Lecture 20: NMPC and summing up

Course in a nutshell:

- Unconstrained optimization
  – Steepest descent, Newton, Quasi-Newton
  – Globalization (line-search and Hessian modification), derivatives
- Constrained optimization
  – Optimality conditions, KKT
  – Linear programming: SIMPLEX
  – Quadratic programming: Active set method
  – Nonlinear programming: SQP
- Control and optimization
  – LQ control
  – MPC
  – Today: Nonlinear MPC, some “practical”/industrial issues on MPC

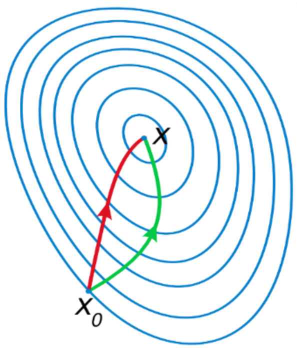
Reference: F&H 4.5, 4.6

Q&A session: Thursday 30 May? (Exam: Monday 03 June)

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)

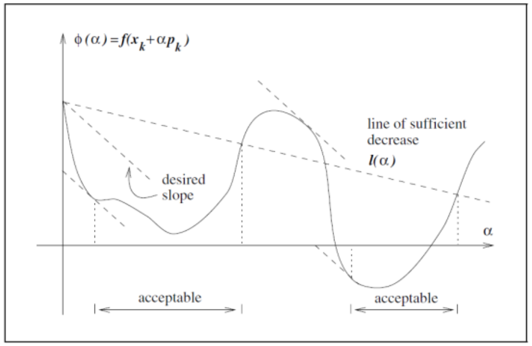
Termination criteria:
Stop when first of these become true:

- $\|\nabla f(x_k)\| \le \epsilon$ (necessary condition)
- $\|x_k - x_{k-1}\| \le \epsilon$ (no progress)
- $\|f(x_k) - f(x_{k-1})\| \le \epsilon$ (no progress)
- $k \le k_{\max}$ (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
  $$p_k = -B_k^{-1} \nabla f(x_k)$$
  $$B_k \approx \nabla^2 f(x_k)$$

Step length (Wolfe):



Quasi-Newton: BFGS method

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 $y_k = \nabla f_{k+1} - \nabla f_k$;

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

$k \leftarrow k + 1$;

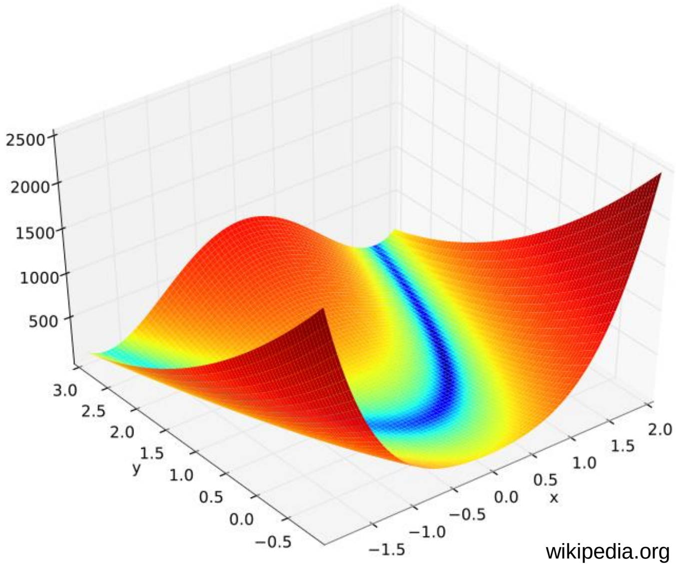
end (while)

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

Example (from book)

- Using steepest descent, BFGS (Quasi-Newton) 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

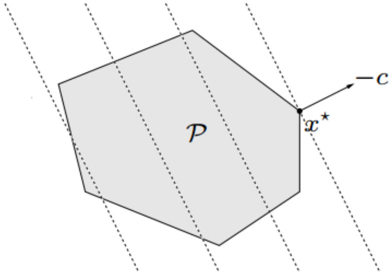
- Last iterations; value of $\|x_k - x^*\|$

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

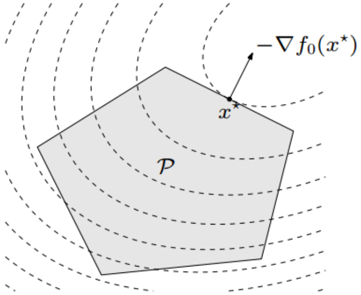
- Linear programming
  - Convex problem
  - Feasible set polyhedron

$$\begin{array}{ll} \text{minimize} & c^\mathsf{T} x \\ \text{subject to} & Ax \le b \\ & Cx = d \end{array}$$



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

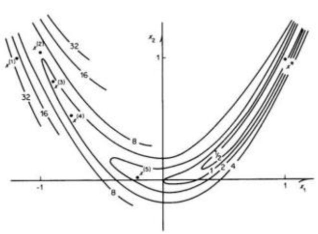
$$\begin{array}{ll} \text{minimize} & \frac{1}{2} x^\mathsf{T} P x + q^\mathsf{T} x \\ \text{subject to} & Ax \le b \\ & Cx = d \end{array}$$



- Nonlinear programming
  - In general non-convex!

$$\begin{array}{ll} \text{minimize} & f(x) \\ \text{subject to} & g(x) = 0 \\ & h(x) \ge 0 \end{array}$$

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



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

KKT conditions (Theorem 12.1)

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

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

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

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

Linear programming, standard form and KKT: recap

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):

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

Check KKT-conditions for BFP

- Given BFP x, and corresponding basis $\mathcal{B}(x)$. Define
$$\mathcal{N}(x) = \{1, 2, \dots, n\} \backslash \mathcal{B}(x)$$
- Partition x, s and c:
$$x_B = [x_i]_{i \in \mathcal{B}(x)} \quad x_N = [x_i]_{i \in \mathcal{N}(x)}$$
- KKT conditions

  KKT-2: $Ax = Bx_B + Nx_N = Bx_B = b$ (since *x* is BFP)

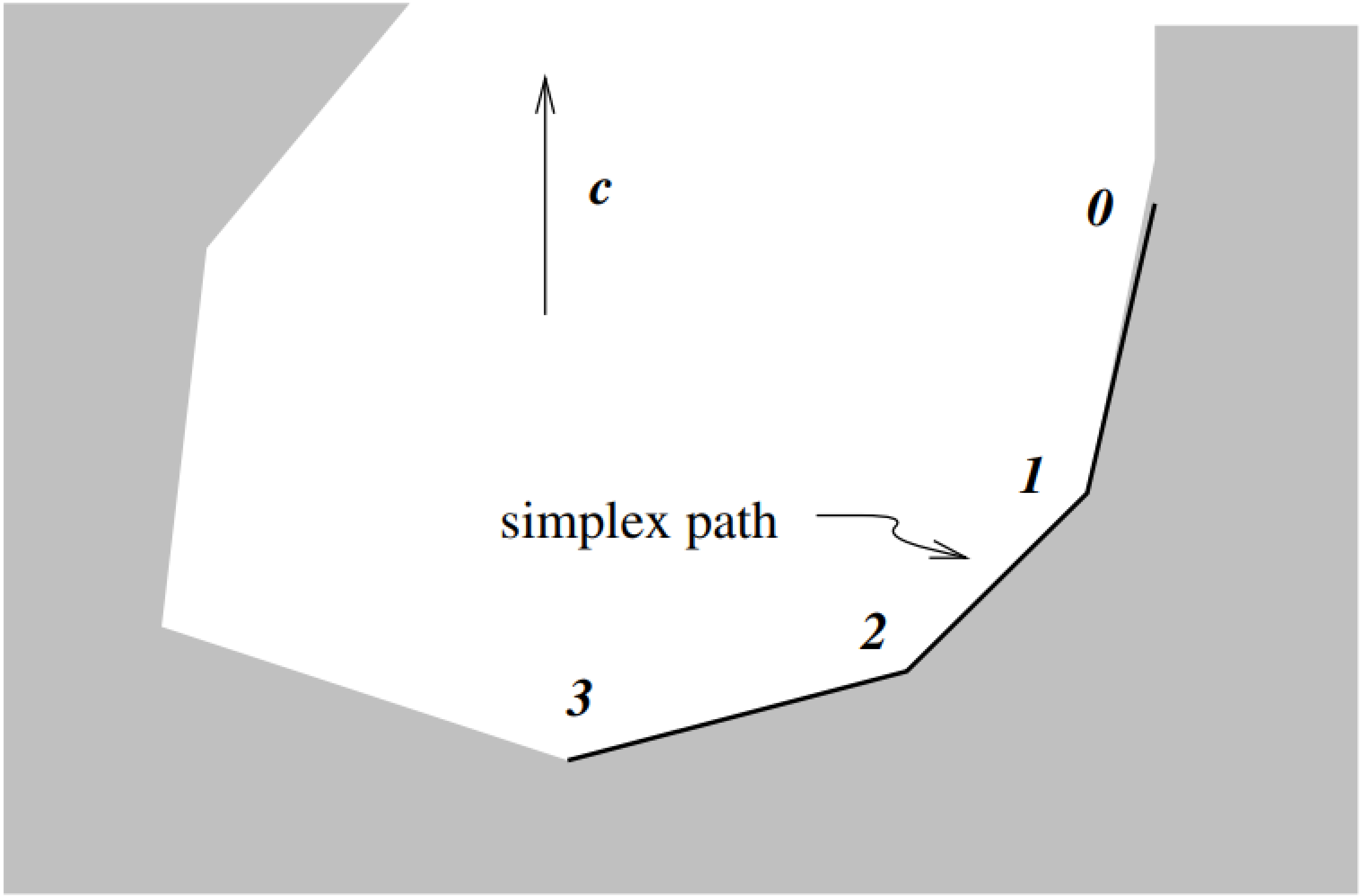
  KKT-3: $x_B = B^{-1}b \geq 0, \quad x_N = 0$ (since *x* is BFP)

  KKT-5: $x^\top s = x_B^\top s_B + x_N^\top s_N = 0$ if we choose $s_B = 0$

  KKT-1: $\begin{bmatrix} B^T \\ N^T \end{bmatrix} \lambda + \begin{bmatrix} s_B \\ s_N \end{bmatrix} = \begin{bmatrix} c_B \\ c_N \end{bmatrix} \Rightarrow \begin{cases} \lambda = B^{-T} c_B \\ s_N = c_N - N^T \lambda \end{cases}$

  KKT-4: Is $s_N \geq 0$?

- If $s_N \geq 0$, then the BFP x fulfills KKT and is a solution
- If not, change basis, and try again
  - E.g. pick smallest element of s_N (index *q*), increase x_q along *Ax*=*b* until x_p becomes zero. Move *q* from $\mathcal{N}$ to $\mathcal{B}$, and *p* from $\mathcal{B}$ to $\mathcal{N}$. This guarantees decrease of objective, and no “cycling” (if non-degenerate).


c
0
1
simplex path
2
3

General QP problem

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

- 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:

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

Defined via active set:

$$\mathcal{A}(x^*) = \mathcal{E} \cup \left\{ i \in \mathcal{I} \middle| a_i^\top x^* = b_i \right\}$$

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

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
      $j \leftarrow \arg\min_{j \in \mathcal{W}_k \cap \mathcal{I}} \hat{\lambda}_j$;
      $x_{k+1} \leftarrow x_k$; $\mathcal{W}_{k+1} \leftarrow \mathcal{W}_k \setminus \{j\}$;
  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)

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

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

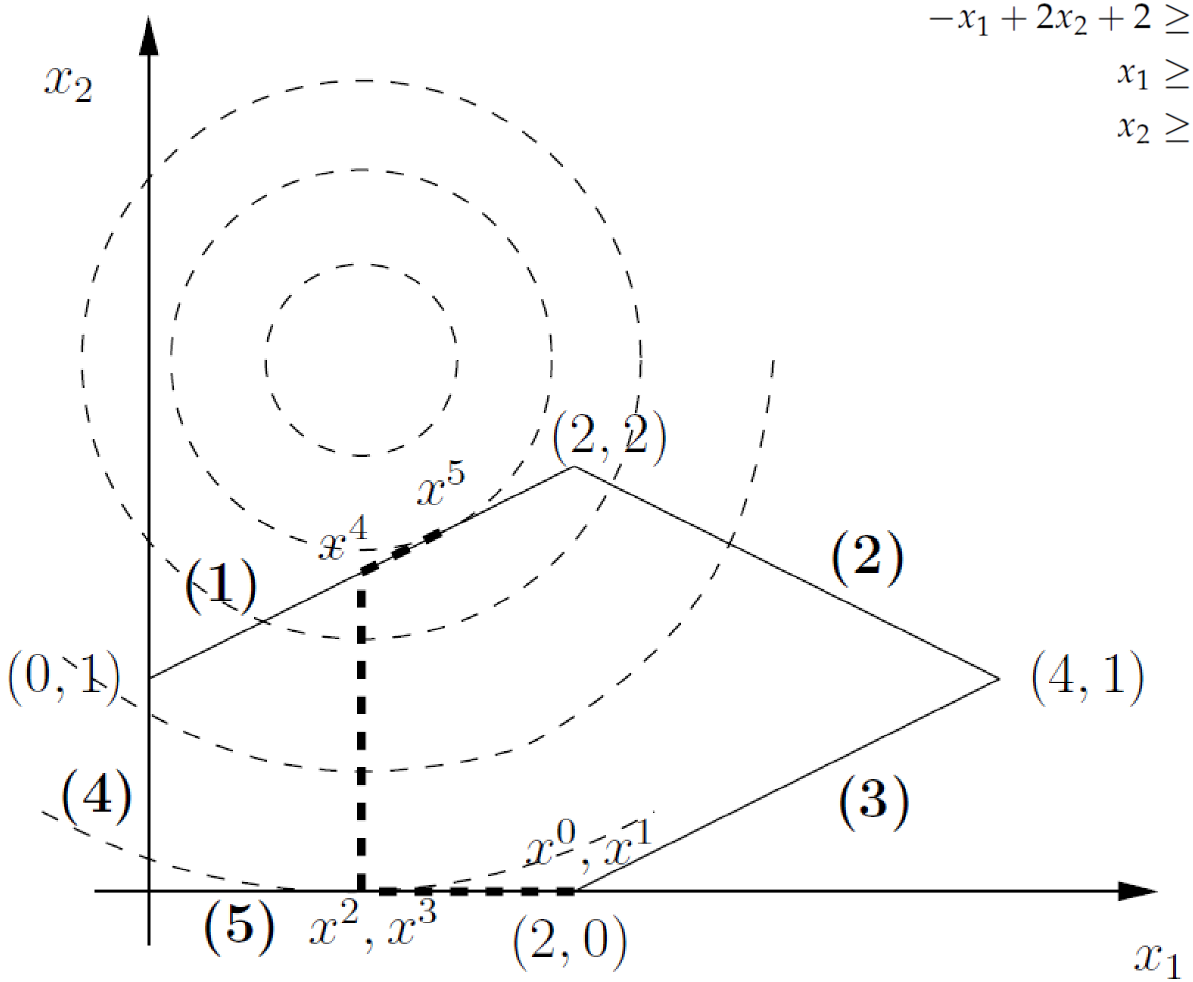
$$\sum_{i \in \hat{\mathcal{W}}} a_i \hat{\lambda}_i = g = G\hat{x} + c, \tag{16.42}$$

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

Example 16.4

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

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



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

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

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

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

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

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

Equality-constrained NLP

$$\min_{x \in \mathbb{R}^n} f(x) \quad \text{subject to} \quad c(x) = 0$$

- Lagrangian: $\mathcal{L}(x, \lambda) = f(x) - \lambda^\top c(x)$
- KKT-system: $F(x, \lambda) = \begin{pmatrix} \nabla_x \mathcal{L}(x, \lambda) \\ c(x) \end{pmatrix} = 0$

$$A(x)^\top = (\nabla c_1(x), \ldots, \nabla c_m(x))$$

- To solve: Use Newton's method for nonlinear equations on KKT-system:

$$\begin{pmatrix} x_{k+1} \\ \lambda_{k+1} \end{pmatrix} = \begin{pmatrix} x_k \\ \lambda_k \end{pmatrix} + \begin{pmatrix} p_k \\ p_{\lambda_k} \end{pmatrix} \quad \text{where} \quad \underbrace{\begin{pmatrix} \nabla^2_{xx} \mathcal{L}(x_k, \lambda_k) & -A^\top(x_k) \\ A(x_k) & 0 \end{pmatrix}}_{\text{Jacobian of } F(x,\lambda) \text{ at } (x_k,\, \lambda_k)} \begin{pmatrix} p_k \\ p_{\lambda_k} \end{pmatrix} = \underbrace{\begin{pmatrix} -\nabla f(x_k) + A^\top(x_k)\lambda_k \\ -c(x_k) \end{pmatrix}}_{-F(x_k,\, \lambda_k)}$$

- Consider this quadratic approximation to the objective (or Lagrangian):

$$\min_{p \in \mathbb{R}^n} f(x_k) + \nabla f(x_k)^\top p + \frac{1}{2} p^\top \nabla^2_{xx} \mathcal{L}(x_k, \lambda_k) p \ \text{ subject to } \ c(x_k) + A(x_k)^\top p = 0$$

- KKT: $\begin{pmatrix} \nabla^2_{xx} \mathcal{L}(x_k, \lambda_k) & -A^\top(x_k) \\ A(x_k) & 0 \end{pmatrix} \begin{pmatrix} p_k \\ l_k \end{pmatrix} = \begin{pmatrix} -\nabla f(x_k) \\ -c(x_k) \end{pmatrix}$

- If we let $l_k = p_{\lambda_k} + \lambda_k = \lambda_{k+1}$, it is clear that the two KKT systems give equivalent solutions
  - Newton-viewpoint: quadratic convergence locally
  - QP-viewpoint: provides a means for practical implementation and extension to inequality constraints
- Assumptions for the above: 1) $A(x_k)$ full row rank (LICQ),
  2) $\nabla^2_{xx} \mathcal{L}(x_k, \lambda_k) > 0$ on tangent space of constraints

Local SQP-algorithm for solving NLPs

Only equality constraints:

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

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, $\nabla^2_{xx}\mathcal{L}_k$, 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)

$$\begin{aligned} \min_p \quad & f_k + \nabla f_k^T p + \tfrac{1}{2} p^T \nabla^2_{xx}\mathcal{L}_k p \\ \text{subject to} \quad & A_k p + c_k = 0. \end{aligned}$$

With inequality constraints (IQP method):

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

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

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)

$$\min_p \quad f_k + \nabla f_k^T p + \tfrac{1}{2} p^T \nabla^2_{xx}\mathcal{L}_k p \tag{18.11a}$$

$$\text{subject to} \quad \nabla c_i(x_k)^T p + c_i(x_k) = 0, \quad i \in \mathcal{E}, \tag{18.11b}$$

$$\nabla c_i(x_k)^T p + c_i(x_k) \geq 0, \quad i \in \mathcal{I}. \tag{18.11c}$$

$$\mu \geq \frac{\nabla f_k^T p_k + (\sigma/2) p_k^T \nabla^2_{xx}\mathcal{L}_k p_k}{(1-\rho)\|c_k\|_1}. \tag{18.36}$$

NLP software

- SNOPT
  - “solves large-scale linear and nonlinear problems; especially recommended if some of the constraints are highly nonlinear, or constraints respectively their gradients are costly to evaluate and second derivative information is unavailable or hard to obtain; assumes that the number of “free” variables is modest.”
  - Licence: Commercial
- IPOPT
  - “interior point method for large-scale NLPs”
  - License: Open source (but good linear solvers might be commercial)
- WORHP
  - SQP solver for very large problems, IP at QP level, exact or approximate second derivatives, various linear algebra options, varius interfaces
  - Licence: Commercial, but free for academia
- KNITRO
  - trust region interior point method, efficient for NLPs of all sizes, various interfaces
  - License: Commercial
- (…and several others, including `fmincon` in Matlab Optimization Toolbox)

- «Decision tree for optimization software»: http://plato.asu.edu/sub/nlores.html

Example: optimization using CasADi

- CasADi (https://casadi.org/)
  - "CasADi is a symbolic framework for numeric optimization implementing automatic differentiation in forward and reverse modes on sparse matrix-valued computational graphs."
  - "…interfaces to IPOPT/BONMIN, BlockSQP, WORHP, KNITRO and SNOPT…"

$$\min_{x,y,z} \; x^2 + 100z^2$$
$$\text{s.t.} \; z + (1-x)^2 - y = 0$$

Define variables

Define objective and constraints

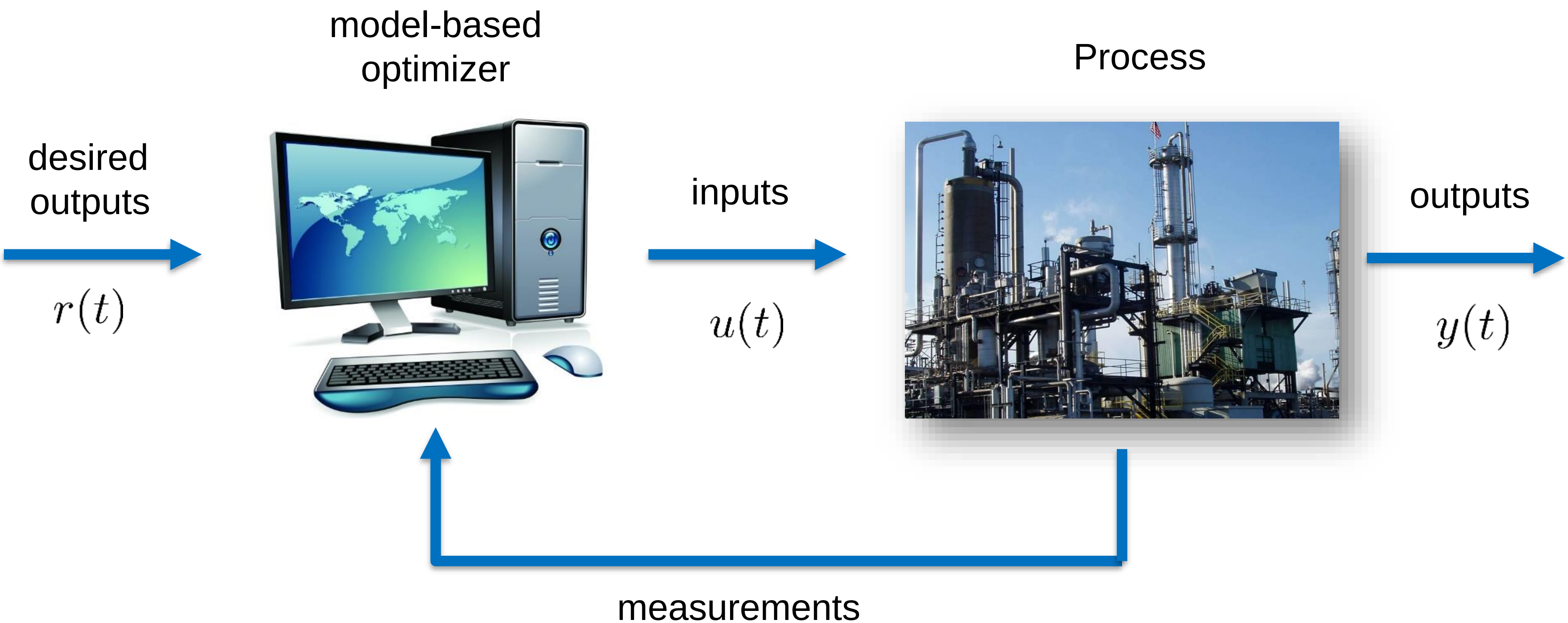
Create solver object

Solve the opt problem

rosenbrock.m

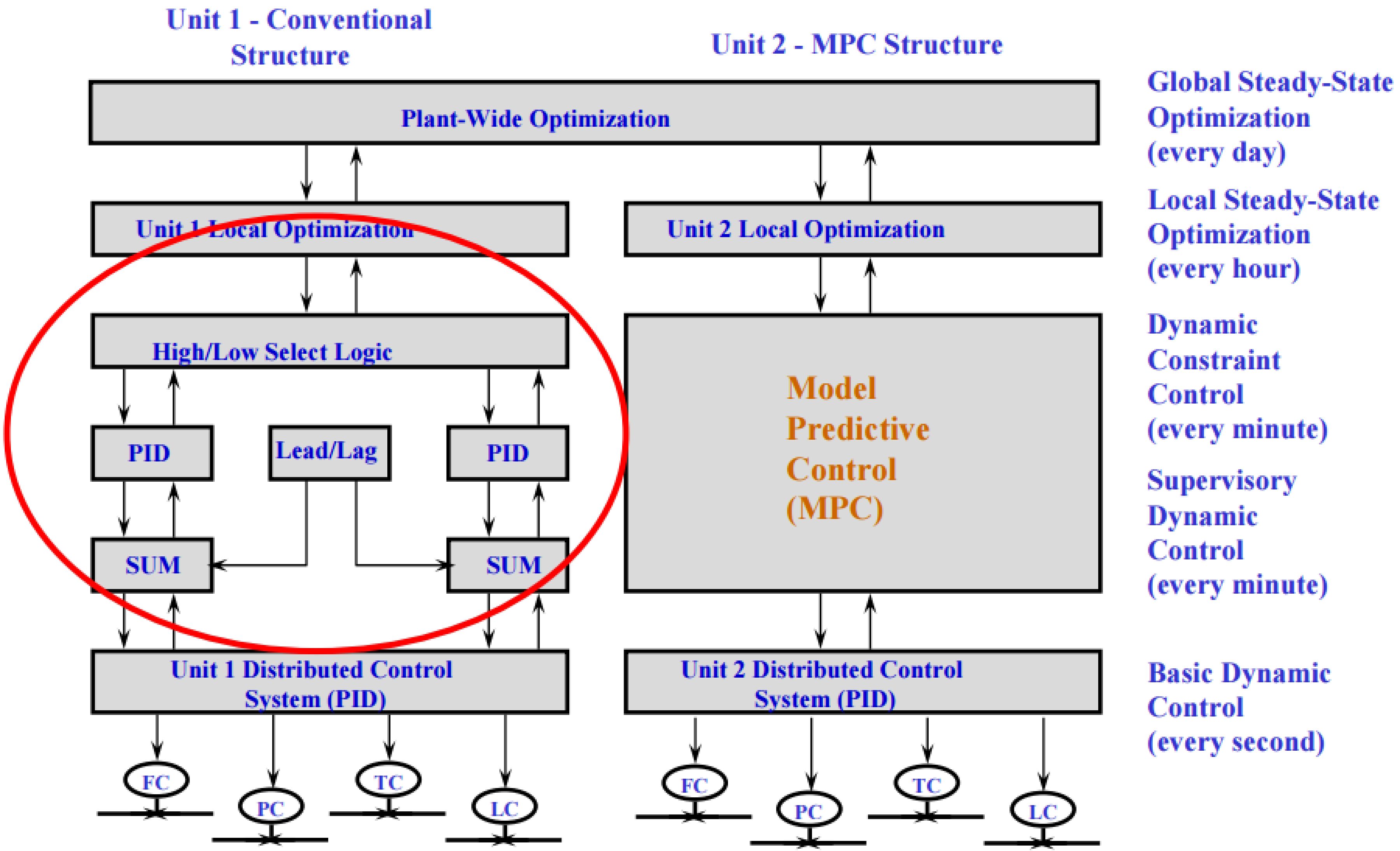
```
import casadi.*

% Create NLP: Solve the Rosenbrock problem:
%     minimize    x^2 + 100*z^2
%     subject to  z + (1-x)^2 - y == 0
x = SX.sym('x');
y = SX.sym('y');
z = SX.sym('z');
v = [x;y;z];
f = x^2 + 100*z^2;
g = z + (1-x)^2 - y;
nlp = struct('x', v, 'f', f, 'g', g);

% Create IPOPT solver object
solver = nlpsol('solver', 'ipopt', nlp);

% Solve the NLP
res = solver('x0' , [2.5 3.0 0.75],... % solution guess
             'lbx', -inf,...           % lower bound on x
             'ubx',  inf,...           % upper bound on x
             'lbg',    0,...           % lower bound on g
             'ubg',    0);             % upper bound on g

% Print the solution
f_opt = full(res.f)          % >> 0
x_opt = full(res.x)          % >> [0; 1; 0]
lam_x_opt = full(res.lam_x)  % >> [0; 0; 0]
lam_g_opt = full(res.lam_g)  % >> 0
```

Model predictive control



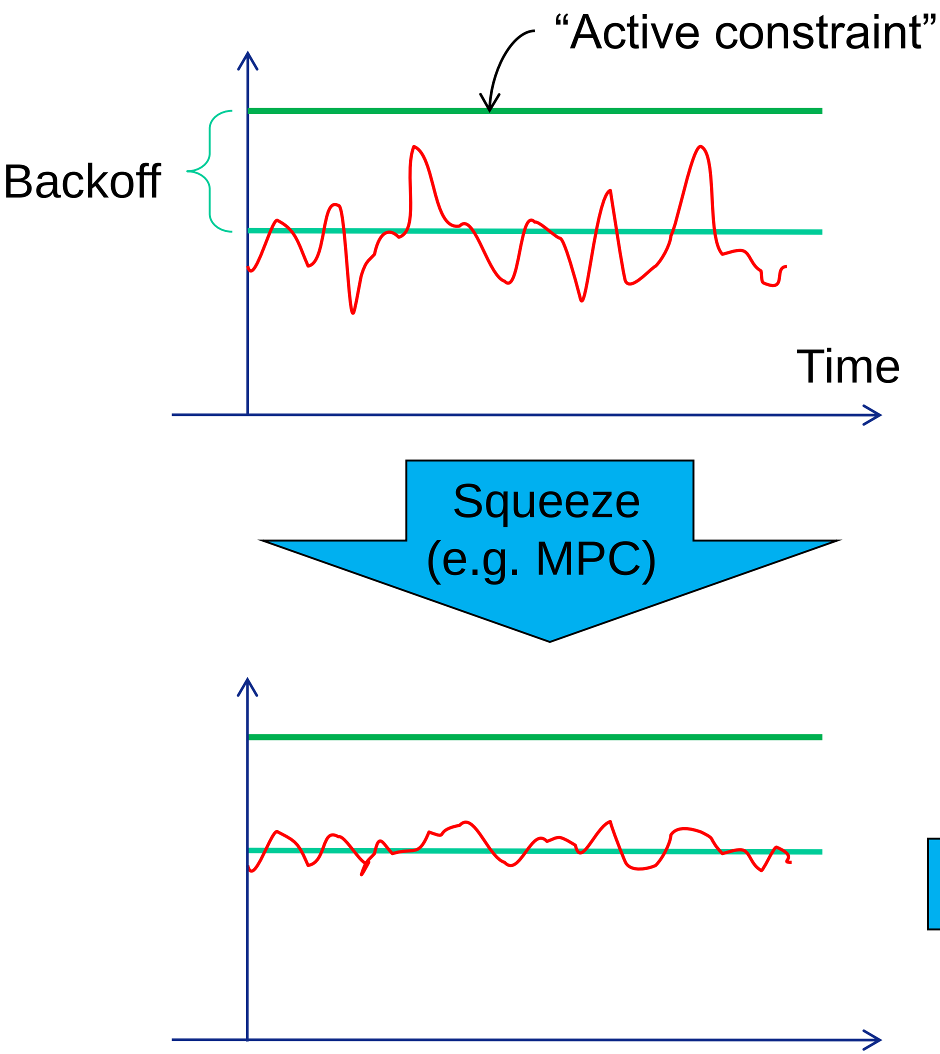
A model of the process is used to compute the control signals (inputs) that optimize predicted future process evolution

Operational hierarchy before and after MPC

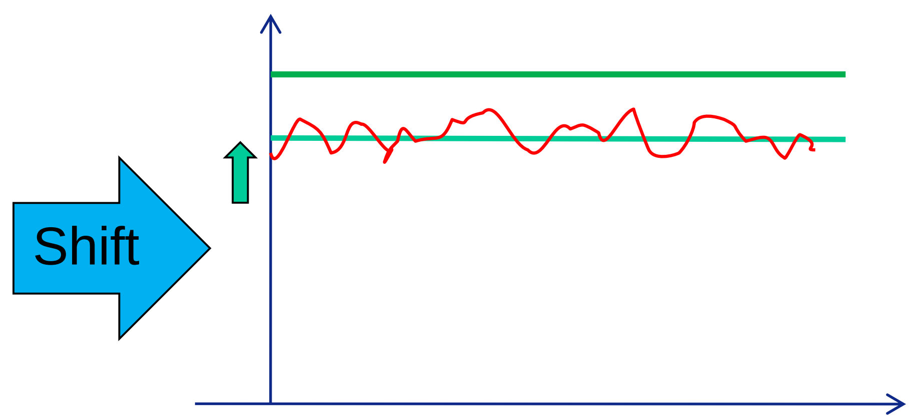


"Squeeze and shift"

How advanced control (MPC) improves economy



Backoff = loss (lower throughput, off-spec product, ...)



Embedded Model Predictive Control

PhD project Giorgio Kufoalor

Statoil

Traditional MPC

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

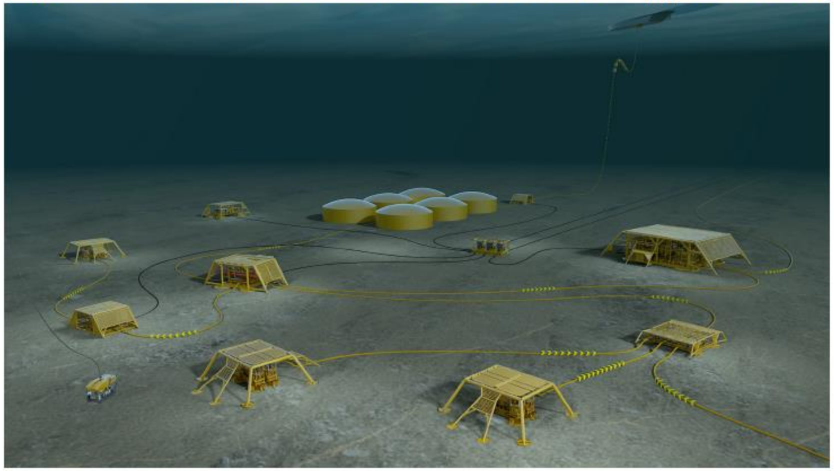
(M. Morari, 2013)

Embedded MPC

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

(M. Morari, 2013)

Embedded MPC for new industrial applications

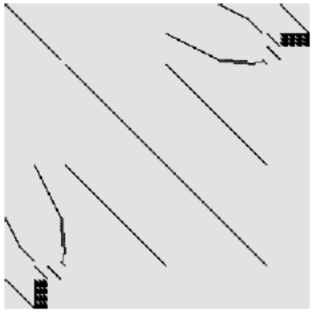


(Statoil subsea factory)

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, T. A. Johansen, L. S. Imsland, **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, L. S. Imsland, T. A. Johansen, 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

Linear MPC; open loop dynamic optimization

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

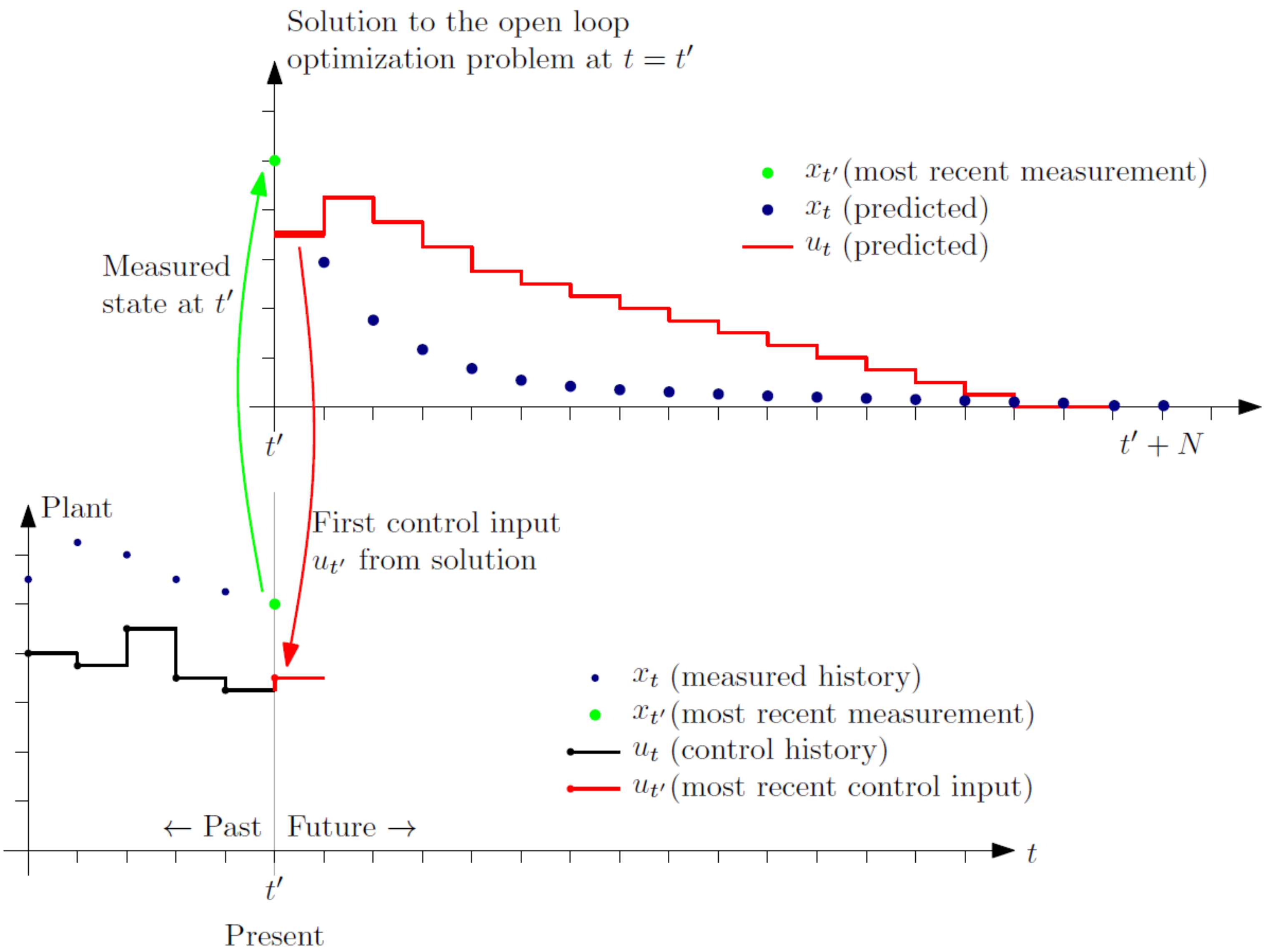
subject to

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

where

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

Model predictive control principle



The three ways of implementing NMPC

- Sequential (single shooting) methods
  - Have only inputs (control moves) as optimization variables, simulate to calculate objective and state constraints (and gradients)
  - Results in “small” optimization problem with no structure
  - “Standard” SQP methods are suitable
- Simultaneous methods
  - Have both inputs and states as optimization variables, include model as equality constraints
  - Results in huge optimization problem, but equality constraints are very structured (“sparse”, a lot of zeros)
  - Must use solvers that exploit structure (e.g. IPOPT)
- Inbetween: 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 “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)
  – In general, discretization scheme important: more accurate discretization may make it possible to use longer sample intervals (however, several aspects influences this decision)

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

Implementation: Tor Aksel Heirung

Output feedback MPC

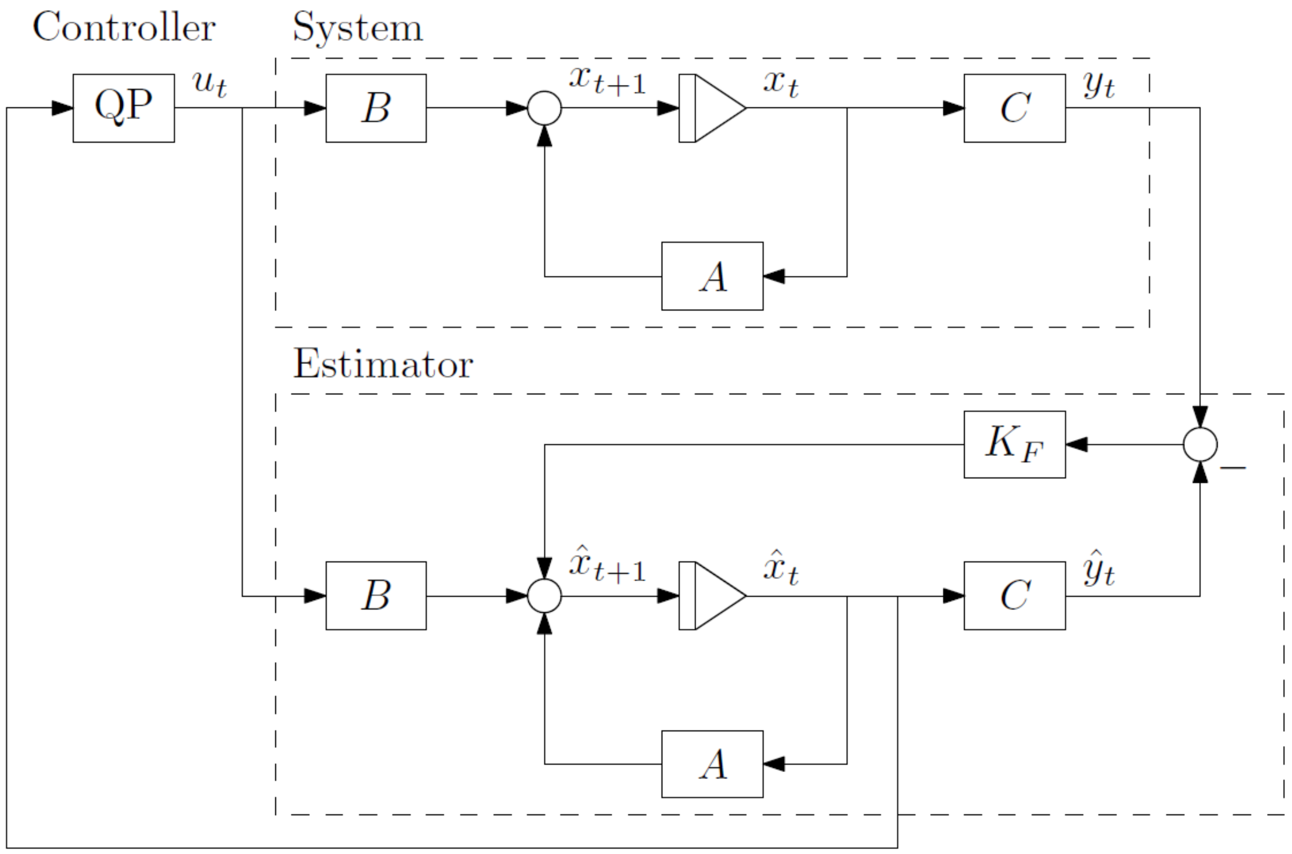


Figure 4.3: The structure of an output feedback linear MPC.

MPC and feasibility

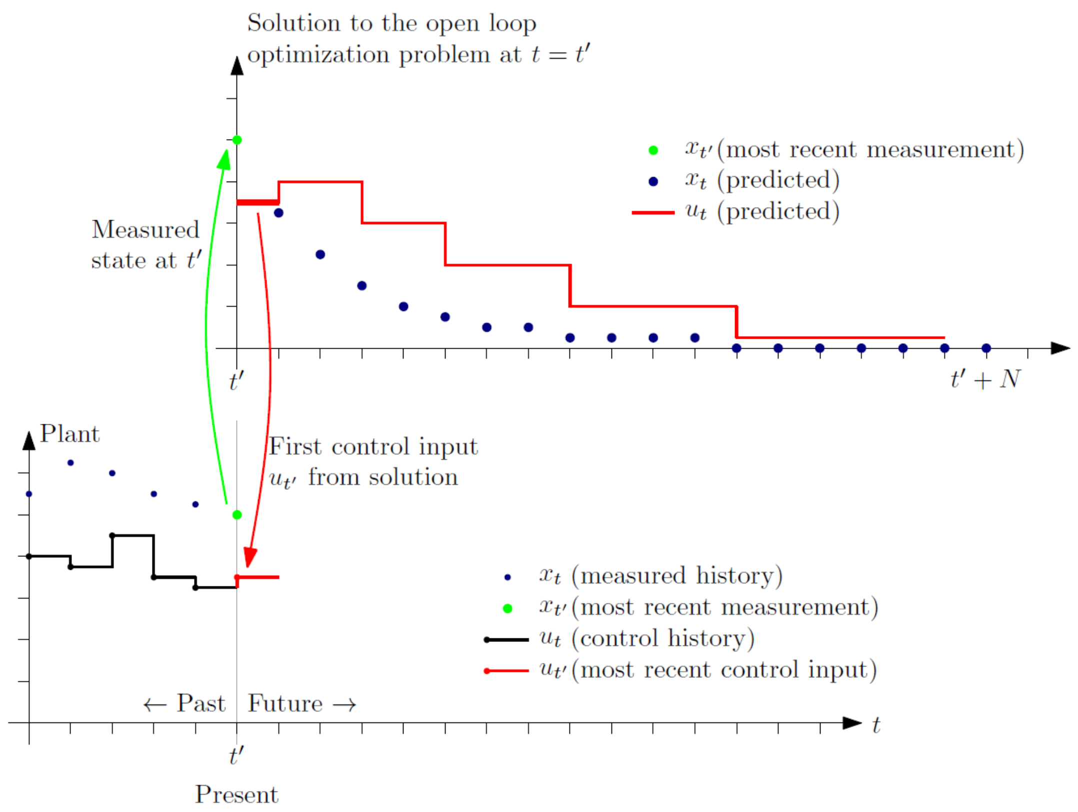
Is there always a solution to the MPC open-loop optimization problem?

- Not necessarily – state (or output) constraints may become infeasible, for example after a disturbance
- Practical solution: Soft constraints (or "exact penalty" formulations)
  - "Soften" ("relax") state constraints by adding "slack variables"

$$
\begin{aligned}
\min_{z \in \mathbb{R}^n} f(z) = \sum_{t=0}^{N-1} \frac{1}{2} x_{t+1}^\top Q_{t+1} x_{t+1} + \frac{1}{2} u_t^\top R_t u_t + \rho^\top \epsilon \\
\text{s.t.} \quad x_{t+1} = A_t x_t + B_t u_t, \quad t = \{0, \dots, N-1\} \\
x^{\text{low}} - \epsilon \leq x_t \leq x^{\text{high}} + \epsilon, \quad t = \{1, \dots, N\}, \qquad \epsilon > 0 \\
\vdots
\end{aligned}
$$

Complexity reduction strategies in MPC

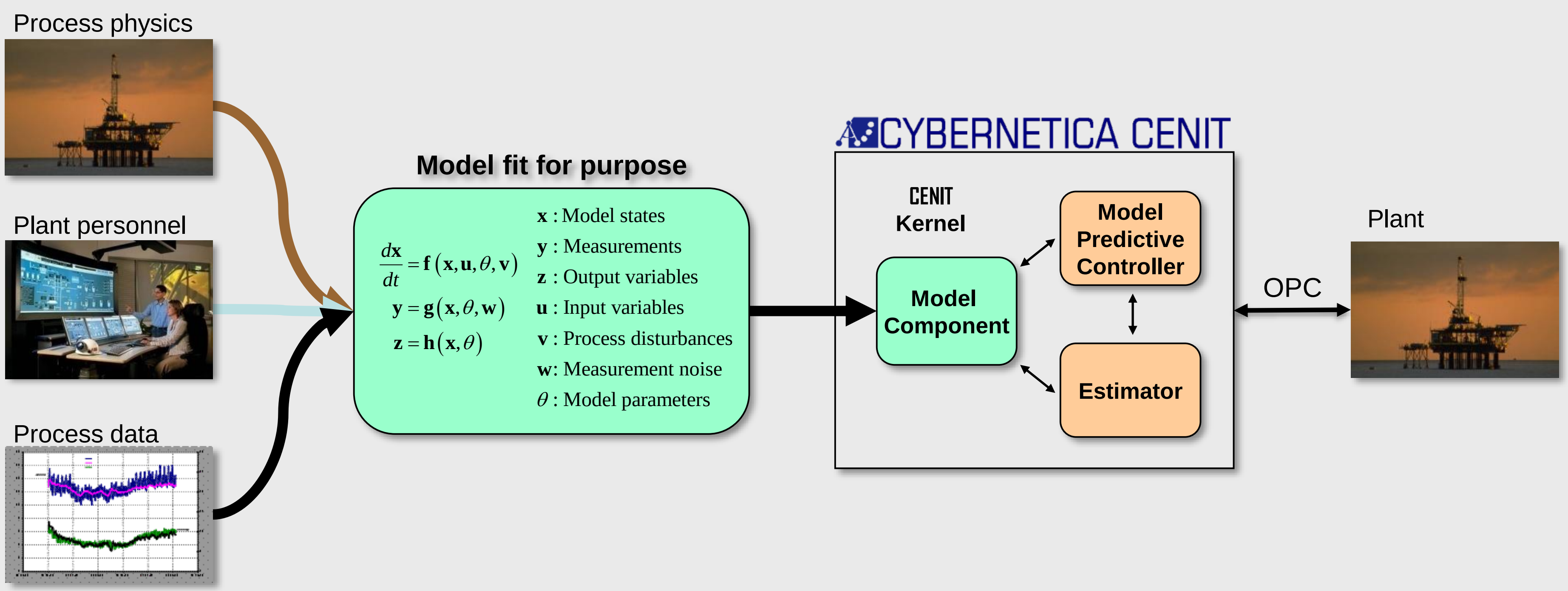
- Input blocking (or move blocking) – reduce number of QP variables



- "Incident points" – reduce number of QP constraints
  - Only check constraints at certain time instants, rather than at all times on horizon

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)



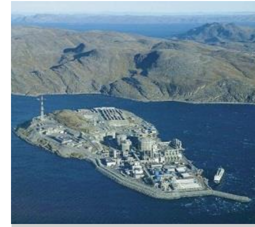
SEPTIC

Statoil Estimation and Prediction Tool for Identification and Control

Total in Statoil:
92 MPC Applications

Åsgard
Norne
Heidrun

#7



#4

Snøhvit

Mongstad

#28

Kollsnes

#5

Gullfaks/Tordis

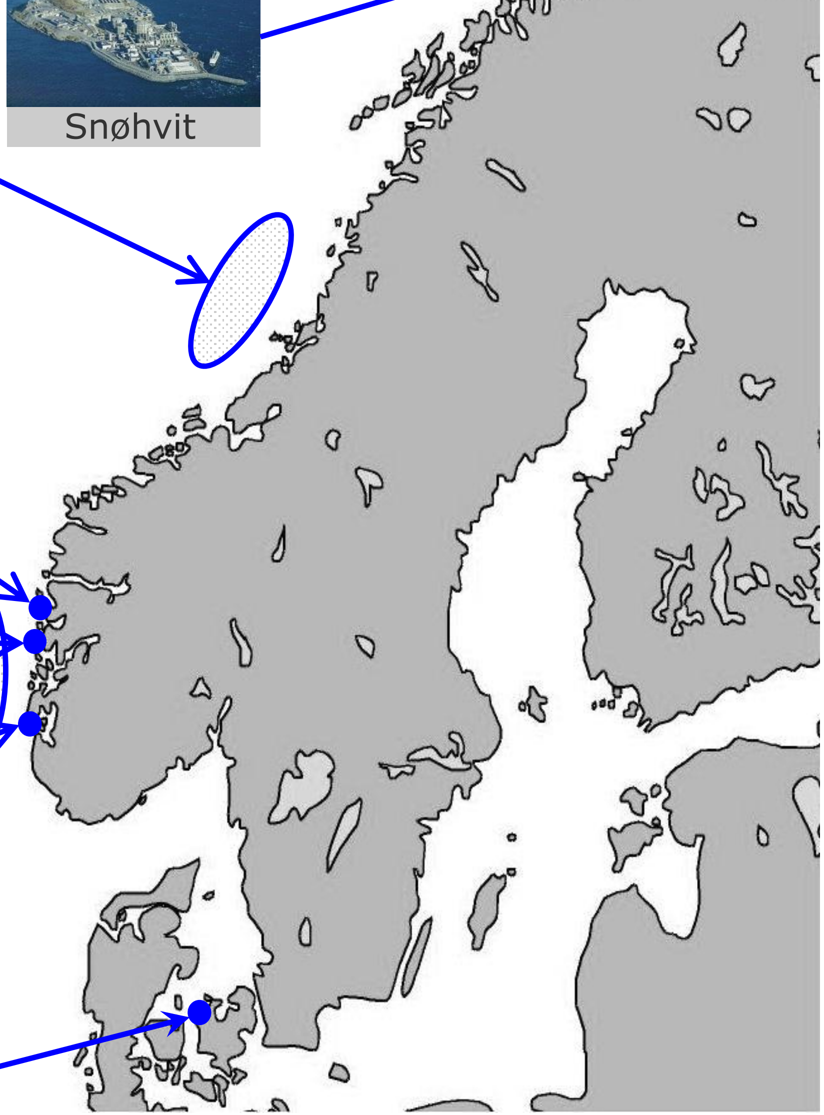
#2

Kårstø

#25

Kalundborg

#21



Stig Strand, Statoil