Numerical Methods for NMPC and MHE

Moritz Diehl, Joachim Ferreau, Boris Houska, Niels Haverbeke
Optimization in Engineering Center (OPTEC), K.U. Leuven, Belgium


*OPTEC, K.U. Leuven / **Stanford University / ***IWR, University of Heidelberg
(First a Distillation NMPC Story for Warm-Up)
Model Predictive Control (MPC)

Always look a bit into the future.

Brain predicts and optimizes: e.g. slow down \textbf{before} curve
Principle of Optimal Feedback Control / Nonlinear MPC: Computations in Model Predictive Control (MPC)

1. Estimate current system state \( x_0 \) (and parameters) from measurements.

2. Solve \textit{in real-time} an optimal control problem:

\[
\min_{x,z,u} \int_{t_0}^{t_0+T_p} L(x,z,u) dt + E(x(t_0+T_p)) \quad \text{s.t.} \quad \begin{cases}
    x(t_0) - x_0 = 0, \\
    \dot{x} - f(x,z,u) = 0, \quad t \in [t_0, t_0+T_p] \\
    g(x,z,u) = 0, \quad t \in [t_0, t_0+T_p] \\
    h(x,z,u) \geq 0, \quad t \in [t_0, t_0+T_p] \\
    r(x(t_0+T_p)) \geq 0.
\end{cases}
\]

3. Implement first control \( u_0 \) for time \( \delta \) at real plant. Set \( t_0 = t_0 + \delta \) and go to 1.

\textbf{Main challenge for MPC: fast and reliable real-time optimization}
Example: Distillation Column (ISR, Stuttgart)

- Aim: to ensure product purity, keep two temperatures \((T_{14}, T_{28})\) constant despite disturbances
- least squares objective:
  \[
  \min \int_{t_0}^{t_0+T_p} \left\| \begin{array}{c}
  T_{14}(t) - T_{14}^{ref} \\
  T_{28}(t) - T_{28}^{ref}
  \end{array} \right\|_2^2 dt
  \]
- control horizon 10 min
- prediction horizon 10 h
- stiff DAE model with 82 differential and 122 algebraic state variables
- Desired sampling time: 30 seconds.
Distillation Online Scenario

- System is in steady state, optimizer predicts constant trajectory:

- Suddenly, system state $x_0$ is disturbed.
- What to do with optimizer?
Conventional Approach

- use offline method, e.g. MUSCOD-II with BFGS (Leineweber, 1999).
- initialize with **new** initial value $x_0$ and integrate system with **old** controls.
- iterate until convergence.

Initialization
Conventional Approach

- use offline method, e.g. MUSCOD-II with BFGS (Leineweber, 1999).
- initialize with new initial value $x_0$ and integrate system with old controls.
- iterate until convergence.
Conventional Approach

- use offline method, e.g. MUSCOD-II with BFGS (Leineweber, 1999).
- initialize with **new** initial value $\alpha_0$ and integrate system with **old** controls.
- iterate until convergence.

**Initialization**

**16th Iteration**

**Solution (32nd Iteration)**
New Approach: Initial Value Embedding

- Initialize with old trajectory, accept violation of $s_0^x - x_0 = 0$
New Approach: Initial Value Embedding

- Initialize with old trajectory, accept violation of $s_0^x - x_0 = 0$

Initialization

First Iteration
New Approach: Initial Value Embedding

- Initialize with old trajectory, accept violation of $s^x_0 - x_q = 0$

Initialization | First Iteration | Solution (3rd Iteration)

First iteration nearly solution!
Very different results after first iteration!

Conventional:

Initial Value Embedding:
Initial Value Embedding

- first iteration is tangential predictor for exact solution (for exact hessian SQP)
- also valid for active set changes
- derivative can be computed before $x_0$ is known: first iteration nearly without delay
Initial Value Embedding

- first iteration is tangential predictor for exact solution (for exact hessian SQP)
- also valid for active set changes
- derivative can be computed before \( x_0 \) is known: first iteration nearly without delay

Why wait until convergence and do nothing in the meantime?
Real-Time Iterations [D. 2001]

Iterate, *while* problem is changing!

- tangential prediction after each change in $x_a$
- solution accuracy is increased with each iteration when $x_a$ changes little
- iterates stay close to solution manifold
Real-Time Iteration Algorithm:

1. **Preparation Step (costly):**
   Linearize system at current iterate, perform partial reduction and condensing of quadratic program.

2. **Feedback Step (short):**
   When new $x_k$ is known, solve condensed QP and implement control $u_k$ immediately.
   Complete SQP iteration. Go to 1.

- short cycle-duration (as one SQP iteration)
- negligible feedback delay ($\approx 1\%$ of cycle)
- nevertheless fully nonlinear optimization
Real-Time Iterations minimize feedback delay
Realization at Distillation Column

[D., Findeisen, Schwarzkopf, Uslu, Allgöwer, Bock, Schlöder, 2002]

- Parameter estimation using dynamic experiments
- Online state estimation with Extended Kalman Filter variant, using only 3 temperature measurements to infer all 82 system states
- Implementation of estimator and optimizer on Linux Workstation.
- Communication with Process Control System via FTP all 10 seconds.
- Self-synchronizing processes.
Computation Times During Application

**Preparation**

![Graph showing computation time over preparation phase](image)

**Feedback**

![Graph showing computation time over feedback phase](image)
Experiments with a Real Distillation Column

Feedflow Change by 20%: Transient Phase (Comparison with PI-Controller)

Transient in 15 minutes instead of 2 hours!
Large Disturbance (Heating), then NMPC

- Overheating by manual control
- NMPC only starts at $t = 1500$ s
- PI-controller not implementable, as disturbance too large (valve saturation)
- NMPC: at start control bound active $\Rightarrow T_{28}$ rises further
- Disturbance attenuated after half an hour
Real vs. Theoretical Optimal Solution

Experimental Closed-Loop

- $T_{28}$ [$^\circ$C]
- $T_{14}$ [$^\circ$C]
- $L_{vol}$ [l/h]
- $Q$ [kW]

Optimal Solution

- $T_{28}$ [$^\circ$C]
- $T_{14}$ [$^\circ$C]
- $L_{vol}$ [l/h]
- $Q$ [kW]
(Now back to the history of NMPC)
Outline of the Talk

PART I: Offline Optimal Control
- NMPC and MHE Problem Statement
- Simultaneous vs. Sequential Formulation
- Newton Type Optimization: IP vs. SQP Methods

PART II: Online Algorithms
- Parametric Sensitivities
- Review of Three Classical Algorithms

PART III: Software and Mechatronic Applications
NMPC Optimal Control Problem in Continuous Time

\[
\begin{align*}
\text{minimize} & \quad \int_0^T L(x(t), u(t)) \, dt + E(x(T)) \\
\text{subject to} & \\
& x(0) - x_0 = 0, \quad \text{(fixed initial value)} \\
& \dot{x}(t) - f(x(t), u(t)) = 0, \quad t \in [0, T], \quad \text{(ODE model)} \\
& h(x(t), u(t)) \geq 0, \quad t \in [0, T], \quad \text{(path constraints)} \\
& r(x(T)) = 0 \quad \text{(terminal constraints)}.
\end{align*}
\]

How to solve these nonlinear problems reliably and fast?
Optimal Control Family Tree

- Hamilton-Jacobi-Bellman Equation: 
  - Tabulation in State Space

- Indirect Methods, Pontryagin: 
  - Solve Boundary Value Problem

- Direct Methods: 
  - Transform into Nonlinear Program (NLP)

- Single Shooting: 
  - Only discretized controls in NLP (sequential)

- Collocation: 
  - Discretized controls and states in NLP (simultaneous)

- Multiple Shooting: 
  - Controls and node start values in NLP (simultaneous)
Optimal Control Family Tree

(curse of dimensionality)

Hamilton-Jacobi-Bellman Equation: 
*Tabulation in State Space*

Indirect Methods, Pontryagin: 
*Solve Boundary Value Problem*

Direct Methods: 
*Transform into Nonlinear Program (NLP)*

Single Shooting: 
*Only discretized controls in NLP (sequential)*

Collocation: 
*Discretized controls and states in NLP (simultaneous)*

Multiple Shooting: 
*Controls and node start values in NLP (simultaneous)*
Optimal Control Family Tree

(curse of dimensionality)

Hamilton-Jacobi-Bellman Equation:
- Tabulation in State Space

(bad inequality treatment)

Indirect Methods, Pontryagin:
- Solve Boundary Value Problem

Direct Methods:
- Transform into Nonlinear Program (NLP)

Single Shooting:
- Only discretized controls in NLP (sequential)

Collocation:
- Discretized controls and states in NLP (simultaneous)

Multiple Shooting:
- Controls and node start values in NLP (simultaneous)
Optimal Control Family Tree

Hamilton-Jacobi-Bellman Equation: Tabulation in State Space

Indirect Methods, Pontryagin: Solve Boundary Value Problem

Direct Methods: Transform into Nonlinear Program (NLP)

Single Shooting: Only discretized controls in NLP (sequential)

Collocation: Discretized controls and states in NLP (simultaneous)

Multiple Shooting: Controls and node start values in NLP (simultaneous)
NMPC Problem in Discrete Time

\[
\begin{align*}
\text{minimize} & \quad \sum_{i=0}^{N-1} L_i(x_i, z_i, u_i) + E(x_N) \\
\text{subject to} & \quad x_0 - \bar{x}_0 = 0, \\
& \quad x_{i+1} - f_i(x_i, z_i, u_i) = 0, \quad i = 0, \ldots, N - 1, \\
& \quad g_i(x_i, z_i, u_i) = 0, \quad i = 0, \ldots, N - 1, \\
& \quad h_i(x_i, z_i, u_i) \leq 0, \quad i = 0, \ldots, N - 1, \\
& \quad r(x_N) \leq 0.
\end{align*}
\]

Structured parametric Nonlinear Program, “mp-NLP”

- Initial Value $\bar{x}_0$ is not known beforehand ("online data")
- Discrete time dynamics often come from ODE simulation ("shooting")
- "Algebraic States" $z$ implicitly defined via third condition, can come from DAEs or from collocation discretization
NMPC = mp-NLP

- Solution manifold is piecewise differentiable
- Critical regions are non-polyhedral
(MHE Problem)

\[
\begin{align*}
\text{minimize} & \quad \|x_0 - \bar{x}_0\|_P^2 + \sum_{i=0}^{N-1} \|y_i - m_i(x_i, z_i, u_i, w_i)\|_Q^2 + \|w_i\|_R^2 \\
\text{subject to} & \quad x_{i+1} - f_i(x_i, z_i, u_i, w_i) = 0, \quad i = 0, \ldots, N - 1, \\
& \quad g_i(x_i, z_i, u_i, w_i) = 0, \quad i = 0, \ldots, N - 1, \\
& \quad h_i(x_i, z_i, u_i, w_i) \leq 0, \quad i = 0, \ldots, N - 1,
\end{align*}
\]

- Online problem data: \(y_i\)
- “Controls” \(w\) account for unknown disturbances. Often many \(w\).
- Initial value is free
- Non-Euclidean norms (L1) can be advantageous (Poster Haverbeke)
Sequential Approach (Single Shooting): Eliminate States

\[
\begin{align*}
\text{minimize} & \sum_{i=0}^{N-1} L_i(\tilde{x}_i(u), \tilde{z}_i(u), u_i) + E(\tilde{x}_N(u)) \\
\text{subject to} & h_i(\tilde{x}_i(u), \tilde{z}_i(u), u_i) \leq 0, \quad i = 0, \ldots, N - 1, \\
& r(\tilde{x}_N(u)) \leq 0.
\end{align*}
\]

Pros:
- Only control degrees of freedom (for NMPC)
- Can couple with “Vanilla NLP” solver

Cons:
- Sparsity of problem lost
- Unstable systems cannot be treated

Historically first “direct” approach (“single shooting”, Sargent&Sullivan 1978)
Simultaneous Approach: Keep States in NLP

**A MULTIPLE SHOOTING ALGORITHM FOR DIRECT SOLUTION OF OPTIMAL CONTROL PROBLEMS**

Hans Georg Bock and Karl J. Plitt

Institut für Angewandte Mathematik, SFB 72, Universität Bonn, 5300 Bonn, Federal Republic of Germany

**Variants:**
Multiple Shooting and Collocation

**Pros:**
- Sparsity of problem kept
- Unstable systems can be treated, nonlinearity reduced

**Cons:**
- Large scale problems
- Need to develop (or use) structure exploiting NLP solver
Outline of the Talk

PART I: Offline Optimal Control

- NMPC and MHE Problem Statement
- Simultaneous vs. Sequential Formulation
- Newton Type Optimization: IP vs. SQP Methods

PART II: Online Algorithms

- Parametric Sensitivities
- Review of Three Classical Algorithms

PART III: Software and Mechatronic Applications
How to solve Nonlinear Programs (NLPs) ?

\[
\begin{align*}
\text{minimize} & \quad F(X) \quad \text{s.t.} \quad \left\{ \begin{array}{c}
G(X) = 0 \\
H(X) \leq 0
\end{array} \right.
\end{align*}
\]

Lagrangian: \[ \mathcal{L}(X, \lambda, \mu) = F(X) + G(X)^T \lambda + H(X)^T \mu \]

Karush Kuhn Tucker (KKT) conditions: for optimal \( X^* \) exist \( \lambda^*, \mu^* \) such that:

\[
\nabla_X \mathcal{L}(X^*, \lambda^*, \mu^*) = 0
\]

\[
G(X^*) = 0
\]

\[
0 \geq H(X^*) \perp \mu^* \geq 0.
\]

Newton type methods try to find points satisfying these conditions. But last condition non-smooth: cannot apply Newton’s method directly. What to do?
Approach 1: Interior Point (IP) Methods

- Replace last condition by smoothed version:

\[
\nabla_X L(X^*, \lambda^*, \mu^*) = 0 \\
G(X^*) = 0 \\
- H_i(X^*) \mu_i^* = \tau, \quad i = 1, \ldots, n_H.
\]

Summarize as \( R(W) = 0 \)

- Solve with Newton’s method, i.e.,
  - Linearize at current guess \( W^k = (X^k, \lambda^k, \mu^k) \):
    \[
    R(W^k) + \nabla R(W^k)^T (W^{k+1} - W^k) = 0
    \]
  - solve linearized system, get new trial point

- For \( \tau \) small, IP problem gets close to original (path-following, self-concordance, polynomial time for convex problems, …)
(Note: IP with fixed $\tau$ makes mp-NLP smooth)
Approach 2: Sequential Quadratic Programming (SQP)


ALGORITHMS FOR NONLINEAR CONSTRAINTS THAT USE LAGRANGIAN FUNCTIONS*

M.J.D. POWELL

University of Cambridge, Cambridge, United Kingdom

Received 10 October 1976

- Linearize all problem functions, solve Quadratic Program (QP):

\[
\begin{align*}
\text{minimize} & \quad F^k_{\text{QP}}(X) \quad \text{s.t.} \\
& G(X^k) + \nabla G(X^k)^T(X - X^k) = 0 \\
& H(X^k) + \nabla H(X^k)^T(X - X^k) \leq 0
\end{align*}
\]

with convex quadratic objective using an approximation of Hessian.

Obtain new guesses for both $X^*$ and $\lambda^*, \mu^*$. 

(Important Variant of SQP: Generalized Gauss-Newton)

If objective is sum of squares:

\[ F(X) = \frac{1}{2} \| R(X) \|^2. \]

then QP objective can well be approximated by:

\[ F_{Q\text{P}}^k(X) = \frac{1}{2} \| R(X^k) + \nabla R(X^k)^T (X - X^k) \|^2. \]

(often extremely good linear convergence)
Difference between IP and SQP?

- Both generate sequence of iterates $X^k, \lambda^k, \mu^k$
- Both need to linearize problem functions in each iteration.

- IP iterations cheaper:
  - IP solves only linear system
  - SQP solves a QP in each iteration (maybe even with an IP method!)

- IP needs more iterations:
  - IP multipliers change slowly, iterates always in interior
  - SQP multipliers jump, active set can quickly be identified

SQP good if problem function evaluations are expensive (shooting methods)
SQP even works if all QP matrices are old. Only constraints and Lagrange gradient (cheap by adjoint differentiation) need to be exact.

Trick: use “modified gradient”

\[ a_k = \nabla_X \mathcal{L}(X^k, \lambda^k, \mu^k) - B_k \lambda^k - C_k \mu^k \]

in QP objective

\[ F^k_{\text{adjQP}}(X) = a_k^T X + \frac{1}{2} (X - X^k)^T A_k (X - X^k). \]

Solve QP with inexact constraints

\[
\begin{array}{l}
\text{minimize} \quad F^k_{\text{adjQP}}(X) \\
\text{s.t.} \quad 
\begin{cases} 
G(X^k) + B_k^T (X - X^k) = 0 \\
H(X^k) + C_k^T (X - X^k) \leq 0.
\end{cases}
\end{array}
\]

In each SQP iteration, solve structured QP:

\[
\begin{align*}
\text{minimize} & \quad \sum_{i=0}^{N-1} L_{QP,i}(x_i, z_i, u_i) + E_{QP}(x_N) \\
\text{subject to} & \quad x_0 - \bar{x}_0 = 0, \\
& \quad x_{i+1} - f'_i - F^x_i x_i - F^z_i z_i - F^u_i u_i = 0, \quad i = 0, \ldots, N-1, \\
& \quad g'_i + G^x_i x_i + G^z_i z_i + G^u_i u_i = 0, \quad i = 0, \ldots, N-1, \\
& \quad h'_i + H^x_i x_i + H^z_i z_i + H^u_i u_i \leq 0, \quad i = 0, \ldots, N-1, \\
& \quad r' + R x_N \leq 0.
\end{align*}
\]

Algebraic Reduction: first eliminate $z$
QP after Algebraic Reduction

minimize  \[
\sum_{i=0}^{N-1} L_{\text{redQP},i}(x_i, u_i) + E_{\text{QP}}(x_N)
\]

subject to
\[
x_0 - \bar{x}_0 = 0,
\]
\[
x_{i+1} - c_i - A_i x_i - B_i u_i = 0, \quad i = 0, \ldots, N - 1,
\]
\[
\bar{h}_i + \bar{H}_i^x x_i + \bar{H}_i^u u_i \leq 0, \quad i = 0, \ldots, N - 1,
\]
\[
r' + R x_N \leq 0.
\]

How to solve this structured QP?
Approach 1: Banded Factorization

- Factorize large banded KKT Matrix e.g. by Riccati based recursion

\[
M = \begin{bmatrix}
\mathbb{I} & Q_0 & S_0 & -A_0^T \\
\mathbb{I} & Q_0 & S_0 & -A_0^T \\
S_0^T & R_0 & -B_0^T \\
-A_0 & -B_0 & \ddots & \mathbb{I} \\
\mathbb{I} & Q_N \end{bmatrix}
\]

- Advantageous for long horizons and many controls (MHE).
- Niels Haverbeke develops fast Riccati-Kalman QP solvers: e.g. MHE with 200 steps, 10 states, 10 “controls” (4000 x 4000 matrix): 20 ms
Approach 2: Condensing - Eliminate all States

- Eliminate states by linear system simulation, keep only controls in QP, solve QP with dense solver

\[
\begin{align*}
\text{minimize} & \quad f_{\text{condQP},i}(\bar{x}_0, u) \\
\text{subject to} & \quad \bar{r} + \bar{R}_x \bar{x}_0 + \bar{R}_u u \leq 0.
\end{align*}
\]

- Note: mp-QP in same form as in Alberto Bemporad’s Keynote
- Can use this QP as fast feedback law for several \( \bar{x}_0 \) (Poster Albersmeyer et al.). We come back to this.
- But QP matrices change after each SQP re-linearization
Classification of Newton Type Optimal Control

(a) Treatment of Inequalities: Nonlinear IP vs. SQP
(b) Nonlinear Iterations: Simultaneous vs. Sequential
(c) Derivative Computations: Full vs. Reduced
(d) Linear Algebra: Banded vs. Condensing
Outline of the Talk

PART I: Offline Optimal Control

- NMPC and MHE Problem Statement
- Simultaneous vs. Sequential Formulation
- Newton Type Optimization: IP vs. SQP Methods

PART II: Online Algorithms

- Parametric Sensitivities
- Review of Three Classical Algorithms

PART III: Software and Mechatronic Applications
Outline of the Talk

PART I: Offline Optimal Control
- NMPC and MHE Problem Statement
- Simultaneous vs. Sequential Formulation
- Newton Type Optimization: IP vs. SQP Methods

PART II: Online Algorithms
- Parametric Sensitivities
- Review of Three Classical Algorithms

PART III: Software and Mechatronic Applications
• SQP type method
• single shooting
• perform only one SQP iteration per problem (“real-time iteration”)
• Method also implemented in “NEPSAC Algorithm” by [De Keyser 1998] and many others [IPCOS, GE, …]. Many applications.
• was missing one important feature...parametric sensitivities
In IP case, smoothed KKT conditions are equivalent to parametric root finding problem: $R(\bar{x}_0, W) = 0$

with solution $W^*(\bar{x}_0)$ depending on initial condition

Based on old solution, can get “tangential predictor” for new one:
Can obtain sensitivity nearly for free in Newton type methods:

\[ W' = W - \left( \frac{\partial R}{\partial W}(\bar{x}_0, W) \right)^{-1} \left[ \frac{\partial R}{\partial \bar{x}_0}(\bar{x}_0, W) \left( \bar{x}'_0 - \bar{x}_0 \right) + R(\bar{x}_0, W) \right] \]
“IP real-time iteration” for sequence of NLPs
A continuation/GMRES method for fast computation of nonlinear receding horizon control

Toshiyuki Ohtsuka*

Department of Computer-Controlled Mechanical Systems, Graduate School of Engineering, Osaka University, 2-1 Yamadaoka, Suita, Osaka 565-0871, Japan

Additional features of Continuation/GMRES:

- matrix free and iterative linear algebra via GMRES
- Interior Point formulation via quadratic slacks
- Single shooting with adjoint gradient computation
- (recently extended to multiple shooting with condensing, faster contraction)

But: IP real-time iteration “overshoots” at active set changes
Ohtsuka (2004) uses for NMPC a variant of IP methods:

\[
\begin{align*}
\text{minimize} \quad & F(X) - \tau \sum_{i=1}^{n_H} Y_i \\
\text{s.t.} \quad & G(X) = 0 \\
& H_i(X) + Y_i^2 = 0, \quad i = 1, \ldots, n_H.
\end{align*}
\]

Seems to work well for fixed penalty parameter. But no self-concordance properties as in usual IP methods.
Solve a full QP with “initial value embedding” [D. et al. 2002].

\[
\begin{align*}
\text{minimize} & \quad f_{\text{condQP},t}(\tilde{x}_0, u) \\
\text{subject to} & \quad \tilde{r} + \tilde{R}^0 \tilde{x}_0 + \tilde{R}^u u \leq 0.
\end{align*}
\]

At smooth parts, delivers same predictor-corrector step as Newton. But is “Generalized Tangential Predictor” valid also across active set changes:
SQP Real-Time Iteration [D. et al 2002]

- long “preparation phase” for linearization, reduction, and condensing
- fast “feedback phase” (QP solution once $\bar{x}_0$ is known). Fast, but…
Stability of System-Optimizer Dynamics?

- System and optimizer are coupled: can numerical errors grow and destabilize closed loop?
- Stability analysis combines concepts from both, **NMPC stability theory** and **convergence theory of Newton-type optimization**.
- Stability shown under mild assumptions (short sampling times, stable NMPC scheme) [Diehl, Findeisen, Allgöwer, 2005]
- Losses w.r.t. optimal feedback control are $O(\kappa^2 \epsilon^2)$ after $\epsilon$ disturbance [Diehl, Bock, Schlöder, 2005]
Newest ideas in opposite directions

**Multi-Level Real-Time Iterations**
[Bock, D. et al. NMPC 05, Wirsching 2007]
Make real-time iterations cheaper.
Four Levels:
A) mp-QP at innermost level
B) Feasibility improvement
C) Optimality Improvement
D) Full re-linearization, only rarely in outer loop
- Allows extremely fast sampling rates at innermost level A (feedback phase).
- Level C allows to converge to NLP solution WITHOUT NEW JACOBIAN EVALUATIONS.

**Advanced Step NMPC**
[Zavala and Biegler 2007]
Combine two well-tested ideas [D. 2001]
- Preparation vs. Feedback Phase
- Tangential Predictor in Feedback
with two new building blocks
- For preparation, iterate next problem to convergence via IP method
- Use IP predictor in feedback phase
Outline of the Talk

PART I: Offline Optimal Control
- NMPC and MHE Problem Statement
- Simultaneous vs. Sequential Formulation
- Newton Type Optimization: IP vs. SQP Methods

PART II: Online Algorithms
- Parametric Sensitivities
- Review of Three Classical Algorithms

PART III: Software and Mechatronic Applications
Solution of Periodic Optimization Problem

Maximize mean power production:
by varying line thickness, period
duration, controls,
subject to
periodicity and other constraints:

Cable 1.3 km long, 7 cm thick,
Kite Area 500 m$^2$, Power 5 MW.
Kite NMPC Problem solved with ACADO

- 9 states, 3 controls
- Penalize deviation from “lying eight”
- Predict half period
- Zero terminal constraint
- 10 multiple shooting intervals

Solve with **SQP real-time iterations**
Kite NMPC: CPU Time per RTI below 50 ms

<table>
<thead>
<tr>
<th>Component</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial-Value Embedding</td>
<td>0.03 ms</td>
</tr>
<tr>
<td>QP solution (qpOASES)</td>
<td>2.23 ms</td>
</tr>
<tr>
<td>Feedback Phase:</td>
<td>3 ms</td>
</tr>
<tr>
<td>(QP after condensing: 30 vars. / 240 constr.)</td>
<td></td>
</tr>
<tr>
<td>Expansion of the QP</td>
<td>0.10 ms</td>
</tr>
<tr>
<td>Simulation and Sensitivities</td>
<td>44.17 ms</td>
</tr>
<tr>
<td>Condensing (Phase I)</td>
<td>2.83 ms</td>
</tr>
<tr>
<td>Preparation Phase:</td>
<td>47 ms</td>
</tr>
</tbody>
</table>

(on Intel Core 2 Duo CPU T7250, 2 GHz)
Summary

- Four classical ideas for Nonlinear MPC:
  - simultaneous optimization: keep states in problem
  - real-time iteration: use linearization in non-converged points
  - fast feedback phase to avoid delays
  - solve mp-QP to get tangential predictor across active set changes
- We develop open source codes for fast MPC (qpOASES, ACADO)
- Powerful tool in mechatronic MPC applications