

EECS260 Optimization — Lecture notes

Based on “Numerical Optimization” (Nocedal & Wright, Springer, 2nd ed., 2006)

Miguel Á. Carreira-Perpiñán
EECS, University of California, Merced

May 2, 2010

1 Introduction

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

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

\mathbf{x} : variables (vector); $f(\mathbf{x})$: objective function (scalar).

Feasible region: set of points satisfying all constraints.

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

- Ex. (fig. 1.1): $\min (x_1 - 2)^2 + (x_2 - 1)^2$ s.t. $\begin{cases} x_1^2 - x_2 \leq 0 \\ x_1 + x_2 \leq 2. \end{cases}$
- Ex.: transportation problem (LP)

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

c_{ij} : shipping cost; x_{ij} : amount of product shipped from factory i to shop j .

- Ex.: LSQ problem: fit a parametric model (e.g. line, polynomial, neural net...) to a data set. Ex. 2.1
- Optimization 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 optimization (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 \rightarrow solution found very slowly or not at all.
- Not covered in the Nocedal-Wright book, or in this course:

- Discrete optimization (integer programming): the variables are discrete. Ex.: integer transportation problem, traveling 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 specialized 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.
- Non-smooth opt: discontinuous derivatives, e.g. L_1 -norm.
Ref: Fletcher.
- Stochastic opt: the model is specified with uncertainty, e.g. $x \leq b$ 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 to a single objective = linear combinations of objectives.
- EM algorithm (Expectation-Maximization): specialized technique for maximum likelihood estimation of probabilistic models.
Ref: McLachlan & Peel; many books on statistics or machine learning.
- Calculus of variations: stationary point of a functional (= function of functions).
- Convex optimization: we’ll see some of this.
Ref: Boyd & Vandenberghe.
- Modeling: 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 optimization (see syllabus).

2 Fundamentals of unconstrained optimization

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

Conditions for a local minimum \mathbf{x}^* (cf. case $n = 1$)



- Global minimizer: $f(\mathbf{x}^*) \leq f(\mathbf{x}) \forall \mathbf{x} \in \mathbb{R}^n$.
- Local minimizer: \exists neighborhood \mathcal{N} of \mathbf{x}^* : $f(\mathbf{x}^*) \leq f(\mathbf{x}) \forall \mathbf{x} \in \mathcal{N}$.
- Strict (or strong) global minimizer: $f(\mathbf{x}^*) < f(\mathbf{x}) \forall \mathbf{x} \in \mathcal{N} \setminus \{\mathbf{x}^*\}$. (Ex. $f(x) = 2$ vs $f(x) = (x-2)^4$.)
- Isolated local minimizer: $\exists \mathcal{N}$ of \mathbf{x}^* such that \mathbf{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): \mathbf{x}^* local min, f cont. diff. in an open neighborhood of $\mathbf{x}^* \Rightarrow \nabla f(\mathbf{x}^*) = \mathbf{0}$ (Pf.: by contradiction: if $\nabla f(\mathbf{x}^*) \neq \mathbf{0}$ then f decreases along the gradient direction.) (Not sufficient condition, ex: $f(x) = x^3$.)
- Stationary point: $\nabla f(\mathbf{x}^*) = \mathbf{0}$.
- Second-order necessary conditions (Th. 2.3): \mathbf{x}^* is local min, f twice cont. diff. in an open neighborhood of $\mathbf{x}^* \Rightarrow \nabla f(\mathbf{x}^*) = \mathbf{0}$ and $\nabla^2 f(\mathbf{x}^*)$ is psd. (Pf.: by contradiction: if $\nabla^2 f(\mathbf{x}^*)$ is not psd then f decreases along the direction where ∇^2 is not psd.)
- Second-order sufficient conditions (Th. 2.4): $\nabla^2 f$ cont. in an open neighborhood of \mathbf{x}^* , $\nabla f(\mathbf{x}^*) = \mathbf{0}$, $\nabla^2 f(\mathbf{x}^*)$ pd $\Rightarrow \mathbf{x}^*$ is a strict local minimizer of f . (Pf.: Taylor-expand f around \mathbf{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 \mathbf{x} .

Convex optimization

- $\mathcal{S} \subset \mathbb{R}^n$ is a convex set if $\mathbf{x}, \mathbf{y} \in \mathcal{S} \Rightarrow \alpha \mathbf{x} + (1 - \alpha) \mathbf{y} \in \mathcal{S}, \forall \alpha \in [0, 1]$.
- $f : \mathcal{S} \subset \mathbb{R}^n \rightarrow \mathbb{R}$ is a convex function if its domain \mathcal{S} is convex and $f(\alpha \mathbf{x} + (1 - \alpha) \mathbf{y}) \leq \alpha f(\mathbf{x}) + (1 - \alpha) f(\mathbf{y}), \forall \alpha \in [0, 1], \forall \mathbf{x}, \mathbf{y} \in \mathcal{S}$.
- Convex opt problem:
 - objective function is convex
 - equality constraints are linear
 - inequality constraints are concave

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 \mathbf{z} with $f(\mathbf{z}) < f(\mathbf{x}^*)$, study the segment $\overline{\mathbf{x}^* \mathbf{z}}$.)

Algorithm overview

- Algorithms look for a stationary point starting from a point \mathbf{x}_0 (arbitrary or user-supplied) \Rightarrow sequence of iterates $\{\mathbf{x}_k\}_{k=0}^\infty$ that terminates when no more progress can be made, or it seems that a solution has been approximated with sufficient accuracy.
- Stopping criterion: can't use $\|\mathbf{x}_k - \mathbf{x}^*\|$ or $|f(\mathbf{x}_k) - f(\mathbf{x}^*)|$. Instead, in practice (given a small $\epsilon > 0$):

- $\|\nabla f(\mathbf{x}_k)\| < \epsilon$
- $\|\mathbf{x}_k - \mathbf{x}_{k-1}\| < \epsilon$ or $\|\mathbf{x}_k - \mathbf{x}_{k-1}\| < \epsilon \|\mathbf{x}_{k-1}\|$
- $|f(\mathbf{x}_k) - f(\mathbf{x}_{k-1})| < \epsilon$ or $|f(\mathbf{x}_k) - f(\mathbf{x}_{k-1})| < \epsilon f(\mathbf{x}_{k-1})$

and also set a limit on k .

- We choose \mathbf{x}_{k+1} given information about f at \mathbf{x}_k (and possibly earlier iterates) so that $f(\mathbf{x}_{k+1}) < f(\mathbf{x}_k)$ (*descent*).

- Move $\mathbf{x}_k \rightarrow \mathbf{x}_{k+1}$: two fundamental strategies, line search and trust region.

- Line search strategy*

- Choose a *direction* \mathbf{p}_k
- Search along \mathbf{p}_k from \mathbf{x}_k for \mathbf{x}_{k+1} with $f(\mathbf{x}_{k+1}) < f(\mathbf{x}_k)$, i.e., *approximately* solve the 1D minimization problem: $\min_{\alpha > 0} f(\mathbf{x}_k + \alpha \mathbf{p}_k)$ where $\alpha = \text{step length}$.

- Trust region strategy*

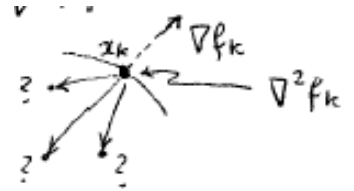
- Construct a model function m_k (typ. quadratic) that is similar to (but simpler than) f around \mathbf{x}_k .
- Search for \mathbf{x}_{k+1} with $m_k(\mathbf{x}_{k+1}) < m_k(\mathbf{x}_k)$ inside a small *trust region* (typ. a ball) around \mathbf{x}_k , i.e., *approximately* solve the n -D minimization problem: $\min_{\mathbf{p}} m_k(\mathbf{x}_k + \mathbf{p})$ s.t. $\mathbf{x}_k + \mathbf{p} \in \text{trust region}$.
- If \mathbf{x}_{k+1} does not produce enough decay in f , shrink the region.

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

- Good: derives maximum benefit from \mathbf{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 \mathbf{x} in a certain direction produce much larger variations in the value of f than do changes to \mathbf{x} in another direction. Some algorithms (e.g. steepest descent) are sensitive to poor scaling while others (e.g. 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).

- Related, but not the same as ill-conditioning.

Ex. $f(\mathbf{x}) = \frac{1}{2} \mathbf{x}^T \begin{pmatrix} 1 & 1+\epsilon \\ 1+\epsilon & 1 \end{pmatrix} \mathbf{x} - \begin{pmatrix} 2+\epsilon \\ 2 \end{pmatrix} \mathbf{x}$.

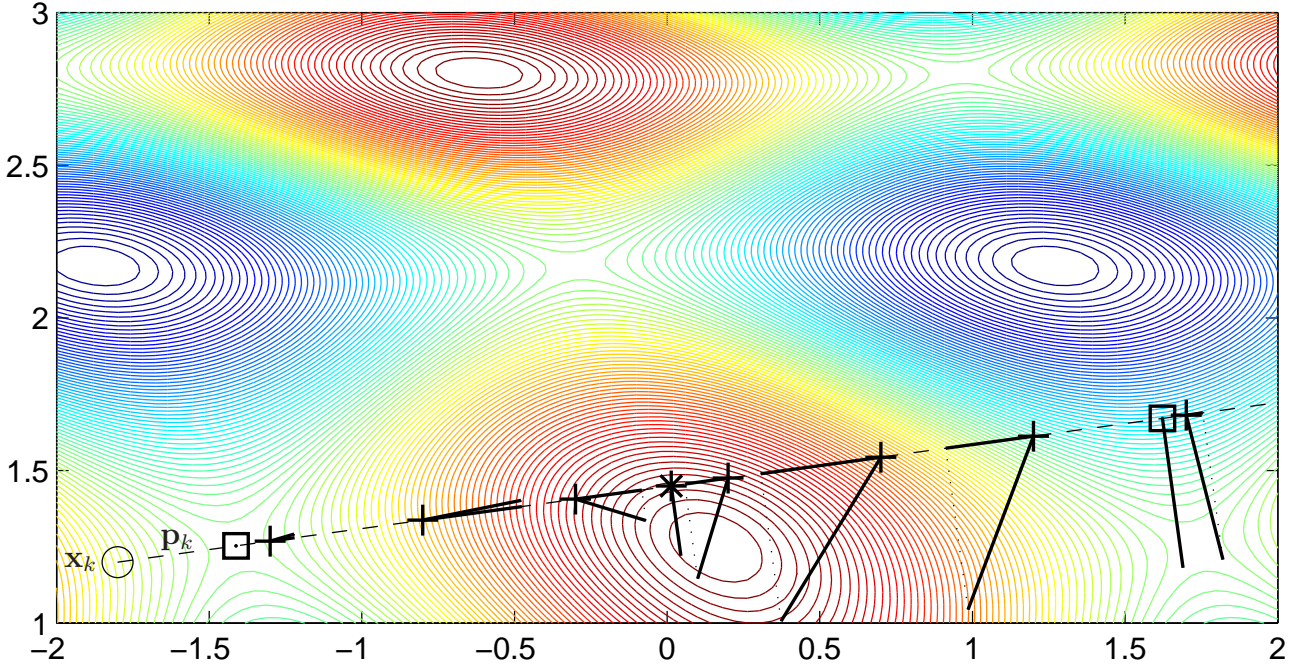
Review: fundamentals of unconstrained optimization

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

- \mathbf{x}^* is a local minimizer $\Rightarrow \begin{cases} \nabla f(\mathbf{x}^*) = 0 & \text{(first order)} \\ \nabla^2 f(\mathbf{x}^*) \text{ psd} & \text{(second order)} \end{cases}$ (necessary condition).
- $\nabla f(\mathbf{x}^*) = 0$, $\nabla^2 f(\mathbf{x}^*)$ pd $\Rightarrow \mathbf{x}^*$ is a strict local minimizer (sufficient condition).
- Stationary point: $\nabla f(\mathbf{x}^*) = 0$; $\nabla^2 f(\mathbf{x}^*)$ $\begin{cases} \text{pd: strict local minimizer} \\ \text{nd: strict local maximizer} \\ \text{not definite (pos \& neg eigenvalues): saddle point} \\ \text{psd: may be non-strict local minimizer} \\ \text{nsd: may be non-strict local maximizer} \end{cases}$

3 Line search methods

Iteration: $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{p}_k$, where α_k is the *step length* (how far to move along \mathbf{p}_k), $\alpha_k > 0$; \mathbf{p}_k is the *search direction*.



Descent direction: $\mathbf{p}_k^T \nabla f_k = \|\mathbf{p}_k\| \|\nabla f_k\| \cos \theta_k < 0$ (angle $< \frac{\pi}{2}$ with $-\nabla f_k$). Guarantees that f can be reduced along \mathbf{p}_k (for a sufficiently smooth step):

$$\begin{aligned} f(\mathbf{x}_k + \alpha \mathbf{p}_k) &= f(\mathbf{x}_k) + \alpha \mathbf{p}_k^T \nabla f_k + \mathcal{O}(\alpha^2) \quad (\text{Taylor's th.}) \\ &< f(\mathbf{x}_k) \quad \text{for all sufficiently small } \alpha > 0 \end{aligned}$$

- The *steepest descent direction*, i.e., the direction along which f decreases most rapidly, is $\mathbf{p}_k = -\nabla f_k$. Pf.: for any \mathbf{p} , α : $f(\mathbf{x}_k + \alpha \mathbf{p}) = f(\mathbf{x}_k) + \alpha \mathbf{p}^T \nabla f_k + \mathcal{O}(\alpha^2)$ so the rate of change in f along \mathbf{p} at \mathbf{x}_k is $\mathbf{p}^T \nabla f_k$ (the directional derivative) $= \|\mathbf{p}\| \|\nabla f_k\| \cos \theta$. Then $\min_{\mathbf{p}} \mathbf{p}^T \nabla f_k$ s.t. $\|\mathbf{p}\| = 1$ is achieved when $\cos \theta = -1$, i.e., $\mathbf{p} = -\nabla f_k / \|\nabla f_k\|$. fig. 2.5
fig. 2.6

This direction is \perp to the contours of f . Pf.: take $\mathbf{x} + \mathbf{p}$ on the same contour line as \mathbf{x} . Then, by Taylor's th.:

$$f(\mathbf{x} + \mathbf{p}) = f(\mathbf{x}) + \mathbf{p}^T \nabla f(\mathbf{x}) + \frac{1}{2} \mathbf{p}^T \nabla^2 f(\mathbf{x} + \epsilon \mathbf{p}) \mathbf{p}, \quad \epsilon \in (0, 1) \Rightarrow \cos \angle(\mathbf{p}, \nabla f(\mathbf{x})) = -\frac{1}{2} \frac{\mathbf{p}^T \nabla^2 f(\mathbf{x} + \epsilon \mathbf{p}) \mathbf{p}}{\|\mathbf{p}\| \|\nabla f(\mathbf{x})\|} \xrightarrow{\|\mathbf{p}\| \rightarrow 0} 0$$

but $\|\mathbf{p}\| \rightarrow 0$ along the contour line means $\mathbf{p} / \|\mathbf{p}\|$ is parallel to its tangent at \mathbf{x} .

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

$$f(\mathbf{x}_k + \mathbf{p}) \approx f_k + \mathbf{p}^T \nabla f_k + \frac{1}{2} \mathbf{p}^T \nabla^2 f_k \mathbf{p} = m_k(\mathbf{p})$$

which is minimized (take derivatives wrt \mathbf{p}) by the Newton direction if $\nabla^2 f_k$ is pd. ([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, $\mathbf{p}_k = -\mathbf{B}_k^{-1} \nabla f_k$ where \mathbf{B}_k is symmetric nonsingular:

- steepest descent: $\mathbf{B}_k = \mathbf{I}$
- Newton's method: $\mathbf{B}_k = \nabla^2 f(\mathbf{x}_k)$
- Quasi-Newton method: $\mathbf{B}_k \approx \nabla^2 f(\mathbf{x}_k)$

If \mathbf{B}_k is pd then \mathbf{p}_k is a descent direction: $\mathbf{p}_k^T \nabla f_k = -\nabla f_k^T \mathbf{B}_k^{-1} \nabla f_k < 0$.

Here, we deal with how to choose the step length given the search direction \mathbf{p}_k . Desirable properties: guaranteed global convergence and rapid rate of convergence.

Step length

Time/accuracy trade-off: 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): $\alpha_k: \min_{\alpha>0} \phi(\alpha) = f(\mathbf{x}_k + \alpha \mathbf{p}_k)$. Too expensive: many evaluations of f , ∇f to find α_k even with moderate precision. (✂ Angle $(\mathbf{p}_k, \widehat{\nabla f_{k+1}}) = ?$)
- Inexact line search: a typical 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 optimization algorithm.

- Reduction in f : $f(\mathbf{x}_k + \alpha_k \mathbf{p}_k) < f(\mathbf{x}_k) \rightarrow$ not enough, can converge before reaching the minimizer.

Wolfe conditions

- ① $f(\mathbf{x}_k + \alpha_k \mathbf{p}_k) \leq f(\mathbf{x}_k) + c_1 \alpha_k \nabla f_k^T \mathbf{p}_k$;
- ② $\nabla f(\mathbf{x}_k + \alpha_k \mathbf{p}_k)^T \mathbf{p}_k \geq c_2 \nabla f_k^T \mathbf{p}_k$.

where $0 < c_1 < c_2 < 1$. Call $\phi(\alpha) = f(\mathbf{x}_k + \alpha \mathbf{p}_k)$, then $\phi'(\alpha) = \nabla f(\mathbf{x}_k + \alpha \mathbf{p}_k)^T \mathbf{p}_k$.

- ① *Sufficient decrease (Armijo condition)* is equivalent to $\phi(0) - \phi(\mathbf{x}_k) \geq \alpha_k (-c_1 \phi'(0))$. fig. 3.3

Rejects too small decreases. The reduction is proportional both to step length α_k and directional derivative $\nabla f_k^T \mathbf{p}_k$. In practice, c_1 is very small, e.g. $c_1 = 10^{-4}$.

It is satisfied for any sufficiently small $\alpha \Rightarrow$ not enough, need to rule out unacceptably small steps.

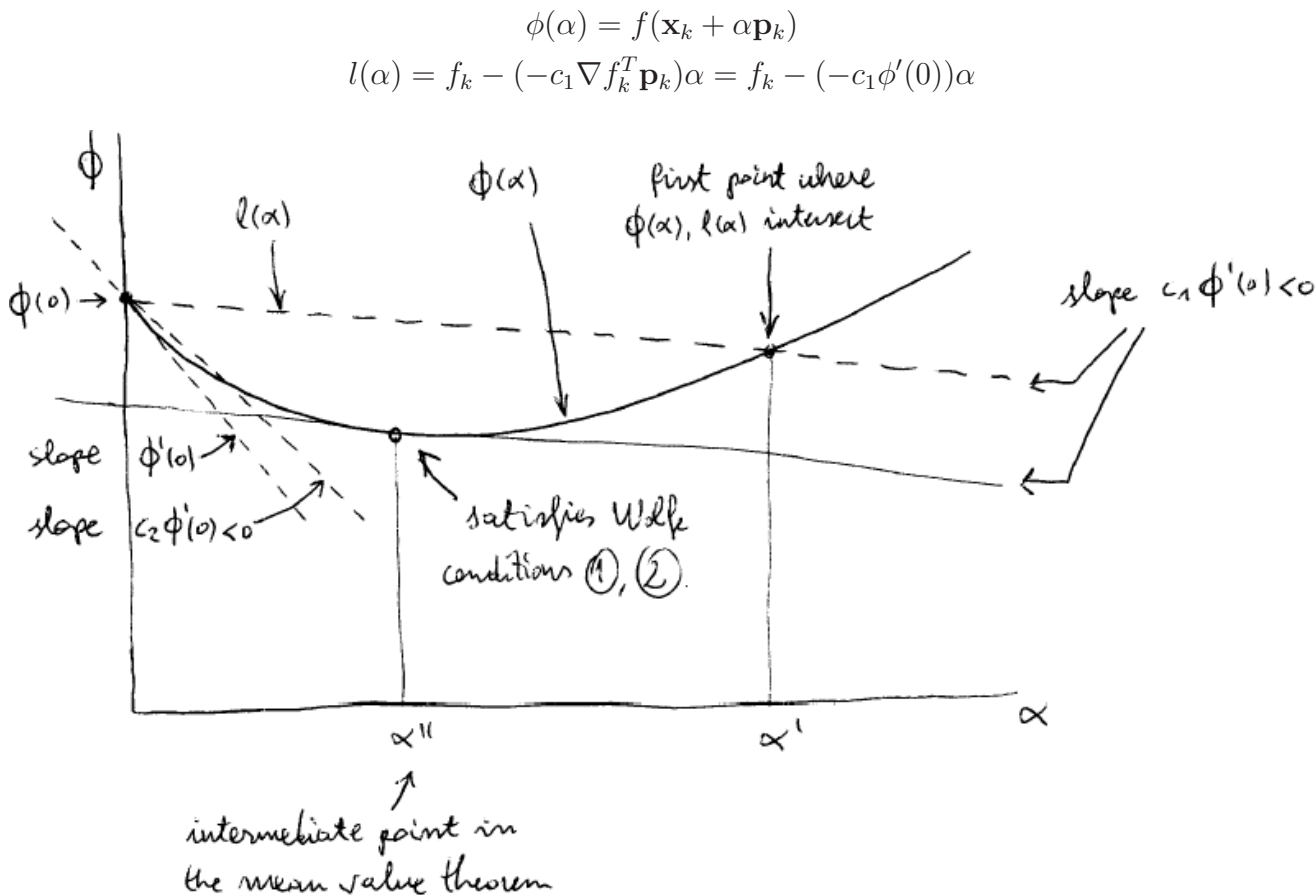
- ② *Curvature condition* is equivalent to $-\phi'(\mathbf{x}_k) \leq -c_2 \phi'(0)$. fig. 3.4

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

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

We will concentrate on the Wolfe conditions in general, and assume they *always* hold when the l.s. is used as part of an optimization 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.; see figure below.)



Other useful conditions

- *Strong Wolfe conditions:* ① + ②' $|\nabla f(\mathbf{x}_k + \alpha_k \mathbf{p}_k)^T \mathbf{p}_k| \leq c_2 |\nabla f_k^T \mathbf{p}_k|$. 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

$$f(\mathbf{x}_k) + (1 - c) \alpha_k \nabla f_k^T \mathbf{p}_k \stackrel{(a)}{\leq} f(\mathbf{x}_k + \alpha_k \mathbf{p}_k) \stackrel{①}{\leq} f(\mathbf{x}_k) + c \alpha_k \nabla f_k^T \mathbf{p}_k, \quad 0 < c < \frac{1}{2}$$

(a) controls step from below, ① is the first Wolfe condition.

Disadvantage: may exclude all minimizers of ϕ . (do the Wolfe conditions exclude minimizers?)

Sufficient decrease and backtracking

Start with largish step size and decrease it (times $\rho < 1$) until it meets the sufficient decrease condition ① (Algorithm 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 too 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, e.g. approximate $\phi(\alpha)$ as a cubic poly.
- There are also derivative-free methods (e.g. 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*: $\|\nabla f_k\| \xrightarrow[k \rightarrow \infty]{} 0$, i.e., convergence to a stationary point for *any* starting point \mathbf{x}_0 . To converge to a minimizer we need more information, e.g. the Hessian.
- We give a theorem for the search direction \mathbf{p}_k to obtain global convergence, focusing on the angle θ_k between \mathbf{p}_k and the steepest descent direction $-\nabla f_k$: $\cos \theta_k = \frac{-\nabla f_k^T \mathbf{p}_k}{\|\nabla f_k\| \|\mathbf{p}_k\|}$, and assuming the Wolfe cond. Similar theorems exist for strong Wolfe, Goldstein cond.
- Important theorem, e.g. shows that the steepest descent method is globally convergent; for other algorithms, it describes how far \mathbf{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 $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{p}_k$ with starting point \mathbf{x}_0 where \mathbf{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 $\mathcal{L} = \{\mathbf{x} : f(\mathbf{x}) \leq f(\mathbf{x}_0)\}$, and that ∇f is Lipschitz continuous on \mathcal{N} ($\Leftrightarrow \exists L > 0 : \|\nabla f(\mathbf{x}) - \nabla f(\tilde{\mathbf{x}})\| \leq L \|\mathbf{x} - \tilde{\mathbf{x}}\| \forall \mathbf{x}, \tilde{\mathbf{x}} \in \mathcal{N}$; weaker than bounded gradient). Then

$$\sum_{k \geq 0} \cos^2 \theta_k \|\nabla f_k\|^2 < \infty \quad (\text{Zoutendijk's condition}) \quad \begin{array}{l} \text{(Proof: ② + L lower bound } \alpha_k; \\ \text{① upper bound } f_{k+1}; \text{ telescope)} \end{array}$$

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

- Examples:
 - *Steepest descent method*: $\mathbf{p}_k = -\nabla f_k \Rightarrow \cos \theta_k = 1 \Rightarrow$ global convergence. Intuitive fig. 3.7 method, but very slow in difficult problems.
 - *Newton-like method*: $\mathbf{p}_k = -\mathbf{B}_k^{-1} \nabla f_k$ with \mathbf{B}_k symmetric, pd and with bounded condition number: $\|\mathbf{B}_k\| \|\mathbf{B}_k^{-1}\| \leq M \forall k$ (ill-cond. $\Rightarrow \nabla f \perp$ Newton dir.) Then $\cos \theta_k \geq \frac{1}{M}$ (Pf.: exe. 3.5) \Rightarrow global convergence.
In other words, if \mathbf{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 (e.g. 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 assuming 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, e.g. 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 looking for a maximizer!). The challenge is to design algorithms will both fast and global convergence.

Rate of convergence

- *Steepest descent*: $\mathbf{p}_k = -\nabla f_k$.

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 \mathbf{x}^* where the Hessian $\nabla^2 f(\mathbf{x}^*)$ is pd. Then $f(\mathbf{x}_{k+1}) - f(\mathbf{x}^*) \leq r^2(f(\mathbf{x}_k) - f(\mathbf{x}^*))$, where $r = \frac{\lambda_n - \lambda_1}{\lambda_n + \lambda_1} = \frac{\kappa - 1}{\kappa + 1}$, $0 < \lambda_1 \leq \dots \leq \lambda_n$ are the eigenvalues of $\nabla^2 f(\mathbf{x}^*)$ and $\kappa = \lambda_n / \lambda_1$ its condition number.

(Pf.: near the min., f is approx. quadratic.) (For quadratic functions (with matrix \mathbf{Q}): $\|\mathbf{x}_{k+1} - \mathbf{x}^*\|_{\mathbf{Q}} \leq r \|\mathbf{x}_k - \mathbf{x}^*\|_{\mathbf{Q}}$ where $\|\mathbf{x}\|_{\mathbf{Q}}^2 = \mathbf{x}^T \mathbf{Q} \mathbf{x}$ and $\frac{1}{2} \|\mathbf{x} - \mathbf{x}^*\|_{\mathbf{Q}}^2 = f(\mathbf{x}) - f(\mathbf{x}^*)$.)

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

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

- *Newton's method*: $\mathbf{p}_k = -\nabla^2 f_k^{-1} \nabla f_k$.

Th. 3.5: assume f twice diff. and $\nabla^2 f$ Lipschitz cont. in a neighborhood of a solution \mathbf{x}^* at the which second-order sufficient cond. hold. Consider the iterates $\mathbf{x}_{k+1} = \mathbf{x}_k - \nabla^2 f_k^{-1} \nabla f_k$. Then, if \mathbf{x}_0 is sufficiently close to \mathbf{x}^* : $(\mathbf{x}_k) \rightarrow \mathbf{x}^*$, $(\|\nabla f_k\|) \rightarrow 0$ both quadratically.

(Pf.: upper bound $\|\mathbf{x}_k + \mathbf{p}_k - \mathbf{x}^*\|$ by a constant $\times \|\mathbf{x}_k - \mathbf{x}^*\|^2$ using Taylor + Lipschitz.)

That is, near the solution, where the Hessian is pd, the convergence rate is *quadratic* if we always take $\alpha_k = 1$ (no line search at all). The theorem does not apply if the Hessian is not pd (away from the solution); practical Newton methods avoid this.

- *Quasi-Newton methods*: $\mathbf{p}_k = -\mathbf{B}_k^{-1} \nabla f_k$ with \mathbf{B}_k symmetric pd.

Th. 3.7: assume f is twice cont. diff. and that the iterates $\mathbf{x}_{k+1} = \mathbf{x}_k - \mathbf{B}_k^{-1} \nabla f_k$ converge to a point \mathbf{x}^* at the which second-order sufficient cond. hold. Then $(\mathbf{x}_k) \rightarrow \mathbf{x}^*$ superlinearly iff $\lim_{k \rightarrow \infty} \frac{\|(\mathbf{B}_k - \nabla^2 f(\mathbf{x}^*)) \mathbf{p}_k\|}{\|\mathbf{p}_k\|} = 0$.

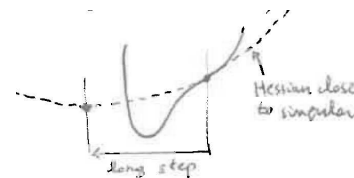
Thus, the convergence rate is *superlinear* if the matrices \mathbf{B}_k become increasingly accurate approximations of the Hessian along the search directions \mathbf{p}_k , and near the solution the step length α_k is always 1, i.e., $\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{p}_k$. In practice, we must always try $\alpha = 1$ in the line search and accept it if it satisfies the Wolfe cond.

Newton's method with Hessian modification

Newton step: solution of the $n \times n$ linear system $\nabla^2 f(\mathbf{x}_k) \mathbf{p}_k^N = -\nabla f(\mathbf{x}_k)$.

- Near a minimizer the Hessian is pd \Rightarrow quadratic convergence (with unit steps $\alpha_k = 1$).

- Away from a minimizer the Hessian may not be pd or may be close to singular $\Rightarrow \mathbf{p}_k^N$ 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.

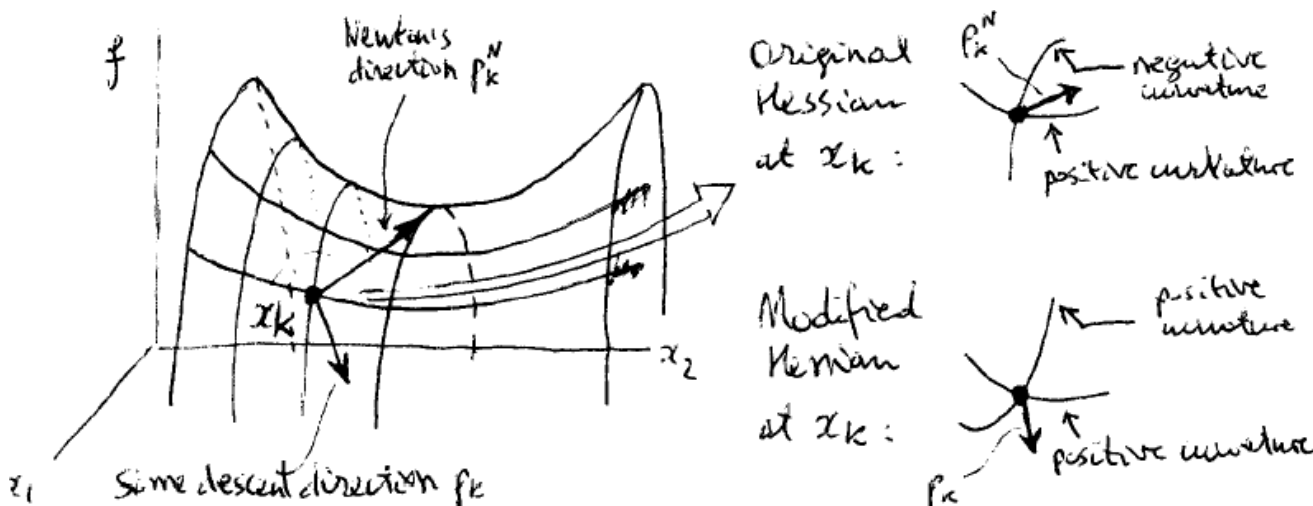


Modified Newton method: modify the Hessian to make it sufficiently pd.

- We solve (by factorizing \mathbf{B}_k , e.g. Cholesky) the system $\mathbf{B}_k \mathbf{p}_k = -\nabla f(\mathbf{x}_k)$ where $\mathbf{B}_k = \nabla^2 f(\mathbf{x}_k) + \mathbf{E}_k$ (modified Hessian) such that \mathbf{B}_k is sufficiently pd. Then we use \mathbf{p}_k (which is a descent direction because \mathbf{B}_k is pd) in a line search with Wolfe conditions.
- Global convergence ($\|\nabla f_k\| \rightarrow 0$) by Zoutendijk's condition if $\kappa(\mathbf{B}_k) = \|\mathbf{B}_k\| \|\mathbf{B}_k^{-1}\| \leq M$ (which can be proven for several types of modifications).
- Quadratic convergence near the minimizer, where the Hessian is pd, if $\mathbf{E}_k = \mathbf{0}$ there.

Diagonal Hessian modification: $\mathbf{E}_k = \lambda \mathbf{I}$ with $\lambda \geq 0$ large enough that \mathbf{B}_k is sufficiently pd.
 $\lambda = \max(0, \delta - \lambda_{\min}(\nabla^2 f(\mathbf{x}_k)))$ for $\delta \geq 0$.

- We want λ as small as possible to preserve Hessian information along the positive curvature directions; but if λ is too small, \mathbf{B}_k is nearly singular and the step too long.
- The method behaves like pure Newton for pd Hessian and $\lambda = 0$, like steepest descent for $\lambda \rightarrow \infty$, and finds some descent direction for intermediate λ .



Other types of Hessian modification exist, but there is no consensus about which one is best:

- Direct modification of the eigenvalues (needs SVD of the Hessian). If $\mathbf{A} = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^T$ (spectral th.) then $\Delta\mathbf{A} = \mathbf{Q} \text{diag}(\max(0, \delta - \lambda_i)) \mathbf{Q}^T$ is the correction with minimum Frobenius norm s.t. $\lambda_{\min}(\mathbf{A} + \Delta\mathbf{A}) \geq \delta$.
- Modified Cholesky factorization (add terms to diagonal on the fly).

Review: line search (l.s.) methods

- Iteration $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{p}_k$ $\left\{ \begin{array}{l} \text{search direction } \mathbf{p}_k \text{ given by optimization method} \\ \text{l.s.} \equiv \text{determine step length } \alpha_k \end{array} \right.$
- We want:
 - Descent direction: $\mathbf{p}_k^T \nabla f_k = \|\mathbf{p}_k\| \|\nabla f_k\| \cos \theta_k < 0$ $\left\{ \begin{array}{ll} \text{steepest descent dir. :} & -\nabla f_k \\ \text{Newton dir. :} & -\nabla^2 f_k^{-1} \nabla f_k \\ \text{Quasi-Newton dir. :} & -\mathbf{B}_k^{-1} \nabla f_k \\ (\mathbf{B}_k \text{ pd} \Rightarrow \text{descent dir.}) \end{array} \right.$
 - Inexact l.s.: approx. solution of $\min_{\alpha > 0} f(\mathbf{x}_k + \alpha \mathbf{p}_k)$ (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.
Ex.: 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 (to a stationary point): $\|\nabla f_k\| \rightarrow 0$.
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 $\cos \theta_k \approx 0$!
Ex.: steepest descent and 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.
- Modified Hessian Newton's method: $\mathbf{B}_k = \nabla^2 f(\mathbf{x}_k) + \lambda \mathbf{I}$ (diagonal modif., others exist) s.t. \mathbf{B}_k suffic. pd.
 $\lambda = 0$: pure Newton step; $\lambda \rightarrow \infty$: steepest descent direction (with step $\rightarrow 0$).
Descent direction with moderate length away from minimizer, pure Newton's method near minimizer.
Global, quadratic convergence if $\kappa(\mathbf{B}_k) \leq M$ and l.s. with Wolfe cond.

4 Trust region methods

Iteration: $\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{p}_k$, where \mathbf{p} is the approximate minimizer of the *model* $m_k(\mathbf{p})$ in a region around \mathbf{x}_k (the *trust region*); if \mathbf{p}_k does not produce a *sufficient decrease* in f , we shrink the region and try again.

- The trust region is typically a ball $B(\mathbf{x}_k; \Delta)$. Other region shapes may also be used: elliptical (to adapt to ill-conditioned Hessians) and box-shaped (with linear constraints.)
- Each time we decrease Δ after failure of a candidate iterate, the step from \mathbf{x}_k is shorter and usually points in a different direction.
- Trade-off 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*: $m_k(\mathbf{p}) = f_k + \nabla f_k^T \mathbf{p}$ s.t. $\|\mathbf{p}\| \leq \Delta_k \Rightarrow \mathbf{p}_k = -\Delta_k \frac{\nabla f_k}{\|\nabla f_k\|}$, i.e., steepest descent with step length α_k given by Δ_k (no news). (🔪 what is the approx. error for the linear model?)
- *Quadratic model*: $m_k(\mathbf{p}) = f_k + \nabla f_k^T \mathbf{p} + \frac{1}{2} \mathbf{p}^T \mathbf{B}_k \mathbf{p}$ where \mathbf{B}_k is symmetric but need not be psd. The approximation error is $\begin{cases} \mathcal{O}(\|\mathbf{p}\|^3) & \text{if } \mathbf{B}_k = \nabla^2 f_k, \text{ trust-region Newton method} \\ \mathcal{O}(\|\mathbf{p}\|^2) & \text{otherwise.} \end{cases}$

In both cases, the model is accurate for small $\|\mathbf{p}\|$, which guarantees we can always find a good step for sufficiently small Δ . (🔪 what happens if $\rho_k < 0$ but $\|\mathbf{p}_k\| < \Delta_k$ with an *arbitrary* m_k ?)

Two issues remain: how to choose Δ_k ? How to find \mathbf{p}_k ?

Choice of the trust-region radius Δ_k Define the ratio $\rho_k = \frac{f_k - f(\mathbf{x}_k + \mathbf{p}_k)}{m_k(\mathbf{0}) - m_k(\mathbf{p}_k)} = \frac{\text{actual reduction}}{\text{predicted reduction}}$.

- Predicted reduction ≥ 0 always (since $\mathbf{p} = \mathbf{0}$ is in the region).
- If actual reduction < 0 the new objective value is larger, so reject the step.
- $\rho_k \begin{cases} \approx 1: \text{ good agreement between } f \text{ and the model } m_k, \text{ so expand } \Delta \text{ if } \|\mathbf{p}_k\| = \Delta_k \\ \text{(otherwise, don't interfere)} \\ > 0 \text{ but not close to 1: keep } \Delta_k \\ \text{close to 0 or negative: shrink } \Delta_k. \end{cases}$

Algorithm 4.1

The optimization subproblem

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

- If \mathbf{B}_k is pd and $\|\mathbf{B}_k^{-1} \nabla f_k\| \leq \Delta_k$ the solution is the unconstrained minimizer $\mathbf{p}_k^B = -\mathbf{B}_k^{-1} \nabla f_k$ (*full step*).
- Otherwise, we compute an *approximate* solution (finding an exact one is too costly).

Approximate solution of the optimization subproblem The *Cauchy point* \mathbf{p}_k^C is the minimizer along $\mathbf{p}_k = -\nabla m_k(\mathbf{0}) = -\mathbf{g}_k$, which lies either at $-\frac{\Delta_k}{\mathbf{g}_k^T \mathbf{g}_k} \mathbf{g}_k$ (boundary) or at $-\tau_k \frac{\Delta_k}{\mathbf{g}_k^T \mathbf{g}_k} \mathbf{g}_k$ with $0 < \tau_k < 1$ (interior). Thus, it is steepest descent with a certain step size, and gives a baseline solution. Several methods improve over the Cauchy point. One approach is based on the following characterization of the exact solution: eq. (4.12)
fig. 4.3

Th. 4.1. \mathbf{p}^* is a global solution of the trust-region problem $\min_{\|\mathbf{p}\| \leq \Delta} m(\mathbf{p}) = f + \mathbf{g}^T \mathbf{p} + \frac{1}{2} \mathbf{p}^T \mathbf{B} \mathbf{p}$ iff \mathbf{p}^* is feasible and $\exists \lambda \geq 0$ such that:

1. $(\mathbf{B} + \lambda \mathbf{I}) \mathbf{p}^* = -\mathbf{g}$
2. $\lambda (\Delta - \|\mathbf{p}^*\|) = 0$ (i.e., $\lambda = 0$ or $\|\mathbf{p}^*\| = \Delta$)
3. $\mathbf{B} + \lambda \mathbf{I}$ is psd.

Algorithm:

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

$\mathbf{B} = \mathbf{Q} \mathbf{\Lambda} \mathbf{Q}^T$ (spectral th. with $\lambda_n \geq \dots \geq \lambda_1$) $\Rightarrow \mathbf{p}(\lambda) = -(\mathbf{B} + \lambda \mathbf{I})^{-1} \mathbf{g} = -\sum_{j=1}^n \frac{\mathbf{q}_j^T \mathbf{g}}{\lambda_j + \lambda} \mathbf{q}_j \Rightarrow \|\mathbf{p}(\lambda)\|^2 = \sum_{j=1}^n \frac{(\mathbf{q}_j^T \mathbf{g})^2}{(\lambda_j + \lambda)^2}$. If $\mathbf{q}_1^T \mathbf{g} \neq 0$, find $\lambda^* > -\lambda_1$ using Newton's method for root finding on $r(\lambda) = \frac{1}{\Delta} - \frac{1}{\|\mathbf{p}(\lambda)\|}$ (since $\frac{1}{\|\mathbf{p}(\lambda)\|} \approx (\lambda + \lambda_1)/\text{constant}$). One can show this is equivalent to Alg. 4.3, which uses Cholesky factorizations (limit to ~ 3 steps). fig. 4.5

Note:

- Using $\mathbf{B} + \lambda \mathbf{I}$ instead of \mathbf{B} in the model transforms the problem into $\min_{\mathbf{p}} m(\mathbf{p}) + \frac{\lambda}{2} \|\mathbf{p}\|^2$, and so for large $\lambda > 0$ the minimizer is strictly inside the region. As we decrease λ the minimizer moves to the region boundary and the theorem holds for that λ .
- If $\lambda > 0$ then the direction is antiparallel to the model gradient and so the region is tangent to the model contour at the solution: $\lambda \mathbf{p}^* = -\mathbf{g} - \mathbf{B} \mathbf{p}^* = -\nabla m(\mathbf{p}^*)$.
- This is useful for Newton's method and is the basis of the Levenberg-Marquardt algorithm for nonlinear least-squares problems.

Other approaches:

- Dogleg method: find minimizer along two-leg path $\mathbf{0} \rightarrow \mathbf{p}_k^C \rightarrow \mathbf{p}_k^B$. fig. 4.4
- Two-dimensional subspace minimization: on the span of the dogleg path.

Global and local convergence

- Under certain assumptions, these approximate algorithms have *global convergence* to a stationary point (Th. 4.6). Essentially, they must ensure a sufficient decrease $m_k(\mathbf{0}) \leq m_k(\mathbf{p}_k)$ at each step; the Cauchy point already achieves such as decrease.
- If using $\mathbf{B}_k = \nabla^2 f_k$ and if the region becomes eventually inactive and we always take the full step, the convergence is *quadratic* (the method becomes Newton's method).

Review: trust-region methods

- Iteration $\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{p}_k$
 - \mathbf{p}_k = approx. minimizer of model m_k of f in trust region: $\min_{\|\mathbf{p}\| \leq \Delta_k} m_k(\mathbf{p})$.
 - \mathbf{p}_k does not produce sufficient decrease \Rightarrow region too big, shrink it and try again.

Insufficient decrease $\Leftrightarrow \rho_k = \frac{f_k - f(\mathbf{x}_k + \mathbf{p}_k)}{m_k(\mathbf{0}) - m_k(\mathbf{p}_k)} \lesssim 0$

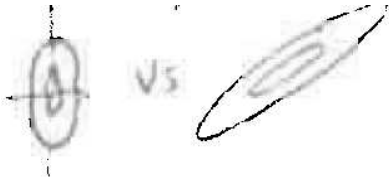
- Quadratic model: $m_k(\mathbf{p}) = f_k + \nabla f_k^T \mathbf{p} + \frac{1}{2} \mathbf{p}^T \mathbf{B}_k \mathbf{p}$. \mathbf{B}_k need not be psd.
Cauchy point: minimizer of m_k within the trust region along the steepest descent direction.
The exact, global minimizer of m_k within the trust region $\|\mathbf{p}\| \leq \Delta_k$ satisfies certain conditions (th. 4.1) that can be used to find an approximate solution. Simpler methods exist (dogleg, 2D subspace min.).
- Global convergence under mild conditions if sufficient decrease; quadratic rate if $\mathbf{B}_k = \nabla^2 f_k$ and if the region becomes eventually inactive and we always take the full step.
- Mainly used for Newton and Levenberg-Marquardt methods.

5 Conjugate gradient methods





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

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

Assume in all this chapter that \mathbf{A} is an $n \times n$ symmetric pd matrix, $\phi(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T \mathbf{A} \mathbf{x} - \mathbf{b}^T \mathbf{x}$ and $\nabla \phi(\mathbf{x}) = \mathbf{A} \mathbf{x} - \mathbf{b} = \mathbf{r}(\mathbf{x})$ (residual).



Idea:

- steepest descent (1 or ∞ iterations): 
- coordinate descent (n or ∞ iterations): 
- Newton's method (1 iteration): 
- CG (n iterations): 

The linear conjugate gradient method

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

Linear system $\mathbf{A} \mathbf{x} = \mathbf{b} \iff$ Optimization problem $\min \phi(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T \mathbf{A} \mathbf{x} - \mathbf{b}^T \mathbf{x}$.

- A set of nonzero vectors $\{\mathbf{p}_0, \mathbf{p}_1, \dots, \mathbf{p}_l\}$ is *conjugate* wrt \mathbf{A} iff $\mathbf{p}_i^T \mathbf{A} \mathbf{p}_j = 0 \ \forall i \neq j$.

Conjugacy \Rightarrow linear independence (Pf.: left-multiply $\sum \sigma_i \mathbf{p}_i$ times $\mathbf{p}_j^T \mathbf{A}$.)

Note $\{\mathbf{A}^{1/2} \mathbf{p}_0, \mathbf{A}^{1/2} \mathbf{p}_1, \dots, \mathbf{A}^{1/2} \mathbf{p}_l\}$ are orthogonal.

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


Conjugate direction method: given a starting point $\mathbf{x}_0 \in \mathbb{R}^n$ and a set of conjugate directions

$\{\mathbf{p}_0, \dots, \mathbf{p}_{n-1}\}$, generate the sequence $\{\mathbf{x}_k\}$ with $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{p}_k$, where $\alpha_k = -\frac{\mathbf{r}_k^T \mathbf{p}_k}{\mathbf{p}_k^T \mathbf{A} \mathbf{p}_k}$

( denominator $\neq 0$?) (exact line search) (Proof: $\mathbf{x}^* = \mathbf{x}_0 + \sum_{i=0}^{n-1} \alpha_i \mathbf{p}_i$.)

- Intuitive idea:

– \mathbf{A} diagonal: quadratic function ϕ can be minimized along the coordinate directions $\mathbf{e}_1, \dots, \mathbf{e}_n$ in n iterations. fig. 5.1

– \mathbf{A} not diagonal: the coordinate directions don't minimize ϕ in n iterations; but the fig. 5.2
variable change $\hat{\mathbf{x}} = \mathbf{S}^{-1} \mathbf{x}$ with $\mathbf{S} = (\mathbf{p}_0 \ \mathbf{p}_1 \ \dots \ \mathbf{p}_{n-1})$ diagonalizes \mathbf{A} : $\hat{\phi}(\hat{\mathbf{x}}) = \phi(\mathbf{S} \hat{\mathbf{x}}) = \frac{1}{2} \hat{\mathbf{x}}^T (\mathbf{S}^T \mathbf{A} \mathbf{S}) \hat{\mathbf{x}} - (\mathbf{S}^T \mathbf{b})^T \hat{\mathbf{x}}$. ( why is \mathbf{S} invertible?)
coordinate search in $\hat{\mathbf{x}} \Leftrightarrow$ conjugate direction search in \mathbf{x} .

- Th. 5.2 (*expanding subspace minimization*): for the conjugate directions method:

– $\mathbf{r}_k^T \mathbf{p}_i = 0$ for $i = 0, \dots, k-1$ (the current residual is \perp to all previous search directions)

(Intuition: if $\mathbf{r}_k = \nabla \phi(\mathbf{x}_k)$ had a nonzero projection along \mathbf{p}_i , it would not be a minimum.)

– \mathbf{x}_k is the minimizer of ϕ over the set $\mathbf{x}_0 + \text{span}\{\mathbf{p}_0, \dots, \mathbf{p}_{k-1}\}$.

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

(Proof: induction plus the fact $\mathbf{r}_{k+1} = \mathbf{r}_k + \alpha_k \mathbf{A} \mathbf{p}_k$ (implied by $\mathbf{r}_k = \mathbf{A} \mathbf{x}_k - \mathbf{b}$ and $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{p}_k$.)

- How to obtain conjugate directions? Many ways, e.g. using the eigenvectors of \mathbf{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 *conjugate gradient method* generates conjugate direction \mathbf{p}_k by using only the previous one, \mathbf{p}_{k-1} :
 - \mathbf{p}_k is a l.c. of $-\nabla\phi(\mathbf{x}_k)$ and \mathbf{p}_{k-1} s.t. being conjugate to $\mathbf{p}_{k-1} \Rightarrow \mathbf{p}_k = -\mathbf{r}_k + \beta_k \mathbf{p}_{k-1}$ with $\beta_k = \frac{\mathbf{r}_k^T \mathbf{A} \mathbf{p}_{k-1}}{\mathbf{p}_{k-1}^T \mathbf{A} \mathbf{p}_{k-1}}$.
 - We start with the steepest descent direction: $\mathbf{p}_0 = -\nabla\phi(\mathbf{x}_0) = -\mathbf{r}_0$.

Algorithm 5.1 (CG; preliminary version): given \mathbf{x}_0 :

$\mathbf{r}_0 \leftarrow \nabla\phi(\mathbf{x}_0) = \mathbf{A}\mathbf{x}_0 - \mathbf{b}, \mathbf{p}_0 \leftarrow -\mathbf{r}_0, k \rightarrow 0$ while $\mathbf{r}_k \neq 0$ $\alpha_k \leftarrow -\frac{\mathbf{r}_k^T \mathbf{p}_k}{\mathbf{p}_k^T \mathbf{A} \mathbf{p}_k}, \mathbf{x}_{k+1} \leftarrow \mathbf{x}_k + \alpha_k \mathbf{p}_k$ $\mathbf{r}_{k+1} = \mathbf{A}\mathbf{x}_{k+1} - \mathbf{b}$ $\beta_{k+1} \leftarrow \frac{\mathbf{r}_{k+1}^T \mathbf{A} \mathbf{p}_k}{\mathbf{p}_k^T \mathbf{A} \mathbf{p}_k}, \mathbf{p}_{k+1} = -\mathbf{r}_{k+1} + \beta_{k+1} \mathbf{p}_k$ $k \leftarrow k + 1$ end	Start with steepest descent dir. from \mathbf{x}_0 $\mathbf{r}_k = 0$ means we are done, which may happen before n steps Exact line search New residual New l.s. direction \mathbf{p}_{k+1} is conjugate to $\mathbf{p}_k, \mathbf{p}_{k-1}, \dots, \mathbf{p}_0$
--	---

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

- Th. 5.3: suppose that the k th iterate of the CG method is not the solution \mathbf{x}^* . Then:
 - $\mathbf{r}_k^T \mathbf{r}_i = 0$ for $i = 0, \dots, k-1$ (the gradients at all iterates are \perp to each other.)
 - $\text{span}\{\mathbf{r}_0, \dots, \mathbf{r}_k\} = \text{span}\{\mathbf{p}_0, \dots, \mathbf{p}_k\} = \text{span}\{\mathbf{r}_0, \mathbf{A}\mathbf{r}_0, \dots, \mathbf{A}^k \mathbf{r}_0\} = \text{Krylov subspace of degree } k \text{ for } \mathbf{r}_0. \text{ (So } \{\mathbf{r}_k\} \text{ orthogonal basis, } \{\mathbf{p}_k\} \text{ basis.)}$
 (Intuitive explanation: compute $\mathbf{r}_k, \mathbf{p}_k$ for $k = 1, 2$ using $\mathbf{r}_{k+1} = \mathbf{r}_k + \mathbf{A} \mathbf{p}_k, \mathbf{p}_{k+1} = -\mathbf{r}_{k+1} + \beta_{k+1} \mathbf{p}_k$.)
 - $\mathbf{p}_k^T \mathbf{A} \mathbf{p}_i = 0$ for $i = 0, \dots, k-1$ (conjugate wrt \mathbf{A} .)

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

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

(Proof: $\mathbf{r}_k^T \mathbf{r}_i = \mathbf{p}_k^T \mathbf{A} \mathbf{p}_i = 0$ follow for $i = k-1$ by construction and for $i < k-1$ by induction.)

- We can simplify a bit the algorithm using the following results:

$$\begin{aligned}
 & - \mathbf{p}_{k+1} = -\mathbf{r}_{k+1} + \beta_{k+1} \mathbf{p}_k \text{ (construction of the } k\text{th direction)} \\
 & - \mathbf{r}_{k+1} = \mathbf{r}_k + \alpha_k \mathbf{A} \mathbf{p}_k \text{ (definition of } \mathbf{r}_k \text{ and } \mathbf{x}_{k+1}) \\
 & - \mathbf{r}_k^T \mathbf{p}_i = \mathbf{r}_k^T \mathbf{r}_i = 0 \text{ for } i < k \text{ (th. 5.2 \& 5.3)} \\
 \Rightarrow & \alpha_k \leftarrow \frac{\mathbf{r}_k^T \mathbf{r}_k}{\mathbf{p}_k^T \mathbf{A} \mathbf{p}_k} = \frac{\|\mathbf{r}_k\|^2}{\|\mathbf{p}_k\|_{\mathbf{A}}^2}, \mathbf{r}_{k+1} \leftarrow \mathbf{r}_k + \alpha_k \mathbf{A} \mathbf{p}_k, \beta_{k+1} \leftarrow \frac{\mathbf{r}_{k+1}^T \mathbf{r}_{k+1}}{\mathbf{r}_k^T \mathbf{r}_k} = \frac{\|\mathbf{r}_{k+1}\|^2}{\|\mathbf{r}_k\|^2} \text{ in algorithm 5.2.}
 \end{aligned}$$

- *Space complexity*: $\mathcal{O}(n)$ since it computes $\mathbf{x}, \mathbf{r}, \mathbf{p}$ at $k+1$ given the values at k : no matrix storage.
- *Time complexity*: the bottleneck is the matrix-vector product $\mathbf{A} \mathbf{p}_k$, which is $\mathcal{O}(n^2)$ (maybe less if \mathbf{A} has structure) \Rightarrow in n steps, $\mathcal{O}(n^3)$, similar to other methods for solving linear systems (e.g. Gauss factorization).

- Advantages: no matrix storage; does not alter \mathbf{A} ; does not introduce fill (for a sparse matrix \mathbf{A}); fast convergence.
- Disadvantages: sensitive to roundoff errors.

It is recommended for *large systems*.

Rate of convergence

- Here we don't mean the asymptotic rate ($k \rightarrow \infty$) 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 \mathbf{A} :
 - Th. 5.4: if \mathbf{A} has only r distinct eigenvalues, CG converges in at most r steps.
 - If the eigenvalues of \mathbf{A} occur in r distinct clusters, CG will approximately solve the problem in r steps..
- Two bounds (using $\|\mathbf{x}\|_{\mathbf{A}}^2 = \mathbf{x}^T \mathbf{A} \mathbf{x}$), useful to estimate the convergence rate in advance if we know something about the eigenvalues of \mathbf{A} :
 - Th. 5.5: $\|\mathbf{x}_{k+1} - \mathbf{x}^*\|_{\mathbf{A}}^2 \leq \left(\frac{\lambda_{n-k} - \lambda_1}{\lambda_{n-k} + \lambda_1}\right)^2 \|\mathbf{x}_0 - \mathbf{x}^*\|_{\mathbf{A}}^2$ if \mathbf{A} has eigenvalues $\lambda_1 \leq \dots \leq \lambda_n$.
 - $\|\mathbf{x}_k - \mathbf{x}^*\|_{\mathbf{A}} \leq 2 \left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^k \|\mathbf{x}_0 - \mathbf{x}^*\|_{\mathbf{A}}$ if $\kappa = \frac{\lambda_n}{\lambda_1}$ is the c.n. (this bound is very coarse).
Recall for steepest descent we had a similar expression but with $\frac{\kappa-1}{\kappa+1}$ instead of $\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}$.
- *Preconditioning*: change of variables $\hat{\mathbf{x}} = \mathbf{C}\mathbf{x}$ so that the new matrix $\hat{\mathbf{A}} = \mathbf{C}^{-T} \mathbf{A} \mathbf{C}^{-1}$ has a clustered spectrum or a small condition number (thus faster convergence). Besides being effective in this sense, a good preconditioner \mathbf{C} should take little storage and allow an inexpensive solution of $\mathbf{C}\mathbf{x} = \hat{\mathbf{x}}$. Finding good preconditioners \mathbf{C} depends on the problem (the structure of \mathbf{A}), e.g. good ones exist when \mathbf{A} results from a discretized PDE.

fig. 5.4

Nonlinear conjugate gradient methods

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

The Fletcher-Reeves method $\begin{cases} \alpha_k \text{ is determined by an inexact line search} \\ \mathbf{r}_k = \nabla f \end{cases}$

Algorithm 5.4: given \mathbf{x}_0 :

Evaluate $f_0 \leftarrow f(\mathbf{x}_0)$, $\nabla f_0 \leftarrow \nabla f(\mathbf{x}_0)$

$\mathbf{p}_0 \leftarrow -\nabla f_0$, $k \leftarrow 0$

while $\nabla f_k \neq 0$

$\mathbf{x}_{k+1} \leftarrow \mathbf{x}_k + \alpha_k \mathbf{p}_k$ with inexact l.s. for α_k

Evaluate ∇f_{k+1}

$\beta_{k+1}^{\text{FR}} \leftarrow \frac{\nabla f_{k+1}^T \nabla f_{k+1}}{\nabla f_k^T \nabla f_k}$, $\mathbf{p}_{k+1} \leftarrow -\nabla f_{k+1} + \beta_{k+1}^{\text{FR}} \mathbf{p}_k$

$k \leftarrow k + 1$

end

- Uses no matrix operations, requires only f and ∇f .
- Line search for α_k : we need each direction $\mathbf{p}_{k+1} = -\nabla f_{k+1} + \beta_{k+1}^{\text{FR}} \mathbf{p}_k$ to be a descent direction, i.e., $\nabla f_{k+1}^T \mathbf{p}_{k+1} = -\|\nabla f_{k+1}\|^2 + \beta_{k+1}^{\text{FR}} \nabla f_{k+1}^T \mathbf{p}_k < 0$.
 - Exact l.s.: α_k is a local minimizer along $\mathbf{p}_k \Rightarrow \nabla f_{k+1}^T \mathbf{p}_k = 0 \Rightarrow \mathbf{p}_{k+1}$ is descent.
 - Inexact l.s.: \mathbf{p}_{k+1} is descent if α_k satisfies the strong Wolfe conditions (lemma 5.6):

$$\begin{aligned} f(\mathbf{x}_k + \alpha_k \mathbf{p}_k) &\leq f(\mathbf{x}_k) + c_1 \alpha_k \nabla f_k^T \mathbf{p}_k \\ |\nabla f(\mathbf{x}_k + \alpha_k \mathbf{p}_k)^T \mathbf{p}_k| &\leq c_2 |\nabla f_k^T \mathbf{p}_k| \end{aligned}$$

where $0 < c_1 < c_2 < \frac{1}{2}$ (note we required a looser $0 < c_1 < c_2 < 1$ in ch. 3).

The Polak-Ribière method

- Differs in the parameter β_k , defined as $\beta_{k+1}^{\text{PR}} = \frac{\nabla f_{k+1}^T (\nabla f_{k+1} - \nabla f_k)}{\|\nabla f_k\|^2}$.
- For strongly convex quadratic functions and exact l.s. $\beta_k^{\text{PR}} = \beta_k^{\text{FR}} = \beta_k^{\text{HS}} = \beta_k$ 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 \mathbf{p}_k is a descent direction.
- Other variants similar to PR: Hestenes-Stiefel $\beta_{k+1}^{\text{HS}} = \frac{\nabla f_{k+1}^T (\nabla f_{k+1} - \nabla f_k)}{(\nabla f_{k+1} - \nabla f_k)^T \mathbf{p}_k}$ (derived by requiring $\mathbf{p}_{k+1}^T \bar{\mathbf{G}}_k \mathbf{p}_k = 0$ for the average Hessian $\bar{\mathbf{G}}_k = \int_0^1 \nabla^2 f(\mathbf{x}_k + t\alpha_k \mathbf{p}_k) dt$), $\beta_{k+1} = \frac{\|\nabla f_{k+1}\|^2}{(\nabla f_{k+1} - \nabla f_k)^T \mathbf{p}_k}$, etc.

Restarts 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 *n-step quadratic convergence*: $\frac{\|\mathbf{x}_{k+n} - \mathbf{x}^*\|}{\|\mathbf{x}_k - \mathbf{x}^*\|^2} \leq M$; intuitively because near the minimum, f is approx. quadratic and so after a restart we will have (approximately) the linear CG method (which requires $\mathbf{p}_0 = \text{steepest descent}$).

For large n (when CG is most useful) restarts may never occur, since an approximate solution may be found in less than n steps.

Global convergence

- With restarts and 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: conjugate gradient methods

- Linear CG: $\mathbf{A}_{n \times n}$ sym. pd: solves $\mathbf{Ax} = \mathbf{b} \Leftrightarrow \min \phi(\mathbf{x}) = \frac{1}{2} \mathbf{x}^T \mathbf{Ax} - \mathbf{b}^T \mathbf{x}$.
 - $\{\mathbf{p}_0, \dots, \mathbf{p}_{n-1}\}$ conjugate wrt $\mathbf{A} \Leftrightarrow \mathbf{p}_i^T \mathbf{A} \mathbf{p}_j = 0 \ \forall i, j, \mathbf{p}_i \neq 0 \ \forall i$.
 - Finds the solution in at most n steps, each an exact line search along a conjugate direction: $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{p}_k$, $\alpha_k = \frac{\mathbf{r}_k^T \mathbf{r}_k}{\mathbf{p}_k^T \mathbf{A} \mathbf{p}_k}$, $\mathbf{r}_k = \nabla \phi(\mathbf{x}_k) = \mathbf{Ax}_k - \mathbf{b}$.
 - At each step, \mathbf{x}_k is the minimizer over the set $\mathbf{x}_0 + \text{span}\{\mathbf{p}_0, \dots, \mathbf{p}_{k-1}\}$; $\mathbf{r}_{k+1} = \mathbf{r}_k + \alpha_k \mathbf{A} \mathbf{p}_k$; and $\mathbf{r}_k^T \mathbf{p}_i = \mathbf{r}_k^T \mathbf{r}_i = 0 \ \forall i < k$.
 - Conjugate direction \mathbf{p}_k is obtained from the previous one and the current gradient: $\mathbf{p}_k = -\mathbf{r}_k + \beta_k \mathbf{p}_{k-1}$ with $\beta_k = \frac{\mathbf{r}_{k+1}^T \mathbf{r}_{k+1}}{\mathbf{r}_k^T \mathbf{r}_k}$.
 - Initial direction is the steepest descent direction $\mathbf{p}_0 = -\nabla \phi(\mathbf{x}_0)$.
- Space complexity $\mathcal{O}(n)$, time complexity $\mathcal{O}(n^3)$.
But often (e.g. when the eigenvalues of \mathbf{A} are clustered, or \mathbf{A} has low c.n.) it gets very close to the solution in $r \ll n$ steps, so $\mathcal{O}(rn^2)$.
- Nonlinear CG: solves $\min f(\mathbf{x})$ where f is nonlinear in general.
 - Fletcher-Reeves: $\mathbf{r}_k = \nabla f(\mathbf{x}_k)$, α_k is determined by an inexact l.s. satisfying the strong Wolfe conditions, $\beta_{k+1}^{\text{FR}} = \frac{\nabla f_{k+1}^T \nabla f_{k+1}}{\nabla f_k^T \nabla f_k}$.
Has global convergence, but to work well in practice it needs restarts (i.e., set $\beta_k = 0$ every n steps).
 - Polak-Ribière: like FR but $\beta_{k+1}^{\text{PR}} = \frac{\nabla f_{k+1}^T (\nabla f_{k+1} - \nabla f_k)}{\nabla f_k^T \nabla f_k}$.
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).

6 Quasi-Newton methods

- Like Newton's method but using a certain \mathbf{B}_k instead of the Hessian.

Like steepest descent and conjugate gradients, they require only the gradient.

By measuring the changes in gradients over iterations, they construct an approximation \mathbf{B}_k to the Hessian whose accuracy improves gradually and results in superlinear convergence.

- Quadratic model of the objective function (correct to first order):

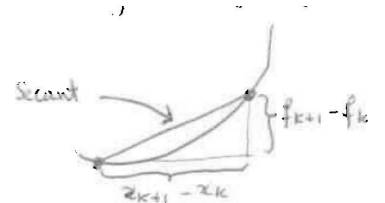
$$m_k(\mathbf{p}) = f_k + \nabla f_k^T \mathbf{p} + \frac{1}{2} \mathbf{p}^T \mathbf{B}_k \mathbf{p}$$

where \mathbf{B}_k is symmetric pd and is updated at every iteration.

- Search direction given by the minimizer of m_k , $\mathbf{p}_k = -\mathbf{B}_k^{-1} \nabla f_k$ (which is descent?).
- Line search $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{p}_k$ with step length chosen to satisfy the Wolfe conditions.

The secant equation

Secant equation: $\mathbf{B}_{k+1} \mathbf{s}_k = \mathbf{y}_k$ where $\mathbf{s}_k = \mathbf{x}_{k+1} - \mathbf{x}_k$ and $\mathbf{y}_k = \nabla f_{k+1} - \nabla f_k$.



- It can be derived:

- From Taylor's th.: $\nabla^2 f_{k+1}(\mathbf{x}_{k+1} - \mathbf{x}_k) \approx \nabla f_{k+1} - \nabla f_k$ (exact if f is quadratic).
- Or by requiring the gradients of the quadratic model m_{k+1} to agree with those of f at \mathbf{x}_{k+1} and \mathbf{x}_k :

$$\begin{aligned} \nabla m_{k+1} &= \nabla f \text{ at } \mathbf{x}_{k+1}: & \text{by construction} \\ \nabla m_{k+1} &= \nabla f \text{ at } \mathbf{x}_k: & \nabla m_{k+1}(-\alpha_k \mathbf{p}_k) = \nabla f_{k+1} - \alpha_k \mathbf{B}_{k+1} \mathbf{p}_k = \nabla f_k. \end{aligned}$$

- It implicitly requires that $\mathbf{s}_k^T \mathbf{y}_k = \mathbf{s}_k^T \mathbf{B}_k \mathbf{s}_k > 0$ (a curvature condition). If f is strongly convex, this is guaranteed (because $\mathbf{s}_k^T \mathbf{y}_k > 0$ for any two points \mathbf{x}_k and \mathbf{x}_{k+1} ; proof: ex. 6.1). Otherwise, it is guaranteed if the line search verifies the 2nd Wolfe condition $\nabla f(\mathbf{x}_k + \alpha_k \mathbf{p}_k)^T \mathbf{p}_k \geq c_2 \nabla f_k^T \mathbf{p}_k$, $0 < c_2 < 1$ (proof: 2nd Wolfe $\Leftrightarrow \nabla f_{k+1}^T \mathbf{s}_k \geq c_2 \nabla f_k^T \mathbf{s}_k \Leftrightarrow \mathbf{y}_k^T \mathbf{s}_k \geq (c_2 - 1) \alpha_k \nabla f_k^T \mathbf{p}_k > 0$).

The secant equation provides only n constraints for $\binom{n}{2}$ dof in \mathbf{B}_k , so it has many solutions. We choose the solution closest to the current matrix \mathbf{B}_k : $\min_{\mathbf{B}} \|\mathbf{B} - \mathbf{B}_k\|$ s.t. \mathbf{B} symmetric pd, $\mathbf{B} \mathbf{s}_k = \mathbf{y}_k$. Different choices of norm are possible; one that allows an easy solution and gives rise to scale invariance is the weighted Frobenius norm $\|\mathbf{A}\|_{\mathbf{W}} = \|\mathbf{W}^{\frac{1}{2}} \mathbf{A} \mathbf{W}^{\frac{1}{2}}\|_F$ (where $\|\mathbf{A}\|_F^2 = \sum_{ij} a_{ij}^2$). \mathbf{W} is any matrix satisfying $\mathbf{W} \mathbf{y}_k = \mathbf{s}_k$ (thus the norm is adimensional, i.e., the solution doesn't depend on the units of the problem).

The DFP method (Davidon-Fletcher-Powell)

If we choose $\mathbf{W}^{-1} = \int_0^1 \nabla^2 f(\mathbf{x}_k + \tau \alpha_k \mathbf{p}_k) d\tau$ (the average Hessian) then the minimizer is unique and is the following rank-2 update:

$$\text{DFP: } \begin{cases} \mathbf{B}_{k+1} = (\mathbf{I} - \gamma_k \mathbf{y}_k \mathbf{s}_k^T) \mathbf{B}_k (\mathbf{I} - \gamma_k \mathbf{s}_k \mathbf{y}_k^T) + \gamma_k \mathbf{y}_k \mathbf{y}_k^T \\ \mathbf{H}_{k+1} = \mathbf{H}_k - \frac{\mathbf{H}_k \mathbf{y}_k \mathbf{y}_k^T \mathbf{H}_k}{\mathbf{y}_k^T \mathbf{H}_k \mathbf{y}_k} + \frac{\mathbf{s}_k \mathbf{s}_k^T}{\mathbf{y}_k^T \mathbf{s}_k} \end{cases} \quad \text{with } \gamma_k = \frac{1}{\mathbf{y}_k^T \mathbf{s}_k} \quad (\mathbf{y}_k^T \mathbf{s}_k > 0 \text{ from earlier curv. cond.})$$

(Pf.: SMW formula)

Where $\mathbf{H}_k = \mathbf{B}_k^{-1}$. Using \mathbf{B}_k directly requires solving $\mathbf{p}_k = -\mathbf{B}_k^{-1} \nabla f_k$, which is $\mathcal{O}(n^3)$, while using \mathbf{H}_k gives us $\mathbf{p}_k = -\mathbf{H}_k \nabla f_k$, which is $\mathcal{O}(n^2)$.

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

We apply the conditions to $\mathbf{H}_{k+1} = \mathbf{B}_{k+1}^{-1}$ rather than \mathbf{B}_{k+1} :

$$\mathbf{H}_{k+1} = \arg \min_{\mathbf{H}} \|\mathbf{H} - \mathbf{H}_k\| \text{ s.t. } \mathbf{H} \text{ symmetric pd, } \mathbf{H}\mathbf{y}_k = \mathbf{s}_k$$

with the same norm as before, where $\mathbf{W}\mathbf{s}_k = \mathbf{y}_k$. For \mathbf{W} = the average Hessian we obtain:

$$\text{BFGS: } \begin{cases} \mathbf{H}_{k+1} = (\mathbf{I} - \rho_k \mathbf{s}_k \mathbf{y}_k^T) \mathbf{H}_k (\mathbf{I} - \rho_k \mathbf{y}_k \mathbf{s}_k^T) + \rho_k \mathbf{s}_k \mathbf{s}_k^T & \text{with } \rho_k = \frac{1}{\mathbf{y}_k^T \mathbf{s}_k} \\ \mathbf{B}_{k+1} = \mathbf{B}_k - \frac{\mathbf{B}_k \mathbf{s}_k \mathbf{s}_k^T \mathbf{B}_k}{\mathbf{s}_k^T \mathbf{B}_k \mathbf{s}_k} + \frac{\mathbf{y}_k \mathbf{y}_k^T}{\mathbf{y}_k^T \mathbf{s}_k} & (\text{Pf.: SMW formula}) \end{cases}$$

We have: $\mathbf{H}_k \text{ pd} \Rightarrow \mathbf{H}_{k+1} \text{ pd}$ (proof: $\mathbf{z}^T \mathbf{H}_{k+1} \mathbf{z} > 0$ if $\mathbf{z} \neq 0$). We take the initial matrix as $\mathbf{H}_0 = \mathbf{I}$ 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 $\mathbf{H}_n = \nabla^2 f^{-1}$. (🔪 Why do many methods work well with quad. f ?)
- BFGS is the best quasi-Newton method. With an adequate line search (e.g. 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 6.1 (BFGS): given starting point \mathbf{x}_0 , convergence tolerance $\epsilon > 0$:

```

H0 ← I, k ← 0
while  $\|\nabla f_k\| > \epsilon$ 
    pk ← −Hk ∇fk                                     search direction
    xk+1 ← xk + αk pk                                     line search with Wolfe cond.
    sk ← xk+1 − xk, yk ← ∇fk+1 − ∇fk, Hk+1 ← BFGS update       update inverse Hessian
    k ← 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. 7).
 - Time: $\mathcal{O}(n^2)$ matrix \times vector, outer products.
- Global convergence if \mathbf{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.
- Local convergence: if BFGS converges, then its order is superlinear under mild conditions.

	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)

By requiring $\mathbf{B}_{k+1} = \mathbf{B}_k + \sigma \mathbf{v} \mathbf{v}^T$ where $\sigma = \pm 1$ and \mathbf{v} is a nonzero vector, and substituting in the secant eq., we obtain:

$$\text{SR1: } \begin{cases} \mathbf{B}_{k+1} = \mathbf{B}_k + \frac{(\mathbf{y}_k - \mathbf{B}_k \mathbf{s}_k)(\mathbf{y}_k - \mathbf{B}_k \mathbf{s}_k)^T}{(\mathbf{y}_k - \mathbf{B}_k \mathbf{s}_k)^T \mathbf{s}_k} \\ \mathbf{H}_{k+1} = \mathbf{H}_k + \frac{(\mathbf{s}_k - \mathbf{H}_k \mathbf{y}_k)(\mathbf{s}_k - \mathbf{H}_k \mathbf{y}_k)^T}{(\mathbf{s}_k - \mathbf{H}_k \mathbf{y}_k)^T \mathbf{y}_k} \end{cases}$$

Generates very good Hessian approximations, often better than BFGS's (indeed BFGS only produces pd \mathbf{B}_k), but:

- Does not necessarily preserve pd \Rightarrow use in trust-region (not l.s.) framework.
- May not satisfy the secant equation $\mathbf{y}_k = \mathbf{B}_{k+1} \mathbf{s}_k$, $\mathbf{s}_k = \mathbf{H}_{k+1} \mathbf{y}_k$ if $(\mathbf{y}_k - \mathbf{B}_k \mathbf{s}_k)^T \mathbf{s}_k = 0 \Rightarrow$ skipping the update if the denominator is small works well in practice:
if $|\mathbf{s}_k^T (\mathbf{y}_k - \mathbf{B}_k \mathbf{s}_k)| < r \|\mathbf{s}_k\| \|\mathbf{y}_k - \mathbf{B}_k \mathbf{s}_k\|$ then $\mathbf{B}_{k+1} = \mathbf{B}_k$ else $\mathbf{B}_{k+1} = \text{SR1}$ [use $r \sim 10^{-8}$].

The Broyden class

$\mathbf{B}_{k+1} = (1 - \phi_k) \mathbf{B}_{k+1}^{\text{BFGS}} + \phi_k \mathbf{B}_{k+1}^{\text{DFP}}$ for $\phi_k \in \mathbb{R}$.

- generalizes BFGS, DFP and SR1
- symmetric
- preserves pd for $\phi_k \in [0, 1]$
- satisfies the secant equation.

Review: quasi-Newton methods

- Newton's method with an approximate Hessian:
 - Quadratic model of objective function f : $m_k(\mathbf{p}) = f_k + \nabla f_k^T \mathbf{p} + \frac{1}{2} \mathbf{p}^T \mathbf{B}_k \mathbf{p}$.
 - Search direction is the minimizer of m_k : $\mathbf{p}_k = -\mathbf{B}_k^{-1} \nabla f_k$.
 - Inexact line search with Wolfe conditions; always try $\alpha_k = 1$ first.
 - \mathbf{B}_k is symmetric pd (for DFP/BFGS) and is updated at each k given the current and previous gradients, so that it approximates $\nabla^2 f_k$.

- Idea: $\nabla^2 f_{k+1}(\underbrace{\mathbf{x}_{k+1} - \mathbf{x}_k}_{\mathbf{s}_k}) \simeq \underbrace{\nabla f_{k+1} - \nabla f_k}_{\mathbf{y}_k}$ (by Taylor's th.).

Secant equation: $\mathbf{B}_{k+1} \mathbf{s}_k = \mathbf{y}_k$; implies $\nabla m_{k+1} = \nabla f$ at $\mathbf{x}_k, \mathbf{x}_{k+1}$.

- DFP method (Davidon-Fletcher-Powell): \mathbf{B}_{k+1} satisfies the secant equation and is closest to \mathbf{B}_k (in a precise sense) \Rightarrow rank-2 update

$$\mathbf{H}_{k+1} (= \mathbf{B}_{k+1}^{-1}) = \mathbf{H}_k - \frac{\mathbf{H}_k \mathbf{y}_k \mathbf{y}_k^T \mathbf{H}_k^T}{\mathbf{y}_k^T \mathbf{H}_k \mathbf{y}_k} + \frac{\mathbf{s}_k \mathbf{s}_k^T}{\mathbf{y}_k^T \mathbf{s}_k} \Rightarrow \mathbf{p}_k = -\mathbf{H}_k \nabla f_k \text{ in } \mathcal{O}(n^2).$$

- BFGS method (Broyden-Fletcher-Goldfarb-Shanno): $\mathbf{H}_{k+1} (= \mathbf{B}_{k+1}^{-1})$ satisfies the secant eq. and is closest to \mathbf{H}_k (in a precise sense) \Rightarrow rank-2 update

$$\mathbf{H}_{k+1} = (\mathbf{I} - \rho_k \mathbf{s}_k \mathbf{y}_k^T) \mathbf{H}_k (\mathbf{I} - \rho_k \mathbf{y}_k \mathbf{s}_k^T) + \rho_k \mathbf{s}_k \mathbf{s}_k^T \text{ with } \rho_k = \frac{1}{\mathbf{y}_k^T \mathbf{s}_k}.$$

Use $\mathbf{H}_0 = \mathbf{I}$. Best quasi-Newton method, self-correcting properties.

- SR1 method (symmetric rank-1): rank-1 update, \mathbf{H}_{k+1} not necessarily pd so use with trust region:

$$\mathbf{H}_{k+1} = \mathbf{H}_k + \frac{(\mathbf{s}_k - \mathbf{H}_k \mathbf{y}_k)(\mathbf{s}_k - \mathbf{H}_k \mathbf{y}_k)^T}{(\mathbf{s}_k - \mathbf{H}_k \mathbf{y}_k)^T \mathbf{y}_k}.$$

- Global convergence: no general results, though the methods are robust in practice.
- Convergence rate: superlinear.

	Newton	Quasi-Newton
Convergence rate	Quadratic	Superlinear
Cost per iteration $\begin{cases} \text{time} \\ \text{space} \end{cases}$	$\mathcal{O}(n^3)$	$\mathcal{O}(n^2)$
	$\mathcal{O}(n^2)$	$\mathcal{O}(n^2)$
Hessian required	yes	no

7 Large-scale unconstrained optimization

- Large problems (today): 10^3 – 10^6 variables.
- In large problems, the following can have a prohibitive cost: factorizing the Hessian (solving for the Newton step), or even computing the Hessian or multiplying times it or storing it (note quasi-Newton algorithms generate dense approximate Hessians even if the true Hessian is sparse).
- In these cases we can use the following:
 - Nonlinear CG is applicable without modification, though not very fast.
 - Sparse Hessian: efficient algorithms (in time and memory) exist to factorize it.
 - This chapter: inexact Newton methods (Newton-CG), limited-memory quasi-Newton methods (L-BFGS).

Inexact Newton methods

Newton step: solution of the $n \times n$ linear system $\nabla^2 f(\mathbf{x}_k) \mathbf{p}_k = -\nabla f(\mathbf{x}_k)$.

- Expensive: computing the Hessian is a major task, $\mathcal{O}(n^2)$, and solving the system is $\mathcal{O}(n^3)$.
- Not robust: far from a minimizer, need to ensure \mathbf{p}_k is descent.

Newton-CG method: solve the system approximately with the linear CG method (efficient), terminating if negative curvature is encountered (robustness); can be implemented as line search or trust region.

Inexact Newton steps Terminate the iterative solver (e.g. CG) when the residual $\mathbf{r}_k = \nabla^2 f(\mathbf{x}_k) \mathbf{p}_k + \nabla f(\mathbf{x}_k)$ (where \mathbf{p}_k is the inexact Newton step) is small wrt the gradient (to achieve invariance wrt scalings of f): $\|\mathbf{r}_k\| \leq \eta_k \|\nabla f(\mathbf{x}_k)\|$, where (η_k) is the *forcing sequence*. Under mild conditions, if the initial \mathbf{x}_0 is sufficiently near a minimizer \mathbf{x}^* and eventually we always try the full step $\alpha_k = 1$, we have:

- Th. 7.1: if $0 < \eta_k \leq \eta < 1 \ \forall k$ then $\mathbf{x}_k \rightarrow \mathbf{x}^*$.
- Th. 7.2: rate of convergence $\begin{cases} \eta_k \rightarrow 0 : & \text{superlinear, e.g. } \eta_k = \min(0.5, \sqrt{\|\nabla f(\mathbf{x}_k)\|}) \\ \eta_k = \mathcal{O}(\|\nabla f(\mathbf{x}_k)\|) : & \text{quadratic, e.g. } \eta_k = \min(0.5, \|\nabla f(\mathbf{x}_k)\|). \end{cases}$

But the smaller η_k , the more iterations of CG we need.

( How many Newton-CG iterations and how many CG steps in total does this require if f is strongly convex quadratic?)

How do we get sufficiently near a minimizer?

Newton-CG method We solve the system with the CG method to an accuracy determined by the forcing sequence, but terminating if negative curvature is encountered ($\mathbf{p}_k^T \mathbf{A} \mathbf{p}_k \leq 0$). If the very 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). Alg. 7.1

The method behaves like:
$$\begin{cases} \text{pd Hessian:} & \text{pure (inexact) Newton direction} \\ \text{nd Hessian:} & \text{steepest descent} \\ \text{not definite Hessian:} & \text{finds some descent direction.} \end{cases}$$

Problem (as in the modified Hessian Newton's method): if the Hessian is nearly singular, the Newton-CG direction can be very long.

Hessian-free Newton methods: the Hessian-vector product $\nabla^2 f_k \mathbf{v}$ can be obtained (exactly or approximately) *without computing the Hessian*, see ch. 8.

- In a trust-region way: limit the line search to a region of size Δ_k .

Limited-memory quasi-Newton methods

- Useful for large problems with costly or nonsparse Hessian.
- They keep simple, compact approximations of the Hessian based on a few n -vectors (rather than an $n \times n$ matrix).
- Linear convergence but fast rate.
- Focus on L-BFGS, which uses curvature information from only the most recent iterations.

Limited-memory BFGS (L-BFGS) BFGS review:

- Step: $\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k \mathbf{H}_k \nabla f_k$.
- Update: $\underbrace{\mathbf{H}_{k+1}}_{\text{dense so we can't use or store}} = \mathbf{V}_k^T \mathbf{H}_k \mathbf{V}_k + \rho_k \mathbf{s}_k \mathbf{s}_k^T$ with
$$\begin{cases} \mathbf{V}_k = \mathbf{I} - \rho_k \mathbf{y}_k \mathbf{s}_k^T \\ \rho_k = 1 / \mathbf{y}_k^T \mathbf{s}_k \\ \mathbf{s}_k = \mathbf{x}_{k+1} - \mathbf{x}_k, \mathbf{y}_k = \nabla f_{k+1} - \nabla f_k. \end{cases}$$

L-BFGS: store modified version of \mathbf{H}_k implicitly, by storing $m \ll n$ of the vector pairs $\{\mathbf{s}_i, \mathbf{y}_i\}$.

- Product $\mathbf{H}_k \nabla f_k$ = sum of inner products involving ∇f_k and the pairs $\{\mathbf{s}_i, \mathbf{y}_i\}$.
- After the new iterate is computed, replace oldest pair with newest one.
- Modest values of m (~ 3 – 20) work ok in practice; best m is problem-dependent.
- Slow convergence in ill-conditioned problems.
- Update, in detail:
 - Iteration k : $\mathbf{x}_k, \{\mathbf{s}_i, \mathbf{y}_i\}$ for $i = k - m, \dots, k - 1$.
 - \mathbf{H}_k = eq. (7.19) by recursively expanding the update and assuming an initial Hessian approximation \mathbf{H}_k^0 , from which the product $\mathbf{H}_k \nabla f_k$ can be computed (algorithm 7.4) in $\mathcal{O}(m)$ vector-vector products, plus the matrix-vector product by $\mathbf{H}_k^0 \implies$ choose it (say) diagonal, in particular $\mathbf{H}_k^0 = \gamma_k \mathbf{I}$ with γ_k = eq. (7.20) helps the direction \mathbf{p}_k to be well scaled and so the step length $\alpha_k = 1$ is mostly accepted.
 - Use l.s. with (strong) Wolfe conditions to make BFGS stable.
 - The first $m - 1$ iterates are as in BFGS.

Algorithm 7.5 (L-BFGS): given \mathbf{x}_0 :

Choose $m > 0$; $k \leftarrow 0$

repeat

Choose \mathbf{H}_k^0

e.g. eq. (7.20)

$\mathbf{p}_k \leftarrow -\mathbf{H}_k \nabla f_k$

algorithm 7.4

$\mathbf{x}_{k+1} \leftarrow \mathbf{x}_k + \alpha_k \mathbf{p}_k$

l.s. with Wolfe conditions

Discard $\{\mathbf{s}_{k-m}, \mathbf{y}_{k-m}\}$ if $k > m$

Store pair $\mathbf{s}_k \leftarrow \mathbf{x}_{k+1} - \mathbf{x}_k$, $\mathbf{y}_k \leftarrow \nabla f_{k+1} - \nabla f_k$

$k \leftarrow k + 1$

until convergence

Relationship with CG methods

- Limited-memory methods historically evolved as improvements of CG methods.
- CG–Hestenes–Stiefel: we have (from $\mathbf{s}_k = \alpha_k \mathbf{p}_k$, $\beta_{k+1}^{\text{HS}} = \frac{\nabla f_{k+1}^T (\nabla f_{k+1} - \nabla f_k)}{(\nabla f_{k+1} - \nabla f_k)^T \mathbf{p}_k}$, $\mathbf{p}_{k+1} = -\nabla f_{k+1} + \beta_{k+1}^{\text{HS}} \mathbf{p}_k$)

$$\mathbf{p}_{k+1} = -\nabla f_{k+1} + \frac{\nabla f_{k+1}^T \mathbf{y}_k}{\mathbf{y}_k^T \mathbf{p}_k} \mathbf{p}_k = -\hat{\mathbf{H}}_{k+1} \nabla f_{k+1} \quad \text{with } \hat{\mathbf{H}}_{k+1} = \mathbf{I} - \frac{\mathbf{s}_k \mathbf{y}_k^T}{\mathbf{y}_k^T \mathbf{s}_k}$$

which resembles quasi-Newton iterates, but $\hat{\mathbf{H}}_{k+1}$ is neither symmetric nor pd.

- The following *memoryless BFGS* is symmetric, pd and satisfies the secant eq. $\mathbf{H}_{k+1} \mathbf{y}_k = \mathbf{s}_k$:

$$\mathbf{H}_{k+1} = \left(\mathbf{I} - \frac{\mathbf{s}_k \mathbf{y}_k^T}{\mathbf{y}_k^T \mathbf{s}_k} \right) \left(\mathbf{I} - \frac{\mathbf{y}_k \mathbf{s}_k^T}{\mathbf{y}_k^T \mathbf{s}_k} \right) + \frac{\mathbf{s}_k \mathbf{s}_k^T}{\mathbf{y}_k^T \mathbf{s}_k} \equiv \begin{cases} \text{BFGS update with } \mathbf{H}_k = \mathbf{I} \\ \text{L-BFGS with } m = 1 \text{ and } \mathbf{H}_k^0 = \mathbf{I}. \end{cases}$$

And, with exact l.s. ($\nabla f_{k+1}^T \mathbf{p}_k = 0 \ \forall k$): $\mathbf{p}_{k+1} = -\mathbf{H}_{k+1} \nabla f_{k+1} \equiv \text{CG-HS} \equiv \text{CG-PR}$.

General limited-memory updating In general, we can represent the quasi-Newton approximation to the Hessian \mathbf{B}_k and inverse Hessian \mathbf{H}_k (BFGS, SR1) as an outer-product form:

$$\mathbf{B}_k = \frac{1}{\gamma_k} \mathbf{I} + \underbrace{\begin{bmatrix} \phantom{\rule{1cm}{0.5cm}} \end{bmatrix}}_n \underbrace{\begin{bmatrix} \phantom{\rule{1cm}{0.5cm}} \end{bmatrix}}_{2m \times 2m} \underbrace{\begin{bmatrix} \phantom{\rule{1cm}{0.5cm}} \end{bmatrix}}_{2m \times n}.$$

This could be used in a trust-region method or in a constrained optimization method. Efficient since updating \mathbf{B}_k costs $\mathcal{O}(mn + m^3)$ and matrix-vector products $\mathbf{B}_k \mathbf{v}$ cost $\mathcal{O}(mn + m^2)$.

Sparse quasi-Newton updates

- Assume we know the sparsity pattern Ω of the true Hessian and demand that the quasi-Newton approximation \mathbf{B}_k to it have that same pattern:

$$\min_{\mathbf{B}} \|\mathbf{B} - \mathbf{B}_k\|_F^2 = \sum_{(i,j) \in \Omega} (B_{ij} - (B_k)_{ij})^2 \text{ s.t. } \mathbf{B} \mathbf{s}_k = \mathbf{y}_k, \mathbf{B} = \mathbf{B}^T, B_{ij} = 0 \text{ for } (i,j) \notin \Omega.$$

Solution \mathbf{B}_{k+1} given by solving an $n \times n$ linsys with the same sparsity pattern, but is not necessarily pd.

- Then, use \mathbf{B}_{k+1} within trust-region method.
- Smaller storage. And, perhaps more accurate approximation? Unfortunately:
 - update not scale invariant under linear transformations
 - disappointing practical performance; seemingly, bad model for \mathbf{B}_k and so poor approximation.

Partially separable functions

- Separable function (e.g. $f(\mathbf{x}) = f_1(x_1, x_3) + f_2(x_2, x_4, x_6) + f_3(x_5)$) \Rightarrow independent optimizations.
- Partially separable function = sum of element functions, each dependent on a few variables \Rightarrow sparse gradient, Hessian for each function; it is efficient to maintain quasi-Newton approximations to each element function. Essentially, we work on a lower-dimensional space for each function.

$$\begin{aligned}
 f(\mathbf{x}) = \sum_i \phi_i(\mathbf{U}_i \mathbf{x}) &\Rightarrow \nabla f(\mathbf{x}) = \sum_i \mathbf{U}_i^T \nabla \phi_i(\mathbf{U}_i \mathbf{x}), \quad \nabla^2 f(\mathbf{x}) = \sum_i \mathbf{U}_i^T \nabla^2 \phi_i(\mathbf{U}_i \mathbf{x}) \mathbf{U}_i \\
 &\Rightarrow \nabla^2 f(\mathbf{x}) \approx \mathbf{B} = \sum_i \mathbf{U}_i^T \mathbf{B}_{[i]} \mathbf{U}_i
 \end{aligned}$$

where \mathbf{U}_i = compactifying matrix (sparse) of $m_i \times n$ and $\mathbf{B}_{[i]}$ = $m_i \times m_i$ quasi-Newton approx. of $\nabla^2 \phi_i$. Then, use within trust-region method: $\mathbf{B}_k \mathbf{p}_k = -\nabla f_k$, which can be solved by linear CG with low-dim operations using \mathbf{U}_i and $\mathbf{B}_{[i]}$ (avoiding explicitly constructing \mathbf{B}_k). See example in page 187.

Review: large-scale unconstrained optimization

- Avoid factorizing or even computing the (approximate) Hessian in $\mathbf{B}_k \mathbf{p}_k = -\nabla f_k$.
- Nonlinear CG (ch. 5), linear convergence, not very fast.
- If sparse Hessian or partially separable function, take advantage of it (e.g. separable low-dim quasi-Newton approximations for each element function).
- Inexact Newton methods:
 - *Newton-CG* solves the Newton direction system approximately with linear CG, terminating when either a sufficiently accurate direction is found (using a forcing sequence) or when negative curvature is found (ensures descent direction).
 - Convergence rate: linear, superlinear or quadratic depending on the forcing sequence.
 - Hessian-free if computing $\nabla^2 f \mathbf{v}$ products without computing $\nabla^2 f$ (ch. 8).
- Limited-memory quasi-Newton methods:
 - *L-BFGS* implicitly constructs approximate Hessians based on $m \ll n$ outer products constructed using the last m pairs $\{\mathbf{s}_i, \mathbf{y}_i\}$.
 - Obtained by unfolding the BFGS formula for \mathbf{H}_{k+1} over the previous m iterations.
 - Linear convergence but generally faster than nonlinear CG.
 - The memoryless L-BFGS ($m = 1$) is similar to nonlinear CG methods.

8 Calculating derivatives

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

Finite-difference derivative approximations

Gradient of $f: \mathbb{R}^n \rightarrow \mathbb{R}$, from Taylor's th.:

$$\frac{\partial f}{\partial x_i} = \begin{cases} \frac{f(\mathbf{x} + \epsilon \mathbf{e}_i) - f(\mathbf{x})}{\epsilon} + \mathcal{O}(\epsilon), & \leftarrow \text{Forward difference, } n+1 \text{ function evaluations} \\ \underbrace{\frac{f(\mathbf{x} + \epsilon \mathbf{e}_i) - f(\mathbf{x} - \epsilon \mathbf{e}_i)}{2\epsilon}}_{\text{Approximate the deriv. with this}} + \underbrace{\mathcal{O}(\epsilon^2)}_{\text{truncation error}} & \leftarrow \text{Central difference, } 2n \text{ function evaluations.} \end{cases}$$

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, $\epsilon \sim u^{\frac{1}{2}}$ with error $\sim u^{\frac{1}{2}}$ for forward diff. and $\epsilon \sim u^{\frac{1}{3}}$ with error $\sim u^{\frac{2}{3}}$ for central diff., where u ($\approx 10^{-16}$ in double precision) is the unit roundoff.

Pf.: assume $\|\nabla^2 f\| \leq L$ and $|f| \leq L_f$ in the region of interest. Then $\frac{\partial f}{\partial x_i}(\mathbf{x}) = \frac{f(\mathbf{x} + \epsilon \mathbf{e}_i) - f(\mathbf{x})}{\epsilon} + \delta_\epsilon$ with $|\delta_\epsilon| \leq \frac{L}{2}\epsilon$. But the machine representation of f at any \mathbf{x} has a relative error $|\text{comp}(f(\mathbf{x})) - f(\mathbf{x})| \leq uL_f$. Thus, the error is bounded by $\frac{L}{2}\epsilon + \frac{2uL_f}{\epsilon}$, which is minimal for $\epsilon^2 = \frac{4L_f u}{L}$ or $\epsilon \sim \sqrt{u}$ (and error $\sim \sqrt{u}$). Similar derivation for the central diff.

Hessian of f : $\mathcal{O}(n^2)$ function evaluations:

$$\frac{\partial^2 f}{\partial x_i \partial x_j}(\mathbf{x}) = \frac{f(\mathbf{x} + \epsilon \mathbf{e}_i + \epsilon \mathbf{e}_j) - f(\mathbf{x} + \epsilon \mathbf{e}_i) - f(\mathbf{x} + \epsilon \mathbf{e}_j) + f(\mathbf{x})}{\epsilon^2} + \mathcal{O}(\epsilon).$$

The roundoff error accumulates badly in this formula.

Jacobian-vector product $\mathbf{J}(\mathbf{x}) \mathbf{p}$ where $\mathbf{r}: \mathbb{R}^n \rightarrow \mathbb{R}^m$ and $\mathbf{J}_{m \times n}$ is the Jacobian of \mathbf{r} :

$$\begin{aligned} \mathbf{J}(\mathbf{x}) \mathbf{p} &= \frac{\mathbf{r}(\mathbf{x} + \epsilon \mathbf{p}) - \mathbf{r}(\mathbf{x})}{\epsilon} + \mathcal{O}(\epsilon) && \text{(by Taylor's th.)} \\ \nabla^2 f(\mathbf{x}) \mathbf{p} &= \frac{\nabla f(\mathbf{x} + \epsilon \mathbf{p}) - \nabla f(\mathbf{x})}{\epsilon} + \mathcal{O}(\epsilon) && \text{(in particular for } \mathbf{r} = \nabla f) \\ \nabla^2 f(\mathbf{x}) \mathbf{e}_i &= \frac{\nabla f(\mathbf{x} + \epsilon \mathbf{e}_i) - \nabla f(\mathbf{x})}{\epsilon} + \mathcal{O}(\epsilon) && \text{(column } i \text{ of the Hessian).} \end{aligned}$$

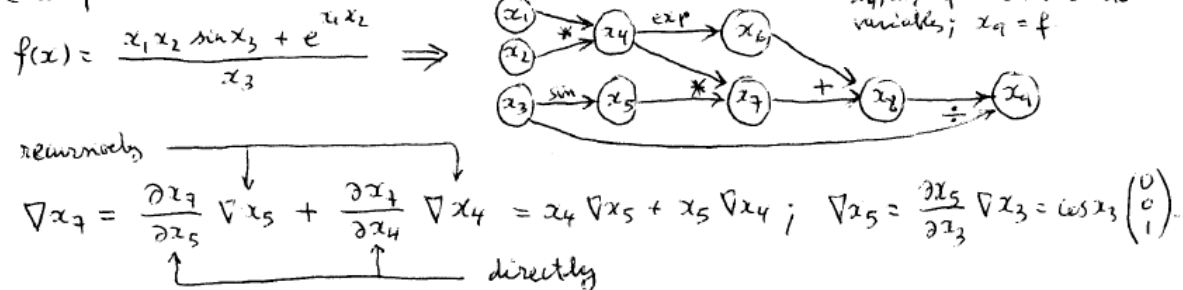
Using the expression for $\nabla^2 f(\mathbf{x}) \mathbf{p}$ in Newton-CG gives rise to a Hessian-free Newton method (ch. 7).

- If the Jacobian or Hessian is sparse, it is possible to reduce the number of function evaluations by cleverly choosing the perturbation vector \mathbf{p} (graph-coloring problem).
- Computationally, the finite-difference approximation of ∇f , etc. can cost more than computing them from their analytical expression (ex.: quadratic f), though this depends on f .
- Numerical gradients are also useful to check whether the expression for a gradient calculated by hand is correct.

Exact derivatives by automatic differentiation

- Build a computational graph of f using intermediate variables.
- Apply the chain rule: $\nabla_{\mathbf{x}} h(\mathbf{y}(\mathbf{x})) = \sum_i \frac{\partial h}{\partial y_i} \nabla y_i(\mathbf{x})$ where $\mathbb{R}^n \xrightarrow{\mathbf{y}} \mathbb{R}^m \xrightarrow{h} \mathbb{R}$.

Example:



- Computes the exact value of f , ∇f or $\nabla^2 f \mathbf{p}$ recursively. Cost (time and space) depends on the structure of f .
- Done automatically by a software tool.
- Disadvantage: simplification of expressions, reuse of operations; e.g. differentiate $\tan x - x$, $\ln\left(\frac{x-1}{x+1}\right)$, $\frac{1}{1+e^{-ax}}$.

Exact derivatives by symbolic differentiation

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

Review: Calculating derivatives

- *Finite-difference approximation* with perturbation ϵ for the gradient, Hessian or Jacobian:
 - Specific finite difference scheme obtained from Taylor's th.
 - ϵ : large truncation error if too large, large roundoff error if too small.
 - ∇f , $\nabla^2 f$ cost $\mathcal{O}(n)$ and $\mathcal{O}(n^2)$ different evaluations of f , respectively; less if sparsity.
- Exact derivatives by *automatic differentiation*: a software tool applies the chain rule recursively. The cost depends on the structure of f .
- Exact derivatives by *symbolic differentiation* with Mathematica, Maple, etc.

9 Derivative-free optimization

Evaluating ∇f in practice is sometimes impossible, e.g.:

- $f(\mathbf{x})$ can be the result of an experimental measurement or a simulation (so analytic form of f unknown).
- Even if known, coding ∇f may be time-consuming or impractical.

Approaches:

- Approximate gradient and possibly Hessian using finite differences (ch. 8), then apply derivative-based method (previous chapters). But:
 - Number of function evaluations may be excessive.
 - Unreliable with noise (= inaccuracy in function evaluation).
- Don't approximate the gradient, instead use function values at a set of sample points and determine a new iterate by a different means (this chapter). But: less developed and less efficient than derivative-based methods; effective only for small problems; difficult to use with general constraints.

If possible, try methods in this order: derivative-based > finite-difference-based > derivative-free.

Finite differences and noise (illustrative analysis)

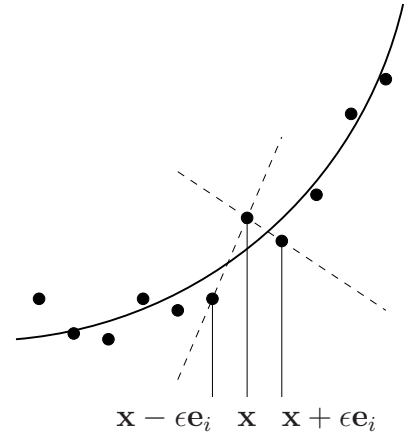
Consider smooth f . Noise in evaluating f can arise because:

- stochastic simulation: random error because finite number of trials
- differential-equation solver (or some other complex numerical procedure): small but nonzero tolerance in calculations.

Write $f(\mathbf{x}) = h(\mathbf{x}) + \phi(\mathbf{x})$ with smooth h and noise ϕ (which need not be a function of \mathbf{x}), consider centered finite-difference approx $\nabla_\epsilon f(\mathbf{x}) = \left(\frac{f(\mathbf{x} + \epsilon \mathbf{e}_i) - f(\mathbf{x} - \epsilon \mathbf{e}_i)}{2\epsilon} \right)_{i=1, \dots, n}$ and let $\eta(\mathbf{x}; \epsilon) = \sup_{\|\mathbf{z} - \mathbf{x}\|_\infty \leq \epsilon} |\phi(\mathbf{z})|$ (noise level at \mathbf{x}). We have:

Lemma 9.1. $\nabla^2 h$ Lipschitz continuous in a neighborhood of the box $\{\mathbf{z} : \|\mathbf{z} - \mathbf{x}\|_\infty \leq \epsilon\}$ with Lipschitz constant L_h . Then:

$$\underbrace{\|\nabla_\epsilon f(\mathbf{x}) - \nabla h(\mathbf{x})\|_\infty}_{\text{approximation error}} \leq \underbrace{L_h \epsilon^2}_{\text{finite-diff. approx. error}} + \underbrace{\frac{\eta(\mathbf{x}; \epsilon)}{\epsilon}}_{\text{noise error}}$$



(Pf.: as for unit roundoff argument in finite diff.)

“If the noise dominates ϵ , no accuracy in $\nabla_\epsilon f$ and so little hope that $-\nabla_\epsilon f$ will be descent.” So, instead of using *close* samples, it may be better to use samples more *widely* separated.

Model-based methods

- Build a model m_k as a quadratic function that interpolates f at an appropriate set of samples. Compute a step with a trust-region strategy (since m_k is usually nonconvex).
- Model: samples $\mathcal{Y} = \{\mathbf{y}^1, \dots, \mathbf{y}^q\} \subset \mathbb{R}^n$ with current iterate $\mathbf{x}_k \in \mathcal{Y}$ and having lowest function value in \mathcal{Y} . Construct $m_k(\mathbf{x}_k + \mathbf{p}) = c + \mathbf{g}^T \mathbf{p} + \frac{1}{2} \mathbf{p}^T \mathbf{G} \mathbf{p}$ (we can't use $\mathbf{g} = \nabla f(\mathbf{x}_k)$, $\mathbf{G} = \nabla^2 f(\mathbf{x}_k)$) by imposing interpolation conditions $m_k(\mathbf{y}^l) = f(\mathbf{y}^l)$, $l = 1, \dots, q$ (linear system). Need $q = \frac{1}{2}(n+1)(n+2)$ (exe. 9.2) and choose points so linsys is nonsingular.
- Step: $\min_{\|\mathbf{p}\|_2 \leq \Delta} m_k(\mathbf{x}_k + \mathbf{p})$, etc. as in trust-region.
- If sufficient reduction: the latest iterate replaces one sample in \mathcal{Y} .
Else: if \mathcal{Y} is adequate (= low condition number of linsys) then reduce Δ else improve \mathcal{Y} .
- Good initial \mathcal{Y} : vertices and edge midpoints of simplex in \mathbb{R}^n (exe. 9.2).
- Naive implementation costs $\mathcal{O}(n^6)$.
Acceleration: update m_k at each iteration rather than recomputing it from scratch.
Even with this it still costs $\mathcal{O}(n^4)$, very expensive.
- Linear model ($\mathbf{G} = \mathbf{0}$): $q = n + 1$ parameters, $\mathcal{O}(n^3)$ step (more slowly convergent).
Hybrid: start with linear steps, switch to quadratic when $q = \frac{1}{2}(n+1)(n+2)$ function values become available.

Coordinate- and pattern-search methods

Search along certain specified directions from the current iterate for a point of lower f value.

Coordinate-descent algorithm (or method of alternating variables) \mathbf{p}_k cycles through the n coordinate dimension $\mathbf{e}_1, \dots, \mathbf{e}_n$ in turn. fig. 9.1

- 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 $\nabla f_k \nrightarrow 0$.
- 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.
- Variants:
 - back-and-forth approach repeats $\mathbf{e}_1 \ \mathbf{e}_2 \ \dots \ \mathbf{e}_{n-1} \ \mathbf{e}_n \ \mathbf{e}_{n-1} \ \dots \ \mathbf{e}_2 \ \mathbf{e}_1 \ \mathbf{e}_2 \ \dots$
 - *Hooke-Jeeves*: after sequence of coordinate descent steps, search along first and last point in the cycle.
- Very useful in special cases, e.g. when alternating over *groups* of variables so that the optimization over each group is easy. Ex.: $f(\mathbf{x}, \mathbf{A}) = \sum_{j=1}^m \|\mathbf{y}_j - \mathbf{A} \mathbf{x}_j\|^2$.

Pattern search Generalizes coordinate search to a richer set of directions at each iteration. At each iterate \mathbf{x}_k :

- Choose a certain set of search directions, $\mathcal{D}_k = \{\mathbf{p}_1, \dots\}$ and define a frame centered at \mathbf{x}_k by points \mathbf{x}_k at a given step length $\gamma_k > 0$ along each direction: $\{\mathbf{x}_k + \gamma_k \mathbf{p}_1, \dots\}$.
- Evaluate f at each frame point:
 - If significantly lower f value found, adopt as new iterate and shift frame center to it. Possibly, increase γ_k (expand the frame).
 - Otherwise, stay at \mathbf{x}_k and reduce γ_k (shrink the frame).
- Possibly, change the directions.

Ex.: algorithm 9.2, which eventually shrinks the frame around a stationary point. Global convergence under certain conditions on the choice of directions:

- (1) At least one direction in \mathcal{D}_k should be descent (unless $\nabla f(\mathbf{x}_k) = \mathbf{0}$), specifically: $\min_{\mathbf{v} \in \mathbb{R}^n} \max_{\mathbf{p} \in \mathcal{D}_k} \cos(\widehat{\mathbf{v}, \mathbf{p}}) \geq \delta$ for constant $\delta > 0$.
- (2) All directions have roughly similar length (so we can use a single step length): $\forall \mathbf{p} \in \mathcal{D}_k$: $\beta_{\min} \leq \|\mathbf{p}\| \leq \beta_{\max}$, for some positive β_{\min} , β_{\max} and all k .

Examples of such \mathcal{D}_k :

fig. 9.2

- Coordinate dirs $\pm \mathbf{e}_1, \dots, \pm \mathbf{e}_n$.
- Simplex set: $\mathbf{p}_i = \frac{1}{2n} \mathbf{e} - \mathbf{e}_i$, $i = 1, \dots, n$ and $\mathbf{p}_{n+1} = \frac{1}{2n} \mathbf{e}$ (where $\mathbf{e} = (1, \dots, 1)^T$).

The coordinate-descent method uses $\mathcal{D}_k = \{\mathbf{e}_i, -\mathbf{e}_i\}$ for some i at each k , which violates condition (1) (exe. 9.9).

Nelder-Mead method (downhill simplex)

- At each iteration we keep $n + 1$ points whose convex hull forms a simplex. (A simplex with vertices $\mathbf{z}_1, \dots, \mathbf{z}_{n+1}$ is nondegenerate or nonsingular if the edge matrix $\mathbf{V} = (\mathbf{z}_2 - \mathbf{z}_1, \dots, \mathbf{z}_{n+1} - \mathbf{z}_1)$ is nonsingular.) At each iteration we replace the worst vertex (in f -value) with a better point obtained by reflecting, expanding or contracting the simplex along the line joining the worst vertex with the simplex center of mass. If we can't find a better point this way, we keep only the best vertex and shrink the simplex towards it.
- Reasonable practical performance but sometimes doesn't converge. The average function value $\frac{1}{n+1} \sum_{i=1}^{n+1} f(\mathbf{x}_i)$ decreases after each step except perhaps after a shrinkage step (exe. 9.11).

Proc. 9.5
fig. 9.4



A conjugate-direction method

- Idea: algorithm that builds a set of conjugate directions using only function values (thus minimizes a strictly convex quadratic function); then extend to nonlinear function.
 - *Parallel subspace property*: let $\mathbf{x}_1 \neq \mathbf{x}_2 \in \mathbb{R}^n$ and $\{\mathbf{p}_1, \dots, \mathbf{p}_l\} \subset \mathbb{R}^n$ l.i. Define the two parallel linear varieties $\mathcal{S}_j = \{\mathbf{x}_j + \sum_{i=1}^l \alpha_i \mathbf{p}_i, \alpha_1, \dots, \alpha_l \in \mathbb{R}\}$, $j = 1, 2$; let \mathbf{x}_1^* and \mathbf{x}_2^* be the minimizers of $f(\mathbf{x}) = \frac{1}{2} \mathbf{x}^T \mathbf{A} \mathbf{x} - \mathbf{b}^T \mathbf{x}$ on \mathcal{S}_1 and \mathcal{S}_2 , resp. $\implies \mathbf{x}_2^* - \mathbf{x}_1^*$ is conjugate to $\{\mathbf{p}_1, \dots, \mathbf{p}_l\}$. fig. 9.3.
- Ex. 2D: given \mathbf{x}_0 and (say) $\mathbf{e}_1, \mathbf{e}_2$: (1) minimize from \mathbf{x}_0 along \mathbf{e}_2 to obtain \mathbf{x}_1 , (2) minimize from \mathbf{x}_1 along \mathbf{e}_1 then \mathbf{e}_2 to obtain $\mathbf{z} \Rightarrow \mathbf{z} - \mathbf{x}_1$ is conjugate to \mathbf{e}_2 . Proof
- Algorithm: starting with n l.i. directions, perform n consecutive exact minimizations each along a current direction, then generate a new conjugate direction which replaces the oldest direction; repeat.

Algorithm 9.3: given \mathbf{x}_0 :

Init: $\mathbf{p}_i = \mathbf{e}_i$ for $i = 1, \dots, n$; $\mathbf{x}_1 \leftarrow \arg \min f$ from \mathbf{x}_0 along \mathbf{p}_n ; $k \leftarrow 1$

repeat

min sequentially from \mathbf{x}_k along $\mathbf{p}_1, \dots, \mathbf{p}_n$ to obtain \mathbf{z}

New dirs: $\mathbf{p}_2, \dots, \mathbf{p}_n, \mathbf{z} - \mathbf{x}_k$

New iterate: $\mathbf{x}_{k+1} \leftarrow \arg \min f$ from \mathbf{z} along $\mathbf{z} - \mathbf{x}_k$

$k \leftarrow k + 1$

until convergence

- For quadratic f , terminates in n steps with total $\mathcal{O}(n^2)$ function evaluations (each one $\mathcal{O}(n^2)$) $\Rightarrow \mathcal{O}(n^4)$. For non-quadratic f the l.s. is inexact (using interpolation) and needs some care. Problem: the directions $\{\mathbf{p}_i\}$ tend to become l.d. Heuristics exist to correct this.
- Useful for small problems.

Implicit filtering

- Essentially, steepest descent at an accuracy level ϵ (the finite-difference parameter) that is decreased over iterations.
- Useful when we can control the accuracy in computing f and $\nabla_\epsilon f$ (e.g. if we can control the tolerance of a differential-equation solver, or the number of trials of a stochastic simulation). A more accurate (less noisy) value costs more computation.
- Algorithm decreases ϵ systematically (but hopefully not as quickly as the decay in error) so as to maintain a reasonable accuracy in $\nabla_\epsilon f(\mathbf{x})$. At each iteration, it performs an Armijo (backtracking) search along $-\nabla_\epsilon f(\mathbf{x})$ which is stopped:
 - when $\|\nabla_{\epsilon_k} f(\mathbf{x})\| \leq \epsilon_k$ (i.e., minimum found with accuracy level ϵ_k , so can decrease ϵ),
 - or we reach a fixed number of backtracking steps (i.e., $\nabla_\epsilon f(\mathbf{x})$ is a poor approximation of ∇f , so need to decrease ϵ).
- Converges if ϵ_k is decreased such that $\epsilon_k^2 + \frac{\eta(\mathbf{x}_k; \epsilon_k)}{\epsilon_k} \rightarrow 0$, i.e., the noise level decreases sufficiently fast as the iterates approach a solution.

Review: derivative-free optimization

Methods that use only function values to minimize f (but not ∇f or $\nabla^2 f$). Less efficient than derivative-based methods, but possibly acceptable for small n or in special cases.

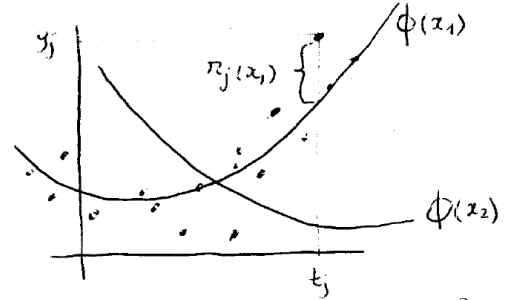
- Finite-difference approximations to the gradient degrade significantly with noise in f .
- *Model-based methods*: build a linear or quadratic model of f by interpolating f at a set of samples and use it with trust-region. Slow convergence rate and very costly steps.
- *Coordinate descent (alternating minimization)*: minimize successively along each variable. If it does converge, its rate of convergence is often much slower than that of steepest descent. Very simple and convenient sometimes.
- *Pattern search*: the iterate \mathbf{x}_k carries a set of directions that is possibly updated based on the values of f along them. Generalizes coordinate descent to a richer direction set.
- *Nelder-Mead method (downhill simplex)*: the iterate \mathbf{x}_k carries a simplex that evolves based on the values of f , falling down and eventually shrinking around a minimizer (if it does converge).
- *Conjugate directions* built using the parallel subspace property: computing the new conjugate direction requires n line searches (CG requires only one).
- *Implicit filtering*: steepest descent at an accuracy level ϵ (the finite-difference parameter) that is decreased over iterations. Useful when we can control the accuracy in computing f and $\nabla_\epsilon f$.

10 Nonlinear least-squares problems

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

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

- Ex.: regression (curve fitting): $r_j = y_j - \phi(\mathbf{x}; t_j)$, $f(\mathbf{x}) = \frac{1}{2} \sum_{j=1}^m (y_j - \phi(\mathbf{x}; t_j))^2$ is the LSQ error of fitting curve $\phi : t \rightarrow y$ (with parameters \mathbf{x}) to the observed data points $\{(t_j, y_j)\}_{j=1}^m$.
If using other norms, e.g. $|r_j|$ or $|r_j|^3$, it won't be a LSQ problem.



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

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

Usually it's easy to compute \mathbf{J} explicitly. Then

$$\begin{aligned} \nabla f(\mathbf{x}) &= \sum_j r_j(\mathbf{x}) \nabla r_j(\mathbf{x}) = \mathbf{J}(\mathbf{x})^T \mathbf{r}(\mathbf{x}) \\ \nabla^2 f(\mathbf{x}) &= \sum_j \nabla r_j(\mathbf{x}) \nabla r_j(\mathbf{x})^T + \sum_j r_j(\mathbf{x}) \nabla^2 r_j(\mathbf{x}) = \overbrace{\mathbf{J}(\mathbf{x})^T \mathbf{J}(\mathbf{x})}^{(*)} + \sum_j r_j(\mathbf{x}) \nabla^2 r_j(\mathbf{x}). \end{aligned}$$

Often $(*)$ is the leading term, e.g.

- 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 problem*: $r_j(\mathbf{x})$ is linear $\forall j \Rightarrow \mathbf{J}(\mathbf{x}) = \mathbf{J}$ constant. Calling $\mathbf{r} = \mathbf{r}(\mathbf{0})$, we have $f(\mathbf{x}) = \frac{1}{2} \|\mathbf{J}\mathbf{x} + \mathbf{r}\|_2^2$ convex[?], $\nabla f(\mathbf{x}) = \mathbf{J}^T(\mathbf{J}\mathbf{x} + \mathbf{r})$, $\nabla^2 f(\mathbf{x}) = \mathbf{J}^T \mathbf{J}$ constant.

([?] is fitting a polynomial to data a linear LSQ problem?)

Minimizer: $\nabla f(\mathbf{x}^*) = \mathbf{0} \Rightarrow \mathbf{J}^T \mathbf{J} \mathbf{x}^* = -\mathbf{J}^T \mathbf{r}$, the *normal equations*: $n \times n$ linear system with pd or psd matrix which can be solved with numerical analysis techniques (Cholesky factorization of $\mathbf{J}^T \mathbf{J}$, or QR or SVD factorization of \mathbf{J} are best depending on the problem; also could use the linear conjugate gradient method for large n .) (If m is very large, do not build \mathbf{J} explicitly but accumulate $\mathbf{J}^T \mathbf{J} = \sum_j \nabla r_j(\mathbf{x}) \nabla r_j(\mathbf{x})^T$ and $\mathbf{J}^T \mathbf{r} = \sum_j r_j(\mathbf{x}) \nabla r_j(\mathbf{x})$).

- For *nonlinear LSQ problems* 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 chapter are applicable too (e.g. Newton's method, if we compute $\nabla^2 r_j$).

Gauss-Newton method

- Line search with Wolfe conditions and a modification of Newton's method: instead of generating the search direction \mathbf{p}_k by solving the Newton eq. $\nabla^2 f(\mathbf{x}_k)\mathbf{p} = -\nabla f(\mathbf{x}_k)$, ignore the second-order term in $\nabla^2 f$ (i.e., approximate $\nabla^2 f_k \approx \mathbf{J}_k^T \mathbf{J}_k$) and solve $\mathbf{J}_k^T \mathbf{J}_k \mathbf{p}_k^{GN} = -\mathbf{J}_k^T \mathbf{r}_k$.
- Equivalent to approximating $\mathbf{r}(\mathbf{x})$ by a linear model $\mathbf{r}(\mathbf{x} + \mathbf{p}) \approx \mathbf{r}(\mathbf{x}) + \mathbf{J}(\mathbf{x})\mathbf{p}$ (*linearization*) and so $f(\mathbf{x})$ by a quadratic model with Hessian $\mathbf{J}(\mathbf{x})^T \mathbf{J}(\mathbf{x})$, then solving the linear LSQ problem $\min_{\mathbf{p}} \frac{1}{2} \|\mathbf{J}_k \mathbf{p} + \mathbf{r}_k\|_2^2$.
- If \mathbf{J}_k has full rank and $\nabla f_k = \mathbf{J}_k^T \mathbf{r}_k \neq \mathbf{0}$ then \mathbf{p}_k^{GN} is a descent direction. (Pf.: evaluate $(\mathbf{p}_k^{GN})^T \nabla f_k < 0$.)
- Saves us the trouble of computing the individual Hessians $\nabla^2 \mathbf{r}_j$.
- *Global convergence* if Wolfe conditions + $\|\mathbf{J}(\mathbf{x})\mathbf{z}\|_2 \geq \gamma \|\mathbf{z}\|_2$ in the region of interest + technical condition (th. 10.1) (Pf.: $\cos \theta_k$ is bounded away from 0 + Zoutendijk's th.) ($\|\mathbf{J}(\mathbf{x})\mathbf{z}\|_2 \Leftrightarrow$ singular values of \mathbf{J} are away from 0 $\Leftrightarrow \mathbf{J}^T \mathbf{J}$ is well conditioned (see ch. 3).)
The theorem doesn't hold if $\mathbf{J}(\mathbf{x}_k)$ is rank-deficient for some k . This occurs when the normal equations are underdetermined (infinite number of solutions for \mathbf{p}_k^{GN}).
- *Rate of convergence* depends on how much the term $\mathbf{J}^T \mathbf{J}$ dominates the second-order term in the Hessian at the solution \mathbf{x}^* ; it is linear but rapid in the small-residual case:
eq. (10.30): $\|\mathbf{x}_k + \mathbf{p}_k^{GN} - \mathbf{x}^*\| \lesssim \|(\mathbf{J}^T(\mathbf{x}^*)\mathbf{J}(\mathbf{x}^*))^{-1} \nabla^2 f(\mathbf{x}^*) - \mathbf{I}\| \|\mathbf{x}_k - \mathbf{x}^*\| + \mathcal{O}(\|\mathbf{x}_k - \mathbf{x}^*\|^2)$.
- *Inexact Gauss-Newton method*: solve the linsys approximately, e.g. with CG.

Levenberg-Marquardt method

- 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 $\mathbf{J}_k^T \mathbf{J}_k$:

$$m_k(\mathbf{p}) = \frac{1}{2} \|\mathbf{r}_k\|^2 + \mathbf{p}^T \mathbf{J}_k^T \mathbf{r}_k + \frac{1}{2} \mathbf{p}^T \mathbf{J}_k^T \mathbf{J}_k \mathbf{p} = \frac{1}{2} \|\mathbf{J}_k \mathbf{p} + \mathbf{r}_k\|_2^2$$

$$\Rightarrow \min_{\mathbf{p}} \frac{1}{2} \|\mathbf{J}_k \mathbf{p} + \mathbf{r}_k\|_2^2 \text{ s.t. } \|\mathbf{p}\| \leq \Delta_k.$$

- A rank-deficient Jacobian is no problem because the step length is bounded by Δ_k .
- Characterization of the solution of the trust-region subproblem (lemma 10.2, direct consequence of th. 4.1 in ch. 4): \mathbf{p}^{LM} is a solution of the trust-region problem $\min_{\|\mathbf{p}\| \leq \Delta} \|\mathbf{J}\mathbf{p} + \mathbf{r}\|_2^2$ iff \mathbf{p}^{LM} is feasible and $\exists \lambda \geq 0$ such that:

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

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

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

- *Global convergence* under certain assumptions.
- *Rate of convergence*: like Gauss-Newton, since near a solution the trust region eventually becomes inactive and the algorithm takes Gauss-Newton steps.

Large-residual problems

If the residuals $r_j(\mathbf{x}^*)$ near the solution \mathbf{x}^* are large, both Gauss-Newton and Levenberg-Marquardt converge slowly, since $\mathbf{J}^T \mathbf{J}$ is a bad model of the Hessian. Options:

- Use a Newton or quasi-Newton method.
- Use a hybrid method, e.g. start with GN/LM then switch to (quasi-)Newton, or apply a quasi-Newton approximation \mathbf{S}_k to the second-order part of the Hessian $\sum_j r_j(\mathbf{x}) \nabla^2 r_j(\mathbf{x})$ and combine with GN: $\mathbf{B}_k = \mathbf{J}_k^T \mathbf{J}_k + \mathbf{S}_k$.

Review: nonlinear least-squares problems

$f(\mathbf{x}) = \frac{1}{2} \sum_{j=1}^m r_j^2(\mathbf{x})$, $m \geq n$, residual $r_j: \mathbb{R}^n \rightarrow \mathbb{R}$ is the error at datum j for a model with parameters \mathbf{x} .

- Simplified form for gradient and Hessian:

- $f(\mathbf{x}) = \frac{1}{2} \|\mathbf{r}(\mathbf{x})\|_2^2$ with $\mathbf{r}(\mathbf{x}) = (r_j(\mathbf{x}))_j$
- $\nabla f(\mathbf{x}) = \mathbf{J}(\mathbf{x})^T \mathbf{r}(\mathbf{x})$ with Jacobian $\mathbf{J}(\mathbf{x}) = \left(\frac{\partial r_j}{\partial \mathbf{x}_i} \right)_{ji}$
- $\nabla^2 f(\mathbf{x}) = \underbrace{\mathbf{J}(\mathbf{x})^T \mathbf{J}(\mathbf{x})}_{\text{use this as approximate Hessian}} + \sum_j r_j(\mathbf{x}) \nabla^2 r_j(\mathbf{x})$.

- *Linear LSQ*: \mathbf{r}_j linear, \mathbf{J} constant, minimizer \mathbf{x}^* satisfies (calling $\mathbf{r} = \mathbf{r}(\mathbf{0})$) the *normal eqs.* $\underbrace{\mathbf{J}^T \mathbf{J}}_{\text{pd or psd}} \mathbf{x}^* = -\mathbf{J} \mathbf{r}$.
- Nonlinear LSQ: GN, LM methods.
- *Gauss-Newton method*:

- Approximate Hessian $\nabla^2 f_k \approx \mathbf{J}_k^T \mathbf{J}_k$, solve for the search direction $\mathbf{J}_k^T \mathbf{J}_k \mathbf{p}_k^{\text{GN}} = -\mathbf{J}_k^T \mathbf{r}_k$, inexact line search with Wolfe conditions.
- Equivalent to linearizing $\mathbf{r}(\mathbf{x} + \mathbf{p}) \approx \mathbf{r}(\mathbf{x}) + \mathbf{J}(\mathbf{x}) \mathbf{p}$.
- Problems if \mathbf{J}_k is rank-defective.

- *Levenberg-Marquardt method*:

- Like GN but with trust region instead of line search: $\min_{\|\mathbf{p}\| \leq \Delta_k} \|\mathbf{J}_k \mathbf{p} + \mathbf{r}_k\|_2^2$.
- No problem if \mathbf{J}_n is rank defective.
- One way to solve the trust-region subproblem approximately: try large $\lambda \geq 0$, solve $(\mathbf{J}_k^T \mathbf{J}_k + \lambda \mathbf{I}) \mathbf{p}_k^{\text{LM}} = -\mathbf{J}_k^T \mathbf{r}_k$, accept \mathbf{p}_k^{LM} if sufficient decrease in f , otherwise try a smaller λ .

- *Global convergence* under certain assumptions.

- *Rate of convergence*: linear but fast if $\mathbf{J}^T(\mathbf{x}^*) \mathbf{J}(\mathbf{x}^*) \approx \nabla^2 f(\mathbf{x}^*)$, which occurs with small residuals ($r_j(\mathbf{x}) \approx 0$) or quasilinear residuals ($\nabla^2 r_j(\mathbf{x}) \approx \mathbf{0}$). Otherwise GN/LM are slow; try other methods instead (quasi-Newton, Newton, etc.) or hybrids that combine the advantages of GN/LM and (quasi-)Newton.

Method of scoring (Gauss-Newton for maximum likelihood)

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

$$\begin{aligned} \text{Gradient } \nabla L &= \frac{1}{p} \nabla p \\ \text{Hessian } \nabla^2 L &= -\frac{1}{p^2} \nabla p \nabla p^T + \frac{1}{p} \nabla^2 p = -\nabla \log p \nabla \log p^T + \frac{1}{p} \nabla^2 p. \end{aligned}$$

Taking expectations wrt the model $p(\mathbf{t}; \mathbf{x})$ we have:

$$\mathbb{E} \{-\nabla^2 L\} = \mathbb{E} \{\nabla \log p \nabla \log p^T\} + \mathbb{E} \left\{ \frac{1}{p} \nabla^2 p \right\} = \text{cov} \{\nabla \log p\}$$

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

In statistical parlance:

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

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

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

Advantages:

- Good approximation to 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 (covariance matrix), so descent directions.
- Particularly simple expressions for the exponential family: $p(\mathbf{t}; \mathbf{x}) = \frac{f(\mathbf{t})}{g(\mathbf{x})} e^{\mathbf{h}(\mathbf{t})^T \Phi(\mathbf{x})}$ with sufficient statistic $\mathbf{h}(\mathbf{t})$ and partition function $g(\mathbf{x}) = \int f(\mathbf{t}) e^{\mathbf{h}(\mathbf{t})^T \Phi(\mathbf{x})} d\mathbf{t}$.

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

$$\text{log-likelihood gradient } \frac{1}{N} \sum_{i=1}^N \nabla \log p(\mathbf{t}_i; \mathbf{x}) = \nabla \Phi(\mathbf{x}) (\mathbb{E}_{\text{data}} \{\mathbf{h}\} - \mathbb{E}_{\text{model}} \{\mathbf{h}\}).$$

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

Missing-data problem Consider \mathbf{t} are observed and \mathbf{z} are missing, so $p(\mathbf{t}; \mathbf{x}) = \int p(\mathbf{t}|\mathbf{z}; \mathbf{x})p(\mathbf{z}; \mathbf{x}) d\mathbf{z}$ (e.g. \mathbf{z} = label of mixture component). We have:

$$\begin{aligned}\nabla \log p(\mathbf{t}; \mathbf{x}) &= \frac{1}{p(\mathbf{t}; \mathbf{x})} \int (\nabla p(\mathbf{t}|\mathbf{z}; \mathbf{x})p(\mathbf{z}; \mathbf{x}) + p(\mathbf{t}|\mathbf{z}; \mathbf{x})\nabla p(\mathbf{z}; \mathbf{x})) d\mathbf{z} \\ &= \frac{1}{p(\mathbf{t}; \mathbf{x})} \int p(\mathbf{z}; \mathbf{x})p(\mathbf{t}|\mathbf{z}; \mathbf{x})(\nabla \log p(\mathbf{t}|\mathbf{z}; \mathbf{x}) + \nabla \log p(\mathbf{z}; \mathbf{x})) d\mathbf{z} = E_{\mathbf{z}|\mathbf{t}} \{ \nabla \log p(\mathbf{t}, \mathbf{z}; \mathbf{x}) \} \\ &= \text{posterior expectation of the complete-data log-likelihood gradient.}\end{aligned}$$

$$\begin{aligned}\nabla^2 \log p(\mathbf{t}; \mathbf{x}) &= \nabla \int p(\mathbf{z}|\mathbf{t}; \mathbf{x}) \nabla \log p(\mathbf{t}, \mathbf{z}; \mathbf{x}) d\mathbf{z} = \\ &= E_{\mathbf{z}|\mathbf{t}} \{ \nabla^2 \log p(\mathbf{t}, \mathbf{z}; \mathbf{x}) \} + \int \nabla p(\mathbf{z}|\mathbf{t}; \mathbf{x}) \nabla \log p(\mathbf{t}, \mathbf{z}; \mathbf{x})^T d\mathbf{z}.\end{aligned}$$

Noting that

$$\begin{aligned}\nabla p(\mathbf{z}|\mathbf{t}; \mathbf{x}) &= \nabla \left(\frac{p(\mathbf{t}, \mathbf{z}; \mathbf{x})}{p(\mathbf{t}; \mathbf{x})} \right) = \frac{1}{p(\mathbf{t}; \mathbf{x})} (\nabla p(\mathbf{t}, \mathbf{z}; \mathbf{x}) - p(\mathbf{z}|\mathbf{t}; \mathbf{x}) \nabla \log p(\mathbf{t}; \mathbf{x})) = \\ &= p(\mathbf{z}|\mathbf{t}; \mathbf{x}) (\nabla \log p(\mathbf{t}, \mathbf{z}; \mathbf{x}) - \nabla \log p(\mathbf{t}; \mathbf{x})),\end{aligned}$$

then the second term is:

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

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

Finally:

$$\begin{aligned}\text{Gradient } \nabla \log p(\mathbf{t}; \mathbf{x}) &= E_{\mathbf{z}|\mathbf{t}} \{ \nabla \log p(\mathbf{t}, \mathbf{z}; \mathbf{x}) \} \\ \text{Hessian } \nabla^2 \log p(\mathbf{t}; \mathbf{x}) &= E_{\mathbf{z}|\mathbf{t}} \{ \nabla^2 \log p(\mathbf{t}, \mathbf{z}; \mathbf{x}) \} + \text{cov}_{\mathbf{z}|\mathbf{t}} \{ \nabla \log p(\mathbf{t}, \mathbf{z}; \mathbf{x}) \} \\ &= \left\{ \begin{array}{c} \text{posterior expectation} \\ \text{of complete-data} \\ \text{log-likelihood Hessian} \end{array} \right\} + \left\{ \begin{array}{c} \text{posterior covariance} \\ \text{of complete-data} \\ \text{log-likelihood gradient} \end{array} \right\}.\end{aligned}$$

Again, ignoring the second-order term we obtain a cheap, pd approximation to the Hessian (Gauss-Newton method)—but for minimizing the likelihood, not for maximizing it!

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

$$\nabla^2 \log p(\mathbf{t}; \mathbf{z}) \simeq -\nabla \log p(\mathbf{t}; \mathbf{x}) \nabla \log p(\mathbf{t}; \mathbf{x})^T.$$

Relation with the EM (expectation-maximization) algorithm

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

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

11 Nonlinear equations

- Problem: *find roots* of the n equations $\mathbf{r}(\mathbf{x}) = \mathbf{0}$ in n unknowns \mathbf{x} where $\mathbf{r}(\mathbf{x}) : \mathbb{R}^n \rightarrow \mathbb{R}^n$. Ex. in p. 271
0, 1, finitely many, or infinitely many roots.

- Many similarities with optimization: Newton's method, line search, trust region...

- Differences:

- In optimization, the local optima can be ranked by objective value.

In root finding, all roots are equally good.

- For quadratic convergence, we need derivative of order $\begin{cases} \text{optimization:} & 2 \\ \text{root-finding:} & 1. \end{cases}$
- Quasi-Newton methods are less useful in root-finding.

- Assume the $n \times n$ Jacobian $\mathbf{J}(\mathbf{x}) = (\frac{\partial r_i}{\partial x_j})_{ij}$ exists and is continuous in the region of interest.

- *Degenerate root*: \mathbf{x}^* with $\mathbf{r}(\mathbf{x}^*) = \mathbf{0}$ and $\mathbf{J}(\mathbf{x}^*)$ singular.



Newton's method

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

Algorithm 11.1: given \mathbf{x}_0 :

```
for  $k = 0, 1, 2 \dots$ 
    solve  $\mathbf{J}_k \mathbf{p}_k = -\mathbf{r}_k$ 
     $\mathbf{x}_{k+1} \leftarrow \mathbf{x}_k + \mathbf{p}_k$ 
end
```

- Newton's method for optimizing an objective function f is the same as applying this algorithm to $\mathbf{r}(\mathbf{x}) = \nabla f(\mathbf{x})$.
- Convergence rate for nondegenerate roots (th. 11.2) $\begin{cases} \text{Jacobian continuous:} & \text{superlinear;} \\ \text{Jacobian Lipschitz cont.:} & \text{quadratic.} \end{cases}$
- Problems:
 - Degenerate roots, e.g. $r(x) = x^2$ produces $x_k = 2^{-k}x_0$ which converge linearly.
 - Not globally convergent: away from a root the algorithm can diverge or cycle; it is not even defined if $\mathbf{J}(\mathbf{x}_k)$ is singular.
 - Expensive to compute \mathbf{J} and solve the system exactly for large n .

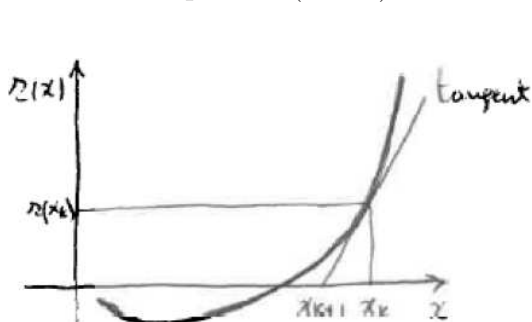
Inexact Newton method

- Exit the linear solver of $\mathbf{J}_k \mathbf{p}_k = -\mathbf{r}_k$ when $\|\mathbf{r}_k + \mathbf{J}_k \mathbf{p}_k\| \leq \eta_k \|\mathbf{r}_k\|$ for $\eta_k \in [0, \eta]$ with constant $\eta \in (0, 1)$. (η_k) = *forcing sequence* (as for Newton's method in optimization). We can't use linear conjugate gradients here because \mathbf{J}_k is not always pd; but there are other linear solvers (also based on Krylov subspaces, i.e., iterative multiplication by \mathbf{J}_k).

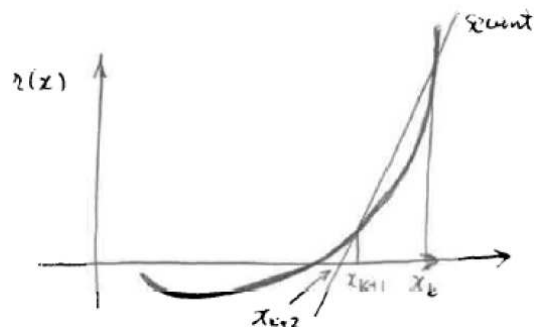
- Convergence rate to a non-degenerate root (th. 11.3) $\begin{cases} \text{linear, if } \eta \text{ sufficiently small} \\ \text{superlinear, if } \eta_k \rightarrow 0 \\ \text{quadratic, if } \eta_k = \mathcal{O}(\|\mathbf{r}_k\|). \end{cases}$

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

- Constructs an approximation to the Jacobian over iterations.
- Write $\mathbf{s}_k = \mathbf{x}_{k+1} - \mathbf{x}_k$, $\mathbf{y}_k = \mathbf{r}_{k+1} - \mathbf{r}_k$: $\mathbf{y}_k = \mathbf{J}_k \mathbf{s}_k + \mathcal{O}(\|\mathbf{s}_k\|^2)$ (Taylor's th.).
- We require the updated Jacobian approximation \mathbf{B}_{k+1} to satisfy the secant equation $\mathbf{y}_k = \mathbf{B}_{k+1} \mathbf{s}_k$ and to minimize $\|\mathbf{B} - \mathbf{B}_k\|_2$, i.e., the smallest possible update that satisfies the secant eq.: $\mathbf{B}_{k+1} = \mathbf{B}_k + \frac{(\mathbf{y}_k - \mathbf{B}_k \mathbf{s}_k) \mathbf{s}_k^T}{\mathbf{s}_k^T \mathbf{s}_k}$ (from lemma 11.4).
- Convergence rate: superlinear if the initial point \mathbf{x}_0 and Jacobian \mathbf{B}_0 are close to the root \mathbf{x}^* and its Jacobian $\mathbf{J}(\mathbf{x}^*)$, resp.; the latter condition can be crucial, but is difficult to guarantee in practice.
- Limited-memory versions exist for large n .
- For a scalar equation ($n = 1$):



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



Secant method: $x_{k+1} = x_k - \frac{r(x_k)}{B_k}$ 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 function*: a function $f : \mathbb{R}^n \rightarrow \mathbb{R}$ that indicates how close \mathbf{x} is to a root, so that by decreasing $f(\mathbf{x})$ we approach a root. In optimization, f is the objective function. In root-finding, there is no unique (and fully satisfactory) way to define a merit function. The most widely used is $f(\mathbf{x}) = \frac{1}{2} \|\mathbf{r}(\mathbf{x})\|_2^2$.
- Problem: each root ($\mathbf{r}(\mathbf{x}) = \mathbf{0}$) is a local minimizer of f but not vice versa, so local minima that are not roots can attract the algorithm.
If a local minimizer \mathbf{x} is not a root then $\mathbf{J}(\mathbf{x})$ is singular (Pf: $\nabla f(\mathbf{x}) = \mathbf{J}(\mathbf{x})^T \mathbf{r}(\mathbf{x}) = \mathbf{0} \xrightarrow{\mathbf{r}(\mathbf{x}) \neq \mathbf{0}} \mathbf{J}(\mathbf{x})$ singular.)
- *Line search methods*: $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{p}_k$ with step length α_k along direction \mathbf{p}_k .
 - We want descent directions for f ($\mathbf{p}_k^T \nabla f(\mathbf{x}_k) < 0$); step length chosen as in ch. 3.
 - Zoutendijk's th.: descent directions + Wolfe conditions + Lipschitz continuous $\mathbf{J} \Rightarrow \sum_{k \geq 0} \cos^2 \theta_k \|\mathbf{J}_k^T \mathbf{r}_k\|^2 < \infty$.
 - So if $\cos \theta_k \geq \delta$ for constant $\delta \in (0, 1)$ and all k sufficiently large $\Rightarrow \nabla f_k = \mathbf{J}_k^T \mathbf{r}_k \rightarrow 0$; and if $\|\mathbf{J}(\mathbf{x})^{-1}\|$ is bounded $\Rightarrow \mathbf{r}_k \rightarrow 0$.

- If well defined, the Newton step is a descent direction for f for $\mathbf{r}_k \neq \mathbf{0}$ (Pf.: $\mathbf{p}_k^T \nabla f_k = -\mathbf{p}_k^T \mathbf{J}_k^T \mathbf{r}_k = -\|\mathbf{r}_k\|^2 < 0$). But $\cos \theta_k = \frac{-\mathbf{p}_k^T \nabla f_k}{\|\mathbf{p}_k\| \|\nabla f_k\|} = \frac{\|\mathbf{r}_k\|^2}{\|\mathbf{J}_k^{-1} \mathbf{r}_k\| \|\mathbf{J}_k^T \mathbf{r}_k\|} \geq \frac{1}{\|\mathbf{J}_k^T\| \|\mathbf{J}_k^{-1}\|} = \frac{1}{\kappa(\mathbf{J}_k)}$, so a large condition number causes poor performance (search direction almost $\perp \nabla f_k$).
- One modification of Newton's direction is $(\mathbf{J}_k^T \mathbf{J}_k + \tau_k \mathbf{I}) \mathbf{p}_k = -\mathbf{J}_k^T \mathbf{r}_k$; a large enough τ_k ensures $\cos \theta_k$ is bounded away from 0 because $\tau_k \rightarrow \infty \Rightarrow \mathbf{p}_k \propto -\mathbf{J}_k^T \mathbf{r}_k$.
- Inexact Newton steps do not compromise global convergence: if at each step $\|\mathbf{r}_k + \mathbf{J}_k \mathbf{p}_k\| \leq \eta_k \|\mathbf{r}_k\|$ for $\eta_k \in [0, \eta]$ and $\eta \in [0, 1)$ then $\cos \theta_k \geq \frac{1-\eta}{2\kappa(\mathbf{J}_k)}$.
- *Trust region methods*: $\min m_k(\mathbf{p}) = f_k + \nabla f_k^T \mathbf{p} + \frac{1}{2} \mathbf{p}^T \mathbf{B}_k \mathbf{p}$ s.t. $\|\mathbf{p}\| \leq \Delta_k$.
 - Algorithm 4.1 from ch. 4 applied to $f(\mathbf{x}) = \frac{1}{2} \|\mathbf{r}(\mathbf{x})\|_2^2$ using $\mathbf{B}_k = \mathbf{J}_k^T \mathbf{J}_k$ as approximate Hessian in the model m_k , i.e., linearize $\mathbf{r}(\mathbf{p}) \approx \mathbf{r}_k + \mathbf{J}_k \mathbf{p}$.
 - The exact solution has the form $\mathbf{p}_k = (-\mathbf{J}_k^T \mathbf{J}_k + \lambda_k \mathbf{I})^{-1} \mathbf{J}_k^T \mathbf{r}_k$ 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 non-degenerate roots) under mild conditions.
 - Quadratic convergence rate if the trust region subproblem is solved exactly for all k sufficiently large.

Continuation/homotopy methods

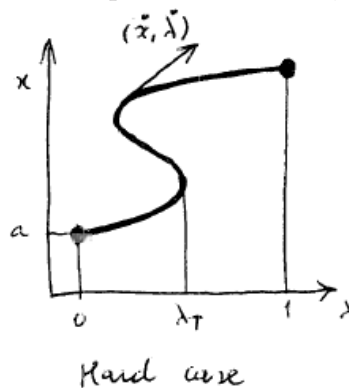
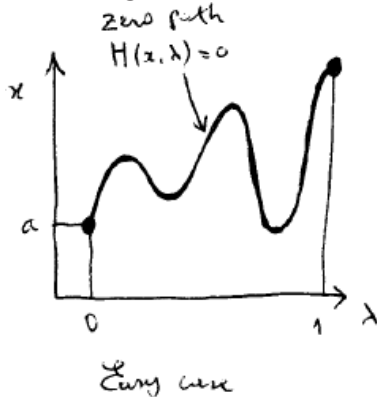
- Problem of Newton-based methods: unless \mathbf{J} is nonsingular in the region of interest, they may converge to a local minimizer of the merit function rather than a root.
- *Continuation methods*: instead of dealing with the original problem $\mathbf{r}(\mathbf{x}) = \mathbf{0}$ directly, establish 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 we move from the easy to the original problem.
- *Homotopy map*: $\mathbf{H}(\mathbf{x}, \lambda) = \underbrace{\lambda \mathbf{r}(\mathbf{x})}_{\substack{\lambda=1 \\ \text{original} \\ \text{problem}}} + (1-\lambda) \underbrace{(\mathbf{x} - \mathbf{a})}_{\substack{\lambda=0 \\ \text{easy problem with} \\ \text{solution } \mathbf{x} = \mathbf{a}}}$ where $\mathbf{a} \in \mathbb{R}^n$ fixed and $\lambda \in \mathbb{R}$.

If $\frac{\partial}{\partial \mathbf{x}} \mathbf{H}(\mathbf{x}, \lambda)$ is nonsingular then $\mathbf{H}(\mathbf{x}, \lambda) = \mathbf{0}$ defines a continuous curve $\mathbf{x}(\lambda)$, the *zero path*.

By the implicit function theorem (th. A.2); ex.: $x^2 - y + 1 = 0$, $\mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{y} + \mathbf{c} = \mathbf{0} \Rightarrow \mathbf{y} = \mathbf{g}(\mathbf{x})$ or $\mathbf{x} = \mathbf{g}(\mathbf{y})$ locally.

We want to follow the path numerically. $\mathbf{x} = \mathbf{a}$ plays the role of initial iterate.

- Naive approach: start from $\lambda = 0$, $\mathbf{x} = \mathbf{a}$; gradually increase λ from 0 to 1 and solve $\mathbf{H}(\mathbf{x}, \lambda) = \mathbf{0}$ using as initial \mathbf{x} the one from the previous λ value; stop after solving for $\lambda = 1$.



But at the turning point λ_T , if we increase λ we lose track of the root \mathbf{x} . To follow the zero path smoothly we need to allow λ to decrease and even to roam outside $[0, 1]$. So we follow the path along the arc-length s (instead of λ).

- Arc-length parametrization of the zero path: $(\mathbf{x}(s), \lambda(s))$ where s = arc length measured from $(\mathbf{a}, 0)$ at $s = 0$. Since $\mathbf{H}(\mathbf{x}(s), \lambda(s)) = \mathbf{0} \forall s \geq 0$, its total derivative wrt s is also $\mathbf{0}$:

$$\frac{d\mathbf{H}}{ds} = \frac{\partial}{\partial \mathbf{x}} \mathbf{H}(\mathbf{x}, \lambda) \dot{\mathbf{x}} + \frac{\partial}{\partial \lambda} \mathbf{H}(\mathbf{x}, \lambda) \dot{\lambda} = \mathbf{0} \quad (n \text{ equations}),$$

where $(\dot{\mathbf{x}}, \dot{\lambda}) = (\frac{d\mathbf{x}}{ds}, \frac{d\lambda}{ds})$ is the tangent vector to the zero path.

- To calculate the tangent vector at a point (\mathbf{x}, λ) notice that:
 1. $(\dot{\mathbf{x}}, \dot{\lambda})$ lies in the null space of the $n \times (n+1)$ matrix $(\frac{\partial \mathbf{H}}{\partial \mathbf{x}}, \frac{\partial \mathbf{H}}{\partial \lambda})$; the null space is 1D if this matrix is full rank. This can be obtained from the QR factorization of $(\frac{\partial \mathbf{H}}{\partial \mathbf{x}}, \frac{\partial \mathbf{H}}{\partial \lambda})$.
 2. Its length is $1 = \|\dot{\mathbf{x}}(s)\|^2 + |\dot{\lambda}(s)|^2 \forall s \geq 0$ (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 correct tangent vector makes an angle of less than $\pi/2$ with the previous tangent.

Procedure 11.7.



- Following the path can now be done by different methods:
 - By solving an initial-value first-order ODE: $\frac{d\mathbf{H}}{ds} = \mathbf{0}$ for $s \geq 0$ with $(\mathbf{x}(0), \lambda(0)) = (\mathbf{a}, 0)$; terminating at an s for which $\lambda(s) = 1$.
 - By a predictor-corrector method: at each iteration k :
 1. Predictor step of length ϵ along the tangent vector: $(\mathbf{x}^P, \lambda^P) = (\mathbf{x}_k, \lambda_k) + \epsilon(\dot{\mathbf{x}}_k, \dot{\lambda}_k)$. This doesn't lie on the zero path, but is close to it.
 2. Corrector step: bring $(\mathbf{x}^P, \lambda^P)$ back to the zero path with a few Newton iterations.

(*What happens if running Newton's method at $\lambda = 1$ from some initial \mathbf{x} ?*)
- The tangent vector is well defined if the matrix $(\frac{\partial \mathbf{H}}{\partial \mathbf{x}}, \frac{\partial \mathbf{H}}{\partial \lambda})$ has full rank. This is guaranteed under certain assumptions:

fig. 11.5

Th. 11.9: \mathbf{r} twice cont. differentiable \Rightarrow for almost all $\mathbf{a} \in \mathbb{R}^n$ there is a zero path from $(\mathbf{a}, 0)$ along which the matrix $(\frac{\partial \mathbf{H}}{\partial \mathbf{x}}, \frac{\partial \mathbf{H}}{\partial \lambda})$ has full rank. If this path is bounded for $\lambda \in [0, 1)$ then it has an accumulation point $(\bar{\mathbf{x}}, 1)$ with $\mathbf{r}(\bar{\mathbf{x}}) = \mathbf{0}$. If $\mathbf{J}(\bar{\mathbf{x}})$ is non-singular, the zero path between $(\mathbf{a}, 0)$ and $(\bar{\mathbf{x}}, 1)$ has finite length.

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

ex. 11.3

- 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.
- Related algorithms:
 - For constrained optimization: quadratic-penalty, log-barrier, interior-point.
 - For (heuristic) global optimization: deterministic annealing.

Review: nonlinear equations

Problem: *find roots* of n equations $\mathbf{r}(\mathbf{x}) = \mathbf{0}$ in n unknowns.

- *Degenerate roots* (singular Jacobian) cause troubles (e.g. slower convergence).
- Similar to optimization but harder: most methods only converge if starting sufficiently near a root.
 - *Newton's method*: linsys given by Jacobian of \mathbf{r} (first deriv. only), quadratic convergence. Inexact steps may be used.
 - *Broyden's method*: quasi-Newton method, approximates Jacobian through differences in \mathbf{r} and \mathbf{x} , superlinear convergence. Limited-memory versions exist.
- *Merit function* $f(\mathbf{x}) = \frac{1}{2} \|\mathbf{r}(\mathbf{x})\|_2^2$ turns root-finding into minimization (so we are guided towards minima), but contains minima that are not roots ($\nabla f(\mathbf{x}) = \mathbf{J}(\mathbf{x})^T \mathbf{r}(\mathbf{x}) = \mathbf{0}$ but $\mathbf{r}(\mathbf{x}) \neq \mathbf{0}$, i.e., singular Jacobian).
 - Line search and trust region strategies may be used. Results obtained in earlier chapters (e.g. for convergence) carry over appropriately.
- *Continuation/homotopy methods* construct a family of problems parameterized over $\lambda \in [0, 1]$ so that $\lambda = 0$ is easy to solve (e.g. $\mathbf{x} - \mathbf{a} = \mathbf{0}$) and $\lambda = 1$ is the original problem ($\mathbf{r}(\mathbf{x}) = \mathbf{0}$).
 - This implicitly defines a path in (\mathbf{x}, λ) that most times is continuous, bounded and goes from $(\mathbf{a}, 0)$ to $(\mathbf{x}^*, 1)$ where \mathbf{x}^* is a root of \mathbf{r} .
 - We follow the path numerically (solving an ODE, or solving a sequence of root-finding problems).
 - More robust than other methods, but higher computational cost.

Summary of methods

Assumptions:

- Typical behavior.

- Evaluation costs:

f 's type	$f(\mathbf{x})$	$\nabla f(\mathbf{x})$	$\nabla^2 f(\mathbf{x})$
quadratic	$\mathcal{O}(n^2)$	$\mathcal{O}(n^2)$	$\mathcal{O}(1)$
other	$\mathcal{O}(n)$	$\mathcal{O}(n)$	$\mathcal{O}(n^2)$

- Appropriate conditions for: the line search (e.g. Wolfe) or trust region strategy; the functions, etc. (e.g. Lipschitz continuity, solution with pd Hessian).

Method		Global convergence	Convergence rate	Cost of one iteration		Derivative order	Quadratic f cost in time
				space	time		
	Steepest descent	Y	linear	$\mathcal{O}(n)$	$\mathcal{O}(n)$	1	∞
Newton	Pure	N	quadratic	$\mathcal{O}(n^2)$	$\mathcal{O}(n^3)$	2	$\mathcal{O}(n^3)$
	Modified-Hessian	Y	quadratic	$\mathcal{O}(n^2)$	$\mathcal{O}(n^3)$	2	$\mathcal{O}(n^3)$
Conjugate-gradient	Fletcher-Reeves	Y	linear	$\mathcal{O}(n)$	$\mathcal{O}(n)$	1	$\mathcal{O}(n^3)$
	Polak-Ribière	N	linear	$\mathcal{O}(n)$	$\mathcal{O}(n)$	1	$\mathcal{O}(n^3)$
Quasi-Newton	DFP, BFGS, SR1	N	superlinear	$\mathcal{O}(n^2)$	$\mathcal{O}(n^2)$	1	$\mathcal{O}(n^3)$
Large-scale	Newton-CG	Y	linear to quadratic dep. on forcing seq.	$\mathcal{O}(n^2)$	$\mathcal{O}(n^2)$ – $\mathcal{O}(n^3)$	2	at least $\mathcal{O}(n^3)$ but finite
	L-BFGS	N	linear	$\mathcal{O}(nm)$	$\mathcal{O}(nm)$	1	∞
Derivative-free	Model-based	N	\leq linear	$\mathcal{O}(n^3)$	$\mathcal{O}(n^4)$	0	$\mathcal{O}(n^6)$
	Coordinate descent	N	\leq linear	$\mathcal{O}(1)$	$\mathcal{O}(n)$	0	∞
	Nelder-Mead	N	\leq linear	$\mathcal{O}(n^2)$	$\mathcal{O}(n)$	0	∞
	Conjugate directions	N	\leq linear	$\mathcal{O}(n^2)$	$\mathcal{O}(n^2)$	0	$\mathcal{O}(n^4)$
Least-squares	Gauss-Newton	N	linear	$\mathcal{O}(n^2)$	$\mathcal{O}(n^3)$	1	$\mathcal{O}(n^3)$
	Levenberg-Marquardt	Y	linear	$\mathcal{O}(n^2)$	$\mathcal{O}(n^3)$	1	$\mathcal{O}(n^3)$

In terms of convergence rate alone, we can rank the methods as:

derivative-free < steepest descent < conjugate-gradient < L-BFGS < Least-squares < quasi-Newton < Newton.

12 Theory of constrained optimization

Optimization problem:

$$\min_{\mathbf{x} \in \mathbb{R}^n} \underbrace{f(\mathbf{x})}_{\text{objective function}} \quad \text{s.t.} \quad \left\{ \begin{array}{ll} c_i(\mathbf{x}) = 0, & i \in \mathcal{E} \\ c_i(\mathbf{x}) \geq 0, & i \in \mathcal{I} \end{array} \right\} \begin{array}{l} \text{finite sets} \\ \text{of indices} \end{array}$$

or equivalently $\min_{\mathbf{x} \in \Omega} f(\mathbf{x})$ where:

- $\Omega = \{\mathbf{x} \in \mathbb{R}^n: \underbrace{c_i(\mathbf{x}) = 0, c \in \mathcal{E}}_{\text{equality constraints}}; \underbrace{c_i(\mathbf{x}) \geq 0, i \in \mathcal{I}}_{\text{inequality constraints}}\}$ the *feasible set*.
- The objective function f and the constraints c_i are all smooth.
- \mathbf{x}^* is a *local solution* iff $\mathbf{x}^* \in \Omega$ and \exists neighborhood \mathcal{N} of \mathbf{x}^* : $f(\mathbf{x}) \geq f(\mathbf{x}^*)$ for $\mathbf{x} \in \mathcal{N} \cap \Omega$.
- \mathbf{x}^* is a *strict local solution* iff $\mathbf{x}^* \in \Omega$ and \exists neigh. \mathcal{N} of \mathbf{x}^* : $f(\mathbf{x}) > f(\mathbf{x}^*)$ for $\mathbf{x} \in \mathcal{N} \cap \Omega$, $\mathbf{x} \neq \mathbf{x}^*$.
- \mathbf{x}^* is a *isolated local solution* iff $\mathbf{x}^* \in \Omega$ and \exists neigh. \mathcal{N} of \mathbf{x}^* : \mathbf{x}^* is only local minimizer in $\mathcal{N} \cap \Omega$.
- At a feasible point \mathbf{x} , the inequality constraint c_i ($i \in \mathcal{I}$) is:
 - *active* iff $c_i(\mathbf{x}) = 0$ (\mathbf{x} is on the boundary for that constraint)
 - *inactive* iff $c_i(\mathbf{x}) > 0$ (\mathbf{x} is interior point for that constraint).
- For inequality constraints, the *constraint normal* $\nabla c_i(\mathbf{x})$ points towards the feasible region and is \perp to the contour $c_i(\mathbf{x}) = 0$. For equality constraints, $\nabla c_i(\mathbf{x})$ is \perp to the contour $c_i(\mathbf{x}) = 0$.
- Mathematical characterization of solutions for unconstrained optimization (reminder):
 - *Necessary conditions*: \mathbf{x}^* local minimizer of $f \Rightarrow \nabla f(\mathbf{x}^*) = \mathbf{0}$, $\nabla^2 f(\mathbf{x}^*)$ psd.
 - *Sufficient conditions*: $\nabla f(\mathbf{x}^*) = \mathbf{0}$, $\nabla^2 f(\mathbf{x}^*)$ pd $\Rightarrow \mathbf{x}^*$ is a strong local minimizer of f .

Here we derive similar conditions for constrained optimization problems. Let's see some examples.

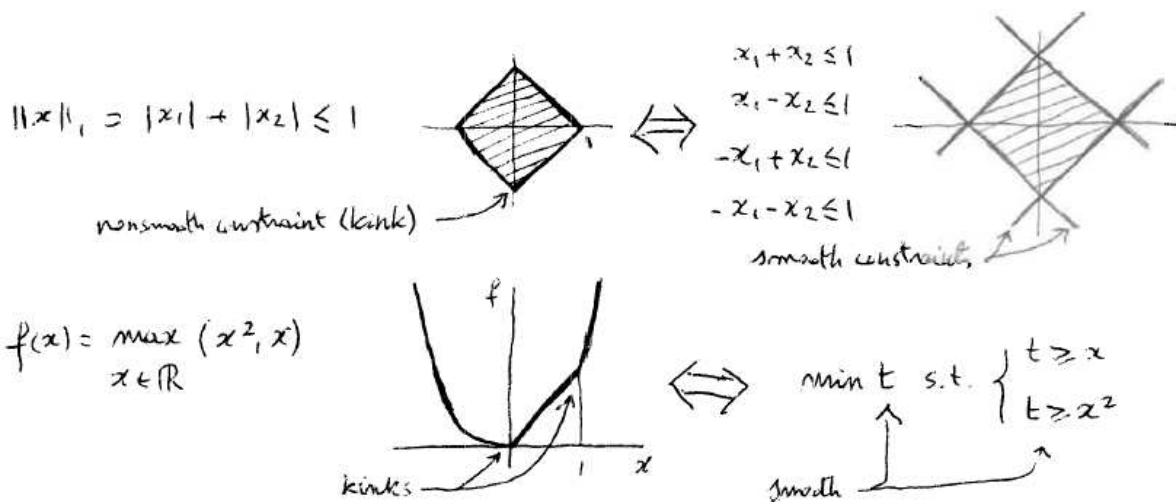
- *Local and global solutions*: constraining can decrease or increase the number of optimizers:

$$\min_{\mathbf{x} \in \mathbb{R}^n} \|\mathbf{x}\|_2^2 \quad \left\{ \begin{array}{l} \text{unconstrained: single solution} \\ \text{constrained s.t. } \|\mathbf{x}\|_2^2 \geq 1: \text{infinite solutions} \\ \text{constrained s.t. } c_1(\mathbf{x}) = 0: \text{several solutions.} \end{array} \right.$$

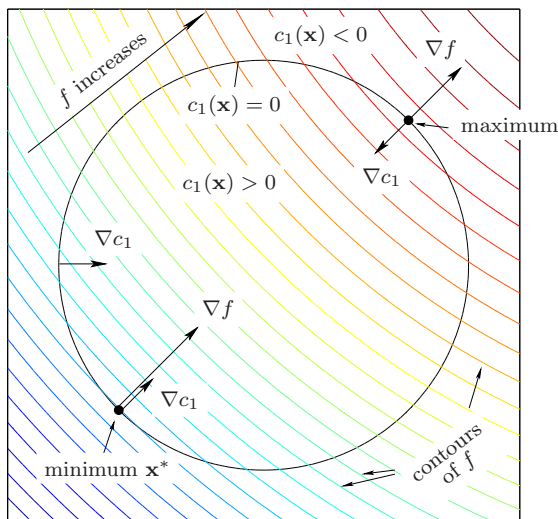


- *Smoothness* of both f and the constraints is important since then we can predict what happens near a point.

Some apparently nonsmooth problems (involving $\|\cdot\|_1$, $\|\cdot\|_\infty$) can be reformulated as smooth:



Case 1: a single equality constraint $c_1(\mathbf{x}) = 0$.



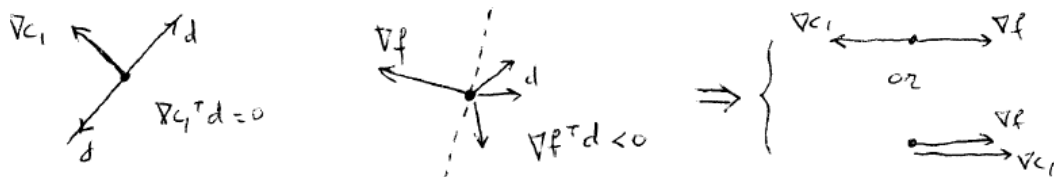
- At \mathbf{x}^* , $\nabla c_1(\mathbf{x}^*) \parallel \nabla f(\mathbf{x}^*)$, i.e., $\nabla f(\mathbf{x}^*) = \lambda_1^* \nabla c_1(\mathbf{x}^*)$ for $\lambda_1^* \in \mathbb{R}$.
- This is necessary but not sufficient since it also holds at the maximum.
- The sign of λ_1^* can be + or - (e.g. use $-c_1$ instead of c_1).
- Compare with exact line search:



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

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

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



Equivalent formulation in terms of the Lagrangian function $\mathcal{L}(\mathbf{x}, \lambda_1) = f(\mathbf{x}) - \underbrace{\lambda_1}_{\text{Lagrangian multiplier}} c_1(\mathbf{x})$: at a solution \mathbf{x}^* , $\exists \lambda_1^* \in \mathbb{R}$: $\nabla_{\mathbf{x}} \mathcal{L}(\mathbf{x}^*, \lambda_1^*) = \mathbf{0}$.

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

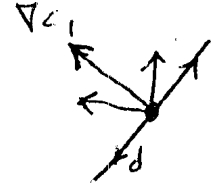
Case 2: a single inequality constraint $c_1(\mathbf{x}) \geq 0$. The solution is the same, but now the sign of λ_1^* matters: at \mathbf{x}^* , $\nabla f(\mathbf{x}^*) = \lambda_1^* \nabla c_1(\mathbf{x}^*)$ for $\lambda_1^* \geq 0$.

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

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

If no improvement is possible:

1. Interior point $c_1(\mathbf{x}) > 0$: any small enough \mathbf{d} satisfies feasibility $\Rightarrow \nabla f(\mathbf{x}) = \mathbf{0}$ (this is the unconstrained case).
2. Boundary point $c_1(\mathbf{x}) = 0$: there cannot be a direction such that $\nabla f(\mathbf{x})^T \mathbf{d} < 0$ and $\nabla c_1(\mathbf{x})^T \mathbf{d} \geq 0$
 $\Rightarrow \nabla f(\mathbf{x})$ and $\nabla c_1(\mathbf{x})$ must point in the same direction
 $\Rightarrow \nabla f(\mathbf{x}) = \lambda_1 \nabla c_1(\mathbf{x})$ for some $\lambda_1 \geq 0 \Rightarrow \overrightarrow{\nabla f} \parallel \overrightarrow{\nabla c_1}$.

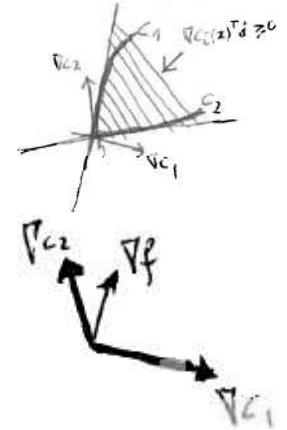


Equivalent formulation: at a solution \mathbf{x}^* , $\exists \lambda_1^* \geq 0$: $\nabla_{\mathbf{x}} \mathcal{L}(\mathbf{x}^*, \lambda_1^*) = \mathbf{0}$ and $\lambda_1^* c_1(\mathbf{x}^*) = 0$. The latter is a complementarity condition $\begin{cases} \lambda_1^* > 0 \text{ and } c_1 \text{ is active, or} \\ \lambda_1^* = 0 \text{ and } c_1 \text{ is inactive.} \end{cases}$

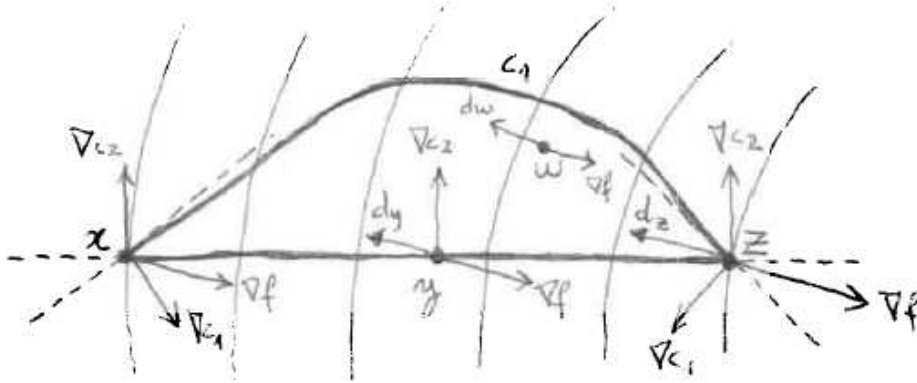
Case 3: two inequality constraints $c_1(\mathbf{x}) \geq 0$, $c_2(\mathbf{x}) \geq 0$. Consider the case of a point \mathbf{x} for which both constraints are active, i.e., $c_1(\mathbf{x}) = c_2(\mathbf{x}) = 0$. A direction \mathbf{d} is a feasible descent direction to first order if $\nabla f(\mathbf{x})^T \mathbf{d} < 0$ and $\nabla c_i(\mathbf{x})^T \mathbf{d} \geq 0$, $i \in \mathcal{I}$.

Intersection of half spaces
defined by $\nabla c_i(\mathbf{x})$

At a solution (and ignoring the case where ∇c_i are parallel), we cannot have a direction \mathbf{d} satisfying $\nabla f = \lambda_1 \nabla c_1 + \lambda_2 \nabla c_2$ with $\lambda_1, \lambda_2 \geq 0 \Rightarrow \nabla f$ is in the positive quadrant of $(\nabla c_1, \nabla c_2)$.



Another example:



$$\text{At } \begin{cases} \mathbf{x}: & \text{no } \mathbf{d} \text{ satisfies } \nabla c_1(\mathbf{x})^T \mathbf{d} \geq 0, \nabla c_2(\mathbf{x})^T \mathbf{d} \geq 0, \nabla f(\mathbf{x})^T \mathbf{d} < 0 \\ \mathbf{z}: & \mathbf{d}_{\mathbf{z}} \text{ satisfies } \nabla c_1(\mathbf{z})^T \mathbf{d} \geq 0, \nabla c_2(\mathbf{z})^T \mathbf{d} \geq 0, \nabla f(\mathbf{z})^T \mathbf{d} < 0 \\ \mathbf{y}: & \mathbf{d}_{\mathbf{y}} \text{ satisfies } c_1(\mathbf{y}) + \nabla c_1(\mathbf{y})^T \mathbf{d} \geq 0 \text{ (} c_1 \text{ not active), } \nabla c_2(\mathbf{y})^T \mathbf{d} \geq 0, \nabla f(\mathbf{y})^T \mathbf{d} < 0 \\ \mathbf{w}: & \mathbf{d}_{\mathbf{w}} \text{ satisfies } c_1(\mathbf{w}) + \nabla c_1(\mathbf{w})^T \mathbf{d} \geq 0, c_2(\mathbf{w}) + \nabla c_2(\mathbf{w})^T \mathbf{d} \geq 0 \text{ (} c_1, c_2 \text{ not active), } \nabla f(\mathbf{w})^T \mathbf{d} < 0. \end{cases}$$

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

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

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

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

- $|\mathcal{E} \cup \mathcal{I}| = m$ constraints.

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

- Active set $\mathcal{A}(\mathbf{x}) = \mathcal{E} \cup \{i \in \mathcal{I} : c_i(\mathbf{x}) = 0\}$. In matrix form: $\mathbf{A}(\mathbf{x}) = [\nabla c_i(\mathbf{x})]_{i \in \mathcal{A}(\mathbf{x})}^T$ of $\cdot \times n$.

- Degenerate constraint behavior: e.g. $c_1^2(\mathbf{x})$ is equivalent to $c_1(\mathbf{x})$ as an equality constraint, but $\nabla(c_1^2) = 2c_1 \nabla c_1 = \mathbf{0}$ at any feasible point, which disables the condition $\nabla f = \lambda_1 \nabla c_1$. We can avoid degenerate behavior by requiring the following constraint qualification:

- Def. 12.4: given \mathbf{x}^* , $\mathcal{A}(\mathbf{x}^*)$, the *linear independence constraint qualification (LICQ)* holds iff the set of active constraint gradients $\{\nabla c_i(\mathbf{x}^*), i \in \mathcal{A}(\mathbf{x}^*)\}$ is l.i. (which implies $\nabla c_i(\mathbf{x}^*) \neq \mathbf{0}$). Equivalently $\mathbf{A}(\mathbf{x}^*)$ has full row rank.

Other constraint qualifications possible in th. 12.1, in partic. “all active constraints are linear”.

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

ex. 12.6

- | | | |
|----|--|--------------------------|
| a) | $\nabla_{\mathbf{x}} \mathcal{L}(\mathbf{x}^*, \boldsymbol{\lambda}^*) = \mathbf{0}$ | n eqs. |
| b) | $c_i(\mathbf{x}^*) = 0 \quad \forall i \in \mathcal{E}$ | |
| c) | $c_i(\mathbf{x}^*) \geq 0 \quad \forall i \in \mathcal{I}$ | |
| d) | $\lambda_i^* \geq 0 \quad \forall i \in \mathcal{I}$ | |
| e) | $\lambda_i^* c_i(\mathbf{x}^*) = 0 \quad \forall i \in \mathcal{E} \cup \mathcal{I}$ | m complementarity eqs. |

Notes:

- $\mathcal{I} = \emptyset$: KKT $\Leftrightarrow \nabla \mathcal{L}(\mathbf{x}^*, \boldsymbol{\lambda}^*) = \mathbf{0}$ (∇ wrt $\mathbf{x}, \boldsymbol{\lambda}$).
In principle solvable by writing $\mathbf{x} = (\mathbf{x}_a, \phi(\mathbf{x}_a))^T$ and solving the unconstrained problem $\min f(\mathbf{x}_a)$.
- Strict complementarity: exactly one of λ_i^* and $c_i(\mathbf{x}^*)$ is zero $\forall i \in \mathcal{I}$. Easier for some algorithms.

Sensitivity of f wrt constraints

λ_i^* = how hard f is pushing or pulling against c_i $\begin{cases} \lambda_i^* = 0 : & f \text{ does not change} \\ \lambda_i^* \neq 0 : & f \text{ changes proportionately to } \lambda_i^*. \end{cases}$

Intuitive proof: infinitesimal perturbation of c_i (inequality constraint): $c_i(\mathbf{x}) \geq \epsilon$. Suppose ϵ is sufficiently small that the perturbed solution $\mathbf{x}^*(\epsilon)$ still has the same active constraints and that the Lagrange multipliers are not affected much. Then (\diamond : Taylor's th. to first order):

$$\left. \begin{aligned} \epsilon &= \overbrace{c_i(\mathbf{x}^*(\epsilon))}^{\epsilon} - \overbrace{c_i(\mathbf{x}^*)}^0 \overset{\diamond}{\approx} (\mathbf{x}^*(\epsilon) - \mathbf{x}^*)^T \nabla c_i(\mathbf{x}^*) \\ 0 &= \underbrace{c_j(\mathbf{x}^*(\epsilon))}_0 - \underbrace{c_j(\mathbf{x}^*)}_0 \overset{\diamond}{\approx} (\mathbf{x}^*(\epsilon) - \mathbf{x}^*)^T \nabla c_j(\mathbf{x}^*) \quad \forall j \neq i, j \in \mathcal{A}(\mathbf{x}^*) \end{aligned} \right\} \Rightarrow$$

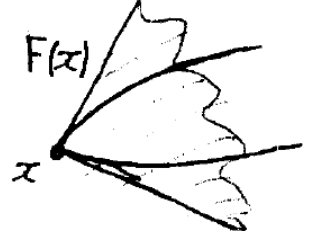
$$f(\mathbf{x}^*(\epsilon)) - f(\mathbf{x}^*) \overset{\diamond}{\approx} (\mathbf{x}^*(\epsilon) - \mathbf{x}^*)^T \nabla f(\mathbf{x}^*) \stackrel{KKT}{=} \sum_{j \in \mathcal{A}(\mathbf{x}^*)} \lambda_j^* (\mathbf{x}^*(\epsilon) - \mathbf{x}^*)^T \nabla c_j(\mathbf{x}^*) \approx \lambda_i^* \epsilon \xrightarrow{\epsilon \rightarrow 0} \frac{df(\mathbf{x}^*(\epsilon))}{d\epsilon} = \lambda_i^*.$$

An active constraint c_i at \mathbf{x}^* is $\begin{cases} \text{strongly} \\ \text{weakly} \end{cases}$ active if $\lambda_i^* \begin{cases} > 0 \\ = 0 \end{cases}$.

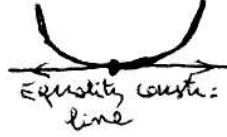
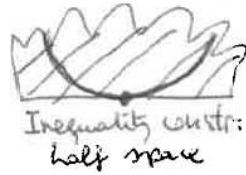
Second order conditions

- Set of linearized feasible directions $\mathcal{F}(\mathbf{x})$ at a feasible point \mathbf{x} , which is a cone (def. 12.3):

$$\mathcal{F}(\mathbf{x}) = \left\{ \mathbf{w} \in \mathbb{R}^n : \begin{array}{l} \mathbf{w}^T \nabla c_i(\mathbf{x}) = 0, \quad \forall i \in \mathcal{E} \\ \mathbf{w}^T \nabla c_i(\mathbf{x}) \geq 0, \quad \forall i \in \mathcal{I} \cap \mathcal{A}(\mathbf{x}) \end{array} \right\}.$$



If LICQ holds, $\mathcal{F}(\mathbf{x})$ is the *tangent cone* to the feasible set at \mathbf{x} .
Other examples of tangent cones $\mathcal{F}(\mathbf{x})$ in 2D:



- First-order conditions hold \Rightarrow a move along any vector $\mathbf{w} \in \mathcal{F}(\mathbf{x}^*)$ either increases f (if $\mathbf{w}^T \nabla f(\mathbf{x}^*) > 0$) or keeps it constant (if $\mathbf{w}^T \nabla f(\mathbf{x}^*) = 0$), to first order.
- The second-order conditions give information in the undecided directions $\mathbf{w}^T \nabla f(\mathbf{x}^*) = 0$, by examining the curvature along them (what are the undecided directions in the unconstrained case?).
- Given $\mathcal{F}(\mathbf{x}^*)$ and some Lagrange multiplier vector $\boldsymbol{\lambda}^*$ satisfying the KKT conditions, define the *critical cone* $\mathcal{C}(\mathbf{x}^*, \boldsymbol{\lambda}^*)$:

$$\mathcal{C}(\mathbf{x}^*, \boldsymbol{\lambda}^*) = \left\{ \mathbf{w} \in \mathcal{F}(\mathbf{x}^*) : \nabla c_i(\mathbf{x}^*)^T \mathbf{w} = 0, \quad \forall i \in \mathcal{I} \cap \mathcal{A}(\mathbf{x}^*) \text{ with } \lambda_i^* > 0 \right\} \subset \mathcal{F}(\mathbf{x}^*)$$

or equivalently

$$\mathcal{C}(\mathbf{x}^*, \boldsymbol{\lambda}^*) = \left\{ \mathbf{w} \in \mathbb{R}^n : \begin{array}{l} \mathbf{w}^T \nabla c_i(\mathbf{x}^*) = 0 \quad \forall i \in \mathcal{E} \\ \mathbf{w}^T \nabla c_i(\mathbf{x}^*) = 0 \quad \forall i \in \mathcal{I} \cap \mathcal{A}(\mathbf{x}^*) \text{ with } \lambda_i^* > 0 \\ \mathbf{w}^T \nabla c_i(\mathbf{x}^*) \geq 0 \quad \forall i \in \mathcal{I} \cap \mathcal{A}(\mathbf{x}^*) \text{ with } \lambda_i^* = 0 \end{array} \right\}.$$

$\mathcal{C}(\mathbf{x}^*, \boldsymbol{\lambda}^*)$ contains the undecided directions for $\mathcal{F}(\mathbf{x})$:

$$\mathbf{w} \in \mathcal{C}(\mathbf{x}^*, \boldsymbol{\lambda}^*) \xrightarrow{\text{KKT}} \mathbf{w}^T \nabla f(\mathbf{x}^*) = \sum_{i \in \mathcal{E} \cup \mathcal{I}} \lambda_i^* \mathbf{w}^T \nabla c_i(\mathbf{x}^*) = 0 \quad (\text{since either } \lambda_i^* = 0 \text{ or } \mathbf{w}^T \nabla c_i(\mathbf{x}^*) = 0).$$

- Second-order necessary conditions (Th. 12.5):** \mathbf{x}^* local solution, LICQ condition holds, KKT conditions hold with Lagrangian multiplier vector $\boldsymbol{\lambda}^* \Rightarrow \mathbf{w}^T \nabla_{\mathbf{xx}}^2 \mathcal{L}(\mathbf{x}^*, \boldsymbol{\lambda}^*) \mathbf{w} \geq 0 \quad \forall \mathbf{w} \in \mathcal{C}(\mathbf{x}^*, \boldsymbol{\lambda}^*)$.
- Second-order sufficient conditions (Th. 12.6):** $\mathbf{x}^* \in \mathbb{R}^n$ feasible point, KKT conditions hold with Lagrange multiplier $\boldsymbol{\lambda}^*$, $\mathbf{w}^T \nabla_{\mathbf{xx}}^2 \mathcal{L}(\mathbf{x}^*, \boldsymbol{\lambda}^*) \mathbf{w} > 0 \quad \forall \mathbf{w} \in \mathcal{C}(\mathbf{x}^*, \boldsymbol{\lambda}^*), \mathbf{w} \neq \mathbf{0} \Rightarrow \mathbf{x}^*$ is a strict local solution. ex. 12.8
ex. 12.9
(What happens with $\mathcal{F}(\mathbf{x}^*)$, $\mathcal{C}(\mathbf{x}^*)$ and $\nabla_{\mathbf{xx}}^2 \mathcal{L}(\mathbf{x}^*, \boldsymbol{\lambda}^*)$ at an interior KKT point \mathbf{x}^* ?)
- Weaker but useful statement of the second-order conditions: assume LICQ and strict complementarity hold, then $\mathcal{C}(\mathbf{x}^*, \boldsymbol{\lambda}^*)$ is the null space of $\mathbf{A}(\mathbf{x}^*)$. If \mathbf{Z} contains a basis of it, then the necessary and the sufficient conditions reduce to the *projected Hessian* $\mathbf{Z}^T \nabla_{\mathbf{xx}}^2 \mathcal{L}(\mathbf{x}^*, \boldsymbol{\lambda}^*) \mathbf{Z}$ being psd or pd, resp.

Review: theory of constrained optimization

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

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

First-order necessary (KKT) conditions:

- *Lagrangian* $\mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}) = f(\mathbf{x}) - \sum_{i \in \mathcal{E} \cup \mathcal{I}} \lambda_i c_i(\mathbf{x})$.
- LICQ: active constraint gradients are l.i.
- Unconstrained opt: $\nabla f(\mathbf{x}^*) = \mathbf{0}$ } n eqs., n unknowns.
- Constrained opt: $\begin{cases} \nabla_{\mathbf{x}} \mathcal{L}(\mathbf{x}^*, \boldsymbol{\lambda}^*) = \mathbf{0} \\ c_i(\mathbf{x}^*) = 0, & i \in \mathcal{E} \\ \lambda_i c_i(\mathbf{x}^*) = 0, & i \in \mathcal{E} \cup \mathcal{I} \\ c_i(\mathbf{x}^*) \geq 0, & i \in \mathcal{I} \\ \lambda_i \geq 0, & i \in \mathcal{I} \end{cases} \begin{cases} m + n \text{ eqs.}, \\ m + n \text{ unknowns} \\ \text{with additional} \\ \text{constraints} \end{cases}$

Second-order conditions:

- *Critical cone* containing the undecided directions:

$$\mathcal{C}(\mathbf{x}^*, \boldsymbol{\lambda}^*) = \left\{ \mathbf{w} \in \mathbb{R}^n : \begin{array}{ll} \mathbf{w}^T \nabla c_i(\mathbf{x}^*) = 0 & \forall i \in \mathcal{E} \\ \mathbf{w}^T \nabla c_i(\mathbf{x}^*) = 0 & \forall i \in \mathcal{I} \cap \mathcal{A}(\mathbf{x}^*) \text{ with } \lambda_i^* > 0 \\ \mathbf{w}^T \nabla c_i(\mathbf{x}^*) \geq 0 & \forall i \in \mathcal{I} \cap \mathcal{A}(\mathbf{x}^*) \text{ with } \lambda_i^* = 0 \end{array} \right\}.$$

- Necessary: $(\mathbf{x}^*, \boldsymbol{\lambda}^*)$ local solution + LICQ + KKT $\Rightarrow \mathbf{w}^T \nabla_{\mathbf{xx}}^2 \mathcal{L}(\mathbf{x}^*, \boldsymbol{\lambda}^*) \mathbf{w} \geq 0 \quad \forall \mathbf{w} \in \mathcal{C}(\mathbf{x}^*, \boldsymbol{\lambda}^*)$.
- Sufficient: $(\mathbf{x}^*, \boldsymbol{\lambda}^*)$ KKT point, $\mathbf{w}^T \nabla_{\mathbf{xx}}^2 \mathcal{L}(\mathbf{x}^*, \boldsymbol{\lambda}^*) \mathbf{w} > 0 \quad \forall \mathbf{w} \in \mathcal{C}(\mathbf{x}^*, \boldsymbol{\lambda}^*) \setminus \{\mathbf{0}\} \Rightarrow \mathbf{x}^*$ strict local sol.

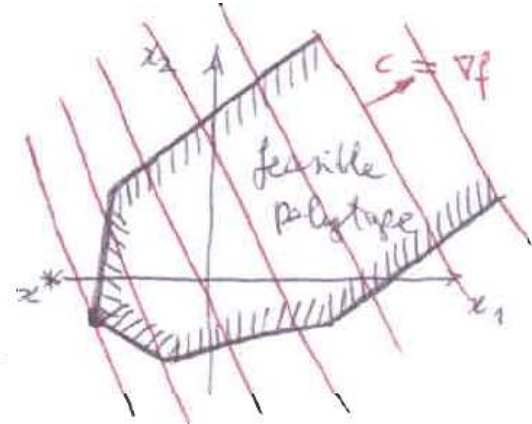
Seek solutions of KKT system, then check whether they are really minimizers (second-order conditions).

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 function: hyperplanes; solution: either none (feasible set is empty or problem is unbounded), one (a vertex) or an infinite number (edge, face, etc.).

(*What happens if there are no inequalities?*)



- **Standard form LP:** $\min_{\mathbf{x}} \mathbf{c}^T \mathbf{x}$ s.t. $\mathbf{Ax} = \mathbf{b}; \mathbf{x} \geq \mathbf{0}$, where $\mathbf{c}, \mathbf{x} \in \mathbb{R}^n$, $\mathbf{b} \in \mathbb{R}^m$, $\mathbf{A}_{m \times n}$. Assume $m < n$ and \mathbf{A} has full row rank (otherwise $\mathbf{Ax} = \mathbf{b}$ contains redundant rows, is infeasible, or defines a unique point).

- Techniques for transformation to standard form:

- $\max \mathbf{c}^T \mathbf{x} \Leftrightarrow -\min (-\mathbf{c})^T \mathbf{x}$.
- Unbounded variable x : split x into *nonnegative and nonpositive parts*: $x = x^+ - x^-$ where $x^+ = \max(x, 0)$ and $x^- = \max(-x, 0)$.
- $\mathbf{x} \leq \mathbf{u}$ or $\mathbf{Ax} \leq \mathbf{b}$: add *slack variables* $\Leftrightarrow \mathbf{x} + \mathbf{w} = \mathbf{u}$, $\mathbf{w} \geq \mathbf{0}$ or $\mathbf{Ax} + \mathbf{y} = \mathbf{b}$, $\mathbf{y} \geq \mathbf{0}$.
- $\mathbf{x} \geq \mathbf{u}$ or $\mathbf{Ax} \geq \mathbf{b}$: add *surplus variables* $\Leftrightarrow \mathbf{x} - \mathbf{w} = \mathbf{u}$, $\mathbf{w} \geq \mathbf{0}$ or $\mathbf{Ax} - \mathbf{y} = \mathbf{b}$, $\mathbf{y} \geq \mathbf{0}$.

Example:

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

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

Optimality conditions

LP is a convex optimization problem \Rightarrow any minimizer is a global minimizer; KKT conditions are necessary *and also sufficient*; LICQ isn't necessary? (*What happens with the second-order conditions?*)

KKT conditions: $\mathcal{L}(\mathbf{x}, \underbrace{\boldsymbol{\lambda}, \mathbf{s}}_{\text{Lagrange multipliers}}) = \mathbf{c}^T \mathbf{x} - \boldsymbol{\lambda}^T (\mathbf{Ax} - \mathbf{b}) - \mathbf{s}^T \mathbf{x}$. If \mathbf{x} is a solution $\Rightarrow \exists! \boldsymbol{\lambda} \in \mathbb{R}^m, \mathbf{s} \in \mathbb{R}^n$:

$$\left. \begin{array}{l} \text{a) } \mathbf{A}^T \boldsymbol{\lambda} + \mathbf{s} = \mathbf{c} \\ \text{b) } \mathbf{Ax} = \mathbf{b} \\ \text{c) } \mathbf{x} \geq \mathbf{0} \\ \text{d) } \mathbf{s} \geq \mathbf{0} \\ \text{e) } x_i s_i = 0, i = 1, \dots, n \Leftrightarrow \mathbf{x}^T \mathbf{s} = 0 \end{array} \right\} \Rightarrow \mathbf{c}^T \mathbf{x} \stackrel{\text{a)}}{=} (\mathbf{A}^T \boldsymbol{\lambda} + \mathbf{s})^T \mathbf{x} = (\mathbf{Ax})^T \boldsymbol{\lambda} \stackrel{\text{b)}}{=} \mathbf{b}^T \boldsymbol{\lambda}.$$

The KKT conditions are also sufficient. Pf.: let $\bar{\mathbf{x}}$ be another feasible point $\Leftrightarrow \mathbf{A}\bar{\mathbf{x}} = \mathbf{b}, \bar{\mathbf{x}} \geq \mathbf{0}$. Then $\mathbf{c}^T \bar{\mathbf{x}} = \underbrace{(\mathbf{A}^T \boldsymbol{\lambda} + \mathbf{s})^T}_{\text{a)}} \bar{\mathbf{x}} = \mathbf{b}^T \boldsymbol{\lambda} + \underbrace{\bar{\mathbf{x}}^T \mathbf{s}}_{\bar{\mathbf{x}}, \mathbf{s} \geq \mathbf{0}} \geq \mathbf{b}^T \boldsymbol{\lambda} = \mathbf{c}^T \mathbf{x}$. And $\bar{\mathbf{x}}$ optimal $\Leftrightarrow \bar{\mathbf{x}}^T \mathbf{s} = 0$.

The dual problem

- *Primal problem*: $\min \mathbf{c}^T \mathbf{x}$ s.t. $\mathbf{Ax} = \mathbf{b}$, $\mathbf{x} \geq \mathbf{0}$.
Dual problem: $\max \mathbf{b}^T \boldsymbol{\lambda}$ s.t. $\mathbf{A}^T \boldsymbol{\lambda} \leq \mathbf{c}$, or $\min -\mathbf{b}^T \boldsymbol{\lambda}$ s.t. $\mathbf{c} - \mathbf{A}^T \boldsymbol{\lambda} \geq \mathbf{0}$ in the form of ch. 12.
 \mathbf{x} : primal variables (n), $\boldsymbol{\lambda}$: dual variables (m).

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

$$\left. \begin{array}{l} \mathbf{Ax} = \mathbf{b} \\ \mathbf{A}^T \boldsymbol{\lambda} \leq \mathbf{c} \\ \mathbf{x} \geq \mathbf{0} \\ x_i (\mathbf{c} - \mathbf{A}^T \boldsymbol{\lambda})_i = 0, \quad i = 1, \dots, n \end{array} \right\} \begin{array}{l} \text{which are identical to the primal problem's KKT} \\ \text{conditions if we define } \mathbf{s} = \mathbf{c} - \mathbf{A}^T \boldsymbol{\lambda}, \text{ i.e.,} \end{array}$$

	Primal	Dual
$\boldsymbol{\lambda}$	Optimal Lag. mult.	Optimal variables
\mathbf{x}	Optimal variables	Optimal Lag. mult.

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

$$\min \begin{pmatrix} -\mathbf{b} \\ \mathbf{b} \\ \mathbf{0} \end{pmatrix}^T \begin{pmatrix} \boldsymbol{\lambda}^+ \\ \boldsymbol{\lambda}^- \\ \mathbf{s} \end{pmatrix} \text{ s.t. } (\mathbf{A}^T \quad -\mathbf{A}^T \quad \mathbf{I}) \begin{pmatrix} \boldsymbol{\lambda}^+ \\ \boldsymbol{\lambda}^- \\ \mathbf{s} \end{pmatrix} = \mathbf{c}, \quad \begin{pmatrix} \boldsymbol{\lambda}^+ \\ \boldsymbol{\lambda}^- \\ \mathbf{s} \end{pmatrix} \geq \mathbf{0}$$

whose dual is

$$\max \mathbf{c}^T \mathbf{z} \text{ s.t. } \begin{pmatrix} \mathbf{A} \\ -\mathbf{A} \\ \mathbf{I} \end{pmatrix} \mathbf{z} \leq \begin{pmatrix} -\mathbf{b} \\ \mathbf{b} \\ \mathbf{0} \end{pmatrix} \Leftrightarrow \min -\mathbf{c}^T \mathbf{z} \text{ s.t. } \mathbf{Az} = -\mathbf{b}, \quad \mathbf{z} \leq \mathbf{0}$$

i.e., the primal with $\mathbf{z} \equiv -\mathbf{x}$.

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

$$0 \leq \mathbf{x}^T \mathbf{s} = \mathbf{x}^T (\mathbf{c} - \mathbf{A}^T \boldsymbol{\lambda}) = \underbrace{\mathbf{c}^T \mathbf{x} - \mathbf{b}^T \boldsymbol{\lambda}}_{\text{gap}} \Leftrightarrow \mathbf{c}^T \mathbf{x} \geq \mathbf{b}^T \boldsymbol{\lambda}.$$

Thus, the dual objective function $\mathbf{b}^T \boldsymbol{\lambda}$ is a lower bound on the primal objective function $\mathbf{c}^T \mathbf{x}$ (*weak duality*); at a solution the gap is 0.

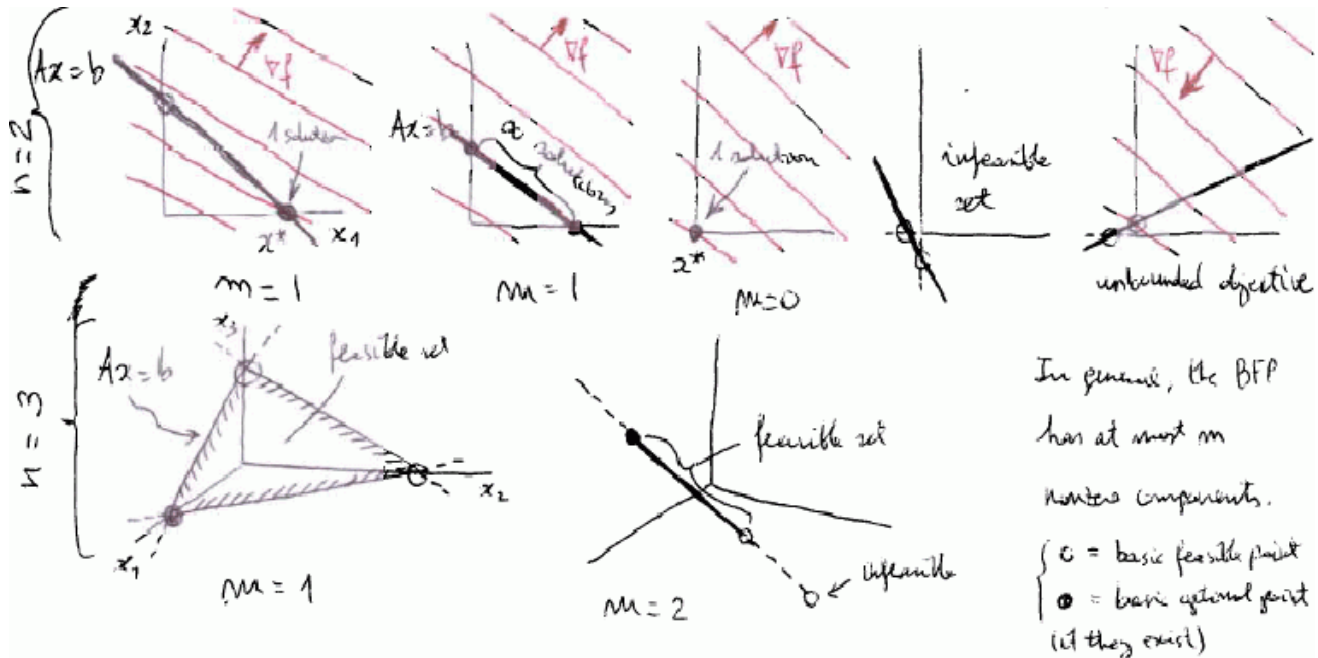
- *Strong duality* (th. 13.1):

1. If either problem (primal or dual) has a (finite) solution, then so does the other, and the objective values are equal.
2. If either problem (primal or dual) is unbounded, then the other problem is infeasible.

- Duality is important in the theory of LP (and convex opt. in general) and in primal-dual algorithms; also, the dual may be easier to solve than the primal.
- *Sensitivity analysis*: how sensitive the global objective value is to perturbations in the constraints \Leftrightarrow find the Lagrange multipliers $\boldsymbol{\lambda}$, \mathbf{s} .

Geometry of the feasible set

$\mathbf{x} \geq 0$ is the n -dim positive quadrant and we consider its intersection with the m -dim ($m < n$) linear subspace $\mathbf{Ax} = \mathbf{b}$. The intersection happens at points \mathbf{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 vectors is a minimizer. Examples:



- \mathbf{x} is a *basic feasible point (BFP)* if \mathbf{x} is a feasible point with at most m nonzero components and we can identify a subset $\mathcal{B}(\mathbf{x})$ of the index set $\{1, \dots, n\}$ such that:

- $\mathcal{B}(\mathbf{x})$ contains exactly m indices
- $i \notin \mathcal{B}(\mathbf{x}) \Rightarrow x_i = 0$
- $\mathbf{B}_{m \times m} = [\mathbf{A}_i]_{i \in \mathcal{B}(\mathbf{x})}$ is nonsingular.

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

$$\mathbf{A} = \begin{pmatrix} x & x & x & x & x \\ x & x & x & x & x \end{pmatrix} \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{matrix} \quad \mathcal{B}(\mathbf{x}) = \{2, 3\} \quad \mathcal{J}(\mathbf{x}) = \{4, 5\}$$

- The simplex method generates a sequence of iterates \mathbf{x}_k that are 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 minimizer).

- Fundamental th. of LP* (th. 13.2): for the standard LP problem:

- \exists feasible point $\Rightarrow \exists$ a BFP
- LP has solutions \Rightarrow at least one such solution is a basic optimal point
- LP is feasible and bounded \Rightarrow it has an optimal solution.

Proof

- All BFPs for the standard LP are vertices of the feasible polytope $\{\mathbf{x}: \mathbf{Ax} = \mathbf{b}, \mathbf{x} \geq 0\}$ and vice versa (th. 13.3). (A vertex is a point that does not lie on a straight line between two other points in the polytope).

Proof

- A LP is *degenerate* if there exists at least one BFP with $\leq m$ nonzero components.



The simplex method

(Not to be confused with Nelder & Mead's downhill simplex method of derivative-free opt.)

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 guarantees 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 $\mathcal{B}(\mathbf{x})$ differs in exactly one component and either decreases the objective or keeps it unchanged. (20)
(10) $\sim 10^5$

The 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):

$$\begin{aligned} \mathbf{B} &= [\mathbf{A}_i]_{i \in \mathcal{B}} & \mathbf{N} &= [\mathbf{A}_i]_{i \in \mathcal{N}} & \Rightarrow \mathbf{A} &= (\mathbf{B} \ \mathbf{N}) \\ \mathbf{x}_\mathbf{B} &= [x_i]_{i \in \mathcal{B}} & \mathbf{x}_\mathbf{N} &= [x_i]_{i \in \mathcal{N}} & \Rightarrow \mathbf{x} &= \begin{pmatrix} \mathbf{x}_\mathbf{B} \\ \mathbf{x}_\mathbf{N} \end{pmatrix}, \text{ also } \mathbf{s} = \begin{pmatrix} \mathbf{s}_\mathbf{B} \\ \mathbf{s}_\mathbf{N} \end{pmatrix}, \ \mathbf{c} = \begin{pmatrix} \mathbf{c}_\mathbf{B} \\ \mathbf{c}_\mathbf{N} \end{pmatrix}. \end{aligned}$$

- Since \mathbf{x} is a BFP we have: \mathbf{B} nonsingular, $\mathbf{x}_\mathbf{B} \geq \mathbf{0}$, $\mathbf{x}_\mathbf{N} = \mathbf{0}$, so KKT c) holds.
- KKT a): $\mathbf{b} = \mathbf{A}\mathbf{x} = \mathbf{B}\mathbf{x}_\mathbf{B} + \mathbf{N}\mathbf{x}_\mathbf{N} \Rightarrow \mathbf{x}_\mathbf{B} = \mathbf{B}^{-1}\mathbf{b}$.
- KKT e): $\mathbf{x}^T \mathbf{s} = \mathbf{x}_\mathbf{B}^T \mathbf{s}_\mathbf{B} = 0 \Rightarrow \mathbf{s}_\mathbf{B} = \mathbf{0}$.
- KKT a): $\mathbf{s} + \mathbf{A}^T \boldsymbol{\lambda} = \mathbf{c} \Rightarrow \begin{pmatrix} \mathbf{s}_\mathbf{B} \\ \mathbf{s}_\mathbf{N} \end{pmatrix} + \begin{pmatrix} \mathbf{B}^T \\ \mathbf{N}^T \end{pmatrix} \boldsymbol{\lambda} = \begin{pmatrix} \mathbf{B}^T \boldsymbol{\lambda} \\ \mathbf{s}_\mathbf{N} + \mathbf{N}^T \boldsymbol{\lambda} \end{pmatrix} = \begin{pmatrix} \mathbf{c}_\mathbf{B} \\ \mathbf{c}_\mathbf{N} \end{pmatrix} \Rightarrow \begin{cases} \boldsymbol{\lambda} = (\mathbf{B}^{-1})^T \mathbf{c}_\mathbf{B} \\ \mathbf{s}_\mathbf{N} = \mathbf{c}_\mathbf{N} - (\mathbf{B}^{-1} \mathbf{N})^T \mathbf{c}_\mathbf{B}. \end{cases}$
- KKT d): $\mathbf{s} \geq \mathbf{0}$: while $\mathbf{s}_\mathbf{B}$ satisfies this, $\mathbf{s}_\mathbf{N} = \mathbf{c}_\mathbf{N} - (\mathbf{B}^{-1} \mathbf{N})^T \mathbf{c}_\mathbf{B}$ may not (if it does, i.e., $\mathbf{s}_\mathbf{N} \geq \mathbf{0}$, we have found an optimal $(\mathbf{x}, \boldsymbol{\lambda}, \mathbf{s})$ 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 $\mathbf{x}_\mathbf{N}$ to 0
 - figure out the effect of increasing x_q on the current BFP $\mathbf{x}_\mathbf{B}$, given that we want it to stay feasible wrt $\mathbf{A}\mathbf{x} = \mathbf{b}$
 - keep increasing x_q until one of components of $\mathbf{x}_\mathbf{B}$ (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 \mathbf{x}^+ the new iterate and \mathbf{x} the current one: we want $\mathbf{A}\mathbf{x}^+ = \mathbf{b} = \mathbf{A}\mathbf{x}$:

$$\mathbf{A}\mathbf{x}^+ = (\mathbf{B} \ \mathbf{N}) \begin{pmatrix} \mathbf{x}_\mathbf{B}^+ \\ \mathbf{x}_\mathbf{N}^+ \end{pmatrix} \stackrel{\diamond}{=} \underbrace{\mathbf{B}}_{m \times m} \underbrace{\mathbf{x}_\mathbf{B}^+}_{m \times 1} + \underbrace{\mathbf{A}_q}_{m \times 1} \underbrace{x_q^+}_{1 \times 1} = \mathbf{B}\mathbf{x}_\mathbf{B} = \mathbf{A}\mathbf{x} \Rightarrow \mathbf{x}_\mathbf{B}^+ = \mathbf{x}_\mathbf{B} - \underbrace{\mathbf{B}^{-1} \mathbf{A}_q x_q^+}_{\text{increase } x_q^+ \text{ till some component of } \mathbf{x}_\mathbf{B}^+ \text{ becomes 0}}$$

($\diamond : x_i^+ = 0$ for $i \in \mathcal{N} \setminus \{q\}$). This operation decreases $\mathbf{c}^T \mathbf{x}$ (pf.: p. 369).

LP nondegenerate & bounded \Rightarrow simplex method terminates at a basic optimal point (th. 13.4).

- 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 \mathbf{s} .

- With inequalities, as indicated by the KKT conditions, an algorithm must determine (implicitly or explicitly) which of them are active at a solution. *Active-set methods*, of which the simplex method is an example:
 - maintain explicitly a set of constraints that estimates the active set at the solution (the complement of the basis \mathcal{B} in the simplex method), and
 - make small changes to it at each step (a single index in the simplex method).

Active-set methods apply also to QP and bound-constrained optimization, but are less convenient for nonlinear programming.

- The simplex method is very efficient in practice (if 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.

Review: linear programming: the simplex method

- *Linear program*:
 - linear objective and constraints
 - convex problem
 - feasible set: polytope
 - number of solutions: 0 (infeasible or unbounded), 1 (a vertex) or ∞ (a face)
 - KKT conditions are necessary and sufficient; LICQ not needed.
- *Primal problem* in LP std form (use slacks, etc. if needed): $\min_{\mathbf{x}} \mathbf{c}^T \mathbf{x}$ s.t. $\mathbf{Ax} = \mathbf{b}$; $\mathbf{x} \geq \mathbf{0}$ with $\mathbf{A}_{m \times n}$.
- *Dual problem*: $\max \mathbf{b}^T \boldsymbol{\lambda}$ s.t. $\mathbf{A}^T \boldsymbol{\lambda} \leq \mathbf{c}$
 - optimal variables \mathbf{x} and Lagrange multipliers $\boldsymbol{\lambda}$ switch roles
 - $\text{dual}(\text{dual}(\text{primal})) = \text{primal}$
 - weak duality: dual objective \leq primal objective for any feasible point; they coincide at the solution
 - strong duality: the primal and the dual, either both have a finite solution, or one is unbounded and the other infeasible.
- Sensitivity analysis: values of the Lagrange multipliers.
- The *simplex method*:
 - tests a finite sequence of polytope vertices with nonincreasing objective values, final one = solution
 - each step requires solving a linear system
 - needs to deal with degenerate and unbounded cases
 - takes $2m$ – $3m$ steps in practice
 - is an *active-set method*.

14 Linear programming: interior-point methods

Interior-point methods	Simplex method
All iterates satisfy the inequality constraints <i>strictly</i> , so in the limit 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 finite sequence of vertices until it finds the optimal one.
Each iteration is expensive to compute but can make significant progress toward the solution.	Usually requires a larger number of inexpensive iterations.
Average-case complexity = worst-case complexity = polynomial.	Average-case complexity: $2m-3m$ iterations (m = number of constraints); worst-case complexity: exponential.

Primal-dual methods

- Standard-form primal LP: $\min \mathbf{c}^T \mathbf{x}$ s.t. $\mathbf{Ax} = \mathbf{b}$, $\mathbf{x} \geq \mathbf{0}$, $\mathbf{c}, \mathbf{x} \in \mathbb{R}^n$, $\mathbf{b} \in \mathbb{R}^m$, $\mathbf{A}_{m \times n}$
Dual LP: $\max \mathbf{b}^T \boldsymbol{\lambda}$ s.t. $\mathbf{A}^T \boldsymbol{\lambda} + \mathbf{s} = \mathbf{c}$, $\mathbf{s} \geq \mathbf{0}$, $\boldsymbol{\lambda} \in \mathbb{R}^m$, $\mathbf{s} \in \mathbb{R}^n$.

- KKT conditions:

$$\left. \begin{array}{l} \mathbf{A}^T \boldsymbol{\lambda} + \mathbf{s} = \mathbf{c} \\ \mathbf{Ax} = \mathbf{b} \\ x_i s_i = 0 \\ i = 1, \dots, n \\ \mathbf{x}, \mathbf{s} \geq \mathbf{0} \end{array} \right\} \begin{array}{l} \text{system of } 2n + m \text{ equations} \\ \text{for } 2n + m \text{ unknowns } \mathbf{x}, \boldsymbol{\lambda}, \mathbf{s} \\ \text{(mildly nonlinear because of } x_i s_i) \end{array} \Leftrightarrow \mathbf{F}(\mathbf{x}, \boldsymbol{\lambda}, \mathbf{s}) = \begin{pmatrix} \mathbf{A}^T \boldsymbol{\lambda} + \mathbf{s} - \mathbf{c} \\ \mathbf{Ax} - \mathbf{b} \\ \mathbf{XSe} \end{pmatrix} = \mathbf{0}$$

$$\mathbf{X} = \text{diag}(x_i), \mathbf{S} = \text{diag}(s_i), \mathbf{e} = \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix}.$$

- Idea: find solutions $(\mathbf{x}^*, \boldsymbol{\lambda}^*, \mathbf{s}^*)$ of this system with a Newton-like method, but modifying the search directions and step sizes to satisfy $\mathbf{x}, \mathbf{s} > \mathbf{0}$ (strict inequality). The sequence of iterates traces a path in the space $(\mathbf{x}, \boldsymbol{\lambda}, \mathbf{s})$, thus the name primal-dual. Solving the system is relatively easy (little nonlinearity) but the nonnegativity condition complicates things. Spurious solutions ($\mathbf{F}(\mathbf{x}, \boldsymbol{\lambda}, \mathbf{s}) = \mathbf{0}$ but $\mathbf{x}, \mathbf{s} \not\geq \mathbf{0}$) abound and do not provide useful information about feasible solutions, so we must ensure to exclude them.

All the vertices of the \mathbf{x} -polytope are associated with a root of \mathbf{F} , but most violate $\mathbf{x}, \mathbf{s} \geq \mathbf{0}$.

- Newton's method to solve nonlinear equations $\mathbf{r}(\mathbf{x}) = \mathbf{0}$ from current estimate \mathbf{x}_k (ch. 11): $\mathbf{x}_{k+1} = \mathbf{x}_k + \Delta \mathbf{x}$ where $\mathbf{J}(\mathbf{x}_k) \Delta \mathbf{x} = -\mathbf{r}(\mathbf{x}_k)$ and $\mathbf{J}(\mathbf{x})$ is the Jacobian of \mathbf{r} .
(Recall that if we apply it to solve $\nabla f(\mathbf{x}) = \mathbf{0}$ we obtain $\nabla^2 f(\mathbf{x}) \mathbf{p} = -\nabla f(\mathbf{x})$, Newton's method for optimization.)
- In our case the Jacobian $\mathbf{J}(\mathbf{x}, \boldsymbol{\lambda}, \mathbf{s})$ takes a simple form. Assuming $\mathbf{x}, \mathbf{s} > \mathbf{0}$ and calling $\mathbf{r}_c = \mathbf{A}^T \boldsymbol{\lambda} + \mathbf{s} - \mathbf{c}$, $\mathbf{r}_b = \mathbf{Ax} - \mathbf{b}$ the residuals for the linear equations, the Newton step is:

$$\mathbf{J}(\mathbf{x}, \boldsymbol{\lambda}, \mathbf{s}) = \begin{pmatrix} \mathbf{0} & \mathbf{A}^T & \mathbf{I} \\ \mathbf{A} & \mathbf{0} & \mathbf{0} \\ \mathbf{S} & \mathbf{0} & \mathbf{X} \end{pmatrix} \Rightarrow \mathbf{J} \begin{pmatrix} \Delta \mathbf{x} \\ \Delta \boldsymbol{\lambda} \\ \Delta \mathbf{s} \end{pmatrix} = \begin{pmatrix} -\mathbf{r}_c \\ -\mathbf{r}_b \\ -\mathbf{XSe} \end{pmatrix} \quad \text{This Newton direction is also called } \textit{affine scaling direction}.$$

Since a full step would likely violate $\mathbf{x}, \mathbf{s} > \mathbf{0}$, we perform a line search so that the new iterate is $\begin{pmatrix} \mathbf{x} \\ \boldsymbol{\lambda} \\ \mathbf{s} \end{pmatrix} + \alpha \begin{pmatrix} \Delta \mathbf{x} \\ \Delta \boldsymbol{\lambda} \\ \Delta \mathbf{s} \end{pmatrix}$ for $\alpha \in (0, 1]$ [$\Rightarrow \alpha < \min(\min_{\Delta x_i < 0}(\frac{-x_i}{\Delta x_i}), \min_{\Delta s_i < 0}(\frac{-s_i}{\Delta s_i}))$].

Still, often $\alpha \ll 1$. Primal-dual methods modify the basic Newton procedure by:

1. Biasing the search direction towards the interior of the nonnegative orthant $\mathbf{x}, \mathbf{s} \geq \mathbf{0}$ (so more room to move within it). We take a less aggressive Newton direction that aims at a solution with $x_i s_i = \sigma \mu > 0$ (*perturbed KKT conditions*) instead of all the way to 0 (this usually allows a longer step α), with:
 - Duality measure $\mu = \frac{\mathbf{x}^T \mathbf{s}}{n}$ = average of the pairwise products $x_i s_i$. It measures closeness to the boundary, and the algorithms drive μ to zero.
 - Centering parameter $\sigma \in [0, 1]$: amount of reduction in μ we want to achieve:
 - $\sigma = 0$: pure N. step towards $(\mathbf{x}_0, \boldsymbol{\lambda}_0, \mathbf{s}_0)$ (*affine-scaling dir.*); aims at reducing μ .
 - $\sigma = 1$: N. step towards $(\mathbf{x}_\mu, \boldsymbol{\lambda}_\mu, \mathbf{s}_\mu) \in \mathcal{C}$ (*centering direction*); aims at centrality.
 Primal-dual methods trade off both aims.
2. Controlling the step α to keep x_i, s_i from moving too close to the boundary of the nonnegative orthant.

Framework 14.1 (Primal-dual path-following): given $(\mathbf{x}^0, \boldsymbol{\lambda}^0, \mathbf{s}^0)$ with $\mathbf{x}^0, \mathbf{s}^0 > \mathbf{0}$

for $k = 0, 1, 2, \dots$

$$\text{Solve } \begin{pmatrix} \mathbf{0} & \mathbf{A}^T & \mathbf{I} \\ \mathbf{A} & \mathbf{0} & \mathbf{0} \\ \mathbf{S}^k & \mathbf{0} & \mathbf{X}^k \end{pmatrix} \begin{pmatrix} \Delta \mathbf{x}^k \\ \Delta \boldsymbol{\lambda}^k \\ \Delta \mathbf{s}^k \end{pmatrix} = \begin{pmatrix} -\mathbf{r}_c \\ -\mathbf{r}_b \\ -\mathbf{X}^k \mathbf{S}^k \mathbf{e} + \sigma_k \mu_k \mathbf{e} \end{pmatrix} \text{ where } \sigma_k \in [0, 1], \mu_k = \frac{(\mathbf{x}^k)^T \mathbf{s}^k}{n}$$

$$(\mathbf{x}^{k+1}, \boldsymbol{\lambda}^{k+1}, \mathbf{s}^{k+1}) \leftarrow (\mathbf{x}^k, \boldsymbol{\lambda}^k, \mathbf{s}^k) + \alpha_k (\Delta \mathbf{x}^k, \Delta \boldsymbol{\lambda}^k, \Delta \mathbf{s}^k) \text{ choosing } \alpha_k \text{ such that } \mathbf{x}^{k+1}, \mathbf{s}^{k+1} > \mathbf{0}$$

end

- The strategies for choosing or adapting σ_k, α_k depend on the particular algorithm.
- If a full step ($\alpha_k = 1$) is taken at any iteration, the residuals $\mathbf{r}_c, \mathbf{r}_b$ become $\mathbf{0}$ and all the subsequent iterates remain strictly feasible?

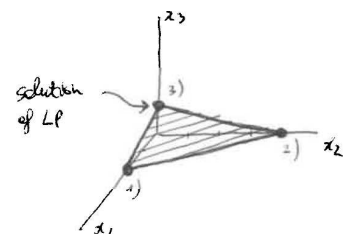
Strict feasibility is preserved: $(\mathbf{x}^k, \boldsymbol{\lambda}^k, \mathbf{s}^k) \in \mathcal{F}^0 \stackrel{?}{\Rightarrow} (\mathbf{x}^{k+1}, \boldsymbol{\lambda}^{k+1}, \mathbf{s}^{k+1}) \in \mathcal{F}^0$.

Examples

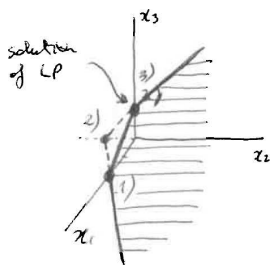
- This shows every vertex of the polytope in \mathbf{x} (i.e., $\mathbf{Ax} = \mathbf{b}$) produces one root of $\mathbf{F}(\mathbf{x}, \boldsymbol{\lambda}, \mathbf{s})$.
 $\min \mathbf{c}^T \mathbf{x}$ s.t. $\mathbf{Ax} = \mathbf{b}, \mathbf{x} \geq \mathbf{0}$, for $\mathbf{c} = (1, 1, 0)^T$, $\mathbf{A} = (1 \ \frac{1}{2} \ 2)$, $b = 2$. KKT conditions:

$$\left. \begin{array}{l} \mathbf{A}^T \boldsymbol{\lambda} + \mathbf{s} = \mathbf{c} \\ \mathbf{Ax} = \mathbf{b} \\ \mathbf{s}^T \mathbf{x} = 0 \\ \mathbf{x}, \mathbf{s} \geq \mathbf{0} \end{array} \right\} \Rightarrow \left. \begin{array}{l} \mathbf{F}(\mathbf{x}, \boldsymbol{\lambda}, \mathbf{s}) = \mathbf{0} \\ \lambda + s_1 = 1 \\ \frac{\lambda}{2} + s_2 = 1 \\ 2\lambda + s_3 = 0 \\ x_1 + \frac{1}{2}x_2 + 2x_3 = 2 \\ s_1 x_1 = 0 \\ s_2 x_2 = 0 \\ s_3 x_3 = 0 \end{array} \right\} \text{ with solutions}$$

- (1) $\lambda = 1, \mathbf{x} = (2, 0, 0)^T, \mathbf{s} = (0, \frac{1}{2}, -2)^T$ infeasible
- (2) $\lambda = 2, \mathbf{x} = (0, 4, 0)^T, \mathbf{s} = (-1, 0, -4)^T$ infeasible
- (3) $\lambda = 0, \mathbf{x} = (0, 0, 1)^T, \mathbf{s} = (1, 1, 0)^T$ feasible



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



- (1) $\lambda = 1, \mathbf{x} = (2, 0, 0)^T, \mathbf{s} = (0, 3, -2)^T$ infeasible
- (2) $\lambda = -\frac{1}{2}, \mathbf{x} = (0, -1, 0)^T, \mathbf{s} = (\frac{3}{2}, 0, 1)^T$ infeasible
- (3) $\lambda = 0, \mathbf{x} = (0, 0, 1)^T, \mathbf{s} = (1, 1, 0)^T$ feasible

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

The central path

Define $\begin{cases} \text{primal-dual feasible set } \mathcal{F} = \{(\mathbf{x}, \boldsymbol{\lambda}, \mathbf{s}): \mathbf{Ax} = \mathbf{b}, \mathbf{A}^T \boldsymbol{\lambda} + \mathbf{s} = \mathbf{c}, \mathbf{x}, \mathbf{s} \geq \mathbf{0}\} \\ \text{primal-dual strictly feasible set } \mathcal{F}^0 = \{(\mathbf{x}, \boldsymbol{\lambda}, \mathbf{s}): \mathbf{Ax} = \mathbf{b}, \mathbf{A}^T \boldsymbol{\lambda} + \mathbf{s} = \mathbf{c}, \mathbf{x}, \mathbf{s} > \mathbf{0}\}. \end{cases}$

Before, we justified taking a step towards $\tau = \sigma\mu > 0$ instead of directly towards 0 in that it keeps us away from the feasible set boundaries and allows longer steps. We can also see this as following a central path through the nonnegative orthant. Parameterize the KKT system in terms of a scalar parameter $\tau > 0$ (*perturbed KKT conditions*):

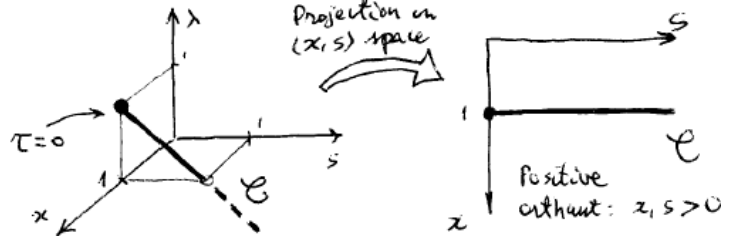
$$\left. \begin{array}{l} \mathbf{A}^T \boldsymbol{\lambda} + \mathbf{s} = \mathbf{c} \\ \mathbf{Ax} = \mathbf{b} \\ x_i s_i = \tau \\ i = 1, \dots, n \\ \mathbf{x}, \mathbf{s} > \mathbf{0} \end{array} \right\} \begin{array}{l} \text{The solution } \mathbf{F}(\mathbf{x}_\tau, \boldsymbol{\lambda}_\tau, \mathbf{s}_\tau) = \mathbf{0} \text{ gives a curve } \mathcal{C} = \{(\mathbf{x}_\tau, \boldsymbol{\lambda}_\tau, \mathbf{s}_\tau): \tau > 0\} \text{ whose} \\ \text{points are strictly feasible, and that converges to a solution for } \tau \rightarrow 0. \text{ This} \\ \text{curve is the } \textit{central path } \mathcal{C}. \text{ The central path is defined uniquely } \forall \tau > 0 \Leftrightarrow \\ \mathcal{F}^0 \neq \emptyset. \end{array}$$

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

Example of central path $\min \mathbf{c}^T \mathbf{x}$ s.t. $\mathbf{Ax} = \mathbf{b}$, $\mathbf{x} \geq \mathbf{0}$ for $\mathbf{x} \in \mathbb{R}$; $\mathbf{A} = \mathbf{b} = \mathbf{c} = 1$.

KKT equations for central path \mathcal{C} :

$$\left. \begin{array}{l} \mathbf{A}^T \boldsymbol{\lambda} + \mathbf{s} = \mathbf{c} \\ \mathbf{Ax} = \mathbf{b} \\ \mathbf{x}^T \mathbf{s} = \tau \\ \tau > 0 \end{array} \right\} \Rightarrow \begin{array}{l} x = 1 \\ s = \tau \\ \lambda = 1 - \tau \end{array}$$



Path-following methods

- They explicitly restrict iterates to a neighborhood 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 $(\mathbf{x}^{k+1}, \boldsymbol{\lambda}^{k+1}, \mathbf{s}^{k+1})$ lies in the neighborhood.



- Examples:
 - $\mathcal{N}_{-\infty}(\gamma) = \{(\mathbf{x}, \boldsymbol{\lambda}, \mathbf{s}) \in \mathcal{F}^0: x_i s_i \geq \gamma\mu, i = 1, \dots, n\}$ for $\gamma \in (0, 1]$ (typ. $\gamma = 10^{-3}$); $\begin{cases} \mathcal{N}_{-\infty}(0) = \mathcal{F} \\ \mathcal{N}_{-\infty}(1) = \mathcal{C}; \end{cases}$
 - $\mathcal{N}_2(\theta) = \{(\mathbf{x}, \boldsymbol{\lambda}, \mathbf{s}) \in \mathcal{F}^0: \|\mathbf{XSe} - \mu\mathbf{e}\|_2 \leq \theta\mu\}$ for $\theta \in [0, 1]$; (typ. $\theta = 0.5$); $\begin{cases} \mathcal{N}_2(1) \neq \mathcal{F} \\ \mathcal{N}_2(0) = \mathcal{C}. \end{cases}$
- Global convergence (th. 14.3):
 - $(\sigma_k) \in [\sigma_{\min}, \sigma_{\max}]$ with $0 < \sigma_{\min} < \sigma_{\max} < 1 \Rightarrow \mu_{k+1} \leq (1 - \frac{\delta}{n}) \mu_k$ for constant $\delta \in (0, 1)$.
- Convergence rate (th. 14.4): (worst-case bound; in practice, almost independent of n)
 - given $\epsilon \in (0, 1)$, $\mathcal{O}(n \log \frac{1}{\epsilon})$ iterations are necessary to find a point for which $\mu_k < \epsilon\mu_0$.
- Homotopy methods for nonlinear eqs. follow tightly a tubular neighborhood of the path. Interior-point path-following methods follow a horn-shaped neighborhood, initially wide.
- Most computational effort is spent solving the linear system for the direction:
 - This is often a sparse system because \mathbf{A} is often sparse.
 - If not, can reformulate (by eliminating variables) as a smaller system: eq. (14.42) or (14.44).

A practical algorithm: Mehrotra's predictor-corrector algorithm

At each iteration:

1. *Predictor step* $(\mathbf{x}', \boldsymbol{\lambda}', \mathbf{s}') = (\mathbf{x}, \boldsymbol{\lambda}, \mathbf{s}) + \alpha(\Delta \mathbf{x}^{\text{aff}}, \Delta \boldsymbol{\lambda}^{\text{aff}}, \Delta \mathbf{s}^{\text{aff}})$: affine-scaling direction (i.e., $\sigma = 0$) and largest step size $\alpha \in [0, 1]$ that satisfies $\mathbf{x}', \mathbf{s}' \geq \mathbf{0}$.
2. *Adapt σ* : compute the effectiveness $\mu_{\text{aff}} = \frac{(\mathbf{x}')^T \mathbf{s}'}{n}$ of this step and set $\sigma = (\mu_{\text{aff}}/\mu)^3$. Thus, if the predictor step is effective, μ_{aff} is small and σ is close to 0, otherwise σ is close to 1.
3. *Corrector step*: compute the step direction using this σ :

$$\mathbf{J} \begin{pmatrix} \Delta \mathbf{x} \\ \Delta \boldsymbol{\lambda} \\ \Delta \mathbf{s} \end{pmatrix} = \begin{pmatrix} -\mathbf{r}_{\mathbf{c}} \\ -\mathbf{r}_{\mathbf{b}} \\ -\mathbf{X}\mathbf{S}\mathbf{e} - \Delta \mathbf{X}^{\text{aff}} \Delta \mathbf{S}^{\text{aff}} \mathbf{e} + \sigma \mu \mathbf{e} \end{pmatrix}.$$

This is an approximation to keeping to the central path:

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

Notes:

- Two linear systems must be solved (predictor and corrector steps) but with the same coefficient matrix.
- Heuristics exist to find a good initial point.
- If LP is infeasible or unbounded, the algorithm typically diverges ($\mathbf{r}_{\mathbf{c}}^k, \mathbf{r}_{\mathbf{b}}^k$ and/or $\mu_k \rightarrow \infty$).
- No convergence theory available for this algorithm (which can occasionally diverge); but it has good practical performance.

Review: linear programming: interior-point methods

- Standard-form primal LP: $\min \mathbf{c}^T \mathbf{x}$ s.t. $\mathbf{Ax} = \mathbf{b}$, $\mathbf{x} \geq \mathbf{0}$, $\mathbf{c}, \mathbf{x} \in \mathbb{R}^n$, $\mathbf{b} \in \mathbb{R}^m$, $\mathbf{A}_{m \times n}$.
- KKT conditions with Lagrange multipliers $\boldsymbol{\lambda}$, \mathbf{s} : $\mathbf{F}(\mathbf{x}, \boldsymbol{\lambda}, \mathbf{s}) = \begin{pmatrix} \mathbf{A}^T \boldsymbol{\lambda} + \mathbf{s} - \mathbf{c} \\ \mathbf{Ax} - \mathbf{b} \\ \mathbf{XSe} \end{pmatrix} = \mathbf{0} \quad \begin{cases} \mathbf{X} = \text{diag}(x_i) \\ \mathbf{S} = \text{diag}(s_i) \\ \mathbf{e} = (1, \dots, 1)^T \end{cases}$.
- Apply Newton's method to solve for $(\mathbf{x}, \boldsymbol{\lambda}, \mathbf{s})$ (primal-dual space) but modify the complementarity conditions as $x_i s_i = \tau = \sigma \mu > 0$ to force iterates to be strictly feasible, i.e., interior ($\mathbf{x}, \mathbf{s} > \mathbf{0}$), and drive $\tau \rightarrow 0$. This affords longer steps α :
 - Pure Newton step: $\mathbf{J} \begin{pmatrix} \Delta \mathbf{x} \\ \Delta \boldsymbol{\lambda} \\ \Delta \mathbf{s} \end{pmatrix} = \begin{pmatrix} -\mathbf{r}_c \\ -\mathbf{r}_b \\ -\mathbf{XSe} + \sigma \mu \mathbf{e} \end{pmatrix}$ with $\mathbf{J}(\mathbf{x}, \boldsymbol{\lambda}, \mathbf{s}) = \begin{pmatrix} \mathbf{0} & \mathbf{A}^T & \mathbf{I} \\ \mathbf{A} & \mathbf{0} & \mathbf{0} \\ \mathbf{S} & \mathbf{0} & \mathbf{X} \end{pmatrix}$.
 - New iterate: $\begin{pmatrix} \mathbf{x} \\ \boldsymbol{\lambda} \\ \mathbf{s} \end{pmatrix} + \alpha \begin{pmatrix} \Delta \mathbf{x} \\ \Delta \boldsymbol{\lambda} \\ \Delta \mathbf{s} \end{pmatrix}$ for $\alpha \in (0, 1]$ that ensures the iterate is sufficiently interior.
 - Duality measure $\mu = \frac{\mathbf{x}^T \mathbf{s}}{n}$ (measures progress towards a solution).
 - Centering parameter σ between 0 (*affine-scaling direction*) and 1 (central path).
- The set of solutions as a function of $\tau > 0$ is called the *central path* \mathcal{C} . It serves as guide to a solution from the interior that avoids non-KKT points. Path-following algorithms follow \mathcal{C} more or less closely.
 - Global convergence: $\mu_{k+1} \leq c \mu_k$ for constant $c \in (0, 1)$ if σ_k is bounded away from 0 and 1.
 - Convergence rate: achieving $\mu_k < \epsilon$ requires $\mathcal{O}(n \log \frac{1}{\epsilon})$ iterations \Rightarrow polynomial complexity.
- Each step of the interior-point method requires solving a linear system (for the Newton step) of $2n + m$ eqs. which is sparse if \mathbf{A} is sparse.
- Fewer, more costly iterations than the simplex method. In practice, preferable in large problems.

15 Fundamentals of algorithms for nonlinear constrained optimization

- General constrained optimization problem:

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

Special cases (for which specialized algorithms exist):

- *Linear programming (LP)*: f , all c_i linear; solved by simplex & interior-point methods.
- *Quadratic programming (QP)*: f quadratic, all c_i linear.
- *Linearly constrained optimization*: all c_i linear.
- *Bound-constrained optimization*: 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. (✍ Is QP convex progr.?)
- *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 algorithms*: sequence of \mathbf{x}_k (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 minimizer (global optimization 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. non-negativity 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 allow 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 modeled as objective function f + penalty, where the penalty includes the constraints. However, this can introduce ill-conditioning.
- *Slack variables* are commonly used to simplify an inequality into a bound, at the cost of having an extra equality and slack variable:

$$c_i(\mathbf{x}) \geq 0 \Rightarrow c_i(\mathbf{x}) - s_i = 0, \quad s_i \geq 0 \quad \forall i \in \mathcal{I}.$$

Categorization 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 and augmented Lagrangian methods.
 - *Penalty methods*: combine objective and constraints into a penalty function $\phi(\mathbf{x}; \mu)$ via a *penalty parameter* $\mu > 0$; e.g. if only equality constraints exist:
 - * $\phi(\mathbf{x}; \mu) = f(\mathbf{x}) + \frac{1}{2\mu} \sum_{i \in \mathcal{E}} c_i(\mathbf{x})^2$
 \Rightarrow unconstrained minimization of ϕ wrt \mathbf{x} for a series of decreasing μ values.
 - * $\phi(\mathbf{x}; \mu) = f(\mathbf{x}) + \frac{1}{2\mu} \sum_{i \in \mathcal{E}} |c_i(\mathbf{x})|$ (exact penalty function)
 \Rightarrow single unconstrained minimization for small enough μ .
 - *Augmented Lagrangian methods*: define a function that combines the Lagrangian and a quadratic penalty; e.g. if only equality constraints exist:
 - * $\mathcal{L}_A(\mathbf{x}, \boldsymbol{\lambda}; \mu) = f(\mathbf{x}) - \sum_{i \in \mathcal{E}} \lambda_i c_i(\mathbf{x}) + \frac{1}{2\mu} \sum_{i \in \mathcal{E}} c_i^2(\mathbf{x})$
 \Rightarrow unconstrained min. of \mathcal{L}_A wrt \mathbf{x} for fixed $\boldsymbol{\lambda}$, μ ; update $\boldsymbol{\lambda}$, decrease μ ; repeat.
 - *Sequential linearly constrained methods*: minimize at every iteration a certain Lagrangian function subject to a linearization of the constraints; useful for large problems.
- Ch. 18: *sequential quadratic programming*: 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.
- Ch. 19: *interior-point methods for nonlinear programming*.
 - *Barrier methods*: add terms to the objective (via a *barrier parameter* $\mu > 0$) that are insignificant when \mathbf{x} is safely inside the feasible set but become large as \mathbf{x} approaches the boundary; e.g. if only inequality constraints exist:
 - * $P(\mathbf{x}; \mu) = f(\mathbf{x}) - \mu \sum_{i \in \mathcal{I}} \log c_i(\mathbf{x})$ (logarithmic barrier function)
 \Rightarrow unconstrained minimization of P wrt \mathbf{x} for a series of decreasing μ values.

Class of methods that...	...replace the original constrained problem by
Quadratic penalty methods	a sequence of unconstrained problems
Log-barrier methods	
Augmented Lagrangian methods	
Nonsmooth exact penalty function methods	a single unconstrained problem
Sequential linearly constrained methods	a sequence of linearly constrained problems
Sequential quadratic programming	a sequence of QP problems

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 15.2': safe elimination.

- Example 15.2: elimination alters the problem: $\min x^2 + y^2$ s.t. $(x - 1)^3 = y^2$ has the solution $\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$. Eliminating $y^2 = (x - 1)^3$ yields $\min x^2 + (x - 1)^3$ which is unbounded ($x \rightarrow -\infty$); 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 linearize the constraints, then apply linear elimination.

Linear elimination Consider $\min f(\mathbf{x})$ s.t. $\mathbf{Ax} = \mathbf{b}$ where $\mathbf{A}_{m \times n}$, $m \leq n$, and \mathbf{A} has full rank (otherwise, remove redundant constraints or determine whether the problem is infeasible). Say we eliminate $\mathbf{x}_B = (x_1, \dots, x_m)^T$ (otherwise permute \mathbf{x} , \mathbf{A} and \mathbf{b}); writing $\mathbf{A} = (\mathbf{B} \ \mathbf{N})$ with $\mathbf{B}_{m \times m}$ nonsingular, $\mathbf{N}_{m \times (n-m)}$ and $\mathbf{x} = \begin{pmatrix} \mathbf{x}_B \\ \mathbf{x}_N \end{pmatrix}$, we have $\mathbf{x}_B = \mathbf{B}^{-1}\mathbf{b} - \mathbf{B}^{-1}\mathbf{N}\mathbf{x}_N$ (remember how to find a basic feasible point in the simplex method), and we can solve the unconstrained problem $\min_{\mathbf{x}_N \in \mathbb{R}^{n-m}} f(\mathbf{x}_B(\mathbf{x}_N), \mathbf{x}_N)$. Ideally we'd like to select \mathbf{B} to be easily factorizable (easier linear system). ex. 15.3

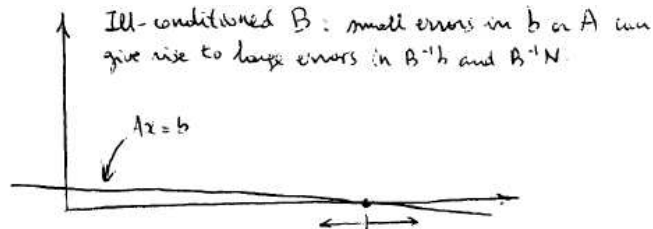
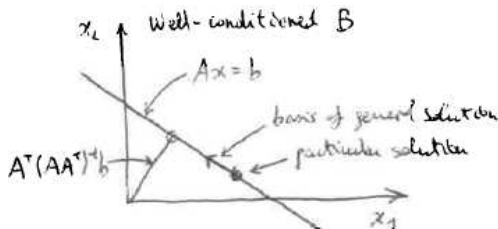
We can also write $\mathbf{x} = \mathbf{Y}\mathbf{b} + \mathbf{Z}\mathbf{x}_N$ with $\mathbf{Y} = \begin{pmatrix} \mathbf{B}^{-1} \\ \mathbf{0} \end{pmatrix}$, $\mathbf{Z} = \begin{pmatrix} -\mathbf{B}^{-1}\mathbf{N} \\ \mathbf{I} \end{pmatrix}$. We have:

- \mathbf{Z} has $n - m$ l.i. columns (due to \mathbf{I} being the lower block) and $\mathbf{AZ} = \mathbf{0} \Rightarrow \mathbf{Z}$ is a basis of the null space of \mathbf{A} .
- The columns of \mathbf{Y} and the columns of \mathbf{Z} are l.i. (pf.: $(\mathbf{Y} \ \mathbf{Z})\boldsymbol{\lambda} = \mathbf{0} \Rightarrow \boldsymbol{\lambda} = \mathbf{0}$); and $\text{null}(\mathbf{A}) \oplus \text{range}(\mathbf{A}^T) = \mathbb{R}^n \Rightarrow \mathbf{Y}$ is a basis of the range space of \mathbf{A}^T and $\mathbf{Y}\mathbf{b}$ is a particular solution of $\mathbf{Ax} = \mathbf{b}$.

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

$$\mathbf{x} = (\text{particular solution of } \mathbf{Ax} = \mathbf{b}) + (\text{general solution of } \mathbf{Ax} = \mathbf{0}).$$

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



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

- 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 optimization algorithm: a step is accepted if it leads to a sufficient reduction in the merit function.

- Unconstrained optimization: objective function = merit function. Also in feasible methods (which force all iterates to be feasible).
- A merit function $\phi(\mathbf{x}; \mu)$ is *exact* if $\exists \mu^* > 0$: $\mu \in (0, \mu^*] \Rightarrow$ any local solution \mathbf{x} of the optimization problem is a local minimizer of ϕ .
- Useful merit functions:

– ℓ_1 exact function:

$$\phi_1(\mathbf{x}; \mu) = f(\mathbf{x}) + \frac{1}{\mu} \sum_{i \in \mathcal{E}} |c_i(\mathbf{x})| + \frac{1}{\mu} \sum_{i \in \mathcal{I}} [c_i(\mathbf{x})]^- \quad [x]^- = \max(0, -x).$$

It is not differentiable. It is exact for $\frac{1}{\mu^*}$ = largest Lagrangian 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 $\mathbf{c}(\mathbf{x}) = \mathbf{0}$ exist:

$$\phi_F(\mathbf{x}; \mu) = f(\mathbf{x}) - \boldsymbol{\lambda}(\mathbf{x})^T \mathbf{c}(\mathbf{x}) + \frac{1}{2\mu} \|\mathbf{c}(\mathbf{x})\|_2^2$$

where $\mathbf{A}(\mathbf{x})$ is the Jacobian of $\mathbf{c}(\mathbf{x})$ and $\boldsymbol{\lambda}(\mathbf{x}) = (\mathbf{A}(\mathbf{x})\mathbf{A}(\mathbf{x})^T)^{-1} \mathbf{A}(\mathbf{x}) \nabla f(\mathbf{x})$ are the least-squares multipliers 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 $\boldsymbol{\lambda}(\mathbf{x})$, it is expensive to evaluate and may be ill-conditioned or not defined.

– *Augmented Lagrangian in \mathbf{x} and $\boldsymbol{\lambda}$* : when only equality constraints exist:

$$\mathcal{L}_A(\mathbf{x}, \boldsymbol{\lambda}; \mu) = f(\mathbf{x}) - \boldsymbol{\lambda}^T \mathbf{c}(\mathbf{x}) + \frac{1}{2\mu} \|\mathbf{c}(\mathbf{x})\|_2^2.$$

Here the iterates are $(\mathbf{x}_k, \boldsymbol{\lambda}_k)$, i.e., a step both in the primal and dual variables. A solution of the optimization problem is a stationary point of \mathcal{L}_A but in general not a minimizer.

Review: fundamentals of algorithms for nonlinear constrained optimization

16 Quadratic programming

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

$$\min_{\mathbf{x} \in \mathbb{R}^n} q(\mathbf{x}) = \frac{1}{2} \mathbf{x}^T \mathbf{G} \mathbf{x} + \mathbf{x}^T \mathbf{c} \text{ s.t. } \begin{cases} \mathbf{a}_i^T \mathbf{x} = b_i, & i \in \mathcal{E} \\ \mathbf{a}_i^T \mathbf{x} \geq b_i, & i \in \mathcal{I} \end{cases} \quad \mathbf{G}_{n \times n} \text{ symmetric}$$

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

- *Convex QP* $\Leftrightarrow \mathbf{G}$ *psd*. Local minimizer(s) also global; not much harder than LP.
- *Non-convex QP* $\Leftrightarrow \mathbf{G}$ *not psd*. Possibly several solutions.

Example: portfolio optimization

- n possible investments (bonds, stocks, etc.) with returns r_i , $i = 1, 2, \dots, n$.
- $\mathbf{r} = (r_1, \dots, r_n)^T$ is a random variable with mean $\boldsymbol{\mu}$ and covariance \mathbf{G} :
 - $\mu_i = \mathbb{E}\{r_i\}$
 - $g_{ij} = \mathbb{E}\{(r_i - \mu_i)(r_j - \mu_j)\}$ = 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 $\mathbf{x} \geq \mathbf{0}$. Return of the portfolio $\mathbf{R} = \mathbf{x}^T \mathbf{r}$, with:
 - mean $\mathbb{E}\{\mathbf{R}\} = \mathbf{x}^T \boldsymbol{\mu}$ (expected return)
 - variance $\mathbb{E}\{(\mathbf{R} - \mathbb{E}\{\mathbf{R}\})^2\} = \mathbf{x}^T \mathbf{G} \mathbf{x}$.
- Wanted portfolio: large expected return, small variance:

$$\max \mathbf{x}^T \boldsymbol{\mu} - \kappa \mathbf{x}^T \mathbf{G} \mathbf{x} \quad \text{s.t.} \quad \sum_{i=1}^k x_i = 1, \quad \mathbf{x} \geq \mathbf{0} \quad \left(\text{Is this convex QP?} \right)$$

where $\kappa \geq 0$ is set by the investor $\begin{cases} \text{conservative investor:} & \text{large } \kappa \\ \text{aggressive investor:} & \text{small } \kappa. \end{cases}$

- In practice, $\boldsymbol{\mu}$ and \mathbf{G} are guesstimated based on historical data and “intuition”.

Equality-constrained QP

- m equality constraints, no inequality constraints:

$$\min_{\mathbf{x}} q(\mathbf{x}) = \frac{1}{2} \mathbf{x}^T \mathbf{G} \mathbf{x} + \mathbf{x}^T \mathbf{c} \quad \text{s.t.} \quad \mathbf{A} \mathbf{x} = \mathbf{b} \quad \mathbf{A} \text{ full rank.}$$

- *KKT conditions*: a solution \mathbf{x}^* verifies (where $\boldsymbol{\lambda}^*$ is the Lagrange multiplier vector)

$$\begin{pmatrix} \mathbf{G} & -\mathbf{A}^T \\ \mathbf{A} & \mathbf{0} \end{pmatrix} \begin{pmatrix} \mathbf{x}^* \\ \boldsymbol{\lambda}^* \end{pmatrix} = \begin{pmatrix} -\mathbf{c} \\ \mathbf{b} \end{pmatrix} \quad \mathbf{x}^* = \mathbf{x} + \mathbf{p} \quad \underbrace{\begin{pmatrix} \mathbf{G} & \mathbf{A}^T \\ \mathbf{A} & \mathbf{0} \end{pmatrix}}_{\text{KKT matrix } \mathbf{K}} \begin{pmatrix} -\mathbf{p} \\ \boldsymbol{\lambda}^* \end{pmatrix} = \begin{pmatrix} \mathbf{g} \\ \mathbf{h} \end{pmatrix} \quad \begin{cases} \mathbf{h} = \mathbf{A} \mathbf{x} - \mathbf{b} \\ \mathbf{g} = \mathbf{c} + \mathbf{G} \mathbf{x} \\ \mathbf{p} = \mathbf{x}^* - \mathbf{x}. \end{cases} \quad \left(\text{What happens if } \mathbf{G} = \mathbf{0} \text{ (LP)?} \right)$$

Let $\mathbf{Z}_{n \times (n-m)} = (\mathbf{z}_1, \dots, \mathbf{z}_{n-m})$ be a basis of $\text{null}(\mathbf{A}) \Leftrightarrow \mathbf{A}\mathbf{Z} = \mathbf{0}$, $\text{rank}(\mathbf{Z}) = n - m$. Call $\mathbf{Z}^T \mathbf{G} \mathbf{Z}$ the *reduced Hessian* (= how the quadratic form looks like in the subspace $\mathbf{A}\mathbf{x} = \mathbf{b}$). Then, if \mathbf{A} has full row rank ($= m$) and $\mathbf{Z}^T \mathbf{G} \mathbf{Z}$ is pd:

- Lemma 16.1: \mathbf{K} is nonsingular (\Rightarrow unique $(\mathbf{x}^*, \boldsymbol{\lambda}^*)$). Proof
- Th. 16.3: $\text{inertia}(\mathbf{K}) = (n, m, 0)$ = number of $(+, -, 0)$ eigenvalues ($\Rightarrow \mathbf{K}$ always indefinite).
In general, $\text{inertia}(\mathbf{K}) = \text{inertia}(\mathbf{Z}^T \mathbf{G} \mathbf{Z}) + (m, m, 0)$.

- Classification of the solutions (assuming the KKT system has solutions $(\mathbf{x}^*, \boldsymbol{\lambda}^*)$): Ex. 16.2

1. Strong local minimizer at $\mathbf{x}^* \Leftrightarrow \mathbf{Z}^T \mathbf{G} \mathbf{Z}$ pd. Proof:
 - Either using the 2nd-order suffic. cond. (note \mathbf{Z} is a basis of $\mathcal{C}(\mathbf{x}^*, \boldsymbol{\lambda}^*) = \text{null}(\mathbf{A})$).
 - Or direct proof that \mathbf{x}^* is the unique, global solution (th. 16.2). Proof
2. Infinite solutions if $\mathbf{Z}^T \mathbf{G} \mathbf{Z}$ is psd and singular.
3. Unbounded if $\mathbf{Z}^T \mathbf{G} \mathbf{Z}$ indefinite.

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

Inequality-constrained QP

- *Optimality conditions*: Lagrangian function $\mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}) = \frac{1}{2} \mathbf{x}^T \mathbf{G} \mathbf{x} + \mathbf{x}^T \mathbf{c} - \sum_{i \in \mathcal{E} \cup \mathcal{I}} \lambda_i (\mathbf{a}_i^T \mathbf{x} - b_i)$. Active set at an optimal point \mathbf{x}^* : $\mathcal{A}(\mathbf{x}^*) = \{i \in \mathcal{E} \cup \mathcal{I} : \mathbf{a}_i^T \mathbf{x}^* = b_i\}$.

- KKT conditions (LICQ not required[?]):

$$\begin{aligned} \mathbf{G}\mathbf{x}^* + \mathbf{c} - \sum_{i \in \mathcal{A}(\mathbf{x}^*)} \lambda_i^* \mathbf{a}_i &= \mathbf{0} \\ \mathbf{a}_i^T \mathbf{x}^* &= b_i \quad \forall i \in \mathcal{A}(\mathbf{x}^*) \\ \mathbf{a}_i^T \mathbf{x}^* &\geq b_i \quad \forall i \in \mathcal{I} \setminus \mathcal{A}(\mathbf{x}^*) \\ \lambda_i^* &\geq 0 \quad \forall i \in \mathcal{I} \cap \mathcal{A}(\mathbf{x}^*). \end{aligned}$$

- Second-order conditions:

1. \mathbf{G} psd (convex QP) $\Rightarrow \mathbf{x}^*$ is a global minimizer (th. 16.4); and unique if \mathbf{G} pd. Proof
2. Strict, unique local minimizer at $\mathbf{x}^* \Leftrightarrow \mathbf{Z}^T \mathbf{G} \mathbf{Z}$ pd, where \mathbf{Z} is a nullspace basis for the active constraint Jacobian matrix $(\mathbf{a}_i^T)_{i \in \mathcal{A}(\mathbf{x}^*)}$.
3. If \mathbf{G} is not psd, there may be more than one strict local minimizer at which the 2nd-order conditions hold (non-convex, or indefinite QP); harder to solve. Determining whether a feasible point is a global minimizer is NP-hard. fig. 16.1

- *Degeneracy* is one of the following situations, which can cause problems for the algorithms: ex. p. 466

- Active constraint gradients are l.d. at the solution, e.g. (but not necessarily) if more than n constraints are active at the solution \Rightarrow numerically difficult to compute \mathbf{Z} .
- Strict complementary condition fails: $\lambda_i^* = 0$ for some active index $i \in \mathcal{A}(\mathbf{x}^*)$ (the constraint is weakly active) \Rightarrow numerically difficult to determine whether a weakly active constraint is active.

- Three types of methods: active-set, gradient-projection, interior-point.

Active-set methods for convex QP

- Convex QP: any local solution is also global.
- They are the most effective methods for small- to medium-scale problems; efficient detection of unboundedness and infeasibility; accurate estimate (typically) of the optimal active set.
- Remember the brute-force approach to solving the KKT systems for all combinations of active constraints: if we knew the optimal active set $\mathcal{A}(\mathbf{x}^*)$ (\equiv the active set at the optimal point \mathbf{x}^*), we could find the solution of the equality-constrained QP problem $\min_{\mathbf{x}} q(\mathbf{x})$ s.t. $\mathbf{a}_i^T \mathbf{x} = b_i$, $i \in \mathcal{A}(\mathbf{x}^*)$. Goal: to determine this set.
- *Active-set method*: start from a guess of the optimal active set; if not optimal, drop one index from $\mathcal{A}(\mathbf{x})$ and add a new index (using gradient and Lag. mult. information); repeat.
 - The simplex method for LP is an active-set method.
 - QP active-set methods may have iterates that aren't 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 objective function q .

Primal active-set method

- Compute a feasible initial iterate \mathbf{x}_0 (some techniques available; pp. 473–474); subsequent iterates will remain feasible.
- Move to next iterate $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{p}_k$: obtained by solving an equality-constrained quadratic subproblem. The constraint set, called *working set* \mathcal{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 \mathbf{a}_i of the constraints in \mathcal{W}_k must be l.i. The quadratic subproblem (solvable as in sec. 16.1):

$$\min_{\mathbf{p}} q(\mathbf{x}_k + \mathbf{p}) \text{ s.t. } \mathcal{W}_k \stackrel{?}{\iff} \min_{\mathbf{p}} \frac{1}{2} \mathbf{p}^T \mathbf{G} \mathbf{p} + (\mathbf{G} \mathbf{x}_k + \mathbf{c})^T \mathbf{p} \text{ s.t. } \mathbf{a}_i^T \mathbf{p} = 0 \ \forall i \in \mathcal{W}_k.$$

Call \mathbf{p}_k the solution of the subproblem. Then:

- Use $\alpha_k = 1$ if possible: if $\mathbf{x}_k + \mathbf{p}_k$ satisfies all constraints (not just those in \mathcal{W}_k) and is thus feasible, set $\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{p}_k$.
- Otherwise, choose α_k as the largest value in $[0, 1)$ for which all constraints are satisfied. Note that $\mathbf{x}_k + \alpha_k \mathbf{p}_k$ satisfies $\mathbf{a}_i^T \mathbf{x}_{k+1} = b_i \ \forall \alpha_k \in \mathbb{R} \ \forall i \in \mathcal{W}_k$, so we need only worry about the constraints not in \mathcal{W}_k ; the result is trivial to compute (similar to the choice of α_k in interior-point methods for LP):

$$\alpha_k \stackrel{?}{=} \min \left(1, \min_{\substack{i \notin \mathcal{W}_k \\ \mathbf{a}_i^T \mathbf{p}_k < 0}} \frac{b_i - \mathbf{a}_i^T \mathbf{x}_k}{\mathbf{a}_i^T \mathbf{p}_k} \right). \quad (16.41)$$

The constraints (typically just one) for which the minimum above is achieved are called *blocking constraints*. The new working set: if $\alpha_k < 1$ (\Leftrightarrow the step along \mathbf{p}_k was blocked by some constraint not in \mathcal{W}_k) then add one of the blocking constraints to \mathcal{W}_k ; otherwise keep \mathcal{W}_k . Note that it is possible that $\alpha_k = 0$; this happens when $\mathbf{a}_i^T \mathbf{p}_k < 0$ for a constraint i that is active at \mathbf{x}_k but not a member of \mathcal{W}_k .

- Iterating this process (where we keep adding blocking constraints and moving \mathbf{x}_k) we must reach a point $\hat{\mathbf{x}}$ that minimizes q over its current working set $\hat{\mathcal{W}}$, or equivalently $\mathbf{p} = \mathbf{0}$ 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 $\sum_{i \in \hat{\mathcal{W}}} \mathbf{a}_i \hat{\lambda}_i = \mathbf{G}\hat{\mathbf{x}} + \mathbf{c}$. So if $\hat{\lambda}_j < 0$ for some $j \in \hat{\mathcal{W}} \cap \mathcal{I}$ then we drop constant j from the working set (since by making this constraint inactive we can decrease q while remaining feasible; th. 16.5) 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).
- 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 \mathbf{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.
- The linear systems can be solved efficiently by updating factorizations (in the KKT matrix, \mathbf{G} is constant and \mathbf{A} changes by one row at most at each step).
- Extension to nonconvex QP possible but complicated.

Algorithm 16.3 (Active-set method for convex QP)

Ex. 16.4

Compute a feasible starting point \mathbf{x}_0

$\mathcal{W}_0 \leftarrow$ subset of the active constraints at \mathbf{x}_0

for $k = 0, 1, 2, \dots$

$\mathbf{p}_k = \arg\min_{\mathbf{p}} \frac{1}{2} \mathbf{p}^T \mathbf{G} \mathbf{p} + (\mathbf{G} \mathbf{x}_k + \mathbf{c})^T \mathbf{p}$ s.t. $\mathbf{a}_i^T \mathbf{p} = 0 \ \forall i \in \mathcal{W}_k$ Equality-constr. quadratic subproblem

if $\mathbf{p}_k = \mathbf{0}$

Solve for $\hat{\lambda}_i$: $\sum_{i \in \mathcal{W}_k} \mathbf{a}_i \hat{\lambda}_i = \mathbf{G}(\mathbf{x}_k + \mathbf{p}_k) + \mathbf{c}$ Compute Lagrange mult. for subproblem

if $\hat{\lambda}_i \geq 0 \ \forall i \in \mathcal{W}_k \cap \mathcal{I}$

stop with solution $\mathbf{x}^* = \mathbf{x}_k$ All KKT conditions hold

else

$j \leftarrow \arg \min_{j \in \mathcal{W}_k \cap \mathcal{I}} \hat{\lambda}_j, \quad \mathcal{W}_{k+1} \leftarrow \mathcal{W}_k \setminus \{j\}$ Remove from the working set that ineq. constr. having the most negative Lag. mult.

$\mathbf{x}_{k+1} \leftarrow \mathbf{x}_k$

end

else

compute $\alpha_k = \min(1, \min \dots)$ from (16.41) $\mathbf{p}_k \neq \mathbf{0}$: we can move \mathbf{x}_k and decrease q

$\mathbf{x}_{k+1} \leftarrow \mathbf{x}_k + \alpha_k \mathbf{p}_k$ Longest step in $[0, 1]$

if $\alpha_k < 1$

$\mathcal{W}_{k+1} \leftarrow \mathcal{W}_k \cup \{\text{one blocking constraint}\}$ There are blocking constraints

else

$\mathcal{W}_{k+1} \leftarrow \mathcal{W}_k$

end

end

end

( Does this algorithm become the simplex method for $\mathbf{G} = \mathbf{0}$ (LP)?)

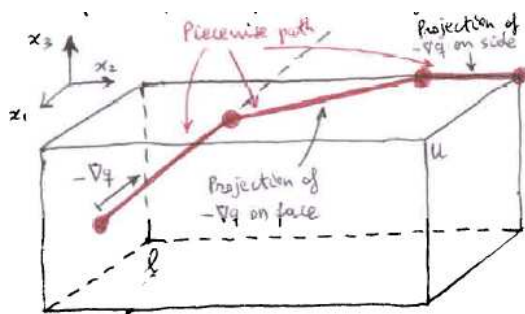
The gradient projection method

- 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:

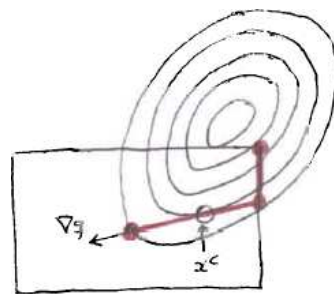
$$\min_{\mathbf{x}} q(\mathbf{x}) = \frac{1}{2} \mathbf{x}^T \mathbf{G} \mathbf{x} + \mathbf{x}^T \mathbf{c} \quad \text{s.t. } \mathbf{l} \leq \mathbf{x} \leq \mathbf{u}$$

$\mathbf{x}, \mathbf{l}, \mathbf{u} \in \mathbb{R}^n$, \mathbf{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 \mathbf{x}° (trivial to obtain); all iterates remain feasible.
- Each iteration consists of two stages; assume current point is \mathbf{x} (which is feasible):
 1. *Find the Cauchy point \mathbf{x}^c* : this is the first minimizer along the steepest descent direction $-\nabla q = -(\mathbf{G}\mathbf{x} + \mathbf{c})$ 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. 486–489).



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




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

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

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

( Does this method converge in a finite number of iterations?)

Interior-point methods

- Appropriate for large problems.
- 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 (exe. 16.21 considers also equality ones):

$$\min_{\mathbf{x}} q(\mathbf{x}) = \frac{1}{2} \mathbf{x}^T \mathbf{G} \mathbf{x} + \mathbf{x}^T \mathbf{c} \text{ s.t. } \mathbf{A} \mathbf{x} \geq \mathbf{b} \quad \text{with } \mathbf{G} \text{ symmetric pd, } \mathbf{A}_{m \times n}.$$

Write KKT conditions, *then* introduce surplus vector $\mathbf{y} = \mathbf{A} \mathbf{x} - \mathbf{b} \geq \mathbf{0}$ (saves m Lag. mult.)[?].

Since the problem is convex, the KKT conditions are not only necessary but also sufficient.

We find minimizers of the QP by finding roots of the KKT system:

$$\begin{array}{l} \text{only} \\ \text{addition} \\ \text{wrt LP} \end{array} \left\{ \begin{array}{l} \mathbf{G} \mathbf{x} - \mathbf{A}^T \boldsymbol{\lambda} = -\mathbf{c} \\ \mathbf{A} \mathbf{x} - \mathbf{y} = \mathbf{b} \\ y_i \lambda_i = 0 \\ i = 1, \dots, n \\ \mathbf{y}, \boldsymbol{\lambda} \geq \mathbf{0} \end{array} \right\} \begin{array}{l} \text{system of } n + 2m \text{ equations} \\ \text{for } n + 2m \text{ unknowns } \mathbf{x}, \mathbf{y}, \boldsymbol{\lambda} \\ \text{(mildly nonlinear because of } y_i \lambda_i) \end{array} \Leftrightarrow \mathbf{F}(\mathbf{x}, \mathbf{y}, \boldsymbol{\lambda}) = \begin{pmatrix} \mathbf{G} \mathbf{x} - \mathbf{A}^T \boldsymbol{\lambda} + \mathbf{c} \\ \mathbf{A} \mathbf{x} - \mathbf{y} - \mathbf{b} \\ \mathbf{Y} \boldsymbol{\Lambda} \mathbf{e} \end{pmatrix} = \mathbf{0}$$

$$\mathbf{Y} = \text{diag}(y_i), \quad \boldsymbol{\Lambda} = \text{diag}(\lambda_i), \quad \mathbf{e} = \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix}.$$

- Central path $\mathcal{C} = \{(\mathbf{x}_\tau, \mathbf{y}_\tau, \boldsymbol{\lambda}_\tau) : \mathbf{F}(\mathbf{x}_\tau, \mathbf{y}_\tau, \boldsymbol{\lambda}_\tau) = \begin{pmatrix} \mathbf{0} \\ \mathbf{0} \\ \tau \mathbf{e} \end{pmatrix}, \tau > 0\} \Leftrightarrow$ solve perturbed KKT system with $y_i \lambda_i = \tau$. Given a current iterate $(\mathbf{x}, \mathbf{y}, \boldsymbol{\lambda})$ with $\mathbf{y}, \boldsymbol{\lambda} > \mathbf{0}$, define the duality measure $\mu = \mathbf{y}^T \boldsymbol{\lambda} / m$ (closeness to the boundary) and the centering parameter $\sigma \in [0, 1]$.
- Newton-like step toward point $(\mathbf{x}_{\sigma\mu}, \mathbf{y}_{\sigma\mu}, \boldsymbol{\lambda}_{\sigma\mu})$ on the central path:

$$\underbrace{\begin{pmatrix} \mathbf{G} & \mathbf{0} & -\mathbf{A}^T \\ \mathbf{A} & -\mathbf{I} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\Lambda} & \mathbf{Y} \end{pmatrix}}_{\text{Jacobian of } \mathbf{F}} \underbrace{\begin{pmatrix} \Delta \mathbf{x} \\ \Delta \mathbf{y} \\ \Delta \boldsymbol{\lambda} \end{pmatrix}}_{\text{step}} = \underbrace{\begin{pmatrix} -\mathbf{r}_c \\ -\mathbf{r}_b \\ -\boldsymbol{\Lambda} \mathbf{Y} \mathbf{e} + \sigma \mu \mathbf{e} \end{pmatrix}}_{-\mathbf{F}(\mathbf{x}, \mathbf{y}, \boldsymbol{\lambda})} \quad \begin{array}{l} \mathbf{r}_c = \mathbf{G} \mathbf{x} - \mathbf{A}^T \boldsymbol{\lambda} + \mathbf{c} \\ \mathbf{r}_b = \mathbf{A} \mathbf{x} - \mathbf{y} - \mathbf{b} \end{array}$$

$$\begin{pmatrix} \mathbf{x}^{k+1} \\ \mathbf{y}^{k+1} \\ \boldsymbol{\lambda}^{k+1} \end{pmatrix} \leftarrow \begin{pmatrix} \mathbf{x}^k \\ \mathbf{y}^k \\ \boldsymbol{\lambda}^k \end{pmatrix} + \alpha_k \begin{pmatrix} \Delta \mathbf{x}^k \\ \Delta \mathbf{y}^k \\ \Delta \boldsymbol{\lambda}^k \end{pmatrix} \text{ choosing } \alpha_k \in [0, 1] \text{ such that } \mathbf{y}^{k+1}, \boldsymbol{\lambda}^{k+1} > \mathbf{0}.$$

- Likewise, we can extend the path-following methods (by defining a neighborhood $\mathcal{N}_{-\infty}(\gamma)$) and Mehrotra's predictor-corrector algorithm.
- Major computation: solving the linear system, more costly than for LP because of \mathbf{G} .
- As in the comparison between the simplex and interior-point methods for LP, for QP:
 - Active-set methods: large number of inexpensive steps; more complicated to implement; preferable if an estimate of the solution is available (“warm start”).
 - Interior-point methods: small number of expensive steps; simpler to implement; the sparsity pattern in the linear system is constant.

Review: quadratic programming

17 Penalty and augmented Lagrangian methods

The quadratic penalty method

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

- Define the following *quadratic-penalty function* with *penalty parameter* $\mu > 0$:

$$Q(\mathbf{x}; \mu) = \underbrace{f(\mathbf{x})}_{\text{objective function}} + \underbrace{\frac{\mu}{2} \sum_{i \in \mathcal{E}} c_i^2(\mathbf{x}) + \frac{\mu}{2} \sum_{i \in \mathcal{I}} ([c_i(\mathbf{x})]^-)^2}_{\text{one term per constraint, which is positive when } \mathbf{x} \text{ violates } c_i \text{ and 0 otherwise}} \quad [y]^- = \max(-y, 0).$$

- Define a sequence of unconstrained minimization subproblems $\min_{\mathbf{x}} Q(\mathbf{x}; \mu_k)$ given a sequence $\mu_k \xrightarrow{k \rightarrow \infty} \infty$. By driving μ to ∞ we penalize the constraint violations with increasing severity, thereby forcing the minimizer of Q closer to the feasible region of the constrained problem.

ex. 17.1

eq. 17.5

Algorithmic framework 17.1 (Quadratic-penalty method)

Given tolerance $\tau_0 > 0$, starting penalty parameter $\mu_0 > 0$, starting point \mathbf{x}_0^s

for $k = 0, 1, 2, \dots$

Find an approximate minimizer \mathbf{x}_k of $Q(\mathbf{x}; \mu_k)$ $\begin{cases} \text{starting at } \mathbf{x}_k^s \\ \text{terminating when } \|\nabla_{\mathbf{x}} Q(\mathbf{x}; \mu_k)\| \leq \tau_k \end{cases}$

if final convergence test satisfied \Rightarrow **stop** with approximate solution \mathbf{x}_k

Choose new $\begin{cases} \text{penalty parameter } \mu_{k+1} > \mu_k \\ \text{starting point } \mathbf{x}_{k+1}^s \\ \text{tolerance } \tau_{k+1} \in (0, \tau_k) \end{cases}$

end

- Smoothness of the penalty terms:

- Equality constraints: c_i^2 has at least[?] as many derivatives as c_i
 \Rightarrow use derivative-based techniques for unconstrained optimization.

- Inequality constraints: $([c_i]^-)^2$ can be less smooth than c_i .

Ex.: for $x_1 \geq 0$, $([x_1]^-)^2 = \min(0, x_1)^2$ has a discontinuous second derivative.

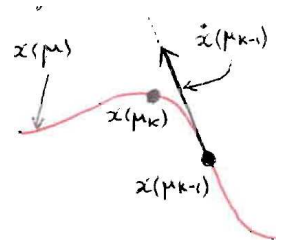
- Choice of starting point \mathbf{x}_k^s for the min. of $Q(\mathbf{x}; \mu_k)$: extrapolate \mathbf{x}_k :

- From previous iterates $\mathbf{x}_{k-1}, \mathbf{x}_{k-2}, \mathbf{x}_{k-3}, \dots$ (in particular, $\mathbf{x}_k^s = \mathbf{x}_{k-1}$).

- Using the tangent to the path $\{\mathbf{x}(\mu), \mu > 0\}$: $\mathbf{x}_k^s = \mathbf{x}_{k-1} + (\mu_k - \mu_{k-1})\dot{\mathbf{x}}$.

The path tangent $\dot{\mathbf{x}} = \frac{d(\mathbf{x}(\mu))}{d\mu}$ can be obtained by total differentiation of $\nabla_{\mathbf{x}} Q(\mathbf{x}; \mu) = \mathbf{0}$ wrt μ :

$$\mathbf{0} = \frac{d\nabla_{\mathbf{x}} Q(\mathbf{x}(\mu); \mu)}{d\mu} = \overbrace{\nabla_{\mathbf{xx}}^2 Q(\mathbf{x}; \mu)}^{\partial \nabla_{\mathbf{x}} Q / \partial \mathbf{x}} \overbrace{\dot{\mathbf{x}}}^{d\mathbf{x}/d\mu} + \overbrace{\frac{1}{\mu} (\nabla_{\mathbf{x}} Q(\mathbf{x}; \mu) - \nabla f(\mathbf{x}))}^{\partial \nabla_{\mathbf{x}} Q / \partial \mu} \left. \vphantom{\frac{d\nabla_{\mathbf{x}} Q(\mathbf{x}(\mu); \mu)}{d\mu}} \right\} \text{Linear system for vector } \dot{\mathbf{x}}.$$



- Choice of $\{\mu_k\}$: adaptive, e.g. if minimizing $Q(\mathbf{x}; \mu_k)$ was:

- expensive: modest increase, e.g. $\mu_{k+1} = 1.5\mu_k$

- cheap: larger increase, e.g. $\mu_{k+1} = 10\mu_k$.

- Choice of $\{\tau_k\}$: $\tau_k \xrightarrow{k \rightarrow \infty} 0$ (the minimization is carried out progressively more accurately).

- *Convergence*: assume $\mu_k \rightarrow \infty$ and consider equality constraints only.

– *Th. 17.1*: \mathbf{x}_k global min. of $Q(\mathbf{x}; \mu_k) \Rightarrow \mathbf{x}_k \rightarrow$ global solution of the constr. problem.

Impractical: requires *global* minimization.

– *Th. 17.2*: if $\tau_k \rightarrow 0$, $\mathbf{x}_k \rightarrow \mathbf{x}^*$:

Proof

* \mathbf{x}^* infeasible $\Rightarrow \mathbf{x}^*$ is a stationary point of $\|\mathbf{c}(\mathbf{x})\|^2$; or

* \mathbf{x}^* feasible, $\nabla c_i(\mathbf{x}^*)$ l.i. $\Rightarrow -\mu_k c_i(\mathbf{x}_k) \rightarrow \lambda_i^* \forall i \in \mathcal{E}$, $(\mathbf{x}^*, \boldsymbol{\lambda}^*)$ satisfy the KKT cond.

Practical: only needs tolerances $\tau_k \rightarrow 0$; gives Lag. mult. estimates; but can converge to infeasible or to any KKT points.

Practical problems

- Q may be unbounded below for some values of μ (ex. in eq. 17.5) \Rightarrow safeguard.
- The penalty function doesn't look quadratic around its minimizer except very close to it (see contours in fig. 17.2).
- Even if $\nabla^2 f(\mathbf{x}^*)$ is well-conditioned, the Hessian $\nabla_{\mathbf{xx}}^2 Q(\mathbf{x}; \mu_k)$ becomes arbitrarily ill-conditioned as $\mu_k \rightarrow \infty$. Consider equality constraints only and define $\mathbf{A}(\mathbf{x})^T = (\nabla c_i(\mathbf{x}))_{i \in \mathcal{E}}$ (matrix of constraint gradients; usually $\text{rank}(\mathbf{A}) < n$):

$$\nabla_{\mathbf{xx}}^2 Q(\mathbf{x}; \mu_k) \stackrel{?}{=} \nabla^2 f(\mathbf{x}) + \sum_{i \in \mathcal{E}} \mu_k c_i(\mathbf{x}) \nabla^2 c_i(\mathbf{x}) + \mu_k \mathbf{A}(\mathbf{x})^T \mathbf{A}(\mathbf{x}).$$

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

$$\nabla_{\mathbf{xx}}^2 Q(\mathbf{x}; \mu_k) \approx \underbrace{\nabla_{\mathbf{xx}}^2 \mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}^*)}_{\text{independent of } \mu_k} + \underbrace{\mu_k \mathbf{A}(\mathbf{x})^T \mathbf{A}(\mathbf{x})}_{\text{rank } |\mathcal{E}| \text{ with nonzero eigenvalues of } \mathcal{O}(\mu_k)}$$

where $\mathcal{L}(\mathbf{x}, \boldsymbol{\lambda})$ is the Lagrangian function (and usually $|\mathcal{E}| < n$). Unconstrained optimization methods have problems with ill-conditioning. For Newton's method we can apply the following reformulation that avoids the ill-conditioning (\mathbf{p} solves both systems[?]):

<u>Newton step</u> $\underbrace{\nabla_{\mathbf{xx}}^2 Q(\mathbf{x}; \mu_k)}_{\text{ill-conditioned} \Rightarrow \text{large error in } \mathbf{p}} \mathbf{p} = -\nabla_{\mathbf{x}} Q(\mathbf{x}; \mu)$	<u>Introducing dummy vector $\boldsymbol{\zeta} = \mu \mathbf{A}(\mathbf{x}) \mathbf{p}$</u> $\underbrace{\begin{pmatrix} \nabla f^2(\mathbf{x}) + \sum_{i \in \mathcal{E}} \mu_k c_i(\mathbf{x}) \nabla^2 c_i(\mathbf{x}) & \mathbf{A}(\mathbf{x})^T \\ \mathbf{A}(\mathbf{x}) & -\frac{1}{\mu_k} \mathbf{I} \end{pmatrix}}_{\text{well-conditioned as } \mu_k \rightarrow \infty^?; \text{ cf. lemma 16.1}} \begin{pmatrix} \mathbf{p} \\ \boldsymbol{\zeta} \end{pmatrix} = \begin{pmatrix} -\nabla_{\mathbf{x}} Q(\mathbf{x}; \mu) \\ \mathbf{0} \end{pmatrix}.$
--	--

This system has dimension $n + |\mathcal{E}|$ rather than n , and is a regularized version of the SQP system (18.6) (the “ $-\frac{1}{\mu_k} \mathbf{I}$ ” term makes the matrix nonsingular even if $\mathbf{A}(\mathbf{x})$ is rank-deficient).

The augmented Lagrangian method is more effective, as it delays the onset of ill-conditioning.

Exact penalty functions

- *Exact penalty function* $\phi(\mathbf{x}; \mu)$: $\exists \mu^* > 0$: $\forall \mu > \mu^*$, any local solution \mathbf{x} of the constrained problem is a local minimizer of ϕ . So we need a single unconstrained minimization of $\phi(\mathbf{x}; \mu)$ for such a $\mu > \mu^*$.

- The quadratic-penalty and log-barrier functions are not exact, so they need $\mu \rightarrow \infty$.

- The ℓ_1 exact penalty function

ex. 17.2

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

ex. 17.3

is exact for μ^* = largest Lagrange multiplier (in absolute value) associated with an optimal solution (th. 17.3). Algorithms based on minimizing ϕ_1 need:

- Rules for adjusting μ to ensure $\mu > \mu^*$ (note minimizing ϕ for large μ is difficult).
- Special techniques to deal with the fact that ϕ_1 is not differentiable at any \mathbf{x} for which $c_i(\mathbf{x}) = 0$ for some $i \in \mathcal{E} \cup \mathcal{I}$ (and such \mathbf{x} must encountered).
- Any penalty function of the type $\phi(\mathbf{x}; \mu) = f(\mathbf{x}) + \mu h(c_1(\mathbf{x}))$ (where $h(0) = 0$ and $h(y) \geq 0 \forall y \in \mathbb{R}$) must be nonsmooth.

Proof
p. 513

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.

Equality constraints only

- In the quadratic-penalty method, the minimizer \mathbf{x}_k of $Q(\mathbf{x}; \mu_k)$ satisfies $c_i(\mathbf{x}_k) \approx -\frac{\lambda_i^*}{\mu_k} \forall i \in \mathcal{E}$ (from th. 17.2) and so $c_i(\mathbf{x}_k) \xrightarrow{\mu_k \rightarrow \infty} 0$. Idea: redefine Q so that its minimizer \mathbf{x}_k satisfies $c_i(\mathbf{x}_k) \approx 0$ (i.e., the subproblem solution better satisfies the equality constraint); this way we will 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 and considering the argument $\boldsymbol{\lambda}$ as estimates for the Lagrange multipliers at a solution (or, as a quadratic-penalty function with objective function $f(\mathbf{x}) - \sum_{i \in \mathcal{E}} \lambda_i c_i(\mathbf{x})$):

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

Near \mathbf{x}^* , the minimizer \mathbf{x}_k of \mathcal{L}_A for $\boldsymbol{\lambda} = \boldsymbol{\lambda}^k$ satisfies $c_i(\mathbf{x}_k) \approx -\frac{1}{\mu_k}(\lambda_i^* - \lambda_i^k) \forall i \in \mathcal{E}$.

Pf.: $\mathbf{0} = \nabla_{\mathbf{x}} \mathcal{L}_A(\mathbf{x}_k, \boldsymbol{\lambda}^k; \mu_k) = \nabla f(\mathbf{x}_k) - \sum_{i \in \mathcal{E}} (\lambda_i^k - \mu_k c_i(\mathbf{x}_k)) \nabla c_i(\mathbf{x}_k) \Rightarrow \lambda_i^* \approx \lambda_i^k - \mu_k c_i(\mathbf{x}_k)$ (KKT cond.).

So if $\boldsymbol{\lambda}^k$ is close to the optimal multiplier vector $\boldsymbol{\lambda}^*$ then $\|\mathbf{c}(\mathbf{x}_k)\|$ will be much smaller than $\frac{1}{\mu_k}$ rather than just proportional to $\frac{1}{\mu_k}$.

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

ex. 17.4

Note that $-\mu_k c_i(\mathbf{x}_k) \rightarrow \begin{cases} \lambda_i^* & \text{quadratic-penalty method} \\ 0 & \text{augmented Lagrangian method.} \end{cases}$

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

- Choice of starting point \mathbf{x}_k^s for the minimization of $\mathcal{L}_A(\mathbf{x}, \boldsymbol{\lambda}^k; \mu_k)$ less critical now (less ill-conditioning), so we can simply take $\mathbf{x}_{k+1}^s \leftarrow \mathbf{x}_k$.
- *Convergence:*
 - *Th. 17.5:* $(\mathbf{x}^*, \boldsymbol{\lambda}^*) = (\text{local solution, Lagrange multiplier})$ at which KKT + LICQ + 2nd-order sufficient conditions hold (\equiv well-behaved solution) $\Rightarrow \exists \bar{\mu} > 0: \forall \mu \geq \bar{\mu}, \mathbf{x}^*$ is a strict local minimizer of $\mathcal{L}_A(\mathbf{x}, \boldsymbol{\lambda}^*; \mu)$.

Pf.: KKT + 2nd-order cond. for constrained problem \Rightarrow KKT + 2nd-order cond. for unconstrained problem $\min_{\mathbf{x}} \mathcal{L}_A$.

Thus \mathcal{L}_A is an exact penalty function for the optimal Lagrange multiplier $\boldsymbol{\lambda} = \boldsymbol{\lambda}^*$ and, if we knew the latter, we would not need to take $\mu \rightarrow \infty$. In practice, we need to estimate $\boldsymbol{\lambda}^*$ over iterates and drive μ sufficiently large; if $\boldsymbol{\lambda}^k$ is close to $\boldsymbol{\lambda}^*$ or if μ_k is large, then \mathbf{x}_k will be close to \mathbf{x}^* (the quadratic-penalty method gives only one option: increase μ_k).

Note that $\mathcal{L}_A(\mathbf{x}, \mathbf{0}; \mu) = Q(\mathbf{x}; \mu)$.

Extension to inequality constraints Three useful formulations:

1. *Bound-constrained formulation:* use slack variables \mathbf{s} to turn inequalities into equalities and bounds: $c_i(\mathbf{x}) \geq 0 \Rightarrow c_i(\mathbf{x}) - s_i = 0, s_i \geq 0 \forall i \in \mathcal{I}$. Consider then the bound-constrained problem (\mathbf{x} absorbs the slacks, and l_i or u_i can be $-\infty$ or $+\infty$):

$$\min_{\mathbf{x} \in \mathbb{R}^n} f(\mathbf{x}) \text{ s.t. } \begin{cases} c_i(\mathbf{x}) = 0, i = 1, \dots, m & \text{equalities} \\ \mathbf{l} \leq \mathbf{x} \leq \mathbf{u} & \text{bounds.} \end{cases}$$

Bound-constrained Lagrangian: augmented Lagrangian using equality constraints only but subject to bounds:

$$\min_{\mathbf{x}} \mathcal{L}_A(\mathbf{x}, \boldsymbol{\lambda}; \mu) = f(\mathbf{x}) - \sum_{i=1}^m \lambda_i c_i(\mathbf{x}) + \frac{\mu}{2} \sum_{i=1}^m c_i^2(\mathbf{x}) \text{ s.t. } \mathbf{l} \leq \mathbf{x} \leq \mathbf{u}.$$

Solve this subproblem approximately, update $\boldsymbol{\lambda}$ and μ , repeat.

The subproblem may be solved with the (nonlinear) gradient-projection method; implemented in the LANCELOT package.

2. *Linearly-constrained formulation:* in the bound-constrained problem, solve the subproblem of minimizing the (augmented) Lagrangian subject to linearization of the constraints:

$$\min_{\mathbf{x}} F_k(\mathbf{x}) \text{ s.t. } \begin{cases} c_i(\mathbf{x}_k) + \nabla c_i(\mathbf{x}_k)^T (\mathbf{x} - \mathbf{x}_k) = 0, i = 1, \dots, m \\ \mathbf{l} \leq \mathbf{x} \leq \mathbf{u} \end{cases}$$

where

- Augmented Lagrangian $F_k(\mathbf{x}) = f(\mathbf{x}) - \overbrace{\sum_{i=1}^m \lambda_i^k \bar{c}_i^k(\mathbf{x})}^{\text{Lagrangian}} + \overbrace{\frac{\mu}{2} \sum_{i=1}^m (\bar{c}_i^k(\mathbf{x}))^2}^{\text{Quadratic penalty}}.$
- Current Lag. mult. estimate $\boldsymbol{\lambda}^k = \text{Lag. mult. for the linearized constraints at } k-1$.
- $\bar{c}_i^k(\mathbf{x}) = c_i(\mathbf{x}) - (c_i(\mathbf{x}_k) + \nabla c_i(\mathbf{x}_k)^T (\mathbf{x} - \mathbf{x}_k)) = \text{true} - \text{linearized} = \text{“second-order } c_i \text{ remainder”}.$

Similar to SQP but with a nonlinear objective (hard subproblem); implemented in MINOS package; particularly effective when most of the constraints are linear.

3. *Unconstrained formulation*: consider again the general constrained problem (without equality constraints for simplicity) and introduce slack variables: $c_i(\mathbf{x}) \geq 0 \Rightarrow c_i(\mathbf{x}) - s_i = 0$, $s_i \geq 0 \forall i \in \mathcal{I}$. Consider the bound-constrained augmented Lagrangian

$$\min_{\mathbf{x}, \mathbf{s}} \mathcal{L}_A(\mathbf{x}, \mathbf{s}, \boldsymbol{\lambda}; \mu) = f(\mathbf{x}) - \sum_{i \in \mathcal{I}} \lambda_i (c_i(\mathbf{x}) - s_i) + \frac{\mu}{2} \sum_{i \in \mathcal{I}} (c_i(\mathbf{x}) - s_i)^2 \text{ s.t. } s_i \geq 0 \forall i \in \mathcal{I}.$$

Apply “coordinate descent” first in \mathbf{s} , then in \mathbf{x} :

- In \mathbf{s} : $\min_{\mathbf{s}} \mathcal{L}_A$ s.t. $s_i \geq 0 \forall i \in \mathcal{I} \Rightarrow$ Solution: $s_i \stackrel{?}{=} \max(0, c_i(\mathbf{x}) - \frac{1}{\mu_k} \lambda_i^k) \forall i \in \mathcal{I}$ because \mathcal{L}_A is a convex, separable quadratic form on \mathbf{s} .
- In \mathbf{x} : substitute the solution s_i in \mathcal{L}_A to obtain:

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

Now, iterate the following:

- Solve approximately the unconstrained problem $\min_{\mathbf{x}} \mathcal{L}_A(\mathbf{x}, \boldsymbol{\lambda}^k; \mu_k)$.

Note that ψ doesn't have a second derivative[?] when $\mu t = \sigma \Leftrightarrow \mu c_i(\mathbf{x}) = \lambda_i$ 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 $c_i(\mathbf{x}^*)$ is 0, so the iterates should stay away from points at which $\mu_k c_i(\mathbf{x}_k) = \lambda_i^k$. Thus it is safe to use Newton's method. For a weakly active constraint $\mu c_i(\mathbf{x}^*) = \lambda_i^* = 0$ does hold.

- Update the Lagrange multipliers as $\lambda_i^{k+1} \leftarrow \max(\lambda_i^k - \mu_k c_i(\mathbf{x}_k), 0) \forall i \in \mathcal{I}$ (since $\lambda_i \geq 0$ for the KKT conditions to hold at the solution).
- Update μ_k , etc.

Review: penalty and augmented Lagrangian methods

18 Sequential quadratic programming (SQP)

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

Local SQP method

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

Linearize the objective function and constraints to obtain the *QP subproblem*:

$$\min_{\mathbf{p}} \frac{1}{2} \mathbf{p}^T \nabla_{\mathbf{xx}}^2 \mathcal{L}_k \mathbf{p} + \nabla f_k^T \mathbf{p} + f_k \quad \text{s.t.} \quad \begin{cases} \nabla c_i(\mathbf{x}_k)^T \mathbf{p} + c_i(\mathbf{x}_k) = 0, & i \in \mathcal{E} \\ \nabla c_i(\mathbf{x}_k)^T \mathbf{p} + c_i(\mathbf{x}_k) \geq 0, & i \in \mathcal{I} \end{cases}$$

where $\nabla_{\mathbf{xx}}^2 \mathcal{L}(\mathbf{x}_k, \boldsymbol{\lambda}_k)$ is the Hessian of the Lagrangian $\mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}) = f(\mathbf{x}) - \boldsymbol{\lambda}^T \mathbf{c}(\mathbf{x})$, for a given $\boldsymbol{\lambda}$ estimate.

($\not\Rightarrow$ Why not use $\nabla^2 f$ instead of $\nabla_{\mathbf{xx}}^2 \mathcal{L}(\mathbf{x}_k, \boldsymbol{\lambda}_k)$? See later.)

($\not\Rightarrow$ Using $\nabla_{\mathbf{x}} \mathcal{L}(\mathbf{x}_k, \boldsymbol{\lambda}_k)$ instead of ∇f_k changes nothing for equality constraints. Why?)

Algorithm 18.1 (local SQP algorithm):

Given initial $\mathbf{x}_0, \boldsymbol{\lambda}_0$

for $k = 0, 1, 2, \dots$

 Evaluate $f_k, \nabla f_k, c_i(\mathbf{x}_k), \nabla c_i(\mathbf{x}_k), \nabla_{\mathbf{xx}}^2 \mathcal{L}(\mathbf{x}_k, \boldsymbol{\lambda}_k)$

$(\mathbf{p}_k, \boldsymbol{\lambda}_{k+1}) \leftarrow$ (solution, Lagrange multiplier) of QP subproblem

$\mathbf{x}_{k+1} \leftarrow \mathbf{x}_k + \mathbf{p}_k$

if convergence test satisfied \Rightarrow **stop** with approximate solution $(\mathbf{x}_{k+1}, \boldsymbol{\lambda}_{k+1})$

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 $\min_{\mathbf{x}} f(\mathbf{x})$ s.t. $\mathbf{c}(\mathbf{x}) = \mathbf{0}$ with $\mathbf{c}(\mathbf{x})^T = (c_1(\mathbf{x}), \dots, c_m(\mathbf{x}))$ and $\mathbf{A}(\mathbf{x})^T = (\nabla c_1(\mathbf{x}), \dots, \nabla c_m(\mathbf{x}))$:

(i) The solution $(\mathbf{p}_k^*, \boldsymbol{\lambda}_k^*)$ of the QP subproblem satisfies:

$$\begin{cases} \nabla_{\mathbf{xx}}^2 \mathcal{L}_k \mathbf{p}_k + \nabla f_k - \mathbf{A}_k^T \boldsymbol{\lambda}_k = \mathbf{0} \\ \mathbf{A}_k \mathbf{p}_k + \mathbf{c}_k = \mathbf{0} \end{cases} \Leftrightarrow \begin{pmatrix} \nabla_{\mathbf{xx}}^2 \mathcal{L}_k & -\mathbf{A}_k^T \\ \mathbf{A}_k & \mathbf{0} \end{pmatrix} \begin{pmatrix} \mathbf{p}_k \\ \boldsymbol{\lambda}_k \end{pmatrix} = \begin{pmatrix} -\nabla f_k \\ -\mathbf{c}_k \end{pmatrix}.$$

(ii) KKT system for the problem: $\mathbf{F}(\mathbf{x}, \boldsymbol{\lambda}) = \begin{pmatrix} \nabla_{\mathbf{x}} \mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}) \\ \mathbf{c}(\mathbf{x}) \end{pmatrix} = \mathbf{0}$ (where $\nabla_{\mathbf{x}} \mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}) = \nabla f(\mathbf{x}) - \mathbf{A}(\mathbf{x})^T \boldsymbol{\lambda}$), for which Newton's method (for root finding) result in a step

$$\begin{pmatrix} \mathbf{x}_{k+1} \\ \boldsymbol{\lambda}_{k+1} \end{pmatrix} = \begin{pmatrix} \mathbf{x}_k \\ \boldsymbol{\lambda}_k \end{pmatrix} + \begin{pmatrix} \mathbf{p}_k \\ \mathbf{p}_{\boldsymbol{\lambda}} \end{pmatrix} \quad \text{where} \quad \underbrace{\begin{pmatrix} \nabla_{\mathbf{xx}}^2 \mathcal{L}_k & -\mathbf{A}_k^T \\ \mathbf{A}_k & \mathbf{0} \end{pmatrix}}_{\text{Jacobian of } \mathbf{F} \text{ at } (\mathbf{x}_k, \boldsymbol{\lambda}_k)^T} \begin{pmatrix} \mathbf{p}_k \\ \mathbf{p}_{\boldsymbol{\lambda}} \end{pmatrix} = \underbrace{\begin{pmatrix} -\nabla f_k + \mathbf{A}_k^T \boldsymbol{\lambda}_k \\ -\mathbf{c}_k \end{pmatrix}}_{-\mathbf{F}(\mathbf{x}_k, \boldsymbol{\lambda}_k)}.$$

(i) \equiv (ii), since the two linear systems have the same solution (define $\mathbf{l}_k = \mathbf{p}_{\boldsymbol{\lambda}} - \boldsymbol{\lambda}_k$).

- Assumptions (recall lemma 16.1 in ch. 16 about equality-constrained QP):

- The constraint Jacobian \mathbf{A}_k has full row rank (LICQ)
- $\nabla_{\mathbf{xx}}^2 \mathcal{L}_k$ is pd on the tangent space of the constraints ($\mathbf{d}^T \nabla_{\mathbf{xx}}^2 \mathcal{L}_k \mathbf{d} > 0 \forall \mathbf{d} \neq \mathbf{0}, \mathbf{A}_k \mathbf{d} = \mathbf{0}$)

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

Do these assumptions hold? They do *locally* (near the solution) if the problem solution satisfies the 2nd-order sufficient conditions. Then Newton's method *converges quadratically*.

- Th. 18.1: $(\mathbf{x}^*, \boldsymbol{\lambda}^*)$ local solution at which KKT + LICQ + 2nd-order + strict complementarity hold \Rightarrow if $(\mathbf{x}_k, \boldsymbol{\lambda}_k)$ is sufficiently close to $(\mathbf{x}^*, \boldsymbol{\lambda}^*)$, there is a local solution of the SQP subproblem whose active set \mathcal{A}_k is $\mathcal{A}(\mathbf{x}^*)$.

At that time, the SQP subproblem correctly identifies the active set at the solution and SQP behaves like Newton steps for an equality-constrained problem.

- To ensure *global convergence* (\equiv from remote starting points), Newton's method needs to be modified (just as in the unconstrained optimization 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 \mathbf{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, e.g. the linearization of inequality constraints may produce an infeasible subproblem

Ex.: linearizing $x \leq 1, x^2 \geq 0$ at $x_k = 3$ results in $3 + p \leq 1, 9 + 6p \geq 0$ which is inconsistent.

19 Interior-point methods for nonlinear programming

- Considered the most powerful algorithms (together with SQP) for large-scale nonlinear programming.
- Extension of the interior-point methods for LP and QP.
- Terms “interior-point methods” and “barrier methods” used interchangeably (but different historical origin).

Interior-point methods as homotopy methods

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

Deriving the KKT conditions (Lagrangian $\mathcal{L}(\mathbf{x}, \mathbf{y}, \mathbf{z}) = f(\mathbf{x}) - \mathbf{y}^T \mathbf{c}_{\mathcal{E}}(\mathbf{x}) - \mathbf{z}^T \mathbf{c}_{\mathcal{I}}(\mathbf{x})$) and introducing in them slacks ($c_i(\mathbf{x}) \geq 0 \Rightarrow c_i(\mathbf{x}) - s_i = 0, s_i \geq 0 \forall i \in \mathcal{I}$) and a parameter $\mu > 0$, we obtain the *perturbed KKT conditions*:

$$\begin{aligned} \nabla f(\mathbf{x}) - \mathbf{A}_{\mathcal{E}}^T(\mathbf{x}) \mathbf{y} - \mathbf{A}_{\mathcal{I}}^T(\mathbf{x}) \mathbf{z} &= \mathbf{0} & \mathbf{c}_{\mathcal{E}}(\mathbf{x}), \mathbf{c}_{\mathcal{I}}(\mathbf{x}): \text{constraint vectors} \\ \mathbf{S} \mathbf{z} - \mu \mathbf{e} &= \mathbf{0} & \mathbf{A}_{\mathcal{E}}, \mathbf{A}_{\mathcal{I}}: \text{constraint Jacobians} \\ \mathbf{c}_{\mathcal{E}}(\mathbf{x}) &= \mathbf{0} & \mathbf{y}, \mathbf{z}: \text{Lagrange multipliers} \\ \mathbf{c}_{\mathcal{I}}(\mathbf{x}) - \mathbf{s} &= \mathbf{0} & \mathbf{S} = \text{diag}(s_i), \mathbf{e} = \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix}. \\ \mathbf{s}, \mathbf{z} &\geq \mathbf{0} \end{aligned}$$

We solve (approximately) this nonlinear system of equations for a sequence $(\mu_k) \xrightarrow{k \rightarrow \infty} 0$ while preserving $\mathbf{s}, \mathbf{z} > \mathbf{0}$, and require the iterates to decrease a merit function (to help converge not just to a KKT point but to a minimizer). This follows a *primal-dual central path* $(\mathbf{x}(\mu), \mathbf{s}(\mu), \mathbf{y}(\mu), \mathbf{z}(\mu))$ that steers through the interior of the primal-dual feasible set, avoiding solutions that satisfy the nonlinear system of equations but not $\mathbf{s}, \mathbf{z} > \mathbf{0}$, and converges to a solution $(\mathbf{x}^*, \mathbf{s}^*, \mathbf{y}^*, \mathbf{z}^*)$ as $\mu \rightarrow 0^+$ under some conditions. The Newton step is

$$\begin{pmatrix} \nabla_{\mathbf{xx}}^2 \mathcal{L} & \mathbf{0} & -\mathbf{A}_{\mathcal{E}}^T(\mathbf{x}) & -\mathbf{A}_{\mathcal{I}}^T(\mathbf{x}) \\ \mathbf{0} & \mathbf{Z} & \mathbf{0} & \mathbf{S} \\ \mathbf{A}_{\mathcal{E}}(\mathbf{x}) & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{A}_{\mathcal{I}}(\mathbf{x}) & -\mathbf{I} & \mathbf{0} & \mathbf{0} \end{pmatrix} \begin{pmatrix} \mathbf{p}_{\mathbf{x}} \\ \mathbf{p}_{\mathbf{s}} \\ \mathbf{p}_{\mathbf{y}} \\ \mathbf{p}_{\mathbf{z}} \end{pmatrix} = - \begin{pmatrix} \nabla f(\mathbf{x}) - \mathbf{A}_{\mathcal{E}}^T(\mathbf{x}) \mathbf{y} - \mathbf{A}_{\mathcal{I}}^T(\mathbf{x}) \mathbf{z} \\ \mathbf{S} \mathbf{z} - \mu \mathbf{e} \\ \mathbf{c}_{\mathcal{E}}(\mathbf{x}) \\ \mathbf{c}_{\mathcal{I}}(\mathbf{x}) - \mathbf{s} \end{pmatrix}.$$

Note that no ill-conditioning arises if the 2nd-order and strict complementarity conditions hold at \mathbf{x}^* , because then either s_i or z_i are nonzero, so the second row of the Jacobian has full row rank. The system can be rewritten in a form that is symmetric (eq. 19.12) but introduces ill-conditioning.

Practical versions of interior-point methods follow line-search or trust-region implementations.

Interior-point methods as barrier methods

In the perturbed KKT conditions, eliminate \mathbf{s} and $\mathbf{x}: i \in \mathcal{I}: \begin{cases} s_i z_i - \mu = 0 \\ c_i(\mathbf{x}) - s_i = 0 \end{cases} \Rightarrow z_i = \frac{\mu}{c_i(\mathbf{x})}$, and substitute in $\nabla_{\mathbf{x}} \mathcal{L}(\mathbf{x}, \mathbf{y}, \mathbf{z})$:

$$\nabla f(\mathbf{x}) - \mathbf{A}_{\mathcal{E}}^T(\mathbf{x}) \mathbf{y} - \sum_{i \in \mathcal{I}} \frac{\mu}{c_i(\mathbf{x})} \nabla c_i(\mathbf{x}) = \mathbf{0} \Leftrightarrow \min_{\mathbf{x}} P(\mathbf{x}; \mu) = f(\mathbf{x}) - \mu \sum_{i \in \mathcal{I}} \log c_i(\mathbf{x}) \text{ s.t. } \mathbf{c}_{\mathcal{E}}(\mathbf{x}) = \mathbf{0}.$$

That is, the interior-point method can be seen as minimizing the *log-barrier function* $P(\mathbf{x}; \mu)$ (subject to the equalities) and taking $\mu \rightarrow 0$.

The primal log-barrier method

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

ex. 19.1

$$P(\mathbf{x}; \mu) = \underbrace{f(\mathbf{x})}_{\text{objective function}} - \underbrace{\mu \sum_{i \in \mathcal{I}} \log c_i(\mathbf{x})}_{\text{log-barrier function}} \begin{cases} \text{infinite everywhere except in } \mathcal{F}^0 \\ \text{smooth inside } \mathcal{F}^0 \\ \text{approaches } \infty \text{ as } \mathbf{x} \text{ approaches the boundary of } \mathcal{F}^0. \end{cases}$$

- The minimizer $\mathbf{x}(\mu)$ of $P(\mathbf{x}, \mu)$ approaches a solution of the constrained problem as $\mu \rightarrow 0^+$.
- Algorithmic framework 19.5 (Primal log-barrier method)*: as for the quadratic penalty but choosing a new barrier parameter $\mu_{k+1} \in (0, \mu_k)$ instead of a new penalty parameter. Similar choices of tolerances, adaptive decrease of μ_k , starting point \mathbf{x}_k^s for the minimization of $P(\mathbf{x}; \mu)$, etc. (Is \mathbf{x}_k^s feasible if extrapolating with the path tangent?)
- The log-barrier function is smooth (if f, c_i are), so, if $\mathbf{x}(\mu) \in \mathcal{F}^0$, no constraints are active and we can use derivative-based techniques for unconstrained optimization.

Convergence

- For convex programs: global convergence.
Th.: $f, \{-c_i, i \in \mathcal{I}\}$ convex functions, $\mathcal{F}^0 \neq \emptyset \Rightarrow$
 - For any $\mu > 0$, $P(\mathbf{x}; \mu)$ is convex in \mathcal{F}^0 and attains a minimizer $\mathbf{x}(\mu)$ (not necessarily unique) on \mathcal{F}^0 ; any local minimizer $\mathbf{x}(\mu)$ is also global.
 - If the set of solutions of the constrained optimization problem is nonempty and bounded and if $\{\mu_k\}$ is a decreasing sequence with $\mu_k \rightarrow 0 \Rightarrow \{\mathbf{x}(\mu_k)\}$ converges to a solution \mathbf{x}^* and $f(\mathbf{x}(\mu_k)) \rightarrow f^*, P(\mathbf{x}(\mu_k)) \rightarrow f^*, P(\mathbf{x}(\mu_k); \mu_k) \rightarrow f^*$.

If there are no solutions or the solution set is unbounded, the theorem may not apply.

- For general inequality-constrained problems: local convergence.
Th.: $\mathcal{F}^0 \neq \emptyset, (\mathbf{x}^*, \boldsymbol{\lambda}^*) = (\text{local solution, Lagrange multiplier})$ at which KKT + LICQ + 2nd-order sufficient conditions hold (\equiv well-behaved solution) \Rightarrow
 - For all sufficiently small $\mu, \exists!$ continuously differentiable function $\mathbf{x}(\mu)$: $\mathbf{x}(\mu)$ is a local minimizer of $P(\mathbf{x}; \mu)$ and $\nabla_{\mathbf{xx}}^2 P(\mathbf{x}; \mu)$ is pd.
 - $(\mathbf{x}(\mu), \boldsymbol{\lambda}(\mu)) \xrightarrow{\mu \rightarrow 0} (\mathbf{x}^*, \boldsymbol{\lambda}^*)$ where $\lambda_i(\mu) = \frac{\mu}{c_i(\mathbf{x}(\mu))}, i \in \mathcal{I}$.

This means there may be sequences of minimizers of $P(\mathbf{x}; \mu)$ that don't converge to a solution as $\mu \rightarrow 0$.

Relation between the minimizers of $P(\mathbf{x}; \mu)$ and a solution $(\mathbf{x}^*, \boldsymbol{\lambda}^*)$: at a minimizer $\mathbf{x}(\mu)$:

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

which is KKT condition a) for the constrained problem ($\nabla_{\mathbf{xx}}^2 \mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}) = \mathbf{0}$). As for the other KKT conditions at $\mathbf{x}(\mu), \boldsymbol{\lambda}(\mu)$; b) $(c_i(\mathbf{x}) \geq 0, i \in \mathcal{I})$ and c) $(\lambda_i \geq 0, i \in \mathcal{I})$ also hold since $c_i(\mathbf{x}(\mu)) > 0$; only the complementarity condition d) fails: $\lambda_i c_i(\mathbf{x}) = \mu > 0$; but it holds as $\mu \rightarrow 0$. The path $\mathcal{C}_p = \{\mathbf{x}(\mu): \mu > 0\}$ is called *primal central path*, and is the projection on the primal variables of the primal-dual central path from the interior-point version.

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

$$\begin{aligned}\nabla_{\mathbf{x}} P(\mathbf{x}; \mu) &= \nabla f(\mathbf{x}) - \sum_{i \in \mathcal{I}} \frac{\mu}{c_i(\mathbf{x})} \nabla c_i(\mathbf{x}) \\ \nabla_{\mathbf{xx}}^2 P(\mathbf{x}; \mu) &= \nabla^2 f(\mathbf{x}) - \sum_{i \in \mathcal{I}} \frac{\mu}{c_i(\mathbf{x})} \nabla^2 c_i(\mathbf{x}) + \sum_{i \in \mathcal{I}} \frac{\mu}{c_i^2(\mathbf{x})} \nabla c_i(\mathbf{x}) \nabla c_i(\mathbf{x})^T.\end{aligned}$$

Near a minimizer $\mathbf{x}(\mu)$ with μ small, from the earlier theorem we have that the optimal Lagrange multipliers can be estimated as $\lambda_i^* \approx \frac{\mu}{c_i(\mathbf{x})}$, so

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

The Newton step can be reformulated as in the quadratic-penalty method to avoid ill-conditioning, and it should be implemented with line-search or trust-region strategy to remain (well) strictly feasible.

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

Splitting an equality constraint $c_i(\mathbf{x}) = 0$ as two inequalities $c_i(\mathbf{x}) \geq 0$, $-c_i(\mathbf{x}) \geq 0$ doesn't work[?], but we can combine the quadratic penalty and the log-barrier:

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

This has similar aspects to the quadratic penalty and barrier methods: algorithm = successive reduction of μ alternated with approximate minimization of B wrt \mathbf{x} ; ill-conditioned $\nabla_{\mathbf{xx}}^2 B$ when μ is small; etc.

To find an initial point which is strictly feasible wrt the inequality constraints, introduce slack variables s_i , $i \in \mathcal{I}$:

$$\begin{aligned}\min_{\mathbf{x}, \mathbf{s}} f(\mathbf{x}) \quad \text{s.t.} \quad & \begin{cases} c_i(\mathbf{x}) = 0, & i \in \mathcal{E} \\ c_i(\mathbf{x}) - s_i = 0, & i \in \mathcal{I} \\ s_i \geq 0, & i \in \mathcal{I} \end{cases} \Rightarrow \\ B(\mathbf{x}, \mathbf{s}; \mu) &= f(\mathbf{x}) - \mu \sum_{i \in \mathcal{I}} \log s_i + \frac{1}{2\mu} \sum_{i \in \mathcal{E}} c_i^2(\mathbf{x}) + \frac{1}{2\mu} \sum_{i \in \mathcal{I}} (c_i(\mathbf{x}) - s_i)^2.\end{aligned}$$

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

Review: interior-point methods for nonlinear programming

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
 - homotopy, path-following
 - 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 minimize 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, e.g.:
 - 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 increase the penalty parameter μ in quadratic penalty, augmented Lagrangian

Given your particular optimization 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 (slacks, elimination, redundant constraints, ...).
- Try to guess good initial iterates.
- Recognize the type of optimization problem: smooth? Separable? LP? QP? Convex? Multiple optima?
- Evaluate costs (time & memory):
 - computing the Hessian
 - solving linear systems (sparse?)

A Math review

Error analysis and floating-point arithmetic

- Floating-point representation of $x \in \mathbb{R}$: $\text{fl}(x) = 2^e \sum_{i=1}^t d_i 2^{-i}$ (t bits for fractional part, $d_1 = 1$, rest of bits for exponent and sign).
- Unit roundoff $u = 2^{-t-1}$ ($\approx 1.1 \times 10^{-16}$ for 64-bit IEEE double precision). Matlab: `eps = 2u`.
- Any x with $|x| \in [2^L, 2^U]$ (where $e \in \{L+1, \dots, U\}$) can be approximated with relative accuracy u : $\frac{|\text{fl}(x) - x|}{|x|} \leq u \Leftrightarrow \text{fl}(x) = x(1 + \epsilon)$ with roundoff error $|\epsilon| \leq u$ (so x and $\text{fl}(x)$ agree to at least 15 decimal digits).
- Roundoff errors accumulate during floating-point operations. An algorithm is *stable* if errors do not grow unbound. A particularly nasty case is *cancellation*: the relative error in computing $x - y$ if x and y are very close is $\lesssim 2u |x| / |x - y| \Rightarrow$ precision loss; or, if x and y are accurate to k digits and they agree in the first \bar{k} , their difference contains only about $k - \bar{k}$ significant digits. So, avoid taking the difference of similar floating-point numbers.

Functions, derivatives, sets

- $f : \mathbb{R}^n \rightarrow \mathbb{R}$

$$\begin{array}{ll} \text{Gradient at } \mathbf{x} \text{ of } f \text{ (} n \times 1 \text{ vector):} & \nabla f(\mathbf{x}) = \begin{pmatrix} \frac{\partial f}{\partial x_1} \\ \vdots \\ \frac{\partial f}{\partial x_n} \end{pmatrix}; \quad \text{Hessian (} n \times n \text{ symmetric matrix):} & \nabla^2 f(\mathbf{x}) : \begin{pmatrix} \frac{\partial^2 f}{\partial x_i \partial x_j} \end{pmatrix}. \end{array}$$

- Directional derivative along direction $\mathbf{p} \in \mathbb{R}^n$:

$$\lim_{\epsilon \rightarrow 0} \frac{f(\mathbf{x} + \epsilon \mathbf{p}) - f(\mathbf{x})}{\epsilon} = \nabla f(\mathbf{x})^T \mathbf{p} \quad [\text{Pf.: Taylor's th.}]$$

- Mean value th.:

- $\phi : \mathbb{R} \rightarrow \mathbb{R}$ continuously differentiable, $\alpha_1 > \alpha_0$:
 $\phi(\alpha_1) - \phi(\alpha_0) = \phi'(\xi)(\alpha_1 - \alpha_0)$ for some $\xi \in (\alpha_0, \alpha_1)$.
- $f : \mathbb{R}^n \rightarrow \mathbb{R}$ cont. diff., for any $\mathbf{p} \in \mathbb{R}^n$:
 $f(\mathbf{x} + \mathbf{p}) - f(\mathbf{x}) = \nabla f(\mathbf{x} + \alpha \mathbf{p})^T \mathbf{p}$ for some $\alpha \in (0, 1)$.

- Taylor's th.:

- $f : \mathbb{R}^n \rightarrow \mathbb{R}$ cont. diff., $\mathbf{p} \in \mathbb{R}^n$. Then
 $f(\mathbf{x} + \mathbf{p}) = f(\mathbf{x}) + \nabla f(\mathbf{x} + t\mathbf{p})^T \mathbf{p}$ for some $t \in (0, 1)$. [Mean value th.]
- If f is twice cont. diff. then:
 $f(\mathbf{x} + \mathbf{p}) = f(\mathbf{x}) + \nabla f(\mathbf{x})^T \mathbf{p} + \frac{1}{2} \mathbf{p}^T \nabla^2 f(\mathbf{x} + t\mathbf{p}) \mathbf{p}$ for some $t \in (0, 1)$.
 $\nabla f(\mathbf{x} + \mathbf{p}) = \nabla f(\mathbf{x}) + \int_0^1 \nabla^2 f(\mathbf{x} + t\mathbf{p}) \mathbf{p} dt$.

- $\mathbf{r} : \mathbb{R}^n \rightarrow \mathbb{R}^m$, Jacobian matrix at \mathbf{x} of \mathbf{r} ($m \times n$ matrix): $\mathbf{J}(\mathbf{x}) : \left(\frac{\partial r_i}{\partial x_j} \right)_{ij}$.

Taylor's th.: $\mathbf{r}(\mathbf{x} + \mathbf{p}) = \mathbf{r}(\mathbf{x}) + \int_0^1 \mathbf{J}(\mathbf{x} + t\mathbf{p}) \mathbf{p} dt$ (if \mathbf{J} is cont. in the domain of interest).

- Cone: set \mathcal{F} verifying: $\mathbf{x} \in \mathcal{F} \Rightarrow \alpha \mathbf{x} \in \mathcal{F} \forall \alpha > 0$. Ex.: $\left\{ \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} : x_1 > 0, x_2 \geq 0 \right\}$.
Cone generated by $\{\mathbf{x}_1, \dots, \mathbf{x}_m\} \subset \mathbb{R}^n$: $\{\mathbf{x} \in \mathbb{R}^n : \mathbf{x} = \sum_{i=1}^m \alpha_i \mathbf{x}_i, \alpha_i \geq 0 \forall i = 1, \dots, m\}$.
Convex hull of $\{\mathbf{x}_1, \dots, \mathbf{x}_m\} \subset \mathbb{R}^n$: $\{\mathbf{x} \in \mathbb{R}^n : \mathbf{x} = \sum_{i=1}^m \alpha_i \mathbf{x}_i, \alpha_i \geq 0 \forall i = 1, \dots, m, \sum_{i=1}^m \alpha_i = 1\}$.

Matrices

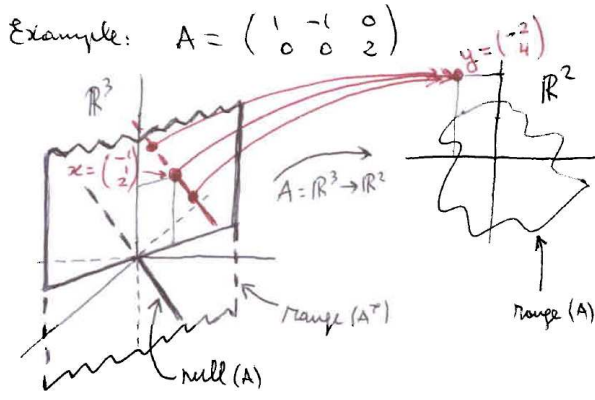
- *Positive definite matrix* $\mathbf{B} \Leftrightarrow \mathbf{p}^T \mathbf{B} \mathbf{p} > 0 \ \forall \mathbf{p} \neq 0$ (pd). *Positive semidefinite* if ≥ 0 (psd).
- *Matrix norm* induced by a vector norm: $\|\mathbf{A}\| = \sup_{\mathbf{x} \neq 0} \frac{\|\mathbf{A}\mathbf{x}\|}{\|\mathbf{x}\|}$.
Ex. $\|\mathbf{A}\|_2 = \text{largest s.v.}$ $\sigma_{\max}(\mathbf{A}) = \text{largest eigenvalue } \lambda_{\max}(\sqrt{\mathbf{A}^T \mathbf{A}})$.
- *Condition number* of a square nonsingular matrix \mathbf{A} : $\kappa(\mathbf{A}) = \|\mathbf{A}\| \|\mathbf{A}^{-1}\| \geq 1$ where $\|\cdot\|$ is any matrix norm. Ex. $\kappa_2(\mathbf{A}) = \sigma_{\max}/\sigma_{\min}$. Square linsys $\mathbf{A}\mathbf{x} = \mathbf{b}$, perturb to $\tilde{\mathbf{A}}\tilde{\mathbf{x}} = \tilde{\mathbf{b}} \Rightarrow \frac{\|\mathbf{x} - \tilde{\mathbf{x}}\|}{\|\mathbf{x}\|} \approx \kappa(\mathbf{A}) \left(\frac{\|\mathbf{A} - \tilde{\mathbf{A}}\|}{\|\mathbf{A}\|} + \frac{\|\mathbf{b} - \tilde{\mathbf{b}}\|}{\|\mathbf{b}\|} \right)$ so ill-conditioned problem \Leftrightarrow large $\kappa(\mathbf{A})$. Ex. in p. 616
- *Eigenvalues and eigenvectors* of a real matrix: $\mathbf{A}\mathbf{u} = \lambda\mathbf{u} = \begin{cases} \lambda \in \mathbb{C} : & \text{eigenvalue} \\ \mathbf{u} \in \mathbb{R}^n : & \text{eigenvector} \end{cases}$.
Matrix $\begin{cases} \text{symmetric:} & \text{all } \lambda \in \mathbb{R}; \text{ eigenvectors of different eigenvalues are } \perp \\ \text{nonsingular:} & \text{all } \lambda \neq 0 \\ \text{pd:} & \text{all } \lambda > 0 \quad (\text{nd: all } \lambda < 0) \\ \text{psd:} & \text{all } \lambda \geq 0 \quad (\text{nsd: all } \lambda \leq 0) \\ \text{not definite:} & \text{mixed-sign } \lambda \end{cases}$
- *Spectral theorem*: \mathbf{A} symmetric, real with eigenvectors $\mathbf{u}_1, \dots, \mathbf{u}_n \in \mathbb{R}^n$ associated with eigenvalues $\lambda_1, \dots, \lambda_n \in \mathbb{R} \Rightarrow \mathbf{A} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^T = \sum_{i=1}^n \lambda_i \mathbf{u}_i \mathbf{u}_i^T$ where $\mathbf{U} = (\mathbf{u}_1 \dots \mathbf{u}_n)$ is orthogonal and $\mathbf{\Lambda} = \text{diag}(\lambda_1, \dots, \lambda_n)$. In other words, a symmetric real matrix can be diagonalized in terms of its eigenvalues and eigenvectors.
Spectrum of \mathbf{A} = eigenvalues of \mathbf{A} .

Linear independence, subspaces

- $\mathbf{v}_1, \dots, \mathbf{v}_k \in \mathbb{R}^n$ are l.i. if $\forall \lambda_1, \dots, \lambda_k \in \mathbb{R}: \lambda_1 \mathbf{v}_1 + \dots + \lambda_k \mathbf{v}_k = \mathbf{0} \Rightarrow \lambda_1 = \dots = \lambda_k = 0$.
- Vectors $\mathbf{u}, \mathbf{v} \neq 0$ are orthogonal ($\mathbf{u} \perp \mathbf{v}$) iff $\mathbf{u}^T \mathbf{v} = 0$ ($= \|\mathbf{u}\| \|\mathbf{v}\| \cos(\mathbf{u}, \mathbf{v})$).
Orthogonal matrix: $\mathbf{U}^{-1} = \mathbf{U}^T$. For the Euclidean norm: $\|\mathbf{U}\mathbf{x}\| = \|\mathbf{x}\|$.
- $\text{span}\{\mathbf{v}_1, \dots, \mathbf{v}_k\} = \{\mathbf{x} : \mathbf{x} = \lambda_1 \mathbf{v}_1 + \dots + \lambda_k \mathbf{v}_k \text{ for } \lambda_1, \dots, \lambda_k \in \mathbb{R}\}$ is the set of all vectors that are linear combination of $\mathbf{v}_1, \dots, \mathbf{v}_k$, or the linear subspace spanned by $\mathbf{v}_1, \dots, \mathbf{v}_k$.
- If $\mathbf{v}_1, \dots, \mathbf{v}_k$ are l.i. then they are a basis of $\text{span}\{\mathbf{v}_1, \dots, \mathbf{v}_k\}$, which has dimension k .

Subspaces of a real matrix $\mathbf{A}_{m \times n} \Leftrightarrow$ linear mapping $\mathbb{R}^n \rightarrow \mathbb{R}^m$

- *Null space*: $\text{null}(\mathbf{A}) = \{\mathbf{x} \in \mathbb{R}^n : \mathbf{A}\mathbf{x} = \mathbf{0}\}$, i.e., the subspace associated with eigenvalue 0.
- *Range space*: $\text{range}(\mathbf{A}) = \{\mathbf{y} \in \mathbb{R}^m : \mathbf{y} = \mathbf{A}\mathbf{x} \text{ for some } \mathbf{x} \in \mathbb{R}^n\}$.
- *Fundamental theorem of linear algebra*: $\text{null}(\mathbf{A}) \oplus \text{range}(\mathbf{A}^T) = \mathbb{R}^n$ (direct sum: $\mathbf{x} \in \mathbb{R}^n \Rightarrow \exists! \mathbf{u}, \mathbf{v} \in \mathbb{R}^n : \mathbf{u} \in \text{null}(\mathbf{A}), \mathbf{v} \in \text{range}(\mathbf{A}^T), \mathbf{x} = \mathbf{u} + \mathbf{v}$). Also: $\text{null}(\mathbf{A}) \cap \text{range}(\mathbf{A}^T) = \{\mathbf{0}\}$, $\text{null}(\mathbf{A}) \perp \text{range}(\mathbf{A}^T)$, $\dim(\text{null}(\mathbf{A})) + \underbrace{\dim(\text{range}(\mathbf{A}^T))}_{=\dim(\text{range}(\mathbf{A}))=\text{rank}(\mathbf{A})} = n$.



Subspace	Basis	Dimension
$\text{null}(\mathbf{A})$	$\begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix}$	1
$\text{range}(\mathbf{A})$	$\begin{pmatrix} 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \end{pmatrix}$	2
$\text{range}(\mathbf{A}^T)$	$\begin{pmatrix} 1 \\ -1 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \\ 2 \end{pmatrix}$	2

Least squares, pseudoinverse and singular value decomposition

Linear system $\mathbf{Ax} = \mathbf{b}$ with m equations, n unknowns, assume \mathbf{A} is full-rank:

- $m = n$: unique solution $\mathbf{x} = \mathbf{A}^{-1}\mathbf{b}$.
- $m < n$: underconstrained, infinite solutions $\mathbf{x} = \mathbf{x}_0 + \mathbf{u}$ $\begin{cases} \mathbf{x}_0 = \text{particular solution} \\ \mathbf{u} \in \text{null}(\mathbf{A}), \text{ with } \dim(\text{null}(\mathbf{A})) = n - m \end{cases}$.

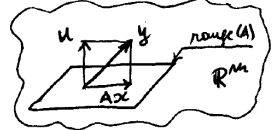
Minimum norm solution ($\min \|\mathbf{x}\|^2$ s.t. $\mathbf{Ax} = \mathbf{b}$): $\mathbf{x} = \mathbf{A}^+\mathbf{b} = \mathbf{A}^T(\mathbf{AA}^T)^{-1}\mathbf{b}$.

Homogeneous system: $\mathbf{Ax} = \mathbf{0}$:

- Unit-norm best approximation ($\min \|\mathbf{Ax}\|^2$ s.t. $\|\mathbf{x}\|^2 = 1$): \mathbf{x} = minor eigenvector of $\mathbf{A}^T\mathbf{A}$.
- $\min \|\mathbf{Ax}\|^2$ s.t. $\|\mathbf{Bx}\|^2 = 1$: \mathbf{x} = minor generalized eigenvector of $\mathbf{A}^T\mathbf{A}, \mathbf{B}^T\mathbf{B}$.
- $m > n$: overconstrained, no solution in general; instead, define LSQ solution as $\min \|\mathbf{Ax} - \mathbf{b}\|^2 \Rightarrow \mathbf{x} = \mathbf{A}^+\mathbf{b} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{b}$.

$\mathbf{AA}^+ = \mathbf{A}(\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T$ is the orthogonal projection on $\text{range}(\mathbf{A})$ (Pf.: write $\mathbf{y} \in \mathbb{R}^m$ as $\mathbf{y} = \mathbf{Ax} + \mathbf{u}$ where $\mathbf{x} \in \mathbb{R}^n$ and $\mathbf{u} \perp \mathbf{Ax} \forall \mathbf{x} \in \mathbb{R}^n$ (i.e., $\mathbf{u} \in \text{null}(\mathbf{A}^T)$). Then $\mathbf{A}(\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{y} = \mathbf{Ax}$).

Likewise, $\mathbf{A}^+\mathbf{A} = \mathbf{A}^T(\mathbf{AA}^T)^{-1}\mathbf{A}$ is the orthogonal projection on $\text{range}(\mathbf{A}^T)$.



Pseudoinverse of $\mathbf{A}_{m \times n}$ is the matrix $\mathbf{A}_{n \times m}^+$ satisfying the Moore-Penrose conditions:

$$\mathbf{A}^+\mathbf{AA}^+ = \mathbf{A}^+; \mathbf{AA}^+\mathbf{A} = \mathbf{A}; \mathbf{AA}^+, \mathbf{A}^+\mathbf{A} \text{ symmetric (uniquely defined)}.$$

Given the SVD of \mathbf{A} as \mathbf{USV}^T : $\mathbf{A}^+ = \mathbf{VS}^+\mathbf{U}^T$ with $s_i^+ = \begin{cases} s_i^{-1} & \text{if } s_i > 0 \\ 0 & \text{if } s_i = 0 \end{cases}$. Particular cases

(assume \mathbf{A} full-rank): $m > n \Rightarrow \mathbf{A}^+ = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T$, $m < n \Rightarrow \mathbf{A}^+ = \mathbf{A}^T(\mathbf{AA}^T)^{-1}$, $m = n \Rightarrow \mathbf{A}^+ = \mathbf{A}^{-1}$.

Singular value decomposition (SVD) of $\mathbf{A}_{m \times n}$, $m \geq n$: $\mathbf{A} = \mathbf{USV}^T = \sum_{i=1}^n s_i \mathbf{u}_i \mathbf{v}_i^T$ with $\mathbf{U}_{m \times n} = (\mathbf{u}_1 \cdots \mathbf{u}_n)$ and $\mathbf{V}_{n \times n} = (\mathbf{v}_1 \cdots \mathbf{v}_n)$ orthogonal, $\mathbf{S} = \text{diag}(s_1, \dots, s_n)$ and singular values $s_1 \geq \dots \geq s_n \geq 0$. Unique up to permutations/multiplicity of s.v.

- $\text{rank}(\mathbf{A}) = p \leq n \Leftrightarrow s_{p+1} = \dots = s_n = 0$: $\mathbf{A} = (\mathbf{U}_p \mathbf{U}_{n-p}) \begin{pmatrix} \mathbf{S}_p & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{pmatrix} \begin{pmatrix} \mathbf{V}_p^T \\ \mathbf{V}_{n-p}^T \end{pmatrix} = \mathbf{U}_p \mathbf{S}_p \mathbf{V}_p^T$ with $\mathbf{U}_p = (\mathbf{u}_1 \cdots \mathbf{u}_p)$ and $\mathbf{V}_{n-p} = (\mathbf{v}_{p+1} \cdots \mathbf{v}_n)$ orthonormal bases of $\text{range}(\mathbf{A})$ and $\text{null}(\mathbf{A})$, resp.
- $\text{rank}(\mathbf{A}) \geq p \Rightarrow \mathbf{U}_p \mathbf{S}_p \mathbf{V}_p^T$ is the best rank- p approximation to \mathbf{A} in the sense of the Frobenius norm ($\|\mathbf{A}\|_F^2 = \text{tr}(\mathbf{AA}^T) = \sum_{i,j} a_{ij}^2$) and the 2-norm ($\|\mathbf{A}\|_2 = \text{largest s.v.}$).

Matrix identities

$$\text{Ranks: } \begin{cases} \mathbf{A}_{m \times n}, \mathbf{B}_{n \times p} : \text{rank}(\mathbf{A}) + \text{rank}(\mathbf{B}) - n \leq \text{rank}(\mathbf{AB}) \leq \min(\text{rank}(\mathbf{A}), \text{rank}(\mathbf{B})) \\ \mathbf{A}_{m \times n}, \mathbf{B}_{m \times n} : \text{rank}(\mathbf{A} + \mathbf{B}) \leq \text{rank}(\mathbf{A}) + \text{rank}(\mathbf{B}) \end{cases}.$$

Inverse of a sum of matrices: given $\mathbf{A}_{p \times p}$, $\mathbf{B}_{p \times q}$, $\mathbf{C}_{q \times q}$, $\mathbf{D}_{q \times p}$, if \mathbf{A} , \mathbf{C} invertible:

$$(\mathbf{A} + \mathbf{BCD})^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{B}(\mathbf{C}^{-1} + \mathbf{DA}^{-1}\mathbf{B})^{-1}\mathbf{DA}^{-1} \quad (\text{Sherman-Morrison-Woodbury formula})$$

Inverse of a matrix by blocks: $\mathbf{A} = \begin{pmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{A}_{21} & \mathbf{A}_{22} \end{pmatrix}$, if \mathbf{A}_{11} , \mathbf{A}_{22} invertible:

$$\mathbf{A}^{-1} = \begin{pmatrix} \mathbf{A}^{11} & \mathbf{A}^{12} \\ \mathbf{A}^{21} & \mathbf{A}^{22} \end{pmatrix} : \quad \begin{cases} \mathbf{A}^{11} = (\mathbf{A}_{11} - \mathbf{A}_{12}\mathbf{A}_{22}^{-1}\mathbf{A}_{21})^{-1}, & \mathbf{A}^{12} = -\mathbf{A}^{11}\mathbf{A}_{12}\mathbf{A}_{22}^{-1} = -\mathbf{A}_{11}^{-1}\mathbf{A}_{12}\mathbf{A}^{22} \\ \mathbf{A}^{22} = (\mathbf{A}_{22} - \mathbf{A}_{21}\mathbf{A}_{11}^{-1}\mathbf{A}_{12})^{-1}, & \mathbf{A}^{21} = -\mathbf{A}^{22}\mathbf{A}_{21}\mathbf{A}_{11}^{-1} = -\mathbf{A}_{22}^{-1}\mathbf{A}_{21}\mathbf{A}^{11} \end{cases}$$

Derivatives:

$$\frac{d(\mathbf{a}^T \mathbf{x})}{d\mathbf{x}} = \nabla_{\mathbf{x}}(\mathbf{a}^T \mathbf{x}) = \mathbf{a} \quad \text{if } \mathbf{a}, \mathbf{x} \in \mathbb{R}^n, \mathbf{a} \text{ independent of } \mathbf{x}$$

$$\frac{d(\mathbf{x}^T \mathbf{A} \mathbf{x})}{d\mathbf{x}} = \nabla_{\mathbf{x}}(\mathbf{x}^T \mathbf{A} \mathbf{x}) = (\mathbf{A} + \mathbf{A}^T)\mathbf{x} \stackrel{\mathbf{A} \text{ symmetric}}{=} 2\mathbf{A}\mathbf{x} \quad \text{if } \mathbf{A}_{n \times n} \text{ independent of } \mathbf{x}$$

$$\mathbf{x}_{m \times 1}, \mathbf{y}_{n \times 1} : \frac{d\mathbf{y}^T}{d\mathbf{x}} = m \times n \text{ Jacobian matrix } \mathbf{J}(\mathbf{x}) = \left(\frac{\partial y_i}{\partial x_j} \right)_{ij}$$

$$f_{1 \times 1}, \mathbf{x}_{n \times 1} : \frac{d^2 f}{d\mathbf{x} d\mathbf{x}^T} = n \times n \text{ Hessian matrix } \nabla^2 f(\mathbf{x}) = \left(\frac{\partial^2 f}{\partial x_i \partial x_j} \right)_{ij}$$

$$\frac{d(\overbrace{\mathbf{x}^T}^{1 \times n} \overbrace{\mathbf{C}}^{n \times m})}{d \underbrace{\mathbf{x}}_{n \times 1}} = \mathbf{C}_{n \times m}, \quad \frac{d(\overbrace{\mathbf{B}}^{m \times n} \overbrace{\mathbf{x}}^{n \times 1})}{d \underbrace{\mathbf{x}^T}_{1 \times n}} = \mathbf{B}_{m \times n} \quad \text{if } \mathbf{B}, \mathbf{C} \text{ independent of } \mathbf{x}$$

$$\text{Product rule: } \frac{d(\mathbf{u}^T \mathbf{v})}{d\mathbf{x}} = \nabla_{\mathbf{x}}(\mathbf{u}^T \mathbf{v}) = \overbrace{\frac{d\mathbf{u}^T}{d\mathbf{x}}}^{n \times n} \mathbf{v} + \overbrace{\frac{d\mathbf{v}^T}{d\mathbf{x}}}^{n \times n} \mathbf{u}, \quad \mathbf{x}, \mathbf{u}, \mathbf{v} \in \mathbb{R}^n$$

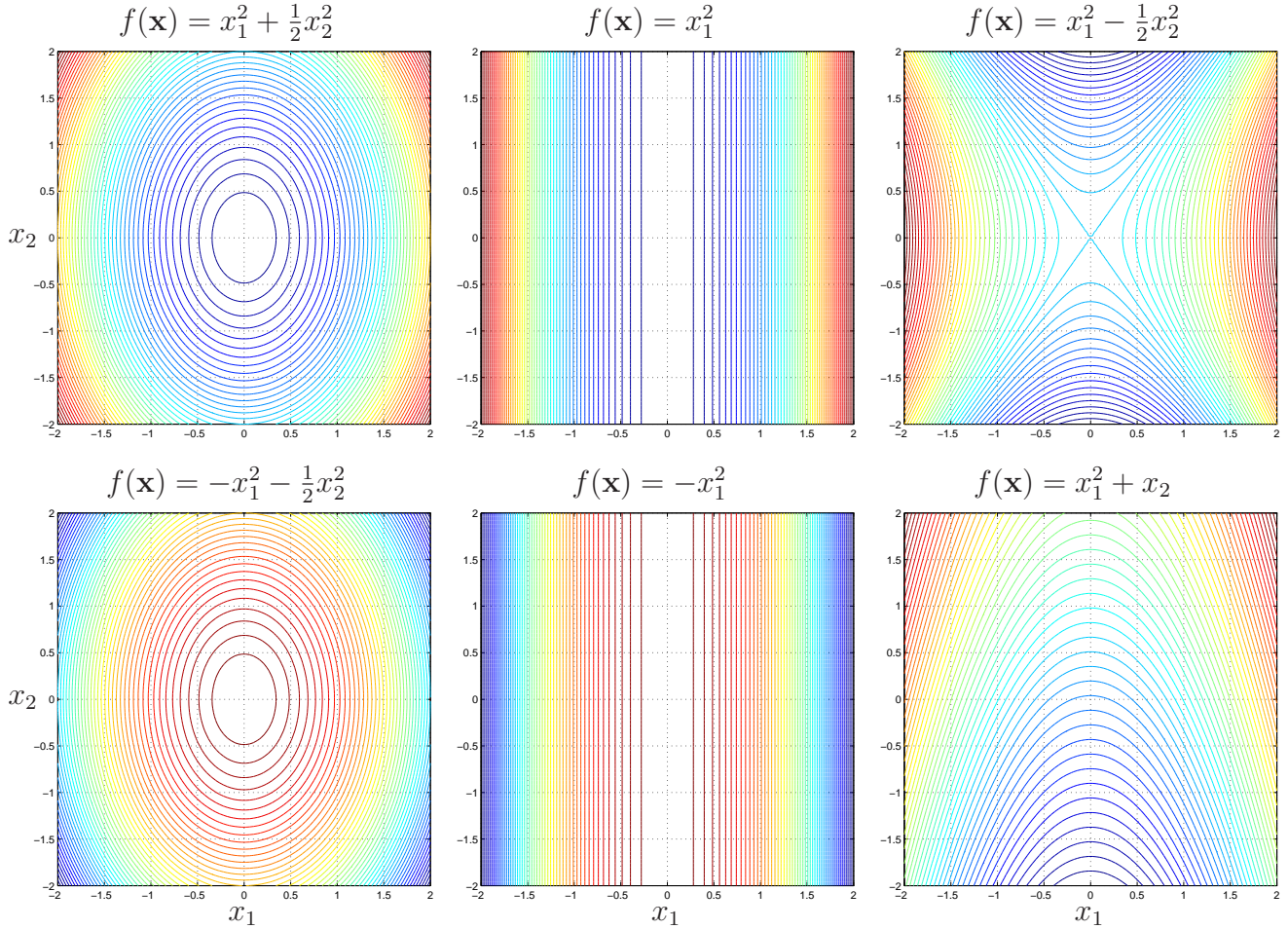
$$\text{Chain rule: } \frac{d\mathbf{y}(\mathbf{x}(t))}{dt} = \underbrace{\frac{d\mathbf{y}^T}{d\mathbf{x}}}_{m \times n} \underbrace{\frac{d\mathbf{x}}{dt}}_{n \times 1} = \sum_{i=1}^n \frac{\partial \mathbf{y}}{\partial x_i} \frac{dx_i}{dt}, \quad \mathbf{x} \in \mathbb{R}^n, \mathbf{y} \in \mathbb{R}^m, t \in \mathbb{R}$$

Quadratic forms

$f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T \mathbf{A} \mathbf{x} + \mathbf{b}^T \mathbf{x} + c$, $\mathbf{A} \in \mathbb{R}^{n \times n}$, $\mathbf{b}, \mathbf{x} \in \mathbb{R}^n$, $c \in \mathbb{R}$. Center and diagonalize it:

1. Ensure \mathbf{A} symmetric: $\mathbf{x}^T \mathbf{A} \mathbf{x} = \mathbf{x}^T \left(\frac{\mathbf{A} + \mathbf{A}^T}{2} \right) \mathbf{x}$.
2. Translate stationary point to origin: $\nabla f = \mathbf{A} \mathbf{x} + \mathbf{b} = \mathbf{0}$, change $\mathbf{y} = \mathbf{x} + \mathbf{A}^+ \mathbf{b} \Rightarrow f(\mathbf{y}) = \frac{1}{2}\mathbf{y}^T \mathbf{A} \mathbf{y} + \mathbf{b}'^T \mathbf{y} + c'$ with $\mathbf{b}' = (\mathbf{I} - \mathbf{A} \mathbf{A}^+) \mathbf{b}$, $c' = -\frac{1}{2} \mathbf{b}^T \mathbf{A}^+ \mathbf{b}$.
Linear terms may remain if \mathbf{A} is singular.
3. Rotate to axis-aligned: $\mathbf{A} = \mathbf{U} \mathbf{\Lambda} \mathbf{U}^T$ (spectral th.), change $\mathbf{z} = \mathbf{U}^T \mathbf{y} \Rightarrow f(\mathbf{z}) = \frac{1}{2}\mathbf{z}^T \mathbf{\Lambda} \mathbf{z} + \mathbf{b}''^T \mathbf{z} + c'$ with $\mathbf{b}'' = \mathbf{U}^T (\mathbf{I} - \mathbf{A} \mathbf{A}^+) \mathbf{b}$.

Considering only the quadratic part $\frac{1}{2}\mathbf{z}^T \mathbf{\Lambda} \mathbf{z} = \frac{1}{2} \sum_{i=1}^n \lambda_i z_i^2$, we have $\mathbf{A} \begin{cases} \text{pd:} & \text{single minimizer} \\ \text{psd:} & \infty \text{ minimizers} \\ \text{not def:} & \text{saddle point(s)} \\ \text{nsd:} & \infty \text{ maximizers} \\ \text{nd:} & \text{single maximizer} \end{cases}$



Order notation

Consider $f(n), g(n) \geq 0$ for $n = 1, 2, 3 \dots$

- Asymptotic upper bound $\mathcal{O}(\cdot)$: f is $\mathcal{O}(g)$ iff $f(n) \leq cg(n)$ for $c > 0$ and all $n > n_0$.
 f is of order g at most.
Ex.: $3n + 5$ is $\mathcal{O}(n)$ and $\mathcal{O}(n^2)$ but not $\mathcal{O}(\log n)$ or $\mathcal{O}(\sqrt{n})$.
- Asymptotic upper bound $o(\cdot)$: f is $o(g)$ iff $\lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = 0$.
 f becomes insignificant relative to g as n grows.
Ex.: $3n + 5$ is $o(n^2)$ and $o(n^{1.3})$ but not $o(n)$.
- Asymptotic tight bound $\Omega(\cdot)$: f is $\Omega(g)$ iff $c_0 g(n) \leq f(n) \leq c_1 g(n)$ for $c_1 \geq c_0 > 0$ and all $n > n_0$.
 f is $\mathcal{O}(g)$ and g is $\mathcal{O}(f)$.
Ex.: $3n + 5$ is $\Omega(n)$ but not $\Omega(n^2)$, $\Omega(\log n)$, $\Omega(\sqrt{n})$.

Cost of operations: assume $n \times 1$ vectors and $n \times n$ matrices.

- Space: vectors are $\mathcal{O}(n)$, matrix are $\mathcal{O}(n^2)$. Less if sparse or structured, e.g. a diagonal matrix or a circulant matrix can be stored as $\mathcal{O}(n)$.
- Time: we count scalar multiplications only.
 - vector \times vector: $\mathcal{O}(n)$.
 - matrix \times vector: $\mathcal{O}(n^2)$.
 - matrix \times matrix: $\mathcal{O}(n^3)$.
 - eigenvalues and eigenvectors: $\mathcal{O}(n^3)$.
 - inversion and linear system solution: $\mathcal{O}(n^3)$.

Less if sparse or structured, e.g. matrix \times vector is $\mathcal{O}(n)$ for a diagonal matrix and for a $\mathcal{O}(n \log n)$ circulant matrix.

Rates of convergence

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

- *Linear convergence*: $\frac{\|\mathbf{x}_{k+1} - \mathbf{x}^*\|}{\|\mathbf{x}_k - \mathbf{x}^*\|} \leq r$ 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 $r \approx 1$ for ill-conditioned problems).
- *Sublinear convergence*: $\lim_{k \rightarrow \infty} \frac{\|\mathbf{x}_{k+1} - \mathbf{x}^*\|}{\|\mathbf{x}_k - \mathbf{x}^*\|} = 1$. Ex.: $x_k = \frac{1}{k}$.
- *Superlinear convergence*: $\lim_{k \rightarrow \infty} \frac{\|\mathbf{x}_{k+1} - \mathbf{x}^*\|}{\|\mathbf{x}_k - \mathbf{x}^*\|} = 0$. Ex.: quasi-Newton methods (typically).
- *Quadratic convergence* (order 2): $\frac{\|\mathbf{x}_{k+1} - \mathbf{x}^*\|}{\|\mathbf{x}_k - \mathbf{x}^*\|^2} \leq M$ 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. Ex.: Newton's method.
- *Order p* : $\frac{\|\mathbf{x}_{k+1} - \mathbf{x}^*\|}{\|\mathbf{x}_k - \mathbf{x}^*\|^p} \leq M$ (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 the particular problem.