (Jones, 2001)
    - Multilevel Coordinate Search (**MCS**) (Huyer, Neumaier, 1999)
  - Response surface methods
    - **Kriging** (Matheron, 1967), **DACE** (Sacks et al., 1989)
    - Efficient global optimization (**EGO**) (Jones, Schonlau, Welch, 1998)
    - **Bayesian optimization** (Brochu, Cora, De Freitas, 2010)
  - Genetic algorithms (**GA**) (Holland, 1975)
  - Particle swarm optimization (**PSO**) (Kennedy, 2010)
  - ...
- **New method**: radial basis function surrogates + inverse distance weighting (**GLIS**) (Bemporad, 2020)

`cse.lab.imtlucca.it/~bemporad/glis`

- We want to auto-tune the linear MPC controller

$$
\begin{aligned}
\min \quad & \sum_{k=0}^{50-1} (y_{k+1} - r(t))^2 + ({\color{red}W^{\Delta u}}(u_k - u_{k-1}))^2 \\
\text{s.t.} \quad & x_{k+1} = Ax_k + Bu_k \\
& y_c = Cx_k \\
& -1.5 \leq u_k \leq 1.5 \\
& u_k \equiv u_{N_u},\ \forall k = {\color{red}N_u}, \ldots, N-1
\end{aligned}
$$



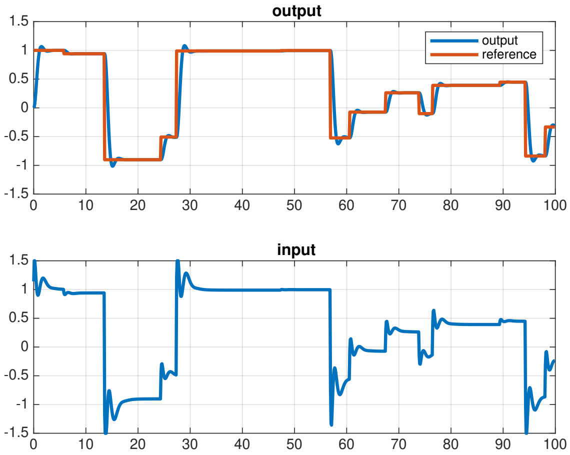
- Calibration parameters: ${\color{red}x = [\log_{10} W^{\Delta u}, N_u]}$

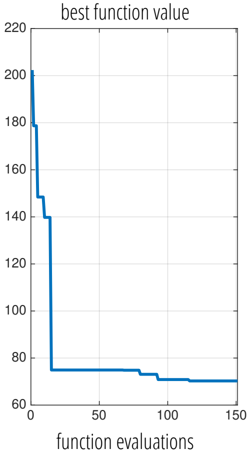
- Range: $-5 \leq x_1 \leq 3$ and $1 \leq x_2 \leq 50$

- Closed-loop performance objective:

$$
f(x) = \sum_{t=0}^{T} \underbrace{(y(t) - r(t))^2}_{\text{track well}} + \underbrace{\frac{1}{2}(u(t) - u(t-1))^2}_{\text{smooth control action}} + \underbrace{2N_u}_{\text{small QP}}
$$

AUTO-TUNING: EXAMPLE



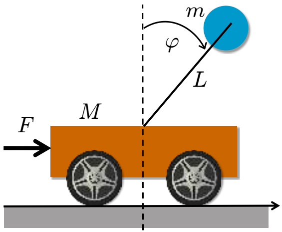


- Result: $x^\star = [-0.2341, 2.3007]$ $\Rightarrow$ $W^{\Delta u} = 0.5833, N_u = 2$

MPC AUTOTUNING EXAMPLE

(Forgione, Piga, Bemporad, 2020)

- Linear MPC applied to cart-pole system: **14** parameters to tune

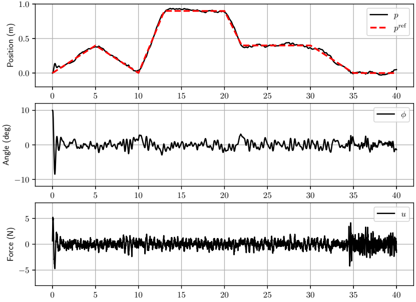


  - sample time
  - weights on outputs and input increments
  - prediction and control horizons
  - covariance matrices of Kalman filter
  - absolute and relative tol of QP solver

- Closed-loop performance score: $J = \int_0^T |p(t) - p_{\text{ref}}(t)| + 30|\phi(t)|dt$
- MPC parameters tuned using 500 iterations of GLIS
- Performance tested with simulated cart on two hardware platforms (PC, Raspberry PI)

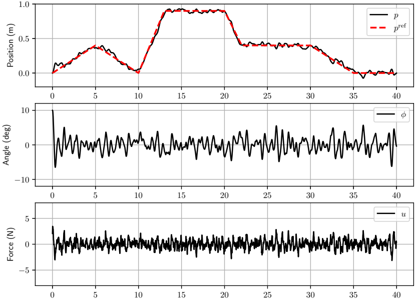
MPC AUTOTUNING EXAMPLE

MPC optimized for **desktop PC**



optimal sample time = **6 ms**

MPC optimized for **Raspberry PI**



optimal sample time = **22 ms**

- Auto-calibration can squeeze max performance out of the available hardware
- Bayesian Optimization gives similar results, but with larger computation effort

AUTO-TUNING: PROS AND CONS

- Pros:
  - Selection of calibration parameters x to test is fully automatic
  - Applicable to any calibration parameter (weights, horizons, solver tolerances, ...)

    (Piga, Forgione, Formentin, Bemporad, 2019) (Forgione, Piga, Bemporad, 2020)
  - Rather arbitrary performance index $f(x)$ (tracking performance, response time, worst-case number of flops, ...)

- Cons:
  - Need to **quantify** an objective function $f(x)$
  - No room for **qualitative** assessments of closed-loop performance
  - Often have **multiple objectives**, not clear how to blend them in a single one

- Current research: **preference-based optimization** (**GLISp**), having human assessments in the loop (**semi-automatic tuning**)

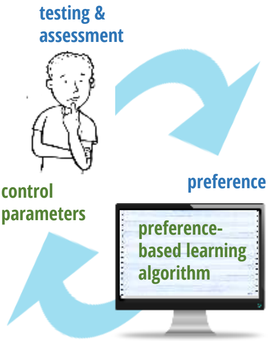
  (Bemporad, Piga, 2019)

  (Zhu, Bemporad, Piga, 2020)

`cse.lab.imtlucca.it/~bemporad/glis`

PREFERENCE-BASED LEARNING (=SEMI-AUTOMATIC TUNING)

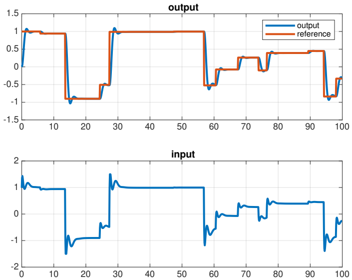
- Latent function = calibrator's (unconscious) score
- The active preference learning algorithm **proposes a new combination** x_{N+1} of parameters to test
- By observing test results, the calibrator expresses a **preference**, telling if x_{N+1} is "better", "similar", or "worse" than current best combination
- Preference learning algorithm: **update the surrogate** $\hat{f}(x)$ of the latent function, optimize the acquisition function, **ask preference**, and **iterate**



PREFERENCE-BASED TUNING: MPC EXAMPLE

- Semi-automatic tuning of $x = [\log_{10} W^{\Delta u}, N_u]$ in linear MPC

$$
\begin{aligned}
\min \quad & \sum_{k=0}^{50-1} (y_{k+1} - r(t))^2 + (W^{\Delta u}(u_k - u_{k-1}))^2 \\
\text{s.t.} \quad & x_{k+1} = Ax_k + Bu_k \\
& y_c = Cx_k \\
& -1.5 \leq u_k \leq 1.5 \\
& u_k \equiv u_{N_u},\ \forall k = N_u, \ldots, N-1
\end{aligned}
$$



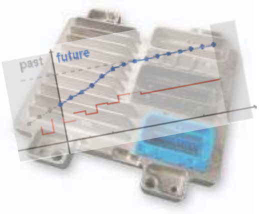
- Same performance index to assess closed-loop quality, but unknown: **only preferences** are available

- Result: $W^{\Delta u} = 0.6888$, $N_u = 2$

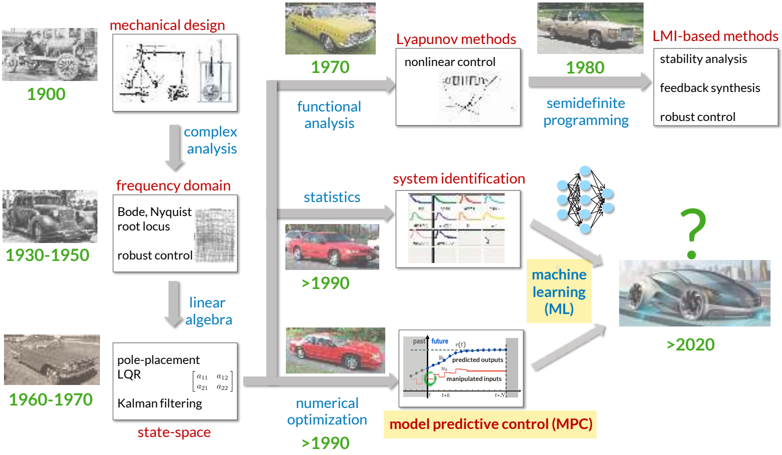
OUTLINE

- **Model Predictive Control** (MPC) (in a nutshell)
- MPC in the **automotive industry**
- **Embedded quadratic programming** (QP) solvers for MPC
- **Calibration** of embedded MPC controllers

➔ Trends in MPC technology (**MPC and machine learning**)



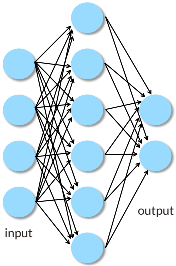
WHERE IS CONTROL ENGINEERING HEADING TO ?



- **MPC** and **ML** = main trends in control R&D in industry !

MACHINE LEARNING (ML)

- Massive set of techniques to **extract mathematical models from data** for **classification, prediction, decision-making**
- Good **mathematical foundations** from artificial intelligence, statistics, optimization
- **Works very well** in practice (despite training is most often a nonconvex optimization problem ...)
- Used in myriads of **very diverse application domains**
- Availability of excellent open-source **software tools** like `scikit-learn`, `Keras/TensorFlow` also explains success



ML FOR MPC

- How can ML be useful in MPC:

  - **Identification** = learn the **prediction model** from data $\begin{cases} x_{k+1} & = & f(x_k, u_k) \\ y_k & = & g(x_k) \end{cases}$

  - **Control** = learn the MPC **control law** from data

    - **reinforcement learning** (best for automatic calibration)
    - **imitation learning** (= approximate explicit MPC)
      (Lenz, Knepper, Saxena, 2015) (Karg, Lucia, 2018)

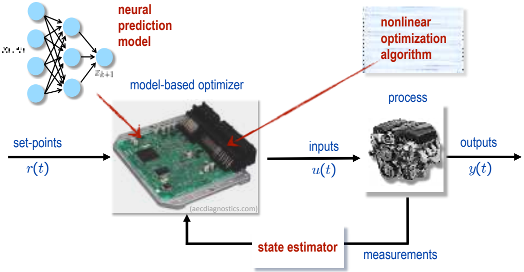
  - **Optimization** = learn (partial) solutions offline for on-line optimization

    - **binary variables** solving parametric MIQP/LP, $\delta^* = \delta(x)$, then solve QP/LP online
      (Masti, Bemporad, 2019) (Masti, Pippia, Bemporad, De Schutter, 2020)
    - **active set** of parametric QP for warm start (Klauco, Kalúz, Kvasnica, 2019)

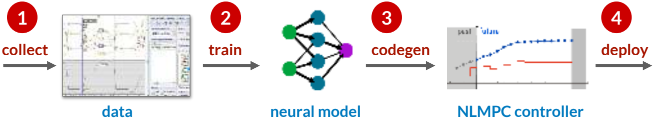
  - **Estimation** = learn how to reconstruct **unmeasured signals** from data (e.g., states)

NLMPC BASED ON DEEP NEURAL NETWORKS

- **Approach**: use a (feedforward deep) neural network model for prediction



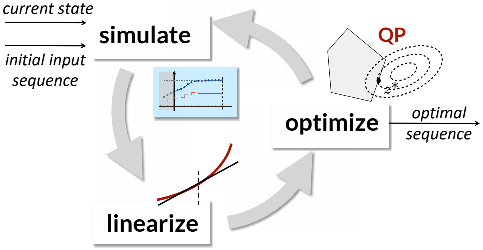
- MPC design workflow:



NONLINEAR MPC

(Gros, Zanon, Quirynen, Bemporad, Diehl, 2020)

- **Nonlinear MPC**: solve a **sequence of LTV-MPC** problems at each sample step



- **Sequential QP** solves the full nonlinear MPC problem, by using well assessed linear MPC/QP technologies

- One QP iteration is often sufficient (= linear time-varying MPC)

- The current state can be estimated, e.g., by **extended Kalman filtering**

ODYS EMBEDDED MPC TOOLSET

- **ODYS Embedded MPC** is a software toolchain for design and deployment of MPC solutions in industrial production

ODYS Advanced Controls & Optimization

- Support for **linear & nonlinear MPC** and **extended Kalman filtering**
- Extremely flexible, all MPC parameters can be changed at runtime (models, cost function, horizons, constraints, ...)
- Integrated with **ODYS QP Solver** for max speed, low memory footprint, and robustness (also in single precision)

odys.it/qp

- Library-free C code, **MISRA-C 2012 compliant**
- Currently used worldwide by several automotive OEMs in R&D and production
- Support for **neural networks** as prediction models (**ODYS Deep Learning**)

odys.it/embedded-mpc

MPC OF ETHYLENE OXIDATION PLANT

- **Chemical process** = **oxidation of ethylene to ethylene oxide** in a nonisothermal continuously stirred tank reactor (CSTR)

$$
\begin{aligned}
&C_2H_4 + \tfrac{1}{2}O_2 \to C_2H_4O\\
&C_2H_4 + 3O_2 \to 2CO_2 + 2H_2O\\
&C_2H_4O + \tfrac{5}{2}O_2 \to 2CO_2 + 2H_2O
\end{aligned}
$$

- **Nonlinear model** (dimensionless variables): (Durand, Ellis, Christofides, 2016)

$$
\left\{
\begin{array}{rcl}
\dot{x}_1 & = & u_1(1 - x_1 x_4)\\
\dot{x}_2 & = & u_1(u_2 - x_2 x_4) - A_1 e^{\frac{\gamma_1}{x_4}}(x_2 x_4)^{\frac{1}{2}} - A_2 e^{\frac{\gamma_2}{x_4}}(x_2 x_4)^{\frac{1}{4}}\\
\dot{x}_3 & = & -u_1 x_3 x_4 + A_1 e^{\frac{\gamma_1}{x_4}}(x_2 x_4)^{\frac{1}{2}} - A_3 e^{\frac{\gamma_3}{x_4}}(x_3 x_4)^{\frac{1}{2}}\\
\dot{x}_4 & = & \frac{u_1(1-x_4) + B_1 e^{\frac{\gamma_1}{x_4}}(x_2 x_4)^{\frac{1}{2}} + B_2 e^{\frac{\gamma_2}{x_4}}(x_2 x_4)^{\frac{1}{4}}}{x_1}\\
 & & + \frac{B_3 e^{\frac{\gamma_3}{x_4}}(x_3 x_4)^{\frac{1}{2}} - B_4(x_4 - T_c)}{x_1}\\
y & = & x_3
\end{array}
\right.
$$

x_1 = gas density
x_2 = ethylene concentration
x_3 **= ethylene oxide concentration**
x_4 = temperature in reactor

u_1 **= feed volumetric flow rate**
u_2 = ethylene concentration in feed

- u_1 = manipulated variables, x_3 = controlled output, u_2 = measured disturbance

MPC OF ETHYLENE OXIDATION PLANT

- **MPC** settings:

sampling time	$T_s = 5\,\mathrm{s}$	measured disturbance @t=200
prediction horizon	$N = 10$	
control horizon	$N_u = 3$	
constraints	$0.0704 \leq u_1 \leq 0.7042$	
cost function	$\sum_{k=0}^{N-1}(y_{k+1} - r_{k+1})^2 + \frac{1}{100}(u_{1,k} - u_{1,k-1})^2$	

- We compare 3 different configurations:
  - NLMPC based on **physical model**
  - Switched linear MPC based on **3 linear models** obtained by linearizing the nonlinear model at $C_2H_4O = \{0.03,\ 0.04,\ 0.05\}$
  - NLMPC based on black-box **neural network** model

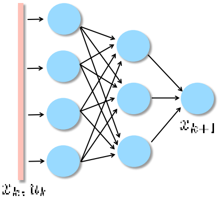
NEURAL NETWORK MODEL OF ETHYLENE OXIDATION PLANT

- Train **state-space neural-network** model

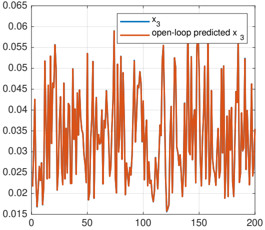
$$x_{k+1} = \mathcal{N}(x_k, u_k)$$

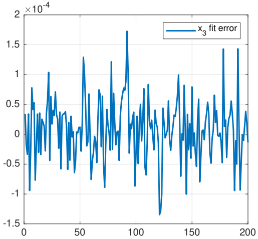
1,000 training samples $\{u_k, x_k\}$
2 layers (6 neurons, 6 neurons)
6 inputs, 4 outputs
sigmoidal activation function

$\rightarrow$ **112 coefficients**



x_3 (validation data)



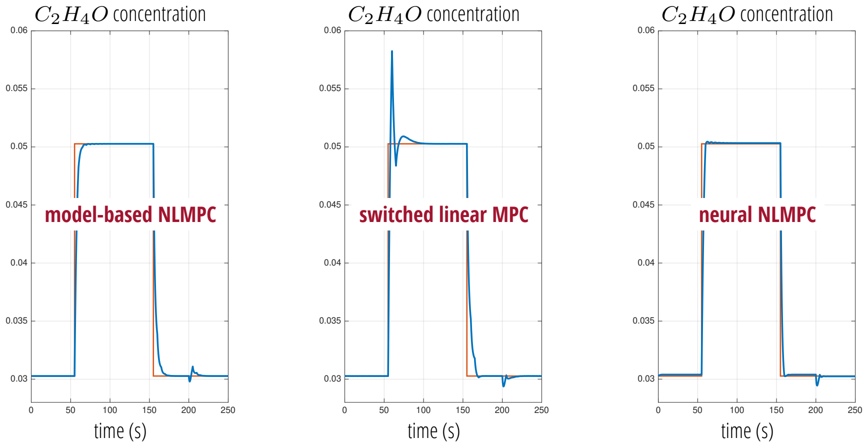


validation sample

- NN model trained by **ODYS Deep Learning** toolset (model fitting + Jacobians $\rightarrow$ neural model in C)

- Model validated on 200 samples.
$x_{3,k+1}$ reproduced from x_k, u_k with max 0.4% error

MPC OF ETHYLENE OXIDATION PLANT - CLOSED-LOOP RESULTS

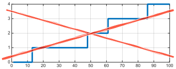


- Neural and model-based NLMPC have **similar** closed-loop performance
- Neural NLMPC requires **no physical model**

ON THE USE OF NEURAL NETWORKS FOR MPC

- Neural prediction models can **speed up** the MPC design a lot

- Experimental **data**:
  - need to cover the operating range well (as in linear SYS-ID)
  - no need to define linear operating ranges with NN's, it is a **one-shot model-learning** step
  - NN coefficients can be updated on-line (=adaptive NLMPC)



- Physical models may **better predict** unseen situations than black box models

- Physical modeling can help driving the choice of the **nonlinear model structure** to use (gray-box models)

CONCLUSIONS

- Long history of success of MPC in the **process industries**
  - multivariable, linear/nonlinear/stochastic systems w/ **constraints**
  - intuitive to **design** and **calibrate**, easy to **reconfigure**

- MPC is now a viable technology in the **automotive industry** too:
  1. modern ECUs can solve MPC problems in **real-time**
  2. advanced **software tools** are available for identification, design, calibration, and deployment of MPC solutions
  3. increasingly tight **requirements** ask for advanced multivariable control solutions
  4. **production managers** are willing to deploy MPC in the vehicle

- MPC based on deep neural models is probably the next step in MPC technology