OptReg cheat sheet / summary

Embla Flatlandsmo

May 2021

DISCLAIMER: This was written by one person just trying to get an overview of the subject. The formulations have not been reviewed by anyone so they might be wrong (or at least not entirely correct). That being said, a lot of this material is from the course book/slides.

Want to contribute? Check out `https://www.overleaf.com/5198544641ddtnxbzfttrp`

Contents

1 Optimization Problems

General formulation

$$\begin{aligned} &\min_{x\in\mathbb{R}^3} f(x) \\ &\text{subject to:} \quad c_i(x) = 0 \quad i \in \mathcal{E} \\ &\qquad\qquad\quad\; c_i(x) \geq 0 \quad i \in \mathcal{I} \end{aligned}$$

For unconstrained optimization, $\mathcal{E} = \mathcal{I} = \emptyset$.

For constrained optimization, the **feasible set** is the set of points x that satisfy the constraints:

$$\Omega = \{x \quad | \quad c_i(x) = 0, i \in \mathcal{E}; \quad c_i(x) \geq 0, i \in \mathcal{I}\} \tag{1}$$

Convexity

In a set: Any straight line segment between two points inside the set S, lies entirely inside S. That is, $x \in S$ and $y \in S$ satisfy $\alpha x + (1-\alpha)y \in S$ for all $\alpha \in (0,1]$.

For a function: If the following property is satisfied:

$$f(\alpha x + (1-\alpha)y) \leq \alpha f(x) + (1-\alpha)f(y), \quad \forall\alpha \in (0,1] \tag{2}$$

Another way to write this property:

$$f(x) \leq \lambda f(z) + (1-\lambda)f(x^*) < f(x^*) \tag{3}$$

where the line segment that joins x^* and $z, (f(z) < f(x^*)$ is given by:

$$x = \lambda z + (1-\lambda)x^*, \quad \text{for some } \lambda \in (0,1]. \tag{4}$$

Convex Programming

Meets the requirements:

- The objective function is a convex function
- The feasible set is a convex set

The feasible set is convex if:

- the equality constraints, $c_i(\cdot), i \in \mathcal{E}$ are linear
- the inequality constraints $c_i(\cdot), i \in \mathcal{I}$ are concave

Global vs Local Minimizers

A point x* is a *global minimizer* if $f(x^*) \leq f(x), \forall x, \quad x \in \mathbb{R}^n$

When f is convex, any local minimizer x* is a global minimizer of f. If in addition f is differentiable, then any stationary point x* is a global minimizer of f

A point x* is a *local minimizer* if there is a neighborhood $\mathcal{N}$ of x such that $f(x^*) \leq f(x) \quad \forall x \in \mathcal{N}$

A point x* is a *strict/strong local minimizer* if there is a neighborhood $\mathcal{N}$ of x* such that $f(x^*) \leq f(x) \quad \forall x \in \mathcal{N}$ with $x \neq x^*$

A point x* is an *isolated local minimizer* if there is a neighbourhood $\mathcal{N}$ of x* such that x* is the only local minimizer of $\mathcal{N}$

Necessary and Sufficient Conditions for Optimality

First-Order Necessary Conditions: If x* is a local minimizer and if f is continuously differentiable in an open neighborhood of x*, then $\nabla f(x^*) = 0$
Second-Order Necessary Conditions: If x* is a local minimizer of f and $\nabla^2 f$ exists and is continuous in an open neighborhood of x*, then $\nabla f(x^*) = 0$ and $\nabla^2 f(x^*)$ is positive semidefinite
Second-Order Sufficient Conditions: Suppose $\nabla^2 f$ is continuous in an open neighborhood of x* and that $\nabla f(x^*) = 0$ and $\nabla^2 f(x^*)$ is positive definite. Then x* is a strict local minimizer of f.

Gradient Approximation

Forward difference:

$$\frac{\partial f}{\partial x_i}(x) \approx \frac{f(x + \epsilon e_i) - f(x)}{\epsilon} \tag{5}$$

The approximation error is linear in terms of ϵ

1.1 Derivative-free optimization

Use sampled function values to determine the new iterate. The optimization problem usually comes in the form

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

where the evaluation of f(x) can come from experimental measurements or stochastic simulations.

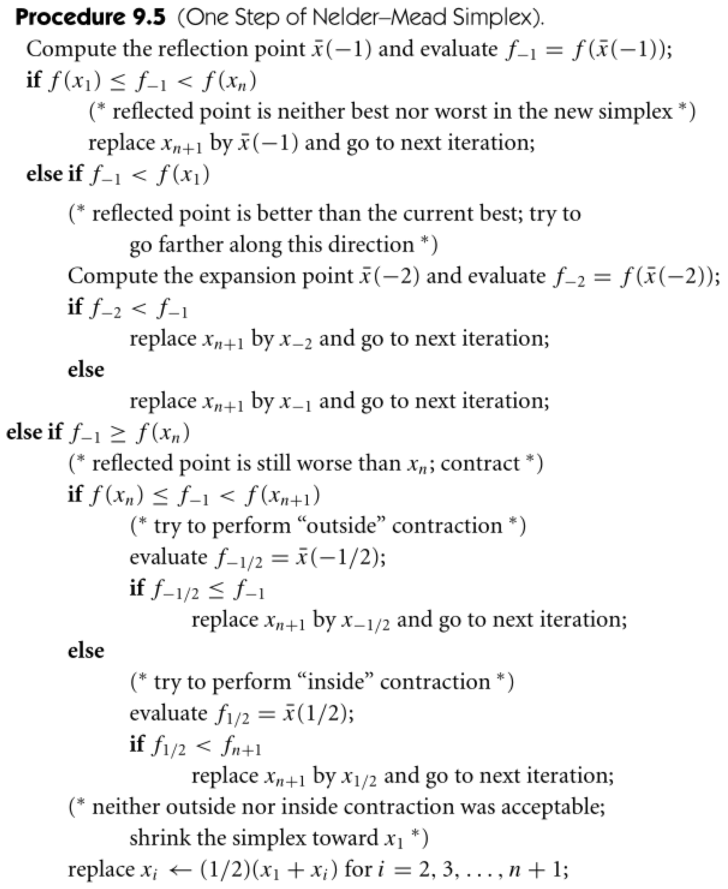
1.1.1 Nelder-Mead method

At any stage of the algorithm, we keep track of $n + 1$ points of interest in $\mathbb{R}^n$, whose convex hull forms a simplex (triangle? pyramid?). In a single iteration, we seek to remove the vertex with the worst function value and replace it with another point with a better value. The new point is obtained by reflecting, expanding or contracting the simplex along the line joining the worst vertex with the centroid of the remaining vertices. If we cannot find a better point in this manner, we retain only the vertex with the best function value, and we shrink the simplex by moving all other vertices toward this value.

Choosing starting points: If we let S denote the initial simplex with vertices$z_1, z_2, ..., z_{n+1}$, the requirement on these $n+1$ points is that the matrix

$$V(S) = [z_2 - z_1, z_3 - z_1, \ldots, z_{n+1} - z_1] \tag{6}$$

is nonsingular. In geometric terms, an invalid set of points in$\mathbb{R}^2$ forms a line. A valid set of points in $\mathbb{R}^2$ forms a triangle.



Procedure 9.5 (One Step of Nelder–Mead Simplex).

```
Compute the reflection point $\bar{x}(-1)$ and evaluate $f_{-1} = f(\bar{x}(-1))$;
if $f(x_1) \le f_{-1} < f(x_n)$
        (* reflected point is neither best nor worst in the new simplex *)
        replace $x_{n+1}$ by $\bar{x}(-1)$ and go to next iteration;
else if $f_{-1} < f(x_1)$
      (* reflected point is better than the current best; try to
            go farther along this direction *)
      Compute the expansion point $\bar{x}(-2)$ and evaluate $f_{-2} = f(\bar{x}(-2))$;
      if $f_{-2} < f_{-1}$
            replace $x_{n+1}$ by $x_{-2}$ and go to next iteration;
      else
            replace $x_{n+1}$ by $x_{-1}$ and go to next iteration;
else if $f_{-1} \ge f(x_n)$
      (* reflected point is still worse than $x_n$; contract *)
      if $f(x_n) \le f_{-1} < f(x_{n+1})$
            (* try to perform "outside" contraction *)
            evaluate $f_{-1/2} = \bar{x}(-1/2)$;
            if $f_{-1/2} \le f_{-1}$
                    replace $x_{n+1}$ by $x_{-1/2}$ and go to next iteration;
      else
            (* try to perform "inside" contraction *)
            evaluate $f_{1/2} = \bar{x}(1/2)$;
            if $f_{1/2} < f_{n+1}$
                    replace $x_{n+1}$ by $x_{1/2}$ and go to next iteration;
      (* neither outside nor inside contraction was acceptable;
            shrink the simplex toward $x_1$ *)
      replace $x_i \leftarrow (1/2)(x_1 + x_i)$ for $i = 2, 3, \ldots, n+1$;
```

Figure 1: One step of Nelder-Mead simplex

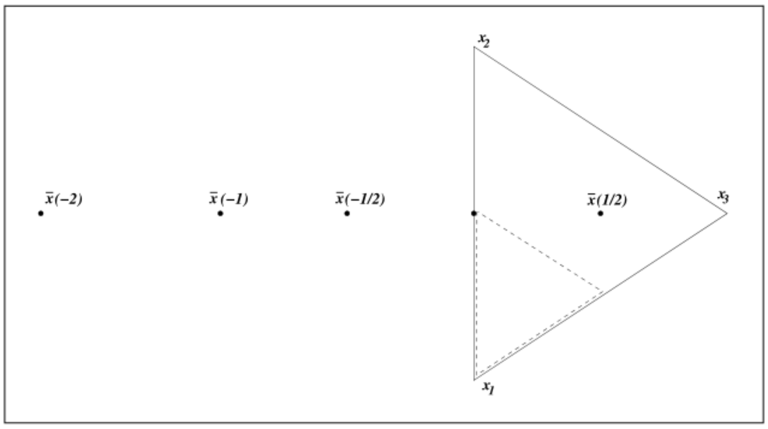


Figure 2: One step of the Nelder-Mead simplex method in $\mathbb{R}^3$, showing current simplex (solid triangle with vertices x_1, x_2, x_3), reflection point $\bar{x}(-1)$, expansion point $\bar{x}(-2)$, inside contraction point $\bar{x}(\frac{1}{2})$, outside contraction point $\bar{x}(-\frac{1}{2})$, and shrunken simplex (dotted triangle).

2 Line Search Methods

Choose a direction p_k and search along this direction from the current iterate x_k for a new iterate with a lower function value. The distance to move along p_k can be found by approximately solving the one-dimensional minimization problem to find a step length α:

$$\min_{\alpha>0} f(x_k + \alpha p_k) \tag{7}$$

That is, each iteration of a line search method is given by:

$$x_{k+1} = x_k + \alpha_k p_k \tag{8}$$

p_k is often required to be a descent direction: it must satisfy $p_k^T \nabla f_k < 0$

Steepest Descent Direction

Go in the direction along which f decreases most rapidly: $p_k = -\nabla f_k$

Newton Direction

$$p_k^N = -(\nabla^2 f_k)^{-1} \nabla f_k \tag{9}$$

Can only be used if $\nabla^2 f_k$ is positive definite. This way, we are sure that $(\nabla^2 f_k)^{-1}$ exists.

Quasi-Newton Search Directions

Replace $\nabla^2 f_k$ with B_k:

$$p_k = -B_k^{-1} \nabla f_k \tag{10}$$

One Hessian approximation is symmetric-rank-one (SR1) formula:

$$B_{k+1} = B_k + \frac{(y_k - B_k s_k)(y_k - B_k s_k)^T}{(y_k - B_k s_k)^T s_k} \tag{11}$$

Another Hessian approximation is the BFGS formula:

$$B_{k+1} = B_k - \frac{B_k s_k s_k^T B_k}{s_k^T B_k s_K} + \frac{y_k y_k^T}{y_k^T s_k} \tag{12}$$

Since B_k^{-1} is used, we could define $B_k^{-1} = H_k$ for ease of computation:

$$H_{k+1} = (\mathbb{I} - \rho_k s_k y_k^T) H_k (\mathbb{I} - \rho_k y_k s_k^T) + \rho_k s_k s_k^T \tag{13}$$

where $s_k = x_{k+1} - x_k = \alpha_k p_k$; $y_k = \nabla f_{k+1} - \nabla f_k$; $\rho_k = \frac{1}{y_k^T s_k}$

2.1 Deciding Step Length

Wolfe Conditions

α_k must satisfy the two conditions:
1: Sufficient decrease

$$f(x_k + \alpha p_k) \leq f(x_k) + c_1 \alpha \nabla f_k^T p_k, \quad c_1 \in (0,1) \tag{14}$$

2: Curvature conditions

$$\nabla f(x_k + \alpha_k p_k)^T p_k \geq c_2 \nabla f_k^T p_k, \quad c_2 \in (c_1, 1) \tag{15}$$

2.2 Convergence Rates

Steepest Descent

Suppose that $f : \mathbb{R}^N \longrightarrow \mathbb{R}$ is twice continuously differentiable, and that the iterates generated by the steepest-descent method with exact line searches converge to a point x* at which the Hessian matrix $\nabla^2 f(x^*)$ is positive definite. Let r be any scalar satisfying

$$r \in (\frac{\lambda_n - \lambda_1}{\lambda_n + \lambda_1}, 1)$$

where $\lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n$ are the eigenvalues of $\nabla^2 f(x^*)$. Then for all k sufficiently large, we have

$$f(x_{k+1}) - f(x^*) \leq r^2[f(x_k) - f(x^*)]$$

Newton's Method

Suppose that f is twice differentiable and that the Hessian $\nabla^2 f(x)$ is Lischitz continuous (smooth) in a neighborhood of a solution x* at which the sufficient conditions are satisfied. Consider the iteration $x_{k+1} = x_k + p_k$ where p_k is given by (9). Then:

- if the starting point x_0 is sufficiently close to x*, the sequence of iterates converges to x*
- the rate of convergence of x_k is quadratic
- the sequence of gradient norms $||\nabla f_k||$ converges quadratically to zero

Notes:

- When at least one of the eigenvalues of the Hessian is zero, the direction p_k^N does not exist since the Hessian is singular and thereby not invertible.
- When$\nabla^2 f_k$ is indefinite (both positive and negative eigenvalues), we cannot say whether p_k^N is a descent direction.

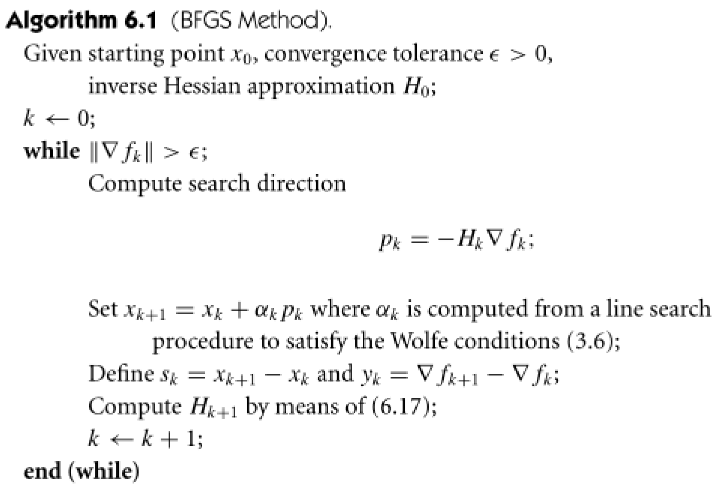
Quasi-Newton Methods

If the search direction of a quasi-Newton method approximates the Newton direction well enough, then the unit step length will satisfy the Wolfe conditions as the iterates converge to the solution.

Suppose that $f : \mathbb{R}^n \longrightarrow \mathbb{R}$ is twice continuously differentiable. Consider the iteration $x_{k+1} = x_k + p_k$, where p_k is given by (10). Assume that x_k converges to a point x* such that $\nabla f(x^*) = 0$ and $\nabla^2 f(x^*)$ is positive definite. Then x_k converges superlinearly if and only if the following condition holds:

$$\lim_{k \to \infty} \frac{||(B_k - \nabla^2 f(x^*))p_k}{||p_k||} = 0 \tag{16}$$

2.3 The 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 $\alpha_k$ 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)
```

We want to use BFGS over Newton when we don't have the capacity (or we don't want to bother) with the Hessian matrix. To implement this efficiently, the line search should:

- either satisfy Wolfe conditions or the strong Wolfe conditions
- always try step length of $\alpha_k = 1$ first, because this step length will eventually always be accepted.

Good Hessian approximations

1. $B_k > 0$ ensures descent direction
2. $B_k \approx \nabla^2 f_k$ ensures fast convergence
3. Cheap computation: Only use gradients to compute B_k

To determine if B_{k+1} is a good approximation it must fulfill the **secant condition**:

$$B_{k+1} s_k = y_k \tag{17}$$

Inverse update formula

We update B according to:

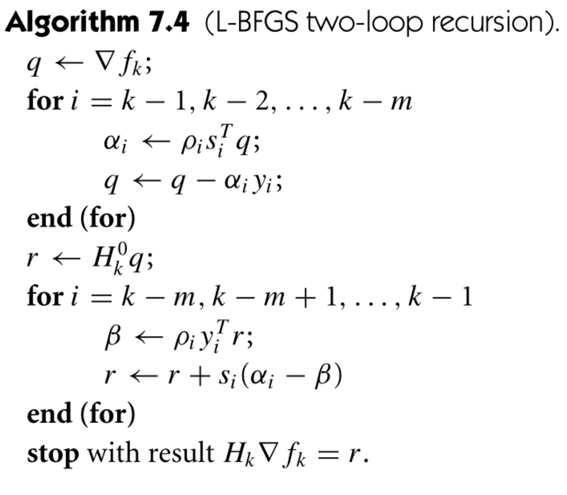
$$B_{k+1} = (\mathbb{I} - \rho_k y_k s_k^\intercal) B_k (\mathbb{I} - \rho_k s_k y_k^{)}) + \rho_k y_k y_k^{'} \quad \rho_k = \frac{1}{y_k^\intercal s_k} \tag{18}$$

But: since we need B_k^{-1} in $p_k = -B_k^{-1}\nabla f_k$ we can just update H_k:

$$H_{k+1} = H_k - \frac{H_k y_k y_k^\intercal H_k}{y_k^\intercal H_k y_k} + \frac{s_k s_k^\intercal}{y_k^\intercal s_k} \tag{19}$$

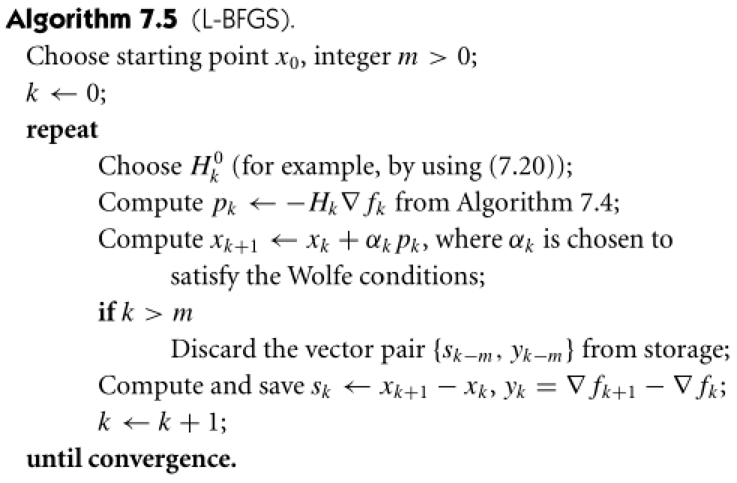
2.4 Limited-Memory BFGS (L-BFGS)

Useful for solving large problem whose Hessian matrices cannot be computed at a reasonable cost or are not sparse. The main idea of L-BFGS is to use curvature information from only the m most recent iterations to construct the Hessian approximation. m depends on the problem, but $3 \leq m \leq 20$ often produce satisfactory results.



Algorithm 7.4 (L-BFGS two-loop recursion).

```
$q \leftarrow \nabla f_k$;
for $i = k-1, k-2, \ldots, k-m$
    $\alpha_i \leftarrow \rho_i s_i^T q$;
    $q \leftarrow q - \alpha_i y_i$;
end (for)
$r \leftarrow H_k^0 q$;
for $i = k-m, k-m+1, \ldots, k-1$
    $\beta \leftarrow \rho_i y_i^T r$;
    $r \leftarrow r + s_i(\alpha_i - \beta)$
end (for)
stop with result $H_k \nabla f_k = r$.
```



Algorithm 7.5 (L-BFGS).

```
Choose starting point $x_0$, integer $m > 0$;
$k \leftarrow 0$;
**repeat**
    Choose $H_k^0$ (for example, by using (7.20));
    Compute $p_k \leftarrow -H_k \nabla f_k$ from Algorithm 7.4;
    Compute $x_{k+1} \leftarrow x_k + \alpha_k p_k$, where $\alpha_k$ is chosen to
        satisfy the Wolfe conditions;
    **if** $k > m$
        Discard the vector pair $\{s_{k-m}, y_{k-m}\}$ from storage;
    Compute and save $s_k \leftarrow x_{k+1} - x_k$, $y_k = \nabla f_{k+1} - \nabla f_k$;
    $k \leftarrow k + 1$;
**until convergence.**
```

Typically we choose $H_k^0 = \gamma_k \mathbb{I}$ where

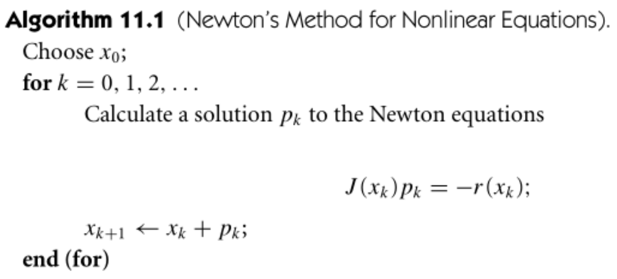
$$\gamma_k = \frac{s_{k-1}^T y_{k-1}}{y_{k-1}^T y_{k-1}} \tag{20}$$

3 Nonlinear Optimization

3.1 Unconstrained Nonlinear Optimization

3.1.1 Newton's Method for Nonlinear Equations

Define the system of non-linear equations as $\boldsymbol{r}(x) = \mathbf{0}$ and the Jacobian of $\boldsymbol{r}$ as $\boldsymbol{J}(x)$. Newton's method for nonlinear equations can then be applied:



Algorithm 11.1 (Newton's Method for Nonlinear Equations).

```
Choose $x_0$;
for $k = 0, 1, 2, \ldots$
    Calculate a solution $p_k$ to the Newton equations

                  $J(x_k)p_k = -r(x_k)$;

    $x_{k+1} \leftarrow x_k + p_k$;
end (for)
```

Note: If $r(x) = \nabla f(x)$ then this method corresponds to Newton's method for minimizing f(x)

Potential shortcomings:

- When starting point is far from a solution, the algorithm can behave weirdly
- When $J(x_k)$ is singular, the Newton step may not be defined
- First-derivative information (i.e. J) may be difficult to obtain
- It may be too expensive to find and calculate the Newton step p_k exactly when n is large
- The root x* in question may be degenerate, that is, J(x*) may be singular.

3.1.2 Merit Functions to improve Newton's method

The most widely used merit function is the sum of squares:

$$f(x) = \frac{1}{2}||r(x)||^2 = \frac{1}{2}\sum_{i=1}^{n} r_i^2(x) \tag{21}$$

Algorithms with global convergence properties can be obtained by applying line-search approach to the sum-of-squares merit function. When it is well defined, we have that the Newton step

$$J(x_k)p_k = -r(x_k) \tag{22}$$

is a descent direction for $f(\cdot)$ whenever $r_k \neq 0$ since

$$p_k^T \nabla f(x_k) = -p_k^T J_k^T r_k = -||r_k||^2 < 0 \tag{23}$$

3.2 Constrained Nonlinear Optimization

Simple Elimination using Linear Constraints

We want to minimize a nonlinear function subject to linear equality constraints:

$$\min f(x) \quad \text{subject to } Ax = b \tag{24}$$

where A is an $m \times n$ matrix with $m \leq n$
Jeg forstår ikke :(((se pdf-side 449 og 452

Equality-constrained NLPs — Newton

Problem formulation:

$$\min_{x \in \mathbb{R}^n} f(x) \quad \text{subject to} \quad c(x) = 0 \tag{25}$$

The Lagrangian and the KKT conditions are given as:

$$\mathcal{L}(x, \lambda) = f(x) - \lambda^T c(x) \tag{26a}$$

$$F(x, \lambda) = \begin{pmatrix} \nabla_x \mathcal{L}(x, \lambda) \\ c(x) \end{pmatrix} = 0 \tag{26b}$$

To solve: use Newton's method for nonlinear equations on KKT conditions:

$$\begin{pmatrix} x_{k+1} \\ \lambda_{k+1} \end{pmatrix} = \begin{pmatrix} x_k \\ \lambda_k \end{pmatrix} + \begin{pmatrix} p_k \\ p_{\lambda_k} \end{pmatrix} \quad \text{where} \tag{27}$$

where

$$\underbrace{\begin{pmatrix} \nabla^2_{xx} \mathcal{L}(x_k, \lambda_k) & -A^\top(x_k) \\ A(x_k) & 0 \end{pmatrix}}_{\text{Jacobian of } F(x,\lambda) \text{ at } (x_k,\lambda_k)} \begin{pmatrix} p_k \\ p_{\lambda_k} \end{pmatrix} = \underbrace{\begin{pmatrix} -\nabla f(x_k) + A^\top(x_k)\lambda_k \\ -c(x_k) \end{pmatrix}}_{-F(x_k,\lambda_k)} \tag{28}$$

$$A(x)^T = (\nabla c_1(x), ..., \nabla c_m(x)) \tag{29}$$

Equality-constrained NLPs — QP

Problem formulation is as in (25), now consider the approximation

$$\min_{p\in\mathbb{R}^n} f\left(x_k\right)+\nabla f\left(x_k\right)^{\top} p+\frac{1}{2} p^{\top} \nabla_{xx}^2 \mathcal{L}\left(x_k, \lambda_k\right) p \text{ subject to } c\left(x_k\right)+A\left(x_k\right)^{\top} p=0 \tag{30}$$

with KKT conditions

$$\left(\begin{array}{cc} \nabla_{xx}^2 \mathcal{L}\left(x_k, \lambda_k\right) & -A^{\top}\left(x_k\right) \\ A\left(x_k\right) & 0 \end{array}\right)\left(\begin{array}{c} p_k \\ l_k \end{array}\right)=\left(\begin{array}{c} -\nabla f\left(x_k\right) \\ -c\left(x_k\right) \end{array}\right) \tag{31}$$

Assumptions:

- $A(x_k)$ full row rank (LICQ)
- $d^T\nabla_{xx}^2\mathcal{L}(x_k,\lambda_k)d > 0$ for all $d \neq 0$ s.t. $A(x_k)d = 0$. i.e. *"positive definite on tangent space of constraints"*

The l_1 merit function

= The 1-norm measurement of the violation of the constraints. (24) is:

$$\phi_1(x;\mu) = f(x) + \mu \sum_{i\in\mathcal{E}} |c_i(x)| + \mu \sum_{i\in\mathcal{I}} [c_i(x)]^- \tag{32}$$

where μ is the penalty parameter.

$$\mu^* = \max\{|\lambda_i^*|, i \in \mathcal{E} \cup \mathcal{I}\} \tag{33}$$

Note: ϕ_1 is not differentiable!

$$D\left(\phi_1\left(x_k;\mu\right);p_k\right) \leq -p_k^T \nabla_{xx}^2 \mathcal{L}_k p_k - \left(\mu - \|\lambda_{k+1}\|_\infty\right) \|c_k\|_1 \tag{34}$$

p_k is a descent direction for the merit function if the Hessian of the Lagrangian is positive definite and if μ is large enough.

Exact Merit Function: A merit function $\phi(x;\mu)$ is exact if there is a positive scalar μ^* such that for any $\mu > \mu^*$, any local solution of the nonlinear programming problem (24) is a local minimizer of $\phi(x;\mu)$

On choosing μ:
Consider the effect of the step on a piecewise quadratic model of ϕ_1:

$$q_\mu(p) = f_k + \nabla f_k^T p + \frac{\sigma}{2} p^T \nabla_{xx}^2 \mathcal{L}_k p + \mu m(p) \tag{35}$$

where

$$m(p) = \|c_k + A_k p\|_1 , \tag{36}$$

If μ satisfies (37), it is left unchanged. Otherwise, it is increased to satisfy the equation with some margin.

$$\mu \geq \frac{\nabla f_k^T p_k + (\sigma/2) p_k^T \nabla_{xx}^2 \mathcal{L}_k p_k}{(1-\rho)\|c_k\|_1} \tag{37}$$

Where $\rho \in (0,1)$ and the constant σ is used to handle cases where $\nabla_{xx}^2 \mathcal{L}_k$ is not positive:

$$\sigma = \begin{cases} 1 & \text{if } p_k^T \nabla_{xx}^2 \mathcal{L}_k p_k > 0 \\ 0 & \text{otherwise} \end{cases} \tag{38}$$

The Maratos Effect

Some algorithms based on merit functions may fail to converge rapidly because they reject steps that make good progress toward a solution. This can slow optimization methods by interfering with good steps away from the solution and by preventing superlinear convergence. Strategies for avoiding the Maratos effect:

- Use a merit function that does not suffer form the Maratos effect, for example *Fletcher's augmented Lagrangian*:

$$\phi_F(x;\mu) = f(x) - \lambda(x)^T c(x) + \frac{1}{2}\mu \sum_{i \in \mathcal{E}} c_i(x)^2 \tag{39}$$

- Use second-order correction in which we add to p_k a step $\hat{p}_k$, which is computed at $c(x_k + p_k)$ and which decreases the constraint violation
- Allow the merit function ϕ to increase on certain iterations; i.e. use nonmonotone strategy

3.3 Sequential Quadratic Programming

i.e. Active-Set Methods for Nonlinear Programming: At each iterate, we model the nonlinear problem as a quadratic programming problem and define the search direction to be the solution of this subproblem.

3.3.1 Local SQP method

Assumptions in the local SQP method

- The constraint Jacobian A(x) has full row rank
- The matrix $\nabla^2_{xx}\mathcal{L}(x,\lambda)$ is positive definite on the tangent space of the constraints, that is, $d^T\nabla^2_{xx}\mathcal{L}(x,\lambda)d > 0$ for all $d \neq 0$ such that $A(x)d = 0$

Equality-Constrained QP approach (EQP)

$$\min f(x) \quad \text{subject to } c(x) = 0 \tag{40}$$

At the iterate (x_k, λ_k) we model (40) using the quadratic program:

$$\min_p \quad f_k + \nabla f_k^T p + \frac{1}{2} p^T \nabla^2_{xx}\mathcal{L}_k p \tag{41a}$$

$$\text{subject to} \quad A_k p + c_k = 0 \tag{41b}$$

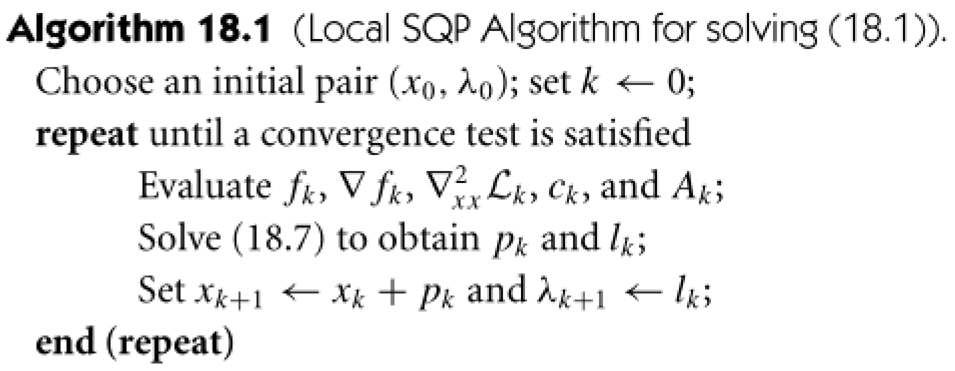
If the assumptions hold, the problem has an unique solution (p_k, l_k) that satisfies

$$\nabla^2_{xx}\mathcal{L}_k p_k + \nabla f_k - A_k = 0 \tag{42a}$$

$$A_k p_k + c_k = 0 \tag{42b}$$

p_k and l_k are solution of the Newton equations:

$$\begin{bmatrix} \nabla^2_{xx}\mathcal{L} & -A_k^T \\ A_k & 0 \end{bmatrix} \begin{bmatrix} p_k \\ p_\lambda \end{bmatrix} = \begin{bmatrix} -\nabla f_k + A_k^T \lambda_k \\ -c_k \end{bmatrix} \tag{43}$$



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

```
Choose an initial pair $(x_0, \lambda_0)$; set $k \leftarrow 0$;
repeat until a convergence test is satisfied
        Evaluate $f_k, \nabla f_k, \nabla^2_{xx}\mathcal{L}_k, c_k$, 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)
```

Figure 3: **NOTE:** (18.1)=(40) and (18.7)=(41)

Inequality-constrained QP approach (IQP)

$$
\begin{aligned}
&\min f(x) && \text{(44a)}\\
&\text{subject to } c_i(x) = 0, \quad i \in \mathcal{E} && \text{(44b)}\\
&c_i(x) \geq 0, i \in \mathcal{I} && \text{(44c)}
\end{aligned}
$$

We linearize both the inequality and equality constraints:

$$
\begin{aligned}
\min_p \quad & f_k + \nabla f_k^T p + \frac{1}{2} p^T \nabla^2_{xx}\mathcal{L}_k p && \text{(45a)}\\
\text{subject to} \quad & \nabla c_i(x_k)^T p + c_i(x_k) = 0, \quad i \in \mathcal{E} && \text{(45b)}\\
& \nabla c_i(x_k)^T p + c_i(x_k) \geq 0, \quad i \in \mathcal{I} && \text{(45c)}
\end{aligned}
$$

Use an algorithm for quadratic programming to solve this problem. The new iterate is given by $(x_k + p_k, \lambda k + 1)$ where p_k is the solution, and λ_{k+1} is the Lagrange multiplier of (45)

The set of active constraints $\mathcal{A}_k$ at the solution of (45) is our guess of the active set at the solution to the nonlinear program. If SQP is able to correctly identify $\mathcal{A}_k$, then it will act like a Newton method for equality-constrained optimization and rapidly converge.

KKT conditions at local solution Suppose that x^* is a local solution of (44) at which the KKT conditions are satisfied for some λ^* . Suppose, too, that the linear independence constraint qualification (LICQ), the strict complementarity condition, and the second-order sufficient conditions hold at (x^*, λ^*). Then if (x_k, λ_k) is sufficiently close to (x^*, λ^*), there is a local solution of the subproblem (45) whose active set $\mathcal{A}_k$ is the same as the active set $\mathcal{A}(x^*)$ of the nonlinear program (44) at x^*.

Inconsistent Linearizations in SQP

(45) might create an infeasible subproblem. To overcome this, we can reformulate (44) as the l_1 penalty problem:

$$\min_{x,v,w,t} f(x) + \mu \sum_{i \in \mathcal{E}} (v_i + w_i) + \mu \sum_{i \in \mathcal{I}} t_i \tag{46a}$$

$$\text{subject to} \quad c_i(x) = v_i - w_i, \quad i \in \mathcal{E}, \tag{46b}$$

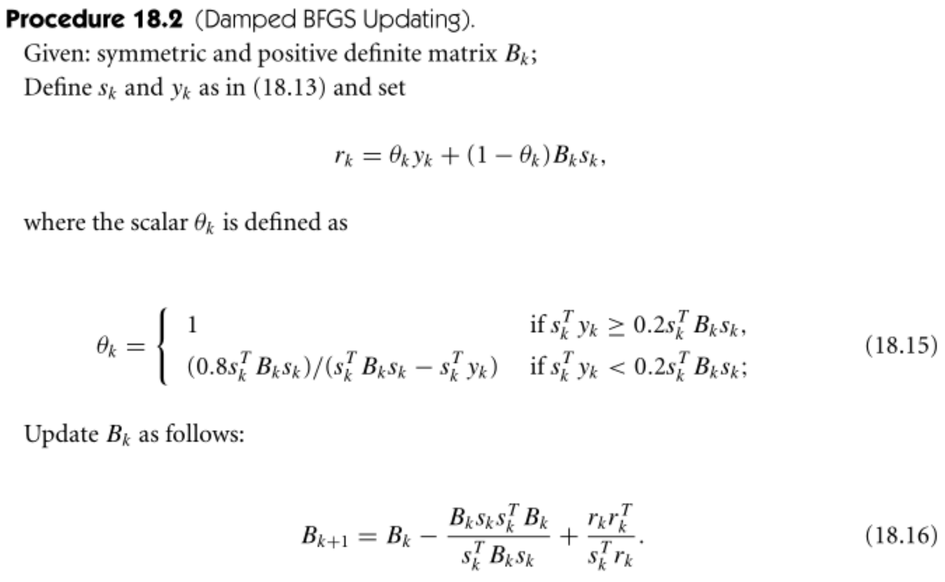
$$c_i(x) \geq -t_i, i \in \mathcal{I}, \tag{46c}$$

$$v, w, t \geq 0 \tag{46d}$$

where $\mu > 0$ is a penalty parameter

3.3.2 Full Quasi-Newton Approximations

Idk... pdf s 557



Procedure 18.2 (Damped BFGS Updating).

Given: symmetric and positive definite matrix B_k;

Define s_k and y_k as in (18.13) and set

$$r_k = \theta_k y_k + (1 - \theta_k) B_k s_k,$$

where the scalar θ_k is defined as

$$\theta_k = \begin{cases} 1 & \text{if } s_k^T y_k \geq 0.2 s_k^T B_k s_k, \\ (0.8 s_k^T B_k s_k)/(s_k^T B_k s_k - s_k^T y_k) & \text{if } s_k^T y_k < 0.2 s_k^T B_k s_k; \end{cases} \tag{18.15}$$

Update B_k as follows:

$$B_{k+1} = B_k - \frac{B_k s_k s_k^T B_k}{s_k^T B_k s_k} + \frac{r_k r_k^T}{s_k^T r_k}. \tag{18.16}$$

Figure 4: Damped BFGS Updating

3.3.3 Merit functions in SQP methods

Merit functions in SQP methods

SQP methods often use a merit function to decide whether a trial step should be accepted. This is because the optimal solution might lie outside the feasible area (i.e. x^* would violate the constraints if we merely looked for a descent direction, Figure 5). **The merit function measures progress both in terms of objective function and in terms of constraints.** In line search method, the merit function controls the step size. For step computations and evaluation of a merit function, inequality constraints are often converted to the form:

$$\bar{c}(x, s) = c(x) - s = 0, \tag{47}$$

where $s \geq 0$ is a vector of slacks (typically not monitored by merit function). In a line search method, a step $\alpha_k p_k$ will be accepted if the sufficient decrease condition holds:

$$\phi_1(x_k + \alpha_k p_k; \mu_k) \leq \phi_1(x_k, \mu_k) + \eta \alpha_k D(\phi_1(x_k; \mu); p_k), \quad \eta \in (0, 1) \tag{48}$$

where $D(\phi_1(x_k; \mu); p_k)$ is the directional derivative of ϕ_1 in the direction p_k

Directional derivative theorem: Let p_k and λ_{k+1} be generated by the SQP iteration (43). Then the directional derivative of ϕ_1 in the direction p_k satisfies

$$D(\phi_1(x_k; \mu); p_k) = \nabla f_k^T p_k - \mu ||c_k||_1 \tag{49}$$

Moreover, we have that

$$D(\phi_1(x_k; \mu); p_k) \leq -p_k^T \nabla_{xx}^2 \mathcal{L}_k p_k - (\mu - ||\lambda_{k+1}||_\infty) ||c_k||_1 \tag{50}$$

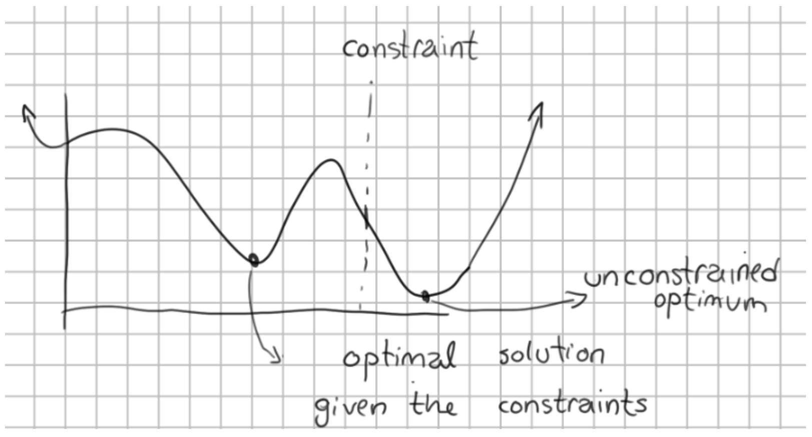
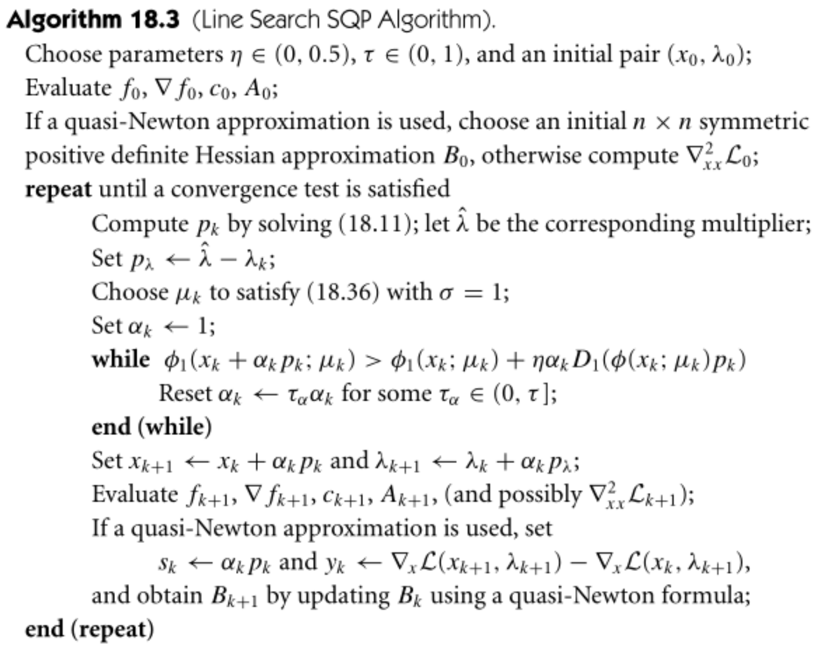


Figure 5: The unconstrained optimum might lie outside of the feasible region, so we need to look for the optimum inside the feasible region by using merit functions.

3.3.4 A Practical Line Search SQP method

Assume the program is convex so we can solve it with the active-set method for quadratic programming:



Algorithm 18.3 (Line Search SQP Algorithm).

```
Choose parameters $\eta \in (0, 0.5)$, $\tau \in (0, 1)$, and an initial pair $(x_0, \lambda_0)$;
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 $\mu_k$ 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)
```

Figure 6: **IMPORTANT:** (18.11)=(44), (18.36)=(51)

$$\mu \geq \frac{\nabla f_k^T p_k + (\sigma/2) p_k^T \nabla^2_{xx}\mathcal{L}_k p_k}{(1-\rho)||c_k||_1} \tag{51}$$

where $\rho \in (0,1)$ and $\sigma = \begin{cases} 1 & \text{if } p_k^T \nabla^2_{xx}\mathcal{L}_k p_k > 0 \\ 0 & \text{otherwise} \end{cases}$

What if $\nabla^2_{xx}\mathcal{L}$ is not positive definite everywhere?

As a result of $\nabla^2_{xx}\mathcal{L}$ not being positive definite, the KKT matrix in calculation of p_k in the SQP algorithm 18.3 is singular.

There are some ways to address this issue:

1. Using quasi-Newton method
2. Adding a multiple of identity matrix
3. Modified Cholesky factorization

Note on nonconvex problems

There is no general way to solve non-convex optimization problems globally, so we cannot guarantee that we will find a global solution using SQP. A simple way to improve our chances is to simply run the SQP algorithm with several different initializations. This will likely lead to different local solutions, which we can compare and use the best one.

4 Linear Constrained Optimization

Active Set

The active set $\mathcal{A}(x)$ at any feasible x consists of the equality constraint indices from $\mathcal{E}$ together with the indices of the inequality constraints i for which $c_i(x) = 0$; that is:

$$\mathcal{A}(x) = \mathcal{E} \cup \{i \in \mathcal{I} | c_i(x) = 0\} \tag{52}$$

At a feasible point x, the inequality constraint $i \in \mathcal{I}$ is said to be *active* if $c_i(x) = 0$ and *inactive* if the strict inequality $c_i(x) > 0$ is satisfied.

The feasible set exists (or is non-empty) as long as no constraints are inconsistent

First-order feasible descent direction

Given a feasible point x and the active constraint set $\mathcal{A}(x)$, the set of linearized feasible directions $\mathcal{F}(x)$ is:

$$\mathcal{F}(x) = \left\{ \begin{array}{ll} d^T \nabla c_i(x) = 0, & \text{for all } i \in \mathcal{E}, \\ d^T \nabla c_i(x) \geq 0, & \text{for all } i \in \mathcal{A}(x) \cap \mathcal{I} \end{array} \right\} \tag{53}$$

Linear Independence Constraint Qualification (LICQ)

Given the point x and the active set $\mathcal{A}(x)$, we say that the linear independence constraint qualification (LICQ) holds if the set of active constraint gradients $\{\nabla c_i(x), i \in \mathcal{A}(x)\}$ is linearly independent.

In general, if LICQ holds, then none of the active constraint gradients can be zero.

First-Order Necessary Conditions (KKT Conditions)

Suppose that x* is a local solution, and that the functions f and c_i are continuously differentiable and that the LICQ holds at x*. Then there is a Lagrange multiplier vector λ^* with components $\lambda_i^*, i \in \mathcal{E} \cup \mathcal{I}$ such that the following conditions are satisfied at (x^*, λ^*):

$$\begin{aligned} \nabla_x \mathcal{L}(x^*, \lambda^*) &= 0, && \quad (54a) \\ c_i(x^*) &= 0, \quad \text{for all } i \in \mathcal{E}, && \quad (54b) \\ c_i(x^*) &\geq 0, \quad \text{for all } i \in \mathcal{I}, && \quad (54c) \\ \lambda_i^* &\geq 0, \quad \text{for all } i \in \mathcal{I}, && \quad (54d) \\ \lambda_i^* c_i(x^*) &= 0, \quad \text{for all } i \in \mathcal{E} \cup \mathcal{I} && \quad (54e) \end{aligned}$$

The first condition can also be written out as:

$$\nabla_x \mathcal{L}(x^*, \lambda^*) = \nabla f(x^*) - \sum_{i \in \mathcal{E} \cup \mathcal{I}} \lambda_i^* \nabla c_i(x^*) \tag{55}$$

Explanation of conditions: (54a) = Stationarity. (54b) and (54c) = Primal feasibility. (54d) = dual feasibility. (54e) = Complementarity condition/Complementarity slackness.

The complementarity condition (54e) imply that either constraint i is active, or $\lambda_i^* = 0$ or possibly both.

Strict Complementarity:
Given a local solution x* and a vector λ^* satisfying (54), we say that the strict complementarity condition holds if exactly one of λ_i^* and $c_i(x^*)$ is zero for each index $i \in \mathcal{I}$. In other words, we have that $\lambda_i^* > 0$ for each $i \in \mathcal{I} \cap \mathcal{A}(x^*)$.

Strict complementarity is desirable because it can make convergence faster, since the determination of the active set is easier when this condition is satisfied.
If strict complementarity holds, (54a) can be written as:

$$0 = \nabla_x \mathcal{L}(x^*, \lambda^*) = \nabla f(x^*) - \sum_{i \in \mathcal{A}(x^*)} \lambda_i^* \nabla c_i(x^*) \tag{56}$$

Lagrange Multipliers and Sensitivity

λ_i^* indicates how hard f is "pushing" or "pulling" the solution x* against the particluar constraint c_i.

Theorem: Let x* be a solution to the optimization problem and suppose that the KKT conditions (54) are satisfied. We say that an inequality constraint c_i is *strongly active* or *binding* if $i \in \mathcal{A}(x^*)$ and $\lambda_i^* > 0$ for some Lagrange multiplier λ^* satisfying the KKT conditions. We say that c_i is *weakly active* if $i \in \mathcal{A}(x^*)$ and $\lambda_i^* = 0$ for all λ^* satisfying the KKT conditions.

Second-Order Sufficient Conditions

Suppose that for some feasible point $x^* \in \mathcal{R}^n$ there is a Lagrange multiplier vector λ^* such that the KKT conditions (54) are satisfied. Suppose also that

$$w^T \nabla^2_{xx} \mathcal{L}(x^*, \lambda^*) w \geq 0, \quad \text{for all } w \in \mathcal{C}(x^*, \lambda^*) \tag{57}$$

Then x* is a strict local solution.

A Fundamental Necessary Condition

If x* is a local solution, then we have

$$\nabla f(x^*)^T d \geq 0, \quad \text{for all } d \in T_\Omega(x^*) \tag{58}$$

Here, $T_\Omega(x^*)$ is the tangent cone: the set of all tangents to Ω at x*.

Second-Order Necessary Conditions

Suppose that x* is a local solution and that the LICQ condition is satisfied. Let λ^* be the Lagrange multiplier vector for which the KKT conditions (54) are satisfied. Then

$$w^T \nabla^2_{xx} \mathcal{L}(x^*, \lambda^*) w \geq 0, \quad \text{for all } w \in \mathcal{C}(x^*, \lambda^*) \tag{59}$$

Here, $\mathcal{C}(x^*, \lambda^*)$ denotes the critical directions of the solution, which is the directions in which the Hessian of the Lagrangian has nonnegative curvature.

4.1 Duality

Dual problem can often be easier to solve. It can be used to obtain a lower bound on the optimal value of the objective of the primal.

Assume only m inequality constraints:

$$\min_{x \in \mathcal{R}^n} f(x) \quad \text{subject to } c(x) \geq 0 \tag{60}$$

where $c(x) = (c_1(x), c_2(x), ..., c_m(x))^T$

The **dual objective function** is defined as:

$$q(\lambda) = \inf_x \mathcal{L}(x, \lambda) \tag{61}$$

where $\mathcal{L}(x, \lambda) = f(x) - \lambda^T c(x)$. There's no completely general approach to finding $\inf_x \mathcal{L}(x, \lambda)$, but in those cases where $\mathcal{L}$ is differentiable and convex with respect to x, any x with $\nabla_x \mathcal{L}(x, \lambda, v) = 0$ will attain the inf.

The **dual problem** is defined as:

$$\max_{\lambda \in \mathcal{R}^n} q(\lambda) \quad \text{subject to } \lambda \geq 0 \tag{62}$$

Dual KKT Conditions

$$\nabla f(\overline{x}) - \nabla c(\overline{x})\overline{\lambda} = 0, \tag{63a}$$
$$c(\overline{x}) \geq 0, \tag{63b}$$
$$\overline{\lambda} \geq 0, \tag{63c}$$
$$\overline{\lambda}_i c_i(\overline{x}) = 0, \quad i = 1, 2, ..., m \tag{63d}$$

$\nabla c(x)$ is the $n \times m$ matrix defined by $\nabla c(x) = [\nabla c_1(x), \nabla c_2(x), ..., \nabla c_m(x)]$. $\overline{x}$ is a solution to the primal. $\overline{\lambda}$ is the solution to the dual problem.

Properties of duality

- The function q is concave and its domain is convex
- **Weak Duality:** For any $\overline{x}$ feasible for the primal and any $\overline{\lambda} \geq 0$, we have $q(\overline{\lambda}) \leq f(\overline{x})$
- **The Lagrange multipliers for the primal are the solutions to the dual problem under certain conditions** (this is essentially due to Wolfe): Suppose $\overline{x}$ is a solution to the primal and suppose f and $-c_i, i = 1, 2, ..., m$ are convex functions on $\mathcal{R}^n$ that are differentiable at $\overline{x}$. Then any $\overline{\lambda}$ for which $(\overline{x}, \overline{\lambda})$ satisfies the KKT conditions of the dual (63a) is a solution of the dual problem.

5 Linear Programming

When the objective functions and all constraints are linear functions of x, the problem is a linear programming problem.

Standard form

$$\min c^T x, \quad \text{subject to } Ax = b, x \geq 0 \tag{64}$$

where c and x are vectors in $\mathbb{R}^n$, b is a vector in $\mathbb{R}^m$ and A is an $m \times n$ matrix.

Slack variables

Linear programs (e.g. simplex method) can only take problems on the form (66). To formulate the problem correctly we can use slack variables.

The original problem (without bounds on x):

$$\min c^T x, \quad \text{subject to } Ax \leq b \tag{65}$$

Introduce slack variables z and make them inequality constraints:

$$\min c^T x, \quad \text{subject to } Ax + z = b, z \geq 0 \tag{66}$$

To get it on standard form (all variables must also be nonnegative), introduce $x = x^+ - x^-$, where $x^+ = \max(x, 0) \geq 0$ and $x^- = \max(-x, 0) \geq 0$:

$$\min \begin{bmatrix} c \\ -c \\ 0 \end{bmatrix}^T \begin{bmatrix} x^+ \\ x^- \\ z \end{bmatrix}, \text{ s.t. } \begin{bmatrix} A & -A & I \end{bmatrix} \begin{bmatrix} x^+ \\ x^- \\ z \end{bmatrix} = b, \begin{bmatrix} x^+ \\ x^- \\ z \end{bmatrix} \geq 0$$

Optimality Conditions

Only the first-order KKT conditions are needed. Since the problem is convex, we know that these conditions are sufficient for a global minimum.
Lagrange multipliers are split into $\lambda \in \mathbb{R}^m$, the multiplier vector for equality constraints $Ax = b$ and $s \in \mathbb{R}^n$, the multiplier vector for the bound constraints $x \geq 0$.
The Lagrangian is given by:

$$\mathcal{L}(x, \lambda, s) = c^T x - \lambda^T (Ax - b) - s^T x \tag{67}$$

The following first-order necessary conditions for x* to be a solution:

$$\begin{aligned} A^T \lambda + s &= c, && (68a)\\ Ax &= b, && (68b)\\ x &\geq 0, && (68c)\\ s &\geq 0, && (68d)\\ x_i s_i &= 0, \quad i = 1, 2, ..., n && (68e) \end{aligned}$$

Dual Problem in Linear Programming

$$\max_{\lambda \in \mathbb{R}^m} b^T \lambda \quad \text{subject to} \quad A^T \lambda \leq c \tag{69}$$

or alternatively using slack variables::

$$\max_{\lambda \in \mathbb{R}^m} b^T \lambda \quad \text{subject to} \quad A^T \lambda + s = c \tag{70}$$

Lagrangian:

$$\mathcal{L}(x, \lambda) = f(x) - \sum_{i \in \mathcal{E} \cup \mathcal{I}} \lambda_i c_i(x) \tag{71}$$

KKT conditions:

$$\begin{aligned} \nabla_x \mathcal{L}(x^*, \lambda^*) &= 0, && && (72a)\\ c_i(x^*) &= 0, && \text{for all } i \in \mathcal{E}, && (72b)\\ c_i(x^*) &\geq 0, && \text{for all } i \in \mathcal{I}, && (72c)\\ \lambda_i^* &\geq 0, && \text{for all } i \in \mathcal{I}, && (72d)\\ \lambda_i^* c_i(x^*) &= 0, && \text{for all } i \in \mathcal{E} \cup \mathcal{I} && (72e) \end{aligned}$$

TODO, side 381. Skjønner ikke dette.

Weak Duality in Linear Programming

Weak duality can be useful for computing bounds.

$$c^T \bar{x} \geq c^T x^* = b^T \lambda^* \geq b^T \bar{\lambda} \tag{73}$$

"For feasible x that will always be $\geq$ the minimum $=$ the maximum of the dual, which is always $\geq$ what we get if we put any feasible λ in the dual."

Duality gap: $c^T \bar{x} - b^T \bar{\lambda}$

Strong Duality in Linear Programming

- If either the dual or the primal has a (finite) solution, then so does the other, and the objective values are equal ($c^T x^* = b^T \lambda^*$)
- If either of the primal or dual is unbounded, then the other problem is infeasible

Sensitivity in Linear Programming

Given LP in the standard form (66), we assume an optimal solution x^* with Lagrangian multipliers λ^*. A small perturbation in b_i : $\tilde{b}_i = b_i + \epsilon$ gives a new solution that fulfills $c^T x^*_{new} = c^T x^* \pm \epsilon \lambda^*_i$

5.1 Basic Feasible Points

Must assume the matrix A has full row rank. This can be achieved by adding slack, surplus and artificial variables.

Definition of a basic feasible point

A vector x is a basic feasible point if it is feasible and if there exists a subset $\mathcal{B}$ of the index set $\{1, 2, ..., n\}$ such that

- $\mathcal{B}$ contains exactly m indices
- $i \notin \mathcal{B} \Rightarrow x_i = 0$ (that is, the bound $x_i \geq 0$ can be inactive only if $i \in \mathcal{B}$)
- The $m \times m$ matrix B defined by $B = [A_i]_{i \in \mathcal{B}}$ is nonsingular, where A_i is the ith column of A.

The Fundamental Theorem of Linear Programming

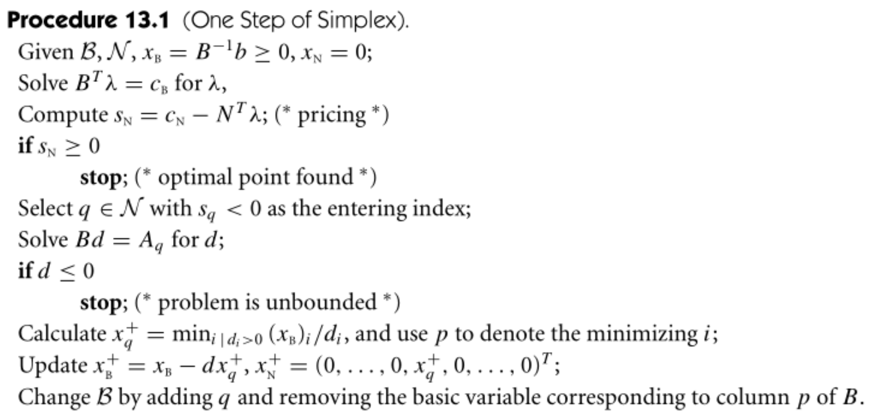
- If (66) has a nonempty feasible region, then there is at least one basic feasible point
- If (66) has solutions, then at least one solution is a basic optimal point
- If (66) is feasible and bounded, then it has an optimal solution

Degeneracy

A basis $\mathcal{B}$ is said to be degenerate if $x_i = 0$ for some $i \in \mathcal{B}$, where x is the basic feasible solution corresponding to $\mathcal{B}$. A linear program (66) is said to be degenerate if it has at least one degenerate basis.

5.2 The Simplex Method

Essentially, we move from vertex to vertex in the feasible region. This is an active set method. In each step, we move from one vertex to an adjacent one for which the basis $\mathcal{B}$ differs in exactly one component.



Procedure 13.1 (One Step of Simplex).

```
Given $\mathcal{B}, \mathcal{N}, x_B = B^{-1}b \geq 0, x_N = 0$;
Solve $B^T\lambda = c_B$ for $\lambda$,
Compute $s_N = c_N - N^T\lambda$; (* pricing *)
if $s_N \geq 0$
    stop; (* optimal point found *)
Select $q \in \mathcal{N}$ with $s_q < 0$ as the entering index;
Solve $Bd = A_q$ for $d$;
if $d \leq 0$
    stop; (* problem is unbounded *)
Calculate $x_q^+ = \min_{i \mid d_i > 0} (x_B)_i / d_i$, and use $p$ to denote the minimizing $i$;
Update $x_B^+ = x_B - dx_q^+$, $x_N^+ = (0, \ldots, 0, x_q^+, 0, \ldots, 0)^T$;
Change $\mathcal{B}$ by adding $q$ and removing the basic variable corresponding to column $p$ of $B$.
```

Simplex Solution Condition

Provided that the linear program (66) is nondegenerate and bounded, the simplex method terminates at a basic optimal point.

6 Quadratic Programming

General quadratic program

$$\min_x \quad q(x) = \frac{1}{2}x^T G x + x^T c \tag{74a}$$

$$\text{subject to} \quad a_i^T x = b_i, \quad i \in \mathcal{E}, \tag{74b}$$

$$a_i^T x \geq b_i, \quad i \in \mathcal{I} \tag{74c}$$

where G is a symmetric $n \times n$ matrix, $\mathcal{E}$ and $\mathcal{I}$ are finite sets of indices, c, x and $\{a_i\}, i \in \mathcal{E} \cup \mathcal{I}$ are vectors in $\mathbb{R}^n$

QPs and convexity

Convex QP: If the Hessian matrix G is positive semidefinite. Then, the problem is often similar in difficulty to a linear program.

Strictly Convex QPs: G is positive definite.

Nonconvex QP: G is an indefinite matrix.

6.1 Equality-constrained Quadratic Programs

EQP formulation

$$\min_x \quad q(x) = \frac{1}{2}x^T G x + x^T c \tag{75a}$$

$$\text{subject to} \quad Ax = b, \quad i \in \mathcal{E} \tag{75b}$$

Where A is the $m \times n$ Jacobian of constraints with $m \leq n$ whose rows are $a_i^T, i \in \mathcal{E}$ and b is the vector in $\mathbb{R}^m$ whose components are $b_i, i \in \mathcal{E}$

The KKT matrix

The first-order necessary conditions for x* to be a solution to (75) state that there is a vector λ^* such that the following system of equations is satisfied:

$$\begin{bmatrix} G & -A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} x^* \\ \lambda^* \end{bmatrix} = \begin{bmatrix} -c \\ b \end{bmatrix} \tag{76}$$

A form that is more useful for computation:

$$\begin{bmatrix} G & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} -p \\ \lambda^* \end{bmatrix} = \begin{bmatrix} g \\ h \end{bmatrix} \tag{77}$$

where λ^* is the vector of Lagrange multipliers, $h = Ax - b$, $g = c + Gx$, $p = x^* - x$

Global solution: Let A have full row rank and assume that the reduced-Hessian matrix $Z^T GZ$ is positive definite. Here, Z denotes the $n \times (n-m)$ matrix whose columns are a basis for the null space of A. Then the vector x* satisfying (76) is the unique global solution to (75)

How to solve KKT system (KKT matrix indefinite, but symmetric):

- Full space: Symmetric indefinite (LDL) factorization: $P^\intercal KP = LBL^\intercal$
- Reduced space: Use Ax=b to eliminate m variables
  - Requires computation of Z, which can be costly
  - Reduced space method faster than full-space when many constraints (if $m - n << n$)

6.2 Inequality-constrained Quadratic Programs

The lagrangian for this problem is:

$$\mathcal{L}(x, \lambda) = \frac{1}{2} x^T Gx + x^T c - \sum_{i \in \mathcal{I} \cup \mathcal{E}} \lambda_i (a_i^T x - b_i) \tag{78}$$

where the active set consists of the indices of the constraints for which equality holds at x*:

$$\mathcal{A}(x^*) = \{i \in \mathcal{E} \cup \mathcal{I} \quad | \quad a_i^T x^* = b_i\} \tag{79}$$

Note: If the active set is already known, the QP can be solved as an EQP

KKT conditions for inequality-constrained QP

$$Gx^* + c - \sum_{i \in \mathcal{A}(x^*)} \lambda_i^* a_i = 0, \tag{80a}$$

$$a_i^T x^* = b_i, \quad \text{for all } i \in \mathcal{A}(x*), \tag{80b}$$

$$a_i^T x^* \geq b_i, \quad \text{for all } i \in \mathcal{I}\mathcal{A}(x^*), \tag{80c}$$

$$\lambda_i^* \geq 0, \quad \text{for all } i \in \mathcal{I} \cap \mathcal{A}(x^*) \tag{80d}$$

Note: If x* satisfies the conditions for some $\lambda_i^*, i \in \mathcal{A}(x^*)$, and G is positive semidefinite, then x*is a global solution of (74a)

6.3 Active-Set Methods for Convex QPs

Active-set methods for QP differ from the simplex method in that the iterates are not necessarily vertices of the feasible region. A general algorithm is:

1. Make a guess of which constraints are active at the optimal solution
2. Solve the corresponding EQP
3. Check KKT conditions.
   (a) If KKT okay: finished
   (b) If not, update guess of active constraints in a smart way, then start over

6.3.1 Primal active-set methods

Find a step from one iterate to the next by solving a quadratic subproblem in which some inequality constraints and all equality constraints are imposed as equalities. This is subset is the *working set* and is denoted at the kth iterate x_k by $\mathcal{W}_k$.

Notes on the active-set method for convex QPs

- Constraint gradients in $\mathcal{W}_0$ must be linearly independent, and our strategy for modifying $\mathcal{W}$ must ensure that this also holds for all subsequent $\mathcal{W}_k$
- When removing constraints, remove the constraint corresponding to the most negative Lagrange multiplier. Beware that this is susceptible to the scaling of the constraints.

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 $\alpha_k$ 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)
```

7 Dynamic Optimization and Model Predictive Control

Two types of dynamic optimization

Quasi-dynamic optimization:

- Optimizes dynamic system by repetitive optimizations on static model
- Key idea: Dynamic changes can be compensated for by frequent reoptimizing on static model
- Works well for slowly varying systems/systems mostly in steady state
- Usually results in smaller problems and less complex formulations, **BUT** cannot handle system with significant dynamics

Dynamic Optimization

- Optimize on a dynamic model
- Solution is a function of time =¿ all decision variables are functions of time
- Necessary when dynamics plays a major role, e.g. systems with frequent changes in operating conditions

7.1 Dynamic Optimization with Linear Models

Problem formulation

The objective function is given by

$$f(z) = \sum_{t=0}^{N} f_t(x_{t+1}, u_t) = \sum_{t=0}^{N-1} \frac{1}{2} x_{t+1}^T Q_{t+1} x_{t+1} + d_{xt+1}^T x_{t+1} + \frac{1}{2} u_t^T R_t u_t + d_{ut}^T u_t \tag{81}$$

where Q_t and R_t are positive semi-definite.
The problem formulation is then:

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

subject to

$$\begin{aligned}
x_{t+1} &= A_t x_t + B_t u_t, && t = 0, \ldots, N-1 \\
x_0, u_{-1} &= \text{ given} \\
x^{\text{low}} &\leq x_t \leq x^{\text{high}}\ , && t = 1, \ldots, N \\
u^{\text{low}} &\leq u_t \leq u^{\text{high}}\ , && t = 0, \ldots, N-1 \\
-\Delta u^{\text{high}} &\leq \Delta u_t \leq \Delta u^{\text{high}}\ , && t = 0, \ldots, N-1
\end{aligned}$$

where

$$\begin{aligned}
Q_t &\succeq 0 && t = 1, \ldots, N \\
R_t &\succeq 0 && t = 0, \ldots, N-1 \\
\Delta u_t &= u_t - u_{t-1} \\
z^\top &= \left(x_1^\top, \ldots, x_N^\top, u_0^\top \ldots, u_{N-1}^\top\right) \\
n &= N \cdot (n_x + n_u)
\end{aligned}$$

Finite horizon optimal open loop control for MIMO system

Optimization objective

$$f(z) = \sum_{t=0}^{N-1} \frac{1}{2} x_{t+1}^T Q x_{t+1} + \frac{1}{2} u_t^T R u_t \tag{82}$$

where Q_t and R_t are positive semi-definite. N = horizon length. $z = \left(x_1^T, ..., x_N^t, u_0^T, ..., u_{N-1}^T\right)$
Rewrite into matrix form $f(z)\frac{1}{2}z^T G z$ where

$$G = \begin{bmatrix} Q_1 & 0 & \cdots & \cdots & \cdots & 0 \\ 0 & \ddots & \ddots & & & \vdots \\ \vdots & \ddots & Q_N & \ddots & & \vdots \\ \vdots & & \ddots & R_0 & \ddots & \vdots \\ \vdots & & & \ddots & \ddots & 0 \\ 0 & \cdots & \cdots & \cdots & 0 & R_{N-1} \end{bmatrix} \tag{83}$$

The state equations can be written as the equality constraint $A_{eq} z = b_{eq}$ with

$$A_{\text{eq}} = \left[\begin{array}{ccccc|ccccc} I & 0 & \cdots & \cdots & 0 & -B_0 & 0 & \cdots & \cdots & 0 \\ -A_1 & I & \ddots & & \vdots & 0 & \ddots & \ddots & & \vdots \\ 0 & \ddots & \ddots & \ddots & \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 & \vdots & & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & -A_N & I & 0 & \cdots & \cdots & 0 & -B_N \end{array}\right]$$

and

$$b_{\text{eq}} = \begin{bmatrix} A_0 x_0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

7.2 Model Predictive Control

MPC definition

At each sampling instant, solve the 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 is better than the open loop solution because it couples open-loop optimization with feedback control.

MPC uses a moving horizon: the prediction horizon changes from one timestep to the next.

Algorithm 2 State feedback MPC procedure

```
for $t = 0, 1, 2, \ldots$ do
    Get the current state $x_t$.
    Solve a dynamic optimization problem on the prediction horizon from $t$
    to $t + N$ with $x_t$ as the initial condition.
    Apply the first control move $u_t$ from the solution above.
end for
```

Figure 7: State Feedback MPC procedure

Algorithm 3 Output feedback MPC procedure

```
for $t = 0, 1, 2, \ldots$ do
    Compute an estimate of the current state $\hat{x}_t$ based on the measured data
    up until time $t$.
    Solve a dynamic optimization problem on the prediction horizon from $t$
    to $t + N$ with $\hat{x}_t$ as the initial condition.
    Apply the first control move $u_t$ from the solution above.
end for
```

Figure 8: Output feedback MPC procedure

Notes on MPC performance

- Offers improved performance through increased disturbance rejection and better tracking performance
- MPC performance depends on the performance of the regulatory controllers. Well tuned controllers (e.g. well-tuned PID) gives better benefit of this.
- MPC automatically adjusts the control strategy to push against and explot the constraints that limit output according to the chosen objective function.

7.2.1 Linear MPC

Pros and cons of a linear MPC controller:

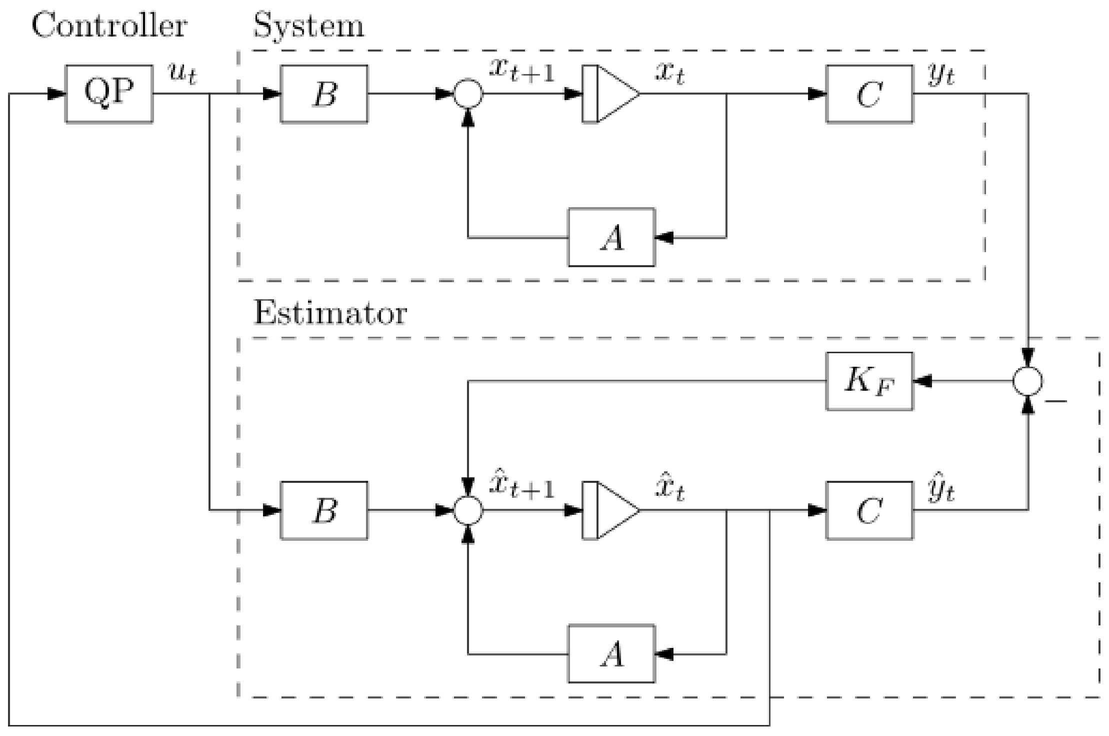


Figure 9: The structure of an output feedback linear MPC

Pros	**Cons**
Ability to include constraints	Increased implementation complexity
Flexibility in defining the cost function. This allows for implementation of fairly complex controllers	More complex stability proofs (if any)
The predictive capabilities; ability to calculate future inputs with respect to a time-varying reference signal or model	Model based (increased modelling effort)
Model Based	
Intuitive	
Multivariable	

Algorithm 4 Linear MPC with state feedback

```
for $t = 0, 1, 2, \ldots$ do
  Get the current state $x_t$.
  Solve the convex QP problem (3.8) on the prediction horizon from $t$ to
  $t + N$ with $x_t$ as the initial condition.
  Apply the first control move $u_t$ from the solution above.
end for
```

Figure 10: Linear MPC with state feedback

Linear MPC equations

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

subject to

$$\begin{aligned} x_{t+1} &= A_t x_t + B_t u_t \\ x_0, u_{-1} &= \text{ given} \\ x^{\text{low}} &\leq x_t \leq x^{\text{high}} \\ u^{\text{low}} &\leq u_t \leq u^{\text{high}} \\ -\Delta u^{\text{high}} &\leq \Delta u_t \leq \Delta u^{\text{high}} \end{aligned}$$

where

$$\begin{aligned} Q_t &\succeq 0 \\ R_t &\succeq 0 \\ R_{\Delta t} &\succeq 0 \end{aligned}$$

Choosing horizon length N

- N must be at least as long as the dominant dynamics (preferably 2-3 times larger than the dominant dynamics)
- Computational time for solving QP must be lower than the control interval

7.2.2 Ensuring feasibility in linear MPC

Why? Because a disturbance might make it so that we end up outside the constraints from x_t to x_{t+1}. To solve the QP, we need to start in the feasible region, so if x_{t+1} has been "knocked out of bounds" by a disturbance, the QP might not have a solution.

Introducing slack variables to ensure feasibility in MPC

Introduce slack variables ϵ and the tuning parameters ρ and $s_1, ..., s_n$, defined such that all their elements are positive

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

subject to

$$\begin{aligned} x_{t+1} &= A_t x_t + B_t u_t \\ x_0, u_{-1} &= \text{ given} \\ x^{\text{low}} - \epsilon &\le x_t \le x^{\text{high}} + \epsilon \\ u^{\text{low}} &\le u_t \le u^{\text{high}} \\ -\Delta u^{\text{high}} &\le \Delta u_t \le \Delta u^{\text{high}} \end{aligned}$$

where

$$\epsilon \in \mathbb{R}^{n_x} \ge 0$$

$$\begin{aligned} &\rho \in \mathbb{R}^{n_x} \ge 0 \\ &S \in \operatorname{diag}\{s_1, \dots, s_{n_x}\}, s_i \ge 0, i = \{1, \dots, n_x\} \end{aligned}$$

7.2.3 Nonlinear MPC

Optimization problem formulation

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

subject to

$$\begin{aligned} x_{t+1} &= g(x_t, u_t) \\ x_0, u_{-1} &= \text{ given} \\ x^{\text{low}} &\le x_t \le x^{\text{high}} \\ u^{\text{low}} &\le u_t \le u^{\text{high}} \\ -\Delta u^{\text{high}} &\le \Delta u_t \le \Delta u^{\text{high}} \end{aligned}$$

where

$$\begin{aligned} Q_t &\succeq 0 \\ R_t &\succeq 0 \\ R_{\Delta t} &\succeq 0 \end{aligned}$$

Note: in the output feedback case, a state stimate is needed, i.e. $\hat{x}_t$

Algorithm 7 Nonlinear MPC with state feedback

```
for $t = 0, 1, 2, \ldots$ do
    Get the current state $x_t$.
    Solve the optimization problem (4.41) on the prediction horizon from $t$
    to $t + N$ with $x_t$ as the initial condition.
    Apply the first control move $u_t$ from the solution above.
end for
```

Figure 11: Nonlinear MPC with state feedback

7.3 Linear Quadratic Control

7.3.1 Finite Horizon LQ control

Finite Horizon LQ control problem formulation

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

subject to

$$\begin{aligned} x_{t+1} &= A_t x_t + B_t u_t \\ x_0 &= \text{ given} \end{aligned}$$

where

$$\begin{aligned} z^\top &= \left(x_1^\top, \ldots, x_N^\top, u_0^\top \ldots, u_{N-1}^\top\right) \\ n &= N \cdot (n_x + n_u) \quad \textit{(n is degrees of freedom)} \end{aligned}$$

Algorithm 6 Output feedback moving horizon LQ

```
for $t = 0, 1, 2, \ldots$ do
    Compute an estimate of the current state $\hat{x}_t$ based on the data up until
    time $t$.
    Compute and apply the control $u_t = -K_t \hat{x}_t$. $K_t$ will be constant for
    all $t$ for an LTI system and an objective function with constant weight
    matrices.
end for
```

Figure 12: Output feedback moving horizon LQ

LQ Control, Batch approach v1 ”Full Space” QP

$$\begin{aligned}
\min_z \sum_{t=0}^{N-1} & \frac{1}{2} x_{t+1}^\top Q x_{t+1} + \frac{1}{2} u_t^\top R u_t \\
\text{s.t.} \quad & x_{t+1} = A x_t + B u_t, \quad t = 0, 1, \dots, N-1 \\
& z = (u_0, x_1, u_1, \dots, u_{N-1}, x_N)^\top
\end{aligned} \tag{84}$$

Formulate with model as equality constraints, all inputs and states as optimization variables:

$$\min_z \frac{1}{2} x^T \begin{pmatrix} R & & & & \\ & Q & & & \\ & & R & & \\ & & & \ddots & \\ & & & & Q \end{pmatrix} z \tag{85a}$$

$$\text{s.t.} \begin{pmatrix} -B & \mathbb{I} & & & & & & & \\ & -A & -B & \mathbb{I} & & & & & \\ & & & -A & -B & \mathbb{I} & & & \\ & & & & \ddots & \ddots & & & \\ & & & & & & -A & -B & \mathbb{I} \end{pmatrix} z = \begin{pmatrix} Ax_0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \tag{85b}$$

$$z = (u_0, x_1, u_1, ..., u_{N-1}, x_N)^T \tag{85c}$$

LQ Control, Batch approach v2 "Reduced Space" QP

Use model to eliminate states as variables (since $x_{t+1} = Ax_t + Bu_t = A(Ax_{t-1} + Bu_{t-1}) + Bu_t$ etc.)

$$\begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{pmatrix} = \begin{pmatrix} A \\ A^2 \\ A^3 \\ \vdots \\ A^N \end{pmatrix} x_0 + \begin{pmatrix} B & & & & \\ AB & B & & & \\ A^2 & AB & B & & \\ \vdots & \vdots & \vdots & \ddots & \\ A^{N-1}B & A^{N-2}B & A^{N-3}B & \dots & B \end{pmatrix} \begin{pmatrix} u_0 \\ u_1 \\ \vdots \\ u_{N-1} \end{pmatrix} \tag{86}$$
$$= S^x x_0 + S^u U$$

Insert into objective function (no constraints):

$$\min_U \frac{1}{2}(S^x x_0 + S^u U)^T \boldsymbol{Q} (S^x x_0 + S^u U) + \frac{1}{2} U^T \boldsymbol{R} U \tag{87}$$

Solution found by setting gradient equal to zero:

$$U = \begin{pmatrix} u_0 \\ u_1 \\ \vdots \\ u_{N-1} \end{pmatrix} = -\big((S^u)^T \boldsymbol{Q} S^u + \boldsymbol{R})^{-1} (S^u)^T \boldsymbol{Q} S^x x_0\big) = -F x_0 \tag{88}$$

LQ Control, Recursive Approach (State feedback)

The solution of the finite horizon LQ problem with $Q_t \succeq 0$ and $R_t \succeq 0$ is given by $u_t = -K_t x_t$ where the feedback gain matrix is derived by

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

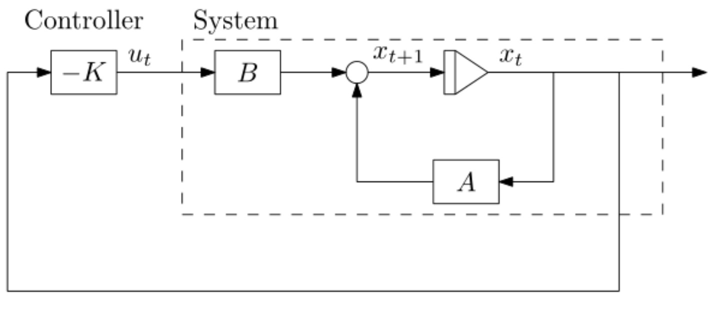


Figure 13: Solution of the LQ control problem, i.e., with state feedback

Notes on solution approaches

- Batch v1, Batch v2 and recursive approach all give same numerical solution. If $Q \succeq 0$ and $R \succ 0$, this solution is unique.
- Batch approaches give open-loop solution, while recursive approach gives closed-loop (feedback) solution $\Rightarrow$ recursive solution is more robust in implementation.
- **Reduced space** has less variables, but normally dense matrices. **Full space** has many variables, but often a lot of sparsity in the matrices (can be explored by a solver)
- It is easy to add constraints as inequalities to batch approaches (as both become convex QPs), but more difficult to add constraints to recursive approach.

7.3.2 Infinite Horizon LQ control

State feedback infinite horizon LQ control

$$\min_z f^\infty(z) = \sum_{t=0}^{\infty} \frac{1}{2} x_{t+1}^\top Q x_{t+1} + \frac{1}{2} u_t^\top R u_t$$

subject to equality constraints

$$\begin{aligned} x_{t+1} &= Ax_t + Bu_t \\ x_0 &= \text{ given} \end{aligned}$$

and

$$\begin{aligned} Q &\succeq 0 \\ R &\succ 0 \end{aligned}$$

where system dimensions are given by

$$\begin{aligned} u_t &\in \mathbb{R}^{n_u} \\ x_t &\in \mathbb{R}^{n_x} \\ z^\top &= \left(u_0^\top, \dots, u_\infty^\top, x_1^\top, \dots, x_\infty^\top\right) \end{aligned}$$

If the system is stabilizable, then there exists a static solution: $P_t \to \bar{P}$ when $t \to \infty$. That is, the recursive Ricatti equation becomes the algebraic Ricatti equation since $P_t = P_{t+1}$ when $N \to \infty$. This solution is equal to the unique positive semidefinite solution of the algebraic Riccati equation.

The solution of the problem is then given by

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

where the feedback gain matrix is derived by

$$K = R^{-1}B^T P(I + BR^{-1}B^T P)^{-1}A \tag{89a}$$

$$P = Q + A^T P(I + BR^{-1}B^T P)^{-1}A \tag{89b}$$

$$P = P^T \succeq 0 \tag{89c}$$

Important:

- This is an infinite dimensional QP, so it cannot be solved using a conventional method since the KKT matrix is infinitely large and not well defined (this is why we use the Riccati equation)
- The objective function must be bounded above, i.e. $f^\infty(z) < \infty$ for some feasible z. This implies that $u_t \to 0$ when $t \to \infty$, otherwise $f^\infty(z) \to \infty$
- (89b) is called the **algebraic Riccati equation**, which is a quadratic equation in the unknown matrix P. Thus, there may exist several (particularly two) solutions to this equation. Only one solution, however, will be positive semi-definite and therefore we specify (89c).

Output feedback infinite horizon LQ control (LQG)

Combine the state feedback infinite horizon LQ control problem formulation with a measurement model $y_t = Cx_t$, the output feedback LQ controller $u_t = -K\hat{x}_t$ and a Kalman filter (92)

$$\begin{aligned} x_{t+1} &= Ax_t + Bu_t \\ y_t &= Cx_t \\ u_t &= -K\hat{x}_t \\ x_0 &= \text{ given} \\ \hat{x}_{t+1} &= A\hat{x}_t + Bu_t + K_F\left(y_t - \hat{y}_t\right) \\ \hat{y}_t &= C\hat{x}_t \\ \hat{x}_0 &= \text{ given} \end{aligned} \tag{90}$$

Or, on a more compact form:

$$\begin{aligned} \xi_{t+1} &= \begin{bmatrix} x_{t+1} \\ \tilde{x}_{t+1} \end{bmatrix} = \begin{bmatrix} A - BK & BK \\ 0 & A - K_FC \end{bmatrix} \xi_t \\ \tilde{x}_t &= x_t - \hat{x}_t \\ \xi_0 &= \text{ given} \end{aligned} \tag{91}$$

The dimension of the augmented state is $\xi_t \in \mathbb{R}^{2n_x}$. $\tilde{x}_t$ refers to the error in the state estimates.

Notes:

- LQG=Linear Quadratic Gaussian Control
- The closed-loop system given by the optimal solution is asymptotically stable if **(1):** (A,B) is stabilizable and **(2):** (A,D) is detectable, where $Q = D^\intercal D$

Stabilizability and Detectability

For the state feedback infinite horizon LQ control problem, let the system (A, B) be stabilizable and (A, D) be detectable. D is defined by $Q = D^TD$. Then the closed loop system given by the optimal solution is asymptotically stable.

Stabilizability = We must be able to influence all unstable modes
Detectability = The objetive function must be sensitive to all the unstable modes

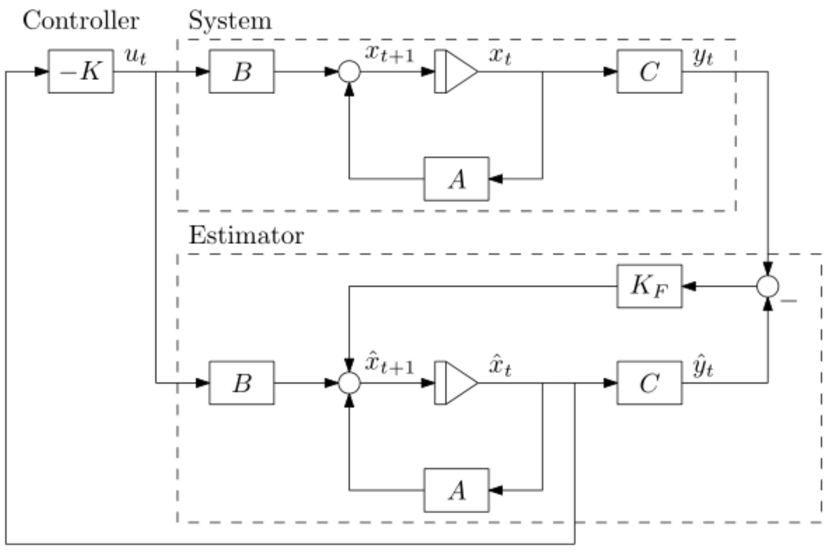


Figure 14: Structure of the LQG controller, i.e. output feedback LQ control

8 Misc

Kalman Filter

$$\begin{aligned} \hat{x}_{t+1} &= A\hat{x}_t + Bu_t + K_F\left(y_t - \hat{y}_t\right) \\ \hat{y}_t &= C\hat{x}_t \\ \hat{x}_0 &= \text{ given} \end{aligned} \tag{92}$$

K_F = Kalman gain, C = measurement matrix, y_t is data measured at each sample. Also, assume linear measurement model:

$$y_t = Cx_t$$

8.1 Matrices

Jacobian definition

$$\mathbf{J} = \left[\begin{array}{ccc} \frac{\partial \mathbf{f}}{\partial x_1} & \cdots & \frac{\partial \mathbf{f}}{\partial x_n} \end{array}\right] = \left[\begin{array}{c} \nabla^{\mathrm{T}} f_1 \\ \vdots \\ \nabla^{\mathrm{T}} f_m \end{array}\right] = \left[\begin{array}{ccc} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \cdots & \frac{\partial f_m}{\partial x_n} \end{array}\right] \tag{93}$$

where $\nabla^T f_i$ is the transpose (row vector) of the gradient of the ith component.

Hessian definition

$$\mathbf{H}(x) := \left[\begin{array}{cccc} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{array}\right] \tag{94}$$

Note that $\mathbf{H}(x) = \boldsymbol{J}(\nabla f)$, the Jacobian of the gradient of f.

Given a vector $v \in \mathbb{R}^n$, the Hessian of f at v is:

$$\boldsymbol{H}(x)(\boldsymbol{v}) := \frac{1}{2}\boldsymbol{v}\boldsymbol{H}(x)\boldsymbol{v}^T \tag{95}$$

Quadratic form expansion

For an equation on the quadratic form, if A is a symmetric matrix, we have:

$$\frac{1}{2}(\boldsymbol{x}-\boldsymbol{y})^T\boldsymbol{A}(\boldsymbol{x}-\boldsymbol{y}) = \frac{1}{2}\boldsymbol{y}^T A\boldsymbol{y} - \boldsymbol{y}^{\boldsymbol{T}}\boldsymbol{A}\boldsymbol{x} + \frac{1}{2}\boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x}$$

Matrix differentiation

Suppose the system is $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}$:

$$\frac{\partial \boldsymbol{y}}{\partial \boldsymbol{x}} = \boldsymbol{A} \tag{96a}$$

$$\frac{\partial \boldsymbol{y}}{\partial \boldsymbol{z}} = \boldsymbol{A}\frac{\partial \boldsymbol{x}}{\partial \boldsymbol{z}} \tag{96b}$$

$$\tag{96c}$$

Suppose α is a scalar defined by $\alpha = \boldsymbol{y}^T\boldsymbol{A}\boldsymbol{x}$, then:

$$\frac{\partial \alpha}{\partial \boldsymbol{x}} = \boldsymbol{y}^T\boldsymbol{A} \tag{97a}$$

$$\frac{\partial \alpha}{\partial \boldsymbol{y}} = \boldsymbol{x}^T\boldsymbol{A}^T \tag{97b}$$

$$\tag{97c}$$

If the scalar α is given by the quadratic form $\alpha = \boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x}$ then:

$$\frac{\partial \alpha}{\partial \boldsymbol{x}} = \boldsymbol{x}^T(\boldsymbol{A} + \boldsymbol{A}^T) \tag{98a}$$

$$\frac{\partial \alpha}{\partial \boldsymbol{z}} = \boldsymbol{x}^T(\boldsymbol{A} + \boldsymbol{A}^T)\frac{\partial \boldsymbol{x}}{\partial \boldsymbol{z}} \tag{98b}$$

If $\boldsymbol{A}$ is a symmetric matrix and α is in the quadratic form as above, then:

$$\frac{\partial \alpha}{\partial \boldsymbol{x}} = 2\boldsymbol{x}^T\boldsymbol{A} \tag{99a}$$

$$\frac{\partial \alpha}{\partial \boldsymbol{z}} = 2\boldsymbol{x}^T\boldsymbol{A}\frac{\partial \boldsymbol{x}}{\partial \boldsymbol{z}} \tag{99b}$$

LU factorization

Used to find a permutation matrix P that exchanges columns of a matrix:

$$PA = LU \tag{100}$$

where

- P is an $n \times n$ permutation matrix (obtained by rearranging the rows of the $n \times n$ identity matrix)
- L is unit lower triangular (lower triangular with diagonal elements equal to 1) U is upper triangular

LU factorization solves a linear system $Ax = b$ in the three step process:

1. form $\tilde{b} = Pb$ by permuting the elements of b
2. solve $Lz = \tilde{b}$ by performing triangular forward-substitution, to obtain the vector z
3. solve $Ux = z$ by performing triangular back-substitution, to obtain the solution vector x

Notes

- If using Gaussian Elimination with Row Partial Pivoting, then the algorithm requires $2n^3/3$ floating-point operations when A is dense

Cholesky factorization

When $A \in \mathbb{R}^{n \times n}$ is symmetric positive definite, we can factorize at half the cost: $n^3/3$ operations. Cholesky factorization produces a matrix L such that

$$A = LL^\intercal \tag{101}$$

Notes:

- Cholesky factorization can be used to verify positive definiteness of a symmetric matrix A.
-

Modified Cholesky factorization

Modifies Hessian when it is not positive definite. Performs Cholesky factorization, but increases diagonal elements encountered during factorization to ensure they are sufficiently positive: