TTK4135 Optimization and Control Course Summary

Martin Brandt

Learning goals:

* Ability to formulate appropriate engineering problems as a mathematical optimization problem.
* Knowledge of typical engineering problems which are suitable for optimization.
* Ability to analyze and solve an optimization problem; in particular linear programs (LP), quadratic programs (QP) and nonlinear programs (NP).
* Ability to analyze and design optimal controllers; in particular Model Predictive Controllers (MPC).
* Knowledge of optimization software.

CONTENTS

1 | UNCONSTRAINED OPTIMIZATION

1.1 | Fundamentals of Unconstrained Optimization

Consider the general unconstrained optimization problem:

$$\min_{x \in \mathbb{R}^n} f(x) \tag{1}$$

Global minimizer: $f(x^*) \leq f(x) \forall x$
Local minimizer: $f(x^*) \leq f(x) \forall x \in \mathcal{N}(x^*)$, where $\mathcal{N}(x^*)$ is a neighbourhood around x.
A strict minimizer has strict inequalities.

Necessary conditions: x^* is a local solution $\implies \nabla f(x^*) = 0 \wedge f \in C^1$
Sufficient conditions: $f \in C^2 \wedge \nabla f(x^*) = 0 \wedge \nabla^2 f(x^*) > 0$

Theorem 1 *If f is convex and x^* is a local solution $\implies$ x^* is a global solution.*

1.2 | Line Search Methods

Consider the general line search:
Line search algorithm

```
given initial guess $x_0$, $k = 0$
while termination criteria not fulfilled do
    find descent direction $p_k$
    traverse along $p_k$ to $x_{k+1} = x_k + \alpha p_k$,    $k++$
end while
```

Termination criteria: $\|\nabla f(x_k)\| \leq \epsilon, \quad \|x_k - x_{k-1}\| \leq \epsilon, \quad |f(x_k) - f(x_{k-1})| \leq \epsilon, \quad k > k_{\max}$

1.2.1 | Descent Directions

* **Steepest descent direction:** $p_k = -\nabla f(x_k)$,   linear convergence
* **Newton direction:** $p_k = -(\nabla^2 f(x_k))^{-1} \nabla f(x_k)$,   quadratic convergence
* **Quasi-Newton direction:** $p_k = -B_k^{-1} \nabla f(x_k)$, $B_k \approx \nabla^2 f(x_k)$,   superlinear convergence

Remark Notice that steepest descent is the linear approximation and Newton direction is the quadratic. By using the curvature information in addition to the gradient we get a better, but more expensive descent direction. In general we need $B_k > 0$ for all descent directions, which is not guaranteed for Newton!

1.2.2 | Scaling

When doing optimization in general we need to be aware of the scaling of the different variables in relation to each other i.e. that some directions give much larger objective function value than others. Steepest descent is very sensitive to scaling, while Newton is scale invariant.

1.2.3 | Globalization Strategies

The ideal way to choose α would be to minimize $\varphi(\alpha) = f(x_k + \alpha p_k)$, which is called *exact line search*, but ain't nobody got time for that!
Instead we do *inexact line search*, where we look for an α that fulfills a sufficient decrease in $\varphi(\alpha)$ and a curvature condition, such that f decreases sufficiently and the slope of f is reduced sufficiently. This can be concretized by the **Wolfe conditions:**

$$f(x_k + \alpha_k p_k) \le f(x_k) + c_1 \alpha_k \nabla f_k^T p_k \tag{2a}$$
$$\nabla f(x_k + \alpha_k p_k)^T p_k \ge c_2 \nabla f_k^T p_k \tag{2b}$$

Step-length selection by interpolation: if Wolfe is not satisfied for α_0, minimize a quadratic interpolation of φ, if Wolfe is still not satisfied minimize a cubic interpolation, if Wolfe is still not satisfied reduce α_0 and retry.

Hessian modification: Newton direction is given by $\nabla^2 f(x_k) p_k = -\nabla f(x_k)$, and since $\nabla^2 f(x_k)$ is typically not positive definite far from the optimum, there might not be a solution or p_k might not be a descent direction. The solution is to replace $\nabla^2 f(x_k)$ with $\nabla^2 f(x_k) + E_k > 0$, or do eigenvalue decomposition and alter the negative eigenvalues to be some small positive δ.

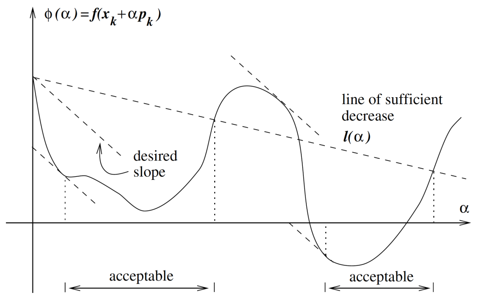

FIGURE 1 Wolfe conditions.

1.2.4 | Quasi-Newton

Model $f(x)$ at x_k:

$$f(x_k + p) \approx m_k(p) = f_k + \nabla f_k^\top p + \frac{1}{2} p^\top \nabla^2 f_k p \implies p_k = \arg\min_p m_k(p) = -\nabla^2 f_k^{-1} \nabla f_k \tag{3}$$

This method i.e. Newton has fast convergence, but it is expensive, so we approximate:

$$f(x_k + p) \approx m_k(p) = f_k + \nabla f_k^\top p + \frac{1}{2} p^\top B_k p \tag{4}$$

We find that

$$B_{k+1} \alpha_k p_k = \nabla f_{k+1} - \nabla f_k = B_{k+1} s_k = y_k \tag{5}$$

This equation (the secant equation) has infinitely many solutions, so we choose the closest to B_k:

$$B_{k+1} = \arg\min_B \|B - B_k\|, \quad B = B^\top > 0, \quad B s_k = y_k \tag{6}$$

For the weighted Fobenius norm we get DFP:

$$B_{k+1} = \left(I - \rho_k y_k s_k^T\right) B_k \left(I - \rho_k s_k y_k^T\right) + \rho_k y_k y_k^T, \quad \rho_k = \frac{1}{y_k^T s_k} \tag{7}$$

But isn't it kind of stupid to find B_{k+1}, when what we really need is $H_k = B_{k+1}^{-1}$, you ask? Yes my friend, yes it is.

Using the holy matrix inversion lemma:

$$(A + UCV)^{-1} = A^{-1} - A^{-1}U(C^{-1} + VA^{-1}U)^{-1}VA^{-1} \tag{8}$$

we get that the inverse DFP is:

$$H_{k+1} = H_k - \frac{H_k y_k y_k^T H_k}{y_k^T H_k y_k} + \frac{s_k s_k^T}{y_k^T s_k} \tag{9}$$

Alternatively, we choose

$$H_{k+1} = \arg\min_H \|H - H_k\|, \quad H = H^\top > 0, \quad H y_k = s_k \tag{10}$$

which is much more logical, and get BFGS:

$$H_{k+1} = \left(\mathcal{I} - \rho_k s_k y_k^T\right) H_k \left(\mathcal{I} - \rho_k y_k s_k^T\right) + \rho_k s_k s_k^T \tag{11}$$

This matrix often get too big, so limited memory BFGS (L-BFGS) stores the m last s_k and y_k and calculates H_k recursively.

1.3 | Calculating Derivatives

1.3.1 | Numerical Differentiation

Directional derivative:

$$\nabla f^\top p \approx \frac{f(x + \epsilon p) - f(x)}{\epsilon} \tag{12}$$

We approximate the gradient by letting $p = e_i$.
One-sided difference:

$$\frac{\partial f}{\partial x_i} = \frac{f(x + \epsilon e_i) - f(x)}{\epsilon} + O(\epsilon) \tag{13}$$

Two-sided difference:

$$\frac{\partial f}{\partial x_i} = \frac{f(x + \epsilon e_i) - f(x - \epsilon e_i)}{2\epsilon} + O(\epsilon^2) \tag{14}$$

The Hessian can be approximated by doing finite differences on the gradient:

$$\nabla^2 f(x) p \approx \frac{\nabla f(x + \epsilon p) - \nabla f(x)}{\epsilon} \tag{15}$$

1.3.2 | Automatic Differentiation

Find analytic values of derivatives by automated use of chain rule. We decompose f into a computational graph and calculate derivatives of all intermediate variables in the graph.
Forward mode: recursively calculate derivatives of sub-expressions with chain rule top-down.
Reverse mode: recursively calculate derivatives from sub-expressions with chain rule bottom-up.

1.4 | Derivative-Free Optimization

Generally, if you can implement derivatives, do it! This is not always possible though, so let's explore derivative-free optimization. Either model-based (build model from data) or based on metaheuristics (like genetic algorithms).

1.4.1 | Nelder-Mead Method

Simplex: generalized triangle in $\mathbb{R}^n$.
Given the initial nonsingular simplex $S = \{x_1, x_2, \dots, x_{n+1}\}$, we define $V(S) = [x_2 - x_1, x_3 - x_1, \dots, x_{n+1} - x_1]$.
Let $f(x_1) \leq f(x_2) \leq \cdots \leq f(x_{n+1})$.
We define the centroid $\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$ and the line $\bar{x}(t) = \bar{x} + t(x_{n+1} - \bar{x})$.

Nelder-Mead iteration

```
compute $\bar{x}(-1)$ and $f_{-1} = f(\bar{x}(-1))$.
if $f(x_1) \leq f_{-1} < f(x_n)$ then
    reflected point is neither best nor worst in new simplex.
    "expand": replace $x_{n+1}$ with $\bar{x}(-1)$
else if $f_{-1} < f(x_1)$ then
    reflected point is better than current best.
    compute $\bar{x}(-2)$ and $f_{-2}$
    if $f_{-2} < f_{-1}$ then
        replace $x_{n+1}$ with $\bar{x}(-2)$
    else
        "expand"
    end if
else
    reflected point is worse than current worse.
    if $f(x_n) \leq f_{-1} < f(x_{n+1})$ then
        compute $\bar{x}(-\frac{1}{2})$ and $f_{-\frac{1}{2}}$
        if $f_{-\frac{1}{2}} < f_{-1}$ then
            try outside contraction
            replace $x_{n+1}$ with $\bar{x}(-\frac{1}{2})$
        end if
    else
        compute $\bar{x}(\frac{1}{2})$ and $f_{\frac{1}{2}}$
        if $f_{\frac{1}{2}} < f_{n+1}$ then
            try inside contraction
            replace $x_{n+1}$ with $\bar{x}(\frac{1}{2})$
        else
            contraction failed, shrink towards $x_1$
            replace $x_i$ with $\frac{1}{2}(x_1 + x_i)$,    $i = 2, 3, \dots, n+1$
        end if
    end if
end if
```

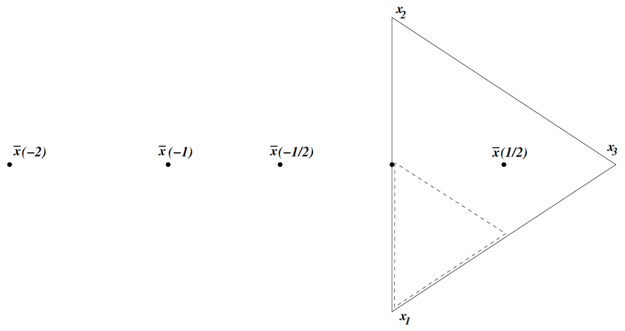

FIGURE 2 One step of Nelder-Mead in two dimensions.

1.5 | Nonlinear Equations

1.5.1 | Nonlinear Least Squares

Consider the nonlinear least squares problem:

$$\min_{x\in\mathbb{R}^n} f(x) = \frac{1}{2}\sum_{j=1}^{m} r_j^2(x) \tag{16}$$

We have the Jacobian $J = \begin{bmatrix}\nabla r_1(x)^\top & \dots & \nabla r_m(x)^\top\end{bmatrix}^\top$ and find that

$$\nabla^2 f(x) = J(x)^T J(x) + \sum_{j=1}^{m} r_j(x)\nabla^2 r_j(x) \approx J(x)^T J(x) \tag{17}$$

The Gauss-Newton method uses this approximation and Newton's method line search to solve the nonlinear least squares problem efficiently.

1.5.2 | Newton's Method

For the nonlinear equation system $r(x) = 0$ we have from Taylor that:

$$r(x_k + p) = r(x_k) + \int_0^1 J(x_k + tp)pdt \quad \approx M_k(p) = r(x_k) + J(x_k)p \tag{18}$$

$$M_k(p_k) = 0 \implies p_k = -J(x_k)^{-1} r(x_k) \tag{19}$$

Newton's method

```
choose $x_0$
while $\|r(x_k)|\ > \epsilon$ do
    solve $J(x_k)p_k = -r(x_k)$
    $x_{k+1} = x_k + p_k, \quad k++$
end while
```

Remark Why Newton? Well, if you think about the problem as minimizing x subject to zero gradient we see that the Jacobian is the Newton direction: $J(x_k) = \nabla^2 f(x_k)$.

Remark Newton's method has quadratic convergence!

Remark Issues: computing the Jacobian is expensive, and it might be singular $\rightarrow$ use Quasi-Newton with Hessian modification. Also, to ensure convergence when far from solution we should use line search, need to implement merit function for globalization.

2 | CONSTRAINED OPTIMIZATION

2.1 | Fundamentals of Constrained Optimization

Consider the general constrained optimization problem:

$$\min_{x\in\mathbb{R}^n} f(x) \quad \text{subject to} \quad \begin{cases} c_i(x) = 0, & i \in \mathcal{E} \\ c_i(x) \geq 0, & i \in \mathcal{I} \end{cases} \tag{20}$$

Feasible set: the region in $\mathbb{R}^n$ that satisfies the constrains, i.e.

$$\Omega = \{x | c_i(x) = 0, \quad i \in \mathcal{E}; c_i(x) \geq 0, \quad i \in \mathcal{I}\} \tag{21}$$

Active set: all active constraints for a given feasible x i.e.

$$\mathcal{A}(x) = \mathcal{E} \cup \{i \in \mathcal{I} | c_i(x) = 0\} \tag{22}$$

2.1.1 | Convexity

The set S is **convex** if: $\quad x, y \in S \implies \alpha x + (1-\alpha)y \in S, \quad \alpha \in [0,1].$
The function f is convex if: $\quad x, y \in S \implies f(\alpha x + (1-\alpha)y) \leq \alpha f(x) + (1-\alpha)f(y), \quad \alpha \in [0,1].$

Convex optimization problems have a convex feasible set and a convex cost function. These problems only have global solutions!

2.1.2 | KKT conditions

The first-order necessary conditions are based on the Lagrangian:

$$\mathcal{L}(x,\lambda) = f(x) - \sum_{i\in\mathcal{E}\cup\mathcal{T}} \lambda_i c_i(x) \tag{23}$$

Then the KKT conditions are:

$$\begin{aligned} \nabla_x \mathcal{L}\left(x^*, \lambda^*\right) &= 0 \\ c_i\left(x^*\right) &= 0, \quad \forall i \in \mathcal{E} \\ c_i\left(x^*\right) &\geq 0, \quad \forall i \in \mathcal{I} \\ \lambda_i^* &\geq 0, \quad \forall i \in \mathcal{I} \\ \lambda_i^* c_i\left(x^*\right) &= 0, \quad \forall i \in \mathcal{E} \cup \mathcal{I} \end{aligned} \tag{24a}$$

The KKT conditions are only true if LICQ holds, which says that *the set of all active constraint gradients ($\nabla c_i(x), i \in \mathcal{A}(x)$) must be linearly independent*.

Remark Sensitivity: Multipliers can be regarded as shadow prices i.e. they show the hidden costs of the constraints.

Remark The gradient is orthogonal to the contour lines, and points into the feasible region.

So $\nabla f^T d$ negative is a descent direction, zero is a feasible direction for equality constraints and positive is a feasible direction for inequalities. The feasible direction is then:

$$\mathcal{F}(x) = \left\{ d \middle| \begin{array}{ll} d^\top \nabla c_i(x) = 0, & \text{for all } i \in \mathcal{E}, \\ d^\top \nabla c_i(x) \geq 0, & \text{for all } i \in \mathcal{A}(x) \cap \mathcal{I} \end{array} \right\} \tag{25}$$

Tangent cone: the set of all tangents to Ω at x^* is called the tangent cone and is denoted by $T_\Omega(x^*)$. The difference between the feasible set and the tangent cone is that the feasible set is algebraic, while the tangent cone is geometric, they are usually similar or identical.

2.1.3 | Second-Order Conditions

The second derivatives of f are needed to determine if a move along the feasible direction with $w^T \nabla f(x^*) = 0$ will increase or decrease f.

We define **The Hessian:**

$$\nabla^2 f(x) = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix} \tag{26}$$

and the **critical cone:**

$$C\left(x^*, \lambda^*\right) = \left\{ w \in \mathcal{F}\left(x^*\right) \middle| \nabla c_i\left(x^*\right)^\top w = 0, \text{ all } i \in \mathcal{A}\left(x^*\right) \cap \mathcal{I} \text{ with } \lambda_i^* > 0 \right\} \tag{27}$$

which are the directions from the feasible directions for which it is not clear from first derivative information alone whether f will increase or decrease.

Given that LICQ and KKT are satisifed the second-order necessary conditions are then:

$$w^\top \nabla_{xx}^2 \mathcal{L}\left(x^*, \lambda^*\right) w \geq 0, \quad \forall w \in C\left(x^*, \lambda^*\right) \tag{28}$$

If the inequality is replaced with a strict inequality, the condition becomes sufficient.

2.2 | Linear Programming

Consider the standard form LP:

$$\min_{x\in\mathbb{R}^n} c^\top x, \quad \text{subject to } Ax = b, x \geq 0 \tag{29}$$

Remark Any linear program can be transformed to standard form by adding slack variables $z \geq 0$ and splitting: $x = (x^+ - x^-), \quad x^+ \geq 0, \quad x^- \geq 0.$

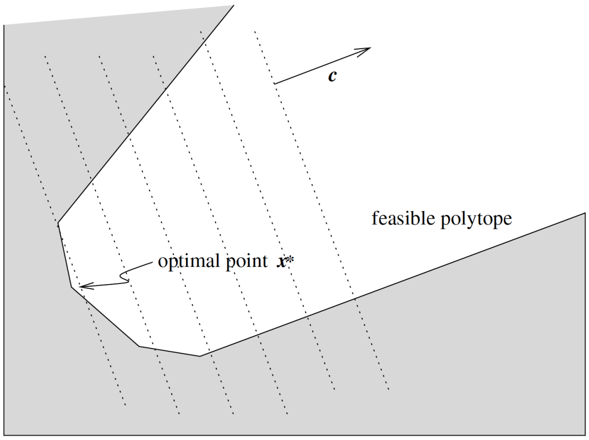

FIGURE 3 Example of a linear program.

Two special cases:

* **Infeasible:** $Ax = b$ has no solution.
* **Unbounded:** there exists a sequence $x_k \in \Omega$ such that $c^T x_k \to -\infty$.

2.2.1 | KKT for Linear Programming

The Lagrangian for a LP is:

$$\mathcal{L}(x, \lambda, s) = c^\top x - \lambda^\top (Ax - b) - s^\top x \tag{30}$$

such that the KKT conditions are:

$$\begin{aligned} A^\top \lambda + s &= c \\ Ax &= b \\ x &\geq 0 \\ s &\geq 0 \\ x_i s_i &= 0, \quad i = 1, 2, \dots, n \end{aligned} \tag{31}$$

Remark The KKT conditions for LPs are necessary and sufficient.

The dual LP:

$$\max_{\lambda\in\mathbb{R}^n} b^\top \lambda, \quad \text{subject to } A^\top\lambda \le c \tag{32}$$

If we compare the primal and dual KKT conditions we observe that they are equivalent.

Weak duality: $c^\top x \ge b^\top \lambda$
String duality: *if primal or dual has a finite solution, so has the other, and the objectives are equal. If primal or dual is unbounded, the other is infeasible.*

2.2.2 | SIMPLEX

We assume that A has full rank.

Basic feasible point: x is a basic feasible point if x is feasible and there exists a subset $\mathcal{B}(x)$ of the index set $1, 2, \ldots, n$ such that:
* $\mathcal{B}(x)$ contains exactly m indices
* $i \notin \mathcal{B} \implies x_i = 0$ i.e. inequality constraints are only inactive if $i \in \mathcal{B}(x)$.
* The $m \times m$ matrix $B = [A_i]_{i\in B}$ is nonsingular.

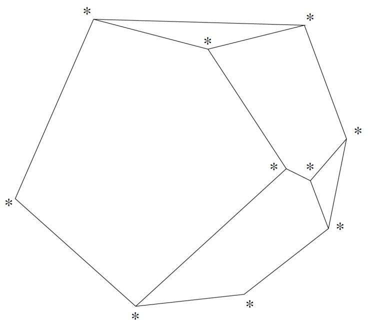

FIGURE 4 Feasible polytope with BFPs.

The SIMPLEX method generates iterates that are BFPs, and converges to a solution if there exists BFPs and at least one of them is a solution.

Theorem 2 *For a standard form LP we have that:*
* *If there is a feasible point, there is a BFP.*
* *If the LP has solutions, one solution is a basic optimal point.*
* *If the LP is feasible and bounded, there is a solution.*

Theorem 3 *All BFPs are vertices of the feasible polytope and vice versa.*

Definition A LP is degenerate if there is a BFP with $x_i = 0$ for $i \in \mathcal{B}(x)$.

Theorem 4 *If a LP is bounded and nondegenerate, then the SIMPLEX method terminates at a basic optimal point.*

We will denote $\mathcal{N}, N, x_N, \ldots$ as the complement to $\mathcal{B}, B, x_B, \ldots$.

SIMPLEX iteration

```
given a BFP $x_B$
solve $B^T\lambda = c_B$,
$s_N = c_N - N^T\lambda$.
if $s_N \ge 0$ then
    return $x_B$
end if
choose index $q \in \mathcal{N}$ such that $s_q < 0$,
```

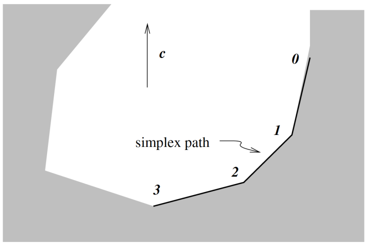

FIGURE 5 SIMPLEX iterations.

```
    solve $Bd = A_q$.
    if $d \leq 0$ then
        stop since problem is unbounded.
    end if
    Increase $x_q$ along $Ax = b$ until another component $x_p$ becomes 0.
    Add $q$ and remove $p$ from $\mathcal{B}$, repeat with new BFP.
```

2.3 | Quadratic Programming

Consider the general QP:

$$\min_{x\in\mathbb{R}^n} q(x) = \frac{1}{2}x^\top Gx + x^\top c \quad \text{subject to} \quad \begin{cases} c_i(x) = a_i^\top x - b_i = 0, & i \in \mathcal{E} \\ c_i(x) = a_i^\top x - b_i \geq 0, & i \in \mathcal{I} \end{cases} \tag{33}$$

Remark The feasible set is convex. The cost function is convex if the reduced Hessian $Z^T GZ$ is positive semi-definite, where the columns of Z span Null(A).

2.3.1 | Equality-constrained QPs

Consider the simplified QP where $\mathcal{I} = \emptyset$. The KKT conditions for the EQP are given by

$$\begin{bmatrix} G & A^\top \\ A & 0 \end{bmatrix}\begin{bmatrix} -p \\ \lambda^* \end{bmatrix} = \begin{bmatrix} c + Gx \\ Ax - b \end{bmatrix} = \begin{bmatrix} g \\ h \end{bmatrix}, \tag{34}$$

where we have done the change of variables $p = x^* - x$.

Alternatively we can use Z, the basis for Null(A), and Y, the basis for Range(A) to define $p = Zp_z + Yp_y$ to eliminate variables and get the reduced solution:

$$\left(Z^\top GZ\right) p_Z = -Z^\top GYp_Y - Z^\top g \tag{35}$$

2.3.2 | Active Set Method for QPs

For the general QP we get the following KKT conditions:

$$\begin{aligned} Gx^* + c - \sum_{i\in\mathcal{A}(x^*)} \lambda_i^* a_i &= 0 \\ a_i^\top x^* - b_i &= 0, \quad \forall i \in \mathcal{A}(x^*) \\ a_i^\top x^* - b_i &\geq 0, \quad \forall i \in \mathcal{I}\backslash\mathcal{A}(x^*) \\ \lambda_i^* &\geq 0, \quad \forall i \in \mathcal{I} \cap \mathcal{A}\left(x^*\right) \end{aligned} \tag{36}$$

Theorem 5 *If x^* satisfies KKT and $G \geq 0$, x^* is a global solution.*

If $\mathcal{A}(x^*)$ is known, the QP reduces to an EQP:

$$\min_x q(x) = \frac{1}{2}x^\top Gx + x^\top c \quad \text{subject to} \quad a_i^\top x - b_i = 0, \quad i \in \mathcal{A}\left(x^*\right), \tag{37}$$

which can be rewritten as:

$$\min_x q(x) = \frac{1}{2}p^\top Gp + g_k^T p \quad \text{subject to} \quad a_i^\top p = 0, \quad i \in \mathcal{W}_k \tag{38}$$

for $p = x - x_k, g_k = Gx_k + c$ and $\mathcal{W}_k$ being the working set for the current iteration.

QP active set iteration

```
given $\mathcal{W}_k$ and $x_k$
solve eq. (38) for $p_k$
if $p_k = 0$ then
  if $\lambda_k \geq 0$ then
    all KKT are fulfilled and $x^* = x_k$
  else
    pick index $i$ of a negative $\lambda_i$ and remove this index from $\mathcal{W}_k$ and repeat
  end if
else
  if $x_{k+1} = x_k + p_k$ is feasible then
    set $\mathcal{W}_{k+1} 0 \mathcal{W}_k$ and start over
  else
    find the blocking constraint:
      $a_i^T(x_k + \alpha_k p_k) \geq B_i$   for$a_i^T p_k < 0, \alpha_k = \min\{\frac{b_i - a_i^T x_k}{a_i^T p_k}, 1\}$
      $x_{k+1} = x_k + \alpha_k p_k, \mathcal{W}_{k+1} = \mathcal{W}_k + \{j\}$, start over
  end if
end if
```

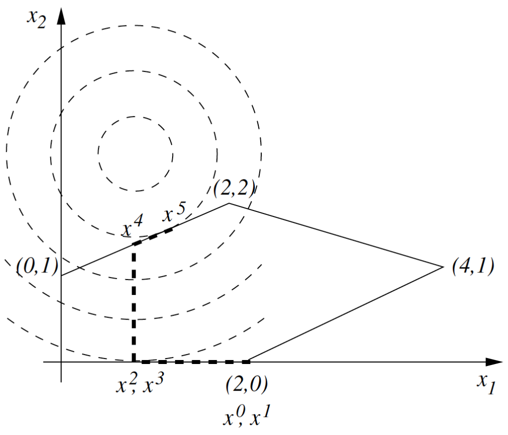

FIGURE 6 Active set QP iterations.

2.4 | Sequential Quadratic Programming

2.4.1 | Local SQP for equality-constrained NLPs

Consider the simplified NLP with only equality constraints:

$$\min_{x\in\mathbb{R}^n} f(x) \quad \text{subject to} \quad c(x)=0 \tag{39}$$

The first order KKT conditions are

$$F(x,\lambda)=\begin{bmatrix}\nabla f(x)-A(x)^T\lambda \\ c(x)\end{bmatrix}=0, \quad A(x)^T=[\nabla c_1(x),\nabla c_2(x),\dots,\nabla c_m(x)] \tag{40}$$

We assume that $A(x)$ has full row rank i.e. LICQ and that $\nabla_x^2\mathcal{L}(x,\lambda)$ is positive definite on the constraint tangent space. Then this is just a nonlinear equation system with $n+m$ equations and variables $\rightarrow$ solve that shit with Newton:

$$\begin{bmatrix}\nabla_x^2\mathcal{L} & -A(x_k)^\top \\ A(x_k) & 0\end{bmatrix}p=-F(x_k,\lambda_k) \tag{41}$$

Or by approximating the Lagrangian by a second order Taylor expansion and minimizing we get the equivalent equation

$$\begin{bmatrix}\nabla_x^2\mathcal{L} & -A(x_k)^\top \\ A(x_k) & 0\end{bmatrix}\begin{bmatrix}p\\ l\end{bmatrix}=-\begin{bmatrix}-\nabla f(x_k)\\ -c(x_k)\end{bmatrix} \tag{42}$$

Local SQP

```
given $x_0, \lambda_0$
**while** $||F(x_k,\lambda)| > \epsilon$ **do**
    solve $\begin{bmatrix}\nabla_x^2\mathcal{L} & -A(x_k)^\top \\ A(x_k) & 0\end{bmatrix}\begin{bmatrix}p\\ l\end{bmatrix}=-\begin{bmatrix}-\nabla f(x_k)\\ -c(x_k)\end{bmatrix}$
    $\begin{bmatrix}x_{k+1}\\ \lambda_{k+1}\end{bmatrix}=\begin{bmatrix}x_k\\ \lambda_k\end{bmatrix}+\begin{bmatrix}p_{x_k}\\ p_{\lambda_k}\end{bmatrix}, \quad k++$
**end while**
```

Remark So the system is really just an EQP! We can therefore interpret this as Newton's method (with quadratic convergence) on the KKT conditions, or sequential solutions to QP approximations of the NLP!

How do we extend to inequalities? We simply add the linearized inequality constraints as well:

$$\begin{aligned} \min_p \quad & f_k+\nabla f_k^T p+\tfrac{1}{2}p^T\nabla_{xx}^2\mathcal{L}_k p \\ \text{subject to} \quad & \nabla c_i(x_k)^T p+c_i(x_k)=0, \quad i\in\mathcal{E} \\ & \nabla c_i(x_k)^T p+c_i(x_k)\geq 0, \quad i\in\mathcal{I} \end{aligned} \tag{43}$$

So literally the entire algorithm is just to solve eq. (43) for p_k with the QP algorithm, update $x_{k+1}=x_k+p_k$ and repeat until convergence. Damn, that was easy. Almost too easy...

But in order to have a robust algorithm, we still need to add line search, BFGS, modification and that stuff...

2.4.2 | SQP Globalization

In the unconstrained case by add modified Hessian, BFGS, line search on $f(x)$ and move in Newton direction.

In the constrained case we modify and do BFGS on $\nabla^2 \mathcal{L}$. Furthermore we need a **merit function** to do the line search, as we want to measure progress in both $f(x)$ and $c(x)$.

l_1 **- merit function:**

$$\varphi(x,\mu)_1 = f(x) + \mu \sum_{i\in\mathcal{E}} |c_i(x)| + \mu \sum_{i\in\mathcal{I}} [c_i(x)]^{-1} \tag{44}$$

where $[z]^{-1} = \max\{0, -z\}$.

Definition $\varphi(x,\mu)$ is **exact** if for any $\mu \le \mu^* < 0$ the local solution of the NLP is a minimizer of φ.

Theorem 6 *φ_1 is exact with $\mu^* = \max\{|\lambda_i^*|, i \in \mathcal{E} \cup \mathcal{I}\}$.*

The directional derivative $\nabla\varphi_1^\top p$ cannot be used since the merit function is not defined everywhere, so we use $D(\varphi_1, p_k) = \nabla f - \mu\|c\|_1$ instead.

Theorem 7 *$D\left(\varphi_1\left(x_k;\mu\right);p_k\right) \le -p_k^T \nabla_{xx}^2 \mathcal{L}_k p_k - \left(\mu - \|\lambda_{k+1}\|_\infty\right)\|c_k\|_1$, and p_k is a descent direction if $\nabla^2 \mathcal{L}_k > 0$ and $\mu > \|\lambda k + 1\|_\infty$*

Sufficient decrease in line search:

$$\phi_1\left(x_k + \alpha_k p_k, \mu_k\right) \le \phi_1\left(x_k, \mu_k\right) + \eta\alpha_k D\left(\phi_1\left(x_k,\mu\right);p_k\right) \tag{45}$$

Maratos-effect: a merit function may reject good steps, must use better merit function or second order corrections to avoid this.

3 | OPTIMAL CONTROL

3.1 | Open Loop Dynamic Optimization

Consider the general dynamic optimization problem:

$$\min_{z} f(z) = \sum_{t=0}^{N-1} f_t(x_{t+1}, u_t) \quad \text{subject to} \quad \begin{cases} x_{t+1} = g_t(x_t, u_t), & t = 0, 1, \dots N-1 \\ h_t(x_t, u_t) \le 0, & t = 0, 1, \dots N-1 \end{cases} \tag{46}$$

where the initial state x_0 is given and N is the prediction horizon.

The standard open loop dynamic optimization problem is to optimize a quadratic cost function subject to a linear model:

$$\min_{z \in \mathbb{R}^n} f(z) = \sum_{t=0}^{N-1} \frac{1}{2} x_{t+1}^\top Q_{t+1} x_{t+1} + d_{xt+1} x_{t+1} + \frac{1}{2} u_t^\top R_t u_t + d_{ut} u_t \tag{47a}$$

$$\text{subject to} \quad x_{t+1} = A_t x_t + B_t u_t, \qquad t = 0, \dots, N-1 \tag{47b}$$

$$x^{\text{low}} \le x_t \le x^{\text{high}}, \qquad t = 1, \dots, N \tag{47c}$$

$$u^{\text{low}} \le u_t \le u^{\text{high}}, \qquad t = 0, \dots, N-1 \tag{47d}$$

$$-\Delta u^{\text{high}} \le \Delta u_t \le \Delta u^{\text{high}}, \qquad t = 0, \dots, N-1 \tag{47e}$$

where $Q_t \ge 0, \quad R_t \ge 0, \quad \Delta u_t = u_t - u_{t-1}, \quad z^\top = \left(x_1^\top, \dots, x_N^\top, u_0^\top \dots, u_{N-1}^\top\right)$, and x_0 and u_{-1} are known.

Remark Notice that we have placed upper and lower bounds on the states, the inputs and the change in inputs. The constraints on the states can be generalized by instead adding constraints in regards to a new variable $y_t = h(x_t)$. Similarly the cost function can weight y instead of x.

Remark A common extension of eq. (47) is to add the weights $\frac{1}{2}\Delta u_t^T R_{\Delta t} \Delta u_t$ to penalize control moves, not just constrain them.

So how do we solve these ginormous problems?
Batch approach 1 (full space): We can rewrite the system to a more compact form:

$$\min_{z \in \mathbb{R}^n} \frac{1}{2} z^\top G z \quad \text{subject to} \quad \begin{cases} A_{\text{eq}} z & = b_{\text{eq}} \\ A_{\text{ineq}} z & \ge b_{\text{ineq}} \end{cases} \tag{48}$$

$$\min_{z \in \mathbb{R}^n} \frac{1}{2} z^\top \begin{bmatrix} R & 0 & 0 & \\ 0 & Q & 0 & \dots \\ 0 & 0 & R & \\ & \vdots & & \ddots \end{bmatrix} z \quad \text{subject to} \quad \begin{bmatrix} -B & I & 0 & \dots & \\ 0 & -A & -B & I & \dots \\ 0 & 0 & 0 & -A & \dots \\ & \vdots & & \ddots & \end{bmatrix} z = \begin{bmatrix} Ax_0 \\ 0 \\ 0 \\ \vdots \end{bmatrix}, \tag{49a}$$

$$\begin{bmatrix} I & 0 & 0 & \dots \\ -I & 0 & 0 & \dots \\ 0 & I & 0 & \dots \\ 0 & -I & 0 & \dots \\ & \vdots & \ddots & \end{bmatrix} z = \begin{bmatrix} u^{\text{high}} \\ -u^{\text{low}} \\ x^{\text{high}} \\ -x^{\text{low}} \\ \vdots \end{bmatrix} \tag{49b}$$

Batch approach 2 (reduced space):
By writing out each state as a linear combination of prior states and inputs we get:

$$x = S^x x_0 + S^u u = \begin{bmatrix} A \\ A^2 \\ A^3 \\ \vdots \\ A^N \end{bmatrix} x_0 + \begin{bmatrix} B & 0 & & \\ AB & B & 0 & \dots \\ A^2B & AB & B & \\ & \vdots & & \ddots \\ A^{N-1}B & A^{N-2}B & \dots & \end{bmatrix} u \tag{50}$$

so that we can solve the reduced problem

$$\min_u \frac{1}{2}(S^x x_0 + S^u u)^\top \bar{Q}(S^x x_0 + S^u u) + \frac{1}{2}u^\top \bar{R} u, \qquad x^{\text{low}} \leq (S^x x_0 + S^u u) \leq x^{\text{high}} \tag{51}$$

3.2 | Model Predictive Control

Let's close that loop baby!
Model predictive control is a form of control in which the current control action is obtained by solving, at each sampling instant, a finite horizon open loop optimal control problem, using the current state of the plant as the initial state; the optimization yields an optimal control sequence and the first control in this sequence is applied to the plant.

MPC algorithm

```
for t = 0,1,2,... do
    determine $x_t$
    solve open loop optimization problem for prediction horizon $t$ to $t + N$ with $x_0 = x_t$
    Apply the first control move $u_0 = u_t$
end for
```

Remark In general we can not guarantee feasibility of the MPC i.e. that the QP has a solution. Usually solved by slacking adding slack variable to the cost function and adding it to the constraints.

Remark Stability is also not guaranteed. Usually we just choose N "large enough", as the proper engineers we are (larger than dominant dynamics).

Remark For output feedback we can add a Kalman filter to estimate x_t to be used in the MPC algorithm. Alternatively we have MHE estimators, which is to the MPC kind of what the Kalman filter is to the LQR, i.e. it uses previous measurements in a optimization problem to estimate the next estimate.

MPC allows us to handle constraints in MIMO control problems transparently, and also we get "optimal" performance $\implies$ \$\$\$. But we do add online complexity, and model of the system is needed.

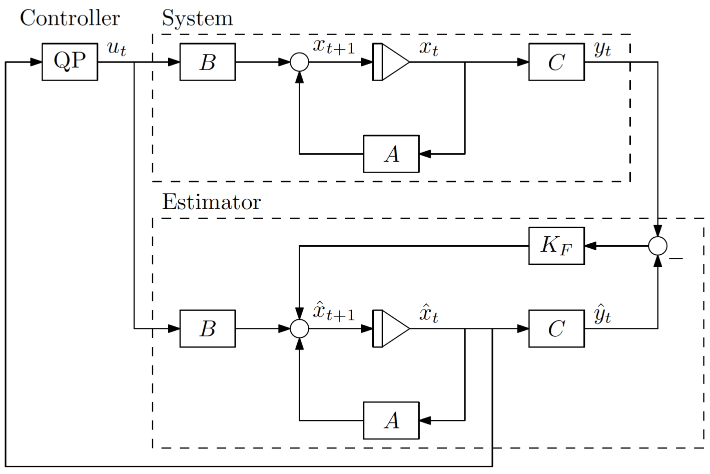

FIGURE 7 An output feedback system

3.2.1 | Reference Tracking

Want to control $y_t = Hx_t$ such that $y_t \to y_{\text{ref}}$. Steady state analysis gives: $y_s = H(I - A)^{-1} Bu_s$. There are many inputs that give the same steady state, so we need **target calculation**:

$$\begin{tikzcd} {} \arrow[r, "y_{\text{ref}}"] & \boxed{\text{Target Calculation}} \arrow[r] & \boxed{K_{\text{MPC}}} \arrow[r, "u_t"] & {} \end{tikzcd}$$

Where the controller now solves:

$$\min_{z \in \mathbb{R}^n} f(z) = \sum_{t=0}^{N-1} \frac{1}{2} (y_{t+1} - Hx_s)^\top Q (y_{t+1} - Hx_s) + \frac{1}{2}(u_t - u_s)^\top R(u_t - u_s) \tag{52}$$

3.2.2 | Offset-free MPC

We add the disturbances d_t as states, use a state estimator to estimate x_t and d_t and use the modified system in the MPC:

$$\begin{bmatrix} x_{t+1} \\ d_{t+1} \end{bmatrix} = \begin{bmatrix} A & A_d \\ 0 & I \end{bmatrix} \begin{bmatrix} x_t \\ d_t \end{bmatrix} + \begin{bmatrix} B \\ 0 \end{bmatrix} u_t \tag{53a}$$

$$y_t = \begin{bmatrix} C & C_d \end{bmatrix} \begin{bmatrix} x_t \\ d_t \end{bmatrix} \tag{53b}$$

3.3 | Linear Quadratic Control

3.3.1 | Finite Horizon LQ Control

We remove all inequalities from the linear dynamic optimization problem and get the LQ problem:

$$\min_{z\in\mathbb{R}^n} f(z) = \sum_{t=0}^{N-1} \frac{1}{2}x_{t+1}^\top Q_{t+1}x_{t+1} + \frac{1}{2}u_t^\top R_t u_t \quad \text{subject to} \quad x_{t+1} = A_t x_t + B_t u_t, \quad t = 0, 1, \dots N-1 \tag{54}$$

Theorem 8 *The LQ solution is given by*

$$u_t = -K_t x_t \tag{55}$$

where the feedback gain matrix K_t is the unique global solution, given by the discrete Ricatti equation:

$$\begin{aligned} K_t &= R_t^{-1}B_t^\top P_{t+1}\left(I + B_t R_t^{-1} B_t^\top P_{t+1}\right)^{-1} A_t, \quad t = 0, \dots, N-1 \\ P_t &= Q_t + A_t^\top P_{t+1}\left(I + B_t R_t^{-1} B_t^\top P_{t+1}\right)^{-1} A_t, \quad t = 0, \dots, N-1 \\ P_N &= Q_N \end{aligned} \tag{56}$$

Remark The gain matrix can be calculated independently of the states.

3.3.2 | Infinite Horizon LQ Control

By observing P_t from the finite horizon LQ controller we see that after a few iterations it converges to a stationary value. It turns out that if we let $N \to \infty$ we get an even simpler controller, with a constant gain matrix that removes the initial transient of the finite horizon controller!

$$\min_{z\in\mathbb{R}^n} f(z) = \sum_{t=0}^{\infty} \frac{1}{2}x_{t+1}^\top Q x_{t+1} + \frac{1}{2}u_t^\top R u_t \quad \text{subject to} \quad x_{t+1} = A_t x_t + B_t u_t, \quad t = 0, 1, \dots N-1 \tag{57}$$

Definition Stabilizability: all uncontrollable modes are stable.

Definition Detectability: all unobservable modes are stable.

Theorem 9 *(A, B) has to be stabilizable for the system to have a solution. Furthermore (A, D) where $Q = D^\top D$ has to be detectable for the solution to be stable.*

Theorem 10 *The infinite horizon LQ (LQR) solution is given by*

$$u_t = -Kx_t \quad \text{for} \quad 0 \le t \le \infty \tag{58}$$

where the feedback gain matrix K is given by:

$$\begin{aligned} K &= R^{-1}B^\top P\left(I + BR^{-1}B^\top P\right)^{-1} A \\ P &= Q + A^\top P\left(I + BR^{-1}B^\top P\right)^{-1} A \\ P &= P^\top \ge 0 \end{aligned} \tag{59}$$

Remark We can reduce the complexity by input blocking (reducing the number of variables by letting u be constant for certain time intervals) and incident points (only measure control variables at certain points in time).

Remark The combination of a LQR and a Kalman filter is usually called the LQG controller.

3.4 | Nonlinear Model Predictive Control

We will now use the same quadratic cost function, but assume a nonlinear model. This means the problem is now no longer convex. Can use SQP to solve the problem. Need to use EKF or UKF for output feedback. Input blocking and incident points could be used. While a quadratic cost function is enough for regulation and tracking, we can further generalize the problem to nonlinear cost functions, e.g. in economic MPC where we optimize the actual economic cost of something, which might be nonlinear.

A | MATRIX CALCULUS

In this course we use the following gradient definition:

$$\nabla f(x) = \left[\frac{\partial f}{\partial x}\right]^\top = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \vdots \\ \frac{\partial f}{\partial x_n} \end{bmatrix} \tag{60}$$

A few nice preliminary matrix calculus equations:

$$\nabla(c^\top x) = c \tag{61}$$

$$\frac{\partial}{\partial x}(Ax) = A \tag{62}$$

$$\nabla(\frac{1}{2}x^\top Gx) = \frac{1}{2}Gx + \frac{1}{2}G^\top x \tag{63}$$