CSE260 Optimization: Lecture notes

Spring semester 2008, UC Merced

Miguel Á. Carreira-Perpiñán

Based mainly on:
Jorge Nocedal and Stephen J. Wright: *Numerical Optimization*. Springer-Verlag, 1999.

CH. 1: INTRODUCTION

* Goal: describe the basic concepts & main state-of-the-art algorithms for continuous optimisation.

* The optimisation problem:

equality constraints (scalar)

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

inequality constraints (scalar)

variables (vector) objective function (scalar)

Feasible region: set of points satisfying all constraints

$$\max f \equiv -\min -f$$

* Ex (see fig. 1.1 in book): $\min (x_1-2)^2+(x_2-1)^2$ s.t. $\begin{cases} x_1^2-x_2\leq 0 \\ x_1+x_2\leq 2 \end{cases}$

* Ex: transportation problem (LP)

$$\min \sum_{ij} c_{ij}x_{ij} \quad \text{s.t.} \begin{cases} \sum_j x_{ij} \leq a_i & \forall i & \text{(capacity of factory } i\text{)} \\ \sum_i x_{ij} \geq b_j & \forall j & \text{(demand of shop } j\text{)} \\ x_{ij}\geq 0 & \forall i,j & \text{(nonnegative production)} \end{cases}$$

shipping cost

amount of product shipped from factory i to shop j

* Ex: LSQ problem: fit a parametric model (eg. line, polynomial, neural net..) to a data set (see Ex. 2.1 in book).

* Optimisation algorithms are iterative: build sequence of points that converges to the solution. Needs good initial point (often by prior knowledge)

* Focus on many-variable problems (but will illustrate in 2D).

* Desiderata for algorithms:
  - Robustness: perform well on wide variety of problems in their class, for any starting point
  - Efficiency: little computer time or storage
  - Accuracy: identify solution precisely (within the limits of fixed-point arithmetic)

They conflict with each other.

* General comment about optimisation (Fletcher): "fascinating blend of theory and computation, heuristics and rigour".

- No universal algorithm: a given algorithm works well with a given class of problems
- Necessary to adapt a method to the problem at hand (by experimenting)
- Not choosing an appropriate algorithm → solution found very slowly, or not at all.

* Not covered in the Nocedal-Wright book, or in this course:

- Discrete optimisation (integer programming): the variables are discrete. Ex. integer transportation problem, travelling salesman problem.
  - Harder to solve than continuous opt (in the latter we can predict the objective function value at nearby points)
  - Too many solutions to count them
  - Rounding typically gives very bad solutions
  - Highly specialised techniques for each problem type

  Ref: Papadimitriou & Steiglitz
- Network opt: shortest paths, max flow, min cost flow, assignments & matchings, MST, dynamic programming, graph partitioning...

  Ref: Ahuja, Magnanti & Orlin.
- Nonsmooth opt: discontinuous derivatives, eg L_1-norm.

  Ref: Fletcher
- Stochastic opt: the model is specified with uncertainty, eg $x \le b$ where b could be given by a probability density function.
- Global opt: find the global minimum, not just a local one. Very difficult. Some heuristics: simulated annealing, genetic algorithms, evolutionary computation.
- Multiobjective opt: one approach is to transform it into a single objective = linear combination of objectives.
- EM algorithms (Expectation-Maximisation): specialised technique for maximum likelihood estimation of probabilistic models.

  Ref: McLachlan & Peel; many books on statistics or machine learning
- Calculus of variations: stationary points of a functional (= function of functions)
- Convex optimisation: we'll see some of this Ref: Boyd & Vandenberghe.

- Derivative-free (direct) methods: in practice, we typically can compute ∇ and possibly ∇^2 cheaply and if so it results in more efficient methods.
  Ref: Luenberger; Fletcher.
- Modelling: the setup of the opt problem, i.e. the process of identifying objective, variables and constraints for a given problem. Very important but application-dependent.
  Ref: Dantzig; Ahuja et al

* Course contents: derivative-based methods for continuous optimisation (see syllabus).

CH. 2: FUNDAMENTALS OF UNCONSTRAINED OPTIMISATION

Problem: $\min f(x)$, $x \in \mathbb{R}^n$

* CONDITIONS FOR A LOCAL MINIMUM x^*

- Global minimiser: $f(x^*) \leq f(x)$ $\forall x \in \mathbb{R}^n$ [Ex:]
- Local " : $\exists$ neighbourhood $\mathcal{N}$ of x^*: $f(x^*) \leq f(x)$ $\forall x \in \mathcal{N}$
- Strict (or strong) local minimiser: $f(x^*) < f(x)$ $\forall x \in \mathcal{N} \setminus \{x^*\}$. [Ex. $f(x)=2$ vs. $f(x)=(x-2)^4$]
- Isolated local minimiser: $\exists$ $\mathcal{N}$ of x^* such that x^* is the only local min. in $\mathcal{N}$. [Ex. $f(x) = x^4 \cos\frac{1}{x} + 2x^4$ with $f(0)=0$, and $x^*=0$]

  All isolated local min are strict.

- First-order necessary conditions (Th. 2.2): x^* local min, f cont. diff. in an open neighbourhood of x^* $\Rightarrow$ $\nabla f(x^*)=0$ [Not sufficient condition, ex: $f(x)=x^3$]
  [Pf. by contradiction: if $\nabla f(x^*) \neq 0$ then f decreases along the gradient direction]
- Stationary point: $\nabla f(x^*)=0$.
- Second-order necessary conditions (Th. 2.3): x^* local min, f twice cont diff in an open neighbourhood of x^* $\Rightarrow$ $\nabla f(x^*)=0$ and $\nabla^2 f(x^*)$ is psd
  [Pf. by contradiction: if $\nabla^2 f(x^*)$ is not psd then f decreases along the directions where ∇^2 is not psd]
- Second-order sufficient conditions (Th. 2.4): $\nabla^2 f$ continuous in an open neighbourhood of x^*, $\nabla f(x^*)=0$, $\nabla^2 f(x^*)$ pd $\Rightarrow$ x^* is a strict local minimiser of f.
  [Pf. Taylor-expand f around x^*] [Not necessary condition, ex: $f(x)=x^4$ at $x^*=0$]

The key for the conditions is that ∇, ∇^2 exist and are continuous. The smoothness of f allows us to predict approximately the landscape around a point x.

* Convex optimisation:

- $S \subset \mathbb{R}^n$ is a convex set if $x, y \in S \Rightarrow \alpha x + (1-\alpha) y \in S \quad \forall \alpha \in [0,1]$
- $f: S \subset \mathbb{R}^n \to \mathbb{R}$ is a convex function if its domain S is convex and

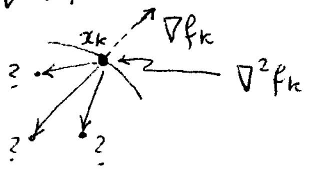
  $f(\alpha x + (1-\alpha) y) \leq \alpha f(x) + (1-\alpha) f(y) \quad \forall \alpha \in [0,1], \; \forall x, y \in S$
- Convex opt problem: $\begin{cases} \text{objective function is convex} \\ \text{equality constraints are linear} \\ \text{inequality} \quad " \quad\quad " \quad \text{concave} \end{cases}$. Ex. linear programming (LP)
- Easier to solve because every local min is a global min.
- Th. 2.5: f convex $\Rightarrow$ any local min is also global

  f convex and differentiable $\Rightarrow$ any stationary point is a global min.

[Pf. by contradiction: assume z with $f(z) < f(x^*)$, study the segment x^* — z]

* ALGORITHM OVERVIEW

- Algorithms look for a stationary point starting from a point x_0 (arbitrary or user-supplied) $\Rightarrow$ sequence of iterates $\{x_k\}_{k=0}^{\infty}$ that terminates when no more progress can be made, or it seems that a solution has been approximated with sufficient accuracy.
- We choose x_{k+1} given information about f at x_k (and possibly earlier iterates) so that $f(x_{k+1}) < f(x_k)$ (descent)
- Move $x_k \to x_{k+1}$: two fundamental strategies, line search and trust region.



– Line search strategy

1. Choose a direction p_k
2. Search along p_k from x_k for x_{k+1} with $f(x_{k+1}) < f(x_k)$, i.e. approximately solve the 1D minimisation problem $\min_{\alpha > 0} f(x_k + \alpha p_k)$ where α = step length.

– Trust region strategy

1. Construct a model function m_k (typ. quadratic) that is similar to (but simpler than) f around x_k.
2. Search for x_{k+1} with $m_k(x_{k+1}) < m(x_k)$ inside a small trust region (typ. a ball) around x_k, i.e., approximately solve the n-D minimisation problem

$$\min_{p} m_k(x_k+p) \text{ s.t. } x_k+p \in \text{trust region.}$$

3. If x_{k+1} does not produce enough decay in f, shrink the region.

In both strategies, the subproblem (step 2) is easier to solve than the real problem. Why not solve the subproblem exactly?

- Good: derives maximum benefit from p_k or m_k; but
- Bad: expensive (many iterations over α) and unnecessary towards the real problem (min f over all $\mathbb{R}^n$).

Both strategies differ in the order in which they choose the direction and the distance of the move:

- Line search: fix direction, choose distance
- Trust region: fix maximum distance, choose direction and actual distance.

* Scaling ("units" of the variables): a problem is poorly scaled if changes to x in a certain direction produce much larger variations in the value of f than do changes to x in another direction. Some algorithms, (eg. steepest descent) are sensitive to poor scaling while others (eg. Newton's method) are not. Generally, scale-invariant algorithms are more robust to poor problem formulations.

Ex: $f(x) = 10^9 x_1^2 + x_2^2$, fig. 2.7

* RATES OF CONVERGENCE

Let $\{x_k\}_{k=0}^{\infty} \subset \mathbb{R}^n$ be a sequence that converges to x^*.

- Linear convergence: $\frac{\|x_{k+1}-x^*\|}{\|x_k-x^*\|} \le r$ for all k sufficiently large, with constant $0<r<1$.
  The distance to the solution decreases at each iteration by at least a constant factor. Ex: steepest descent (and $r \simeq 1$ for ill-conditioned problems).
- Superlinear convergence: $\lim_{k\to\infty} \frac{\|x_{k+1}-x^*\|}{\|x_k-x^*\|} = 0$. Ex: quasi-Newton methods (typic.)
- Quadratic convergence (order 2): $\frac{\|x_{k+1}-x^*\|}{\|x_k-x^*\|^2} \le M$ for all k sufficiently large, with constant $M>0$ (not necessarily <1). We double the number of digits at each iteration. Quadratic faster than superlinear faster than linear.
- Order p: $\frac{\|x_{k+1}-x^*\|}{\|x_k-x^*\|^p} \le M$ (rare for $p>2$).

The speed of an algorithm depends mainly on the order p (in the long run) and on r (for $p=1$), and more weakly on M. The values of r, M depend on the algorithm and on the particular problem.

REVIEW OF CONDITIONS FOR A MINIMISER

Assuming the derivatives $\nabla f(x^*)$, $\nabla^2 f(x^*)$ exist and are continuous in a neighbourhood of x^*:

- x^* is a local minimiser $\Rightarrow \begin{cases} \nabla f(x^*) = 0 & (1^{st} \text{ order}) \\ \nabla^2 f(x^*) \text{ psd} & (2^{nd} \text{ order}) \end{cases}$ $\begin{pmatrix}\text{necessary} \\ \text{condition}\end{pmatrix}$

- $\nabla f(x^*) = 0$, $\nabla^2 f(x^*)$ pd $\Rightarrow$ x^* is a strict local minimiser $\begin{pmatrix}\text{sufficient} \\ \text{condition}\end{pmatrix}$

- Stationary point: $\nabla f(x^*) = 0$; $\nabla^2 f(x^*) \begin{cases} \text{pd: strict local minimiser} \\ \text{nd: '' '' maximiser} \\ \text{not definite (pos, neg eigenvalues): saddle point} \\ \text{psd: may be nonstrict local minimiser} \\ \text{nsd: '' '' '' '' maximiser} \end{cases}$

CH. 3: LINE SEARCH METHODS

search direction

Iteration: $x_{k+1} = x_k + \alpha_k p_k$

step length (how far to move along p_k), $\alpha_k > 0$

Descent direction: $p_k^T \nabla f_k = \|p_k\| \|\nabla f_k\| \cos\theta_k < 0$ (angle $< \frac{\pi}{2}$ with $-\nabla f_k$)

Guarantees that f can be reduced along p_k (for a sufficiently small step):

$$f(x_k + \varepsilon p_k) = f(x_k) + \varepsilon p_k^T \nabla f_k + O(\varepsilon^2) \qquad (\text{Taylor's th.})$$

$$< f(x_k) \text{ for all sufficiently small } \varepsilon > 0$$

- The steepest descent direction, i.e., the direction along which f decreases most rapidly, is $p_k = -\nabla f_k$. Pf: by Taylor's th. for any p, α:

$$f(x_k + \alpha p) = f(x_k) + \alpha p^T \nabla f_k + O(\alpha^2)$$

so the rate of change in f along p at x_k is $p^T \nabla f_k$ (the directional derivative) $= \|p\| \|\nabla f_k\| \cos\theta$. Then $\min_p p^T \nabla f_k$ s.t. $\|p\| = 1$ is achieved when $\cos\theta = -1$, i.e., $p = -\nabla f_k / \|\nabla f_k\|$. Fig. 2.5, 2.6.

This direction is $\perp$ to the contours of f.

- The Newton direction is $p_k = -\nabla^2 f_k^{-1} \nabla f_k$. This corresponds to assuming f is locally quadratic and jumping directly to its minimum. Pf: by Taylor's th.:

$$f(x_k + p) \simeq f_k + p^T \nabla f_k + \frac{1}{2} p^T \nabla^2 f_k\, p = m_k(p) \qquad \nabla^2 f_k \text{ is pd.}$$

which is minimised (take derivatives wrt p) by the Newton direction if ~~the Hessian~~

Q: what happens if assuming f is locally linear (order 1)?

In a line search the Newton direction has a natural step length of 1.

- For most algorithms, $p_k = -B_k^{-1} \nabla f_k$ where B_k is symmetric, nonsingular:
  - Steepest descent: $B_k = I$
  - Newton's method: $B_k = \nabla^2 f(x_k)$
  - Quasi-Newton method: $B_k \simeq \nabla^2 f(x_k)$

  If B_k is pd then p_k is a descent direction: $p_k^T \nabla f_k = -\nabla f_k^T B_k^{-1} \nabla f_k < 0$.

Here, we deal with how to choose the step length given the search direction p_k.

Desirable properties: guaranteed global convergence and rapid rate of convergence.

* STEP LENGTH

Time / accuracy tradeoff: want to choose α_k to give a substantial reduction in f but not to spend much time on it.

- Exact line search (global or local min): α_k: $\min_{\alpha>0} \phi(\alpha) = f(x_k + \alpha p_k)$.

  Too expensive: many evaluations of f, ∇f to find α_k even with moderate precision.
- Inexact linesearch: a typical ~~the~~ l.s. algorithm will try a sequence of α values and stop when certain conditions hold.

We want easily verifiable theoretical conditions on the step length that allow to prove convergence of an optimisation algorithm.

- Reduction in f: $f(x_k + \alpha_k p_k) < f(x_k)$ → not enough, can converge before reaching the minimiser.

* Wolfe conditions:

$$\left.\begin{array}{ll} ① & f(x_k + \alpha_k p_k) \le f(x_k) + c_1 \alpha_k \nabla f_k^T p_k \\ ② & \nabla f(x_k + \alpha_k p_k)^T p_k \geq c_2 \nabla f_k^T p_k \end{array}\right\} \quad 0 < c_1 < c_2 < 1.$$

Call $\phi(\alpha) = f(x_k + \alpha p_k)$, then $\phi'(\alpha) = \nabla f(x_k + \alpha p_k)^T p_k$.

① Sufficient decrease (Armijo condition) is equivalent to $\phi(0) - \phi(x_k) \geq \alpha_k(-c_1 \phi'(0))$.

The reduction is proportional both to $\left\langle \begin{array}{l} \text{step length } \alpha_k \\ \text{directional derivative } \nabla f_k^T p_k \end{array}\right.$. Fig. 3.3.

In practice, c_1 is very small, eg. $c_1 = 10^{-4}$.

It is satisfied for all sufficiently small α ⇒ not enough, need to rule out unacceptably small steps.

② Curvature condition is equivalent to $-\phi'(\alpha_k) \le -c_2 \phi'(0)$.

Reason: if the slope at α, $\phi'(\alpha)$, is strongly negative, it's likely we can reduce f significantly by moving further. Fig. 3.4, 3.5.

In practice $c_2 = \begin{cases} 0.9 & \text{if } p_k \text{ is chosen by a Newton or quasi-Newton method} \\ 0.1 & \text{'' '' '' '' '' '' nonlinear conjugate gradient method.} \end{cases}$

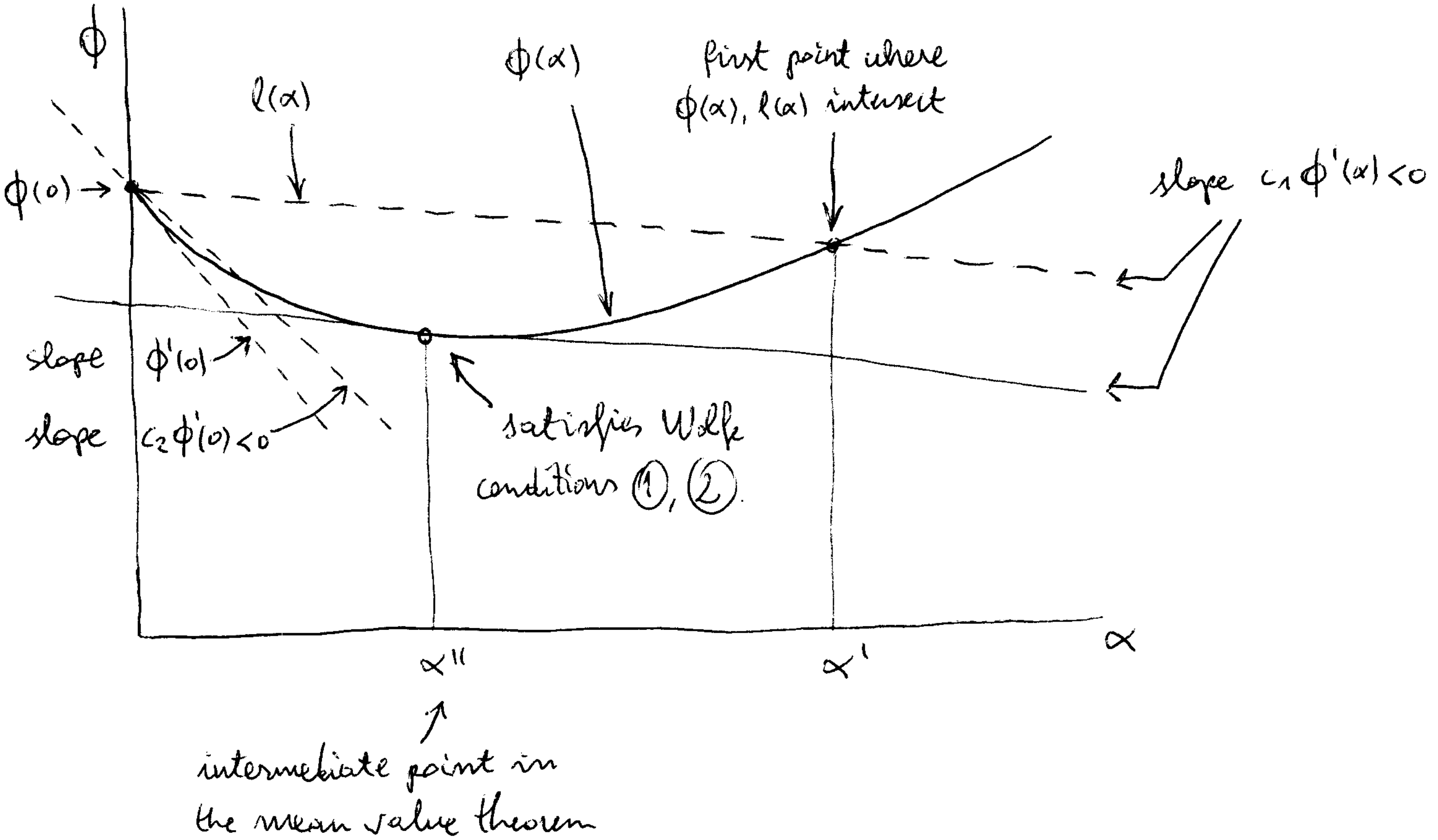
We will concentrate on the Wolfe conditions in general, and assume they always hold when the l.s. is used as part of an optimisation algorithm (allows convergence proofs).

Lemma 3.1: there always exist step lengths that satisfy the Wolfe (also the strong Wolfe) conditions if f is smooth and bounded below. [Pf: mean value th.]

PROOF OF LEMMA 3.1 (STEP LENGTHS FOR WOLFE CONDITIONS)

$$\phi(\alpha) = f(x_k + \alpha_k p_k)$$

$$\ell(\alpha) = f_k - (-c_1 \nabla f_k^T p_k)\alpha = f_k - (-c_1 \phi'(0))\alpha$$



Other useful conditions:

- Strong Wolfe conditions: ① + ②' $|\nabla f(x_k+\alpha_k p_k)^T p_k| \le c_2 |\nabla f_k^T p_k|$

  We don't allow $\phi'(\alpha_k)$ to be too positive, so we exclude points that are far from stationary points of ϕ.

- Goldstein conditions: fig. 3.6

$$f(x_k) + (1-c)\alpha_k \nabla f_k^T p_k \le f(x_k+\alpha_k p_k) \overset{\textcircled{1}}{\le} f(x_k) + c\,\alpha_k \nabla f_k^T p_k\,, \quad 0<c<\tfrac{1}{2}$$

  Controls step from below Disadvantage: may exclude all minimisers of ϕ

  Q: do the Wolfe conditions exclude minimisers?

* Sufficient decrease and backtracking: start with largish step size and decrease it (times $\rho<1$) until it meets the sufficient decrease condition ①. Proc. 3.1

- It's a heuristic approach to avoid a more careful l.s. that satisfies the Wolfe cond.
- It always terminates because ① is satisfied by sufficiently small α.
- Works well in practice because the accepted α_k is near (times ρ) the previous α, which was rejected for being to long.
- The initial step length $\bar{\alpha}$ is 1 for Newton and quasi-Newton methods.

* step-length selection algorithms

- They take a starting value of α and generate a sequence $\{\alpha_i\}$ that satisfies the Wolfe cond. Usually they use interpolation, eg. approximate $\phi(\alpha)$ as a cubic poly.
- There are also derivative-free methods (eg. the golden section search) but they are less efficient and can't benefit from the Wolfe cond. (to prove global convergence).
- We'll just use backtracking for simplicity.

* CONVERGENCE OF LINE SEARCH METHODS

- Global convergence: $\|\nabla f_k\| \xrightarrow[k\to\infty]{} 0$, i.e., convergence to a stationary point for any starting point x_0. To ensure convergence to a minimiser we need more information, eg. the Hessian.
- We give a ~~condition~~ theorem for the search direction p_k to obtain global convergence, focusing on the angle θ_k between p_k and the steepest descent direction $-\nabla f_k$: $\cos\theta_k = \frac{-\nabla f_k^T p}{\|\nabla f_k\|\,\|p\|}$, and assuming the Wolfe cond. Similar theorems exist for strong Wolfe, Goldstein cond.
- Important theorem, eg. shows that the steepest descent method is globally convergent; for other algorithms, it describes how far p_k can deviate from the steepest descent direction and still give rise to a globally convergent iteration.

Th. 3.2 (Zoutendijk): consider an iterative method $x_{k+1} = x_k + \alpha_k p_k$ with starting point x_0 where p_k is a descent direction and α_k satisfies the Wolfe conditions. Suppose f is bounded below in $\mathbb{R}^n$ and cont. diff. in an open set $\mathcal{N}$ containing the level set $\mathcal{L} = \{x : f(x) \le f(x_0)\}$, and that ∇f is Lipschitz continuous on $\mathcal{N}$ weaker than bounded gradient ($\Leftrightarrow \exists L > 0 : \|\nabla f(x) - \nabla f(\tilde{x})\| \le L \|x - \tilde{x}\| \ \forall x, \tilde{x} \in \mathcal{N}$). Then:

$$\sum_{k \ge 0} \cos^2\theta_k \, \|\nabla f_k\|^2 < \infty \qquad \text{(Zoutendijk's condition)}$$

[Proof]

- Zoutendijk's condition implies $\cos^2\theta_k \|\nabla f_k\|^2 \to 0$. Thus, if ~~$\theta_k$~~ $\cos\theta_k \ge \delta > 0 \ \forall k$ for fixed δ then $\|\nabla f_k\| \to 0$ (global convergence).

Examples:

- Steepest descent method: $p_k = -\nabla f_k \Rightarrow \cos\theta_k = 1 \Rightarrow$ global convergence. Intuitive method, but very slow in difficult problems. Fig. 3.7
- Newton-like method: $p_k = -B_k^{-1} \nabla f_k$ with B_k symmetric, pd and with bounded condition number: $\|B_k\| \|B_k^{-1}\| \le M \ \forall k$. Then $\cos\theta_k \ge \frac{1}{M}$ (see exercise 3.5) $\Rightarrow$ global convergence. why? ill-cond $\Rightarrow \nabla f \perp$ Newton dir.
  In other words, if B_k are pd (which is required for descent directions), have bounded c.n. and the step lengths satisfy the Wolfe conditions $\Rightarrow$ global convergence. This includes steepest descent, some Newton and quasi-Newton methods.

- For some methods (eg conjugate gradients) we may have directions that are almost $\perp \nabla f_k$ when the Hessian is ill conditioned. It is still possible to show global convergence by ensuring that we take a steepest descent step from time to time. "Turning" the directions toward $-\nabla f_k$ so that $\cos\theta_k < \delta$ for some preselected $\delta > 0$ is generally a bad idea: it slows down the method (difficult to choose a good δ) and also destroys the invariance properties of quasi-Newton methods.

- Fast convergence can sometimes conflict with global convergence, eg. steepest descent is globally convergent but quite slow; Newton's method converges very fast when near a solution but away from the solution its steps may not even be descent (indeed, it may~~be~~ be looking for a maximiser!). The challenge is to design algorithms with both fast and global convergence.

* RATE OF CONVERGENCE

For quadratic functions (with matrix Q): $\|x_{k+1} - x^*\|_Q \leq r\,\|x_k - x^*\|_Q$ where $\|x\|_Q = x^TQx$ and $\frac{1}{2}\|x - x^*\|_Q^2 = f(x) - f(x^*)$.

- Steepest descent: $p_k = -\nabla f_k$.

Th. 3.4: assume f is twice cont. diff. and that the iterates generated by the steepest descent method with exact line searches converge to a point where the Hessian $\nabla^2 f(x^*)$ is pd. Then $f(x_{k+1}) - f(x^*) \leq r^2(f(x_k) - f(x^*))$, where $r = \left(\frac{\lambda_n - \lambda_1}{\lambda_n + \lambda_1}\right) = \left(\frac{\kappa - 1}{\kappa + 1}\right)$, $0 < \lambda_1 \leq \cdots \leq \lambda_n$ are the eigenvalues of $\nabla^2 f(x^*)$ and $\kappa = \lambda_n / \lambda_1$ its condition number. [Pf idea: near the min, f is approx. quadratic]

Thus, the convergence rate is linear, with two extremes:

- Very well conditioned Hessian: $\lambda_1 \simeq \lambda_n$; very fast, since the steepest descent direction approximately points to the minimiser.
- Ill-conditioned Hessian: $\lambda_1 << \lambda_n$; very slow, zigzagging behaviour. This is the typical situation in practice.

- Quasi-Newton methods: $p_k = -B_k^{-1}\nabla f_k$ with B_k symmetric pd.

The convergence rate is superlinear iff:

1. $\lim_{k\to\infty} \frac{\|(B_k - \nabla^2 f(x^*))p_k\|}{\|p_k\|} = 0$, i.e., the matrices B_k become increasingly accurate approximations of the Hessian along the search directions p_k.
2. Near the solution the step length α_k is always 1, i.e., $x_{k+1} = x_k + p_k$.

Thus, in practice we must always try $\alpha = 1$ in the line search and accept it if it satisfies the Wolfe conditions.

- Newton's method: $p_k = -\nabla^2 f_k^{-1}\,\nabla f_k$.

Near the solution, where the Hessian is pd, the convergence rate is quadratic if we always take $\alpha_k = 1$. The theorems do not apply if the Hessian is not pd (away from the solution); practical Newton methods avoid this.

- Coordinate-descent algorithm (or method of alternating variables): p_k cycles through the n coordinate directions $e_1, \dots, e_n$ in turn. Fig. 3.8.
  - May not converge, iterating indefinitely without approaching a stationary point, if the gradient becomes more and more $\perp$ to the coordinate directions. Then, $\cos\theta_k$ approaches 0 sufficiently rapidly that the Zoutendijk condition is satisfied even when $\nabla f_k \not\to 0$.

- If it does converge, its rate of convergence is often much slower than that of steepest descent, and this gets worse as n increases.
- Advantages: very simple, does not require calculation of derivatives, convergence rate ok if the variables are loosely coupled.

CH. 4: TRUST-REGION METHODS

Iteration: $x_{k+1} = x_k + p_k$, where p_k is the approximate minimiser of the model $m_k(p)$ in a region around x_k (the trust region); if p_k does not produce a ~~[illegible]~~ sufficient decrease in f, we shrink the region and try again.

- The trust region is typically a ball $B(x_k; \Delta)$
  Elliptical and box-shaped regions may also be used.
- Each time we decrease Δ after failure of a candidate iterate, the step from x_k is shorter and usually points in a different direction.
- Tradeoff in Δ: $\begin{cases} \text{too small: good model, but can only take a small step, so slow convergence} \\ \text{too large: bad model, we may have to reduce } \Delta \text{ and repeat} \end{cases}$

  In practice, we increase Δ if previous steps showed the model reliable. Fig. 4.1.
- Linear model: $m_k(p) = f_k + \nabla f_k^T p$ s.t. $\|p\| \leq \Delta_k \Rightarrow p_k = -\Delta_k \frac{\nabla f_k}{\|\nabla f_k\|}$,
  i.e., steepest descent with step length α_k given by Δ_k (no news).
  Q: what is the approx. error for the linear model?
- Quadratic model: $m_k(p) = f_k + \nabla f_k^T p + \frac{1}{2} p^T B_k p$ where B_k is symmetric.
  The approximation error is $\begin{cases} \mathcal{O}(\|p\|^3) \text{ if } B_k = \nabla^2 f_k & \text{(trust-region Newton method)} \\ \mathcal{O}(\|p\|^2) \text{ otherwise} \end{cases}$

  In both cases, the model is accurate for small $\|p\|$, which guarantees we can always find a good step for sufficiently small Δ. Two issues remain: how to choose Δ_k? how to find p_k?

  Q: what happens if $\rho_k < 0$ but $\|p_k\| < \Delta_k$ with an arbitrary m_k?

1. Choice of the trust-region radius Δ_k

Define the ratio $\rho_k = \dfrac{f_k - f(x_k + p_k)}{m_k(0) - m_k(p_k)} = \dfrac{\text{actual reduction}}{\text{predicted reduction}}$.

- Predicted reduction ≥ 0 always (since $p = 0$ is in the region)
- If actual reduction < 0 the new objective value is larger, so reject the step.

- ρ_k:
  - $\simeq 1$: good agreement between f and the model m_k, so expand Δ_k if $\|p_k\| = \Delta_k$ (otherwise, don't interfere)
  - > 0 but not close to 1: keep Δ_k
  - close to 0 or negative: shrink Δ_k

See algorithm 4.1.

2. The optimisation subproblem

$$\min_{p \in \mathbb{R}^n} m_k(p) = f_k + \nabla f_k^T p + \tfrac{1}{2} p^T B_k p \quad \text{s.t.} \quad \|p\| \leq \Delta_k$$

- If B_k is pd and $\|B_k^{-1} \nabla f_k\| \leq \Delta_k$ the solution is the unconstrained minimiser $p_k = -B_k^{-1} \nabla f_k$ (full step)
- Otherwise, we compute an approximate solution. One approach is based on the following characterisation of the exact solution:

Th. 4.3. p^* is a global solution of the trust-region problem $\min\limits_{\|p\| \leq \Delta} m(p) = f + g^T p + \frac{1}{2} p^T B p$ iff p^* is feasible and $\exists\, \lambda \geq 0$ such that:

a) $(B + \lambda I) p^* = -g$

b) $\lambda (\Delta - \|p^*\|) = 0$ (i.e., $\lambda = 0$ or $\|p^*\| = \Delta$)

c) $B + \lambda I$ is psd

Algorithm:

1. Try $\lambda = 0$, solve $Bp^* = -g$ and see if $\|p^*\| \leq \Delta$.
2. If $\|p^*\| > \Delta$, define $p(\lambda) = -(B + \lambda I)^{-1} g$ for λ sufficiently large that $B + \lambda I$ is pd and seek a smaller value $\lambda > 0$ such that $\|p(\lambda)\| = \Delta$ (1D root-finding for λ; iterative solution factorising the matrix $B + \lambda I$).

Note that using $B + \lambda I$ instead of B in the model transforms the problem into $\min\limits_p m(p) + \frac{\lambda}{2} \|p\|^2$, and so for large $\lambda > 0$ the minimiser is strictly inside the region. As we decrease λ, the minimiser moves to the region boundary and the theorem holds for that λ.

- This is useful for Newton's method and is the basis of the Levenberg-Marquardt algorithm for nonlinear least-squares problems.
- Under certain assumptions, this algorithm has global convergence (Th. 4.9) if using $B_k = \nabla^2 f_k$.

REVIEW OF LINE SEARCH METHODS (l.s.)

- Iteration $x_{k+1} = x_k + \alpha_k p_k$ $\begin{cases} \text{search direction } p_k \text{ given by optimisation method} \\ \text{l.s. = determine step length } \alpha_k \end{cases}$

- We want:
  - Descent direction: $p_k^T \nabla f_k = \|p_k\| \|\nabla f_k\| \cos\theta_k < 0$ $\begin{cases} \text{steepest descent dir.: } -\nabla f_k \\ \text{Newton dir: } -\nabla f_k^{2} \nabla f_k \\ \text{Quasi-Newton dir: } -B_k^{-1} \nabla f_k \end{cases}$
    (B_k pd $\Rightarrow$ descent dir)
  - Inexact l.s.: approx. solution of $\min_{\alpha > 0} f(x_k + \alpha p_k)$ (faster convergence of the overall algorithm).
    Even if the l.s. is inexact, if α_k satisfies certain conditions at each k then the overall algorithm has global convergence.
    An example are the Wolfe conditions (others exist), most crucially the sufficient decrease in f. A simple l.s. algorithm that often (not always) satisfies the Wolfe cond. is backtracking (better ones exist).

- Global convergence: $\|\nabla f_k\| \to 0$ (to a stationary point)
  Zoutendijk's th: descent dir + Wolfe + mild cond. on f $\Rightarrow \sum_{k\geq 0} \cos^2\theta_k \|\nabla f_k\|^2 < \infty$.
  Corollary: $\cos\theta_k \geq \delta > 0 \ \forall k \Rightarrow$ global convergence. But we often want $\cos\theta_k \simeq 0$!
  Ex: steepest descent, some Newton-like methods have global convergence.

- Convergence rate:
  - Steepest descent: linear, $r = \left(\frac{\lambda_n - \lambda_1}{\lambda_n + \lambda_1}\right)^2$; slow for ill-conditioned problems.
  - Quasi-Newton: superlinear under certain conditions.
  - Newton: quadratic near the solution.

REVIEW OF TRUST-REGION METHODS

- Iteration $x_{k+1} = x_k + p_k$
    - p_k = approx. minimiser of model m_k of f in trust region: $\min\limits_{\|p\| \leq \Delta_k} m_k(p)$.
    - p_k does not produce sufficient decrease ⇒ region too big. shrink it and try again.

Insufficient decrease $\Leftrightarrow$ $\rho_k = \dfrac{f_k - f(x_k + p_k)}{m_k(0) - m_k(p_k)} \lesssim 0$.

- Quadratic model: $m_k(p) = f_k + \nabla f_k^T p + \frac{1}{2} p^T B_k p$.

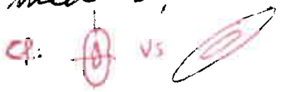
  The exact solution of this in the trust region $\|p\| \leq \Delta_k$ satisfies certain conditions (th. 4.3) that can be used to find an approximate solution.
- Mainly useful for Newton and Levenberg-Marquardt methods.

CH. 5: CONJUGATE GRADIENT METHODS

- Linear conjugate gradient method: solves a large linear system of equations.
- Nonlinear " " " : adaptation of the linear CG for nonlinear optimiz.

Key feature: requires no matrix storage, faster than steepest descent.

Assume in all this chapter that A is an $n \times n$ symmetric pd matrix,

$\phi(x) = \frac{1}{2} x^T A x - b^T x$ and $\nabla \phi(x) = Ax - b = r(x)$. Cf:  vs

- steepest descent (1 or ∞ its.)
- coordinate descent (n or ∞ its.)
- Newton's method (1 it.)
- CG (n its.)

✱ THE LINEAR CONJUGATE GRADIENT METHOD

Iterative method for solving the two equivalent problems (i.e, both have the same, unique solution x^*):

$$\text{Linear system } Ax = b \iff \text{Optimization problem } \min \phi(x) = \tfrac{1}{2} x^T A x - b^T x$$

- A set of nonzero vectors $\{p_0, p_1, \dots, p_l\}$ is <u>conjugate</u> wrt A iff $p_i^T A p_j = 0 \ \forall i \neq j$.

Conjugacy ⇒ linear independence [Proof: left-multiply $\sum \sigma_i p_i$ times $p_j A$]

- Th. 5.1: we can minimise ϕ in n steps at most by successively minimising ϕ along the n vectors in a conjugate set.

<u>conjugate direction method</u>: given a starting point $x_0 \in \mathbb{R}^n$ and a set of conjugate directions $\{p_0, \dots, p_{n-1}\}$, generate the sequence $\{x_k\}$ with $x_{k+1} = x_k + \alpha_k p_k$ where $\alpha_k = -\frac{r_k^T p_k}{p_k^T A p_k}$ (exact line search). [Proof: $x^* = x_0 + \sum_{i=0}^{n-1} \alpha_i p_i$]

← why $\neq 0$?

- Intuitive idea:
  - A diagonal: quadratic function ϕ can be minimised along the coordinate directions $e_1, \dots, e_n$ in n iterations. Fig. 5.1.
  - A not diagonal: the coordinate directions don't minimise ϕ in n iterations (Fig 5.2); but the variable change $\hat{x} = S^{-1} x$ with $S = (p_0\ p_1 \cdots p_{n-1})$ diagonalises A: $\hat{\phi}(\hat{x}) = \phi(S\hat{x}) = \frac{1}{2} \hat{x}^T (S^T A S) \hat{x} - (S^T b)^T \hat{x}$.

  why invertible?

  Coordinate search in $\hat{x}$ ⇔ conjugate direction search

- Th. 5.2 (<u>expanding subspace minimization</u>): for the conjugate directions method:
  - $r_k^T p_i = 0$ for $i = 0, \dots, k-1$ (the current residual is ⊥ to all previous search directions). Intuition: if $r_k = \nabla\phi(x_k)$ had a nonzero projection along p_i, it would not be a minimum.
  - x_k is the minimiser of ϕ over the set $x_0 + \mathrm{span}\{p_0, \dots, p_{k-1}\}$

That is, the method minimises ϕ piecewise, one direction at a time.

[Proof: induction plus the fact $r_{k+1} = r_k + \alpha_k A p_k$]. $\begin{cases} r_k = Ax_k - b \\ x_{k+1} = x_k + \alpha_k p_k \end{cases}$

- How to obtain conjugate directions? Many ways, eg. using the eigenvectors of A or transforming a set of l.i. vectors into conjugate directions with a procedure similar to Gram-Schmidt. But these are computationally expensive!

- The <u>conjugate gradient method</u> generates conjugate direction p_k by using only the previous one, p_{k-1}:
  - p_k is a l.c. of $-\nabla\phi(x_k)$ and p_{k-1} s.t. being conjugate to p_{k-1} ⇒

$$p_k = -r_k + \beta_k p_{k-1} \quad \text{with} \quad \beta_k = \frac{r_k^T A p_{k-1}}{p_{k-1}^T A p_{k-1}}.$$

  - We start with the steepest descent direction: $p_0 = -\nabla\phi(x_0) = -r_0$.

Algorithm 5.1 (CG - preliminary version): given x_0

```
$r_0 \leftarrow \nabla\phi(x_0) = Ax_0 - b$, $p_0 \leftarrow -r_0$, $k \leftarrow 0$        [Start with steepest descent dir. from $x_0$]
while $r_k \neq 0$                                                         [$r_k = 0$ means we are done, which may happen before n steps]
    $\alpha_k \leftarrow -\frac{r_k^T p_k}{p_k^T A p_k}$, $x_{k+1} \leftarrow x_k + \alpha_k p_k$                    [Exact line search]
    $r_{k+1} \leftarrow A x_{k+1} - b$                                     [New residual]
    $\beta_{k+1} \leftarrow \frac{r_{k+1}^T A p_k}{p_k^T A p_k}$, $p_{k+1} \leftarrow -r_{k+1} + \beta_{k+1} p_k$        [New l.s. direction $p_{k+1}$ is conjugate to $p_k, p_{k-1}, \dots, p_0$]
    $k \leftarrow k+1$
end
```

To prove the algorithm works, we need to prove it builds a conjugate direction set.

- Th. 5.3: suppose that the kth iterate of the CG method is not the solution x^*. Then:

  by construction: at the l.s. minimiser, the gradient is ⊥ to the direction search $p_{k-1} \in \text{span}(r_0,\dots,r_{k-1})$

  - $r_k^T r_i = 0$ for $i = 0, \dots, k-1$ (the gradients at all iterates are ⊥ to each other)

  $\{r_k\}$ orthogonal basis, $\{p_k\}$ basis

  - $\text{span}(r_0, \dots, r_k) = \text{span}(p_0, \dots, p_k) = \text{span}(r_0, Ar_0, \dots, A^k r_0)$ = Krylov subspace of degree k for r_0

  Intuitive explanation: compute r_k, p_k for $k=1,2$ using $\begin{cases} r_{k+1} = r_k + \alpha_k A p_k \\ p_{k+1} = -r_{k+1} + \beta_{k+1} p_k \end{cases}$

  - $p_k^T A p_i = 0$ for $i = 0, \dots, k-1$ (conjugate)

  by construction

Thus the sequence $\{x_k\}$ converges to x^* in at most n steps.

Important: the theorem needs that the first direction be the steepest descent dir.

[Proof: induction]

- We can simplify a bit the algorithm using the following results:
  - $p_{k+1} = -r_{k+1} + \beta_{k+1} p_k$ (construction of the kth direction)
  - $r_{k+1} = r_k + \alpha_k A p_k$
  - $r_k^T p_i = r_k^T r_i = 0$ for $i < k$ (th. 5.2 & 5.3)

Thus $\alpha_k \leftarrow \dfrac{r_k^T r_k}{p_k^T A p_k}$ ↖ $\dfrac{\|r_k\|^2}{\|p_k\|_A^2}$; $r_{k+1} \leftarrow r_k + \alpha_k A p_k$, $\beta_{k+1} \leftarrow \dfrac{r_{k+1}^T r_{k+1}}{r_k^T r_k}$ ↖ $\dfrac{\|r_{k+1}\|^2}{\|r_k\|^2}$ in algorithm 5.2

- ~~Thus~~ Space complexity: $\Theta(n)$ since it computes x, r, p at $k+1$ given the values at k:
- no matrix storage.
- Time complexity: the bottleneck is the matrix-vector product Ap_k which is $\mathcal{O}(n^2)$ (maybe less if A has structure) ⇒ in n steps: $\mathcal{O}(n^3)$, similar to other methods for solving linear systems (eg. Gauss factorisation).
- Advantages: no matrix storage; does not alter A; does not introduce fill (for sparse matrix A); fast convergence.
- Disadvantages: sensitive to rounding errors.

It is recommended for large systems.

✱ Rate of convergence

- Here we don't mean the asymptotic rate ($k \to \infty$) because CG converges in at most n steps for a quadratic function. But CG can get very close to the solution in quite less than n steps, depending on the eigenvalue structure of A:

- If A has only $r < n$ distinct eigenvalues, CG converges in at most r iterations.
- If the eigenvalues of A occur in r distinct clusters, CG will approximately solve the problem in r steps. Fig. 5.4.

- Two bounds (using $\|x\|_A^2 = x^T A x$), useful to estimate the convergence rate in advance if we know something about the eigenvalues of A:
  - $\|x_{k+1} - x^*\|_A^2 \le \left(\frac{\lambda_{n-k} - \lambda_1}{\lambda_{n-k} + \lambda_1}\right)^2 \|x_0 - x^*\|_A^2$ if A has eigenvalues $\lambda_1 \le \cdots \le \lambda_n$.
  - $\|x_k - x^*\|_A \le \left(\frac{\sqrt{\kappa} - 1}{\sqrt{\kappa} + 1}\right)^{2k} \|x_0 - x^*\|_A$ if $\kappa = \frac{\lambda_n}{\lambda_1}$ is the c.n. (this bound is very worse)

  Recall that for steepest descent we had a similar expression but with $\frac{\kappa - 1}{\kappa + 1}$ instead of $\left(\frac{\sqrt{\kappa} - 1}{\sqrt{\kappa} + 1}\right)^2$.

- Preconditioning: change of variables $\hat{x} = Cx$ so that the new matrix $\hat{A} = C^{-T} A C^{-1}$ has a clustered spectrum or a small condition number (thus faster convergence). Finding good preconditioners C depends on the problem (the structure of A).

* NONLINEAR CONJUGATE GRADIENT METHODS

We adapt the linear CG (which minimises a quadratic function ϕ) for a nonlinear function f.

- The Fletcher-Reeves method — α_k is determined by an inexact line search; $r_k = \nabla f$

  Algorithm 5.4: given x_0

```
Evaluate $f_0 \leftarrow f(x_0)$,   $\nabla f_0 \leftarrow \nabla f(x_0)$
$p_0 \leftarrow -\nabla f_0$,   $k \leftarrow 0$
while $\nabla f_k \neq 0$
    $x_{k+1} \leftarrow x_k + \alpha_k p_k$ with inexact l.s. for $\alpha_k$
    Evaluate $\nabla f_{k+1}$
    $\beta_{k+1}^{FR} \leftarrow \frac{\nabla f_{k+1}^T \nabla f_{k+1}}{\nabla f_k^T \nabla f_k}$,   $p_{k+1} \leftarrow -\nabla f_{k+1} + \beta_{k+1}^{FR} p_k$
    $k \leftarrow k+1$
end
```

Uses no matrix operations, requires only f and ∇f.

- Line search for α_k: we need each direction $p_{k+1} = -\nabla f_{k+1} + \beta_{k+1}^{FR} p_k$ to be a descent direction, ie, $\nabla f_{k+1}^T p_{k+1} = -\|\nabla f_{k+1}\|^2 + \beta_{k+1}^{FR} \nabla f_{k+1}^T p_k < 0$.
  - Exact l.s.: α_k is a local minimiser along p_k $\Rightarrow$ $\nabla f_{k+1}^T p_k = 0$ $\Rightarrow$ p_{k+1} is descent
  - Inexact l.s.: p_{k+1} is descent if α_k satisfies the strong Wolfe conditions (lemma 5.6):

$$\left.\begin{aligned} f(x_k + \alpha_k p_k) &\le f(x_k) + c_1 \alpha_k \nabla f_k^T p_k \\ |\nabla f(x_k + \alpha_k p_k)^T p_k| &\le c_2 |\nabla f_k^T p_k| \end{aligned}\right\} \quad 0 < c_1 < c_2 < \frac{1}{2}$$

(note we required a looser $0<c_1<c_2<1$ in ch. 3)

- <u>The Polak – Ribière method</u>
  - Differs in the parameter β_k, defined as $\beta_{k+1}^{PR} = \dfrac{\nabla f_{k+1}^T (\nabla f_{k+1} - \nabla f_k)}{\|\nabla f_k\|^2}$.
  - For strongly convex quadratic functions and exact l.s., $\beta_k^{PR} = \beta_k^{FR} = \beta_k$ for linear CG (since the successive gradients are mutually $\perp$).
  - For nonlinear functions in general, with inexact l.s., PR is empirically more robust and efficient than FR.
  - The strong Wolfe conditions don't guarantee that p_k is a descent direction.

- <u>Restarts</u>: restarting the iteration every n steps (by setting $\beta_k = 0$, i.e., taking a steepest descent step) periodically refreshes the algorithm and works well in practice. It leads to <u>n-step quadratic convergence</u>: $\dfrac{\|x_{k+n} - x^*\|}{\|x_k - x^*\|^2} \le M$ intuitively because near the minimum, f is approx. quadratic and so after a restart we will have (approximately) the linear CG method (which requires p_0 = steepest descent).
  - For large n (when CG is most useful) restarts may never occur, since an approximate solution may be found in less than n steps.

- <u>Global convergence:</u>
  - With restarts and ~~global convergence~~ the strong Wolfe conditions, the algorithms (FR, PR) have global convergence since they include as a subsequence the steepest descent method (which is globally convergent with the Wolfe conditions).
  - Without restarts
    - FR has global convergence with the strong Wolfe conditions above
    - PR does not have global convergence, even though in practice it is better.
  - In general, the theory on the rate of convergence of CG is complex and assumes exact l.s.

REVIEW OF CONJUGATE GRADIENT METHODS

* Linear CG: $A_{n\times n}$ sym. pd: solves $Ax = b \Leftrightarrow \phi(x) = \min \frac{1}{2}x^TAx - b^Tx$

- $\{p_0, \dots, p_{n-1}\}$ conjugate wrt $A \Leftrightarrow p_i^TAp_j = 0\ \forall i j$, $p_i \neq 0\ \forall i$
- Finds the solution in at most n steps, each an exact line search along a conjugate direction: $x_{k+1} = x_k + \alpha_k p_k$, $\alpha_k = \frac{r_k^Tr_k}{p_k^TAp_k}$, $r_k = \nabla\phi(x_k) = Ax_k - b$
- At each step, x_k is the minimiser over the set $x_0 + \text{span}(p_0, \dots, p_{k-1})$; $r_{k+1} = r_k + \alpha_k A p_k$; and $r_k^T p_i = r_k^T r_i = 0\ \forall i < k$.
- Conjugate direction p_k is obtained from the previous one and the current gradient: $p_k = -r_k + \beta_k p_{k-1}$ with $\beta_k = \frac{r_{k+1}^T r_{k+1}}{r_k^T r_k}$.
- Initial direction is the steepest descent direction: $p_0 = -\nabla\phi(x_0)$.
- Space complexity $\Theta(n)$, time complexity $O(n^3)$
  But often (eg. when the eigenvalues of A are clustered, or A has low c.n.) it gets very close to the solution in $r \ll n$ steps, so $\Theta(rn^2)$.

* Nonlinear CG: solves $\min f(x)$ where f is nonlinear in general

- Fletcher-Reeves: $r_k = \nabla f(x_k)$, α_k is determined by an inexact l.s. satisfying the strong Wolfe conditions, $\beta_{k+1}^{FR} = \frac{\nabla f_{k+1}^T \nabla f_{k+1}}{\nabla f_k^T \nabla f_k}$.

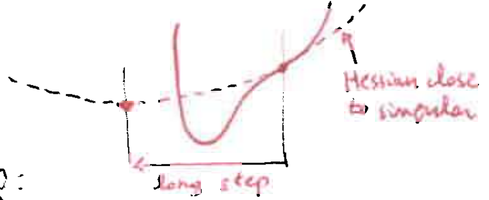
  Has global convergence, but to work well in practice it needs restarts (ie, set $\beta_k = 0$ every n steps).
- Polak-Ribière: like FR but $\beta_{k+1}^{PR} = \frac{\nabla f_{k+1}^T(\nabla f_{k+1} - \nabla f_k)}{\nabla f_k^T \nabla f_k}$.

  Works better than FR in practice, even though it has no global convergence.

* In summary: better method than steepest descent, very useful for large n (little storage).

CH. 6: PRACTICAL NEWTON METHODS

- Newton step: solution of the linear system $\overset{n\times n}{\nabla^2 f(x_k)}\, p_k^N = -\nabla f(x_k)$
  - Computing the Hessian is a major task, $\mathcal{O}(n^2)$.
  - Near a minimiser the Hessian is pd $\Rightarrow$ quadratic convergence (with unit steps $\alpha_k = 1$). Away from a minimiser the Hessian may not be pd or may be close to singular $\Rightarrow$ p_k^N may be an ascent direction or too long. A too long direction may not be good even if it is a descent direction, because it violates the spirit of Newton's method (which relies on a quadratic approximation valid near the current iterate); thus, it may require many iterations in the line search.



  - Need to solve the system efficiently.
- Robustness: two strategies to ensure a good quality step:
  - Newton-CG method: solve the system with the CG method, terminating if negative curvature is encountered; can be implemented as line search or trust region.
  - Modified Newton method: modify the Hessian to make it sufficiently p.d.
- Efficiency:
  - Newton-CG: inexact Newton step, ie, terminate the CG iteration before an exact solution is found (don't want to spend much effort in the subproblem, as in the l.s.)
  - Modified Newton: take advantage of sparsity structure (if available) in the Hessian.
- In general, these methods become (approximately) the pure Newton step if the Hessian is pd, and otherwise they find a descent direction in some heuristic way.

✳ Inexact Newton steps

Terminate the iterative solver (CG) when the residual $r_k = \nabla^2 f(x_k)\, p_k + \nabla f(x_k)$ (where p_k is the inexact Newton step) is small wrt the gradient (to achieve invariance wrt scalings of f): $\|r_k\| \leq \eta_k \|\nabla f(x_k)\|$ where $0 < \eta_k \leq \eta < 1$ $\forall k$ and $\{\eta_k\}$ is the <u>forcing sequence</u>.

$$\text{Rate of convergence} \begin{cases} \eta_k \to 0: \text{ superlinear}, & \text{eg } \eta_k = \min(0.5, \sqrt{\|\nabla f(x_k)\|}) \\ \eta_k = \mathcal{O}(\|\nabla f(x_k)\|): \text{ quadratic}, & \text{eg } \eta_k = \min(0.5, \|\nabla f(x_k)\|) \end{cases}$$

but the smaller η_k, the more iterations of CG we need

✳ Newton-CG method

$\begin{cases} A \text{ nd: seeks a maximiser because } p_k^T A p_k < 0 \Rightarrow \alpha_k < 0 \text{ } (p_k \text{ remains descent}) \\ A \text{ not def: seeks a saddle, but some directions } (p_k^T A p_k = 0) \text{ have no stationary point} \end{cases}$

We solve the system with the CG method, terminating if negative curvature is encountered ($p_k^T A p_k \leq 0$); if the first direction is of negative curvature, use the steepest descent direction instead. Then, we use the resulting direction:

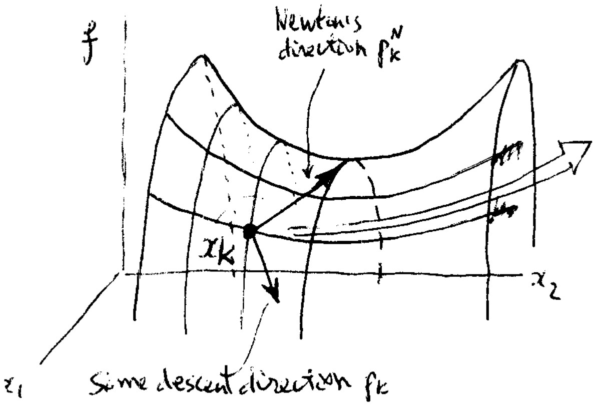
- For a line search (inexact, with appropriate conditions: Wolfe, Goldstein; or Armijo backtracking line search). Problem: if the Hessian is nearly singular the Newton-CG direction can be very long.
- In a trust-region way: limit the line search to a region of size Δ_k.

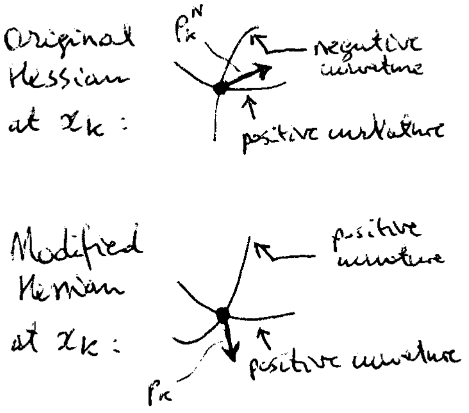
The method behaves like pure Newton for pd Hessian, like steepest descent for nd Hessian, and finds some descent direction for not definite Hessian.

* Modified Newton method

We solve (by factorising B_k, eg. Cholesky factorisation) the system $B_k p_k = -\nabla f(x_k)$ where $B_k = \nabla^2 f(x_k) + \lambda I$ (modified Hessian) with $\lambda \geq 0$ large enough that B_k is sufficiently pd. Then we use p_k (which is a descent direction because B_k is pd) in a line search with Wolfe conditions

The method behaves like pure Newton for pd Hessian and $\lambda = 0$, like steepest descent for $\lambda \to \infty$, and finds some descent direction for intermediate λ:





We want λ as small as possible to preserve Hessian information along the positive curvature directions; but if λ is too small, B_k is nearly singular and the step too long.

- Other types of Hessian modification exist, but there is no consensus about which one is best
- Global convergence ($\|\nabla f_k\| \to 0$) if $\kappa(B_k) = \|B_k\| \|B_k^{-1}\| \leq M$ (ch. 3)
- Quadratic convergence near the minimiser, where the Hessian is pd.

* Trust-region Newton methods

Don't need the Hessian to be pd. See ch. 4: min $m_k(p) = f_k + \nabla f_k^T p + \frac{1}{2} p^T B_k p$ s.t. $\|p\| \leq \Delta_k$; with $B_k = \nabla^2 f(x_k)$. One algorithm: find $\lambda \geq 0$ such that the conditions of th. 4.3 hold $\Rightarrow$ 1D root-finding for λ, but requires solving $(B_k + \lambda I) p^* = -\nabla f_k$ (by factorising $B_k + \lambda I$).

Have global convergence; and under some conditions (th. 6.4), if using $B_k = \nabla^2 f(x_k)$ (which is pd near the minimiser) then the trust region size Δ_k becomes inactive for all k sufficiently large, so quadratic convergence.

REVIEW OF PRACTICAL NEWTON METHODS

- Pure Newton step: approximate f quadratically with true Hessian, jump directly to minimiser: $\nabla^2 f_k \, p_k^N = -\nabla f_k$.
- Great convergence rate (quadratic) but:
  - Computing the Hessian is hard, $\mathcal{O}(n^2)$
  - Solving for p_k^N is hard, $\mathcal{O}(n^3)$
  - Hessian may not be sufficiently pd, so p_k^N may be ascent or too long
- Modifications of the pure Newton method:
  - Inexact Newton steps: approximate solution of the system (use linear conjugate gradient method, stop before n steps).
  - Newton-CG method: solve system with CG, stop if negative curvature
  - Modified Newton method: use $B_k = \nabla^2 f_k + \lambda I$ instead of the Hessian with $\lambda \geq 0$ large enough that B_k is sufficiently pd. $\lambda = \begin{cases} 0: & \text{pure Newton} \\ \infty: & \text{steepest descent} \end{cases}$
  - Trust-region Newton method: pure Newton step subject to $\|p_k^N\| \leq \Delta_k$.
- Global convergence under some conditions
- Quadratic rate near the minimiser (where the Hessian will be pd).

CH. 7: CALCULATING DERIVATIVES

Approximate or automatic techniques to compute the gradient, Hessian or Jacobian if difficult by hand

* Finite-difference derivative approximations

Example: $f: \mathbb{R}^n \rightarrow \mathbb{R}$, $$\frac{\partial f}{\partial x_i} = \begin{cases} \frac{f(x+\varepsilon e_i) - f(x)}{\varepsilon} + \mathcal{O}(\varepsilon) \leftarrow \text{Forward difference} \\ \underbrace{\frac{f(x+\varepsilon e_i) - f(x-\varepsilon e_i)}{2\varepsilon}}_{\text{Approximate the derivative with this}} + \mathcal{O}(\varepsilon^2) \leftarrow \text{Central difference} \end{cases}$$

[Proof: Taylor's th.]

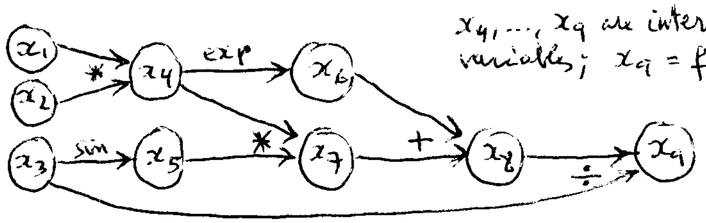
- Needs careful choice of ε: as small as possible but not too close to the machine precision (to avoid roundoff errors). As a rule of thumb, $\varepsilon \simeq \sqrt{u}$ for forward diff. and $\varepsilon \simeq u^{2/3}$ for central diff., where u ($= 10^{-16}$ in double precision) is the unit roundoff.
- The gradient requires $\left\{ \begin{array}{l} \text{forward diff: } n+1 \\ \text{central diff: } 2n \end{array} \right\}$ function evaluations.
- The Hessian requires $\mathcal{O}(n^2)$ function evaluations, less if sparse.
- Also useful to check whether a gradient calculated by hand is correct.

* Automatic differentiation

- Build a computational graph of f using intermediate variables
- Apply the chain rule: $\nabla_x h(y(x)) = \sum_i \frac{\partial h}{\partial y_i} \nabla y_i(x)$.

Example:

$$f(x) = \frac{x_1 x_2 \sin x_3 + e^{x_1 x_2}}{x_3} \Longrightarrow$$



$x_4, \dots, x_9$ are intermediate variables; $x_9 = f$

recursively

$$\nabla x_7 = \frac{\partial x_7}{\partial x_5} \nabla x_5 + \frac{\partial x_7}{\partial x_4} \nabla x_4 = x_4 \nabla x_5 + x_5 \nabla x_4 \; ; \; \nabla x_5 = \frac{\partial x_5}{\partial x_3} \nabla x_3 = \cos x_3 \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}.$$

directly

- Compute the values of f, ∇f recursively
- Done automatically by a software tool.

- Disadvantage: simplification of expressions, reuse of operations; eg differentiate $\tan x - x$, $\ln\left(\frac{x-1}{x+1}\right)$, $\frac{1}{1+e^{-ax}}$.

* Symbolic differentiation

- Produce an algebraic expression for the gradient, etc. Packages: Mathematica, Maple ...

CH. 8: QUASI-NEWTON METHODS

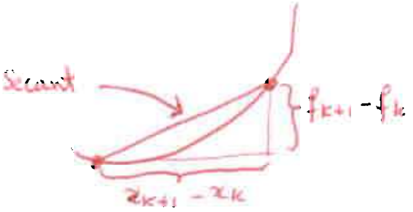
- Like steepest descent and conjugate gradients, they require only the gradient.
- By measuring the changes in gradients over iterations, they construct an approximation to the Hessian whose accuracy improves gradually and results in superlinear convergence.
- Idea (from Taylor's th.): $\nabla^2 f_{k+1} \underbrace{(x_{k+1} - x_k)}_{s_k} \simeq \underbrace{\nabla f_{k+1} - \nabla f_k}_{y_k}$ → Exact if f is quadratic
- Line search $x_{k+1} = x_k + \alpha_k p_k$ with step length chosen to satisfy the Wolfe conditions.
- Quadratic model of the objective function (correct to first order):

$$m_k(p) = f_k + \nabla f_k^T p + \frac{1}{2} p^T B_k p$$

where B_k is symmetric pd and is updated at every iteration.

- The search direction is given by the minimiser of m_k, $p_k = -B_k^{-1} \nabla f_k$ (which is a descent direction) ⇒ like Newton's method with B_k instead of the Hessian.
→ why?

* The DFP method (Davidon – Fletcher – Powell)



B_{k+1} is chosen to satisfy the following conditions:

① $\nabla m_{k+1} = \nabla f$ at x_k and x_{k+1} ⇒ $\nabla m_{k+1}(-\alpha_k p_k) = \nabla f_{k+1} - \alpha_k B_{k+1} p_k = \nabla f_k$

⇒ $\boxed{B_{k+1} s_k = y_k}$ (<u>secant equation</u>)

Note that this implicitly requires that $s_k^T y_k = s_k^T B_k s_k > 0$ (a curvature condition). If f is strongly convex, this is guaranteed (because $s_k^T y_k > 0$ for any two points x_k and x_{k+1}; proof: exercise 8.1). Otherwise, it is guaranteed if the line search verifies the 2nd Wolfe condition $\nabla f(x_k + \alpha_k p_k)^T p_k \geq c_2 \nabla f_k^T p_k$, $0 < c_2 < 1$ (proof: 2nd Wolfe ⇔ $\nabla f_{k+1}^T s_k \geq c_2 \nabla f_k^T s_k$ ⇔ $y_k^T s_k \geq (c_2 - 1)\alpha_k \nabla f_k^T p_k > 0$).

② The secant equation has many solutions; we choose the closest to the current matrix B_k: $\min_B \|B - B_k\|$ s.t. B symmetric pd, $B s_k = y_k$.
Different choices of norm are possible; one that allows an easy solution and gives rise to scale invariance is the weighted Frobenius norm $\|A\|_W = \|W^{\frac{1}{2}} A W^{\frac{1}{2}}\|_F$ (where $\|A\|_F^2 = \sum_{ij} a_{ij}^2$). W is any matrix satisfying $W y_k = s_k$ (thus the norm is adimensional, i.e., the solution doesn't depend on the units of the problem)

If we choose $W^{-1} = \int_0^1 \nabla^2 f(x_k + \tau\alpha_k p_k)\, d\tau$ (the average Hessian) then the minimiser is unique and is the following rank-2 update:

$$\text{DFP:} \begin{cases} B_{k+1} = (I - \gamma_k y_k s_k^T)\, B_k\, (I - \gamma_k s_k y_k^T) + \gamma_k y_k y_k^T & \text{with } \gamma_k = \dfrac{1}{y_k^T s_k} \\ H_{k+1} = H_k - \dfrac{H_k y_k y_k^T H_k}{y_k^T H_k y_k} + \dfrac{s_k s_k^T}{y_k^T s_k} & \end{cases}$$

$\to y_k^T s_k > 0$ (curvature condition required earlier)

[proof: Sherman-Morrison-Woodbury formula]

For $\boxed{H_k = B_k^{-1}}$: using B_k directly requires solving $p_k = -B_k^{-1} \nabla f_k$ which is $O(n^3)$, while using H_k gives ~~us~~ $p_k = -H_k \nabla f_k$ which is $O(n^2)$.

* The BFGS method (Broyden - Fletcher - Goldfarb - Shanno)

We apply the conditions to $\overset{=B_{k+1}^{-1}}{H_{k+1}}$ rather than B_{k+1}: $\min_H \|H - H_k\|$ s.t. H symmetric pd, $H y_k = s_k$ with the same norm as before where $W s_k = y_k$. For W = the average Hessian we obtain:

$$\text{BFGS:} \begin{cases} H_{k+1} = (I - \rho_k s_k y_k^T)\, H_k\, (I - \rho_k y_k s_k^T) + \rho_k s_k s_k^T & \text{with } \rho_k = \dfrac{1}{y_k^T s_k} \\ B_{k+1} = B_k - \dfrac{B_k s_k s_k^T B_k}{s_k^T B_k s_k} + \dfrac{y_k y_k^T}{y_k^T s_k} & \end{cases}$$

[proof: SMW formula]

We have: H_k pd $\Rightarrow$ H_{k+1} pd [proof: $z^T H_{k+1} z > 0 \ \forall z \neq 0$]

why do many methods work well with quadratic f?

We take the initial matrix as $H_0 = I$ for lack of better knowledge.

- For quadratic f and if an exact line search is performed, then DFP, BFGS, SR1 converge to the exact minimizer in n steps and $H_n = \nabla^2 f$.
- BFGS is the best quasi-Newton method. With an adequate line search (eg. Wolfe conditions), BFGS has effective self-correcting properties (and DFP is not so effective): a poor approximation to the Hessian will be improved in a few steps, thus being stable wrt roundoff error.

Algorithm 2.1 (BFGS): given starting point x_0, convergence tolerance $\varepsilon > 0$, $H_0 \leftarrow I$, $k \leftarrow 0$

```
while $\|\nabla f_k\| \geq \varepsilon$
    $p_k \leftarrow -H_k \nabla f_k$                                   [search direction]
    $x_{k+1} \leftarrow x_k + \alpha_k p_k$                           [line search with Wolfe cond.]
    $s_k \leftarrow x_{k+1} - x_k$,  $y_k \leftarrow \nabla f_{k+1} - \nabla f_k$,  $H_{k+1} \leftarrow$ BFGS update
    $k \leftarrow k+1$
end
```

- Always try $\alpha_k = 1$ first in the line search (this step will always be accepted eventually). Empirically, good values for c_1, c_2 in the Wolfe conditions are: $c_1 = 10^{-4}$, $c_2 = 0.9$.

- Cost per iteration:
  - Space: $\mathcal{O}(n^2)$ matrix storage. For large problems, techniques exist to modify(*) the method to take less space, though converging more slowly (see ch. 9).
  - Time: $\mathcal{O}(n^2)$ matrix × vector, outer products
- Global convergence if B_k have a bounded condition number + Wolfe conditions (see ch 3); but in practice this assumption may not hold. There aren't truly global convergence results, though the methods are very robust in practice.

	Newton	Quasi-Newton
Convergence rate	quadratic	superlinear
cost per iteration (time)	$\mathcal{O}(n^3)$ linear system	$\mathcal{O}(n^2)$
$\nabla^2 f$ required	yes	no

* The SR1 method (symmetric, rank-1)

$$\underline{SR1}: \begin{cases} B_{k+1} = B_k + \dfrac{(y_k - B_k s_k)(y_k - B_k s_k)^T}{(y_k - B_k s_k)^T s_k} \\[2ex] H_{k+1} = H_k + \dfrac{(s_k - H_k y_k)(s_k - H_k y_k)^T}{(s_k - H_k y_k)^T y_k} \end{cases}$$

- Generates very good Hessian approximations, often better than BFGS, but:
  - Does not necessarily preserve pd ⇒ use in trust-region (not l.s.) framework
  - May not satisfy the secant equation $y_k = B_{k+1} s_k$, $s_k = H_{k+1} y_k$ if $(y_k - B_k s_k)^T s_k = 0$ ⇒ skipping the update if the denominator is small works well in practice:
    if $|s_k^T (y_k - B_k s_k)| < r \|s_k\| \|y_k - B_k s_k\|$ then $B_{k+1} = B_k$ else $B_{k+1} = SR1$ [use $r \sim 10^{-8}$]

* The Broyden class

$$B_{k+1} = (1-\phi_k) B_{k+1}^{BFGS} + \phi_k B_{k+1}^{DFP} \quad \text{for } \phi_k \in \mathbb{R}$$

- Generalizes BFGS, DFP and SR1.
- Symmetric
- Preserves pd for $\phi_k \in [0,1]$
- Satisfies the secant equation

(*) – Represent H_k in terms of a few $n \times 1$ vectors
– Take advantage of sparsity in the Hessian
– Exploit "partial separability" of f, eg. $f(x) = f_1(x_1, x_3) + f_2(x_2, x_4, x_6) + f_3(x_5)$

REVIEW OF QUASI-NEWTON METHODS

- Newton's method with an approximate Hessian:
  - Quadratic model of objective function f: $m_k(p) = f_k + \nabla f_k^T p + \frac{1}{2} p^T B_k p$
  - Search direction is the minimiser of m_k: $p_k = -B_k^{-1} \nabla f_k$.
  - Inexact line search with Wolfe conditions; always try $\alpha_k = 1$ first.
  - B_k is symmetric pd and is updated at each k given the current and previous gradients, so that it approximates $\nabla^2 f_k$.
- Idea: $\nabla^2 f_{k+1} (\underbrace{x_{k+1} - x_k}_{s_k}) \simeq \underbrace{\nabla f_{k+1} - \nabla f_k}_{y_k}$ (by Taylor's th.)

<u>Secant equation</u>: $B_{k+1} s_k = y_k$; implies $\nabla m_{k+1} = \nabla f$ at x_k, x_{k+1}.

- <u>DFP method (Davidon-Fletcher-Powell)</u>: B_{k+1} satisfies the secant equation and is closest to B_k (in a precise sense) $\Rightarrow$ rank-2 update

$$H_{k+1} (= B_{k+1}^{-1}) = H_k - \frac{H_k y_k y_k^T H_k^T}{y_k^T H_k y_k} + \frac{s_k s_k^T}{y_k^T s_k} \Rightarrow p_k = -H_k \nabla f_k \text{ in } \mathcal{O}(n^2)$$

- <u>BFGS method (Broyden-Fletcher-Goldfarb-Shanno)</u>: H_{k+1} $(= B_{k+1}^{-1})$ satisfies the secant eq. and is closest to H_k (in a precise sense) $\Rightarrow$ rank-2 update

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

Use $H_0 = I$.

Best quasi-Newton method, self-correcting properties.

- <u>SR1 method (symmetric rank-1)</u>: rank-1 update

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

- <u>Global convergence</u>: no general results, though the methods are robust in practice

<u>Convergence rate</u>: superlinear

		Newton	Quasi-Newton
convergence rate		quadratic	superlinear
cost per iteration	time	$\mathcal{O}(n^3)$	$\mathcal{O}(n^2)$
	space	$\mathcal{O}(n^2)$	$\mathcal{O}(n^2)$
Hessian required		yes	no

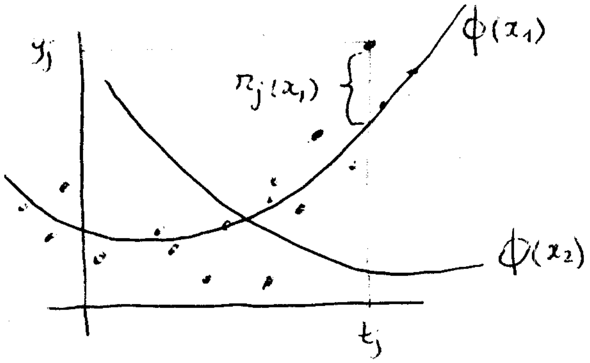
CH. 10: NONLINEAR LEAST-SQUARES PROBLEMS

- Least-squares (LSQ) problem: $f(x) = \frac{1}{2}\sum_{j=1}^{m} r_j^2(x)$ where the residuals $r_j : \mathbb{R}^n \to \mathbb{R}$, $j=1,\dots,m$ are smooth and $m \geq n$.

- Arise very often in practice when fitting a parametric model to observed data; $r_j(x)$ is the error for datum j with model parameters x; "min f" means finding the parameter values that best match the model to the data.

- Example: regression (curve fitting): $r(x) = \frac{1}{2}\sum_{j=1}^{m} (y_j - \phi(x;t_j))^2$ is the LSQ error of fitting curve $\phi : t \to y$ (with parameters x) to the observed data points $\{(t_j, y_j)\}_{j=1}^{m}$.

  If using other norms, eg. $|r_j|$ or $|r_j|^3$, it won't be a LSQ problem.



- The special form of f simplifies the minimisation problem. Write $f(x) = \frac{1}{2}\|r(x)\|_2^2$ in terms of the residual vector $r : \mathbb{R}^n \to \mathbb{R}^m$

$$r(x) = \begin{pmatrix} r_1(x) \\ \vdots \\ r_m(x) \end{pmatrix} \quad \text{with Jacobian} \quad J(x) = \left(\frac{\partial r_j}{\partial x_i}\right)_{\substack{j=1,\dots,m \\ i=1,\dots,n}} \quad \left(\begin{matrix} m \times n \text{ matrix of first} \\ \text{partial derivatives} \end{matrix}\right)$$

Usually it's easy to compute J explicitly. Then

$$\nabla f(x) = \sum_j r_j(x)\nabla r_j(x) = J(x)^T r(x)$$

$$\nabla^2 f(x) = \sum_j \nabla r_j(x)\nabla r_j(x)^T + \sum_j r_j(x)\nabla^2 r_j(x) = \overbrace{J(x)^T J(x)}^{\circledast} + \sum_j r_j(x)\nabla^2 r_j(x)$$

Often $\circledast$ is the leading term, eg

- if r_j are small around the solution ("small-residual case")
- if r_j are approximately linear around the solution

So we get a pretty good approximation of the Hessian for free.

- Linear LSQ problems: $r_j(x)$ is linear $\forall j \Rightarrow J(x) = J$ constant. Calling $r = r(0)$ we have $f(x) = \frac{1}{2}\|Jx + r\|_2^2$ convex, $\nabla f(x) = J^T(Jx + r)$, $\nabla^2 f(x) = J^T J$ constant.

  ↳ why?

- Minimiser: $\nabla f(x^*) = 0 \Rightarrow J^TJ x^* = -Jr$, the <u>normal equations</u>: $n \times n$ linear system with pd or psd matrix which can be solved with numerical analysis techniques (see book; also could use the linear conjugate gradient method for large n).
- For <u>nonlinear LSQ problems</u> f isn't necessarily convex. We see 2 methods (Gauss-Newton, Levenberg-Marquardt) which take advantage of the particular form of LSQ problems; but any of the methods we have seen in earlier chapters are applicable too (eg. Newton's method, if we compute $\nabla^2 r_j$).

* <u>Gauss-Newton method</u>

- Line search with Wolfe conditions and a modification of Newton's method: instead of generating the search direction p_k by solving the Newton equations $\nabla^2 f(x_k) p = -\nabla f(x_k)$, we ignore the second-order term from $\nabla^2 f$ (i.e., approximate $\nabla^2 f_k \simeq J_k^T J_k$) and solve $J_k^T J_k \, p_k^{GN} = -J_k^T r_k$.
- Equivalent to approximating $r(x)$ by a linear model $r(x+p) \approx r(x) + J(x)p$ (<u>linearisation</u>) and so $f(x)$ by a quadratic model with Hessian $J(x)^T J(x)$, then solving the linear LSQ problem $\min_p \frac{1}{2} \| J_k p + r_k \|_2^2$.

why? Evaluate $(p_k^{GN})^T \nabla f_k < 0$

- If J_k has full rank and $\nabla f_k = J_k^T r_k \neq 0$ then p_k^{GN} is a descent direction.
- Saves us the trouble of computing the individual Hessians $\nabla^2 r_j$.
- <u>Global convergence</u> if Wolfe conditions + $\| J(x) z \|_2 \geq \gamma \| z \|_2$ in the region of interest + technical condition (th. 10.1). [Proof: $\cos\theta_k$ is bounded away from 0 + Zoutendijk's th.]

⇔ Singular values of J are away from 0 ⇔ J^TJ is well conditioned (see ch. 3)

The theorem doesn't hold if $J(x_k)$ is rank-deficient for some k. This occurs when the normal equations are undetdetermined (infinite number of solutions for p_k^{GN}).

- <u>Rate of convergence</u> depends on how much the term J^TJ dominates the second-order term in the Hessian; it is linear but rapid in the small-residual case.

* <u>Levenberg-Marquardt method</u>

- Same modification of Newton's method as in the Gauss-Newton method but with a trust region instead of a line search.

- Spherical trust region with radius Δ_k, quadratic model for f with Hessian $J_k^T J_k$:

$$m_k(p) = \frac{1}{2}\|r_k\|^2 + p^T J_k^T r_k + \frac{1}{2} p^T J_k^T J_k p = \frac{1}{2}\|J_k p + r_k\|_2^2$$

$$\Rightarrow \min_p \frac{1}{2}\|J_k p + r_k\|_2^2 \quad \text{s.t.} \quad \|p\| \le \Delta_k$$

- A rank-deficient Jacobian is no problem because the step length is bounded by Δ_k

- Characterisation of the solution of the trust-region subproblem (lemma 10.3, direct consequence of th. 4.3 in ch. 4): p^{LM} is a solution of the trust-region problem $\min_{\|p\| \le \Delta} \|Jp + r\|_2^2$ iff $\exists \lambda \ge 0$ such that:

  a) $(J^T J + \lambda I)\, p^{LM} = -J^T r$

  b) $\lambda(\Delta - \|p^{LM}\|) = 0$

  Search for λ: start with large λ, reduce it till the corresponding p^{LM} from a) produces a sufficient decrease (defined in some way) in f.

- Global convergence under certain assumptions.

- Rate of convergence similar to Gauss-Newton, since both methods ignore the second-order component of the Hessian. Near a solution with small residuals, the trust region eventually becomes inactive and the algorithm takes Gauss-Newton steps.

* Large-residual problems

If the residuals $r_j(x^*)$ near the solution x^* are large, both Gauss-Newton and Levenberg-Marquardt converge slowly, since $J^T J$ is a bad model of the Hessian. In that case it is better to use a Newton or quasi-Newton method.

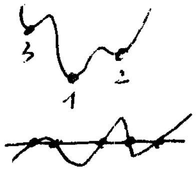
REVIEW OF NONLINEAR LEAST-SQUARES PROBLEMS

- $f(x) = \frac{1}{2}\sum_{j=1}^{m} r_j^2(x)$ where $m \geq n$ and residual $r_j : \mathbb{R}^n \to \mathbb{R}$ is the error at datum j for a model with parameters x.
- Simplified form for gradient and Hessian:
  - $f(x) = \frac{1}{2}\|r(x)\|_2^2$ with $r(x) = (r_j(x))_j$
  - $\nabla f(x) = J(x)^T r(x)$ with Jacobian $J(x) = \left(\frac{\partial r_j}{\partial x_i}\right)_{ji}$
  - $\nabla^2 f(x) = \underbrace{J(x)^T J(x)}_{\text{use this as approximate Hessian}} + \sum_j r_j(x)\, \nabla^2 r_j(x)$
- Linear LSQ: r_j linear, J constant, the minimiser x^* satisfies (calling $r = r(0)$) the normal equations $\underbrace{J^T J}_{\text{pd or psd}} x^* = -J r$
- Nonlinear LSQ: GN, LM methods
- Gauss-Newton method:
  - Approximate Hessian $\nabla^2 f_k \simeq J_k^T J_k$, solve for the search direction $J_k^T J_k p_k^{GN} = -J_k^T r_k$; inexact line search with Wolfe conditions.
  - Equivalent to linearising $r(x+p) \simeq r(x) + J(x)p$
  - Problems if J_k is rank-defective
- Levenberg-Marquardt method:
  - Like GN but with trust region instead of line search: $\min_{\|p\| \leq \Delta_k} \|J_k p + r_k\|_2^2$.
  - No problem if J_k is rank-defective
  - One way to solve the trust-region subproblem approximately: try large $\lambda \geq 0$, solve $(J_k^T J_k + \lambda I)\, p_k^{LM} = -J_k^T r_k$, accept p_k^{LM} if sufficient decrease in f, otherwise try a smaller λ.
- Global convergence under certain assumptions.
- Rate of convergence: linear but rapid if small residuals

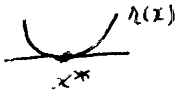
Using $J(x)^T J(x)$ instead of $\nabla^2 f(x)$ works well if small residuals ($r_j(x) \simeq 0$) or quasilinear residuals ($\nabla^2 r_j(x) \simeq 0$). Otherwise the GN, LM methods are slow; try others instead (quasi-Newton, etc.).

CH. 11: NONLINEAR EQUATIONS

- Problem: find roots of the equation $\boxed{r(x)=0}$ where $r: \mathbb{R}^n \to \mathbb{R}^n$.
- Many similarities with optimisation: Newton's method, line search, trust region...
- Differences:
  - In optimisation, the local optima can be ranked by the objective value. In root-finding, all roots are equally good.
  - For quadratic convergence we need derivatives of order $\begin{cases} \text{optimisation: } 2 \\ \text{root-finding: } 1. \end{cases}$
  - Quasi-Newton methods are less useful in root-finding.



- Assume the $n \times n$ Jacobian $J(x) = \left(\dfrac{\partial r_i(x)}{\partial x_j}\right)_{ij}$ exists and is continuous in the region of interest.
- Degenerate root: x^* with $r(x^*)=0$ and $J(x^*)$ singular.



* Newton's method

- Taylor's th.: linearise $r(x+p) = r(x) + J(x)p + O(\|p\|^2)$ and use as model; find its root.

Algorithm 11.1:

```
given $x_0$;
for $k = 0, 1, 2 \ldots$
        solve $J_k p_k = -r_k$
        $x_{k+1} \leftarrow x_k + p_k$
end
```

- Newton's method for optimising an objective function f is the same as applying this algorithm to $r(x) = \nabla f(x)$.
- Convergence rate for nondegenerate roots $\begin{cases} \text{- Jacobian continuous: superlinear} \\ \text{- Jacobian Lipschitz continuous: quadratic.} \end{cases}$
- Problems:
  - Degenerate roots, e.g. $r(x) = x^2$ produces $x_k = 2^{-k} x_0$ which converges linearly.
  - Not globally convergent: away from a root the algorithm can diverge or cycle; it is not even defined if $J(x_k)$ is singular.
  - Expensive to compute J and solve the system exactly for large n.

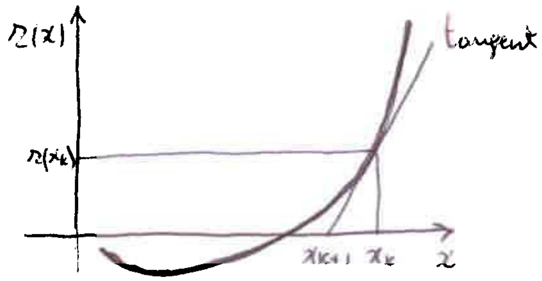
* Inexact Newton's method

- Exit the linear solver of $J_k p_k = -r_k$ when $\|r_k + J_k p_k\| \leq \eta_k \|r_k\|$ for $\eta_k \in [0, \eta]$ with constant $\eta \in (0,1)$. $\{\eta_k\}$ = forcing sequence (as for Newton's method in optimisation). We can't use linear conjugate gradients here because J_k isn't always pd; but there are other linear solvers (also based on Krylov subspaces, ie, iterative multiplication by J_k).

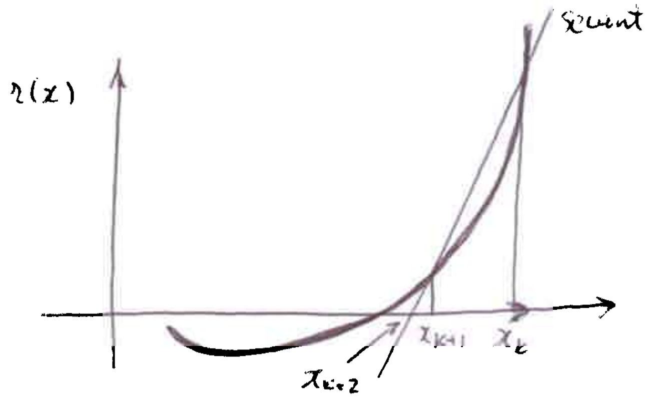
- Convergence rate to a nondegenerate root $\begin{cases} \text{linear, if } \eta \text{ sufficiently small} \\ \text{superlinear, if } \eta_k \to 0 \\ \text{quadratic, if } \eta_k = \mathcal{O}(\|r_k\|). \end{cases}$

* Broyden's method (secant or quasi-Newton method)

- Constructs an approximation to the Jacobian over iterations.
- write $s_k = x_{k+1} - x_k$, $y_k = r_{k+1} - r_k$: $y_k = J_k s_k + \mathcal{O}(\|s_k\|^2)$ (Taylor's th.).
- We require the updated Jacobian approximation B_{k+1} to satisfy the secant equation $y_k = B_{k+1} s_k$ and to minimise $\|B - B_k\|_2$, ie, the smallest possible update that satisfies the secant eq.: $B_{k+1} = B_k + \frac{(y_k - B_k s_k) s_k^T}{s_k^T s_k}$.
- Convergence rate: superlinear if the initial Jacobian B_0 is close to the Jacobian at the root $J(x^*)$; the latter can be crucial, but is difficult to guarantee in practice
- for a scalar equation $(n=1)$:



Newton's method: $x_{k+1} = x_k - \frac{r(x_k)}{r'(x_k)}$



Secant method: $x_K = x_k - \frac{r(x_k)}{B_k}$ with $B_k = \frac{r(x_k) - r(x_{k-1})}{x_k - x_{k-1}}$ independent of B_{k-1}.

* Practical methods:

- Line search and trust region techniques to ensure convergence away from a root.
- Merit functions: a function $f: \mathbb{R}^n \to \mathbb{R}$ that indicates how close x is to a root, so that by decreasing $f(x)$ we approach a root. In optimization, f is the objective function. In root-finding, there is no unique ~~wo~~ (and fully satisfactory) way to define a merit function. The most widely used is $f(x) = \frac{1}{2}\|r(x)\|_2^2$.
- Problem: each root $(r(x)=0)$ is a local minimiser of f but not vice versa, so local minima that are not roots can attract the algorithm.

If a local minimiser x is not a root then $J(x)$ is singular (pf. $\nabla f(x) = J(x)^T r(x) = 0 \xRightarrow{r(x) \neq 0} J(x)$ singular).

- <u>Line search methods</u>: $x_{k+1} = x_k + \alpha_k p_k$ with step length α_k along direction p_k.
- We want descent directions for r ($p_k^T \nabla f(x_k) < 0$); the step length is chosen as in ch. 3.
- Zoutendijk's th.: descent directions + Wolfe conditions + Lipschitz-continuous J $\Rightarrow$ $\sum_{k \geq 0} \cos^2\theta_k \, \|J_k^T r_k\|^2 < \infty$.
- $\Rightarrow$ if $\cos\theta_k \geq \delta$ for constant $\delta \in (0,1)$ and k sufficiently large $\Rightarrow \nabla f_k = J_k^T r_k \to 0$; and <u>if $\|J(x)^{-1}\|$ is bounded</u> $\Rightarrow r_k \to 0$.
- If well defined, the Newton step is a descent direction for f for $r_k \neq 0$ (pf. $p_k \nabla f_k = -p_k^T J_k^T r_k = -\|r_k\|^2 < 0$). But $\cos\theta_k = \frac{-p_k^T \nabla f_k}{\|p_k\| \, \|\nabla f_k\|} = \frac{\|r_k\|^2}{\|J_k^{-1} r_k\| \, \|J_k^T r_k\|} \geq \frac{1}{\|J_k^T\| \, \|J_k^{-1}\|} = \frac{1}{\kappa(J_k)}$, so a large condition number causes poor performance (search direction almost $\perp \nabla f_k$).
- One modification of Newton's direction is $(J_k^T J_k + \tau_k I) p_k = -J_k^T r_k$; a large enough τ_k ensures $\cos\theta_k$ is bounded away from 0 because $\tau_k \to \infty \Rightarrow p_k \propto -J_k^T r_k$.
- Inexact Newton steps do not compromise global convergence: if at each step $\|r_k + J_k p_k\| \leq \eta_k \|r_k\|$ for $\eta_k \in [0, \eta]$ and $\eta \in [0,1)$ then $\cos\theta_k \geq \frac{1-\eta}{2\kappa(J_k)}$.

- <u>Trust region methods</u>: $\min m_k(p) = f_k + \nabla f_k^T p + \frac{1}{2} p^T B_k p$ s.t. $\|p\| \leq \Delta_k$.
  - Algorithm 4.1 from ch. 4 applied to $f(x) = \frac{1}{2}\|r(x)\|_2^2$ using $B_k = J_k^T J_k$ as approximate Hessian in the model m_k, i.e., linearise $r(p) \simeq r_k + J_k p$.
  - The exact solution has the form $p_k = (-J_k^T J_k + \lambda_k I)^{-1} J_k^T r_k$ for some $\lambda_k \geq 0$, with $\lambda_k = 0$ if the unconstrained solution is in the trust region. The Levenberg-Marquardt algorithm searches for such λ_k.
  - Global convergence (to nondegenerate roots) under mild conditions.
  - Quadratic convergence rate if the trust region subproblem is solved exactly for all k sufficiently large.

* <u>Continuation / homotopy methods</u>

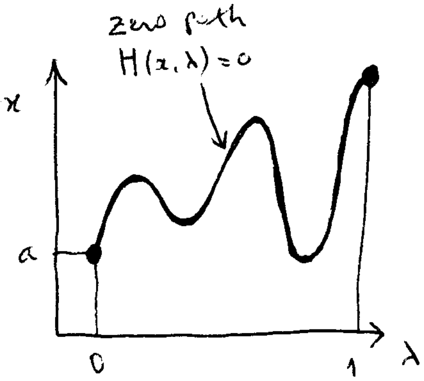
- Problem of Newton-based methods: unless J is nonsingular in the region of interest, they may converge to a local minimiser of the merit function rather than a root.
- <u>Continuation methods</u>: instead of dealing with the original problem $r(x) = 0$ directly, establish a continuous sequence of root-finding problems that converges to the original problem

but starts from an easy problem; then solve each problem in the sequence, tracking the root as ~~th~~ we moves from the easy to the original problem.

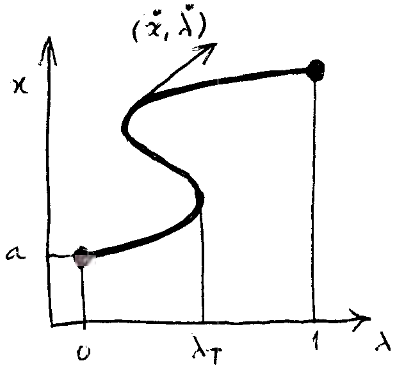
- Homotopy map: $H(x,\lambda) = \lambda r(x) + (1-\lambda)(x-a)$ where $a \in \mathbb{R}^n$ fixed and $\lambda \in \mathbb{R}$.

  $\lambda = \begin{cases} 0: & \text{easy problem with solution } x = a \\ 1: & \text{original problem.} \end{cases}$

  Start from $\lambda = 0$, $x = a$; gradually increase λ from 0 to 1 and solve $H(x,\lambda) = 0$ using as initial x the one from the previous λ value; stop after solving for $\lambda = 1$.



Easy case



Hard case

At the turning point λ_T, if we increase λ we lose track of the root x. To follow the zero path smoothly we need to allow λ to decrease and even to roam outside $[0,1]$.

- Arc-length parameterisation of the homotopy path: $(x(s), \lambda(s))$ where s = arc length measured from $(a,0)$ at $s=0$. Since $H(x(s),\lambda(s)) = 0$ $\forall s \geq 0$, its total derivative wrt s is also 0: $\frac{dH}{ds} = \frac{\partial}{\partial x} H(x,\lambda)\,\dot{x} + \frac{\partial}{\partial \lambda} H(x,\lambda)\,\dot{\lambda} = 0$, where $(\dot{x},\dot{\lambda}) = \left(\frac{dx}{ds}, \frac{d\lambda}{ds}\right)$ is the tangent vector to the zero path. To calculate the tangent vector at a point (x,λ) notice that:

  1) $(\dot{x},\dot{\lambda})$ lies in the nullspace of the $n \times (n+1)$ matrix $\left(\frac{\partial H}{\partial x} \;\; \frac{\partial H}{\partial \lambda}\right)$; the null space is 1D if this matrix is full rank.
  2) Its length is $1 = \|\dot{x}(s)\|^2 + |\dot{\lambda}(s)|^2$ $\forall s \geq 0$ (s is arc length, so unit speed).
  3) We need to choose the correct sign to ensure we move forward along the path; a heuristic that works well is to choose the sign so that the current tangent vector makes an angle of less than $\pi/2$ with the previous tangent.

Following the path can now be done by solving an initial-value first-order ODE: $\frac{dH}{ds} = 0$ for $s \geq 0$ with $H(0) = (a,0)$; terminating at an s for which $\lambda(s) = 1$.

- The tangent vector is well defined if the matrix $\left(\frac{\partial H}{\partial x} \;\; \frac{\partial H}{\partial \lambda}\right)$ has full rank. This is guaranteed under certain assumptions:

Th. 11.11: r twice differentiable $\Rightarrow$ for almost all $a \in \mathbb{R}^n$ there is a zero path from $(a, 0)$ along which the matrix $\left(\frac{\partial H}{\partial x} \; \frac{\partial H}{\partial \lambda} \right)$ has full rank. If this path is bounded for $\lambda \in [0, 1)$ then it has an accumulation point $(\bar{x}, 1)$ with $r(\bar{x}) = 0$. If $J(\bar{x})$ is non-singular, the zero path between $(a, 0)$ and $(\bar{x}, 1)$ has finite length.

Thus, unless we are unfortunate in the choice of a, the continuation algorithm will find a path that either diverges or leads to a root $\bar{x}$ if $J(\bar{x})$ is nonsingular. However, divergence can occur in practice.

- Continuation methods can fail in practice with even simple problems and they require considerable computation; but they are generally more reliable than merit-function methods.

CH. 12: THEORY OF CONSTRAINED OPTIMIZATION

Optimisation problem $\min\limits_{x\in\mathbb{R}^n} f(x)$ s.t. $\begin{cases} c_i(x)=0, & i\in\mathcal{E} \\ c_i(x)\geq 0, & i\in\mathcal{I}\end{cases}$ ← finite sets of indices

$f(x)$ ← objective function

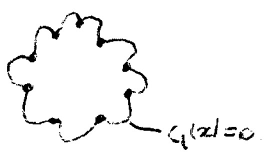
or equivalently $\min\limits_{x\in\Omega} f(x)$ where:

- $\Omega = \{x\in\mathbb{R}^n: \overbrace{c_i(x)=0,\ i\in\mathcal{E}}^{\text{equality constr}};\ \overbrace{c_i(x)\geq 0,\ i\in\mathcal{I}}^{\text{inequality constraints}}\}$ <u>feasible set</u>
- The objective function f and the constraints c_i are all smooth
- x^* is a <u>local solution</u> iff $x^*\in\Omega$ and $\exists$ neighbourhood $\mathcal{N}$ of x^*: $f(x)\geq f(x^*)$ for $x\in\mathcal{N}\cap\Omega$.
- x^* is a <u>strict local solution</u> iff $x^*\in\Omega$ and $\exists$ neighbourhood $\mathcal{N}$ of x^*: $f(x) > f(x^*)$ for $x\in\mathcal{N}\cap\Omega$, $x\neq x^*$
- x^* is an <u>isolated local solution</u> iff $x^*\in\Omega$ and $\exists$ neighbourhood $\mathcal{N}$ of x^*: x^* is the only local minimiser in $\mathcal{N}\cap\Omega$.
- At a feasible point x, the inequality contraint c_i $(i\in\mathcal{I})$ is:
  - <u>active</u> iff $c_i(x)=0$ (x is on the boundary for that constraint)
  - <u>inactive</u> iff $c_i(x)>0$ (x is interior point " " ")
- For inequality constraints, the <u>contraint normal</u> $\nabla c_i(x)$ points towards the feasible region and is $\perp$ to the contour $c_i(x)=0$. For equality contraints, $\nabla c_i(x)$ is $\perp$ to the contour $c_i(x)=0$.
- Mathematical characterisation of solutions for unconstrained optimisation (reminder):
  - <u>Necessary conditions</u>: x^* local minimiser of $f \Rightarrow \nabla f(x^*)=0$, $\nabla^2 f(x^*)$ psd
  - <u>Sufficient conditions</u>: $\nabla^{2} f(x^*)=0$, $\nabla^2 f(x^*)$ pd $\Rightarrow x^*$ is a strong local minimiser of f.

Here we derive similar conditions for contrained optimisation problems. Let's see some examples.

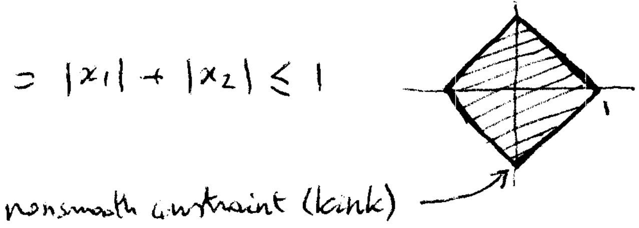
- <u>Local and global solutions</u>: constraining can decrease or increase the number of optimisers:

$\min\limits_{x\in\mathbb{R}^n} \|x\|_2^2 \begin{cases} \text{unconstrained: single solution} \\ \text{constrained s.t. } \|x\|_2^2\geq 1 \text{: infinite solutions} \\ \text{constrained s.t. } c_1(x)=0 \text{: several solutions}\end{cases}$



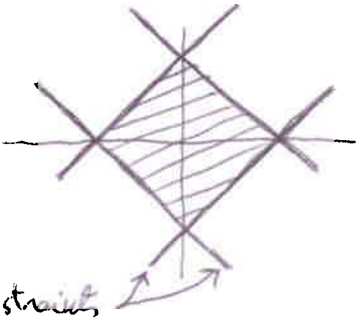
- Smoothness of both f and the constraints is important since then we can predict what happens near a point.

$\|x\|_1 = |x_1| + |x_2| \le 1$

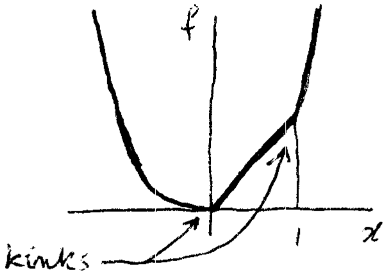


$\iff$

$$\begin{aligned} x_1 + x_2 &\le 1 \\ x_1 - x_2 &\le 1 \\ -x_1 + x_2 &\le 1 \\ -x_1 - x_2 &\le 1 \end{aligned}$$



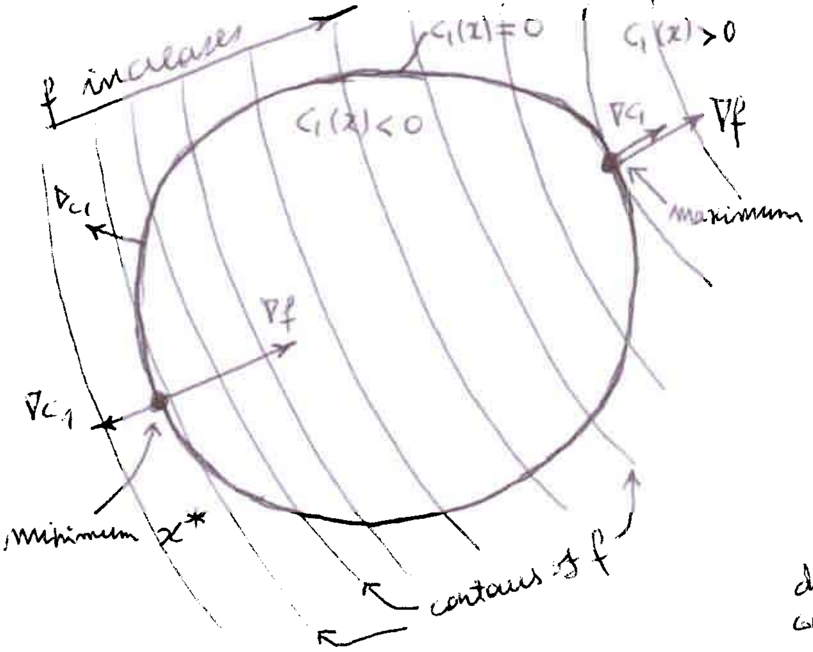
$$f(x) = \max_{x \in \mathbb{R}} (x^2, x)$$



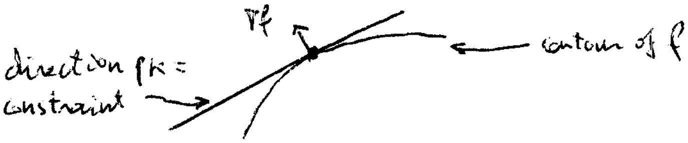
$\iff$ $\min t$ s.t. $\begin{cases} t \ge x \\ t \ge x^2 \end{cases}$

smooth

- Example: a single equality constraint $c_1(x) = 0$: at x^* $\nabla c_1(x^*) \parallel \nabla f(x^*)$, i.e. $\nabla f(x^*) = \lambda_1^* \nabla c_1(x^*)$ for $\lambda_1^* \in \mathbb{R}$.



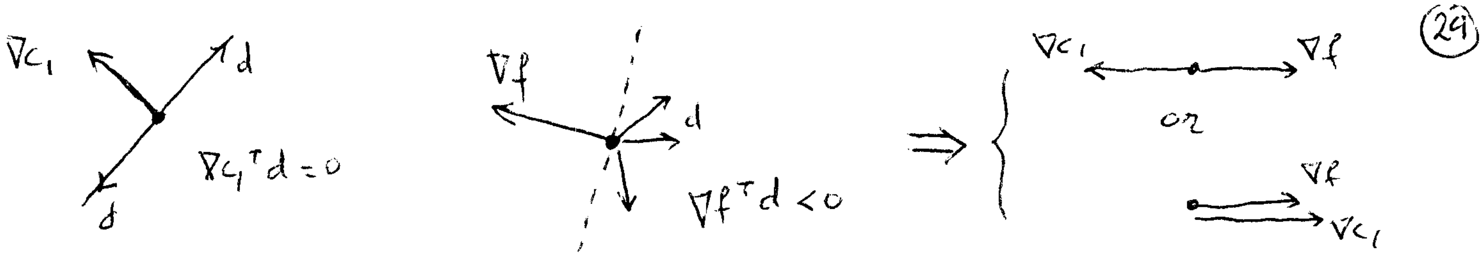
- This is necessary but not sufficient since it also holds at the maximum.
- The sign of λ_1^* can be + or − (e.g. use $-c_1$ instead of c_1).
- Compare with exact line search:



Pf: consider a feasible point at x, i.e., $c_1(x) = 0$. An infinitesimal move to $x + d$:

- retains feasibility if $\nabla c_1(x)^T d = 0$ (by Taylor's th: $0 = c_1(x+d) \simeq \overset{0}{\cancel{c_1(x)}} + \nabla c_1(x)^T d$) (contour line)
- decreases f if $\nabla f(x)^T d < 0$ (by Taylor's th: $0 > f(x+d) - f(x) \simeq \nabla f(x)^T d$) (descent direction)

Thus if no improvement is possible then there cannot be a direction d such that $\nabla c_1(x)^T d = 0$ and $\nabla f(x)^T d < 0$ $\Rightarrow$ $\nabla f(x) = \lambda_1 \nabla c_1(x)$ for some $\lambda_1 \in \mathbb{R}$.



Equivalent formulation in terms of the Lagrangian function $\mathcal{L}(x, \lambda_1) = f(x) - \underset{\substack{\uparrow \\ \text{Lagrange multiplier}}}{\lambda_1} c_1(x)$:

at a solution x^*, $\exists \lambda_1^* \in \mathbb{R}$: $\nabla_x \mathcal{L}(x^*, \lambda_1^*) = 0$.

Idea: to optimise equality-constrained problems, search for stationary points of the Lagrangian.

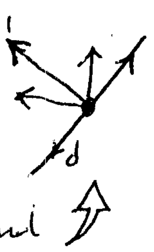
- Example: a single inequality constraint $c_1(x) \geq 0$: the solution is the same, but now the sign of λ_1^* matters: at x^*, $\nabla f(x^*) = \lambda_1^* \nabla c_1(x^*)$ for $\lambda_1^* \geq 0$.

  Pf. consider a feasible point at x, i.e., $c_1(x) \geq 0$. An infinitesimal move to $x + d$:

  - retains feasibility if $c_1(x) + \nabla c_1(x)^T d \geq 0$ (by Taylor's th: $0 \leq c_1(x+d) \simeq c_1(x) + \nabla c_1(x)^T d$).
  - decreases f if $\nabla f(x)^T d < 0$ as before.

  If no improvement is possible:

  a) Interior point $c_1(x) > 0$: any small enough d satisfies feasibility $\Rightarrow \nabla f(x) = 0$ (this is the unconstrained case)



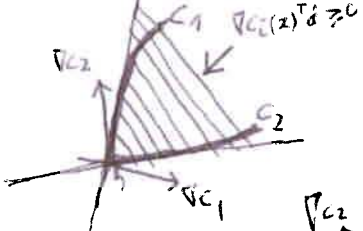
  b) Boundary point $c_1(x) = 0$: there cannot be a direction such that $\nabla f(x)^T d < 0$ and $\nabla c_1(x)^T d \geq 0$ $\Rightarrow$ $\nabla f(x)$ and $\nabla c_1(x)$ must point in the same direction $\Rightarrow$ $\nabla f(x) = \lambda_1 \nabla c_1(x)$ for some $\lambda_1 \geq 0$. $\Rightarrow$ $\xrightarrow{\nabla f} \nabla c_1$

  Equivalent formulation: at a solution x^* $\exists \lambda_1^* \geq 0$: $\nabla_x \mathcal{L}(x^*, \lambda_1^*) = 0$ and $\overbrace{\lambda_1^* c_1(x^*) = 0}^{\circledast}$

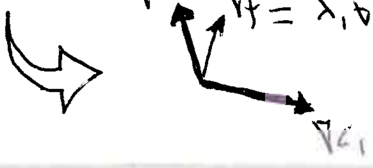
  $\circledast$ is a complimentarity condition $<$ $\lambda_1^* > 0$ and c_1 is active, or / $\lambda_1^* = 0$ and c_1 is inactive.

- Example: two inequality constraints $c_1(x) \geq 0$, $c_2(x) \geq 0$: consider the case of a point x for which both constraints are active, ie, $c_1(x) = c_2(x) = 0$. A direction d is a feasible descent direction to first order if $\nabla f(x)^T d < 0$ and

  $\underbrace{\nabla c_i(x)^T d \geq 0, \ i \in \mathcal{I}}_{\text{Intersection of half spaces defined by } \nabla c_i(x)}$

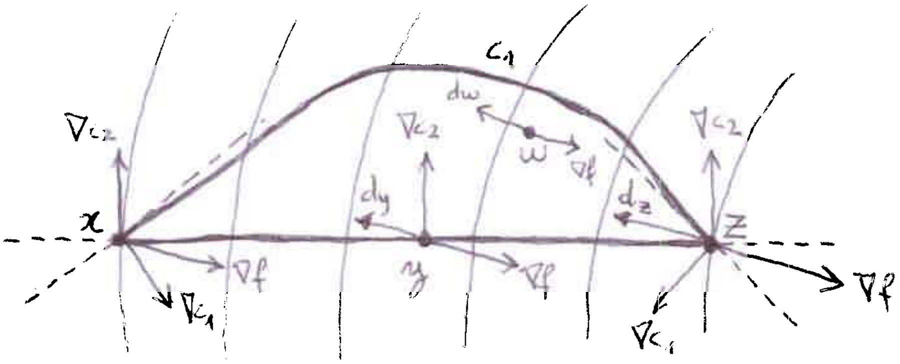


  At a solution, we cannot have a direction d satisfying that (ignore the case where ∇c_i are parallel).



  $\nabla f = \lambda_1 \nabla c_1 + \lambda_2 \nabla c_2$ with $\lambda_1, \lambda_2 \geq 0$

  $\Leftrightarrow$ ∇f is in positive quadrant of $(\nabla c_1, \nabla c_2)$.



At { x: no d satisfies $\nabla c_1(x)^T d \geq 0$, $\nabla c_2(x)^T d \geq 0$, $\nabla f(x)^T d < 0$

z: d_z satisfies $\nabla c_1(z)^T d \geq 0$, $\nabla c_2(z)^T \geq 0$, $\nabla f(z)^T d < 0$

y: d_y satisfies $c_1(y) + \nabla c_1(y)^T d \geq 0$, ($c_1$ not active), $\nabla c_2(y)^T d \geq 0$, $\nabla f(y)^T d < 0$

w: d_w satisfies $c_1(w) + \nabla c_1(w)^T d \geq 0$, $c_2(w) + \nabla c_2(w)^T d \geq 0$ (c_1, c_2 not active), $\nabla f(w)^T d < 0$

Equivalent formulation: $\mathcal{L}(x, \lambda) = f(x) - \sum_i \lambda_i c_i(x)$. At a solution x^*, $\exists\, \lambda^* \geq 0$ ($\equiv \lambda_i^* \geq 0$): $\nabla_x \mathcal{L}(x^*, \lambda^*) = 0$ and $\lambda_i^* c_i(x^*) = 0$ (complementarity condition).

* First-order necessary conditions for optimality (Karush - Kuhn - Tucker conditions)

They relate the gradient of f and of the constraints at a solution.

Consider the constrained optimization problem $\min_{x \in \mathbb{R}^n} f(x)$ s.t. $\begin{cases} c_i(x) = 0, & i \in \mathcal{E} \\ c_i(x) \geq 0, & i \in \mathcal{I} \end{cases}$

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

Active set $\mathcal{A}(x) = \mathcal{E} \cup \{ i \in \mathcal{I} : c_i(x) = 0 \}$

Degenerate constraint behaviour: eg $c_1^2(x)$ is equivalent to $c_1(x)$ as an equality constraint, but $\nabla(c_1^2) = 2 c_1 \nabla c_1 = 0$ at any feasible point, which disables the condition $\nabla f = \lambda_1 \nabla c_1$. We can avoid degenerate behaviour by requiring the following constraint qualification (others possible):

LICQ (def. 12.1): given x^*, $\mathcal{A}(x^*)$, the linear independence constraint qualification (LICQ) holds iff the set of active constraint gradients $\{ \nabla c_i(x^*),\ i \in \mathcal{A}(x^*) \}$ is l.i.

(which implies $\nabla c_i(x^*) \neq 0$)

Th. 12.1 (KKT conditions): x^* local solution of the optimisation problem, LICQ holds at $x^* \Rightarrow \exists!\ \lambda^* \in \mathbb{R}$ (Lagrange multipliers) such that:

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

Do example 12.4

Note: If $\mathcal{I} = \emptyset$ then KKT $\Leftrightarrow$ $\nabla \mathcal{L}(x^*, \lambda^*) = 0$ (∇ wrt x, λ) which is in principle solvable by writing $x = (x_a, \phi(x_a))^T$ and solving the unconstrained problem $\min f(x_a)$.

- Sensitivity of f wrt constraints:

λ_i^* = how hard f is pushing or pulling against c_i $\Big\langle$ $\lambda_i^* = 0$: f doesn't change; $\lambda_i^* \neq 0$: f changes proportionally to λ_i^*

Intuitive proof: infinitesimal perturbation of c_i (inequality constraint): $c_i(x) \geq \varepsilon$ suppose ε is sufficiently small that the perturbed solution $x^*(\varepsilon)$ still has the same active constraints and that the Lagrange multipliers aren't much affected.

Then:

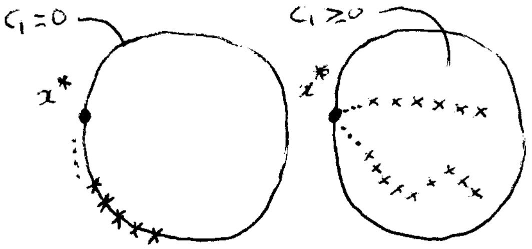
Taylor's th. (1st-order)

$$\begin{aligned} \varepsilon &= \overbrace{c_i(x^*(\varepsilon))}^{\varepsilon} - \overbrace{c_i(x^*)}^{0} \simeq (x^*(\varepsilon) - x^*)^T \nabla c_i(x^*) \\ 0 &= \underbrace{c_j(x^*(\varepsilon))}_{0} - \underbrace{c_j(x^*)}_{0} \simeq (x^*(\varepsilon) - x^*)^T \nabla c_j(x^*) \quad \forall j \neq i,\ j \in \mathcal{A}(x^*) \end{aligned}$$

$$\Rightarrow f(x^*(\varepsilon)) - f(x^*) \simeq (x^*(\varepsilon) - x^*)^T \nabla f(x^*) \overset{\text{KKT}}{=\!=} \sum_{j \in \mathcal{A}(x^*)} \lambda_j^* (x^*(\varepsilon) - x^*)^T \nabla c_j(x^*)$$

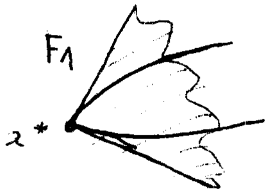
$$\simeq \lambda_i^* \varepsilon \xrightarrow{\varepsilon \to 0} \frac{df(x^*(\varepsilon))}{d\varepsilon} = \lambda_i^*$$

- Feasible sequence: sequence (z_k) that converges to a feasible point x^* and such that z_k is feasible for sufficiently large k. A local solution x of the optimisation problem is a point at which all feasible sequences verify $f(z_k) \geq f(x)$ for suffic. large k. A limiting direction of a feasible sequence is any vector d such that $\lim\limits_{z_k \in S_d} \frac{z_k - x}{\|z_k - x\|} = d$ where S_d is a subsequence of (z_k).



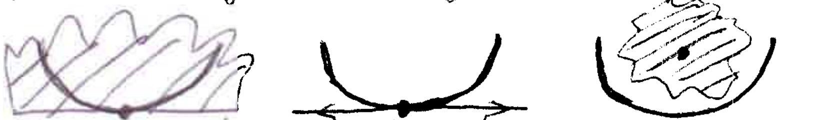
- Def. 12.4: given a point x^* and the active constraint set $\mathcal{A}(x^*)$:

$$F_1 = \left\{ \alpha d : \ d \in \mathbb{R}^n, \ \alpha > 0, \ \begin{array}{ll} d^T \nabla c_i = 0 & \forall i \in \mathcal{E} \\ d^T \nabla c_i \geq 0 & \forall i \in \mathcal{I} \cap \mathcal{A}(x^*) \end{array} \right\}$$



If LICQ holds, F_1 is the tangent cone to the feasible set at x^*.

F_1 is the set of all positive multiples of all limiting directions of all possible feasible sequences. Other examples of tangent cones F_1 in 2D:



Inequality constr.: half space

Equality constr.: line

Inequality constr., interior point: all space

* Second-order conditions

- First-order conditions hold $\Rightarrow$ a move along any vector $w \in F_1$ either increases f ($w^T \nabla f(x^*) > 0$) or keeps it constant ($w^T \nabla f(x^*) = 0$), to first order.
- Second-order conditions give information in the undecided directions $w^T \nabla f(x^*) = 0$, by examining the curvature (cf. unconstrained case).
- Def: given F_1 and some Lagrange multiplier vector λ^* satisfying the KKT, define the following subset of F_1:

$$F_2(\lambda^*) = \{ w \in F_1 : \ \nabla c_i(x^*)^T w = 0, \ \forall i \in \mathcal{I} \cap \mathcal{A}(x^*) \text{ with } \lambda_i^* > 0 \}. \text{ or equiv.}$$

$$w \in F_2(\lambda^*) \Leftrightarrow \begin{cases} \nabla c_i(x^*)^T w = 0 & \forall i \in \mathcal{E} \\ \nabla c_i(x^*)^T w = 0 & \forall i \in \mathcal{I} \cap \mathcal{A}(x^*) \text{ with } \lambda_i^* > 0 . \\ \nabla c_i(x^*)^T w \geq 0 & \forall i \in \mathcal{I} \cap \mathcal{A}(x^*) \text{ with } \lambda_i^* = 0 \end{cases}$$

$F_2(\lambda^*)$ contains the undecided directions for F_1: $w \in F_2(\lambda^*) \xrightarrow{KKT} w^T \nabla f(x^*) =$

$$= \sum_{i \in \mathcal{E} \cup \mathcal{I}} \lambda_i^* \, w^T \nabla c_i(x^*) = 0 \text{ since either } \lambda_i^* = 0 \text{ or } w^T \nabla c_i(x^*) = 0.$$

- Second-order necessary conditions (Th. 12.5): x^* local solution, LICQ condition holds, KKT conditions hold with Lagrange multiplier vector λ^* $\Rightarrow$ $w^T \nabla_{xx} \mathcal{L}(x^*, \lambda^*) w \geq 0 \quad \forall w \in F_2(\lambda^*)$.
- Second-order sufficient conditions (Th. 12.6): $x^* \in \mathbb{R}^n$ feasible point, KKT conditions hold with Lagrange multiplier λ^*, $w^T \nabla_{xx} \mathcal{L}(x^*, \lambda^*) w > 0 \ \forall w \in F_2(\lambda^*), w \neq 0$ $\Rightarrow$ x^* is a strict local solution.

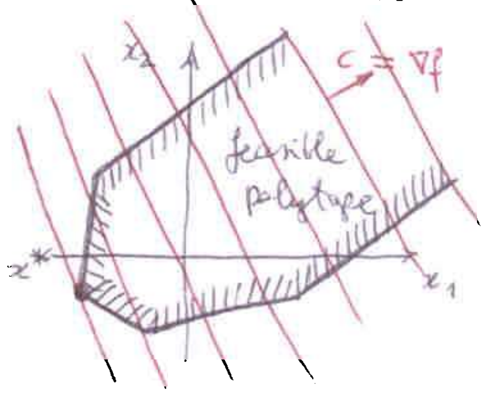
$\triangleright$ examples 12.7, 12.8.

CH. 13: LINEAR PROGRAMMING: THE SIMPLEX METHOD

* Linear program (LP): linear objective function, linear constraints (equality + inequality); feasible set: polytope (= convex, connected set with flat faces); contours of objective: hyperplanes; solution: either none (feasible set is empty), one (a vertex) or an infinite number (edge, face, etc).

- Standard form LP: $\boxed{\min c^Tx \text{ s.t. } Ax=b,\ x\geq 0}$, where $c,x\in\mathbb{R}^n$, $b\in\mathbb{R}^m$, $A_{m\times n}$.

Assume $m<n$ and A has full row rank (otherwise $Ax=b$ contains redundant rows, is infeasible, or defines a unique point).



Examples of transformation to standard form:

- $\max c^Tx \Leftrightarrow \min (-c)^Tx$
- Unbounded variable x:

  Split x into nonnegative and nonpositive parts: $x=x^+-x^-$ where $x^+=\max(x,0)$ and $x^-=\max(-x,0)$. Example:

$$\begin{matrix}\min c^Tx \\ \text{s.t. } Ax\geq b\end{matrix} \Leftrightarrow \min \begin{pmatrix}c\\-c\\0\end{pmatrix}^T\begin{pmatrix}x^+\\x^-\\z\end{pmatrix} \text{ s.t. } \begin{pmatrix}A & -A & -I\end{pmatrix}\begin{pmatrix}x^+\\x^-\\z\end{pmatrix}=b,\quad \begin{pmatrix}x^+\\x^-\\z\end{pmatrix}\geq 0$$

- $x\leq u$ or $Ax\leq b$: add slack variables $\Leftrightarrow$ $x+w=u$, $w\geq 0$ or $Ax+y=b$, $y\geq 0$.
  Equivalently: $x\geq u$ or $Ax\geq b$: add surplus variable $\Leftrightarrow$ $x-w=u$, $w\geq 0$ or $Ax-y=b$, $y\geq 0$

- LP is a very special case of constrained optimisation, but popular because of its simplicity and the availability of software.
- Commercial software accepts LPP in non-standard form.

* Optimality conditions: LP is a convex optimisation problem $\Rightarrow$ any minimiser is a global minimiser; the first-order conditions (KKT) are necessary and also sufficient; the LICQ condition isn't necessary. [Q: what happens with the second-order conditions?]

KKT conditions: $\mathcal{L}(x,\underbrace{\pi,s}_{\text{Lagrange multipliers}}) = c^Tx - \pi^T(Ax-b) - s^Tx$. x is a solution $\Rightarrow \exists\, \pi, s$:

$$\left.\begin{array}{ll} a) & A^T\pi + s = c \\ b) & Ax = b \\ c) & x\geq 0 \\ d) & s\geq 0 \\ e) & x_is_i = 0,\ i=1,\dots,n \Leftrightarrow x^Ts=0 \end{array}\right\} \Rightarrow c^Tx \overset{a)}{=} (A^T\pi+s)^Tx = (Ax)^T\pi \overset{b)}{=} b^T\pi.$$

The KKT conditions are also sufficient. Pf: let $\bar{x}$ be another feasible point $\Leftrightarrow A\bar{x}=b,\ \bar{x}\geq 0$. Then $c^T\bar{x} \overset{a)}{=} (A^T\pi+s)^T\bar{x} = b^T\pi + \bar{x}^Ts \underset{\llcorner\ \bar{x},\, s\,\geq 0}{\geq} b^T\pi = c^Tx$. And $\bar{x}$ optimal $\Leftrightarrow \bar{x}^Ts=0$.

* The dual problem:

- Primal problem: $\min \overset{\text{primal variables }(n)}{c^Tx}$ s.t. $Ax=b,\ x\geq 0$

  Dual problem: $\max \underset{\text{dual variables }(m)}{b^T\pi}$ s.t. $A^T\pi\leq c$, or $\min -b^T\pi$ s.t. $c-A^T\pi\geq 0$ in form of ch. 12

- Dual of the dual = primal. Pf: restate dual in LP standard form by introducing slack variables $s\geq 0$ (so that $A^T\pi+s=c$) and splitting the unbounded variables π into $\pi=\pi^+-\pi^-$ with $\pi^+,\pi^-\geq 0$. Then we can write the dual as:

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

whose dual is $\max c^Tz$ s.t. $\begin{pmatrix}A\\ -A\\ I\end{pmatrix}z\leq\begin{pmatrix}-b\\ b\\ 0\end{pmatrix} \Leftrightarrow \min -c^Tz$ s.t. $Az=-b,\ z\leq 0 \Leftrightarrow$ primal with $z=-x$.

- KKT conditions for the dual: $\mathcal{L}(\pi,x) = -b^T\pi - x^T(c-A^T\pi)$. π is a solution $\Rightarrow \exists x$:

$$\left.\begin{aligned} Ax&=b\\ A^T\pi&\leq c\\ x&\geq 0\\ x_i(c-A^T\pi)_i&=0,\quad i=1,\dots,n\end{aligned}\right\}$$

which are identical to the primal problem's KKT conditions if we define $s=c-A^T\pi$, i.e.:

	Primal	Dual
π	Optimal Lagrange multipliers	Optimal variables
x	Optimal variables	Optimal Lagrange multipliers

- Duality gap: given a feasible vector x for the primal ($\Leftrightarrow Ax=b,\ x\geq 0$) and a feasible vector (π,s) for the dual ($\Leftrightarrow A^T\pi+s=c,\ s\geq 0$) we have:

$$0\leq x^Ts = x^T(c-A^T\pi) = \underbrace{c^Tx-b^T\pi}_{\text{gap}} \Leftrightarrow c^Tx\geq b^T\pi$$

Thus, the dual objective function $b^T\pi$ is a lower bound on the primal objective function c^Tx; at a solution the gap is 0.

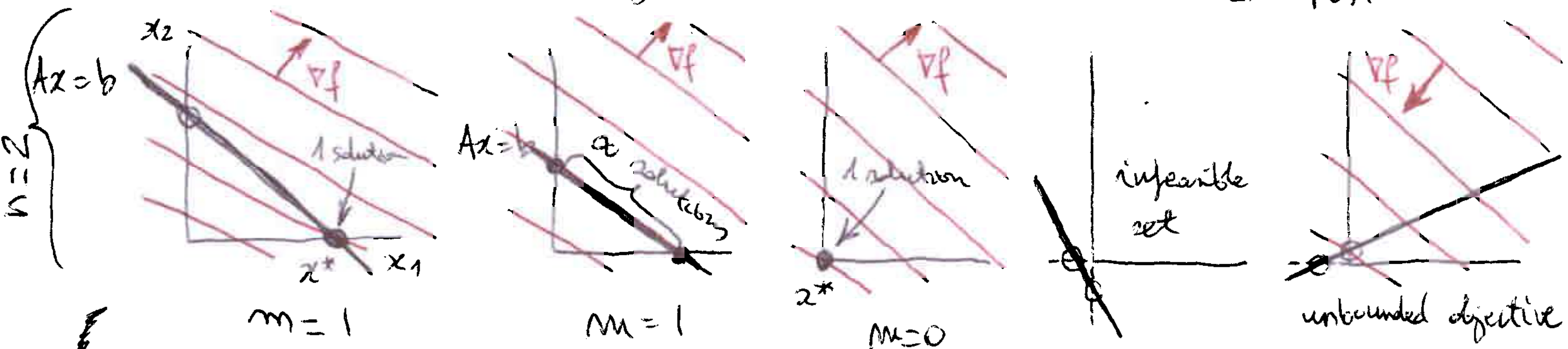
- Duality theorem of LP (th. 13.1):
  a) Primal has a solution with finite optimal objective value $\Rightarrow$ so does the dual, and the objective values are equal (and vice versa).
  b) Primal has an unbounded objective $\Rightarrow$ dual has no feasible points (and vice versa).

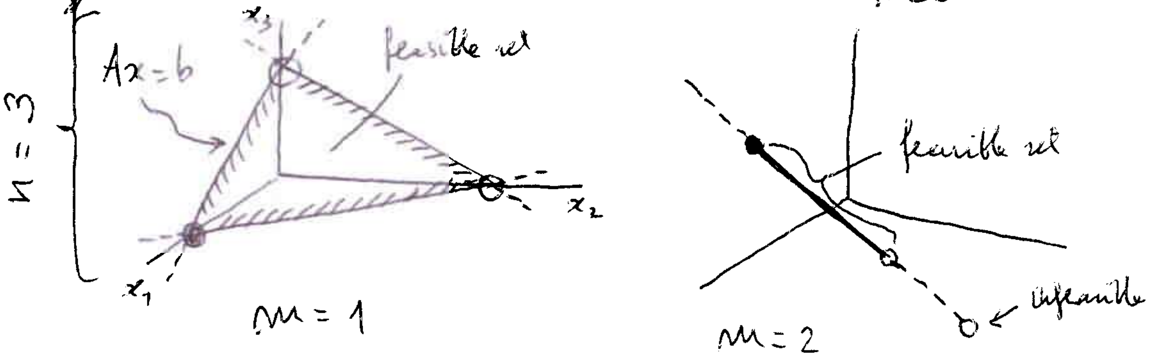
- Sensitivity analysis: how sensitive the optimal objective value is to perturbations in the constraints $\Leftrightarrow$ find the Lagrange multipliers π, s.

- Duality is important in the theory of LP (and convex opt. in general); also, the dual may be easier to solve than the primal. and in primal-dual algorithms

* Geometry of the feasible set

$x \geq 0$ is the n-dim positive quadrant and we consider its intersection with the m-dim. $(m < n)$ linear subspace $Ax = b$. The intersection happens at points x having at most m nonzeros, which are the vertices of the feasible polytope. If the objective is bounded then at least one of these vertices is a minimiser. Examples:





In general, the BFP has at most m nonzero components.

- ○ = basic feasible point
- ● = basic optimal point

(if they exist)

- x is a basic feasible point (BFP) if x is a feasible point with at most m nonzero components and we can identify a subset $B(x)$ of the index set $\{1, \dots, n\}$ such that:
  - $B(x)$ contains exactly m indices
  - $i \notin B(x) \Rightarrow x_i = 0$
  - $B_{m \times m} = [A_i]_{i \in B(x)}$ is nonsingular

Example: $n = 5$, $m = 2$:

$$A = \begin{array}{c} \begin{pmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \end{pmatrix} \\ \begin{array}{ccccc} & 2 & 3 & & \end{array} \end{array} \quad \begin{array}{cc} 0 & \\ \times & 2 \\ \times & 3 \\ 0 & \\ 0 & \end{array}$$

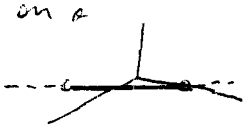
$\mathcal{B}(x) = \{2, 3\}$

$\mathcal{N}(x) = \{1, 4, 5\}$

- The simplex method generates a sequence of iterates x_k that are ~~basic~~ BFPs and converges (in a finite number of steps) to a solution, if the LP has BFPs and at least one of them is a basic optimal point (= a BFP which is a minimiser).

- **Fundamental th. of LP** (th. 13.2): for the standard LP problem:
  - ~~there~~ $\exists$ feasible point $\Rightarrow$ $\exists$ a BFP
  - LPP has solutions $\Rightarrow$ at least one such solution is a basic optimal point.
  - LPP is feasible and bounded $\Rightarrow$ it has an optimal solution. [Proof]

- Th 13.3: all BFPs for the standard LPP are vertices of the feasible polytope $\{x : Ax = b,\ x \geq 0\}$ and vice versa. [a vertex is a point that does not lie on a straight line between two other points in the polytope] [Proof]



- A LPP is <u>degenerate</u> if there exists at least one BFP with $\leq m$ nonzero components.

* <u>The simplex method</u> → Not to be confused with Nelder and Mead's downhill simplex method (derivative-free optimisation method).

There are at most $\binom{n}{m}$ different sets of basic indices $\mathcal{B}$, so a brute-force way to find a solution would be to try them all and check the KKT conditions. The simplex algorithm does better than this: it generates a sequence of iterates all of which are BFPs (thus vertices of the polytope). Each step moves from one vertex to an adjacent vertex for which the set of basic indices $\mathcal{B}(x)$ differs in exactly one component and either decreases the objective or keeps it unchanged.

$\rightarrow \binom{20}{10} \sim 10^5$

- Move: we need to decide which index to change in the basic set $\mathcal{B}$ (by taking it out and replacing it with one index from outside $\mathcal{B}$, i.e., from $\mathcal{N} = \{1, \dots, n\} \setminus \mathcal{B}$). Write the KKT conditions in terms of $\mathcal{B}$ and $\mathcal{N}$ (partitioned matrices and vectors):

$$B = [A_i]_{i \in \mathcal{B}} \qquad N = [A_i]_{i \in \mathcal{N}} \quad \Rightarrow \quad A = \begin{pmatrix} B & N \end{pmatrix}$$

$$x_B = [x_i]_{i \in \mathcal{B}} \qquad x_N = [x_i]_{i \in \mathcal{N}} \quad \Rightarrow \quad x = \begin{pmatrix} x_B \\ x_N \end{pmatrix}, \text{ also } s = \begin{pmatrix} s_B \\ s_N \end{pmatrix}, \ c = \begin{pmatrix} c_B \\ c_N \end{pmatrix}.$$

Since x is a BFP we have: B nonsingular, $x_B \geq 0$, $x_N = 0$ (so KKT c) holds).

KKT a) $b = Ax = Bx_B + Nx_N \Rightarrow x_B = B^{-1}b$

KKT e) $x^T s = x_B^T s_B = 0 \Rightarrow s_B = 0$

KKT a) $s + A^T\pi = c \Rightarrow \begin{pmatrix} s_B \\ s_N \end{pmatrix} + \begin{pmatrix} B^T \\ N^T \end{pmatrix}\pi = \begin{pmatrix} B^T\pi \\ s_N + N^T\pi \end{pmatrix} = \begin{pmatrix} c_B \\ c_N \end{pmatrix} \Rightarrow \begin{cases} \pi = (B^{-1})^T c_B \\ s_N = c_N - (B^{-1}N)^T c_B \end{cases}$

KKT d) $s \geq 0$: while s_B satisfies this, $s_N = c_N - (B^{-1}N)^T c_B$ may not (if it does, i.e., $s_N \geq 0$, we have found an optimal (x, π, s) and we have finished). Thus we take out one of the indices $q \in \mathcal{N}$ for which $s_q < 0$ (there are usually several) and:

- allow x_q to increase from 0
- fix all other components of x_N to 0
- figure out the effect of increasing x_q on the current BFP x_B, given that we want it to stay feasible wrt $Ax = b$

– keep increasing x_q until one of the components of x_B (say, that of x_p) is driven to 0.

– p leaves $\mathcal{B}$ to $\mathcal{N}$, q enters $\mathcal{B}$ from $\mathcal{N}$.

Formally, call x^+ the new iterate and x the current one: we want $Ax^+=b=Ax$:

$$Ax^+ = (B\ N)\begin{pmatrix}x_B^+\\ x_N^+\end{pmatrix} \overset{x_i^+=0 \text{ for } i\in\mathcal{N}\setminus\{q\}}{=} \underbrace{B}_{m\times m}\underbrace{x_B^+}_{m\times 1} + \underbrace{A_q}_{m\times 1}\underbrace{x_q^+}_{1\times 1} = Bx_B + Ax \Rightarrow x_B^+ = x_B - \underbrace{B^{-1}A_q x_q^+}_{}$$

Increase x_q^+ till some component of x_B^+ becomes 0

This operation decreases c^Tx (pf.: p. 377).

Th. 13.4: If the LPP is nondegenerate and bounded, the simplex method terminates at a basic optimal point.

- The practical implementation of the simplex needs to take care of some details:
  - Degenerate & unbounded cases
  - Efficient implementation of the linear system solution
  - Selection of the entering index from among the several negative components of s.

- The simplex method is very efficient in practice (it typically requires $2m$ to $3m$ iterations) but it does have a worst-case complexity that is exponential in n. This can be demonstrated with a pathological n-dim problem where the feasible polytope has 2^n vertices, all of which are visited by the simplex method before reaching the optimal point.

CH. 14: LINEAR PROGRAMMING: INTERIOR-POINT METHODS

Interior-point methods	Simplex method
All iterates satisfy the inequality constraints *strictly*, so they approach the solution from the inside (in some methods from the outside) but never lie on the boundary of the feasible set.	Moves along the boundary of the feasible polytope, testing a sequence of vertices until it finds the optimal one.
Each iteration is expensive to compute but can make significant progress towards the solution.	Usually requires a larger number of inexpensive iterations.
Average-case complexity = worst-case complexity = polynomial	Average-case complexity: 2m - 3m (m = number of constraints); worst-case complexity: exponential.

* Primal-dual methods

- Standard-form primal LPP: $\min c^Tx$ s.t. $Ax=b$, $x \geq 0$ $\quad c, x \in \mathbb{R}^n$, $b \in \mathbb{R}^m$, $A_{m\times n}$

  Dual LPP: $\max b^T\lambda$ s.t. $A^T\lambda + s = c$, $s \geq 0$ $\quad \lambda \in \mathbb{R}^m$, $s \in \mathbb{R}^n$

- KKT conditions:

$$\left.\begin{aligned} A^T\lambda + s &= c \\ Ax &= b \\ x^Ts &= 0 \end{aligned}\right\} \begin{array}{l}\text{System of } 2n+m \text{ equations} \\ \text{for } 2n+m \text{ unknowns } x, \lambda, s \\ \text{(mildly nonlinear because of } x^Ts) \end{array} \iff F(x,\lambda,s) = \begin{pmatrix} A^T\lambda + s - c \\ Ax - b \\ XSe \end{pmatrix} = 0$$

$$x, s \geq 0 \qquad\qquad X = \operatorname{diag}(x_i),\ S = \operatorname{diag}(s_i),\ e = \begin{pmatrix}1\\ \vdots \\ 1\end{pmatrix}.$$

- Idea: find solutions (x^*, λ^*, s^*) of this system with a Newton-like method, but modifying the search directions and step sizes to satisfy $x, s > 0$ (strict inequality). The sequence of iterates traces a path in the space (x, λ, s), thus the name primal-dual. Solving the system is relatively easy (little nonlinearity) but the nonnegativity condition complicates things. Infeasible solutions ($F(x,\lambda,s) = 0$ but not $x, s \geq 0$) abound and do not provide useful information about feasible solutions, so we must ensure to exclude them. All the vertices of the x-polytope are associated with a root of F, but most violate $x, s \geq 0$.
- Newton's method to solve nonlinear equations $f(x) = 0$ from current estimate x_k (ch. 11): $x_{k+1} = x_k + \Delta x$ where $J(x_k)\Delta x = -f(x_k)$ and $J(x)$ is the Jacobian of f.
- [recall that if we apply it to solving $\nabla f(x) = 0$ we obtain $\nabla^2 f(x)\, p = -\nabla f(x)$, Newton's method for optimisation]

* Example showing how every vertex of the polytope in x (i.e., $Ax=b$) produces one root of $F(x,\lambda,s)$:

min c^Tx s.t. $Ax=b$, $x\geq 0$. For $c=\begin{pmatrix}1\\1\\0\end{pmatrix}$, $A=(1\ \tfrac{1}{2}\ 2)$, $b=2$:

KKT conditions:

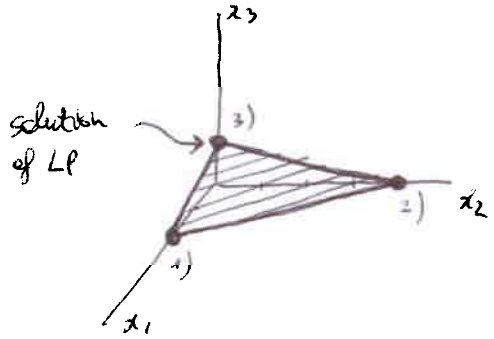
$$\left.\begin{aligned} A^T\lambda+s&=c\\ Ax&=b\\ s^Tx&=0\\ x,s&\geq 0\end{aligned}\right\}\Rightarrow \overbrace{\left.\begin{aligned}\lambda+s_1&=1\\ \frac{\lambda}{2}+s_2&=1\\ 2\lambda+s_3&=0\\ x_1+\tfrac{1}{2}x_2+2x_3&=2\\ (1-\lambda)x_1&=0\\ (1-\tfrac{\lambda}{2})x_2&=0\\ \lambda x_3&=0\end{aligned}\right\}}^{F(x,\lambda,s)=0}$$

with solutions

1) $\lambda=1$, $x=\begin{pmatrix}2\\0\\0\end{pmatrix}$, $s=\begin{pmatrix}0\\1/2\\-2\end{pmatrix}$ infeasible

2) $\lambda=2$, $x=\begin{pmatrix}0\\4\\0\end{pmatrix}$, $s=\begin{pmatrix}-1\\0\\-4\end{pmatrix}$ infeasible

3) $\lambda=0$, $x=\begin{pmatrix}0\\0\\1\end{pmatrix}$, $s=\begin{pmatrix}1\\1\\0\end{pmatrix}$ feasible

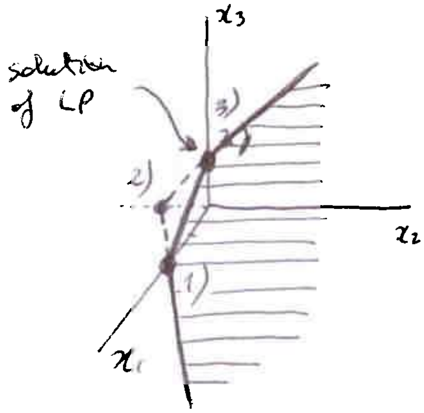


Another example: $A=(1\ \ -2\ \ \ 2)$ above. The solutions of $F(x,\lambda,s)=0$ are:

1) $\lambda=1$, $x=\begin{pmatrix}2\\0\\0\end{pmatrix}$, $s=\begin{pmatrix}0\\3\\-2\end{pmatrix}$ infeasible

2) $\lambda=-\frac{1}{2}$, $x=\begin{pmatrix}0\\-1\\0\end{pmatrix}$, $s=\begin{pmatrix}3/2\\0\\1\end{pmatrix}$ infeasible

3) $\lambda=0$, $x=\begin{pmatrix}0\\0\\1\end{pmatrix}$, $s=\begin{pmatrix}1\\1\\0\end{pmatrix}$ feasible



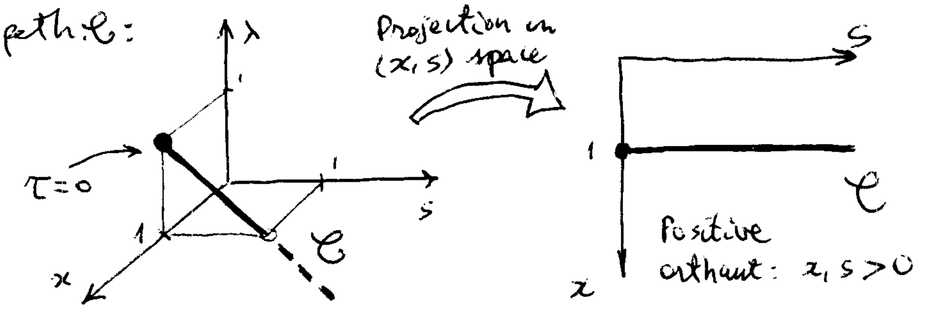
Thus the need to steer away from the boundary till we approach the solution.

* Example of central path: min c^Tx st. $Ax=b$, $x\geq 0$ for $x\in\mathbb{R}$; $A=b=c=1$

KKT equations for central path $\mathcal{C}$:

$$\left.\begin{aligned} A^T\lambda+s&=c\\ Ax&=b\\ x^Ts&=\tau\end{aligned}\right\}\quad \begin{aligned} x&=1\\ s&=\tau\\ \lambda&=1-\tau\end{aligned}$$

$\tau>0$



- In our case the Jacobian $J(x,\lambda,s)$ takes a simple form. Assuming the current point (x,λ,s) is strictly feasible $(x,s>0)$ the Newton step is:

$$\boxed{J(x,\lambda,s) = \begin{pmatrix} 0 & A^T & I \\ A & 0 & 0 \\ S & 0 & X \end{pmatrix}} \Rightarrow J\begin{pmatrix} \Delta x \\ \Delta\lambda \\ \Delta s \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ -XSe \end{pmatrix}$$

This is called the <u>affine scaling direction</u>

Since a full step would likely violate $x,s \geq 0$, we perform a line search so that the new iterate is $\begin{pmatrix} x \\ \lambda \\ s \end{pmatrix} + \alpha \begin{pmatrix} \Delta x \\ \Delta\lambda \\ \Delta s \end{pmatrix}$ for $\alpha \in (0,1]$. Still, often $\alpha \ll 1$. Primal-dual methods modify the basic Newton procedure by:

$$\longrightarrow \alpha < \min\left(\min_{\Delta x_i<0} \frac{-x_i}{\Delta x_i}, \min_{\Delta s_i<0} \frac{-s_i}{\Delta s_i}\right)$$ WHY?

1) Biasing the search direction towards the interior of the nonnegative orthant $x,s \geq 0$ (so more room to move within it).
2) keeping x_i, s_i from moving too close to the boundary of the nonnegative orthant.

* <u>The central path</u>

- Define: <u>feasible set $\mathcal{F}$</u> $= \{(x,\lambda,s) : Ax=b,\ A^T\lambda+s=c,\ x,s\geq 0\}$

  <u>strictly feasible set $\mathcal{F}^o$</u> $= \{(x,\lambda,s) : Ax=b,\ A^T\lambda+s=c,\ x,s>0\}$

Parameterise the KKT system in terms of a scalar parameter $\tau>0$:

$$\left.\begin{aligned} A^T\lambda + s &= c \\ Ax &= b \\ x_i s_i &= \tau \quad i=1,\dots,n \\ x,s &\geq 0 \end{aligned}\right\}$$

The solution $F(x_\tau,\lambda_\tau,s_\tau)=0$ gives a curve $\mathcal{C} = \{(x_\tau,\lambda_\tau,s_\tau) : \tau>0\}$ whose points are strictly feasible, and that converges to a solution for $\tau \to 0$. This curve is the <u>central path $\mathcal{C}$.</u>

The central path is defined uniquely $\forall \tau>0 \Leftrightarrow \mathcal{F} \neq \emptyset$.

The central path guides us to a solution along a route that steers clear of spurious solutions by keeping all x and s components strictly positive and decreasing the pairwise products $x_i s_i$ to 0 at roughly the same rate. A Newton step towards points in $\mathcal{C}$ is biased toward the interior of the nonnegative orthant $x,s\geq 0$ and so they can usually be longer than the pure Newton steps for F (which aims at a point in $\mathcal{F}$).

- The biased search direction is given by defining $\tau = \sigma\mu$ where:
  - <u>Duality measure</u> $\mu = \dfrac{x^Ts}{n}$ = average of the pairwise products $x_i s_i$. It measures closeness to the boundary, and the algorithms drive μ to zero.

- Centering parameter $\sigma \in [0,1]$

  $\sigma = 0$: pure Newton step towards (x_0, λ_0, s_0) (affine-scaling direction); aims at reducing μ.

  $\sigma = 1$: Newton step towards $(x_\mu, \lambda_\mu, s_\mu) \in \mathcal{C}$ (centering direction); aims at centrality.

  Primal-dual methods trade off both aims.

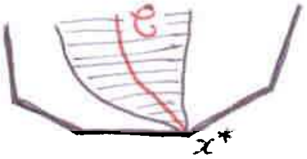
The Newton step: $J \begin{pmatrix} \Delta x \\ \Delta \lambda \\ \Delta s \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ -XSe + \sigma\mu e \end{pmatrix}$.

* General framework for primal-dual algorithms (14.1)

```
Given $(x^0, \lambda^0, s^0) \in \mathcal{F}^0$
for $k = 1, 2, \dots$
    Solve $\begin{pmatrix} 0 & A^T & I \\ A & 0 & 0 \\ S^k & 0 & X^k \end{pmatrix} \begin{pmatrix} \Delta x^k \\ \Delta \lambda^k \\ \Delta s^k \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ -X^k S^k e + \sigma_k \mu_k e \end{pmatrix}$ where $\sigma_k \in [0,1]$, $\mu_k = \frac{(x^k)^T s^k}{n}$
    $(x^{k+1}, \lambda^{k+1}, s^{k+1}) \leftarrow (x^k, \lambda^k, s^k) + \alpha_k (\Delta x^k, \Delta \lambda^k, \Delta s^k)$ choosing $\alpha_k$ such that $x^{k+1}, s^{k+1} > 0$
end
```

- The strategies for choosing or adapting σ_k, α_k depend on the particular algorithm.
- Strict feasibility is preserved: $(x^k, \lambda^k, s^k) \in \mathcal{F}^0 \Rightarrow (x^{k+1}, \lambda^{k+1}, s^{k+1}) \in \mathcal{F}^0$ [why?]
- Finding ~~an~~ a starting point that is strictly feasible is difficult. One approach are infeasible-interior-point methods:
  - choose a starting point with $x^0, s^0 > 0$ (always possible)
  - use this step equation: $J \begin{pmatrix} \Delta x \\ \Delta \lambda \\ \Delta s \end{pmatrix} = \begin{pmatrix} -r_c \\ -r_b \\ -XSe + \sigma\mu e \end{pmatrix}$

  where $r_b = Ax - b$, $r_c = A^T\lambda + s - c$ are the residuals for the linear equations. This is still a Newton step towards $(x_{\sigma\mu}, \lambda_{\sigma\mu}, s_{\sigma\mu}) \in \mathcal{C}$, but it tries to correct the infeasibility in the equality constraints in a single step. If a full step ($\alpha_k = 1$) is taken at any iteration, the residuals become 0 and all subsequent iterates remain strictly feasible. [why?]



* Path-following methods

- They explicitly restrict iterates to a neighbourhood of the central path $\mathcal{C}$, thus following $\mathcal{C}$ more or less strictly. That is, we choose $\alpha_k \in [0,1]$ as large as possible but so that $(x^{k+1}, \lambda^{k+1}, s^{k+1})$ lies in the neighbourhood. An example of neighbourhood is

  $\mathcal{N}_{-\infty}(\gamma) = \{(x, \lambda, s) \in \mathcal{F}^0 : x_i s_i \geq \gamma\mu,\ i = 1, \dots, n\}$ for $\gamma \in (0,1]$ (typically $\gamma = 10^{-3}$).

  $\mathcal{N}_{-\infty}(0) = \mathcal{F}$.

- Most computational effort is spent solving the linear system for the direction; however, this is often a sparse system because A is often sparse.
- They are globally convergent. (it can be proven that $\mu_{k+1} \leq C\mu_k$ for constant $C \in (0,1)$ if $\forall k$ $\sigma_k \in [\sigma_{min}, \sigma_{max}]$ with $0 < \sigma_{min} < \sigma_{max} < 1$). Convergence rate: given $\varepsilon \in (0,1)$, $\mathcal{O}(n \log \frac{1}{\varepsilon})$ iterations are necessary to find a point for which $\mu_k < \varepsilon$.

* <u>Mehrotra predictor-corrector algorithm</u>

At each iteration:

1) Compute the affine-scaling direction (i.e., $(\Delta x, \Delta \lambda, \Delta s)$ for $\sigma = 0$) and the largest step size $\alpha \in [0,1]$ that satisfies $x, s \geq 0$, resulting in a predictor step to (x', λ', s').

2) Compute the effectiveness $\mu_{aff} = \frac{(x')^T s'}{n}$ of this step and set $\sigma = (\mu_{aff}/\mu)^3$ (adaptive σ). Thus, if the predictor step is effective, μ_{aff} is small and σ is close to 0, otherwise σ is close to 1.

3) Compute the step direction using this σ (corrector step):

$$J \begin{pmatrix} \Delta x \\ \Delta \lambda \\ \Delta s \end{pmatrix} = \begin{pmatrix} -r_c \\ -r_b \\ -XSe - \Delta X^{aff} \Delta S^{aff} e + \sigma \mu e \end{pmatrix}$$

This is an approximation to keeping to the central path:

$$(x_i + \Delta x_i)(s_i + \Delta s_i) = \sigma\mu \Rightarrow x_i \Delta s_i + s_i \Delta x_i = \sigma\mu - \underbrace{\Delta x_i \Delta s_i}_{\text{approximated with } \Delta x_i^{aff},\ \Delta s_i^{aff}} - x_i s_i$$

- No convergence theory available for this algorithm (which can occasionally diverge); but it has good practical performance.

CH. 15: FUNDAMENTALS OF ALGORITHMS FOR NONLINEAR CONSTRAINED OPTIM.

- General constrained optimisation problem: $\min_{x\in\mathbb{R}^n} f(x)$ s.t. $\begin{cases} c_i(x)=0, & i\in\mathcal{E} \\ c_i(x)\geq 0, & i\in\mathcal{I} \end{cases}$, $f, \{c_i\}$ smooth.

  Special cases (for which specialised algorithms exist):

  - Linear programming (LP): f, all c_i linear; solved by simplex and interior-point methods.
  - Quadratic programming (QP): f quadratic, all c_i linear.
  - Linearly constrained optimisation: all c_i linear.
  - Bound-constrained optimisation: constraints are only of the form $x_i \geq l_i$ or $x_i \leq u_i$.
  - Convex programming: f convex, equality c_i linear, inequality c_i concave. [Q: is QP convex programming?]

- Brute-force approach: guess which inequality constraints are active ($\lambda_i^* \neq 0$), try to solve the nonlinear equations given by the KKT conditions directly and then check whether the resulting solutions are feasible. If there are m inequality constraints and k are active, we have $\binom{m}{k}$ combinations and so altogether $\binom{m}{0}+\binom{m}{1}+\dots+\binom{m}{m}=2^m$ combinations, which is wasteful unless we can really guess which constraints are active. Solving a nonlinear system of equations is still hard because the root-finding algorithms are not guaranteed to find a solution from arbitrary starting points.

- Iterative algorithm: sequence of x_k ~~that~~ (and possibly of Lagrange multipliers associated with the constraints) that converges to a solution. The move to a new iterate is based on information about the objective and constraints, and their derivatives, at the current iterate, possibly combined with information gathered in previous iterates. Termination occurs when a solution is identified accurately enough, or when further progress can't be made.

  Goal: to find a local minimiser (global optimisation is too hard).

- Initial study of the problem: try to show whether the problem is infeasible or unbounded; try to simplify the problem.

- Hard constraints: they cannot be violated during the algorithm's run, e.g. nonnegativity of x if $\sqrt{x}$ appears in the objective function. Need feasible algorithms, which are slower than algorithms that allow the iterates to be infeasible, since they can't follow shortcuts across infeasible territory; but the objective is a merit function, which spares us the need to introduce a more complex merit function that accounts for constraint violations.

Soft constraints: they may be modelled as objective function f + penalty, where the penalty includes the constraints. However, this can introduce ill-conditioning.

* Categorisation of algorithms

- Ch. 16: quadratic programming: it's an important problem by itself and as part of other algorithms; the algorithms can be tailored to specific types of QP.
- Ch. 17: penalty, barrier and augmented Lagrangian methods.

– Penalty methods: combine objective and constraints into a penalty function $\phi(x;\mu)$ via a penalty parameter $\mu > 0$; eg if only equality constraints exist:

- $\phi(x;\mu) = f(x) + \frac{1}{2\mu}\sum_{i\in\mathcal{E}} c_i^2(x)$ ⇒ unconstrained minimisation of ϕ wrt x for a series of decreasing μ values.
- $\phi(x;\mu) = f(x) + \frac{1}{\mu}\sum_{i\in\mathcal{E}} |c_i(x)|$ (exact penalty function) ⇒ single unconstrained minimisation for small enough μ.

– Barrier methods: add terms to the objective (via a barrier parameter $\mu > 0$) that are insignificant when x is safely inside the feasible set but become large as x approaches the boundary; eg. if only inequality constraints exist:

$P(x;\mu) = f(x) - \mu \sum_{i\in I} \log c_i(x)$ (logarithmic barrier function) ⇒ solve for decreasing values of μ.

– Augmented Lagrangian methods: define a function that combines the Lagrangian function and a quadratic penalty; eg. if only equality constraints exist:

$\mathcal{L}_A(x,\lambda;\mu) = f(x) - \sum_{i\in\mathcal{E}} \lambda_i c_i(x) + \frac{1}{2\mu}\sum_{i\in\mathcal{E}} c_i^2(x)$ ⇒ minimise $\mathcal{L}_A$ wrt x for fixed λ, μ; update λ, decrease μ; repeat.

– Sequential linearly constrained methods: minimise at every iteration a certain Lagrangian function subject to a linearisation of the constraints; useful for large problems.

- Chapter 18: <u>sequential quadratic programming</u>: model the problem as a QP subproblem; solve it by ensuring a certain merit function decreases; repeat. They are effective in practice. Although the QP subproblem is relatively complicated, they typically require fewer function evaluations than some of the other methods.

* Elimination of variables

- Goal: eliminate some of the constraints and so simplify the problem. This must be done with care because the problem may be altered, or ill-conditioning may appear.
- Example 15.0: safe elimination.
- Example 15.1: elimination alters the problem: min x^2+y^2 s.t. $(x-1)^3 = y^2$ has the solution $\binom{x}{y} = \binom{1}{0}$. Eliminating $y^2 = (x-1)^3$ yields min $x^2 + (x-1)^3$ which is unbounded ($x \to -\infty$); the mistake is that this elimination ignores the implicit constraint $x \geq 1$ (since $y^2 \geq 0$) which is active at the solution.

  In general, nonlinear elimination is tricky. Instead, many algorithms linearise the constraints, then apply linear elimination.

- <u>Linear elimination</u>: consider min $f(x)$ st. $Ax = b$ where $A_{m\times n}$, $m \leq n$. and A has full rank (otherwise, remove redundant constraints or determine whether the problem is infeasible). Say we eliminate $x_B = (x_1, \dots, x_m)^T$ (otherwise permute x, A and b); writing $A = (B\ N)$ with $B_{m\times m}$ nonsingular, $N_{m\times(n-m)}$ and $x = \binom{x_B}{x_N}$, we have $x_B = B^{-1}b - B^{-1}N x_N$ (remember how to find a basic feasible point in the simplex method), and we can solve the unconstrained problem $\min_{x_N \in \mathbb{R}^{n-m}} f(x_B(x_N), x_N)$ (example 15.2). Ideally we'd like to select B to be easily factorisable (easier linear system).

  We can also write $x = Yb + Zx_N$ with $Y = \begin{pmatrix} B^{-1} \\ 0 \end{pmatrix}$, $Z = \begin{pmatrix} -B^{-1}N \\ I \end{pmatrix}$. Since:

  1) Z has $n-m$ l.i. columns (due to I being the lower block) and $AZ = 0$, Z is a basis of the null space of A.

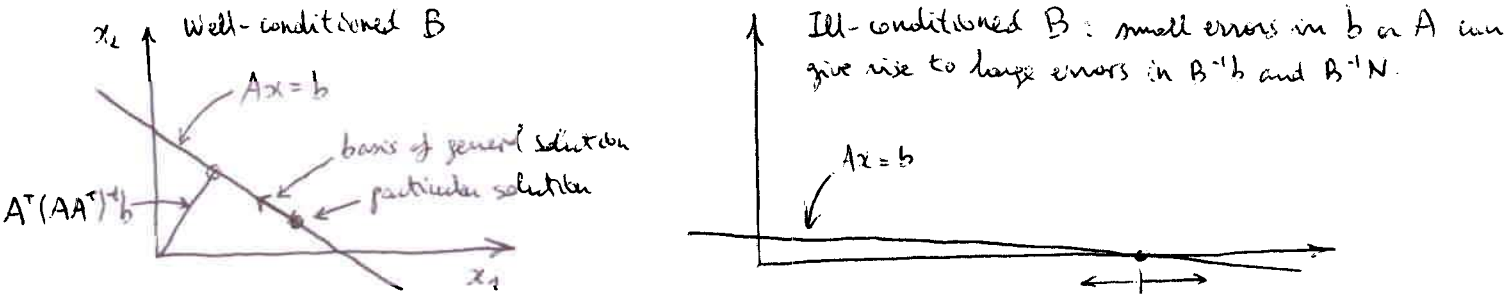
  $\text{null}(A) \oplus \text{range}(A^T) = \mathbb{R}^n$

  2) The columns of Y and the columns of Z are l.i. (pf. $(Y\ Z)\lambda = 0 \Rightarrow \lambda = 0$) $\Rightarrow$ Y is a basis of the range space of A^T, and Yb is a particular solution of $Ax = b$.

Thus the elimination technique expresses feasible points as the sum of a particular solution of $Ax=b$ plus a displacement along the null space of A:

x = (particular solution of $Ax=b$) + (general solution of $Ax=0$).

But linear elimination can give rise to numerical instability, e.g. for $n=2$:


x_2
Well-conditioned B
$Ax=b$
basis of general solution
particular solution
$A^T(AA^T)^{-1}b$
x_1
Ill-conditioned B: small errors in b or A can give rise to large errors in $B^{-1}b$ and $B^{-1}N$.
$Ax=b$

This can be improved by choosing as the particular solution that having minimum norm: min $\|x\|_2$ s.t. $Ax=b$, which is $x_p = A^T(AA^T)^{-1}b$ (pf. apply KKT to min $\frac{1}{2}x^Tx$ s.t. $Ax=b$). Both this x_p and Z can be computed in a numerically stable way using the QR decomposition of A, though the latter is costly if A is large (even if sparse).

- If inequality constraints exist, eliminating equality constraints is worthwhile if the inequality constraints don't get more complicated.

* Measuring progress: merit functions $\phi(x;\mu)$

- A merit function measures a combination of decrease in the objective and feasibility via a penalty parameter $\mu>0$ which controls the tradeoff; several definitions exist. They help to control the optimisation algorithm: a step is accepted if it leads to a sufficient reduction in the merit function.
- Unconstrained optimisation: objective function = merit function. Also in feasible methods (which ~~en~~force all iterates to be feasible).
- A merit function $\phi(x;\mu)$ is exact if $\exists \mu^*>0$: $\mu\in(0,\mu^*] \Rightarrow$ any local solution x of the optimisation problem is a local minimiser of ϕ.
- Useful merit functions:
  - l_1 exact function: $\phi_1(x;\mu) = f(x) + \frac{1}{\mu}\sum_{i\in\mathcal{E}}|c_i(x)| + \frac{1}{\mu}\sum_{i\in\mathcal{I}}[c_i(x)]^-$ where $[x]^- = \max(0,-x)$. It is not differentiable. It is exact for $\frac{1}{\mu^*}$ = largest Lagrange multiplier (in absolute value) associated with an optimal solution. Many algorithms using this function adjust μ heuristically to ensure $\mu<\mu^*$.

It is inexpensive to evaluate but it may reject steps that make good progress toward the solution (Maratos effect).

- Fletcher's augmented Lagrangian: when only equality constraints $Ax=b$ exist, $\phi_F(x;\mu) = f(x) - \lambda^T(x)c(x) + \frac{1}{2\mu}\sum_{i\in\mathcal{E}} c_i(x)^2$, where $\lambda(x) = (AA^T)^{-1}A\,\nabla f(x)$ are the least-squares multiplier estimates. It is differentiable and exact and does not suffer from the Maratos effect, but since it requires the solution of a linear system to obtain $\lambda(x)$, it is expensive to evaluate and may be ill-conditioned or not defined.
- Augmented Lagrangian in x and λ: when only equality constraints exist, $\mathcal{L}_A(x,\lambda;\mu) = f(x) - \lambda^T c(x) + \frac{1}{2\mu}\|c(x)\|_2^2$. Here the iterates are (x_k, λ_k), i.e., a step both in the primal and dual variables. A solution of the optimisation problem is a stationary point of $\mathcal{L}_A$ but in general not a minimiser.

CHAPTER 16: QUADRATIC PROGRAMMING

- Quadratic program (QP): quadratic objective function, linear constraints.

$$\min_{x\in\mathbb{R}^n} q(x) = \frac{1}{2}x^TGx + x^Td \quad \text{s.t.} \begin{cases} a_i^Tx = b_i, & i\in\mathcal{E} \\ a_i^Tx \geq b_i, & i\in\mathcal{I} \end{cases} \quad G_{n\times n} \text{ symmetric}$$

Can always be solved in a finite number of iterations (exactly how many depends on G and on the number of inequality constraints).

Convex QP $\Leftrightarrow$ G psd. Local minimiser(s) also global; not much harder than LP.

Nonconvex QP $\Leftrightarrow$ G not psd. Possibly several solutions.

- Example: portfolio optimisation
  - n possible investments (bonds, stocks, etc.) with returns r_i, $i=1,\dots,n$.
  - $r = \begin{pmatrix} r_1 \\ \vdots \\ r_n \end{pmatrix}$ is a random variable with mean μ and covariance G:

    $\mu_i = E[r_i]$, $\quad g_{ij} = E[(r_i-\mu_i)(r_j-\mu_j)]$ = tendency of the returns of investments i,j to move together

    Usually high μ_i means high g_{ii}.
  - An investor constructs a portfolio by putting a fraction $x_i\in[0,1]$ of the available funds into investment i, with $\sum_{i=1}^n x_i = 1$ and $x\geq 0$.

    Return of the portfolio $R = x^Tr$, with:
    - . Mean $E[R] = x^T\mu$ (expected return)
    - . Variance $E[(R-E[R])^2] = x^TGx$
  - Wanted portfolio: large expected return, small variance:

    $$\max\ x^T\mu - \kappa x^TGx \quad \text{s.t.} \quad \sum_{i=1}^n x_i = 1, \quad x\geq 0. \qquad \text{[Q: convex QP!]}$$

    where $\kappa \geq 0$ is set by the investor $<$ conservative investor: large κ / aggressive investor: small κ
  - In practice, μ and G are guesstimated based on historical data and "intuition".

* Equality-constrained QP

- m equality constraints, no inequality constraints:

$$\min_x q(x) = \frac{1}{2}x^TGx + x^Td \quad \text{s.t.} \quad Ax=b \qquad A \text{ full rank}$$

- KKT conditions: a solution x^* verifies (where λ^* is the Lagrange multiplier vector)

$$\begin{pmatrix} G & -A^T \\ A & 0 \end{pmatrix}\begin{pmatrix} x^* \\ \lambda^* \end{pmatrix} = \begin{pmatrix} -d \\ b \end{pmatrix} \overset{x^* = x+p}{\Longleftrightarrow} \underbrace{\begin{pmatrix} G & A^T \\ A & 0 \end{pmatrix}}_{\text{KKT matrix } K}\begin{pmatrix} -p \\ \lambda^* \end{pmatrix} = \begin{pmatrix} g \\ c \end{pmatrix} \quad \begin{cases} c = Ax-b \\ g = d+Gx \\ p = x^*-x \end{cases}$$

[Q: what happens if $G=0$ (LP)?]

Let $Z_{n\times(n-m)} = (z_1 \cdots z_{n-m})$ be a basis of null (A) $\Leftrightarrow$ $AZ=0$, rank$(Z) = n-m$.

Call Z^TGZ the reduced Hessian. ($\equiv$ how the quadratic form looks like in the subspace $Ax=b$).

- Classification of the solutions (assuming the KKT system has solutions $\begin{pmatrix} x^* \\ \lambda^* \end{pmatrix}$):

1) Strong local minimiser at x^* $\Leftrightarrow$ Z^TGZ pd. Proof:
   - Either using the 2nd-order sufficient conditions (note that Z is a basis of $F_2(\lambda^*) = \text{null}(A)$)
   - or direct proof that x^* is a global solution (th. 16.2)

2) Infinite solutions if Z^TGZ is psd and singular.

3) Unbounded if Z^TGZ indefinite.

A full rank, Z^TGZ pd $\Rightarrow \begin{cases} K \text{ is nonsingular } \left(\Rightarrow \text{unique } \begin{pmatrix} x^* \\ \lambda^* \end{pmatrix}\right) \text{ by lemma 16.1} \\ \text{inertia}(K) = (n, m, 0) \; (\Rightarrow K \text{ always indefinite}) \text{ lemma 16.3} \end{cases}$

$+ \quad - \quad 0$ ← number of pos/neg/0 eigenvalues

- The KKT system can be solved with various linear algebra techniques (note that linear conjugate gradients are not applicable). why?

* Inequality-constrained QP

- Three types of methods $\begin{cases} \text{- active-set} \\ \text{- gradient-projection} \\ \text{- interior-point} \end{cases}$ and also general ones $\begin{cases} \text{- augmented Lagrangian (ch. 17)} \\ \text{- exact penalty S}\ell_1\text{QP (ch. 18)} \end{cases}$

- Optimality conditions: Laplacian function $\mathcal{L}(x,\lambda) = \frac{1}{2}x^TGx + x^Td - \sum_{i\in I\cup E} \lambda_i(a_i^Tx - b_i)$.

Active set at an optimal point x^*: $\mathcal{A}(x^*) = \{i \in I \cup E : a_i^Tx^* = b_i\}$.

- KKT conditions: it can be proven that the LICQ condition (active constraint gradients are l.i.) is not required.

$$Gx^* + d - \sum_{i\in\mathcal{A}(x^*)} \lambda_i^* a_i = 0$$

$$a_i^Tx^* = b_i \quad \forall i \in \mathcal{A}(x^*)$$

$$a_i^Tx^* \geq b_i \quad \forall i \in I \setminus \mathcal{A}(x^*)$$

$$\lambda_i^* \geq 0 \quad \forall i \in I \cap \mathcal{A}(x^*)$$

– 2nd-order conditions:

1) Strong local minimiser at x^* ⇔ $Z^T G Z$ pd where Z is a nullspace basis for the active constraint Jacobian matrix $(a_i)^T_{i \in \mathcal{A}(x^*)}$.

x^* is also a global solution (th. 16.2).

2) If G is not pd, there may be more than one strict local minimiser at which the 2nd-order conditions hold (nonconvex, or indefinite, problem); harder to solve.

Examples: fig 16.1.

- Degeneracy is one of the following situations, which can cause problems for the algorithms (ex. p. 456):
  - Active constraint gradients are l.d. at the solution, e.g. (but not necessarily) if more than n constraints are active at the solution ⇒ numerically difficult to compute Z.
  - Strict complementarity condition fails: $\lambda_i^* = 0$ for some active index $i \in \mathcal{A}(x^*)$ (the constraint is weakly active) ⇒ numerically difficult to determine whether a weakly active constraint is active.

* Active-set methods for convex QP

- Convex QP: any local solution is also global.
- These are the most effective methods for small- to medium-scale problems.
- Remember the brute-force approach to solving the KKT systems for all combinations of active constraints: if we knew the optimal active set $\mathcal{A}(x^*)$ (≡ the active set at the optimal point x^*), we could find the solution of the equality-constrained QP problem $\min_x q(x)$ s.t. $a_i^T x = b_i$, $i \in \mathcal{A}(x^*)$. Goal: to determine this set.
- Active-set method: start from a guess of the optimal active set; if not optimal, drop one index from $\mathcal{A}(x)$ and add a new index (using gradient & Lagrange multiplier information); repeat.
  - The simplex method for LP is an active-set method.
  - Active-set methods for QP may have iterates that are not in vertices of the feasible polytope.

Three types of active-set methods: primal, dual, and primal-dual. We focus on primal methods, which generate iterates that remain feasible wrt the primal problem while steadily decreasing the primal objective function f.

- Primal active-set method

– Compute a feasible initial iterate x_0 (some techniques available; pp. 462-463); subsequent iterates will remain feasible.

– Move to next iterate $x_{k+1} = x_k + \alpha_k p_k$: obtained by solving an equality-constrained quadratic subproblem. The constraint set, called working set W_k, consists of all the equality constraints and some of the inequality constraints taken as equality constraints (i.e., assuming they are active); the gradients a_i of the constraints in W_k must be l.i. The quadratic subproblem (solvable as in sec. 16.1):

$$\min_p \; q(x_k+p) \text{ s.t. } W_k \overset{[why?]}{\Longleftrightarrow} \min_p \; \frac{1}{2}p^TGp + (Gx_k+d)^Tp \text{ s.t. } a_i^Tp = 0 \;\; \forall i \in W_k$$

Call p_k the solution of the subproblem. Then:

- Use $\alpha_k = 1$ if possible: if $x_k + p_k$ satisfies all constraints (not just those in W_k) and is thus feasible, set $x_{k+1} = x_k + p_k$.
- otherwise, choose α_k the largest value in $[0,1)$ for which all constraints are satisfied. Note that $x_k + \alpha_k p_k$ satisfies $a_i^T x_{k+1} = b_i$ $\forall \alpha_k \in \mathbb{R}$ $\forall i \in W_k$ [why?] so we need only worry about the constraints not in W_k; the result is trivial to compute (similar to the choice of α_k in interior-point methods for LP):

$$\alpha_k = \min\left(1, \min_{\substack{i \notin W_k \\ a_i^T p_k < 0}} \frac{b_i - a_i^T x_k}{a_i^T p_k}\right) \quad [why?]$$

(typically just one)

The constraints for which the min is achieved are called blocking constraints.

The new working set: if $\alpha_k < 1$ ($\Leftrightarrow$ the step along p_k was blocked by some constraint not in W_k) then add one of the blocking constraints to W_k; otherwise keep W_k.

Note that it is possible that $\alpha_k = 0$; this happens when $a_i^T p_k < 0$ for a constraint i that is active at x_k but not a member of W_k.

– Iterating this process (where we keep adding blocking constraints and moving x_k) we must reach a point $\hat{x}$ that minimizes q over its current working set $\hat{W}$, or equivalently $p = 0$ occurs. Now, is this also a minimizer of the QP problem, i.e., does it satisfy the KKT conditions? Only if the Lagrange multipliers for the inequality constraints in the working set are nonnegative. The Lagrange multipliers are the solution of $\sum_{i \in \hat{W}} a_i \hat{\lambda}_i = G\hat{x} + d$ [why?] So if $\hat{\lambda}_j < 0$ for some $j \in \hat{W} \cap I$ then we drop constraint j from the working set (since by making this constraint inactive we can decrease q while remaining feasible; th. 16.4) and go back to iterate.

If there are several $\hat{\lambda}_j < 0$ one typically chooses the most negative one since the rate of decrease of q is proportional to $|\hat{\lambda}_j|$ if we remove constraint j (other heuristics possible).

Algorithm 16.1 (Active-set method for convex QP)

Example 16.3

```
Compute a feasible starting point $x_0$
$W_0 \leftarrow$ subset of the active constraints at $x_0$
for $k = 0, 1, 2, \ldots$
    $p_k = \arg\min_p \frac{1}{2} p^T G p + (G x_k + d)^T p$  s.t.  $a_i^T p = 0 \ \forall i \in W_k$      [Equality-constrained quadratic subproblem]
    if $p_k = 0$
        solve for $\hat{\lambda}_i$:  $\sum_{i \in W_k} a_i \hat{\lambda}_i = G(x_k + p_k) + d$      [Compute Lagrange multipliers for subproblem]
        if $\hat{\lambda}_i \geq 0 \ \forall i \in W_k \cap I$
            STOP with solution $x^* = x_k$      [All KKT conditions hold]
        else
            $j \leftarrow \arg\min_{j \in W_k \cap I} \hat{\lambda}_j$,  $W_{k+1} \leftarrow W_k \setminus \{j\}$      [Remove from the working set that inequality constraint having the most negative Lag. mult.]
            $x_{k+1} \leftarrow x_k$
        end
    else      [$p_k \neq 0$: we can move $x_k$ and decrease f]
        compute $\alpha_k = \min(1, \min \ldots)$ from (16.29)      [Largest step in [0,1]]
        $x_{k+1} \leftarrow x_k + \alpha_k p_k$
        if $\alpha_k < 1$      [There are blocking constraints]
            $W_{k+1} \leftarrow W_k \cup \{$one blocking constraint$\}$      [Add one of them to the working set]
        else
            $W_{k+1} \leftarrow W_k$
        end
    end
end
```

Q: does this algorithm become the simplex algorithm for $G = 0$ (LP)?

- If $\alpha_k > 0$ at each step, this algorithm converges in a finite number of iterations since there is a finite number of working sets. In rare situations the algorithm can cycle: a sequence of consecutive iterations results in no movement of x_k while the working set undergoes deletions and additions of indices and eventually repeats itself. Although this can be dealt with, most

QP implementations simply ignore it.

- It can be extended to deal with indefinite QP; however, the algorithm may get stuck at a feasible stationary points (which satisfy the KKT conditions though possibly not the 2nd-order ones).

✱ The gradient-projection method

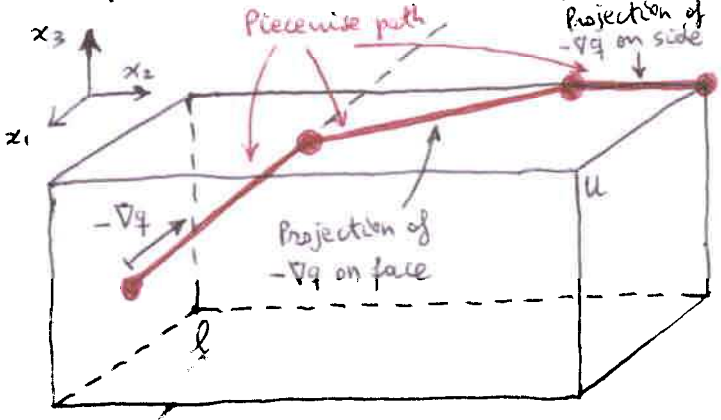
- Active-set method: the working set changes by only one index at each iteration, so many iterations are needed for large-scale problems.
- Gradient-projection method: large changes to the active set (from those constraints that are active at the current point, to those that are active at the Cauchy point).
- Most efficient on bound-constrained QP, on which we focus:

$$\min_{x} q(x) = \tfrac{1}{2}x^T G x + x^T d \quad \text{s.t.} \quad l \leq x \leq u$$

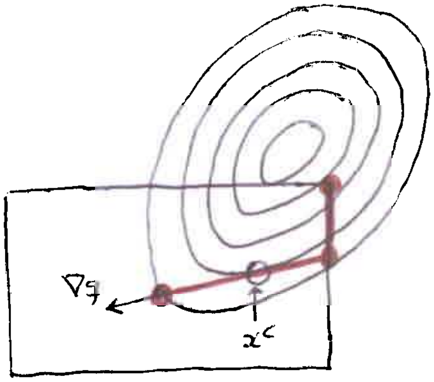
$x, l, u \in \mathbb{R}^n$; G symmetric (not necessarily pd); not all components need to be bounded.

- The feasible set is a box.
- Idea: steepest descent but bending along the box faces.
- Needs a feasible starting point x^0 (trivial to obtain); all iterates remain feasible.
- Each iteration consists of 2 stages; assume current point is x (which is feasible):

1) Find the Cauchy point x^c: this is the first minimiser along the steepest descent direction $-\nabla q = -(Gx+d)$ piecewise-bent to satisfy the constraints. To find it, search along $-\nabla q$; if we hit a bound (a box face), bend the direction (by projecting it on the face) and keep searching along it, and so on, resulting in a piecewise linear path (exact formulas in pp. 477–479).


x_3
x_2
x_1
Piecewise path
Projection of $-\nabla q$ on side
u
$-\nabla q$
Projection of $-\nabla q$ on face
l

x^c is somewhere in this path (depending on the quadratic form $q(x)$). Note there can be several minimisers along the path.



2) Approximate solution of QP subproblem where the active constraints are taken as equality constraints, i.e., the components of x^c that hit the bounds are kept fixed:

$$\min_{x} q(x) \quad \text{s.t.} \begin{cases} x_i = x_i^c, & i \in \mathcal{A}(x^c) \\ l_i \leq x_i \leq u_i, & i \notin \mathcal{A}(x^c) \end{cases}$$

This is almost as hard as the original QP problem; but all we need for global convergence is a point x^+ with $q(x^+) \le q(x^c)$ and is feasible wrt the subproblem constraints. x^c itself works. Something even better can be obtained by applying linear conjugate gradients to $\min q(x)$ s.t. $x_i = x_i^c$, $i \in \mathcal{A}(x^c)$ and stopping when either negative curvature appears, or a bound $l_i \le x_i \le u_i$, $i \notin \mathcal{A}(x^c)$ is violated.

- The gradient-projection method can be applied to general linear constraints (not just bounds), but finding the piecewise path costs much more computation, which is not worth.

Q: does this method converge in a finite number of iterations?

* Interior-point methods

- A simple extension of the primal-dual interior-point approach of LP works for convex QP. The algorithms are easy to implement and efficient for some problems.
- Consider for simplicity only inequality constraints (exercise 16.18 considers also equality ones):

$$\min_x q(x) = \frac{1}{2} x^T G x + x^T d \quad \text{s.t. } Ax \ge b \quad \text{with } G \text{ symmetric pd, } A_{m\times n}.$$

Introduce surplus vector $y = Ax - b \ge 0$.

Since the problem is convex, the KKT conditions are not only necessary but also sufficient. We find minimisers of the QP by finding roots of the KKT system:

$$\left.\begin{array}{r} \overset{\text{only addition wrt LP}}{\boxed{Gx}} - A^T\lambda + d = 0 \\ Ax - y - b = 0 \\ y^T\lambda = 0 \end{array}\right\} \begin{array}{l} \text{System of } n+2m \text{ equations} \\ \text{for } n+2m \text{ unknowns } x, y, \lambda \\ \text{(mildly nonlinear because of } y^T\lambda) \end{array} \Leftrightarrow F(x,y,\lambda) = \begin{pmatrix} Gx - A^T\lambda + d \\ Ax - y - b \\ Y\Lambda e \end{pmatrix} = 0$$

$$y, \lambda \ge 0 \qquad\qquad Y = \text{diag}(y_i),\ \Lambda = \text{diag}(\lambda_i),\ e = \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix}$$

- Central path $\mathcal{C} = \left\{(x_\tau, y_\tau, \lambda_\tau) : F(x_\tau, y_\tau, \lambda_\tau) = \begin{pmatrix} 0 \\ 0 \\ \tau e \end{pmatrix},\ \tau > 0\right\} \Leftrightarrow$ Solve KKT system with $y_i\lambda_i = \tau$.

Given a current iterate (x, y, λ) with $y, \lambda > 0$, define the duality measure $\mu = \frac{y^T\lambda}{m}$ (closeness to the boundary) and the centering parameter $\sigma \in [0,1]$.

- Newton-like step toward point $(x_{\sigma\mu}, y_{\sigma\mu}, \lambda_{\sigma\mu})$ on the central path:

$$\underbrace{\begin{pmatrix} G & -A^T & 0 \\ A & 0 & -I \\ 0 & Y & \Lambda \end{pmatrix}}_{\text{Jacobian of } F} \underbrace{\begin{pmatrix} \Delta x \\ \Delta\lambda \\ \Delta y \end{pmatrix}}_{\text{step}} = \underbrace{\begin{pmatrix} -r_d \\ -r_b \\ -\Lambda Y e + \sigma\mu e \end{pmatrix}}_{-F(x,y,\lambda)} \qquad \begin{array}{l} r_d = Gx - A^T\lambda + d \\ r_b = Ax - y - b \end{array}$$

$(x^{k+1}, y^{k+1}, \lambda^{k+1}) \leftarrow (x^k, y^k, \lambda^k) + \alpha_k(\Delta x^k, \Delta y^k, \Delta\lambda^k)$ choosing $\alpha_k \in [0,1]$ such that $y^{k+1}, \lambda^{k+1} > 0$.

- Likewise, we can extend the path-following methods (by defining a neighbourhood $\mathcal{N}_{-\infty}(\gamma)$) and Mehrotra's predictor-corrector algorithm.

CH. 17: PENALTY, BARRIER AND AUGMENTED LAGRANGIAN METHODS

Class of methods that ...	replace the original constrained problem by
- Quadratic penalty method - Log-barrier method - Augmented Lagrangian method (method of multipliers)	a sequence of unconstrained problems
- Exact penalty function methods	a single unconstrained problem
- Sequential linearly constrained methods	a sequence of linearly constrained problems
- Sequential quadratic programming (ch.18)	a sequence of QP problems

✱ The quadratic penalty method

$\min_x f(x)$ s.t. $\begin{cases} c_i(x) = 0, & i\in\mathcal{E} \\ c_i(x) \geq 0, & i\in\mathcal{I} \end{cases}$. Define the following quadratic-penalty function:

one term per constraint, which is positive when x violates c_i and 0 otherwise

$$Q(x;\mu) = \overbrace{f(x)}^{\text{objective function}} + \frac{1}{2\mu}\overbrace{\sum_{i\in\mathcal{E}} c_i^2(x)} + \frac{1}{2\mu}\overbrace{\sum_{i\in\mathcal{I}} \left([c_i(x)]^-\right)^2}$$

with penalty parameter $\mu > 0$, where $[y]^- = \max(-y, 0)$.

Define a sequence of unconstrained minimisation subproblems $\min_x Q(x;\mu_k)$ given a sequence $\{\mu_k\} \xrightarrow[k\to\infty]{} 0^+$. By driving μ to 0 we penalise the constraint violations with increasing severity, thereby forcing the minimiser of Q closer to the feasible region of the constrained problem. Algorithmic framework 17.1:

```
Given tolerance $\tau_0 > 0$, starting penalty parameter $\mu_0 > 0$, starting point $x_0^s$
for $k = 0, 1, 2, \ldots$
    Find an approximate minimiser $x_k$ of $Q(x;\mu_k)$ { starting at $x_k^s$
                                                          { terminating when $\|\nabla Q(x;\mu_k)\| \leq \tau_k$
    If final convergence test satisfied ⇒ STOP with approximate solution $x_k$
    Choose new { penalty parameter $\mu_{k+1} \in (0, \mu_k)$
               { starting point $x_{k+1}^s$
               { tolerance $\tau_{k+1} \in (0, \tau_k)$
end
```

Example 17.1

- Smoothness of the penalty terms:
  - Equality constraints: c_i^2 has at least WHY? as many derivatives as c_i ⇒ use derivative-based techniques for unconstrained optimisation.
  - Inequality constraints: $([c_i]^-)^2$ can be less smooth than c_i, e.g. for $x_1 \geq 0$, $([x_1]^-)^2 = \min(0, x_1)^2$ has a discontinuous second derivative.
- Starting point x_k^s for the minimisation of $Q(x;\mu_k)$: given by $x_{k-1}, x_{k-2}, \cdots$
- Choice of $\{\mu_k\}$: adaptive, e.g. if minimising $Q(x;\mu_k)$ was $\begin{cases} \text{expensive: modest decrease, eg. } \mu_{k+1} = 0.7\mu_k \\ \text{cheap: larger decrease, eg } \mu_{k+1} = 0.1\mu_k. \end{cases}$
- Convergence: assume $\mu_k \to 0^+$.
  - Th. 17.1: if each x_k is the exact global minimiser of $Q(x;\mu_k)$ ⇒ $\{x_k\}$ converges to a solution of the constrained problem. [Impractical: requires global minimisation]
  - Th. 17.2: if $\tau_k \to 0$, $x_k \to x^*$ and the gradient constraints $\nabla c_i(x^*)$ are l.i. ⇒ $-\frac{c_i(x_k)}{\mu_k} \to \lambda_i^*$ $\forall i \in \mathcal{E}$ and (x^*, λ^*) satisfy the KKT conditions. ↳ remember LICQ

    [Practical: only needs tolerances $\tau_k \to 0$] [Do proof]
  - When the problem is infeasible, often the quadratic-penalty method converges to a stationary point or minimiser of $\|c(x)\|^2$.

- Practical problems: the penalty function doesn't look quadratic around its minimiser except very close to it (see contours in fig. 17.2); and, even if $\nabla^2 f(x^*)$ is well-conditioned, the Hessian $\nabla^2_{xx} Q(x;\mu_k)$ becomes ill-conditioned as $\mu_k \to 0$. Consider equality constraints only and define $A(x)^T = (\nabla c_i(x))_{i \in \mathcal{E}}$ (matrix of constraint gradients): usually rank $(A) < n$

$$\nabla^2_{xx} Q(x;\mu_k) = \nabla^2 f(x) + \sum_{i \in \mathcal{E}} \frac{c_i(x)}{\mu_k} \nabla^2 c_i(x) + \frac{1}{\mu_k} A(x)^T A(x) \qquad \text{WHY?}$$

Near a minimiser, from th. 17.2 we have $\frac{c_i(x)}{\mu_k} \simeq -\lambda_i^*$ and so

$$\nabla^2_{xx} Q(x;\mu_k) \simeq \underbrace{\nabla^2_{xx} \mathcal{L}(x, \lambda^*)}_{\text{independent of } \mu_k} + \underbrace{\frac{1}{\mu_k} A(x)^T A(x)}_{\text{becomes very large as } \mu_k \to 0 \text{, with rank} < n}$$

where $\mathcal{L}(x,\lambda)$ is the Lagrangian funct.

Unconstrained optimisation methods have problems with ill-conditioning. For Newton's method we can apply the following reformulation that avoids the ill-conditioning:

Newton step   Introduce dummy vector ζ

$$\underbrace{\nabla^2_{xx} Q(x;\mu)\cdot p}_{\text{ill-conditioned} \Rightarrow \text{large error in } p} = -\nabla_x Q(x;\mu) \iff \underbrace{\begin{pmatrix} \nabla^2 f(x) + \sum_{i\in\mathcal{E}} \frac{c_i(x)}{\mu_k}\nabla^2 c_i(x) & A(x)^T \\ A(x) & -\mu_k I \end{pmatrix}}_{\text{well-conditioned as } \mu_k \to 0 \text{ WHY?}} \begin{pmatrix} p \\ \zeta \end{pmatrix} = \begin{pmatrix} -\nabla_x Q(x;\mu) \\ 0 \end{pmatrix}$$

p solves both systems WHY?

- In general, the method of multipliers is more effective since it tends to avoid ill-conditioning.

✱ The logarithmic barrier method

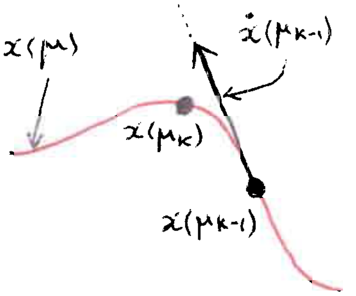
- Consider the inequality-constrained problem $\min_x f(x)$ s.t. $c_i(x) \geq 0$, $i \in \mathcal{I}$. Strictly feasible region $\mathcal{F}^0 = \{x \in \mathbb{R}^n : c_i(x) > 0 \; \forall i \in \mathcal{I}\}$, assume nonempty. Define the <u>log-barrier function</u> (through a <u>barrier parameter</u> $\mu > 0$):

$$P(x;\mu) = \underbrace{f(x)}_{\text{objective function}} - \underbrace{\mu \sum_{i \in \mathcal{I}} \log c_i(x)}_{\text{log-barrier function}}$$

log-barrier function: infinite everywhere except in $\mathcal{F}^0$; smooth inside $\mathcal{F}^0$; approaches ∞ as x approaches the boundary of $\mathcal{F}^0$

Ex. 17.2

- The minimiser $x(\mu)$ of $P(x;\mu)$ approaches a solution of the constrained problem as $\mu \to 0^+$.
- <u>Algorithmic framework 17.2 (log-barrier)</u>: ~~like~~ as for the quadratic penalty but choosing a new barrier parameter $\mu_{k+1} \in (0, \mu_k)$ instead of a new penalty parameter.
- The log-barrier function is smooth (if f, c_i are), so if $x(\mu) \in \mathcal{F}^0$, no constraints are active and we can use derivative-based techniques for unconstrained optimisation.
- Good starting point x_k^s for the minimisation of $P(x;\mu_k)$:
  - could extrapolate x_k from previous iterates $x_{k-1}, x_{k-2}, \cdots$; or
  - Extrapolate directly using the tangent to the path $\{x(\mu), \mu > 0\}$: $x_k^s = x_{k-1} + (\mu_k - \mu_{k-1})\dot{x}$. The path tangent $\dot{x} = \frac{dx(\mu)}{d\mu}$ can be obtained by total differentiation of $\nabla_x P(x;\mu) = 0$ wrt μ:



$$\underset{\uparrow}{0} = \frac{d \nabla_x P(x(\mu),\mu)}{d\mu} = \underbrace{\nabla^2_{xx} P(x;\mu)}_{\frac{\partial \nabla_x P}{\partial x}} \underbrace{\dot{x}}_{\frac{dx}{d\mu}} - \underbrace{\sum_{i\in\mathcal{I}} \frac{1}{c_i(x)} \nabla c_i(x)}_{\frac{\partial \nabla_x P}{\partial \mu}} \Bigg\} \text{ Linear system for vector } \dot{x}$$

Because $\nabla_x P(x(\mu),\mu) = 0$ by definition of $x(\mu)$

- Choice of $\{\mu_k\}$: adaptive, as with quadratic penalty.
- Convergence:
  - For convex programs: global convergence.

    Th. 17.3: $f, \{-c_i, i \in I\}$ convex functions, $\mathcal{F}^\circ \neq \phi \Rightarrow$

    a) For any $\mu > 0$, $P(x;\mu)$ is convex in $\mathcal{F}^\circ$ and attains a minimiser $x(\mu)$ (not necessarily unique) on $\mathcal{F}^\circ$; any local minimiser $x(\mu)$ is also global.

    b) If the set of solutions of the constrained optimisation problem is nonempty and bounded and if $\{\mu_k\}$ is a decreasing sequence with $\mu_k \to 0 \Rightarrow \{x(\mu_k)\}$ converges to a solution x^* and $f(x(\mu_k)) \to f^*$, $P(x(\mu_k);\mu_k) \to f^*$.

    [If there are no solutions or the solution set is unbounded the theorem may not apply.]
  - For general inequality-constrained problems: local convergence.

    Th. 17.4: $\mathcal{F}^\circ \neq \phi$, (x^*, λ^*) = (local solution, Lagrange multiplier) at which KKT + LICQ + 2nd-order sufficient conditions hold (≡ well-behaved solution) $\Rightarrow$

    a) For all sufficiently small μ, $\exists!$ continuously differentiable function $x(\mu)$: $x(\mu)$ is a local minimiser of $P(x;\mu)$ and $\nabla^2_{xx} P(x;\mu)$ is pd.

    b) $(x(\mu), \lambda(\mu)) \xrightarrow[\mu \to 0]{} (x^*, \lambda^*)$ where $\lambda_i(\mu) = \frac{\mu}{c_i(x(\mu))}$, $i \in I$.

    [This means there may be sequences of minimisers of $P(x;\mu)$ that don't converge to a solution as $\mu \to 0$.]
- Relation between the minimisers of $P(x;\mu)$ and a solution (x^*, λ^*): at a minimiser $x(\mu)$,

$$\nabla_x P(x(\mu);\mu) = 0 = \nabla f(x(\mu)) - \sum_{i \in I} \underbrace{\boxed{\frac{\mu}{c_i(x(\mu))}}}_{\text{define as } \lambda(\mu)} \nabla c_i(x(\mu)) = \nabla f(x(\mu)) - \sum_{i \in I} \lambda_i(\mu) \nabla c_i(x(\mu))$$

which is KKT condition a) for the constrained problem ($\nabla_{xx} \mathcal{L}(x,\lambda) = 0$). As for the other KKT conditions at $x(\mu)$, $\lambda(\mu)$: b) ($c_i(x) \geq 0$, $i \in I$) and c) ($\lambda_i \geq 0$, $i \in I$) also hold since $c_i(x(\mu)) > 0$; and the complementarity condition d) fails: $\lambda_i c_i(x) = \mu > 0$, but it holds as $\mu \to 0$. The path $\mathcal{C}_p = \{x(\mu) : \mu > 0\}$ is called (primal) central path.

- Practical problems: as with the quadratic-penalty method, the barrier function looks quadratic only very near its minimiser; and the Hessian $\nabla^2_{xx} P(x;\mu_k)$ becomes ill-conditioned as $\mu_k \to 0$:

$$\nabla_x P(x;\mu) = \nabla f(x) - \sum_{i\in I} \frac{\mu}{c_i(x)} \nabla c_i(x)$$

$$\nabla^2_{xx} P(x;\mu) = \nabla^2 f(x) - \sum_{i\in I} \frac{\mu}{c_i(x)} \nabla^2 c_i(x) + \sum_{i\in I} \frac{\mu}{c_i^2(x)} \nabla c_i(x) \nabla c_i(x)^T$$

Near a minimizer $x(\mu)$ with μ small, from th. 17.4 we have that the optimal Lagrange multiplier can be estimated as $\lambda_i^* \simeq \frac{\mu}{c_i(x)} \Rightarrow$

$$\nabla^2_{xx} P(x;\mu) \simeq \underbrace{\nabla^2_{xx} \mathcal{L}(x,\lambda^*)}_{\text{independent of } \mu} + \underbrace{\sum_{i\in I} \frac{1}{\mu} (\lambda_i^*)^2 \nabla c_i(x) \nabla c_i(x)^T}_{\substack{\text{for the active constraints } (\lambda_i^* \neq 0), \\ \text{becomes very large as } \mu \to 0, \text{ with rank} < n}}$$

$\boxed{\text{Ex. 17.3}}$

However, the effects of ill-conditioning are less severe than in the quadratic-penalty method.

The Newton step can be reformulated as before to avoid ill-conditioning, and it should be implemented with line-search or trust-region strategy to remain strictly feasible.

- <u>Relation to primal-dual interior-point methods</u>: write KKT conditions for the constrained problem and introduce slack variables s_i, $i \in I$ (where $c(x) = \begin{pmatrix} c_1(x) \\ \vdots \\ c_m(x) \end{pmatrix}$):

$$\left.\begin{aligned} \nabla f(x) - \sum_{i\in I} \lambda_i \nabla c_i(x) &= 0 \\ c(x) - s &= 0 \\ \lambda_i s_i &= \mu,\ i \in I \\ \lambda, s &\geq 0 \end{aligned}\right\} \text{Nonlinear system of equations} \quad F_\mu(x,\lambda,s) = \begin{pmatrix} \nabla f(x) - A(x)^T \lambda \\ c(x) - s \\ \Lambda S e - \mu e \end{pmatrix} = 0$$

$\Lambda = \operatorname{diag}(\lambda_i)$, $S = \operatorname{diag}(s_i)$, $e = \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix}$, $A(x)^T = \left(\nabla c_i(x)\right)_{i\in I}$

The solution of the system as a function of μ traces the <u>primal-dual central path</u> $\mathcal{C}_{pd} = \{x(\mu), \lambda(\mu), s(\mu) : \mu > 0\}$ whose projection on the primal variables is the primal central path $\mathcal{C}_p$. $\mathcal{C}_{pd}$ steers through the interior of the primal-dual feasible set, avoiding spurious solutions that satisfy the nonlinear system of equations but not $\lambda, s > 0$, and converges to a solution as $\mu \to 0$. In primal-dual interior-point algorithms we solve the nonlinear system with Newton's method (the interior-point methods for LP and QP are particular cases of this). Newton step: $J_k \begin{pmatrix} \Delta x \\ \Delta\lambda \\ \Delta s \end{pmatrix} = -F_\mu(x_k, \lambda_k, s_k) \Rightarrow \begin{pmatrix} x \\ \lambda \\ s \end{pmatrix}_{k+1} = \begin{pmatrix} x \\ \lambda \\ s \end{pmatrix}_k + \alpha \begin{pmatrix} \Delta x \\ \Delta\lambda \\ \Delta s \end{pmatrix}$.

In the log-barrier method, we eliminate s and λ directly from the system:

$$\left.\begin{aligned} c(x) - s &= 0 \\ \lambda_i s_i &= \mu \end{aligned}\right\} \Rightarrow \lambda_i = \frac{\mu}{c_i(x)} \Rightarrow \nabla f(x) - \sum_{i\in I} \frac{\mu}{c_i(x)} \nabla c_i(x) = 0 \Leftrightarrow \min_x f(x) - \mu \sum_{i\in I} \log c_i(x)$$

$i \in I$

- Equality constraints: $\min_{x} f(x)$ s.t. $\begin{cases} c_i(x)=0, & i\in\mathcal{E} \\ c_i(x)\geq 0, & i\in\mathcal{I} \end{cases}$

Splitting an equality constraint $c_i(x)=0$ as two inequalities $c_i(x)\geq 0$, $-c_i(x)\geq 0$ doesn't work (WHY?) but we can combine the quadratic penalty and the log-barrier: $B(x;\mu) = f(x) - \mu \sum_{i\in\mathcal{I}} \log c_i(x) + \frac{1}{2\mu}\sum_{i\in\mathcal{E}} c_i^2(x)$. This has similar aspects to the quadratic penalty and barrier methods: algorithm = successive reduction of μ alternated with approximate minimisation of B wrt x; ill-conditioned $\nabla^2_{xx} B$ when μ is small; etc.

- To find an initial point which is strictly feasible wrt inequality constraints, introduce slack variables s_i, $i\in\mathcal{I}$: $\min_{x,s} f(x)$ s.t. $\left\{\begin{array}{ll} c_i(x)=0, & i\in\mathcal{E} \\ c_i(x)-s_i=0, & i\in\mathcal{I} \\ s_i\geq 0, & i\in\mathcal{I} \end{array}\right\} \Rightarrow$

$$B(x,s;\mu) = f(x) - \mu\sum_{i\in\mathcal{I}} \log s_i + \frac{1}{2\mu}\sum_{i\in\mathcal{E}} c_i^2(x) + \frac{1}{2\mu}\sum_{i\in\mathcal{I}} (c_i(x)-s_i)^2.$$

Now, any point $\binom{x}{s}$ with $s>0$ lies in the domain of B.

* Exact penalty functions

- Exact penalty function $\phi(x;\mu)$: $\exists \mu^* > 0: \forall \mu\in(0,\mu^*]$, any local solution x of the constrained problem is a local minimiser of ϕ. So we need a single unconstrained minimisation of $\phi(x;\mu)$ for such a $\mu\in(0,\mu^*]$.
- The quadratic-penalty and log-barrier functions are not exact, so we need $\mu\to 0$.
- The l_1 exact penalty function $\phi_1(x;\mu) = f(x) + \frac{1}{\mu}\sum_{i\in\mathcal{E}}|c_i(x)| + \frac{1}{\mu}\sum_{i\in\mathcal{I}}[c_i(x)]^-$ is exact for $\frac{1}{\mu^*}$ = largest Lagrange multiplier (in absolute value) associated with an optimal solution. Algorithms based on minimising ϕ_1 need:
  - Rules for adjusting μ to ensure $\mu<\mu^*$.
  - Special techniques to deal with the fact that ϕ_1 is not differentiable at any x for which $c_i(x)=0$ for some $i\in\mathcal{E}\cup\mathcal{I}$ (and such x must be encountered).

* Augmented Lagrangian method (method of multipliers)

- Modification of the quadratic-penalty method to reduce the possibility of ill-conditioning by introducing explicit estimates of the Lagrange multipliers at each iterate.
- Tends to yield less ill-conditioned subproblems than the log-barrier method and doesn't

need strictly feasible iterates for the inequality constraints.

- Consider first the equality-constrained problem. In the quadratic-penalty method, the minimiser x_k of $Q(x;\mu_k)$ satisfies $c_i(x_k) \simeq -\mu_k \lambda_i^* \;\forall i \in \mathcal{E}$ (from th. 17.2) and so $c_i(x_k) \xrightarrow[\mu_k \to 0]{} 0$. Idea: redefine Q so that its minimiser x_k satisfies $c_i(x_k) \simeq 0$ (i.e., the subproblem solution better satisfies the equality constraint); this way we'll get better iterates when μ_k is not so small and delay the appearance of ill-conditioning.

- Define the augmented Lagrangian function by adding a quadratic penalty to the Lagrangian:

$$\mathcal{L}_A(x,\lambda;\mu) = f(x) - \sum_{i\in\mathcal{E}} \lambda_i c_i(x) + \frac{1}{2\mu} \sum_{i\in\mathcal{E}} c_i^2(x).$$

Considering this as a quadratic-penalty function with objective function $f(x) - \sum_{i\in\mathcal{E}} \lambda_i c_i(x)$, the minimiser x_k of $\mathcal{L}_A$ for $\lambda = \lambda_i^k$ satisfies (th. 17.2) $c_i(x_k) \simeq -\mu_k(\lambda_i^* - \lambda_i^k) \;\forall i \in \mathcal{E}$. WHY? S if λ^k is close to the optimal multiplier vector λ^* then $\|c(x_k)\|$ will be much smaller than μ_k rather than just proportional to μ_k. Also note that $\mathcal{L}_A(x,\lambda^*;\mu)$ is an exact penalty function ~~[illegible]~~ (of course, we don't know λ^*). WHY?

Now we need an update equation for λ^{k+1} so that it approximates λ^* more and more accurately; the relation $c(x_k) \simeq -\mu_k(\lambda^* - \lambda^k)$ suggests $\lambda^{k+1} \leftarrow \lambda^k - \frac{c(x_k)}{\mu_k}$.

Example 17.4

- Algorithmic framework 17.3 (method of multipliers — equality constraints): as for the quadratic-penalty method but using $\mathcal{L}_A(x,\lambda;\mu)$ and updating $\lambda^{k+1} \leftarrow \lambda^k - \frac{c(x_k)}{\mu_k}$ where x_k is the (approximate) minimiser of $\mathcal{L}_A(x,\lambda^k;\mu_k)$ and with given starting point λ^0.

- Choice of starting point x_k^s for the minimisation of $\mathcal{L}_A(x,\lambda^k;\mu_k)$ is less critical now (less ill-conditioning), so we can simply take $x_{k+1}^s \leftarrow x_k^s$.

- Extension to inequality constraints (assume no equality constraints for simplicity):
  - Introduce slack variables: $c_i(x) \geq 0 \Rightarrow c_i(x) - s_i = 0,\; s_i \geq 0 \;\forall i \in \mathcal{I}$.
  - Define the augmented Lagrangian only for the equality constraints:

$$\mathcal{L}_A(x,s,\lambda;\mu) = f(x) - \sum_{i\in\mathcal{I}} \lambda_i (c_i(x) - s_i) + \frac{1}{2\mu}\sum_{i\in\mathcal{I}} (c_i(x) - s_i)^2.$$

  - Define the bound-constrained subproblem: $\min_{x,s} \mathcal{L}_A(x,s,\lambda;\mu)$ s.t. $s_i \geq 0 \;\forall i \in \mathcal{I}$.

One approach to solve this subproblem (implemented in the LANCELOT package) is to use the gradient-projection method. Another approach is to apply "coordinate descent" first in s, then in x:

- In s: $\min_s \mathcal{L}_A$ s.t. $s_i \geq 0\ \forall i \in \mathcal{I}$ $\Rightarrow$ Solution: $s_i = \max(0, c_i(x) - \mu_k \lambda_i^k)\ \forall i \in \mathcal{I}$ because $\mathcal{L}_A$ is a convex quadratic form on s. WHY?
- In x: substitute the solution s_i in $\mathcal{L}_A$ to obtain WHY?

$$\mathcal{L}_A(x, \lambda^k; \mu_k) = f(x) + \sum_{i \in \mathcal{I}} \psi(c_i(x), \lambda_i^k; \mu_k) \quad \text{where} \quad \psi(\underbrace{t, \sigma; \mu}_{\text{all scalar}}) = \begin{cases} -\sigma t + \frac{t^2}{2\mu} & \text{if } t - \mu\sigma \leq 0 \\ -\frac{\mu\sigma^2}{2} & \text{otherwise} \end{cases}$$

Now solve approximately the unconstrained problem $\min_x \mathcal{L}_A(x, \lambda^k; \mu_k)$. Note that ψ doesn't have a second derivative when $t = \sigma\mu$ (WHY?) $\Leftrightarrow$ $c_i(x) = \lambda_i \mu$ for some $i \in \mathcal{I}$. Fortunately this rarely happens, since from the strict complementarity KKT condition (if it holds) exactly one of λ_i^* and $c_i(x^*)$ is 0, so the iterates should stay away from points at which $c_i(x_k) = \lambda_i^k \mu_k$. Thus it is safe to use Newton's method. For a weakly active constraint $c_i(x^*) = \lambda_i^* \mu = 0$ does hold.

- Finally, update the Lagrange multipliers as $\lambda_i^{k+1} \leftarrow \max\left(\lambda_i^k - \frac{c_i(x_k)}{\mu_k}, 0\right)\ \forall i \in \mathcal{I}$ (since $\lambda_i \geq 0$ for the KKT conditions to hold at the solution).

- Convergence: consider only equality constraints for simplicity. Th. 17.5: (x^*, λ^*) = (local solution, Lagrange multiplier) at which KKT + LICQ + 2nd-order sufficient conditions hold ($\equiv$ well-behaved solution) $\Rightarrow \exists \bar{\mu} > 0$: $\forall \mu \in (0, \bar{\mu}]$, x^* is a strict local minimiser of $\mathcal{L}_A(x, \lambda^*; \mu)$. [Thus $\mathcal{L}_A$ is an exact penalty function for the optimal Lagrange multiplier and we need not take $\mu \to 0$]. In practice, we need to estimate λ over iterates and drive μ sufficiently small.

* Sequential linearly constrained methods (SLC)

- Idea: at each step, linearise constraints and minimise the Lagrangian subject to them.
- Mainly used for large problems and particularly effective when most of the constraints are linear; implemented in MINOS package. Solving the subproblem is hard; SQP methods are generally preferable.
- Consider first equality constraints only. The SLC subproblem is $\min_x F_k(x)$ s.t. $\nabla c_i^T(x)(x - x_k) + c_i(x_k) = 0$, $i \in \mathcal{E}$, where

$$F_k(x) = \overbrace{f(x) - \sum_{i \in \mathcal{E}} \lambda_i^k \bar{c}_i^k(x)}^{\text{Lagrangian}} + \overbrace{\frac{1}{2\mu} \sum_{i \in \mathcal{E}} (\bar{c}_i^k(x))^2}^{\text{Quadratic penalty}} \quad \text{with penalty parameter } \mu > 0$$

λ^k = current Lagrange multiplier estimate = Lagrange multiplier for the SLC subproblem at $k-1$

$\bar{c}_i^k(x) = c_i(x) - (c_i(x_k) + \nabla c_i(x_k)^T (x - x_k))$ = true − linearised

- For inequality constraints: introduce slacks, transform constraints into equalities and bounds; solve the SLC subproblem subject to them.

CH. 18: SEQUENTIAL QUADRATIC PROGRAMMING (SQP)

- One of the most effective approaches for nonlinearly constrained optimisation, large or small.

* Local SQP method

General nonlinear programming problem: $\min_x f(x)$ s.t. $\begin{cases} c_i(x) = 0, & i \in \mathcal{E} \\ c_i(x) \geq 0, & i \in \mathcal{I} \end{cases}$

Linearise the objective function and constraints to obtain the QP subproblem:

$$\min_p \ \frac{1}{2} p^T W_k p + \nabla f_k^T p \quad \text{s.t.} \begin{cases} \nabla c_i(x_k)^T p + c_i(x_k) = 0, & i \in \mathcal{E} \\ \nabla c_i(x_k)^T p + c_i(x_k) \geq 0, & i \in \mathcal{I} \end{cases}$$

where $W_k = \nabla^2_{xx} \mathcal{L}(x_k, \lambda_k)$ is the Hessian of the Lagrangian $\mathcal{L}(x,\lambda) = f(x) - \lambda^T c(x)$.

- Algorithm 18.1 (local SQP algorithm)

```
Given initial $x_0$, $\lambda_0$
for $k = 0, 1, 2, \dots$
    Evaluate $\nabla f_k$, $c_i(x_k)$, $\nabla c_i(x_k)$, $W_k = \nabla^2_{xx} \mathcal{L}(x_k, \lambda_k)$
    $(p_k, \lambda_{k+1}) \leftarrow$ (solution, Lagrange multiplier) of QP subproblem
    $x_{k+1} \leftarrow x_k + p_k$
    if convergence test satisfied $\Rightarrow$ STOP with approximate solution $(x_{k+1}, \lambda_{k+1})$.
end
```

- Intuitive idea: the QP subproblem is Newton's method applied to the optimality conditions of the problem. Consider only equality constraints for simplicity and write $\min_x f(x)$ s.t. $c(x) = 0$ with $c(x) = \begin{pmatrix} c_1(x) \\ \vdots \\ c_m(x) \end{pmatrix}$ and $A(x)^T = \left(\nabla c_1(x) \dots \nabla c_m(x) \right)$.

(i) The solution $\begin{pmatrix} p_k \\ \mu_k \end{pmatrix}$ of the QP subproblem satisfies $\begin{cases} W_k p_k + \nabla f_k - A_k^T \mu_k = 0 \\ A_k p_k + c_k = 0 \end{cases} \Leftrightarrow \begin{pmatrix} W_k & -A_k^T \\ A_k & 0 \end{pmatrix} \begin{pmatrix} p_k \\ \mu_k \end{pmatrix} = \begin{pmatrix} -\nabla f_k \\ -c_k \end{pmatrix}$.

(ii) The KKT system for the problem is $F(x,\lambda) = \begin{pmatrix} \overbrace{\boxed{\nabla f(x) - A(x)^T \lambda}}^{\nabla_x \mathcal{L}(x,\lambda)} \\ c(x) \end{pmatrix} = 0$, for which Newton's method (for root-finding) results in a step

$$\begin{pmatrix} x_{k+1} \\ \lambda_{k+1} \end{pmatrix} = \begin{pmatrix} x_k \\ \lambda_k \end{pmatrix} + \begin{pmatrix} p_k \\ p_\lambda \end{pmatrix} \quad \text{where} \quad \underbrace{\begin{pmatrix} W_k & -A_k^T \\ A_k & 0 \end{pmatrix}}_{\text{Jacobian of } F \text{ at } \begin{pmatrix} x_k \\ \lambda_k \end{pmatrix}} \begin{pmatrix} p_k \\ p_\lambda \end{pmatrix} = \underbrace{\begin{pmatrix} -\nabla f_k + A_k^T \lambda_k \\ -c_k \end{pmatrix}}_{-F(x_k, \lambda_k)}$$

(i) and (ii) are equivalent, since the two linear systems have the same solution (define $\mu_k = p_k + \lambda_k$).

- Assumptions: (recall lemma 16.1 in ch. 16 about equality-constrained QP)
  - The constraint Jacobian A_k has full row rank (LICQ)
  - W_k is pd on the tangent space of the constraints ($d^T W_k d > 0$ $\forall d \neq 0$, $A_k d = 0$)

  ⇒ the KKT matrix is nonsingular and the linear system has a unique solution.

If the problem solution satisfies the 2nd-order sufficient conditions then these assumptions hold locally (near the solution) and Newton's method converges quadratically.

- To ensure global convergence (≡ from remote starting points), Newton's method needs to be modified (just as in the unconstrained optimisation case). This includes defining a merit function (which evaluates the goodness of an iterate, trading off reducing the objective function but improving feasibility) and applying the strategies of:
  - Line search: modify the Hessian of the quadratic model to make it pd, so that p_k is a descent direction for the merit function.
  - Trust region: limit the step size to a region so that the step produces sufficient decrease of the merit function (the Hessian need not be pd).

Additional issues need to be accounted for, eg. the linearisation of inequality constraints may produce an infeasible subproblem (example: linearising $x \leq 1$, $x^2 \geq 0$ at $x_k = 3$ results in $3 + p \leq 1$, $9 + 6p \geq 0$ which is inconsistent).

MAXIMUM LIKELIHOOD ESTIMATION: GAUSS-NEWTON & METHOD OF SCORING

* Maximum likelihood estimation of parameters x given observations $\{t_i\}$: $\max_x \frac{1}{N}\sum_{i=1}^N \log p(t_i;x)$ where $p(t;x)$ is a pmf or pdf in t. Call $L(t;x) = \log p(t;x)$. Then (all derivatives are wrt x in this section, and assume we can interchange ∇ and $\int$):

Gradient $\nabla L = \frac{1}{p}\nabla p$

Hessian $\nabla^2 L = -\frac{1}{p^2}\nabla p \nabla p^T + \frac{1}{p}\nabla^2 p = -\nabla \log p \, \nabla \log p^T + \frac{1}{p}\nabla^2 p$

Taking expectations wrt the model $p(t;x)$ we have:

$$E\{-\nabla^2 L\} = E\{\nabla \log p \, \nabla \log p^T\} + \cancel{E\{\tfrac{1}{p}\nabla^2 p\}} = \mathrm{cov}\{\nabla \log p\}$$

since $E\{\nabla \log p\} = \int p \frac{1}{p}\nabla p = \nabla \int p = 0$ and $E\{\frac{1}{p}\nabla^2 p\} = \int p \frac{1}{p}\nabla^2 p = \nabla^2 \int p = 0$.

In statistical parlance:

- Observed information: $-\nabla^2 L$
- Expected information: $E\{-\nabla^2 L\} = E\{\nabla \log p \, \nabla \log p^T\}$ (Fisher information matrix)
- Score: $\nabla \log p = \nabla L$

Two ways of approximating the log-likelihood Hessian $\frac{1}{N}\sum_{i=1}^N \nabla^2 \log p(t_i;x)$ using only the first-order term on $\nabla \log p$:

- Gauss-Newton: sample observed information $J(x) = \frac{1}{N}\sum_{i=1}^N \nabla \log p(t_i;x) \nabla \log p(t_i;x)^T$.
- Method of scoring: expected information $J(x) = E\{\nabla \log p \, \nabla \log p^T\}$. This requires computing an integral, but its form is often much simpler than that of the observed information (eg for the exponential family).

Advantages:

- Good approximation to the Hessian (the second-order term is small on average if the model fits well the data)
- Cheaper to compute (only requires first derivatives)
- Positive definite so descent directions.
- Particularly simple expressions for the exponential family: $p(t;x) = \frac{f(t)}{g(x)} e^{h(t)^T \phi(x)}$ with sufficient statistic $h(t)$ and partition function $g(x) = \int f(t) e^{h(t)^T \phi(x)} dt$.

  $\nabla \log p = -\frac{1}{g}\nabla g + \nabla \phi(x) h(t) = \nabla \phi(x)(h(t) - E\{h(t)\}) \Rightarrow$

  log-likelihood gradient $\frac{1}{N}\sum_{i=1}^N \nabla \log p(t_i;x) = \nabla \phi(x)(E_{\text{data}}\{h\} - E_{\text{model}}\{h\})$.

  Typically $\phi(x) = x$ so $\nabla \phi(x) = I$, in which case $E\{-\nabla^2 L\} = -\nabla^2 L$.

* For a missing-data problem where t are observed and z are missing we have $p(t;x) = \int p(t|z;x)\, p(z;x)\, dz$ (eg. z = label of mixture component). We have:

$$\nabla \log p(t;x) = \frac{1}{p(t;x)} \int \left(\nabla p(t|z;x) \cdot p(z;x) + p(t|z;x)\, \nabla p(z;x) \right) dz =$$

$$= \frac{1}{p(t;x)} \int p(z;x)\, p(t|z;x) \left(\nabla \log p(t|z;x) + \nabla \log p(z;x) \right) dz = E_{z|t} \{ \nabla \log p(t,z;x) \}$$

= posterior expectation of the complete-data log-likelihood gradient.

$$\nabla^2 \log p(t;x) = \nabla \int p(z|t;x)\, \nabla \log p(t,z;x)\, dz =$$

$$= E_{z|t} \{ \nabla^2 \log p(t,z;x) \} + \int \nabla p(z|t;x)\, \nabla \log p(t,z;x)^T\, dz.$$

Noting that $\nabla p(z|t;x) = \nabla \left(\frac{p(t,z;x)}{p(t;x)} \right) = \frac{1}{p(t;x)} \left(\nabla p(t,z;x) - p(z|t;x)\, \nabla \log p(t;x) \right) =$

$= p(z|t;x) \left(\nabla \log p(t,z;x) - \nabla \log p(t;x) \right)$, then the second term is:

$$E_{z|t} \{ (\nabla \log p(t,z;x) - \nabla \log p(t;x))\, \nabla \log p(t,z;x)^T \} = \operatorname{cov}_{z|t} \{ \nabla \log p(t,z;x) \}$$

since $\nabla \log p(t;x) = E_{z|t} \{ \nabla \log p(t,z;x) \}$.

Finally:

Gradient $\nabla \log p(t;x) = E_{z|t} \{ \nabla \log p(t,z;x) \}$

Hessian $\nabla^2 \log p(t;x) = \underbrace{E_{z|t} \{ \nabla^2 \log p(t,z;x) \}}_{\text{posterior expectation of complete-data loglikelihood Hessian}} + \underbrace{\operatorname{cov}_{z|t} \{ \nabla \log p(t,z;x) \}}_{\text{posterior covariance of the complete-data log-likelihood gradient.}}$

Again, ignoring the second-order term we obtain a cheap, positive-definite approximation to the Hessian (Gauss-Newton method) — but for minimising the likelihood, not for maximising it!

We can still use the first-order, negative-definite approximation from before:

$$\nabla^2 \log p(t; x) \simeq -\nabla \log p(t; x)\, \nabla \log p(t; x)^T.$$

* Relation with the EM (expectation-maximisation) algorithm:

- E step: compute $p(z|t; x^{old})$ and
$$Q(x; x^{old}) = E_{z|t; x^{old}}\{\log p(t, z; x)\}$$
- M step: $\max\limits_{x} Q(x; x^{old})$

We have $\nabla \log p(t; x) = \left.\frac{\partial Q(x'; x)}{\partial x'}\right|_{x'=x} = E_{z|t}\{\nabla \log p(t, z; x)\}$.

Final comments

Fundamental ideas underlying most methods:

- Optimality conditions (KKT, 2nd-order):
  - check whether a point is a solution
  - suggest algorithms
- Sequence of subproblems that converges to our problem, where each subproblem is easy:
  - line search
  - trust region
  - quadratic penalty, log-barrier, augmented Lagrangian
  - interior-point
  - SQP
- Simpler function valid near current iterate (e.g. linear, quadratic with Taylor's th.): allows to predict the function locally.
- Derivative $\approx$ finite difference (e.g. secant equation).
- Approximate vs exact solution of the subproblem: want to minimise overall computation.
- Heuristics are useful to invent algorithms, but they must be backed by theory guaranteeing good performance (e.g. l.s. heuristics are ok as long as Wolfe conditions hold).
- Problem-dependent heuristics:
  - Restarts in nonlinear conjugate gradients
  - How accurately to solve the subproblem (forcing sequences, tolerances)
  - How to choose the centering parameter σ in interior-point methods
  - How to decrease the penalty parameter μ in quadratic penalty, log-barrier, augmented Lagrangian

Given your particular optimisation problem:

- No best method in general; use your understanding of basic methods and fundamental ideas to choose an appropriate method, or to design your own.
- Simplify the problem if possible.
- Try to guess good initial iterates.
- Recognise the type of optimisation problem: smooth? LP? QP? Convex? Multiple optima?
- Evaluate costs (time & memory):
  - computing the Hessian
  - solving linear systems (sparse?)