MODEL PREDICTIVE CONTROL FOR AUTOMOTIVE PRODUCTION

Alberto Bemporad

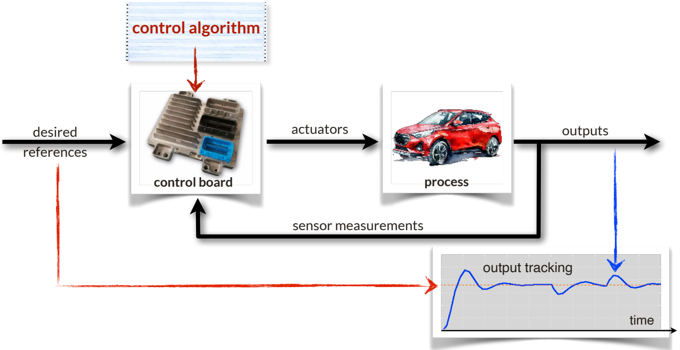
imt.lu/ab

IMT SCHOOL FOR ADVANCED STUDIES LUCCA
www.imtlucca.it

ODYS Advanced Controls & Optimization
www.odys.it

November 9, 2023

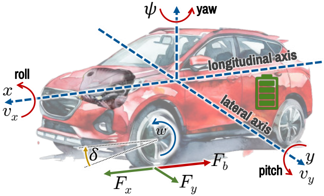
VEHICLE CONTROL



- **Vehicle control** = use of **algorithms** for manipulating **actuators** in real time based on **sensor** measurement feedback to **ensure proper vehicle behavior**

- Vehicle controls are fundamental for:
  - **efficiency** (optimized operations, energy management) **[cleaner environment!]**
  - passenger **comfort** and **safety** (advanced driver assistance systems) **[save lives!]**

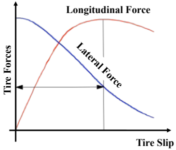
VEHICLE CONTROL



- **Examples** of vehicle control systems:

  Electronic Stability Control (**ESC**), Traction Control System (**TCS**), Adaptive Cruise Control (**ACC**), Lane Keeping Assist (**LKA**), Anti-lock Braking System (**ABS**), Engine Control Unit (**ECU**), Transmission Control Unit (**TCU**), ..., **Autonomous Driving** (**AD**)

- **Complexity** of vehicle control problems:
  - **multiple actuators** (e.g., 4 traction/braking forces, front/rear steering, electric motors, ...)
  - **nonlinearities** and **uncertainties** (e.g., tire forces)
  - **highly coupled** dynamics and **interactions** of many control systems (engine control, transmission control, heat distribution, ...)

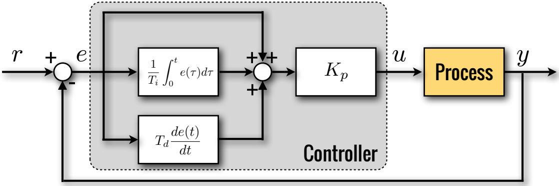


(Borrelli, Bemporad, Tseng, Hrovat, 2006)

Control is a fundamental software component for proper vehicle operations

CLASSICAL CONTROL

- Proportional Integrative Derivative (PID) controllers are the most used controllers in industrial automation since the ’30s



PID Controller

✔ **Single-loops** are very easy to tune, just **3 parameters to calibrate**

✔ **Few lines of C code**, minimal memory and throughput requirements

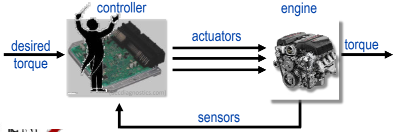
✔ **No process model** required, just output measurements

PIDs widely used in vehicle control. So why consider new control methods?

CONTROL REQUIREMENTS

- Increasing **requirements** (emissions, fuel efficiency, passenger comfort, ...)

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

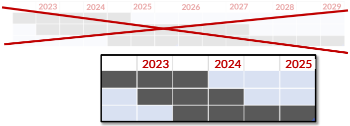


  - **increasing number** of actuators (e.g., due to electrification)

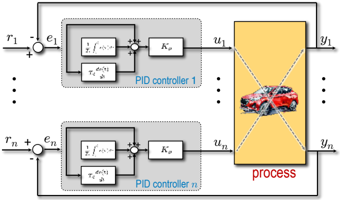
  - take into account **limited range** of actuators

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

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



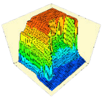
PID CONTROL: LIMITATIONS



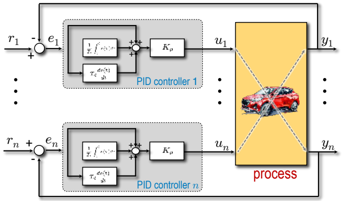
✖ **Multi-input/multi-output** systems: dynamical coupling requires tuning multiple PID loops together

- ☹ Surgically changing a PID loop tuning may have bad consequences on other loops, due to dynamical **interactions**
- ☹ Lookup-table complexity increases **exponentially** (e.g.: 5 inputs, 10 values each $\rightarrow 10^5$ entries)
- ☹ Hard to coordinate multiple actuators **optimally**
- ☹ The calibration might need to be completely redone for a new vehicle model

	A	B	C	D	E
1	Input 1	Input 2	Input 3	Input 4	Input 5
2	0.0119	0.0046	0.0287	0.0155	0.0012
3	0.0318	0.0154	0.0292	0.0225	0.0067
4	0.0344	0.043	0.0305	0.0326	0.0336
5	0.0357	0.0497	0.0377	0.0424	0.0358
6	0.0462	0.0598	0.0855	0.0527	0.068
7	0.054	0.076	0.0987	0.0596	0.0688
8	0.0759	0.0782	0.1068	0.0605	0.0908
9	0.0971	0.0811	0.1111	0.0714	0.0911
10	0.0975	0.0838	0.1174	0.0835	0.0942
11	0.119	0.0844	0.1366	0.0967	0.1056



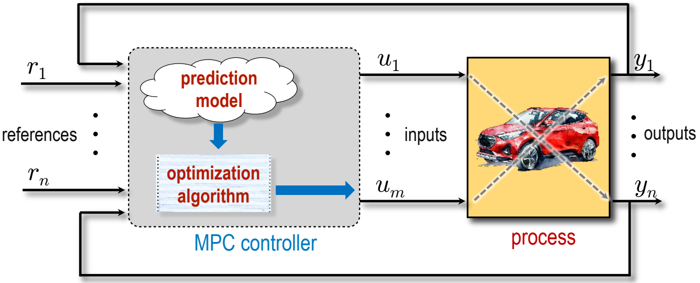
PID CONTROL: LIMITATIONS



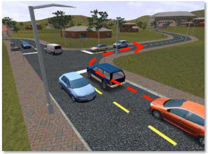
- ✖ Handling **input constraints** require additional **anti-windup** design
- ✖ **Output constraints** are much harder to handle
- ✖ Limited **preview** (derivative term =1st order extrapolation of future output)
- ✖ No explicit performance index optimized at runtime
- ✖ Resilience to **actuator faults** requires further design effort

Classical control can be inadequate (time-consuming & suboptimal design)

MODEL PREDICTIVE CONTROL (MPC)



- **Key idea:** At each sample step, use a (simplified) dynamical **(M)odel** of the process to **(P)redict** its future evolution and choose the "best" **(C)ontrol** action accordingly



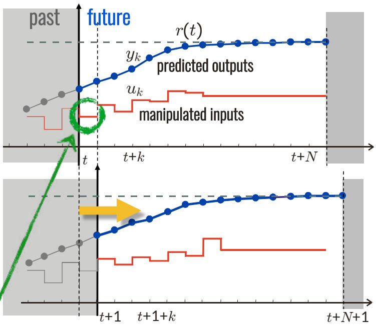
MODEL PREDICTIVE CONTROL

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

$$\min_{u_0,\ldots,u_{N-1}} \sum_{k=0}^{N-1} \|y_k - r(t)\|_2^2 + \rho\|u_k - u_r(t)\|_2^2$$

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

→ **numerical optimization problem**



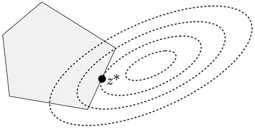
1. **estimate** current state $x(t)$
2. **optimize** wrt $\{u_0, \ldots, u_{N-1}\}$
3. only **apply** optimal u_0 as input $u(t)$

Repeat at all time steps t

LINEAR MPC

- **Linear** prediction model: real-time optimization = **Quadratic Program (QP)**

$$\min_z \quad \frac{1}{2} z' H z + x'(t) F' z$$
$$\text{s.t.} \quad Gz \leq W + Sx(t)$$



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

- The MPC concept dates back to the 60's (Rafal, Stevens, 1968) (Propoi, 1963)

- MPC is used in the process industries since the 80's (Qin, Badgewell, 2003)

Today APC (advanced process control) = MPC

©SimulateLive.com

RESEARCH ON 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

Fiat

General Motors

ODYS
Advanced Controls & Optimization

Most automotive OEMs are looking into MPC solutions today

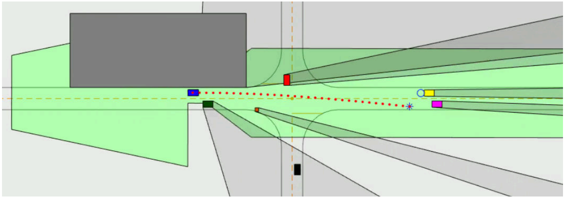
MPC FOR AUTONOMOUS DRIVING / DRIVER-ASSISTANCE SYSTEMS

(Graf Plessen, Bernardini, Esen, Bemporad, 2018)

- Coordinate **torque request** and **steering** to achieve **safe and comfortable** autonomous driving with no collisions

- MPC combines **path planning**, **path tracking**, and **obstacle avoidance**

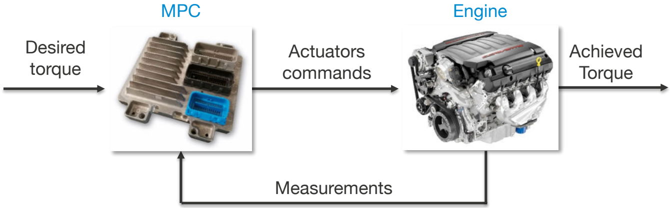
- **Stochastic prediction models** used to account for uncertainty (other vehicles/pedestrians, driver's requests)



ODYS Advanced Controls & Optimization

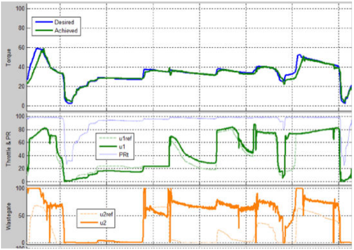
MPC OF GASOLINE TURBOCHARGED ENGINES

- 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)

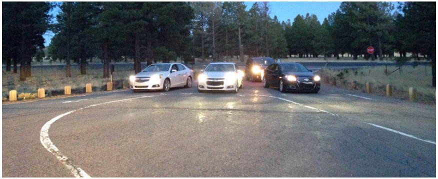


engine operating at low pressure (66 kPa)

MPC IN AUTOMOTIVE PRODUCTION

- MPC of turbocharged gasoline engine in GM production since 2018
  (Bemporad, Bernardini, Long, Verdejo, 2018)

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

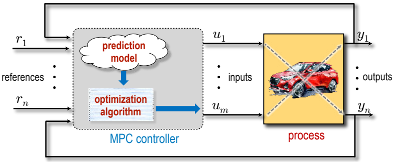


First known mass production of MPC in the automotive industry

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

ODYS real-time optimization and embedded MPC software is currently running on **3+ million vehicles** worldwide

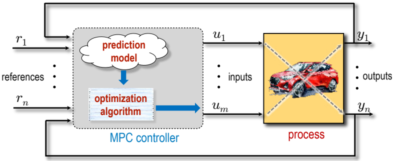
MODEL PREDICTIVE CONTROL (MPC)



$$
\begin{aligned}
\min \quad & \sum_{k=0}^{N-1} \|y_k - r_{t+k}\|_2^2 + \rho \|u_k - u_{r,t+k}\|_2^2 \\
\text{s.t.} \quad & x_{k+1} = f(x_k, u_k) \\
& y_k = g(x_k) \\
& u_{\min} \leq u_k \leq u_{\max} \\
& y_{\min} \leq y_k \leq y_{\max}
\end{aligned}
$$

- Naturally coordinates **multiple inputs and outputs** and over-actuated systems (# inputs $>$ # outputs)
- Naturally handles **input and output constraints**
- Very easily includes **preview** on references/measured disturbances
- Design easy to **transfer** to new models (**no lookup tables**)
- Controller easily reconfigurable online to **handle faults** (resilience)

MODEL PREDICTIVE CONTROL (MPC)



$$\min \sum_{k=0}^{N-1} \|y_k - r_{t+k}\|_2^2 + \rho\|u_k - u_{r,t+k}\|_2^2$$

$$\text{s.t.} \quad x_{k+1} = f(x_k, u_k)$$
$$y_k = g(x_k)$$
$$u_{\min} \le u_k \le u_{\max}$$
$$y_{\min} \le y_k \le y_{\max}$$

Price to pay:

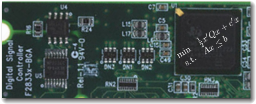
- ✖ Nontrivial C code, requires **formulating and solving QP problems** at runtime
- ✖ Requires a **process model** (physical modeling and/or system identification) (similar to all **model-based control-design** methods)
- ✖ Multiple parameters to calibrate (models, weights, solver tolerances, ...)

EMBEDDED QUADRATIC OPTIMIZATION

EMBEDDED SOLVERS IN PRODUCTION

- Many QP algorithms exist today, but not all are suitable for **embedded control**

Key requirements for deploying QP in production:



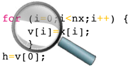
1. **speed (throughput)**
   - **worst-case** execution time less than sampling interval
   - also fast on **average** (to free the processor to execute other tasks)

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

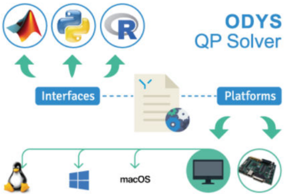
5. **code simple enough** to be validated/verified/certified
   (library-free C code, easy to check by production engineers)



ODYS QP SOLVER

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

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



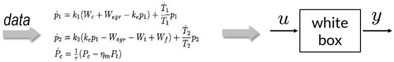
- Implements a **proprietary** state-of-the-art method for QP
- Completely written in **ANSI-C** and **MISRA-C 2012** compliant
- **Fast**, **robust** (also in single precision), **low-memory** requirements
- **Optimized version for MPC** available ($\approx$ 50% faster)
- Licensed to several automotive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

odys.it/qp

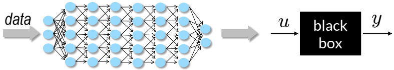
PREDICTION MODELS FOR MPC

PREDICTION MODELS FOR MPC

- **Physical models** might be already available from digital twins



- **Black-box system identification** is a mature technology (ARX, N4SYD, neural networks, ...)

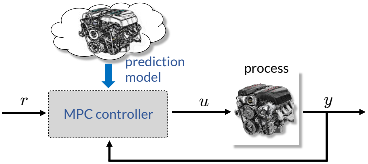


- **Gray-box** (or **physics-informed**) models: mix of the two, can be quite effective

- Should the model be **perfect?**

> "All models are wrong, but some are useful."
>
> (George E. P. Box)

- A model is a **good model** for MPC if
  - captures the **main dynamics** of the process
  - the resulting MPC closed-loop **performs well**



NONLINEAR SYS-ID BASED ON NEURAL NETWORKS

- **Neural networks** proposed for nonlinear system identification since the '90s

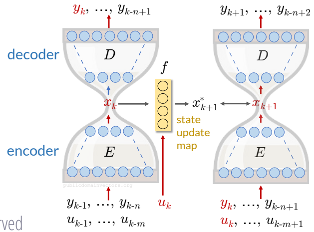
  (Narendra, Parthasarathy, 1990) (Hunt et al., 1992) (Suykens, Vandewalle, De Moor, 1996)

- **NNARX** models: use a **feedforward neural network** to approximate the nonlinear difference equation $y_t \approx \mathcal{N}(y_{t-1}, \ldots, y_{t-n_a}, u_{t-1}, \ldots, u_{t-n_b})$

- **Neural state-space** models:

  - **w/ state data**: fit a neural network model $x_{t+1} \approx \mathcal{N}_x(x_t, u_t), \quad y_t \approx \mathcal{N}_y(x_t)$

  - **I/O data only**: set x_t = value of an inner layer of the network (Prasad, Bequette, 2003) such as an **autoencoder** (Masti, Bemporad, 2021)



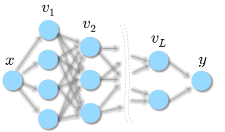
- **Recurrent neural networks** (RNNs): more appropriate for open-loop prediction, but more difficult to train than feedforward NNs

RECURRENT NEURAL NETWORKS

- **Recurrent Neural Network** (RNN) model:

$$\begin{aligned} x_{k+1} &= f_x(x_k, u_k, \theta_x) \\ y_k &= f_y(x_k, \theta_y) \\ f_x, f_y &= \text{feedforward neural network} \end{aligned}$$

(e.g.: general RNNs, LSTMs, RESNETS, physics-informed NNs, ...)



$$v_j = A_j f_{j-1}(v_{j-1}) + b_j$$

$$\theta = (A_1, b_1, \ldots, A_L, b_L)$$

- **Training problem**: given a dataset $\{u_0, y_0, \ldots, u_{N-1}, y_{N-1}\}$ solve

$$\begin{aligned} \min_{\substack{\theta_x, \theta_y \\ x_0, x_1, \ldots, x_{N-1}}} \quad & r(x_0, \theta_x, \theta_y) + \frac{1}{N}\sum_{k=0}^{N-1} \ell(y_k, f_y(x_k, \theta_y)) \\ \text{s.t.} \quad & x_{k+1} = f_x(x_k, u_k, \theta_x) \end{aligned}$$

- **Main issue**: x_k are **hidden states**, i.e., are **unknowns** of the problem

OFFLINE AND ONLINE TRAINING RNNS BY EKF

(Puskorius, Feldkamp, 1994) (Wang, Huang, 2011) (Bemporad, 2023)

- Estimate both hidden states x_k and parameters θ_x, θ_y by **EKF** based on model

$$\begin{cases} x_{k+1} = f_x(x_k, u_k, \theta_{xk}) + \xi_k \\ \begin{bmatrix} \theta_{x(k+1)} \\ \theta_{y(k+1)} \end{bmatrix} = \begin{bmatrix} \theta_{xk} \\ \theta_{yk} \end{bmatrix} + \eta_k \\ y_k = f_y(x_k, \theta_{yk}) + \zeta_k \end{cases}$$

Ratio $\mathrm{Var}[\eta_k]/\ \mathrm{Var}[\zeta_k]$ related to **learning-rate** of training algorithm

Inverse of initial matrix P_0 related to **ℓ_2-penalty** on θ_x, θ_y

- RNN and its hidden state x_k can be estimated **on line** from a streaming dataset $\{u_k, y_k\}$, and/or **offline** by processing multiple epochs of a given dataset

- Can handle **general smooth strongly convex** loss fncs/regularization terms

- Can add **ℓ_1-penalty** $\lambda \left\| \begin{bmatrix} \theta_x \\ \theta_y \end{bmatrix} \right\|_1$ to **sparsify** θ_x, θ_y by changing EKF update into

$$\begin{bmatrix} \hat{x}(k|k) \\ \theta_x(k|k) \\ \theta_y(k|k) \end{bmatrix} = \begin{bmatrix} \hat{x}(k|k-1) \\ \theta_x(k|k-1) \\ \theta_y(k|k-1) \end{bmatrix} + M(k)e(k) - \lambda P(k|k-1) \begin{bmatrix} 0 \\ \mathrm{sign}(\theta_x(k|k-1)) \\ \mathrm{sign}(\theta_y(k|k-1)) \end{bmatrix}$$

TRAINING RNNS BY SEQUENTIAL LEAST-SQUARES AND ADMM

(Bemporad, 2023)

- Use the **alternating direction method of multipliers** (ADMM) by splitting

$$\begin{array}{rl} \min_{\theta_x,\theta_y,x_0,\nu_x,\nu_y} & r(x_0,\theta_x,\theta_y)+\sum_{k=0}^{N-1}\ell(y_k,f_y(x_k,\theta_y))+g(\nu_x,\nu_y) \\ \text{s.t.} & x_{k+1}=f_x(x_k,u_k,\theta_x) \\ & \begin{bmatrix}\nu_x\\ \nu_y\end{bmatrix}=\begin{bmatrix}\theta_x\\ \theta_y\end{bmatrix} \end{array}$$

- Each ADMM iteration requires solving a standard **least-squares** problem

- Either **line-search** (LS) or a **trust-region** method (Levenberg-Marquardt) (LM) is used while optimizing:

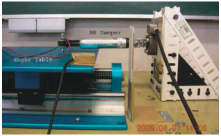
  - **NAILS** = Nonconvex ADMM Iterations and Sequential LS with Line Search

  - **NAILM** = Nonconvex ADMM Iterations and Sequential LS with Levenberg-Marquardt

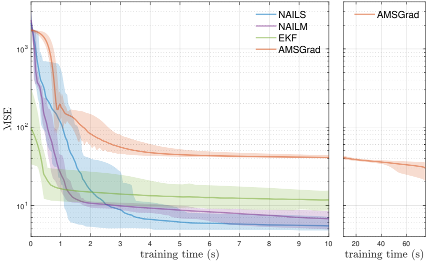
TRAINING RNNS BY SEQUENTIAL LS AND ADMM

(Bemporad, 2023)

- **Example**: **magneto-rheological fluid damper**
  N=2000 data used for training, 1499 for testing the model
  (Wang, Sano, Chen, Huang, 2009)



- RNN model: 4 states, shallow NNs w/ **4 neurons**, **I/O feedthrough**



MSE loss on training data, mean value and range over 20 runs from different random initial weights

NAILS = GNN method with line search

NAILM = GNN method with LM steps

Best Fit Rate	training	test
NAILS	**94.41** (0.27)	89.35 (2.63)
NAILM	94.07 (0.38)	89.64 (2.30)
EKF	91.41 (0.70)	87.17 (3.06)
AMSGrad	84.69 (0.15)	80.56 (0.18)

TRAINING RNNS BY SEQUENTIAL LS AND ADMM

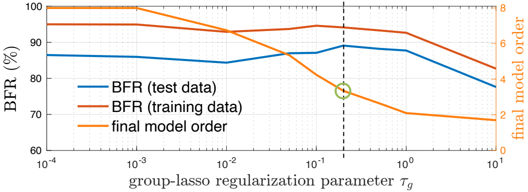
(Bemporad, 2023)

- Fluid-damper example: **Lasso regularization** $g(\nu_x, \nu_y) = 0.2\|\nu_x\|_1 + 0.2\|\nu_y\|_1$

training algorithm	BFR training	BFR test	sparsity %	CPU time	# epochs
NAILS	91.00 (1.66)	87.71 (2.67)	**65.1** (6.5)	**11.4** s	250
NAILM	**91.32** (1.19)	**87.80** (1.86)	64.1 (7.4)	11.7 s	250
EKF	89.27 (1.48)	86.67 (2.71)	47.9 (9.1)	13.2 s	50
AMSGrad	91.04 (0.47)	88.32 (0.80)	16.8 (7.1)	64.0 s	2000
Adam	90.47 (0.34)	87.79 (0.44)	8.3 (3.5)	63.9 s	2000
DiffGrad	90.05 (0.64)	87.34 (1.14)	7.4 (4.5)	63.9 s	2000

$\approx$ same fit than SGD/EKF but sparser models and faster (CPU: Apple M1 Pro)

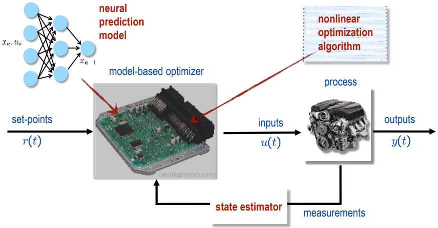
- Fluid-damper example: **group-Lasso regularization** $g(\nu_i^g) = \tau_g \sum_{i=1}^{n_x} \|\nu_i^g\|_2$ to zero entire rows and columns and **reduce state-dimension** automatically



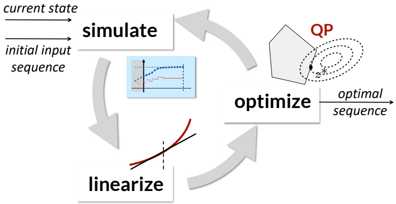
good choice: $n_x = 3$ (best fit on test data)

NONLINEAR MPC BASED ON NEURAL NETWORKS

- **Approach**: use a neural network model for prediction



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



ODYS EMBEDDED MPC TOOLSET

- **ODYS Embedded MPC** is a software toolchain for design and deployment of MPC solutions in industrial production
- 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 **MPC-specific version** of **ODYS QP Solver**
- 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`

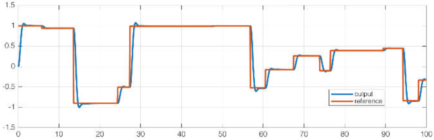
CALIBRATION AND CRITICAL SCENARIO DETECTION

BEST MPC CALIBRATION

- The design depends on a vector x of **control parameters**
- x = (weights, covariance matrices, solver thresholds, ...)
- 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**

$$\min_x f(x)$$

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, ...)
  - ✔ Rather arbitrary performance index $f(x)$ (tracking performance, response time, worst-case number of flops, ...)
- **Cons**:
  - ✖ The calibrator must **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

ACTIVE PREFERENCE LEARNING

(Bemporad, Piga, *Machine Learning*, 2021)

- Objective function $f(x)$ is not available (**latent function**)

- We can only express a **preference** between two choices:

$$\begin{array}{ll} x_1 \text{ “better” than } x_2 & [f(x_1) < f(x_2)] \\ x_1 \text{ “as good as” } x_2 & [f(x_1) = f(x_2)] \\ x_2 \text{ “better” than } x_1 & [f(x_1) > f(x_2)] \end{array}$$

- We want to find a global optimum $x^\star$ that is “better” than any other x

- **Active preference learning**: iteratively propose a new sample to compare

- **Key idea**: learn a **surrogate** of the (latent) objective function from preferences

SEMI-AUTOMATIC CALIBRATION BY PREFERENCE-BASED LEARNING

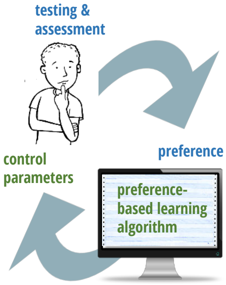
- Use **preference-based optimization** (**GLISp**) algorithm for **semi-automatic tuning** of MPC (Zhu, Bemporad, Piga, 2021) (Bemporad, Piga, 2021)

- Latent function = calibrator's (unconscious) control performance score

- GLISp **proposes a new combination** x_{N+1} of control parameters to test

- The calibrator expresses a **preference**: x_{N+1} is "**better**", "**similar**", or "**worse**" than current best

- Preference learning algorithm iterates:
(1) **update the surrogate** $\hat{f}(x)$ of the latent function,
(2) optimize the acquisition function, (3) **ask preference**



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

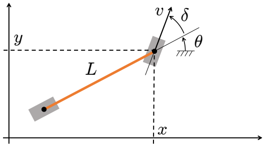
`pip install glis`

PREFERENCE-BASED TUNING: MPC EXAMPLE

(Zhu, Bemporad, Piga, 2021)

- Example: calibration of a simple MPC for lane-keeping (2 inputs, 3 outputs)

$$\begin{cases} \dot{x} &= v\cos(\theta+\delta) \\ \dot{y} &= v\sin(\theta+\delta) \\ \dot{\theta} &= \frac{1}{L}v\sin(\delta) \end{cases}$$



- Multiple control objectives:

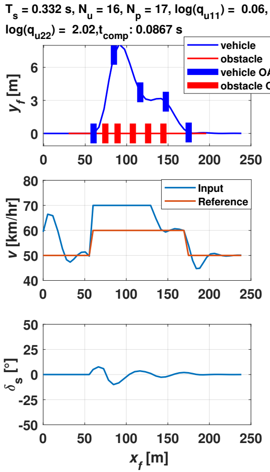
  “*optimal obstacle avoidance*”, “*pleasant drive*”, “*CPU time small enough*”, …

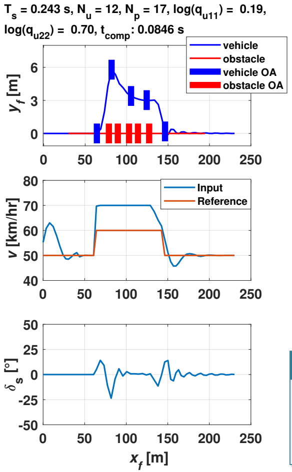
  ➡ **not easy to quantify in a single function**

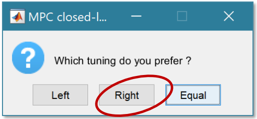
- 5 MPC parameters to tune:
  - **sampling time**
  - prediction and control **horizons**
  - **weights** on input increments Δv, $\Delta\delta$

PREFERENCE-BASED TUNING: MPC EXAMPLE

- Preference query window:

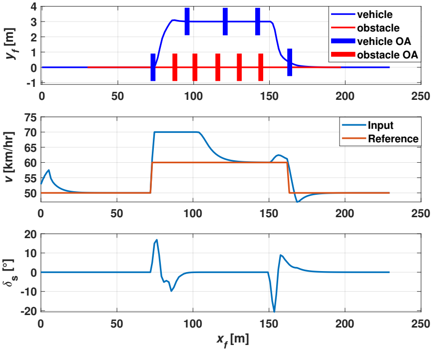






PREFERENCE-BASED TUNING: MPC EXAMPLE

- Convergence after 50 GLISp iterations (=49 queries):



Optimal MPC parameters:

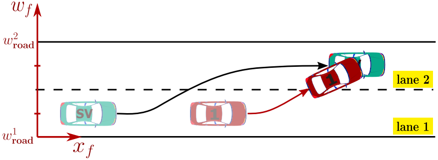
- sample time = 85 ms (CPU time = 80.8 ms)
- prediction horizon = 16
- control horizon = 5
- weight on Δv = 1.82
- weight on $\Delta\delta$ = 8.28

- **Note**: no need to define a closed-loop performance index explicitly!

WORST-CASE SCENARIO DETECTION

(Zhu, Bemporad, Kneissl, Esen, 2023)

- **Goal**: detect **undesired closed-loop scenarios** (=**corner-cases**)
- Let x = parameters defining the scenario (e.g., initial conditions, disturbances, ...)
- **Critical scenario** = vector x^* for which the closed-loop behavior is critical

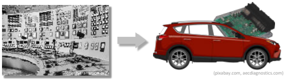


- **Critical scenario detection** = find the **worst** combination x^* of scenario parameters by solving the **global optimization problem**

$$\min_x f(x)$$

CONCLUSIONS

- Long history of success of MPC in the **process industries**, now spreading to the **automotive** industry



- MPC technology completely ready for mass production:

  1. modern ECUs can solve MPC problems in **real-time**
  2. industry-grade **MPC software** is available for design, calibration, and deployment

- **Key enabler** for adopting MPC: **production managers** that are willing to adopt such a new advanced control technology

- In **software-defined vehicles**, control is an **essential software component**: same hardware + different controls = drastically different performance!

Control innovation is essential for automotive market success