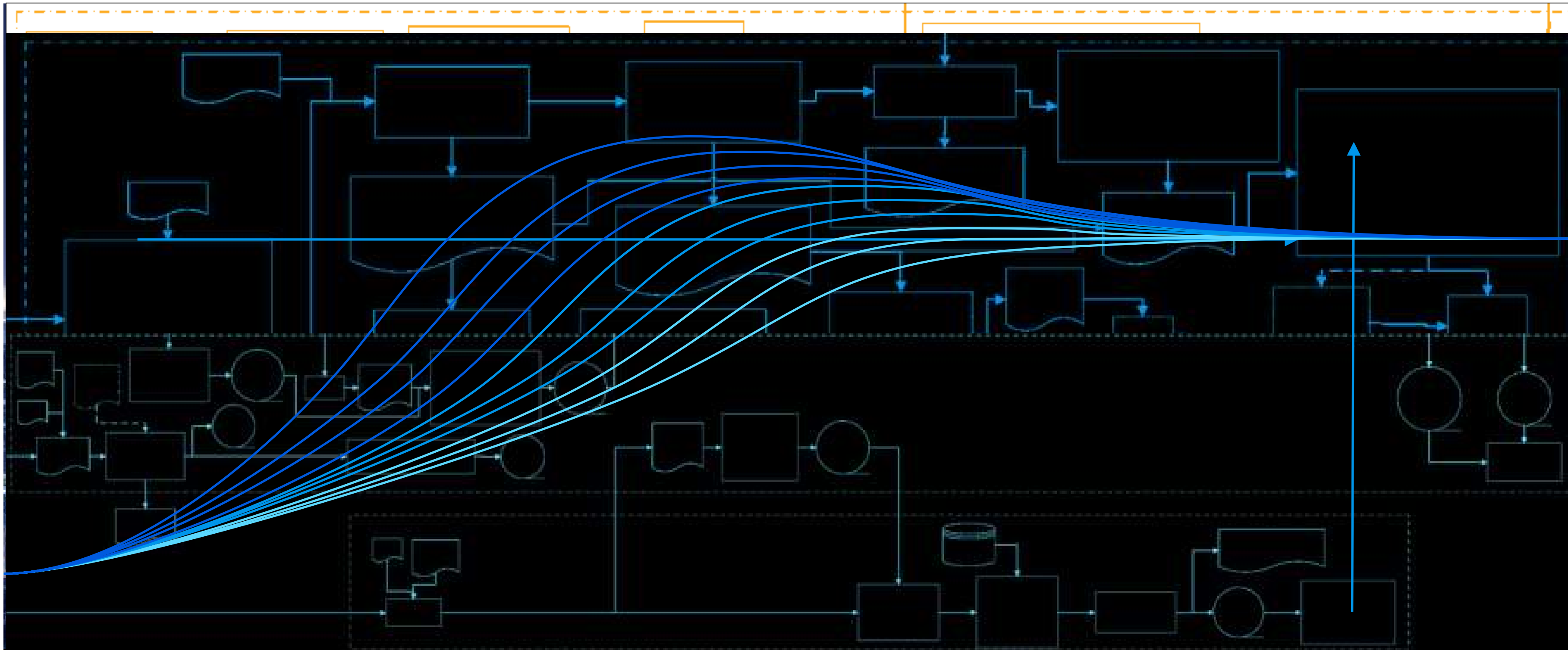


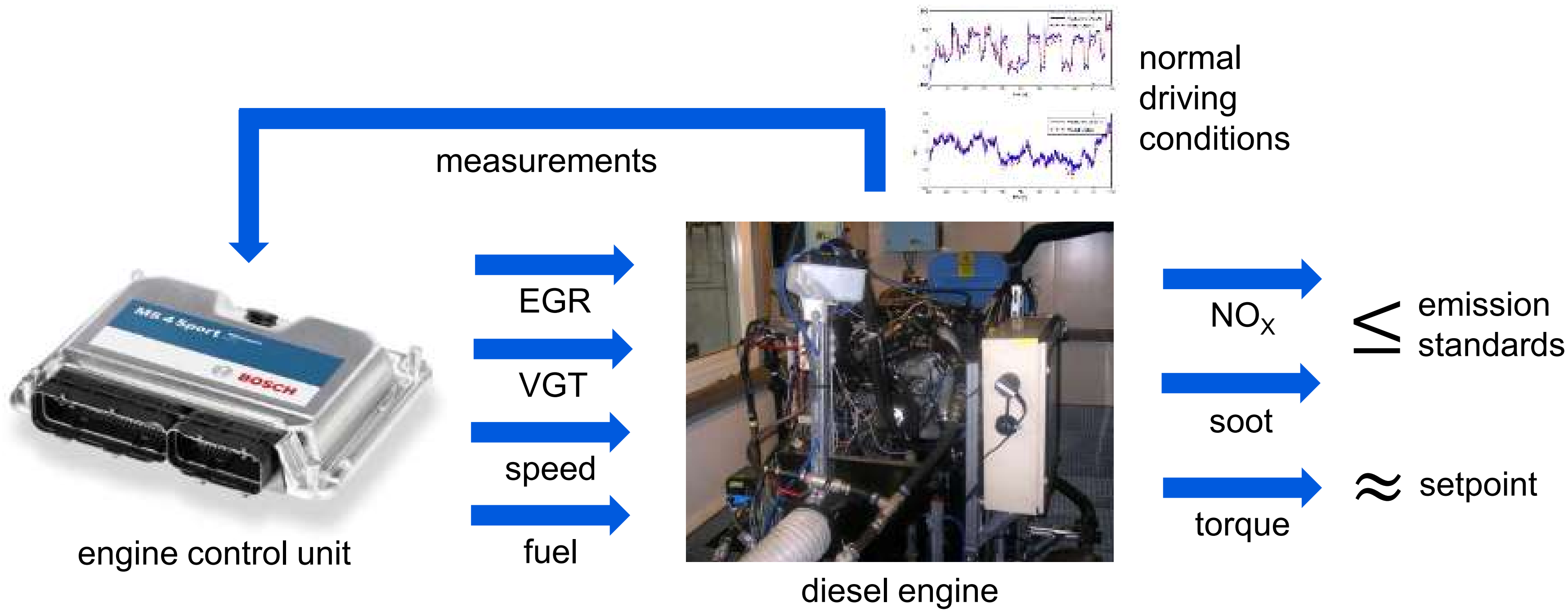
Hans Joachim Ferreau, ABB Corporate Research, 14/4/2016

Embedded Model Predictive Control in Industrial Applications

Power and productivity
for a better world™

ABB

Motivating example of a challenging control problem



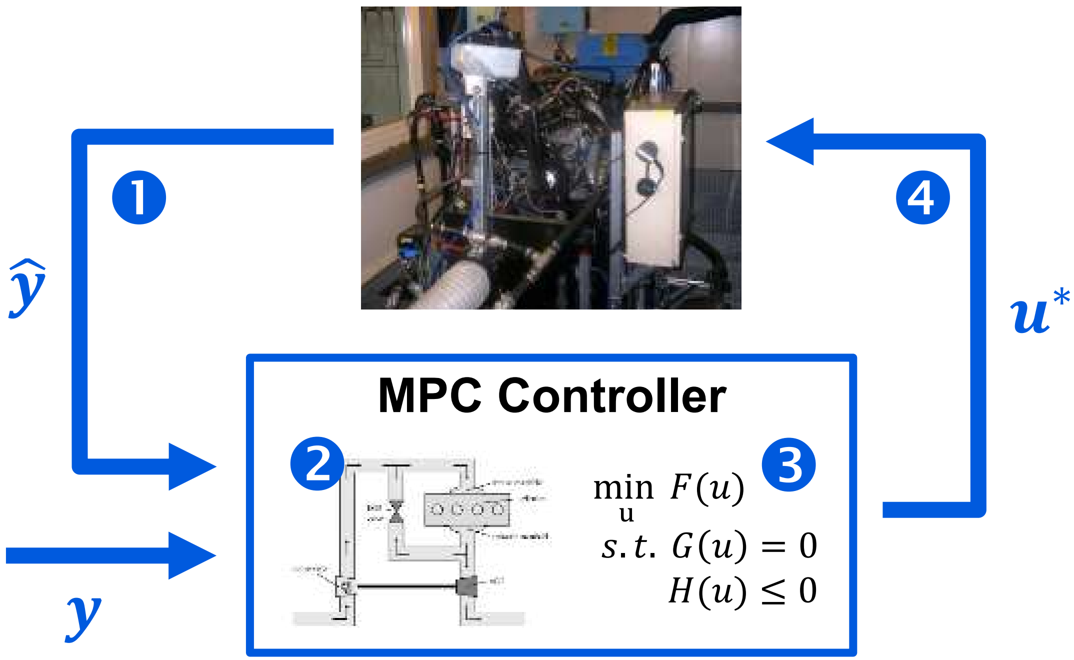
Characteristics:

- **Multiple inputs/outputs**
- **Constraints**
- Nonlinear, coupled dynamics
- Hardly ever in steady-state

Challenges:

- Fast sampling times
- **Limited computational resources**
- Controller has to run **extremely reliable**

Model Predictive Control
on embedded hardware



1) **Measure** or estimate current system state

2) **Predict** future behavior using dynamic model

3) **Optimize** behaviour using optimization algorithm

4) **Apply** optimized inputs to system

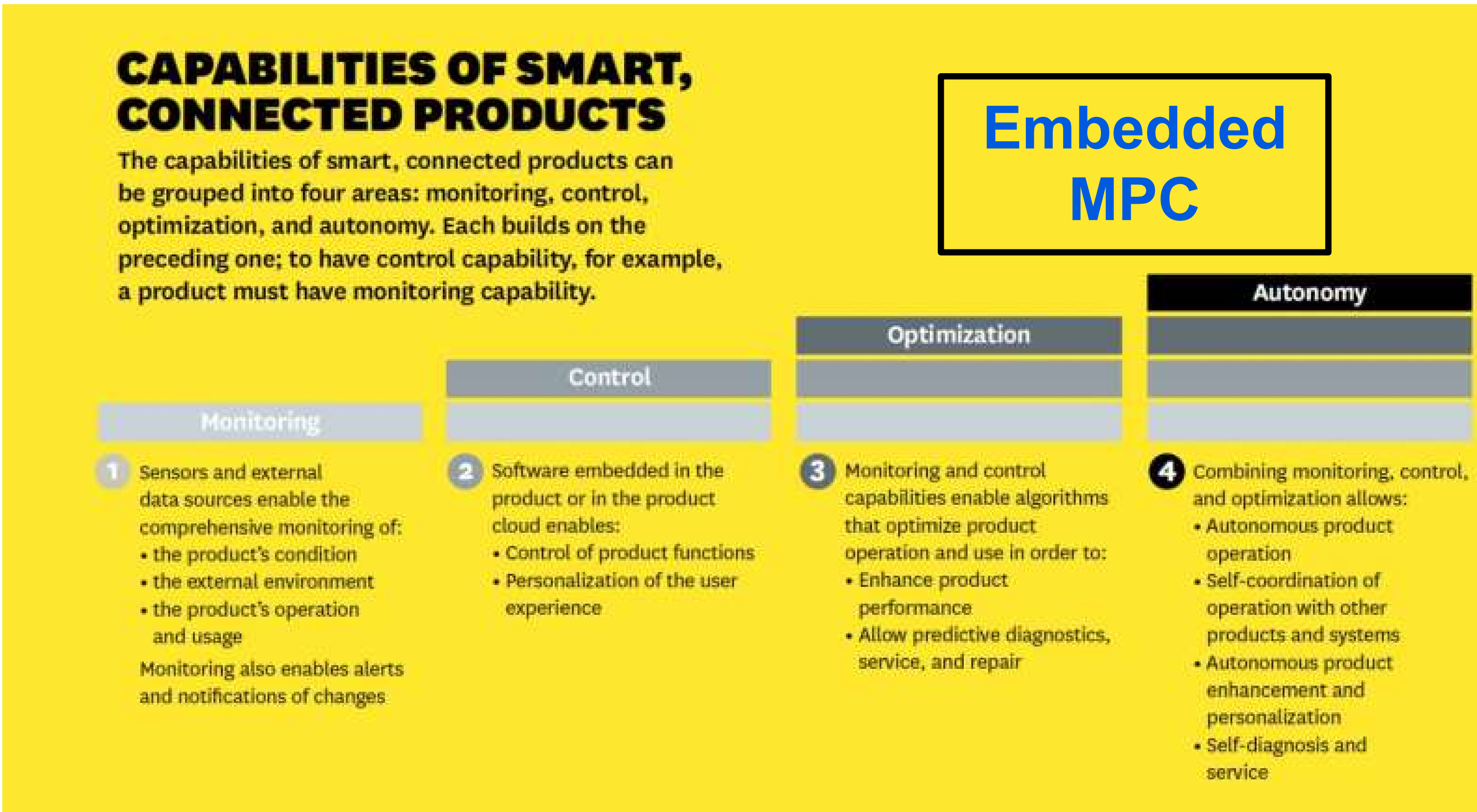
Why embedded?

- Controller hardware highly integrated into product
- Guaranteed **communication latency** (safety critical!)
- Hardware may be much **cheaper** and **more energy-efficient**

What makes Embedded MPC special?

	Server-based Optimization	Embedded Optimization
Reliability	important, but operator can still override controller	**crucial** as typically no user-interaction possible
Computation time	couple of seconds and above	seconds and below (often **millisecond range**)
Software dependencies	easy to link external libraries	**self-contained code** strongly preferred
Memory management	dynamic or static	typically static (or even in hardware)
Number representation	double precision	double/single precision or even fixed-point

Embedded MPC
As part of “Smart” Products



Source: M.E. Porter, J.E. Heppelmann: How Smart, Connected Products Are Transforming Competition, Harvard Business Review, Nov. 2014.

Outline

- ✓ Motivation
- **High-Speed Linear MPC**
- Embedded Nonlinear MPC
- Selected Applications at ABB
- Conclusions

Model Predictive Control
QP and general NLP

Linear OCP:

$$
\begin{aligned}
OCP(x_0):\quad & \min_{x(\cdot),u(\cdot)} \int_{t_0}^{t_0+t_p} x(t)^T Q x(t) + u(t)^T R u(t)\, dt \\
s.t.\quad & x(t_0) = x_0 \\
& \dot{x}(t) = Ax(t) + Bu(t) \quad \forall\, t \in [t_0, t_0+t_p] \\
& 0 \geq Cx(t) + Du(t) \quad \forall\, t \in [t_0, t_0+t_p] \\
& 0 \geq \tilde{C}x(t_0+t_p)
\end{aligned}
$$

Quadratic Program (QP):

$$
\begin{aligned}
QP(x_0):\quad & \min_{z} \tfrac{1}{2} z^T H z + z' g \\
& s.t.\ Bz = b(x_0) \\
& Az \leq a
\end{aligned}
$$

Nonlinear OCP:

$$
\begin{aligned}
OCP(x_0):\quad & \min_{x(\cdot),u(\cdot)} \int_{t_0}^{t_0+t_p} J\big(x(t), u(t)\big)\, dt + P\Big(x\big(t_0+t_p\big)\Big) \\
s.t.\quad & x(t_0) = x_0 \\
& \dot{x}(t) = f\big(x(t), u(t)\big) \quad \forall\, t \in [t_0, t_0+t_p] \\
& 0 \geq c\big(x(t), u(t)\big) \quad \forall\, t \in [t_0, t_0+t_p] \\
& 0 \geq \tilde{c}\Big(x\big(t_0+t_p\big)\Big)
\end{aligned}
$$

Nonlinear Program (NLP):

$$
\begin{aligned}
NLP(x_0):\quad & \min_{z}\ F(z) \\
& s.t.\ \ G(z, x_0) = 0 \\
& H(z) \leq 0
\end{aligned}
$$

(or MINLP in case of binary variables)

Embedded Linear MPC
The quest for fast and reliable solvers

- Embedded applications have triggered major academic efforts to develop **highly efficient solvers:**

First-order	gradient method, primal FGM, dual FGM, GPAD, FiOrdOs
Active-set	quadprog (primal), QLD (dual), qpOASES (primal-dual)
Interior-point	primal barrier, CVXGEN (primal-dual), FORCES (primal-dual), HPMPC
Others	PQP, qpDUNES (Newton-type), ADMM, MPT (explicit MPC)

- Best choice is **highly problem-dependent** due to:
  - **numerical properties** of MPC formulation
  - **implementation aspects** (e.g. target hardware)

Numerical Properties
Sparse vs. dense QP formulation

- MPC leads to **specially-structured QP problems:**
  - specific sparsity pattern
  - parametric dependency

- How to **exploit problem sparsity**?

a) Using sparse solver:

$$QP_s(x_0):\ \min_z\ \tfrac{1}{2} z' H_s z + z' g_s$$
$$s.t.\ \ B_s z = b_s(x_0)$$
$$A_s z \le a_s$$

or

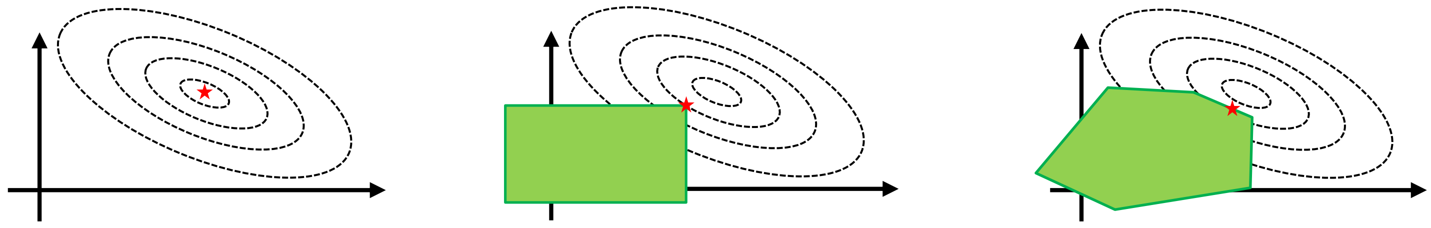
b) Eliminate states:

$$QP_d(x_0):\ \min_z\ \tfrac{1}{2} z' H_d z + z' g_d(x_0)$$
$$s.t.\ \ A_d z \le a_d(x_0)$$

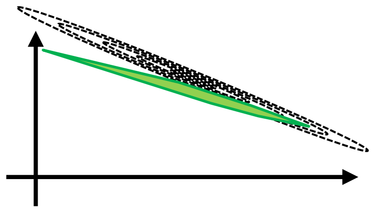
- Parametric dependency can be exploited by **warm-starts**

Numerical Properties
Constraints and objective functions

- **Inequality constraints** are a main reason to use MPC… and they also **make solving the QP more demanding**



- **Badly conditioned QP problems** (due to unstable dynamics, scaling, etc.)



- What if QP problem becomes **infeasible?**
- Some methods cannot handle **semi-definite objective**

Implementation Aspects

- **Reliability is key!** (find a sufficiently accurate solution in time)
- Is **computation time** constant or (strongly) varying, predictable, bounded, or unknown? In any case, short enough?
- Do **warm-starts** help? (average vs. worst-case execution time)
- **Code size**, programming language, software dependencies, memory management
- Suitability for **parallel execution** on multi-core (or even hybrid) architectures
- Suitability for fixed-point implementation (e.g. on FPGA)

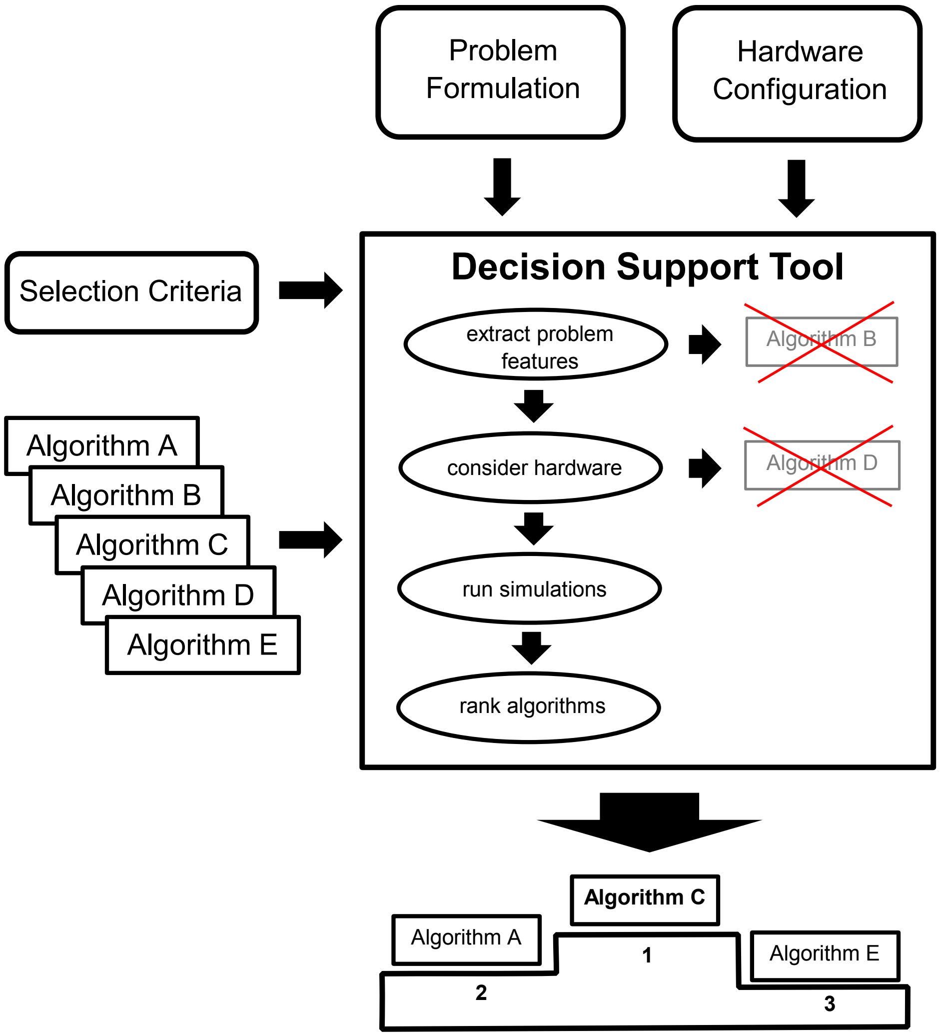
Existing Linear MPC Algorithms
A rough overview

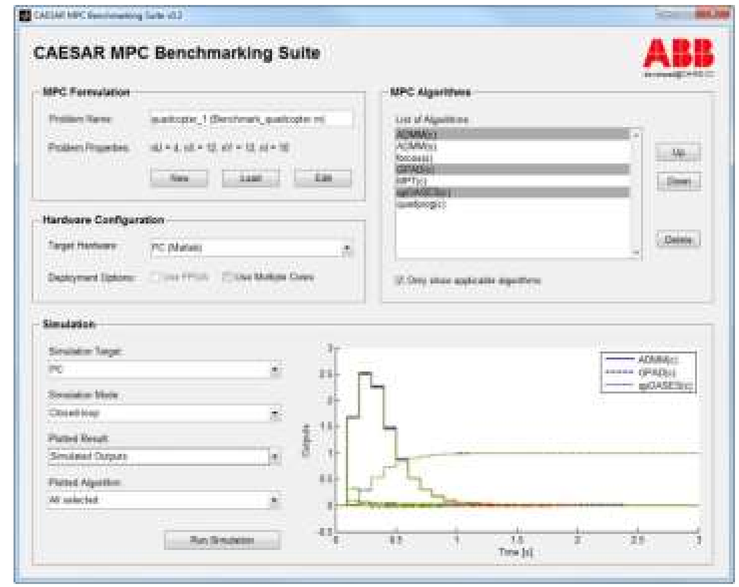
- **First-order methods:**
  - compute step towards solution of unconstrained QP
  - project to feasible set (difficult for general constraints)
- **Active-set methods:**
  - guess which inequalities hold with equality at solution
  - solve resulting equality-constrained QP (almost trivial)
  - check if guess was correct, update guess if not
- **Interior-point methods:**
  - remove inequalities, but penalize constraint violations in objective function (non-quadratic term, e.g. logarithmic)
  - solve resulting equality-constrained NLP with Newton's method
- **Explicit methods and others**

Linear MPC can run reliably
at kHz sampling times
even on embedded platforms!

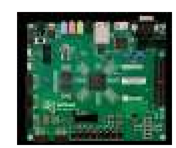
How to choose the algorithm?

Decision Support Tool for MPC
Benchmarking and ranking





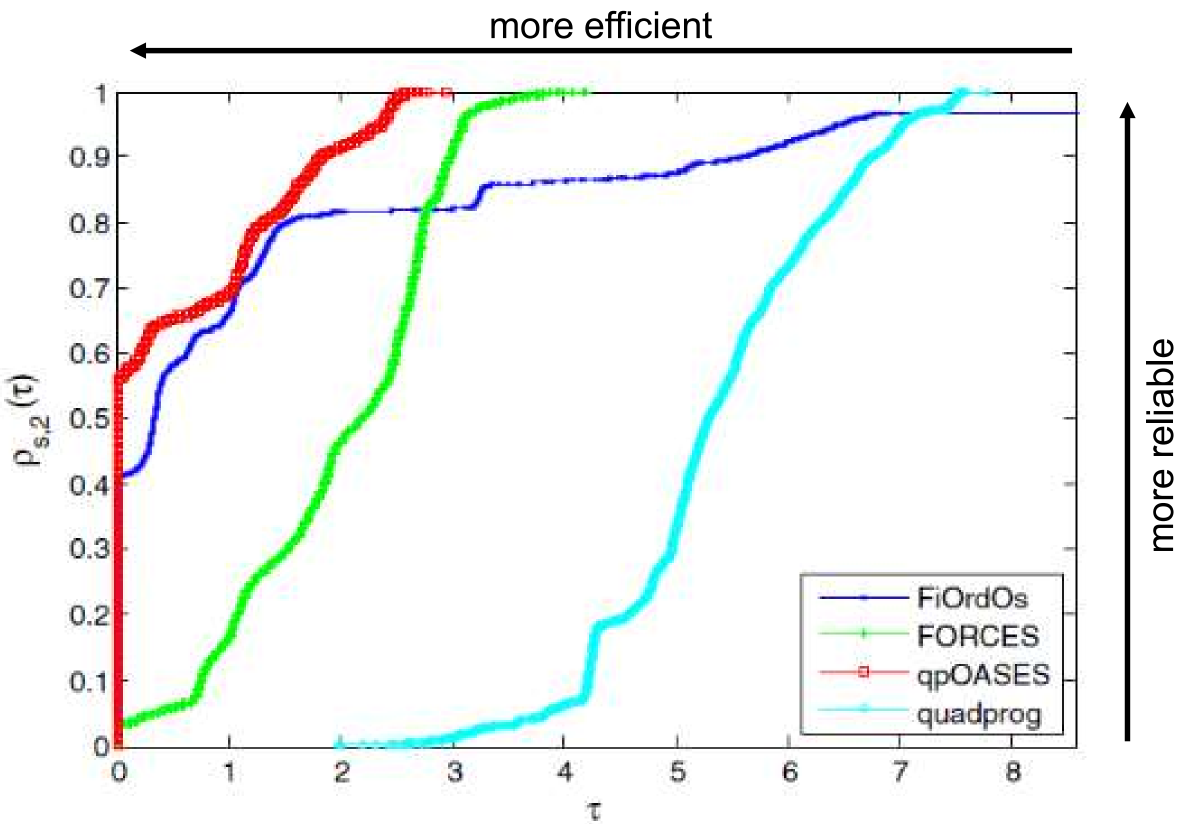
- Matlab-based tool
- **Compares up to 12 algorithms on PC, PEC2, PEC3, Xilinx' Zynq**



(joint work with Helfried Peyrl)

MPC Benchmarking Suite
Illustrative results: speed

- Overall **computational performance on 14 MPC benchmark examples:**
  - > 2500 QP instances
  - 2 - 12 states
  - 1 - 4 control inputs
  - 3 - 100 intervals
  - different constraints

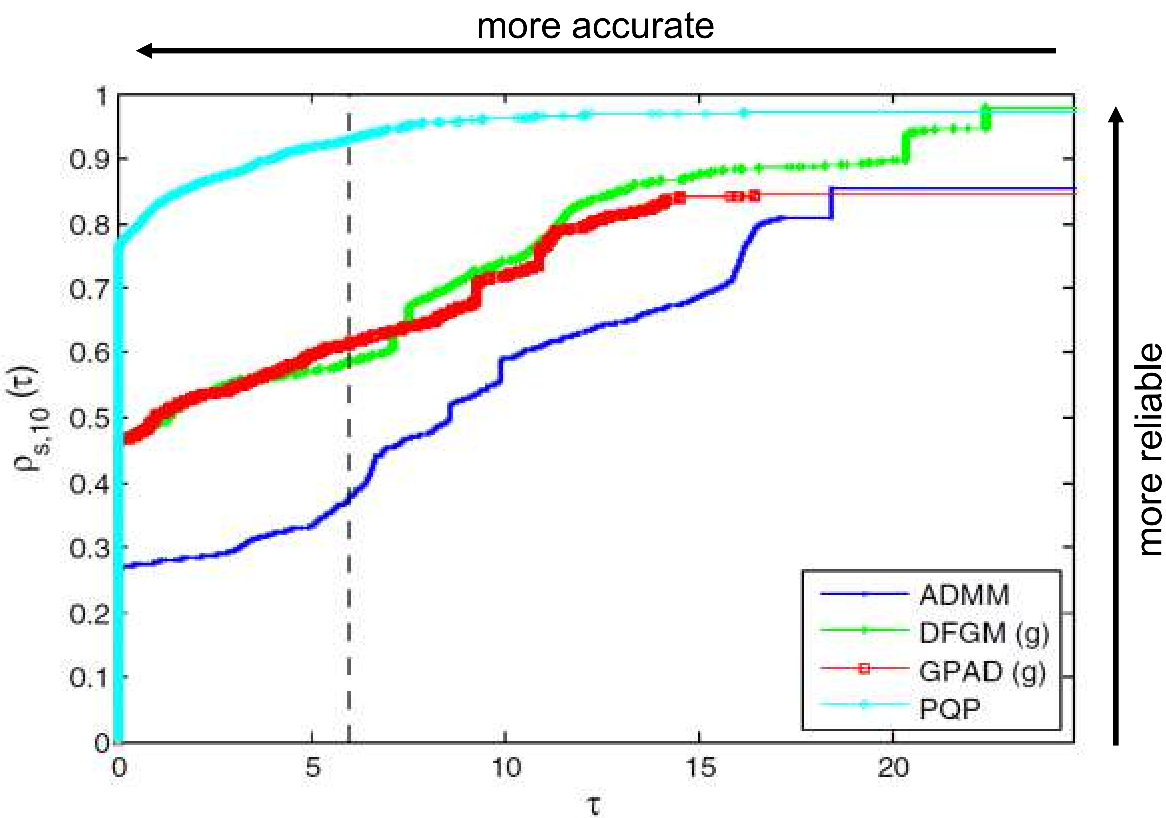


see Kouzoupis, Zanelli, Peyrl, Ferreau (2015)

- *Remarks:*
  - solver-specific termination criterion and default options
  - no warm-starts

MPC Benchmarking Suite
Illustrative results: accuracy

- Comparing **accuracy of “first-order methods“** with fixed number of iterations:
  - $\tau = 0$ defined as relative error less or equal than 10^{-8}

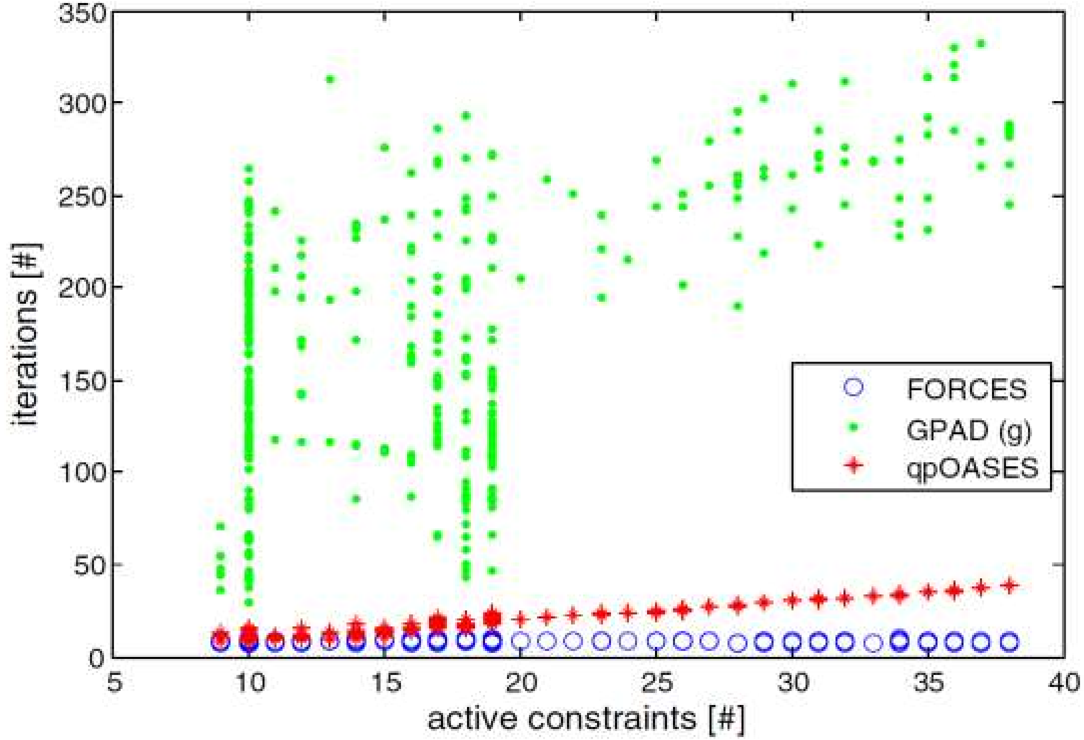


see Kouzoupis, Zanelli, Peyrl, Ferreau (2015)

- *Remark:*
  - no problem-specific tuning for ADMM

MPC Benchmarking Suite
Illustrative results: #iterations

- Comparing **number of iterations vs. number of active constraints** for a specific example:



- *Remark:*
  - qpOASES performing cold-starts

see Kouzoupis, Zanelli, Peyrl, Ferreau (2015)

qpOASES
An Implementation of the **O**nline **A**ctive **SE**t **S**trategy

- qpOASES solves QP problems of the following form:

$$\min_{z} \ \tfrac{1}{2} z' H z + z' g(x_0)$$
$$s.t. \ \ \underline{b}(x_0) \le z \ \le \bar{b}(x_0)$$
$$\underline{c}(x_0) \le Az \le \bar{c}(x_0)$$

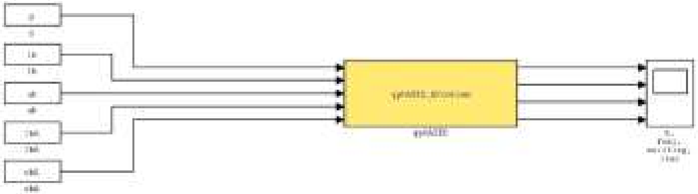
- **C/C++ implementation** with dense linear algebra, developed since 2007 *see e.g. Ferreau, Kirches, Potschka, Bock, Diehl (2014)*
- **Reliable and efficient** for solving small- to medium-scale QPs (when states have been eliminated from MPC problem)
- **Self-contained code** (optionally, LAPACK/BLAS can be linked)
- Distributed as **open-source software** (GNU LGPL), download at: **https://projects.coin-or.org/qpOASES**

qpOASES
Interfaces and Applications

- Matlab / Octave / Scilab

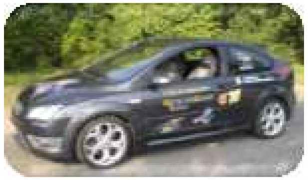
```
[x,fval,exitflag,iter,lambda] = qpOASES( H,g,A,lb,ub,lbA,ubA )
```

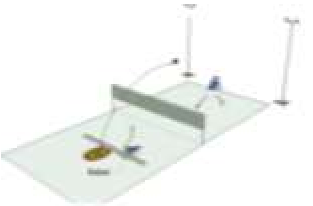
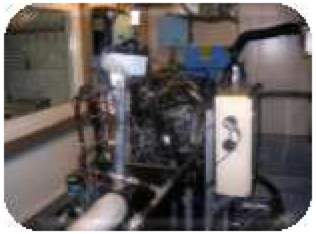
- Simulink (dSPACE / xPC Target)



A few applications:





Outline

- ✓ Motivation
- ✓ High-Speed Linear MPC
- **Embedded Nonlinear MPC**
- Selected Applications at ABB
- Conclusions

Model Predictive Control
QP and general NLP

Linear OCP:

$$
\begin{aligned}
OCP(x_0):\quad & \min_{x(\cdot),u(\cdot)} \int_{t_0}^{t_0+t_p} x(t)^T Q x(t) + u(t)^T R u(t)\, dt \\
s.t.\quad & x(t_0) = x_0 \\
& \dot{x}(t) = Ax(t) + Bu(t) \quad \forall\, t \in [t_0, t_0 + t_p] \\
& 0 \geq Cx(t) + Du(t) \quad \forall\, t \in [t_0, t_0 + t_p] \\
& 0 \geq \tilde{C}x(t_0 + t_p)
\end{aligned}
$$

Quadratic Program (QP):

$$
\begin{aligned}
QP(x_0):\quad & \min_{z} \tfrac{1}{2} z^T H z + z' g \\
& s.t.\ Bz = b(x_0) \\
& Az \leq a
\end{aligned}
$$

Nonlinear OCP:

$$
\begin{aligned}
OCP(x_0):\quad & \min_{x(\cdot),u(\cdot)} \int_{t_0}^{t_0+t_p} J\big(x(t), u(t)\big)\, dt + P\left(x\big(t_0 + t_p\big)\right) \\
s.t.\quad & x(t_0) = x_0 \\
& \dot{x}(t) = f\big(x(t), u(t)\big) \quad \forall\, t \in [t_0, t_0 + t_p] \\
& 0 \geq c\big(x(t), u(t)\big) \quad \forall\, t \in [t_0, t_0 + t_p] \\
& 0 \geq \tilde{c}\left(x\big(t_0 + t_p\big)\right)
\end{aligned}
$$

Nonlinear Program (NLP):

$$
\begin{aligned}
NLP(x_0):\quad & \min_{z}\ F(z) \\
& s.t.\ G(z, x_0) = 0 \\
& H(z) \leq 0
\end{aligned}
$$

Solution Methods for Nonlinear MPC
Direct methods

- **Direct methods** first replace the continuous control input trajectory $u(t)$ by a finite-dimensional parameterization U
- Typically a **piecewise constant control parameterization** is used (on a partition $t_0 < t_1 < \ldots < t_{N-1} = t_0 + t_p$):

$$U = (u_0, u_1, \ldots, u_{N-1}) = \big(u(t_0), u(t_1), \ldots, u(t_{N-1})\big)$$

$$\textbf{OCP} \xrightarrow{\textbf{discretize}} \textbf{NLP} \xrightarrow{\textbf{optimize}} \textbf{Solution}$$

- The way the states are discretized leads to different variants:
  - **single shooting** (sequential approach)
  - **multiple shooting**
  - **collocation** (simultaneous approach)

Direct Methods for Nonlinear MPC
Solving the NLP

- NLPs can be solved efficiently using Newton-type methods:
  - **Interior-Point methods** (e.g. IPOPT)
  - **Sequential Quadratic Programming**

IP Methods:

✓ rather constant runtime

✓ easy to exploit sparsity

✗ difficult to warm-start

✗ need 2nd order derivatives

SQP Methods:

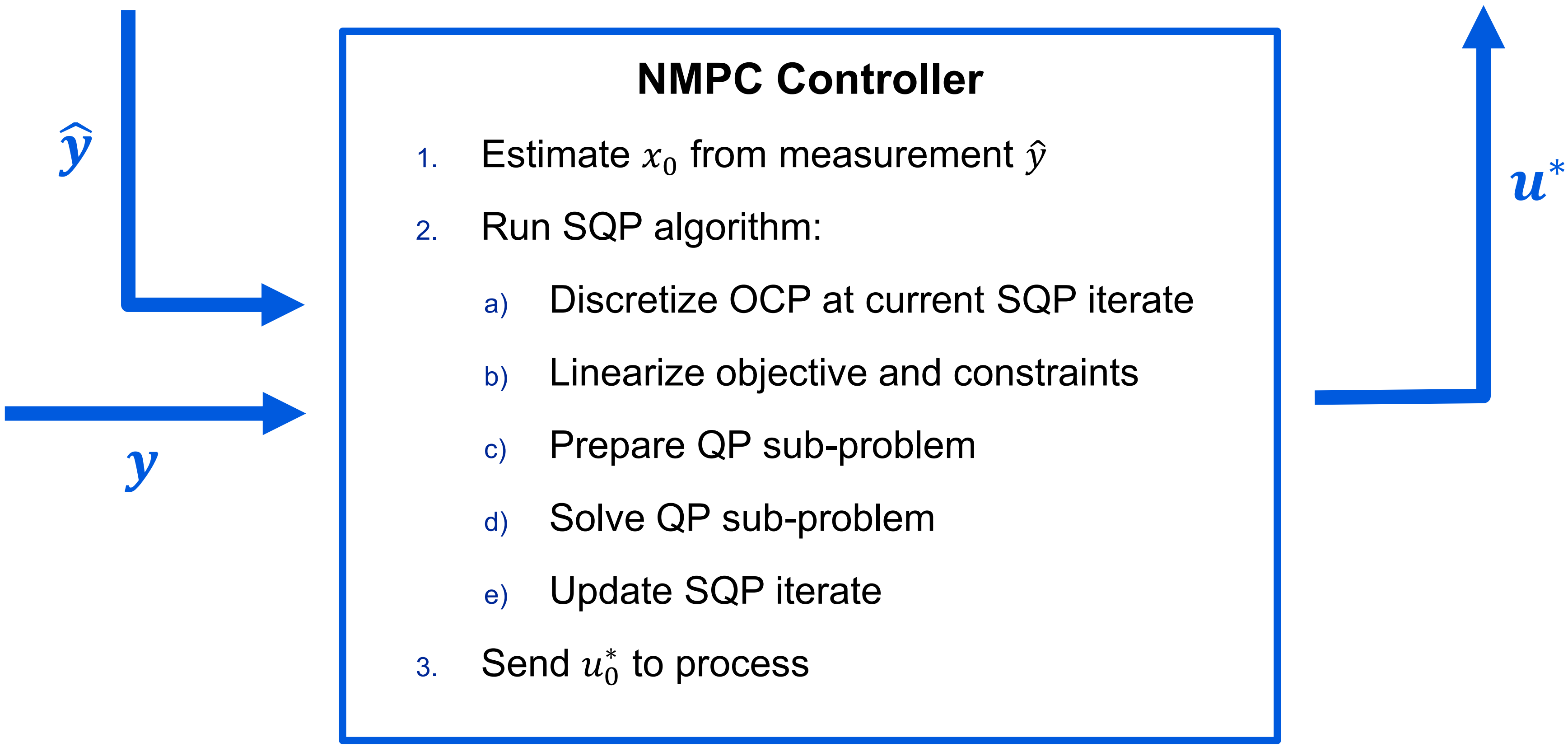
✓ rely on solving QPs

✓ easy to warm-start

✓ 1st order derivatives enough

✗ more variable runtime

SQP Algorithm for Nonlinear MPC

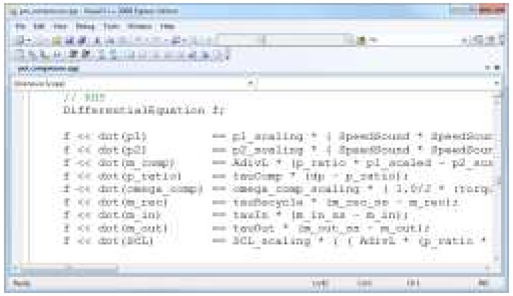


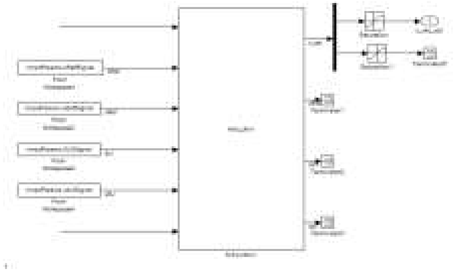
- **Real-time iteration scheme:** only perform one iteration of a full-step Gauss-Newton SQP scheme

  see Diehl (2001), Diehl et al. (2002)

SQP Algorithm for Nonlinear MPC
ACADO Toolkit

- ACADO Code Generation:
  see Houska, Ferreau, Diehl (2011)
  - Takes **symbolic NMPC problem formulation** in C++ or Matlab
  - **Auto-generates efficient, customized and self-contained C code** implementing SQP algorithm for NMPC
  - Compiles NMPC algorithm into **Simulink S function**





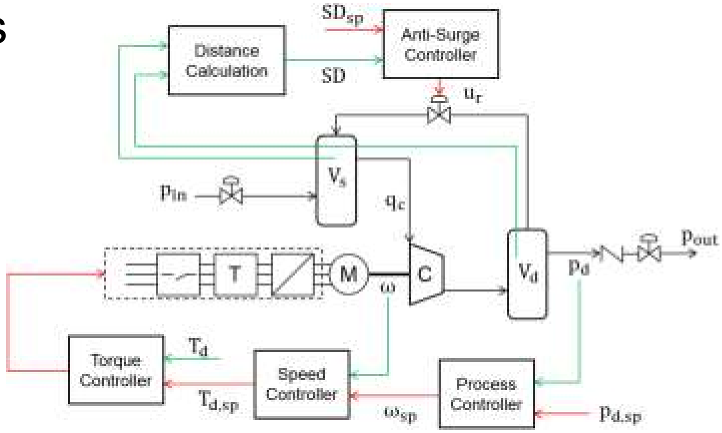
- Developed since 2009 at KU Leuven (now U Freiburg)
- Open-source: www.acadotoolkit.org

Outline

- ✓ Motivation
- ✓ High-Speed Linear MPC
- Embedded Nonlinear MPC
- **Selected Applications at ABB**
- Conclusions

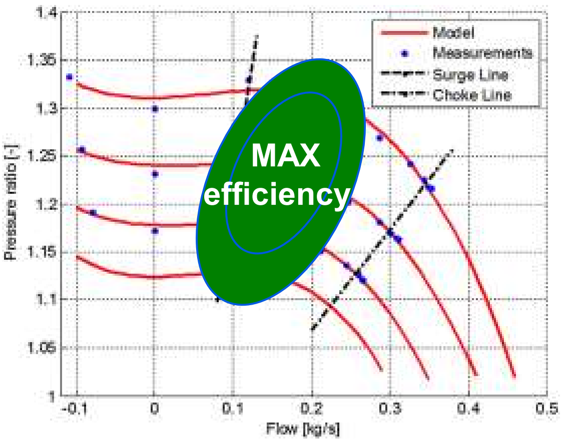
MPC for Compressor Control
Challenges

- Up to 97% of compressor lifetime costs are engergy costs



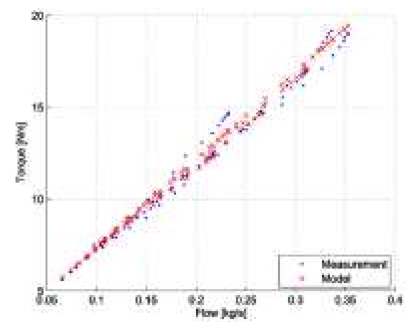
- **Goal: Combined anti-surge and process control** to operate gas compressors more efficiently

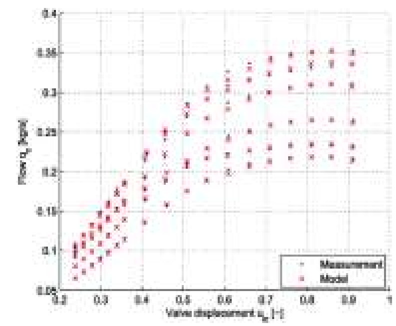
- **Challenges:**
  - **Nonlinear, coupled dynamics**
  - Time delays
  - **Safety critical**
  - Millisecond sampling times



MPC for Compressor Control
Tests at PLCRC

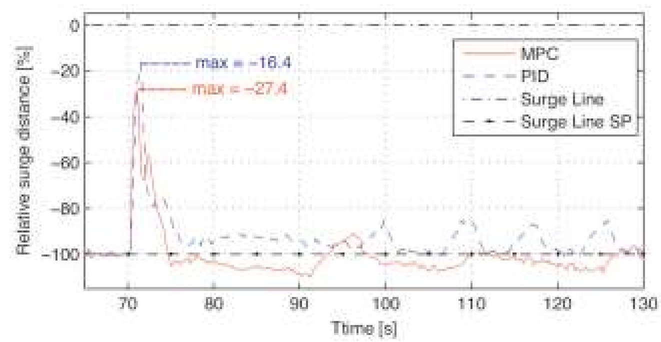
- **Setup:**
  - Compressor test rig with 15kW variable-speed drive
  - Identified nonlinear grey-box model with 5 states
  - Linearized **MPC algorithm** using qpOASES
  - Kalman filter for state estimation
  - Running with **50ms sampling time on AC 800PEC**





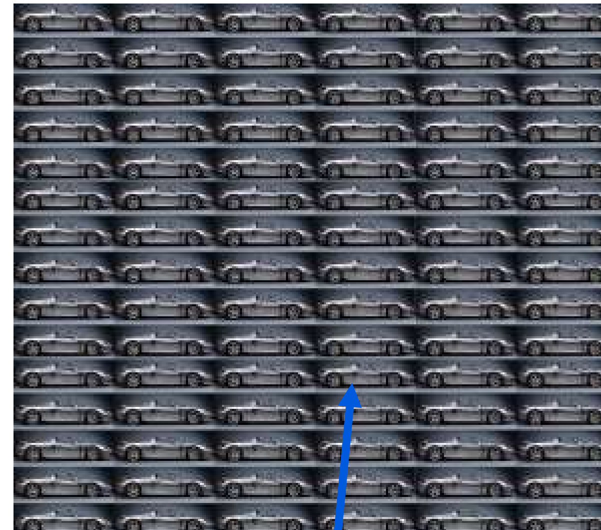
- **Results:**
  - **10% more distance to surge**
  - **50% faster process control**

  see Cortinovis, Ferreau, Lewandowski, Mercangöz (2015)



NMPC for Load Commutated Inverters
48 Megawatt at 1 kHz

=



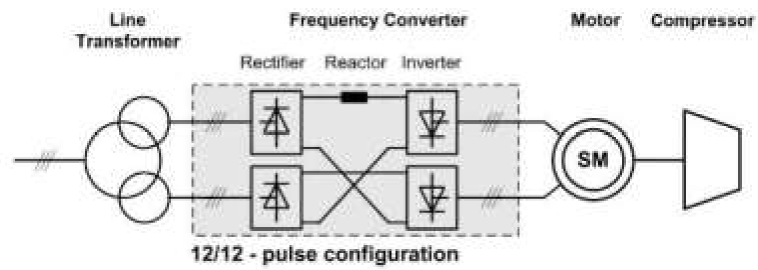
Thomas Besselmann

Synchronous machine
(48 Megawatt)

106 Porsche Carrera GTs
(48 MW at full-throttle)

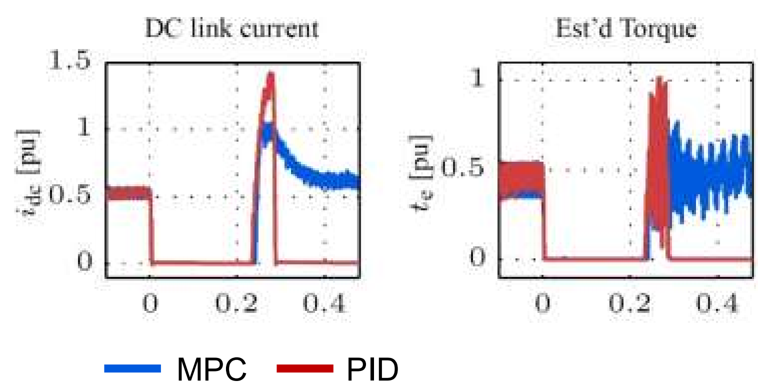
NMPC for Load Commutated Inverters
48 Megawatt at 1kHz

- Load commutated inverters (LCIs) play an important role in powering electrically-driven compressor stations
- **Goal:** Enable **LCIs to ride through partial loss of grid voltage**



- **Solution:**
  - **Auto-generated NMPC algorithm** (ACADO/qpOASES)
  - Running at **1kHz on AC 800PEC**
- **Results:**
  - Successfully tested **on a 48 MW pilot plant installation**
  - **Works where PID solution fails!**

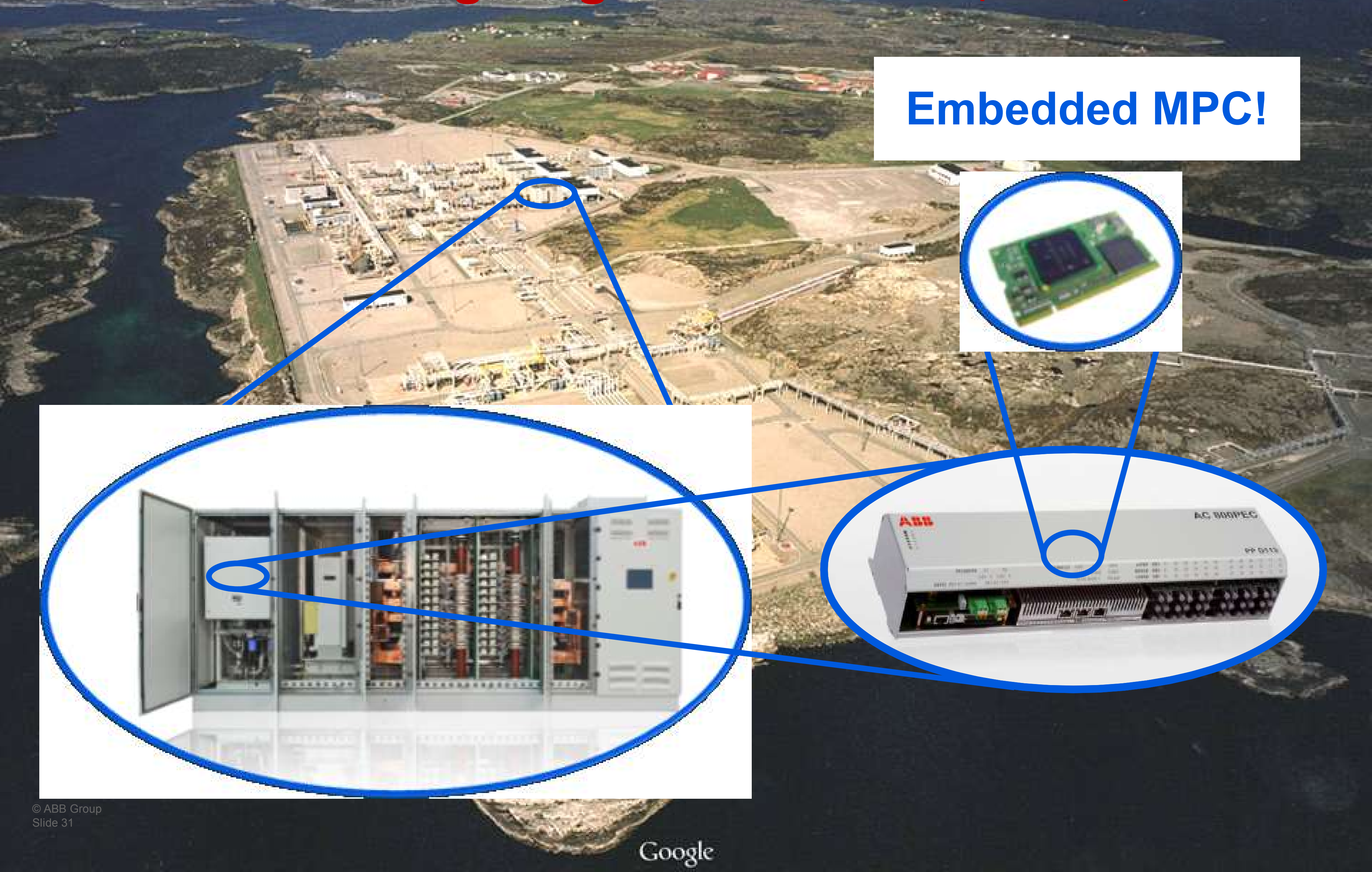
see Besselmann, Van de moortel, Almer, Jörg, Ferreau (2016)



«Kollsnes accounts for more than 40% of all Norwegian gas deliveries» *(Gassco)*

Illustration: Statoil

«Kollsnes accounts for more than 40% of all Norwegian gas deliveries» *(Gassco)*



Outline

- ✓ Motivation
- ✓ High-Speed Linear MPC
- ▪ Embedded Nonlinear MPC
- ✓ Selected Applications at ABB
- ▪ **Conclusions**

Conclusions

- **MPC can run reliably at millisecond sampling times,** even on embedded controller hardware
- If numerical performance is crucial, **care must be taken to choose the most appropriate implementation**
- **Many more applications** may benefit from embedded MPC (enabling to “smart” products)

Power and productivity
for a better world™

ABB