Nonlinear Optimal Control for Embedded Applications

Moritz Diehl
Systems Control and Optimization Laboratory
Department of Microsystems Engineering (IMTEK) &
Department of Mathematics
University of Freiburg

CORTOPT Aalborg,
August 22-23, 2016

Complex Sensor Actuator Systems

SENSORS

- GPS
- acceleration
- radar
- vision
- ...

How to connect ?

ACTUATORS

- flight surfaces
- steering wheel
- motor speeds
- joint torques
- ...



Embedded Optimization

Solve, in real-time and repeatedly, an optimization problem that depends on the incoming stream of input data, to generate a stream of output data.

Embedded Optimization

Solve, in real-time and repeatedly, an optimization problem that depends on the incoming stream of input data, to generate a stream of output data.

Example: Parametric Quadratic Programming

x

$$u = \arg\min_{u} \begin{bmatrix} x \\ u \end{bmatrix}^T \begin{bmatrix} Q & S \\ S^T & R \end{bmatrix} \begin{bmatrix} x \\ u \end{bmatrix} + \begin{bmatrix} x \\ u \end{bmatrix}^T \begin{bmatrix} g \\ h \end{bmatrix} \quad \text{s.t. } Au + Bx \leq b$$

u

Embedded Optimization: a CPU-Intensive Map

x → EMBEDDED OPTIMIZATION → u

$$u = \arg\min_{u} \begin{bmatrix} x \\ u \end{bmatrix}^T \begin{bmatrix} Q & S \\ S^T & R \end{bmatrix} \begin{bmatrix} x \\ u \end{bmatrix} + \begin{bmatrix} x \\ u \end{bmatrix}^T \begin{bmatrix} g \\ h \end{bmatrix} \quad \text{s.t. } Au + Bx \leq b$$

Surprisingly powerful!
Nearly every map of interest can be generated by embedded convex optimisation…

The ubiquity of parametric convex optimization

THEOREM [Baes, D., Necoara, 2008]
Every continuous map

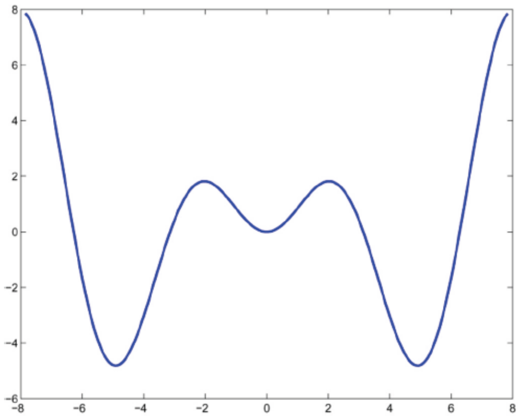
$$\begin{aligned} \mu : \mathbb{R}^{n_x} &\rightarrow \mathbb{R}^{n_u} \\ x &\mapsto u = \mu(x) \end{aligned}$$

can be represented as parametric convex program (PCP):

$$\mu(x) = \arg\min_{u} g(u, x) \quad \text{s.t.} \quad (u, x) \in \Gamma$$

PCP: objective and feasible set jointly convex in parameters and variables (x, u).

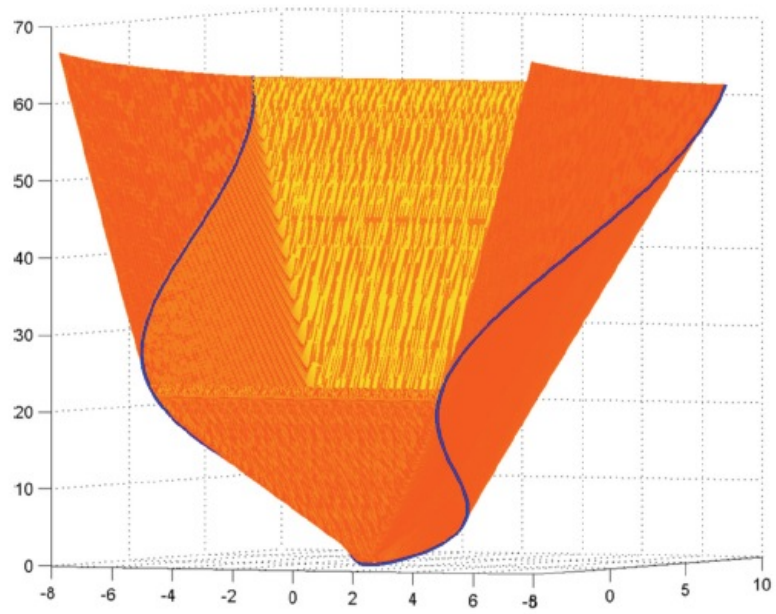
(Sketch of Proof)



Given: graph of $\mu(x)$

Construct epigraph E of $g(u, x)$

1. "Bend" graph of $\mu(x)$ using strictly convex $g^0(x)$
2. Add upward rays.
3. Take convex hull.
4. Show that minima are preserved.



$$
\begin{aligned}
S &:= \{(x, \mu(x), t) | x \in \Omega, g^0(x) \leq t\} \\
E &:= \operatorname{conv}(S)
\end{aligned}
$$

Overview

- Embedded Optimization
- **Time Optimal Motions in Mechatronics**
- Real-Time Optimization Methods and Software
- Four Experimental NMPC Applications

Time-Optimal Point-To-Point Motions [PhD Vandenbrouck 2012]

Fast oscillating systems (cranes, plotters, wafer steppers, …)

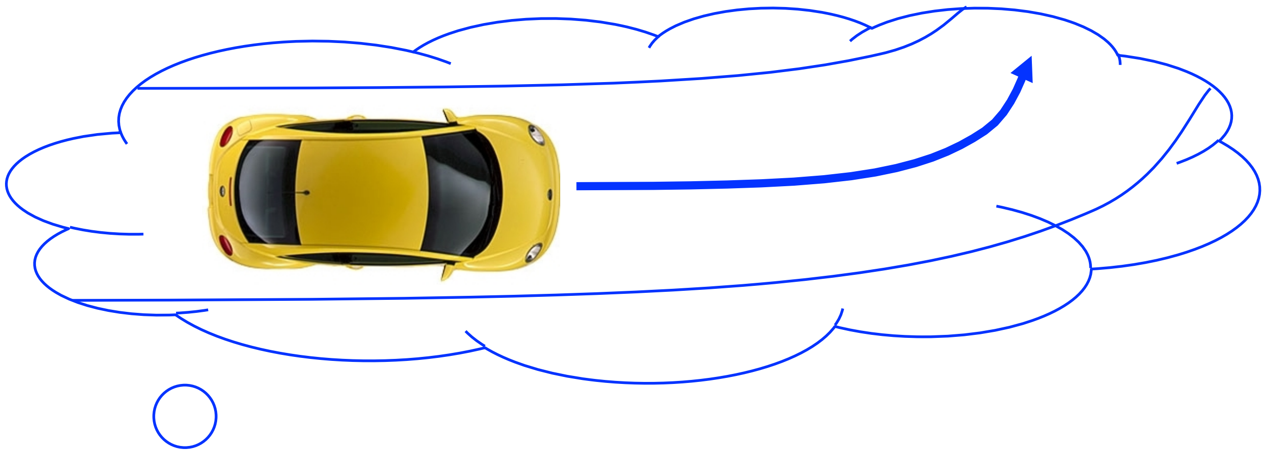
Control aims:

- reach end point as fast as possible
- do not violate constraints
- no residual vibrations

Idea: formulate as embedded optimization problem
in form of Model Predictive Control (MPC)

Model Predictive Control (MPC)

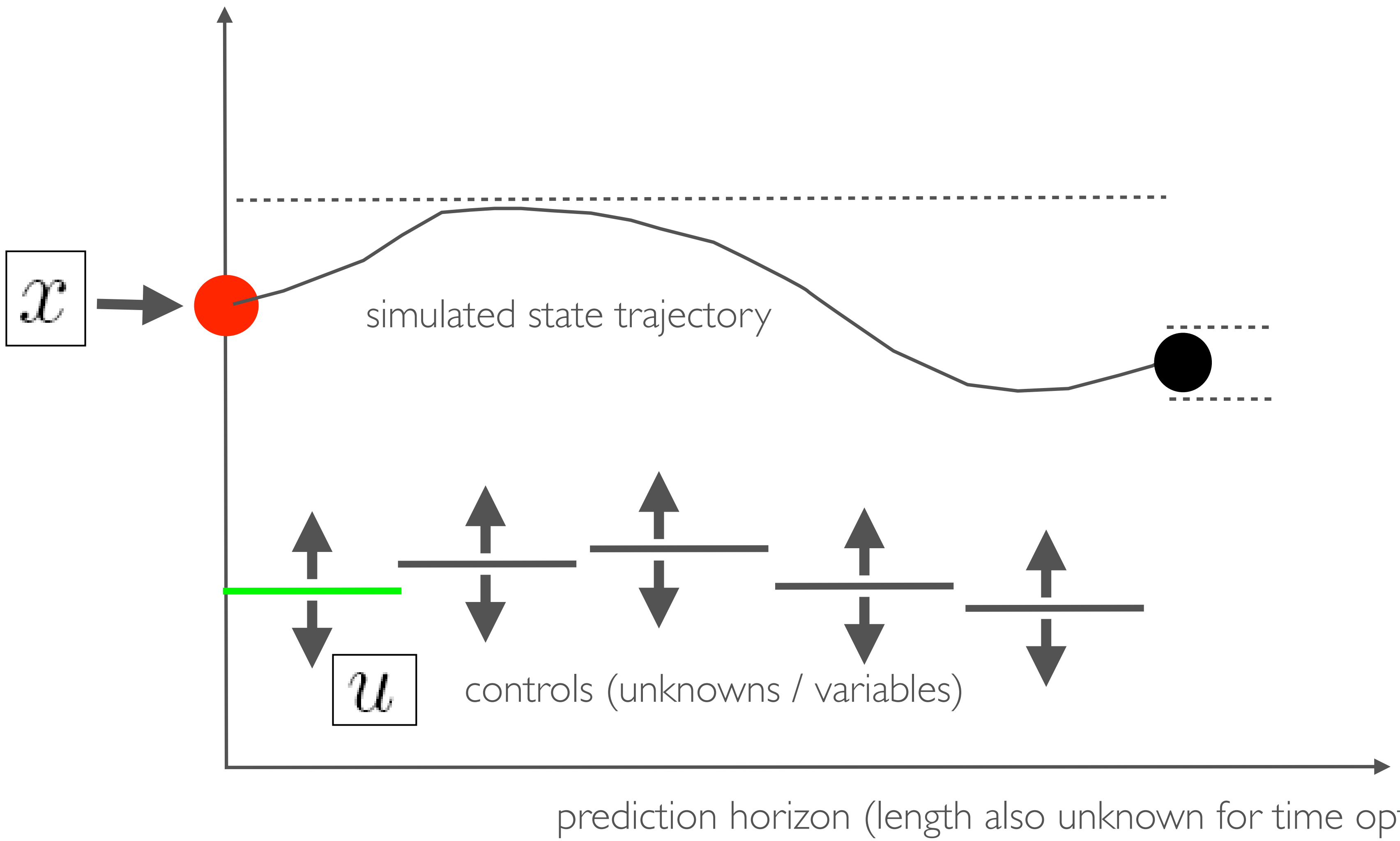
Always look a bit into the future



Example: driver predicts and optimizes, and therefore slows down before a curve

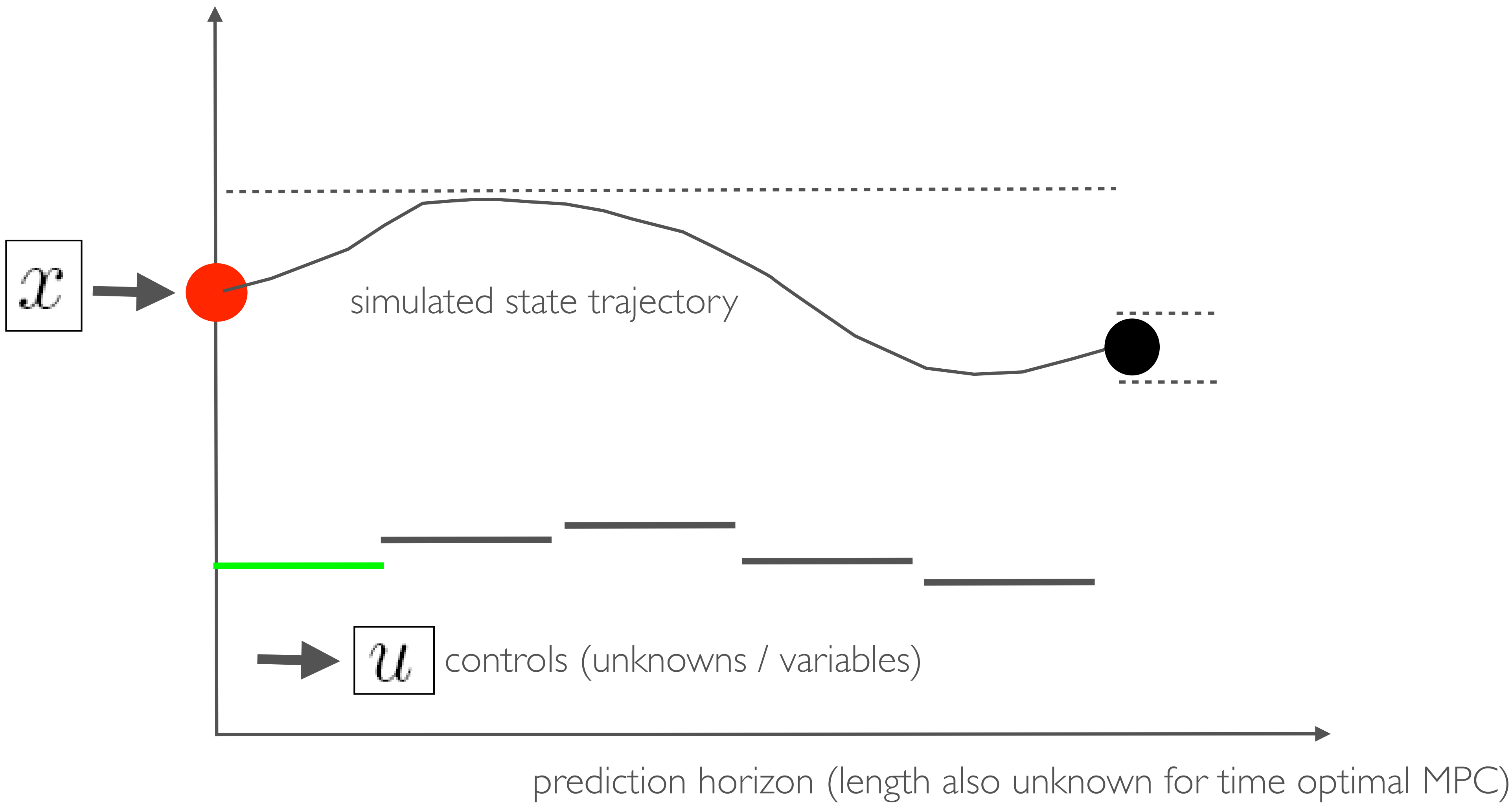
Optimal Control Problem in MPC

For given system state ***x***, which controls ***u*** lead to the best objective value without violation of constraints ?



Optimal Control Problem in MPC

For given system state ***x***, which controls ***u*** lead to the best objective value without violation of constraints ?



Time Optimal MPC of a Crane

SENSORS

- line angle
- cart position

→

MPC

→

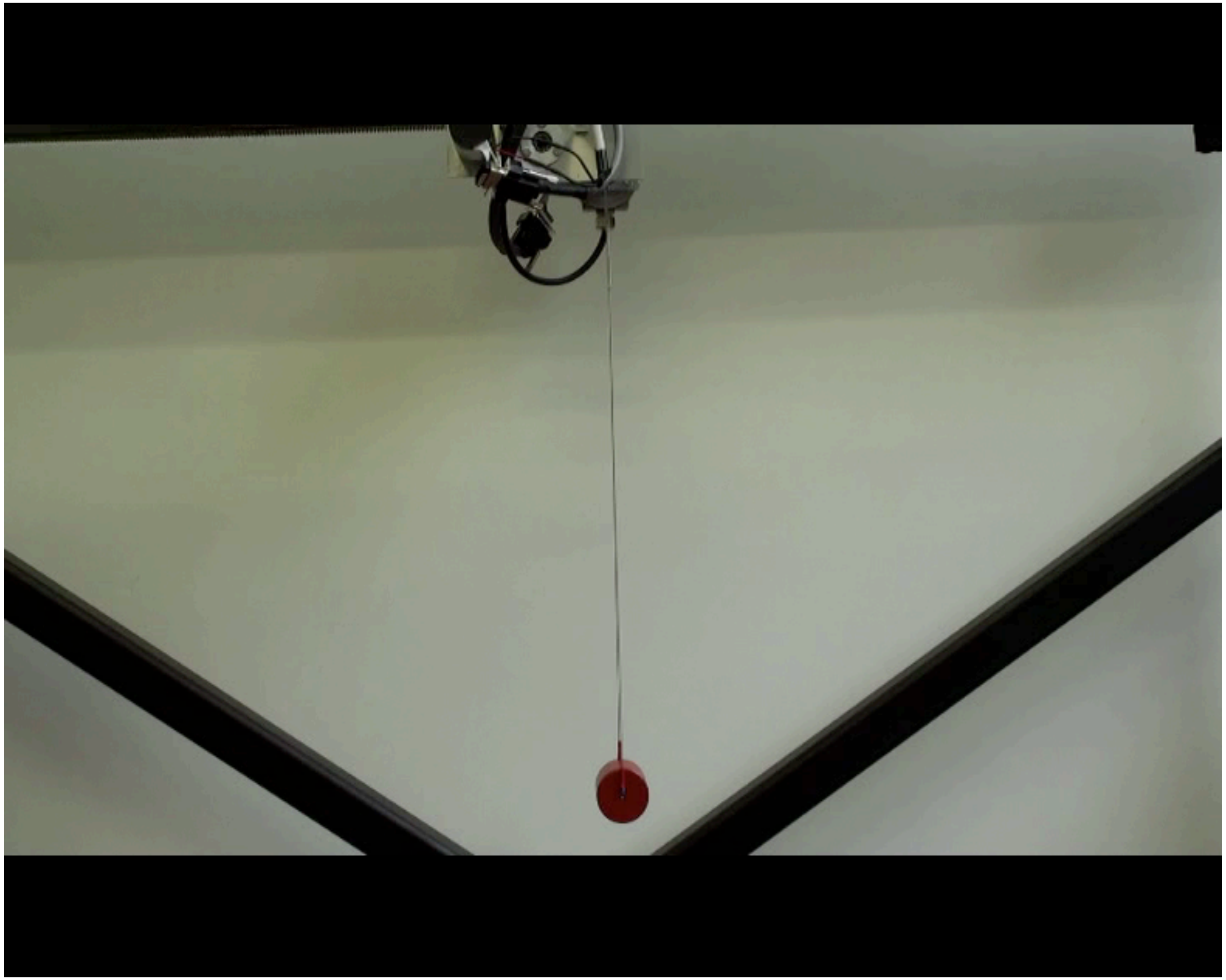
ACTUATOR

- cart motor

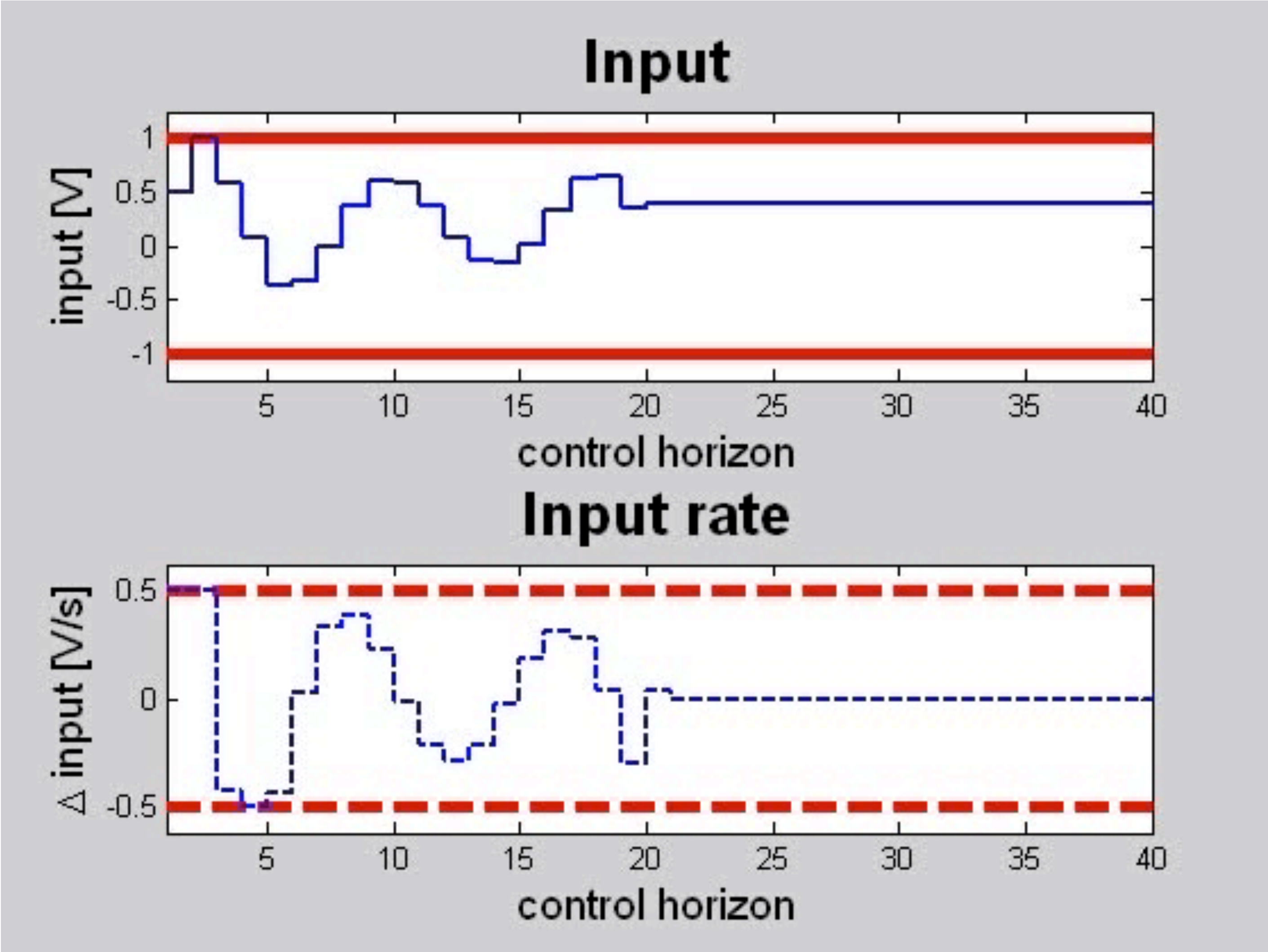
Hardware: xPC Target.   Software: qpOASES [Ferreau, D., Bock, 2008]

Time Optimal MPC of a Crane

Univ. Leuven [Vandenbrouck, Swevers, D.]



Optimal solutions varying in time (inequalities matter)

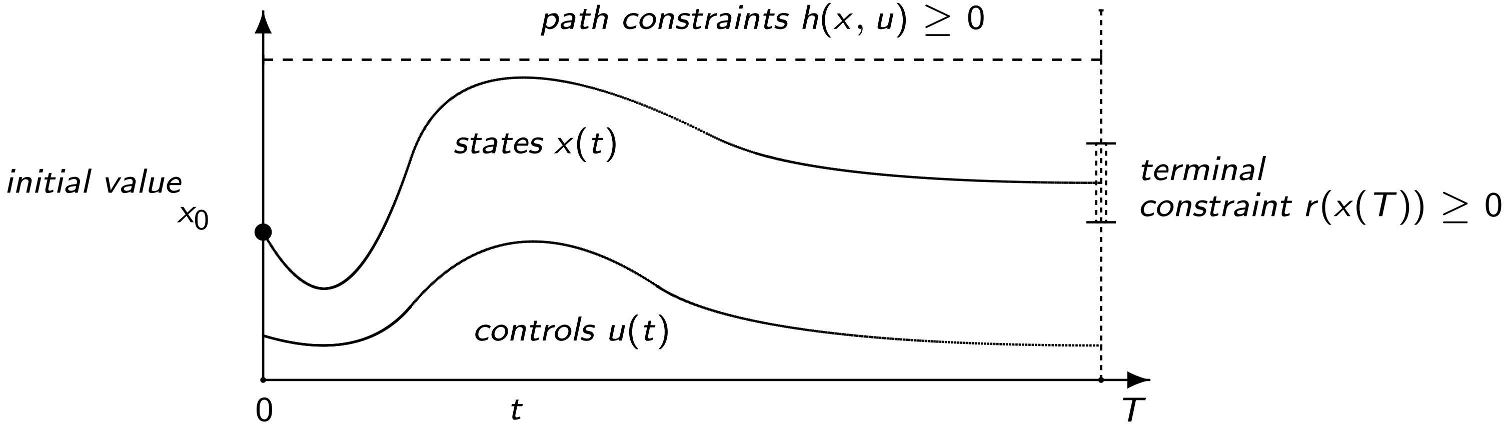


Solver qpOASES [PhD H.J. Ferreau, 2011], [Ferreau, Kirches, Potschka, Bock, D. , A parametric active-set algorithm for quadratic programming, Mathematical Programming Computation, 2014]

Overview

- Embedded Optimization
- Time Optimal Motions in Mechatronics
- **Real-Time Optimization Methods and Software**
- Four Experimental NMPC Applications

Simplified Optimal Control Problem in ODE

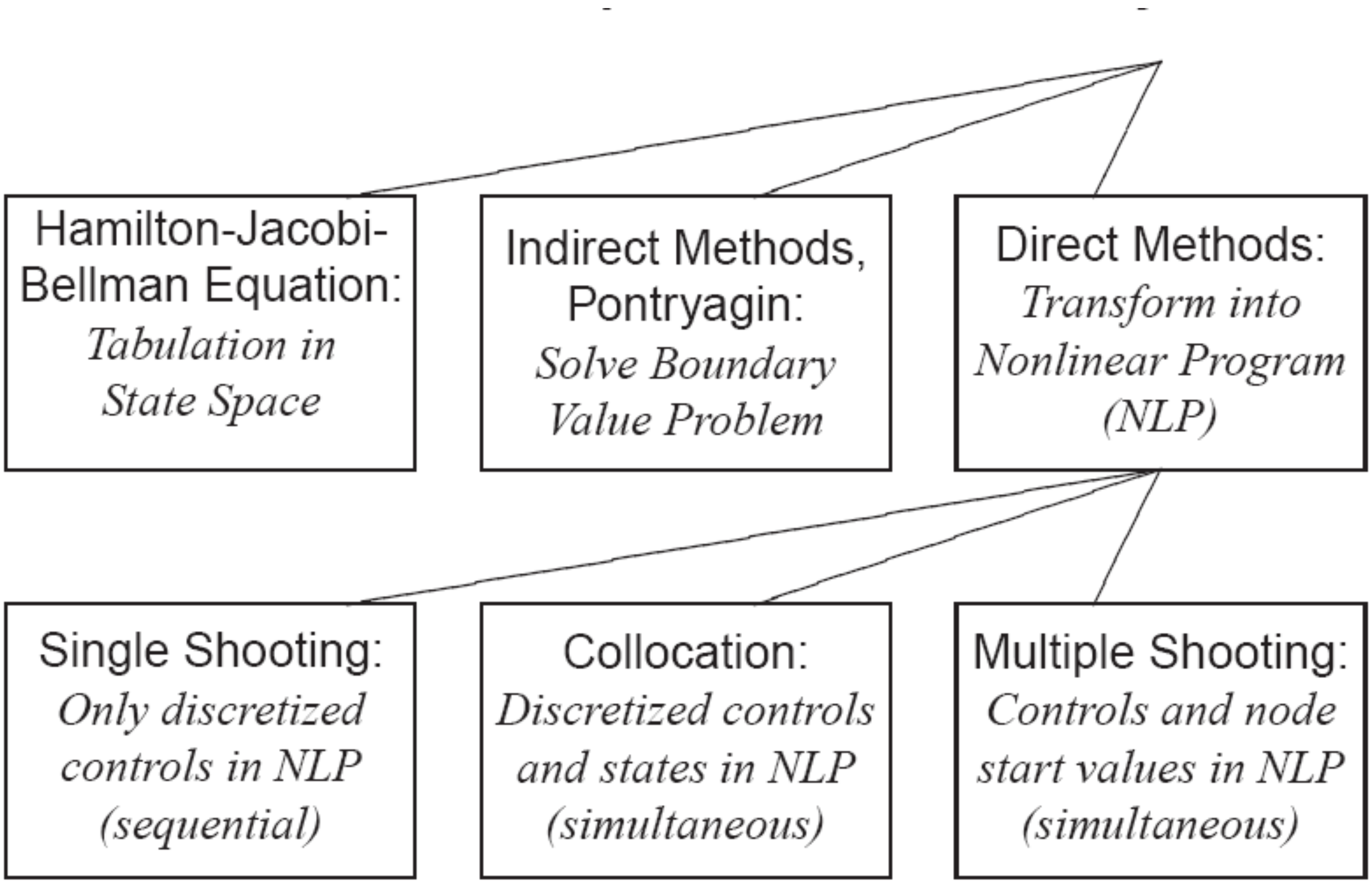


$$\underset{x(\cdot),u(\cdot)}{\text{minimize}} \quad \int_0^T L(x(t), u(t))\, dt \;+\; E\,(x(T))$$

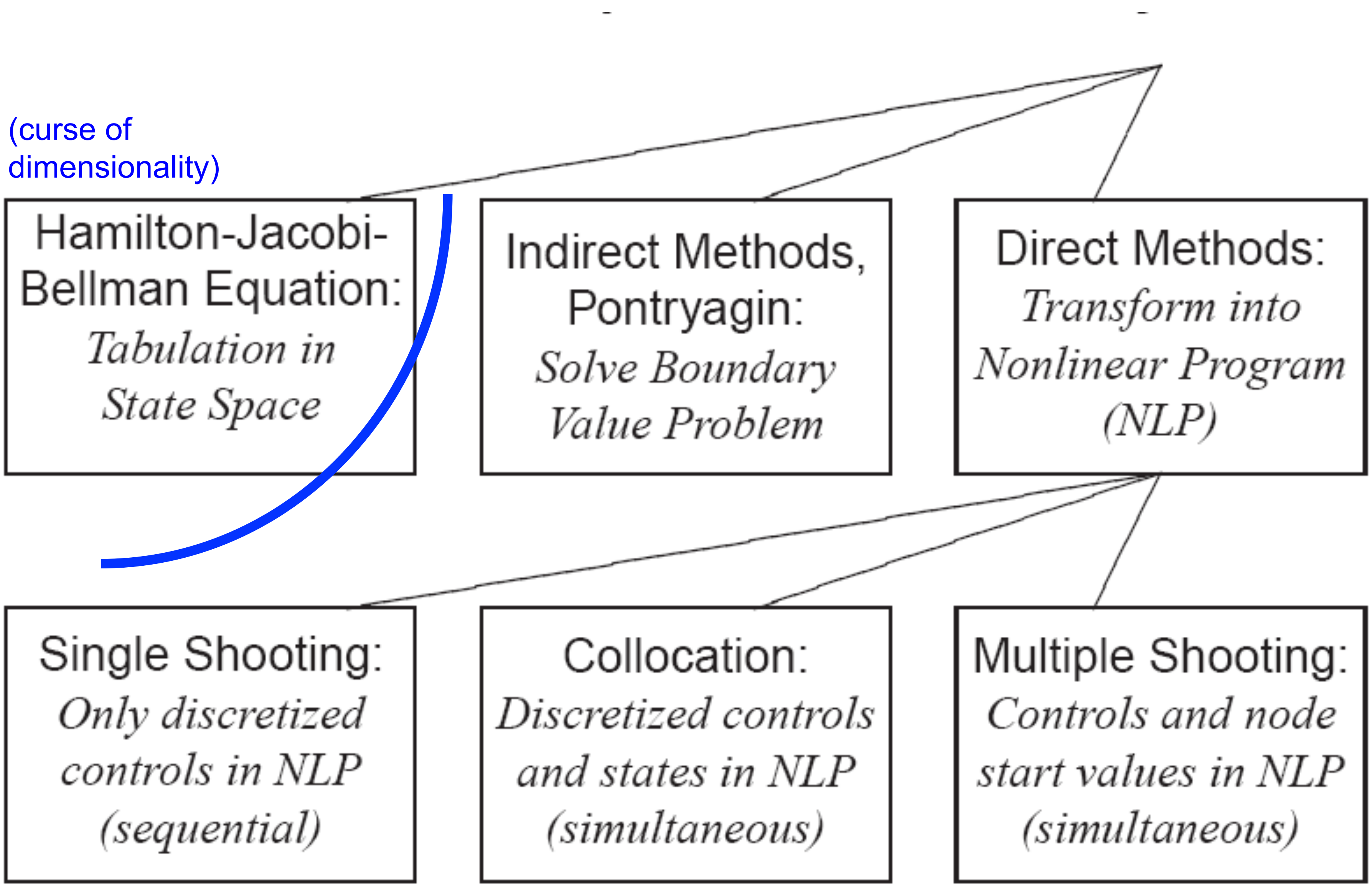
subject to

$$\begin{aligned} x(0) - x_0 &= 0, & & \text{(fixed initial value)} \\ \dot{x}(t) - f(x(t), u(t)) &= 0, & t \in [0, T], & \quad \text{(ODE model)} \\ h(x(t), u(t)) &\geq 0, & t \in [0, T], & \quad \text{(path constraints)} \\ r\,(x(T)) &\geq 0 & & \text{(terminal constraints)} \end{aligned}$$

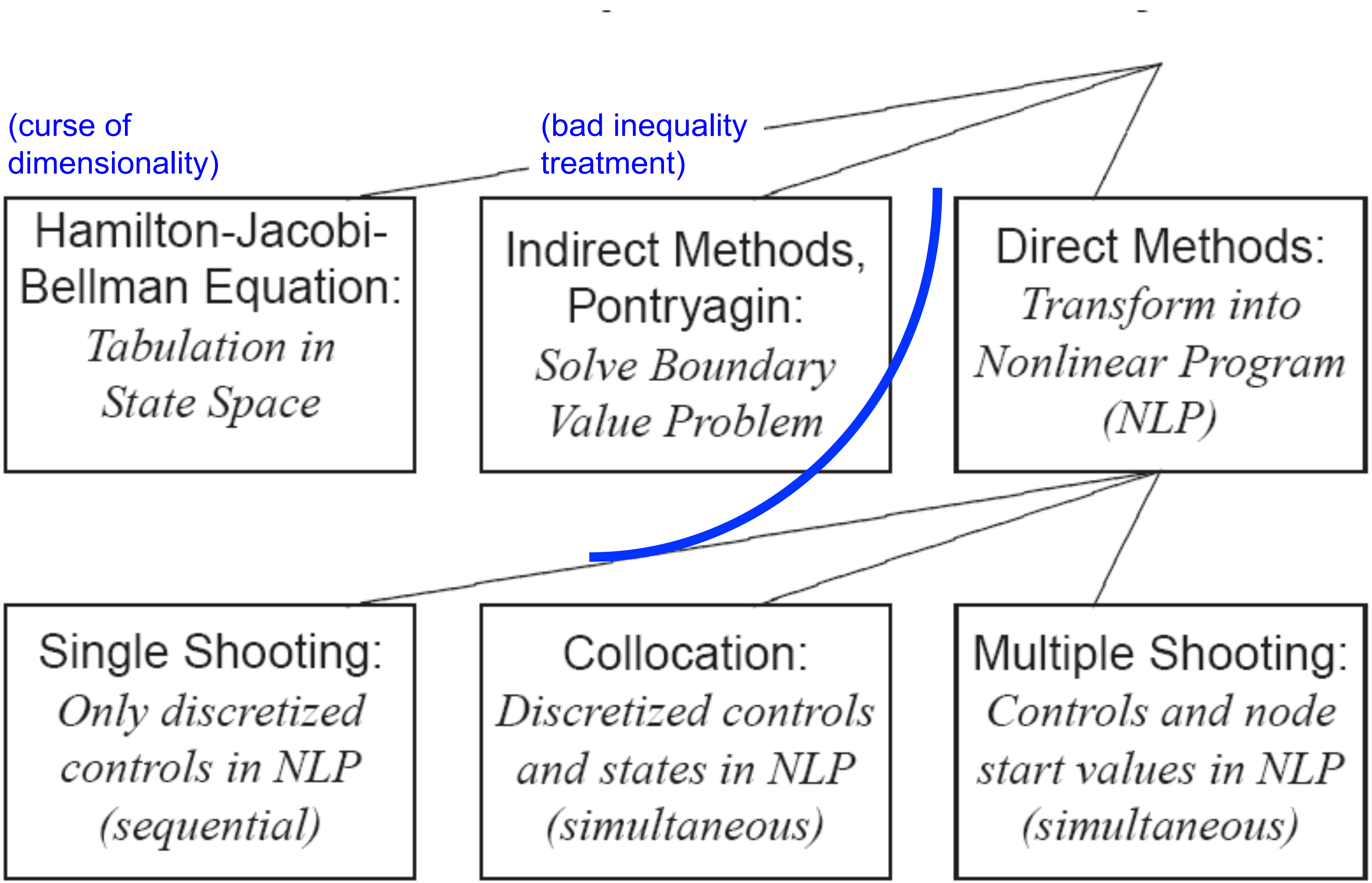
Optimal Control Solution Methods - Family Tree



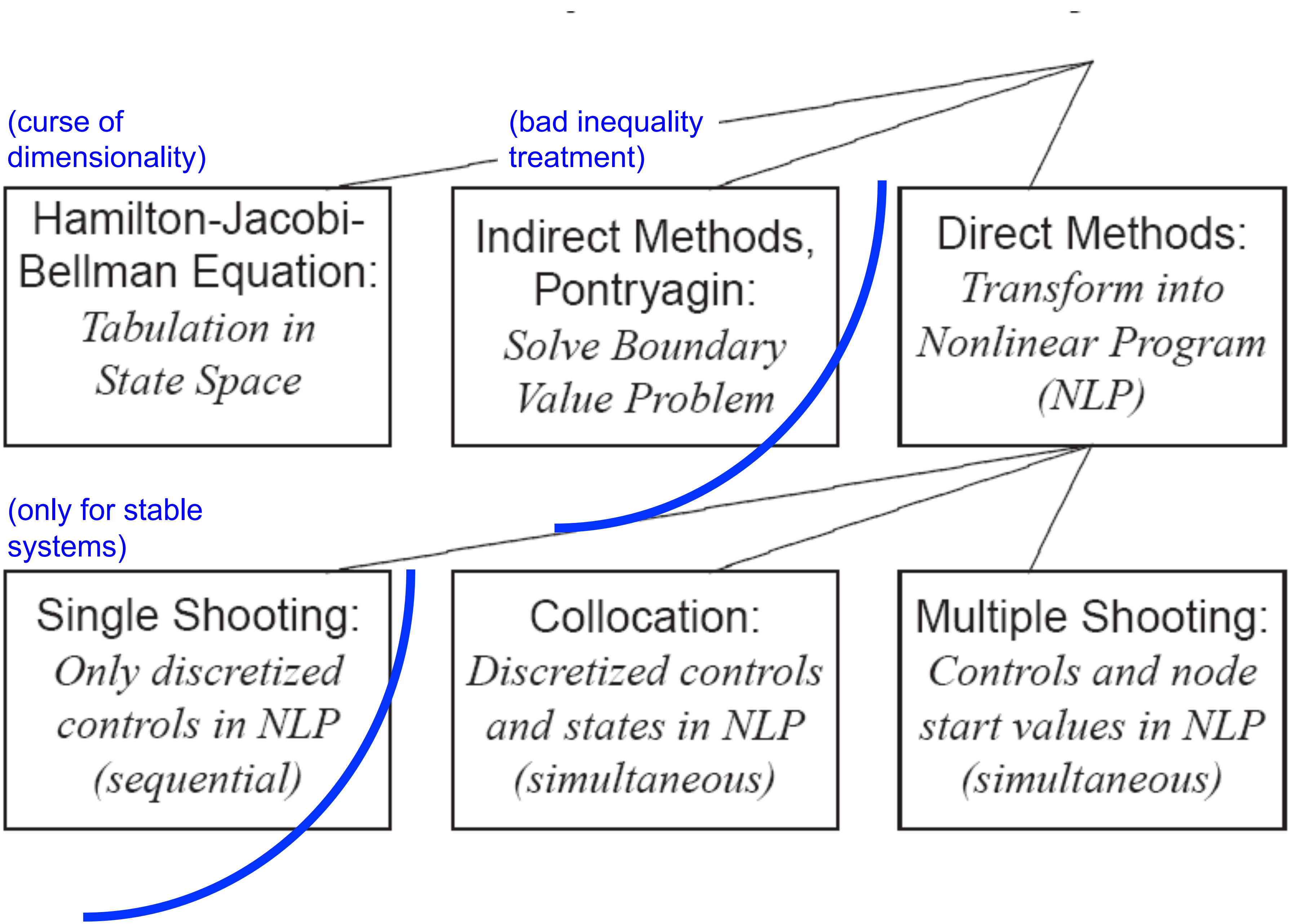
Optimal Control Solution Methods - Family Tree



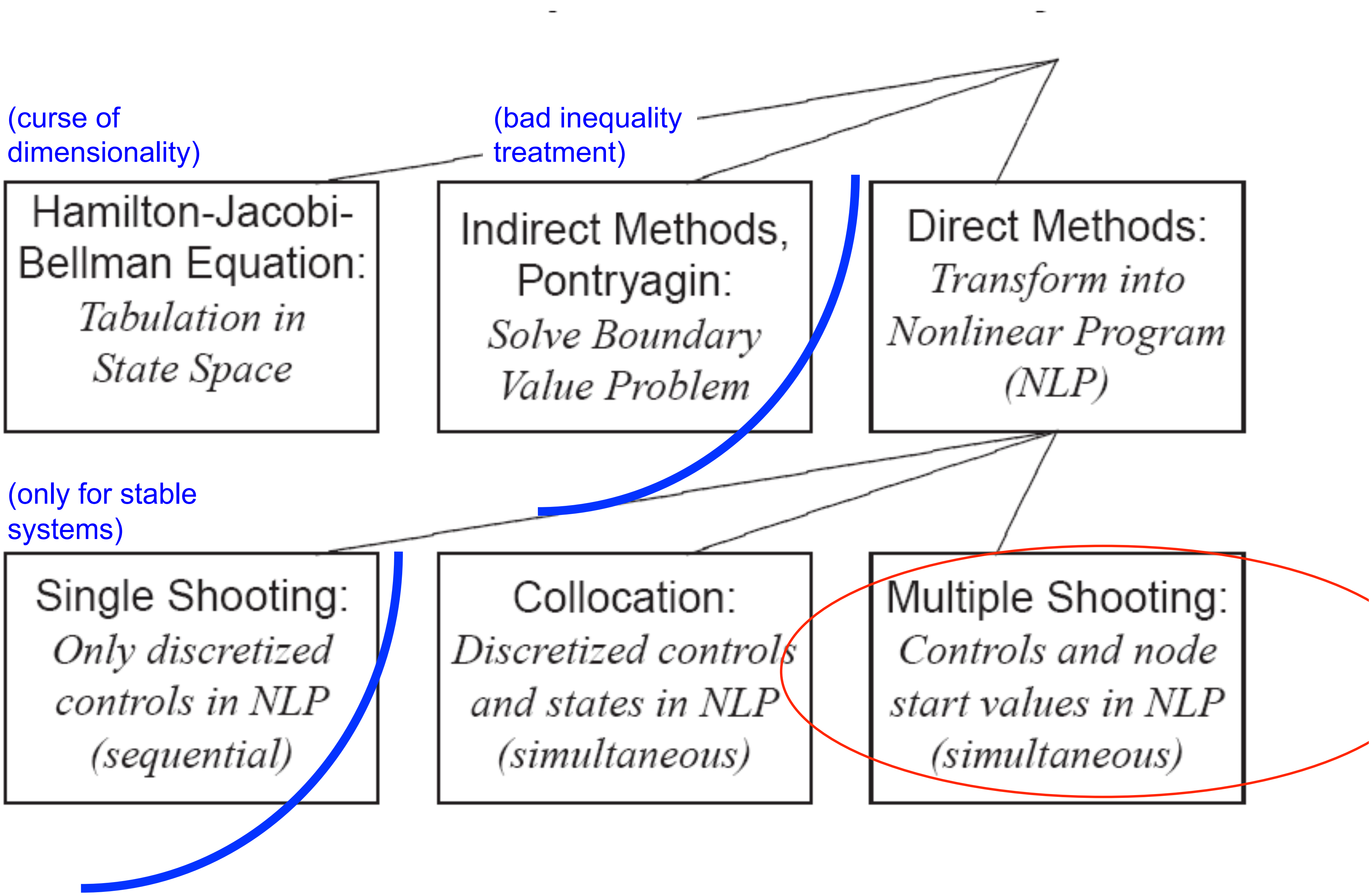
Optimal Control Solution Methods - Family Tree



Optimal Control Solution Methods - Family Tree

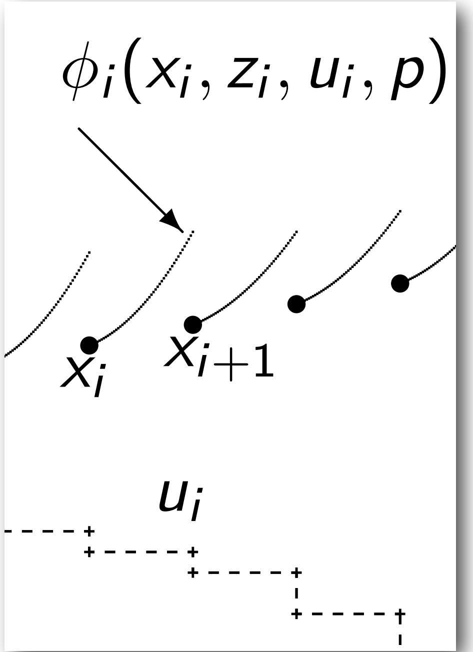


Optimal Control Solution Methods - Family Tree



Direct Multiple Shooting [Bock and Plitt, 1981] [Leineweber et al. 1999]

Hans Georg Bock



- Discretize controls e.g. piecewise constant

$$u(t) = u_i \quad \text{for} \quad t \in [t_i, t_{i+1}]$$

- Solve relaxed DAE on each interval $[t_i, t_{i+1}]$ numerically, starting with artificial initial values x_i, z_i. Obtain trajectory pieces, and state at end of interval $\phi_i(x_i, z_i, q_i, p)$.

$$\begin{aligned}
&\underset{x,z,u,p}{\text{minimize}} \quad \sum_{i=0}^{N-1} l_i(x_i, z_i, u_i, p) \quad + \quad E\,(x_N, p) \\
&\text{subject to} \\
& x_{i+1} - \phi_i(x_i, z_i, u_i, p) = 0, \ i = 0, \dots, N-1, \quad \text{(continuity)} \\
& g(x_i, z_i, u_i, p) = 0, \ i = 0, \dots, N-1, \quad \text{(algebraic consistency)} \\
& h(x_i, z_i, u_i, p) \geq 0, \ i = 0, \dots, N, \quad \text{(discretized path constr.)} \\
& r\,(x_0, x_N, p) \geq 0. \quad \text{(boundary conditions)}
\end{aligned}$$

Real-Time Iterations [PhD Diehl 2001, Heidelberg]

1) Keep states in problem - use direct multiple shooting [1]
2) Exploit convexity via Generalized Gauss-Newton [2]
3) Use tangential predictors for short feedback delay [3]
4) Iterate while problem changes (Real-Time Iterations) [4]
5) Auto-generate custom solvers in plain-C [5,6] (no if, no malloc)

[1] Bock & Plitt, IFAC WC, 1984
[2] Bock 1983
[3] Bock, D. et al, 1999
[4] D. et al., 2002 / 2005
[5] Mattingley & Boyd, 2009
[6] Houska et al.: Automatica, 2011.

Open source toolkit: ACADO CodeGen [6]

Dynamic Optimization Problem in MPC

$$
\begin{array}{lrcll}
\underset{x,\,z,\,u}{\text{minimize}} & \displaystyle\sum_{i=0}^{N-1} L_i(x_i, z_i, u_i) & + & E\,(x_N) & \\
\text{subject to} & x_0 - \bar{x}_0 & = & 0, & \\
& x_{i+1} - f_i(x_i, z_i, u_i) & = & 0, & i = 0, \dots, N-1, \\
& g_i(x_i, z_i, u_i) & = & 0, & i = 0, \dots, N-1, \\
& h_i(x_i, z_i, u_i) & \leq & 0, & i = 0, \dots, N-1, \\
& r\,(x_N) & \leq & 0. &
\end{array}
$$

Structured parametric Nonlinear Program (pNLP)

Initial Value $\bar{x}_0$ is often not known beforehand (“online data” in MPC)

Discrete time dynamics come from ODE simulation (“multiple shooting”)

Dynamic Optimization Problem in MPC

$$
\begin{array}{llcll}
\underset{x,\,z,\,u}{\text{minimize}} & \displaystyle\sum_{i=0}^{N-1} L_i(x_i, z_i, u_i) & + & E\,(x_N) & \\
\text{subject to} & x_0 - \bar{x}_0 & = & 0, & \\
& x_{i+1} - f_i(x_i, z_i, u_i) & = & 0, & i = 0, \ldots, N-1, \\
& g_i(x_i, z_i, u_i) & = & 0, & i = 0, \ldots, N-1, \\
& h_i(x_i, z_i, u_i) & \leq & 0, & i = 0, \ldots, N-1, \\
& r\,(x_N) & \leq & 0. &
\end{array}
$$

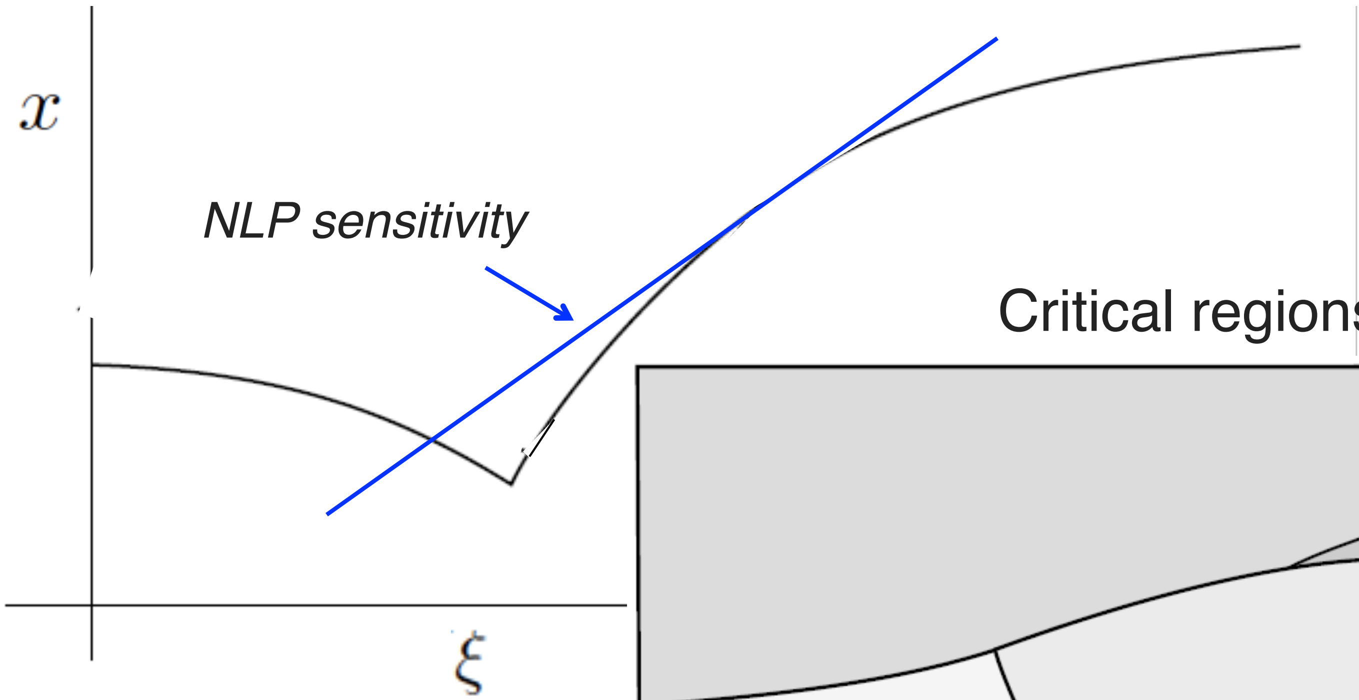
Summarize as

$$
\begin{array}{ll}
\min\limits_{x \in \mathbb{R}^n} & f(x) \\
\text{s.t.} & g(x) + M\xi = 0, \\
& x \in \Omega,
\end{array}
$$

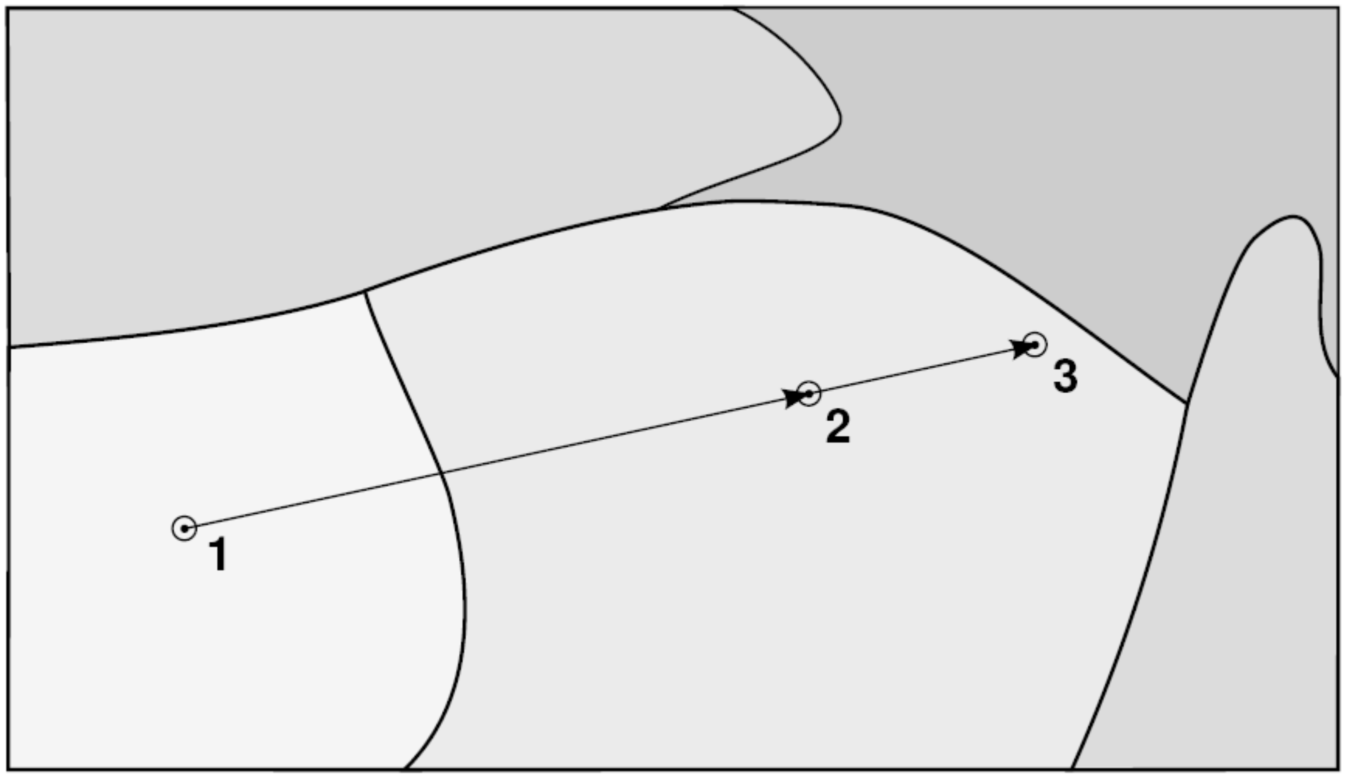
with convex f and Ω

Nonlinear MPC = parametric Nonlinear Programming

Solution manifold is piecewise differentiable (kinks at active set changes)



Critical regions are non-polyhedral



How to deal with a sequence of large parameter changes?

NLP Pathfollowing!

Real-Time Iteration (Sequential Convex Programming)

Step 1: Linearize nonlinear constraints at x^k to obtain convex problem:

$$\boxed{\begin{array}{ll} \min\limits_{x \in \mathbb{R}^n} & f(x) \\ \text{s.t.} & g(x^\kappa) + g'(x^\kappa)(x - x^\kappa) + M\xi = 0, \\ & x \in \Omega. \end{array}}$$

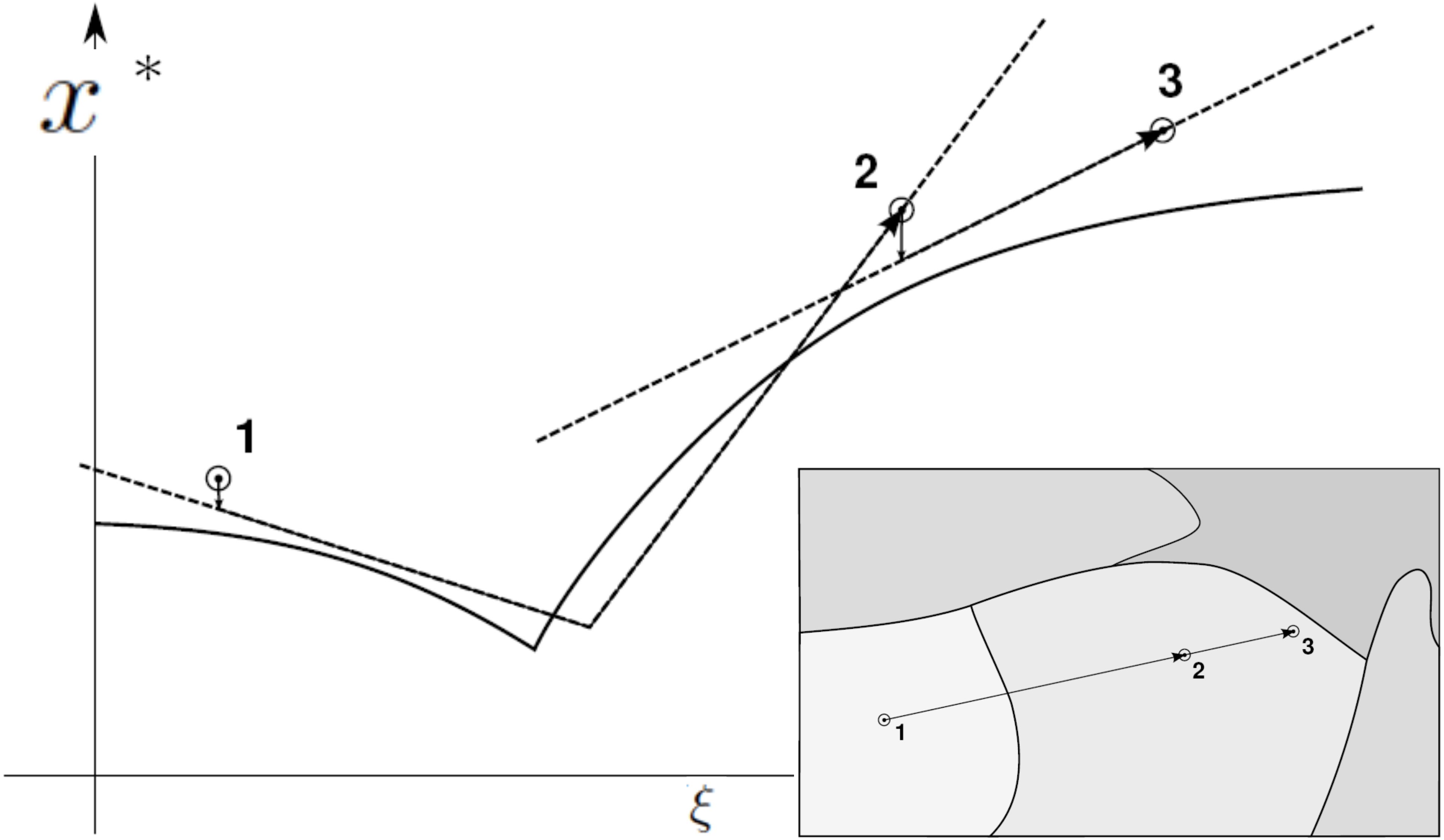
Step 2: Get new value of parameter ξ and solve convex problem - typically a quadratic program (QP) - to obtain next iterate. Go to step 1.

[Diehl, Bock, Schloeder, Findeisen, Nagy, Allgower, JPC, 2002]

[Zavala, Anitescu, SICON, 2010]

[Tran Dinh, Savorgnan, Diehl, SIOPT, 2013]

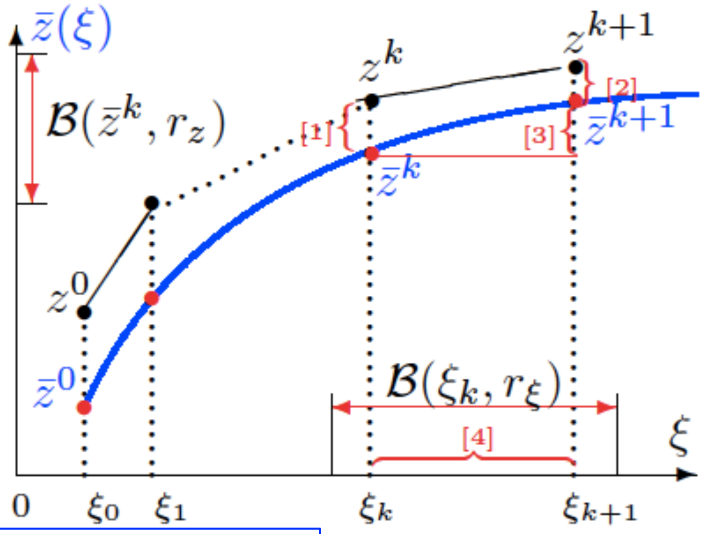
Real-Time Iteration



Tangential prediction even across active set changes

Can divide computations in "preparation" and "feedback phase" [D. 2001]

Real-Time Iteration Contraction Estimate



$$z^k := (x^k, y^k)$$

Contraction estimate for primal dual errors:

$$\boxed{\begin{aligned} \|z^{k+1} - \bar{z}^{k+1}\| \le & \left(\alpha + c_1 \|z^k - \bar{z}^k\|\right) \|z^k - \bar{z}^k\| \\ & + \left(c_2 + c_3 \|\xi_{k+1} - \xi_k\|\right) \|\xi_{k+1} - \xi_k\| \end{aligned}}$$

Contraction depends on bounds on nonlinearity, Jacobian error, and on strong regularity. Contraction rate independent of active set changes!

[Tran Dinh, Savorgnan, Diehl, SIOPT, 2013]

Computations in one Real-Time Iteration

NLP

$$
\begin{array}{llcl}
\underset{x,\,z,\,u}{\text{minimize}} & \displaystyle\sum_{i=0}^{N-1} L_i(x_i, z_i, u_i) & + & E\,(x_N) \\
\text{subject to} & x_0 - \bar{x}_0 & = & 0, \\
& x_{i+1} - f_i(x_i, z_i, u_i) & = & 0, \quad i = 0, \ldots, N-1, \\
& g_i(x_i, z_i, u_i) & = & 0, \quad i = 0, \ldots, N-1, \\
& h_i(x_i, z_i, u_i) & \leq & 0, \quad i = 0, \ldots, N-1, \\
& r\,(x_N) & \leq & 0.
\end{array}
$$

1) Linearize constraints:
Integration & sensitivities

Sparse QP

$$
\begin{array}{llcl}
\underset{x,\,u}{\text{minimize}} & \displaystyle\sum_{i=0}^{N-1} L_{\mathrm{redQP},i}(x_i, u_i) & + & E_{\mathrm{QP}}\,(x_N) \\
\text{subject to} & x_0 - \bar{x}_0 & = & 0, \\
& x_{i+1} - c_i - A_i x_i - B_i u_i & = & 0, \quad i = 0, \ldots, N-1, \\
& \bar{h}_i + \bar{H}_i^x x_i + \bar{H}_i^u u_i & \leq & 0, \quad i = 0, \ldots, N-1, \\
& r' + R x_N & \leq & 0.
\end{array}
$$

2) Condense sparse QP

Condensed small QP

$$
\begin{array}{lll}
\underset{u}{\text{minimize}} & f_{\mathrm{condQP},i}(\bar{x}_0, u) & \\
\text{subject to} & \bar{r} + \bar{R}^{x_0} \bar{x}_0 + \bar{R}^u u & \leq \quad 0.
\end{array}
$$

3) Solve condensed QP

Computations in one Real-Time Iteration

NLP

$$\begin{array}{lrcll}
\underset{x,\,z,\,u}{\text{minimize}} & \displaystyle\sum_{i=0}^{N-1} L_i(x_i, z_i, u_i) & + & E\,(x_N) & \\
\text{subject to} & x_0 - \bar{x}_0 & = & 0, & \\
& x_{i+1} - f_i(x_i, z_i, u_i) & = & 0, & i = 0, \ldots, N-1, \\
& g_i(x_i, z_i, u_i) & = & 0, & i = 0, \ldots, N-1, \\
& h_i(x_i, z_i, u_i) & \leq & 0, & i = 0, \ldots, N-1, \\
& r\,(x_N) & \leq & 0. &
\end{array}$$

Sparse QP

$$\begin{array}{lrcll}
\underset{x,\,u}{\text{minimize}} & \displaystyle\sum_{i=0}^{N-1} L_{\text{redQP},i}(x_i, u_i) & + & E_{\text{QP}}\,(x_N) & \\
\text{subject to} & x_0 - \bar{x}_0 & = & 0, & \\
& x_{i+1} - c_i - A_i x_i - B_i u_i & = & 0, & i = 0, \ldots, N-1, \\
& \bar{h}_i + \bar{H}_i^x x_i + \bar{H}_i^u u_i & \leq & 0, & i = 0, \ldots, N-1, \\
& r' + R x_N & \leq & 0. &
\end{array}$$

Condensed small QP

$$\begin{array}{lrcl}
\underset{u}{\text{minimize}} & f_{\text{condQP},i}(\bar{x}_0, u) & & \\
\text{subject to} & \bar{r} + \bar{R}^{x_0}\bar{x}_0 + \bar{R}^u u & \leq & 0.
\end{array}$$

Can prepare without knowing $\bar{x}_0$

"Preparation phase"

1) Linearize constraints: Integration & sensitivities

2) Condense sparse QP

3) Solve condensed QP

Computations in one Real-Time Iteration

NLP

$$
\begin{array}{lrcll}
\underset{x,\,z,\,u}{\text{minimize}} & \displaystyle\sum_{i=0}^{N-1} L_i(x_i, z_i, u_i) & + & E\,(x_N) & \\
\text{subject to} & x_0 - \bar{x}_0 & = & 0, & \\
& x_{i+1} - f_i(x_i, z_i, u_i) & = & 0, & i = 0, \ldots, N-1, \\
& g_i(x_i, z_i, u_i) & = & 0, & i = 0, \ldots, N-1, \\
& h_i(x_i, z_i, u_i) & \leq & 0, & i = 0, \ldots, N-1, \\
& r\,(x_N) & \leq & 0. &
\end{array}
$$

1) Linearize constraints:
Integration & sensitivities

Sparse QP

$$
\begin{array}{lrcll}
\underset{x,\,u}{\text{minimize}} & \displaystyle\sum_{i=0}^{N-1} L_{\text{redQP},i}(x_i, u_i) & + & E_{\text{QP}}\,(x_N) & \\
\text{subject to} & x_0 - \bar{x}_0 & = & 0, & \\
& x_{i+1} - c_i - A_i x_i - B_i u_i & = & 0, & i = 0, \ldots, N-1, \\
& \bar{h}_i + \bar{H}_i^x x_i + \bar{H}_i^u u_i & \leq & 0, & i = 0, \ldots, N-1, \\
& r' + R x_N & \leq & 0. &
\end{array}
$$

2) Condense sparse QP

Condensed small QP

$$
\begin{array}{lrcl}
\underset{u}{\text{minimize}} & f_{\text{condQP},i}(\bar{x}_0, u) & & \\
\text{subject to} & \bar{r} + \bar{R}^{x_0} \bar{x}_0 + \bar{R}^u u & \leq & 0.
\end{array}
$$

“Feedback phase”

3) Solve condensed QP

Alternative: skip condensing, directly solve sparse QP

NLP

$$
\begin{array}{llcll}
\underset{x,\,z,\,u}{\text{minimize}} & \displaystyle\sum_{i=0}^{N-1} L_i(x_i, z_i, u_i) \quad + \quad E\,(x_N) & & & \\
\text{subject to} & x_0 - \bar{x}_0 & = & 0, & \\
 & x_{i+1} - f_i(x_i, z_i, u_i) & = & 0, & i = 0, \dots, N-1, \\
 & g_i(x_i, z_i, u_i) & = & 0, & i = 0, \dots, N-1, \\
 & h_i(x_i, z_i, u_i) & \leq & 0, & i = 0, \dots, N-1, \\
 & r\,(x_N) & \leq & 0. &
\end{array}
$$

1) Linearize constraints:
Integration & sensitivities

Sparse QP

$$
\begin{array}{llcll}
\underset{x,\,u}{\text{minimize}} & \displaystyle\sum_{i=0}^{N-1} L_{\text{redQP},i}(x_i, u_i) \quad + \quad E_{\text{QP}}\,(x_N) & & & \\
\text{subject to} & x_0 - \bar{x}_0 & = & 0, & \\
 & x_{i+1} - c_i - A_i x_i - B_i u_i & = & 0, & i = 0, \dots, N-1, \\
 & \bar{h}_i + \bar{H}_i^x x_i + \bar{H}_i^u u_i & \leq & 0, & i = 0, \dots, N-1, \\
 & r' + R x_N & \leq & 0. &
\end{array}
$$

2) Solve sparse QP, e.g.
with Riccati type algorithm

Feedback phase

Past algorithmic developments [PhD students at KU Leuven]

Hans Joachim Ferreau

- Online active set strategy QP solver **qpOASES** [Ferreau, Bock, D IJC 2007], [Ferreau, Kirches, Potschka, Bock, D., MathProgC 2014]. Well tested in dozens of academic and industrial applications.

Boris Houska

- Autogeneration of plain-C nonlinear optimal control solvers in **ACADO** [Houska, Ferreau, D., Automatica 2011]

Joel Andersson

- Algorithmic Differentiation and Optimal Control Modelling Environment **CasADi** [Andersson, Akesson, D., LNCSE 2012]

[all our software is open-source and comes under industry friendly LGPL license]

Past algorithmic developments (2)

Milan Vukov

- **ACADO Code Generation** consolidation, block sparse condensing, parallelization, moving horizon estimation

Janick Frasch

- Dual Newton active set strategy QP solver **qpDUNES** for long horizon optimal control [Frasch 2014], [Frasch, Sager, D. 2015]

Joris Gillis

- **CasADi** consolidation, sparse Hessian generation, differentiable implicit solvers for Lyapunov equations

Recent algorithmic developments (in Freiburg)

Rien Quirynen

Dimitris Kouzoupis

Robin Verschueren

Gianluca Frison

Andrea Zanelli

- Auto-Generated Implicit Integrators with Higher Order Derivatives [Quirynen, Vukov, Zanon, D.,OCAM, 2014]
- Tree-Sparsity Exploiting Optimisation Algorithms for Nonlinear Model Predictive Control
- Convex Second Derivative Approximations for Economic Nonlinear Model Predictive Control
- Efficient Register Management for Block Sparse Linear Algebra Solvers, Riccati QP Solver **HPMPC** (PhD at DTU Lyngby, currently postdoc in Freiburg)
- Inexact Newton type methods for microsecond Nonlinear MPC

Recent algorithmic developments (in Freiburg)

Rien Quirynen

- Auto-Generated Implicit Integrators with Higher Order Derivatives [Quirynen, Vukov, Zanon, D.,OCAM, 2014]

Dimitris Kouzoupis

- Tree-Sparsity Exploiting Optimisation Algorithms for Nonlinear Model Predictive Control

Robin Verschueren

- Convex Second Derivative Approximations for Economic Nonlinear Model Predictive Control

Gianluca Frison

- Efficient Register Management for Block Sparse Linear Algebra Solvers, Riccati QP Solver **HPMPC** (PhD at DTU Lyngby, currently postdoc in Freiburg)

Andrea Zanelli

- Inexact Newton type methods for microsecond Nonlinear MPC

Implicit Runge-Kutta / Collocation Integrators for ODE

Approximate solution of Ordinary Differential Equation (ODE)

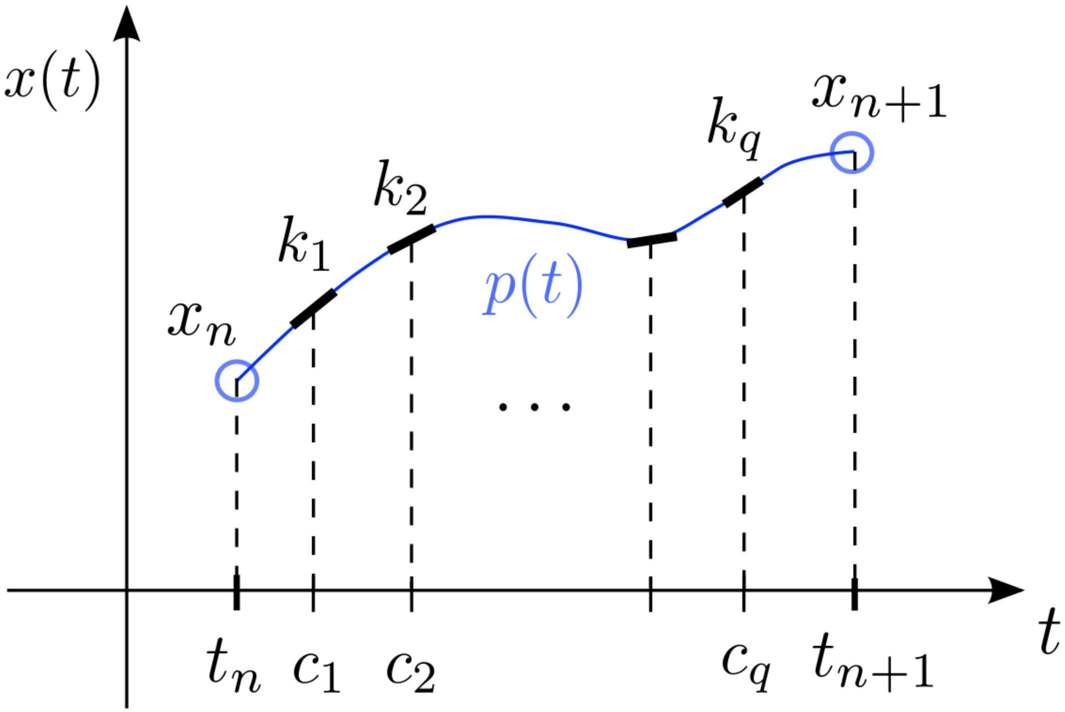
$$0 = f(t, \dot{x}(t), x(t))$$

by an interpolation polynomial $p(t)$ that satisfies derivative matching conditions on q collocation points:

$$p(t_n) = x_n$$

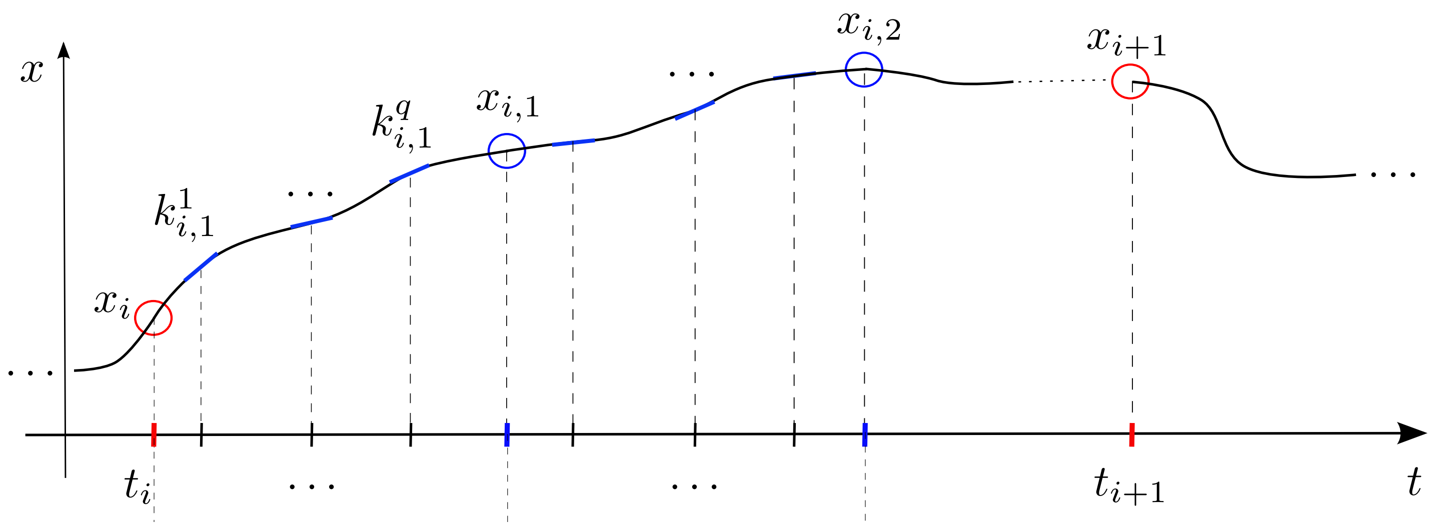
$$0 = f(t_i, \dot{p}(t_i), p(t_i))$$

$$\text{for} \quad i = 1, \ldots, q.$$



Integration over one multiple shooting interval

Concatenate several integrator steps together



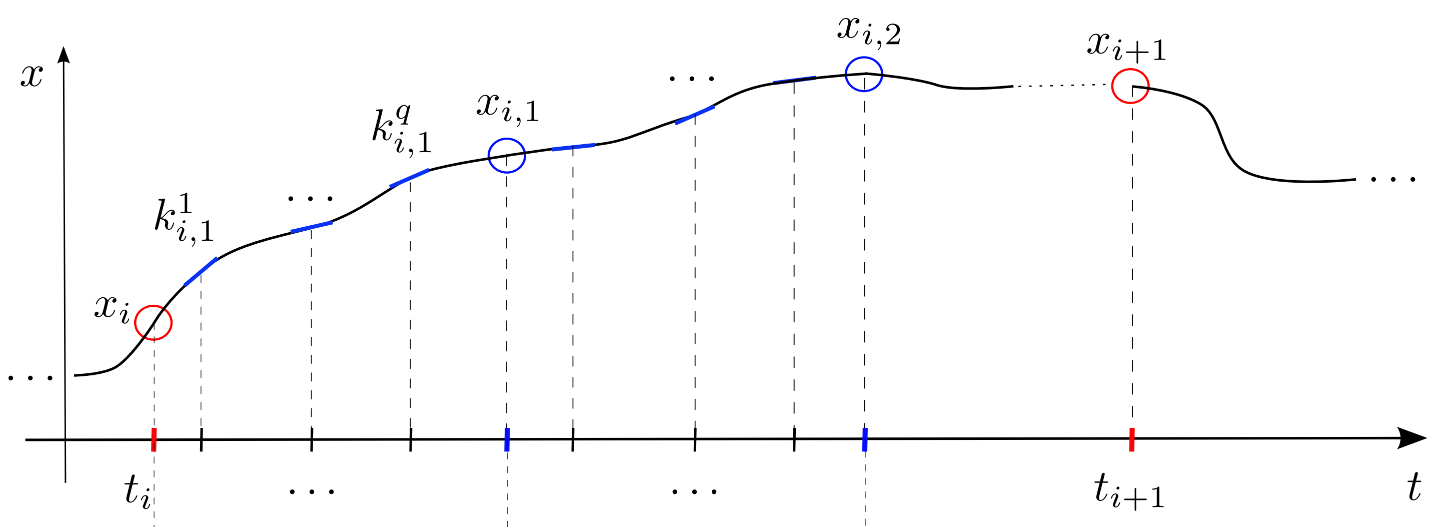
and summarise as implicit discrete time system (with large vector K_n of internal variables):

$$x_{n+1} = F(x_n, K_n, u)$$

$$0 = G(x_n, K_n, u),$$

Integration over one multiple shooting interval

Concatenate several integrator steps together



and summarise as implicit discrete time system (with large vector K_n of internal variables):

$$x_{n+1} = F(x_n, K_n, u)$$

$$0 = G(x_n, K_n, u),$$

after Newton convergence also compute sensitivities:

$$\begin{bmatrix} \frac{\mathrm{d}x_{n+1}}{\mathrm{d}x_0} & \frac{\mathrm{d}x_{n+1}}{\mathrm{d}u} \end{bmatrix} = \begin{bmatrix} \frac{\partial F_n}{\partial x}\frac{\mathrm{d}x_n}{\mathrm{d}x_0} & \frac{\partial F_n}{\partial x}\frac{\mathrm{d}x_n}{\mathrm{d}u} + \frac{\partial F_n}{\partial u} \end{bmatrix} + \frac{\partial F_n}{\partial K}\begin{bmatrix} \frac{\mathrm{d}K_n}{\mathrm{d}x_0} & \frac{\mathrm{d}K_n}{\mathrm{d}u} \end{bmatrix}$$

$$\begin{bmatrix} \frac{\mathrm{d}K_n}{\mathrm{d}x_0} & \frac{\mathrm{d}K_n}{\mathrm{d}u} \end{bmatrix} = -\frac{\partial G_n}{\partial K}^{-1}\begin{bmatrix} \frac{\partial G_n}{\partial x}\frac{\mathrm{d}x_n}{\mathrm{d}x_0} & \frac{\partial G_n}{\partial x}\frac{\mathrm{d}x_n}{\mathrm{d}u} + \frac{\partial G_n}{\partial u} \end{bmatrix},$$

Standard vs Lifted Collocation Integrator

Direct Multiple Shooting with Collocation Integrator

$$
\begin{aligned}
\min_{X,\,U} \quad & \sum_{i=0}^{N-1} l(x_i, u_i) + m(x_N) \\
\text{s.t.} \quad & 0 = x_0 - \hat{x}_0 \\
& 0 = \phi(x_i, u_i) - x_{i+1}, \quad i \in \mathbb{Z}_0^{N-1}
\end{aligned}
$$

$\phi(\bar{x}_i, \bar{u}_i)$
and derivatives

$\bar{x}_i, \bar{u}_i$

$$G(w_i, K_i) = 0$$

ΔK

Newton-type scheme

Standard vs Lifted Collocation Integrator

Direct Multiple Shooting with Collocation Integrator

$$\min_{X,U} \quad \sum_{i=0}^{N-1} l(x_i,u_i) + m(x_N)$$

$$\begin{aligned} \text{s.t.} \quad 0 &= x_0 - \hat{x}_0 \\ 0 &= \phi(x_i,u_i) - x_{i+1}, \quad i \in \mathbb{Z}_0^{N-1} \end{aligned}$$

$\phi(\bar{x}_i,\bar{u}_i)$
and derivatives

$\bar{x}_i, \bar{u}_i$

$$G(w_i,K_i) = 0$$

ΔK

Newton-type scheme

Direct Multiple Shooting with Lifted Collocation Integrator

$$\min_{X,U} \quad \sum_{i=0}^{N-1} l(x_i,u_i) + m(x_N)$$

$$\begin{aligned} \text{s.t.} \quad 0 &= x_0 - \hat{x}_0 \\ 0 &= x_i + B\,K_i - x_{i+1}, \quad i \in \mathbb{Z}_0^{N-1} \end{aligned}$$

$\Delta\tilde{K}_i, K_i^w$
and derivatives

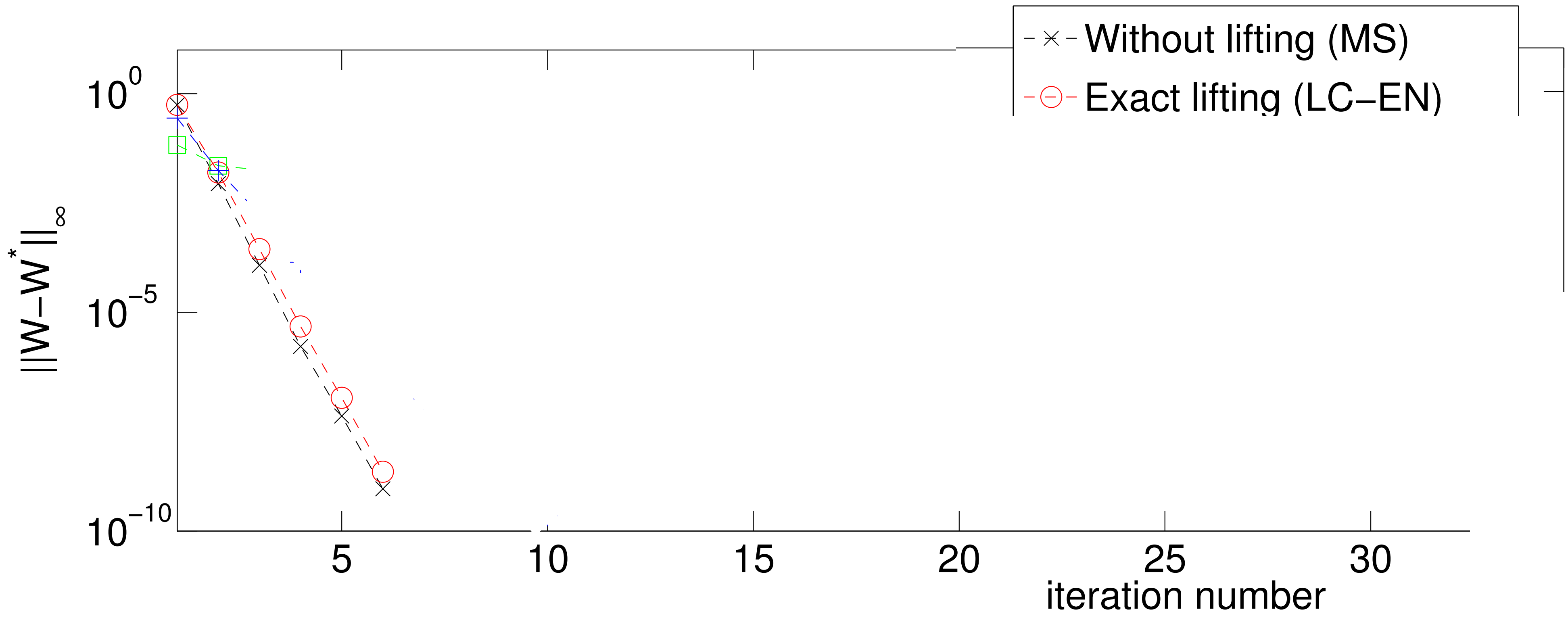
$\bar{x}_i, \bar{u}_i$

$$\Delta\tilde{K}_i = -\frac{\partial G(\bar{z}_i)}{\partial K_i}^{-1} G(\bar{z}_i)$$

$$K_i^w = -\frac{\partial G(\bar{z}_i)}{\partial K_i}^{-1} \frac{\partial G(\bar{z}_i)}{\partial w_i}$$

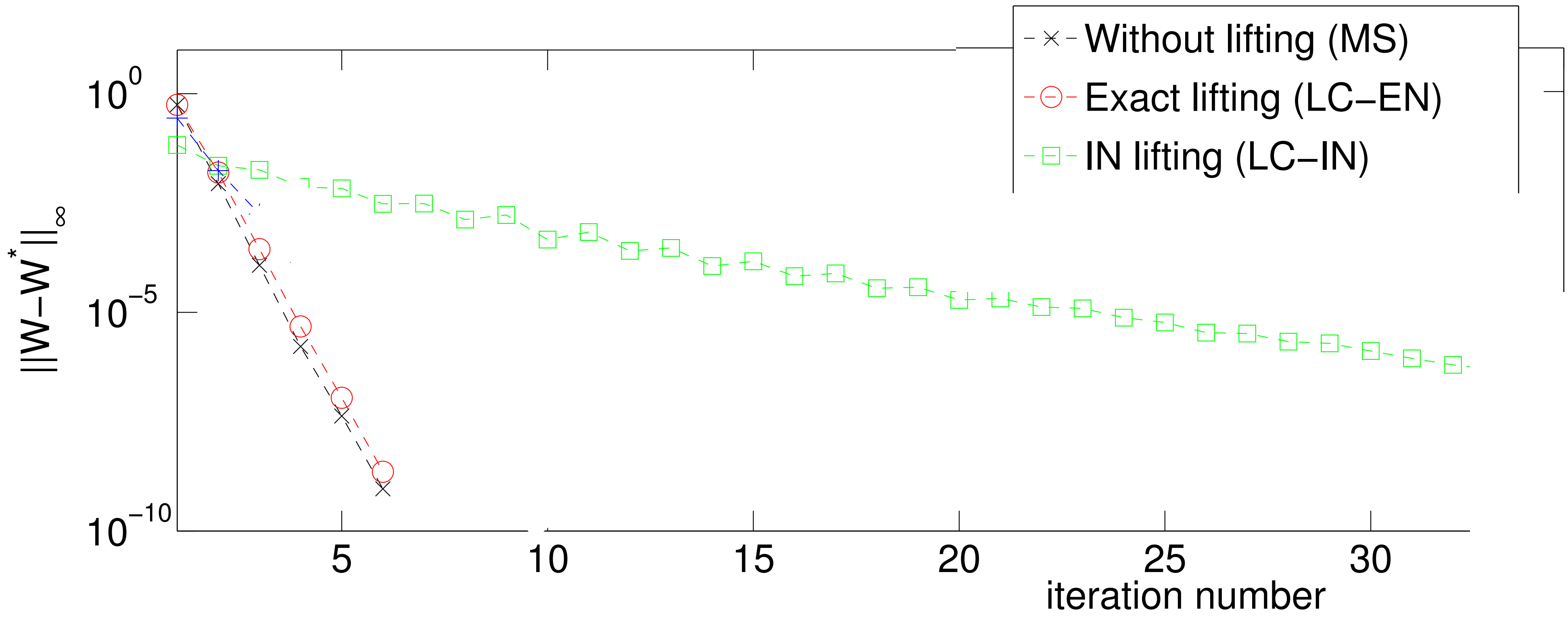
one hidden Newton iteration only,
equivalent to direct collocation…

SQP type iterations for an optimal control test problem



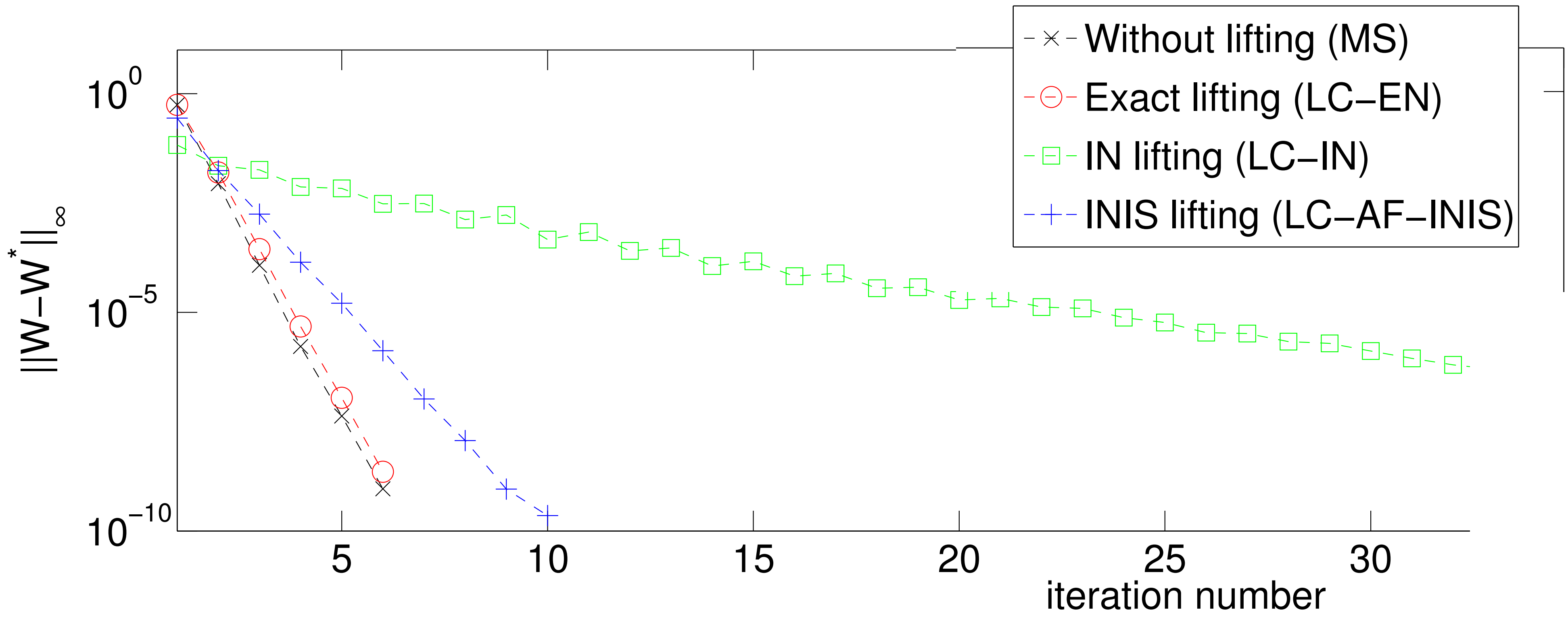
- Lifting makes iterations cheaper at same convergence speed

SQP type iterations for an optimal control test problem



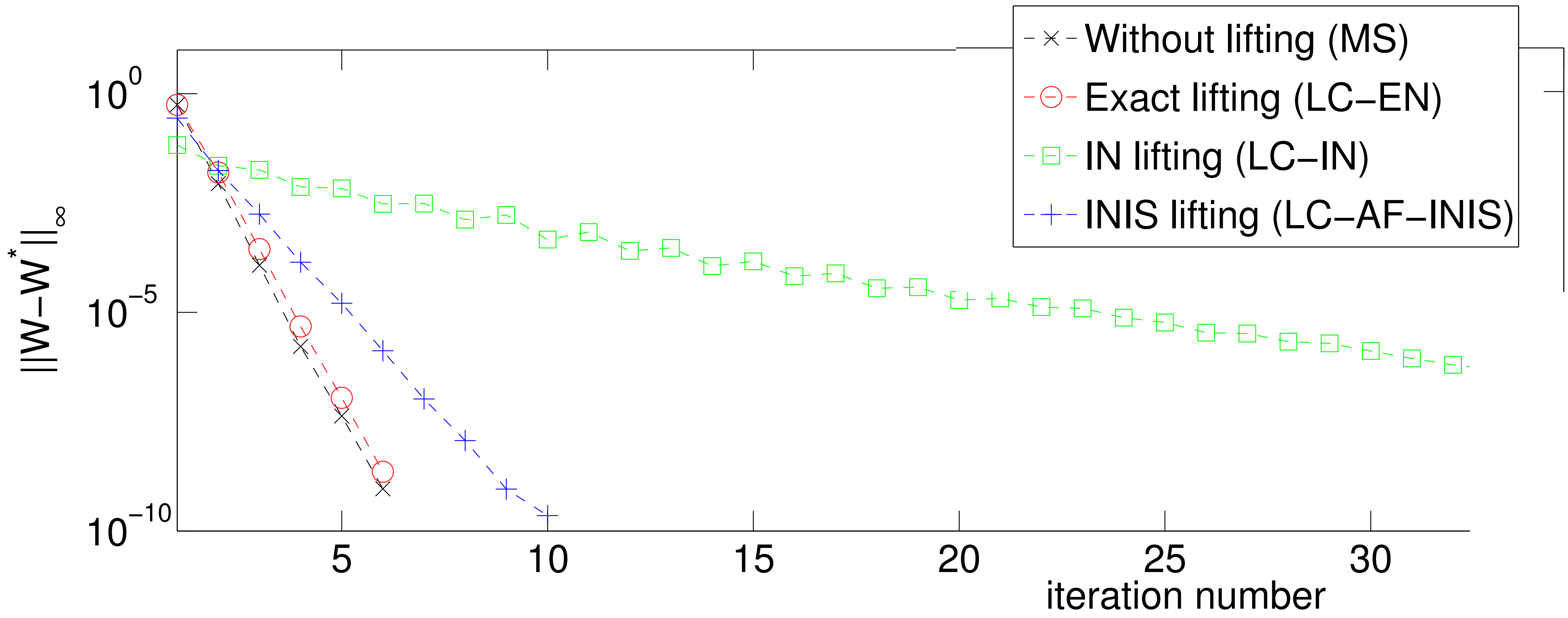
- Lifting makes iterations cheaper at same convergence speed
- can make iterations even cheaper by **Inexact Newton (IN),** but with slower convergence

SQP type iterations for an optimal control test problem



- Lifting makes iterations cheaper at same convergence speed
- can make iterations even cheaper by **Inexact Newton (IN),** but with slower convergence
- can improve contraction rate by **Inexact Newton with Iterated Sensitivities (INIS)**

SQP type iterations for an optimal control test problem



- Lifting makes iterations cheaper at same convergence speed
- can make iterations even cheaper by **Inexact Newton (IN),** but with slower convergence
- can improve contraction rate by **Inexact Newton with Iterated Sensitivities (INIS)**
  - *theory highlight: INIS recovers same contraction rate as forward problem!*

CPU Time Comparison per SQP Iteration

Rien Quirynen

Detailed timing results for exact Hessian based SQP on the time optimal OCP using $n_{\mathrm{m}} = 5$ masses or $n_{\mathrm{x}} = 24+\mathbf{1}$ states ($N_{\mathrm{s}} = 3, q = 4$), where one iteration of direct collocation (6.7) based on `Ipopt` takes about 270 ms.

	Without lifting (MS)	Exact lifting (LC-EN)	IN lifting (LC-IN)	INIS lifting (LC-INIS)
simulation	**87.23 ms**	51.33 ms	15.50 ms	**15.48 ms**
condensing	2.07 ms	2.08 ms	2.05 ms	2.06 ms
regularization	1.72 ms	1.82 ms	1.86 ms	1.86 ms
QP solution	5.69 ms	6.13 ms	5.67 ms	5.50 ms
total SQP step	96.77 ms	61.55 ms	25.09 ms	**24.92 ms**

Case Study: Quadratic Programming improvements 2012-2016 (all algorithms re-activated on same computer on 22.6.2016)

Andrea Zanelli

Case Study: Quadratic Programming improvements 2012-2016 (all algorithms re-activated on same computer on 22.6.2016)

Andrea Zanelli

Comparison of different algorithmic QP solution approaches, using **ACADO** on Linux Laptop, CPU i7 with 3.1 GHz [A. Zanelli]

Hanging Chain Optimal Control Benchmark

- 27 states, 3 controls, state and control constraints,
- vary MPC control horizon length from N=10 to N=100 intervals
- direct multiple shooting leads to sparse NLP with N*(27+3) variables, N*5 state constraints, N*6 input bounds (3000 variables, 500 state constraints, 600 input bounds for N=100)

Case Study: Quadratic Programming improvements 2012-2016 (all algorithms re-activated on same computer on 22.6.2016)

Andrea Zanelli

Comparison of different algorithmic QP solution approaches, using **ACADO** on Linux Laptop, CPU i7 with 3.1 GHz [A. Zanelli]

Hanging Chain Optimal Control Benchmark

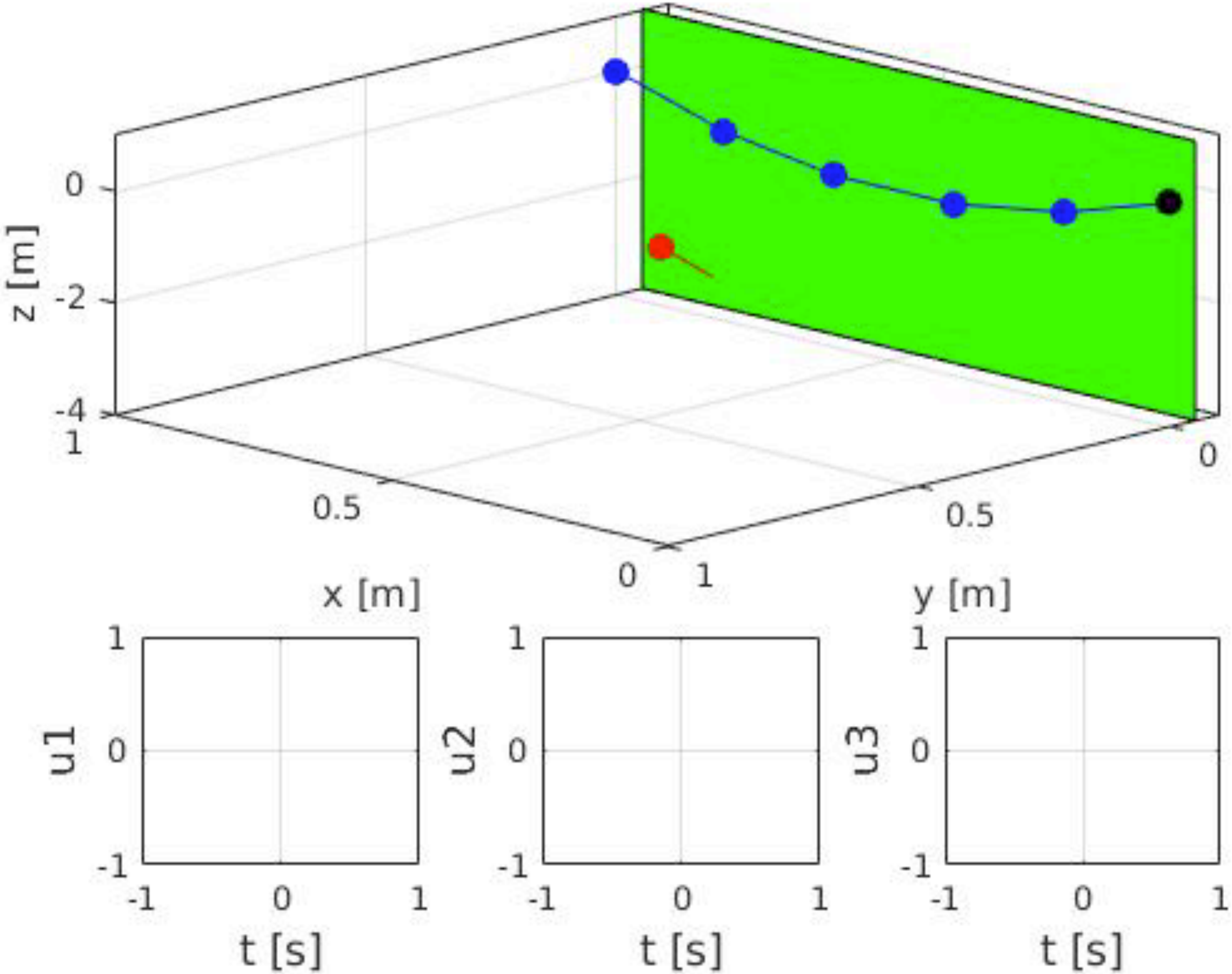
- 27 states, 3 controls, state and control constraints,
- vary MPC control horizon length from N=10 to N=100 intervals
- direct multiple shooting leads to sparse NLP with N*(27+3) variables, N*5 state constraints, N*6 input bounds (3000 variables, 500 state constraints, 600 input bounds for N=100)

Rien Quirynen

- Always use: Numerical integration with code generated Implicit Runge Kutta (IRK-GL2) method [Quirynen 2012], two integration steps per interval.

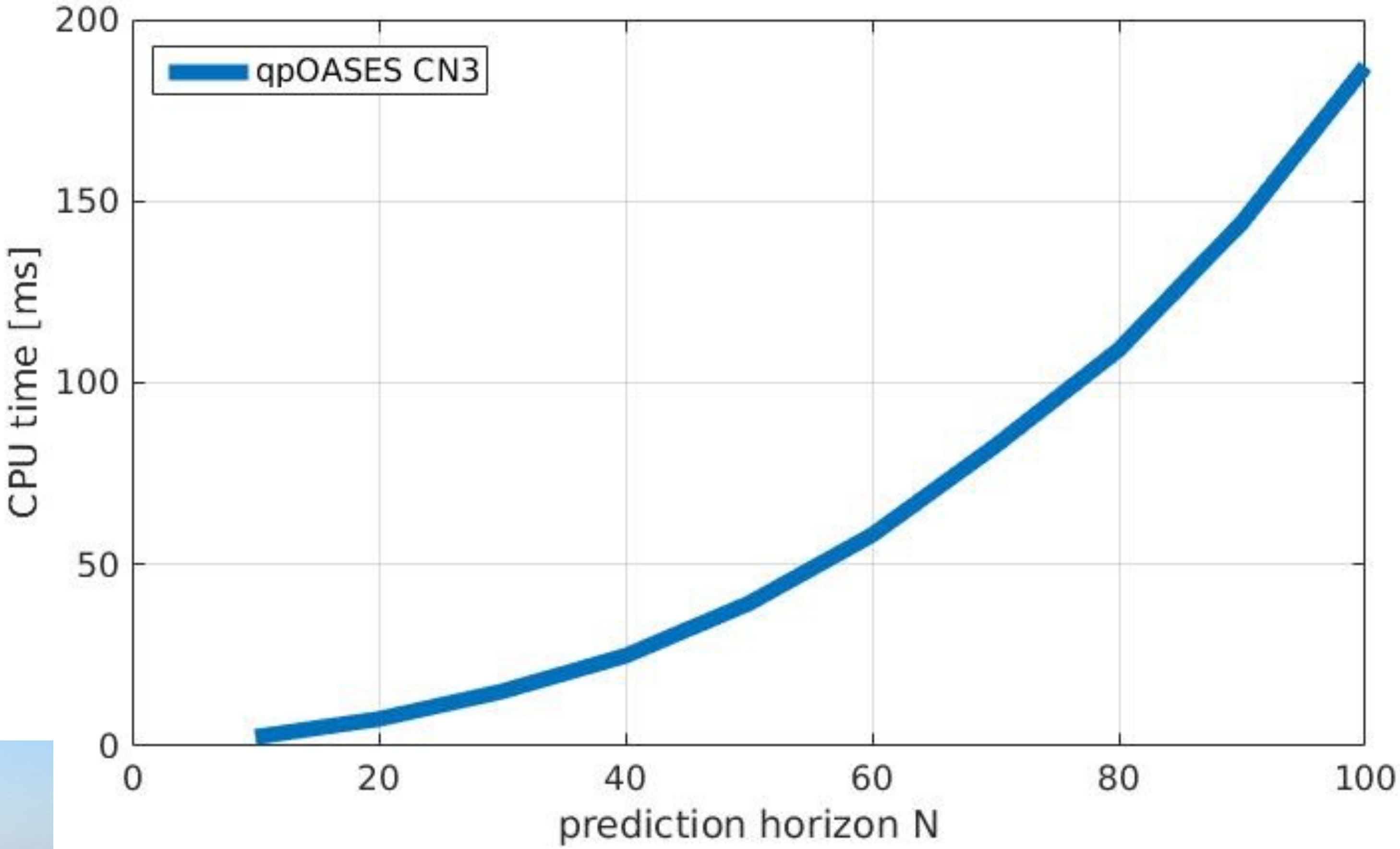
Hanging Chain with Glass Wall Benchmark

Andrea Zanell



2012: ACADO Code Generation with **Condensing**

- efficient block sparse condensing with **$O(N^3)$ complexity**
- qpOASES to solve “condensed” QPs (with 3*N variables)



Milan Vukov

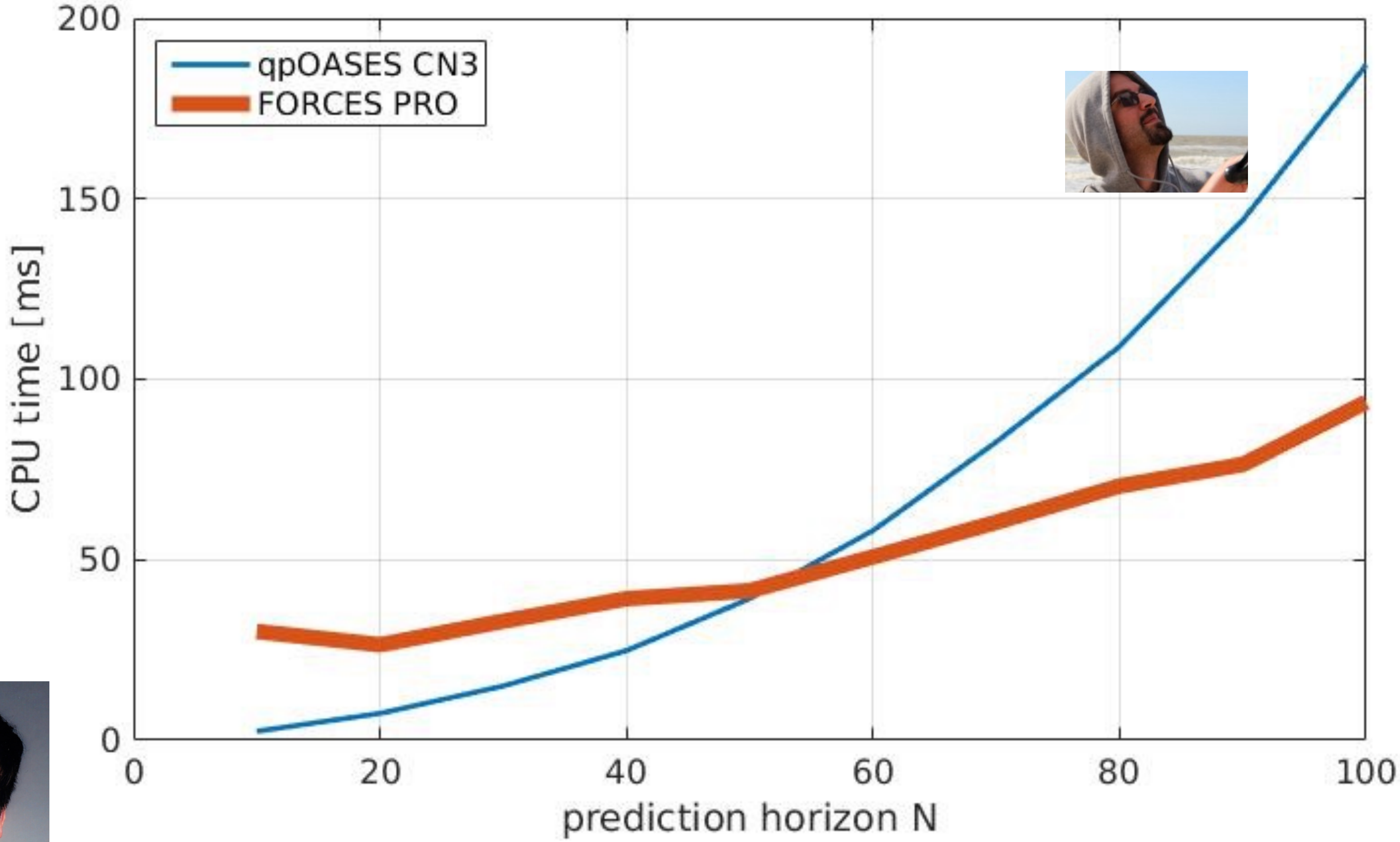
2013: Code generated sparse QP solver **FORCES**

- use interior point method with sparse linear algebra
- code generate Riccati solvers with **O(N) complexity**
- include FORCES as QP solver in ACADO (M. Vukov)

Alexander Domahidi [PhD ETH 2013]

2013: Code generated sparse QP solver **FORCES**

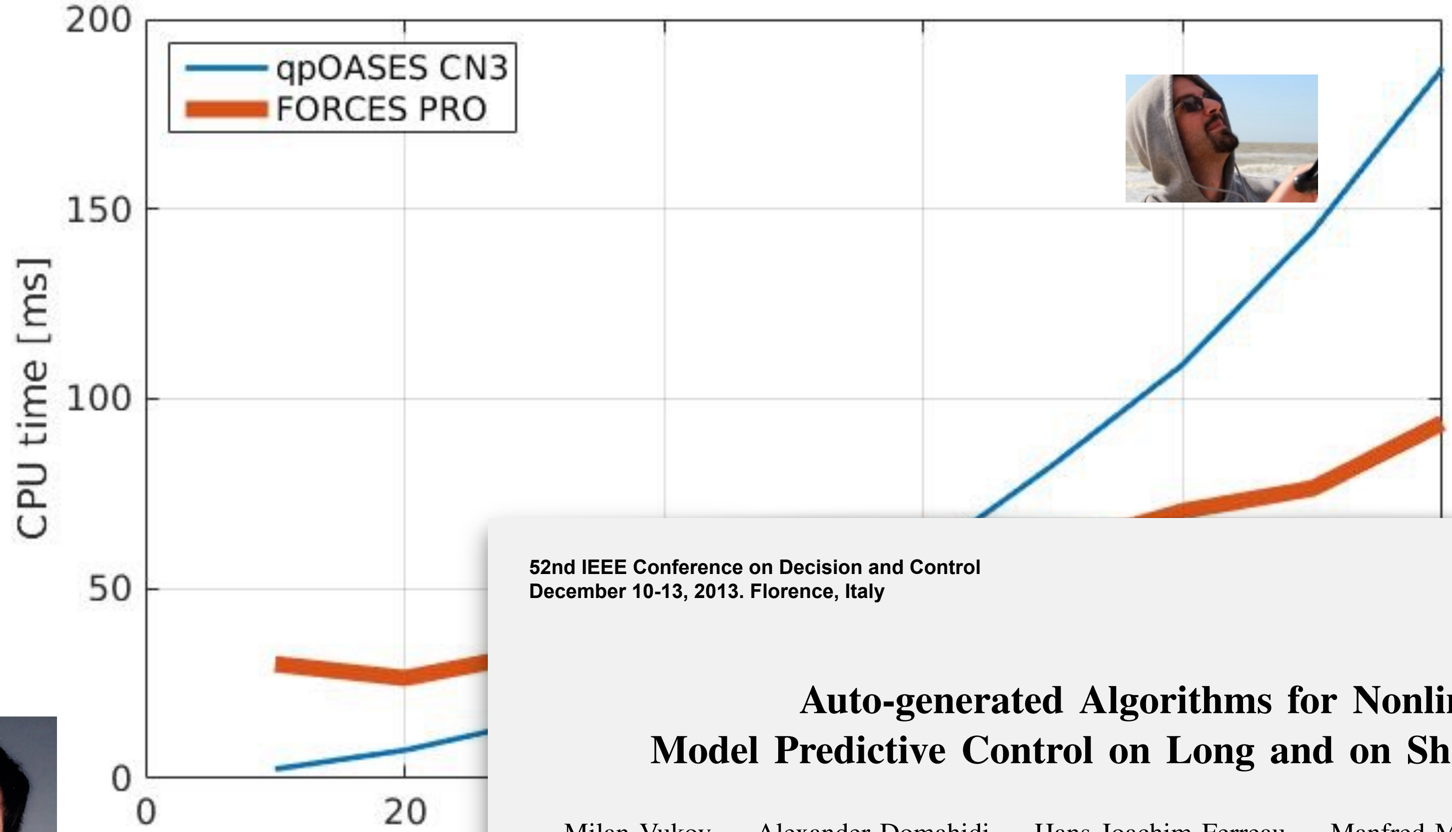
- use interior point method with sparse linear algebra
- code generate Riccati solvers with **O(N) complexity**
- include FORCES as QP solver in ACADO (M. Vukov)



Alexander Domahidi [PhD ETH 2013]

2013: Code generated sparse QP solver **FORCES**

- use interior point method with sparse linear algebra
- code generate Riccati solvers with **O(N) complexity**
- include FORCES as QP solver in ACADO (M. Vukov)



52nd IEEE Conference on Decision and Control
December 10-13, 2013. Florence, Italy

Auto-generated Algorithms for Nonlinear Model Predictive Control on Long and on Short Horizons

Milan Vukov Alexander Domahidi Hans Joachim Ferreau Manfred Morari Moritz Diehl

Alexander Domahidi [PhD ETH 2013]

2013: A Surprising Improvement in **Condensing**

Reorder block matrix multiplications, reduce **$O(N^3)$ to $O(N^2)$ complexity!**

Independently discovered by G. Frison (DTU Lyngby) and J. Andersson (Leuven). Implemented efficiently by M. Vukov in ACADO.

Milan Vukov

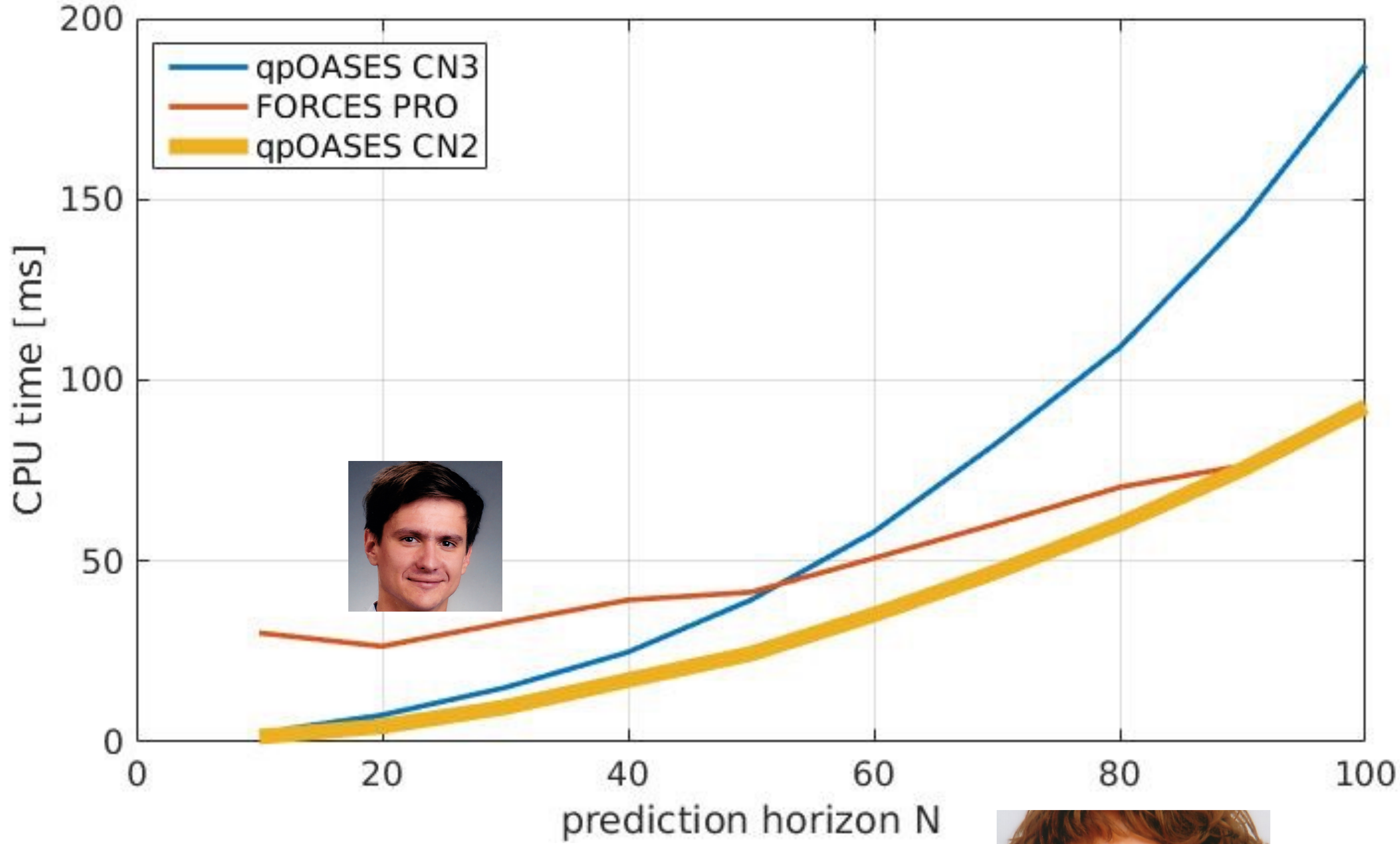
Gianluca Frison

Joel Andersson

2013: A Surprising Improvement in **Condensing**

Reorder block matrix multiplications, reduce **$O(N^3)$ to $O(N^2)$ complexity!**

Independently discovered by G. Frison (DTU Lyngby) and J. Andersson (Leuven). Implemented efficiently by M. Vukov in ACADO.



Milan Vukov

Gianluca Frison

Joel Andersson

2014/15: **qpDUNES** and **Block-Partial Condensing**

Janick Frasch

- "Dual Newton Strategy" qpDUNES (Frasch et al. 2015) uses Lagrangian decomposition, solves many small QPs independently with qpOASES, and performs sparse Cholesky factorisations
- block-partial condensing (first described by Daniel Axehill) boosts performance of qpDUNES (Dimitris Kouzoupis)

Daniel Axehill

Dimitris Kouzoupis

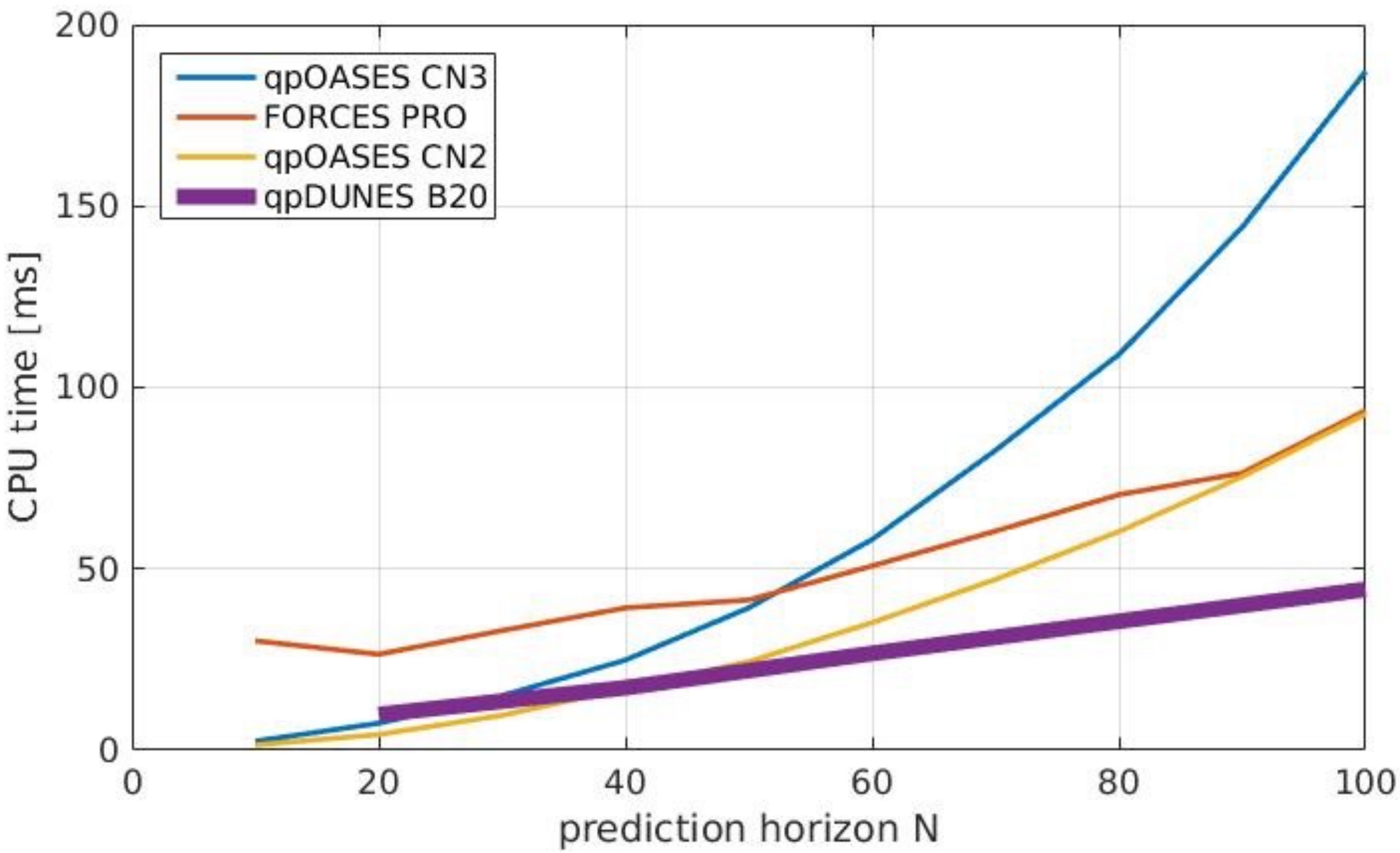
2014/15: **qpDUNES** and **Block-Partial Condensing**

Janick Frasch

- "Dual Newton Strategy" qpDUNES (Frasch et al. 2015) uses Lagrangian decomposition, solves many small QPs independently with qpOASES, and performs sparse Cholesky factorisations
- block-partial condensing (first described by Daniel Axehill) boosts performance of qpDUNES (Dimitris Kouzoupis)

Daniel Axehill

Dimitris Kouzoupis



2016: Efficient Register Management in **HPMPC**

- use interior point method with Riccati solver of **O(N) complexity**
- use register size specific blocking of dense matrices to reduce memory movements and obtain near peak CPU performance **(Gianluca Frison)**
- include in ACADO (M. Vukov, A. Zanelli)

Gianluca Frison

2016: Efficient Register Management in **HPMPC**

- use interior point method with Riccati solver of **O(N) complexity**
- use register size specific blocking of dense matrices to reduce memory movements and obtain near peak CPU performance **(Gianluca Frison)**
- include in ACADO (M. Vukov, A. Zanelli)

Gianluca Frison

Preprints, 5th IFAC Conference on Nonlinear Model Predictive Control
September 17-20, 2015. Seville, Spain

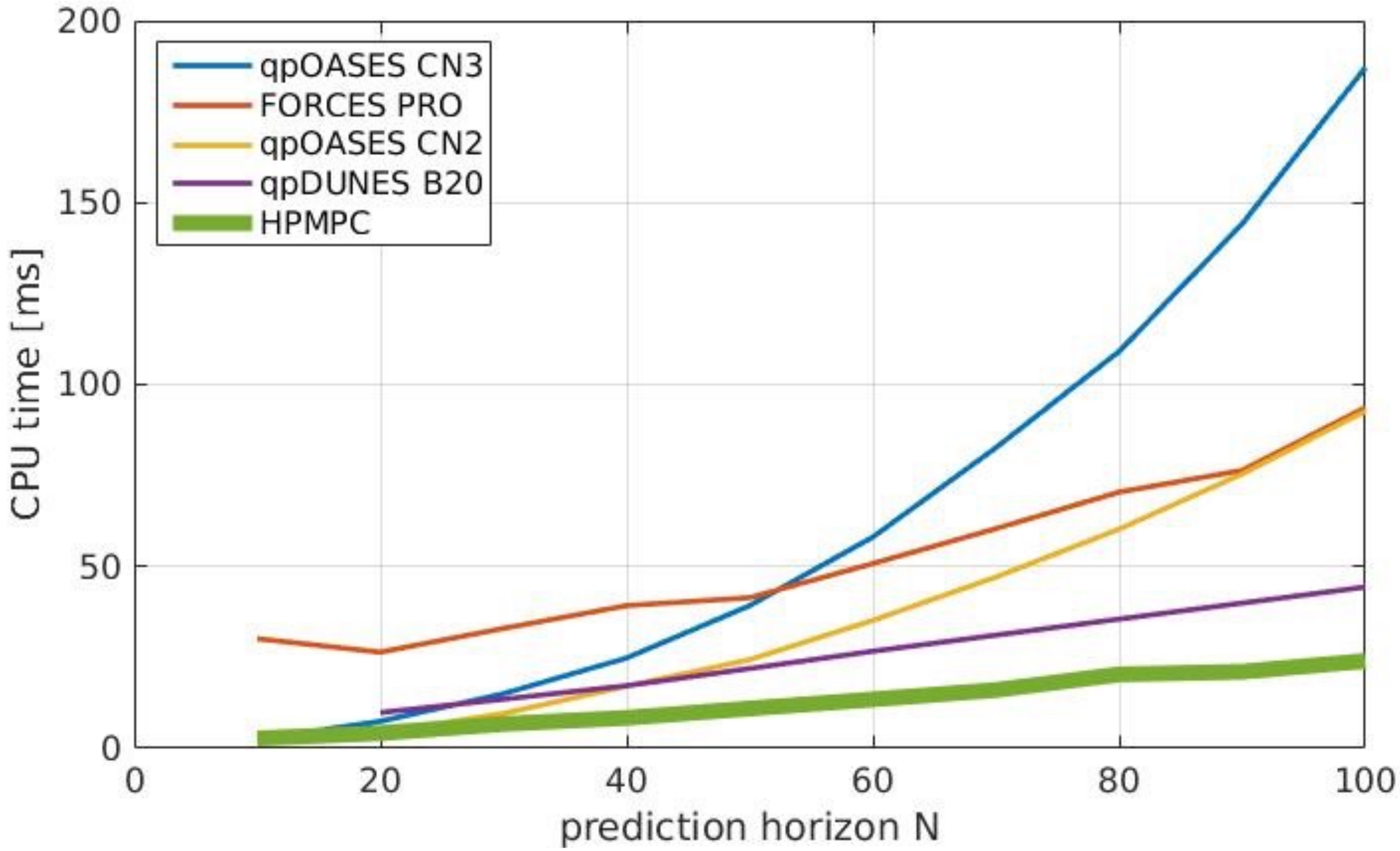
High-Performance Small-Scale Solvers for Moving Horizon Estimation

Gianluca Frison * Milan Vukov ** Niels Kjølstad Poulsen *
Moritz Diehl **,*** John Bagterp Jørgensen *

2016: Efficient Register Management in **HPMPC**

- use interior point method with Riccati solver of **O(N) complexity**
- use register size specific blocking of dense matrices to reduce memory movements and obtain near peak CPU performance **(Gianluca Frison)**
- include in ACADO (M. Vukov, A. Zanelli)

Gianluca Frison

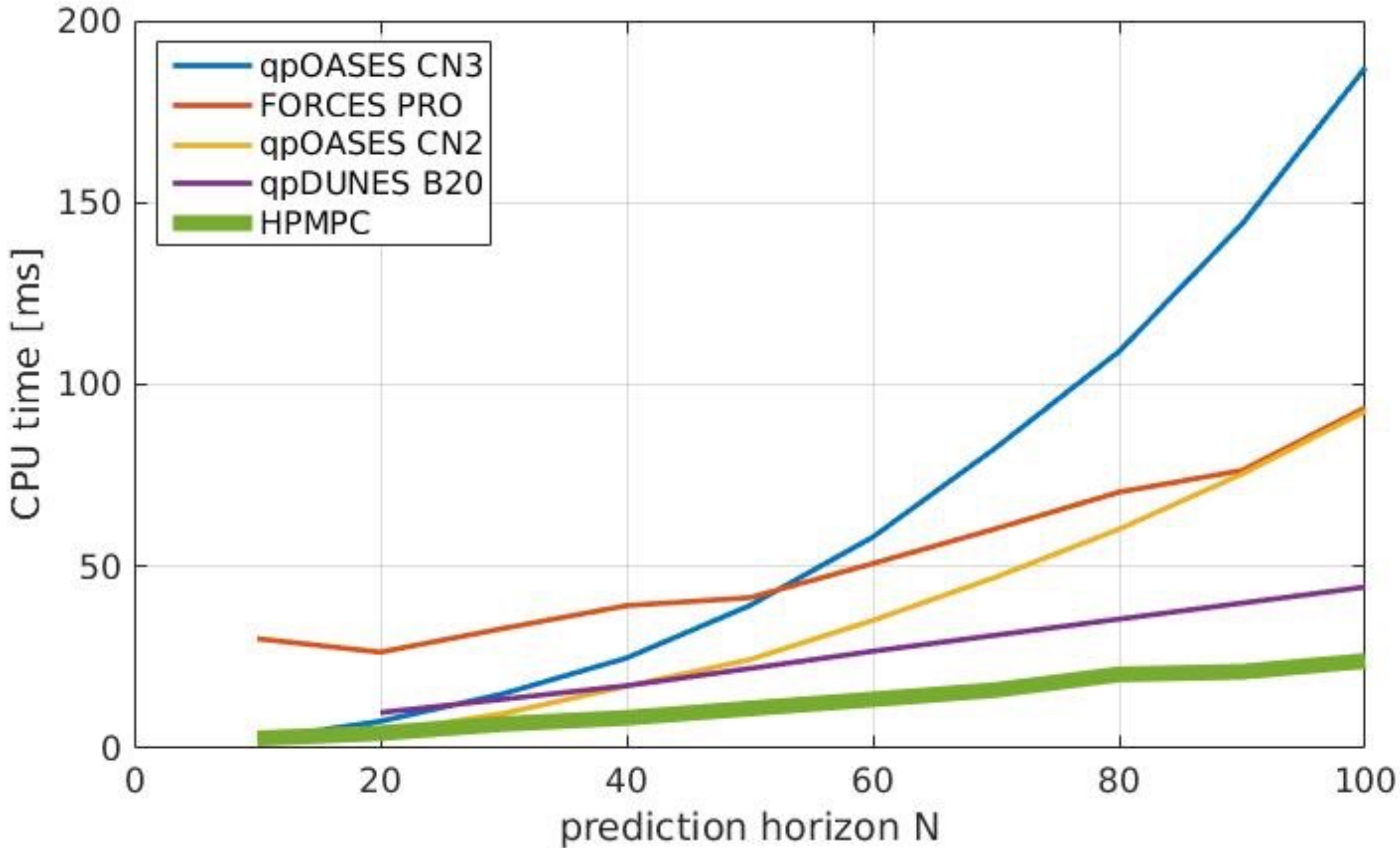


2016: Efficient Register Management in **HPMPC**

- use interior point method with Riccati solver of **O(N) complexity**
- use register size specific blocking of dense matrices to reduce memory movements and obtain near peak CPU performance **(Gianluca Frison)**
- include in ACADO (M. Vukov, A. Zanelli)

Gianluca Frison

expect further improvement for block-partial condensing (~2)



Overview

- Embedded Optimization
- Time Optimal Motions in Mechatronics
- Real-Time Optimization Methods and Software
- **Four Experimental NMPC Applications**

MPC in Practice: 2x Embedded Optimisation

SENSORS

- GPS
- acceleration
- radar
- vision
- ...

→

Moving Horizon Estimation:
fit model to data

↓

Nonlinear Model Predictive Control:
make optimal plan,
realize only first step

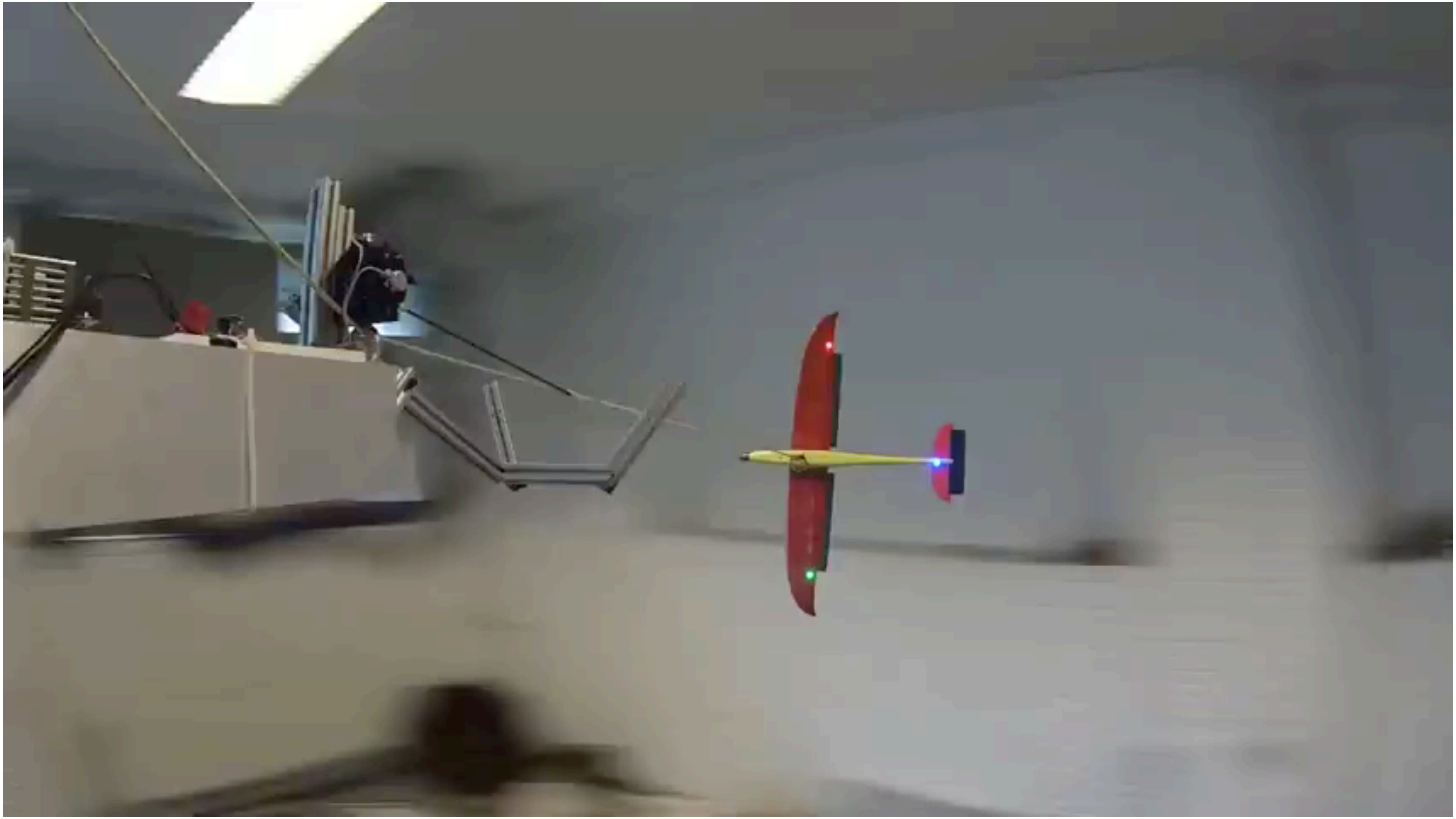
→

ACTUATORS

- flight surfaces
- steering wheel
- motor speeds
- joint torques
- …

1) Flight Carousel for Tethered Airplanes

Experiments within the ERC Project HIGHWIND Leuven/Freiburg



Milan Vukov

Moving Horizon Estimation and Nonlinear Model Predictive Control on the Flight Carousel

(sampling time 50 Hz, using ACADO Code Generation)

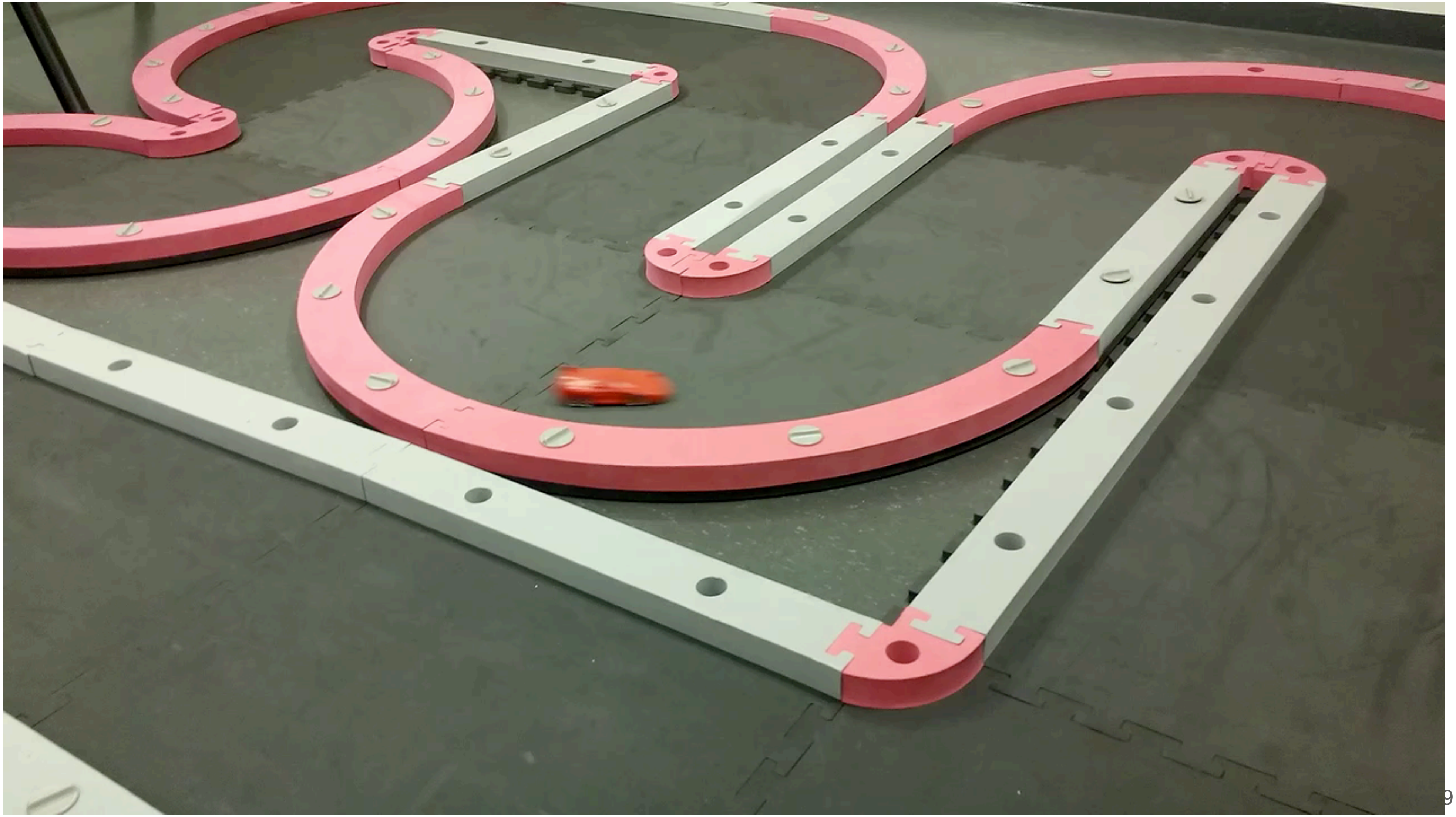
Closed loop experiments with NMPC & NMHE

2) Nonlinear MPC Example: time-optimal "racing" of model cars

Robin Verschueren

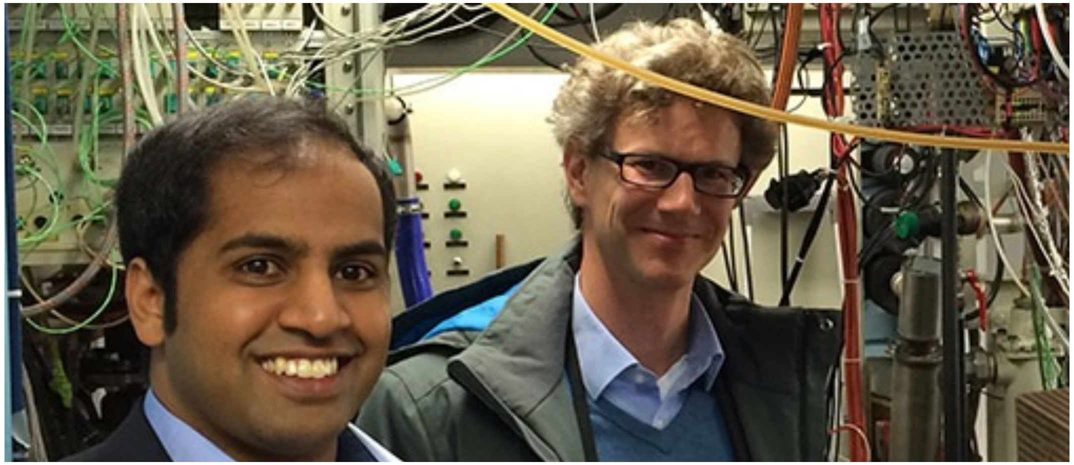
Freiburg/Leuven/ETH/Siemens-PLM. 100 Hz sampling time using ACADO [Verschueren, De Bruyne, Zanon, Frasch, D. CDC 2014]

(Nonlinear MPC video from 22.6.2016 in Freiburg)



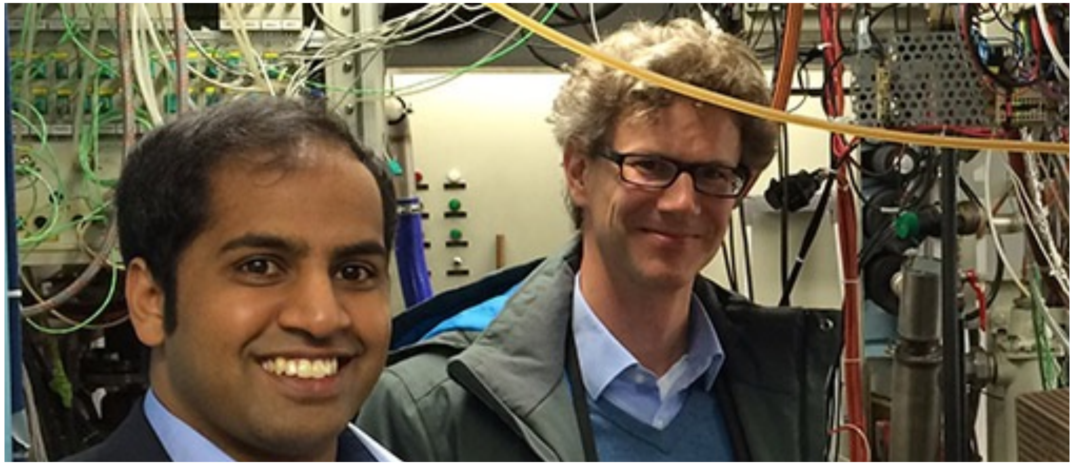
3) Nonlinear MPC of Two-Stage Turbocharger with ACADO

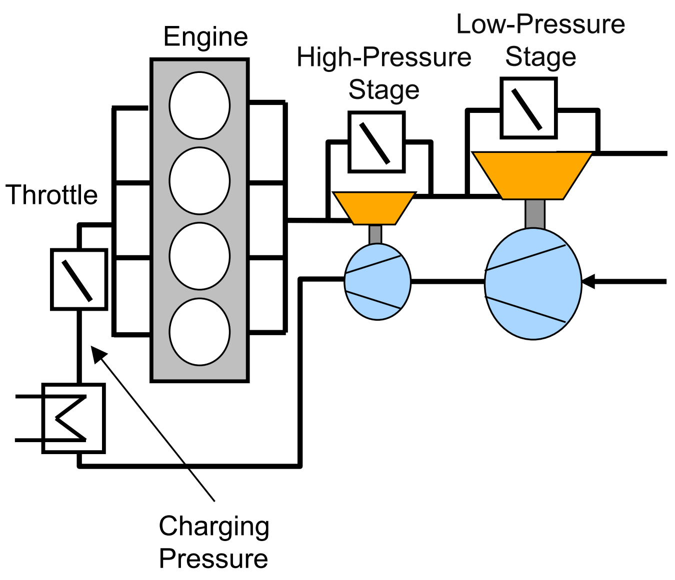
Cooperation with Dr. Thiva Albin (RWTH Aachen) and **Rien Quirynen**



3) Nonlinear MPC of Two-Stage Turbocharger with ACADO

Cooperation with Dr. Thiva Albin (RWTH Aachen) and **Rien Quirynen**





Actuated Variables

- Wastegate-
  High Pressure
- Wastegate-
  Low Pressure

Controlled Variable / Sensors

- Charging pressure
- No sensors in the exhaust gas path

test car of RWTH Aachen

3) Nonlinear MPC of Two-Stage Turbocharger with ACADO

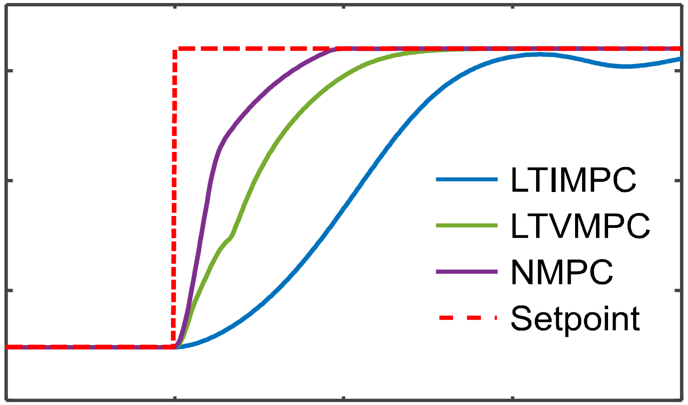
Cooperation with Dr. Thiva Albin (RWTH Aachen) and **Rien Quirynen**

- use nonlinear DAE model with 4 states, 2 controls
- use ACADO Code Generation from MATLAB
- export C-code into Simulink
- deploy on dSPACE Autobox

$$
\begin{aligned}
\frac{d}{dt}\left(\frac{1}{2}J_{tc,hp}n_{tc,hp}^2\right) &= P_{t,hp} - P_{c,hp} \\
\frac{d}{dt}\left(\frac{1}{2}J_{tc,lp}n_{tc,lp}^2\right) &= P_{t,lp} - P_{c,lp} \\
P_{c,hp} &= \dot{m}_{c,hp}c_pT_{uc,hp}\frac{1}{\eta_{s,c,hp}}\left(\Pi_{c,hp}^{\frac{\kappa-1}{\kappa}} - 1\right) \\
P_{c,lp} &= \dot{m}_{c,lp}c_pT_{amb}\frac{1}{\eta_{s,c,lp}}\left(\Pi_{c,lp}^{\frac{\kappa-1}{\kappa}} - 1\right) \\
P_{t,hp} &= \dot{m}_{t,hp}c_pT_{ut,hp}\eta_{s,t,hp}\left(1 - \Pi_{t,hp}^{\frac{1-\kappa}{\kappa}}\right) \\
P_{t,lp} &= \dot{m}_{t,lp}c_pT_{ut,lp}\eta_{s,t,lp}\left(1 - \Pi_{t,lp}^{\frac{1-\kappa}{\kappa}}\right)
\end{aligned}
$$

3) Nonlinear MPC of Two-Stage Turbocharger with ACADO

Nonlinear MPC superior to Linear MPC in simulations:



Implemented in test car of RWTH Aachen and tested on a test drive and the road.

[driving a happy M.D. to Aachen Hbf on 2.11.2015]

4) Electrical Compressor Control at ABB (Norway)

- work of Dr. Joachim Ferreau and Dr. Thomas Besselmann, ABB
- nonlinear MPC with qpOASES and ACADO, 1ms sampling time
- first tests at 48 MW Drive
- currently, 15% of Norwegian Gas Exports are controlled by Nonlinear MPC

4) Electrical Compressor Control at ABB (Norway)

- work of Dr. Joachim Ferreau and Dr. Thomas Besselmann, ABB
- nonlinear MPC with qpOASES and ACADO, 1ms sampling time
- first tests at 48 MW Drive
- currently, 15% of Norwegian Gas Exports are controlled by Nonlinear MPC

```
Joachim Ferreau (email from 7.3.2016):

The NMPC installations in Norway
(actually 5 compressors at two
different sites) are doing fine since
last autumn – roughly 80 billion NMPC
instances solved by now. In addition,
they have proven to work as expected
when handling external voltage dips.
```

Summary

embedded optimization uses more CPU resources than classical filters, but allows the development of **more powerful** nonlinear control and estimation algorithms with **wider range of validity**

good numerical methods can solve **nonlinear optimal control** problems at **milli- and microsecond sampling times**

open source tools ACADO and CasADi well-tested in dozens of optimal control and embedded MPC applications: cranes, wafer steppers, race cars, combustion engines, electrical drives, tethered airplanes, power converters,…

need **differentiable simulation models** for numerical efficiency

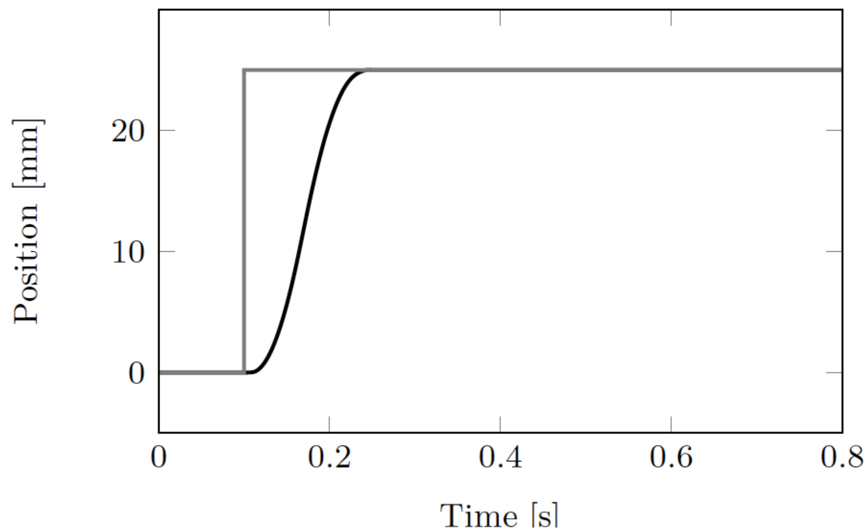
Thank you

CasADi

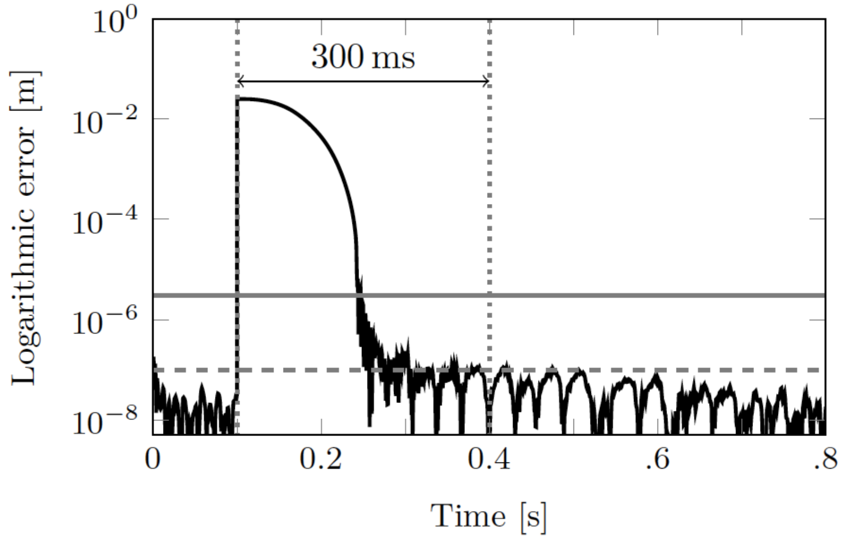
- “Computer Algebra System for Automatic Differentiation”
- Implements AD on sparse matrix-valued computational graphs
- Open-source tool (LGPL): www.casadi.org, developed by Joel Andersson and Joris Gillis
- Front-ends to C++, Python, Octave & MATLAB

- Symbolic model import from Modelica (via Jmodelica.org)
- Interfaces to: SUNDIALS, CPLEX, qpOASES, IPOPT, KNITRO, …
- “Write efficient optimal control solver in a few lines”

Time Optimal MPC at ETEL (CH): 25cm step, 100nm accuracy



TOMPC at 250 Hz (+PID with 12 kHz)



Lieboud's results after 1 week at ETEL:

- 25 cm step in 300 ms
- 100 nm accuracy

equivalent to: „fly 2,5 km with MACH15, stop with 1 mm position accuracy"