NTNU | Norwegian University of Science and Technology

TTK4135 – Lecture 20
Summing up / Nonlinear MPC / Control applications

Lecturer: Lars Imsland

Outline

- Summing up optimization
- Today: Control and (nonlinear) optimization
  - Nonlinear MPC, some “practical”/industrial issues on MPC
  - Reference: F&H 4.5, 4.6
- Some examples from literature using concepts from this course

Numerical optimization in a nutshell (in this course)

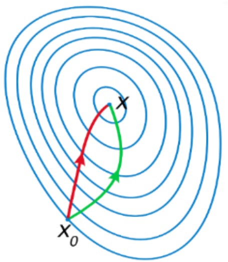
- Unconstrained optimization
  - Search directions: Steepest descent, Newton, Quasi-Newton
  - Globalization: Line-search and (possibly) Hessian modification (**Alg. 6.1**)
- Constrained optimization
  - Optimality conditions, KKT
  - Linear programming: Standard form, KKT conditions, BFPs, SIMPLEX method (**Proc. 13.1**)
  - Quadratic programming: EQP/QP, KKT system, Active set method (**Alg. 16.3**)
  - Nonlinear programming: Nonlinear equations (**Alg. 11.1**), SQP-method (**Alg. 18.3**)

Learning goal Ch. 2, 3 and 6: Understand this slide

Line-search unconstrained optimization

$$\min_x f(x)$$

1. Initial guess x_0
2. While termination criteria not fulfilled
   a) Find descent direction p_k from x_k
   b) Find appropriate step length α_k ; set $x_{k+1} = x_k + \alpha_k p_k$
   c) $k = k+1$
3. $x_M = x^*$? (possibly check sufficient conditions for optimality)



A comparison of steepest descent and Newton's method. Newton's method uses curvature information to take a more direct route. (wikipedia.org)

Termination criteria:
Stop when first of these become true:

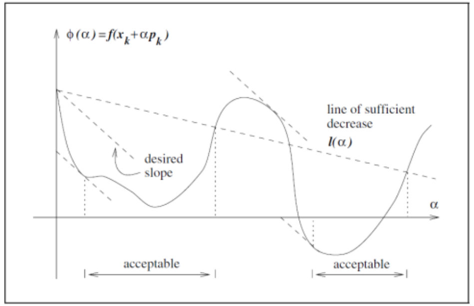
- $\|\nabla f(x_k)\| \leq \epsilon$ (necessary condition)
- $\|x_k - x_{k-1}\| \leq \epsilon$ (no progress)
- $\|f(x_k) - f(x_{k-1})\| \leq \epsilon$ (no progress)
- $k \leq k_{\max}$ (kept on too long)

Descent directions:

- Steepest descent
  $p_k = -\nabla f(x_k)$
- Newton
  $p_k = -(\nabla^2 f(x_k))^{-1} \nabla f(x_k)$
- Quasi-Newton
  $p_k = -B_k^{-1} \nabla f(x_k)$
  $B_k \approx \nabla^2 f(x_k)$

How to calculate derivatives – Ch. 8

Step length line search (Wolfe):



How many iterations? (Convergence rates)

Quasi-Newton: BFGS method

$$p_k = -B_k^{-1}\nabla f(x_k)$$
$$B_k \approx \nabla^2 f(x_k)$$
$$H_k = B_k^{-1}$$

Algorithm 6.1 (BFGS Method).

Given starting point x_0, convergence tolerance $\epsilon > 0$,
inverse Hessian approximation H_0;

$k \leftarrow 0$;

while $\|\nabla f_k\| > \epsilon$;

Compute search direction

$$p_k = -H_k \nabla f_k;$$

Set $x_{k+1} = x_k + \alpha_k p_k$ where α_k is computed from a line search procedure to satisfy the Wolfe conditions (3.6);

Define $s_k = x_{k+1} - x_k$ and $y_k = \nabla f_{k+1} - \nabla f_k$;

We use only gradient!

Compute H_{k+1} by means of (6.17);

$k \leftarrow k + 1$;

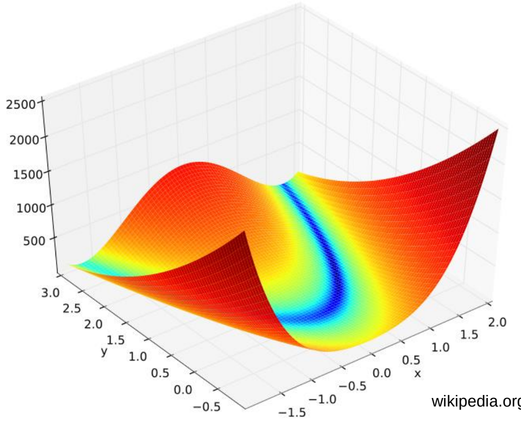
end (while)

$$H_{k+1} = (I - \rho_k s_k y_k^T) H_k (I - \rho_k y_k s_k^T) + \rho_k s_k s_k^T$$

Example (from book)

- Using steepest descent, BFGS and inexact Newton on Rosenbrock function

$$f(x) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2$$



wikipedia.org

- Iterations from starting point (-1.2,1):
  - Steepest descent: 5264
  - BFGS: 34
  - Newton: 21

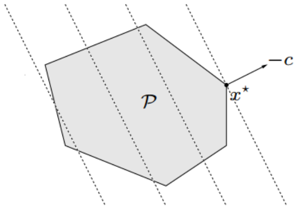
- Last iterations; value of $\|x_k - x^*\|$

steepest descent	BFGS	Newton
1.827e-04	1.70e-03	3.48e-02
1.826e-04	1.17e-03	1.44e-02
1.824e-04	1.34e-04	1.82e-04
1.823e-04	1.01e-06	1.17e-08

Types of Constrained Optimization Problems

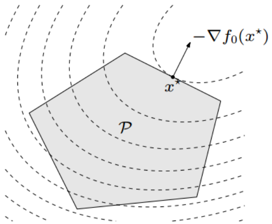
- Linear programming
  - Convex problem
  - Feasible set polyhedron

$$\begin{aligned} \min \quad & c^\top x \\ \text{subject to} \quad & Ax \le b \\ & Cx = d \end{aligned}$$



- Quadratic programming
  - Convex problem if $P \geq 0$
  - Feasible set polyhedron

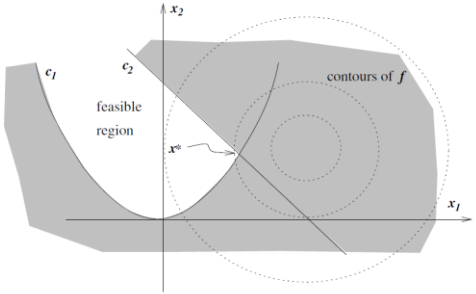
$$\begin{aligned} \min \quad & \frac{1}{2} x^\top P x + q^\top x \\ \text{subject to} \quad & Ax \le b \\ & Cx = d \end{aligned}$$



- Nonlinear programming
  - In general non-convex!

$$\begin{aligned} \min \quad & f(x) \\ \text{subject to} \quad & g(x) = 0 \\ & h(x) \ge 0 \end{aligned}$$

$$\min_{x \in \mathbb{R}^n} f(x) \qquad \text{subject to} \quad \begin{aligned} & c_i(x) = 0, \quad i \in \mathcal{E}, \\ & c_i(x) \ge 0, \quad i \in \mathcal{I}. \end{aligned}$$



KKT conditions (Theorem 12.1)

$$\min_{x \in \mathbb{R}^n} f(x) \qquad \text{subject to} \quad \begin{array}{ll} c_i(x) = 0, & i \in \mathcal{E}, \\ c_i(x) \geq 0, & i \in \mathcal{I}. \end{array}$$

KKT-conditions (First-order necessary conditions): If x^* is a local solution and LICQ holds, then there exist λ^* such that

$$\begin{aligned} \nabla_x \mathcal{L}(x^*, \lambda^*) &= 0, && && \text{(stationarity)} \\ c_i(x^*) &= 0, && \forall i \in \mathcal{E}, && \\ c_i(x^*) &\geq 0, && \forall i \in \mathcal{I}, && \text{(primal feasibility)} \\ \lambda_i^* &\geq 0, && \forall i \in \mathcal{I}, && \text{(dual feasibility)} \\ \lambda_i^* c_i(x^*) &= 0, && \forall i \in \mathcal{E} \cup \mathcal{I}. && \text{(complementarity condition/ complementary slackness)} \end{aligned}$$

Starting point for all algorithms for constrained optimization in this course!

Linear programming, standard form and KKT

LP: $\min_{x \in \mathbb{R}^n} c^T x$ subject to $\begin{cases} a_i x = b_i, & i \in \mathcal{E} \\ a_i x \geq b_i, & i \in \mathcal{I} \end{cases}$

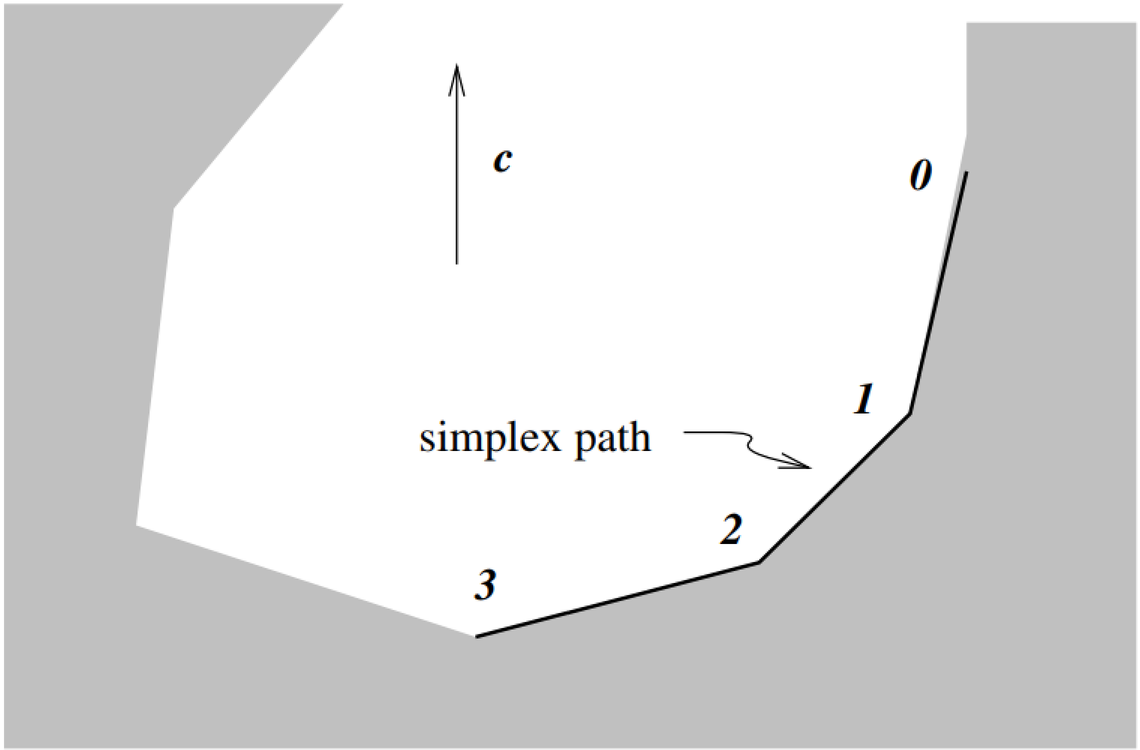
LP, standard form: $\min_{x \in \mathbb{R}^n} c^T x$ subject to $\begin{cases} Ax = b \\ x \geq 0 \end{cases}$

Lagrangian: $\mathcal{L}(x, \lambda, s) = c^T x - \lambda^T (Ax - b) - s^T x$

KKT-conditions (LPs: necessary *and* sufficient for optimality):

$$
\begin{aligned}
A^T \lambda^* + s^* &= c, \\
Ax^* &= b, \\
x^* &\geq 0, \\
s^* &\geq 0, \\
x_i^* s_i^* &= 0, \quad i = 1, 2, \ldots, n
\end{aligned}
$$

Simplex method: BFP and KKT



General QP problem

$$\min_x \frac{1}{2} x^\top G x + x^\top c$$
$$\text{s.t. } a_i^\top x = b_i, \qquad i \in \mathcal{E}$$
$$a_i^\top x \geq b_i, \qquad i \in \mathcal{I}$$

- Lagrangian

$$\mathcal{L}(x^*, \lambda^*) = \frac{1}{2} x^\top G x + x^\top c - \sum_{i \in \mathcal{E} \cup \mathcal{I}} \lambda_i (a_i^\top x - b_i)$$

- KKT conditions

General:

$$\boxed{\begin{aligned} Gx^* + c - \sum_{i \in \mathcal{E} \cup \mathcal{I}} \lambda_i^* a_i &= 0 \\ a_i^\top x^* &= b_i, & i \in \mathcal{E} \\ a_i^\top x^* &\geq b_i, & i \in \mathcal{I} \\ \lambda_i^* &\geq 0, & i \in \mathcal{I} \\ \lambda_i^* (a_i^\top x^* - b_i) &= 0, & i \in \mathcal{E} \cup \mathcal{I} \end{aligned}}$$

Defined via active set:

$$\boxed{\begin{aligned} &\mathcal{A}(x^*) = \mathcal{E} \cup \{ i \in \mathcal{I} | a_i^\top x^* = b_i \} \\ &Gx^* + c - \sum_{i \in \mathcal{A}(x^*)} \lambda_i^* a_i = 0 \\ &a_i^\top x^* = b_i, \qquad i \in \mathcal{A}(x^*) \\ &a_i^\top x^* \geq b_i, \qquad i \in \mathcal{I} \backslash \mathcal{A}(x^*) \\ &\lambda_i^* \geq 0, \qquad i \in \mathcal{A}(x^*) \cap \mathcal{I} \end{aligned}}$$

Active set method for convex QP

Algorithm 16.3 (Active-Set Method for Convex QP).

Compute a feasible starting point x_0;
Set $\mathcal{W}_0$ to be a subset of the active constraints at x_0;
for $k = 0, 1, 2, \ldots$
  Solve (16.39) to find p_k;
  if $p_k = 0$
    Compute Lagrange multipliers $\hat{\lambda}_i$ that satisfy (16.42),
      with $\hat{\mathcal{W}} = \mathcal{W}_k$;
    if $\hat{\lambda}_i \geq 0$ for all $i \in \mathcal{W}_k \cap \mathcal{I}$
      stop with solution $x^* = x_k$;
    else
      $j \leftarrow \arg\min_{j \in \mathcal{W}_k \cap \mathcal{I}} \hat{\lambda}_j$;
      $x_{k+1} \leftarrow x_k$; $\mathcal{W}_{k+1} \leftarrow \mathcal{W}_k \backslash \{j\}$;
  else (* $p_k \neq 0$ *)
    Compute α_k from (16.41);
    $x_{k+1} \leftarrow x_k + \alpha_k p_k$;
    if there are blocking constraints
      Obtain $\mathcal{W}_{k+1}$ by adding one of the blocking
        constraints to $\mathcal{W}_k$;
    else
      $\mathcal{W}_{k+1} \leftarrow \mathcal{W}_k$;
end (for)

$$\min_p \quad \tfrac{1}{2} p^T G p + g_k^T p \tag{16.39a}$$

$$\text{subject to} \quad a_i^T p = 0, \quad i \in \mathcal{W}_k. \tag{16.39b}$$

$$\sum_{i \in \hat{\mathcal{W}}} a_i \hat{\lambda}_i = g = G\hat{x} + c, \tag{16.42}$$

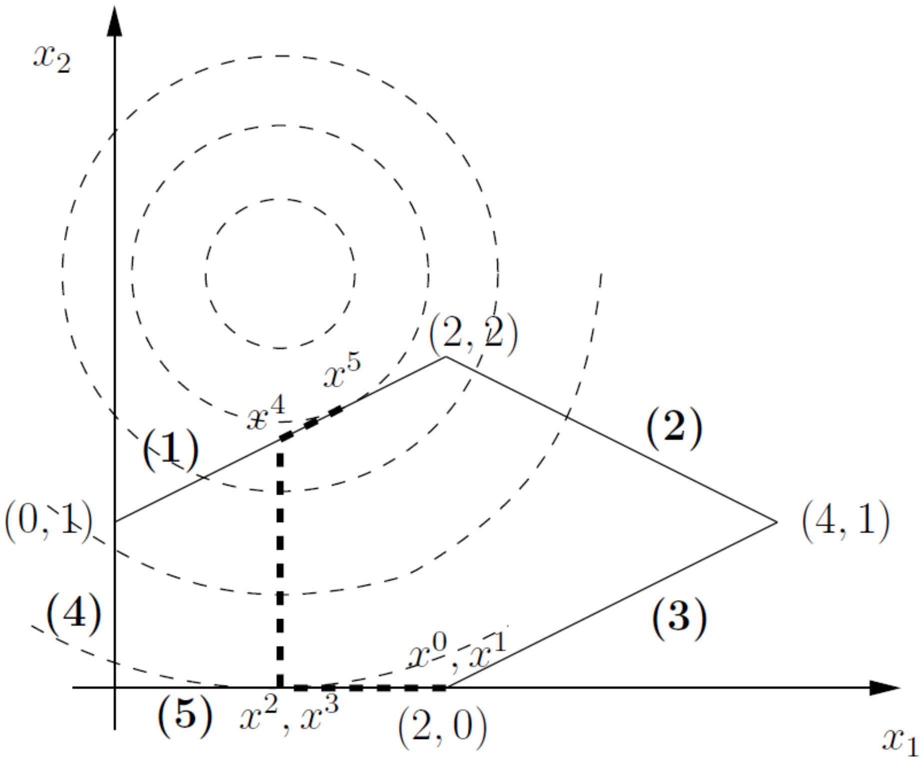
$$\alpha_k \stackrel{\text{def}}{=} \min \left(1, \min_{i \notin \mathcal{W}_k,\, a_i^T p_k < 0} \frac{b_i - a_i^T x_k}{a_i^T p_k}\right). \tag{16.41}$$

No degeneracy and *G>0*: Active set method converges in finite number of iterations.

Example 16.4

$$\min_x q(x) = (x_1-1)^2 + (x_2-2.5)^2$$

$$\begin{aligned} \text{subject to} \quad x_1 - 2x_2 + 2 &\geq 0 & (1)\\ -x_1 - 2x_2 + 6 &\geq 0 & (2)\\ -x_1 + 2x_2 + 2 &\geq 0 & (3)\\ x_1 &\geq 0 & (4)\\ x_2 &\geq 0 & (5) \end{aligned}$$



$$G = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}, \quad c = \begin{bmatrix} -2 \\ -5 \end{bmatrix}$$

$$a_1 = \begin{bmatrix} 1 & -2 \end{bmatrix}^\mathsf{T}, \quad b_1 = -2$$

$$a_2 = \begin{bmatrix} -1 & -2 \end{bmatrix}^\mathsf{T}, \quad b_2 = -6$$

$$a_3 = \begin{bmatrix} -1 & 2 \end{bmatrix}^\mathsf{T}, \quad b_3 = -2$$

$$a_4 = \begin{bmatrix} 1 & 0 \end{bmatrix}^\mathsf{T}, \quad b_4 = 0$$

$$a_5 = \begin{bmatrix} 0 & 1 \end{bmatrix}^\mathsf{T}, \quad b_5 = 0$$

Local SQP-algorithm for solving NLPs

Only equality constraints:

$$\min f(x) \quad \text{subject to } c(x) = 0$$

$$\min_p \; f_k + \nabla f_k^T p + \tfrac{1}{2} p^T \nabla_{xx}^2 \mathcal{L}_k p \qquad \text{subject to } A_k p + c_k = 0.$$

Algorithm 18.1 (Local SQP Algorithm for solving (18.1)).

Choose an initial pair (x_0, λ_0); set $k \leftarrow 0$;
repeat until a convergence test is satisfied
  Evaluate f_k, ∇f_k, $\nabla_{xx}^2 \mathcal{L}_k$, c_k, and A_k;
  Solve (18.7) to obtain p_k and l_k;
  Set $x_{k+1} \leftarrow x_k + p_k$ and $\lambda_{k+1} \leftarrow l_k$;
end (repeat)

With inequality constraints:

$$\min_{x \in \mathbb{R}^n} f(x) \quad \text{subject to} \quad \begin{cases} c_i(x) = 0, & i \in \mathcal{E} \\ c_i(x) \geq 0, & i \in \mathcal{I} \end{cases}$$

$$\begin{aligned} \min_p \quad & f_k + \nabla f_k^T p + \tfrac{1}{2} p^T \nabla_{xx}^2 \mathcal{L}_k p \\ \text{subject to} \quad & \nabla c_i(x_k)^T p + c_i(x_k) = 0, \quad i \in \mathcal{E}, \\ & \nabla c_i(x_k)^T p + c_i(x_k) \geq 0, \quad i \in \mathcal{I}. \end{aligned}$$

Thm 18.1: Alg. 18.1 identifies (eventually) the optimal active set of constraints (under assumptions). After, it behaves like Newton's method for equality constrained problems.

A practical line search SQP method

Algorithm 18.3 (Line Search SQP Algorithm).

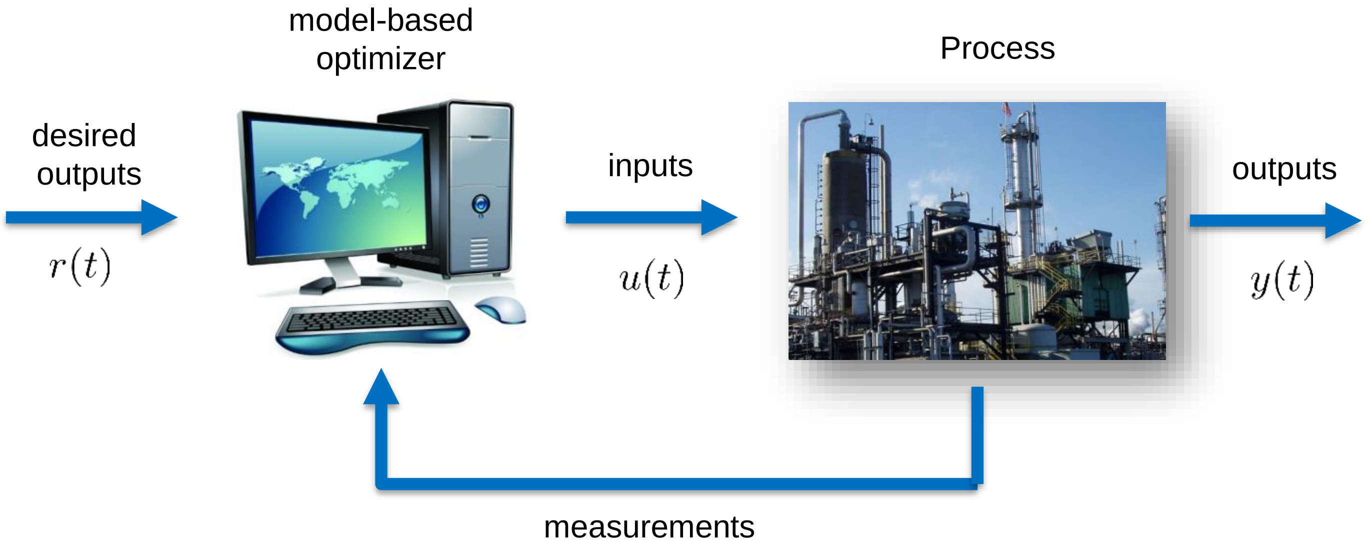
Choose parameters $\eta \in (0, 0.5)$, $\tau \in (0, 1)$, and an initial pair (x_0, λ_0);
Evaluate $f_0, \nabla f_0, c_0, A_0$;
If a quasi-Newton approximation is used, choose an initial $n \times n$ symmetric positive definite Hessian approximation B_0, otherwise compute $\nabla^2_{xx}\mathcal{L}_0$;
repeat until a convergence test is satisfied
  Compute p_k by solving (18.11); let $\hat{\lambda}$ be the corresponding multiplier;
  Set $p_\lambda \leftarrow \hat{\lambda} - \lambda_k$;
  Choose μ_k to satisfy (18.36) with $\sigma = 1$;
  Set $\alpha_k \leftarrow 1$;
  while $\phi_1(x_k + \alpha_k p_k; \mu_k) > \phi_1(x_k; \mu_k) + \eta\alpha_k D_1(\phi(x_k; \mu_k) p_k)$
    Reset $\alpha_k \leftarrow \tau_\alpha \alpha_k$ for some $\tau_\alpha \in (0, \tau]$;
  end (while)
  Set $x_{k+1} \leftarrow x_k + \alpha_k p_k$ and $\lambda_{k+1} \leftarrow \lambda_k + \alpha_k p_\lambda$;
  Evaluate $f_{k+1}, \nabla f_{k+1}, c_{k+1}, A_{k+1}$, (and possibly $\nabla^2_{xx}\mathcal{L}_{k+1}$);
  If a quasi-Newton approximation is used, set
    $s_k \leftarrow \alpha_k p_k$ and $y_k \leftarrow \nabla_x \mathcal{L}(x_{k+1}, \lambda_{k+1}) - \nabla_x \mathcal{L}(x_k, \lambda_{k+1})$,
  and obtain B_{k+1} by updating B_k using a quasi-Newton formula;
end (repeat)

$$
\begin{aligned}
\min_p \quad & f_k + \nabla f_k^T p + \tfrac{1}{2} p^T \nabla^2_{xx}\mathcal{L}_k p && \text{(18.11a)}\\
\text{subject to} \quad & \nabla c_i(x_k)^T p + c_i(x_k) = 0, \quad i \in \mathcal{E}, && \text{(18.11b)}\\
& \nabla c_i(x_k)^T p + c_i(x_k) \geq 0, \quad i \in \mathcal{I}. && \text{(18.11c)}
\end{aligned}
$$

Compare Alg. 6.1:

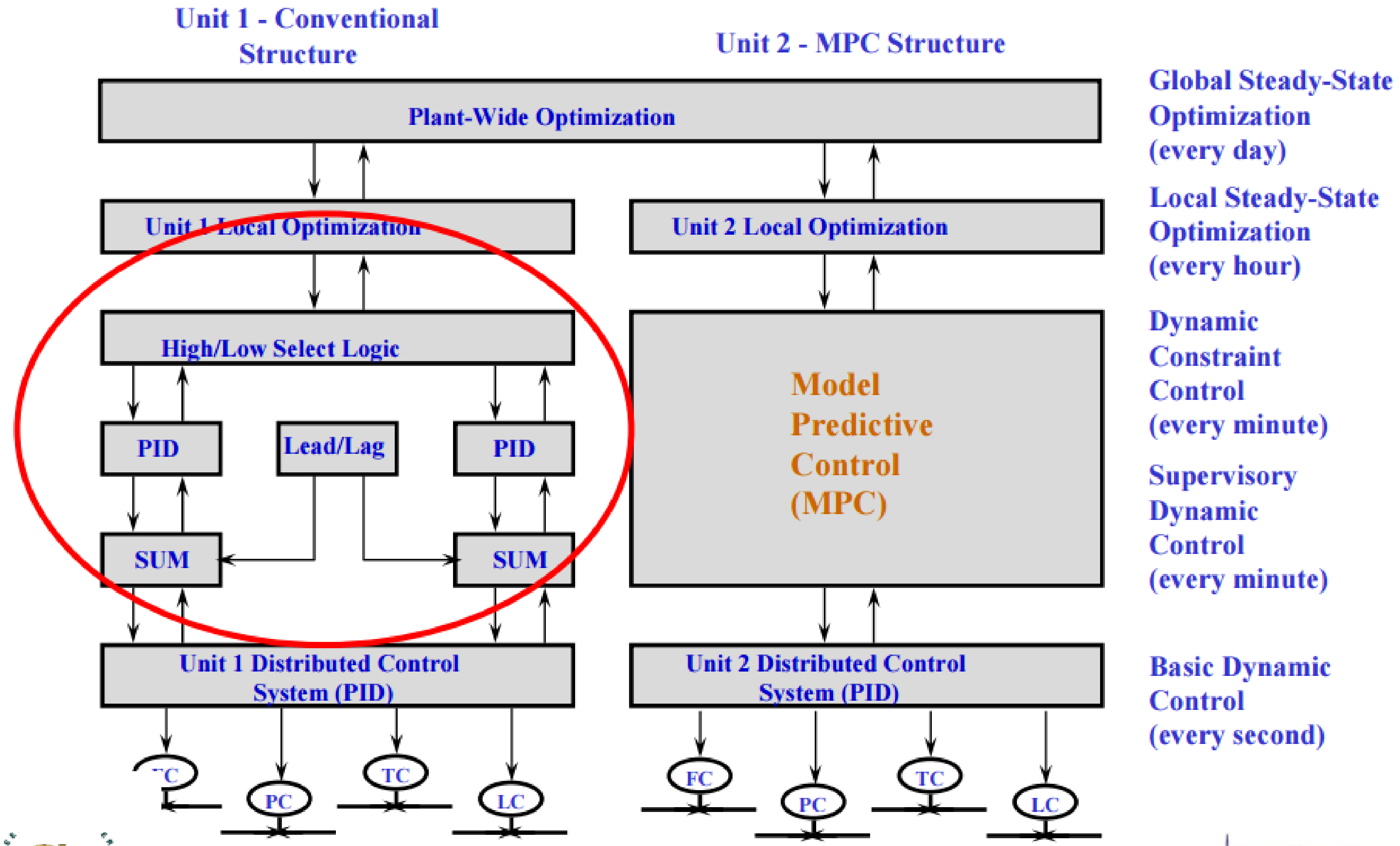
- Search direction from quadratic approximation
- Line search
- Update Hessian using BFGS

Model predictive control



A model of the process is used to compute the control signals (inputs) that optimize predicted future process behavior

Process control hierarchy before and after MPC



Embedded Model Predictive Control

PhD project Giorgio Kufoalor

Traditional MPC

- Successful in process industries
- Sampling times of minutes
- Powerful computing platforms

(M. Morari, 2013)

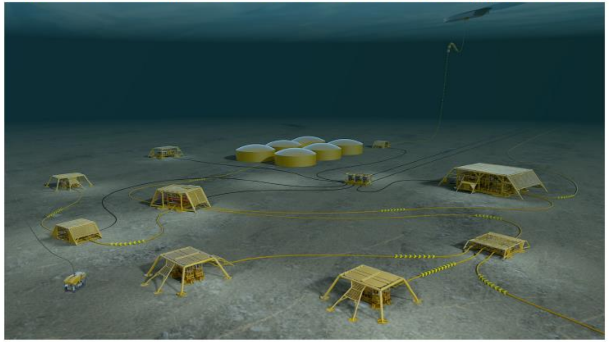
MIND THE GAP

Embedded MPC

- Small, high performance plants
- Sampling times of ms to ns
- Limited embedded platform

(M. Morari, 2013)

Embedded MPC for new industrial applications



(Statoil subsea factory)

Main contributions to fill the gap

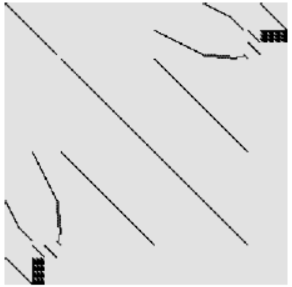
PhD project Giorgio Kufoalor

- Step-response MPC on ultra-reliable, resource constrained, industrial hardware

- Detailed study on *MPC formulations* and *solver methods* to achieve fast and reliable solutions
  - *Achieve significant savings*, both in computations and memory usage
  - Exploiting problem structure and specifics of computing platform
  - Automatic code generation (almost…)

- Development of new multistage QP framework, tailored to step-response MPC

- All extensively tested on a realistic subsea compact separation simulator using *hardware-in-the-loop* testing



D. K. Kufoalor, G. Frison, L. Imsland, T. A. Johansen, J. B. Jørgensen, **Block Factorization of Step Response Model Predictive Control Problems**, J. Process Control, Vol. 53, May, pp. 1–14, 2017;
D. K. M. Kufoalor, S. Richter, L. Imsland, T. A. Johansen, **Enabling Full Structure Exploitation of Practical MPC Formulations for Speeding up First-Order Methods**, 56th IEEE Conference on Decision and Control, 2017
D. K. M. Kufoalor, L. S. Imsland, T. A. Johansen, **Efficient Implementation of Step Response Models for Embedded Model Predictive Control**, Computers & Chemical Engineering, Volume 90, July, Pages 121–135, 2016
D. K. M. Kufoalor, V. Aaker, T. A. Johansen, L. Imsland, G. O. Eikrem, **Automatically Generated Embedded Model Predictive Control: Moving an Industrial PC-based MPC to an Embedded Platform**, J. Optimal Control - Applications and Methods, vol. 36, pp. 705–727, 2015

Open-loop optimization with linear state-space model

QP

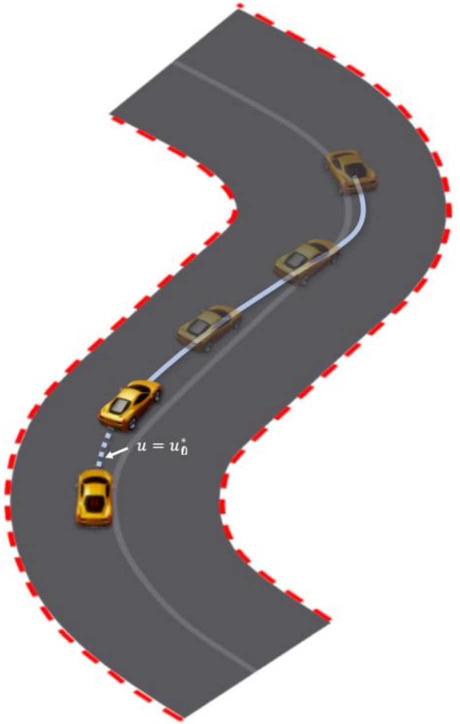
$$\min_{z\in\mathbb{R}^n} f(z) = \sum_{t=0}^{N-1} \frac{1}{2} x_{t+1}^\top Q_{t+1} x_{t+1} + d_{x,t+1} x_{t+1} + \frac{1}{2} u_t^\top R_t u_t + d_{u,t} u_t + \frac{1}{2} \Delta u_t^\top S \Delta u_t$$

subject to

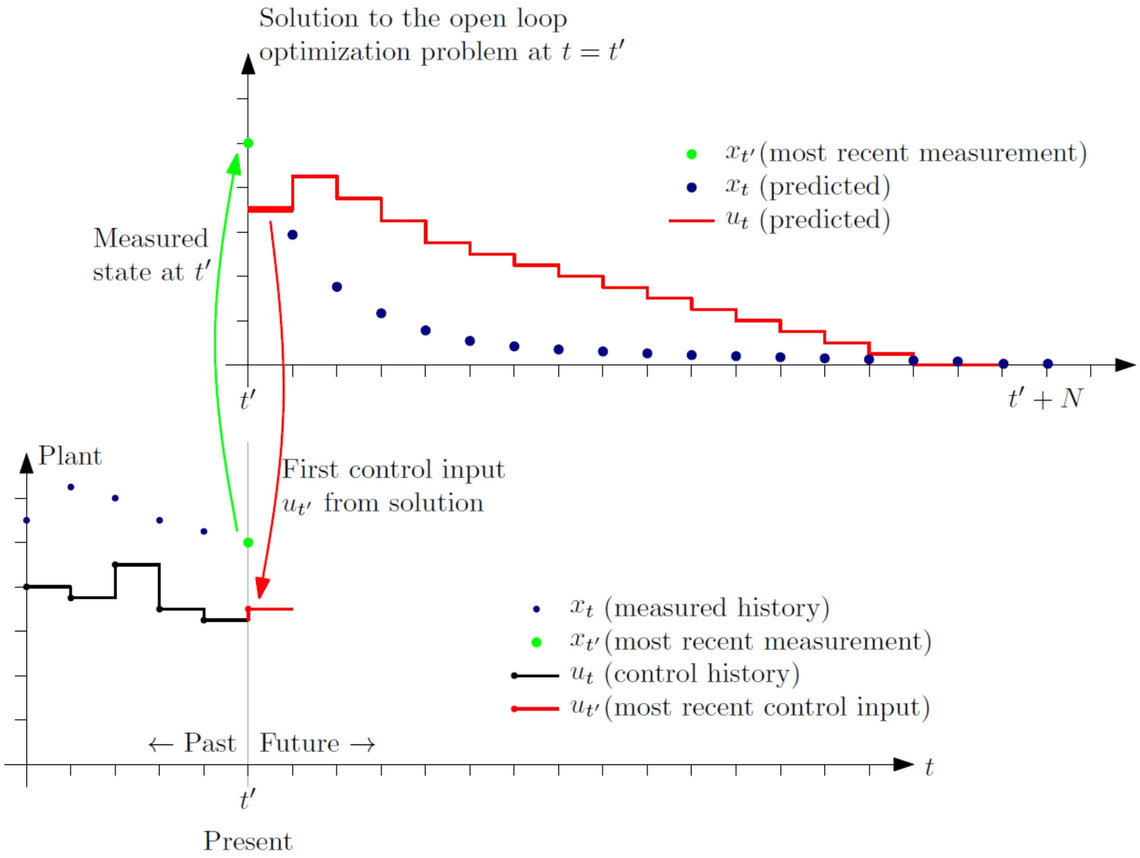
$$\begin{aligned} x_{t+1} &= A_t x_t + B_t u_t, \quad t = \{0,\dots,N-1\} \\ x^{\text{low}} &\leq x_t \leq x^{\text{high}}, \quad t = \{1,\dots,N\} \\ u^{\text{low}} &\leq u_t \leq u^{\text{high}}, \quad t = \{0,\dots,N-1\} \\ -\Delta u^{\text{high}} &\leq \Delta u_t \leq \Delta u^{\text{high}}, \quad t = \{0,\dots,N-1\} \end{aligned}$$

where

$$\begin{aligned} & x_0 \text{ and } u_{-1} \text{ is given} \\ & \Delta u_t := u_t - u_{t-1} \\ & z^\top := (u_0^\top, x_1^\top, \dots, u_{N-1}^\top, x_N^\top) \\ & n = N \cdot (n_x + n_u) \\ & Q_t \succeq 0 \quad t = \{1,\dots,N\} \\ & R_t \succ 0 \quad t = \{0,\dots,N-1\} \end{aligned}$$



Model predictive control principle



Nonlinear MPC

The three ways of implementing NMPC

- Sequential (single shooting) methods
  - Only inputs as optimization variables, simulate to calculate objective and state constraints (and gradients)
  - "Small" optimization problem with no structure
  - Standard SQP methods are suitable
- Simultaneous methods
  - Both inputs and states as optimization variables, include model as equality constraints
  - Huge optimization problem, but constraints and gradients are very structured ("sparse", a lot of zeros)
  - Must use solvers that exploit this structure (e.g. IPOPT)
- In-between method: Multiple shooting
  - Divide horizon into "sub-horizons", use single-shooting on each sub-horizon and add equality constraints to "glue" each sub-horizon together
  - Results in medium-sized "block-structured" optimization problem
  - Ideally use solvers that exploit this structure (but not many exists)

- What is best? Depends…

NMPC example: van der Pol

- Controlled van der Pol oscillator

$$\dot{x}_1 = x_2$$
$$\dot{x}_2 = -x_1 + e(1 - x_1^2)x_2 + u$$

- Discretization (here: Euler)

- Stability dependent on horizon length
- Importance of “warm-start”

Implementation: Tor Aksel Heirung

Output feedback MPC

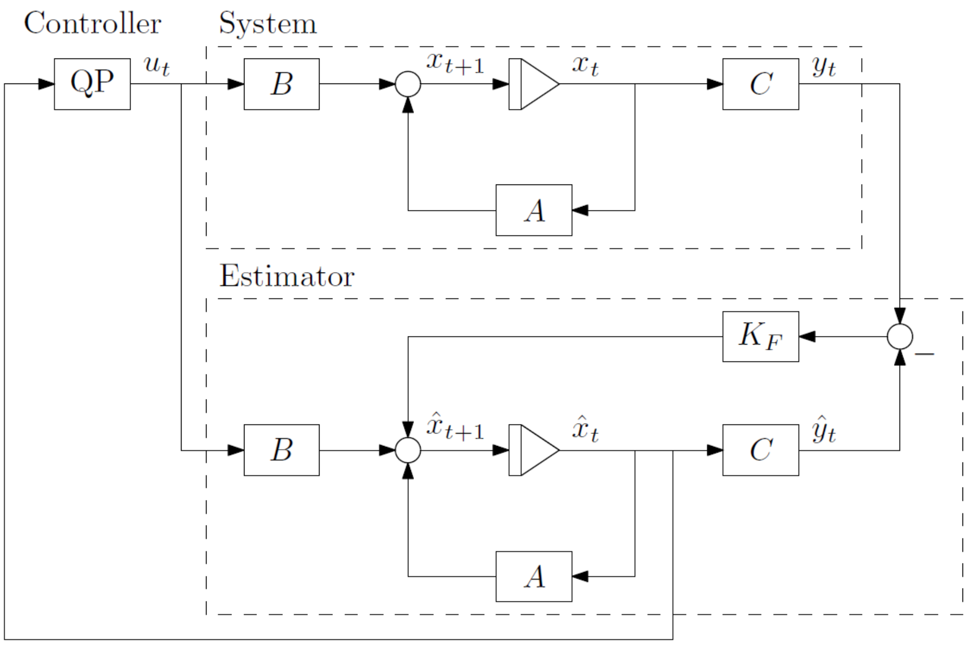
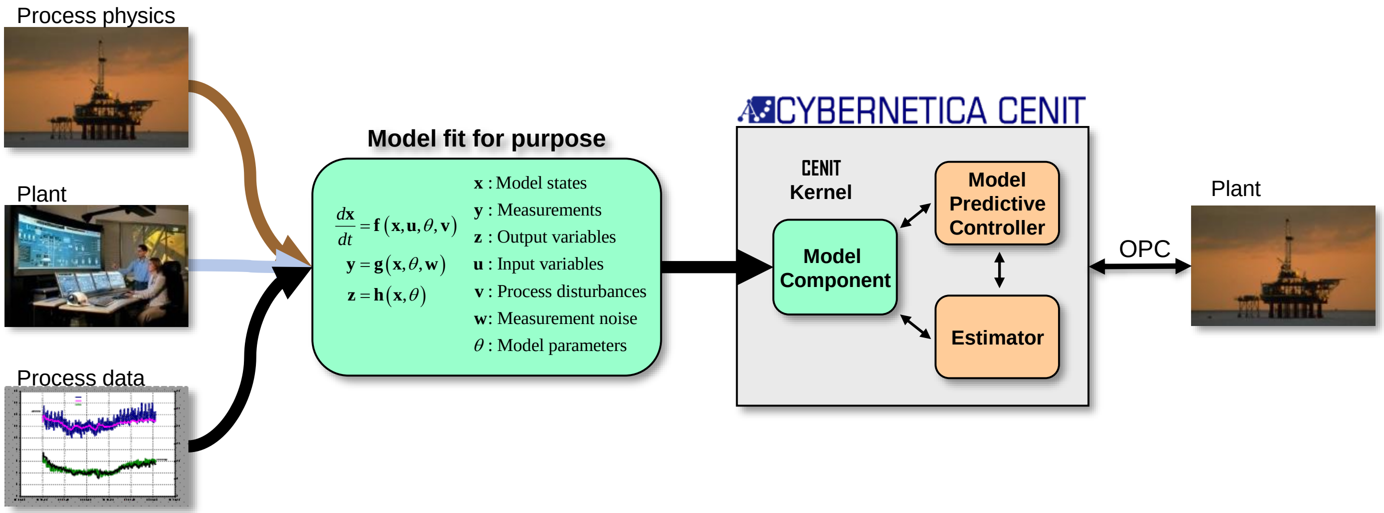


Figure 4.3: The structure of an output feedback linear MPC.

Output feedback NMPC

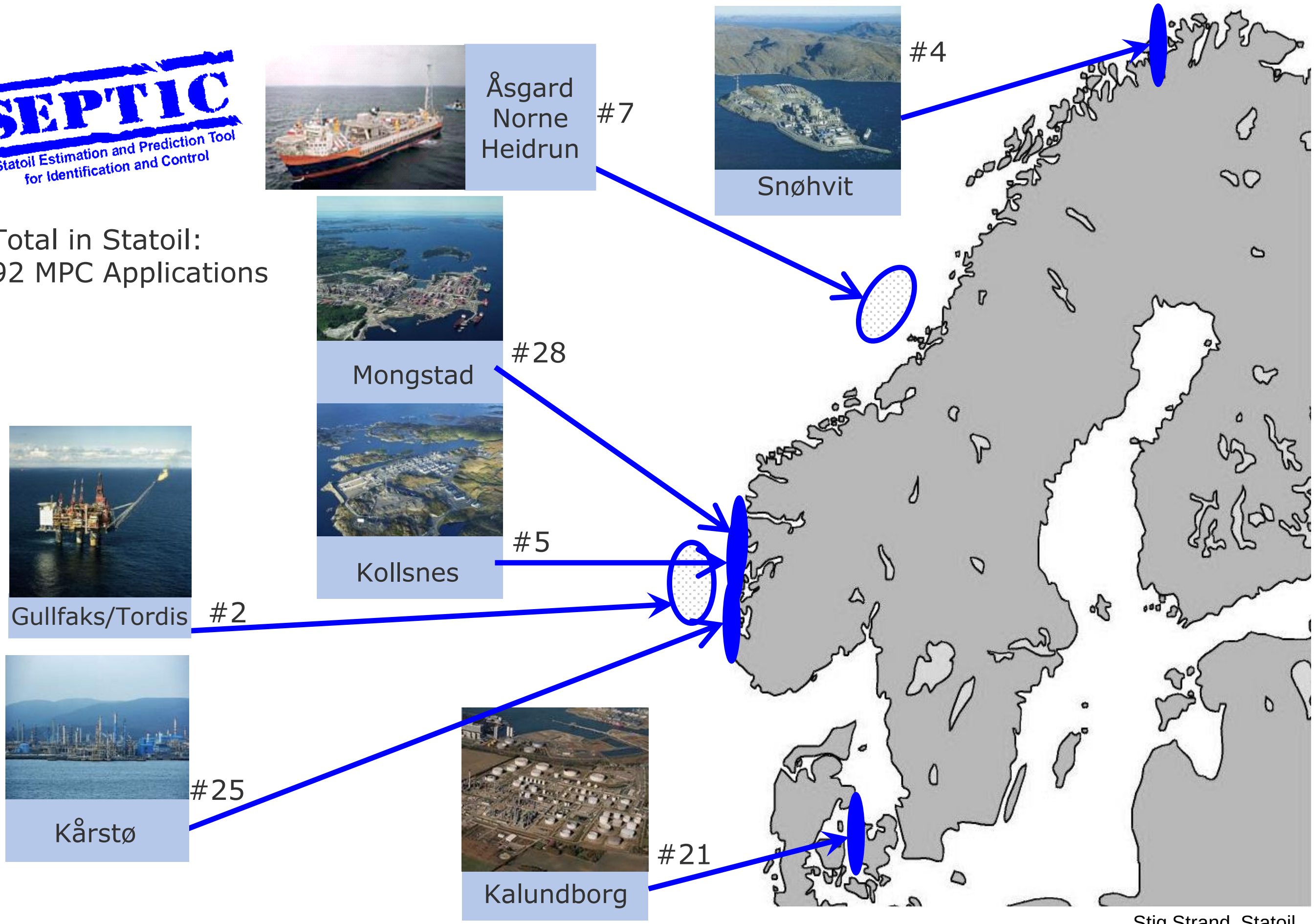
Cybernetica

- [Cybernetica](#) provides advanced model-based control systems for the process industry
  - Based on non-linear first principles (mechanistic) models
  - Nonlinear state- and parameter estimation (EKF, MHE)
  - Online dynamic optimization (nonlinear model predictive control, NMPC)



SEPTIC
Statoil Estimation and Prediction Tool for Identification and Control

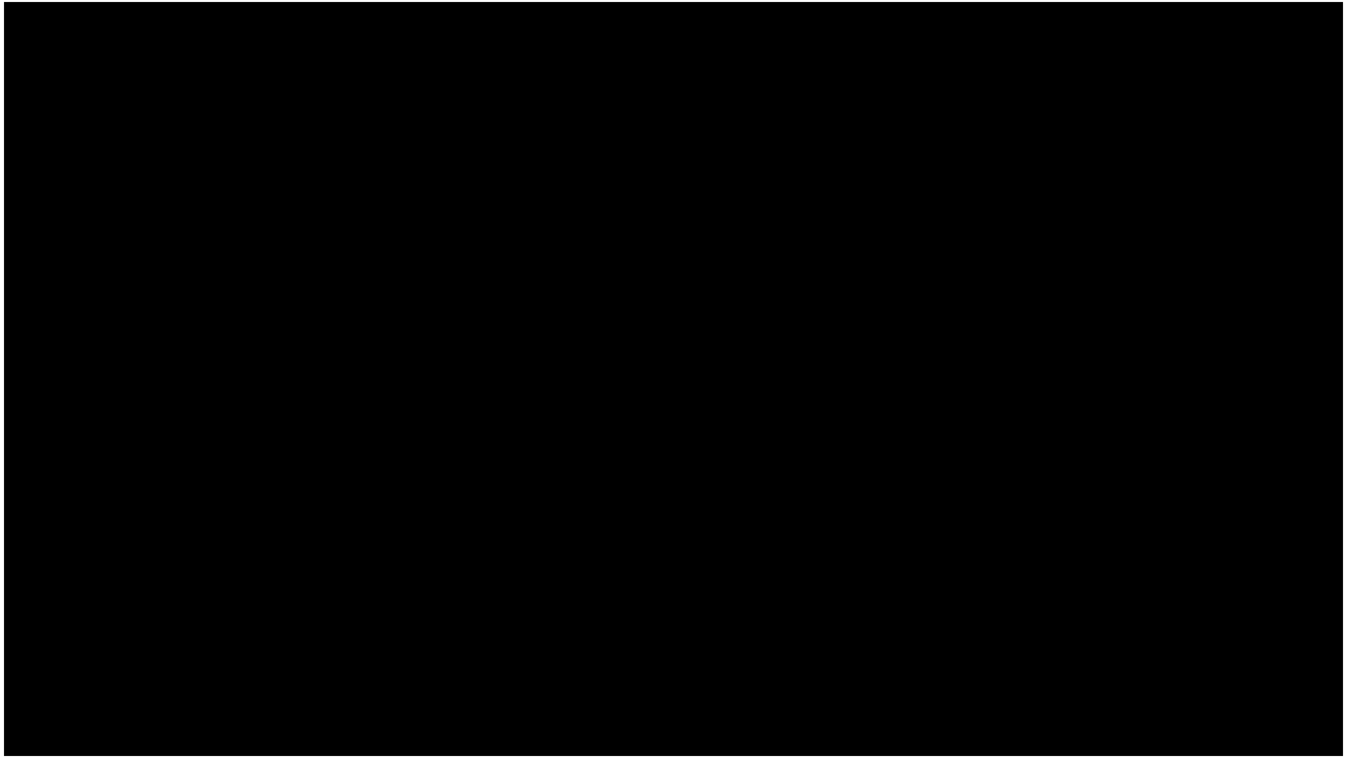
Total in Statoil:
92 MPC Applications



Stig Strand, Statoil

SOME EXAMPLES

Applied LQR



Source: https://doi.org/10.1109/HUMANOIDS.2014.7041387

Applied LQR

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + g(q) = S^T\tau + J_c(q)^T f_c$$

1. Modify dynamics to null-space of the contact constraint
2. Linearize the model around desired pose $(q_0, \dot{q}_0, \tau_0)$

$$\dot{x} = Ax + Bu, \qquad x^T = [\Delta q^T, \Delta\dot{q}^T], \qquad u^T = [\Delta\tau]$$

3. Calculate the infinite horizon linear quadratic regulator

$$J = \int_0^\infty x^T Q x + u^T R u$$

4. Apply the input

$$\tau = \tau_0 - Kx$$

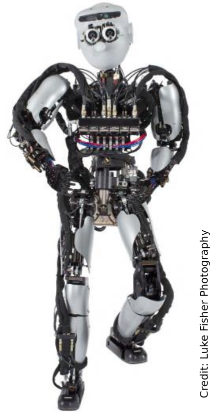
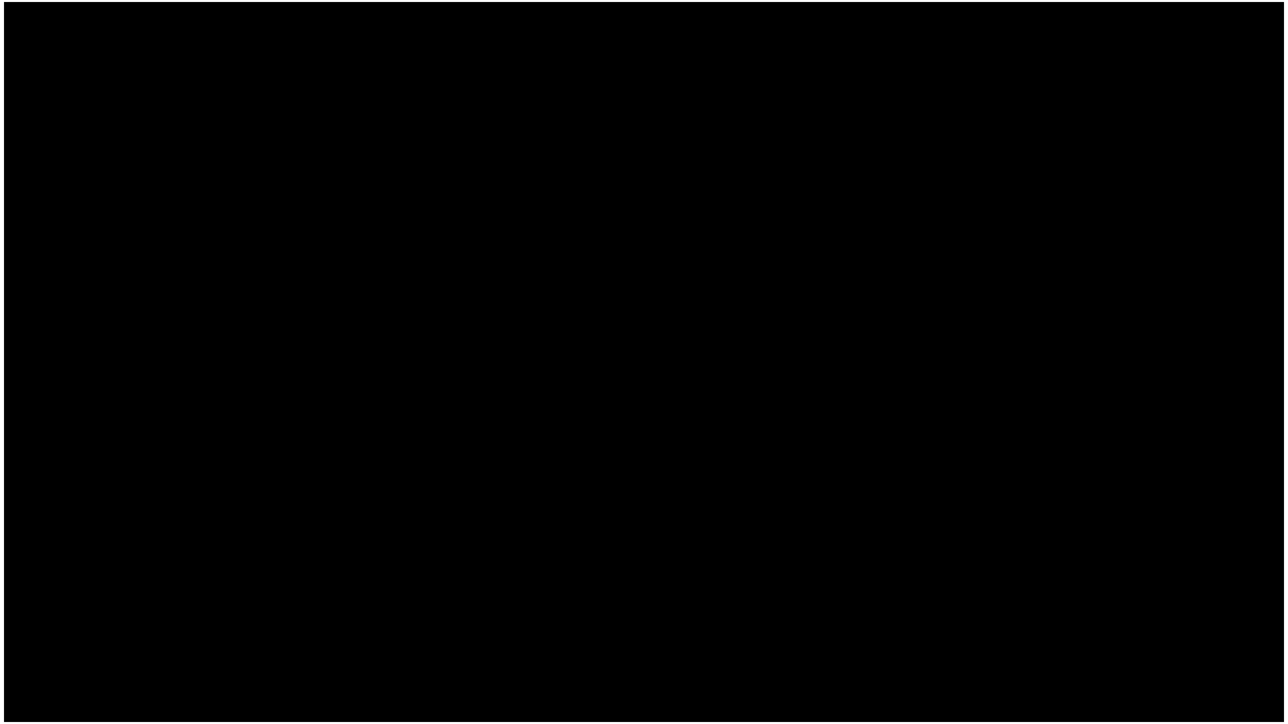


Fig. 1: Hydraulically actuated torque controlled Sarcos humanoid used for experiments.

Source: https://doi.org/10.1109/HUMANOIDS.2014.7041387

Applied Open-Loop Dynamic Optimization



Source: https://arxiv.org/pdf/2003.04882.pdf

Applied Open-Loop Dynamic Optimization

1. Split the problem:
   - Lap time offline
   - Tracking online
2. Solve a "periodic" problem

$$\begin{aligned}\min_{\mathbf{X},\mathbf{U}} \quad & \sum_{k=0}^{N} j_{\mathrm{LTO}}(\mathbf{x}_k, \mathbf{u}_k) \\ \text{s.t.} \quad & \mathbf{x}_{k+1} = f_s^d(\mathbf{x}_k, \mathbf{u}_k) \\ & f_s^d(\mathbf{x}_N, \mathbf{u}_N) = \mathbf{x}_0\end{aligned}$$

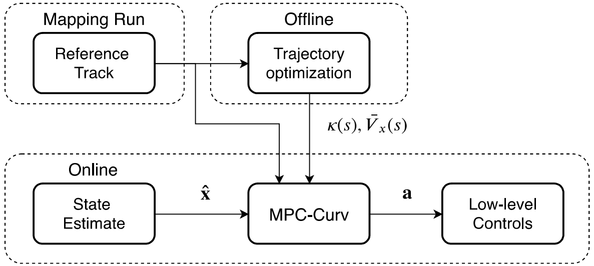
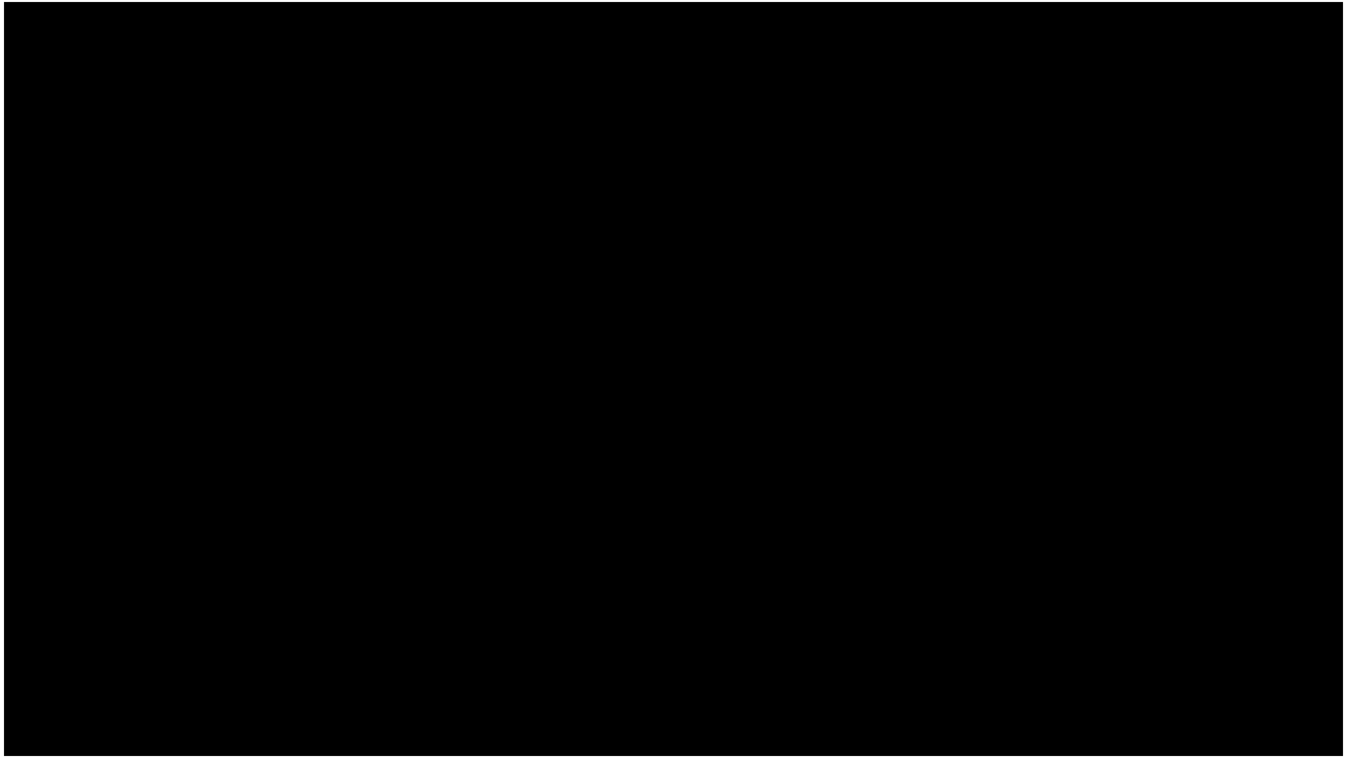


Fig. 3. The hierarchical controller uses the reference track in both stages of the hierarchical controller. First the lap time optimization (LTO) problem computes a reference path, whose curvature $\kappa(s)$ and speed profile $\bar{V}_x(s)$ are later used by the NMPC.

3. Apply NMPC for closed-loop

Source: https://arxiv.org/pdf/2003.04882.pdf

MPC using distributed SQP



Source: http://doi.org/10.1109/TCST.2018.2871397

MPC using distributed SQP

Algorithm 1 Schematic description of the Distributed SQP. The arguments of the functions are removed for brevity.

Coordinator initializes the problem.
while exit conditions not fulfilled **do**
    Coordinator broadcasts $\boldsymbol{T}$.
    Each vehicle solves (8)
    Each vehicle returns $\nabla V_i, \nabla^2 V_i, g_i, \nabla g_i, \nabla^2 g_i$.
    Coordinator solves the SQP sub-problem (21).
    Coordinator and vehicles compute α.
    Coordinator takes step (20).

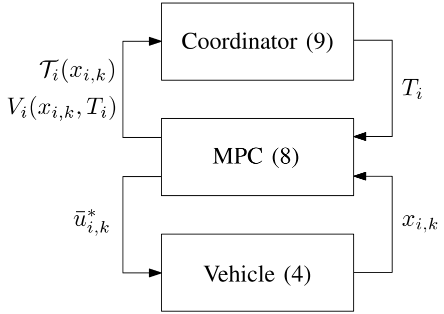
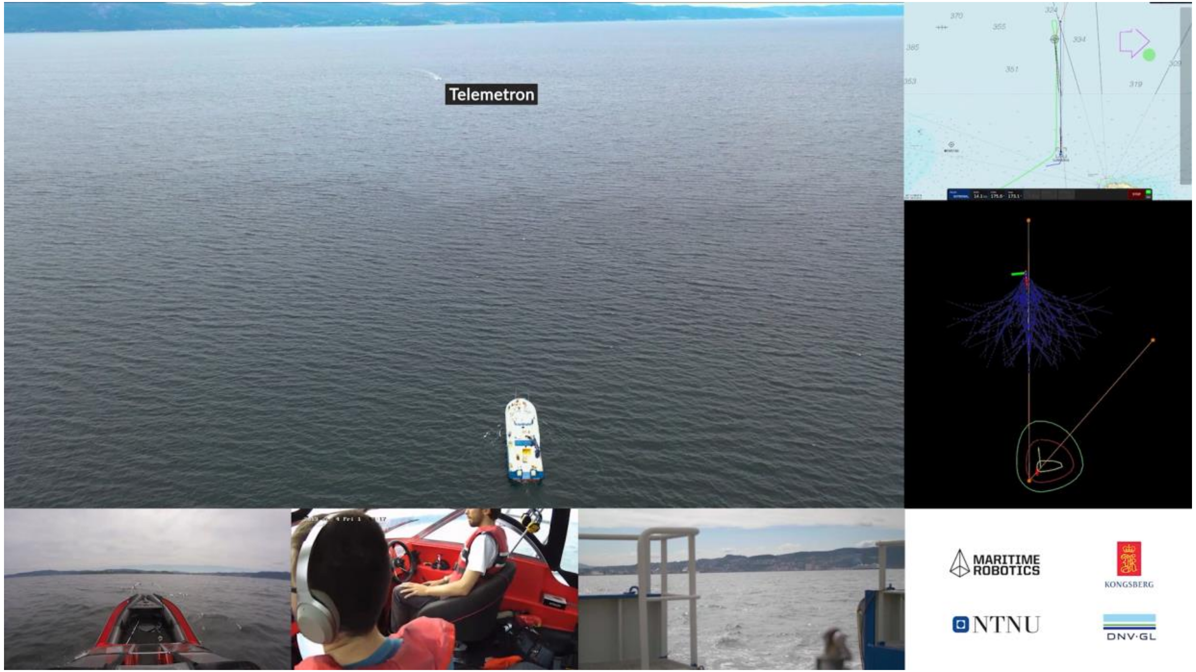


Fig. 2: Schematic illustration of the bi-level control structure for one vehicle. The coordinator is in closed-loop with all vehicles in the same way.

Branching Course MPC



Source: Bjørn-Olav Holtung Eriksen
https://doi.org/10.1002/rob.21900

Branching Course MPC

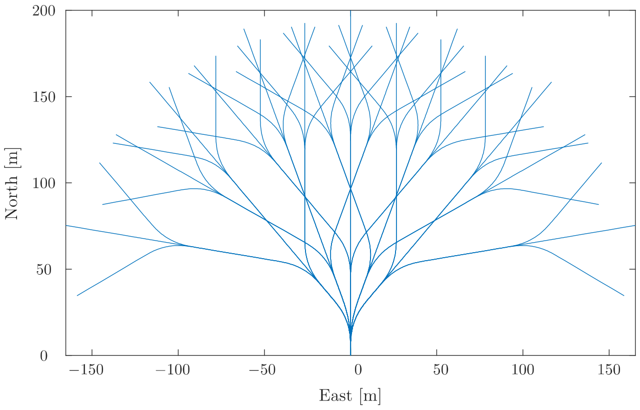


Figure 10: A set of predicted pose trajectories with 3 levels.