Technion – Israel Institute of Technology

Minerva Optimization Center

Technion City, Haifa 32000, Israel
Fax 972-4-8235194

OPTIMIZATION II
NUMERICAL METHODS
FOR
NONLINEAR CONTINUOUS OPTIMIZATION

A. Nemirovski

Spring Semester 1999

Contents

Lecture 1

Introduction

1.1 Optimization methods: the purpose

Our course is devoted to *numerical methods for nonlinear continuous optimization*, i.e., for solving problems of the type

$$\textit{minimize} \;\; f(x) \;\; \textit{s.t.} \;\; g_i(x) \leq 0,\, i = 1, ..., m;\;\; h_j(x) = 0,\, j = 1, ..., k. \tag{1.1.1}$$

Here x varies over $\mathbf{R}^n$, and the *objective* $f(x)$, same as the functions g_i and h_j, are smooth enough (normally we assume them to be at least once continuous differentiable). The constraints

$$g_i(x) \leq 0,\, i = 1, ..., m; \quad h_j(x) = 0,\, j = 1, ..., k$$

are referred to as *functional* constraints, divided in the evident manner into *inequality* and *equality* constraints.

We refer to (1.1.1) as to *nonlinear* optimization problems in order to distinguish between these problems and Linear Programming programs; the latter correspond to the case when all the functions f, g_i, h_j are linear. And we mention *continuous optimization* in the description of our subject to distinguish between it and *discrete optimization*, where we look for a solution from a discrete set, e.g., comprised of vectors with integer coordinates (Integer Programming), vectors with 0-1 coordinates (Boolean Programming), etc.

Problems (1.1.1) arise in huge variety of applications, since whenever people make decisions, they try to make them in an "optimal" manner. If the situation is simple enough, so that the candidate decisions can be parameterized by finite-dimensional vectors, and the quality of these desicions can be characterized by finite set of "computable" criteria, the concept of "optimal" decision typically takes the form of problem (1.1.1). Note that in real-world applications this preliminary phase – *modelling the decision making as an optimization problem with computable objective and constraints* – is, normally, much more difficult and creative than the subsequent phase when we solve the resulting problem. In our course, anyhow, we do not touch this modelling phase, and focus on technique for solving optimization programs.

Recall that problems (1.1.1) were our main subject already in the course "Optimization I", where we have developed optimality conditions for these problems. We remember that one can form a square system of nonlinear equations and a system of inequalities wich together define certain set – the one of *Karush-Kuhn-Tucker* points of the problem – which, under mild regularity conditions, contains all optimal solutions to the problem. The Karush-Kuhn-Tucker system of equations and inequalities typically has finitely many solutions, and if we are clever enough to

find analytically all these solutions, then we could look through them and to choose the one with the best value of the objective, thus getting the optimal solution in a closed analytical form. The difficulty, anyhow, is that as a rule we are *not* so clever to solve analytically the Karush-Kuhn-Tucker system, same as are unable to find the optimal solution analytically by other means. In all these "difficult" cases – and basically all optimization problems coming from real-world applications are difficult in this sense – all we may hope for is a numerical routine, an algorithm which allows to approximate numerically the solutions we are interested in. Thus, numerical optimization methods form the main tool for solving real-world optimization problems.

1.2 Preliminary Classification of Optimization Methods

It should be stressed that one hardly can hope to design a single optimization method capable to solve efficiently all nonlinear optimization problems – these problems are too diverse. In fact there are numerous methods, and each of them is oriented onto certain restricted family of optimization problems.

1.2.1 Classification of Nonlinear Optimization Problems

Traditionally, the Nonlinear Optimization Problems (1.1.1) are divided into two large classes:

- *Unconstrained problems* – no inequality or equality constraints are present. The generic form of an unconstrained problem, consequently, is

$$\textit{minimize} \;\; f(x) \;\; s.t. \;\; x \in \mathbf{R}^n, \tag{1.2.1}$$

  f being smooth (at least once continuously differentiable) function on $\mathbf{R}^n$;

- *Constrained problems*, involving at least one inequality or equality constraint.

The constrained problems, in turn, are subdivided into several classes, according to whether there are nonlinear constraints, inequality constraints, and so on; in the mean time we shall speak about this in more details.

According to the outlined classification of optimization problems, the optimization methods are primarily partitioned into those for unconstrained and constrained optimization. Although the more simpler unconstrained problems are not that frequently met in applications, methods for unconstrained optimization play very important role: they are used directly to solve unconstrained problems and indirectly, as building blocks, in many methods of constrained minimization.

1.2.2 Iterative nature of optimization methods

Methods for numerical solving nonlinear optimization problems are, in their essence, *iterative routines*: as applied to problem (1.1.1), a method typically is unable to find exact solution in finite number of computations. What a method generates, is an infinite sequence $\{x_t\}$ of *approximate solutions*. The next *iterate* x_{t+1} is formed, according to certain rules, on the basis of *local information* of the problem collected along the previous iterates. The portion of information $\mathcal{I}_t$ obtained at a current iterate x_t is a vector comprised of the values of the objective and the constraints at x_t and, possibly, of the gradients or even higher-order derivatives of these functions at x_t. Thus, when forming x_{t+1}, the method "knows" the values and the derivatives, up

to certain fixed order, of the objective and the constraints at the previous iterates $x_1, ..., x_t$, and this information is all information on the problem available to the method when it generates the new iterate x_{t+1}. This iterate, consequently, is certain function of the information accumulated so far:

$$x_{t+1} = X_{t+1}(\mathcal{I}_1, \mathcal{I}_2, ..., \mathcal{I}_t).$$

The collection of the *search rules* $X_t(\cdot)$ predetermines behaviour of the method as applied to an arbitrary problem; consequently, the *method itself can be identified with the collection* $\{X_t\}_{t=1}^{\infty}$. Note that the list of arguments in X_t is comprised of $(t-1)$ portions of local information; in particular, the list of arguments in the very first search rule X_1 is empty, so that this "function" is simply a fixed vector specified by the description of the method – the *starting point*.

From the outlined general scheme of an iterative routine it follows that optimization methods are classified not only according to the types of problems the methods are solving, but also according to the type of local information they use. From this "information" viewpoint, the methods are divided into

- *zero-order* routines using only values of the objective and the constraints and not using their derivatives;
- *first-order* routines using the values and the gradients of the objective and the constraints;
- *second-order* routines using the values, the gradients and the Hessians (i.e., matrices of second-order derivatives) of the objective and the constraints.

In principle, of course, we could speak about methods of orders larger than 2; these latter methods, anyhow, never are used in practice. Indeed, to use a method of an order k, you should provide possibility to compute partial derivatives of the objective and the constraints up to order k. In the multi-dimensional case it is not that easy even for $k = 1$ and even in the case when your functions are given by explicit analytical expressions (which is not always the case). And there is "explosure of difficulties" in computing higher order derivatives: for a function of n variables, there are n first order derivatives to be computed, $\frac{n(n+1)}{2}$ second order derivatives, $\frac{n(n+1)(n+2)}{2\times 3}$ third order derivatives, etc.; as a result, even in the medium-scale case with n being several tens the difficulties with programming, computation time, memory required to deal with higher-order derivatives make usage of these derivatives too expensive. On the other hand, there are no serious theoretical advantages in exploiting derivatives of order higher than 2, so there is nothing to compensate the cost of computing these derivatives.

1.3 Convergence of Optimization Methods

As a matter of fact, we cannot expect a nonlinear problem to be solved *exactly* in finite number of steps; all we can hope for is that the sequence of iterates $\{x_t\}$ generated by the method in question converges, as $t \to \infty$, to the solution set of the problem. In the theory of Numerical Optimization, the convergence of an optimization method on certain family of problems is exactly what gives the method right to be qualified as a tool for solving problems from the family. Convergence is not the only characteristic of a method, but this is the property which makes it a theoretically valid optimization routine. Our local goal is to present a kind of general framework (originating from Zangwill) which allows to prove convergence of optimization algorithms.

1.3.1 Abstract Iterative Algorithms

We already know what is a general optimization method: this is a set of rules for generating sequential iterates on the basis of information on the problem accumulated so far. In particlar, the larger is the index of the rule, the larger is the list of arguments in it. In the majority of actual methods, anyhow, the "prehistory" $(\mathcal{I}_1, ..., \mathcal{I}_t)$ of the process to the moment $t+1$ is in fact compressed to a vector μ_t of fixed dimension, and the next iterate is certain function of the current iterate x_t, of this "memory", and, of course, of the problem, let it be called P, the method is applied to:

$$x_{t+1} = \mathcal{X}_P(x_t, \mu_t). \tag{1.3.1}$$

In simple methods, the entire "prehistory" of the process is compressed to the current iterate x_t, so that the memory vector simply is empty; the future behaviour of these *memoryless* methods depends on the problem and the current iterate only, not on how the method came to this iterate.

In the general case, when the memory vector is non-empty, it, in turn, is updated from step to step according to the new portion of information on the problem:

$$\mu_{t+1} = \mathcal{M}_P(x_t, \mu_t); \tag{1.3.2}$$

here $\mathcal{X}_P(\cdot)$, $\mathcal{M}_P(\cdot)$ are some vector-valued functions specific for the method in question.

Relations (1.3.1) - (1.3.2) specify the behaviour of the method, up to the starting values of x_0 and μ_0; we may convert these relations into equivalent relation

$$\begin{pmatrix} x_{t+1} \\ \mu_{t+1} \end{pmatrix} \equiv \xi_{t+1} = \Xi_P(\xi_t) \equiv \begin{pmatrix} \mathcal{X}_P(x_t, \mu_t) \\ \mathcal{M}_P(x_t, \mu_t) \end{pmatrix} \tag{1.3.3}$$

Thus, for the majority of methods one can represent their behaviour on a problem P by simple recurrency

$$\xi_{t+1} = \Xi_P(\xi_t) \tag{1.3.4}$$

at the cost of extending the current iterate x_t to the "state vector" ξ_t belonging to a larger than our initial $\mathbf{R}^n$ space $\mathbf{R}^N$. Of course, to specify completely the trajectory $\{\xi_t\}$, we should indicate, along with Ξ, the initial point ξ_0 of the trajectory.

Example 1.3.1 *Consider an unconstrained minimization problem*

$$f(x) \to \min \mid \; x \in \mathbf{R}^n$$

and the simplest gradient-type method given by the recurrency

$$x_{t+1} = x_t - \gamma \nabla f(x_t),$$

where the stepsize $\gamma > 0$ is fixed. To represent this method in the form (1.3.4), no memory vector μ should be introduced: all "prehistiry" is accumulated in the last iterate x_t, and the method from the very beginning is in the form (1.3.4), with $\xi_t = x_t$.

The recurrency (1.3.4) involves a point-to-point mapping $\Xi_P(\cdot)$, and the properties of this mapping are responsible for all properties of the method in question, in particular, for its convergence. To give convenient and general sufficient conditions of convergence in terms of Ξ, it turns out to be reasonable to make one step of generalization more and to extend the point-to-point mapping Ξ to a point-to-set mapping Θ. Namely, consider the following abstract situation.

Definition 1.3.1 [Abstract iterative algorithm] *We are given a set $X \in \mathbf{R}^N$ and a mapping $\Theta(\cdot)$ which sets into correspondence to a point $\xi \in X$ a nonempty subset $\Theta(\xi)$ in X; we shall call the pair $(X, \Theta(\cdot))$ an abstract iterative algorithm $\mathcal{A}$. Given a initial point $\xi_0 \in X$, we define a trajectory of $\mathcal{A}$ as any sequence $\{\xi_t\}_{t=1}^{\infty}$ which satisfies the relations*

$$\xi_t \in \Theta(\xi_{t-1}), \quad t = 1, 2, ... \tag{1.3.5}$$

Let us stress that, since $\Theta(\cdot)$ is a point-to-set mapping, the trajectory of $\mathcal{A}$ is not, generally speaking, uniquely defined by the initial point ξ_0.

Let us say that recurrency (1.3.4), started at certain initial point ξ_0, is *covered* by an abstract iterative algorithm $\mathcal{A}$, if the trajectory of this recurrency is one of the trajectories (1.3.5) of the latter algorithm.

What we are about to do is to indicate natural sufficient conditions for convergence of abstract iterative algorithms; the goal is, of course, to use these conditions to prove convergence of methods of the type (1.3.1) - (1.3.2) "covered" by these abstract algorithms. Before coming to these conditions, it makes sense to explain why we have passed from the natural "point-to-point" presentation (1.3.4) of an iterative routine to its "point-to-set" covering (1.3.5). The reason is very simple: typically, as we shall see, the same basic scheme of an optimization process can be implemented in many different ways and thus – in many different, although close to each other, optimization routines. These different implementations of the same scheme usually can be covered by a single abstract iterative algorithm. Proving once for ever convergence of this abstract algorithm, we in one step prove conergence all implementations in question. Moreover, analysing the abstract algorithm, we understand which elements of the corresponding scheme indeed are essential for convergence and which elements can be varied without violating this crucial property. The resulting "freedom in implementation" is extremely important from the practical viewpoint: we get opportunity to adjust the "flexible elements" of the construction to the specific properties of problems we are interested in. An adjustment of this type for sure preserves convergence and may improve significantly practical performance of the method.

1.3.2 Convergence of an Abstract Iterative Algorithm

Let

$$\mathcal{A} = (X \subset \mathbf{R}^n, \Theta(\cdot))$$

be an abstract iterative algorithm. In order to analyze its convergence, we first of all should specify *towards what* the algorithm should converge. In the general convergence analysis, it is simply assumed that we are given a subset $\Gamma \subset X$ which is the desired *solution set*; in applications of this analysis to particular optimization methods we, of course, should properly specify this set Γ to get meaningful results.

Two crucial for convergence properties of an abstract iterative algorithm are its *descent* nature and *closedness*. These propertries are as follows.

Descent algorithms

Definition 1.3.2 [Descent function] *We say that a <u>continuous</u> function $Z(\xi) : X \to \mathbf{R}$ is a descent function (or a Lyapunov function) for an abstract iterative algorithm $\mathcal{A} = (X, \Theta(\cdot))$ with respect to a solution set Γ, if*

- $[\xi \in X \backslash \Gamma, \ \ \theta \in \Theta(\xi)] \Rightarrow Z(\theta) < Z(\xi)$;

- $[\xi \in \Gamma,\ \theta \in \Theta(\xi)] \Rightarrow Z(\theta) \leq Z(\xi)$.

We call an abstract iterative algorithm $\mathcal{A} = (X, \Theta(\cdot))$ descent with respect to a solution set $\Gamma \subset X$, if $\mathcal{A}$ admits a descent function with respect to Γ.

The definition is very clear – the mapping $\xi \mapsto \Theta(\xi)$ should not increase the descent function Z – the value of Z at any point from $\Theta(\xi)$ should be at most the value at ξ; and if ξ is not a solution, then the values of Z at the points from $\Theta(\xi)$ should be strictly less than the one at ξ.

Example 1.3.2 *When interested in solving unconstrained problem*

$$f(x) \to \min \mid\ x \in \mathbf{R}^n$$

by a memoryless method (so that ξ is the same as the original design vector x), the following three ways to specify Γ can be considered:

- A: Γ *is the set of global minimizers of f*
- B: Γ *is the set of local minimizers of f*
- C: Γ *is the set of critical points of f, i.e., the points where $\nabla f = 0$*

The natural candidate to the role of $Z(\cdot)$ in all three cases is f itself, and in the case of C, in addition, the function $|\nabla f(x)|$.

With these choices of the solution set and $Z(\cdot)$, the fact that $Z(\cdot)$ indeed is descent for an abstract iterative algorithm $\mathcal{A} = (X, \Theta(\cdot))$ means that any step of the type

$$x \mapsto y \in \Theta(x)$$

should

- *in the case of A reduce the value of the objective f, provided that x is not a global minimizer of f, and should not increase the objective, if x is a global minimizer of f;*
- *in the case of B reduce the value of the objective, provided that x is not a local minimizer of f, and should not increase the objective, if x is a local minimizer of f;*
- *in the case of C with $Z \equiv f$ reduce f, provided that x is not a critical point of f, and should not increase f, if x is a critical point;*
- *in the case of C with $Z \equiv |\nabla f|$ reduce $|\nabla f|$, provided that x is not a critical point of f, otherwise should map x onto itself or onto another critical point of f.*

Closed point-to-set mappings

Definition 1.3.3 [Closed mapping] *Let $X \subset \mathbf{R}^N$ and $Y \subset \mathbf{R}^M$, and let $\Theta(\cdot)$ be a point-to-set mapping from X to Y, so that $\Theta(\xi)$, $\xi \in X$, is a nonempty subset in Y.*

Let $\xi \in X$. We say that the point-to-set mapping Θ is closed at ξ, if for every two sequences $\{\xi_t \in X\}$ and $\{\theta_t \in \Theta(\xi_t)\}$ relations

$$\xi_t \to x,\ t \to \infty; \quad \theta_t \to \theta,\ t \to \infty$$

always imply that

$$\theta \in \Theta(\xi).$$

The mapping Θ is called closed on X, if it is closed at every point of X.

Example 1.3.3 *Let $\Theta(\cdot)$ be a point-to-point mapping on X which is continuous at $\xi \in X$ with respect to X:*

$$\forall\ (\xi_i \in X, \xi_i \to \xi,\ i \to \infty): \quad \Theta(\xi_i) \to \Theta(\xi),\ i \to \infty.$$

Then Θ is closed at ξ.

A point-to-set mapping Θ is closed on a closed set $X \in \mathbf{R}^N$ if and only if the graph of Θ – the set

$$\{(\xi, \theta) \in \mathbf{R}^N \times \mathbf{R}^M \mid \theta \in \Theta(\xi)\}$$

– is a closed set in $\mathbf{R}^{N+M}$.

As we shall see in a while, closedness of the mapping associated with an abstract iterative algorithm is crucial for the convergence of the algorithm. This is why it is important to know how to verify closedness. In many cases the mapping we are interested in is combined of simpler mappings, and it would be fine to have a possibility to reduce the analisys of closedness for the "combined" mapping to the one for its components. Let us formulate and prove the most useful statement of this type related to *composition* of point-to-set mappings.

Definition 1.3.4 [Composition of mappings] *Let $A(\cdot)$ be a point-to-set mapping from X to Y, and let $B(\cdot)$ be a point-to-set mapping from Y to Z. The composition $C = BA$ of these mappings is defined as the point-to-set mapping of X to Z given by*

$$C(\xi) = \bigcup_{\eta \in A(\xi)} B(\eta),\ \ \xi \in X.$$

Proposition 1.3.1 [Closedness of composition] *Let A be a point-to-set mapping from X to Y, B be a point-to-set mapping from Y to Z and C be the composition of A and B.*

Let $\xi \in X$. Assume that

- (i) *A is closed at ξ;*
- (ii) *A is locally bounded at ξ, i.e., whenever $\xi_t \to \xi$ as $t \to \infty$ ($\xi_t \in X$) and $\eta_t \in A(\xi_t)$, the sequence $\{\eta_t\}$ is bounded;*
- (iii) *B is closed at every point of $A(\xi)$.*

Then C is closed at ξ.

Proof. Let $\xi_t \to \xi$, $t \to \infty$, and let $\zeta_t \in C(\xi_t)$ are such that $\zeta_t \to \zeta$ as $t \to \infty$; we should prove that $\zeta \in C(\xi)$.

Indeed, from the construction of the composition it follows that there exist $\eta_t \in A(\xi_t)$ such that $\zeta_t \in B(\eta_t)$, $t = 1, 2, ...$ Since A is locally bounded at ξ (see (ii)), the sequence $\{\eta_t\}$ is bounded; consequently, there exists a converging subsequence $\{\eta_{t_s}\}$, $t_1 < t_2 < ...$:

$$\lim_{s \to \infty} \eta_{t_s} = \eta.$$

We have $\xi_{t_s} \to \xi$, $s \to \infty$, $\eta_{t_s} \in A(\xi_{t_s})$, $\eta_{t_s} \to \eta$, $s \to \infty$; since A is closed at ξ by (i), these relations imply that

$$\eta \in A(\xi). \tag{1.3.6}$$

Now we have $\eta_{t_s} \in Y$, $\eta_{t_s} \to \eta \in A(\xi)$ as $s \to \infty$, and $\zeta_{t_s} \in B(\eta_{t_s})$, $\zeta_{t_s} \to \zeta$ as $s \to \infty$. Since the mapping B is closed at the point $\eta \in A(\xi)$ by (iii), we conclude that $\zeta \in B(\eta) \subset C(\xi)$ (the final inclusion follows from the definition of composition due to (1.3.6)). ∎

Corollary 1.3.1 *Let $A : X \to Y$ and $B : Y \to Z$ be point-to-set mappings, let A be closed at $\xi \in X$, B be closed at every point of $A(\xi)$, and let Y be bounded. Then the composition $C = BA$ is closed at ξ.*

Indeed, if Y is bounded, then, of course, A is locally bounded at any point, and it suffices to apply Proposition 1.3.1.

Corollary 1.3.2 *Let $A : X \to Y$ be a point-to-point mapping and $B : Y \to Z$ be a point-to-set mapping. Assume that A is continuous at $\xi \in X$ with respect to X and B is closed at $A(x)$. Then the composition $C = BA$ is closed at ξ.*

Indeed, A is closed at ξ (Example 1.3.3) and clearly locally bounded at ξ; it remains to apply Proposition 1.3.1.

Global Convergence Theorem

Now we are ready to establish

Theorem 1.3.1 ["Global Convergence Theorem"] *Let $\mathcal{A} = (X, \Theta(\cdot))$ be an abstract iterative algorithm, $\Gamma \subset X$ be a given solution set and $\xi_0 \in X$ be a given initial point. Consider a sequence $\{\xi_t\}$ satisfying the inclusions*

$$\xi_t \in \Theta(\xi_{t-1}),\ t = 1, 2, ...$$

and assume that

- (i) [descent property] *there exists a function $Z(\cdot)$ which is descent for $\mathcal{A}$ with respect to the solution set Γ;*
- (ii) [closedness outside Γ] *The mapping $\Theta(\cdot)$ is closed at every point outside Γ;*
- (iii) [compactness of the trajectory] *There exists a compact set $S \subset X$ which contains the sequence $\{\xi_t\}$.*

Then limiting points of the trajectory $\{\xi_t\}$ exist, and every point of this type belongs to the solution set Γ.

In particular, if, in addition to (i) - (iii), *$\Gamma = \{\xi^*\}$ is a singleton, then the sequence $\{\xi_t\}$ converges to x^*.*

Proof. Since the sequence $\{\xi_t\}$ is contained in a compact subset S of X (see (iii)), it possesses limiting points. Let ξ be a limiting point of $\{\xi_t\}$, so that there is a convergent to ξ subsequence of our sequence:

$$\xi^s \equiv \xi_{t_s} \to \xi,\ s \to \infty \quad [t_1 < t_2 < ...]; \tag{1.3.7}$$

by passing to a new subsequence, we may assume that

$$\theta^s \equiv \xi_{t_s+1} \to \theta,\ s \to \infty. \tag{1.3.8}$$

We should prove that $\xi \in \Gamma$. Assume, on contrary, that $\xi \notin \Gamma$, and let us lead this assumption to a contradiction.

By construction,

$$\theta^s \equiv \xi_{t_s+1} \in \Theta(\xi^s);\ \xi^s \to \xi,\ \theta^s \to \theta,\ s \to \infty. \tag{1.3.9}$$

Besides this, we have assumed that $\xi \notin \Gamma$, so that Θ is closed at ξ by (ii); this fact combined with (1.3.9) implies that

$$\theta \in \Theta(\xi).$$

The latter relation combined with the descent property (i) and the fact that $\xi \notin \Gamma$ results in

$$Z(\theta) < Z(\xi). \tag{1.3.10}$$

The resulting relation can be immediately led to a contradiction. Indeed, from the descent property (i) it follows that $Z(\xi_{t+1}) \leq Z(\xi_t)$, so that the sequence $\{Z(\xi_t)\}$ is monotonically nonincreasing. Along the subsequence $t = t_s$, $s = 1, 2, ...$, the points ξ_t converge to ξ, and, since $Z(\cdot)$ is continuous (the definition of a descent function), the monotone sequence $\{Z(\xi_t)\}$ has a converging subsequence and, consequently, converges itself. It follows that

$$Z(\xi_t) - Z(\xi_{t+1}) \to 0,\ t \to \infty. \tag{1.3.11}$$

At the same time, along the sequence $t = t_s$, $s = 1, 2, ...$, of values of t we have $\xi_{t_s} = \xi^s \to \xi$, $\xi_{t_s+1} = \theta^s \to \theta$, and since Z is continuous, $Z(\xi_{t_s}) - Z(\xi_{t_s+1}) = Z(\xi^s) - Z(\theta^s) \to Z(\xi) - Z(\theta)$. The latter quantity is positive by (1.3.10), which contradicts (1.3.11). ∎

1.3.3 Rates of convergence

The fact of convergence of an optimization (and any other) computational method is the "weakest" property which gives the method right to exist. In principle, there are as many methods with this property as you wish, and the question is how to rank these mehods and which of them to recommended for practical usage. In the traditional Nonlinear Optimization this problem is resolved by looking at the *asymptotic rate of convergence* measured as follows.

Assume that the method as applied to problem P generates sequence of iterates which converges to the solution set X_P^* of the problem. To define the *rate of convergence*, we first introduce an *error function* $\mathrm{err}(x)$ which measures the quality of an approximate solution x; this function should be positive outside X_P^* and should be zero at the latter set.

There are several ways for reasonable choice of the error function. E.g., we always can use as it the distance from the approximate solution to the solution set:

$$\mathrm{dist}_P(x) = \inf_{x^* \in X_P^*} |x - x^*|;$$

another choice of the error function could be the residual in terms of the objective and constraints, like

$$\mathrm{res}_P(x) = \inf_{x \in X_P^*} \max\{f(x) - f(x^*); [g_1(x)]_+; ...; [g_m(x)]_+; |h_1(x)|; ...; |h_k(x)|\},$$

$[a]_+ = \max(a, 0)$ being the positive part of a real a, etc.

For properly chosen error function (e.g., for dist_P), convergence of the iterates to the solution set implies that the scalar sequence

$$r_t = \mathrm{err}(x_t)$$

converges to 0, and we measure the quality of convergence by the rate at which the nonnegative reals r_t tend to zero.

The standard classification here is as follows:

- [linear convergence] a sequence $\{r_t \geq 0\}$ such that for some $q \in (0,1)$, some $C < \infty$ and all t one has
$$r_t \leq Cq^t$$
are called *linearly converging to 0 with convergence ratio q*; the simplest example of such a sequence is $r_t = Cq^t$. The lower bound of those q for which $\{r_t\}$ linearly converges to 0 with the convergence ratio q is called the *convergence ratio* of the sequence.

  E.g., for the sequence $r_t = Cq^t$, same as for the sequence $\{r_t = C(q+\epsilon_t)^t\}$, $\epsilon_t \to 0$, $t \to \infty$, the convergence ratio is q, although the second sequence, generally speaking, does not converge to 0 with the convergence ratio q (it, anyhow, converges to 0 linearly with convergence ratio q' for any $q' \in (q,1)$).

  It is immediately seen that a sufficient condition for a sequence $\{r_t > 0\}$ to be linearly converging with convergence ratio $q \in (0,1)$ is to satisfy the property
$$\limsup_{t\to\infty} \frac{r_{t+1}}{r_t} < q.$$

- [sub- and superlinear convergence] a sequence which converges to 0, but is not linearly converging (e.g., the sequence $r_t = t^{-1}$), is called *sublinearly* converging. A sequence which linearly converges to zero with any positive convergence ratio (so that the convergence ratio of the sequence is 0) is called *superlinearly converging* (e.g., the sequence $r_t = t^{-t}$).

  A sufficient condition for a sequence $\{r_t > 0\}$ to be superlinearly converging is
$$\lim_{t\to\infty} \frac{r_{t+1}}{r_t} = 0.$$

- [convergence of order $p > 1$] a sequence $\{r_t \geq 0\}$ converging to 0 is called to have *convergence order $p > 1$*, if for some C and all large enough t one has
$$r_{t+1} \leq Cr_t^p.$$
  The upper bound of those p for which the sequence converges to 0 with order p is called the *order of convergence* of the sequence.

  E.g., the sequence $r_t = a^{(p^t)}$ $(a \in (0,1), p > 1)$ converges to zero with order p, since $r_{t+1}/r_t^p = 1$. The sequences converging to 0 with order 2 have special name – they are called *quadratically convergent*.

  Of course, a sequence converging to 0 with order $p > 1$ is superlinearly converging to 0 (but, generally speaking, not vice versa).

Traditionally, the rate of convergence of iterative numerical routines is measured by the rank of the corresponding sequence of errors $\{r_t = \mathrm{err}(x_t)\}$ in the above scale; in particular, we may speak about *sublinearly*, *linearly*, *superlinearly*, *quadratically* converging methods, same as about methods with order of convergence $p > 1$. It is common to think that the better is the rate of convergence of a method, the more preferable is the method itself. Thus, a linearly converging method is thought to be better than a sublinearly converging one; among two linearly converging methods the more preferable is the one with the smaller convergence ratio; a superlinearly converging method is prefered to a linearly converging one, etc. Of course, all these preferences

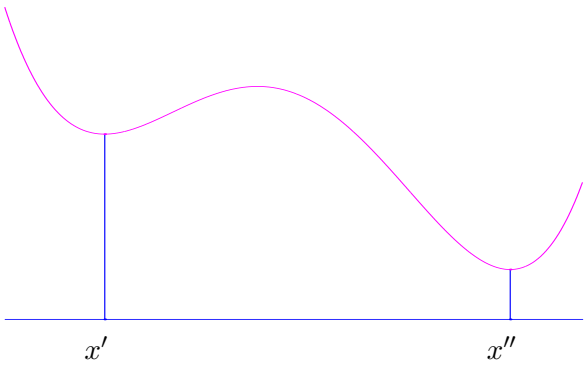
are "conditional", provided that there are no significant differences in computational complexity of steps, etc.

We should stress that the rate of convergence, same as the very property of convergence, is an asymptotic characteristic of the sequence of errors; it does not say *when* the announced rate of convergence occurs, i.e., what are the values of C or/and "large enough" values of t mentioned in the corresponding definitions. For concrete methods, the bounds on the indicated quantities typically can be extracted from the convergence proofs, but it does not help a lot – the bounds are usually very complicated, rough and depend on "invisible" quantitive characteristics of the problem like the magnitides of the higher-order derivatives, condition numbers of Hessians, etc. It follows that *one should not overestimate the importance of the rate-of-convergence ranking of the methods*. This traditional approach gives a kind of orientation, nothing more; unfortunately, there seems to be no purely theoretical way to get detailed ranking of numerical optimization methods. As a result, practical recommendations on which method to use are based on different theoretical and empirical considerations: theoretical rate of convergence, actual behaviour on test problems, numerical stability, simplicity and robustness, etc.)

1.3.4 Global and Local solutions

The crucial intrinsic difficulty in Nonlinear Optimization is that *we cannot expect a numerical optimization method to approximate a globally optimal solution to the problem*.

The difficulty has its roots in *local* nature of the information on the problem which is available to the methods. Assume, e.g., that we are minimizing the function shown on the picture:



The function has two local minimizers, x' and x''. Observing small enough neighbourhood of every one of these minimzers, it is impossible to guess that in fact there exists another one. As a result, any "normal" method of nonlinear optimization as applied to the objective in question and being started from a small neighbourhood of the "wrong" (local, not global) minimizer x', will converge to x' – the local information on f available for the method contains no hint to x''!

It would be wrong to say that the difficulty is absolutely unavoidable. We could run the method for different starting points, or even could look through the values of the objective along a sequence which is dense in $\mathbf{R}$ and define x_t as the best, in terms of the values of f, of the first t point of the sequence. The latter "method" can be easily extended to general constrained multidimensional problems; one can immediately prove its convergence to the global solution; the method is simple in implementation, etc. There is only one small drawback in the method: the tremendous number of function evaluations required to solve a problem within inaccuracy ϵ.

It can be easily seen that the outlined method, as applied to a problem

$$f(x) \to \min \mid \; x \in \mathbf{R}^n, g_1(x) = |x|^2 \leq 1$$

with Lipschitz continuous, with constant 1, objective f, requires, *in the worst case*, at least $(3\epsilon)^{-n}$ steps to find a point x_ϵ with the residual $f(x_\epsilon) - \min_{|x|\leq 1} f$ in terms of the objective $\leq \epsilon$.

When $\epsilon = 0.03$ and $n = 20$ (very modest accuracy and dimension requirements), the number of steps becomes $> 10^{20}$, and this is the *lower* complexity bound!

Moreover, for the family of problems in question the lower bound $(3\epsilon)^{-n}$ on the number of function evaluations required to *guarantee* residual $\leq \epsilon$ is valid for *an arbitrary* optimization method which uses only local information on the objective.

Thus, we can approximate, within any given error $\epsilon > 0$, *the global* solution to any optimization problem; but to say that *the best* we can promise is to do this, for $\epsilon = 0.03$, $n = 20$, in 10^{20} steps – is worse than to say nothing.

As a consequence of the above considerations (same as of other, more advanced results of *Information-Based Complexity Theory of Optimization*), we come to the following important, although a desperating, conclusion:

It makes no sense to expect an optimization method to be able to approximate, with a reasonable inaccuracy in a reasonable time, global solution to all optimization problems of a given, even moderate, size.

In fact all we may hope to do in reasonable time is to find tight approximations to certain (not necessarily corresponding to the optimal solution) Karush-Kuhn-Tucker point of an optimization problem (in the unconstrained case – to a critical point of the objective). In simple cases we may hope also to approximate a locally optimal solution, without any garanties of its global optimality.

There is, anyhow, a "solvable case" when we do can approximate at reasonable complexity *globally optimal* solution to an optimization problem. This is the case when the problem is *convex* (i.e., the functions f and g_i, $i = 1, ..., m$, are convex, while h_j, if any, are linear). Properties of convex optimization problems and the numerical methods for these problems form the subject of Convex Programming. Convex Programming is, in its nature, simpler and, consequently, much more advanced than the general Nonlinear Optimization. In particular, in Convex Programming we are able to point out methods with quite reasonable *global* (not asymptotical!) rate of convergence which are capable to *guarantee*, at a reasonable computational cost, high-accuracy approximations of *globally* optimal solutions even in general-type convex programs.

I would be happy to restrict our course to the nice world of Convex Programming, but we cannot afford it to ourselves: in actual applications we, unfortunately, too often meet with nonconvex problems, and we have no choice but to solve them – even at the cost of weakening the notion of "optimal solution" from the global to a local one (or even to the one of a Karush-Kuhn-Tucker point).

Assignment # 1 (Lecture 1)

Exercise 1.3.1 *Let point-to-set mapping $\Theta(\cdot)$ from $\mathbf{R}^n$ into $\mathbf{R}^n$ be defined as*

$$\Theta(\xi) = \{\eta \in \mathbf{R}^n \mid \eta^T \xi \leq 0\}.$$

Is the mapping closed?

Exercise 1.3.2 *Consider the abstract iterative algorithm $\mathcal{A} = (\mathbf{R}, \Theta(\cdot))$, where*

$$\Theta(\xi) = \xi + 1$$

is a point-to-point mapping. Set the solution set Γ to be empty, and define $Z(\xi)$ as $\exp\{-\xi\}$.

1) Prove that Z is a descent function for $\mathcal{A}$ with respect to Γ

2) Prove that Θ is closed at any point of $X = \mathbf{R}$

3) Look at the (clearly diverging) trajectory $\xi_{t+1} = \Theta(\xi_t)$, $\xi_0 = 0$, of our abstract itrative algorithm. What prevents to apply the Global Convergence Theorem?

Exercise 1.3.3 *Consider the abstract iterative algorithm $\mathcal{A} = (X = [0,1], \Theta(\cdot))$, where*

$$\Theta(\xi) = \begin{cases} [0, \xi), & 0 < \xi \leq 1 \\ \{0\}, & \xi = 0 \end{cases}$$

Define the solution set Γ to be $\{0\}$ and set $Z(\xi) = \xi$.

1) Prove that Z is a descent function for $\mathcal{A}$ with respect to Γ

2) Consider the sequence $\{\xi_t\}$ such that $\xi_0 = 1$ and $1/2 < \xi_{t+1} < \xi_t$ for all t (e.g., $\xi_{t+1} = \frac{1}{2}[\frac{1}{2} + \xi_t]$). Prove that the sequence is a trajectory of the abstract iterative algorithm in question, although it does not converge to the solution set. What prevents to apply the Global Convergence Theorem?

Exercise 1.3.4 *Prove statements indicated in Example 1.3.3.*

Lecture 2

Line Search

This lecture is devoted to *one-dimensional* unconstrained optimization, i.e., to numerical methods for solving problems

$$f(x) \to \min \mid x \in \mathbf{R}, \tag{2.0.1}$$

f being *at least* continuous function on the axis; these methods usually are called *line search*.

Our interest in line search comes, of course, not from the fact that there are many applied one-dimensional problems, but rather from the fact that line search is a component of basically all methods for multidimensional optimization. Typically, the scheme of a multidimensional method for unconstrained minimization is as follows: looking at local behaviour of the objective f at the current iterate x_t, the method chooses current "direction of movement" d_t (which, normally, is a descent direction of the objective, i.e., $d_t^T \nabla f(x_t) < 0$) and then performs a step in this direction:

$$x_t \mapsto x_{t+1} = x_t + \alpha_t d_t$$

in order to achieve certain progress in the objective value, i.e., to ensure that $f(x_{t+1}) < f(x_t)$. And in the majority of methods the step in the direction d_t is chosen by one-dimensional minimization of the function

$$\phi(\alpha) = f(x_t + \alpha d_t).$$

Thus, line search technique is crucial for basically all multidimensional optimization methods.

2.1 Zero-Order Line Search

We start with the *zero-order* line search, i.e., with methods for solving (2.0.1) which use the values of f only, not the derivatives.

The methods we are about to develop solve not the problem (2.0.1) as it is, but the problem

$$f(x) \to \min \mid a \le x \le b \tag{2.1.1}$$

of minimizing the objective over a given finite ($-\infty < a < b < \infty$) segment $[a, b]$ of the real axis. To ensure well-posedness of the problem, we make the following assumption:

f is *unimodal* on $[a, b]$, i.e., possesses a unique local minimum x^* on the segment.

This assumption, as it is easily seen, implies that f is stirctly decreasing in $[a, b]$ to the left of x^*:

$$a \le x' < x'' \le x^* \Rightarrow f(x') > f(x'') \tag{2.1.2}$$

and is strictly increasing in $[a, b]$ to the right of x^*:

$$x^* \leq x' < x'' \leq b \Rightarrow f(x') < f(x''). \tag{2.1.3}$$

Indeed, if (2.1.2) were false, there would exist x' and x'' such that

$$a \leq x' < x'' \leq x^*, \;\; f(x') \leq f(x'').$$

It follows that the set of minimizers of f on $[a, x'']$ contains a minimizer, x_*, which differs from x'' [1]. Since x_* is a minimizer of f on $[a, x'']$ and x_* differs from x'', x_* is a local minimizer of f on $[a, b]$, while it was assumed that the only local minimizer of f on $[a, b]$ is x^*; this gives the desired contradiction. Verification of (2.1.3) is similar.

Note that relations (2.1.2) - (2.1.3), in turn, imply that f is unimodal on $[a, b]$ and even on every smaller segment $[a', b'] \subset [a, b]$.

Given that f is unimodal on $[a, b]$, we immediately can point out a strategy for approximating x^*, namely, as follows. Let us choose somehow two points x^- and x^+ in (a, b),

$$a < x^- < x^+ < b,$$

and let us compute the values of f at these points. The basic observation is that

if [case A] $f(x^-) \leq f(x^+)$, *then* x^* *is to the left of* x^+ [indeed, if x^* were to the right of x^+, then we would have $f(x^-) > f(x^+)$ by (2.1.2)], *and if* [case B] $f(x^-) \geq f(x^+)$, *then* x^* *is to the right of* x^- ["symmetric" reasoning].

Consequently, in the case of A we can replace the initial "uncertainty segment" $\Delta_0 = [a, b]$ with the new segment $\Delta_1 = [a, x^+]$, and in the case of B – with the segment $\Delta_1 = [x^-, b]$; in both cases the new "uncertainty segment" Δ_1 covers x^* and is strictly less than Δ_0. Since, at is was already mentioned, the objective, being unimodal on the initial segment $\Delta_0 = [a, b]$, is unimodal also on the smaller segment $\Delta_1 \subset \Delta_0$, we may iterate this procedure – choose two points in Δ_1, compute the values of the objective at these points, compare the results and replace Δ_1 with smaller uncertainty segment Δ_2, still containing the desired solution x^*, and so on.

Thus, we come to the following

Algorithm 2.1.1 [Conceptual zero-order minimization of unimodal function on $[a, b]$]
Initialization: *Set* $\Delta_0 = [a, b]$, $t = 1$
Step t: *Given previous uncertainty segment* $\Delta_{t-1} = [a_{t-1}, b_{t-1}]$,

- *choose search points* x_t^-, x_t^+: $a_{t-1} < x_t^- < x_t^+ < b_{t-1}$;
- *compute* $f(x_t^-)$ *and* $f(x_t^+)$;
- *define the new uncertainty segment as follows: in the case of* $f(x_t^-) \leq f(x_t^+)$ *set* $\Delta_t = [a_{t-1}, x_t^+]$, *otherwise set* $\Delta_t = [x_t^-, b_{t-1}]$;
- *replace* t *with* $t + 1$ *and loop.*

[1] look: if x'' itself is not a minimizer of f on $[a, x'']$, then *any* minimizer of f on $[a, x'']$ can be chosen as x_*; if x'' is a minimizer of f on $[a, x'']$, then x' also is a minimizer, since $f(x') \leq f(x'')$, and we can set $x_* = x'$

It is immediately seen that we may linear convergence of the lengths of subsequent uncertainty segments to 0, thus coming to a linearly converging algorithm of approximating x^*. E.g., if x_t^-, x_t^+ are chosen to split Δ_{t-1} it into three equal parts, we ensure $|\Delta_{t+1}| = \frac{2}{3}|\Delta_t|$ ($|\Delta|$ stands for the length of a segment Δ), thus coming to the linearly converging, with the convergence ratio $\sqrt{2/3}$, algorithm for approximating x^*:

$$|x^* - x^k| \leq \left(\frac{2}{3}\right)^{\lfloor k/2 \rfloor} |b - a|, \tag{2.1.4}$$

k being the # of function evaluations performed so far and x^k being an arbitrary point of the uncertainty segment $\Delta_{\lfloor k/2 \rfloor}$ formed after k function evaluations.

Estimate (2.1.4) is fine – we have nonasymptotical linear convergence rate with problem-independent convergence ratio. Could there be something better?

The answer is "yes". The way to improve the rate of convergence is to note that one of the two search points used to pass from Δ_t to Δ_{t+1} will for sure be inside Δ_{t+1}, and we could try to make it the search point used to pass from Δ_{t+1} to Δ_{t+2}; with this strategy, *the cost of updating Δ_t into Δ_{t+1} will be one function evaluation, not two of them* (except the very first updating $\Delta_0 \to \Delta_1$ which still will cost two function evaluations). There are two ways to implement this new smart strategy – the optimal one ("Fibonacci search") and the suboptimal ("Golden search").

2.1.1 Fibonacci search

The Fibonacci search can be used when we know in advance the number $N > 2$ of function evaluations we are going to perform.

Given N, consider the sequence of the first $N+1$ *Fibonacci integers* $F_0, F_1, F_2, ..., F_N$ defined by the recurrency

$$F_0 = F_1 = 1; F_k = F_{k-1} + F_{k-2}$$

(the first 10 integers in the sequence are $1, 1, 2, 3, 5, 8, 13, 21, 34, 55$). The method is as follows: given $\Delta_0 = [a, b]$, we set

$$d_0 = |b - a|,$$

choose two first search points x_1^- and x_1^+ at the distance

$$d_1 = \frac{F_{N-1}}{F_N} d_0$$

from the right and from the left endpoints of Δ_0, respectively (since $F_N/F_{N-1} = (F_{N-1} + F_{N-2})/F_{N-1} = 1 + F_{N-2}/F_{N-1} < 2$, we have $d_1 > d_0/2$, so that $x_1^- < x_1^+$). The length of the new uncertainty segment Δ_1 clearly will be d_1.

What we are about to do is to iterate the above step, with N replaced by $N-1$. Thus, now we should evaluate f at two points x_2^-, x_2^+ of the segment Δ_1 placed at the distance

$$d_2 = \frac{F_{N-2}}{F_{N-1}} d_1 \quad \left[= \frac{F_{N-2}}{F_{N-1}} \frac{F_{N-1}}{F_N} d_0 = \frac{F_{N-2}}{F_N} d_0\right] \tag{2.1.5}$$

from the right and the left endpoint of Δ_1. The crucial fact (which takes its origin in the arithmetic properties of the Fibonacci numbers) is that

one of these two required points where f should be computed is already processed – this is the one of the previous two points which belongs to Δ_1.

Indeed, assume, without loss of generality, that $\Delta_1 = [a, x_1^+]$ (the case $\Delta_1 = [x_1^-, b]$ is completely similar), so that the one of the first two search point belonging to Δ_1 is x_1^-. We have

$$x_1^- - a = (b - d_1) - a = (b - a) - d_1 = d_0 - d_1 = d_0\left(1 - \frac{F_{N-1}}{F_N}\right) =$$

[since $F_N = F_{N-1} + F_{N-2}$ and $d_2 = \frac{F_{N-2}}{F_N} d_0$]

$$= d_0 \frac{F_{N-2}}{F_N} = d_2.$$

Thus, only one of the two required points of Δ_1 is new for us, and another comes from the previous step; consequently, in order to update Δ_1 into Δ_2 we need one function evaluation, not two of them. After this new function evaluation, we are able to replace Δ_1 with Δ_2. To process Δ_2, we act exactly as above, but with N replaced by $N - 2$; here we need to evaluate f at the two points of Δ_2 placed at the distance

$$d_3 = \frac{F_{N-3}}{F_{N-2}} d_2 \quad [= \frac{F_{N-3}}{F_N} d_0, \text{ see (2.1.5)}]$$

from the endpoints of the segment, and again one of these two points already is processed.

Iterating this procedure, we come to the segment Δ_{N-1} which covers x^*; the length of the segment is

$$d_{N-1} = \frac{F_1}{F_N} d_0 = \frac{b - a}{F_N},$$

and the total # of evaluations of f required to get this segment is N (we need 2 evaluations of f to pass from Δ_0 to Δ_1 and one evaluation per every of $N - 2$ subsequent updatings $\Delta_t \mapsto \Delta_{t+1}$, $1 \le t \le N - 2$).

Taking, as approximation of x^*, any point x^N of the segment Δ_{N-1}, we have

$$|x^N - x^*| \le |\Delta_N| = \frac{b - a}{F_N}. \tag{2.1.6}$$

To compare (2.1.6) with the accuracy estimate (2.1.4) of our initial – unsophisticated – method, note that

$$F_t = \frac{1}{\lambda + 2}\left[(\lambda + 1)\lambda^t + (-1)^t \lambda^{-t}\right], \ \lambda = \frac{1 + \sqrt{5}}{2} > 1.^{2)} \tag{2.1.7}$$

[2]Here is the computation: the Fibonacci numbers satisfy the homogeneous finite-difference equation

$$x_t - x_{t-1} - x_{t-2} = 0$$

and initial conditions $x_0 = x_1 = 1$. To solve a finite difference homogeneous equation, one should first look for its *fundamental solutions* – those of the type $x_t = \lambda^t$. Substituting $x_t = \lambda^t$ into the equation, we get a quadratic equation for λ:

$$\lambda^2 - \lambda - 1 = 0,$$

and we come to two fundamental solutions:

$$x_t^{(i)} = \lambda_i^t, \ i = 1, 2, \text{ with } \lambda_1 = \frac{1 + \sqrt{5}}{2} > 1, \ \lambda_2 = -1/\lambda_1.$$

Any linear combination of these fundamental solutions again is a solution to the equation, and to get $\{F_t\}$, it remains to choose the coefficients of the combination to fit the initial conditions $F_0 = F_1 = 1$. As a result, we come to (2.1.7). A surprise is that the expression for *integer* quantities F_t involves irrational number!

Consequently, (2.1.6) results in

$$|x^N - x^*| \leq \frac{\lambda+2}{\lambda+1}\lambda^{-N}|b-a|(1+o(1)), \tag{2.1.8}$$

where $o(1)$ denotes a function of N which converges to 0 as $N \to \infty$).

We see that the convergence ratio for the Fibonacci search is

$$\lambda^{-1} = \frac{2}{1+\sqrt{5}} = 0.61803...$$

which is much better than the ratio $\sqrt{2/3} = 0.81649...$ given by (2.1.4).

It can be proved that the Fibonacci search is, in certain precise sense, the *optimal*, in terms of the accuracy guaranteed after N function evaluations, zero-order method for minimizing an unimodal function on a segment. In spite of this fine theoretical property, the method is not that convenient from the practical viewpoint: we should choose in advance the number of function evaluations to be performed (i.e., to tune the method to certain chosen in advance accuracy), which is sometimes inconvenient. The *Golden search* method we are about to present is free of this shortcoming and at the same time is, for not too small N, basically as efficient as the original Fibonacci search.

2.1.2 Golden search

The idea of the method is very simple: at k-th step of the N-step Fibonacci search, we choose two search points in the segment Δ_{k-1}, and each of these points divides the segment (from the closer endpoint to the more far one) in the ratio

$$[1 - F_{N-k}/F_{N-k+1}] \,:\, [F_{N-k}/F_{N-k+1}]\,,$$

i.e., in the ratio $F_{N-k-1} : F_{N-k}$. According to (2.1.7), this ratio, for large $N-k$, is close to $1:\lambda$, $\lambda = (1+\sqrt{5})/2$. And in the Golden search implementation of Algorithm 2.1.1 we choose, at *every* step, the search points x_t^- and x_t^+ to divide the previous segment of uncertainty $\Delta_{t-1} = [a_{t-1}, b_{t-1}]$ in the ratio $1:\lambda$:

$$x_t^- = \frac{\lambda}{1+\lambda}a_{t-1} + \frac{1}{1+\lambda}b_{t-1}; \quad x_t^+ = \frac{1}{1+\lambda}a_{t-1} + \frac{\lambda}{1+\lambda}b_{t-1}. \tag{2.1.9}$$

Here again, for $t \geq 2$, one of the search points required to update Δ_{t-1} into Δ_t is already processed in course of updating Δ_{t-2} into Δ_{t-1}. To see it, it suffices to consider the case when $\Delta_{t-2} = [\alpha, \beta]$ and $\Delta_{t-1} = [\alpha, x_{t-1}^+]$ (the "symmetric" case $\Delta_{t-1} = [x_{t-1}^-, \beta]$ is completely similar). Denoting $d = \beta - \alpha$, we have

$$x_{t-1}^- = \alpha + \frac{1}{1+\lambda}d, \;\; x_{t-1}^+ = \alpha + \frac{\lambda}{1+\lambda}d; \tag{2.1.10}$$

now, we are in situation $\Delta_{t-1} = [\alpha, x_{t-1}^+]$, so that the second of the two search points needed to update Δ_{t-1} into Δ_t is

$$x_t^+ = \alpha + \frac{\lambda}{1+\lambda}(x_{t-1}^+ - \alpha) = \alpha + \frac{\lambda^2}{(1+\lambda)^2}d$$

(see the second equality in (2.1.10)). The latter quantity, due to the first equality in (2.1.10) and the characteristic equation $\lambda^2 = 1 + \lambda$ giving λ, is nothing but x_{t-1}^-:

$$\lambda^2 = 1 + \lambda \Leftrightarrow \frac{1}{1+\lambda} = \frac{\lambda^2}{(1+\lambda)^2}.$$

Thus, in the Golden search, same as in the Fibonacci search, each updating $\Delta_{t-1} \mapsto \Delta_t$, except the very first one, requires a single function evaluation, not two of them. The length of the uncertainty segment is reduced by every updating by factor

$$\frac{\lambda}{1+\lambda} = \frac{1}{\lambda},$$

i.e.,

$$|\Delta_t| = \lambda^{-t}(b-a).$$

After $N \geq 2$ function evaluations (i.e., after $t = N-1$ steps of the Golden search) we can approximate x^* by (any) point x^N of the resulting segment Δ_{N-1}, and inaccuracy bound will be

$$|x^N - x^*| \leq |\Delta_{N-1}| \leq \lambda^{1-N}(b-a). \tag{2.1.11}$$

Thus, we have the same linear rate of convergence (and with the same convergence ratio) as for the Fibonacci search, but now the method is "stationary" – we can perform as many steps of it as we wish.

2.2 Bisection

The theoretical advantage of the zero-order methods, like the Fibonacci search and the Golden search, is that these methods use the minimal information on the objective – its values only. Besides this, the methods have a very wide field of applications – the only requirement imposed on the objective is to be unimodal on a given segment whch localizes the minimizer to be approximated. And even under these extremely mild restrictions these methods are linearly converging with objective-independent converging ratio; moreover, the corresponding *efficiency estimates* (2.1.8) and (2.1.11) are non-asymptotical: they do not contain "uncertain" constant factors and are valid for all values of N. At the same time, typically our objective is better behaved than a general unimodal function, e.g., is smooth enough. Making use of these additional properties of the objective, we may improve the behaviour of the line search methods.

Let us look what happens if we are solving problem (2.1.1) with *smooth* – continuously differentiable – objective. Same as above, assume that the objective is unimodal on $[a, b]$. In fact we make a little bit stronger assumption:

(A): *the minimizer x^* of f on $[a, b]$ is an interior point of the segment, and $f'(x)$ changes its sign at x^*:*

$$f'(x) < 0, x \in [a, x^*); \quad f'(x) > 0, x \in (x^*, b]$$

[unimodality + differentiability imply only that $f'(x) \leq 0$ on $[a, x^*)$ and $f'(x) \geq 0$ on $(x^*, b]$].

Besides these restrictions on the problem, assume, as it is normally the case, that we are able to compute not only the value, but also the derivative of the objective at a given point.

Under these assumptions we can solve (2.1.1) by definitely the simplest possible method – the *bisection*. Namely, let us compute f' at the midpoint x_1 of $\Delta_0 = [a, b]$. There are three possible cases:

- $f'(x_1) > 0$. This case, according to (A), is possible if and only if $x^* < x_1$, and we can replace the initial segment of uncertainty with $\Delta_1 = [a, x_1]$, thus reducing the length of the uncertainty segment by factor 2;

- $f'(x_1) < 0$. Similarly to the previous case, this inequality is possible if and only if $x^* > x_1$, and we can replace the initial segment of uncertainty with $[x_1, b]$, again reducing the length of the uncertainty segment by factor 2;

- $f'(x_1) = 0$. Accodring to (A), it is possible if and only if $x_1 = x^*$, and we can terminate with exact minimizer at hand.

In the first two cases our objective clearly possesses property (A) with respect to the new segment of uncertainty, and we can iterate the construction. Thus, we come to

Algorithm 2.2.1 [Bisection]
Initialization: *set* $\Delta_0 = [a, b]$, $t = 1$
Step t: *Given previous uncertainty segment* $\Delta_{t-1} = [a_{t-1}, b_{t-1}]$,

- *define current search point* x_t *as the midpoint of* Δ_{t-1}*:*

$$x_t = \frac{a_{t-1} + b_{t-1}}{2};$$

- *compute* $f'(x_t)$*;*

- *in the case of* $f'(x_t) = 0$ *terminate and claim that* x_t *is the exact solution to (2.1.1). Otherwise set*

$$\Delta_t = \begin{cases} [a_{t-1}, x_t], & f'(x_t) > 0 \\ [x_t, b_{t-1}], & f'(x_t) < 0 \end{cases}$$

replace t *with* $t + 1$ *and loop.*

From the above considertaions we immediately conclude that

Proposition 2.2.1 [Linear convergence of Bisection]
Under assumption (A), for any $t \geq 1$*, either the Bisection search termninates in course of the first* t *steps with exact solution* x^**, or* t*-th uncertainty segment* Δ_t *is well-defined, covers* x^* *and is of the length* $2^{-t}(b - a)$*.*

Thus, the Bisection method converges linearly with convergence ratio 0.5*.*

Remark 2.2.1 The convergence reatio of the Bisection algorithm is better than the one 0.61803... for Fibonacci/Golden search. There is no contradiction with the announced optimality of the Fibonacci search: the latter is optimal among the *zero-order* methods for minimizing unimodal functions, while Bisection is a first-order method.

Remark 2.2.2 The Bisection method can be viewed as the "limiting case" of the conceptual zero-order Algorithm 2.1.1: when, in the latter algorithm, we make both the search points x_t^- and x_t^+ close to the midpoint of the uncertainty segment Δ_{t-1}, the result of comparison between $f(x_t^-)$ and $f(x_t^+)$ which governs the choice of the new uncertainty segment in Algorithm 2.1.1 is given by the sign of f' at the midpoint of Δ_{t-1}.

Remark 2.2.3 Note that the assumption (A) can be weakened. Namely, let us assume that f' changes its sign at the segment $[a,b]$: $f'(a)<0$, $f'(b)>0$; and we assume nothing about the derivative in (a,b), except its continuity. In this case we still can successfully use the Bisection method to approximate a *critical point* of f in (a,b), i.e., a point where $f'(x)=0$. Indeed, from the description of the method it is immediately seen that what we do is generating a sequence of enclosed segments $\Delta_0 \supset \Delta_1 \supset \Delta_2 \supset ...$, with the next segment being twice smaller than the previous one, with the property that f' changes it sign from $-$ to $+$ when passing from the left endpoint of every segment Δ_t to its right endpoint. This process can be terminated only in the case when the current iterate x_t is a critical point of f. If it does not happen, then the enclosed segments Δ_t have a unique common point x^*, and since in any neighbourhood of the point there are points both with positive and negative values of f', we have $f'(x^*)=0$ (recall that f' is continuous). This is the critical point of f to which the algorithm converges linearly with convergence ratio 0.5.

The indicated remark explains the nature of the bisection algorithm. This is an algorithm for finding zero of the function f' rather than for minimizing f itself (under assumption (A), of course, this is the same - to minimize f on $[a,b]$ or to find the zero of f' on (a,b)). And the idea of the algorithm is absolutely trivial: given that the zero of f' is *bracketed* by the initial uncertainty segment $\Delta_0=[a,b]$ (i.e., that f' at the endpoints of the segment is of different sign), we generate the sequence of enclosed segments, also bracketing zero of f', as follows: we split the previous segment $\Delta_t=[a_{t-1},b_{t-1}]$ by its midpoint x_t into two subsegments $[a_{t-1},x_t]$ and $[x_t,b_{t-1}]$. Since f' changes its sign when passing from a_{t-1} to b_{t-1}, it changes its sign either when passing from a_{t-1} to x_t, or when passing from x_t to b_{t-1} (provided that $f'(x_t)\neq 0$, so that we can speak about the sign of $f'(x_t)$; if $f'(x_t)=0$, we are already done). We detect on which of the two subsegments f' in fact changes sign and take it as the new uncertainty segment Δ_t; by construction, it also brackets a zero of f'.

2.3 Curve fitting

The line search methods considered so far possess, under unimodality assumption, nice *objective-independent global linear convergence*. May we hope for something better? Of course, yes: it would be fine to get something superlinearly converging. If the objective is "well-behaved" – smooth enough – we have good chances to accelerate convergence, at least at the final stage, by *curve fitting*, i.e., by approximating the objective by a simple function with analytically computable minimum. The natural way is to approximate f by a polynomial, choosing the coefficients of the polynomial in order to fit it to the observed values (and derivatives, if we are able to compute these derivatives) of f at several "most perspective" iterates. An iteration of a pure curve fitting algorithm is as follows:

- at the beginning of the iteration, we have certain set of "working points" where we already have computed the values and, possibly, certain derivatives of the objective. Given these data, we compute the *current approximating polynomial* p which should have the same values and derivatives at the working points as those of the objective
- after approximating polynomial p is computed, we find analytically its minimizer and take it as the new search point
- we compute the value (and, possibly, the derivatives) of the objective at the new search point and update the set of working points, adding to it the last search point (along with

the information on the objective at this point) and excluding from this set the "worst" of the previous working points, and then loop.

The idea underlying the outlined approach is very simple: if we somehow can enforce the procedure to converge, the working points will eventually be at certain small distance d from the minimizer of f. If f is smooth enough, the error in approximating f by p in the d-neighbourhood of working points will be of order of d^{q+1}, q being the degree of p, and the error in approximating f' by p' will be of order of d^q. Consequently, we may hope that the distance from the minimizer of p (i.e., the zero of p') to the minimizer of f (the zero of f') will be of order of d^q, which gives us good chances for superlinear convergence.

Of course, what is said is nothing but a very rough idea. Let uf look at several standard implementations.

2.3.1 Newton's method

Assume that we are solving problem (2.0.1) with twice continuously differentiable objective f, and that, given x, we can compute $f(x)$, $f'(x)$, $f''(x)$. Under these assumptions we can apply to the problem the *Newton* method as follows:

Algorithm 2.3.1 [One-dimensional Newton method]
Initialization: *choose somehow starting point* x_0
Step t: *given the previous iterate* x_{t-1},

- *compute* $f(x_{t-1}), f'(x_{t-1}), f''(x_{t-1})$ *and approximate* f *around* x_{t-1} *by its second-order Taylor expansion*

$$p(x) = f(x_{t-1}) + f'(x_{t-1})(x - x_{t-1}) + \frac{1}{2}f''(x_{t-1})(x - x_{t-1})^2;$$

- *choose as* x_t *the minimizer of the quadratic function* $p(\cdot)$*:*

$$x_t = x_{t-1} - \frac{f'(x_{t-1})}{f''(x_{t-1})},$$

replace t *with* $t + 1$ *and loop.*

The Newton method, *started close to a nondegenerate local minimizer* x^* *of* f (i.e., close to a point x^* satisfying the sufficient second order optimality condition: $f'(x^*) = 0$, $f''(x^*) > 0$), converges to x^* quadratically:

Proposition 2.3.1 [Local quadratic convergence of the Newton method]
Let $x^* \in \mathbf{R}$ *be a nondegenerate local minimizer of smooth function* f*, i.e., a point such that* f *is three times continuously differentiable in a neighbourhood of* x^* *with* $f'(x^*) = 0$, $f''(x^*) > 0$. *Then the Newton iterates converge to* x^* *quadratically, provided that the starting point* x_0 *is close enough to* x^*.

Proof. Let $g(x) = f'(x)$, so that $g(x^*) = 0$, $g'(x^*) > 0$ and

$$x_t = x_{t-1} - \frac{g(x_{t-1})}{g'(x_{t-1})}.$$

Since $g = f'$ is twice continuously differentiable in a neighbourhood of x^* and $g'(x^*) > 0$, there exist positive constants k_1, k_2, r such that

$$|x' - x^*|, |x'' - x^*| \leq r \Rightarrow |g'(x') - g'(x'')| \leq k_1|x' - x''|, \;\; g'(x') \geq k_2. \tag{2.3.1}$$

Now let

$$\rho = \min\{r; \frac{k_2}{k_1}\}. \tag{2.3.2}$$

Assume that for certain t the iterate x_{t-1} belongs to the ρ-neighbourhood

$$U_\rho = [x^* - \rho, x^* + \rho]$$

of the point x^*. Then $g'(x_{t-1}) \geq k_2 > 0$ (due to (2.3.1); note that $\rho \leq r$), so that the Newton iterate x_t of x_{t-1} is well-defined. We have

$$x_t - x^* = x_{t-1} - x^* - \frac{g(x_{t-1})}{g'(x_{t-1})} =$$

[since $g(x^*) = 0$]

$$x_{t-1} - x^* - \frac{g(x_{t-1}) - g(x^*)}{g'(x_{t-1})} = \frac{g(x^*) - g(x_{t-1}) - g'(x_{t-1})(x^* - x_{t-1})}{g'(x_{t-1})}.$$

The numerator in the resulting fraction is the reminder in the first order Taylor expansion of g at x_{t-1}; due to (2.3.1) and since $|x_{t-1} - x^*| \leq \rho \leq r$, it does not exceed in absolute value the quantity $\frac{1}{2}k_1|x^* - x_{t-1}|^2$. The denominator, by the same (2.3.1), is at least k_2. Thus,

$$x_{t-1} \in U_\rho \Rightarrow |x_t - x^*| \leq \frac{k_1}{2k_2}|x_{t-1} - x^*|^2. \tag{2.3.3}$$

Due to the origin of ρ, (2.3.3) implies that

$$|x_t - x^*| \leq |x_{t-1} - x^*|/2;$$

we see that the trajectory of the Newton method, once reaching U_ρ, never leaves this neighbourhood and converges to x^* linearly with convergence ratio 0.5. It for sure is the case when $x_0 \in U_\rho$, and let us specify the "close enough" in the statement of the proposition just as inclusion $x_0 \in U_\rho$. With this specification, we get that the trajectory converges to x^* linearly, and from (2.3.3) it follows that in fact the order of convergence is (at least) 2. ∎

Remark 2.3.1 Both the assumptions that $f''(x^*) > 0$ and that x_0 is close enough are essential[3)]. E.g., as applied to the smooth convex function

$$f(x) = x^4$$

(with degenerate minimizer $x^* = 0$), the method becomes

$$x_t = x_{t-1} - \frac{1}{3}x_{t-1} = \frac{2}{3}x_{t-1};$$

[3] in fact, the assumption $f''(x^*) > 0$ can be replaced with $f''(x^*) < 0$, since the trajectory of the method remains unchanged when f is replaced with $-f$ (in other words, the Newton method does not distinguish between the local minima and local maxima of the objective). We are speaking about the case of $f''(x^*) > 0$, not the one of $f''(x^*) < 0$, simply because the former case is the only important for minimization.

in this example the method converges, but linearly rather than quadratically.

As applied to strictly convex smooth function

$$f(x) = \sqrt{1+x^2}$$

with unique (and nondegenerate) local (and global as well) minimizer $x^* = 0$, the method becomes, as it is immediately seen,

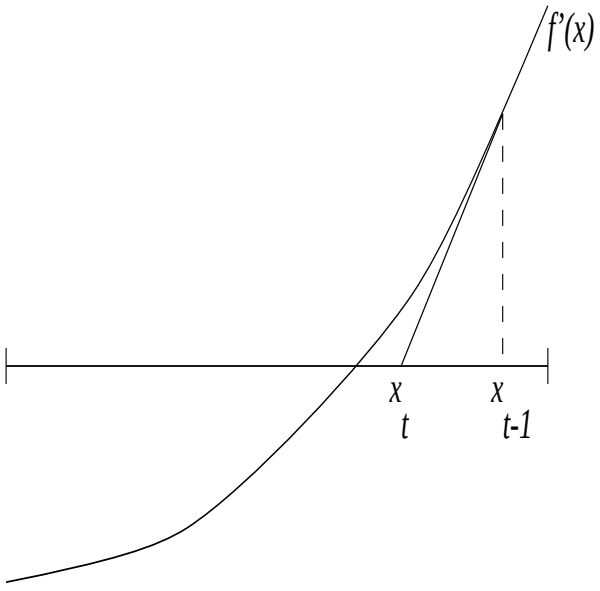
$$x_t = -x_{t-1}^3;$$

this procedure converges (very fast: with order 3) to 0 provided that the starting point is in $(-1, 1)$, and diverges to infinity – also very fast – if $|x_0| > 1$.

In fact the Newton method is a *linearization method for finding zero of f'*: given the previous iterate x_{t-1}, we linearize $g = f'$ at this point and take as x_t the solution to the linearization

$$g(x_{t-1}) + g'(x_{t-1})(x - x_{t-1}) = 0$$

of the actual equation $g(x) = 0$.



Newton method as zero-finding routine

2.3.2 Regula Falsi (False Position) method

This method, same as the Newton one, is based on approximating f by a quadratic polynomial, but here this polynomial is constructed via two working points with first-order information rather than via a single working point with second-order information. The method, in its most straightforward form, is as follows. Given two latest iterates x_{t-1}, x_{t-2}, along with the values f and f' at these iterates, we approximate $f''(x_{t-1})$ by the finite difference

$$\frac{f'(x_{t-1}) - f'(x_{t-2})}{x_{t-1} - x_{t-2}}$$

and use this approximation to approximate f by a quadratic function

$$p(x) = f(x_{t-1}) + f'(x_{t-1})(x - x_{t-1}) + \frac{1}{2}\frac{f'(x_{t-1}) - f'(x_{t-2})}{x_{t-1} - x_{t-2}}(x - x_{t-1})^2.$$

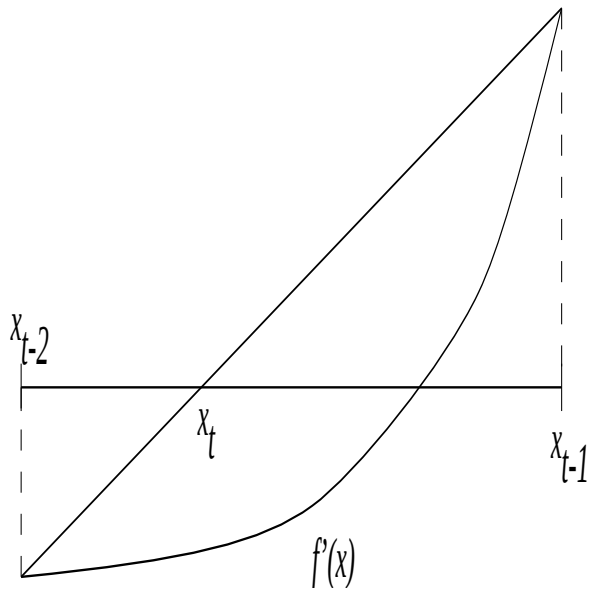
The new iterate is the minimizer of this quadratic function:

$$x_t = x_{t-1} - f'(x_{t-1})\frac{x_{t-1} - x_{t-2}}{f'(x_{t-1}) - f'(x_{t-2})}. \tag{2.3.4}$$

Note that although the polynomial p is chosen in asymmetric with respect to x_{t-1} and x_{t-2} way (it is tangent to f at x_{t-1}, but even not necessarily coincides with f at x_{t-2}), the minimizer x_t of this polynomial is symmetric with respect to the pair of working points; as it is immediately seen, the right hand side of (2.3.4) can be equivalently rewritten as

$$x_t = x_{t-2} - f'(x_{t-2})\frac{x_{t-1} - x_{t-2}}{f'(x_{t-1}) - f'(x_{t-2})}.$$

The geometry of the method is very simple: same as the Newton method, this is the method which actually approximates the zero of $g(x) = f'(x)$ (look: the values of f are not involved into the recurrency (2.3.4)). In the Newton method we, given the value and the derivative of g at x_{t-1}, approximate the graph of g by its tangent at x_{t-1} line $g(x_{t-1}) + g'(x_{t-1})(x - x_{t-1})$ and choose x_t as the point where this tangent line crosses the x-axis. In the Regula Falsi method we, given the values of g at two points x_{t-1} and x_{t-2}, approximate the graph of g by the secant line passing through $(x_{t-1}, g(x_{t-1}))$ and $(x_{t-2}, g(x_{t-2}))$ and choose as x_t the point where this secant line crosses the x-axis.



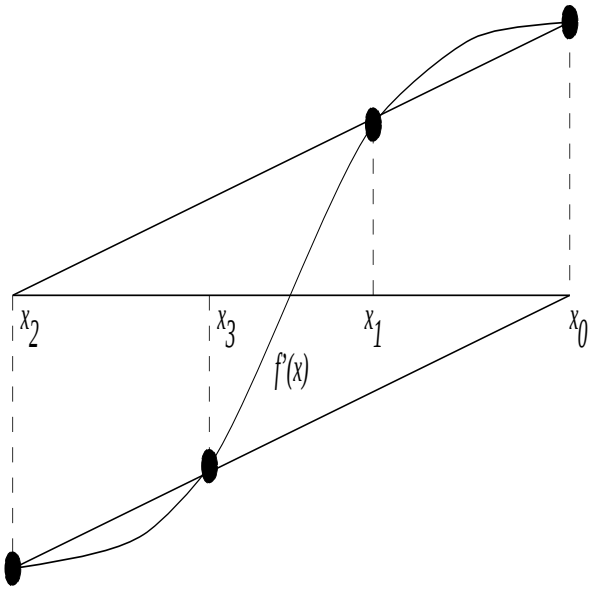
Regula Falsi method as zero-finding routine

The local rate of convergence of the method is given by the following

Proposition 2.3.2 [Local superlinear convergence of Regula Flasi method]
Let x^* be a nondegenerate minimizer of a smooth function f, namely, a point such that f is three times continuously differentiable in a neighbourhood of x^* with $f'(x^*) = 0$, $f''(x^*) > 0$. Then, for starting pair x_0, x_1 of two distinct points close enough to x^* the method converges to x^* superlinearly with order of convergence

$$\lambda = \frac{1 + \sqrt{5}}{2}.$$

Note that, same as for the Newton method, the assumptions of the proposition, especially the one of closeness of x_0 and x_1 to x^*, are essential: badly started, the method may be non-converging:



Cycle in Regula Falsi: the trajectory is $x_0, x_1, x_2, x_3, x_0, x_1, ...$

2.3.3 Cubic fit

Approximating polynomials used in curve fitting methods are of small order – not greater than 3, which is clear: to run the method, we should be able to compute the minimizer of a polynomial analytically. The *Cubic fit* is based on this "highest-order" approximation. At the step t of the method, given the latest two iterates $x' = x_{t-1}$ and $x'' = x_{t-2}$ along with the values of f and f' at the points, one defines the cubic polynomial $p(\cdot)$ such that

$$p(x') = f(x'),\ p'(x') = f'(x'),\ p(x'') = f(x''),\ p'(x'') = f'(x'')$$

(these are four linear equations on four coefficients of the polynomial; if $x' \neq x''$, which, as we shall see, always can be assumed, the equations uniquely define p). As the next iterate, one chooses the local minimizer of p. If exists, the minimizer is given by the relations

$$x_t = x' - (x' - x'')\left[\frac{u_1 + u_2 - f'(x')}{f'(x'') - f'(x') + 2u_2}\right],$$

$$u_1 = f'(x') + f'(x'') - 3\frac{f(x') - f(x'')}{x' - x''},\ \ u_2 = \sqrt{u_1^2 - f'(x')f'(x'')}. \qquad (2.3.5)$$

The step is for sure well-defined if $f'(x')$ and $f'(x'')$ are of opposite signs ("V-shape"; compare with Bisection). One can prove that if x^* is a nondegenerate local minimizer of a smooth enough function f, then the method, *started close enough to* x^*, converges to x^* quadratically.

2.3.4 Safeguarded curve fitting

There are many other curve fitting schemes: the one based on rational approximation of the objective, the one where the objective is approximated by a quadratic polynomial according to

its values at three-point working set, etc. All curve fitting methods have common advantage: superlinear local convergence for well-behaved functions. And all these methods have common disadvantage as well: if the method is started far from optimum, the method can diverge (see, e.g., Remark 2.3.1). To overcome this severe shortcoming, it is recommended to combine reliable linearly converging (in the case of unimodal objectives) methods like Golden search or Bisection with curve fitting. The idea is as follows. At step t of the method we have previous uncertainty segment $\Delta_{t-1} = [a_{t-1}, b_{t-1}]$ ($f'(x_{t-1}) < 0$, $f'(b_{t-1}) > 0$), as well as certain working set W_t comprised of several previous iterates where we know the values of f and, possibly, the derivatives of f. Same as in the curve fitting methods, we use the working set to form polynomial (or rational) approximation p to f and compute analytically the minimizer of p (or something close to this minimizer). In the curve fitting methods the resulting point, let it be called u, is used as the next iterate. In contrast to this, in the safeguarded curve fitting it is used as the next iterate only if u is "reasonably placed" with respect to Δ_{t-1}; if it is not the case, the next iterate is chosen according to the Bisection/Golden search scheme, depending on whether we use the derivatives of f.

The notion of a "reasonably placed" u contains several tests. First of all, u should be well-defined and belong to Δ_{t-1}, but this is not all. It is recommended, e.g., not to use u if it is too far from the best (with the smallest value of f) point of the working set, say, is at the distance larger than $\frac{1}{2}|\Delta_{t-1}|$ (since in this case the Bisection step is likely to reduce inaccuracy better than the curve fitting one). After the step – the curve fitting or the "safeguarding" (Bisection/Golden search) one – is performed, one updates the working set and the uncertainty segment.

Of course, what is said is not a description of a concrete method, but a very rough idea which admits many implementations (detailed specification of which curve fitting method to use, what is a "reasonably placed" u, rules for updating the working set, etc.) With good implementation, the safeguarded curve fitting combines the advantages of the Bisection/Golden search (global linear convergence with objective-independent rate in the unimodal case) and those of curve fitting (superlinear local convergence for well-behaved objectives).

2.4 Unexact Line Search

As it was already mentioned, the main application of line search methods is inside algorithms for multi-dimensional optimization. In these algorithms we always allow only small number of steps of the line search subroutine at each iteration of the master algorithm, otherwise the overall complexity of the master method will be too large. Moreover, in many multidimensional algorithms we are not that interested in high-accuracy solutions of one-dimensional subproblems; what is crucial for the master method, is reasonable progress in solving these subproblems. Whenever it is the case, we can terminate line search relatively far from the actual solution of the subproblem in question, using certain simple tests for "resonable progress". Let us describe two most popular tests of this type.

2.4.1 Armijo's rule

Consider the situation which is typical for application of line search technique inside multi-dimensional master method. At an iteration of the latter method we have *current iterate* $x \in \mathbf{R}^n$ and *search direction* $d \in \mathbf{R}^n$ which is a *descent* direction for our multivariate objective $f(\cdot) : \mathbf{R}^n \to \mathbf{R}$:

$$d^T \nabla f(x) < 0. \tag{2.4.1}$$

The goal is to reduce "essentially" the value of the objective by a step

$$x \mapsto x + \gamma^* d$$

from x in the direction d.

Assume that f is continuously differentiable. Then the function

$$\phi(\gamma) = f(x + \gamma d)$$

of one variable also is once continuously differentiable; moreover, due to (2.4.1), we have

$$\phi'(0) < 0,$$

so that for small positive γ one has

$$\phi(\gamma) - \phi(0) \approx \gamma\phi'(0) < 0.$$

Our desire is to choose a "'reasonably large" stepsize $\gamma^* > 0$ which results in the progress $\phi(\gamma^*) - \phi(0)$ in the objective "of order of $\gamma^*\phi'(0)$". The Armijo test for this requirement is as follows:

Armijo's Test:
we fix once for ever constants $\epsilon \in (0,1)$ (popular choice is $\epsilon = 0.2$) and $\eta > 1$ (say, $\eta = 2$ or $\eta = 10$) and say that candidate value $\gamma > 0$ is appropriate, if the following two relations are satisfied:

$$\phi(\gamma) \le \phi(0) + \epsilon\gamma\phi'(0) \tag{2.4.2}$$

[this part of the test says that the progress in the value of ϕ given by the stepsize γ is "of order of $\gamma\phi'(0)$"]

$$\phi(\eta\gamma) \ge \phi(0) + \epsilon\eta\gamma\phi'(0) \tag{2.4.3}$$

[this part of the test says that γ is "maximal in order" stepsize which satisfies (2.4.2) – if we multiply γ by η, the increased value fails to satisfy (2.4.2), at least to satisfy it as a strict inequality]

Under assumption (2.4.1) and the additional (very natural) assumption that f (and, consequently, ϕ) is below bounded, the Armijo test is *consistent*: there do exist values of $\gamma > 0$ which pass the test. To see it, it suffices to notice that

A. (2.4.2) is satisfied for all small enough positive γ.
Indeed, since ϕ is differentiable, we have

$$0 > \phi'(0) = \lim_{\gamma \to +0} \frac{\phi(\gamma) - \phi(0)}{\gamma},$$

whence

$$\epsilon\phi'(0) \ge \frac{\phi(\gamma) - \phi(0)}{\gamma}$$

for all small enough positive γ (since $\epsilon\phi'(0) > \phi'(0)$ due to $\phi'(0) < 0, \epsilon \in (0,1)$); the resulting inequality is equivalent to (2.4.2);

B. (2.4.2) is not valid for all large enough values of γ.
Indeed, the right hand side of (2.4.2) tends to $-\infty$ as $\gamma \to \infty$, due to $\phi'(0) < 0$, and the left hand side is assumed to be below bounded.

Now let us choose an arbitrary positive $\gamma = \gamma_0$ and test whether it satisfies (2.4.2). If it is the case, let us replace this value sequentially by $\gamma_1 = \eta\gamma_0$, $\gamma_2 = \eta\gamma_1$, etc., each time verifying whether the new value of γ satisfies (2.4.2). According to **B**, this cannot last forever: for certain $s \geq 1$ γ_s for sure fails to satisfy (2.4.2). When it happens for the first time, the quantity γ_{s-1} turns out to satisfy (2.4.2), while the quantity $\gamma_s = \eta\gamma_{s-1}$ fails to satisfy (2.4.2), which means that $\gamma = \gamma_{s-1}$ passes the Armijo test.

If the initial γ_0 does not satisfy (2.4.2), we replace this value sequentially by $\gamma_1 = \eta^{-1}\gamma_0$, $\gamma_2 = \eta^{-1}\gamma_1$, etc., each time verifying whether the new value of γ still does not satify (2.4.2). According to **A**, this cannot last forever: for certain $s \geq 1$, γ_s for sure satisfies (2.4.2). When it happens for the first time, γ_s turns out to satisfy (2.4.2), while $\gamma_{s-1} = \eta\gamma_s$ fails to satisfy (2.4.2), and $\gamma = \gamma_s$ passes the Armijo test.

Note that the presented proof in fact gives an explicit (and fast) algorithm for finding a stepsize passing the Armijo test, and this algorithm can be used (and often is used) in Armijo-aimed line search instead of more accurate (and normally more time-consuming) line search methods from the previous sections.

2.4.2 Goldstein test

Another popular test for "sufficient progress" in line search is as follows:

Goldstein test:
we fix one for ever constant $\epsilon \in (0, 1/2)$ and say that candidate value $\gamma > 0$ is appropriate, if

$$\phi(0) + (1-\epsilon)\gamma\phi'(0) \leq \phi(\gamma) \leq \phi(0) + \epsilon\gamma\phi'(0). \tag{2.4.4}$$

Here again relation (2.3.4) and below boundedness of f imply consistency of the test.

2.4.3 Closedness of the line search mappings

To be able to apply the general convergence analysis scheme to the algorithms which use line search we need to know something about closedness of the mappings given by line search. To this end the following three statements will be useful:

Proposition 2.4.1 [Closedness of the mapping given by exact line search]
Let $f : \mathbf{R}^n \to \mathbf{R}$ be a continuous function with compact level sets: $f(x) \to \infty$ as $|x| \to \infty$. Then the point-to-set mapping

$$S(x, d) : \mathbf{R}^n \times \mathbf{R}^n \to \mathbf{R}^n$$

given by

$$S(x, d) = \mathrm{Argmin}\{f(y) \mid y = x + \gamma d, \gamma \geq 0\}$$

(i.e., the image of a pair "point $x \in \mathbf{R}^n$, direction $d \in \mathbf{R}^n$" under the mapping which puts into correspondence to (x, d) the set of minimizers of f on the ray $\{x + \gamma d \mid \gamma \geq 0\}$) is closed at any point (x, d) with $d \neq 0$.

Proposition 2.4.2 [Closedness of the mapping given by line search with Armijo termination]
Let $f : \mathbf{R}^n \to \mathbf{R}$ be a continuously differentiable function which is below bounded on $\mathbf{R}^n$, let $\epsilon \in (0, 1)$, and let $\eta > 1$. Then the point-to-set mapping

$$S(x, d) : \mathbf{R}^n \times \mathbf{R}^n \to \mathbf{R}^n$$

given by

$$S(x,d) = \{y = x+\gamma d \mid \gamma > 0,\ f(x+\gamma d) \le f(x)+\epsilon\gamma d^T\nabla f(x),\ f(x+\eta\gamma d) \ge f(x)+\epsilon\eta\gamma d^T\nabla f(x)\}$$

(i.e., the image of the pair "point $x \in \mathbf{R}^n$, direction $d \in \mathbf{R}^n$" under the mapping which puts into correspondence to (x,d) the set of all points on the ray $\{x+\gamma d \mid \gamma > 0\}$ with γ satisfying the Armijo test) is well-defined (i.e., $S(x,d) \neq \emptyset$) and is closed at any point (x,d) such that d is a descent dirrection of f at x, i.e., $d^T\nabla f(x) < 0$.

Proposition 2.4.3 [Closedness of the mapping given by line search with Goldstein termination] *Let $f : \mathbf{R}^n \to \mathbf{R}$ be a continuously differentiable function which is below bounded on $\mathbf{R}^n$, and let $\epsilon \in (0, 1/2)$. Then the mapping*

$$S(x,d) : \mathbf{R}^n \times \mathbf{R}^n \to \mathbf{R}^n$$

given by

$$S(x,d) = \{y = x+\gamma d \mid \gamma > 0,\ f(x) + (1-\epsilon)\gamma d^T\nabla f(x) \le f(x+\gamma d) \le f(x) + \epsilon\gamma d^T\nabla f(x)\}$$

(i.e., the image of the pair "point $x \in \mathbf{R}^n$, direction $d \in \mathbf{R}^n$" under the mapping which puts into correspondence to (x,d) the set of all points on the ray $\{x+\gamma d \mid \gamma > 0\}$ with γ satisfying the Goldstein test) is closed at any point (x,d) such that d is a descent dirrection of f at x, i.e., $d^T\nabla f(x) < 0$.

Proof of Proposition 2.4.1. First of all, since f is continuous and tends to ∞ as $|x| \to \infty$, f attains its minimum on every closed nonempty subset of $\mathbf{R}^n$ (why?), so that $S(x,d)$, for every pair (x,d), is a nonempty subset of $\mathbf{R}^n$, as is required by the definition of a point-to-set mapping. It remains to prove that if (x,d) is such that $d \neq 0$, then $S(\cdot,\cdot)$ is closed at (x,d). In other words, we should prove that if $x_t \to x$, $d_t \to d$ and $y_t \in S(x_t,d_t)$, $y_t \to y$ as $t \to \infty$, then $y \in S(x,d)$, i.e., y is a minimizer of f on the ray $R = \{x+\gamma d \mid \gamma \ge 0\}$.

Let $R_t = \{x_t+\gamma d_t \mid \gamma \ge 0\}$, so that y_t is a minimizer of f on R_t, and let $y_t = x_t + \gamma_t d_t$. Since $\gamma_t d_t = y_t - x_t \to y - x$ and $d_t \to d \neq 0$ as $t \to \infty$, we see that the sequence $\{\gamma_t\}$ has a limit γ as $t \to \infty$ (look at the quantities $|y_t - x_t| = \gamma_t|d_t|$). Consequently,

$$y = \lim_{t\to\infty}[x_t + \gamma_t d_t] = x + \gamma d;$$

since $\gamma_t \ge 0$, we have also $\gamma \ge 0$, so that $y \in R$.

It remains to verify that y is a minimizer of f on R. Let $y' = x + \gamma' d$, $\gamma' \ge 0$, be a point of R. Setting $y'_t = x_t + \gamma' d_t$, we have

$$y'_t \to y',\ t \to \infty,$$

whence, due to continuity of f,

$$f(y'_t) \to f(y'),\ t \to \infty.$$

Since $y'_t \in R_t$ and y_t is a minimizer of f on R_t, we have

$$f(y_t) \le f(y'_t).$$

As $t \to \infty$, the right hand side here, as we just have seen, converges to $f(y')$, while the left hand one converges to $f(y)$ (recall that y_t converge to y and f is continuous). We conclude that

$f(y) \le f(y')$, and this is true for an arbitrary point $y' \in R$. Thus, y indeed is a minimizer of f on R. ∎

Proof of Proposition 2.4.2. Let (x, d) be such that $d^T \nabla f(x) < 0$. As we know from the discussion devoted to the Armijo rule, $S(x, d)$ is nonempty, and all we need is to prove closedness of the mapping. Thus, let

$$x_t \to x,\ d_t \to d,\ y_t \to y,\ t \to \infty \quad [y_t \in S(x_t, d_t)], \tag{2.4.5}$$

and let us prove that $y \in S(x, d)$.

We have $y_t = x_t + \gamma_t d_t$, with $\gamma_t > 0$ satisfying the corresponding Armijo test:

$$f(x_t + \gamma_t d_t) \le f(x_t) + \epsilon\gamma_t d_t^T \nabla f(x_t); \quad f(x_t + \eta\gamma_t d_t) \ge f(x_t) + \epsilon\eta\gamma_t d_t^T \nabla f(x_t). \tag{2.4.6}$$

Since $d^T \nabla f(x) < 0$ by asumption, so that $d \neq 0$, we, as in the proof of Proposition 2.4.1, have $\gamma_t \to \gamma$; this observation combined with (2.4.5) and (2.4.6), in view of continuous differentiability of f, implies that

$$f(x + \gamma d) \le f(x) + \epsilon\gamma d^T \nabla f(x); \quad f(x + \eta\gamma d) \ge f(x) + \epsilon\eta\gamma d^T \nabla f(x). \tag{2.4.7}$$

This is what we need, but not all we need: in the definition of Armijo-acceptable stepsize γ, besides (2.4.7), it is required $\gamma > 0$ (zero stepsize trivially satisfies (2.4.2) - (2.4.3), but is of no interest). By its origin, γ is the limit of a sequence of positive reals γ_t, so that it is nonnegative. But why it is positive? This is the focus point of the proof.

To prove positivity of γ, it suffices to prove that γ_t are bounded away from zero (are $\ge \delta$ for properly chosen $\delta > 0$). To this end let us set

$$\alpha = -d^T \nabla f(x) > 0$$

and let us note that the function $h^T \nabla f(z)$ is continuous in h, z (since f is continuously differentiable); at the point $h = d, z = x$ this function equals to $-\alpha < 0$ and therefore there exists positive ρ such that in the ρ-neighbourhood of d, x the function is $\ge -\alpha\zeta$ and $\le -\alpha\zeta^{-1}$:

$$|z - x| + |h - d| \le \rho \Rightarrow -\alpha\zeta \le h^T \nabla f(z) \le -\alpha\zeta^{-1}, \tag{2.4.8}$$

where $\zeta > 1$ is chosen close enough to 1 to satisfy the inequality

$$\epsilon\zeta^2 < 1. \tag{2.4.9}$$

From (2.4.5) it follows that large enough T and all $t \ge T$ one has

$$|x_t - x| + |d_t - d| \le \rho/2; \quad |d|/2 \le |d_t| \le 2|d|. \tag{2.4.10}$$

I claim that if $t \ge T$, then $\gamma_t \ge 0.25|d|^{-1}\eta^{-1}\rho$ (in particular, $\{\gamma_t\}$ are bounded away from 0, as required). Indeed, assume, on contrary, that for certain $t \ge T$ one has

$$0 \le \gamma_t \le 0.25|d|^{-1}\eta^{-1}\rho, \tag{2.4.11}$$

and let us lead this assumption to a contradiction.

By the Lagrange Mean Value Theorem we have

$$f(x_t + \eta\gamma_t d_t) - f(x_t) = \eta\gamma_t d_t^T \nabla f(x'), x' = x_t + \theta\eta\gamma_t d_t$$

for certain $\theta \in (0,1)$. In view of (2.4.10) and (2.4.11) the pair $h = d_t, z = x'$ satisfies the premise in (2.4.8), so that in view of (2.4.8) one has

$$f(x_t + \eta\gamma_t d_t) - f(x_t) \leq -\eta\gamma_t\zeta^{-1}\alpha.$$

By similar reasoning applied to $h = d_t, z = x_t$,

$$d_t^T \nabla f(x_t) \geq -\alpha\zeta.$$

Combining the resulting inequalities, we come to

$$f(x_t + \eta\gamma_t d_t) - f(x_t) \leq \eta\gamma_t\zeta^{-2} d_t^T \nabla f(x_t) < \epsilon\eta\gamma_t d_t^T \nabla f(x_t)$$

(note that $\epsilon < \zeta^{-2}$ by (2.4.9), while $\eta\gamma_t > 0$ and $d_t^T \nabla f(x_t) < 0$). The resulting inequality contradicts the fact that γ_t passes the Armijo test. ∎

Proof of Proposition 2.4.3 is completely similar to the one of Proposition 2.4.2.

Assignment # 2 (Lecture 2)

Exercise 2.4.1 [Golden search] *Write a code implementing the Golden search and run it on several unimodal test functions on your choice.*

Exercise 2.4.2 [Bisection] *Write a code implementing the Bisection and run it on several unimodal test functions on your choice.*

Run 50 steps of the Bisection algorithm on the (non-unimodal) function

$$f(x) = -\sin\left(\frac{2\pi}{\frac{2}{17}+x}\right) \quad [x \geq 0]$$

with the initial uncertainty segments (a) $[0,1]$*; (b)* $[0,4]$*, taking as the result the midpoint of the final uncertainty segment. Why the results are different?*

Exercise 2.4.3 [Golden search vs Bisection] Assume that the problem (2.1.1) to be solved satisfies assumption (A) (Section 2.2), and that the derivatives of the objective are available. What should be prefered – the Golden search or the Bisection?

Of course, Bisection has better convergence (convergence ratio 0.5 versus 0.618... for the Golden search), but this comparison is unfair: the Golden search does not use derivatives, and switching off the part of the code which computes f', normally, save the overall computation time, in spite of larger # of steps, to the same accuracy, in the Golden search.

The actual reason to prefer Bisection is that this method, normally, is more numerically stable. Indeed, assume that we are solving (2.1.1) and everything – the values of f, f', f'' in $[a,b]$, same as a and b themselves, are "normal reals" – those of order of 1. And assume that we are interested in reducing the initial uncertainty segment to the one of length ϵ. What are the accuracies ϵ we can achieve in actual (unexact) computations?

The rough reasoning is as follows: to run the Golden search, we should compare values of the objective, at the final steps – at points at the distance $O(\epsilon)$ from the minimizer. At these points, the values of f differ from the optimal value (and, consequently, from each other) by $O(\epsilon^2)$. In order to ensure correct comparison of the values (and an incorrect one may make all the subsequent computations senseless), the absolute inaccuracy ϵ^* of machine representation of a number of order of 1 (for double precision Fortran/C computations ϵ^* is something like 10^{-16}) should be less than the above $O(\epsilon^2)$. Thus, the values of ϵ we indeed can achieve in Golden search should be of order of $O(\sqrt{\epsilon^*})$.

In the Bisection method, we should compare the values of f' with 0; if all intermediate results in the code computing the derivative are of order of 1, the derivative is computed with absolute inaccuracy $\leq c\epsilon^*$, with certain constant c. If $f''(x^*)$, x^* being the minimizer of f on $[a,b]$, is positive of order of 1 ("the minimizer is numerically well-conditioned"), then at the distance $\geq C\epsilon$ away from x^* the actual values of f' are, in absolute values, at least $C'\epsilon$, C' being certain constant. We see that if x is at the distance ϵ away from x^* and ϵ is such that $C'\epsilon > c\epsilon^*$ (i.e., the magnitude of $f'(x)$ is greater than the absolute error in computing $f'(x)$), then the sign of the actually computed $f'(x)$ will be the same as the exact sign of $f'(x)$, and the bisection step will be correct. Thus, under the above assumptions we can expect that the Bisection will be able to reach accuracy $\epsilon = c(C')^{-1}\epsilon^* = O(\epsilon^*)$ (compare with $O(\sqrt{\epsilon^*})$ for the Golden search).

In order to test this reasoning, I run the Golden search and the Bisection on the problem

$$f(x) = (x+1)^2 \to \min \mid -2 \leq x \leq 1.$$

To my surprise (I am unexperienced in error analysis!), both the methods solved the problem within accuracy of order of 10^{-16}. After a short thought, I realized what was wrong and was able to update the objective to observe the outlined phenomenon.

Could you

a) Guess what is wrong in the indicated example?

b) Correct the example and observe the phenomenon?

Exercise 2.4.4 [Newton's method] *Run the Newton minimization method at the functions*

1) $f(x) = \frac{1}{2}x^2 - x - \frac{1}{2}\exp\{-2x\}$ *(starting point* 0.5*)*

2) $f(x) = x^4 \exp\{-x/6\}$ *(starting point* 1.0*)*

Exercise 2.4.5 [Safeguarded cubic fit] *Try to implement the Safeguarded Polynomial Approximation approach (Section 2.3.4) in the following situation:*

- *the problem is*
$$f(x) \to \min \mid x \in [a, b],$$
with $[a, b]$ *being an uncertainty segment:* $f'(a) < 0$, $f'(b) > 0$*;*
- *the data available at a search point* x *are* $f(x), f'(x)$*;*
- *the curve to be fitted is cubic polynomial which coincides, along with the first-order derivative, with* f *on two-point working set.*

The task is a little bit diffuse, and you are welcome to test your own ideas.

If I were you, I would start with testing the following simple scheme:

1^0. Given current uncertainty segment $\Delta_{t-1} = [a_{t-1}, b_{t-1}]$, with $f'(a_{t-1}) < 0$, $f'(b_{t-1}) > 0$, and the working set a_{t-1}, b_{t-1} (so that we know the values of f and f' at a_{t-1} and b_{t-1}), we compute the cubic polynomial p with the same as those of f values and first order derivatives at the working set, and compute its (unique) local minimizer lying in Δ_{t-1}, let this minimizer be u (explicit formulae for p and u are given in Section 2.3.3).

2^0. Two cases are possible:

l) u is in the left half of the segment Δ_{t-1} [in this case let us say that a_{t-1} is the "perspective" endpoint of the segment Δ_{t-1}]

r) u is in the right half of the segment Δ_{t-1} [in this case we say that b_{t-1} is the prespective endpoint of Δ_{t-1}]

In the case of l) let us choose, as the next search point,

$$x^+ = u + \alpha(u - a_{t-1}),$$

and in the case of r) – the point

$$x^+ = u - \alpha(b_{t-1} - u),$$

$\alpha > 0$ being small positive real (I would try $\alpha = 0.01$ or smth. like).

3^0. After $f(x^+)$ and $f'(x^+)$ are computed, we check whether the sign of $f'(x^+)$ is *opposite* to the sign of f' at the perspective endpoint, let it be denoted x^-, of the segment Δ_{t-1}. If it is so (*good case*), we take, as Δ_t, the segment with the endpoints x^- and x^+; note that f' changes its sign from $-$ to $+$ when passing from the left endpoint of Δ_{t-1} to the right endpoint, as it is required for an uncertainty segment, and that the length of Δ_t is at most $(0.5 + \alpha)$ times the length of the previous uncertainty segment Δ_{t-1}.

It is possible, of course, that the sign of f' at x^+ is the same as at x^- (*bad case*). In the case of l) it means that $f'(x^+)$ and $f'(b_{t-1})$ are of different signs (since $f'(a_{t-1}) \equiv f'(x^-)$ is of the same sign as $f'(x^+)$, and the signs of f' at a_{t-1} and b_{t-1} are different), and we perform an additional bisection step in the segment $[x^+, b_{t-1}]$ to get a twice smaller uncertainty segment Δ_t. In the case of r) we, similarly, perform an additional bisection step in the segment $[a_{t-1}, x^+]$ to get a new uncertainty segment Δ_t.

We see that in one (in the good case) or two (in the bad one) function evaluations we come from a given uncertainty segment to a new uncertainty segment with smaller, by factor at least $0.5 + \alpha$ (in the good case) or 0.5 (in the bad one), length. Consequently, the scheme in question is "safeguarded" – it converges linearly with objective-independent rate.

Now let me explain why this procedure seems to have high convergence rate at the final stage. Consider step t and assume that f is "well-behaved" and that we are at the final stage of the process, so that p is good enough approximation of f on Δ_{t-1}. Let x^* be the minimizer of f on Δ_{t-1}, and let, for the sake of definiteness, a_{t-1} be the one of the endpoints of the uncertainty segment which is closer to x^*. Under these assumptions we may expect that

a) u is much closer to x^* than a_{t-1}, and since x^* is in the left half of Δ_{t-1}, u typically also will be in this left half, so that the case l) will occur;

b) due to the "extra step" ($x^+ = u + \alpha(u - a_{t-1})$ instead of $x^+ = u$), our next iterate will be to the right of x^*, so that we will meet with the good case, and no additional bisection step will be needed;

c) The new iterate x^+ will be "almost" as close to x^* as u (α is small!), so that the method will behave itself "almost" as the pure curve fitting scheme, which, in good cases, possesses fast local convergence.

Note that the "extra step" mentioned in b) is aimed exactly to have good chances to get, without additional bisection step, new uncertainty segment "sufficiently smaller" than the previous one, so that the safeguarded policy in the above scheme is "distributed" between this extra step and additional bisection steps.

I do not think that the indicated scheme is "the best one"; it is nothing but what came to my mind. You are welcome to play with this scheme or to invent something better!

Lecture 3

Gradient Descent

Starting from this lecture, we shall speak about methods for solving unconstrained multidimensional problems

$$f(x) \to \min \mid x \in \mathbf{R}^n. \tag{3.0.1}$$

From now on, let us make the following assumptions:

- (A) the objective f in (3.0.1) is continuously differentiable;
- (B) the problem in question is solvable: the set

$$X^* = \operatorname*{Argmin}_{\mathbf{R}^n} f$$

  is nonempty.

Our today lecture is devoted to the oldest and most widely known method for (3.0.1) - the *Gradient Descent*.

3.1 The method

3.1.1 The idea

The idea of the method is very simple. Assume that we are at certain point x, and that we have computed $f(x)$ and $\nabla f(x)$. Assume that x is not a critical point of f: $\nabla f(x) \neq 0$ (this is the same as to say that x is not a Karush-Kuhn-Tucker point of the problem). Then $g = -\nabla f(x)$ is a *descent* direction of f at x:

$$\frac{d}{d\gamma}|_{\gamma=0} f(x - \gamma \nabla f(x)) = -|\nabla f(x)|^2 < 0;$$

moreover, this is the *best* among the descent directions h (normalized to have the same length as that one of g) of f at x: for any h, $|h| = |g|$, one has

$$\frac{d}{d\gamma}|_{\gamma=0} f(x + \gamma h) = h^T \nabla f(x) \geq -|h||\nabla f(x)| = -|\nabla f(x)|^2$$

(we have used tha Cauchy inequality), the inequality being equality if and only if $h = g$.

The indicated observation demonstrates that in order to improve x – to form a new point with smaller value of the objective – it makes sense to perform a step

$$x \mapsto x + \gamma g \equiv x - \gamma \nabla f(x)$$

from x in the antigradient direction; with properly chosen stepsize $\gamma > 0$, such a step will for sure decrease f. And in the Gradient Descent method, we simply iterate the above step. Thus, the generic scheme of the method is as follows

Algorithm 3.1.1 [Generic Gradient Descent]
Initialization: *choose somehow starting point* x_0 *and set* $t = 1$
Step t: *at the beginning of step* t *we have previous iterate* x_{t-1}. *At the step we*

- *compute* $f(x_{t-1})$ *and* $\nabla f(x_{t-1})$*;*
- *choose somehow a positive stepsize* γ_t*, set*

$$x_t = x_{t-1} - \gamma_t \nabla f(x_{t-1}), \tag{3.1.1}$$

replace t *with* $t + 1$ *and loop.*

Thus, the generic Gradient Descent method is simply the recurrency (3.1.1) with certain rule for choosing stepsizes $\gamma_t > 0$; normally, the stepsizes are given by a kind of line search applied to the univariate functions

$$\phi_t(\gamma) = f(x_{t-1} - \gamma \nabla f(x_{t-1})).$$

3.1.2 Standard implementations

Different versions of line search result in different versions of the Gradient Descent method. Among these versions, one should mention

- ArD [Gradient Descent with Armijo-terminated line search]: the stepsize $\gamma_t > 0$ at iteration t where $\nabla f(x_{t-1}) \neq 0$ is chosen according to the Armijo test (Section 2.4.1):

$$\begin{aligned} f(x_{t-1} - \gamma_t \nabla f(x_{t-1})) &\leq f(x_{t-1}) - \epsilon \gamma_t |\nabla f(x_{t-1})|^2; \\ f(x_{t-1} - \eta \gamma_t \nabla f(x_{t-1})) &\geq f(x_{t-1}) - \epsilon \eta \gamma_t |\nabla f(x_{t-1})|^2, \end{aligned} \tag{3.1.2}$$

  $\epsilon \in (0, 1)$ and $\eta > 1$ being the parameters of the method. And if x_{t-1} is a critical point of f, i.e., $\nabla f(x_{t-1}) = 0$, the choice of $\gamma_t > 0$ is absolutely unimportant: independently of the value of γ_t, (3.1.1) will result in $x_t = x_{t-1}$.

- StD [Steepest Descent]: γ_t minimizes f along the ray $\{x_{t-1} - \gamma \nabla f(x_{t-1}) \mid \gamma \geq 0\}$:

$$\gamma_t \in \operatorname*{Argmin}_{\gamma \geq 0} f(x_{t-1} - \gamma \nabla f(x_{t-1})). \tag{3.1.3}$$

Of course, the Steepest Descent is a kind of idealization: in nontrivial cases we are unable to minimize the objective along the search ray *exactly*. Moreover, to make this idealization valid, we should assume that the corresponding steps are well-defined, i.e., that

$$\operatorname*{Argmin}_{\gamma \geq 0} f(x - \gamma \nabla f(x)) \neq \emptyset$$

for every x; in what follows, this is assumed "by default" whenever we are speaking about the Steepest Descent.

In contrast to the Steepest Descent, the Gradient Descent with Armijo-terminated line search is quite "constructive" – we know from Section 2.4.1 how to find a stepsize passing the Armijo test.

3.2 Convergence of the Gradient Descent

3.2.1 General Convergence Theorem

We start with the following theorem which establishes, under very mild restrictions, global convergence of the Gradient Descent to the set of *critical points* of f – to the set

$$X^{**} = \{x \in \mathbf{R}^n \mid \nabla f(x) = 0\}.$$

Theorem 3.2.1 [Global convergence of Gradient Descent] *For both* StD *and* ArD *, the following statements are true:*

(i) *If the trajectory $\{x_t\}$ of the method is bounded, then the trajectory possesses limiting points, and all these limiting points are critical points of f;*

(ii) *If the level set*

$$S = \{x \in \mathbf{R}^n \mid f(x) \leq f(x_0)\}$$

of the objective is bounded, then the trajectory of the method is bounded (and, consequently, all its limiting points, by (i)*, belong to X^{**}).*

Proof. (ii) is an immediate consequence of (i), since both ArD and StD clearly are *descent* methods:

$$x_t \neq x_{t-1} \Rightarrow f(x_t) < f(x_{t-1}). \tag{3.2.1}$$

Therefore the trajectory, for each of the methods, is contained in the level set S; since under assumption of (ii) this set is bounded, the trajectory also is bounded, as claimed in (ii).

It remains to prove (ii). Let us start with ArD . To prove its convergence to X^{**}, let us apply the general scheme for convergence analysis from Lecture 1.

1^0. First of all, let us bring the method into the framework required by our general scheme. The method is memoryless, so that we can cover it by the abstract iterative algorithm

$$\mathcal{A} = (X = \mathbf{R}^n; \Theta(\cdot)),$$

where the point-to-set mapping Θ is defined as follows:

$$\Theta(x) = \begin{cases} \{y \mid y = x - \gamma \nabla f(x) \text{ with } \gamma \text{ satisfying the Armijo test}\} & ,\nabla f(x) \neq 0 \\ \{x\} & ,\nabla f(x) = 0 \end{cases}$$

Now, the algorithm $\mathcal{A}$ is descent with respect to the solution set $\Gamma = X^{**}$, the descent finction being $Z(x) = f(x)$, as is immediately seen from (3.1.2). Let us prove that the mapping $\Theta(\cdot)$ is closed at any point $x \notin \Gamma$. To this end let us note that Θ can be represented as composition of two mappings:

$$\Theta = SD,$$

where

- the inner mapping D is the point-to-point mapping from $\mathbf{R}^n$ to $\mathbf{R}^n \times \mathbf{R}^n$ given by

$$D(x) = (x, -\nabla f(x));$$

- the outer mapping S is the point-to-set mapping from $\mathbf{R}^n \times \mathbf{R}^n$ into $\mathbf{R}^n$ given by

$$S(x,d) = \begin{cases} \{y = x + \gamma d \text{ with } \gamma \text{ satisfying the Armijo test}\} & ,d^T \nabla f(x) < 0 \\ \{x\} & ,d^T \nabla f(x) \geq 0 \end{cases}$$

The inner mapping is a continuous point-to-point mapping (look at assumption (A) in the beginning of the lecture). Now, if $x \notin X^{**}$, then $D(x) = (x, d \equiv -\nabla(x))$ is such that d is a descent direction of f at x: $d^T \nabla f(x) < 0$. By Proposition 2.4.2, the outer mapping S is closed at the point $D(x)$. Consequently, by Corollary 1.3.2, the composite mapping Θ is closed at x.

Now the desired convergence statement is readily given by the Global Convergence Theorem 1.3.1 (we just have seen that the assumptions (ii) and (iii) of Theorem 1.3.1 are satisfied, and the remaining assumption (i) of the Theorem is nothing but the premise in the statement we are proving).

The proof of (i) for the case of StD is completely similar; the only difference is that we should refer to Proposition 2.4.1 instead of Proposition 2.4.2. ∎

3.2.2 Limiting points of Gradient Descent

We have proved that the standard versions of the Gradient Descent, under the assumption that the trajectory is bounded, converge to the set X^{**} of critical points of the objective. This set for sure contains the set X^* of global minimizers of f, same as the set of local minimizers of the objective, but this is not all: X^{**} contains also all local maximizers of f and the saddle points of the function, if any exists. An important question is whether a limiting point of the trajectory of the Gradient Descent may be something we are not interested in – a a critical point which is *not* a local minimizer of the objective.

To understand what is going on, let us consider the case of very smooth (say, infinitely many times differentiable) f with bounded level set $S = \{x \mid f(x) \le f(x_0)\}$. Since both the versions under consideration – ArD and StD – are descent methods, the trajectory remains in S; thus, all what is important for us is the the behaviour of f in S.

In the case of "general position"[1] f possesses a finite set of critical points in S, and all these points are nondegenerate, i.e., with a nondegenerate Hessian $\nabla^2 f$. Let x^* be a critical point of f; let us look whether it may happen that this point is a limiting point of the trajectory $\{x_t\}$ of the Gradient Descent.

In fact it is better to speak about *continuous time* approximation of the method – the one where the trajectory $x(t)$ is function of *continuous* argument t given by the differential equation

$$\frac{d}{dt}x(t) = -\nabla f(x(t)) \tag{3.2.2}$$

The actual trajectory $\{x_t\}$ is a finite-difference approximation of the solution to this differential equation; the behaviour of the actual trajectory is more or less close to the one of $x(t)$. Of course, "more or less" is not a rigorous notion, but now we do not pretend to get rigorous results (although some of our conclusions can be proved rigorously).

Case A: $\nabla^2 f(x^*)$ is positive definite, so that x^* is a local minimizer of f (as it is said by the Sufficient condition for local optimality). It is easily seen that x^* can be a limiting point of the trajectory $x(t)$; the trajectory even converges to x^*, provided that it once came close enough to x^*. To verify this, it suffices to look at the behaviour of the function $V(x) = |x - x^*|^2$ along the trajectory: one has

$$\frac{d}{dt}V(x(t)) = 2(x(t) - x^*)^T \frac{d}{dt}x(t) = -(x(t) - x^*)^T \nabla f(x(t)).$$

[1] It means that the indicated properties can be ensured by properly chosen arbitrarily small perturbation of the original f, e.g., by adding to it a small linear form

When $x(t)$ is close to x^*, we have

$$\nabla f(x(t)) = \nabla f(x(t)) - \nabla f(x^*) \approx \nabla^2 f(x^*)(x(t) - x^*);$$

it is a simple exercise in Calculus to verify that the above $\approx$ is sufficient to make correct conclusion on the sign of $(x(t) - x^*)^T \nabla f(x(t))$: it will be the same (provided, of course, that $x(t)$ is close enough to x^*) as the sign of the product $(x(t) - x^*)^T \nabla^2 f(x^*)(x(t) - x^*)$, i.e., will be $+$ (recall that we are in the situation of positive definite $\nabla^2 f(x^*)$). Consequently, if, for certain $\bar{t}$, $V(x(\bar{t}))$ is small enough and nonzero (i.e., $x(\bar{t})$ is close enough to x^* and differs from x^*), then, for the same $\bar{t}$, we have $\frac{d}{dt} V(x(\bar{t})) < 0$; it immediately follows that, starting with $t = \bar{t}$, the function $V(x(t))$ will monotonically decrease and, moreover, will converge to 0. Thus, the trajectory $x(t)$, once coming close to x^*, will then approach x^*. In other words, a (strict[2]) local minimizer of f is an attractor for the vector field given by the right hand side of (3.2.2).

Case B: $\nabla^2 f(x^*)$ is negative definite, so that x^* is a (strict) local maximizer of f. Such a point never will be a limiting point of the trajectory, provided that the trajectory does not start at the point (since x^* is a critical point, the trajectory, started at x^*, will never leave the point). Indeed, in the case in question the vector field $-\nabla f(x)$ in a small ball neighbourhood U of x^* looks "outwards" of x^*, i.e., in this neighbourhood, in contrast to what we had in the case A,

$$x \neq x^* \Rightarrow (x - x^*)^T (-\nabla f(x)) > 0.$$

It follows that the trajectory never will enter U from outside of U, and if it starts somewhere in U (not at x^*!), it goes away from x^* until it leaves U.

Case C: $\nabla^2 f(x^*)$ is indefinite, so that x^* is a saddle point of f. To simplify things, let us assume that f is quadratic (so that (3.2.2) is linear differential equation, and we can write down the solution analytically). The result here is as follows: there exists an affine set $\mathcal{S}$ of dimension $< n$ such that the trajectory, started at this set, converges to x^*; and started outside this set, it never comes too close to x^*. Since it is "practically impossible" to start the trajectory *exactly* at "thin" set $\mathcal{S}$, we can conlcude that, at least in the case of a quadratic f, the trajectory never will converge to a saddle point of f. Now, since in the non-quadratic case locally everything is very similar to the quadratic one, we may say that it is "extremely unlikely" to get a saddle point of f as a limiting point of the Gradient Descent trajectory.

The results of our unformal considertaions are as follows: we may be "practically sure" that every limiting point of the Gradient Descent trajectory is a local minimizer of the objective, not a local maximizer or a saddle point of f. At the same time, there are no guarantees that the trajectory converges to a global minimizer of f, since *every* strict local minimizer of f is an attractor of the anti-gradient vector field $-\nabla f(x)$. Whether the trajectory converges to a local minimizer or to a global one, depends on the choice of the starting point.

3.3 Rates of convergence

3.3.1 Rate of global convergence: general $C^{1,1}$ case

As we already know, under the assumption of item (ii) of Theorem 3.2.1 (i.e., when the level set $S = \{x \mid f(x) \leq f(x_0)\}$ is bounded), the versions of the Gradien Descent mentioned in

[2] i.e., with positive definite $\nabla^2 f$

the Theorem converge to the set X^{**} of critical points of f. What can be said about the *non-asymptotical* rate of convergence? The answer depends on how we measure the inaccuracy. If we use to this purpose something like the distance

$$\mathrm{dist}(x, X^{**}) = \min_{y \in X^{**}} |y - x|$$

from an approximate solution x to X^{**}, no nontrivial efficiency estimates can be done: the convergence of the quantities $\mathrm{dist}(x_t, X^{**})$ to 0 can be arbitrarily slow. There is, anyhow, another accuracy measure,

$$\epsilon_f(x) = |\nabla f(x)|^2,$$

more suitable for our purposes. Note that the set X^{**} towards which the trajectory converges is exactly the set where $\epsilon_f(\cdot) = 0$, so that $\epsilon_f(x)$ indeed can be viewed as something which measures the "residual in the inclusion $x \in X^{**}$". And it turns out that we point out the rate at which this residual converges to 0:

Proposition 3.3.1 [Non-asymptotical rate of convergence of Gradient Descent]
Assume that the objective f is a $\mathrm{C}^{1,1}$ function, i.e., it is continuously differentiable with Lipschitz continuous gradient:

$$|\nabla f(x) - \nabla f(y)| \leq L_f |x - y|, \quad \forall x, y \in \mathbf{R}^n. \tag{3.3.1}$$

Then for any integer $N > 0$:

(i) *For the started at x_0 trajectory $\{x_t\}$ of* StD *one has*

$$\epsilon_f[t] \equiv \min_{0 \leq t < N} |\nabla f(x_t)|^2 \leq \frac{2L_f}{N}[f(x_0) - \min f]. \tag{3.3.2}$$

(ii) *For the started at x_0 trajectory $\{x_t\}$ of* ArD *one has*

$$\epsilon_f[t] \equiv \min_{0 \leq t < N} |\nabla f(x_t)|^2 \leq \frac{\eta L_f}{2\epsilon(1-\epsilon)N}[f(x_0) - \min f], \tag{3.3.3}$$

$\epsilon \in (0, 1)$, $\eta > 1$ being the parameters of the underlying Armijo test.

Proof [non-obligatory].
1^0. Let us start with the following simple

Lemma 3.3.1 *Under assumption of the Theorem one has*

$$f(y) \leq f(x) + (y - x)^T \nabla f(x) + \frac{L_f}{2}|y - x|^2, \quad \forall x, y \in \mathbf{R}^n. \tag{3.3.4}$$

Proof of Lemma. Let $\phi(\gamma) = f(x + \gamma(y - x))$. Note that ϕ is continuously differentiable (since f is) and

$$|\phi'(\alpha) - \phi'(\beta)| = |(y - x)^T(\nabla f(x + \alpha(y - x)) - \nabla f(x + \beta(y - x))| \leq$$

[Cauchy's inequality]

$$\leq |y - x||\nabla f(x + \alpha(y - x)) - \nabla f(x + \beta(y - x))| \leq$$

[(3.3.1)]

$$\leq |y - x|^2 L_f |\alpha - \beta|.$$

Thus,

$$|\phi'(\alpha) - \phi'(\beta)| \leq L_f |y-x|^2 |\alpha - \beta|, \quad \forall \alpha, \beta \in \mathbf{R}. \tag{3.3.5}$$

We have

$$f(y) - f(x) - (y-x)^T \nabla f(x) = \phi(1) - \phi(0) - \phi'(0) = \int_0^1 \phi'(\alpha) d\alpha - \phi'(0) =$$

$$= \int_0^1 [\phi'(\alpha) - \phi'(0)] d\alpha \leq$$

[see (3.3.5)]

$$\leq \int_0^1 |y-x|^2 L_f \alpha d\alpha = \frac{L_f}{2} |y-x|^2,$$

as required in (3.3.4). ∎

2^0. Now we are ready to prove (i). By construction of the Steepest Descent,

$$f(x_t) = \min_{\gamma \geq 0} f(x_{t-1} - \gamma \nabla f(x_{t-1})) \leq$$

[by Lemma 3.3.1]

$$\leq \min_{\gamma \geq 0} \left[f(x_{t-1}) + [-\gamma \nabla f(x_{t-1})]^T \nabla f(x_{t-1}) + \frac{L_f}{2} |\gamma \nabla f(x_{t-1})|^2 \right] =$$

$$= f(x_{t-1}) + |\nabla f(x_{t-1})|^2 \min_{\gamma \geq 0} \left[-\gamma + \frac{L_f}{2} \gamma^2 \right] = f(x_{t-1}) - \frac{1}{2L_f} |\nabla f(x_{t-1})|^2.$$

Thus, we come to the following important inequality:

$$f(x_{t-1}) - f(x_t) \geq \frac{1}{2L_f} |\nabla f(x_{t-1})|^2 \tag{3.3.6}$$

– the progress in the objective at a step of Steepest Descent is at least of order of the squared norm of the gradient at the previous iterate.

To conclude the proof, it suffices to note that, due to the monotonicity of the method, the total progress in the objective in course of certain segment of steps cannot be more than the initial residual $f(x_0) - \min f$ in the objective value; consequently, in a long segment, there must be a step with small progress, i.e., with small norm of the gradient. To make this reasoning quantitive, let us take sum of inequalities (3.3.6) over $t = 1, ..., N$, coming to

$$\frac{1}{2L_f} \sum_{t=0}^{N-1} |\nabla f(x_t)|^2 \leq f(x_0) - f(x_N) \leq f(x_0) - \min f.$$

The left hand side here is $\geq \frac{N}{2L_f} \min_{0 \leq t < N} |\nabla f(x_t)|^2$, and (3.3.2) follows. ∎

3^0. The proof of (ii) is a little bit more involved, but follows the same basic idea: the progress at a step of ArD can be small only if the gradient at the previous iterate is small, and the progress at certain step from a long segment of the steps must be small, since the total progress cannot be larger than the initial residual. Thus, in a long segment of steps we must pass through a point with small norm of the gradient.

The quantitive reasoning is as follows. First of all, progress in the objective at a step t of ArD is not too small, provided that both γ_t and $|\nabla f(x_{t-1})|^2$ are not too small:

$$f(x_{t-1}) - f(x_t) \geq \epsilon \gamma_t |\nabla f(x_{t-1})|^2; \tag{3.3.7}$$

this is immediate consequence of the first inequality in (3.1.2). Second, γ_t is not too small. Indeed, by Lemma 3.3.1 applied with $x = x_{t-1}, y = x_{t-1} - \eta\gamma_t \nabla f(x_{t-1})$ we have

$$f(x_{t-1} - \eta\gamma_t \nabla f(x_{t-1})) \leq f(x_{t-1}) - \eta\gamma_t |\nabla f(x_{t-1})|^2 + \frac{L_f}{2}\eta^2\gamma_t^2 |\nabla f(x_{t-1})|^2,$$

while by the second inequality in (3.1.2)

$$f(x_{t-1} - \eta\gamma_t \nabla f(x_{t-1})) \geq f(x_{t-1}) - \epsilon\eta\gamma_t |\nabla f(x_{t-1})|^2.$$

Combining these inequalities, we get

$$(1-\epsilon)\eta\gamma_t |\nabla f(x_{t-1})|^2 \leq \frac{L_f}{2}\eta^2\gamma_t^2 |\nabla f(x_{t-1})|^2.$$

Since $\gamma_t > 0$, in the case of $\nabla f(x_{t-1}) \neq 0$ we obtain

$$\gamma_t \geq \frac{2(1-\epsilon)}{\eta L_f}; \tag{3.3.8}$$

in the case of $\nabla f(x_{t-1}) = 0$ γ_t, as we remember, can be chosen in arbitrary way without influencing the trajectory (the latter in any case will satisfy $x_{t-1} = x_t = x_{t+1} = ...$), and we may assume that γ_t *always* satisfies (3.3.8).

Combining (3.3.7) and (3.3.8), we come to the following inequality (compare with (3.3.6):

$$f(x_{t-1}) - f(x_t) \geq \frac{2\epsilon(1-\epsilon)}{\eta L_f} |\nabla f(x_{t-1})|^2. \tag{3.3.9}$$

Now the proof can be completed exactly as in the case of the Steepest Descent. ∎

Remark 3.3.1 The efficiency estimate of Proposition 3.3.1 gives sublinearly converging to 0 *non-asymptotical* upper bound on the inaccuracies $\epsilon_f(\cdot)$ of the iterates. Note, anyhow, that this is a bound on the inaccuracy of *the best* (with the smallest norm of the gradient) of the iterates generated in course of the first N steps of the method, not on the inaccuracy of the *last* iterate x_N (the quantities $|\nabla f(x_t)|^2$ may oscillate, in contrast to the values $f(x_t)$ of the objective).

3.3.2 Rate of global convergence: convex $C^{1,1}$ case

Theorem 3.2.1 says that under mild assumptions the trajectories of ArD and StD converge to the set X^{**} of critical points of f. If we assume, in addition, that f is *convex*, so that the set of critical points of f is the same as the set of global minimizers of the function, we may claim that the trajectories converge to the optimal set of the problem. Moreover, in the case of convex $C^{1,1}$ objective (see Proposition 3.3.1) we can get non-asymptotical efficiency estimates in terms of the residuals $f(x_t) - \min f$, and under additional nondegeneracy assumption (see below) – also in terms of the distances $|x_t - x^*|$ from the iterates to the optimal solution.

To simplify our considerations and to make them more "practical", in what follows we restrict ourselves with the Armijo-based version ArD of the Gradient Descent.

Convex $C^{1,1}$ case

Proposition 3.3.2 [Rate of global convergence of ArD in convex $C^{1,1}$ case]
Let the parameter ϵ in the ArD *method be ≥ 0.5, and let f be a convex $C^{1,1}$ function with a nonempty set X^* of global minimizers. Then*

(i) *The trajectory $\{x_t\}$ of* ArD *converges to certain point $x^* \in X^*$;*

(ii) *For every $N \geq 1$ one has*

$$f(x_N) - \min f \leq \frac{\eta L_f \operatorname{dist}^2(x_0, x^*)}{4(1-\epsilon)N}, \tag{3.3.10}$$

where L_f is the Lipschitz constant of $\nabla f(\cdot)$ and

$$\operatorname{dist}(x, X^*) = \min_{y \in X^*} |y - x|. \tag{3.3.11}$$

Proof [non-obligatory].
1^0. Let x^* be a point from X^*, and let us look how the squared distances

$$d_t^2 = |x_t - x^*|^2$$

vary with t. We have

$$d_t^2 = |x_t - x^*|^2 \equiv |[x_{t-1} - \gamma_t \nabla f(x_{t-1})] - x^*|^2 = |[x_{t-1} - x^*] - \gamma_t \nabla f(x_{t-1})|^2 =$$

$$= |x_{t-1} - x^*|^2 - 2\gamma_t (x_{t-1} - x^*)^T \nabla f(x_{t-1}) + \gamma_t^2 |\nabla f(x_{t-1})|^2. \tag{3.3.12}$$

Since f is convex, from the Gradient Inequality

$$f(y) \geq f(x) + (y - x)^T \nabla f(x) \ \ \forall x, y \in \mathbf{R}^n$$

it follows that

$$(x_{t-1} - x^*)^T \nabla f(x_{t-1}) \geq f(x_{t-1}) - f(x^*) = f(x_{t-1}) - \min f.$$

This inequality combined with (3.3.12) results in

$$d_t^2 \leq d_{t-1}^2 - \gamma_t \left[2\epsilon_{t-1} - \gamma_t |\nabla f(x_{t-1})|^2 \right], \ \ \epsilon_s \equiv f(x_s) - \min f \geq 0. \tag{3.3.13}$$

According to (3.3.7), we have

$$\gamma_t |\nabla f(x_{t-1})|^2 \leq \frac{1}{\epsilon}[f(x_{t-1}) - f(x_t)] = \frac{1}{\epsilon}[\epsilon_{t-1} - \epsilon_t].$$

Combining this inequality with (3.3.13), we get

$$d_t^2 \leq d_{t-1}^2 - \gamma_t \left[(2 - \epsilon^{-1})\epsilon_{t-1} + \epsilon^{-1}\epsilon_t \right]. \tag{3.3.14}$$

Since, by assumption, $1/2 \leq \epsilon$, and clearly $\epsilon_s \geq 0$, the quantity in the parentheses in the right hand side is nonnegative. We know also from (3.3.8) that

$$\gamma_t \geq \bar{\gamma} = \frac{2(1-\epsilon)}{\eta L_f},$$

so that (3.3.14) results in

$$d_t^2 \le d_{t-1}^2 - \bar{\gamma}\left[(2-\epsilon^{-1})\epsilon_{t-1} + \epsilon^{-1}\epsilon_t\right]. \tag{3.3.15}$$

We conclude, consequently, that

(*) *The distances from the points x_t to (any) point $x^* \in X^*$ do not increase with t. In particualr, the trajectory is bounded.*

From (*) it immediately follows that $\{x_t\}$ converges to certain point $\bar{x}^* \in X^*$, as claimed in (i). Indeed, by Theorem 3.2.1 the trajectory, being bounded, has all its limiting points in the set X^{**} of critical points of f, or, which is the same (f is convex!), in the set X^* of global minimizers of f. Let $\bar{x}^*$ be one of these limiting points, and let us prove that in fact $\{x_t\}$ converges to $\bar{x}^*$. To this end note that the sequence $|x_t - \bar{x}^*|$, which, as we know from (*), is non-increasing, has 0 as its limiting point; consequently, the sequence converges to 0, so that $x_t \to \bar{x}^*$ as $t \to \infty$, as claimed.

It remains to prove (3.3.10). Taking sum of inequalities (3.3.15) over $t = 1, ..., N$, we get

$$N\bar{\gamma}\left[(2-\epsilon^{-1})\epsilon_{t-1} + \epsilon^{-1}\epsilon_t\right] \le d_0^2 - d_N^2 \le d_0^2 \equiv |x_0 - x^*|^2.$$

Since $\epsilon_0 \ge \epsilon_1 \ge \epsilon_2 \ge ...$ (our method is descent – it never decreases the values of the objective!), the left hand side in the resulting inequality can only become smaller if we replace all ϵ_t with ϵ_N; thus, we get

$$2N\bar{\gamma}\epsilon_N \le |x_0 - x^*|^2, \tag{3.3.16}$$

whence, substituting the expression for $\bar{\gamma}$,

$$\epsilon_N \le \frac{\eta L_f |x_0 - x^*|^2}{4(1-\epsilon)N};$$

since the resulting inequality is valid for all $x^* \in X^*$, (3.3.10) follows. ∎

Strongly convex $\mathbf{C}^{1,1}$ case

Proposition 3.3.2 deals with the case of *smooth convex* f, but there were no assumptions on the non-degeneracy of the minimizer – the minimizer might be non-unique, and the graph of f could be very "flat" around X^*. Under additional assumption of *strong convexity* of f we may get better convergence results.

The notion of strong convexity is given by the following

Definition 3.3.1 [Strongly convex function] *A function $f : \mathbf{R}^n \to \mathbf{R}$ is called strongly convex with the parameters of strong convexity (l_f, L_f), $0 < l_f \le L_f \le \infty$, if f is continuously differentiable and satisfies the inequalities*

$$f(x) + (y-x)^T\nabla f(x) + \frac{l_f}{2}|y-x|^2 \le f(y) \le f(x) + (y-x)^T\nabla f(x) + \frac{L_f}{2}|y-x|^2, \ \ \forall x, y \in \mathbf{R}^n. \tag{3.3.17}$$

Strongly convex functions traditionally play the role of "excellent" objectives, and this is the family on which the theoretical convergence analysis of optimization methods is normally performed. For our further purposes it is worthy to know how to detect strong convexity and what are the basic properties of strongly convex functions; this is the issue we are coming to.

The most convenient sufficient condition for strong convexity is given by the following

Proposition 3.3.3 [Criterion of strong convexity for twice continuously differentiable functions]
Let $f : \mathbf{R}^n \to \mathbf{R}$ be twice continuously differentiable function, and let (l_f, L_f), $0 < l_f \leq L_f < \infty$, be two given reals. f is strongly convex with parameters l_f, L_f if and only if the spectrum of the Hessian of f at every point $x \in \mathbf{R}^n$ belongs to the segment $[l_f, L_f]$:

$$l_f \leq \lambda_{\min}(\nabla^2 f(x)) \leq \lambda_{\max}(\nabla^2 f(x)) \leq L_f \ \ \forall x \in \mathbf{R}^n, \tag{3.3.18}$$

where $\lambda_{\min}(A), \lambda_{\max}(A)$ denote the minimal, respectively, the maximal eigenvalue of a symmetric matrix A and $\nabla^2 f(x)$ denotes the Hessian (the matrix of the second order derivatives) of f at x.

Example 3.3.1 *The convex quadratic form*

$$f(x) = \frac{1}{2} x^T A x - b^T x + c,$$

A being positive definite symmetric matrix, is strongly convex with the parameters $l_f = \lambda_{\min}(A)$, $L_f = \lambda_{\max}(A)$.

The most important for us properties of strongly convex functions are summarized in the following statement:

Proposition 3.3.4 *Let f be strongly convex with parameters (l_f, L_f). Then*
(i) *The level sets $\{x \mid f(x) \leq a\}$ of f are compact for every real a;*
(ii) *f attains its global minimum on $\mathbf{R}^n$, the minimizer being unique;*
(iii) *$\nabla f(x)$ is Lipschitz continuous with Lipschitz constant L_f.*

Now we come back to Gradient Descent. The following important proposition says that ArD , as applied to a strongly convex f, possesses global linear convergence:

Proposition 3.3.5 [Linear convergence of ArD as applied to strongly convex f]
Let a strongly convex, with parameters (l_f, L_f), function f be minimized by ArD *started at certain point x_0, and let the parameter ϵ of the Armijo test underlying the method be $\geq 1/2$. Then, for every integer $N \geq 1$, one has*

$$|x_N - x^*| \leq \theta^N |x_0 - x^*|, \quad \theta = \sqrt{\frac{Q_f - (2 - \epsilon^{-1})(1 - \epsilon)\eta^{-1}}{Q_f + (\epsilon^{-1} - 1)\eta^{-1}}}, \tag{3.3.19}$$

where x^* is the (unique, due to Proposition 3.3.4.(ii)) minimizer of f and

$$Q_f = \frac{L_f}{l_f} \tag{3.3.20}$$

is the condition number of f.

Besides this,

$$f(x_N) - \min f \leq \theta^{2N} Q_f [f(x_0) - \min f]. \tag{3.3.21}$$

Thus, the method globally linearly converges with convergence ratio θ (note that $\theta \in (0, 1)$ due to $\epsilon \in [1/2, 1)$).

Proof [non-obligatory].
1^0. According to Proposition 3.3.4, f is $C^{1,1}$ convex function which attains its minimum, and the gradient of f is Lipschitz continuous with constant L_f. Consequently, all conclusions of the proof of Proposition 3.3.2 are valid, in particular, relation (3.3.14):

$$d_t^2 \equiv |x_t - x^*|^2 \le d_{t-1}^2 - \bar\gamma \left[(2-\epsilon^{-1})\epsilon_{t-1} + \epsilon^{-1}\epsilon_t\right], \ \bar\gamma = \frac{2(1-\epsilon)}{\eta L_f}, \ \epsilon_s = f(x_s) - \min f. \quad (3.3.22)$$

Applying (3.3.17) to the pair $(x = x^*, y = x_s)$ and taking into account that $\nabla f(x^*) = 0$, we get

$$\epsilon_s \ge \frac{l_f}{2}|x_s - x^*|^2 = \frac{l_f}{2} d_s^2;$$

therefore (3.3.22) implies

$$d_t^2 \le d_{t-1}^2 - \frac{\bar\gamma l_f}{2}\left[(2-\epsilon^{-1})d_{t-1}^2 + \epsilon^{-1} d_t^2\right],$$

or, substituting the expression for $\bar\gamma$ and rearranging the expression,

$$d_t^2 \le \theta^2 d_{t-1}^2, \quad (3.3.23)$$

with θ given by (3.3.19), and (3.3.19) follows.

It remains to prove (3.3.21). To this end it suffices to note that, due to the first inequality in (3.3.17) applied with $x = x^*, y = x_0$, one has

$$|x_0 - x^*|^2 \le \frac{2}{l_f}[f(x_0) - f(x^*)] = \frac{2}{l_f}[f(x_0) - \min f], \quad (3.3.24)$$

while the second inequality in (3.3.17) applied with $x = x^*, y = x_N$ says that

$$f(x_N) - \min f \equiv f(x_N) - f(x^*) \le \frac{L_f}{2}|x_N - x^*|^2;$$

consequently,

$$f(x_N) - \min f \le \frac{L_f}{2}|x_N - x^*|^2 \le$$

[see (3.3.19)]

$$\le \frac{L_f}{2}\theta^{2N}|x_0 - x^*|^2 \le$$

[see (3.3.24)]

$$\le \frac{L_f}{l_f}\theta^{2N}[f(x_0) - \min f],$$

as required in (3.3.21). ∎

Global rate of convergence in convex $C^{1,1}$ case: summary

The results given by Propositions 3.3.2 and 3.3.5 can be summarized as follows. Assume that we are solving the problem

$$f(x) \to \min$$

with convex $C^{1,1}$ objective (i.e., $\nabla f(x)$ is a Lipschitz continuous vector field), and assume that f possesses a nonempty set X^* of global minimizers. And assume that we are minimizing f by ArD with properly chosen parameter ϵ, namely, $1/2 \le \epsilon < 1$. Then

- A. In the general case, where no strong convexity of f is imposed, the trajectory $\{x_t\}$ of the method converges to certain point $\bar{x}^* \in X^*$, and the residuals in terms of the objective – the quantities $\epsilon_N = f(x_N) - \min f$ – go to zero at least as $O(1/N)$, namely, they satisfy the estimate

$$\epsilon_N \leq \frac{\eta L_f \operatorname{dist}^2(x_0, X^*)}{4(1-\epsilon)} \frac{1}{N}. \tag{3.3.25}$$

  Note that

  - no quantitive assertions on the rate of convergence of the quantities $|x_N - \bar{x}^*|$ can be given; all we know is that these quantities converge to 0, but the convergence can be as slow as you wish. Namely, given an arbitrary decreasing sequence $\{d_t\}$ converging to 0, one can point out a $C^{1,1}$ convex function f on 2D plane such that $L_f = 1$, $\operatorname{dist}(x_0, X^*) = d_0$ and $\operatorname{dist}(x_t, X^*) \geq d_t$ for every t;
  - estimate (3.3.25) establishes correct order of convergence to 0 of the residuals in terms of the objective: for properly chosen $C^{1,1}$ convex function f on the 2D plane one has

$$\epsilon_N \geq \frac{\alpha}{N}, \ N = 1, 2, ...$$

    with certain positive α.

- B. If f is strongly convex with parameters (l_f, L_f), then the method converges linearly:

$$|x_N - x^*| \leq \theta^N |x_0 - x^*|, \ \ f(x_N) - \min f \leq Q_f \theta^{2N} [f(x_0) - \min f],$$

$$\theta = \sqrt{\frac{Q_f - (2 - \epsilon^{-1})(1-\epsilon)\eta^{-1}}{Q_f + (\epsilon^{-1} - 1)\eta^{-1}}}, \tag{3.3.26}$$

  $Q_f = L_f / l_f$ being the condition number of f.

Note that the convergence ratio θ (or θ^2, depending on which accuracy measure – the distance from the iterate to the optimal set or the residual in terms of the objective – we use) tends to 1 as the condition number of the problem goes to infinity (as people say, as the problem becomes *ill-conditioned*). When Q_f is large, we have

$$\theta \approx 1 - p Q_f^{-1}, \ \ p = (1-\epsilon)\eta^{-1}, \tag{3.3.27}$$

so that to decrease the upper bound (3.3.26) on $|x. - x^*|$ by an absolute constant factor, say, by factor 10, it requires $O(Q_f)$ steps of the method. In other words, what we can extract from (3.3.26) is that

(**) *the number of steps of the method resulting in a given in advance progress in accuracy (the one required to decrease the initial distance from the optimal set by a given factor, say, 10^6), is proportional to the condition number Q_f of the objective.*

Of course, this conclusion is derived from an *upper* bound on inaccuracy; it might happen that our upper bounds "underestimate" actual performance of the method. It turns out, anyhow, that our bounds are tight, and the conclusion is valid:

in the case of strongly convex objective f (which is the best case in unconstrained optimization)

the number of steps of the Gradient Descent required to reduce initial inaccuracy (measured either as the distance from the optimal set or as the residual in terms of the objective) *by a given factor is typically proportional to the condition number of* f.

To justify the claim, let us look what happens in the case of *quadratic* objective.

3.3.3 Rate of convergence in the quadratic case

Let us look what happens if Gradient Descent is applied to a strongly convex quadratic objective

$$f(x) = \frac{1}{2}x^T Ax - b^T x + c.$$

A being symmetric positive definite matrix. As we know from Example 3.3.1, f is strongly convex with the parameters $l_f = \lambda_{\min}(A)$, $L_f = \lambda_{\max}(A)$ (the minimal and the maximal eigenvalues of A, respectively).

It is convenient to speak about the Steepest Descent rather than about the Armijo-based Gradient Descent (in the latter case our considerations would suffer from uncertainty in the choice of the stepsizes).

We have the following relations:

- The gradient of the function f is given by the relation

$$g(x) \equiv \nabla f(x) = Ax - b; \tag{3.3.28}$$

in particular, the unique minimizer of f is given by the equation (the Fermat rule)

$$Ax^* = b. \tag{3.3.29}$$

Note also that, as it is seen from one-line computation,

$$f(x) = E(x) + f(x^*), \;\; E(x) = \frac{1}{2}(x - x^*)^T A(x - x^*); \tag{3.3.30}$$

note that $E(\cdot)$ is nothing but inaccuracy in terms of the objective.

- The trajectory of the Steepest Descent is given by the recurrency

$$x_{t+1} = x_t - \gamma_{t+1} g_t, \;\; g_t \equiv g(x_t) \equiv \nabla f(x_t) = Ax_t - b = A(x_t - x^*), \tag{3.3.31}$$

where γ_{t+1} is minimizer of the strongly convex quadratic function $\phi(\gamma) = f(x_t - \gamma g_t)$ of real variable γ. Solving equation $\phi'(\gamma) = 0$ which identifies γ_{t+1}, we get

$$\gamma_{t+1} = \frac{g_t^T g_t}{g_t^T A g_t}; \tag{3.3.32}$$

thus, (3.3.31) becomes

$$x_{t+1} = x_t - \frac{g_t^T g_t}{g_t^T A g_t} g_t. \tag{3.3.33}$$

- Explicit computation results in [3)]

$$E(x_{t+1}) = \left\{1 - \frac{(g_t^T g_t)^2}{[g_t^T A g_t][g_t^T A^{-1} g_t]}\right\} E(x_t). \tag{3.3.34}$$

Now we can obtain the convergence rate of the method from the following

Lemma 3.3.2 [Kantorovich] *Let A be a positive definite symmetric matrix with the condition number (the ratio between the largest and the smallest eigenvalue) Q. Then for any nonzero vector x one has*

$$\frac{(x^T x)^2}{[x^T Ax][x^T A^{-1} x]} \geq \frac{4Q}{(1+Q)^2}.$$

Proof. It is known from Linear Algebra that a symmetric $n \times n$ matrix A is orthogonally equivalent to a diagonal matrix S (i.e., $A = USU^T$ with orthogonal U), the eigenvalues $\lambda_1 \leq \lambda_2 \leq ... \leq \lambda_n$ of A being the diagonal entries of S. Denoting $y = U^T x$, we see that the left hand side in the inequality in question is

$$\frac{(\sum_i y_i^2)^2}{(\sum_i \lambda_i y_i^2)(\sum_i \lambda_i^{-1} y_i^2)}. \tag{3.3.35}$$

This quantity remains unchanged if all y_i's are multiplied by common nonzero factor; thus, without loss of generality we may assume that $\sum_i y_i^2 = 1$. Further, the quantity in question remains unchanged if all λ_i's are multiplied by common positive factor; thus, we may assume that $\lambda_1 = 1$, so that $\lambda_n = Q$ is the condition number of the matrix A. Setting $a_i = y_i^2$, we come to the necessity to prove that

if $u = \sum_i a_i \lambda_i$, $v = \sum_i a_i \lambda_i^{-1}$, where $0 \leq a_i$, $\sum_i a_i = 1$, and $1 \leq \lambda_i \leq Q$, then $uv \leq (1+Q)^2/(4Q)$.

This is easy: due to its origin, the point (u, v) on the 2D plane is convex combination, the coefficients being a_i, of the points $P_i = (\lambda_i, \lambda_i^{-1})$ belonging to the arc Γ on the graph of the function $\eta = 1/\xi$, the arc corresponding to the segment $[1, Q]$ of values of ξ (ξ, η are the

[3] Here is the computation: since $\phi(\gamma)$ is a convex quadratic form and γ_{t+1} is its minimizer, we have

$$\phi(0) = \phi(\gamma_{t+1}) + \frac{1}{2}\gamma_{t+1}^2 \phi'';$$

due to the origin of ϕ, we have $\phi'' = g_t^T A g_t$, so that

$$E(x_t) - E(x_{t+1}) \equiv f(x_t) - f(x_{t+1}) \equiv \phi(0) - \phi(\gamma_{t+1}) = \frac{1}{2}\gamma_{t+1}^2 [g_t^T A g_t],$$

or, due to (3.3.32),

$$E(x_t) - E(x_{t+1}) = \frac{(g_t^T g_t)^2}{2 g_t^T A g_t}.$$

At the same time by (3.3.30), (3.3.31) one has

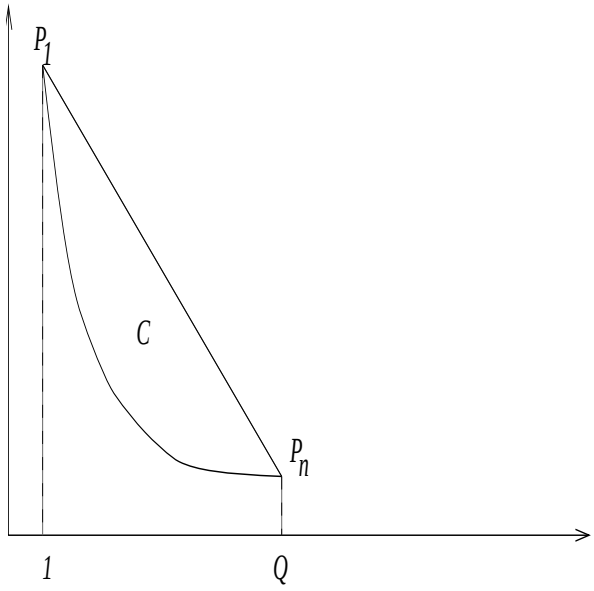
$$E(x_t) = \frac{1}{2}(x_t - x^*)^T A (x_t - x^*) = \frac{1}{2}[A^{-1} g_t]^T A [A^{-1} g_t] = \frac{1}{2} g_t^T A^{-1} g_t.$$

Combining the resulting relations, we come to

$$\frac{E(x_t) - E(x_{t+1})}{E(x_t)} = \frac{(g_t^T g)^2}{[g_t^T A g_t][g_t^T A^{-1} g_t]},$$

as required in (3.3.34).

coordinates on the plane). Consequently, (u, v) belongs to the convex hull C of Γ. This convex hull is as you see on the picture:



Arc Γ and its convex hull

The largest, over $(u, v) \in C$, product uv corresponds to the case when (u, v) belongs to the line segment $[P_1, P_n]$ bounding C from above, so that

$$uv \leq \max_{0 \leq a \leq 1} [(a + (1-a)Q)(a + \frac{1-a}{Q})];$$

the right hand side maximum can be explicitly computed (it corresponds to $a = 1/2$), and the resulting value is $(Q+1)^2/(4Q)$, as claimed. ∎

Combining Lemma 3.3.2 and (3.3.34), we come to the following

Proposition 3.3.6 [Convergence ratio for the Steepest Descent as applied to strongly convex quadratic form]
As applied to a strongly convex quadratic form f with condition number Q, the Steepest Descent converges linearly with the convergence ratio not worse than

$$1 - \frac{4Q}{(Q+1)^2} = \left(\frac{Q-1}{Q+1}\right)^2, \tag{3.3.36}$$

namely, for all N one has

$$f(x_N) - \min f \leq \left(\frac{Q-1}{Q+1}\right)^{2N} [f(x_0) - \min f]. \tag{3.3.37}$$

Note that the Proposition says that the convergence ratio is *not worse* than $(Q-1)^2(Q+1)^{-2}$; the actual convergence ratio depends on the starting point x_0. It is known, anyhow, that (3.3.37) gives correct description of the rate of convergence: for "almost all" starting points, the process indeed converges at the rate close to the indicated upper bound. Since the convergence ratio given by Proposition is $1 - O(1/Q)$ (cf. (3.3.27)), quantitive conclusion (**) from the previous subsection indeed is valid, even in the case of strongly convex *quadratic* f.

3.3.4 Local rate of convergence of Steepest Descent

Relation (3.3.37) is a *non-asymptotical* efficiency estimate of the Steepest Descent in the *quadratic* case. In the *non-quadratic nondegenerate* case the method admits similar *asymptotic* efficiency estimate. Namely, one can prove the following

Theorem 3.3.1 [Local rate of convergence of Steepest Descent]
Assume that the trajectory $\{x_t\}$ of the Steepest Descent as applied to f converges to a point x^* which is a nondegenerate local minimizer of f, namely, is such that f is twice continuously differentiable in a neighbourhood of x^* and the Hessian $\nabla^2 f(x^*)$ of the objective at x^* is positive definite.

Then the trajectory converges to x^* linearly, and the convergence ratio of the sequence $f(x_t) - f(x^*)$ of residuals in terms of the objective is at least

$$\left(\frac{Q-1}{Q+1}\right)^2,$$

Q being the condition number of $\nabla^2 f(x^*)$:

$$(\forall \epsilon > 0\ \exists C_\epsilon < \infty): \quad f(x_N) - f(x^*) \leq C_\epsilon \left(\frac{Q-1}{Q+1} + \epsilon\right)^{2N},\ N = 1, 2, ... \tag{3.3.38}$$

3.4 Conclusions

Let us summarize our knowledge of Gradient Descent. We know that

- In the most general case, under mild regularity assumptions, both StD and ArD converge to the set of critical points of the objective (see Theorem 3.2.1), and there is certain guaranteed sublinear rate of global convergence in terms of the quantities $|\nabla f(x_N)|^2$ (see Proposition 3.3.1);
- In the convex $C^{1,1}$ case ArD converges to a global minimizer of the objective (provided that such a minimizer exists), and there is certain guaranteed (sublinear) rate of global convergence in terms of the residuals in the objective $f(x_N) - \min f$ (see Proposition 3.3.2);
- In the strongly convex case ArD converges to the unique minimizer of the objective, and both distances to the minimizer and the residuals in terms of the objective admit global linearly converging to zero upper bounds. The corresponding convergence ratio is given by the condition number of the objective Q (see Proposition 3.3.5) and is of the type $1 - O(1/Q)$, so that the number of steps required to reduce the initial inaccuracy by a given factor is proportional to Q (this is an upper bound, but typically it reflects the actual behaviour of the method);
- It the quadratic case - globally, and in the nonquadratic one – asymptotically, StD converges linearly with the convergence ratio $1 - O(1/Q)$, Q being the condition number of the Hessian of the objective at the minimizer towards which the method converges (in the quadratic case, of course, this Hessian simply is the matrix of our quadratic form).

This is what we know. What should be conclusions – is the method good, or bad, or what? As it normally is the case in numerical optimization, we are unable to give a definite answer: there are

too many different criteria to be taken into account. What we can do, is to list advantages and disadvantages of the method. Such a knowledge provides us with a kind of orientation: when we know what are the strong and the weak points of an optimization method and given a particular application we are interested in, we can decide "how strong in the case in question are the strong points and how weak are the weak ones", thus getting possibility to choose the solution method better fitting the situation. As about the Gradient Descent, the evident strong points of the method are

- broad family of problems where we can guarantee global convergence to a critical point (normally - to a local minimizer) of the objective;
- simplicity: at a step of the method, we need single evaluation of ∇f and a small number of evaluations of f (the evaluations of f are required by the line search; if one uses ArD with simplified line search mentioned in Section 2.4.1, this number indeed is small). Note that each evaluation of f is accompanied by small ($O(n)$, n being the dimension of the design vector) number of arithmetic operations.

The most important weak point of the method is relatively low rate of convergence: even in the strongly convex quadratic case, the method converges linearly. This itself is not that bad; what indeed is bad, is that the convergence ratio is too sensitive to the condition number Q of the objective. As we remember, the number of steps of the method, for a given progress in accuracy, is proportional to Q. And this indeed is too bad, since in applications we typically meet with ill-conditioned problems, with condition numbers of orders of thousands and millions; whenever this is the case, we hardly can expect something good from Gradient Descent, at least when we are interested in high-accuracy solutions.

It is worthy to understand what is the geometry underlying slowing down the Gradient Descent in the case of ill-conditioned objective. Consider the case of strongly convex quadratic f. The level surfaces

$$S_\delta = \{x \mid f(x) = \min f + \delta\}$$

of f are homothetic ellipsoids centered at the minimizer x^* of f; the squared half-axes of these ellipsoids are inverse proportional to the eigenvalues of $A = \nabla^2 f$. Indeed, as we know from (3.3.30),

$$f(x) = \frac{1}{2}(x - x^*)^T A(x - x^*) + \min f,$$

so that in the orthogonal coordinates x_i associated with the orthonormal eigenbasis of A and the origin placed at x^* we have

$$f(x) = \frac{1}{2}\sum_i \lambda_i x_i^2 + \min f,$$

λ_i being the eigenvalues of A. Consequently, the equation of S_δ in the indicated coordinates is

$$\sum_i \lambda_i x_i^2 = 2\delta.$$

Now, if A is ill-conditioned, the ellipsoids S_δ become a kind of "valleys" – they are relatively narrow in some directions (those associated with smallest half-axes of the ellipsoids) and relatively long in other directions (associated with the largest half-axes). The gradient – which is orthogonal to the level surface – on the dominating part of this surface looks "almost across the

valley", and since the valley is narrow, the steps turn out to be short. As a result, the trajectory of the method is a kind of short-step zig-zag movement with slow overall trend towards the minimizer.

What should be stressed is that in the case in question there is nothing intrinsically bad in the problem itself; all difficulties come from the fact that we relate the objective to a "badly chosen" initial coordinates. Under appropriate *non-orthogonal* linear transformation of coordinates (pass from x_i to $y_i = \sqrt{\lambda_i} x_i$) the objective becomes perfectly conditioned – it becomes the sum of suqares of the coordinates, so that condition number now equals 1, and the Gradient Descent, *being run in the new coordinates,* will go directly towards the minimizer. The problem, of course, is that the Gradient Descent is *associated with a once for ever fixed initial Euclidean coordinates* (since the underlying notion of gradient is a Euclidean notion: different Euclidean structures result in different gradient vectors of the same function at the same point). If these initial coordinates are badly chosen for a given objective f (so that the condition number of f with respect to these coordinates is large), the Gradient Descent will be slow, although if we were clever enough to perform first appropriate *scaling* – linear non-orthogonal transformation of the coordinates – and then run Gradient Descent in these new coordinates, we might obtain fast convergence. In our next Lecture we will consider the famous *Newton method* which, in a sense, is nothing but "locally optimally scaled" Gradient Descent, with the scaling varying from step to step.

Assignment # 3 (Lecture 3)

Obligatory problems

Exercise 3.4.1 *Prove that in the Steepest Descent any two subsequent directions of movement are mutually orthogonal. Derive from this that in the 2D case all directions of movement on even steps are collinear to each other, and the directions of movement at the odd steps also are collinear to each other.*

Exercise 3.4.2 *Write the code implementing the* ArD *(or* StD *, on your choice) and apply it to the following problems:*

- *Rosenbrock problem*

$$f(x) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2 \to \min \mid x = (x_1, x_2) \in \mathbf{R}^2,$$

the starting point is $x_0 = (-1.2, 1)$.

The Rosenbrock problem is a well-known test example: it has a unique critical point $x^* = (1, 1)$ (the global minimizer of f); the level lines of the function are banana-shaped valleys, and the function is nonconvex and rather badly conditioned.

- *Quadratic problem*

$$f_\alpha(x) = x_1^2 + \alpha x_2^2 \to \min \mid x = (x_1, x_2) \in \mathbf{R}^2.$$

Test the following values of α

$$10^{-1}; 10^{-4}; 10^{-6}$$

and for each of these values test the starting points

$$(1, 1); (\sqrt{\alpha}, 1); (\alpha, 1).$$

How long it takes to reduce the initial inaccuracy, in terms of the objective, by factor 0.1?

- *Quadratic problem*

$$f(x) = \frac{1}{2} x^T A x - b^T x, \;\; x \in \mathbf{R}^4,$$

with

$$A = \begin{pmatrix} 0.78 & -0.02 & -0.12 & -0.14 \\ -0.02 & 0.86 & -0.04 & 0.06 \\ -0.12 & -0.04 & 0.72 & -0.08 \\ -0.14 & 0.06 & -0.08 & 0.74 \end{pmatrix}, \quad b = \begin{pmatrix} 0.76 \\ 0.08 \\ 1.12 \\ 0.68 \end{pmatrix}, \quad x_0 = 0.$$

Run the method intil the norm of the gradient at the current iterate is becomes less than 10^{-6}*. Is the convergence fast or not?*

Those using MatLab can compute the spectrum of A and to compare the theoretical upper bound on convergence rate with the observed one.

- *Experiments with Hilbert matrix. Let* $H^{(n)}$ *be the* $n \times n$ *Hilbert matrix:*

$$(H^{(n)})_{ij} = \frac{1}{i + j - 1}, \;\; i, j = 1, ..., n.$$

This is a symmetric positive definite matrix (since $x^T H^{(n)} x = \int_0^1 (\sum_{i=1}^n x_i t^{i-1})^2 dt \geq 0$*, the inequality being strict for* $x \neq 0$*).*

For $n = 4, 8, 16$ *perform the following experiments:*

- *choose somehow n-dimensional nonzero vector x^*, e.g., $x^* = (1, ..., 1)^T$;*
- *compute $b = H^{(n)}x^*$;*
- *Apply your Gradient Descent code to the quadratic function*

$$f(x) = \frac{1}{2}x^T H^{(n)}x - b^T x,$$

the starting point being $x_0 = 0$. Note that x^* is the unique minimizer of f.

- *Terminate the method when you will get $|x_N - x^*| \leq 10^{-4}$, not allowing it, anyhow, to run more than 10,000 steps.*

What will be the results?

Those using MatLab can try to compute the condition number of the Hilbert matrices in question.

If you choose to implement ArD *, play with the parameters ϵ and η of the method to get the best possible convergence.*

Optional problems

Exercise 3.4.3 *Prove that if x^* is a strict local minimizer of a twice continuously differentiable f, then* ArD *with the choice of the stepsize as the smallest one satisfying the Armijo test, being started close enough to x^*, converges to x^*.*

Exercise 3.4.4 *Prove that if x^* is a strict local maximizer of twice continuously differentiable f (i.e., $\nabla f(x^*) = 0$ and $\nabla^2 f(x^*)$ is negative definite), then the trajectories of* ArD *and* StD *do not have x^* as a limiting point, provided that x^* does not belong to the trajectory (the latter may happen in* ArD *because of "badly chosen" stepsize, and in both of the methods – when x^* is the starting point).*

Exercise 3.4.5 *Prove that if x^* is a strict local minimizer of a twice continuously differentiable f, then* ArD *with the choice of the stepsize as the smallest one satisfying the Armijo test, being started close enough to x^*, converges to x^*.*

Exercise 3.4.6 *Prove items (i) and (ii) of Proposition 3.3.4.*

Exercise 3.4.7 *Prove item (iii) of Proposition 3.3.4.*

Lecture 4

Newton's Method

We continue investigating methods for unconstrained minimization problem

$$f(x) \to \min \mid x \in \mathbf{R}^n.$$

What is on our agenda is the famous *Newton method* based on local quadratic model of f. To get possibility to speak about this model, we assume from now on that f is twice continuously differentiable.

4.1 The Basic Newton Method

The idea of the method is very simple; we have already used this idea in the univariate case (Lecture 2). Given current iterate x, the value $f(x)$, the gradient $\nabla f(x)$ and the Hessian $\nabla^2 f(x)$ of the objective at x, we approximate f around x by its second order Taylor expansion

$$f(y) \approx f(x) + (y-x)^T \nabla f(x) + \frac{1}{2}(y-x)^T [\nabla^2 f(x)](y-x)$$

and take as the next iterate the minimizer of the right hand side quadratic form of y. To get this minimizer, we differentiate the right hand side in y and set the gradient to 0, which results in the equation with respect to y

$$[\nabla^2 f(x)](y-x) = -\nabla f(x).$$

This is a linear system with respect to y; assuming the matrix of the system (i.e., the Hessian $\nabla^2 f(x)$) nonsingular, we can write down the solution as

$$y = x - [\nabla^2 f(x)]^{-1} \nabla f(x).$$

In the Basic Newton method, we simply iterate the indicated updating:

Algorithm 4.1.1 [Basic Newton Method] *Given starting point x_0, run the recurrence*

$$x_t = x_{t-1} - [\nabla^2 f(x_{t-1})]^{-1} \nabla f(x_{t-1}). \tag{4.1.1}$$

The indicated method is not necessarily well-defined (e.g., what to do when the Hessian at the current iterate turns out to be singular?) We shall address this difficulty, same as several other difficulties which may occur in the method, later. Our current goal is to establish the fundamental result on the method – its *local quadratic convergence in the non-degenerate case:*

Theorem 4.1.1 [Local Quadratic Convergence of the Newton method in the nondegenerate case]
Assume that f is three times continuously differentiable in a neighbourhood of $x^* \in \mathbf{R}^n$, and that x^* is a nondegenerate local minimizer of f, i.e., $\nabla f(x^*) = 0$ and $\nabla^2 f(x^*)$ is positive definite. Then the Basic Newton method, starting close enough to x^*, converges to x^* quadratically.

Proof. Let U be a convex neighbourhood of x^* where the third order partial derivatives of f (i.e., the second order partial derivatives of the components of ∇f) are bounded. In this neighbourhood, consequently,

$$| - \nabla f(y) - \nabla^2 f(y)(x^* - y)| \equiv |\nabla f(x^*) - \nabla f(y) - \nabla^2 f(y)(x^* - y)| \leq \beta_1 |y - x^*|^2 \qquad (4.1.2)$$

with some constant β_1 (we have applied to the components of ∇f the standard upper bound on the remainder in the first order Taylor expansion: if $g(\cdot)$ is a scalar function with bounded second order derivatives in U, then

$$|g(x) - g(y) - \nabla g(y)(x - y)| \leq \beta |y - x|^2$$

for some $\beta < \infty$ [1)] and all $x, y \in U$, cf. Lemma 3.3.1).

Since $\nabla^2 f(x^*)$ is nonsingular and $\nabla^2 f(x)$ is continuous at $x = x^*$, there exists a smaller neighbourhood $U' \subset U$ of x^*, let it be the centered at x^* ball of radius $r > 0$, such that

$$y \in U' \Rightarrow |[\nabla^2 f(y)]^{-1} \leq \beta_2 \qquad (4.1.3)$$

for some constant $beta_2$; here and in what follows, for a matrix A $|A|$ denotes the *operator norm* of A, i.e.,

$$|A| = \max_{|h| \leq 1} |Ah|,$$

the right hand side norms being the standard Euclidean norms on the corresponding vector spaces.

Now assume that certain point x_t of the trajectory of the Basic Newton method on f is close enough to x^*, namely, is such that

$$x_t \in U'', \quad U'' = \{x \mid |x - x^*| \leq \rho \equiv \min[\frac{1}{2\beta_1\beta_2}, r]\}. \qquad (4.1.4)$$

We have

$$|x_{t+1} - x^*| = |x_t - x^* - [\nabla^2 f(x_t)]^{-1} \nabla f(x_t)| =$$

$$= |[\nabla^2 f(x_t)]^{-1} \left[\nabla^2 f(x_t)(x_t - x^*) - \nabla f(x_t)\right]| \leq |[\nabla^2 f(x_t)]^{-1}| \, | - \nabla f(x_t) - \nabla^2 f(x_t)(x^* - x_t)| \leq$$

[by (4.1.3) and (4.1.2)]

$$\leq \beta_1\beta_2 |x_t - x^*|^2.$$

Thus, we come to

$$x_t \in U'' \Rightarrow |x_{t+1} - x^*| \leq \beta_1\beta_2 |x_t - x^*|^2 \quad [\leq (\beta_1\beta_2 |x_t - x^*|)|x_t - x^*| \leq 0.5 |x_t - x^*|]. \qquad (4.1.5)$$

We see that the new iterate x_{t+1} is at least twice closer to x^* than x_t and, consequently, $x_{t+1} \in U''$. Thus, reaching U'' at certain moment $\bar{t}$ (this for sure happens when the trajectory is started in U''), the trajectory never leaves this neighbourhood of x^*, and

$$|x_{t+1} - x^*| \leq \beta_1\beta_2 |x_t - x^*|^2 \leq 0.5 |x_t - x^*|, \; t \geq \bar{t},$$

[1] note that the magnitude of β is of order of the magnitude of second order derivatives of g in U

so that the trajectory converges to x^* quadratically. ∎

The indicated theorem establishes fast – quadratic – local convergence of the Basic Newton method to a nondegenerate local minimizer of f, which is fine. At the same time, we remember from Lecture 2 that even in the univariate case and for smooth convex objective, the Newton method not necessarily possesses global convergence: started not too close to minimizer, it may diverge. It follows that we cannot rely on this method "as it is" – in actual computations, how could we know that the starting point is "close enough" to the minimizer? Thus, some modifications are needed in order to make the method globally converging. Let us look at the standard modifications of this type.

4.2 Modifications of the Basic Newton Method

4.2.1 Incorporating line search

The Basic Newton method at each iteration performs a unit step:

$$x_{t+1} = x_t = e(x_t)$$

in the *Newton direction*

$$e(x) = -[\nabla^2 f(x)]^{-1}\nabla f(x). \tag{4.2.1}$$

It prescribes, consequently, both the search direction and the stepsize (just 1) in the direction. The first idea how to "cure" the method to make it globally convergent is to use only the direction given by the method, but not the stepsize; as about the stepsize, we could choose it by a kind of line search aimed to achieve "significant progress" in the objective (compare with the Gradient Descent). Thus, we come to the *Newton method with line search* given by the recurrence

$$x_{t+1} = x_t + \gamma_{t+1}e(x_t),\ e(x_t) = [\nabla^2 f(x_t)]^{-1}\nabla f(x_t), \tag{4.2.2}$$

where the stepsize $\gamma_{t+1} \geq 0$ is given by a line search. In particular, we could speak about

- *"Steepest Newton method"*:

  $$\gamma_{t+1} \in \mathrm{Argmin}\{f(x_t + \gamma e(x_t)) \mid \gamma \geq 0\},$$

  or

- the *Armijo-based Newton method*, where γ_{t+1} is given by the Armijo-terminated line search,

  or

- the *Goldstein-based Newton method*, where γ_{t+1} is given by the goldstein-terminated line search.

We could expect the indicated modifications to make the method globally converging; at the same time, we may hope that close enough to a nondegenerate local minimizer of f, the indicated line search will result in stepsize close to 1, so that the asymptotical behaviour of the modified method will be similar to the one of the basic method (provided, of course, that the modified method indeed converges to a nondegenerate local minimizer of f). Whether these hopes indeed

are valid or not, it depends on f, and at least one property of f seems to be necessary to make our hopes valid: $e(x)$ should be a *descent* direction of f at a non-critical x:

$$\nabla f(x) \neq 0 \Rightarrow e^T(x)\nabla f(x) \equiv -(\nabla f(x))^T[\nabla^2 f(x)]^{-1}\nabla f(x) < 0. \tag{4.2.3}$$

Indeed, if there exists x with nonzero $\nabla f(x)$ such that the Newton direction $e(x)$ is not a descent direction of f at x, it is unclear whether there exists a step in this direction which reduces f. If, e.g., $f(x + \gamma e(x))$ is a nondecreasing function of $\gamma > 0$, then the Steepest Newton method started at x clearly will never leave the point, thus converging to (simply staying at) non-critical point of f. Similar difficulties occur with the Armijo- and Goldstein-based Newton methods: is $e(x)$ is not a descent direction of f at x, then the Armijo/Goldstein-terminated line search makes no sense at all.

We see that one may hope to prove convergence of the Newton method with line search only if f satisfies the property

$$\nabla f(x) \neq 0 \Rightarrow (\nabla f(x))^T[\nabla^2 f(x)]^{-1}\nabla f(x) > 0.$$

The simplest way to impose this property is to assume that f is convex with nonsingular Hessian:

$$\nabla^2 f(x) > 0 \quad \forall x \quad \left[\equiv h^T[\nabla^2 f(x)]h > - \forall x \forall h \neq 0\right]; \tag{4.2.4}$$

indeed, if the matrix $\nabla f(x)$ is positive definite at every x, then, as it is known from Linear Algebra (and can be proved immediately), the matrix $[\nabla^2 f(x)]^{-1}$ also is positive definite at every x, so that (4.2.3) takes place. It indeed turns out that under assumption (4.2.4) the line search versions of the Newton method possess global convergence:

Proposition 4.2.1 *Let f be a twice continuously differentiable convex function with the Hessian $\nabla^2 f(x)$ being positive definite at every point x, and let x_0 be such that the level set*

$$S = \{x \mid f(x) \leq f(x_0)\}$$

associated with x_0 is bounded. Then the Steepest Newton method, same as the Armijo/Goldstein-based Newton method started at x_0 converges to the unique global minimizer of f.

We shall not prove this proposition here; it will be obtained as an immediate corollary of a forthcoming more general statement.

The "line search" modification of the Basic Newton method is not quite appropriate: as we just have seen, in this modification we meet with severe difficulties when the Newton direction at certain point is not a descent direction of the objective. Another difficulty is that the Newton direction, generally speaking, can simply be undefined – $\nabla^2 f(x)$ may be singular at a point of the trajectory. what to do in this situation? We see that in order to make the Newton method reliable, we need to modify not only the stepsize used in the method, but also the search direction itself, at least in the cases when it is "bad" (is not a descent direction of f or simply is undefined). What we are about to do is to present several general-purpose versions of the method.

4.2.2 Variable Metric Methods

Let us start with developing certain general approach which covers both the Gradient and the Newton methods and allows to understand how to "cure" the Newton method. To outline the

idea, let us come back to the Gradient Descent. What in fact we did in this method? Given previous iterate x, we used local *linear* approximation of the objective:

$$f(y) \approx f(x) + (y-x)^T \nabla f(x) \tag{4.2.5}$$

and choose, as the next direction of movement, the "most perspective" of descent directions of the right hand side. Now, how we were comparing the directions to choose the "most perspective" one? We took the unit ball

$$W = \{d \mid d^T d \leq 1\}$$

of directions and choose in this ball the direction which minimizes the value

$$\bar{f}(x+d) \equiv f(x) + d^T \nabla f(x)$$

of the approximate objective. This direction, as it is immediately seen, is simply the normalized anti-gradient direction

$$-|\nabla f(x)|^{-1} \nabla f(x),$$

and in the Gradient Descent we used it as the current direction of movement, choosing the stepsize in order to achieve "significant" progress in the objective value. Note that instead of minimizing $\bar{f}(x+d)$ on the ball W, we could minimize the quadratic function

$$\hat{f}(d) = d^T \nabla f(x) + \frac{1}{2} d^T d$$

over $d \in \mathbf{R}^n$; the result will be simply the anti-gradient direction $-\nabla f(x)$. This is not the same as the above normalized anti-gradient direction, but the difference in normalization is absolutely unimportant for us – in any case we indent to use line search in the generated direction, so that what in fact we are interested in is the *search ray* $\{x + \gamma d \mid \gamma \geq 0\}$, and proportional, with positive coefficient, directions result in the same ray.

With the outlined interpretation of the Gradient Descent as a method with line search and the search direction given by minimization of the linearized objective $\bar{f}(x+d)$ over $d \in W$, we may ask ourselves: why we use in this scheme the unit ball W, not something else? E.g., why not to use an ellipsoid

$$W_A = \{d \mid d^T A d \leq 1\},$$

A being a positive definite symmetric matrix?
Recall that an ellipsoid in properly chosen coordinates of $\mathbf{R}^n$ becomes a ball (the coordinates are obtained from the standard ones by linear nonsingular transformation). Consequently, a method of the outlined type with W replaced by W_A, i.e., the one where the search direction is given by

$$d = \operatorname*{argmin}_{d \in W_A} \bar{f}(x+d) \tag{4.2.6}$$

has the same "right to exist" as the Gradient Descent. This new "scaled by matrix A Gradient Descent" is nothing but the usual Gradient Descent, but associated with the coordinates on $\mathbf{R}^n$ where W_A becomes the unit ball. Note that the usual Gradient Descent corresponds to the case $A = I$, I being the unit matrix. Now, the initial coordinates are absolutely "occasional" – they have nothing in common with the problem we are solving; consequently, we have no reason to prefer these particular coordinates. Moreover, if we were lucky to adjust the coordinates we use to the "geometry" of the objective (cf. concluding discussion in the previous Lecture), we could get a method with better convergence than the one of the usual Gradient Descent.

Same as above, the direction given by (4.2.6) is, up to renormalization (the latter, as it was already explained, is unimportant – it is "suppressed" by line search), nothing but the direction given by the minimization of the quadratic form

$$\hat{f}_A(d) = d^T \nabla f(x) + \frac{1}{2} d^T A d; \tag{4.2.7}$$

minimizing the right hand side with respect to d (to this end it suffices to solve the Fermat equation $\nabla_d \hat{f}_A(d) \equiv \nabla f + Ad = 0$), we come to the explicit form of the search direction:

$$d = -A^{-1} \nabla f(x). \tag{4.2.8}$$

Note that this direction for sure is a descent direction of f at x, provided that x is not a critical point of f:

$$\nabla f(x) \neq 0 \Rightarrow d^T \nabla f(x) = -(\nabla f(x))^T A^{-1} \nabla f(x) < 0$$

(recall that A is symmetric positive definite, whence A^{-1} also is symmetric positive definite)., so that we are in a good position to apply to f line search in the direction d.

The summary of our considerations is as follows: choosing a positive definite symmetric matrix A, we can associate with it "A-anti-gradient direction" $-A^{-1}\nabla f(x)$, which is a descent direction of f at x (provided that $\nabla f(x) \neq 0$). And we have the same reasons to use this direction in order to improve f as those to use the standard anti-gradient direction (given by the same construction with $Q = I$).

Now we can make one step of generalization more: why should we use at each step of the method a once for ever fixed matrix A instead of varying this matrix from iteration to iteration? The "geometry" of the objective varies along the trajectory, and it is natural to adjust the matrix A to this varying geometry. Thus, we come to the following generic scheme of a *Variable Metric method:*

Algorithm 4.2.1 [Generic Variable Metric method]
Initialization: choose somehow starting point x_0 and set $t = 0$
Step t: given previous iterate x_{t-1},

- *compute $f(x_{t-1})$, $\nabla f(x_{t-1})$ and, possibly, $\nabla^2 f(x_{t-1})$;*
- *choose somehow positive definite symmetric matrix A_t and compute the A_t-anti-gradient direction*

  $$d_t = -A_t^{-1} \nabla f(x_{t-1})$$

  of f at x_{t-1};
- *perform line search from x_{t-1} in the direction d_t, thus getting new iterate*

  $$x_t = x_{t-1} + \gamma_t d_t \equiv x_{t-1} - \gamma_t A_t^{-1} \nabla f(x_{t-1}),$$

  replace t with $t+1$ and loop.

The outlined scheme covers all methods we know so far: to get different versions of the Gradient Descent, we should set $A_t \equiv I$ and should specify the version of line search to be used. With $A_t = \nabla^2 f(x_{t-1})$, we get, as d_t, the Newton direction of f at x_{t-1}, and we come to the Newton method with line search; further specifying the line search by the "programmed" rule $\gamma_t = 1$, we get the Basic Newton method. Thus, *the Basic Newton method is nothing but the Gradient*

Descent scaled by the Hessian of the objective at the current point. Note, anyhow, that the "Newton choice" $A_t = \nabla^2 f(x_{t-1})$ is compatible with the outlined scheme (where A_t should be symmetric positive definite) only when $\nabla^2 f(x_{t-1})$ is positive definite[2]; from the above discussion we know that if it is not the case, we indeed have no reasons to use the Newton direction and should somehow modify it to make it descent. Thus, the generic Algorithm 4.2.1 covers all we know to the moment and provides us with good idea how to "cure" the Newton method at a "bad" iterate: *at such an iterate, we should modify the actual Hessian to make the result positive definite in order to get a descent direction of the objective.* Thus, we come to the family of *Modified Newton methods* – those given by the generic Algorithm 4.2.1 where we use as A_t, whenever it is possible, the Hessian $\nabla^2 f(x_{t-1})$ of the objective. In what follows we shall specify several popular modifications of this type; before this, anyhow, we should check whether our new scheme indeed achieves our target – whether this scheme is globally convergent.

4.2.3 Global convergence of a Variable Metric method

We are about to prove that the generic Variable Metric method (Algorithm 4.2.1), under some reasonable restrictions on the matrices A_t, globally converges to the set of critical points of f. The restrictions are very simple: in the generic method, A_t should be symmetric positive definite, and what we need is "uniform" positive definiteness.

Let us say that Algorithm 4.2.1 is *unifromly descent* with parameters p, P, $0 < p \leq P < \infty$, if the rules for choosing A_t in the algorithm ensure that

$$\lambda_{\min}(A_t) \geq p, \;\; \lambda_{\max}(A_t) \leq P, \;\; t = 1, 2, ... \tag{4.2.9}$$

(as always, $\lambda_{\min}(A)$and $\lambda_{\max}(A)$ denote the minimal and the maximal eigenvalues of a symmetric matrix A). Thus, "uniform descentness" simply means that the matrices A_t never become "too large" (their maximal eigenvalues are bounded away from infinity) and never become "almost degenerate" (their minimal eigenvalues are bounded away from zero).

Theorem 4.2.1 [Global convergence of uniformly descent Variable Metric method]
Let f be twice continuously differentiable function, and let x_0 be such that the level set

$$S = \{x \in \mathbf{R}^n \mid f(x) \leq f(x_0)\}$$

is bounded. Assume that f is minimized by a uniformly descent Variable Metric method started at x_0, and assume that the line search used in the method is either the exact one-dimensional line search, or the Armijo-terminated, or the Goldstein-terminated one. Then the trajectory of the method is bounded, the objective is non-increasing along the trajectory, and all limiting points of the trajectory (which for sure exist, since the trajectory is bounded) belong to the set

$$X^{**} = \{x \mid \nabla f(x) = 0\}$$

of critical points of f.

Proof. We are going to apply our general scheme for convergence analysis. To this end we should cover the method in question by an abstract iterative algorithm and to check that this algorithm satisfies the assumptions of the Global Convergence Theorem 1.3.1.

[2] the scheme requires also symmetry of A_t, but here we have no problems: since f is from the very beginning assumed to be twice continuously differentiable, its Hessians for sure are symmetric

1^0. *Constructing the Abstract Iterative Algorithm.* The abstract iterative algorithm $\mathcal{A}$ we are about to construct is of the type

$$\mathcal{A} = (X = \mathbf{R}^n, \Theta(\cdot)),$$

so that all we need is to define the underlying point-to-set mapping $\Theta(\cdot)$. We define this mapping as the composition

$$\Theta = \mathcal{S}\mathcal{D},$$

where

- the inner point-to-set mapping $\mathcal{D}$ is the mapping from $\mathbf{R}^n$ into $\mathbf{R}^n \times \mathbf{R}^n$; this mapping puts into correspondence to a point $x \in \mathbf{R}^n$ the set $\mathcal{D}(x)$ of all *search elements* (x, d) ("the point x and a search direction $d \in \mathbf{R}^n$") for which the following relations are satisfied:

  $$d^T \nabla f(x) \le -P^{-1}|\nabla f(x)|^2; \quad |d| \le p^{-1}|\nabla f(x)|, \tag{4.2.10}$$

  $0 < p \le P < \infty$ being the parameters responsible for the uniform descent property of the method in question;

- the outer point-to-set mapping $\mathcal{S}$ is the line search mapping

  $$(\text{point } x \in \mathbf{R}^n, \text{ direction } d \in \mathbf{R}^n) \mapsto$$

  $$\mapsto \text{"the result of line search applied to } f \text{ at } x \text{ in the direction } d\text{"}$$

  associated with the corresponding version of the line search (exact, Armijo- or Goldstein-terminated, wee Section 2.4.3).

  For the Armijo- and the Goldstein-terminated line search, which initially are defined only at pairs (x, d) with d being a descent direction of f at x, we, same as in the previous Lecture, extend the mapping onto all pairs (x, d) by setting $\mathcal{S}(x, d) = \{x\}$ in the case when $d^T \nabla f(x) \ge 0$.

It is immediately seen that the resulting abstract iterative algorithm covers the method under consideration. Indeed, all we should prove is that the directions d_t generated in the algorithm are such that the pairs (x_{t-1}, d_t) belong to the image of x_{t-1} under the point-to-set mapping $\mathcal{D}$, i.e., that

$$-(\nabla f(x_{t-1}))^T A_t^{-1} \nabla f(x_{t-1}) \le -P^{-1}|\nabla f(x_{t-1})|^2, \quad |A_t^{-1}\nabla f(x_{t-1})| \le p^{-1}|\nabla f(x_{t-1})|.$$

This is evident: since the eigenvalues of A_t, by definition of the uniform descent property, are between p and P, the eigenvalues of the matrix A_t^{-1} are between P^{-1} and p^{-1}; consequently, for any vector h one has

$$h^T A_t^{-1} h \ge P^{-1} h^T h, \quad |A_t^{-1} h| \le p^{-1}|h|$$

(compute the indicated quantities in the orthonormal eigenbasis of A_t!).

Now we are basically done. First of all, the constructed abstract iterative algorithm $\mathcal{A}$ is descent with respect to the solution set $\Gamma = X^{**}$, the descent function being $Z \equiv f$. Descentness is an immediate consequence of the already mentioned fact that $A_t^{-1}\nabla f(x)$ is a descent direction of f at any point outside X^{**} and is zero at any point of the latter set; recall that the method

includes line search, so that it decreases the objective whenever the search direction is descent. Thus, assumption (i) of Theorem 1.3.1 is satisfied. Since the method is descent, the trajectory $\{x_t\}$ belongs to the level set S of the objective; this set is assumed to be bounded, and, consequently, the trajectory is bounded, as required by assumption (iii) of Theorem 1.3.1. To get convergence of the method as a consequence of Theorem 1.3.1, it remains to verify assumption (ii) of the Theorem, i.e., closedness of the mapping Θ at any point $x \notin X^{**}$. To this end let us note that the mapping $\mathcal{D}$ clearly is closed and locally bounded at any point x (closedness is an immediate consequence of relations (4.2.10) and the continuity of ∇f, local boundedness follows from the second inequality in (4.2.10)). Besides this, we already have mentioned that if x is not a critical point of f, then the d-component of any pair $(x, d) \in \mathcal{D}(x)$ is a descent direction of f at x, so that the outer mapping $\mathcal{S}$ is closed at any pair (x, d) of this type (Propositions 2.4.1 - 2.4.3). According to Corollary 1.3.1, the composite mapping Θ is closed at x, and we are done. ∎

As a corollary of Theorem 4.2.1, we immediately get the result announced in Proposition 4.2.1. Indeed, in the situation addressed by the Proposition f is convex twice continuously differentiable with positive definite Hessian, and the level set S of f associated with the starting point x_0 is bounded and therefore compact. The trajectory of the method the Proposition speaks about is contained in S (since the method by construction is descent), so that the matrices A_t used in this method are the Hessians of f at certain points from the compact set S. Since $\nabla^2 f(x)$ is continuous in x, is positive definite at any x and S is compact, the matrices $\nabla^2 f(x)$ are uniformly in $x \in S$ bounded:

$$P \equiv \max_{x \in S} \lambda_{\max}(\nabla^2 f(x)) < \infty$$

and are "uniformly positive definite":

$$p \equiv \min_{x \in S} \lambda_{\min}(\nabla^2 f(x)) > 0.$$

Thus, the method in question is uniformly descent, and all assumptions of Theorem 4.2.1 indeed are satisfied.

4.2.4 Implementations of the Modified Newton method

Now let us come back to the Newton method. As we remember, our goal is to modify it in order to assure global convergence (at least the same as the one for the Gradient Descent); and, of course, we would not like to loose the fine local convergence properties of the method given by Theorem 4.1.1. We already have an idea how to modify the method: we should use line search in a descent direction given by the Variable Metric scheme, where we use, as A_t, the Hessian $\nabla^2 f(x_{t-1})$, if the latter matrix is positive definite; if it is not the case, we use as A_t certain positive definite modification of the Hessian. Of course, from both theoretical and numerical reasons, we should modify $\nabla^2 f(x_{t-1})$ not only when it fails to be positive definite, but also when it is "close" to a non-positive-definite matrix. Indeed, if we do not modify $\nabla^2 f(x_{t-1})$ when it is close to a non-positive-definite matrix, i.e., when the minimal eigenvalue of the Hessian, being positive, is small, we are unable to ensure the uniform descent property of the method and, consequently, are unable to ensure global convergence; this is a theoretical reason. The practical one is that in the outlined case the condition number of the Hessian is large, so that when solving numerically the Newton system

$$\nabla^2 f(x_{t-1})e = -\nabla f(x_{t-1})$$

in order to find the Newton direction, we meet with severe numerical problems, and the actually computed direction will be far from the exact one.

Thus, our general tactics in the Newton-based implementation of Algorithm 4.2.1 should be as follows: given x_{t-1}, we compute

$$H_{t-1} \equiv \nabla^2 f(x_{t-1})$$

and check whether this matrix is "well positive definite"; if it is the case, we use this matrix as A_t, otherwise replace H_{t-1} with certain "well positive definite" modification A_t of H_{t-1}. Let us look at several implementations of the outlined scheme.

4.2.5 Modifications based on Spectral Decomposition

Recall that, as it is known from Linear Algebra, any symmetric $n \times n$ matrix A possesses *spectral decomposition*

$$A = UDU^T,$$

U being an $n \times n$ orthogonal matrix:

$$U^T U = I,$$

and D being a diagonal matrix. The diagonal entries of D are exactly the eigenvalues of A, while the columns of U are the associated normalized eigenvectors. In particular, A is positive definite if and only if the diagonal entries of D are positive.

In the Modified Newton methods based on spectral decomposition one finds the indicated decomposition of the Hessian at every step:

$$H_t \equiv \nabla^2 f(x_t) = U_t D_t U_t^T, \tag{4.2.11}$$

and compares the eignevlaues of H_t – i.e., the diagonal entries of D_t – with a once for ever chosen "threshold" $\delta > 0$. If all the eigenvalues are $\geq \delta$, we qualify H_t as "well positive definite" and use it as A_{t+1}. If some of the eigenvalues of H_t are $< \delta$ (e.g., are negative), we set

$$A_{t+1} = U_t \bar{D}_t U_t^T,$$

where $\bar{D}_t$ is diagonal matrix obtained from D by replacing the diagonal entries which are smaller than δ with δ. Another way to "correct" H_t is to replace the negative diagonal values in D_t by their absolute values (and then to replace by δ those diagonal entries, if any, which are less than δ).

Both indicated strategies result in

$$\lambda_{\min}(A_t) \geq \delta, \ t = 1, 2, ...,$$

and never increase "significantly" the norm of the Hessian:

$$\lambda_{\max}(A_t) \leq \max\left[|\lambda_{\max}(H_t)|, |\lambda_{\min}(H_t)|, \delta\right];$$

as a result, the associated modified Newton method turns out to be uniformly descent (and thus globally converging), provided that the level set of f associated with the starting point is bounded (so that the Hessians along the trajectory are uniformly bounded). A drawback of the approach is its relatively large arithmetic cost: to find spectral decomposition (4.2.11) to machine precision, it normally requires between $2n^3$ and $4n^3$ arithmetic operations, n being the row size of H_t (i.e., the design dimension of the optimization problem). As we shall see in a while, this is, in a sense, too much.

4.2.6 Levenberg-Marquardt Modification

In the *Levenberg-Marquardt* modification of the Newton method we choose A_{t+1} as the "regularization"

$$A_{t+1} = \epsilon_t I + H_t \tag{4.2.12}$$

of the actual Hessian, where the "regularization parameter" $\epsilon_t \geq 0$ is chosen to make the right hand side "well positive definite" – to have all its eigenvalues at least the chosen in advance positive threshold δ. This is the same as to ensure that $A_{t+1} \geq \delta I$ [3)].

To find the desired ϵ_t, we first check whether the requirement

$$H_t > \delta I$$

is satisfied; if it is the case, we choose $\epsilon_t = 0$ and $A_{t+1} = H_t$, thus getting the pure Newton direction. Now assume that the inequality $H_t \geq \delta I$ dose not take place. The matrix

$$A(\epsilon) = \epsilon I + H_t - \delta I$$

for sure is positive definite when $\epsilon > 0$ is large enough, and in the case in question it is not positive definite for $\epsilon = 0$. Thus, there exists the smallest $\epsilon = \epsilon^* \geq 0$ for which $A(\epsilon)$ is positive semidefinite. And what we do in the Levenberg-Marquardt scheme are several steps of the bisection routine to find a "tight" upper bound ϵ_t for ϵ^*. The resulting upper bound is used to create A_t according to (4.2.12).

The essence of the matter is, of course, how we verify whether a given trial value of ϵ is appropriate, i.e., whether it results in positive semidefinite $A(\epsilon)$. It would be absolutely senseless to answer this question by computing spectral decomposition of $A(\epsilon)$ – this computation is exactly what we are trying to avoid. Fortunately, there is a much more efficient way to check whether a given symmetric matrix is positive definite – Cholesky factorization.

Cholesky Factorization

It is known from Linear Algebra, that a symmetric $n \times n$ matrix A is positive definite is and only if it admits factorization

$$A = LDL^T, \tag{4.2.13}$$

where

- L is lower-triangular $n \times n$ matrix with unit diagonal entries;
- D is diagonal matrix with positive diagonal entries.

The Cholesky Factorization is an algorithm which computes the factors L and D in decomposition (4.2.13), if such a decomposition exists. The algorithm is given by the recurrence

$$d_j = a_{ii} - \sum_{s=1}^{j-1} d_s l_{js}^2, \tag{4.2.14}$$

[3] from now on, for symmetric matrices A, B the inequality $A \geq B$ means that $A - B$ is positive semidefinite, and $A > B$ means that $A - B$ is positive definite. Note that, for a symmetric matrix A, the relation $\lambda_{\max}(A) \leq a$ is equivalent to $aI - A \geq 0$, while $\lambda_{\min}(A) \geq a$ is equivalent to $A - aI \geq 0$

$$l_{ij} = \frac{1}{d_j}\left(a_{ij} - \sum_{s=1}^{j-1} d_s l_{js} l_{is}\right), \quad j \le i \le n, \tag{4.2.15}$$

(d_j is j-th diagonal entry of D, l_{ij} and a_{ij} are the entries of L and A, respectively). The indicated recurrence allows to compute D and L, if they exist, in

$$\mathcal{C}_n = \frac{n^3}{6}(1 + o(1))$$

arithmetic operations ($o(1) \to 0$, $n \to \infty$), in numerically stable manner. Note that L is computed in the "column by column" fashion: the order of computations is

$$d_1 \to l_{1,1}, l_{2,1}, ..., l_{n,1} \to d_2 \to l_{2,2}, l_{3,2}, ..., l_{n,2} \to d_3 \to l_{3,3}, l_{4,3}, ...l_{n,3} \to ... \to d_n \to l_{n,n};$$

if the right hand side in (4.2.14) turns out to be nonpositive for some j, this indicates that A is not positive definite.

The main advantage of Cholesky factorization is not only its ability to check in $\mathcal{C}_n$ computations whether A is positive definite, but also to get, as a byproduct, the factorization (4.2.13) of a positive definite A. With this factorization, we can immediately solve the linear system

$$Ax = b;$$

the solution x to the system can be identified by backsubstitutions – sequential solving, for u, v and x, two triangular and one diagonal systems

$$Lu = b; \quad Dv = u; \quad L^T x = v,$$

and this computation requires only $O(n^2)$ arithmetic operations. The resulting method – Cholesky decomposition with subsequent backsubstitutions (the *square root method*) – is thought to be the most efficient in the operation count and most numerically stable Linear Algebra routine for solving linear systems with general-type symmetric positive definite matrices.

Coming back to the Levenberg-Marquardt scheme, we observe that one can check positive definiteness of $A(\epsilon)$ for a given ϵ in about $n^3/6$ arithmetic operations – something from 12 to 24 times cheaper than the cost of spectral decomposition. Thus, the Levenberg-Marquardt scheme with 5-10 bisection steps in ϵ (this is sufficient for "smart" implementation of the scheme) is numerically less expensive then the scheme based on spectral decomposition.

4.2.7 Modified Cholesky Factorization scheme

The next implementation of the outlined Modified Newton approach, and the most practical one, is based on *Modified Cholesky Decomposition*. The idea is to apply to the actual Hessian H_t the Cholesky Factorization algorithm until it is possible, i.e., until the computed values of d_j are at least prescribed threshold $\delta > 0$. When we "meet with an obstacle" – the right hand side of (4.2.14) becomes less than δ, – we increase the corresponding diagonal entry in H_t to make the right hand side of (4.2.14) equal to δ and continue the computation. As a result, we get Cholesky factorization of certain matrix

$$A_{t+1} = H_t + \Delta_t$$

which differs from H_t by a diagonal matrix Δ_t, and this is the modification we use in Algorithm 4.2.1. If H_t is "well positive definite", the "regularizing component" Δ_t automatically turns out

to be zero, and we get the pure Newton direction, as it should be. The main advantage of the indicated scheme as compared to the Levenberg-Marquardt one is that we need no bisection and get the desired modification with, basically, the same effort as in a single Cholesky factorization of a positive definite matrix. Moreover, we end up with decomposed A_{t+1}, so that we are ready to find the desired direction d_{t+1} at a low – quadratic in n – arithmetic cost. Thus, in the outlined scheme the arithmetic cost of a step (including the expenses for modification of the Hessian when this modification is needed) is close to the arithmetic cost of solving the Newton system for the case when no modification is needed.

An additional reasonable idea is to allow in course of the factorization interchanges in order of columns/rows of the processed matrix, which allows to postpone, until it is possible, the necessity in modifying diagonal entries. As a result of these interchanges, what will be in fact decomposed according to (4.2.13) is not the matrix $A_{t+1} = H_t + \Delta_t$ itself, but the permuted matrix $\Pi_t^T A_{t+1} \Pi_t$. Here Π_t is a permutation matrix, i.e., the one obtained from the unit matrix by certain permutation of columns. For our purposes it is the same what to decompose – the original matrix or a permuted one.

The algorithm for the outlined scheme of modifying the Hessian is as follows[4]

Algorithm 4.2.2 *[Modified Cholesky Factorization]*
Input: *Hessian H to be modified and decomposed; "threshold" $\delta > 0$; "machine zero" $\epsilon^* > 0$*
Output: *lower-triangular matrix L with unit diagonal entries; diagonal matrix D with positive diagonal entries d_j; permutation matrix Π_n such that*

$$LDL^T = \Pi_n^T H \Pi_n + \Delta,$$

Δ being a diagonal matrix.
Preprocessing: *set*

$$\beta^2 = \max\{\gamma, \xi/\nu, \epsilon^*\},$$

where

- *γ is the maximum of absolute values of the diagonal entries of H;*
- *ξ is the maximum of absolute values of the non-diagonal entries of H;*
- *$\nu = \sqrt{n^2 - 1}$, $n > 1$ being the row size of H.*

Initialization: *set the column index j to 1. Set $c_{ii} = H_{ii}$, $i = 1, ..., n$. Set $\Pi_0 = I$.*
Step j: *we already have*

- *first $j-1$ diagonal entries d_s, $1 \le s \le j-1$, of D,*
- *first $j-1$ columns l_{is}, $1 \le i \le n$, $1 \le s \le \min[j-1, i]$, of L,*
- *auxiliary quantities c_{is}, $s = 1, ..., j-1$, $i = j, ..., n$, and c_{ii}, $i = j, ..., n$; these quantities should be treated as filling the first $j-1$ columns in last $n-j+1$ rows in certain lower-triangular matrix C_{j-1} and last $n-j+1$ diagonal entries in the matrix (draw picture for $j = 2$)*
- *permutation matrix Π_{j-1}.*

[4] see Gill, Ah.E., Murray, W. and Wright, M.H., *Practical Optimization*, Academic Press, 1981

At step j we perform the following actions:

1. Row and column interchanges: *find the smallest index q such that $|c_{qq}| = \max_{j\le i\le n} |c_{ii}|$. Interchange*

- *all information corresponding to rows and columns q and j in H;*
- *columns q and j in Π_{j-1}, thus coming to the matrix Π_j;*
- *rows q and j in the already computed part (i.e., in the first $j-1$ columns) of L;*
- *rows q and j in the first $j-1$ columns of matrix C_{j-1};*
- *c_{jj} and c_{qq}.*

2. Compute j-th row of L: *set*

$$l_{js} = c_{js}/d_s, \;\; s = 1, ..., j-1;$$

compute the quantities

$$c_{ij} = H_{ij} - \sum_{s=1}^{j-1} l_{js} c_{is}, \;\; i = j+1, ..., n,$$

thus computing part of the "new" (those absent in C_{j-1}) entries in C_j, and set

$$\theta_j = \max_{j+1\le i\le n} |c_{ij}|$$

($\theta_j = 0$ for $j = n$).

3. Compute j-th diagonal element of D: *define*

$$d_j = \max\{\delta, |c_{jj}|, \theta_j^2/\beta^2\}$$

and the diagonal modification

$$\Delta_{jj} = d_j - c_{jj}.$$

If $j = n$, terminate.

4. Update c_{ii}'s: *replace c_{ii} by $c_{ii} - c_{ii}^2/d_j$, $i = j+1, ..., n$, thus completing the computation of C_j. Set j to $j+1$ and go to the next step.*

It can be proved that with the presented modification algorithm, the entries r_{ij} in the matrix $R = LD^{1/2}$ and the diagonal entries d_j in D satisfy the inequalities

$$|r_{ij}| \le \beta; \quad d_j \ge \delta; \tag{4.2.16}$$

it can derived from this relation that the "modified Hessian"

$$A \equiv H + \Pi_n \Delta \Pi_n^T \equiv H + E$$

which actually is decomposed by the Algorithm 4.2.2, satisfies the bounds

$$pI \le A \le p'(|H| + \epsilon^*),$$

where p and p' are positive constants depending on the row size n of the matrix and on δ only, not on H. It is easily seen that the indicated property ensures uniform descentness of Algorithm 4.2.1, provided that the level set of f associated with the starting point is bounded.

It can be proved also that if H itself is "well positive definite", namely, it admits Cholesky factorization

$$H = LDL^T$$

with diagonal entries in D not less than δ, then the Algorithm results in zero correction matrix E, as it should be for a reasonable modifying algorithm.

Last remark relates to the memory requirements. Normally, the Hessian matrix H_t in the Modified Newton method is used only to compute the (modified) Newton direction. It follows that when running Algorithm 4.2.2, we can overwrite the relevant parts of H_t by already formed entries of L and C_j. Thus, with proper implementation no additional memory as compared to the one to store H_t is needed.

4.2.8 The Newton Method: how good it is?

We have investigated the basic version of the Newton method, along with several modifications aimed to make the method globally converging. Finally, what are the basic advantages and disadvantages of the method?

The main advantage of the method is its fast (quadratic) local convergence to a nondegenerate local minimizer of the objective, provided that we were lucky to bring the trajectory close enough to such a minimizer.

Rigorously speaking, the indicated attractive property is possessed only by the basic version of the method. For a modified version, the indicated phenomenon takes place only when at the final phase of the process the modified method becomes close to the basic Newton method, i.e., it eventually uses nearly Newton search directions and nearly unit stepsizes. Whether this indeed is a case, it depends, first of all, on whether the modification we use manages to drive the trajectory to a small neighbourhood of a nondegenerate local minimizer of f; if it is not so, we have no reasons to expect fast convergence. Thus, let us assume that *the trajectory of the modified Newton method under consideration converges to a nondegenerate local minimizer x^* of f*[5]. Is then the convergence asymptotically quadratic?

The answer depends on the rules for modifying the Newton direction and for those of line search. Namely, if the Hessian of the objective at x^* is not only positive definite (this is a qualitative property which means nothing), but is "well positive definite", so that the technique used for modifying the Hessian remains it unchanged in a neighbourhood of x^*, then the search direction used in the method eventually becomes the exact Newton direction, as it should be for fast local convergence. This is fine but insufficient: to get something asymptotically close to the Basic Newton method, we need also nearly unit stepsizes at the final phase of the process; whether it is so or not, it depends on line search we use. One can prove, e.g., that the required property of "asymptotically unit stepsizes in the Newton direction" is ensured by the exact line search. To get the same behaviour it in the Armijo-terminated line search, the parameters ϵ and η of the underlying Armijo test should be chosen properly (e.g., $\epsilon = 0.2$ and $\eta = 10$), and we should always start line search with testing the unit stepsize.

In spite of all indicated remarks which say that the modifications of the Basic Newton method aimed to ensure global convergence may spoil the theoretical quadratic convergence of the basic method (either because of bad implementation, or due to nearly-degeneracy of the minimizer the trajectory converges to), the Newton-based methods should be qualified as the most efficient tool

[5] this, e.g., for sure is the case when the modification is uniformly descent and f is strongly convex: here the only critical point is a nondegenerate global minimizer, while convergence to the set of critical points is given by Theorem 4.2.1

for smooth unconstrained minimization. The actual reason of the efficiency of these methods is their intrinsic ability (spoiled, to some extent, by the modifications aimed to ensure global convergence) to adjust themselves to the "local geometry" of the objective.

The main shortcoming of the Newton-type methods is their relatively high computational cost. To run such a method, we should be able to compute the Hessian of the objective and should solve at each step an $n \times n$ system of linear equations. If the objective is too complicated and/or the dimension n of the problem is large, these requirements may become "too costly" from the viewpoint of programming, execution time and memory considerations. In order to overcome these drawbacks, significant effort was invested into theoretical and computational development of first-order routines (those not using second-order derivatives) capable to "imitate" the Newton method. These are the methods we are about to consider in our nearest lectures.

4.3 Newton Method and Self-Concordant Functions

The traditional results on the Newton method establish no more than its fast *asymptotical* convergence; global efficiency estimates can be proved only for modified versions of the method, and these global estimates never are better than those for the Gradient Descent. In this section we consider a family of objectives – the so called *self-concordant ones* – where the Newton method admits excellent *global* efficiency estimates. This family underlies the most recent and advanced *Interior Point methods for large-scale Convex Optimization*.

4.3.1 Preliminaries

The traditional "starting point" in the theory of the Newton method – Theorem 4.1.1 – possesses an evident drawback (which, anyhow, remained unnoticed by generations of researchers). The Theorem establishes local quadratic convergence of the Basic Newton method as applied to a function f with positive definite Hessian at the minimizer, this is fine; but what is the "quantitive" information given by the Theorem? What indeed is the "region of quadratic convergence" $\mathcal{Q}$ of the method – the set of those starting points from which the method converges quickly to x^*? The proof provides us with certain "constructive" description of $\mathcal{Q}$, but look – this description involves differential characteristics of f like the magnitude of the third order derivatives of f in a neighbourhood of x^* (this quantity underlies the constant β_1 from the proof) and the bound on the norm of inverted Hessian in this neighbourhood (the constant β_2; in fact this constant depends on β_1, the radius of the neighbourhood and the smallest eigenvalue of $\nabla^2 f(x^*)$). Besides this, the "fast convergence" of the method is described in terms of the the behaviour of the standard Euclidean distances $|x_t - x^*|$. All these quantities – magnitudes of third-order derivatives of f, norms of the inverted Hessian, distances from the iterates to the minimizer – are "frame-dependent": they depend on the choice of Euclidean structure on the space of variables, on what are the orthonormal coordinates used to compute partial derivatives, Hessian matrices and their norms, etc. When we vary the Euclidean structure (pass from the original coordinates to another coordinates via a non-orthogonal linear transformation), all these quantities somehow vary, same as the description of $\mathcal{Q}$ given by Theorem 4.1.1. On the other hand, when passing from one Euclidean structure on the space of variables to another, we do not vary neither the problem, nor the Basic Newton method. Indeed, the latter method is independent of any a priori coordinates, as it is seen from the following "coordinateless" description of the method:

To find the Newton iterate x_{t+1} of the previous iterate x_t, take the second order Taylor expansion of f at x_t and choose, as x_{t+1}, the minimizer of the resulting quadratic form.

Thus, the coordinates are responsible only for the *point of view* we use to investigate the process and are absolutely irrelevant to the process itself. And the results of Theorem 4.1.1 in their quantitive part (same as other traditional results on the Newton method) reflect this "point of view", not only the actual properties of the Newton process! This "dependence on viewpoint" is a severe drawback: how can we get correct impression of actual abilities of the method looking at the method from an "occasionally chosen" position? This is exactly the same as to try to get a good picture of a landscape directing the camera in a random manner.

4.3.2 Self-concordance

After the drawback of the traditional results is realized, could we choose a proper point of view – to orient our camera properly, at least for "good" objectives? Assume, e.g., that our objective f is convex with nondegenerate Hessian. Then at every point x there is a natural, intrinsic for the objective, Euclidean structure on the space of variables, namely, the one given by the Hessian of the objective at x; the corresponding norm is

$$|h|_{f,x} = \sqrt{h^T \nabla^2 f(x) h} \equiv \sqrt{\frac{d^2}{dt^2}|_{t=0} f(x+th)}. \tag{4.3.1}$$

Note that the first expression for $|h|_{f,x}$ seems to be "frame-dependent" – it is given in terms of coordinates used to compute inner product and the Hessian. But in fact the value of this expression is "frame-independent", as it is seen from the second representation of $|h|_{f,x}$.

Now, from the standard results on the Newton method we know that the behaviour of the method depends on the magnitudes of the third-order derivatives of f. Thus, these results are expressed, among other, in terms of upper bounds

$$\left| \frac{d^3}{dt^3}|_{t=0} f(x+th) \right| \leq \alpha$$

on the third-order directional derivatives of the objective, *the derivatives being taken along unit in the standard Euclidean metric directions h*. What happens if we impose similar upper bound on the third-order directional derivatives *along the directions of the unit $|\cdot|_{f,x}$ length* rather than along the directions of the unit "usual" length? In other words, what happens if we assume that

$$|h|_{f,x} \leq 1 \Rightarrow \left| \frac{d^3}{dt^3}|_{t=0} f(x+th) \right| \leq \alpha \quad ?$$

Since the left hand side of the concluding inequality is of homogeneity degree 3 with respect to h, the indicated assumption is equivalent to the one

$$\left| \frac{d^3}{dt^3}|_{t=0} f(x+th) \right| \leq \alpha |h|^3_{f,x} \quad \forall x \; \forall h.$$

Now, the resulting inequality, qualitatively, remains true when we *scale* f – replace it by λf with positive constant λ, but the value of α varies: $\alpha \mapsto \lambda^{-1/2}\alpha$. We can use this property to normalize the constant factor α, e.g., to set it equal to 2 (this is the most technically convenient normalization).

Thus, we come to the main ingredient of the notion of a

self-concordant function: a three times continuously differentiable convex function f satisfying the inequality

$$\left|\frac{d^3}{dt^3}|_{t=0}f(x+th)\right| \leq 2|h|_{f,x}^3 \equiv 2\left[\frac{d^2}{dt^2}|_{t=0}f(x+th)\right]^{3/2} \qquad \forall h \in \mathbf{R}^n. \tag{4.3.2}$$

We do not insist on f to be defined everywhere; it suffices to assume that the domain of f is an open convex set $Q_f \subset \mathbf{R}^n$, and that (4.3.2) is satisfied at every point $x \in Q_f$. The second part of the definition of a self-concordant function is that

Q_f is a "natural domain" of f, so that f possesses the barrier property with respect to Q_f – blows up to infinity when a sequence of interior points of Q_f approaches a boundary point of the domain:

$$\forall\{x_i \in Q_f\}: \; x_i \to x \in \partial Q_f, i \to \infty \Rightarrow f(x_i) \to \infty, \; i \to \infty. \tag{4.3.3}$$

Of course, the second part of the definition imposes something on f only when the domain of f is less than the entire $\mathbf{R}^n$.

Note that the definition of a self-concordant function is "coordinateless" – it imposes certain inequality between third- and second-order directional derivatives of the function and certain behaviour of the function on the boundary of its domain; all notions involved are "frame-independent".

4.3.3 Self-concordant functions and the Newton method

It turns out that the Newton method *as applied to a self-concordant function f* possesses extremely nice *global* convergence properties. Namely, one can more or less straightforwardly prove the following statements:

Proposition 4.3.1 [Self-concordant functions and the Newton method]
Let f be strongly self-concordant, and let $\nabla^2 f$ be nondegenerate at some point of Q_f (this for sure is the case when Q_f does not contain lines, e.g., is bounded). Then

- (i) [Nondegeneracy] *$\nabla f(x)$ is positive definite at every point $x \in Q_f$;*
- (ii) [Existence of minimizer] *If f is below bounded (which for sure is the case when Q_f is bounded), then f attains its minimum on Q_f, the minimizer being unique;*
- (iii) [Damped Newton method] *The started at arbitrary point $x_0 \in Q_f$ process*

$$x_{t+1} = x_t - \frac{1}{1+\lambda(f,x_t)}[\nabla^2 f(x_t)]^{-1}\nabla f(x_t), \quad \lambda(f,x) = \sqrt{(\nabla f(x))^T[\nabla^2 f(x)]^{-1}\nabla f(x)} \tag{4.3.4}$$

  – the Newton method with particular stepsizes

$$\gamma_{t+1} = \frac{1}{1+\lambda(f,x_t)}$$

  – possesses the following properties:

  - (iii.1) *The process keeps the iterates in Q_f and is therefore well-defined;*

– (iii.2) *If f is below bounded on Q_f (which for sure is the case if $\lambda(f,x)<1$ for some $x\in Q_f$) then $\{x_t\}$ converges to the unique minimizer x_f^* of f on Q_f;*

– (iii.3) *Each step of the process (4.3.4) decreases f "significantly", provided that $\lambda(f,x_t)$ is not too small:*

$$f(x_t)-f(x_{t+1})\geq \lambda(f,x_t)-\ln\left(1+\lambda(f,x_t)\right); \tag{4.3.5}$$

– (iii.4) *For every t, one has*

$$\lambda(f,x_t)<1\Rightarrow f(x_t)-f(x_t^*)\leq -\ln\left(1-\lambda(f,x_t)\right)-\lambda(f,x_t) \tag{4.3.6}$$

and

$$\lambda(f,x_{t+1})\leq \frac{2\lambda^2(f,x_t)}{1-\lambda(f,x_t)}. \tag{4.3.7}$$

The indicated statements demonstrate extremely nice *global convergence* properties of the Damped Newton method (4.3.4) as applied to a self-concordant function f. Namely, assume that f is self-concordant with nondegenerate Hessian at certain (and then, as it was mentioned in the above proposition, at any) point of Q_f. Assume, besides this, that f is below bounded on Q_f (and, consequently, attains its minimum on Q_f by (ii)). According to (iii), the Damped Newton method is keeps the iterates in A_f. Now, we may partition the trajectory into two parts:

- the initial phase: from the beginning to the first moment, let it be called t^*, when $\lambda(f,x_t)\leq 1/4$;
- the final phase: starting from the moment t^*.

According to (iii.3), at every step of the initial phase the objective is decreased at least by *absolute* constant

$$\kappa=\frac{1}{4}-\ln\frac{5}{4}>0;$$

consequently,

- *the initial phase is finite and is comprised of no more than*

$$\mathcal{N}_{\rm ini}=\frac{f(x_0)-\min_{Q_f}f}{\kappa}$$

iterations.

Starting with $t=t^*$, we have in view of (4.3.6):

$$\lambda(f,x_{t+1})\leq\frac{2\lambda^2(f,x_t)}{1-\lambda(f,x_t)}\leq\frac{1}{2}\lambda(f,x_t);$$

thus,

- *starting with $t=t^*$, the quantities $\lambda(f,x_t)$ converge quadratically to 0 with objective-independent rate.*

According to (4.3.7),

- *starting with $t=t^*$, the residuals in terms of the objective $f(x_t)-\min_{Q_f}f$ also converge quadratically to zero with objective-independent rate.*

Combining the above observations, we observe that

- *the number of steps of the Damped Newton method required to reduce the residual* $f(x_t) - \min f$ *in the value of a self-concordant below bounded objective to a prescribed value* $\epsilon < 0.1$ *is no more than*

$$N(\epsilon) \leq O(1) \left[[f(x_0) - \min f] + \ln \ln \frac{1}{\epsilon} \right], \tag{4.3.8}$$

  $O(1)$ *being an absolute constant.*

It is also worthy of note what happens when we apply the Damped Newton method to a *below unbounded* self-concordant f. The answer is as follows:

- *for a below unbounded* f *one has* $\lambda(f,x) \geq 1$ *for every* x *(see (iii.2)), and, consequently, every step of the method decreases* f *at least by the absolute constant* $1 - \ln 2$ *(see (iii.3)).*

The indicated picture gives a "frame-" and "objective-independent" description of the *global* behaviour of the Damped Newton method as applied to a below bounded self-concordant function. Note that the quantity $\lambda(f,x)$ used to describe the behaviour of the method at the first glance is "coordinate dependent" (see (4.3.4)), but in fact this quantity is "coordinateless". Indeed, one can easily verify that

$$\frac{\lambda^2(f,x)}{2} = \hat{f}(x) - \min_y \hat{f}(y),$$

where

$$\hat{f}(y) = f(x) + (y-x)^T \nabla f(x) + \frac{1}{2}(y-x)^T \nabla^2 f(x)(y-x)$$

is the second-order Taylor expansion of f at x. This is a coordinateless definition of $\lambda(f,x)$.

Note that the region of quadratic convergence of the Damped Newton method as applied to a below bounded self-concordant function f is, according to (iii.4), the set

$$\mathcal{Q}_f = \{x \in Q_f \mid \lambda(f,x) \leq \frac{1}{4}\}. \tag{4.3.9}$$

4.3.4 Self-concordant functions: applications

At the first glance, the family of self-concordant functions is rather "thin" – the functions are given by certain "strict" differential inequality, and "a general", even convex and smooth, f hardly may happen to be self-concordant. Thus, what for these elegant results on the behaviour of the Newton method on self-concordant functions?

The answer is as follows: it turns out that (even constrained) Convex Programming problems of reasonable (in principle – of arbitrary) analytic structure can be reduced to a "small series" of problems of minimizing self-concordant functions. Applying to these auxiliary problems the Damped Newton method, we come to the theoretically most efficient (and extremely efficient in practice) *Interior Point Polynomial Time methods for Convex Optimization*. Appearance of these methods (starting with the landmark paper of N. Karmarkar (1984), where the first method of this type for Linear Programming was proposed) definitely was the main event in Optimization during the last decade, it completely changed the entire area of large-scale Convex Optimization, in particular, Linear Programming.

Right now I am not going to speak about Interior Point methods in more details; we shall come back to these methods at the end of our course. What should be stressed now is that *the crucial point in the design of the Interior Point methods is our ability to construct "good" self-concordant functions with prescribed domains.* To this end it is worthy to note how to construct self-concordant functions. Here the following "raw materials" and "combination rules" are useful:

Raw materials: basic examples of self-concordant functions. For the time being, the following examples are sufficient:

- [Convex quadratic (e.g., linear) form] The convex quadratic function
$$f(x) = \frac{1}{2}x^T Ax - b^T x + c$$
(A is symmetric positive semidefinite $n \times n$ matrix) is self-concordant on $\mathbf{R}^n$;
- [Logarithm] The function
$$-\ln(x)$$
is self-concordant with the domain $\mathbf{R}_+ = \{x \in \mathbf{R} \mid x > 0\}$;
- [Extension of the previous example: Logarithmic barrier, linear/quadratic case] Let
$$Q = \{x \in \mathbf{R}^n \mid \phi_j(x) < 0,\ j = 1, ..., m\}$$
be a nonempty set in $\mathbf{R}^n$ given by m strict convex quadratic (e.g., linear) inequalities. Then the function
$$f(x) = -\sum_{i=1}^{m} \ln(-\phi_i(x))$$
is self-concordant with the domain equal to Q.

Combination rules: simple operations with functions preserving self-concordance

- [Linear combination with coefficients ≥ 1] Let f_i, $i = 1, ...m$, be self-concordant functions with the domains Q_{f_i}, let these domains possess a nonempty intersection Q, and let $\alpha_i \geq 1$, $i = 1, ..., m$, be given reals. Then the function
$$f(x) = \sum_{i=1}^{m} \alpha_i f_i(x)$$
is self-concordant with the domain equal to Q.

  In particular, the sum of a self-concordant function and a convex quadratic function (e.g., a linear one) is self-concordant;
- [Affine substitution] Let $f(x)$ be self-concordant with the domain $Q_f \subset \mathbf{R}^n$, and let $x = A\xi + b$ be an affine mapping from $\mathbf{R}^k$ into $\mathbf{R}^n$ with the image intersecting Q_f. Then the composite function
$$g(\xi) = f(A\xi + b)$$
is self-concordant with the domain
$$Q_g = \{\xi \mid A\xi + b \in Q_f\}$$
being the inverse image of Q_f under the affine mapping in question.

To justify self-concordance of the indicated functions, same as the validity of the combination rules, only minimal effort is required; at the same time, these examples and rules give almost all required to establish excellent global efficiency estimates for Interior Point methods as applied to Linear Programming and Convex Quadratically Constrained Quadratic programming.

After we know examples of self-concordant functions, let us look how our now understanding of the behaviour of the Newton method on such a function differs from the one given by Theorem 4.1.1. To this end consider a particular self-concordant function – the logarithmic barrier

$$f(x) = -\ln(\delta - x_1) - \ln(\delta + x_1) - \ln(1 - x_2) - \ln(1 + x_2)$$

for the 2D rectangle

$$D = \{x \in \mathbf{R}^2 \mid |x_1| < \delta, |x_2| < 1\};$$

in what follows we assume that the rectangle is "wide", i.e., that

$$\delta >> 1.$$

This function indeed is self-concordant (see the third of the above "raw material" examples). The minimizer of the function clearly is the origin; the region of quadratic convergence of the Damped Newton method is given by

$$\mathcal{Q} = \{x \in D \mid \frac{x_1^2}{\delta^2 + x_1^2} + \frac{x_2^2}{1 + x_2^2} \leq \frac{1}{32}\}$$

(see (4.3.9)). We see that the region of quadratic convergence of the Damped Newton method is large enough – it contains, e.g., 8 times smaller than D concentric to D rectangle D'. Besides this, (4.3.8) says that in order to minimize f to inaccuracy, in terms of the objective, ϵ, starting with a point $x_0 \in D$, it suffices to perform no more than

$$O(1)\left[\ln \frac{1}{\| x_0 \|} + \ln \ln \frac{1}{\epsilon}\right]$$

steps, where $O(1)$ is an absolute constant and

$$\| x \| = \max\{\frac{|x_1|}{\delta}, |x_2|\}.$$

Now let us look what Theorem 4.1.1 says. The Hessian $\nabla^2 f(0)$ of the objective at the minimizer is

$$H = \begin{pmatrix} 2\delta^{-2} & 0 \\ 0 & 2 \end{pmatrix},$$

and $|H^{-1}| = O(\delta^2)$; in, say, 0.5-neighbourhood U of $x^* = 0$ we also have $|[\nabla^2 f(x)]^{-1}| = O(\delta^2)$. The third-order derivatives of f in U are of order of 1. Thus, in the notation from the proof of Theorem 4.1.1 we have $\beta_1 = O(1)$ (this is the magnitude of the third order derivatives of f in U), $U' = U$, $r = 0.5$ (the radius of the circle $U' = U$) and $\beta_2 = O(\delta^2)$ (this is the upper bound on the norm of the inverted Hessian of f in U'). According to the proof, the region U'' of quadratic convergence of the Newton method is ρ-neighbourhood of $x^* = 0$ with

$$\rho = \min[r, (2\beta_1\beta_2)^{-1}] = O(\delta^{-2}).$$

Thus, according to Theorem 4.1.1, the region of quadratic convergence of the method becomes the smaller the larger is δ, while the actual behaviour of this region is quite opposite.

In this simple example, the aforementioned drawback of the traditional approach – its "frame-dependence" – is clearly seen. Applying Theorem 4.1.1 to the situation in question, we used extremely bad "frame" – Euclidean structure. If we were clever enough to scale the variable x_1 before applying Theorem 4.1.1 – to divide it by δ – it would become absolutely clear that the behaviour of the Newton method is absolutely independent of δ, and the region of quadratic convergence of the method is a once for ever fixed "fraction" of the rectangle D.

To extract certain moral from this story about self-concordance, let me note that it is one of the examples of what is Mathematics and progress in Mathematics: the known from XVII Century very natural and simple (and seemingly perfectly well understood decades ago) optimization method gave rise to one of the most advanced and most recent breakthroughs in Optimization.

Assignment # 4 (Lecture 4)

Exercise 4.3.1 *Assume that f is twice continuously differentiable with nondegenerate (not necessarily positive definite) Hessian at x. As we know, the Newton direction*

$$e(x) = -[\nabla^2 f(x)]^{-1}\nabla f(x)$$

not necessarily is a descent direction for f at x. Prove that $e(x)$ always is a descent direction of

$$g(\cdot) = |\nabla f(\cdot)|^2,$$

i.e., that

$$g(x) > 0 \Rightarrow e^T(x)\nabla g(x) < 0.$$

Could you use this observation to build (on the paper, not on a computer) a Newton-type algorithm for approximating critical points of f ?

Exercise 4.3.2 *Write a code implementing the Modified Newton method. Use the Levenberg-Marquardt scheme and the line search subroutine which you were supposed to create when doing Assignment 2.*
Think about

- *reasonable choice of the parameter δ and the bisection policy in the "modifying the Hessian" part of the algorithm;*
- *implementation of line search aimed to use unit stepsizes whenever they pass the termination test you use in your line search*

If you are working with MATLAB, you could use the MATLAB Cholesky factorization, otherwise you are welcome to write your own Cholesky code.
Test your code on

- *The Rosenbrock function*

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

  the starting point being $(-1.2, 1.0)^T$;
- *The logarithmic barrier*

  $$f(x) = -\sum_{i=1}^{10} \ln(i^2 - x_i^2) \quad [x \in \mathbf{R}^{10}],$$

  the starting point being $(x_0)_i = i - \epsilon i$, $i = 1, ..., 10$, for $\epsilon \in \{0.001; 0.01; 0.1\}$.

Exercise 4.3.3 *Write a code implementing the Damped Newton method (4.3.4) and run the code on the function*

$$f(x) = \frac{1}{\epsilon}\sum_{i=1}^{10} i x_i - \sum_{i=1}^{10} \ln(1 - x_i^2)$$

(by the way, is the function self-concordant?), the starting point being $x_0 = 0$. Test the values $\epsilon \in \{1; 0.1; 0.01; 0.005\}$.
Run the method until the inequality $\lambda(f, x_t) \leq 10^{-6}$ is satisfied and look at the pattern of values of $\lambda(f, \cdot)$ along the trajectory.
Could you explain why the number of steps of the method increases as ϵ decreases?

Lecture 5

The Conjugate Gradient Method

We come to a new family of methods for unconstrained minimization problem

$$f(x) \to \min \mid \; x \in \mathbf{R}^n \tag{5.0.1}$$

– the *conjugate direction methods*. The idea is to accelerate the rate of convergence of the Gradient Descent, avoiding at the same time explicit usage of second-order information which makes the Newton-type methods computationally expensive.

The idea behind the conjugate direction methods comes from efficient iterative technique for minimizing quadratic function (same as the idea of the Newton-type methods comes from direct Linear Algebra technique for this latter problem), so that general conjugate direction methods are natural extensions of those for quadratic minimization. And it is natural to start with conjugate directions for quadratic minimization.

5.1 Quadratic forms, Linear systems and Conjugate directions

Let H be a $n \times n$ positive definite symmetric matrix, and let

$$f(x) = \frac{1}{2} x^T A x - b^T x$$

being the quadratic form for which H is the Hessian matrix. Since H is positive definite, f is strongly convex (Lecture 3); the unique global minimizer x^* of f is the solution to the Fermat equation

$$\nabla f(x) \equiv Hx - b = 0; \tag{5.1.1}$$

due to strong convexity, this is the only critical point of f. Thus, it is the same – to minimize a strongly convex quadratic form f on $\mathbf{R}^n$ and to solve a linear $n \times n$ system of equations

$$Hx = b \tag{5.1.2}$$

with symmetric positive definite matrix $H = \nabla^2 f$.

In our previous lecture we spoke about direct Linear Algebra technique for solving (5.1.2) – namely, about the one based on Cholesky decomposition of H. There are other direct Linear Algebra methods to solve this problem – Gauss elimination, etc. When saying that these methods are "direct", I mean that they work with H "as a whole", and until the matrix is processed, no approximate minimizers of f (i.e., approximate solutions to (5.1.2)) are generated; and after

H is processed, we get the exact minimizer (in actual computations – exact up to influence of rounding errors). In contrast to this, the so called *iterative* methods for minimizing f ($\equiv$ for solving (5.1.2)) generate a sequence of approximate minimizers converging to the exact one. We already know a method of this type – the Steepest Descent; this method is aimed to minimize nonquadratic functions, but one can apply it to a quadratic objective as well.

Today we shall study another family of iterative methods – those based on *conjugate directions*; in contrast to Steepest Descent, such a method minimizes quadratic form *exactly* in no more than n steps, n being the dimension of the design vector (i.e., the size of linear system (5.1.2)).

The methods in question are based on important notion of *conjugate directions*.

5.1.1 Conjugate directions

Definition 5.1.1 [Conjugate directions] *Let H be a symmetric $n \times n$ positive definite matrix. We say that two vectors $d_1, d_2 \in \mathbf{R}^n$ are conjugate with respect to H (synonyms: H-conjugate, H-orthogonal), if*

$$d_1^T H d_2 = 0.$$

A set of vectors $d_1, ..., d_k$ is called H-orthogonal if $d_i H d_j = 0$ whenever $i \neq j$.

The meaning of the notion is clear: a symmetric positive definite matrix H defines a new inner product

$$\langle x, y \rangle = x^T H y$$

on $\mathbf{R}^n$; symmetry and positive definiteness of H imply that this indeed is an inner product[1]. In other words, H equips $\mathbf{R}^n$ with new Euclidean structure, and vectors are H-conjugate if and only if they are orthogonal in this Euclidean structure.

Given a linear combination

$$d = \sum_{i=1}^{k} \alpha_i d_i$$

of orthogonal, in the usual sense, nonzero vectors d_i, one can easily restore the coefficients via the value of the sum:

$$\alpha_i = \frac{d^T d_i}{d_i^T d_i}.$$

Similar fact takes place for linear combinations of H-orthogonal nonzero vectors:

Lemma 5.1.1 *Let H be symmetric positive definite matrix, and $d_0, ..., d_k$ be nonzero H-orthogonal vectors. Then the coefficients α_i in a linear combination*

$$d = \sum_{i=0}^{k} \alpha_i d_i \tag{5.1.3}$$

can be restored via d by the explicit formulae

$$\alpha_i = \frac{d^T H d_i}{d_i^T H d_i}, \quad i = 0, ..., k \tag{5.1.4}$$

(the denominators are nonzero due to positive definiteness of H; recall that d_i are nonzero by assumption).

[1] i.e., function of pair of vectors from $\mathbf{R}^n$ which is linear with respect to each argument, is symmetric – remains unchanged when the arguments are swapped, and is positive definite: $\langle x, x \rangle > 0$ whenever $x \neq 0$

Proof is immediate: taking the usual inner product of both sides on (5.1.3) with Hd_j, we get

$$d^T H d_j = \alpha_j d_j^T H d_j$$

(due to H-orthogonality of d_i with $i \neq j$ to d_j, the right hand side terms associated with $i \neq j$ vanish), and we come to (5.1.4).

In fact relations (5.1.4) are nothing but the standard formulae for Fourier coefficients of a vector in orthogonal (in the Euclidean structure $\langle u, v\rangle = u^T H v$) basis $d_0, ..., d_{n-1}$:

$$x = \sum_{i=0}^{n-1} \frac{\langle x, d_i\rangle}{\langle d_i, d_i\rangle} d_i \qquad \blacksquare$$

Corollary 5.1.1 *If $d_0, ..., d_k$ are H-orthogonal, H being a symmetric positive definite matrix, then the set of vectors $d_0, ..., d_k$ is linearly independent.*

Indeed, assuming that certain linear combination $\sum_{i=0}^{k} \alpha_i d_i$ of $d_0, ..., d_k$ is zero, we conclude from (5.1.4) that all the coefficients are zero. Thus, only trivial (with all coefficients being 0) combination of d_i can be equal to 0, and this is exactly linear independence of the vectors.

Now comes the first critical point of the Lecture: explicit formulae for the solution to (5.1.2) in terms of a complete (with n elements) system of H-orthogonal nonzero directions $d_0, ..., d_{n-1}$:

Proposition 5.1.1 [Solution to (5.1.2) via Conjugate Directions]
Let H be a positive definite symmetric $n \times n$ matrix, and let $d_0, ..., d_{n-1}$ be a system of n nonzero H-orthogonal vectors. Then the solution x^* to the system

$$Hx = b$$

is given by the formula

$$x^* = \sum_{i=0}^{n-1} \frac{b^T d_i}{d_i^T H d_i} d_i. \tag{5.1.5}$$

Proof. Since $d_0, ..., d_{n-1}$ are nonzero and H-orthogonal, they are linearly independent (Corollary 5.1.1); and since this is a system of n linearly independent vectors in $\mathbf{R}^n$, this is a basis in $\mathbf{R}^n$, so that the solution x^* to our system is a linear combination

$$x^* = \sum_{i=1}^{n-1} \alpha_i d_i$$

of the vectors d_i. According to Lemma 5.1.1, the coefficients of the combination are

$$\alpha_i = \frac{(x^*)^T H d_i}{d_i^T H d_i} = \frac{(Hx^*)^T d_i}{d_i^T H d_i} =$$

[since $Hx^* = b$]

$$= \frac{b^T d_i}{d_i^T H d_i}. \qquad \blacksquare$$

Remark 5.1.1 An equivalent reformulation of Proposition is that

$$x^* \equiv H^{-1}b = \sum_{i=0}^{n-1} \frac{b^T d_i}{d_i^T H d_i} d_i \equiv \left[\sum_{i=1}^{n-1} \frac{1}{d_i^T H d_i} d_i d_i^T \right] b;$$

since this equality is valid for every b, the matrix in the brackets is exactly H^{-1}:

$$H^{-1} = \sum_{i=0}^{n-1} \frac{1}{d_i^T H d_i} d_i d_i^T, \tag{5.1.6}$$

and we end up with a formula which represents the inverse to a positive definite symmetric matrix H as a sum of rank 1 matrices coming from a complete (comprised of n nonzero vectors) system of H-orthogonal vectors.

5.2 Method of Conjugate Directions: quadratic case

It is convenient for us to reformulate the statement of the Proposition as an assertion about certain iterative algorithm:

Theorem 5.2.1 [Conjugate Direction Theorem] *Let H be a positive definite symmetric $n \times n$ matrix, b be a vector and*

$$f(x) = \frac{1}{2} x^T H x - b^T x$$

be the quadratic form associated with H and b.

Let, further, $d_0, ..., d_{n-1}$ be a system of nonzero H-orthogonal vectors, and let x_0 be an arbitrary starting point. The iterative process

$$x_{t+1} = x_t + \gamma_{t+1} d_t, \quad \gamma_{t+1} = -\frac{d_t^T g_t}{d_t^T H d_t}, \ \ t = 0, ..., n-1, \tag{5.2.1}$$

where g_t is the gradient of f at x_t:

$$g_t = \nabla f(x_t) = Hx - b, \tag{5.2.2}$$

converges to the unique minimizer x^* of f ($\equiv$ the unique solution to the linear system $Hx = b$) in n steps: $x_n = x^*$.

We see that, given a complete (with n elements) system $d_0, ..., d_{n-1}$ of nonzero H-orthogonal vectors, we can associate with it recurrence (5.2.1) – the *Conjugate Direction method* – and find the exact minimizer of f in n steps.

Proof of the Conjugate Direction Theorem. From (5.2.1) we have

$$x_k - x_0 = \sum_{t=0}^{k-1} \gamma_{t+1} d_t,$$

whence

$$d_k^T H(x_k - x_0) = \sum_{t=0}^{k-1} \gamma_{t+1} d_k^T H d_t = 0. \tag{5.2.3}$$

On the other hand, by (5.1.3)

$$x^* - x_0 = \sum_{t=0}^{n-1} \frac{d_t^T H(x^* - x_0)}{d_t^T H d_t} d_t =$$

[by (5.2.3)]

$$= \sum_{t=0}^{n-1} \frac{d_t^T H(x^* - x_t)}{d_t^T H d_t} d_t =$$

[since $H(x^* - x_t) = b - Hx_t = -\nabla f(x_t) = -g_t$]

$$= \sum_{t=0}^{n-1} \left[-\frac{d_t^T g_t}{d_t^T H d_t} \right] d_t =$$

[definition of γ_{t+1}, see (5.2.1)]

$$= \sum_{t=0}^{n-1} \gamma_{t+1} d_t = x_n - x_0,$$

so that $x^* = x_n$. ∎

5.3 Descent properties of Conjugate Direction method

Let, as above,

$$f(x) = \frac{1}{2} x^T H x - b^T x$$

be a strongly convex quadratic form on $\mathbf{R}^n$ (so that H is a positive definite symmetric $n \times n$ matrix), and let $d_0, ..., d_{n-1}$ be a system of n nonzero H-orthogonal vectors. Our local goal is to establish certain important (and in fact characteristic) property of the trajectory $\{x_0, ..., x_n\}$ of the Conjugate Direction method associated with $d_0, ..., d_{n-1}$. We already know that x_n is the minimizer of f on the entire space $\mathbf{R}^n$; and the indicated property is that every x_t, $0 \le t \le n$, is the minimizer of f on the affine subspace

$$M_t = x_0 + \mathcal{B}_{t-1},$$

where

$$\mathcal{B}_{t-1} = \mathrm{Lin}\{d_0, ..., d_{t-1}\}$$

is the linear span of the vectors $d_0, ..., d_{t-1}$ (here, for homogeneity, the linear span of an empty set of vectors is $\{0\}$: $\mathcal{B}_0 = \{0\}$). The affine sets $M_0, ..., M_n$ form an increasing sequence:

$$\{x_0\} = M_0 \subset M_1 \subset ... \subset M_{n-1} \subset M_n = \mathbf{R}^n \tag{5.3.1}$$

which "links" x_0 and the entire space, and we come to a very nice picture: the Conjugate Direction method generates subsequently the minimizers of f on the elements of this "chain". The announced property is given by the following

Proposition 5.3.1 *For every t, $1 \le t \le n$, the vector x_t is the minimizer of f on the affine plane*

$$x_0 + \mathcal{B}_{t-1},$$

where

$$\mathcal{B}_{t-1} = \mathrm{Lin}\{d_0, ..., d_{t-1}\}$$

is the linear span of $d_0, ..., d_{t-1}$. *In particular,* x_t *minimizes* f *on the line*

$$l_t = \{x_{t-1} + \gamma d_{t-1} \mid \gamma \in \mathbf{R}\}.$$

Proof. Of course, "in particular" part of the statement follows from its "general" part: by construction, $l_t \subset x_0 + \mathcal{B}_{t-1}$ and $x_t \in l_t$ (see (5.2.1)); knowing that x_t minimizes f on $x_0 + \mathcal{B}_{t-1}$, we could immediately conclude that it minimizes f on $l_t \subset x_0 + \mathcal{B}_{t-1}$ as well.

To prove that x_t minimizes f on $x_0 + \mathcal{B}_{t-1}$, it suffices to prove that the gradient g_t of f at x_t is orthogonal to $\mathcal{B}_{t-1}$, i.e., is orthogonal to $d_0, ..., d_{t-1}$ [2].

According to the Conjugate Direction Theorem, we have $x_n = x^* \equiv H^{-1}b$, whence

$$x_t - x^* = x_t - x_n = -\sum_{i=t}^{n-1} \gamma_{i+1} d_i,$$

whence

$$g_t = Hx_t - b = Hx_t - Hx^* = H(x_t - x^*) = -\sum_{i=t}^{n-1} \gamma_{i+1}[Hd_i].$$

Since $d_0, ..., d_{n-1}$ are H-orthogonal, each vector Hd_i, $i \geq t$, is orthogonal to all vectors d_j, $0 \leq j < t$, and, consequently, is orthogonal to $\mathcal{B}_{t-1}$; as we just have seen, g_t is a linear combination of Hd_i, $i = t, ..., n-1$, so that g_t also is orthogonal to $\mathcal{B}_{t-1}$. ∎

5.4 Conjugate Gradient method: quadratic case

To the moment, we have associated to a n-element system of nonzero H-orthogonal vectors $d_0, ..., d_{n-1}$, H being a symmetric positive definite $n \times n$ matrix, the Conjugate Direction method (5.2.1) for minimizing quadratic forms

$$f(x) = \frac{1}{2}x^T Hx - b^T x; \tag{5.4.1}$$

what we know is that the iterates x_t of the method minimize f on the sequence of expanding affine sets (5.3.1), and the n-th iterate is the global minimizer of f. All this is fine, but where to get the underlying complete system of H-orthogonal vectors d_i?

As we remember, H-orthogonality is nothing but the standard orthogonality, but taken with respect to the inner product $\langle u, v\rangle = u^T Hv$ instead of the standard inner product $u^T v$; consequently, we can get a complete system of H-orthogonal vectors from any sequence of n linearly independent vectors by applying to the latter system the Gram-Schmidt orthogonalization process (of course, in the inner product $\langle u, v\rangle$, not in the standard one!) [those who do not remember

[2] indeed, f is convex, so that a necessary and sufficient condition for f to attain its minimum over an affine set $M = x_0 + L$, L being a linear subspace in $\mathbf{R}^n$, at certain point $\bar{x} \in M$, is the orthogonality of $\nabla f(\bar{x})$ to L. To see it, represent M as an image of $\mathbf{R}^k$, $k = \dim L$, under affine mapping $u \mapsto \bar{x} + Au$ and set $\phi(u) = f(\bar{x} + Au)$; ϕ is convex, since f is, and ϕ attains its minimum over $\mathbf{R}^k$ at $u = 0$ if and only if f attains its minimum on M at $\bar{x}$. Now, the necessary and sufficient condition for ϕ to attain its minimum at $u = 0$ is the Fermat condition $\nabla\phi(0) = 0$. Since $\nabla\phi(0) = A^T \nabla f(\bar{x})$, we come to the condition $A^T \nabla f(\bar{x}) = 0$; and since, by the standard Linear Algebra facts, $\mathrm{Ker}\, A^T$ is the orthogonal complement to $\mathrm{Im}\, A$ and by construction $\mathrm{Im}\, A = L$, relation $A^T \nabla f(\bar{x}) = 0$ is equivalent to the orthogonality of $\nabla f(\bar{x})$ and L.

what does it mean Gram-Schmidt orthogonalization, may ignore the above sentence, same as the next one, without any harm: we will not exploit this process in its general form]. It turns out, anyhow, that there exists certain particular sequence extremely convenient for the process; this is the *Krylov sequence*

$$g_0 \equiv Hx_0 - b, Hg_0, H^2g_0, H^3g_0, ...$$

The Conjugate Direction method based on directions given by $\langle \cdot, \cdot \rangle$-orthogonalization of the indicated sequence has a special name – the *Conjugate Gradient method* – and possesses extremely attractive optimality properties. This is the method we come to.

5.4.1 Description of the method

The Conjugate Gradient method (CG for short) for minimizing strongly convex quadratic form (5.4.1) is as follows:

Algorithm 5.4.1 [Conjugate Gradient method]
Initialization: *choose arbitrary starting point* x_0 *and set*

$$d_0 = -g_0 \equiv -\nabla f(x_0) = b - Hx_0;$$

set $t = 1$.
Step t: *if* $g_{t-1} \equiv \nabla f(x_{t-1}) = 0$, *terminate,* x_{t-1} *being the result. Otherwise set*
[new iterate]

$$x_t = x_{t-1} + \gamma_t d_{t-1}, \quad \gamma_t = -\frac{g_{t-1}^T d_{t-1}}{d_{t-1}^T H d_{t-1}}, \tag{5.4.2}$$

[new gradient]

$$g_t = \nabla f(x_t) \equiv Hx_t - b, \tag{5.4.3}$$

[new direction]

$$d_t = -g_t + \beta_t d_{t-1}, \quad \beta_t = \frac{g_t^T H d_{t-1}}{d_{t-1}^T H d_{t-1}}, \tag{5.4.4}$$

replace t *with* $t + 1$ *and loop.*

We are about to prove that the presented algorithm is a Conjugate Direction method:

Theorem 5.4.1 [Conjugate Gradient Theorem]
The Conjugate Gradient method is a Conjugate Direction method: if the algorithm does not terminate at step t, *then*

(i_t) *The gradients* $g_0, ..., g_{t-1}$ *of* f *at the points* $x_0, ..., x_{t-1}$ *are nonzero and*

$$\mathrm{Lin}\{g_0, g_1, ..., g_{t-1}\} = \mathrm{Lin}\left\{g_0, Hg_0, ..., H^{t-1}g_0\right\}; \tag{5.4.5}$$

(ii_t) *The directions* $d_0, ..., d_{t-1}$ *are nonzero and*

$$\mathrm{Lin}\{d_0, ..., d_{t-1}\} = \mathrm{Lin}\left\{g_0, Hg_0, ..., H^{t-1}g_0\right\}; \tag{5.4.6}$$

(iii_t) *The directions* $d_0, ..., d_{t-1}$ *are* H*-orthogonal:*

$$d_i^T H d_j = 0, \ \ 0 \le i < j \le t-1; \tag{5.4.7}$$

(iv) *One has*

$$\gamma_t = \frac{g_{t-1}^T g_{t-1}}{d_{t-1}^T H d_{t-1}}; \tag{5.4.8}$$

(v) *One has*

$$\beta_t = \frac{g_t^T g_t}{g_{t-1}^T g_{t-1}}. \tag{5.4.9}$$

In particular, the algorithm terminates no later than after n steps with the result being the exact minimizer of f.

Proof.
1^0. We first prove (i)-(iii) by induction on t.

Base $t = 1$. We should prove that if the algorithm does not terminate at the first step (i.e., if $g_0 \neq 0$), then (i_1) - (iii_1) are valid. This is evident, since, by the description of the method, $g_0 = d_0$, and there is nothing to prove in (iii_1).

Step $t \mapsto t+1$. Assume that (i_s) - (iii_s) are valid for $s \leq t$ and that the algorithm does not terminate at the step $t+1$, i.e., that $g_t \neq 0$, and let us derive from this assumption (i_{t+1}) - (iii_{t+1}).

First of all, consider the directions $d_0, ..., d_{t-1}$. Since (ii_s) and (iii_s) are valid for $s \leq t$, this is a sequence of nonzero H-orthogonal directions, and we clearly can extend it, by adding appropriate directions $\bar{d}_t, ..., \bar{d}_{n-1}$, to a complete system of H-orthogonal directions[3]. Now let us look at the Conjugate Direction method associated with this extended system of directions. Comparing (5.4.2) and (5.2.1), we see that the first $t+1$ points of the trajectory of this latter method are the same as the first $t+1$ points $x_0, ..., x_t$ of the Conjugate Gradient algorithm in question. In particular, by Proposition 5.3.1,

$$g_s^T d_l = 0, \;\; 0 \leq l < s \leq t \tag{5.4.10}$$

– the gradient of f at a point x_s, $s \leq t$, is orthogonal to the linear spans of the directions $d_0, ..., d_{s-1}$. And since the linear span of $d_0, ..., d_{s-1}$ is, by (ii_s), the same as the linear span of $g_0, ..., g_{s-1}$, we conclude that the gradients $g_0, ..., g_t$ are mutually orthogonal:

$$g_s^T g_l = 0, \;\; 0 \leq l < s \leq t. \tag{5.4.11}$$

We have from (5.4.2)

$$g_t = Hx_t - b = H(x_{t-1} + \gamma_t d_{t-1}) - b = [Hx_{t-1} - b] + \gamma_t H d_{t-1} = g_{t-1} + \gamma_t H d_{t-1}.$$

By (i_{t-1}) and (ii_{t-1}), both g_{t-1} and d_{t-1} belong to $\mathrm{Lin}\{g_0, Hg_0, ..., H^{t-1}g_0\}$, so that the above computation demonstrates that $g_t \in \mathrm{Lin}\{g_0, Hg_0, ..., H^t g_0\}$, which combined with ($\mathrm{i}_t$) means that

$$\mathrm{Lin}\{g_0, ..., g_t\} \subset \mathrm{Lin}\{g_0, Hg_0, ..., H^t g_0\}.$$

Since $g_0, ..., g_t$ are nonzero and mutually orthogonal (see (5.4.11)), the left hand side subspace in the latter inclusion is of dimension $t+1$, while the right hand side subspace is of dimension at most $t+1$; a $t+1$-dimensional linear subspace can be enclosed into a linear subspace of dimension $\leq t+1$ only if the subspaces are equal, and we come to (i_{t+1}).

[3] same as any system of orthogonal nonzero vectors can be extended to an orthogonal basis; recall that H-orthogonality is nothing but the orthogonality with respect to certain Euclidean structure!

To prove (ii_{t+1}), note that by (5.4.4)

$$d_t = -g_t + \beta_t d_{t-1};$$

both right hand side vectors are from $\mathrm{Lin}\{g_0, Hg_0, ..., H^t g_0\}$ (by (ii_t) and already proved (i_{t+1})), so that $d_t \in \mathrm{Lin}\{g_0, Hg_0, ..., H^T g_0\}$; combining this observation with (ii_t), we come to

$$\mathrm{Lin}\{d_0, ..., d_t\} \subset \mathrm{Lin}\{g_0, Hg_0, ..., H^T g_0\}. \tag{5.4.12}$$

Besides this, $d_t \neq 0$ (indeed, g_t is nonzero and is orthogonal to d_{t-1} by (5.4.10)). Now let us prove that d_t is H-orthogonal to $d_0, ..., d_{t-1}$. From formula for d_t we have

$$d_t^T H d_s = -g_t^T H d_s + \beta_t d_{t-1}^T H d_s. \tag{5.4.13}$$

When $s = t-1$, the right hand side is 0 by definition of β_t; and when $s < t-1$, both terms in the right hand side are zero. Indeed, the first term is zero due to (ii_t): this relation implies that

$$Hd_s \in \mathrm{Lin}\{Hg_0, H^2 g_0, ..., H^s g_0\},$$

and the right hand side subspace is, due to (i_{s+1}) ($s < t-1$, so that we can use (i_{s+1})!), contained in the linear span of the gradients $g_0, ..., g_{s+1}$, and g_t is orthogonal to all these gradients by virtue of (5.4.12) (recall that $s < t-1$). The second term in the right hand side of (5.4.13) vanishes because of (iii_t).

Thus, the right hand side in (5.4.13) is zero for all $s < t$; in other words, we have proved (iii_{t+1}) – the vectors $d_0, ..., d_t$ indeed are H-orthogonal. We already know that $d_t \neq 0$; consequently, in view of (ii_t), all $d_0, ..., d_t$ are nonzero. Since, as we already know, these vectors are H-orthogonal, they are linearly independent (Corollary 5.1.1). Consequently, inclusion in (5.4.12) is in fact equality, and we have proved (ii_{t+1}). The inductive step is completed.

2^0. To prove (iv), note that if $t > 1$ then, by (5.4.4),

$$-g_{t-1}^T d_{t-1} = g_{t-1}^T g_{t-1} - \beta_{t-1} g_{t-1}^T d_{t-2},$$

and the second term in the right hand side is 0 due to (5.4.11); thus, $-g_{t-1}^T d_{t-1} = g_{t-1}^T g_{t-1}$ for $t > 1$. For $t = 1$ this relation also is valid (since, by the initialization rule, $d_0 = -g_0$). Substituting equality $-g_{t-1}^T d_{t-1} = g_{t-1}^T g_{t-1}$ into formula for γ_t from (5.4.2), we get (5.4.8).

To prove (v), note that $g_t^T g_{t-1} = 0$ (see (5.4.12). Besides this, from (5.4.2) we have

$$Hd_{t-1} = \frac{1}{\gamma_t}[g_t - g_{t-1}]$$

(note that $\gamma_t > 0$ due to (iv) and (i_t), so that we indeed can rewrite (5.4.2) in the desired way); taking inner product with g_t, we get

$$g_t^T H d_{t-1} = \frac{1}{\gamma_t} g_t^T g_t = \frac{d_{t-1}^T H d_{t-1}}{g_{t-1}^T g_{t-1}} g_t^T g_t$$

(we have used (5.4.8)); substituting the result into (5.4.4), we come to (5.4.9).

3^0. It remains to prove the "in particular" part of the statement – i.e., that the method terminates with exact minimizer of f in no more than n steps. This is immediate: as we know from the proof of (i)-(iii), if the method does not terminate at step t, the points $x_0, ..., x_t$ are the first $t+1$ points of the trajectory of certain Conjugate Direction method as applied to f. It follows that $t \leq n$ – since in n steps Conjugate Directions method for sure finds the exact solution and comes therefore to $g_n = 0$. ∎

CG and Three-Diagonal representation of a Symmetric matrix

This subsection is non-obligatory Assume that we are minimizing (5.4.1) by the Conjugate Gradient algorithm; for the sake of simplicity, assume also that the method terminates in exactly n steps, so that in course of its run we obtain n nonzero gradients $g_0, ..., g_{n-1}$. As we know from the proof of the Conjugate Gradient Theorem (see (5.4.12)), these gradients are mutually orthogonal, so that after normalization

$$f_i = |g_i|^{-1} g_i$$

of g_i we get an orthonormal basis in $\mathbf{R}^n$. How does the matrix H look in this basis? The answer is very impressive:

the matrix in the basis $\{f_i\}$ is 3-diagonal: $f_i^T H f_j = 0$ whenever $|i-j| > 1$.

The proof is immediate: assume, e.g., that $i > j+1$ and let us prove that $f_i^T H f_j = 0$. As we know from Theorem 5.4.1.(i), $f_j \in \mathrm{Lin}\{g_0, Hg_0, ..., H^j g_0\}$, whence $Hf_j \in \mathrm{Lin}\{Hg_0, ..., H^{j+1} g_0\}$, and the latter subspace is contained in $\mathrm{Lin}\{g_0, ..., g_{j+1}\}$ (the same Theorem 5.4.1.(i)). Since, as it was already mentioned, g_i is orthogonal to $g_0, ..., g_{i-1}$ and $j+1 < i$, we have $f_i^T H f_j = 0$, as claimed.

The necessity to find an orthonormal basis where a given symmetric matrix becomes 3-diagonal occurs in many applications, e.g., when it is necessary to find the spectrum (all eigenvalues) of a large and sparse (with close to 0 percentage of nonzero entries) symmetric matrix. The Conjugate Gradient algorithm is certain "starting point" in developing tools for finding such a basis[4].

5.4.2 Rate of convergence of the Conjugate Gradient method

According to Theorem 5.4.1 and Proposition 5.3.1, if CG as applied to (5.4.1) does not terminate during first t steps, x_t is the minimizer of f on the affine plane

$$M_t = x_0 + \mathrm{Lin}\{d_0, ..., d_{t-1}\} = x_0 + \mathcal{E}_t,$$

where, according to (5.4.6),

$$\mathcal{E}_t = \mathrm{Lin}\{g_0, Hg_0, ..., H^{t-1} g_0\}, \quad g_0 = Hx_0 - b. \tag{5.4.14}$$

Equivalent description of $\mathcal{E}_t$ is that this is the space comprised of all vectors of the form $p(H)g_0$, p being a polynomial of degree $\le t-1$:

$$\mathcal{E}_t = \{p(H)g_0 \mid p(z) = \sum_{i=0}^{t-1} p_i z^i\}. \tag{5.4.15}$$

Given these observations, we immediately can establish the following

Proposition 5.4.1 *Let*

$$E(x) = f(x) - \min f$$

[4] please do not think that the problem in question can be solved by straightforward application of the CG: the influence of rounding errors makes the actually computed gradients very far from being mutually orthogonal!

be the residual in terms of the objective associated with quadratic form (5.4.1), and let x_t be t-th iterate of the Conjugate Gradient method (if the method terminates in course of first t steps, then, by definition, x_t is the result of the method, i.e., the exact minimizer of f). Then

$$E(x_t) = \min_{p\in\mathcal{P}_{t-1}} \frac{1}{2}(x_0 - x^*)^T H[I - Hp(H)]^2(x_0 - x^*), \tag{5.4.16}$$

where x_0 is the starting point, x^* is the exact minimizer of f, I is the unit matrix, and $\mathcal{P}_k$ is the family of all polynomials of degree $\le k$.

Proof is immediate: since f is a quadratic form, we have

$$f(x) = f(x^*) + (x - x^*)^T\nabla f(x^*) + \frac{1}{2}(x - x^*)^T[\nabla^2 f](x - x^*) = f(x^*) + \frac{1}{2}(x - x^*)^T H(x - x^*)$$

(we have taken into account that $\nabla f(x^*) = 0$ and $\nabla^2 f = H$), whence

$$E(x) = \frac{1}{2}(x - x^*)H(x - x^*). \tag{5.4.17}$$

Substituting

$$x = x_0 + p(H)g_0 = x_0 + p(H)(Hx_0 - b) = x_0 + p(H)H(x_0 - x^*),$$

we get $x - x^* = -[I - Hp(H)](x_0 - x^*)$, whence

$$E(x_0 + p(H)g_0) = \frac{1}{2}(x_0 - x^*)^T H[1 - Hp(H)]^2(x - x^*).$$

When p runs through the family $\mathcal{P}_{t-1}$, the point $x_0 + p(H)g_0$, as it already was explained, runs through the affine plane $M_t = x_0 + \mathcal{E}_t$; x_t, as we know, is the minimizer of f (and, consequently, E) on this plane, and (5.4.16) follows. ∎

What we are interested in, is the following corollary of Proposition 5.4.1:

Corollary 5.4.1 *Let Σ be the spectrum (the set of distinct eigenvalues) of H, and let x_t be t-th point of the trajectory of CG as applied to f. Then*

$$E(x_t) \le E(x_0) \min_{q\in\mathcal{P}_t^*} \max_{\lambda\in\Sigma} q^2(\lambda), \tag{5.4.18}$$

where $\mathcal{P}_t^*$ is the set of all polynomials $q(z)$ of degree $\le t$ equal 1 at $z = 0$.

Besides this,

$$E(x_t) \le \left[\frac{1}{2}|x_0 - x^*|^2\right] \min_{q\in\mathcal{P}_t^*} \max_{\lambda\in\Sigma} \lambda q^2(\lambda). \tag{5.4.19}$$

Proof. Let $e_1, ..., e_n$ be an orthonormal basis comprised of eigenvectors of H, and let $\lambda_1, ..., \lambda_n$ be the corresponding eigenvalues. Let

$$x_0 - x^* = \sum_{i=1}^n s_i e_i$$

be the expansion of $x_0 - x^*$ in the indicated basis. Then, for any polynomial r,

$$r(H)(x_0 - x^*) = \sum_{i=1}^n r(\lambda_i) s_i e_i;$$

applying this identity to the polynomial $r(z) = z(1 - zp(z))^2$, $p \in \mathcal{P}_{t-1}$, we get

$$(x_0 - x^*)^2 H[1 - Hp(H)]^2 (x_0 - x^*) = \left[\sum_{i=1}^n s_i e_i\right]^T \left[\sum_{i=1}^n \lambda_i (1 - \lambda_i p(\lambda_i))^2 s_i e_i\right] =$$

$$= \sum_{i=1}^n (1 - \lambda_i p(\lambda_i))^2 \lambda_i s_i^2. \tag{5.4.20}$$

The resulting quantity clearly can be bounded from above by

$$S \equiv \left[\max_{\lambda\in\Sigma}(1 - \lambda p(\lambda))^2\right] \sum_{i=1}^n \lambda_i s_i^2 = \left[\max_{\lambda\in\Sigma}(1 - \lambda p(\lambda))^2\right] \left[(x_0 - x^*)^T H (x_0 - x^*)\right] =$$

[see (5.4.17)].

$$= 2\left[\max_{\lambda\in\Sigma}(1 - \lambda p(\lambda))^2\right] E(x_0).$$

Combining the resulting inequality with (5.4.16), we come to

$$E(x_t) \le E(x_0) \min_{p\in\mathcal{P}_{t-1}} \max_{\lambda\in\Sigma}(1 - \lambda p(\lambda)^2.$$

When p runs through $\mathcal{P}_{t-1}$, the polynomial $q(z) = 1 - zp(z)$ clearly runs through the entier $\mathcal{P}_t^*$, and (5.4.18) follows.

Relation (5.4.19) is proved similarly; the only difference is that instead of bounding from above the right hand side of (5.4.20) by the quantity S, we bound the expression by

$$\left[\max_{\lambda\in\Sigma} \lambda(1 - \lambda p(\lambda))^2\right] \sum_{i=1}^n s_i^2,$$

the second factor in the bound being $|x_0 - x^*|^2$. ∎

Corollary 5.4.1 provides us with a lot of information on the rate of convergence of the Conjugate Gradient method:

- A. Rate of convergence in terms of Condition number It can be proved that for any segment $\Delta = [l, L]$, $0 < l < L < \infty$, and for any positive integer s there exists a polynomial $q_s \in \mathcal{P}_s^*$ with

  $$\max_{\lambda\in\Delta} q_s^2(\lambda) \le 4\left[\frac{\sqrt{Q} - 1}{\sqrt{Q} + 1}\right]^{2s}, \quad Q = \frac{L}{l}.$$

  Combining (5.4.18) and the just indicated result where one should substitute

  $$l = \lambda_{\min}(H), L = \lambda_{\max}(H),$$

  we get the following *non-asymptotical* efficiency estimate for CG as applied to (5.4.1):

  $$E(x_N) \le 4\left[\frac{\sqrt{Q_H} - 1}{\sqrt{Q_H} + 1}\right]^{2N} E(x_0), \quad N = 1, 2, ... \tag{5.4.21}$$

  where

  $$Q_H = \frac{\lambda_{\max}(H)}{\lambda_{\min}(H)}$$

is the condition number of the matrix H.

We came to a *nonasymptotical linear rate of convergence* with the convergence ratio

$$\left[\frac{\sqrt{Q_H}-1}{\sqrt{Q_H}+1}\right]^2;$$

for large Q_h, this ratio is of the form $1-O(1)\sqrt{Q_H}$, so that the number of steps required to improve initial inaccuracy by a given factor ϵ is, *independently of the value of* n, bounded from above by $O(1)\sqrt{Q_H}\ln(1/\epsilon)$. The number of required steps is proportional to the *square root* of the condition number of the Hessian, while for the Steepest Descent in the quadratic case similar quantity is proportional to the condition number itself (see Lecture 3); this indeed is a great difference!

- B. Rate of convergence in terms of $|x_0 - x^*|$ It can be proved also that for any $L > 0$ and any integer $s > 0$ there exists a polynomial $r_s \in \mathcal{P}_s^*$ such that

$$\max_{0\le\lambda\le L} \lambda r^2(\lambda) \le \frac{L}{(2s+1)^2}.$$

Since Σ for sure is contained in the segment $[0, L = |H|]$, $|H|$ being the norm of matrix H, we can use just indicated result and (5.4.19) to get nonasymptotical (and independent on the condition number of H) sublinear efficiency estimate

$$E(x_N) \le \frac{|H||x_0 - x^*|^2}{2}\frac{1}{(2N+1)^2}, \quad N = 1, 2, ... \tag{5.4.22}$$

This result resembles sublinear global efficiency estimate for Gradient Descent as applied to a convex $C^{1,1}$ function (Lecture 3, Proposition 3.3.2); note that a convex quadratic form (5.4.1) indeed is a $C^{1,1}$ function with Lipschitz constant of gradient equal to $|H|$. As compared to the indicated result about Gradient Descent, where the convergence was with the rate $O(1/N)$, the convergence given by (5.4.22) is twice better in order – $O(1/N^2)$.

- C. Rate of convergence in terms of the spectrum of H The above results established rate of convergence of CG in terms of bounds – both lower and upper or only the upper one – on the eigenvalues of H. If we take into account details of the distribution of the eigenvalues, more detailed information on convergence rate can be obtained. Let

$$\lambda_{\max}(H) \equiv \lambda_1 > \lambda_2 > ... > \lambda_m \equiv \lambda_{\min}(H)$$

be *distinct* eigenvalues of H written down in descent order. For every $k \le s$ let

$$\pi_k(z) = \prod_{i=1}^{k}(1 - z/\lambda_i) \in \mathcal{P}_k^*,$$

and let $q_{s,k} \in \mathcal{P}_s^*$ be the polynomial such that

$$\max_{\lambda_{\min}(H)\le\lambda\le\lambda_{k+1}} q_{s,k}^2(\lambda) \le 4\left[\frac{\sqrt{Q_{k,H}}-1}{\sqrt{Q_{k,H}+1}}\right]^{2s}, \quad Q_{k,H} = \frac{\lambda_{k+1}}{\lambda_{\min}(H)}$$

(existence of such a polynomial is mentioned in item A). It is clear that

$$\max_{\lambda\in\Sigma}[\pi_k(\lambda)q_{s,k}(\lambda)]^2 \le 4\left[\frac{\sqrt{Q_{k,H}}-1}{\sqrt{Q_{k,H}}+1}\right]^{2s}$$

(indeed, π_k vanishes on the spectrum of H to the right of λ_{k+1} and is in absolute value ≤ 1 between zero and λ_{k+1}, while $q_{s,k}$ satisfies the required bound to the left of λ_{k+1} in Σ). Besides this, $\pi_k q_{s,k} \in \mathcal{P}^*_{k+s}$. Consequently, by (5.4.18)

$$E(x_N) \le 4 \min_{1\le s\le N}\left\{\left[\frac{\sqrt{Q_{N-s,H}}-1}{\sqrt{Q_{N-s,H}}+1}\right]^{2s}\right\} E(x_0)$$

– this is an extension of the estimate (5.4.21) (this latter estimate corresponds to the case when we eliminate the outer min and set $s = N$ in the inner brackets, which results in $Q_{0,H} = Q_H$). We see also that if $N \ge m$, m being the number of distinct eigenvalues in H, then $E(x_N) = 0$ (set $q = \pi_m$ in (5.4.18)); thus, in fact

CG finds the exact minimizer of f in at most as many steps as many distinct eigenvalues are there in matrix H.

Taking into account the detailes of the spectrum of H, one can strengthen the estimate of item B as well.

5.4.3 Conjugate Gradient algorithm for quadratic minimization: advantages and disadvantages

Let us summarize our knowledge on the CG algorithm for (strongly convex) quadratic minimization, or, which is the same, for solving linear systems

$$Hx = b$$

with positive definite symmetric matrix H.

First of all, note that the method is simple in implementation – as simple as the Gradient Descent: a step of the method requires not more than 2 multiplications of vectors by matrix A. Indeed, literally reproducing formulae (5.4.2) - (5.4.4), you need 2 matrix-vector multiplications:

$d_{t-1} \to Hd_{t-1}$ to find γ_t,

and

$x_t \to Hx_t$ to find g_t.

In fact only the first of these matrix-vector multiplication is necessary, since g_t can be computed recursively:

$$g_t = g_{t-1} + \gamma_t H d_{t-1} \quad [\text{since } \; g_t - g_{t-1} = H(x_t - x_{t-1}) = \gamma_t H d_{t-1}].$$

Note that all remaining actions at a step are simple – taking inner products and linear combinations of vectors, and all these actions together cost $O(n)$ arithmetic operations. Thus, the arithmetic cost of a step in CG (same as that one for Steepest Descent) is

$$O(n) + [\text{cost of a single matrix-vector multiplication } d \to Hd].$$

This is a very important fact. It demonstrates that *sparsity of H* – relatively small number N ($N << n^2$) of nonzero entries – can be immediately utilized by CG. Indeed, in the "dense" case

matrix-vector multiplication costs $2n^2$ multiplications and additions, and this is the principal term in the arithmetic cost of a step of the method; in the sparse case this principal term reduces to $2N << 2n^2$.

Large-scale linear systems of equations typically have matrices H which either are extremely sparse (something $0.01\% - 1\%$ of nonzero entries), or are not sparse themselves, but are products of two sparse matrices ("implicit spasity", e.g., the least square matrices arising in Tomography); in this latter case matrix-vector multiplications are as cheap as if H itself were sparse. If the size of the matrix is large enough (tens of thousands; in Tomography people deal with sizes of order of 10^5 - 10^6) and no sparsity – explicit or "implicit" – is present, then, typically, there are no ways to solve the system. Now, if the matrix of the system is sparse and the pattern of the nonzero entries is good enough, one can solve the system by a kind of Cholesky decomposition or Gauss elimination, both the methods being modified in order to work with sparse data and not to destroy sparsity in course of their work. If the matrix is large and sparsity is not "well-structured" or is "implicit", the direct methods of Linear Algebra are unable to solve the system, and all we can do is to use an iterative method, like Steepest Descent or CG. Here we exploit the main advantage of an iterative method based on matrix-vector multiplications – cheap step and modest memory requirements.

The indicated advantages of iterative methods are shared by both Steepest Descent and Conjugate Gradient. But there is an important argument in favour of CG – its better rate of convergence. In fact, *the Conjugate Gradient algorithm possesses the best*, in certain exact sense, *rate of convergence an iterative method* (i.e., the one based on matrix-vector multiplications) *may have*.

These are the main advantages of the CG – simplicity and theoretical optimality among the iterative methods for quadratic minimization. And the main disadvantage of the method is its sensitivity to the condition number of the matrix of the system – although less than the one for Steepest Descent (see item A in the discussion above), but still rather unpleasant. Theoretically, all bad we could expect of an ill-conditioned H is that the convergence of CG will be slow, but after n steps (as always, n is the size of H) the method should magically bring us the exact solution. The influence of rounding errors makes this attractive picture absolutely unrealistic. Even with moderate condition number of H, the method will not find exact solution in n steps, but will come rather close to it; and with large condition number, the n-th approximate solution can be even worse than the initial one. Therefore when people, by some reasons, are interested to solve a moderate-size linear system by CG, they allow the method to run $2n$, $4n$ or something like steps (I am saying about "moderate size" systems, since for large-scale ones $2n$ or $4n$ steps of CG simply cannot be carried out in reasonable time).

The conclusion here should be as follows: if you are solving a linear system with symmetric positive definite matrix and the size of the system is such that direct Linear Algebra methods – like Cholesky decomposition – can be run in reasonable time, it is better to use these direct methods, since they are much more numerically stable and less sensitive to the conditioning of the system than the iterative methods. It makes sense to solve the system by CG only when the direct methods cannot be used, and in this case your chances to solve the problem heavily depend on whether you can exploit explicit or implicit sparsity of the matrix in question and especially on how well-conditioned is the matrix.

5.5 Extensions to non-quadratic problems

To the moment we in fact dealt not with unconstrained minimization, but with the Linear Algebra problem of solving a linear system with positive definite symmetric matrix. All this work in fact was aimed to develop extensions of the Conjugate Gradient algorithm on the nonquadratic case, and it is time now to come to these extensions.

The idea behind the extensions in question is as follows: the Conjugate Gradient Algorithm in its basic version 5.4.1:

(A): $g_0 = \nabla f(x_0);\ d_0 = -g_0;$

(B): $x_t = x_{t-1} + \gamma_t d_{t-1}, \quad \gamma_t = -\frac{g_{t-1}^t d_{t-1}}{d_{t-1}^T H d_{t-1}};$

(C): $g_t = \nabla f(x_t)$:

(D): $d_t = -g_t + \beta_t d_{t-1}, \quad \beta_t = \frac{g_t^T H d_{t-1}}{d_{t-1}^T H d_{t-1}};$

"almost ignores" the quadratic nature of f: the matrix H is involved only in the formulae for scalars γ_t (the stepsize) and β_t (the coefficient in the updating formulae for the search directions). If we were able to eliminate the presence of H completely and to describe the process in terms of f and ∇f only, we would get a recurrence CG* which, formally, could be applied to an arbitrary objective f, and in the case of strongly convex quadratic objective would become our basic method CG. This latter method solves quadratic problem *exactly* in n steps; since close to a nondegenerate local minimizer x^* a general smooth objective f is very similar to a strongly convex quadratic one f_q, we could hope that CG* applied to f and started close to x^* would "significantly" reduce inaccuracy in n steps. Now we could again apply to f n steps of the same routine CG*, but with the starting point given by the first n steps, again hopefully significantly reducing inaccuracy, and so on. If, besides this, we were clever enough to ensure global convergence of the indicated "cyclic" routine, we would get a globally converging method with good asymptotical behaviour.

This is the idea, and now let us look how to implement it. First of all, we should eliminate H in formulae for the method in the quadratic case. It is easy to do it with the formula for the stepsize γ_t. Indeed, we know that CG is a Conjugate Direction method and therefore x_t is the minimizer of f along the line passing through x_{t-1} in the direction d_{t-1} (Proposition 5.3.1). Thus, we may replace (B) by equivalent, in the case of strongly convex quadratic f, rule

(B*): $x_t = x_{t-1} + \gamma_t d_{t-1}, \quad \gamma_t \in \mathrm{Argmin}_{\gamma \in \mathbf{R}} f(x_{t-1} + \gamma d_{t-1});$

this new rule makes sense for a non-quadratic f as well.

It remains to eliminate H in (D). This can be done, e.g., via the identity given by Theorem 5.4.1.(v):

$$\beta_t = \frac{g_t^T g_t}{g_{t-1}^T g_{t-1}}.$$

With these substitutions, we come to the following

Algorithm 5.5.1 [Fletcher-Reeves Conjugate Gradient method for minimization of a general-type function f over $\mathbf{R}^n$]
Initialization: *choose arbitrary starting point* x_0. *Set cycle counter* $k = 1$.
Cycle k:
Initialization of the cycle: *given* x_0, *compute*

$$g_0 = \nabla f(x_0), \;\; d_0 = -g_0;$$

Inter-cycle loop: *for* $t = 1, ..., n$:
a) if $g_{t-1} = 0$, *terminate,* x_{t-1} *being the result produced by the method, otherwise set* $x_t = x_{t-1} + \gamma_t d_{t-1}$, *where* γ_t *minimizes* $f(x_{t-1} + \gamma d_{t-1})$ *over* $\gamma \in \mathbf{R}$;
b) compute $g_t = \nabla f(x_t)$;
c) set $d_t = -g_t + \beta_t d_{t-1}$, *with* $\beta_t = \frac{g_t^T g_t}{g_{t-1}^T g_{t-1}}$.
If $t < n$, *replace* t *with* $t + 1$ *and go to a)*
Restart: *replace* x_0 *with* x_n, *replace* k *with* $k + 1$ *and go to new cycle.*

The Fletcher-Reeves algorithm is not the only extension of the quadratic Conjugate Gradient algorithm onto non-quadratic case. There are many other ways to eliminate H from Algorithm 5.4.1, and each one gives rise to a non-quadratic version of CG. E.g., one can rewrite the relation

$$\beta_t = \frac{g_t^T g_t}{g_{t-1}^T g_{t-1}} \tag{5.5.1}$$

equivalently in quadratic case as

$$\beta_t = \frac{(g_t - g_{t-1})^T g_t}{g_{t-1}^T g_{t-1}} \tag{5.5.2}$$

(as we remember from the proof of Theorem 5.4.1, in the quadratic case g_t is orthogonal to g_{t-1}, so that both equations for β_t are in this case equivalent). When we replace the formula for β_t in the Fletcher-Reevs method by (5.5.2), we again obtain a method (the *Polak-Ribiere* one) for unconstrained minimization of smooth general-type functions, and this method also becomes the quadratic CG in the case of quadratic objective. It should be stressed that in the nonquadratic case the Polak-Ribiere method differs from the Fletcher-Reeves one, since relations (5.5.1) and (5.5.2) are equivalent only in the case of quadratic f.

5.5.1 Global and local convergence of Conjugate Gradient methods in non-quadratic case

Proposition 5.5.1 *[Global convergence of the Fletcher-Reeves and the Polak-Ribiere methods] Let a continuously differentiable function* $f : \mathbf{R}^n \to \mathbf{R}$ *be minimized by the Fletcher-Reeves or the Polak-Ribiere versions of the Conjugate Gradient method. Assume that the starting point* x_0 *of the first cycle is such that the level set*

$$S = \{x \mid \; f(x) \leq f(x_0)\}$$

is bounded, and let x^k *be the starting point of cycle* k *(i.e., the result of the previous cycle). Then the sequence* $\{x^k\}$ *is bounded, and all limiting points of this sequence are critical points of* f.

The proof, basically, repeats the one for the Steepest Descent, and I shall only sketch it. The first observation is that the objective never increases along the trajectory, since all steps of the method are based on precise line search. In particular, the trajectory never leaves the compact set S.

Now, the crucial observation is that the first step of cycle k is the usual Steepest Descent step from x^k and therefore it "significantly" decreases f, provided that the gradient of f at x^k is not very small [5]. Since the subsequent inter-cycle steps do not increase the objective, we conclude that the the sum of progresses in the objective values at the Steepest Descent steps starting the cycles is bounded from above (by the initial residual in terms of the objective). Consequently, these progresses tend to zero as $k \to \infty$, and due to the aforementioned relation

$$\text{small progress in the objective at a Steepest Descent step from } x^{k-1} \Rightarrow$$

$$\Rightarrow \text{small } |\nabla f(x^{k-1})|$$

we conclude that $\nabla f(x^k) \to 0$, $k \to \infty$. Thus, any limiting point of the sequence $\{x^k\}$ is a critical point of f.

The actual justification of non-quadratic extensions of the Conjugate Gradient method is the following proposition which says that the property "finite n-step convergence" of the method in the quadratic case transforms into the property of "n-step quadratic convergence" in the nonquadratic one:

Proposition 5.5.2 [Asymptotical "n-step quadratic" convergence of the Fletcher-Reeves and Polak-Ribiere methods in nonquadratic nondegenerate case]
Let $f : \mathbf{R}^n \to \mathbf{R}$ be three times continuously differentiable function, and let x^* be a nondegenerate local minimizer of f, so that $\nabla f(x^*) = 0$ and $\nabla^2 f(x^*)$ is positive definite. Assume that f is minimized by the Fletcher-Reeves or Polak-Ribiere versions of the Conjugate Gradient algorithm, and assume that the sequence $\{x^k\}$ of points starting the cycles of the algorithm converges to x^*. Then the sequence $\{x^k\}$ converges to x^* quadratically:

$$|x^{k+1} - x^*| \le C|x^k - x^*|^2$$

for certain $C < \infty$ and all k.

We should stress that the quadratic convergence indicated in the theorem is *not* the quadratic convergence of the subsequent search points generated by the method: in the Proposition we speak about "squaring the distance to x^*" in n steps of the method, not after each step, and for large n this "n-step quadratic convergence" is not that attractive.

[5] all is going on within a compact set S where f is continuously differentiable; therefore for $x \in S$ we have

$$f(x+h) \le f(x) + h^T \nabla f(x) + \epsilon(|h|)|h|, \ \ \forall (h, |h| \le 1),$$

with *independent of* $x \in S$ reminder $\epsilon(s) \to 0$, $s \to 0$. Consequently, properly chosen step from a point $x \in S$ in the anti-gradient direction indeed decreases f at least by $\psi(|\nabla f(x)|)$, with some positive on the ray $s > 0$ and nondecreasing on the ray function $\psi(s)$

Assignment # 5 (Lecture 5)

Exercise 5.5.1 *Write a code implementing the Conjugate Gradient method for minimizing strongly convex quadratic forms (i.e., for solving systems*

$$Hx = b$$

with positive definite symmetric matrices H).

Run the experiment as follows:

- *choose an integer n and generate randomly $n \times n$ matrix A;*
- *generate a $n \times n$ diagonal matrix D with positive diagonal entries d_i, $i = 1, ..., n$;*
- *compute $H = A^T DA$, thus coming to a positive definite symmetric $n \times n$ matrix H;*
- *generate randomly n-dimensional vector x^* and set $b = Hx^*$; now you have linear system*

  $$Hx = b$$

  with known in advance exact solution x^*;
- *apply to the system your CG code, starting the process at $x_0 = 0$. Run the code until the residual in the system $|Hx_k - b|$ becomes less than $10^{-6}|b|$, x_k being the current iterate of the method. If, due to influence of rounding errors, you will be unable to terminate after n steps, run more steps of the method, and compare the following two policies:*
  - *restart after every n steps, i.e., after the first n steps of the method run it again as from the very beginning, but use, as x_0, the result of the first n steps; after n steps more, again restart the method, using, as x_0, the result of the second n-step series of iterations, etc.;*
  - *no restarts at all: from the very beginning till the very end, use relations from Algorithm 5.4.1, as if you were not expecting the method to get the exact solution in n steps.*
- *during the run, display the errors $|x_k - x^*|$ and $|Hx_k - b|$.*

Choose n as you wish and test the diagonal matrices D with

$$d_i = \delta^{\frac{i-1}{n-1}}, \;\; i = 1, ..., n,$$

with δ taking values $1, 10, 100, 1000, 10^5$.
What happens when δ increases? Why?

Exercise 5.5.2 *Write codes implementing the Fletcher-Reeves and the Polak-Ribiere Conjugate Gradient methods. Apply the codes to the Rosenbrock problem*

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

the starting point being $(-1.2, 1.0)$.

Exercise 5.5.3 [Gram-Schnidt Orthogonalization]
Let H be a positive definite symmetric $n \times n$ matrix and $p_0, ..., p_{n-1}$ be a linearly independent set of n vectors in $\mathbf{R}^n$. Prove that the vectors d_i given by the Gram-Shcmidt orthogonalization process, i.e., by the recurrence

$$d_0 = p_0; d_t = p_t - \sum_{i=0}^{t-1} \frac{p_t^T H d_i}{d_i^T H d_i} d_i,\ 1 \leq t < n,$$

form an H-orthogonal system of nonzero vectors.

Exercise 5.5.4 [Non-obligatory] *Consider the Tschebyshev polynomial given on the segment $[-1, 1]$ by the formula*

$$T_n(x) = \cos(n \operatorname{arc\ cos} x).$$

1) Prove that T_n indeed is a polynomial of the degree n, and that outside $[-1, 1]$ this polynomial is given by the formula

$$T_n(x) = \operatorname{ch}(n \operatorname{arc\,ch} x)$$

($\operatorname{ch} z = \frac{\exp\{z\}+\exp\{-z\}}{2}$, $\operatorname{arc\,ch} x$ is the function inverse to $\operatorname{ch}(\cdot)$: $\operatorname{ch}(\operatorname{arc\,ch} x) \equiv x$).

2) Let $0 < l < L < \infty$ be two positive reals, and let t be positive integer. Prove that the polynomial

$$q_t(z) = \frac{T_t\left(\frac{L+l-2z}{L-l}\right)}{T_t\left(\frac{L+l}{L-l}\right)}$$

belongs to $\mathcal{P}_t^*$ and that

$$\max_{l \leq z \leq L} |q_t(z)| \leq 2\left[\frac{\sqrt{Q}-1}{\sqrt{Q}+1}\right]^t, \quad Q = \frac{L}{l}.$$

Derive from this observation the fact announced in the beginnning of item A, Section 5.4.2.

Lecture 6

Quasi-Newton Methods

We continue with methods for solving unconstrained minimization problem

$$f(x) \to \min \mid \; x \in \mathbf{R}^n \tag{6.0.1}$$

with twice continuously differentiable f; now we are interested in the so called *quasi-Newton* methods. Same as the Conjugate Gradient, these methods try to imitate the behaviour of the Newton recurrence avoiding at the same time explicit usage of the second-order information.

6.1 Motivation

The quasi-Newton methods belong to the generic family of *variable metric routines* (Lecture 4, Section 4.2.2). Recall that the generic Variable Metric Algorithm 4.2.1 is the recurrence of the type

$$x_{t+1} = x_t - \gamma_{t+1} S_{t+1} \nabla f(x_t), \tag{6.1.1}$$

where S_{t+1} are symmetric positive definite matrices (in the notation of Algorithm 4.2.1, $S_{t+1} = A_{t+1}^{-1}$). As about stepsizes γ_{t+1}, they are given by a kind of line search in the direction

$$d_{t+1} = -S_{t+1} \nabla f(x_t);$$

these directions are descent for f at non-critical points x_t:

$$\nabla f(x_t) \neq 0 \Rightarrow d_{t+1}^T \nabla f(x_t) \equiv -(\nabla f(x_t))^T S_{t+1} \nabla f(x_t) < 0 \tag{6.1.2}$$

(S_t is positive definite!).

As we remember from Lecture 4, a good choice of S_t should make the matrices $A_{t+1} = S_{t+1}^{-1}$ "positive definite corrections" of the Hessians $\nabla^2 f(x_t)$, and the Modified Newton methods (Lecture 4) more or less straightforwardly implemented this idea: there we sequentially

- computed the Hessians $\nabla^2 f(x_t)$,
- modified them, if necessary, to make the resulting matrices A_{t+1} "well positive definite", and, finally,
- computed $S_{t+1} = A_{t+1}^{-1}$ to get the search directions d_{t+1}.

In the quasi-Newton methods we use another approach to implement the same idea: we compute the matrices S_{t+1} recursively without explicit usage of the Hessians of the objective and inverting these Hessians. The recurrence defining S_{t+1} is aimed to ensure, at least in good cases, that

$$S_{t+1} - [\nabla^2 f(x_t)]^{-1} \to 0, \ t \to \infty, \tag{6.1.3}$$

which, basically, means that the method asymptotically becomes close to the Newton one and therefore quickly converges. To support the latter claim and to make it quantitive, let us look what happens in the "best possible" case of strongly convex quadratic f.

Theorem 6.1.1 [Variable Metric Method with precise linesearch in the quadratic case] *Let*

$$f(x) = \frac{1}{2}x^T H x - b^T x$$

be a strongly convex quadratic form on $\mathbf{R}^n$ (so that the matrix H is positive definite and symmetric), and let f be minimized by the Variable Metric method (6.1.3) with exact linesearch:

$$\gamma_{t+1} = \operatorname*{argmin}_{\gamma} f(x_t + \gamma d_{t+1}).$$

Then the residuals in terms of the objective

$$E(x_t) \equiv f(x_t) - \min_x f(x)$$

satisfy the recurrence

$$E(x_{t+1}) \le E(x_t) \left[\frac{Q_{t+1} - 1}{Q_{t+1} + 1}\right]^2, \tag{6.1.4}$$

where Q_{t+1} is the condition number (ratio of the largest eigenvalue to the smallest one) of the matrix

$$H_{t+1} = S_{t+1}^{1/2} H S_{t+1}^{1/2}.$$

Proof. Let is fix t, and let us denote, for the sake brevity, S_{t+1} simply by S. Let also

$$\phi(\xi) = f(S^{1/2}\xi); \quad \xi_t = S^{-1/2}x_t; \quad \xi_{t+1} = S^{-1/2}x_{t+1},$$

so that

$$\phi(\xi_t) = f(x_t), \ \ \phi(\xi_{t+1}) = f(x_{t+1}). \tag{6.1.5}$$

By construction, x_{t+1} is the minimizer of f on the ray starting at x_t and directed by d_{t+1}; consequently, ξ_{t+1} is the minimizer of ϕ at the ray starting at ξ_t and directed by the vector

$$\delta_{t+1} = S^{-1/2} d_{t+1}.$$

Now,

$$\delta_{t+1} = S^{-1/2} d_{t+1} =$$

[definition of d_{t+1}]

$$= -S^{-1/2} S \nabla f(x_t) = -S^{1/2} \nabla f(x_t) =$$

[since $\phi(\xi) = f(S^{1/2}\xi)$ and, consequently, $\nabla\phi(\xi) = S^{1/2}\nabla f(S^{1/2}\xi)$]

$$= -\nabla\phi(\xi_t).$$

Thus, ξ_{t+1} *is nothing but the Steepest Descent iterate of* ξ_t.

Now,

$$\phi(\xi) = f(S^{1/2}\xi) = \frac{1}{2}\xi^T S^{1/2} H S^{1/2}\xi - b^T S^{1/2}\xi$$

is a strongly convex quadratic function with the Hessian matrix

$$\bar{H} = S^{1/2} H S^{1/2};$$

from Proposition 3.3.6 on the convergence rate of Steepest Descent as applied to a quadratic function we know that

$$\phi(\xi_{t+1}) - \min\phi \le [\phi(\xi_t) - \min\phi]\left[\frac{Q-1}{Q+1}\right]^2,$$

$Q = Q_{t+1}$ being the condition number of the matrix $S^{1/2} H S^{1/2} \equiv \nabla^2\phi$. Due to (6.1.5) and evident relation $\min\phi = \min f$ (ϕ is obtained from f by one-to-one substitution of argument), the resulting inequality is nothing but (6.1.4). ∎

We conclude from (6.1.4) that if f is strongly convex quadratic and the matrix $S_{t+1}^{1/2}\nabla^2 f S_{t+1}^{1/2}$ is close to the unit matrix I, so that Q_{t+1} is close to 1, then step t of our Variable Matrix method will multiply the residual $E(\cdot)$ by a factor close to 0. Since it is the same to say that the matrix $S^{1/2}\nabla^2 f S^{1/2}$ is close to the unit matrix and to say that S is close to $[\nabla^2 f]^{-1}$ (multiply the approximate equality $S^{1/2}\nabla^2 f S^{1/2} \approx I$ from the left and from the right by $S^{-1/2}$), we see that in the case of strongly convex quadratic f relation (6.1.3) (accompanied by exact line search) results in

$$E(x_{t+1})/E(x_t) \to 0, \;\; t \to \infty,$$

i.e., in superlinear convergence. This is in the quadratic case; but as always, the asymptotic properties of an optimization method in a non-quadratic case are similar to those in the strongly convex quadratic one (provided, of course, that the trajectory converges to a nondegenerate local minimizer of the objective), so that (6.1.3) indeed should result in good asymptotic rate of convergence, at least for good objectives.

The problem is how to ensure (6.1.3) at a "cheap" computational cost. The simplest way is to set

$$S_t \equiv [\nabla^2 f(x_0)]^{-1},$$

assuming, of course, that $\nabla^2 f(x_0)$ is positive definite (the "frozen" Newton method). Here we do not get (6.1.3), so that we have no hope for superlinear convergence. At the same time, under favourable circumstances – when the Hessians of f at different points are more or less "similar" to each other – we may hope that such a "scaling" will significantly accelerate the convergence. Indeed, we can observe from the proof of Theorem 6.1.1 that the indicated scaling makes the process – viewed in the coordinates $\xi = [\nabla^2 f(x_0)]^{1/2} x$ – the Steepest Descent as applied to the updated objective $\phi(\xi) = f([\nabla^2 f(x_0)]^{-1/2}\xi)$. If the Hessians of f are "similar" to each other, namely, we have (at least along the trajectory of the method)

$$l\nabla^2 f(x_0) \le \nabla^2 f(x) \le L\nabla^2 f(x_0)$$

with moderate L and not too small l, then the function ϕ will be, as it is immediately seen, strongly convex with parameters (l, L), so that the Steepest Descent as applied to ϕ (i.e., the "frozen" Newton method as applied to f) will converge linearly with the ratio $[(Q-1)/(Q+1)]^2$, $Q = L/l$ being the condition number of the transformed objective. This condition number

may be much smaller than the one of the initial objective, and whenever this is the case, the "frozen" Newton method will be much faster than the Steepest Descent as applied to the original objective. Although there are many "if" and "may" in the above paragraph, it is worthy to keep in mind the aforementioned possibility.

Now let us switch to more sophisticated policies of generating matrices S_t used in quasi-Newton methods.

6.2 Approximating inverse Hessians via first order information

6.2.1 The idea

The idea behind the below policies for ensuring (6.1.3) is very simple. We intend to generate S_t recursively. Namely, assume that at the step t (when updating x_{t-1} into x_t) we used certain approximation S_t of the matrix

$$[\nabla^2 f(x_{t-1})]^{-1},$$

so that x_t was obtained from x_{t-1} by certain step in the direction

$$-S_t g_{t-1}, \quad g_s \equiv \nabla f(x_s).$$

Thus,

$$p_t \equiv x_t - x_{t-1} \equiv -\gamma_t S_t g_{t-1}, \tag{6.2.1}$$

γ_t being the stepsize given by linesearch. The first-order information on f at the points x_t and x_{t-1} allows us to define the vectors

$$q_t \equiv g_t - g_{t-1}. \tag{6.2.2}$$

If the step p_t is small in norm (as it should be the case at the final stage) and f is twice continuously differentiable (which we assume from now on), then

$$q_t \equiv g_t - g_{t-1} \equiv \nabla f(x_t) - \nabla f(x_{t-1}) \approx [\nabla^2 f(x_{t-1})](x_t - x_{t-1}) \equiv [\nabla^2 f(x_{t-1})]p_t; \tag{6.2.3}$$

of course, if f is quadratic, $\approx$ is simply an equality, independently of whether p_t is small or not.

Thus, after the step t we have enriched our information on the Hessian: now we have not only the previous approximation S_t to $[\nabla^2 f(x_{t-1})]^{-1}$, but also some approximation (namely, q_t) to the vector $[\nabla^2 f(x_{t-1})]p_t$. We can use this additional information to update the previous approximation to the inverse Hessian. The simplest and the most natural idea is to impose on S_{t+1} the following condition (which is an equality version of the approximate equality (6.2.3)):

$$S_{t+1} q_t = p_t.$$

This relation, of course, can be satisfied by infinitely many updatings $S_t \mapsto S_{t+1}$, and there are different "natural" ways to specify this updating. When the policy of the updatings is fixed, we get a particular version of the generic Variable Metric algorithm, up to the freedom in the linesearch tactics. We shall always assume in what follows that this latter issue – which linesearch to use – is resolved in favour of the exact linesearch. Thus, the generic algorithm we are going to consider is as follows:

Algorithm 6.2.1 [Generic Quasi-Newton method]
Initialization: *choose somehow starting point x_0, the initial positive definite symmetric matrix S_1, compute $g_0 = \nabla f(x_0)$ and set $t = 0$.*
Step t: *given x_{t-1}, $g_{t-1} = \nabla f(x_{t-1}) \neq 0$ and S_t, check whether $g_{t-1} = 0$. If it is the case, terminate (x_{t-1} is a critical point of f), otherwise*

- *set*
$$d_t = -S_t g_{t-1};$$
- *perform exact line search from x_{t-1} in the direction d_t, thus getting new iterate*
$$x_t = x_{t-1} + \gamma_t d_t;$$
- *compute $g_t = \nabla f(x_t)$ and set*
$$p_t = x_t - x_{t-1},\ q_t = g_t - g_{t-1};$$
- (U): *update S_t into positive definite symmetric matrix S_{t+1}, maintaining the relation*
$$S_{t+1} q_t = p_t, \tag{6.2.4}$$
replace t with $t+1$ and loop.

When specifying the only "degree of freedom" in the presented generic algorithm, i.e., the rules for (U), our targets are <u>at least</u> the following:

- (A) the matrices S_t should be symmetric positive definite;
- (B) in the case of strongly convex quadratic f the matrices S_t should converge (ideally – in finite number of steps) to the inverse Hessian $[\nabla^2 f]^{-1}$.

The first requirement is the standard requirement for Variable Metric algorithms; it was motivated in Lecture 4. The second requirement comes from our final goal – to ensure, at least in good cases (when the trajectory converges to a nondegenerate local minimizer of f), that that $S_{t+1} - [\nabla^2 f(x_t)]^{-1} \to 0$, $t \to \infty$; as we remember, this property underlies fast – superlinear – asymptotical convergence of the algorithm.

Implementing (U) in a way which ensures the indicated properties and incorporating, in addition, certain policies which make the algorithm globally converging (e.g., restarts, which we already have used in non-quadratic extensions of the Conjugate Gradient method), we will come to globally converging methods with nice, in good cases, asymptotical convergence properties. In the rest of this lecture we carry this plan out, focusing on implementations of (U) compatible with requirements (A) and (B). The updating formulae $S_t \mapsto S_{t+1}$ we are about to design involve only the first-order information on f and, of course, do *not* use the hypothesis that f is quadratic – recall that we are interested in first-order algorithms which can be applied to general objectives. The "necessary condition" for such a general formula to be of interest for us is its ability to ensure (B); thus, in what follows we *design* "general" formulae and "test" them in the strongly convex quadratic case.

After these preliminary remarks, let us come to concrete policies for (U).

6.3 Rank 1 corrections

With this policy, one tries to update S_t into S_{t+1} in the simplest fashion – by adding to S_t a rank 1 matrix $\alpha_t z_t z_t^T$:
$$S_{t+1} = S_t + \alpha_t z_t z_t^T \tag{6.3.1}$$

with certain scalar α_t and vector z_t. These quantities are chosen to ensure (6.2.4):

$$p_t = S_{t+1}q_t = S_tq_t + \alpha_t[z_t^Tq_t]z_t. \tag{6.3.2}$$

From this relation,

$$z_t = \omega_t[p_t - S_tq_t];$$

substituting this expression into (6.3.2), we get

$$p_t - S_tq_t = \beta_t[(p_t - S_tq_t)^Tq_t][p_t - S_tq_t] \quad [\beta_t = \alpha_t\omega_t^2];$$

to get β_t, let us take the inner product of both sides in the latter equality with q_t, which results in

$$\beta_t = \frac{1}{(p_t - S_tq_t)^Tq_t}.$$

Thus, we come to the following formula for Rank 1 version of (U):

$$S_{t+1} = S_t + \frac{1}{(p_t - S_tq_t)^Tq_t}[p_t - S_tq_t][p_t - S_tq_t]^T. \tag{6.3.3}$$

Note that the right hand side in this formula not necessarily is well-defined: the denominator there can vanish. In this situation we distinguish between two cases. First, when $p_t - S_tq_t = 0$, so that $S_{t+1} = S_t$ satisfies the required relation $p_t = S_{t+1}q_t$. In this "good" case we, by definition, assign to the right hand side of (6.3.3) the value S_t [1)]. Second, when the vector $p_t - S_tq_t$ is nonzero, but is orthogonal to q_t. In this case let us agree that the right hand side is undefined, so that the Rank 1 updating cannot be used at the step in question.

In fact in the case of quadratic f updating (6.3.3), under mild nondegeneracy assumptions, results in $S_{n+1} = H^{-1}$. The underlying Linear Algebra statement is as follows:

Proposition 6.3.1 *Let H and S_1 be fixed symmetric matrices, let $p_1, ..., p_k$ be arbitrary vectors, and let S_t, $t = 2, 3, ..., k+1$, be defined by the recurrence (6.3.3):*

$$S_{t+1} = S_t + \frac{1}{(p_t - S_tq_t)^Tq_t}[p_t - S_tq_t][p_t - S_tq_t]^T,$$

where

$$q_t = Hp_t.$$

Then

($*_k$) $\quad p_i = S_{k+1}q_i,\ i = 1, ..., k.$

Proof is given by induction on k.

Base $k = 1$ is trivially valid: the updating rule (6.3.3) was derived exactly to satisfy the requirement

$$S_{t+1}q_t = p_t, \tag{6.3.4}$$

so that $S_2q_1 = p_1$, as required in (6.3.2) for $k = 1$.

Step. Assume $k > 1$, and let we already know that ($*_{k-1}$) is valid, i.e., that

$$S_kq_i = p_i, \quad i = 1, ..., k-1. \tag{6.3.5}$$

[1] in what follows I shall skip considerations related to this "favourable" case; it can be easily checked that occurrence of this situation never violates our forthcoming conclusions

To derive from this assumption $(*_k)$, we act as follows. First of all, (6.3.4) says that the equality $S_{k+1}q_k = p_k$ indeed is valid. Thus, all we should prove to verify $(*_k)$ is that the updating $S_k \mapsto S_{k+1}$ does not violate equalities (6.3.5). This is easy. We have from the updating formula (6.3.3):

$$S_{k+1}q_i = S_kq_i + [(p_k - S_kq_k)^Tq_i]y_k, \;\; y_k = \frac{1}{(p_k - S_kq_k)^Tq_k}(p_k - S_kq_k).$$

We have

$$(p_k - S_kq_k)^Tq_i = p_k^Tq_i - q_k^T(S_kq_i) =$$

[inductive hypothesis (6.3.5)]

$$= p_k^Tq_i - q_k^Tp_i =$$

[since, by assumption, $q_l = Hp_l$]

$$= p_k^THp_i - p_k^THp_i = 0;$$

thus,

$$S_{k+1}q_i = S_kq_i = p_i$$

(we again have used the inductive hypothesis). ∎

When applying Algorithm 6.2.1 with the Rank 1 implementation (6.3.3) of (U) to a strongly convex quadratic function

$$f(x) = \frac{1}{2}x^THx - b^Tx,$$

we are in situation $q_t = Hp_t$ for all t, so that the above Proposition can be applied; we therefore get for the case in question

$$p_i = S_{t+1}q_i \quad [\equiv S_{t+1}Hp_i],\; i = 1, ..., t.$$

In particular, if $p_1, ..., p_n$ (n is the dimension of the space) are linearly independent, we have $S_{n+1}Hp_i = p_i$ for vectors p_i forming a basis in $\mathbf{R}^n$, whence $S_{n+1}H = I$, or

$$S_{n+1} = H^{-1}.$$

Consequently (see Theorem 6.1.1),

$$x_{n+1} = x^*$$

is the exact minimizer of f. Thus, in the quadratic case the method in question results in $S_{n+1} = H^{-1}$, provided that the first n sequential search directions are linearly independent. We could expect, therefore, that in a non-quadratic case the scheme would result in $S_t - [\nabla^2 f(x_{t-1})]^{-1} \to 0$, $t \to \infty$, provided that the trajectory converges to a nondegenerate local minimizer of the objective, and this is exactly what we wish. There is, anyhow, a serious drawback of our simple scheme: generally speaking, it does not maintain positive definiteness of the matrices S_t (one can verify that this is guaranteed under additional assumption that $(p_t - S_tq_t)^Tq_t > 0$, which is not always the case). Thus, the Rank 1 scheme for (U) has certain limitations; this is why we should look for more sophisticated policies.

6.4 Davidon-Fletcher-Powell method

In this method, (U) is given by

$$S_{t+1} = S_t + \frac{1}{p_t^T q_t} p_t p_t^T - \frac{1}{q_t^T S_t q_t} S_t q_t q_t^T S_t. \tag{6.4.1}$$

We are about to demonstrate that (6.4.1) is well-defined, results in positive definite S_{t+1} and maintains (6.2.4).

Proposition 6.4.1 *Let R be a positive definite symmetric matrix, and let p and q be two vectors such that*

$$p^T q > 0. \tag{6.4.2}$$

Then the matrix

$$R' = R + \frac{1}{p^T q} p p^T - \frac{1}{q^T R q} R q q^T R \tag{6.4.3}$$

is symmetric positive definite and satisfies the relation

$$R' q = p. \tag{6.4.4}$$

Proof. 1^0. Let us prove that R' satisfies (6.4.4). Indeed, we have

$$R'q = Rq + \frac{1}{p^T q}(p^T q)p - \frac{1}{q^T R q}(q^T R q)Rq = Rq + p - Rq = p.$$

2^0. It remains to verify that R' is positive definite. Let x be an arbitrary nonzero vector; we should prove that $x^T R' x > 0$. Indeed,

$$x^T R' x = x^T R x + \frac{(x^T p)^2}{p^T q} - \frac{(x^T R q)^2}{q^T R q};$$

setting $a = R^{1/2}x, b = R^{1/2}q$, we can rewrite the relation as

$$x^T R' x = \frac{(a^T a)(b^T b) - (a^T b)^2}{b^T b} + \frac{(x^T p)^2}{p^T q}.$$

Since, by assumption, $p^T q > 0$, the second fraction is nonnegative. The first one also is nonnegative by Cauchy's inequality. Thus, $x^T R' x \geq 0$, while we need to prove that this quantity is positive. To this end it suffices to verify that both numerators in the right hand side of the latter relation cannot vanish simultaneously. Indeed, the first numerator can vanish only if a is proportional to b (this is said by Cauchy's inequality: it becomes equality only when the vectors in question are proportional). If a is proportional to b, then x is proportional to q (see the origin of a and b). But if $x = sq$ for some nonzero (x is nonzero!) s, then the second numerator is $s^2(q^T p)^2$, and we know that $q^T p$ is positive. ∎

Now we can prove that (6.4.1) indeed can be used as (U):

Proposition 6.4.2 *Let at a step t of Algorithm 6.2.1 with (U) given by (6.4.1) the matrix S_t be positive definite and $g_{t-1} = \nabla f(x_{t-1}) \neq 0$ (so that x_{t-1} is not a critical point of f, and the step t indeed should be performed). Then S_{t+1} is well-defined, is positive definite and satisfies (6.2.4).*

Proof. It suffices to verify that $q_t^T p_t > 0$; then we would be able to get all we need from Proposition 6.4.1 (applied with $R = S_t, q = q_t, p = p_t$).

Since $g_{t-1} \neq 0$ and S_t is positive definite, the direction $d_t = -S_t g_{t-1}$ is a descent direction of f at x_{t-1} (see (6.1.2)). Consequently, the exact linesearch results in a nonzero stepsize, and $p_t = x_t - x_{t-1} = \gamma_t d_t \neq 0$. We have

$$q_t^T p_t = \gamma_t (g_t - g_{t-1})^T d_t =$$

[since x_t is a minimizer of f on the ray $\{x_{t-1} + \gamma d_t \mid \gamma > 0\}$ and therefore g_t is orthogonal to the direction d_t of the ray]

$$= -\gamma_t g_{t-1}^T d_t = \gamma_t g_{t-1}^T S_t g_{t-1} > 0$$

(S_t is positive definite). ∎

Now we can establish the main property of the Davidon-Fletcher-Powell method – its finite convergence in the quadratic case. Moreover, we shall prove that in this case the trajectory generated by the method initialized with $S_1 = I$ is exactly the one of the Conjugate Gradient method, so that the DFP (Davidon-Fletcher-Powell) method with the indicated initialization is a Conjugate Gradient method – in the quadratic case it becomes the standard Conjugate Gradient.

Theorem 6.4.1 [DFP as a Conjugate Gradient method]
Let the DFP method be applied to a strongly convex quadratic objective

$$f(x) = \frac{1}{2} x^T H x - b^T x.$$

Then

(i) *For any t until the termination moment (i.e., for any t such that $g_t = \nabla f(x_t) \neq 0$) one has:*

1) The directions $p_1, ..., p_t$ are nonzero and H-orthogonal:

(1$_t$) $p_i^T H p_j = 0,\ 1 \leq i < j \leq t;$

2) One has

(2$_t$) $S_{t+1} H p_i = p_i,\ 1 \leq i \leq t.$

(ii) *The method is a Conjugate Directions one and, consequently, it terminates in at most n steps with exact minimizer of f. If it terminates exactly in n steps, then*

$$S_{n+1} = H^{-1}.$$

(iii) *If S_1 is proportional to the unit matrix, then the trajectory of the method coincides with the one the Conjugate Gradient method started at x_0.*

Proof. Let T be the termination moment (i.e., the moment t where it turns out that $g_t = 0$). A priori we do not know whether such a moment exists, and if it is not the case (in fact it is), we set $T = \infty$.

1^0. First note that if $t \leq T$, then

$$q_t = \nabla f(x_t) - \nabla f(x_{t-1}) = H(x_t - x_{t-1}) = H p_t; \tag{6.4.5}$$

consequently, from Proposition 6.4.2 one has

$$S_{t+1} H p_t = S_{t+1} q_t = p_t. \tag{6.4.6}$$

Besides this, from the same Proposition $p_t \neq 0$.
2^0. Let us prove (1_t) and (2_t) by induction on $t \leq T$.
Base $t = 1$ is valid: (1_1) is true by trivial reasons, and (2_1) follows from (6.4.6) applied with $t = 1$.
Step $t - 1 \Rightarrow t$. Let $1 < t \leq T$ and let (1_{t-1}) and (2_{t-1}) be valid; we should prove that (1_t) and (2_t) also are valid.

Since f is quadratic, we have for any s, $1 \leq s < t$:

$$g_{t-1} = g_s + H(x_{t-1} - x_s) = g_s + H(p_{s+1} + p_{s+2} + ... + p_{t-1}),$$

whence

$$p_s^T g_{t-1} = p_s^T g_s + p_s^T H(p_{s+1} + ... + p_{t-1}) =$$

[by (1_{t-1})]

$$= p_s^T g_s =$$

[since x_s is the minimizer of f on the open ray $\{x_{s-1} + \gamma p_s \mid \gamma > 0\}$ and therefore g_s is orthogonal to the direction p_s of the ray]

$$= 0.$$

Thus,

$$p_s^T g_{t-1} = 0, \ \ 1 \leq s < t; \tag{6.4.7}$$

applying (2_{t-1}), we get

$$0 = p_s^T g_{t-1} = p_s^T H S_t g_{t-1} = 0, \ 1 \leq s < t. \tag{6.4.8}$$

Now,

$$p_t = -\gamma_t S_t g_{t-1}$$

with positive γ_t, whence $S_t g_{t-1} = -\gamma_t^{-1} p_t$; substituting this expression into (6.4.8), we get

$$p_s^T H p_t = 0, \ 1 \leq s < t,$$

which combined with (1_{t-1}) results in (1_t).

We now have for $1 \leq s < t$:

$$q_t^T S_t H p_s =$$

[by (2_{t-1})]

$$= q_t^T p_s =$$

[since $q_t = H p_t$]

$$= p_t^T H p_s =$$

[by already proved (1_t) and since $s < t$]

$$= 0,$$

so that

$$q_t^T S_t H p_s = 0, \ \ 1 \leq s < t. \tag{6.4.9}$$

Consequently, from (6.4.1) for $1 \leq s < t$ we have

$$S_{t+1} H p_s = S_t H p_s + \frac{p_t^T H p_s}{p_t^T q_t} p_t - \frac{q_t^T S_t H p_s}{q_t^T S_t q_t} S_t q_t =$$

[since $S_t H p_s = p_s$ by (2_{t-1}), $p_t^T H p_s = 0$ by (1_t) and by (6.4.9)]

$$= p_s.$$

Thus,

$$S_{t+1} H p_s = p_s$$

for $1 \leq s < t$; this relation is valid for $s = t$ as well due to (6.4.6). Thus, we have proved (2_t). The induction is completed, and (i) is proved.

3^0. Relation (1_t) along with the fact that x_t is the minimizer of f on the open ray $\{x_{t-1} + \gamma x_t \mid \gamma > 0\}$ and consequently (f is convex!) on the line passing through x_{t-1} in the direction p_t demonstrates that the method indeed is the Conjugate Directions one associated with the recursively generated system $\{p_i\}$ of H-orthogonal directions. Indeed, we know from (1_t) that the directions $p_1, ..., p_T$ are nonzero H-orthogonal; by Proposition 5.3.1, a Conjugate Directions method associated with certain system of H-orthogonal vectors is (clearly uniquely) defined by the starting point and the requirement that the current iterate x_t is obtained from the previous one x_{t-1} by one-dimensional minimization of f along the line passing through x_{t-1} in the direction p_t [2)], and this is exactly the case with the DFP method as applied to a strongly convex quadratic objective.

Since in the quadratic case DFP becomes a Conjugate Directions method, it indeed terminates in at most n steps (Theorem 5.2.1). If it terminates in exactly n steps, then from (2_{n+1}) it follows that $S_{n+1} H p_i = p_i$ for a complete system $p_1, ..., p_n$ of nonzero H-orthogonal directions; such a system is a basis in $\mathbf{R}^n$ (Corollary 5.1.1), so that $S_{n+1} H = I$, as claimed in (ii). Thus, (ii) is proved.

4^0. It remains to verify that the DFP method started with $S_1 = \lambda I$ ($\lambda > 0$) is the Conjugate Gradient method. To see this, it suffices to prove that x_t is the minimizer of f on the t-th Krylov space

$$\mathcal{K}_t = x_0 + \mathrm{Lin}\{g_0, H g_0, ... H^{t-1} g_0\}, \;\; g_0 = \nabla f(x_0);$$

indeed, by Theorems 5.2.1 and 5.4.1, the trajectory of the Conjugate Gradient method is exactly the sequence of minimizers of f on these spaces.

Now, since we already know that DFP is a Conjugate Directions method, to prove that x_t is the minimizer of f on $\mathcal{K}_t$ is the same as to prove that

$$\mathrm{Lin}\{g_0, H g_0 ..., H^{t-1} g_0\} = \mathrm{Lin}\{p_1, ..., p_t\} \tag{6.4.10}$$

for every $t \leq T$ (see Theorem 5.2.1). In other words, we should prove that $p_t, t \leq T$, is a linear combination of $g_0, H g_0, ..., H^{t-1} g_0$; it will demonstrate that the right hand side in (6.4.10) is contained in the left hand one, and since the left hand side has dimension t – nonzero H-orthogonal vectors $p_1, ..., p_t$ are linearly independent! – and the right hand one – at most t, the inclusion of the right hand side into the left hand one in fact means their coincidence.

To prove that p_t is a linear combination of $g_0, H g_0, ..., H^{t-1} g_0$, let us use induction on t. The statement is valid for $t = 1$: $p_1 = -\gamma_1 S_1 g_0 = -\gamma_1 \lambda g_0$, as required. Now assume that $p_1, ..., p_{t-1}$ are linear combinations of the vectors $g_0, H g_0, ... H^{t-2} g_0$. From recurrence (6.4.1) governing S_i it immediately follows that S_t differs from $S_1 = \lambda I$ by a sum of $2(t-1)$ rank 1 matrices $\alpha_s z_s z_s^T$,

[2] in Proposition 5.3.1 this direction was called d_{t-1}, because there the directions were indiced starting with 0; now the initial direction index is 1

with reals α_s and vectors z_s of the form p_i or $S_i q_i = S_i H p_i$, where $i \leq t-1$. By inductive hypothesis, therefore, all z_s are linear combinations of $g_0, Hg_0, ..., H^{t-1}g_0$, so that

$$p_t = -\gamma_t S_t g_{t-1} = -\gamma_t S_1 g_{t-1} - \gamma_t [S_t - S_1] g_{t-1} = -\gamma_t \lambda g_{t-1} + \sum_{i=0}^{t-1} \beta_i H^i g_0.$$

It remains to note that $g_{t-1} = g_0 + H(x_{t-1} - x_0)$, and by the inductive hypothesis $H(x_{t-1} - x_0)$ is a linear combination of $Hg_0, ..., H^{t-1}g_0$. ∎

6.5 The Broyden family

The DFP version of (U) is certain rank 2 formula – the updated matrix S_{t+1} differs from S_t by a matrix of rank 2. The *Broyden* family of quasi-Newton methods is based on another rank 2 updating formula which we are about to introduce.

To get the starting point for our developments, note that relation (6.2.4), i.e.,

$$S_{t+1} q_t = p_t,$$

for the case of symmetric positive definite S_{t+1}, is equivalent to the relation

$$H_{t+1} p_t = q_t \tag{6.5.1}$$

with symmetric positive definite $H_{t+1} \equiv S_{t+1}^{-1}$. Thus,

each policy $\mathcal{P}$ of generating positive definite symmetric matrices S_t maintaining (6.2.4) induces certain policy $\mathcal{P}^*$ of generating positive definite symmetric matrices $H_t = S_t^{-1}$ maintaining (6.5.1),

and, of course, vice versa:

each policy $\mathcal{P}^*$ of generating positive definite symmetric matrices H_t maintaining (6.5.1) induces certain policy of generating positive definite symmetric matrices $S_t = H_t^{-1}$ maintaining (6.2.4).

Now, we know how to generate matrices H_t satisfying relation (6.5.1) – this is, basically, the same problem as we already have studied, but with swapped q_t and p_t. Thus,

given any one of the above updating formulae for S_t, we can construct a "complementary" updating formula for H_t by replacing, in the initial formula, all S's by H's and interchanging q's and p's.

For example, the complementary formula for the Rank 1 scheme (6.3.3) is

$$H_{t+1} = H_t + \frac{1}{p_t^T (q_t - H_t p_t)} [q_t - H_t p_t][q_t - H_t p_t]^T; \tag{6.5.2}$$

The complementary formula for the DFP updating scheme (6.4.1) is

$$H_{t+1} = H_t + \frac{1}{q_t^T p_t} q_t q_t^T - \frac{1}{p_t^T H_t p_t} H_t p_t p_t^T H_t, \tag{6.5.3}$$

– it is called the Broyden-Fletcher-Goldfarb-Shanno updating of H_t.

We have the following analogy to Proposition 6.4.2:

Proposition 6.5.1 *Let H_t be positive definite symmetric matrix, let x_{t-1} be an arbitrary point with $g_{t-1} = \nabla f(x_{t-1}) \neq 0$, and let d_t be an arbitrary descent direction of f at x_{t-1} (i.e., $d_t^T g_{t-1} < 0$). Let, further, x_t be the minimizer of f on the ray $\{x_{t-1} + \gamma d_t \mid \gamma \geq 0\}$, and let*

$$p_t = x_t - x_{t-1},\ q_t = \nabla f(x_t) - \nabla f(x_{t-1}) \equiv g_t - g_{t-1}.$$

Then the matrix H_{t+1} given by (6.5.3) is symmetric positive definite and satisfies (6.5.1).

Proof. From the proof of Proposition 6.4.2 we know that $p_t^T q_t > 0$, and it remains to apply Proposition 6.4.1 to the data $R = H_t$, $p = q_t$, $q = p_t$. ∎

According to Proposition 6.5.1, we can look at (6.5.3) as at certain maintaining (6.5.1) policy $\mathcal{P}^*$ of updating positive definite symmetric matrices H_t. As we know, every policy of this type induces certain policy of updating positive definite matrices $S_t = H_t^{-1}$ which maintains (6.2.4). In our case the induced policy is given by

$$S_{t+1} = \left[S_t^{-1} + \frac{1}{p_t^T q_t} q_t q_t^T - \frac{1}{p_t^T S_t^{-1} p_t} S_t^{-1} p_t p_t^T S_t^{-1} \right]^{-1}. \tag{6.5.4}$$

It can be shown by a direct computation (which I skip), that the latter relation is nothing but

(I) $\quad S_{t+1}^{BFGS} = S_t + \frac{1+q_t^T S_t q_t}{(p_t^T q_t)^2} p_t p_t^T - \frac{1}{p_t^T q_t} \left[p_t q_t^T S_t + S_t q_t p_t^T \right]$

(we write ^{BFGS} to indicate that this is the Broyden-Fletcher-Goldfarb-Shanno updating). Due to Proposition 6.5.1, (I) is a correct version of (U): independently of what is positive definite symmetric S_t, it results in positive definite symmetric S_{t+1} satisfying the relation

(*) $\qquad S_{t+1} q_t = p_t,$

exactly as the Davidon-Fletcher-Powell updating

(II) $\quad S_{t+1}^{DFP} = S_t + \frac{1}{q_t^T p_t} p_t p_t^T - \frac{1}{q_t^T S_t q_t} S_t q_t q_t^T S_t.$

Now we have two updating formulae – (I) and (II); they transform a positive definite matrix S_t into positive definite matrices S_{t+1}^{BFGS} and S_{t+1}^{DFP}, respectively, satisfying (*). Since (*) is linear in S_{t+1}, *any convex combination of the resulting matrices also satisfies (*).* Thus, we come to the *Broyden* implementation of (U) given by

$$S_{t+1}^{\phi} = (1-\phi) S_{t+1}^{DFP} + \phi S_{t+1}^{BFGS}; \tag{6.5.5}$$

here $\phi \in [0,1]$ is the parameter specifying the updating. Note that the particular case of $\phi = 0$ corresponds to the Davidon-Fletcher-Powell method.

One can verify by a direct computation that

$$S_{t+1}^{\phi} = S_{t+1}^{DFP} + \phi v_{t+1} v_{t+1}^T, \quad v_{t+1} = (q_t^T S_t q_t)^{1/2} \left[\frac{1}{p_t^T q_t} p_t - \frac{1}{q_t^T S_t q_t} S_t q_t \right]. \tag{6.5.6}$$

From the considerations which led us to (6.5.5), we get the following

Corollary 6.5.1 *A Broyden method, i.e., Algorithm 6.2.1 with (U) given by (6.5.5), $\phi \in [0,1]$ being the parameter of the method (which may vary from iteration to iteration), is a quasi-Newton method: it maintains symmetry and positive definiteness of the matrices S_t and ensures (6.2.4).*

In the quadratic case we have the following proposition (completely similar to Theorem 6.4.1):

Proposition 6.5.2 [Broyden method as a Conjugate Directions method]
Any Broyden method as applied to a strongly convex quadratic objective

$$f(x) = \frac{1}{2}x^T Hx - b^T x$$

becomes a Conjugate Directions method: if it does not terminate in course of the first t steps, then the directions $p_1, ..., p_t$ are nonzero H-orthogonal and the trajectory is exactly the one of the Conjugate Directions method associated with the directions $p_1, ..., p_t$. In particular, the method terminates in at most n steps with exact solution.

If the method does not terminate in course of the first t steps, then

$$S_{t+1}Hp_i = p_i,\ 1 \le i \le t; \tag{6.5.7}$$

in particular, if the method terminates exactly after n steps, one has

$$S_{n+1} = H^{-1}.$$

If S_0 is proportional to the unit matrix, then the trajectory of the method on f is exactly the one of the Conjugate Gradient method.

Proof is completely similar to the one of Theorem 6.4.1 and is skipped. In fact, there is no necessity to prove anything, due to the following remarkable fact:

It can be verified that *all Broyden methods, independently of the choice of the parameter ϕ, being started from the same pair (x_0, S_1), equipped with the same exact line search and being applied to the same problem, generate the same sequence of iterates (although not the same sequence of matrices H_t!).*

Thus, in the case of exact line search all methods from the Broyden family are the same. The methods became different only when inexact line search is used; although inexact line search is forbidden by our theoretical considerations, it is always the case in actual computations.

Broyden methods are thought to be the most efficient practically versions of the Conjugate Gradient and quasi-Newton methods. After intensive numerical testing of different policies of tuning the parameter ϕ, it was found that the best is the simplest policy $\phi \equiv 1$, i.e., the pure Broyden-Fletcher-Goldfarb-Shanno method.

Remark 6.5.1 Practical versions of the Broyden methods.
In practical versions of Broyden methods, exact line search is replaced with inexact one. Besides the standard requirements of "significant decrease" of the objective in course of the line search (like those given by the Armijo test), here we meet with specific additional requirement: the line search should ensure the relation

$$p_t^T q_t > 0. \tag{6.5.8}$$

In view of Proposition 6.4.1 this relation ensures that the updating formulae (I) and (II) (and, consequently, the final formula (6.5.5) with $\phi \in [0, 1]$) maintain positive definiteness of S_t's and relation (6.2.4), i.e., the properties crucial for the quasi-Newton methods.

Relation (6.5.8) is ensured by the exact line search (we know this from the proof of Proposition 6.4.2), but of course not only by it: the property is given by a strict inequality and therefore,

being valid for the stepsize given by the exact line search, is for sure valid if the stepsize is close enough to the "exact" one.

Another important implementation issue is as follows. Under assumption (6.5.8), updating (6.5.5) should maintain positive definiteness of the matrices S_t. In actual computations, anyhow, rounding errors may eventually destroy this crucial property, and when it happens, the method may become behave itself crazy. To overcome this difficlty, in good implementations people store and update from step to step not the matrices S_t themselves, but their Cholesky factors: lower-triangular matrices C_t such that $S_t = C_t C_t^T$, or, more frequently, the Cholesky factors C'_t of the inverse matrices S_t^{-1}. Updating formula (6.5.5) implies certain routines for updatings $C_t \mapsto C_{t+1}$ (respectively, $C'_t \mapsto C'_{t+1}$), and these are the formulae in fact used in the algorithms. The arithmetic cost of implementing such a routine is $O(n^2)$, i.e., is of the same order of complexity as the original formula (6.5.5); on the other hand, it turns out that this scheme is much more stable with respect to rounding errors, as far as the descent properties of the actually computed search directions are concerned.

6.6 Convergence of Quasi-Newton methods

6.6.1 Global convergence

In practice, quasi-Newton methods are usually executed in *continuous fashion:* Algorithm 6.2.1 is started at certain x_0 with certain positive definite S_1 and is run "forever" with the chosen version of (U). For such a scheme, *global convergence* is proved only for certain versions of the method and only under strong assumptions on f.

Of course, there is no difficulty in proving global convergence for the *scheme with restarts*, where one resets current S_t after every m steps to the initial (positive definite) value of the matrix; here m is certain fixed "cycle duration" (compare with the non-quadratic Conjugate Gradient methods from the previous lecture). For the latter scheme, it is easy to prove that if the level set

$$L = \{x \mid f(x) \leq f(x_0)\}$$

associated with the initial point is bounded, then the trajectory is bounded and all limiting points of the sequence $\{x_{mk}\}_{k=0}^{\infty}$ are critical points of f. The proof is quite similar to the one of Proposition 5.5.1: the steps with indices $1 + mt$ are

$$x_{mk+1} \in \mathrm{Argmin}\{f(x_{mk} - \gamma S \nabla f(x_{mk})) \mid \gamma \geq 0\},$$

S being a once for ever fixed symmetric positive definite matrix (the one to which S_t is reset at the restart steps). Such a step decreases the objective "significantly", provided that $\nabla f(x_{mk})$ is not small[3)]; this property can be immediately derived (try to do it!) from positive definiteness of S, continuous differentiability of the objective and boundedness of L. At the remaining steps the objective never increases due to the linesearch, and we conclude that the sum over t of progresses $f(x_{t-1}) - f(x_t)$ in the objective value is bounded. Thus, the progress at step t tends to 0 as $t \to \infty$, and, consequently, $\nabla f(x_{mk}) \to 0$ as $k \to \infty$ – otherwise, as it was indicated, the progresses at the restart steps could not tend to 0 as $k \to \infty$. Thus, $\nabla f(x_{mk}) \to 0$, $k \to \infty$, so that every limiting point of the sequence $\{x_{mk}\}_k$ indeed is a critical point of f.

[3]here is the exact formulation: for every $\epsilon > 0$ there exists $\delta > 0$ such that if, for some k, $|\nabla f(x_{mk})| \geq \epsilon$, then $f(x_{mk+1}) \leq f(x_{mk}) - \delta$

6.6.2 Local convergence

As far as *local* convergence is concerned, we, as always, are interested in the following question:

let x^* be a nondegenerate local minimizer of f (i.e., $\nabla f(x^*) = 0$, $\nabla^2 f(x^*)$ is positive definite). Assume that the trajectory of the method in question converges to x^*; what can be said about the asymptotical rate of convergence?

We are about to answer this question for the most frequently used BFGS method (recall that it is the Broyden method with $\phi \equiv 1$).

First of all, consider the the scheme with restarts and assume, for the sake of simplicity, that $m = n$ and $S = I$, S being the matrix to which S_t is reset after every n steps. Besides this, assume that the objective is smooth – three times continuously differentiable. In this case one can expect that the convergence of the sequence $\{x_{mk}\}_{k=0}^{\infty}$ to x^* will be quadratic. Indeed, in the strongly convex quadratic case the BFGS method with the initialization in question, as we remember from Proposition 6.5.2, becomes the Conjugate Gradient method, so that in fact we are speaking about a nonquadratic extension of the CG and can use the reasoning from the previous lecture. Our expectations indeed turn out to be valid.

Anyhow, the "squaring the distance to x^* in every n steps" is not that attractive property, especially when n is large. Moreover, the scheme with restarts is not too reasonable from the asymptotical point of view: all our motivation of the quasi-Newton scheme came from the desire to ensure in a good case (when the trajectory converges to a nondegenerate local minimizer of f) the relation $S_t - \nabla^2 f(x_t) \to 0$, $t \to \infty$, in order to make the method asymptotically similar to the Newton one; and in the scheme with restarts we destroy this property when resetting S_t to the unit matrix at the restarting steps. To utilize the actual potential of the quasi-Newton scheme, we should avoid restarts, at least asymptotically (at the beginning they might become necessary to ensure global convergence).

It is very difficult to prove something good about local convergence of a quasi-Newton method executed in "continuous fashion" without restarts. There is, anyhow, the following remarkable result of Powell (1976):

Theorem 6.6.1 *Consider the Broyden-Fletcher-Goldfarb-Shanno method (i.e., the Broyden method with $\phi \equiv 1$) without restarts and assume that the method converges to a nondegenerate local minimizer x^* of a three times continuously differentiable function f. Then the method converges to x^* superlinearly.*

This result has also extensions onto the "practical" – with properly specified inexact line search – versions of the method.

Assignment # 6 (Lecture 6)

Exercise 6.6.1 *Write a code implementing the BFGS method and run the code on*

1) the strongly convex quadratic functions described in Exercise 5.5.1 (use the scheme without restarts with S_1 set to the unit matrix)

Compare the results with those obtained for the Conjugate Gradient method when doing Exercise 5.5.1

2) the Rosenbrock problem

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

the starting point being $(-1.2, 1.0)$ (test both the scheme with and without restarts; set S_1 to the unit matrix)

Compare the results with those obtained for the Fletcher-Reeves and Polak-Ribiere methods when doing Exercise 5.5.2

3) the function described in Exercise 4.3.3

Compare the results with those obtained for the Damped Newton method when doing Exercise 4.3.3

Exercise 6.6.2 *1) Prove the following important Sherman-Morrison formula:*
if A is nonsingular $n \times n$ matrix and B, C are $n \times k$ matrices, then

$$(A + BC^T)^{-1} = A^{-1} - A^{-1}B(I_k + C^T A^{-1} B)^{-1} C^T A^{-1},$$

where I_k denotes the unit $k \times k$ matrix. The exact meaning of the equality is as follows: if the expression in one of the sides of the equality makes sense, then both the expressions make sense and are equal to each other.

Hint*: reduce the general case to the one of $A = I_n$; in this latter case, check by direct computation that the product of $(I_n + BC^T)$ and the right hand side is the unit matrix.*

The Sherman-Morrison formula is very useful in actual computations. Indeed, in many cases the following situation occurs: you know the inverse to certain "basic" matrix A and should compute the inverse to perturbed matrix $\bar{A} = A + BC^T$, where the perturbation BC^T is of relatively small rank (i.e., in the above notation $k << n$). The Sherman-Morrison formula demonstrates that the computation of $\bar{A}^{-1}$ – inversion of "large" (of order $n \times n$) perturbed matrix – can be reduced to inversion of "small" (of order $k \times k$) matrix $I_k + C^T A^{-1} B$.

In particular, in the case of "rank 1" correction ($k = 1$) computation of $\bar{A}^{-1}$, given A, B, C and A^{-1}, requires $O(n^2)$ arithmetic operations only; the particular version of the Sherman-Morrison formula for the case of $k = 1$ is called the *Rank 1 correction formula*.

2) [non-obligatory part] *Applying twice Rank 1 correction formula to (6.5.4), prove formula (I).*

How to remember the Sherman-Morrison formula. I personally always derive the formula from its particular case corresponding to $A = I$, and use the following "formal series trick" (which in fact explains the origin of the Sherman-Morrison identity): substituting formally into the formal series

$$(I + x)^{-1} = I - x + x^2 - x^3 - x^4 + ...$$

$x = BC^T$, we get

$$(I + BC^T)^{-1} = I - BC^T + BC^T BC^T - BC^T BC^T BC^T + ... =$$

$$= I - B[I - C^T B + C^T B C^T + ...]C^T = I - B(I + C^T B)^{-1} C^T.$$

Of course, this is just to remember the formula, not to prove it!

Lecture 7

Convex Programming

Today we start the second part of the Course – methods for solving *constrained* optimization problems. This lecture is in sense something special – we shall speak about *Convex Programming problems*. A Convex Programming problem is the one of the form

$$f_0(x) \to \min \mid \ f_i(x) \leq 0,\ i = 1, ..., m,\ x \in G \subset \mathbf{R}^n, \tag{7.0.1}$$

where

- G – the *domain* of the problem – is a closed *convex* set in $\mathbf{R}^n$;
- the objective f_0 and the constraints f_i, $i = 1, ..., m$, are *convex* functions on $\mathbf{R}^n$.

Convex Programming is, in a sense, the "solvable case" in Nonlinear Optimization: as we shall see in a while, convex programs, in contrast to the general ones, can be solved efficiently: one can approximate their *global* solutions by *globally linearly converging, with objective-independent ratio,* methods. This phenomenon – possibility to approximate efficiently global solutions – has no analogy in the general nonconvex case[1] and comes from nice geometry of convex programs.

Computational tools for Convex Programming is the most developed, both theoretically and algorithmically, area in Continuous Optimization, and what follows is nothing but a brief introduction to this rich area. In fact I shall speak about only one method – the *Ellipsoid algorithm*, since it is the simplest way to achieve the main goal of the lecture – to demonstrate that Convex Programming indeed is "solvable". Practical conclusion of this theoretical in its essence phenomenon is, in my opinion, very important, and I would formulate it as follows:
When modelling a real-world situation as an optimization program, do your best to make the program convex; I strongly believe that it is more important than to minimize the number of design variables or to ensure smoothness of the objective and the constraints. Whenever you are able to cast the problem as a convex program, you get in your disposal wide spectrum of efficient and reliable optimization tools.

7.1 Preliminaries

Let me start with recalling to you two important notions of Convex Analysis – those of a *subgradient* of a convex function and a *separating plane*.

[1] Recall that in all our previous considerations the best we could hope for was the convergence to a local minimizer of the objective, and we never could guarantee this local minimizer to be the global one. Besides this, basically all our rate-of-convergence results were asymptotical and depended on local properties of the objective.

7.1.1 Subgradients of convex functions

In our previous considerations, we always required the objective to be smooth – at least once continuously differentiable – and used the first order information (the gradients) or the second order one (the gradients and the Hessians) of the objective. In Convex Programming, there is no necessity to insist on differentiability of the objective and the constraints, and the role of gradients is played by *subgradients*. Recall that if g is a convex function on $\mathbf{R}^n$, then at each point $x \in \mathbf{R}^n$ there exists at least one *subgradient* $g'(x)$ of g – vector such that

$$g(y) \geq g(x) + (y-x)^T g'(x). \tag{7.1.1}$$

Geometrically: the graph of the linear function

$$y \mapsto g(x) + (y-x)^T g'(x)$$

is everywhere below the graph of g itself, and at the point x both graphs touch each other. Generally speaking, subgradient of g at a point x is not unique; it is unique if and only if g is differentiable at x, and in this case the unique subgradient of g at x is exactly the gradient $\nabla g(x)$ of g at x.

When speaking about methods for solving (7.0.1), we always assume that we have in our disposal a *First order oracle*, i.e., a routine which, given on input a point $x \in \mathbf{R}^n$, returns on output the values $f_i(x)$ and certain subgradients $f_i'(x)$, $i = 0, ..., m$ of the objective and the constraints at x.

7.1.2 Separating planes

Given a closed convex set $G \subset \mathbf{R}^n$ and a point x outside G, one can always *separate x from G by a hyperplane*, i.e., to point out a *separator* – vector e such that

$$e^T x > \sup_{y \in G} e^T y. \tag{7.1.2}$$

Geometrically: whenever $x \notin G$ (G is convex and closed), we can point out a *hyperplane*

$$\Pi = \{y \mid e^T y = a\}$$

such that x and G belong to opposite open half-spaces into which Π splits $\mathbf{R}^n$:

$$e^T x > a \ \& \ \sup_{y \in G} e^T y < a.$$

When speaking about methods for solving (7.0.1), we always assume that we have in our disposal a *Separation oracle* for G, i.e., a routine which, given on input a point $x \in \mathbf{R}^n$, reports whether $x \in G$, and if it is not the case, returns "the proof" of the relation $x \notin G$, i.e., a vector e satisfying (7.1.2).

Remark 7.1.1 Implementation of the oracles. The assumption that when solving (7.0.1), we have in our disposal a First order oracle for the objective and the constraints and a Separation oracle for the domain G are crucial for our abilities to solve (7.0.1) efficiently. Fortunately, this assumption indeed is valid in all "normal" cases. Indeed, typical convex functions f arising in applications are either differentiable – and then, as in was already mentioned, subgradients are

the same as gradients, so that they are available whenever we have an "explicit" representation of the function – or are upper bounds of differentiable convex functions:

$$f(x) = \sup_{\alpha \in A} f_\alpha(x).$$

In the latter case we have no problems with computing subgradients of f in the *discrete case* – when the index set A is finite. Indeed, it is immediately seen from the definition of a subgradient that if $\alpha(x)$ is the index of the largest at x (or a largest, if there are several of them) of the functions f_α, then the gradient of $f_{\alpha(x)}$ at the point x is a subgradient of f as well. Thus, if all f_α are "explicitly given", we can simply compute all of them at x and choose the/a one largest at the point; its value will be the value of f at x, and its gradient at the point will be a subgradient of f at x.

Now, the domain G of (7.0.1) typically is given by a set of convex inequalities:

$$G = \{x \in \mathbf{R}^n \mid g_j(x) \leq 0,\ j = 1, ..., k\}$$

with explicitly given and simple – differentiable – convex functions g_j, e.g.

$$G = \{x \mid (x - a)^T (x - a) - R^2 \leq 0\}$$

(centered at a Euclidean ball of radius R) or

$$G = \{x \mid a_i \leq x \leq b_i,\ i = 1, ..., n\}$$

(a box). Whenever it is the case, we have no problems with Separation oracle for G: given x, it suffices to check whether all the inequalities describing G are satisfied at x. If it is the case, then $x \in G$, otherwise $g_i(x) > 0$ for some i, and it is immediately seen that $\nabla g_i(x)$ can be chosen as separator.

It should be stressed, anyhow, that not all convex programs admit "efficient" oracles. E.g., we may meet with a problem where the objective (or a constraint) is given by "continuous" maximization like

$$f(x) = \max_{\alpha \in [0,1]} f_\alpha(x)$$

("semiinfinite programs"); this situation occurs in the problems of best uniform approximation of a given function $\phi(\alpha)$, $0 \leq \alpha \leq 1$, by a linear combination $\sum_{i=1}^n x_i \phi_i(\alpha)$ of other given functions; the problem can be written down as a "simple" convex program

$$f(x) \equiv \max_{\alpha \in [0,1]} |\phi(\alpha) - \sum_{i=1}^n x_i \phi_i(\alpha)| \to \min;$$

this is a convex program, but whether it can or cannot be solved efficiently, it heavily depends on whether we know how to perform efficiently maximization in α required to compute the value of f at a point.

7.2 The Ellipsoid Method

We are about to present one of the most general algorithms for convex optimization – the *Ellipsoid method*. The idea of the method is more clear when we restrict ourselves with the problem of the type

$$f(x) \to \min \mid x \in G \subset \mathbf{R}^n \tag{7.2.1}$$

of minimizing a convex function f on a *solid* G, i.e., a closed and bounded convex set with a nonempty interior. In the mean time we shall see that general problem (7.0.1) can be easily reduced to (7.2.1).

7.2.1 The idea

The method extends onto multi-dimensional case the scheme of the simplest method for univariate minimization, namely, of the Bisection (Lecture 2). The idea is very simple: let us take an arbitrary interior point $x_1 \in G$ and compute a subgradient $f'(x_1)$ of the objective at the point, so that

$$f(x) - f(x_1) \geq (x - x_1)^T f'(x_1). \tag{7.2.2}$$

It is possible that $f'(x_1) = 0$; then (7.2.2) says that x_1 is a global minimizer of f on $\mathbf{R}^n$, and since this global minimizer belongs to G, it is an optimal solution to the problem, and we can terminate. Now assume that $f'(x_1) \neq 0$ and let us ask ourselves what can be said about localization of the optimal set, let it be called X^*. The answer is immediate:

from (7.2.2) it follows that if x belongs to the open half-space

$$\Pi_1^+ = \{x \mid (x - x_1)^T f'(x_1) > 0\},$$

so that the right hand side in (7.2.2) is positive at x, then x for sure is not optimal: (7.2.2) says that $f(x) > f(x_1)$, so that x is worse than feasible solution x_1. Consequently, the optimal set (about which we initially knew only that $X^* \subset G$) belongs to the new localizer

$$G_1 = \{x \in G \mid (x - x_1)^T f'(x_1) \leq 0\}.$$

This localizer again is a convex solid (as the intersection of G and a closed half-space with the boundary passing through the interior point x_1 of G) *and is smaller than G* (since an interior point x_1 of G is a boundary point of G_1).

Thus, choosing somehow an interior point x_1 in the "initial localizer of the optimal set" $G \equiv G_0$ and looking at $f'(x_1)$, we either terminate with exact solution, or can perform a cut – pass from G_0 to a smaller solid G_1 which also contains the optimal set. In this latter case we can iterate the construction – choose somehow an interior point x_2 in G_1, compute $f'(x_2)$ and, in the case of $f'(x_2) \neq 0$, perform a new cut – replace G_1 with the new localizer

$$G_2 = \{x \in G_1 \mid (x - x_2)^T f'(x_2) \leq 0\},$$

and so on. With the resulting recurrence, we either terminate at certain step with exact solution, or generate a sequence of shrinking solids

$$G = G_0 \supset G_1 \supset G_2 \supset ...,$$

every of the solids containing the optimal set.

It can be guessed that if x_t are chosen properly, then the localizers G_t "shrink to X^* at certain reasonable rate", and the method converges. This is, e.g., the case with Bisection: there all G_t are segments (what else could be a solid on the axis?), and x_t is chosen as the center of G_{t-1}; as a result, the lengths of the segments G_t go to 0 linearly with the ratio $\frac{1}{2}$, and since all the localizers contain X^*, the method converges at the same linear rate.

In multi-dimensional case the situation is much more difficult:

- the sets G_t can be more or less arbitrary solids, and it is not clear what does it mean a "center" of G_t, i.e., how to choose $x_{t+1} \in G_t$ in order to achieve "significant shrinkage" of the localizer at a step (by the way, how to measure this shrinkage?) On the other hand, it is clear that if x_{t+1} is badly chosen (e.g., is close to the boundary of G_t) and the new cut is "badly oriented", the next localizer G_{t+1} may be "almost as large as G_t";
- in the one-dimensional case the linear sizes of the localizers tend to 0 linearly with ratio $\frac{1}{2}$, and this is the source of convergence of x_t to the minimizer of f. It is absolutely clear that in the multi-dimensional case we cannot enforce the linear sizes of G_t to go to 0 (look what happens if $G = G_0$ is the unit square on the plane and f does not depend on the first coordinate; in this case all localizers will be rectangles of the same width, and consequently we cannot enforce their linear sizes to go to 0). Thus, we should think carefully how to measure sizes of the localizers – with bad choice of the notion of size, we are unable to push it to 0.

The above remarks demonstrate that it is not that straightforward – to extend Bisection onto the multi-dimensional case. Nevertheless, there are satisfactory ways to do it.

7.2.2 The Center-of-Gravity method

The first which comes to mind is to measure the sizes of the localizers as their n-dimensional volumes $\mathrm{Vol}(G_t)$ and to use, as x_{t+1}, the *center of gravity* of G_t:

$$x_{t+1} = \frac{1}{\mathrm{Vol}(G_t)} \int_{G_t} x dx.$$

A (very nontrivial) geometrical fact is that with this *Center of Gravity* policy we get linear convergence of the volumes of G_t to 0:

$$\mathrm{Vol}(G_t) \leq \left[1 - \left(\frac{n}{n+1}\right)^n\right]^t \mathrm{Vol}(G_0),$$

which in turn implies linear convergence in terms of the residual in the objective value: if x^t is the best – with the smallest value of the objective – of the points $x_1, ..., x_t$, then

$$f(x^t) - \min_G f \leq \left[1 - \left(\frac{n}{n+1}\right)^n\right]^{t/n} [\max_G f - \min_G f]. \tag{7.2.3}$$

(7.2.3) demonstrates *global linear convergence of the Center-of-Gravity method with objective-independent convergence ratio*

$$\kappa(n) = \left[1 - \left(\frac{n}{n+1}\right)^n\right]^{1/n} \leq (1 - \exp\{-1\})^{1/n}.$$

Consequently, *to get an ϵ-solution to the problem – to find a point $x^t \in G$ with*

$$f(x^t) - \min_G f \leq \epsilon[\max_G f - \min_G f]$$

– it requires to perform no more than

$$\lfloor \frac{\ln \frac{1}{\epsilon}}{\ln \frac{1}{\kappa(n)}} \rfloor \leq 2.13 n \ln \frac{1}{\epsilon} + 1 \tag{7.2.4}$$

steps of the method[2)].

Look how strong is the result: our *global* efficiency estimate is objective-independent: no condition numbers (and even no smoothness assumptions at all) are involved! f may be nonsmooth, may possess all kinds of degeneracy, etc., and all this does not influence the estimate!

Let me add that the Center-of-Gravity method is, in certain precise sense, an optimal method for convex optimization.

A weak point of the method is the necessity to find, at every step, the center of gravity of the previous localizer. To find the center of gravity of a general type multi-dimensional solid (and even of a general type polytope) is, computationally, an extremely difficult problem. Of course, if G (and then every G_t) is a polytope, this problem is "algorithmically solvable" – one can easily point out a method which solves the problem in finitely many arithmetic operations. Unfortunately, the number of arithmetic operations required by all known methods for finding the center of gravity grows exponentially with n, so that are unable to to compute centers of gravity in reasonable time already in the dimension 5-10. As a result, the Center-of-Gravity method is of "academic interest" only – it cannot be used as a computational tool.

7.2.3 From Center-of-Gravity to the Ellipsoid method

The Ellipsoid method can be viewed as certain "computationally tractable" approximation to the Center-of-Gravity method. The idea is to enforce all the localizers to have "nice geometry" convenient for finding the centers of gravity, namely, to be ellipsoids. Assume, e.g., that $G_0 = G$ is an ellipsoid. Then there is no difficulty to point out the center of G_0, so that we have no problems with the first step of the Center-of-Gravity method. Unfortunately, the resulting localizer, let it be called G_1^+, will be not an ellipsoid, but a "half-ellipsoid" – the intersection of ellipsoid G_0 and a half-space with the boundary hyperplane passing through the center of G_0; this solid is not that convenient for finding the center of gravity. To restore the geometry of the localizer, let us cover the half-ellipsoid G_1^+ by the ellipsoid of the smallest volume G_1 containing G_1^+, and let us take this G_1 as our new localizer. Note that G_1 indeed is a localizer of the optimal set X^* (since the latter is contained already in G_1^+, and G_1 covers G_1^+). Now we are in the same situation as at the very beginning, but with G_0 replaced with G_1, and can iterate the construction – to take the center x_2 of the ellipsoid G_1, to perform a new cut, getting a new half-ellipsoid G_1^+ which covers the optimal set, to embed it into the ellipsoid G_2 of the smallest possible volume, etc. In this scheme, we actually make a kind of trade-off between efficiency of the routine and computational complexity of a step: when extending the "actual localizers" – half-ellipsoids – to ellipsoids, we add points which for sure could not be optimal solutions, and thus slow the procedure down. At the cost of this slowing the process down we, anyhow, enable ourselves to deal with "simple sets" – ellipsoids, and thus reduce dramatically the computational complexity of a step.

There are two points which should be clarified.

- first, it is unclear in advance whether we indeed are able to decrease at linear rate the volumes of sequential localizers and thus get a converging method – it could happen that, when extending the half-ellipsoid G_{t+1}^+ to the ellipsoid G_{t+1} of the smallest volume containing G_{t+1}^+, we come back to the previous ellipsoid G_t, so that no progress in volume is achieved. Fortunately, this is <u>not</u> the case: the procedure reduces the volumes of the sequential ellipsoids G_t by factor $\kappa^*(n) \leq \exp\{-\frac{1}{2n}\}$, thus enforcing the volumes of G_t

[2]here and in what follows we use the standard notation $\lfloor a \rfloor$ to denote the smallest integer $\geq$ a real a

to go to 0 at linear rate with the ratio $\kappa^*(n)$ [3]. This ratio is worse than the one for the Center-of-Gravity method (there the ratio was at most absolute constant $1-\exp\{-1\}$, now it is dimension-dependent constant close to $1-\frac{1}{2n}$); but we still have linear convergence!

- second, it was assumed that G is an ellipsoid. What to do if it is not so? The answer is easy: let us choose, as G_0, an arbitrary ellipsoid which covers the domain G of the problem; such a G_0 for sure will be a localizer of the optimal set, and this is all we need. This answer is good, but there is a weak point: it may happen that the center x_1 of the ellipsoid G_0 is outside G; how should we perform the first cut in this case? Moreover, this "bad" situation – when the center x_{t+1} of the current localizer G_t is outside G – may eventually occur even when $G_0 = G$ is an ellipsoid: at each step of the method, we add something to the "half" of the previous localizer, and this something could contain points not belonging to G. As a result, G_t, generally speaking, is not contained in G, and it may happen that the center of G_t is outside G. And to the moment we know how to perform cuts only through points of G, not through arbitrary points of $\mathbf{R}^n$.

  We can immediately resolve the indicated difficulty. Given the previous ellipsoidal localizer G_t and its center x_{t+1}, let us ask the Separation oracle whether $x_{t+1} \in G$. If it is the case, we have no problems with the cut – we call the First order oracle, get a subgradient of f at x_{t+1} and use it as it was explained above to produce the cut. Now, if x_{t+1} is not in G, the Separation oracle will return a separator e:

$$e^T x_{t+1} > \max_{x \in G} e^T x.$$

  Consequently, all the points $x \in G$ satisfy the inequality

$$(x - x_{t+1})^T e < 0,$$

  and we can use e for the cut, setting

$$G_{t+1}^+ = \{x \in G_t \mid (x - x_{t+1})^T e \leq 0\}.$$

  Since the inequality which "cuts G_{t+1}^+ off G_t" is satisfied on the entire G and, consequently, on X^*, and the latter set was assumed to belong to G_t (G_t is a localizer!), we conclude that $X^* \subset G_{t+1}^+$, and this is all we need.

After all explanations and remarks, we can pass to formal description of the Ellipsoid method.

7.2.4 The Algorithm

How to represent an ellipsoid

An ellipsoid is a geometrical entity; to run an algorithm, we should deal with numerical representations of these entities. The most convenient for our goals is to represent an ellipsoid as the image of the unit Euclidean ball under nonsingular affine mapping:

$$E = E(c, B) = \{x = c + Bu \mid u^T u \leq 1\}. \tag{7.2.5}$$

[3] The fact that even after extending G_{t+1}^+ to G_{t+1} we still have progress in volume heavily depends on the specific geometrical properties of ellipsoids; if. e.g., we would try to replace the ellipsoids with boxes, we would fail to ensure the desired progress. The ellipsoids, anyhow, are not the only solids appropriate for our goals; we could use simplices as well, although with worse progress in volume per step

Here $c \in \mathbf{R}^n$ is the center of the ellipsoid and B is an $n \times n$ nonsingular matrix. Thus, to say "an ellipsoid" in what follows means exactly to say "the set (7.2.5) given by a pair (c, B) comprised of vector $c \in \mathbf{R}^n$ and a nonsingular $n \times n$ matrix B"; you may think about this representation as about the *definition* of an ellipsoid.

The volume the ellipsoid $E(c, b)$ is

$$\mathrm{Vol}(E(c, B)) = |\mathrm{Det}\ B| v(n),$$

$v(n)$ being the volume of the unit n-dimensional Euclidean ball V_n; indeed, $E(c, B)$ is the image of V_n under affine mapping, B being the matrix of the homogeneous part of the mapping, and such a transformation multiplies the volumes by $|\mathrm{Det}\ B|$.

We need the following simple

Lemma 7.2.1 *Let $n > 1$, let*

$$E = E(c, B) = \{x = c + Bu \mid u^T u \leq 1\}$$

be an ellipsoid in $\mathbf{R}^n$, and let

$$E^+ = \{x \in E \mid (x - c)^T e \leq 0\} \quad [e \neq 0]$$

be a "half-ellipsoid" – the intersection of E and a half-space with the boundary hyperplane passing through the center of E. Then E^+ can be covered by ellipsoid $E' = E(c', B')$ of the volume

$$\mathrm{Vol}(E') = \kappa^*(n)\, \mathrm{Vol}(E),$$

$$\kappa^*(n) = \frac{n^2}{n^2 - 1}\sqrt{\frac{n-1}{n+1}} \leq \exp\{-\frac{1}{2(n-1)}\}. \tag{7.2.6}$$

The parameters c' and B' of the ellipsoid E' are given by

$$B' = \alpha(n) B - \gamma(n)(Bp)p^T, \quad c' = c - \frac{1}{n+1} Bp, \tag{7.2.7}$$

where

$$\alpha(n) = \left\{\frac{n^2}{n^2-1}\right\}^{1/4}, \quad \gamma(n) = \alpha(n)\sqrt{\frac{n-1}{n+1}}, \quad p = \frac{B^T e}{\sqrt{e^T B B^T e}}.$$

To prove the lemma, it suffices to reduce the situation to the similar one with E being the unit Euclidean ball $V = V_n$; indeed, since E is the image of V under the affine transformation $u \mapsto Bu + c$, the half-ellipsoid E^+ is the image, under this transformation, of the half-ball

$$V^+ = \{u \in V \mid (B^T e)^T u \leq 0\} = \{u \in V \mid p^T u \leq 0\}.$$

Now, it is quite straightforward to verify that a half-ball indeed can be covered by an ellipsoid V' with the volume being the required fraction of the volume of V; you may take

$$V' = \{x \mid (x + \frac{1}{n+1}p)^T [\alpha(n) I_n - \gamma(n) p p^T]^{-2} (x + \frac{1}{n+1}p) \leq 1\}.$$

It remains to note that the image of V' under the affine transformation which maps the unit ball V onto the ellipsoid E is an ellipsoid which clearly contains the half-ellipsoid E^+ and is in the same ratio of volumes with respect to E as V' is with respect to the unit ball V (since the ratio of volumes remains invariant under affine transformations). The ellipsoid E' given in formulation of the lemma is nothing but the image of the above V' under our affine transformation. ∎

The Ellipsoid algorithm

The algorithm is as follows

Algorithm 7.2.1 [The Ellipsoid algorithm for convex program $f(x) \to \min \mid x \in G \subset \mathbf{R}^n$]
Assumptions:

- *G is a solid (bounded and closed convex set with a nonempty interior)*
- *we are given First order oracle for f and Separation oracle for G*
- *we are able to point out an ellipsoid $G_0 = E(c_0, B_0)$ which contains G.*

Initialization: *set $t = 1$*
Step t: Given ellipsoid $G_{t-1} = E(c_{t-1}, B_{t-1})$, set $x_t = c_{t-1}$ and act as follows:
1) [Call the Separation Oracle] *Call the Separation oracle, x_t being the input. If the oracle says that $x_t \in G$, call the step t productive and go to 2), otherwise call the step t non-productive, set e_t equal to the separator returned by the oracle:*

$$x \in G \Rightarrow (x - x_t)^T e_t < 0$$

and go to 3)
2) [Call the First order oracle] *Call the First order oracle, x_t being the input. If $f'(x_t) = 0$, terminate – $x_t \in G$ is the minimizer of f on the entire $\mathbf{R}^n$ (see the definition (7.1.1) of a subgradient) and is therefore an optimal solution to the problem. In the case of $f'(x_t) \neq 0$ set*

$$e_t = f'(x_t)$$

and go to 3).
3) [Update the ellipsoid] *Update the ellipsoid $E(c_{t-1}, B_{t-1})$ into $E(c_t, B_t)$ according to formula (7.2.7) with $e = e_t$, i.e., set*

$$B_t = \alpha(n) B_{t-1} - \gamma(n)(B_{t-1} p_t) p_t^T, \quad c_t = c_{t-1} - \frac{1}{n+1} B_{t-1} p_{t-1},$$

where

$$\alpha(n) = \left\{ \frac{n^2}{n^2 - 1} \right\}^{1/4}, \quad \gamma(n) = \alpha(n) \sqrt{\frac{n-1}{n+1}}, \quad p_t = \frac{B_{t-1}^T e_t}{\sqrt{e_t^T B_{t-1} B_{t-1}^T e_t}}.$$

Replace t with $t + 1$ and go to the next step.
Approximate solution x^t generated by the method after t steps is, by definition, the best (with the smallest value of f_0) of the points x_j generated at the productive steps $j \leq t$ [if the steps $1, ..., t$ are non-productive, x^t is undefined].

7.2.5 The Ellipsoid algorithm: rate of convergence

We are about to prove the main result on the Ellipsoid method.

Theorem 7.2.1 [Rate of convergence of the Ellipsoid algorithm]
Let $n > 1$, and let convex program (7.2.1) satisfying the assumptions from the description of the Ellipsoid algorithm 7.2.1 be solved by the algorithm. Let

$$N(\epsilon) = \rfloor 2n(n-1) \ln \left(\frac{\mathcal{V}}{\epsilon} \right) \lfloor,$$

where $0 < \epsilon < 1$ *and*

$$\mathcal{V} = \left[\frac{\mathrm{Vol}(E(c_0, B_0))}{\mathrm{Vol}(G)}\right]^{1/n}$$

Then, for any $\epsilon \in (0,1)$*, the approximate solution* $x^{N(\epsilon)}$ *found by the method in course of the first* $N(\epsilon)$ *steps, is well-defined and is an* ϵ*-solution to the problem, i.e., belongs to* G *and satisfies the inequality*

$$f(x^{N(\epsilon)}) - \min_G f \le \epsilon[\max_G f - \min_G f]. \tag{7.2.8}$$

Proof. For the sake of brevity, let $N = N(\epsilon)$. We may assume that the method does not terminate in course of the first N steps – otherwise there is nothing to prove: the only possibility for the method to terminate is to find an exact optimal solution to the problem.

Let us fix $\epsilon' \in (\epsilon, 1)$, and let x^* be an optimal solution to (7.2.1) (it exists, since the domain of the problem is compact and the objective, being convex on $\mathbf{R}^n$, is continuous (we had such a theorem in the course Optimization I) and therefore attains its minimum on compact set G.
1^0. Set

$$G^* = x^* + \epsilon'(G - x^*),$$

so that G^* is the image of G under homothety transformation with the center at x^* and the coefficient ϵ'.

By construction, we have

$$\mathrm{Vol}(G^*) = (\epsilon')^n \,\mathrm{Vol}(G) > \epsilon^n \,\mathrm{Vol}(G) \tag{7.2.9}$$

("homothety with coefficient $\alpha > 0$ in $\mathbf{R}^n$ multiplies volumes by α^n"). On the other hand, by Lemma 7.2.1 we have $\mathrm{Vol}(E(c_t, B_t)) \le \exp\{-1/(2(n-1))\}\,\mathrm{Vol}(E(c_{t-1}, B_{t-1}))$, whence

$$\mathrm{Vol}(E(c_N, B_N) \le \exp\{-\frac{N}{2(n-1)}\}\,\mathrm{Vol}(E(c_0, B_0)) \le$$

[definition of $N = N(\epsilon)$]

$$\le \mathcal{V}^{-N}\epsilon^N \,\mathrm{Vol}(E(c_0, B_0)) =$$

[definition of $\mathcal{V}$]

$$= \frac{\mathrm{Vol}(G)}{\mathrm{Vol}(E(c_0, B_0))}\epsilon^N \,\mathrm{Vol}(E(c_0, B_0)) = \epsilon^N \,\mathrm{Vol}(G).$$

Comparing the resulting inequality with (7.2.9), we conclude that $\mathrm{Vol}(E(c_N, B_N)) < \mathrm{Vol}(G^*)$, so that

$$G^* \backslash E(c_N, B_N) \ne \emptyset. \tag{7.2.10}$$

2^0. According to (7.2.10), there exists

$$y \in \mathrm{Vol}(G^*) \backslash E(c_N, B_N).$$

I claim that

$$y = (1-\epsilon')x^* + \epsilon' z \text{ for some } z \in G; \qquad (y - x_t)^T e_t > 0 \text{ for some } t \le N \tag{7.2.11}$$

The first relation in (7.2.11) comes from the inclusion $y \in G^*$: by definition,

$$G^* = x^* + \epsilon'(G - x^*) = \{x^* + \epsilon'(z - x^*) \mid z \in G\}.$$

To prove the second relation in (7.2.11), note that from the first one $y \in G \subset E(c_0, B_0)$, while by the origin of y we have $y \notin E(c_N, B_N)$. Consequently, there exists $t \leq N$ such that

$$y \in E(c_{t-1}, B_{t-1}) \ \& \ y \notin E(c_t, B_t). \tag{7.2.12}$$

According to our policy of updating the ellipsoids (see Lemma 7.2.1), $E(c_t, B_t)$ contains the half-ellipsoid

$$E^+(c_{t-1}, B_{t-1}) = \{x \in E(c_{t-1}, B_{t-1}) \mid (x - x_t)^T e_t \leq 0\};$$

this inclusion and (7.2.12) demonstrate that $(y - x_t)^T e_t > 0$, as required in the second relation in (7.2.11).

3^0. Now let us look what happens at the step t given by the second relation in (7.2.11). First of all, I claim that the step t is productive: $x_t \in G$. Indeed, otherwise, by construction of the method, e_t would be a separator of x_t and G:

$$(x - x_t)^T e_t < 0 \quad \forall x \in G,$$

but this relation, as we know from (7.2.11), is violated at $y \in G$ and therefore cannot take place.

Thus, t is productive, whence, by construction of the method, $e_t = f'(x_t)$. Now the second relation in (7.2.11) reads

$$(y - x_t)^T f'(x_t) > 0,$$

whence, by definition of subgradient, $f(y) > f(x_t)$. This inequality, along with productivity of the step t and the definition of approximate solutions, says that x^N is well-defined and

$$f(x^N) \leq f(y). \tag{7.2.13}$$

4^0. Now we are done. By the first relation in (7.2.11) and due to convexity of f we have

$$f(y) \leq (1 - \epsilon')f(x^*) + \epsilon' f(z),$$

whence

$$f(y) - \min_G f \equiv f(y) - f(x^*) \leq \epsilon'(f(z) - f(x^*)) \leq \epsilon'[\max_G f - \min_g f] \qquad [z \in G\ !];$$

combining the resulting inequality with (7.2.13), we get

$$f(x^N) - \min_G f \leq \epsilon'[\max_G f - \min_G f].$$

The latter inequality is valid for all $\epsilon' \in (\epsilon, 1)$, and (7.2.8) follows. ∎

7.2.6 Ellipsoid method for problems with functional constraints

To the moment we spoke about the Ellipsoid method as applied to problem (7.2.1) without functional constraints. There is no difficulty to extend the method onto general convex problems (7.0.1). To this end it suffices to note that (7.0.1) can be immediately rewritten as problem of the type (7.2.1), namely, as

$$f(x) \equiv f_0(x) \to \min \mid \ x \in \hat{G} \equiv \{x \in G \mid \ f_i(x) \leq 0,\ i = 1, ..., m\}. \tag{7.2.14}$$

All we need to solve this latter problem by the Ellipsoid algorithm is

- to equip (7.2.14) by the Separation and the First order oracles

  this is immediate: the First order oracle for (7.2.14) is readily given by the one for (7.0.1). The Separation oracle for $\hat{G}$ can be immediately constructed from the Separation oracle for G and the First order oracle for (7.0.1): given $x \in \mathbf{R}^n$, we first check whether the point belongs to G, If it is not the case, then it, of course, does not belong to $\hat{G}$ as well, and the separator (given by the Separation oracle for G on the input x) separates x from $\hat{G}$. Now, if $x \in G$, we can ask the First order oracle about the values and subgradients of $f_1, ..., f_m$ at x. If all the values are nonpositive, then $x \in \hat{G}$; if one of this values is positive, then $x \notin \hat{G}$ and, moreover, the vector $f_i'(x)$ ($i \geq 1$ is such that $f_i(x) > 0$) can be used as a separator of x and $\hat{G}$ (Remark 7.1.1)

- to ensure that $\hat{G}$ is a solid

  (we shall simply assume this).

The resulting algorithm is as follows

Algorithm 7.2.2 [The Ellipsoid algorithm for convex program (7.0.1)]
Assumptions:

- $\hat{G} = \{x \in G \mid f_i(x) \leq 0, i = 1, ..., m\}$ *is a solid (bounded and closed convex set with a nonempty interior)*
- *we are given First-order oracle for* $f_0, ..., f_m$ *and Separation oracle for* G
- *we are able to point out an ellipsoid* $G_0 = E(c_0, B_0)$ *which contains* $\hat{G}$.

Initialization: *set* $t = 1$
Step t: *Given ellipsoid* $G_{t-1} = E(c_{t-1}, B_{t-1})$, *set* $x_t = c_{t-1}$ *and act as follows:*

1) [Call the Separation oracle] *Call the Separation oracle for* G, x_t *being the input. If the oracle says that* $x_t \in G$, *go to 2), otherwise call the step* t *non-productive, set* e_t *equal to the separator returned by the oracle:*

$$x \in G \Rightarrow (x - x_t)^T e_t < 0$$

and go to 4)

2) [Call the First order oracle] *Call the First order oracle,* x_t *being the input, and check whether* $f_i(x_t) \leq 0$, $i = 1, ..., m$. *If* $f_i(x_t) \leq 0$ *for all* $i \geq 1$, *call the step* t *productive and look at* $f_0'(x_t)$. *If* $f_0'(x_t) = 0$, *terminate – *x_t *is feasible for (7.0.1) and is the minimizer of* f *on the entire* $\mathbf{R}^n$ *(see the definition (7.1.1) of a subgradient), whence it is an optimal solution to the problem. In the case of* $f_0'(x_t) \neq 0$ *set*

$$e_t = f_0'(x_t)$$

and go to 4).

3) [The case of $x_t \in G$ and $f_i(x_t) > 0$ for some $i \geq 1$] *Call step* t *non-productive and find* $i \geq 1$ *such that* $f_i(x_t) > 0$ *(when we arrive at 3), such an* i *exists), set*

$$e_t = f_i'(x_t)$$

and go to 4).

4) [Updating the ellipsoid] *Update the ellipsoid* $E(c_{t-1}, B_{t-1})$ *into* $E(c_t, B_t)$ *according to formula (7.2.7) with* $e = e_t$*, i.e., set*

$$B_t = \alpha(n)B_{t-1} - \gamma(n)(B_{t-1}p_t)p_t^T, \quad c_t = c_{t-1} - \frac{1}{n+1}B_{t-1}p_{t-1},$$

where

$$\alpha(n) = \left\{\frac{n^2}{n^2-1}\right\}^{1/4}, \quad \gamma(n) = \alpha(n)\sqrt{\frac{n-1}{n+1}}, \quad p_t = \frac{B_{t-1}^T e_t}{\sqrt{e_t^T B_{t-1} B_{t-1}^T e_t}}.$$

Replace t *with* $t+1$ *and go to the next step.*
Approximate solution x^t *generated by the method after* t *steps is, by definition, the best (with the smallest value of* f*) of the points* x_j *generated at the productive steps* $j \le t$ *[if the steps* $1, ..., t$ *are non-productive,* x^t *is undefined].*

The following result is an immediate corollary of Theorem 7.2.1.

Theorem 7.2.2 [Rate of convergence of the Ellipsoid algorithm on problems with functional constraints]
Let $n > 1$*, and let convex program (7.0.1) satisfying the assumptions from the description of the Ellipsoid algorithm 7.2.2 be solved by the algorithm. Let*

$$N(\epsilon) = \lfloor 2n(n-1)\ln\left(\frac{\mathcal{V}}{\epsilon}\right)\rfloor,$$

where $0 < \epsilon < 1$ *and*

$$\mathcal{V} = \left[\frac{\mathrm{Vol}(E(c_0, B_0))}{\mathrm{Vol}(\hat{G})}\right]^{1/n}$$

Then, for any $\epsilon \in (0,1)$*, the approximate solution* $x^{N(\epsilon)}$ *found by the method in course of the first* $N(\epsilon)$ *steps, is well-defined and is an* ϵ*-solution to the problem, i.e., belongs to* $\hat{G}$ *and satisfies the inequality*

$$f(x^{N(\epsilon)}) - \min_{\hat{G}} f \le \epsilon[\max_{\hat{G}} f - \min_{\hat{G}} f]. \tag{7.2.15}$$

7.3 Ellipsoid method and Complexity of Convex Programming

The Ellipsoid method implies fundamental theoretical results on *complexity* of Convex Programming; let us briefly discuss these theoretical issues.

7.3.1 Complexity: what is it?

When speaking about complexity of a class of computational problems, we are interested to answer the following crucial question:

Given a family $\mathcal{P}$ *of problem instances and required accuracy* ϵ*, what can be said about the computational effort sufficient to solve each instance to the prescribed accuracy?*

This is an informal question, of course, and to answer it, we should first formalize the question itself, namely, to say

- What does it mean "a family of problems" and how we measure accuracy of approximate solutions;

- What does it mean "effort of computational process"? What is the model of the process and how the effort is measured?

To realize that these indeed are points which should be clarified, consider the following "universal solution method" (which in fact is the main method to solve problems): think! Is it a "computational method"? What is the "effort" for this method?

There are basically two approaches to formalize the complexity-related notions. The first, which is more convenient for "continuous" computational problems, is as follows (according to our goals, I shall speak only about optimization problems in the form (7.2.1)).

Real Arithmetic Complexity Model:

- A family of problems $\mathcal{P}$ is a set of problems (7.2.1) such that a particular member $p = (f, G)$ of the family is encoded by a finite-dimensional *data vector* $d(p)$. The dimension of the data vector is called the *size* of the instance.

  Example 1. Linear Programming over Reals. Here the instances are of the form

  $$f(x) = c^T x \to \min \mid \; x \in G = \{x \in \mathbf{R}^n \mid \; Ax \leq b\},$$

  A being $m \times n$ matrix and b being m-dimensional vector. The data vector of a problem instance is comprised of the pair n, m and of $n + m + nm$ entries of c, b, A written down in a once for ever fixed order (e.g., first the n entries of c, then the m entries of b and finally the nm entries of A in the row-wise order).

  Example 2. Quadratically Constrained Quadratic Programming. Here the instances are of the form

  $$f_0(x) = \frac{1}{2} x^T H_0 x - b_0^T x \to \min$$

  subject to

  $$x \in G = \{x \in \mathbf{R}^n \mid \; f_i(x) = \frac{1}{2} x^T H_i x - b_i^T x + c_i \leq 0, \; i = 1, ..., m\},$$

  and the data vector is comprised of n, m and the entries of the matrices H_i, the vectors b_i and the reals c_i written down in a once for ever fixed order.

  The number of examples can be easily extended; basically, the typical families of problems in question are comprised of problems of a "fixed generic analytical structure", and the data vector is the set of numerical coefficients of the (fixed by the description of the family) analytical expressions corresponding to the particular instance in question.

  In what follows we restrict ourselves with the families comprised of *solvable* problems with *bounded* feasible sets.

- An ϵ-solution x to a problem instance $p = (f, G)$ is a feasible ($x \in G$) solution such that

  $$f(x) - \min_G f \leq \epsilon [\max_G f - \min_G f];$$

  there are other definitions of an ϵ-solution, but let us restrict ourselves with the indicated one.

- A computational algorithm for $\mathcal{P}$ is a program for an idealized computer capable to perform operations of exact real arithmetic (four arithmetic operations and computation of elementary functions like $\sqrt{\cdot}$, $\sin(\cdot)$, $\log(\cdot)$, etc.). When solving a problem instance from $\mathcal{P}$, the algorithm gets on input the data vector of the instance and the required accuracy ϵ and starts to process these data; after finitely many elementary steps the algorithm should terminate and return the vector $x_{\rm out}$ of the corresponding dimension, which should be an ϵ-solution to the input instance. The computational effort of solving the instance is, by definition, the total # of elementary steps (arithmetic operations) performed in course of processing the instance.

The second way to formalize the complexity-related notions, the way which is the most widely used in theoretical Computer Science and in Combinatorial Optimization, is given by

Algorithmic Complexity Model:

- A family of problems is a set of problems (7.2.1) such that a particular member $p = (f, G)$ of the family is encoded by an *integer* $i(p)$; the # L of digits in this integer is the bit size of the instance p.

  Example 3. Integer programming. Here the problem instances are of the form

  $$f(x) = c^T x \to \min \mid x \in G = \{x \in \mathbf{Z}^n \mid Ax \le b\},$$

  where $\mathbf{Z}^n$ is the space of n-dimensional vectors with integer entries, c and b are vectors of dimensions n and m with integer entries, and A is an $m \times n$ matrix with integer entries.

  To encode the data (which form a collection of integers) as a single integer, one can act as follows:

  first, write down the data as a finite sequence of binary integers (similarly to Example 1, where the data were real);

  second, encode the resulting *sequence* of binary integers as a *single* integer, e.g., by representing

  - binary digit 0 as 00

  - binary digit 1 as 11

  - blank between integers as 01

  - sign – at an integer as 10

  Example 4. Linear Programming over Rationals. This is a subfamily of the Linear Programming family (Example 1), where we impose on all the entries of c, b, A the requirement to be rational numbers. To encode a problem instance by a single binary integer, it suffices to write the data as a sequence of binary integers, same as in Example 3 (with the only difference that now every rational element of the data is represented by two sequential integers, its numerator and denominator), and then encode the sequence, as it was explained above, by a single binary integer.

  In what follows, speaking about Algorithmic Complexity model, we always assume that

  - the families in question are comprised of solvable problems with bounded feasible sets

- any problem instance from the family admits a solution which can be naturally encoded by a binary integer

  The second assumption is clearly valid for the Integer Programming (Example 3). It is also valid for Linear Programming over Rationals (Example 4), since it can be easily proved that a solvable LP program with integer data admits a solution with rational entries; and we already know that any finite sequence of integers/rationals can be naturally encoded by a single integer.

- Normally, in Algorithmic Complexity model people are not interested in approximate solutions and speak only about exact ones; consequently, no problem with measuring accuracy occurs.

- A computational algorithm is an algorithm in any of (the equivalent to each other) definitions given in Mathematical Logic; you loose nothing when thinking of a program for an idealized computer which is capable to store in memory as many finite binary words as you need and to perform bit-wise operations with these words. The algorithm as applied to a problem instance p from $\mathcal{P}$ gets on input the code $i(p)$ of the instance and starts to process it; after finitely many elementary steps, the algorithm should terminate and return the code of an exact solution to the instance. The computational effort is measured as the total # of elementary bit-wise operations performed in course of the solution process.

Let me stress the difference between the notions of the size of an instance in the Real Arithmetic and the Algorithmic Complexity models. In the first model, the size of an instance is, basically, the # of real coefficients in the natural analytical representation of the instance, and the size is independent of the numerical values of these coefficients – they may be arbitrary reals. E.g. the Real Arithmetic size of any LP program over Reals (Example 1) with $n = 2$ variables and $m = 3$ inequality constraints is the dimension of the vector

$$(2, 3, c_1, c_2, b_1, b_2, b_3, A_{11}, A_{12}, A_{21}, A_{22}, A_{31}, A_{32}),$$

i.e., 13. In contrast to this, the Algorithmic Complexity size of an LP program over Rationals (Example 4) with $n = 2$ variables and $m = 3$ constraints can be arbitrarily large, if the rational data are "long".

We can observe similar difference between the measures of computational effort in the Real Arithmetic and the Algorithmic Complexity models. In the first of them, the effort required, say, to add two reals is one, independently of what the reals are. In the second model, we in principle cannot deal with reals – only with integers; and the effort to add two N-digit integers is not 1 – it is proportional to N.

7.3.2 Computational Tractability = Polynomial Solvability

After the main complexity-related notions – family of problems, ϵ-solution, solution algorithm and its computational effort – are formalized, we may ask ourselves

What is the complexity of a given family of problems, i.e., the best possible, over all solution algorithms, computational effort sufficient to solve all instances of a given size to a given accuracy?

Before trying to answer this question, we may ask ourselves a more rough (and, in a sense, more important) question:

Is the family in question computationally tractable, i.e. is its complexity a "reasonable" function of the size of instances?

If the answer to this second question is negative, then we may forget about the first one: if we know that to solve an instance of size l from a family $\mathcal{P}$ it may require *at least* 2^l elementary steps, then we may not bother much whether the actual complexity is 2^l, 2^{5l} or $2^{(2^l)}$: in the first case, the maximal size of instances we are able to solve for sure in reasonable time is something like 30, in the second – 6, in the third – 5; this difference normally means nothing, since the sizes of actually interesting problems usually are hundreds, thousands and tens of thousands.

Now, the notion of "computationally tractable" complexity in theoretical studies of the last 30 years is unanimously understood as the one of "polynomial time" solvability. This latter notion first was introduced in the Algorithmic Complexity model and was defined as follows:

A solution algorithm $\mathcal{A}$ for a family $\mathcal{P}$ of problem instances is called polynomial, if the computational effort required by $\mathcal{A}$ to solve a problem instance of an arbitrary bit size L never exceeds $q(L)$, q being certain polynomial; here all complexity-related notions are understood according to the Algorithmic Complexity model.
$\mathcal{P}$ is called polynomially solvable, if it admits polynomial solution algorithm.

The analogy of this definition for the Real Arithmetic Complexity model is as follows:

A solution algorithms $\mathcal{A}$ for a family $\mathcal{P}$ of problem instances is called R-polynomial, if the computational effort required by $\mathcal{A}$ to solve a problem instance p of an arbitrary size l to an arbitrary accuracy $\epsilon \in (0,1)$ never exceeds

$$q(l) \ln\left(\frac{\mathcal{V}(p)}{\epsilon}\right), \tag{7.3.1}$$

where q is certain polynomial and $\mathcal{V}(p)$ is certain data-dependent quantity; here all complexity-related notions are understood according to the Real Arithmetic Complexity model.

The definition of an R-polynomial algorithm admits a very natural interpretation. The quantity $\ln(\mathcal{V}(p)/\epsilon)$ can be viewed as # of accuracy digits in an ϵ-solution; with this interpretation, (7.3.1) means that *the arithmetic cost per accuracy digit is bounded from above by a polynomial of the dimension of the data vector.*

7.3.3 R-Polynomial Solvability of Convex Programming

Invention of the Ellipsoid method (1976) allowed to establish the following important

Theorem 7.3.1 [R-polynomial solvability of Convex Programming]
Consider a family $\mathcal{P}$ of convex problems (7.2.1) and assume that

(i) *all problem instances $p = (f, G)$ from the family are bounded (i.e., their feasible domains G are bounded) and strictly feasible (i.e., their feasible domains G possess nonempty interiors); moreover, given the data vector $d(p)$ of the instance, one can in polynomial in the size $l_p = \dim d(p)$ of the instance number of arithmetic operations compute an ellipsoid which covers G along with a lower bound $v_p > 0$ for the volume of G;*

(ii) *given an instance $p = (f, G) \in \mathcal{P}$ and a point $x \in \mathbf{R}^{n_p}$, n_p being the # of variables in p, on can in polynomial in l_p number of arithmetic operations*

- *imitate the Separation oracle for G on the input x, i.e., check the inclusion $x \in G$ and, if it is not valid, compute a separator e:*

$$\sup_{y \in G} e^T y < e^T x;$$

- *imitate the First order oracle for f, i.e., compute $f(x)$ and $f'(x)$.*

Then there exists an R-polynomial algorithm for $\mathcal{P}$.

Proof is immediate: it suffices to use the Ellipsoid algorithm. Indeed,

- (i) says that, given on input the data vector $d(p)$ of a problem instance $p \in \mathcal{P}$, we can in polynomial in $l_p = \dim d(p)$ number of arithmetic operations compute an ellipsoid $E(c_0, B_0)$ and thus start the Ellipsoid algorithm. As a byproduct, we observe that the number of variables n_p in p is bounded from above by a polynomial of the size l_p of the instance (otherwise it would require too many operations simply to write down the center of the ellipsoid).

- according to the description of the method, to perform a step of it, we should imitate the Separation and, possibly, the First order oracle at current point x_t (according to (iii), it takes polynomial in l_p number of arithmetic operations) and then should perform $O(n_p^2)$ arithmetic operations to update the previous ellipsoid into the current one according to Algorithm 7.2.1.3); by virtue of the previous remark, n_p is bounded from above by a polynomial in l_p, so that the overall arithmetic effort at a step is polynomial in l_p;

- according to Theorem 7.2.1, the method will find an ϵ-solution to p in the number of steps not exceeding

$$N(\epsilon) = \lfloor 2(n_p - 1) n_p \ln\left(\frac{\mathcal{V}}{\epsilon}\right) \rfloor, \quad \mathcal{V} = \left[\frac{\mathrm{Vol}(E(c_0, B_0))}{\mathrm{Vol}(G)}\right]^{1/n_p};$$

  we have

$$N(\epsilon) \le N'(\epsilon) = \lfloor 2(n_p - 1) n_p \ln\left(\frac{\mathrm{Vol}^{1/n_p}(E(c_0, B_0))}{v_p^{1/n_p}\epsilon}\right) \rfloor$$

  (see (i)); according to (i), we can compute $N'(\epsilon)$ in polynomial in l_p number of operations and terminate the process after $N'(p)$ steps, thus getting an ϵ-solution to p.

The overall arithmetic effort to find an ϵ-solution to p, in view of the above remarks, is bounded from above by

$$r(l_p) N'(\epsilon) \le q(l_p) \ln\left(\frac{\mathcal{V}(p)}{\epsilon}\right), \quad \mathcal{V}(p) \equiv \left[\frac{\mathrm{Vol}(E(c_0, B_0))}{v_p}\right]^{1/n_p},$$

both $r(\cdot)$ and $q(\cdot)$ being certain polynomials, so that the presented method indeed is R-polynomial.

■

When thinking of Theorem 7.3.1, you should take into account that the "unpleasant" assumptions (i) are completely technical and normally can be ensured by slight regularization of problem instances. Assumption (ii) is satisfied in all "non-pathological" applications, so that in fact Theorem 7.3.1 can be qualified as a General Theorem on R-Polynomial Solvability of

Convex Programming. I should add that this is, in a sense, an "existence theorem": it claims that in Convex Programming there exists a "universal" R-polynomial solution routine, but it does not say that this is the best possible R-polynomial routine for all particular cases. The main practical drawback of the Ellipsoid method is that it is "slow" – the # of iterations required to solve the problem within a once for ever fixed accuracy ϵ grows quadratically with the dimension n of the problem. This quadratic in n growth of the effort makes it basically impossible to use the method as a practical tool in dimensions like hundreds and thousands. Recently, for many important families of Convex Programming problems (Linear, Quadratically Constrained Quadratic, Geometric and some others) more specialized and more efficient *interior point polynomial algorithms* were developed; these methods are capable to struggle with large-scale problems of real-world origin.

I would say that the Ellipsoid method, as a practical tool, is fine for not large – up to 20-30 variables – convex problems. The advantages of the method are:

- simplicity in implementation and good numerical stability
- universality
- low order of dependence of computational effort on m – the # of functional constraints

  The latter remark relates to Algorithm 7.2.2): the number of steps required to achieve a given accuracy is simply independent of m, and the effort per step is at most proportional to the m (the only place where the number of constraints influence the effort is the necessity to check feasibility of the current iterate and to point out a violated constraint, if any exists). As a result, for "high and narrow" convex programs (say, with up to 20-30 variables and thousands of constraints as you wish) the Ellipsoid method seems to be one of the best known Convex Programming routines.

7.4 Polynomial solvability of Linear Programming

Surprisingly, the most important "complexity consequences" of the Ellipsoid method – which is a purely "continuous optimization" algorithm – relate to *Algorithmic Complexity*. The most known of these consequences is

7.4.1 Polynomial Solvability of Linear Programming over Rationals

Some History

Everybody knows that Linear Programming programs – both with real and and rational data – can be solved efficiently (and are solved almost for 50 years) by the Simplex Algorithm, which is a finite Real Arithmetic routine for LP capable to solve huge practical linear programs. What I am not sure is that everybody knows what does it mean "efficiently" in the above sentence. This is *practical efficiency* – when solving an LP program

$$c^T x \to \min \mid Ax \leq b \tag{7.4.1}$$

with n variables and $m > n$ inequality constraints, the Simplex *normally* performs $2m$ - $4m$ iterations to find the solution. Anyhow, theoretically Simplex is not a polynomial algorithm: since the beginning of sixties, it is known that there exist simple LP programs p_n with $n = 1, 2, 3, ...$ variables and integer data of the total bit size $L_n = O(n)$ such that some versions

of the Simplex method solve p_n in no less than 2^n steps. Consequently, the aforementioned versions of the Simplex method are not polynomial – for a polynomial algorithm, the solution time should be bounded by a polynomial of $L_n = O(n)$, i.e., of n. Similar "bad examples" were constructed for other versions of the Simplex method; and although nobody was able to prove that *no* version of Simplex can be polynomial, nobody was also lucky to point out a polynomial version of the Simplex method. Moreover, since mid-sixties, where the Algorithmic Complexity approach became standard, and till 1979 nobody knew whether LP over Rationals (Example 4 above) is or is not polynomially solvable.

7.4.2 Khachiyan's Theorem

The positive answer on the fundamental question whether LP over Rationals is polynomially solvable was given only in 1979 by L. Khachiyan; the main tool used in the proof was the Ellipsoid method invented 3 years earlier.

Below I sketch the reasoning of Khachiyan.

Step 1: from Optimization to Feasibility

The first step of the construction reduces the *optimization LP program* (7.4.1) to the *Feasibility program* for a system of linear inequalities. To this end it suffices to write down both the inequalities of (7.4.1) and of its LP dual and to replace minimization in the primal problem and maximization in the dual one with the linear constraint that the duality gap should be zero. This results in the following system of linear inequalities and equations:

$$Ax \leq b; \quad A^T y = c; \quad y \leq 0; \quad c^T x = b^T y \tag{7.4.2}$$

with unknowns $x \in \mathbf{R}^n$ and $y \in \mathbf{R}^m$. The Duality Theorem in Linear Programming says that the original problem (7.4.1) is solvable if and only if (7.4.2) is, and if it is the case, then x-components of the solutions to (7.4.2) are exactly the optimal solutions to (7.4.1).

Assembling x and y in a single vector of variables u, representing u as the difference of new *nonnegative* variable vectors v and w and, finally, assembling v and w into single vector z (all these manipulations are standard tricks in LP), we we can rewrite (7.4.2) equivalently as the problem

$$\text{find} \;\; z \in \mathbf{R}^N : \;\; Qz \leq q, \;\; z \geq 0; \tag{7.4.3}$$

here $N = 2(n + m)$, and Q, q are certain matrix and vector with rational entries coming, in an evident fashion, from the initial data c, b, A. Note that the bit size L_1 of the data in (7.4.3) is of order of the bit size L of the initial data in (7.4.1); in fact $L_1 \leq 10L$.

It is convenient to pass from problem (7.4.3) with rational data to an equivalent problem with integer data; to this end it suffices to compute the common denominator of all the fractions involved into (7.4.3) and to multiply all the data by this common denominator. As a result, we come to the problem

$$\text{find} \;\; x \in \mathbf{R}^N : \;\; Rz \leq r, \;\; z \geq 0; \tag{7.4.4}$$

the properties of this problem are as follows:

- (7.4.3) is equivalent to the LP program (7.4.1): both the problems are solvable/unsolvable simultaneously; if they are solvable, then any solution to (7.4.4) can be converted in an explicit and simple manner into optimal solution to (7.4.1);

- the data in (7.4.4) are integer, and their bit size L_2 is bounded by a polynomial (something like $200L^2$) of the bit size L of the initial data [4)].

It follows that *if we were able to solve in polynomial time the Feasibility problem (7.4.4), we would get, as a byproduct, a polynomial algorithm for the initial problem (7.4.1).*

Step 2: from Feasibility to Solvability

Instead of attacking the Feasibility problem (7.4.4) (where we are asked to check whether (7.4.4) is solvable and if it is the case – are asked to point out a solution), let us for the time being restrict our goals with the following relaxed solvability version of (7.4.4):

(S) *given the data of (7.4.4), detect whether the problem is solvable.*

The simplification, as compared to (7.4.4), is that now we are not obliged to point out an explicit solution to a solvable instance.

Problem (S) is the problem which is solved in polynomial time by the Ellipsoid method. This is done as follows.

- **From (S) to (S′).** Problem (S) clearly is equivalent to the following optimization problem: let
$$f(z) = \max\left[(Rz)_1 - r_1|; (Rz)_2 - r_2; ...; (Rz)_M - r_M|\right]$$
(M is the number of rows in R) be the residual function of the system of inequalities $Rz \leq r$; this function clearly is nonnegative at a nonnegative z if and only if z is a feasible solution to (7.4.4). Consequently, (S) is exactly the following problem:

  (S′): detect whether $\min_{z\in\mathbf{R}^N, z\geq 0} f(z) \leq 0$

  ((S) is solvable if and only if the answer in (S′) is "yes").

- **From (S′) to (S″).** The next step is given by the following simple

  Lemma 7.4.1 *The answer in* (S′) *is "yes" if and only if it is "yes" in the following problem*

  (S″) detect whether $\min_{z:0\leq z_i\leq 2^{L_2},\, i=1,...,N} f(z) \leq 0$

  L_2 being the bit length of the data in (S).

  Proof. If the answer in (S″) is "yes", then, of course, the answer in (S′) also is "yes". It remains to prove, therefore, that if the answer in (S′) is "yes" (or, which is the same, if (S) is solvable), then the answer in (S″) also is "yes". This is easy. The solution set of (S), being nonempty, is a nonempty closed convex polyhedral set (here and in what follows I use the standard terminology of Convex Analysis; this terminology, along with all required facts, was presented in the course Optimization I); since (S) involves nonnegativity

[4] the latter property comes from the fact that the common denominator of the entries in (7.4.3) is an integer of bit size at most $L_1 \leq 10L$; therefore when passing from (7.4.3) to (7.4.4), we increase the bit size of each entry by at most $10L$. Since even the total bit size of the entries in (7.4.3) is at most $10L$, the bit size of an entry in (7.4.4) is at most $20L$; and there are at most $10L$ entries. All our estimates are extremely rough, but it does not matter – all we are interested in is to ensure polynomiality of the bit size of the transformed problem in the bit size of the initial one

constraints, this set does not contain lines, and therefore, due to the well-known fact of Convex Analysis, possesses an extreme point $\bar{z}$. From the standard facts on the structure of extreme points of a polyhedral set given by (7.4.4) it is known that the vector z^* comprised of nonzero entries of $\bar{z}$, if such entries exist, satisfy *nonsingular* system of linear equations of the form

$$\bar{R}z^* = \bar{r},$$

where

- $\bar{R}$ is a $k \times k$ nonsingular submatrix in R (k is the # of nonzero entries in $\bar{z}$)
- $\bar{r}$ is certain fragment of the right hand side vector r

(R and r are given by (7.4.4)).

It follows that the entries in z^* are given by the Cramer rules – they are ratios of certain $k \times k$ determinants taken from the $k \times (k+1)$ matrix $[\bar{R}|\bar{r}]$:

$$z_i^* = \frac{\Delta_i}{\Delta_0}.$$

All entries in this matrix are integers, and the total bit size of the entries does not exceed L_2. It follows that all the determinants are, in absolute value, at most 2^{L_2} [5]. Thus, the numerators in the Cramer formulae are $\leq L_2$ in absolute values, while the denominator (being a nonzero integer) is in absolute value ≥ 1. Consequently, $|z_i^*| \leq 2^{L_2}$.

Thus, all nonzero entries in $\bar{z}$ are $\leq 2^{L_2}$ in absolute values. Since $\bar{z}$ is a solution to (S), this is a point where f is nonnegative. We conclude that if the answer in (S) is "yes", then f attains nonpositive value in the box $0 \leq z_i \leq 2^{L_2}$, $1 \leq i \leq N$, so that the answer in (S$''$) also is "yes". ∎

- **(S$''$) as a convex program.** To solve problem (S$''$) is exactly the same as to check whether the optimal value in the optimization program

$$f(z) \to \min \mid z \in G = \{z \in \mathbf{R}^n \mid 0 \leq z_i \leq 2^{L_2},\ i = 1, ..., N\} \tag{7.4.5}$$

is or is not positive. The objective in the problem is easily computable convex function (since it is maximum of M linear forms), and the domain G of the problem is a simple solid - a box. Remark 7.1.1 explains how to imitate the First order and the Separation oracles for the problem; we can immediately point out the initial ellipsoid which contains G (simply the Euclidean ball circumscribed around the cube G). Thus, we are able to solve the problem by the Ellipsoid method. From Theorem 7.2.1 (where one should estimate the quantities $\mathcal{V}$ and $\max_G f - \min_G f$ via L_2; this is quite straightforward) it follows that in

[5] This is a consequence of the Hadamard inequality: the absolute value of a determinant (≡ the volume of the parallelotope spanned by the rows of the determinant) does not exceed the product of the Euclidean lengths of the rows of the determinant (product of the edges of the parallelotope). Consequently, $\log_2|\Delta_i|$ does not exceed the sum of binary logs of the Euclidean lengths of the rows of $[\bar{R}|\bar{r}]$. It remains to note that the binary logarithm of the Euclidean length of an integer vector clearly does not exceed the total bit length of the vector:

$$\frac{1}{2}\log_2(a_1^2 + ... + a_k^2) \leq \frac{1}{2}\log_2[(1+a_1^2)(1+a_2)^2...(1+a_k^2)] = \sum_{i=1}^{k}\frac{1}{2}\log_2[1+a_i^2] \leq \sum_{i=1}^{k}\log_2[1+|a_i|]$$

and the latter expression clearly is $\leq$ the total # of binary digits in integers $a_1, ..., a_k$.

order to approximate the optimal value f^* in (7.4.5) within a prescribed *absolute* accuracy $\nu > 0$, it suffices to perform

$$N_\nu = O(1)N^2[L_2 + \ln \frac{1}{\nu}]$$

steps with at most $O(1)(M+N)N$ arithmetic operations per step, which gives totally

$$\mathcal{M}(\nu) = O(1)N^3(M+N)[L_2 + \ln \frac{1}{\nu}] \tag{7.4.6}$$

arithmetic operations ($O(1)$ here and in what follows are positive absolute constants).

All this looks fine, but in order to detect whether the optimal value f^* in (7.4.5) is or is not nonpositive (i.e., whether (S) is or is not solvable), we should distinguish between two "infinitesimaly close to each other" hypotheses $f^* \le 0$ and $f^* > 0$, which seems to require the exact value of f^*; and all we can do is to approximate "quickly" f^* to a prescribed accuracy $\nu > 0$, not to find f^* exactly.

Fortunately, our two hypotheses are not infinitesimaly close to each other – there is a "gap" between them. Namely, it can be proved that

if the optimal value f^* in (7.4.5) is positive, it is not too small, namely $f^* \ge 2^{-\pi(L_2)}$ with certain polynomial $\pi(\cdot)$ [6].

The latter remark says that to distinguish between the cases $f^* \le 0$ and $f^* > 0$ means in fact to distinguish between $f^* \le 0$ and $f^* \ge 2^{-\pi(L_2)}$; and to this end it suffices to restore f^* within absolute accuracy like $\nu = 2^{-\pi(L_2)-2}$. According to (7.4.6), this requires $O(1)N^3(N+M)[L_2 + \pi(L_2)]$ arithmetic operations, which is not more than a polynomial of L_2 and, consequently, of L (since L_2 is bounded from above by a polynomial of L).

Thus, *we indeed can decide whether f^* is or is not nonpositive or, which is the same, whether* (S) *is or is not solvable, in polynomial in L number of arithmetic operations.*

This is not exactly what we need: our complexity model counts not arithmetic, but bit-wise operations. It turns out, anyhow (the verification is straightforward, although very dull), that *one can apply the Ellipsoid method with inexact arithmetic instead of the exact one, rounding the intermediate results to a polynomial in L number of accuracy digits, and it still allows to restore f^* within required accuracy.* With the resulting inexact computations, the bit-wise overall effort becomes polynomial in L, and the method becomes polynomial.

Thus, (S) *is polynomially solvable.*

[6] The proof (I skip the details) is completely similar to that one of Lemma 7.4.1: we again exploit the fact that f^* is the optimal value in certain LP program with integer data of the polynomial in L_2 total bit size, namely, in the program

$$t \to \min \mid \; t \ge (Pz)_i - p_i,\; i = 1, ..., N,\; 0 \le z_i \le 2^{L_2}.$$

The optimal value in this problem is achieved at an extreme point, and this point, same as in Lemma 7.4.1, is rational with not too large numerators and denominators of the entries. Consequently, the optimal value of t is rational with not too large numerator and denominator, and such a fraction, if positive, of course, is not too close to 0.

From Solvability back to Feasibility

It remains to explain how, given possibility to solve in polynomial time the Solvability problem (S), one could solve in polynomial time the Feasibility problem (7.4.4) and thus – the original problem (7.4.1).

First of all, we solve (S) – we already know how to do it in polynomial time. If the answer in (S) is "no", then (7.4.4) is unsolvable, and we are done. It remains to consider the case when the answer in (S) is "yes"; here we should point out explicitly the solution to the system

$(*_0)$: $\quad Pz \leq p,\ z \geq 0.$

Let us act as follows. Take the first inequality $P_1^T z \leq p_1$ in the system $Pz \leq p$ and make it equality. This will give us a new system $(*_1^+)$ of, basically, the same bit size. as the one of $(*_0)$. Let us check in polynomial time whether this new system is solvable. If it is so, let $(*_1)$ be the system obtained from $(*_0)$ by replacing the first inequality with the equality; note that this system is solvable, and all solutions to it are solutions to $(*_0)$. If $(*_0^+)$ is unsolvable, it means that the hyperplane $\Pi = \{P_1^T z = p_1\}$ does not intersect the solution set of $(*_0)$. Since the latter set is nonempty and convex, the only possibility for the hyperplane Π not to intersect this set is when the inequality $P_1^T z \leq p_1$ is <u>redundant</u> in system $(*_0)$ – when eliminating this inequality from the system, we do not vary the solution set. If it is the case, let us define $(*_1)$ as the system obtained from $(*_0)$ by eliminating the first inequality.

Let us look what we get. Solving in polynomial time problem (S) associated with the system $(*_1^+)$ with basically the same bit size as the one of (7.4.4), we have updated the initial system of inequalities $(*_0)$ into a new system $(*_1)$; this new system is solvable, and every solution to it is a solution to $(*_0)$ as well, and the new system has one inequality less than the initial system.

Now let us apply the same trick to system $(*_1)$, trying to make the equality the second inequality $P_2^T z \leq p_2$ of the initial system; as a result, we will "kill" this second inequality – either make it the equality, or eliminate it at all – and all solutions to the resulting system $(*_2)$ (which for sure will be solvable) will be solutions to $(*_1)$ and, consequently, to the initial system $(*_0)$.

Proceeding in this manner, we in M steps (M is the row size of P) will "kill" all the inequalities $Pz \leq p$ - some of them will be eliminated, and some - transformed into equalities. Now let us kill in in the same fashion the inequalities $z_i \geq 0$, $i = 1, ..., N$. As a result, in N more steps we shall "kill" <u>all</u> inequalities in the original system $(*_0)$, including the nonnegativity ones, and shall end up with a system of *linear equations*. According to our construction, the resulting system $(*_{M+N})$ will be solvable and every solution to it will be a solution to $(*_0)$.

It follows that to get a solution to $(*_0)$ it remains to solve the resulting solvable system $(*_{M+N})$ of linear equations by any standard Linear Algebra routine (all these routines are polynomial).

Note that the overall process requires to solve $N + M$ Solvability problems of, basically, the same bit size as (7.4.4) and to solve a system of linear equations, again of the same bit size as (7.4.4); thus, the overall complexity is polynomial in the size of (7.4.4).

The proof of polynomial time solvability of LP over Rationals – which might look long and dull – in fact uses absolutely simple and standard arguments; the only nonstandard – and the key one – argument is the Ellipsoid method.

7.4.3 More History

Although the Ellipsoid method as applied to LP turns out to be polynomial – and therefore, theoretically, much more efficient than the non-polynomial Simplex method – in practical computations the Simplex (which – it is a kind of mystery why – never works according to its disastrous theoretical worst case efficiency bound) is incredibly better than the Ellipsoid method, so that it makes absolutely no sense to think that the Ellipsoid method can be competitive with Simplex as a practical LP tool. Nevertheless, the "complexity approach" to LP proved itself to be fruitful not only in theory. In 1984, N. Karmarkar developed a new polynomial time algorithm for LP – the first of the interior point methods which are in great fashion now – and this method turned out to be quite competitive with the Simplex method in practical computations, not speaking about great theoretical advantage of polynomiality shared by the method of Karmarkar.

Lecture 8

Constrained Minimization: Elements of Theory

We start investigating numerical methods for *constrained optimization*, and this lecture will be mainly devoted to the summary of notation, terminology and optimality conditions. Those of you who passed through the course "Optimization I" should be familiar with the material, even in a little bit more general form; therefore in what follows I skip part of the proofs. I think, anyhow, that it makes sense to summarize our expected knowledge.

8.1 Nonlinear Programming problems

A general Nonlinear Programming program is the problem

$$\begin{array}{rcl} \text{(P)}: & & \\ \text{minimize} & & f(x) \\ \text{subject to} & & \\ \text{[equality constraints:]} & h(x) \equiv \begin{pmatrix} h_1(x) \\ h_2(x) \\ \cdots \\ h_m(x) \end{pmatrix} & = \quad 0, \\ \text{[inequality constraints:]} & g(x) \equiv \begin{pmatrix} g_1(x) \\ g_2(x) \\ \cdots \\ g_k(x) \end{pmatrix} & \leq \quad 0. \end{array}$$

Here

- the design vector x varies over certain $\mathbf{R}^n$;
- the objective $f(x)$, the equality constraints $h_i(x)$, $i = 1, ..., m$, and the inequality constraints $g_j(x)$, $j = 1, ..., k$, are smooth (at least twice continuously differentiable) functions on $\mathbf{R}^n$.

The standard terminology related to problem (P) is as follows:

- a feasible solution to (P) is any point $x \in \mathbf{R}^n$ which satisfies all the equality and inequality

constraints of the problem[1)]. (P) is called feasible, is it has at least one feasible solution; the set of all these feasible solutions is called the feasible set of the problem.

- For a feasible solution x, some of the inequality constraints can be satisfied at x as *strict* inequalities (i.e., $g_j(x) < 0$), and some – as equalities: $g_j(x) = 0$. The inequality constraints of this latter type are called active at x^*, and those of the former type – nonactive. The reason for this terminology is clear: for a nonactive at x inequality constraint we have $g_j(x) < 0$; from continuity of g_j it follows that the constraint is satisfied in a neighbourhood of x as well; in other words, such an inequality *locally* does not participate in the problem: it makes no influence on feasibility/infeasibility of candidate solutions close to x (of course, "far" from x such an inequality can also come into the play). In contrast to this, an active at x inequality cannot be neglected even in a small neighbourhood of the point: normally, it influences feasibility/infeasibility of close to x candidate solutions.

- If among *feasible* solutions of (P) there exists (at least) one x^* with the smallest value of the objective:

  $$f(x^*) \leq f(x) \text{ for every } x \text{ feasible for (P)},$$

  then (P) is called solvable, and any (feasible!) x^* with the indicated property is called a (globally) optimal solution to (P);

- If a feasible solution x^* is "the best" – with the smallest value of the objective – among all *feasible* solutions *close enough* to x^*:

  $$\exists r > 0: \quad f(x^*) \leq f(x) \text{ for every } x \text{ feasible for (P) and such that } |x - x^*| < r,$$

  then x^* is called locally optimal solution to (P).

Among general Nonlinear Programming problems (P), the following two particular cases should be mentioned:

- equality constrained problems (only equality constraints $h_i(x) = 0$ and no inequality constraints $g_j(x) \leq 0$ are present),

  and

- inequality constrained problems (vice versa, only inequality constraints $g_j(x) \leq 0$ and no equality constraints $h_i(x) = 0$ are present).

Note that there are various possibilities to convert a general problem (P) into an equivalent *equality constrained* problem; e.g., you can associate with every inequality constraint

$$g_j(x) \leq 0$$

additional slack variable s_j and replace this inequality with the equality

$$h_{m+j}(x, s) = g_j(x) + s_j^2 = 0;$$

[1] note that we use the same word "constraints" for functions h_i and g_j and for relations like $h_i(x) = 0$, $g_j(x) \leq 0$ involving these functions. This saves words and never causes misunderstandings; e.g., it is absolutely clear that when saying "the gradients of the constraints", we speak about the functions (only a function, not a relation, may have a gradient!), while saying "such and such x satisfies such and such constraint", we speak about relations, not functions.

adding these equalities to the list of equality constraints of the original problem, you come to a new problem

$$f(x) \to \min \mid\ h_i(x) = 0,\ i = 1, ..., m,\ h_i(x, s) = 0,\ i = m+1, ..., m+k$$

with $m + k$ equality constraints and extended vector of design variables $(x, s_1, ..., s_k) \in \mathbf{R}^{n+k}$. This equality constrained problem (P′) clearly is equivalent to the original problem (P), namely:

- a point x is feasible for (P) if and only if it can be extended, by a properly chosen $s = (s_1, ..., s_k)$ to a feasible solution (x, s) for (P′);
- a point x^* is globally/locally optimal solution of (P) if and only if it can be extended, by properly chosen $s^* = (s_1^*, ..., s_k^*)$, to a globally, respectively, locally optimal solution to (P′).

It follows that in principle we could restrict our considerations with equality constrained problems only. This, anyhow, makes no sense, since when passing from (P) to (P′), we extend the dimension of the design vector which not always is desirable. There are also cases when the transformation (P) $\mapsto$ (P′) "kills" nice structure of the original problem. The most evident example here is *Convex Programming* known to us from the previous lecture. In our current format, (P) is a convex program if it is inequality constrained and all the objective and the constraints are convex functions; we also could allow equality constraints, but only linear one, since a nonlinear equality constraint, generally speaking, defines a "curved" surface, so that the feasible set of the problem more ore less inavoidable becomes nonconvex, in contrast to what should be the case in Convex Programming. When passing from a convex program (P) to an equivalent equality constrained program, we loose convexity – as it was just explained, a problem with nonlinear equality constraints (and these are exactly the constraints we get after conversion) normally is nonconvex; as a result, in the case in question we loose possibility to solve the problem by efficient Convex Programming methods.

Thus, although possible theoretically, it makes no sense to restrict ourselves once for ever with the equality constrained problems only. At the same time, it would be too complicated to speak all the time about the most general case where both inequality and equality constraints are present. What people normally do (and what we shall do as well) is a kind of compromise: usually we shall consider separately the equality and the inequality constrained cases. As about the general "mixed" case, you should remember that all said about the equality constrained case can be extended, via the aforementioned transformation (P)$\mapsto$(P′), onto this mixed case.

8.2 Geometry of Constraints and Optimality Conditions

A good way to imagine an optimization problem is to think of it as of the problem of minimizing the objective over the feasible set, let it be called S. Let us look what is this feasible set geometrically, and what are the conditions for a point from this set to be a locally optimal solution. This latter issue is of great importance for us, since these are the optimality conditions which govern many of the algorithms to be described. These algorithms check whether at the current iterate the optimality condition is satisfied, and if it is violated, update the iterate, guided by the "residuals" in the conditions. This is exactly as in unconstrained minimization: the necessary optimality condition is given by vanishing of the gradient of the objective at a point (the Fermat rule); the "residual" in this condition is the gradient itself. And the gradient is exactly the quantity which governed all our actions in unconstrained minimization.

8.2.1 Equality constrained case

Surfaces and tangent planes

Consider the case of an equality constrained problem (P); here the feasible set is given by a number of smooth scalar equalities:

$$S = \{x \in \mathbf{R}^n \mid h_i(x) = 0,\, i = 1, ..., m\}; \tag{8.2.1}$$

of course, we can replace this system of scalar equations by a single vector equation:

$$S = \{x \in \mathbf{R}^n \mid h(x) \equiv \begin{pmatrix} h_1(x) \\ h_2(x) \\ \cdots \\ h_m(x) \end{pmatrix} = 0\}. \tag{8.2.2}$$

Geometrically, a system of scalar equations defines a *surface* in $\mathbf{R}^n$, so that an equality constrained program is to minimize a given objective over a given surface. The word "surface" in the above sentence, anyhow, has no precise mathematical meaning, and the sentence itself appeals to geometric intuition, not to detailed understanding. Indeed, consider the equation

$$h(x) \equiv x_1^2 + x_2^2 - 1 = 0 \tag{8.2.3}$$

in 2D plane; everybody knows that this equation defines the unit circumference centered at the origin; this doubtless is quite respectable "surface", better to say curve. Now consider the equation

$$h(x) \equiv x_2^2 - x_1^2 = 0 \tag{8.2.4}$$

which looks as good as the previous one. But noticing that

$$h(x) = (x_2 - x_1)(x_2 + x_1),$$

we discover that the equation $h(x) = 0$ defines the set comprised of two straight lines:

$$l_1 = \{x \mid x_2 = x_1\} \text{ and } l_2 = \{x \mid x_2 = -x_1\}.$$

In a neighbourhood of the origin, which is the intersection point of the lines, is this pair of lines "one surface" or "two of them"? Geometrically, of course, two! Let me add that *an arbitrary closed subset in $\mathbf{R}^n$ can be represented as the set of zeros of smooth (infinitely many times continuously differentiable) scalar function*, so that it makes no geometric sense at all to think that "a smooth equality constraint defines a surface" – a general type closed subset in $\mathbf{R}^n$ – and even on the one-dimensional axis – may be absolutely pathologic from the geometric viewpoint, and in fact no geometric intuition can be applied to such a set.

We see that in order to get a "geometrically tractable" feasible sets and to avoid singularities which, generally speaking, make it impossible to say something reasonable about the set and, consequently, about the optimization problem associated with the set, we should impose on the constraints some additional, compared to smoothness, regularity assumptions. The most evident geometrical property of a "respectable" surface is that it should admit, at every point from it, a "tangent plane" – be well-approximated by a plane.

The natural candidate onto the role of the tangent plane to the surface S at a point $x \in S$ is the set H_x of those points y which satisfy the system of *linear* equations obtained by linearization at x of the smooth equations defining S. These linearizations are

$$\bar{h}_i[y] = h_i(x) + (y - x)^T \nabla h_i(x) = (y - x)^T \nabla h_i(x)$$

(the second equality comes from $h_i(x) = 0$; recall that $x \in S$). Consequently,

$$T_x = \{y \mid (y-x)\nabla h_i(x) = 0,\ i = 1, ..., m\} = \{y \mid \nabla h(x)(y-x) = 0\}, \tag{8.2.5}$$

$$\nabla h(x) = \begin{pmatrix} [\nabla h_1(x)]^T \\ \cdots \\ [\nabla h_m(x)]^T \end{pmatrix}$$

being the $m \times n$ derivative of the vector-valued mapping $h(\cdot)$ at x.

Whether this candidate indeed is good, it depends on the situation. E.g., for the circumference S given by (8.2.3) the indicated equation defines, for $x = (1, 0)^T$,

$$T_x = \{y \mid y_1 = 1\};$$

this indeed is the equation of the geometric tangent line to the circumference at the point in question.

For the pair of lines S given by equation (8.2.4) relation (8.2.5) applied at the point $x' = (1, 1)^T$ results in

$$T_{x'} = \{y \mid y_1 = y_2\};$$

since in a small enough neighbourhood of x' S coincides with the line $x_1 = x_2$, this is fine tangent line – locally it even not approximates, but reproduces exactly the surface! But at the point $x'' = (0, 0)^T$, which also belongs to S, relation (8.2.5) results in

$$T_{x''} = \mathbf{R}^2,$$

which makes no geometrical sense[2)].

Thus, there should be certain *regularity assumptions* which imply that (8.2.5) indeed is, geometrically, a tangent plane to S at x; and since the existence of the tangent plane is, geometrically, the most basic property of a "respectable surface", these assumptions imply, as a byproduct, that S itself "is a surface". The indicated assumptions are given by one of the most general and important theorems of Analysis – the Implicit Function Theorem; they simply say that the set of gradients of the constraints at x should be linearly independent.

Definition 8.2.1 *Let*

$$h(x) \equiv \begin{pmatrix} h_1(x) \\ h_2(x) \\ \cdots \\ h_m(x) \end{pmatrix} = 0 \tag{8.2.6}$$

be a system of equality constraints, and let $\bar{x}$ be a solution to the system. This solution is called regular for the system, if the vectors

$$\nabla h_i(\bar{x}),\ i = 1, ..., m,$$

are linearly independent, or, which is the same, if the $m \times n$ matrix $\nabla h(\bar{x})$ with the rows being the gradients of the constraints $h_1, ..., h_m$ at $\bar{x}$ has full row rank m.

[2]You may think that in the latter case everything also is ok: in a neighbourhood of the origin, which is the intersection of the two straight lines comprising S, S is not a "respectable" geometrical surface, so that there is nothing strange that (8.2.5) fails to represent the tangent line to S at the point – no such a line exists at all! Unfortunately, even a quite respectable surface can be defined by a badly chosen system of equality constraints, and in this case (8.2.5) fails to represent a geometrically well-defined object. Look what happens if we represent circumference S by the equation

$$h(x) \equiv (x_1^2 + x_2^2 - 1)^2 = 0;$$

applying (8.2.5), you will get a crazy answer $T_x = \mathbf{R}^2$ for every $x \in S$!

The next statement says that if x is a regular solution for system (8.2.6), then (8.2.5) indeed defines the tangent plane to S at the point (and also explains what the words "tangent plane" mean).

Theorem 8.2.1 [Theorem on Tangent Plane] *Let S be given by (8.2.6), and let $x \in S$ be a regular solution of the corresponding system of equations with twice continuously differentiable constraints h_i, $i = 1, ..., m$. Then the plane T_x given by (8.2.5) is the tangent plane to S at x, namely, there exists constant C such that*

- *the distance from an arbitrary point $x' \in S$ of the surface to T_x is of the second order of magnitude as compared to the distance from x' to x:*

$$\forall(x' \in S) \quad \exists(y' \in T_x): \quad |x' - y'| \leq C|x' - x|^2;$$

- *vice versa, the distance from an arbitrary point $y' \in T_x$ of the tangent plane to the surface is of the second order of magnitude as compared to the distance from y' to x:*

$$\forall(y' \in T_x) \quad \exists(x' \in S): \quad |x' - y'| \leq C|y' - x|^2.$$

The Theorem provides us, among other, with exact translation to rigorous language of the geometric notion of a "plane tangent to a surface at a point x of the surface" – roughly speaking, the plane should approximate the surface locally within accuracy which is "infinitesimal of highest order" as compared to the distance to x.

First order optimality condition

Now let us ask ourselves the following crucial question:

(?) *given an equality constrained problem*

$$\text{(P)} \quad f(x) \to \min \mid h(x) \equiv \begin{pmatrix} h_1(x) \\ \dots \\ h_m(x) \end{pmatrix} = 0 \tag{8.2.7}$$

– the problem of minimizing smooth objective f over the surface S given by equality constraints, and given a feasible solution x^* to the problem, what are the conditions – necessary, sufficient, or both – for x^* to be locally optimal?

The first idea which comes to our mind is as follows. Assume that x^* is a *regular* point of the system of constraints in question, so that the plane T tangent to the surface S at the point x^* is well-defined. This plane is a "tight" approximation of the surface around x^*, and it is natural to conjecture that x^* should be optimal solution to the "approximate problem" where S is replaced with T, and the actual objective is replaced with its linearization at the point x^* – with the linear form $\bar{f}(x) = f(x^*) + (x - x^*)^T \nabla f(x^*)$. Let us believe in this conjecture and look what are the consequences. If x^* is local minimizer of the linear form $\bar{f}$ in the plane T – this plane, geometrically, is certain $\mathbf{R}^k$ – then, by the usual Fermat rule, the derivative of $\bar{f}$ taken at x^* in every direction along the plane T should be zero:

$$e^T \nabla f(x^*) = 0 \quad \forall e : x^* + e \in T.$$

We know what is the tangent plane T: according to (8.2.5), it is given by the system of linear equations

$$(y - x^*)^T \nabla h_i(x^*) = 0, \;\; i = 1, ..., m,$$

or, which is the same, by the system

$$\nabla h(x^*)(y - x^*) = 0.$$

In particular, a direction e looks along T (i.e., $x^* + e \in T$) if and only if

$$e^T \nabla h_i(x^*) = 0,\, i = 1, ..., m \quad [\Leftrightarrow \nabla h(x^*)e = 0].$$

Thus, our conjecture is as follows:

if x^* is locally optimal in (P), then $\nabla f(x^*)$ should be orthogonal to the linear subspace

$$L = T - x^* = \{e \mid \nabla h(x^*)e = 0\}.$$

Now, from Linear Algebra it is known that a vector p (in our case – $\nabla f(x^*)$) is orthogonal to the linear subspace given by the system

$$Ae = 0$$

(in our case $A = \nabla h(x^*)$) if and only if the vector belongs to the image space of the matrix A^T: $p + A^T\lambda = 0$ for certain vector λ, or, which is the same, that p is linear combination of the rows of A, the entries of $-\lambda$ being the coefficients of the combination. It follows that our conjecture is equivalent to the following assertion:

If x^* is locally optimal for (P), then there exist m-dimensional vector λ^* (the vector of *Lagrange multipliers*) such that

$$\nabla f(x^*) + [\nabla h(x^*)]^T \lambda^* \equiv \nabla f(x^*) + \sum_{i=1}^{m} \lambda_i^* \nabla h_i(x^*) = 0 \tag{8.2.8}$$

(note that the rows of $\nabla h(x^*)$ are exactly the gradients of h_i).

Note that relation (8.2.8) can be equivalently written down as the equation

$$\nabla_x L(x^*; \lambda^*) = 0,$$

where

$$L(x; \lambda) = f(x) + \sum_{i=1}^{m} \lambda_i h_i(x)$$

is the Lagrange function of problem (P).

All our conclusions were based upon our initial conjecture: if x^* is locally optimal in (P), then it is also locally optimal in the approximate problem where the feasible surface S is replaced with the tangent plane T, and the objective f – with its linearization $\bar{f}$. This conjecture, and consequently, all its conclusions turn out to be true:

Theorem 8.2.2 [First Order Necessary Optimality conditions for Equality Constrained Problem – the Lagrange rule]
Let x^* be a locally optimal solution of the equality constrained problem (8.2.7), and let it be a regular point for the system of equality constraints of the problem. Then the following two equivalent to each other properties take place:

- *The directional derivative of f taken at x^* in every direction along the plane*

$$T = \{y \mid \nabla h(x^*)(y - x^*) = 0\}$$

tangent to the surface at x^* is zero:

$$\nabla h(x^*)e = 0 \Rightarrow e^T \nabla f(x^*) = 0; \tag{8.2.9}$$

- *there exists uniquely defined by x^* vector of Lagrange multipliers λ_i^*, $i = 1, ..., m$, such that*

$$\nabla f(x^*) + \sum_{i=1}^{m} \lambda_i^* \nabla h_i(x^*) = 0,$$

or, which is the same,

$$\nabla_x L(x^*; \lambda^*) = 0,$$

$$L(x; \lambda) = f(x) + \sum_{i=1}^{m} \lambda_i h_i(x) \equiv f(x) + \lambda^T h(x)$$

being the Lagrange function of the problem.

Proof will be outlined only.

What we should prove is the implication (8.2.9), i.e., the first of the two equivalent formulations of the necessary optimality conditions, since the second of them is obtained from the first one by the standard Linear Algebra arguments (in particular, uniqueness of λ^* comes from regularity of x^*, i.e., from linear independence of $\nabla h_i(x^*)$: since these vectors are linearly independent, the gradient of the objective may have at most one representation as a linear combination of the gradient of the constraints).

Let us prove (8.2.9) by contradiction. Namely, assume that there exists a direction e parallel to the tangent plane:

$$\nabla h(x^*)e = 0$$

and non-orthogonal to $\nabla f(x^*)$. By replacing e, if necessary, with $-e$ we can assume that e is a descent direction of the objective:

$$e^T \nabla f(x^*) < 0.$$

Now consider the ray

$$y_t = x^* + te;$$

since e is parallel to T, this ray belongs to the tangent plane. We have

$$\frac{d}{dt}|_{t=0} f(y_t) = e^T \nabla f(x^*) \equiv -\alpha < 0,$$

whence for all small enough nonnegative t, say, $0 \leq t \leq \delta$,

$$f(y_t) \leq f(x^*) - \frac{\alpha}{2} t. \tag{8.2.10}$$

Since x^* is regular for the system of constraints of the problem, by Theorem 8.2.1 there exists, for every t, a point $x_t \in S$ such that

$$|y_t - x_t| \leq C|y_t - x^*|^2 = C_1 t^2, \;\; C_1 = C|e|^2;$$

from this inequality, x_t (and, of course, y_t) converge to x^* as $t \to +0$. Since f is continuously differentiable, it is Lipschitz continuous in a neighbourhood of x^*, so that for small enough positive t we have

$$|f(x_t) - f(y_t)| \leq C_2|x_t - y_t| \leq C_1C_2t^2.$$

Combining this inequality with (8.2.10), we obtain

$$f(x_t) \leq f(x^*) - \frac{\alpha}{2}t + C_1C_2t^2$$

for all small enough positive t. The right hand side of the latter inequality is strictly less than $f(x^*)$, provided that t is positive and small enough; we see that for indicated t x_t is a feasible solution of the problem (recall that $x_t \in S$ by construction) with better than that one of x^* value of the objective. As $t \to +0$, these better solutions converge to x^*; consequently, x^* is not locally optimal, and this is the desired contradiction. ∎

The main advantage of the First Order Necessary Optimality Condition is that it allows to reduce the problem of identifying all candidates to the role of *regular for the system of constraints* locally optimal solutions to (8.2.7) to solving a *square system of (nonlinear) equations*. Indeed, such a solution should, for certain $\lambda = \lambda^*$, satisfy the system of n equations $\nabla_x L(x;\lambda) = 0$, and, besides this, satisfy the m equations $h_i(x) = 0$, $i = 1, ..., m$ – the equality constraints of the problem. Noticing that $h_i(x) = \frac{\partial}{\partial\lambda_i}L(x,\lambda)$, we can write down this system of $n+m$ equations with $n+m$ unknowns x and λ in very nice symmetric form:

$$\nabla_x L(x;\lambda) = 0, \tag{8.2.11}$$

$$\nabla_\lambda L(x;\lambda) = 0. \tag{8.2.12}$$

Second Order Optimality conditions

Same as in the unconstrained case with the Fermat rule, the First Order Optimality conditions for equality constrained problems given by the Lagrange rule are not quite satisfactory. Namely, these are only necessary and not sufficient conditions for local optimality (and how could they be sufficient? They do not distinguish between (constrained) local minima and local maxima of f!). In the unconstrained case, the Fermat rule is accompanied by the Second Order Optimality condition:

- [necessary part] if a critical point x^* of f (i.e., satisfying the Fermat rule $\nabla f(x^*) = 0$) is local minimum of f, then the Hessian of f at x^* is positive semidefinite:

$$e^T\nabla^2 f(x^*)e \geq 0 \quad \forall e \in \mathbf{R}^n.$$

- [sufficient part] if a critical point x^* of f is such that the Hessian of f at x^* is positive definite:

$$e^T\nabla^2 f(x^*)e > 0 \quad \forall 0 \neq e \in \mathbf{R}^n,$$

then x^* is local minimizer of f.

One could think that the extension of these second order conditions onto the equality constrained case if as follows: at a locally optimal solution, the Hessian of the objective should be positive semidefinite along the tangent plane, and if it is positive definite along this plane and the first order optimality conditions are satisfied, then the point is locally optimal; this idea comes from

considering the approximate version of the problem where the actual feasible surface is replaced with its tangent plane at the point in question. *The indicated conjecture is not true at all*, and it is clear why: the second order optimality conditions come from analyzing the second-order approximation of the objective on the feasible set. When this set is "curved", as it is the case in equality constrained problems with nonlinear equality constraints, the "curvature" of the feasible surface influences the second-order behaviour of the objective on the surface, and we are unable to get correct conclusions looking at the behaviour of the objective on the tangent plane (the latter plane is only first-order approximation of the feasible surface). The correct statement deals not with the objective, but with the Lagrange function, and is as follows:

Theorem 8.2.3 [Second Order Optimality Conditions for Equality Constrained problems] *Let x^* be a feasible solution for the equality constrained problem (8.2.7), let x^* be regular for the system of constraints of the problem, and let*

$$L(x;\lambda) = f(x) + \sum_{i=1}^{m} \lambda_i h_i(x) = f(x) + \lambda^T h(x)$$

be the Lagrange function of the problem. Then

- [Second Order Necessary Optimality condition]

  Let x^* be locally optimal solution of the problem. Then x^* satisfies the First Order Optimality condition:

  $$\exists \lambda^* : \quad \nabla_x L(x^*;\lambda^*) = 0; \quad \nabla_\lambda L(x^*;\lambda^*) = 0 \tag{8.2.13}$$

  and, besides this, the Hessian of the Lagrangian with respect to x, reduced to the tangent plane, is positive semidefinite:

  $$\nabla h(x^*)e = 0 \Rightarrow e^T[\nabla_x^2 L(x^*;\lambda^*)]e \geq 0. \tag{8.2.14}$$

- [Second Order Sufficient Optimality condition]

  Let x^* satisfy the First Order Optimality condition

  $$\exists \lambda^* : \quad \nabla_x L(x^*;\lambda^*) = 0; \quad \nabla_\lambda L(x^*;\lambda^*) = 0 \tag{8.2.15}$$

  and let, besides this, the Hessian of the Lagrangian with respect to x, reduced to the tangent plane, be positive definite:

  $$\nabla h(x^*)e = 0, \ e \neq 0 \Rightarrow e^T[\nabla_x^2 L(x^*;\lambda^*)]e > 0. \tag{8.2.16}$$

  Then x^* is locally optimal solution to the problem.

Please remember that

The "second order part" of the Second Order Optimality condition deals only with the restriction of the Hessian of the Lagrangian onto the tangent plane, not with the entire Hessian; there could be "directions of negative curvature" for the Hessian at x^*, but only transversal (not tangent) to the feasible set ones.

Proof of the Theorem is skipped.

It is worthy to introduce a special name for "good" locally optimal solutions to (8.2.7):

Definition 8.2.2 *A feasible solution x^* to problem (8.2.7) which is*

- *regular for the constraints of the problem*
  and
- *satisfies the Second Order Sufficient Optimality conditions (8.2.15) - (8.2.16)*

is called nondegenerate solution to the problem.

Note that the Second Order Sufficient Optimality conditions (8.2.15) - (8.2.16) involve not only x^*, but also the Lagrange multipliers λ^*, so that it is unclear in advance whether the above definition is "correct", i.e., whether it indeed represents certain property of x^* alone, not one of the pair (x^*, λ^*). Luckily, since x^* is regular for the constraints, the vector of Lagrange multipliers λ^* satisfying (8.2.15), if exists, is uniquely defined by x^*, so that the definition indeed is correct. Note also that, by Theorem 8.2.3, a nondegenerate solution of (8.2.7) is locally optimal.

Lagrange Multipliers and Sensitivity Analysis

The Lagrange multipliers λ^* associated with the equality constrained problem (8.2.7) have very natural interpretation: they are responsible for variations of the optimal value with respect to violations of the constraints. Namely, let us embed problem (8.2.7) into the family of problems

$$(\mathrm{P_b}) \quad f(x) \to \min \mid h(x) \equiv \begin{pmatrix} h_1(x) \\ \dots \\ h_m(x) \end{pmatrix} = b, \tag{8.2.17}$$

b being the m-dimensional "parameter vector" (the original problem corresponds to $b = 0$), and let us look what happens with a (nondegenerate) local solution to the problem when the parameter vector varies; this is called *sensitivity analysis*, and this is a very important phase of processing real-world optimization problems. The answer is given by the following

Theorem 8.2.4 *Consider parameterized family of problems (8.2.17), let x^* be a nondegenerate solution to problem (P_0), and let λ^* be the corresponding vector of Lagrange multipliers from the First Order Optimality condition. Then there exist*

- *small enough neighbourhood U of the point x^* in the space $\mathbf{R}^n$ of the design vectors,*
- *small enough neighbourhood V of the point $b = 0$ in the space $\mathbf{R}^m$ of the parameters,*
- *continuously differentiable functions*

$$x^*(b) : V \to U \subset \mathbf{R}^n; \;\; \lambda^*(b) : V \to \mathbf{R}^m \quad [x^*(0) = x^*, \lambda^*(0) = \lambda^*]$$

such that

- *whenever b is close enough to 0, namely, whenever $b \in V$, $x^*(b) \in U$ is a nondegenerate locally optimal solution to the "perturbed" problem (P_b), $\lambda^*(b)$ being the corresponding vector of Lagrange multipliers; $x^*(b)$ is the only point in U which satisfies the First Order Necessary Optimality condition in perturbed problem;*

- *for $b \in V$, $-\lambda_i^*(b)$ are the derivatives of the "local optimal value"*

$$f^*(b) \equiv f(x^*(b))$$

with respect to b:

$$\nabla_b f^*(b) = -\lambda^*(b). \tag{8.2.18}$$

Proof is given by the more or less straightforward application of the Implicit Function Theorem; I skip it here (it is outlined in the optional exercise to the lecture).

8.2.2 General case

Now consider the case when the problem includes both equality and inequality constraints, so that it is of the form

$$f(x) \to \min \mid x \in F \subset \mathbf{R}^n, \tag{8.2.19}$$

and the feasible set is given by

$$F = \{x \in \mathbf{R}^n \mid h(x) = 0,\ g(x) \leq 0\}, \quad h(x) = \begin{pmatrix} h_1(x) \\ \dots \\ h_m(x) \end{pmatrix},\ g(x) = \begin{pmatrix} g_1(x) \\ \dots \\ g_k(x) \end{pmatrix}. \tag{8.2.20}$$

Geometrically, the feasible set is the intersection of the already known to us *surface* S given by the equality constraints and the *domain* G given by the inequalities (normally, a system of smooth inequalities defines a domain – a set with a nonempty interior; on this interior the inequalities are satisfied as strict – $<$ instead of $\leq$, and the boundary is characterized by the fact that at least one of the inequalities becomes equality). As in the equality constrained case, in order for the feasible set to be "respectable" – to fit geometric intuition and to enable us to characterize solutions to the problem – it should satisfy reasonable regularity assumptions. The most convenient for us assumption of this type is given by the following modification of Definition 8.2.1:

Definition 8.2.3 [Regular point of a general system of constraints] *Let x be a point satisfying system (8.2.20) of equality and inequality constraints, and let*

$$I(x) = \{j : g_j(x) = 0\}$$

be the set of indices of the inequality constraints <u>active</u> at x. x is called <u>regular</u> for the system of constraints in question, if the set comprised of

- *the gradients $\nabla h_i(x)$, $i = 1, ..., n$, of all the equality constraints*
- *the gradients $\nabla g_j(x)$, $j \in I(x)$, of the inequality constraints active at x,*

all the gradients being taken at x, is linearly independent.

The natural conjecture is that in a neighbourhood of a regular feasible solution x, the feasible set F given by (8.2.20) can be well-approximated by the polyhedral set which we get when linearizing all the constraints at x. The latter set is

$$\{y \in \mathbf{R}^n \mid h_i(x) + (y-x)^T \nabla h_i(x) = 0,\ i = 1, ..., m, g_j(x) + (y-x)^T \nabla g_j(x) \leq 0,\ j = 1, ..., k\}. \tag{8.2.21}$$

In this representation we may skip the terms $h_i(x)$ – they are equal to 0, since x is feasible. Moreover, we may drop the linear (in y) inequality constraints coming from the nonactive at x^* original inequality constraints; this, of course, may vary the resulting set, but for sure remains it unchanged in a small enough neighbourhood of x (in such a neighbourhood both the nonactive inequality constraints and their linearizations are satisfied automatically). And since we think only about local structure of the feasible set, we may forget about the nonactive constraints at all. After the non-active inequality constraints are dropped, we may forget about the terms $g_j(x)$ – for active constraints these terms are zeros.

Thus, we may conjecture that in a neighbourhood of a regular feasible solution x the feasible set F should be well-approximated by the set

$$\bar{T}_x = \{y \in \mathbf{R}^n \mid \nabla h(x^*)(y-x) = 0; (y-x)^T \nabla g_j(x) \leq 0,\ j \in I(x)\} \tag{8.2.22}$$

where, as always, $I(x)$ is the set of indices of the inequality constraints active at x. Geometrically, (8.2.22) is a convex cone with the vertex at x; of course, this cone is contained in the tangent plane to the surface S given by the equality constraints[3].

The indicated conjecture indeed is true: from the Implicit Function Theorem one can derive the following

Theorem 8.2.5 *Let x be a regular solution of the system (8.2.20) of equality and inequality constraints. Then the solution set F of the system is, locally at x, well-approximated by the cone (8.2.22), namely, there exists constant C such that*

- *the distance from a point $y' \in \bar{T}_x$ to F is of second order of magnitude as compared to the distance from y' to x:*

$$\forall (y' \in \bar{T}_x) \quad \exists (x' \in F): \quad |y'-x'| \leq C|y'-x|^2;$$

- *the distance from a point $x' \in F$ to $\bar{T}_x$ is of second order of magnitude as compared to the distance from x' to x:*

$$\forall (x' \in F) \quad \exists (y' \in \bar{T}_x): \quad |x'-y'| \leq C|x'-x|^2.$$

First Order Optimality condition

The origin of the Lagrange rule (Theorem 8.2.2) – the first order Optimality condition for equality constrained problems – was simple: this was the optimality condition for the "approximate" problem which we get from the original one when replacing the objective and constraints by their linearizations at a point in question. Similarly, the most natural way to conjecture what should be the first order optimality conditions for a feasible solution x^* in the general problem (8.2.19) - (8.2.20) is to replace the problem with the Linear Programming problem obtained by linearlization of the objective and all the constraints at x^* (i.e., by replacing the actual feasible set by the cone $\bar{T}_{x^*}$, and replacing the actual objective by its linearization at x^*). Looking at the resulting Linear Programming program and applying the Linear Programming Duality Theorem, we come to the condition as follows:

The necessary and sufficient condition for x^* to be optimal solution to the linearized problem

$$f(x^*) + (x-x^*)\nabla f(x^*) \to \min \mid x \in \bar{T}_{x^*}$$

[3] note that the cone should not necessary be pointed, i.e., it may contain straight lines; in particular, it can be an affine set, e.g., the entire tangent plane to S, as it is the case when no inequality constraints are active at x

is existence of Lagrange multipliers λ_i^* for the equality constraints and nonnegative Lagrange multipliers μ_j^* for the active at x^* inequality constraints, such that

$$\nabla f(x^*) + \sum_{i=1}^{m} \lambda_i^* \nabla h_i(x^*) + \sum_{j \in I(x^*)} \mu_j^* \nabla g_j(x^*) = 0.$$

Same as in the equality constrained case, the conjecture which led us the these conclusions, turns out to be true (provided that x^* is regular for the constraints):

Theorem 8.2.6 [Karush-Kuhn-Tucker Necessary Optimality Conditions]
Let x^* be a locally optimal solution to problem (8.2.19) - (8.2.20), and assume that x^* is regular for the constraints (8.2.20) of the problem. Then there exist uniquely defined

- *Lagrange multipliers λ_i^*, $i = 1, ..., n$, for the equality constraints*
- *nonnegative Lagrange multipliers μ_j^* for the active at x^* inequality constraints [$j \in I(x^*) = \{j \leq k : g_j(x^*) = 0\}$]*

such that

$$\nabla f(x^*) + \sum_{i=1}^{m} \lambda_i^* \nabla h_i(x^*) + \sum_{j \in I(x^*)} \mu_j^* \nabla g_j(x^*) = 0. \tag{8.2.23}$$

Proof of the Theorem is skipped.

In the latter theorem, the Lagrange multipliers μ_j^* were associated only with the *active* inequality constraints. We loose nothing when associating these multipliers also with non-active inequality constraints and setting for these constraints $\mu_j^* = 0$. With this convention, the latter Theorem can be reformulated in terms of the Lagrange function

$$L(x; \lambda, \mu) = f(x) + \sum_{i=1}^{m} \lambda_i h_i(x) + \sum_{j=1}^{k} \mu_j g_j(x)$$

of the problem as follows (cf. (8.2.11) - (8.2.12)):

Corollary 8.2.1 *Assume that feasible solution x^* to the constrained problem (8.2.19) - (8.2.20) is regular for the constraints (8.2.20). If x^* is locally optimal, then there exists uniquely defined vector (λ^*, μ^*) of Lagrange multipliers such that (x^*, λ^*, μ^*) is the solution to the corresponding KKT (Karush-Kuhn-Tucker) system of equalities and inequalities with unknowns x, λ, μ:*

equations:			
$\nabla_x L(x; \lambda, \mu)$	$=$	0	[*KKT equation*; *n scalar equations*]
$\nabla_\lambda L(x; \lambda, \mu)$	$=$	0	[*primal feasibility* $h_j(x) = 0$; *m scalar equations*]
$\mu_j g_j(x)$	$=$	0	[*complementary slackness*; *k scalar equations*]
inequalities:			
$g_j(x)$	$\leq$	0	[*primal feasibility*]
μ_j	$\geq$	0, $j = 1, ..., k$	[*nonnegativity of Lagrange multipliers for inequalities*]

Note that the number of unknowns $(n + m + k)$ in the KKT system equals to the number of equality constraints in it; thus, we may expect that the system has finitely many solutions. If we were clever enough to find all these solutions analytically, we could select among them the one with the smallest value of the objective; according to the Corollary, it would be the globally optimal solution to the problem, provided that the latter exists and is a regular point of the system of constraints. Unfortunately, we normally are unable to find explicitly even a particular KKT point of the problem (i.e., a particular solutions to the KKT system).

Second Order Optimality condition: General case

Extension of Theorem 8.2.5 onto the case of problems with both equality and inequality constraints is as follows:

Theorem 8.2.7 [Second Order Optimality conditions for problem (8.2.19) - (8.2.20)]
Let x^* be a feasible solution to problem (8.2.19) - (8.2.20) which is a regular point of the system of constraints (8.2.20), and let $I(x^*) = \{j \mid g_j(x^*) = 0\}$ be the set of indices of the active at x^* inequality constraints. Then

- [Second Order Necessary Optimality condition]

  If x^* is locally optimal solution to the problem, then there exist uniquely defined by x^* Lagrange multipliers λ^*, μ^* such that (x^*, λ^*, μ^*) is a KKT point of the problem, i.e., a solution to the KKT system presented in Corollary 8.2.1. Moreover, the Hessian of the Lagrangian with respect to x $\nabla_x^2 L(x^*; \lambda^*, \mu^*)$ is positive semidefinite on the plane tangent to the surface given by equality and active inequality constraints of the problem:

$$\nabla h_i(x^*)e = 0,\ i = 1, ..., m,\ \nabla g_j(x^*)e = 0,\ j \in I(x^*) \Rightarrow e^T[\nabla_x^2 L(x^*; \lambda^*, \mu^*)e \geq 0.$$

- Second Order Sufficient Optimality condition]

  Assume that there exist Lagrange multipliers λ^*, μ^* such that $(x^*; \lambda^*, \mu^*)$ is a KKT point of the problem, and assume that the Hessian of the Lagrangian with respect to x

$$\nabla_x^2 L(x^*; \lambda^*, \mu^*)$$

  is positive definite on the plane tangent to the surface, the surface being given by equality and those active inequality constraints for which $\mu_j^* > 0$

$$e \neq 0, \nabla h_i(x^*)e = 0,\ i = 1, ..., m,\ \nabla g_j(x^*)e = 0,\ j \in I^*(x^*) \equiv j \in I(x^*) \mid m_j > 0 \Rightarrow$$

$$\Rightarrow e^T[\nabla_x^2 L(x^*; \lambda^*, \mu^*)]e > 0.$$

  Then x^* is locally optimal solution to the problem.

Proof of the Theorem is skipped.

It makes sense to introduce a special name for "good" locally optimal solutions to (8.2.19) - (8.2.20):

Definition 8.2.4 *A feasible solution x^* to (8.2.19) - (8.2.20) which*

- *is regular for the constraints of the problem,*

- *satisfies the Sufficient Second Order Optimality condition from Theorem 8.2.7,*

  and

- *is such that all given by the Second Order Optimality condition Lagrange multipliers μ_j^* for the active at x^* inequality constraints are strictly positive rather than simply nonnegative,*

is called a nondegenerate locally optimal solution to the problem.

The third requirement imposed in the definition might look strange – what for is it? Already the first two of the requirements enforce x^* to be locally optimal! In fact, all three requirements make sense – all of them are needed by

Sensitivity Analysis

Given problem (8.2.19) - (8.2.20), let it be called (P), we can embed it into the $m+k$-parameter family of problems

$$(\mathrm{P_{b,d}}) \quad f(x) \to \min \mid h(x) \equiv \begin{pmatrix} h_1(x) \\ \dots \\ h_m(x) \end{pmatrix} = b,\ g(x) \equiv \begin{pmatrix} g_1(x) \\ \dots \\ g_k(x) \end{pmatrix} \le d; \tag{8.2.24}$$

in this notation, (P) itself is the problem $(\mathrm{P}_{0,0})$.

It turns out that, same as in the case of equality constrained problems (see Section 8.2.1), the Lagrange multipliers are responsible for the derivatives of the "local optimal value" with respect to the parameters b and d of the family (8.2.24):

Theorem 8.2.8 *Consider parameterized family of problems (8.2.24), let x^* be a nondegenerate solution to problem $(\mathrm{P}_{0,0})$, and let λ^*, μ^* be the corresponding vector of Lagrange multipliers, so that $(x^*; \lambda^*, \mu^*)$ is a KKT point of the problem $(\mathrm{P}_{0,1})$. Then there exist*

- *small enough neighbourhood X of the point x^* in the space $\mathbf{R}^n$ of the design vectors,*
- *small enough neighbourhood V of the point $(b,d) = (0,0)$ in the space $\mathbf{R}^m \times \mathbf{R}^k$ of the parameters,*
- *continuously differentiable functions*

$$x^*(b,d) : V \to X \subset \mathbf{R}^n; quad\lambda^*(b,d) : V \to \mathbf{R}^m, \quad \mu^*(b,d) : V \to \mathbf{R}^k$$

$$[x^*(0,0) = x^*, \lambda^*(0,0) = \lambda^*, \mu^*(0,0) = \mu^*]$$

such that

- *whenever (b,d) is close enough to $(0,0)$, namely, whenever $(b,d) \in V$, $x^*(b,d) \in X$ is a nondegenerate locally optimal solution to the "perturbed" problem $(\mathrm{P}_{b,d})$, $\lambda^*(b,d)$, $\mu^*(b,d)$ being the corresponding vectors of Lagrange multipliers; $(x^*(b,d), \lambda^*(b,d), \mu^*(b,d))$ is the only KKT point of problem $(P_{b,d})$ in the set of points (x, λ, μ) with $x \in X$.*
- *for $(b,d) \in V$, $-\lambda_i^*(b,d)$ and $-\mu_j^*(b,d)$ are the derivatives of the "local optimal value"*

$$f^*(b,d) \equiv f(x^*(b,d))$$

  with respect to b_i, d_j, respectively:

$$\nabla_b f^*(b,d) = -\lambda^*(b,d); \quad \nabla_d f^*(b,d) = -\mu^*(b,d). \tag{8.2.25}$$

Assignment # 8 (Lecture 8)

Obligatory problems

Exercise 8.2.1 *A curve in the 2D plane is given by the equation*

$$h(x) \equiv x_1^3 - x_2^2 = 0.$$

Which points of the curve are regular with respect to this equality?

Exercise 8.2.2 *Find the edges of the 3D box of a given surface area possessing, under this restriction, the maximum possible volume.*

Exercise 8.2.3 *Prove that in a inequality constrained problem with convex objective and all the inequality constraints the KKT Optimality Condition (Corollary 8.2.1) is sufficient condition for a feasible solution x^* to be globally optimal.*
Hint: *Take into account that from convexity of the objective and the constraints*

$$\begin{array}{rcl} f(x) & \geq & f(x^*) + (x - x^*)^T \nabla f(x^*) \\ g_j(x) & \geq & g_j(x^*) + (x - x^*)^T \nabla f(x^*) \end{array}$$

for all x.

Exercise 8.2.4 *Apply the Largange rule (Theorem 8.2.2) to the equality constrained problem*

$$f(x) \equiv x_1x_2 + x_2x_3 + x_3x_1 \rightarrow \min \mid x_1 + x_2 + x_3 = 3.$$

How many solutions to the Lagrange system (8.2.11) - (8.2.12) you have got? Which of them, you think, are locally optimal solutions? To answer the latter question, you may use the Second Order Optimality condition.

Optional problems

The goal of the below exercise is to prove the Sensitivity Analysis Theorem 8.2.8. To this end let us start with the formulation of one of the most fundamental theorems of Calculus:

The Implicit Function Theorem. *Let U' be a neighbourhood of a point $u^* \in \mathbf{R}^N$, V' be a neighbourhood of a point $v^* \in \mathbf{R}^M$, and let*

$$\Phi(u, v) : U' \times V' \rightarrow \mathbf{R}^N$$

be $s \geq 1$ times continuously differentiable vector-valued function of $u \in U'$ and $v \in V'$ such that

$$\Phi(u^*, v^*) = 0$$

and the $N \times N$ matrix

$$\Phi'_u(u^*, v^*) \equiv \frac{\partial}{\partial u}|_{u=u^*, v=v^*} \Phi(u, v)$$

is nonsingular.

Then there exist

- *a neighbourhood $U \subset U'$ of u^* in $\mathbf{R}^N$ and a neighbourhood $V \subset V'$ of v^* in $\mathbf{R}^M$,*

- *s times continuously differentiable function*

$$u^*(v) : V \to U$$

– *the implicit function) given by the system of equations*

$$\Phi(u, v) = 0$$

such that

- $\Phi(u^*(v), v) \equiv 0$, $v \in V$;
- *for every $v \in V$, the point $u^*(v) \in U$ is the unique in U solution of the equality system $\Phi(u, v) = 0$, u being the unknown in the system.*

Exercise 8.2.5 *Derive Theorem 8.2.8 from the Implicit Function Theorem.*
Hint: Here is the sketch of the proof.

1^0. Notation. Let x^* be the nondegenerate locally optimal solution to $(\mathrm{P}_{0,0})$ in question, and let λ^* and μ^* be the corresponding Lagrange multipliers. Without loss of generality, we may assume that all inequality constraints of the problem are active at x^* – the nonactive constraints locally do not influence the problem, and we can drop theme.

2^0. Let us set

$$N = n + m + k; \quad M = m + k.$$

We partition vectors $u \in \mathbf{R}^N$ into parts

$$u = (x, \lambda, \mu)$$

of the sizes n, m, k, respectively, and set

$$u^* = (x^*, \lambda^*, \mu^*).$$

Similarly, we partition vectors $v \in \mathbf{R}^M$ into parts

$$v = (b, d)$$

of the sizes m and k, respectively, and set

$$v^* = (0, 0).$$

Now let us define the mapping

$$\Phi(u, v) = \begin{pmatrix} \nabla f(x) + \sum_{i=1}^m \lambda_i h_i(x) + \sum_{j=1}^k \mu_j \nabla g_j(x) \\ h(x) - b \\ \mu \times (g(x) - d) \end{pmatrix}$$

where $\mu \times q$ is the *dot product* of k-dimensional vectors μ and q, i.e., the k-dimensional vector given by $[\mu \times q]_j = \mu_j q_j$.

Note that the system of N equations

$$\Phi(u, v) = 0$$

is nothing but the equation part of the KKT system corresponding to the problem $(\mathrm{P}_{b,d})$.

3^0. Let us verify that the introduced data satisfy the premise of the Implicit Function Theorem (where one can set $U' = \mathbf{R}^N$, $V' = \mathbf{R}^M$). Indeed,

$$\Phi(u^*, v^*) = 0$$

since x^* is nondegenerate solution of (P_{v^*})=$(\mathrm{P}_{0,0})$, and λ^*, μ^* are the corresponding Lagrange multipliers. All we should prove is that the matrix $\Phi'_u(u^*, v^*)$ is nondegenerate. To this end it suffices to verify that

$$\phi'_u(u^*, v^*) \begin{pmatrix} dx \\ d\lambda \\ d\mu \end{pmatrix} = 0$$

only for zero $dx, d\lambda, d\mu$.

Direct computation demonstrates that the latter equality is nothing but the system of linear equalities

$$\begin{array}{lrcl}
(\alpha) & [\nabla^2 f(x^*) + \sum_{i=1}^m \lambda_i^* \nabla^2 h_i(x^*) + & & \\
& \sum_{j=1}^k \mu_j^* \nabla^2 g_j(x^*)]dx + \sum_{i=1}^m d\lambda_i \nabla h_i(x^*) + + \sum_{j=1}^k d\mu_j \nabla g_j(x^*) & = & 0 \\
(\beta) & (dx)^T \nabla h_i(x^*) & = & 0,\ i = 1, ..., m \\
(\gamma) & \mu_j (dx)^T \nabla g_j(x^*) & = & 0,\ j = 1, ..., m
\end{array}$$

(when computing last k rows in the matrix $\Phi'_u(u^*, v^*)$, you should take into account that, by virtue of our convention from 1^0, $g_j(x^*) = 0$ for all j, so that the terms $d\mu_j g_j(x^*)$ which occur when differentiating Φ in fact vanish).

From (β) and (γ) it follows that dx is orthogonal to the gradients of all equality constraints and to the gradients of those inequality constraints for which $\mu_j^* > 0$, all the gradients being taken at x^*. With this in mind, let us take the inner product of both sides of (α) (this is an n-dimensional vector equality) and the vector dx; we will get

$$(dx)^T \nabla_x^2 L(x^*; \lambda^*, \mu^*) dx = 0,$$

which, by definition of a nondegenerate solution and due to the already established orthogonality of dx and the gradients of the equality constraints and the gradients of those inequality constraints with $\mu_j^* > 0$, implies that $dx = 0$. Since $dx = 0$, (α) results in

$$\sum_{i=1}^m d\lambda_i \nabla h_i(x^*) + \sum_{j=1}^k d\mu_j \nabla g_j(x^*),$$

whence $d\lambda = 0$ and $d\mu = 0$ (x^* is regular for the constraints, so that the set comprised of gradients of all the constraints is linearly independent; here again we use our convention that all the inequality constraints are active at x^*). Thus, $dx = 0, d\lambda = 0, d\mu = 0$, as claimed.

4^0. Since, by 3^0, the premise of the Implicit Function Theorem is satisfied at (u^*, v^*) with $s = 1$, the theorem implies that there exists a neighbourhood V of the point $v^* = (0, 0)$ in the plane of parameters identifying problems (P_v) from our family, a neighbourhood U of the point u^* and once continuously differentiable function

$$u^*(v) \equiv: (x^*(b, d), \lambda^*(b, d), \mu^*(b, d)) : V \to U \quad [v = (b, d)]$$

such that

A:

$$\Phi(u^*(v), v) \equiv 0, \;\; v \in V,$$

and for $v \in V$ the point $u^*(v)$ is the unique in U solution to the system of equations

$$\Phi(u, v) = 0.$$

Since x^* is nondegenerate, $\mu^* > 0$; since $u^*(\cdot)$ is continuous, passing, if necessary, to smaller U and V, we may assume that

B: $\mu^*(v) > 0$ when $v \in V$.

Now let X' be a neighbourhood of x^* such that the gradients of the functions $h_i(x), g_j(x)$, $i = 1, ..., m$, $j = 1, ..., k$ are linearly independent for $x \in X'$; since these gradients are linearly independent at $x = x^*$ (x^* is a regular point of the constraints!), such a neighbourhood exists. Some points x from X' can be extended by properly chosen Lagrange multipliers to triples (x, λ, μ) satisfying the KKT equation, and some, perhaps, cannot; let X'' be the set of those $x \in X'$ where these multipliers exist. Since the gradients of all the constraints at the points from X' are linearly independent, the Lagrange multipliers for $x \in X''$ are uniquely defined and continuous on X''; when $x = x^*$, these multipliers are λ^*, μ^*. Consequently, there exists small enough neighbourhood, let it be called X, of x^* such that

C: the set of gradients of all the constraints at any point $x \in X$ is linearly independent, and if $x \in X$ can be extended by Lagrange multipliers λ, μ to a triple (x, λ, μ) satisfying the KKT equation, then $(x, \lambda, \mu) \in U$.

Shrinking, if necessary, V, we can ensure that

D: $x^*(v) \in X$ when $v \in V$.

Now note that the matrix $\nabla_x^2 L(x; \lambda, \mu)$ is continuous in x, λ, μ, and the plane

$$\bar{T}(x) = \{e \mid \; e^T \nabla h_i(x) = 0, e^T \nabla g_j(x) = 0, i = 1, ..., m, j = 1, ..., n\}$$

is, in the natural sense, continuous in $x \in X'$. When $x = x^*$, $\lambda = \lambda^*$, $\mu = \mu^*$, the matrix $\nabla_x^2 L(x; \lambda, \mu)$ is positive definite on the plane $\bar{T}(x^*)$; by continuity reasons, it follows that the matrix $\nabla_x^2 L(x^*(v); \lambda^*(v), \mu^*(v))$ is positive definite on $\bar{T}(x^*(v))$, whenever v is close enough to v^*. Again shrinking V, we may assume that the latter property is satisfied when $v \in V$:

E: the matrix $\nabla_x^2 L(x^*(v); \lambda^*(v), \mu^*(v))$ is positive definite on the linear subspace $\bar{T}(x^*(v))$ when $v \in V$.

Now we are basically done. By construction, when $v = (b, d) \in V$, the point $x^*(v)$ along with Lagrange multipliers $\lambda^*(v), \mu^*(v)$ satisfies the equations of the KKT system (by **A**) and $\mu^*(v) > 0$ (by **B**); the latter two properties clearly imply that $x^*(v)$ is the KKT point for the problem (P_v). By **C** and **E** this point is nondegenerate locally optimal solution to (P_v). Besides this, when $v \in V$, $x^*(v)$ is the only point in X which satisfies the necessary First Order Optimality condition for (P_v). Indeed, let $x \in X$ admit extension, by Lagrange multipliers λ, μ, to a triple satisfying the KKT equation associated with (P_v). By **C** we have $u = (x, \lambda, \mu) \in U$, so that u is a belonging to U solution to the system of equations

$$\Phi(\cdot, v) = 0.$$

By **A** such a solution is uniquely defined and is $(x^*(v), \lambda^*(v), \mu^*(v))$, as claimed.

Thus, we have proved all required statements excluding (8.2.25). The proof of the latter relation is immediate: we know from the Implicit Function Theorem that $x^*(v)$ is differentiable

in $v \in V$; let x' be the derivative of this mapping, taken at certain point $\bar{v} \in V$ along certain direction $\delta v = (\delta b, \delta d)$. Taking inner product of both sides of the equality

$$\nabla_x f(x^*(\bar{v})) + \sum_{i=1}^{m} \lambda_i^*(\bar{v}) \nabla_x h_i(x^*(\bar{v})) + \sum_{j=1}^{k} \mu^*(\bar{v}) \nabla_x g_j(x^*(\bar{v})) = 0$$

and x' and taking into account that

$$(x')^T \nabla_x f(x^*(\bar{v})) = \frac{d}{dt}|_{t=0} f^*(\bar{v} + t\delta v)$$

("chain rule"), we get

$$\frac{d}{dt}|_{t=0} f^*(\bar{v} + t\delta v) = -\sum_{i=1}^{m} \lambda_i^*(\bar{v})(x')^T \nabla_x h_i(x^*(\bar{v})) - \sum_{j=1}^{k} \mu_j^*(\bar{v})(x')^T \nabla_x g_j(x^*(\bar{v})). \qquad (8.2.26)$$

We have, identically in $v = (b, d) \in V$,

$$h_i(x^*(v)) = b_i,\ i = 1, ..., m; \quad g_j(x^*(v)) = d_j,\ j = 1, ..., k$$

(the fact that all g_j are active at $x^*(v)$ comes from the KKT system due to $\mu_j^*(v) > 0$, $j = 1, ..., k$, see **B**. Differentiating these equalities at the point $\bar{v}$ in the direction δv, we observe that the right hand side in (8.2.26) is equal to

$$-\sum_{i=1}^{k} \lambda_i^*(\bar{v}) \delta b_i - \sum_{j=1}^{k} \mu_j^*(\bar{v}) \delta d_j,$$

and (8.2.26) results in (8.2.25).

Lecture 9

Primal Methods

The traditional methods for general type constrained minimization

$$\begin{array}{lrcl} \text{(P)} & & & \\ & f(x) & \to & \min \\ \text{s.t.} & & & \\ & h_i(x) & = & 0,\ i = 1, ..., m, \\ & g_j(x) & < & 0,\ j = 1, ..., k \end{array}$$

(when saying "general type problems", I mean not necessarily convex ones; Convex Programming is another and much nicer story) can be, roughly speaking, separated into the following groups:

- primal methods, where we, roughly speaking, try to act similarly to what we did in the unconstrained case – i.e., to move along the feasible set in a way which ensures, at each step, progress in the objective;
- barrier and penalty methods, where we reduce (P) to a series of "approximating" the problem unconstrained programs;
- Lagrange multipliers methods, where we focus on the *dual problem* associated with (P); this dual problem is either unconstrained one (when (P) is equality constrained), or has simple nonnegativity constraints (when (P) includes inequalities) and is therefore simpler than (P). When solving the dual problem, we get, as a byproduct, approximate solutions to (P) itself. Note that a posteriori the Lagrange multiplier methods, same as the penalty/barrier ones, reduce (P) to a sequence of unconstrained problems, but in a "smart" manner quite different from the straightforward penalty/barrier scheme;
- SQP (Sequential Quadratic Programming) methods. The SQP methods, in contrast to all previous ones, neither try to improve the objective staying within the feasible set, nor approximate the constrained problem by unconstrained ones, but directly solve the KKT system of (nonlinear) equations associated with (P) by a kind of the Newton method.

Today we shall speak about Primal methods. As it will become clear later, their actual field of applications are *linearly constrained problems* – when all the equality and inequality constraints of the problem are linear. It make, anyhow, sense not to restrict ourselves from the very beginning with the linearly constrained case – theoretically, linearity of constraints is not that crucial.

Before passing to the main body of the lecture, let me make an important comment.

When solving an unconstrained minimization problem, we were aimed to find the optimal solution; but the best we indeed were able to do was to ensure convergence to the set of critical points of the objective, the set of points where the First Order Necessary Optimality condition – the Fermat rule – is satisfied. Similarly, the best we can do in constrained optimization is *to ensure convergence to the set of KKT points of the problem – those points where the First Order Necessary Optimality condition is satisfied.* Whether it fits our actual goal or not – this is another story, sometimes with happy end (e.g., in the case of *convex* problem a KKT point is for sure globally optimal solution), sometimes – not, but this is all we can achieve.

During all this lecture, we make the following assumption on problem (P) in question:

Regularity: The problem is *regular*, i.e., it is feasible and every feasible solution is a regular point (see Definition 8.2.3, Lecture 8) for the system of constraints.

9.1 Method of Feasible Directions

Just to start our considerations, let me briefly outline the idea of the oldest primal method – the Feasible Directions one. Those who have passed through the course Optimization I, already know what the method is – we used it to demonstrate the immediate applications of the Optimality conditions.

The Feasible Directions method can be applied only to a problem without *nonlinear* equality constraints. To simplify considerations, assume that all our constraints are *linear inequalities*, so that the problem is

$$\text{(LIP)} \quad f(x) \to \min \mid \; g_j(x) \equiv a_j^T x - b_j \le 0, \; j = 1, ..., k. \quad [x \in \mathbf{R}^n] \tag{9.1.1}$$

The idea of the method is as follows: the *First Order Necessary Optimality conditions (the KKT conditions) are "constructive": if they are not satisfied at a given feasible solution x, then we can explicitly point out a better – with a smaller value of the objective – feasible solution x^+.*

Indeed, we remember from the previous lecture that the KKT conditions were obtained from the following construction: given feasible solution x, we associate with it *linearization of (LIP) at x* – the auxiliary Linear Programming program

$$\text{(LIP}_x\text{)} \quad \bar{f}(y) \equiv f(x) + (y - x)^T \nabla f(x) \to \min$$

subject to

$$(y - x)^T a_j \equiv (y - x)^T \nabla g_j \le 0, \; j \in I(x) = \{j \mid \; g_j(x) = 0\};$$

x is a KKT point of (9.1.1) if and only if x is optimal solution to (P_x) (this was exactly the conjecture which led us to the KKT condition).

From this "if and only if" statement we conclude that if x is not a KKT point of (9.1.1), then x is not optimal solution to (P_x). In other words (pass in (P_x) from variable y to $d = y - x$ and note that $a_i^T x = b_i, \; i \in I(x)$), there exists a *descent direction* d – direction satisfying

$$d^T \nabla f(x) < 0, \; d^T \nabla g_j \le 0, \; j \in I(x).$$

When performing a small enough step in this direction, we improve the objective (by the same reasons as in the Gradient Descent) and do not violate the constraints which are active at x. Since small enough step for sure does not violate the remaining – nonactive at x – constraints,

we indeed improve the objective and do not loose feasibility. Now we can iterate the construction at the new point, an so on.

Normally, at a non-KKT point there exist many descent directions. In the Feasible Directions method we choose the one which is "most perspective" – along which the objective decreases at the highest possible rate. Of course, to choose the most perspective direction, we should normalize somehow the candidates (multiplying a given descent direction by large scalar factor, we multiply by the same factor the rate of progress in the objective, clearly not improving the direction itself). The standard normalization here is given by the restriction

$$|d_i| \leq 1,\ i = 1, ..., n, \tag{9.1.2}$$

so that the direction in question is the solution to the following LP program:

$$d^T \nabla f(x) \to \min \mid d^T \nabla g_j \leq 0,\ j \in I(x),\ |d_i| \leq 1,\ i = 1, ..., n. \tag{9.1.3}$$

Normalization (9.1.2) instead of more natural normalization like $|d| \leq 1$ is motivated by the desire for the direction to be determined via an LP, and thus effectively solvable, program.

After the "most perspective" direction d is found, we define the largest stepsize γ, let it be called $\bar{\gamma}$, among the stepsizes which keep the point $x + \gamma d$ feasible, and define the next iterate x^+ via the linesearch applied to f on the segment $[0, \bar{\gamma}]$:

$$x^+ \in \mathrm{Argmin}\{f(y) \mid y = x + \gamma d,\ 0 \leq \gamma \leq \bar{\gamma}\}.$$

Then we replace x with x^+ and loop.

The outlined idea can be naturally extended onto inequality constrained problems with nonlinear constraints (i.e. where g_j are not necessarily linear; in this case, of course, ∇g_j arising in (9.1.3) should be replaced with $\nabla g_j(x)$). The extension is not quite straightforward:

- extending "literally" the construction of the search direction onto nonlinear case, we meet with unpleasant possibility that the direction d given by (9.1.3) may result in $d^T \nabla g_j(x) = 0$ for certain $j \in I(x)$; if it is the case, it well may happen that the maximal allowed by the constraints stepsize $\bar{\alpha}$ from x along the direction d is zero – the linearization of g_j at x is constant along the direction d, so that the (nonlinear!) constraint g_j may become positive after even a small step in the direction. It follows that we should use in (9.1.3) strict inequalities like $d^T \nabla g_j(x) \leq -\alpha |d|$, with reasonably chosen $\alpha > 0$ rather than nonstrict inequalities $d^T \nabla g_j(x) \leq 0$, at least for the nonlinear g_j;

- a weak point of the above "naive" version of the Feasible Direction method (which is seen also in linearly constrained case) is that choosing the search direction we do not even look at the nonactive constraints. This for sure makes no sense: a constraint which is nonactive at a point, but is "almost active" – with very small in absolute value negative $g_j(x)$ (when choosing the direction, we did not look at this constraint at all) well can rapidly increase in the chosen direction and become active very close to the point, thus resulting in a very small stepsize; and it may happen very far from the KKT set! In fact everybody who at least once worked with computer understands in advance that, numerically, it makes no sense to partition the constraints into "active" and "nonactive" – in actual computations, we may speak only about "essentially satisfied" and "essentially violated" constraints.

As a result of the indicated drawbacks, the above "naive" version of the Feasible Directions method as applied to nonlinear problems may be bad even in good (convex) situations: it may

converge to a "false" point (not a KKT one). The intrinsic mathematical explanation of this phenomenon is that the corresponding algorithmic mapping is not closed even outside the KKT set (see Lecture 1). There are several ways to modify the Feasible Directions method in order to make the corresponding mapping closed and thus to ensure its convergence to the KKT set; one of this versions was presented in Lecture 9 of the course Optimization I. I am not going to go in more details here – my goal was mainly to indicate how careful should you be when implementing natural constructions!

9.2 Active Set Methods

In order to motivate two typical primal methods to be considered – the *Gradient Projection* and the *Reduced Gradient* ones – let us start with *equality constrained* problems.

9.2.1 Equality constrained case: Gradient Projection scheme

Assume that we are solving a regular equality constrained problem

$$f(x) \to \min \mid h(x) = \begin{pmatrix} h_1(x) \\ \dots \\ h_m(x) \end{pmatrix} = 0, \tag{9.2.1}$$

and let S be the feasible surface of the problem.

What we could do in order to solve the problem, it depends on the nature of constraints.

In the equality constrained case – where h is linear – we, basically, can use all machinery for unconstrained minimization we have developed so far. Indeed, geometrically our problem is an unconstrained one: we should minimize the objective over a plane, the affine set given by a number of linear equality constraints. We can use this system of constraints to represent part of the design variables as linear functions of another part of the variables, thus partitioning the variables into "dependent" and "independent" ones. Substituting these expressions into the objective, we reduce our problem to an unconstrained one with smaller number of design variables (the independent ones), and it remains to solve the resulting unconstrained problem.

Of course, in actual computations we should not necessarily carry this scheme out explicitly. We can work with the problem in its original format, reproducing in the initial variables this or that optimization method for the equivalent unconstrained problem, thus coming to "linear equality constrained versions" of unconstrained minimization methods. These versions are, of course, more complicated from the algorithmic viewpoint, they may be less numerically stable, due to complications in Linear Algebra routines, etc.; apart of these implementation issues, all convergence properties of the original methods are inherited by their modifications capable to handle linear equality constraints.

The case of nonlinear equality constraints is much more complicated. Let us look at one of the schemes which can be used in this difficult case – at the *Gradient Projection scheme*.

Assume that we have formed current iterate $x_t \in S$, and let

$$T_t = x + L_t, \;\; L_t = \{e \mid \nabla h(x_t)e = 0\}$$

be the plane tangent to S at x_t.

It is possible that the gradient of the objective at x_t is orthogonal to the surface (i.e., to the linear subspace L_t). Then x_t satisfies the conjecture from which we derived the First Order

Necessary Optimality conditions (see Section 8.2.1, Lecture 8), so that x_t is the KKT point of the problem. If it is the case, we terminate. Now let $\nabla f(x_t)$ be non-orthogonal to the surface. Then minus projection of the gradient onto L_t is a descent direction of the objective. It can be easily seen that the direction is given by

$$d_{t+1} = -P_t \nabla f(x_t),$$

$$P_t = I - [\nabla h(x_t)]^T \left(\nabla h(x_t)[\nabla h(x_t)]^T\right)^{-1} \nabla h(x_t) \tag{9.2.2}$$

being the matrix of the orthogonal projection onto L_t.

Performing a step

$$x_t \mapsto x_t^+ = x_t + \gamma_{t+1} d_{t+1}$$

from x_t in the direction d_{t+1} with a small stepsize $\gamma_t > 0$, we get progress in the objective which is, up to the highest order terms, proportional to the stepsize with positive proportionality coefficient, while the distance from the shifted point to the feasible surface is at most proportional to the *squared* stepsize (Tangent Plane Theorem 8.2.1, Lecture 8). It follows that if we were able to pass from the shifted point to a close point of the feasible surface, we would, for small stepsizes, get a new feasible point x_{t+1} with better value of the objective (we gain something proportional to the stepsize when shifting the point along the tangent plane and then loose something at most proportional to the squared stepsize when projecting the shifted point back to the surface; for small stepsizes, the balance is in our favour). Now we could apply similar updating to x_{t+1}, and so on. All this is very similar to the Gradient Descent scheme in unconstrained minimization; in this latter case, our "feasible surface" is the entire space, and we need no projection.

In the general case, of course, the projection may cause severe difficulties: to project a point onto a "curved" surface, this is nontrivial computational problem which, normally, cannot be solved with finite computational effort. What we indeed can do with finite computational effort is to come close to the surface, within a prescribed small tolerance. If the point to be projected is close enough to the surface, the latter problem can be easily solved even when the required tolerance is small. Indeed, to project the point onto the surface is, basically, the same as to solve certain system of nonlinear equations, and when our point is close to the surface, it means that we have a good approximate solution to this system; in order to transform this "good" approximate solution to an "excellent" one (that one fitting the tolerance), we could apply to the system a Newton-type method with good local rate of convergence (I am speaking about the Newton method for solving systems of equations, not the one for minimization). In more detailed description, this looks as follows. Given x_t^+, let us try to find a displacement *orthogonal* to L_t such that x_t^+ plus the displacement is in S. The directions orthogonal to L_t, i.e., to the kernel of the $m \times n$ matrix $\nabla h(x_t)$, belong to the image of the transposed matrix, i.e., are of the form $[\nabla h(x_t)]^T \alpha$ for some $\alpha \in \mathbf{R}^m$. Thus, our problem is to find $\alpha^* \in \mathbf{R}^m$ such that

$$h(x_t^+ + [\nabla h(x_t)]^T \alpha^*) = 0.$$

This is a particular case of a system

$$\eta(\alpha) = 0$$

with m nonlinear equations and m unknowns; the standard way to solve the system is to use the Newton method for solving systems of equations (not the optimization one!):

$$\alpha_{i+1} = \alpha_i - [\eta'(\alpha_i)]^{-1} \eta(\alpha_i)$$

(we find the new approximation as the solution to the linearized equation $\eta(\alpha_i)+\eta'(\alpha_i)[\alpha_{i+1}-\alpha_i=0$; the known to us Newton *minimization* method for unconstrained problem $\phi(x)\to\min$ is exactly the above recurrence applied to the Fermat equation $\nabla\phi(x)=0$).

For the particular system we are interested in the Newton recurrence becomes

$$\alpha_{i+1}=\alpha_i-\left([\nabla h(y_i)][\nabla h(x_t^+)]^T\right)^{-1}h(y_i),\;\; y_i=x_t^+ +[\nabla h(x_t^+)]^T\alpha_i;$$

The computational disadvantage of this recurrence is that we should at each step compute and invert certain $m\times m$ matrix. Since we intend to apply the reccurence in the situation when the starting point x_t^+ is close enough to x_t and is "very close" to S, we expect that all the points y_i will be close to x_t and may therefore pass from the Newton reccurence to a "frozen Newton" one (compare with Section 6.1, Lecture 6):

$$\alpha_{i+1}=\alpha_i-\left[\nabla h(x_t^+)][\nabla h(x_t^+)]^T\right)^{-1}h(y_i),\;\; y_i=x_t^+ +[\nabla h(x_t^+)]^T\alpha_i.$$

The induced recurrence for y_i is simply

$$y_{i+1}=y_i-[\nabla h(x_t)]^T\left([\nabla h(x_t)][\nabla h(x_t)]^T\right)^{-1}h(y_i)\equiv y_i-(I-P_t)h(y_i),\;\; y_1=x_t^+. \qquad (9.2.3)$$

It can be proved that this recurrence converges linearly, provided that h_i are three times continuously differentiable and the stepsize γ_{t+1} is small enough; the less is the stepsize, the faster is the convergence – the closer to 0 is the convergence ratio. Thus, we indeed can quickly reduce the distance from the shifted point x_t^+ to the surface S – to get a new point x_{t+1} "belonging to the surface S within given small tolerance", say, within machine accuracy, and then iterate the outlined "predictor-corrector" transformation $x_t\mapsto x_{t+1}$.

Not coming in further details here, let us assume (this is a reasonable idealization of our demonstrated actual abilities) that after shifting point along the tangent plane we indeed can move the shifted point to the closest point of the feasible surface. Note that there is at least one case when our assumption indeed is reasonable – and even is not an idealization: this is the case of linear equality constraints in the original problem. In this case the feasible surface coincides with the tangent plane, and the "projection difficulty" does not occur at all.

9.2.2 Inequality Constrained case: Active Set scheme

Now let us look how the outlined idea can be modified in order to handle general type constrained problems. For the sake of convenience, consider the inequality constrained case (it will become clear in a while that the extensions of what we are about to do onto the mixed case are obvious). Thus, let our problem be

$$f(x)\to\min\mid\; g_j(x)\le 0,\; j=1,...,k.\quad [x\in\mathbf{R}^n] \qquad (9.2.4)$$

If we could guess what is the active set of the problem – what are the constraints which are active at the solution – we could reduce the situation to the previous one – to the case of equality constraints, simply by eliminating the inequality constraints which are not in the active set and making the remaining inequality constraints the equality ones. The difficulty is, of course, how to identify the correct active set. The idea of the *active set* methods is

- to use, at every phase of the process, certain current *guess* for the true active set – this guess is called *working set*;

- at a phase of the process associated with a given working set, to convert the problem to an equality constrained one by eliminating the inequalities which are not in the working set and making the remaining inequalities equalities, and to solve the resulting equality constrained problem by some method, e.g., of the type outlined in the previous subsection;

- when running the method for the equality constrained problem, to control whether current information is consistent with the conjecture that the current working set indeed is active; when inconsistency is detected, we correct our guess for the true active set and proceed with the updated working set.

There are two components in the outlined general *active set* scheme –

(1) how to detect that the current working set should be updated and how to update it when inconsistency is detected;

(2) how to solve the equality constrained problem associated with the current working set.

We will consider these two components separately. For the sake of simplicity, in what follows we assume that the feasible set of our (regular) problem is bounded.

Extending working set

Assume that when processing the current working set W, i.e., when optimizing the objective over the corresponding *working surface*

$$S_W = \{x \mid g_j(x) = 0,\ j \in W\}$$

by a method of the type indicated in Section 9.2.1, we are about to pass from an iterate x_t satisfying all inequality constraints not included into the working set to an iterate x_{t+1} which violates some of these constraints. Then it makes sense not to pass from x_t to x_{t+1}, but to extend the current working set with the indices of the constraints we are about to violate (thus reducing the dimension of the current working surface) and to proceed with these new working set and working surface.

Of course, it is easy to say "to proceed with the new working set", but how to do it? To deal with the new working surface, we should "stand at it"; and what we do have are two points x_t and x_{t+1} which are "at opposite sides" of the new working surface; no one of them, generally speaking, is exactly at the latter surface.

Our abilities to reach the new working surface depend on whether the problem in question has only linear or also nonlinear constraints.

In the case of nonlinear constraints we should apply to x_t, or to x_{t+1}, or to a reasonable "mixture" of these points, the "projection" routine similar to the one described in Section 9.2.1. If x_t is close to x_{t+1} (as it is normally the case when travelling along a curved surface), then both the points are close to the new working surface, and we are in a good position to project on it.

in the equality constrained case the situation is much better. Indeed, in basically all known to us unconstrained optimization methods which we could use to minimize the objective at the current working surface (let us better call it *working plane* – it indeed is a plane), x_{t+1} is obtained from x_t by a kind of linesearch in certain search direction d_{t+1} parallel to the working plane. Before performing linesearch, we could easily compute the largest of those stepsizes γ from x_t along the direction d_{t+1} which keep the shifted point $x_t + \gamma d_{t+1}$ feasible. If this "largest admissible by the constraints stepsize", let it be called $\bar{\gamma}$, is finite, then at the point $x_t + \bar{\gamma} d_{t+1}$ one or several inequality constraints which are not in the current working set become active.

Now let us replace the usual linesearch from x_t in the direction d_{t+1} with its "restricted" version, where we do not allow the stepsize to be greater than $\bar{\gamma}$, and let this linesearch result in the new iterate $x_{t+1} = x_t + \gamma_{t+1} d_{t+1}$. If $\gamma_{t+1} < \bar{\gamma}$, then no "new" (those not in the previous working set) inequality constraints become active at x_{t+1}, and we can proceed with the previous working set and working plane. Otherwise $\gamma_{t+1} = \bar{\gamma}$, so that one or more new inequality constraints become active at x_{t+1}. In this latter case we add indices of all these constraints to the current working set, thus extending it and shrinking the working plane, and proceed with this new working plane; note that x_{t+1} belongs to the latter plane, and no necessity in "projection" occurs.

Shrinking working set

After certain number of sequential extensions of the working set mentioned in the previous subsection, we will travel along certain fixed surface S' – indeed, in the process in question the working set is only increased, and it cannot become larger than the set of indices of *all* constraints we have. According to our policy of extending the working set – it is extended each time when the current iterate violates one of the constraints not in the working set – all out iterates at this phase will be feasible. Assume, for the sake of definiteness, that we trace our working surfaces by the Gradient Projection method presented in Section 9.2.1. Then – by the same reasons as in the case of unconstrained minimization by the Gradient Descent – our process will converge to the critical set of the objective at S', i.e., to the set of points of S' where the gradient of the objective is orthogonal to the surface. Indeed, while the gradient of the objective is not "almost orthogonal" to S' (i.e., the projection of the gradient onto the tangent plane remains greater than certain positive δ), the progress in the objective per step remains at least certain $\epsilon(\delta) > 0$, and since the objective is below bounded on the feasible set (the latter is compact!), the number of these steps cannot be too large[1]. With a reasonable idealization, therefore, we may assume that at certain moment we will arrive at a critical point of the objective at S', let the point be called x. At this point, as it was explained in the beginning of Section 9.2.1, we will have

$$\nabla f(x) + \sum_{j \in W} \lambda_j \nabla g_j(x) = 0,$$

with properly chosen λ_j. If all these Lagrange multipliers are nonnegative, we are done – we have found a KKT point of the original problem. Now assume that certain multiplier λ_j is negative. I claim that then it makes sense to eliminate j from the current working set – the projection of $\nabla f(x)$ on the tangent plane to the new surface S'' (corresponding to the reduced working set) will have positive inner product with $\nabla g_j(x)$. Let us for a moment believe in this fact; what are the consequences? They are immediate. The first step of our "surface-following" method as applied to the reduced working set will, basically, be a step along the projection of $-\nabla f(x)$ onto the tangent plane to S'' (plus "second-order" corrector step aimed to put the shifted point onto S''). The direction of the step, as was announced, has negative inner product with $\nabla g_j(x)$, so that the step will *reduce* the constraint g_j in the first order with respect to the stepsize terms. The second order effects caused by the corrector step and nonlinearity of the constraint g_j will be unable to compensate this effect, provided that the stepsize is small enough. It follows that our step will keep the point feasible; and, as a step of our "surface

[1] of course, this reasoning requires certain restrictions on the policy for choosing stepsizes. In the unconstrained case, these restrictions could be given by the Armijo rule or something like this. Similar restrictions should be imposed also in the constrained case; we, anyhow, are not going to go in too deep details here in order not to overload our course with "second-order" complications of purely theoretical origin, basically irrelevant to practical computations

tracing" method from a point where the projection of ∇f onto the (new) working surface is nonzero, it will reduce the objective. Thus, we will generate a better approximate solution – still feasible and with improved value of the objective. Now we can continue tracing surface S''.

It is time now to justify the announced claim about inner product. Our situation is as follows: we are given certain linearly independent set of vectors $\phi_j, j \in W$ (these are the gradients of the constraints with the indices from the working set W; they are linearly independent, since we have assumed that the problem is regular) and vector ϕ (the gradient of the objective) which is linear combination of ϕ_j. Without loss of generality we may assume that $W = \{1, ..., l\}$, so that

$$\phi = \sum_{j=1}^{l} \lambda_j \phi_j.$$

We know also that at least one of the coefficients λ_j is negative, let it be the coefficient λ_1. And what we should prove is that the projection ψ of the vector ϕ on the linear subspace

$$L = \{e \mid \phi_j^T e = 0,\ j = 2, ..., l\}$$

(the tangent plane to the surface associated with the reduced working set $\{2, ..., l\}$) has positive inner product with ϕ_1. This is immediate: by definition of the projection, ψ belongs to L, whence

$$\psi^T \phi_j = 0,\ j = 2, ..., l;$$

taking weighted sum of these equalities with coefficients λ_j, we get

$$0 = \psi^T (\sum_{j=2}^{l} \lambda_j \phi_j) =$$

[since by assumption $\sum_{j=1}^{l} \lambda_j \phi_j = -\phi$, whence $\sum_{j=2}^{i} \lambda_j \phi_j = -\phi - \lambda_1 \phi_1$]

$$= \psi^T (-\phi - \lambda_1 \phi_1),$$

or

$$\psi^T \phi = -\lambda_1 \psi^T \phi_1 = |\lambda_1| \psi^T \phi_1 \tag{9.2.5}$$

(recall that λ_1 is negative). The left hand side in this relation is nonnegative (as inner product of a vector and its orthogonal projection onto a linear subspace), and it is zero if and only if $\psi = 0$, i.e., if and only if ϕ itself is orthogonal to L. But if ϕ is orthogonal to L (which is given as the solution set to the system of linear homogeneous equations $\phi_i^T e = 0$, $i = 2, ..., l$), then the vector ϕ should be linear combination of the vectors involved into the system:

$$\phi = \sum_{j=2}^{m} \lambda'_j \phi_j.$$

Thus, we have two different representations of ϕ as a linear combination of $\phi_1, ..., \phi_l$ – one with negative coefficient λ_1 at ϕ_1 and another with zero coefficient at the vector; this is impossible, since the vectors $\phi_1, ..., \phi_l$ were assumed to be linearly independent.

Thus, $\psi^T \phi > 0$, and we conclude from (9.2.5) that $\psi^T \phi_1 > 0$, as claimed. ∎

Active Set methods

In an active set method we combine the two aforementioned tactics – minimize the objective over the working surface associated with the current working set, until one of the following three possibilities is met:

- we are about to leave the feasible set, i.e., to crosse at least one of the hyper-surfaces $\{x \mid g_j(x) = 0\}$ with j not in the current working set; when it happens, we add j to the current working set and proceed with the updated working set;
- we have met a feasible point which (within a given small tolerance) is a critical point of the objective at the current working surface: the gradient of the objective "almost" is linear combination of the gradients of the constraints associated with the current working set. When it happens, we look at the signs of the coefficients λ_j in this linear combination and
  - terminate, if all the coefficients are "essentially nonnegative" (are nonnegative within another small tolerance); we are at an (approximate) KKT point of the problem;
  - eliminate from the working set the index of one of the constraints which corresponds to "essentially negative" coefficient λ_j and proceed with the new working set.

Remark 9.2.1 *To the moment, we spoke about inequality constrained problems only. Now it is absolutely clear that the equality constraints do not bring any troubles: we simply should always keep them in the working set, and that is it!*

An immediate question is – what can be said about the outlined procedure? May we expect certain reasonable behaviour of it – e.g., termination with a KKT point (approximate, within the tolerances we use to distinguish between "essentially nonnegative" and "essentially negative" Lagrange multipliers), or we can stay all the time far from the KKT set of the problem, switching from one working surface to another?

It turns out that under not too restrictive assumptions the things are not bad. Indeed, let us speak about the idealized active set method, where we move along the current working surface decreasing the objective and update the working set whenever crossing the surface $\{x \mid g_j(x) = 0\}$ of an inequality constraint not in the current working set (when it happens, we extend the working set) or until a critical point of the objective at the current working surface is found (in the latter case, we either terminate, or shrink the working set, as explained above). The idealization is that we, first, are able to move along the surface rather than along its "tight" neighbourhood, and, second, that when staying at the current working surface for a long enough time, we reach the critical set of the objective at the surface. As it was explained, this is quite reasonable idealization.

Theorem 9.2.1 [Active Set Theorem] *Assume that problem (P) is regular (all feasible points are regular for the constraints) and with nonempty bounded feasible set. Assume also that there is exactly one critical point of the objective at each tentative working surface (a nonempty set obtained by eliminating some of inequalities of (P) and replacing the remaining inequalities with equalities). Then the idealized active set method terminates with KKT point of the problem.*

Proof. The only possibility for the method to terminate is to find a KKT point of the problem, so that all we need is to prove that the method terminates. Assume that it is not the case. The trajectory of the method can be partitioned into stages between sequential shrinkages of the

working set, and every stage, in turn – into phases where the working set remains unchanged. According to our idealization, each of the phases is finite. The number of phases at a given stage also is finite (since from phase to phase within a given stage the working set increases, and it cannot become larger than the set of indices of all the constraints). Thus, all we need is to prove that the number of stages is finite. To this end let us look at the working surfaces we deal with at the concluding phases of the stages. We shrink the working set only when a critical point of the objective at the current working surface is met. On a given working surface, by assumption, there is unique critical point. It follows that to meet twice the same working surface as concluding a stage means to visit twice the same point, which is impossible: the objective in our routine monotonically decreases until termination!

Since different stages are concluded by different working surfaces, and there are finitely many working surfaces, there are finitely many stages. ∎

Now we turn to the main ingredient of an active set method – to the method for minimizing the objective over the current working surface, and to the resulting routines.

9.3 Gradient Projection and Reduced Gradient methods

9.3.1 The Gradient Projection method

The Gradient Projection method is the Active Set method where minimization of the objective over the current working surface is carried out via the Gradient Projection scheme presented in Section 9.2.1. As far as the case of a general – nonlinear – problem is concerned, there is nothing to add to this sentence. It makes, anyhow, sense to look at the method in more details in the *linearly constrained case* – when all the equality and inequality constraints of the problem are linear:

$$h_i(x) = a_i^T x - b_i,\ i = 1, ..., m; \quad g_j(x) = a_{m+j}^T x - b_{m+j}.$$

In this nice and important case the procedure simplifies significantly.

- First of all, we have no troubles with corrections – the point x_t on the current working surface S (which now is just a plane), shifted in the projected onto S antigradient direction of the objective, i.e., the point

  $$x_t^+ = x_t + \gamma_{t+1} d_{t+1},\ \ d_{t+1} = -P_t \nabla f(x_t),$$

  P_t being the current projection matrix, remains in S independently of the stepsize γ_{t+1}, so that this point is the same as x_{t+1};

- Second, it is easy to choose the stepsize γ_{t+1}. To this end as we remember from Section 9.2.2 it suffices to identify the largest stepsize $\bar{\gamma}$ which ensures feasibility of the shifted point:

  $$\bar{\gamma} = \sup\{\gamma > 0 \mid\ a_i^T(x_t + \gamma d_{t+1}) \le b_i,\ i \notin W\},$$

  where W is the current working set (recall that it all the time includes the indices of all equality constraints). Of course, it is immediate to compute this $\bar{\gamma}$:

  $$\bar{\gamma} = \min\{\frac{b_i - a_i^T x}{a_i^T d_{t+1}} \mid\ i \notin W,\ a_i^T d_{t+1} > 0\}.$$

After $\bar\gamma$ is found, it is easy to choose a good stepsize γ_{t+1}, e.g., by the "restricted linesearch":

$$\gamma_{t+1} \in \mathrm{Argmin}\{f(x_t + \gamma d_{t+1}) \mid 0 \le \gamma \le \bar\gamma\}.$$

If it turns out that $\gamma_{t+1} = \bar\gamma$, the iterated point will make active at least one of previously non-working linear inequality constraints; according to our general tactics, we should add the indices of all these constraints to the working set. Note that in the case in question we simply find ourselves on the new working plane.

- Third, it is easy to update the projection matrix P_t from step to step. Indeed, when both steps in question deal with the same working set, the matrix simply remains the same. And if the steps relate to different working sets, then the corresponding planes are given by the systems of linear equations which, normally, differ by one inequality (either eliminated from the earlier system, or added to it). With this modification of the working plane, the projector to it can be updated in $O(n^2)$ operations by a kind of Sherman-Morrison formula (Exercise 6.6.2).

9.3.2 The Reduced Gradient method

The idea of the Reduced Gradient method is very close to the one of the Gradient Projection method: Reduced Gradient method also implements the Active Set scheme, and the only difference is how we minimize objective over the current working surface S. In the Reduced Gradient method, we use certain parameterization of S, namely, use the equations defining the surface to express some of our design variables ("the dependent variables") via the rest of the variables (the "independent" ones). The resulting parameterization of the working surface allows to express the objective on this surface via the independent variables. We minimize the resulting function of "independent" variables by the unconstrained Gradient Descent, thus minimizing the actual objective over the working surface. Of course, all the time we take care of the inequality constraints not included into the current working set and use the Active Set strategy to update the working set and working surface; with each updating of this type, we renew our partitioning of the variables into "dependent" and "independent".

I am not going to present the detailed description of the Reduced Gradient method in the nonlinear case; let us focus on the linearly constrained one.

Same as in Linear Programming, introducing slack variables we can rewrite a linearly constrained problem (P) in the following *standard form*:

$$f(x) \to \min \mid Ax = b,\ x \ge 0 \quad [x \in \mathbf{R}^n] \tag{9.3.1}$$

where A is $m \times n$ matrix. To simplify things, let us assume that every m columns of A are linearly independent, and any basic feasible solution (a vertex of the feasible polyhedron – feasible point with at most m positive coordinates) has exactly m positive coordinates (the same nondegeneracy assumption often is used in LP). It immediately follows that the number of positive entries at *any* feasible solution is at least m.

Given a feasible solution $\bar x$, we always can partition the vector of variables as

$$x = (y, z)$$

with m "dependent" variables y which are positive at $\bar x$ and remaining $n - m$ "independent variables" z (they are nonnegative at $\bar x$. Let B and C be the submatrices of A corresponding to

this partitioning of the design vector; then the problem can be rewritten as

$$f(y,z) \to \min \mid \; By + Cz = b, \; y, z \geq 0. \tag{9.3.2}$$

Since B is an $m \times m$ submatrix of the matrix A and thus, due to the nondegeneracy assumption, is nondegenerate, we can solve the system of linear equations with respect to the dependent variables:

$$y = y(z) \equiv B^{-1}(b - Cz),$$

thus expressing our objective on the feasible set via the independent variables; expressed in terms of these variables, the objective becomes

$$\phi(z) = f(B^{-1}(b - Cz), z).$$

With the resulting parameterization of the feasible plane and the objective in terms of the independent variables, we can move along the plane of independent variables in order to reduce the value of the objective. The gradient of ϕ – the *reduced gradient* of f – is

$$r = \nabla_z f(y,z) + C^T (B^{-1})^T \nabla_y f(y,z).$$

The KKT condition for the original problem, as it is easily seen, can be expressed in terms of the reduced gradient as follows:

$$z_i > 0 \Rightarrow r_i = 0; \quad z_i = 0 \Rightarrow r_i \geq 0. \tag{9.3.3}$$

In the Reduced Gradient method as applied to linearly constrained problem (9.3.1), we at each phase deal with certain partition of the vector of design variables x into the "dependent" part y (which should be positive at the iterates of the phase) and "independent part" z and with certain *working sets* W comprised of part of the indices of the independent variables; at the current iterate, the independent variables with the indices from the working set are zero, and the remaining independent variables are positive. The working surface corresponding to the current working set is, of course, the plane

$$S = \{x = (y,z) \mid \; By + Cz = b, \; z_i = 0, \, i \in W(z)\},$$

where B and C are the submatrices of A corresponding to the partitioning of the variables into dependent and independent ones.

At a step of the method, given current iterate $x = (y,z)$ and current working set W such that

$$\begin{array}{lrcl} (\alpha) & Ax & = & b \\ (\beta) & y & > & 0 \\ (\gamma) & z_i & > & 0, \; i \in W \\ (\delta) & z_i & = & 0, \; i \notin W \end{array}$$

we act as follows.

1) we compute the reduced gradient r and check whether the KKT condition (9.3.3) is satisfied. If it is the case, we terminate – x is a KKT point of the problem.

If (9.3.2) is <u>not</u> satisfied, two possibilities may occur:

- (I) among the components r_i of the reduced gradient with indices $i \notin W$ there are nonzeros. If it is the case, we form the search direction d in the space of independent variables according to

$$d_i = \begin{cases} -r_i, & i \notin W \\ 0, & i \in W \end{cases} \tag{9.3.4}$$

  do not vary the working set and go to 2)

- (II) all components r_i of the reduced gradient with indices $i \notin W$ are zeros. Since we are in the situation where (9.3.3) is not satisfied, there exist an index $i \in W$ such that $r_i < 0$. We eliminate this index from the working set, thus updating this set (this is exactly the case when our Active Set strategy requires to shrink the working set), define the search direction d in the space of independent variables according to (9.3.4) *applied to shrinked working set* and go to 2)

2) When we arrive at 2), we have a nonzero direction d in the plane of independent variables with the following properties:

- d is descent direction for ϕ at the point z: $r^T d < 0$ (this is clear from (9.3.4));
- all components of d_i with entries from the current working set W' are zeros (I write W' instead of W, since we might update the working set if case (II) was met);
- if $z_i = 0$ for some $i \notin W'$, then $d_i > 0$ (indeed, all entries of z with indices not in W are positive by (γ), so that the only possibility to encounter $i \notin W'$ with $z_i = 0$ is to meet with case (II); in this situation i is the eliminated index, and $d_i = -r_i$ is positive by construction.

Now we can perform a step from z in the direction d, thus passing to the new iterate

$$z^+ = z + \gamma^* d, \quad y^+ = y - \gamma^* B^{-1} C d.$$

The stepsize should, of course, be chosen to minimize the objective ϕ along the search direction; we, anyhow, should impose on the stepsize certain upper bound to maintain feasibility of the updated pair (y^+, z^+). By construction, this pair, independently of the stepsize, will satisfy the linear equations of the problem, so that all we need is to ensure nonnegativity of y^+ and z^+. The corresponding upper bound $\bar{\gamma}$ on the stepsize is given by

$$\bar{\gamma} = \sup\{\gamma \mid z + \gamma d \geq 0,\ y - \gamma^* B^{-1} C d \geq 0\}.$$

We can easily compute this bound (compare with the Gradient Projection method for the case of linear constraints); the important observation is that this bound is positive. Indeed, $y > 0$ by (β), so that the requirement that y^+ should be nonnegative is compatible with small enough positive stepsizes. Similarly, the entries of z with indices $\notin W$ are positive, and the requirement that the corresponding entries in z^+ should be nonzero also is compatible with small enough positive stepsizes. The entries of z_i with $i \notin W'$ are zero, same as the corresponding entries of d_i, so here all positive stepsizes are admissible. If there are independent variables z_i which do not fall into the two considered categories (i.e., those with $i \in W$ and $i \notin W'$), it means that $W' \neq W$, which may happen only when the case (II) was met. It this case,

the only index in $W\backslash W'$ is the one eliminated in (II); for this index by construction $z_i = 0$ and $d_i > 0$, so that here again all positive stepsizes are admissible.

Now we can choose γ^*, e.g., by "restricted minimization" of the objective along the direction d:

$$\gamma^* \in \mathrm{Argmin}\{\phi(z + \gamma d) \mid 0 \leq \gamma \leq \bar{\gamma}\},$$

and the corresponding step will for sure improve the value of the objective.

3) After the step $x = (y, z) \mapsto x^+ = (z^+, y^+)$ is performed, we update, if necessary, our structures – the partitioning of the variables into dependent and independent ones and the working set. Namely,

- It may happen that some entries in y^+ are zero. If it is the case, we update the partition in order to make the new dependent variables of x^+ positive (as we know, it is possible due to the nondegeneracy assumption). With this new partition, we declare the new working set W^+ to be comprised of the indices of zero entries of the independent components of x^+ (we do not exclude $W^+ = \emptyset$).
- It may happen that all entries of y^+ are positive, but some entries of z^+ with indices not in W' become zero. If it happens, we preserve the partition and extend the working set W' by adding to it the indices of the zero entries of z^+.
- Last, it may happen that all entries of y^+, same as all entries of z^+ with indices not in W' are positive (it means that $\gamma^* < \bar{\gamma}$). Here we preserve the partition, and our new working set is W'.

It is easily seen that the new iterate, along with the new partition and new working set, satisfies requirements (α) - (γ), and we can iterate the process.

9.3.3 Geometry of Gradient Projection and Reduced Gradient methods in linearly constrained case

The geometry of our actions in the linearly constrained case, for both the Gradient Projection and Reduced Gradient methods, is quite clear. In both the methods, we travel along the feasible polyhedron of our problem. At each step, we are at a point of the relative interior of certain facet of the polyhedron (this facet is the feasible part of our current working plane) and are trying to improve the objective by moving along certain descent direction of it in the facet. The only difference between the methods is how this descent direction is determined. In the Gradient Projection method, the direction is simply the projection of the antigradient of f onto the working plane. In the Reduced Gradient method, we parameterize somehow the points of the working plane and take the gradient of the objective with respect to the parameters. It can be easily seen that this is the same as to compute the "full-dimensional" gradient of the objective *with respect to certain new Euclidean metric on* $\mathbf{R}^n$ and to project the resulting vector on the plane, of course, to project in the same new Euclidean metric. After the descent direction is formed, we can meet with several possibilities:

- the direction in fact is <u>not</u> descent – it is zero. If it happens, we are at the KKT point of the objective on the working plane – the antigradient of the objective is linear combination of the linear forms defining the plane. If the coefficients in this combination are of proper signs (i.e., the coefficients corresponding to the linear forms coming from inequality constraints are nonnegative), the point in question is a KKT point of the problem, and we terminate. If it is not the case, then we replace our current working facet with larger one (which

is obtained when eliminating the linear equality associated with negative coefficient); the descent direction we get in this larger facet turns out to be nonzero, i.e., indeed descent, and we proceed with this larger facet. Note that in the case in question the new search direction "looks inwards the new facet" – we can perform certain positive step in this direction, not leaving this facet.

Thus, we either terminate with a KKT point of the problem, or find ourselves in the situation when the search direction is descent for the objective and certain nontrivial step in this direction keeps us in the facet (which now is not necessary the one with which we started the iteration – the facet might have been enlarged!).

- If we are in the just mentioned "good" situation – either from the very beginning of the iteration, or after "bad" constraint is eliminated and the initial facet is replaced with a larger one – we perform linesearch along the descent direction we have. In this linesearch we impose on the stepsize the restriction not to push the point out of the facet we now have. After the stepsize if found and the point is shifted, we either find ourselves on the "facet of the facet" – several new linear constraints which were not previously involved into the description of the working plane now have become active – or no new constraints are met. In the first case, we replace the facet with the smaller one given by all constraints of the problem active at the new iterate, in the second – proceed with the facet we have.

Linear Programming case

In the case of *linear* objective and constraints – i.e., in the Linear Programming case – what we do in both methods are just the simplex iterations. More exactly, it becomes so after several initial steps – we should not necessarily start with the working set resulting in a vertex of the feasible polyhedron! Our behaviour at the beginning of the process is as follows. Assume that we start in the relative interior of the feasible set – no inequality constraints are active at the starting point x, and let, for the sake of simplicity, the feasible set be bounded. Our first step is to choose somehow descent direction of the objective in the plane given by the equality constraints. Then we minimize the objective along this direction within the feasible set. Since the objective is linear, the result x' of this minimization will belong to the relative boundary of the feasible set – to certain facet of it. Now we repeat the same construction in the facet and, as a result, come to a facet of smaller dimension and improve the objective value, and so on. In no more than k steps, k being the number of inequalities in the problem, we arrive at a vertex of the feasible polytope[2]. Starting with this moment, we indeed perform simplex iterations: our working plane is a point, so that we for sure are at the KKT point of the objective in this working plane. If the vertex where we are is not optimal, then our Active Set strategy enforces us to replace the point by a one dimension larger facet; this enlarged facet will be of dimension 1, i.e., it will be an edge of the feasible polytope. When minimizing the objective over this edge, as it is predicted by our Active Set strategy, we move to another vertex with better value of the objective, and so on, until the optimal vertex is reached; this is exactly what the Simplex Method does.

The "preliminary phase" of the Gradient Projection method in the LP case – the one before the pure simplex iterations are started – is called *purification*. This is an important algorithm – it

[2]we ignore the "degenerate case" when our current facet is a non-vertex one, but the gradient of the objective is orthogonal to the plane of the facet. We could easily handle this case as well by saying that if it happens, then the search direction we currently use is an arbitrary direction along the plane of the facet

allows to transform efficiently an arbitrary feasible solution to an LP program into a basic feasible solution with better (or at least the same) value of the objective. (When saying "efficient", I mean that the updating takes at most k Gradient Projection steps, and each step requires at most quadratic in $m + n + k$ number of arithmetic operations.) The necessity in such a transformation occurs when we solve linear programs by *interior point methods* where all the iterates strictly satisfy all the inequality constraints. After an approximate solution of this type is found, we may use purification to improve it; if solution is good enough, then purification will transform this *approximate* solution into the *exact* one.

9.4 Linearly Constrained Quadratic Programming program

We already know that in the Linear Programming case the Active Set methods become the Simplex Method – one of the most frequently used and most powerful methods for LP. There is another favourable situation for the Active Set strategy – the one when the constraints are *linear* and the objective is quadratic (*Linearly Constrained Quadratic Programming program – LCQP*). This is a particular case, of course, but it is of great importance by its own right and also due to the fact that LCQP programs arise, as auxiliary subproblems, in one of the most powerful practical methods for constrained minimization – in Sequential Quadratic Programming (SQP) which we will consider in the mean time.

Assume that the constraints are linear and the objective is quadratic

$$f(x) = \frac{1}{2}x^T Hx - d^T x$$

with positive definite Hessian H.

At a step t of the method, given current working set W, the corresponding working plane S_W and a belonging to the plane feasible for the original problem solution x_t, we can explicitly point out the optimum x^W of the objective over S_W. Indeed, the plane can be represented by a system of linear equations

$$A_W x = b_W,$$

with A_W being $m_W \times n$ matrix of full row rank; this system comes from the original system of linear equality and inequality constraints by eliminating some of the inequalities and replacing the remaining inequalities (those with indices in W) by equalities.

The KKT system for the problem

$$(\mathrm{P}_W) \qquad f(x) \to \min \mid A_W x = b_W$$

is the square linear system

$$\begin{array}{rcl} Hx + A_W^T \lambda & = & d \\ A_W x & = & b_W \end{array}$$

From the fact that H is positive definite and A_W is of full row rank (recall that all problems we consider are assumed to be regular!) it immediately follows that the system is nonsingular. The x-component of the solution to the system is the unique global minimizer x^W of f on the working plane (we deal with regular convex program, so that the KKT conditions are necessary and sufficient for optimality), while the λ-component of the solution λ^W is the corresponding vector of Lagrange multipliers.

If (case 1) x^W is feasible for the original problem and the entries of λ^W associated with the inequality constraints of the original system are nonnegative, then x^W is KKT point of the

original problem, and λ^W is, essentially, the vector of Lagrange multipliers for this problem (all we need is to set to 0 the Lagrange multipliers associated with inequality constraints not in the working set).

Now, if (case 2) x^W is feasible for the original problem, but some of the entries in λ^W associated with the inequality constraints of the original problem are negative, we may eliminate from the the working set the index of any one of these constraints. As we know from our general considerations, there will be a feasible for the original problem point on the new (extended by one dimension) working plane with the value of the objective less than at x^W; consequently, the optimal value of the objective on the feasible (for the original problem) part of the new working plane also will be less than the optimal value of the objective at the current working plane. Our new iterate in case 2 will be $x_{t+1} = x^W$.

Last, it is possible (case 3) that x^W is not feasible for the original problem. Looking at the constraints, we can immediately identify the last point x_t^+ of the segment $[x_t, x^W]$ (note that this segment belongs to the current working plane; it starts within and ends outside the feasible set of the original problem) which is still feasible for the original problem. At this point, some of the inequality constraints with indices not in the current working set are active; as always, we add their indices to the working set, thus updating it, and define the new iterate as $x_{t+1} = x_t^+$.

Thus, at the step in question we either terminate with a KKT point of the problem, i.e., with a globally optimal solution to it (we deal with convex program, so that the KKT condition is sufficient for optimality), or update somehow the working set and the iterate (cases 2 and 3). Having updated the working set and the iterate, we loop – apply the same actions to the new working set and iterate, etc.

An extremely good news about the presented Active Set method is that it finds the global solution to the problem in question in finite number of steps; thus, we are in the situation when all the idealizations of the Working Set Theorem are realities! The proof of indicated statement goes along the lines of the one of the Working Set Theorem, and it is part of the assignment accompanying this lecture.

Concluding comments. As we are used to do, let us summarize the advantages and disadvantages of the Primal methods, specifically the Projected Gradient and the Reduced Gradient ones. The advantages of the methods are in their generality and especially in the fact that they generate *feasible* solutions with *improving* from step to step values of the objective; there are cases when these properties are of great importance for the end-user.

The main disadvantage of the Projected Gradient and Reduced Gradient methods is that in practice they are very slow as applied to *nonlinearly constrained* problems. The reason for this is clear: when there are nonlinear inequality constraints, the methods are enforced to travel along "curved" surfaces; consequently, we are unable to avoid projections – iterative returns to a tight neighbourhood of the current working surface. In order to ensure reasonable complexity of the projections in the Gradient Projection method, we are enforced to perform relatively short steps along the tangent plane to the current working surface – this is basically the only possibility to get a good starting point for the projection subroutine; as a result of these short steps, the method may too slowly approach the KKT set. Similar difficulties occur in the Reduced Gradient method as applied to a nonlinearly constrained problem.

In contrast to this, the *linearly constrained case* is much better suited for the methods in question, and the Gradient Projection and Reduced Gradient routines are quite reasonable and frequently used algorithms for practical minimization under linear constraints. We have also have seen that in two important particular cases of linearly constrained problems – those with linear and convex quadratic objectives – there exist specific versions of the Active Set methods

which solve the problems in question in finitely many steps; the Active Set methods definitely are the best traditional routines for LP and convex Linearly Constrained Quadratic Programming.

Last, in the linearly constrained case there is no necessity to minimize the objective over the current working plane by the Gradient Descent, as it is done in the Gradient Projection and, in fact, also in the Reduced Gradient method (with the only difference that here the Gradient Descent deals with certain parameterization of the feasible plane, or, which is the same, with certain specific Euclidean structure in the plane). As we know, there are much more efficient methods for unconstrained minimization, e.g., the Newton and the quasi-Newton ones. It is clear from the discussion in the beginning of Section 9.2.1 that all these methods can be used, without conceptual difficulties, for minimization under linear equality constraints and, consequently, may be used as "working horses" in Active Set methods, thus giving rise to efficient algorithms of this latter type.

Assignment # 9 (Lecture 9)

Exercise 9.4.1 *Convert the problem (cf. (9.1.3)*

$$c^T d \to \min \mid \; Ad \le 0, \; \max_i |d_i| \le 1 \quad [d \in \mathbf{R}^n]$$

into an LP program.

Are you able to perform similar conversion when the constraint

$$\max_i |d_i| \le 1$$

is replaced with

$$\sum_i |d_i| \le 1 \qquad ?$$

Exercise 9.4.2 *Let L be a linear subspace in $\mathbf{R}^n$ given by the system of linear homogeneous equations*

$$Ax = 0$$

(A is $m \times n$ matrix of full row rank). Prove that the matrices

$$P = I - A^T[AA^T]^{-1}A, \quad Q = A^T[AA^T]^{-1}A$$

are the matrices of orthogonal projectors of $\mathbf{R}^n$ onto L and onto the orthogonal complement $L^\perp$ of L, respectively (i.e., that $Px \in L$ for every x with $Px - x$ being orthogonal to L, and that $Px = x$ for $x \in L$; the same for Q, with L replaced with $L^\perp$). [compare with (9.2.2)]

Exercise 9.4.3 *Prove that the Active Set method from Section 9.4 as applied to a regular linearly constrained Quadratic Programming program with strongly convex objective indeed finds the exact solution to the problem in finitely many steps.*

Hint. 1) Since the objective is strictly convex, all the minimizers of the objective over the working planes are well-defined; consequently, the method also is well-defined.

2) The only possibility for the method to terminate is to find a KKT point (thus, global optimal solution) of the problem. Therefore all you need is to demonstrate that the method is finite. Assume that it is not the case and lead this assumption to contradiction, using the following arguments:

- there could not be too long sequences of steps where only case (3) takes place; thus, case (2) should occur infinitely many times
- the values of the objective along the sequence of iterates x_1, x_2,... never increase
- if at a step t case (2) takes place, then the objective will be strictly increased at step $t+1$

Derive from these observations that it is impossible to meet twice case (2) with the same working set.

Lecture 10

Penalty and Barrier Methods

This lecture is devoted to the *penalty* and the *barrier* methods; as far as the underlying ideas are concerned, these methods implement the simplest approach to constrained optimization – approximate a constrained problem by unconstrained ones. Let us look how it is done.

10.1 The idea

10.1.1 Penalty methods: equality constrained case

To get the idea of the construction, consider an equality constrained problem

$$\text{(ECP)} \quad f(x) \to \min \mid h_i(x) = 0,\ i = 1, ..., m \quad [x \in \mathbf{R}^n].$$

In order to approximate this constrained problem by an unconstrained one, let us add to our objective a term which "penalizes" violation of constraints; the simplest term of this type is

$$\frac{1}{2}\rho \sum_{i=1}^{m} h_i^2(x),$$

where $\rho > 0$ is "penalty parameter". This term is zero on the feasible set and is positive outside this set; if ρ is large, then the penalizing term is large everywhere except tight neighbourhood of the feasible set.

Now let us add the penalty term to the objective. From the above discussion it follows that the resulting "combined objective"

$$f_\rho(x) = f(x) + \frac{1}{2}\rho \sum_{i=1}^{m} h_i^2(x) \tag{10.1.1}$$

possesses the following properties:

- at the feasible set, it coincides with the actual objective;
- for large ρ, is is large outside tight neighbourhood of the feasible set (indeed, outside such a neighbourhood the penalty term becomes larger and larger as the penalty parameter grows, while the objective remains as it was).

From these properties it immediately follows that

$$\lim_{\rho\to\infty} f_\rho(x) = \begin{cases} f(x), & x \text{ feasible} \\ +\infty, & \text{otherwise} \end{cases};$$

Thus, we could say that the limit of f_ρ as $\rho \to \infty$ is the function taking values in the extended real axis (with $+\infty$ added) which coincides with f on the feasible set and is $+\infty$ otherwise this set; it is clear that *unconstrained* local/global minimizers of this limit are exactly the *constrained* local, respectively, global minimizers of f. This *exact coincidence* takes place only in the limit; we could, anyhow, expect that "close to the limit", for large enough values of the penalty parameter, the unconstrained minimizers of the penalized objective are close to the constrained minimizers of the actual objective. Thus, solving the unconstrained problem

$$f_\rho(x) \to \min$$

for large enough value of ρ, we may hope to get good approximations to the solutions of (ECP). As we shall see in the mean time, under mild regularity assumptions all these "could expect" and "may hope" indeed take place.

10.1.2 Penalty methods: general constrained case

The idea of penalization can be easily carried out in the case when there are inequality constraints as well. Given a general type constrained problem

$$\text{(GCP)} \quad f(x) \to \min \mid h_i(x) = 0,\ i = 1, ..., m,\ g_j \le 0,\ j = 1, ..., k,$$

we could penalize the inequality constraints by the term

$$\frac{1}{2}\rho \sum_{j=1}^{k} (g_j^+(x))^2,$$

where

$$a^+ = \max[a, 0] = \begin{cases} a, & a \ge 0 \\ 0, & a < 0 \end{cases}$$

is the "positive part" of a real a. The resulting penalty term is zero at any point where all the inequality constraints are satisfied and is positive (and proportional to ρ) at any point where at least one of the inequalities is violated.

Adding to the objective penalty terms for both equality and inequality constraints, we come to the penalized objective

$$f_\rho(x) = f(x) + \frac{1}{2}\rho \sum_{i=1}^{m} h_i^2(x) + \frac{1}{2}\rho \sum_{j=1}^{k} (g_j^+(x))^2; \tag{10.1.2}$$

same as above, we can expect that the unconstrained minimizers of the penalized objective approach the constrained minimizers of the actual objective as the penalty parameter goes to infinity. Thus, solutions of the unconstrained problem

$$f_\rho(x) \to \min,$$

for large ρ, are good approximations to the solutions of (GCP).

10.1.3 Barrier methods

The idea of the *barrier* methods is similar; the only difference is that instead of allowing violations of the constraints and punishing these violations, we now prevent the constraints to be violated by a kind of interior penalty which blows up to infinity as a constraint is about to be violated. This "interior penalty" approach can be normally used in the case of *inequality constrained* problems with "full-dimensional" feasible set. Namely, consider an inequality constrained problem

$$\text{(ICP)} \quad f(x) \to \min \mid g_j(x) \le 0,\ j = 1, ..., k,$$

and assume that the feasible domain G of the problem is such that

- the interior int G of the domain G is nonempty, and every g_j is strictly negative in int G
- every point from G can be represented as the limit of a sequence of points from the interior of G

Assume also, just for the sake of simplicity, that G is bounded.

Given problem with the indicated properties, one can in many ways define an *interior penalty function* (also called a *barrier*) for the feasible domain G, i.e., a function F defined on the interior of G and such that

- F is continuously differentiable on int G
- $F(x_i) \to \infty$ for any sequence of points $x_i \in$ int G converging to a boundary point x of G

E.g., one can set

$$F(x) = \sum_{j=1}^{k} \frac{1}{-g_j(x)}$$

(*the Carrol barrier*), or

$$F(x) = -\sum_{j=1}^{k} \ln(-g_j(x))$$

(*the logarithmic barrier*), or something else.

Now consider the following "aggergate":

$$F_\rho(x) = f(x) + \frac{1}{\rho} F(x),$$

where $\rho > 0$ is penalty parameter. The function F_ρ is well-defined and smooth on int G and goes to ∞ along every sequence of points from int G converging to a boundary point of G (indeed, along such a sequence F possesses the required behaviour, while f remains bounded due to continuity of the objective). In particular, the level sets of F_ρ – the sets of the type

$$\{x \in \text{int } G \mid F_\rho(x) \le a\}$$

are closed[1)]. Since they are also bounded (recall that G is assumed to bounded), they are compact sets, and F_ρ, being continuous on such a compact set, attains its minimum on it; the

[1]indeed, to prove closedness of a level set, let it be called L, is the same as to prove that if $f_\rho(x_i) \le a < \infty$ for certain sequence of points $\{x_i\}$ converging to a point x, then $F_\rho(x) \le a$ (a closed set is by definition, the one which contains the limits of all converging sequences comprised of elements of the set). A priori x might be either an interior, or a boundary point of G. The second possibility should be excluded, since if it is the case, then, due to already indicated properties of F_ρ, it would be $F_\rho(x_i) \to \infty$ as $i \to \infty$, which is impossible, since F_ρ is above bounded on every level set. Thus, $x \in$ int G; but then F_ρ is continuous at x, and since $F_\rho(x_i) \le a$ and $x_i \to x$, $i \to \infty$, we get $F_\rho(x) \le a$, as required.

corresponding minimizer clearly is a minimizer of F_ρ on int G as well.

Thus, for every positive ρ the function F_ρ attains its minimum on int G. At the same time, when the penalty parameter ρ is large, F_ρ "almost everywhere in int G" is "almost equal" to f – indeed, due to the factor $\frac{1}{\rho}$ at F in F_ρ, the contribution of the interior penalty term for large ρ is not negligible only in a thin, the smaller the larger is ρ, neighbourhood of the boundary. From this observation it is natural to guess (and it turns out to be indeed true) that the minimizers of F_ρ on int G are, for large ρ, close to the optimal set of (ICP) and could therefore be treated as good approximate solutions to (ICP).

Now, the problem

$$F_\rho(x) \to \min$$

is, formally, a constrained problem – since the domain of the objective is int G rather than entire $\mathbf{R}^n$. Nevertheless, we have basically the same possibilities to solve the problem as if it was unconstrained. Indeed, any *descent* (i.e., forming a sequence of iterates along which the objective never increases) method for unconstrained minimization, as applied to F_ρ and started at an interior point of G, never will come too close to the boundary of G (since, as we know, close to the boundary F_ρ is large, and along the trajectory of the method F_ρ is not greater than at the starting point). It means that the behaviour of the method as applied to F_ρ will, basically, be the same as if F_ρ was defined everywhere – the method simply will not feel that the objective is only partially defined. Thus, the barrier scheme in fact reduces the constrained minimization problem to an unconstrained one (or, better to say, allows to approximate the constrained problem by "essentially unconstrained" problem).

After the ideas of penalty and barrier schemes are outlined, let us come to more detailed investigation of the schemes.

10.2 Penalty methods

Let us investigate the penalty scheme in more details. The main questions we should focus on are

- Whether indeed unconstrained minimizers of the penalized objective f_ρ converge, as $\rho \to \infty$, to the solutions of the constrained problem?
- What are our possibilities to minimize the penalized objective?

For the sake of definiteness, let us focus on the case of equality constrained problem (ECP) (the results for tthe general case are similar).

10.2.1 Convergence of the penalty scheme

The first – and very simple – statement is as follows:

Theorem 10.2.1 *Let the objective f in problem (ECP) possess bounded level sets:*

$$f(x) \to \infty, \ |x| \to \infty,$$

and let (ECP) be feasible. Then, for any positive ρ, the set of global minimizers $X^*(\rho)$ of the penalized objective f_ρ is nonempty. Moreover, if X^* is the set of globally optimal solutions to

(ECP), then, for large ρ, the set X^*_ρ is "close" to X^*: for any $\epsilon > 0$ there exists $\rho = \rho(\epsilon)$ such that the set $X^*(\rho)$, for all $\rho \geq \rho(\epsilon)$, is contained in ϵ-neighbourhood

$$X^*_\epsilon = \{x \mid \exists x^* \in X^* : \ |x - x^*| < \epsilon\}$$

of the optimal set of (ECP).

Proof. First of all, (ECP) is solvable. Indeed, let x_0 be a feasible solution to the problem, and let U be the corresponding level set of the objective:

$$U = \{x \mid \ f(x) \leq f(x_0)\}.$$

By assumption, this set is bounded and, due to continuity of f, is closed; therefore it is compact. Further, the feasible set S of the problem also is closed (since the constraints are continuous); consequently, the set

$$U_f = U \cap S$$

– the set of all feasible solutions not worse, in terms of the values of f, then the feasible solution x_0 – is bounded and closed, therefore is compact (and nonempty – it contains x_0). It is clear that the original problem is equivalent to the one of minimizing the objective over U_f, and the latter problem, being a problem of minimizing a continuous function on compact set, is solvable.

By similar reasons, every unconstrained problem

$$(\mathrm{P}_\rho) \quad f_\rho(x) \to \min$$

also is solvable. I claim that

- the optimal value f^*_ρ of (P_ρ) is not greater than the optimal value f^* in (ECP);
- if x^*_ρ is an optimal solution to (P_ρ), then

$$f(x^*_\rho) \leq f^*; \tag{10.2.1}$$

- the optimal set $X^*(\rho)$ of (P_ρ) is contained in U.

Indeed, if x^* is an optimal solution to (ECP), then

$$f_\rho(x^*) = f(x^*) = f^*,$$

so that the optimal value in (P_ρ) can be only $\leq$ the one in (ECP), which justifies the first claim. Further, due to nonnegativity of the penalty term we have

$$f(x^*_\rho) \leq f_\rho(x^*_\rho) = \min_x f_\rho(x) \leq f_\rho(x^*) = f^*,$$

which justifies the second claim. And this second claim immediately implies that $x^*_\rho \in U$ by construction of U.

Our observations immediately result in the desired conclusions. Indeed, we already have proved that $X^*(\rho)$ is nonempty, and all we need is to verify that, for large ρ, the set $X^*(\rho)$ is contained in tight neighbourhood of X^*. Assume that it is not the case: there exist positive ϵ and a sequence $\rho_i \to \infty$ such that $X^*(\rho_i)$ is not contained in X^*_ϵ, so that one can choose $x^*_i \in X^*(\rho_i)$ in such a way that x^*_i is outside X^*_ϵ. According to the third claim, the points x^*_i belong to U and form therefore a bounded sequence. Passing to a subsequence, we may assume

that x_i^* converge to certain point x as $i \to \infty$. I claim that x is an optimal solution to (ECP) (this will give us the desired contradiction: since $x_i^* \to x \in X^*$, the points x_i^* for all large enough i must be in X_ϵ^* – look at the definition of the latter set – and we have chosen x_i not to be in X_ϵ^*). To prove that x is an optimal solution to (ECP), we should prove that $f(x) \leq f^*$ and that x is feasible. The first inequality is readily given by (10.2.1) – it should be satisfied for $x_\rho^* = x_i^*$, and the latter points converge to x; recall that f is continuous. Feasibility of x is evident: otherwise $h(x) \neq 0$ and, since $x_i^* \to x$ and h is continuous, for all large enough i one has

$$|h(x_i^*)| \geq a = \frac{1}{2}|h(x)| > 0,$$

whence for these i

$$f_{\rho_i}^* = f_{\rho_i}(x_i^*) = \frac{1}{2}\rho_i |h(x_i^*)|^2 + f(x_i^*) \geq \frac{a}{2}\rho_i + f(x_i^*) \to \infty, \ i \to \infty$$

(note that $\rho_i \to \infty$, while $f(x_i^*) \to f(x)$), which contradicts our first claim. ∎

The formulated theorem is not that useful: we could conclude something reasonable from its statement, if we were able to approximate the *global* solutions to (P_ρ), which is the case only when f_ρ is convex (as it, e.g., happens when f is convex and the equality constraints are linear). What we indeed need is a *local* version of the theorem. This version is as follows:

Theorem 10.2.2 *Let x^* be a nondegenerate locally optimal solution to (ECP) (see Definition 8.2.2). Then there exists a neighbourhood V of x^* (an open set containing x^*) and $\bar{\rho} > 0$ such that for every $\rho \geq \bar{\rho}$ the penalized objective f_ρ possesses in V exactly one critical point $x^*(\rho)$. This point is a nondegenerate local minimizer of f_ρ and a minimizer of f_ρ in V, and $x^*(\rho) \to x^*$ as $\rho \to \infty$.*

Proof [non-obligatory]. The simplest way to prove the theorem is to reduce the situation to the case when the constraints are linear. This can be done as follows: since x^* is nondegenerate, the gradients of the constraints at x^* are linearly independent. From the appropriate version of the Implicit Function Theorem (we again use this magic Calculus tool) it follows that you can choose locally new coordinates y (nonlinearly related to the original ones!) in which our m constraints will be simply the first coordinate functions. Namely, there exist

- a neighbourhood V' of the point x^*
- a neighbourhood W' of the origin in $\mathbf{R}^n$
- a one-to-one mapping $x = X(y)$ from W' onto V' with inverse $y = Y(x)$

such that

- $X(\cdot)$ and $Y(\cdot)$ are continuously differentiable as many times as the constraints h (i.e., at least twice; recall that we once for ever restricted our considerations to problems with twice continuously differentiable data)
- $x^* = X(0)$ ("x^* in the coordinates y becomes the origin")
- $h_i(X(y)) \equiv y_i$, $i = 1, ..., m$ ("in y-coordinates the constraints become the first m coordinate functions").

Now let us pass in (ECP) from coordinates x to coordinates y, which results in the problem

$$(\mathrm{ECP}') \quad \phi(y) \equiv f(X(y)) \to \min \mid y_i \equiv h_i(X(y)) = 0,\ i = 1, ..., m;$$

this, of course, makes sense only in the neighbourhood W' of the origin in y-variables.

1^0. I claim that $y^* = 0$ is nondegenerate solution to the problem (ECP$'$). Indeed, we should prove that this is a regular point for the constraints in the problem (which is evident) and that the Second Order Sufficient Optimality condition takes place at the point, i.e, there exist λ^* such that the Lagrange function

$$L'(y, \lambda) = \phi(y) + \sum_{i=1}^{m} \lambda_i y_i$$

satisfies

$$\nabla_y L'(y^*, \lambda^*) = 0$$

and

$$d^T \nabla_y^2 L'(y^*, \lambda^*) d > 0$$

for every nonzero vector d which is orthogonal to the gradients of the constraints of (ECP$'$). And what we know is that there exists λ^* which ensures similar properties of the Lagrange function

$$L(x, \lambda) = f(x) + \sum_{i=1}^{m} \lambda_i h_i(x)$$

of the original problem at $x = x^*$. What we shall prove is that the latter λ^* satisfies all our needs in (ECP$'$) as well. Indeed, we clearly have

$$L'(y, \lambda^*) = L(X(y), \lambda^*),$$

whence, denoting by $X'(y)$ the $n \times n$ matrix of derivative of the mapping $X(\cdot)$ at y,

$$\nabla_y L'(y^*, \lambda^*) = [X'(0)]^T \nabla_x L(x^*, \lambda^*) = [X'(0)]^T 0 = 0$$

– $y^* = 0$ satisfies the first order part of the Optimality condition.

Further, $d^T \nabla_y [h(X(y))] = d^T [X'(y)]^T \nabla_x h(X(y))$, so that d is orthogonal to the gradients of $y_i \equiv h_i(X(y))$, $i = 1, ..., m$, if and only if $\bar{d} = X'(0)d$ is orthogonal to $\nabla h_i(x^*)$, $i = 1, ..., m$. Note also that $X'(0)$ is nonsingular (differentiate the identity $Y(X(y)) \equiv y$ to get $[X'(0)]^{-1} = Y'(x^*)$), so that d is nonzero if and only if $X'(0)d$ is.

Now let d be a nonzero vector which is orthogonal to the gradients of the constraints of (ECP$'$). As it was just explained, $\bar{d} = X'(0)d$ is a nonzero vector orthogonal to the gradients of h_i at x^*, and we have

$$d^T \nabla_y^2 L'(y^*, \lambda^*) d = d^T [\nabla_y^2|_{y=y^*} L(X(y), \lambda^*)] d =$$

[differentiating twice the superposition in direction d]

$$d^T [X'(0)]^T \nabla_x^2 L(x, \lambda^*) X'(0) d + [\nabla_x L(x^*, \lambda^*)]^T \frac{d^2}{dt^2}|_{t=0} X(td) =$$

[the second term is zero due to the origin of λ^*]

$$= \bar{d}^T \nabla_x^2 L(x^*, \lambda^*) \bar{d} > 0$$

[again due to the origin of λ^*], and we see that the second order part of the required conditions also is satisfied. ∎

2^0. Now note that (ECP′) is a problem with linear constraints, so that the Hessian with respect to y of the Lagrangian of the problem, independently of the values of the Lagrange multipliers, is $\nabla_y^2\phi(y)$. In particular, the "second-order part" of the fact that $y^*=0$ is nondegenerate solution to (ECP′) simply says that $H=\nabla_y^2\phi(y^*)$ is positive definite on the plane

$$L=\{d \mid d_i=0,\ i=1,...,m\}$$

(this the tangent plane at y^* to the feasible surface of the problem). This is the key argument in the proof of the following crucial fact:

(*) *function*

$$\phi_\rho(y)=\phi(y)+\frac{1}{2}\rho\sum_{i=1}^m y_i^2$$

– the penalized objective of (ECP′) – is, for large enough ρ, strictly convex in a properly chosen convex neighbourhood W of $y^*=0$, i.e., there exists small enough convex neighbourhood $W\subset W'$ of $y^*=0$ (simply an open ball of certain small enough positive radius) and $\rho^*>0$ such that

$$\nabla_y^2\phi_\rho(y)$$

is positive definite whenever $y\in W$ and $\rho\geq\rho^*$.

The proof of (*) goes through the following simple

Lemma 10.2.1 *Let A be a symmetric $n\times n$ matrix and L be a linear subspace in $\mathbf{R}^n$, and let P be the orthoprojector on L. Assume that matrix A is positive definite on L:*

$$d^TAd\geq\alpha|d|^2\quad\forall d\in L$$

with certain $\alpha>0$. Then there exists ρ^* such that the matrix

$$A+\rho(I-P)$$

is positive definite whenever $\rho\geq\rho^*$ and is such that

$$d^T(A+\rho(I-P))d\geq\frac{\alpha}{2}|d'|^2+\frac{\rho}{2}|d''|^2\quad\forall d\in\mathbf{R}^n,$$

$d'=Pd$ and $d''=(I-P)d$ being the projections of d onto L and onto the orthogonal complement of L, respectively.

Moreover, ρ^* can be chosen depending only on α and on an upper bound β for the norm

$$|A|=\max_{d:|d|\leq1}|Ad|$$

of the matrix A.

Proof. Let $\beta\geq|A|$ and $\gamma>0$. We have

$$d^T(A+\rho(I-P))d=d^TAd+\rho|d''|^2=(d'+d'')^TA(d'+d'')+\rho|d''|^2=$$

$$=(d')^TAd'+2(d')^TAd''+(d'')^TAd''+\rho|d''|^2\geq$$

[since $(d')^T A d' \geq \alpha |d'|^2$ and, by Cauchy's inequality,

$$|2(d')^T A d''| = 2|d'||Ad''| \leq 2|A||d'||d''| \leq 2\beta |d'||d''| \leq \frac{\beta}{\gamma}|d'|^2 + \beta\gamma |d''|^2$$

(note that $|2uv| \leq \gamma^{-1}u^2 + \gamma v^2$ – this is nothing but the inequality between the arithmetic and the geometric mean)]

$$\geq \alpha |d'|^2 - \frac{\beta}{\gamma}|d'|^2 - \beta\gamma |d''|^2 + \rho |d''|^2.$$

Setting here $\gamma = \frac{2\beta}{\alpha}$, we get

$$d^T(A + \rho(I-P))d \geq \frac{\alpha}{2}|d'|^2 + [\rho - \frac{2\beta^2}{\alpha}]|d''|^2;$$

choosing finally $\rho^* = \frac{4\beta^2}{\alpha}$ and assuming $\rho \geq \rho^*$, so that $\rho - \frac{2\beta^2}{\alpha} \geq \frac{\rho}{2}$, we come to

$$d^T(A + \rho(I-P)d \geq \frac{\alpha}{2}|d'|^2 + \frac{\rho}{2}|d''|^2$$

for all $\rho \geq \rho^*$. ∎

Now we are ready to prove (*). Indeed, since $\nabla^2\phi(y)$ is continuous in y and $\nabla^2\phi(y^*)$ is positive definite on

$$L = \{d \mid \ d_i = 0,\ i = 1, ..., m\},$$

we could choose

- small enough ball W centered at $y^* = 0$ and contained in W'
- small enough positive α
- large enough positive β

such that

$$d^T[\nabla^2\phi(y)]d \geq \alpha |d|^2,\ y \in W, d \in L,$$

P being the orthoprojector on L, and

$$|\nabla^2\phi(y)| \leq \beta,\ y \in W.$$

Now note that

$$\nabla_y^2 \phi_\rho(y) \equiv \nabla_y^2 \left[\phi(y) + \frac{\rho}{2}\sum_{i=1}^m y_i^2\right] = \nabla^2\phi(y) + \rho(I-P),$$

P being the orthoprojector onto L. According to Lemma 10.2.1, there exists $\rho^* > 0$ such that all the matrices

$$\nabla^2\phi_\rho(y)$$

corresponding to $y \in W$ and $\rho \geq \rho^*$ are positive definite, moreover, satisfy

$$d^T[\nabla^2\phi_\rho(y)]d \geq \frac{\alpha}{2}|d'|^2 + \frac{\rho}{2}|d''|^2,\ \ d' = Pd,\ d'' = (I-P)d. \tag{10.2.2}$$

Thus, whenever $\rho \geq \rho^*$, the Hessian of the function ϕ_ρ is positive definite in W, so that ϕ_ρ is convex in W.

4^0. Let $\rho \geq \rho^*$. From (10.2.2) it immediately follows that

$$\phi_\rho(y) \geq \phi_\rho(0) + y^T \nabla_y \phi(0) + \frac{\alpha}{4}|y'|^2 + \frac{\rho}{4}|y''|^2, \;\; y' = Py,\; y'' = (I-P)y''. \tag{10.2.3}$$

The gradient of ϕ_ρ at $y^* = 0$ is, first, independent of ρ (it is simply the gradient of ϕ at the point) and, second, is orthogonal to L, as it is given by the "first order part" of the fact that $y^* = 0$ is a nondegenerate solution to (ECP′). Consequently, (10.2.3) can be rewritten as

$$\phi_\rho(y) \geq \phi_\rho(0) + (y'')^T g + \frac{\alpha}{4}|y'|^2 + \frac{\rho}{4}|y''|^2, \tag{10.2.4}$$

$g = \nabla_y \phi(y^*)$ being a once for ever fixed vector. From this relation it easily follows[2)]

(**) *there exists $\bar\rho \geq \rho^*$ such that $\phi_\rho(y) > \phi_\rho(0)$ whenever $\rho \geq \bar\rho$ and y is a boundary point of the ball W.*

Now let $\rho \geq \bar\rho$. The function ϕ_ρ, being continuous on the closure of W (which is a closed ball, i.e., a compact set), attains its minimum on $\mathrm{cl}\, W$, and, due to strong convexity of the function, the minimizer, let it be called $y^*(\rho)$, is unique. By (**), this minimizer cannot be a boundary point of $\mathrm{cl}\, W$ – on the boundary ϕ_ρ is greater than at $y^* = 0$, i.e., it is a point from W. Since ϕ_ρ is smooth and $y^*(\rho) \in W$, $y^*(\rho)$ is a critical point of ϕ_ρ. There is no other critical point of ϕ_ρ in W, since ϕ_ρ is convex and therefore a critical point of the function is its minimizer on $\mathrm{cl}\, W$, and we know that such a minimizer is unique. Note that $y^*(\rho)$ is a nondegenerate local minimizer of ϕ_ρ, since ϕ_ρ is with positive definite Hessian. Last, $y^*(\rho) \to y^* = 0$ as $\rho \to \infty$. Indeed, we have

$$\phi_\rho(y^*(\rho)) \leq \phi_\rho(0),$$

whence, denoting $y_L(\rho) = Py^*(\rho)$, $y_{L^\perp}(\rho) = (I-P)y^*(\rho)$ and applying (10.2.4),

$$g^T y_{L^\perp}(\rho) + \frac{\alpha}{4}|y_L(\rho)|^2 + \frac{\rho}{4}|y_{L^\perp}(\rho)|^2 \leq 0,$$

or, with the Cauchy inequality,

$$|g||y_{L^\perp}(\rho)| \geq \frac{\alpha}{4}|y_L(\rho)|^2 + \frac{\rho}{4}|y_{L^\perp}(\rho)|^2.$$

From this inequality it immediately follows that $|y_{L^\perp}(\rho)| \to 0$ as $\rho \to \infty$ (why?), and with this observation the same inequality results also in $|y_L(\rho)| \to 0$, $\rho \to \infty$, so that indeed $y^*(\rho) \to y^* = 0$, $\rho \to \infty$.

[2]here is the derivation: we should prove that

$$(y'')^T g + \frac{\alpha}{4}|y'|^2 + \frac{\rho}{4}|y''|^2 > 0$$

whenever y is on the boundary of W and ρ is large enough; recall that W is centered at the origin ball of certain raduis $r > 0$. Denoting $s = |y''|^2$ and taking into account that $|y'|^2 = r^2 - s$ and $(y'')^T g \geq -cs^{1/2}$ by Cauchy's inequality, we reduce the problem in question to the following one: prove that

$$(!) \quad \min_{0 \leq s \leq r^2} \theta_\rho(s),\; \theta_\rho(s) = \left[\frac{\alpha}{4}r^2 + \frac{\rho-\alpha}{4}s - c\sqrt{s}\right]$$

is positive, provided that ρ is large enough. This is evident (split the entire segment $[0, r^2]$ where s varies into the segment Δ where $cs^{1/2} \leq \frac{\alpha}{8}r^2$ – in this segment $\theta_\rho(s)$ is positive whenever $\rho > \alpha$ – and the complementary segment Δ', and note that in this complementary segment s is bounded away from zero and therefore θ_ρ for sure is positive for all large enough values of ρ.

5^0. Thus, we have established the statement we are going to prove – but for the "locally equivalent to (ECP)" problem (ECP′) rather than for the actual problem of interest: we have pointed out a neighbourhood W of the point y^* such that the penalized objective ϕ_ρ of (ECP′), for all large enough ρ, has in this neighbourhood exactly one critical point, which is a nondegenerate minimizer of ϕ_ρ in the neighbourhood; and as $\rho \to \infty$, this critical point $y^*(\rho)$ converges to $y^* = 0$. Now let us take the image V of W under our substitution of variables mapping $y \mapsto X(y)$. We will get a neighbourhood of x^*, and in this neighbourhood we clearly have

$$f_\rho(x) = \phi_\rho(Y(x)).$$

Now, Y is one-to-one differentiable mapping of V onto W with differentiable inverse X; it immediately follows that a point x is a critical point (or a minimizer) of f_ρ in V if and only if $Y(x)$ is a critical point, respectively, a minimizer of ϕ_ρ in W; in particular, for $\rho \geq \bar\rho$ f_ρ indeed possesses unique critical point $x^*(\rho) = X(y^*(\rho))$ in V, and this is the minimizer of f_ρ in V. As $\rho \to \infty$, we have $y^*(\rho) \to y^* = 0$, whence $x^*(\rho) \to X(0) = x^*$. The only property of $x^*(\rho)$ we did not verify so far is that it is *nondegenerate* local minimizer of f_ρ, i.e., that $\nabla_x^2 f_\rho(x^*(\rho))$ is positive definite; we know that it is the case for ϕ_ρ and $y^*(\rho)$, but our substitution of variables is *nonlinear* and therefore, generally speaking, does not preserve positive definiteness of Hessians. Fortunately, it does preserve positive definiteness of the Hessians *taken at critical points*: if $Y(x)$ is twice continuously differentiable mapping with differentiable inverse and ψ is such that $\nabla_y \psi(\bar y) = 0$, $\bar y = Y(\bar x)$, then, as it is immediately seen, the Hessian of the composite function $g(x) = \psi(Y(x))$ at the point $\bar x$ is given by

$$\nabla_x^2 g(\bar x) = [Y'(\bar x)]^T \nabla_y^2 \psi(\bar y) Y'(\bar x)$$

(in the general case, there are also terms coming from the first order derivative of ψ at $\bar y$ and second order derivatives of $Y(\cdot)$, but in our case, when $\bar y$ is a critical point of ψ, these terms are zero), so that $\nabla_x^2 g(\bar x)$ is positive definite if and only if $\nabla_y^2 \psi(\bar y)$ is so. Applying this observation to $\psi = \phi_\rho$ and $\bar y = y^*(\rho)$, we obtain the last fact we need – nondegeneracy of $x^*(\rho)$ as an unconstrained local minimizer of f_ρ. ∎

10.2.2 Interesting properties of the path $x^*(\rho)$

In this subsection we assume that we are in situation of Theorem 10.2.2; our goal is to establish several useful properties of the given by the Theorem *path of unconstrained minimizers* $x^*(\rho), \rho \geq \bar\rho$ of the penalized objective in the neighbourhood V of x^*. The properties are as follows

- Let
$$X^+(\rho) = \{x \in V \mid |h(x)| \leq |h(x^*(\rho))|\}.$$
Then
$$x^*(\rho) \in \operatorname*{Argmin}_{x \in X^+(\rho)} f(x) \tag{10.2.5}$$
("$x^*(\rho)$ minimizes f on the set of all those points from V where the constraints are violated at most as they are violated at $x^*(\rho)$").

Indeed, $x^*(\rho)$ evidently belongs to $X^+(\rho)$; if there were a point x in $X^+(\rho)$ with $f(x) < f(x^*(\rho))$, we would have

$$f_\rho(x) = f(x) + \frac{\rho}{2}|h(x)|^2 <$$

[since $f(x) < f(x^*(\rho))$ and $|h(x)| \le |h(x^*(\rho))|$ due to $x \in X^+(\rho)$]

$$< f(x^*(\rho)) + \frac{\rho}{2}|h(x^*(\rho))|^2 = f_\rho(x^*(\rho)).$$

The resulting inequality is impossible, since $x^*(\rho)$ minimizes f_ρ in V.

- [Monotonicity of the optimal value of the penalized objective] The optimal (in V) values $f_\rho(x^*(\rho))$ of the penalized objectives do not decrease with ρ.

Indeed, if $\rho \le \rho'$, then clearly $f_\rho(x) \le f_{\rho'}(x)$ everywhere in V, and consequently the same inequality holds for the minimal values of the functions.

- [Monotonicity of violations of the constraints] The quantities

  $$v(\rho) = |h(x^*(\rho))|$$

  do not increase with ρ ("the larger is the penalty parameter, the less are violations of the constraints at the solution to the penalized problem")

Indeed, assume that $\rho' > \rho''$, and let $x' = x^*(\rho')$, $x'' = x^*(\rho'')$. We have

$$[f_{\rho'}(x'') \equiv] \quad f(x'') + \frac{\rho'}{2}|h(x'')|^2 \ge f(x') + \frac{\rho'}{2}|h(x')|^2 \quad [\equiv f_{\rho'}(x')]$$

and, similarly,

$$f(x') + \frac{\rho''}{2}|h(x')|^2 \ge f(x'') + \frac{\rho''}{2}|h(x'')|^2.$$

Taking sum of these inequalities, we get

$$\frac{\rho' - \rho''}{2}|h(x'')|^2 \ge \frac{\rho' - \rho''}{2}|h(x')|^2$$

whence, due to $\rho' > \rho''$, $v(\rho'') = |h(x'')| \ge |h(x')| = v(\rho')$, as required.

- [Monotonicity of the actual objective] The values of the actual objective f along the path $x^*(\rho)$ do not decrease with ρ.

Indeed, we already know that

$$f(x^*(\rho)) = \min_{x: x \in V, |h(x)| \le v(\rho)} f(x).$$

According to the previous statement, the sets in the right hand side over which the minimum is taken do not increase with ρ, and, consequently, the minimal value of f over these sets does not decrease with ρ.

The indicated properties demonstrate the following nice behaviour of the path $x^*(\rho)$: as the penalty parameter grows, the path approaches the constrained minimizer x^* of (ECP); the values of the objective along the path are always better (more exactly, not worse) than at x^* and increase (actually, do not decrease) with ρ, approaching the optimal value of the constrained problem from the left. Similarly, the violation of constraints becomes smaller and smaller as ρ grows and approaches zero. In other words, for every finite value of the penalty parameter it turns out to be profitable to violate constraints, getting, as a result, certain progress in the actual objective; and as the penalty parameter grows, this violation, same as progress in the objective, monotonically goes to zero.

An additional important property of the path is as follows:

- [Lagrange multipliers and the path] The quantities

$$\lambda_i(\rho) = \rho h_i(x^*(\rho))$$

tend, as $\rho \to \infty$, to the optimal Lagrange multipliers λ^* associated with x^*.

Indeed, $x^*(\rho)$ is a critical point of f_ρ, whence

$$\nabla_x f_\rho(x^*(\rho)) = \nabla_x f(x^*(\rho)) + \sum_{i=1}^m \lambda_i(\rho)\nabla_x h(x^*(\rho)) = 0.$$

Thus, $\lambda_i(\rho)$ are the coefficients in the representation of the vector $\psi(\rho) \equiv -\nabla_x f(x^*(\rho))$ as a linear combination of the vectors $\psi_i(\rho) = \nabla_x h_i(x^*(\rho))$. As we know, $x^*(\rho) \to x^*$ as $\rho \to \infty$, so that $\psi(\rho) \to \psi \equiv \nabla_x f(x^*)$ and $\psi_i(\rho) \to \psi_i \equiv \nabla_x h_i(x^*)$. Since the vectors $\psi_1, ..., \psi_m$ are linearly independent, from the indicated convergencies it follows (why?) that $\lambda_i(\rho) \to \lambda_i^*$, $\rho \to \infty$, where λ_i^* are the (uniquely defined) coefficients in the representation of $-\psi$ as a linear combination of ψ_i, i.e., are the Lagrange multipliers.

Remark 10.2.1 Similar results and properties take place also for the penalty method as applied to the general type constrained problem (GCP).

Remark 10.2.2 The quadratic penalty term we used is not, of course, the only option; for (ECP), we could use penalty term of the form

$$\rho\Phi(h(x))$$

as well, with smooth function Φ which is zero at the origin and positive outside the origin (in our considerations, Φ was set to $\frac{1}{2}|u|^2$); similar generalizations are possible for (GCP). The results for these more general penalties, under reasonable assumptions on Φ, would be similar to those as for the particular case we have considered.

10.2.3 Penalty method: advantages and drawbacks

Now it is time to look what are our abilities to solve the unconstrained problems

$$(\mathrm{P}_\rho) \quad f_\rho(x) \to \min$$

which, as we already know, for large ρ are good approximations of the constrained problem in question. *In principle* we can solve these problems by any one of unconstrained minimization methods we know, and this is definitely a great advantage of the approach.

There is, anyhow, a severe weak point of the construction – to approximate well the constrained problem by unconstrained one, we *must* deal with large values of the penalty parameter, and this, as we shall see in a while, unavoidably makes the unconstrained problem (P_ρ) *ill-conditioned* and thus – very difficult for any unconstrained minimization methods sensitive to the conditioning of the problem. And all the methods for unconstrained minimization we know, except, possibly, the Newton method, are "sensitive" to conditioning (e.g., in the Gradient Descent the number of steps required to achieve an ϵ-solution is, asymptotically, inverse proportional to the condition number of the Hessian of objective at the optimal point). Even the Newton method, which does not react on the conditioning explicitly – it is "self-scaled" – suffers a lot as applied to an ill-conditioned problem, since here we are enforced to invert ill-conditioned

Hessian matrices, and this, in actual computations with their rounding errors, causes a lot of troubles. The indicated drawback – ill-conditioness of auxiliary unconstrained problems – is the main disadvantage of the "straightforward" penalty scheme, and because of it the scheme is not that widely used now and is in many cases replaced with more smart modified Lagrangian scheme (in the mean time, we shall look at the latter scheme).

It is time now to justify the above claim that problem (P_ρ) is, for large ρ, ill-conditioned. Indeed, assume that we are in the situation of Theorem 10.2.2, and let us compute the Hessian H_ρ of f_ρ at the point $x = x^*(\rho)$. The computation yields

$$H_\rho = \left[\nabla_x^2 f(x) + \sum_{i=1}^m [\rho h_i(x)]\nabla_x^2 h_i(x)\right] + \rho[\nabla_x h(x)]^T[\nabla_x h(x)].$$

We see that H_ρ is comprised of two terms: the matrix in the brackets, let it be called L_ρ, and the proportional to ρ matrix $\rho M_\rho \equiv [\nabla_x h(x)]^T[\nabla_x h(x)]$. When $\rho \to \infty$, then, as we know, $x = x^*(\rho)$ converges to x^* and $\rho h_i(x^*)$ converge to the Lagrange multipliers λ_i^* of (ECP), so that L_ρ possesses quite respectable limit L, namely, the Hessian of the Lagrange function $\nabla_x^2 L(x^*, \lambda^*)$. The matrix M_ρ also possesses limit, namely,

$$M = [\nabla_x h(x^*)]^T[\nabla_x h(x^*)];$$

this limit, as it is clearly seen, is a matrix which vanishes on the tangent at x^* plane T to the feasible surface of the problem and is nondegenerate on the orthogonal complement $T^\perp$ to this tangent plane. Since M_ρ is symmetric, both T and $T^\perp$ are invariant for M, and M possesses $n - m$ eigenvectors with zero eigenvalues – these vectors span T – and m eigenvectors with positive eigenvalues – these latter vectors span $T^\perp$. Since

$$H_\rho = L_\rho + \rho M_\rho,$$

we conclude that the spectrum of H_ρ, for large ρ, is as follows: there are $n - m$ eigenvectors "almost in T" with the eigenvalues "almost equal" to those of the reduction of L onto T; since L is positive definite on T, these eigenvalues are positive reals. Now, H_ρ possesses m eigenvectors "almost orthogonal" to T with eigenvalues "almost equal" to ρ times the nonzero eigenvalues of M. Thus, excluding trivial cases $m = 0$ (no constraints at all) and $m = n$ (locally unique feasible solution x^*), the eigenvalues of H_ρ form two groups – group of $n - m$ asymptotically constant positive reals and group of m reals asymptotically proportional to ρ. We conclude that the condition number of H_ρ is of order of ρ, as it was claimed.

10.3 Barrier methods

10.3.1 "Classical" barrier scheme

The situation with the *barrier* (interior penalty) methods in their "classical" from outlined in Section 10.1.3 is very similar to the one with penalty methods. It indeed is true (and is easily verified) that the solutions to the modified problem

$$(P_\rho) \quad F_\rho(x) \equiv f(x) + \frac{1}{\rho}F(x) \to \min,$$

F being the interior penalty for the feasible domain of (ICP), converge to the optimal set of the problem:

Theorem 10.3.1 *Let F be an interior penalty function for the feasible domain G of (ICP), and assume that the feasible domain is bounded and is the closure of its interior* int G. *Then the set $X^*(\rho)$ of minimizers of F_ρ on* int G *is nonempty, and these sets converge, as $\rho \to \infty$, to the optimal set X^* of (ICP): for every $\epsilon > 0$ there exists $\bar\rho$ such that $X^*(\rho)$, for all $\rho \geq \bar\rho$, is contained in the ϵ-neighbourhood*

$$X^*_\epsilon = \{x \in G \mid \exists x^* \in X^* : |x - x^*| < \epsilon\}$$

of the optimal set of (ICP).

Proof is completely similar to that one of Theorem 10.2.1. First of all, X^* is nonempty (since f is continuous and the feasible domain is bounded and closed and is therefore a compact set). The fact that $X^*(\rho)$ are nonempty for all positive ρ was proved in Section 10.1.3. To prove that $X^*(\rho)$ is contained, for large ρ, in a tight neighbourhood of X^*, let us act as follows. Same as in the proof of Theorem 10.2.1, it suffices to lead to a contradiction the assumption that there exists a sequence $\{x_i \in X^*(\rho_i)\}$ with $\rho_i \to \infty$ which converges to a point $x \in G \backslash X^*$. Assume that it is the case; then $f(x) > \min_G f + \delta$ with certain positive δ. Let x^* be a global minimizer of f on G. Since G is the closure of int G, x^* can be approximated, within an arbitrarily high accuracy, by points from int G, and since f is continuous, we can find a point $x' \in$ int G such that

$$f(x') \leq f(x^*) + \frac{\delta}{2}.$$

We have

$$F_{\rho_i}(x_i) = \min_{x \in \text{int } G} F_{\rho_i}(x) \leq F_{\rho_i}(x'), \tag{10.3.1}$$

whence

$$f(x_i) + \frac{1}{\rho_i} F(x_i) \leq f(x') + \frac{1}{\rho_i} F(x').$$

Since F is a barrier for *bounded* domain G, F is below bounded on int G (since it attains its minimum on int G – use the reasoning from Section 10.1.3 for the case of $f \equiv 0$). Thus, $F(x) \geq a > -\infty$ for all $x \in$ int G, and therefore (10.3.1) implies that

$$f(x_i) \leq f(x') + \frac{1}{\rho_i} F(x') - \frac{1}{\rho_i} a.$$

As $i \to \infty$, the right hand side in this inequality tends to $f(x') \leq f(x^*) + \frac{\delta}{2}$, and the left hand side tends to $f(x) \geq f(x^*) + \delta$; since $\delta > 0$, we get the desired contradiction. ∎

If I were writing this lecture 5-8 years ago, I would proceed with the statements similar to the one of Theorem 10.2.2 and those on behaviour of the path of minimizers of F_ρ and conclude with the same laments "all this is fine, but the problems of minimization of F_ρ normally (when the solution to the original problem is on the boundary of G; otherwise the problem actually is unconstrained) become the more ill-conditioned the larger is ρ, so that the difficulties of their numerical solution grow with the penalty parameter". When indeed writing this lecture, I would say something quite opposite: *there exists important situations when the difficulties in numerical minimization of F_ρ do not increase with the penalty parameter, and the overall scheme turns out to be theoretically efficient and, moreover, the best known so far.* This change in evaluation of the scheme is the result of recent "interior point revolution" in Optimization which I have already mentioned in Lecture 4. Let us look what is the revolution and what is the progress caused by the revolution.

10.3.2 Self-concordant barriers and path-following scheme

Assume from now on that our problem (ICP) is convex (the revolution we are speaking about deals, at least directly, with Convex Programming only). It is well-known that convex program (ICP) can be easily rewritten as a program with *linear objective*; indeed, it suffices to extend the vector of design variables by one variable, let it be called t, more and to rewrite (ICP) as the problem

$$t \to \min \mid g_j(x) \le 0,\ j = 1, ..., k,\ f(x) - t \le 0.$$

The resulting problem still is convex and has linear objective.

To save notation, assume that the outlined conversion is carried out in advance, so that already the original problem has linear objective and is therefore of the form

$$\text{(P)} \quad f(x) \equiv c^T x \to \min \mid x \in G \subset \mathbf{R}^n.$$

Here the feasible set G of the problem is convex (we are speaking about convex programs!); we also assume that it is *closed, bounded and possesses a nonempty interior.*

Our abilities to solve (P) efficiently by an interior point method depend on our abilities to point out a "good" interior penalty F for the feasible domain. What we are interested in is a ϑ-self-concordant barrier F; the meaning of these words is given by the following

Definition 10.3.1 [Self-concordant barrier] *Let $\vartheta \ge 1$. We say that a function F is ϑ-self-concordant barrier for the feasible domain D of problem (P), if*

- *F is self-concordant function on* int G *(Section 4.3.2, Lecture 4), i.e., three times continuously differentiable convex function on* int G *possessing the barrier property (i.e., $F(x_i) \to \infty$ along every sequence of points $x_i \in$ int G converging to a boundary point of G) and satisfying the differential inequality*

$$|\frac{d^3}{dt^3}|_{t=0} F(x + th)| \le 2\left[h^T F''(x) h\right]^{3/2} \quad \forall x \in \text{int } G\ \forall h \in \mathbf{R}^n;$$

- *F satisfies the differential inequality*

$$|h^T F'(x)| \le \sqrt{\vartheta}\sqrt{h^T F''(x) h} \quad \forall x \in \text{int } G\ \forall h \in \mathbf{R}^n. \tag{10.3.2}$$

An immediate example is as follows (cf. "Raw materials" in Section 4.3.4, Lecture 4):

Example 10.3.1 [Logarithmic barrier for a polytope] *Let*

$$G = \{x \in \mathbf{R}^n \mid a_j^T x \le b_j,\ j = 1, ..., m\}$$

be a polytope given by a list of linear inequalities satisfying the Slater condition (i.e., there exists $\bar{x}$ such that $a_j^T \bar{x} < b_j$, $j = 1, ..., m$). Then the function

$$F(x) = -\sum_{j=1}^{m} \ln(b_j - a_j^T x)$$

is an m-self-concordant barrier for G.

In the mean time, we shall justify this example (same as shall consider the crucial issue of *how* to find a self-concordant barrier for a given feasible domain). For the time being, let us focus on another issue: how to solve (P), *given a ϑ-self-concordant barrier for the feasible domain of the problem.*

What we intend to do is to use the *path-following scheme* associated with the barrier – certain very natural implementation of the barrier method.

Path-following scheme

When speaking about the barrier scheme, we simply claimed that the minimizers of the aggregate F_ρ approach, as $\rho \to \infty$, the optimal set of (P). The immediate recommendation coming from this claim could be: choose "large enough" value of the penalty and solve, for the chosen ρ, the problem (P_ρ) of minimizing F_ρ, thus coming to a good approximate solution to (P). It makes, anyhow, sense to come to the solution of (P_ρ) *gradually*, by solving sequentially problems (P_{ρ_i}) along an increasing sequence of values of the penalty parameter. Namely, assume that the barrier F we use is nondegenerate, i.e., $F''(x)$ is positive definite at every point $x \in \text{int } G$ (note that this indeed is the case when F is self-concordant barrier for a bounded feasible domain, see Proposition 4.3.1.(i)). Then the optimal set of F_ρ for every positive ρ is a singleton (we already know that it is nonempty, and the uniqueness of the minimizer follows from convexity and nondegeneracy of F_ρ). Thus, we have a *path*

$$x^*(\rho) = \operatorname*{argmin}_{\text{int } G} F_\rho(\cdot);$$

as we know from Theorem 10.3.1, this path converges to the optimal set of (P) as $\rho \to \infty$; besides this, it can be easily seen that the path is continuous (even continuously differentiable) in ρ. In order to approximate $x^*(\rho)$ with large values of ρ via the path-following scheme, we *trace the path* $x^*(\rho)$, namely, generate sequentially approximations $x(\rho_i)$ to the points $x^*(\rho_t)$ along certain diverging to infinity sequence $\rho_0 < \rho_1 < ...$ of values of the parameter. This is done as follows:

given "tight" approximation $x(\rho_t)$ to $x^*(\rho_t)$, we update it into "tight" approximation $x(\rho_{t+1})$ to $x^*(\rho_{t+1})$ as follows:

- *first, choose somehow a new value $\rho_{t+1} > \rho_t$ of the penalty parameter*
- *second, apply to the function $F_{\rho_{t+1}}(\cdot)$ a method for unconstrained minimization started at $x(\rho_t)$, and run the method until closeness to the new target point $x^*(\rho_{t+1})$ is restored, thus coming to the new iterate $x(\rho_{t+1})$ close to the new target point of the path.*

All this is very close to what we did when tracing feasible surface with the Gradient Projection scheme; our hope is that since $x^*(\rho)$ is continuous in ρ and $x(\rho_t)$ is "close" to $x^*(\rho_t)$, for "not too large" $\rho_{t+1} - \rho_t$ the point $x(\rho_t)$ will be "not too far" from the new target point $x^*(\rho_{t+1})$, so that the unconstrained minimization method we use will quickly restore closeness to the new target point. With this "gradual" movement, we may hope to arrive near $x^*(\rho)$ with large ρ faster than by attacking the problem (P_ρ) directly.

All this was known for many years; and the progress during last decade was in transforming these qualitative ideas into exact quantitive recommendations.

Namely, it turned out that

- **A.** The best possibilities to carry this scheme out are when the barrier F is ϑ-self-concordant; the less is the value of ϑ, the better;
- **B.** The natural measure of "closeness" of a point $x \in \text{int } G$ to the point $x^*(\rho)$ of the path is the Newton decrement of the self-concordant function

$$\Phi_\rho(x) = \rho F_\rho(x) \equiv \rho c^T x + F(x)$$

at the point x, i.e., the quantity

$$\lambda(\Phi_\rho, x) = \sqrt{[\nabla_x \Phi_\rho(x)]^T [\nabla_x^2 \Phi_\rho(x)]^{-1} \nabla_x \Phi_\rho(x)}$$

(cf. Proposition 4.3.1.(iii)). More specifically, the notion "x is close to $x^*(\rho)$" is convenient to define as the relation

$$\lambda(\Phi_\rho, x) \leq 0.05 \tag{10.3.3}$$

(in fact, 0.05 in the right hand side could be replaced with arbitrary absolute constant < 1, with slight modification of subsequent statements; I choose this particular value for the sake of simplicity)

Now, what do all these words "the best possibility" and "natural measure" actually mean? It is said by the following two statements.

- **C.** *Assume that x is close, in the sense of (10.3.3), to a point $x^*(\rho)$ of the path $x^*(\cdot)$ associated with a ϑ-self-concordant barrier for the feasible domain G of problem (P). Let us increase the parameter ρ to the larger value*

$$\rho^+ = \left(1 + \frac{0.08}{\sqrt{\vartheta}}\right) \rho \tag{10.3.4}$$

and replace x by its damped Newton iterate (cf. (4.3.4), Lecture 4)

$$x^+ = x - \frac{1}{1 + \lambda(\Phi_{\rho^+}, x)} [\nabla_x^2 \Phi_{\rho^+}(x)]^{-1} \nabla_x \Phi_{\rho^+}(x). \tag{10.3.5}$$

Then x^+ is close, in the sense of (10.3.3), to the new target point $x^*(\rho^+)$ of the path.

C. says that we are able to trace the path (all the time staying close to it in the sense of **B.**) increasing the penalty parameter linearly in the ratio $(1 + 0.08\vartheta^{-1/2})$ and accompanying each step in the penalty parameter by a single Newton step in x. And why we should be happy with this, it is said by

- **D.** *If x is close, in the sense of (10.3.3), to a point $x^*(\rho)$ of the path, then the inaccuracy, in terms of the objective, of the point x as of an approximate solution to (P) is bounded from above by $2\vartheta\rho^{-1}$:*

$$f(x) - \min_{x \in G} f(x) \leq \frac{2\vartheta}{\rho}. \tag{10.3.6}$$

D. says that the inaccuracy of the iterates $x(\rho_i)$ formed in the above path-following procedure goes to 0 as $1/\rho_i$, while **C.** says that we are able increase ρ_i linearly, at the cost of a single Newton step per each updating of ρ. Thus, we come to the following

Theorem 10.3.2 *Assume that we are given*

- (i) *ϑ-self-concordant barrier F for the feasible domain G of problem (P)*
- (ii) *starting pair (x_0, ρ_0) with $\rho_0 > 0$ and x_0 being close, in the sense of (10.3.3), to the point $x^*(\rho_0)$.*

Consider the path-following method (cf. (10.3.4) - (10.3.5))

$$\rho_{t+1} = \left(1 + \frac{0.08}{\sqrt{\vartheta}}\right)\rho_t; \quad x_{t+1} = x_t - \frac{1}{1+\lambda(\Phi_{\rho_{t+1}}, x_t)}[\nabla_x^2 \Phi_{\rho_{t+1}}(x_t)]^{-1}\nabla_x \Phi_{\rho_{t+1}}(x_t). \qquad (10.3.7)$$

Then the iterates of the method are well-defined, belong to the interior of G and the method possesses linear global rate of convergence:

$$f(x_t) - \min_G f \leq \frac{2\vartheta}{\rho_0}\left(1 + \frac{0.08}{\sqrt{\vartheta}}\right)^{-t}. \qquad (10.3.8)$$

In particular, to make the residual in f less than a given $\epsilon > 0$, it suffices to perform no more that

$$N(\epsilon) \leq \lfloor 20\sqrt{\vartheta}\ln\left(1 + \frac{20\vartheta}{\rho_0\epsilon}\right)\rfloor \qquad (10.3.9)$$

Newton steps.

We see that the parameter ϑ of the self-concordant barrier underlying the method is responsible for the *Newton complexity* of the method – the factor at the log-term in the complexity bound (10.3.9).

Remark 10.3.1 The presented result does not explain how to *start* tracing the path – how to get initial pair (x_0, ρ_0) close to the path. This turns out to be a minor difficulty: given in advance a strictly feasible solution $\bar{x}$ to (P), we could use the same path-following scheme (applied to certain artificial objective) to come close to the path $x^*(\cdot)$, thus arriving at a position from which we can start tracing the path. In our very brief outline of the topic, it makes no sense to go in these " details of initialization"; it suffices to say that the necessity to start from approaching $x^*(\cdot)$ basically does not violate the overall complexity of the method.

It makes sense if not to *prove* the aforementioned statements – the complete proofs, although rather simple, go beyond the scope of our today lecture – but at least to *motivate* them – to explain what is the role of self-concordance and "magic inequality" (10.3.2) in ensuring properties **C.** and **D.** (this is all we need – the Theorem, of course, is an immediate consequence of these two properties).

Let us start with **C.** – this property is much more important. Thus, assume we are at a point x close, in the sense of (10.3.3), to $x^*(\rho)$. What this inequality actually says?

Let us denote by

$$\| h \|_{H^{-1}} = (h^T H^{-1} h)^{1/2}$$

the *scaled* Euclidean norm given by the inverse to the Hessian matrix

$$H \equiv \nabla_x^2 \Phi_\rho(x) = \nabla_x^2 F(x)$$

(the equality comes from the fact that Φ_ρ and F differ by a *linear* function $\rho f(x) \equiv \rho c^T x$). Note that by definition of $\lambda(\cdot,\cdot)$ one has

$$\lambda(\Phi_s, x) = \| \nabla_x \Phi_s(x) \|_{H^{-1}} \equiv \| sc + F'(x) \|_{H^{-1}}.$$

Due to the last formula, the closeness of x to $x^*(\rho)$ (see (10.3.3)) means exactly that

$$\| \nabla_x \Phi_\rho(x) \|_{H^{-1}} \equiv \| \rho c + F'(x) \|_{H^{-1}} \leq 0.05,$$

whence, by the triangle inequality,

$$\| \rho c \|_{H^{-1}} \leq 0.05 + \| F'(x) \|_{H^{-1}} \leq 0.05 + \sqrt{\vartheta} \tag{10.3.10}$$

(the concluding inequality here is given by (10.3.2) [3)], and this is the main point where this component of the definition of a self-concordant barrier comes into the play).

From the indicated relations

$$\lambda(\Phi_{\rho^+}, x) = \| \rho^+ c + F'(x) \|_{H^{-1}} \leq \| (\rho^+ - \rho)c \|_{H^{-1}} + \| \rho c + F'(x) \|_{H^{-1}} =$$

$$= \frac{|\rho^+ - \rho}{\rho} \| \rho c \|_{H^{-1}} + \lambda(\Phi_\rho, x) \leq$$

[see (10.3.4), (10.3.10)]

$$\leq \frac{0.08}{\sqrt{\vartheta}}(0.05 + \sqrt{\vartheta}) + 0.05 \leq 0.134$$

(note that $\vartheta \geq 1$ by Definition 10.3.1). According to Proposition 4.3.1.(iii.3), Lecture 4, the indicated inequality says that *we are in the domain of quadratic convergence of the damped Newton method as applied to self-concordant function* Φ_{ρ^+}; namely, the indicated Proposition says that

$$\lambda(\Phi_{\rho^+}, x^+) \leq \frac{2(0.134)^2}{1 - 0.134} < 0.05.$$

as claimed in **C.**. Note that this reasoning heavily exploits self-concordance of F ∎

To establish property **D.**, it requires to analyze in more details the notion of a self-concordant barrier, and I am not going to do it here. Just to demonstrate where ϑ comes from, let us prove an estimate similar to (10.3.6) for the particular case when, first, the barrier in question is the standard logarithmic barrier given by Example 10.3.1 and, second, the point x is exactly the point $x^*(\rho)$ rather than is close to the latter point. Under the outlined assumptions we have

$$x = x^*(\rho) \Rightarrow \nabla_x \Phi_\rho(x) = 0 \Rightarrow$$

[substitute expressions for Φ_ρ and F]

$$\rho c + \sum_{j=1}^{m} \frac{a_j}{b_j - a_j^T x} = 0 \Rightarrow$$

[take inner product with $x - x^*$, x^* being an optimal solution to (P)]

$$\rho c^T (x - x^*) = \sum_{j=1}^{m} \frac{a_j^T (x^* - x)}{b_j - a_j^T x} \leq$$

[take into account that $a_j^T (x^* - x) = a_j^T x^* - a_j^T x \leq b_j - a_j^T x$ due to $x^* \in G$]

$$\leq m,$$

whence

$$c^T (x - x^*) \leq \frac{m}{\rho} \equiv \frac{\vartheta}{\rho}$$

(for the case in question $\vartheta = m$). This estimate is twice better than (10.3.6) – this is because we have considered the case of $x = x^*(\rho)$ rather than the one of x close to $x^*(\rho)$.

[3] indeed, for a positive definite symmetric matrix H it clearly is the same (why?) to say that $|g|_{H^{-1}} \leq \alpha$ and to say that $|g^T h| \leq \alpha |h|_H$ for all h

Applications

Linear Programming. The most famous (although, I believe, not the most important) application of Theorem 10.3.2 deals with Linear Programming, when G is a polytope and F is the standard logarithmic barrier for this polytope (see Example 10.3.1). For this case, the Newton complexity of the method[4)] is $O(\sqrt{m})$, m being the # of linear inequalities involved into the description of G. Each Newton step costs, as it is easily seen, $O(mn^2)$ arithmetic operations, so that the *arithmetic cost per accuracy digit* – number of arithmetic operations required to reduce current inaccuracy by absolute constant factor – turns out to be $O(m^{1.5}n^2)$. Thus, we get a polynomial time solution method for LP, with complexity characteristics typically (for m and n of the same order) better than those for the Ellipsoid method (Lecture 7). Note also that with certain "smart" implementation of Linear Algebra, the above arithmetic cost can be reduced to $O(mn^2)$; this is the best known so far *cubic in the size of the problem* upper complexity bound for Linear Programming.

To increase list of application examples, note that our abilities to solve in the outlined style a convex program of a given structure are limited only by our abilities to point out self-concordant barrier for the corresponding feasible domain. *In principle*, there are no limits at all – it can be proved that *every* closed convex domain in $\mathbf{R}^n$ admits a self-concordant barrier with the value of parameter at most $O(n)$. This "universal barrier" is given by certain multivariate integral and is too complicated for actual computations; recall that we should form and solve Newton systems associated with our barrier, so that we need it to be "explicitly computable".

Thus, we come to the following important question:

How to construct "explicit" self-concordant barriers. There are many cases when we are clever enough to point out "explicitly computable self-concordant barriers" for convex domains we are interested in. We already know one example of this type – Linear Programming (although we do not know to the moment why the standard logarithmic barrier for a polytope given by m linear constraints is m-self-concordant; why it is so, it will become clear in a moment). What helps us to construct self-concordant barriers and to evaluate their parameters are the following extremely simple combination rules, completely similar to those for self-concordant functions (see Section 4.3.4, Lecture 4):

- [Linear combination with coefficients ≥ 1] Let F_i, $i = 1, ...m$, be ϑ_i-self-concordant barriers for the closed convex domains G_i, let the intersection G of these domains possess a nonempty interior Q, and let $\alpha_i \geq 1$, $i = 1, ..., m$, be given reals. Then the function

$$F(x) = \sum_{i=1}^{m} \alpha_i F_i(x)$$

  is $(\sum_{i=1}^m \alpha_i \vartheta_i)$-self-concordant barrier for G.

- [Affine substitution] Let $F(x)$ be ϑ-self-concordant barrier for the closed convex domain $G \subset \mathbf{R}^n$, and let $x = A\xi + b$ be an affine mapping from $\mathbf{R}^k$ into $\mathbf{R}^n$ with the image intersecting int G. Then the composite function

$$F^+(\xi) = F(A\xi + b)$$

[4] recall that it is the factor at the logarithmic term in (10.3.9), i.e., the # of Newton steps sufficient to reduce current inaccuracy by an absolute constant factor, say, by factor 2; cf. with the stories about polynomial time methods from Lecture 7

is ϑ-self-concordant barrier for the closed convex domain

$$G^+ = \{\xi \mid A\xi + b \in G\}$$

which is the inverse image of G under the affine mapping in question.

The indicated combination rules can be applied to the "raw materials" as follows:

- [Logarithm] The function
$$-\ln(x)$$
is 1-self-concordant barrier for the nonnegative ray $\mathbf{R}_+ = \{x \in \mathbf{R} \mid x > 0\}$;

[the indicated property of logarithm is given by 1-line computation]

- [Extension of the previous example: Logarithmic barrier, linear/quadratic case] Let
$$G = \mathrm{cl}\{x \in \mathbf{R}^n \mid \phi_j(x) < 0,\ j = 1, ..., m\}$$
be a nonempty set in $\mathbf{R}^n$ given by m convex quadratic (e.g., linear) inequalities satisfying the Slater condition. Then the function
$$f(x) = -\sum_{i=1}^{m} \ln(-\phi_i(x))$$
is m-self-concordant barrier for G.

[for the case when all the functions f_i are linear, the conclusion immediately follows from the Combination rules (a polyhedral set given by m linear inequalities is intersection of m half-spaces, and a half-space is inverse image of the nonnegative axis under affine mapping; applying the combination rules to the barrier $-\ln x$ for the nonnegative ray, we get, without any computations, that the standard logarithmic barrier is for a polyhedral set is m-self-concordant). For the case when there are also quadratic forms among f_i, you need a 1-2 lines of computations. Note that the case of *linear* f_i covers the entire Linear Programming, while the case of *convex quadratic* f_i covers much wider family of quadratically constrained convex quadratic problems.]

- The function
$$F(t, x) = -\ln(t^2 - x^T x)$$
is 2-self-concordant barrier for the "ice-cream cone"
$$\mathbf{K}^n_+ = \{(t, x) \in \mathbf{R} \times \mathbf{R}^n \mid t \geq |x|\};$$
the function
$$F(X) = -\ln \mathrm{Det}\ X$$
is n-self-concordant barrier for the cone $\mathbf{S}^n_+$ of positive definite symmetric $n \times n$ matrices.

One hardly could imagine how wide is the class of applications – from Combinatorial optimization to Structural Design and Stability Analysis/Synthesis in Control – of the latter two barriers, especially of the $\ln \mathrm{Det}$ -one.

10.3.3 Concluding remarks

The path-following scheme results in the most efficient, in terms of the worst-case complexity analysis, interior point methods for Linear and Convex Programming. From the practical viewpoint, anyhow, this scheme, in its aforementioned form, looks bad. The severe practical drawback of the scheme is its "short-step nature" – according to the scheme, the penalty parameter should be updated by the "programmed" rule (10.3.4), and this makes the actual performance of the method more or less close to the one given by (10.3.9). Thus, in the straightforward implementation of the scheme the complexity estimate (10.3.9) will be not just the theoretical upper bound on the worst-case complexity of the method, but the indication of the "typical" performance of the algorithm. And a method *actually* working according to the complexity estimate (10.3.9) could be fine theoretically, but it definitely will be of very restricted practical interest in the large-scale case. E.g., in LP program with $m \approx 10^5$ inequality constraints and $n \approx 10^4$ variables (these are respectable, but in no sense "outstanding" sizes for a practical LP program) estimate (10.3.9) predicts something like hundreds of Newton steps with Newton systems of the size $10^4 \times 10^4$; even in the case of good sparsity structure of the systems, such a computation would be much more time consuming than the one given by the Simplex Method.

In order to get "practical" path-following methods, we need a *long-step* tactics – rules for *on-line* adjusting the stepsizes in the penalty parameter to, let me say, local curvature of the path, rules which allow to update parameter as fast as possible – possible from the viewpoint of the actual "numerical circumstances" the method is in rather than from the viewpoint of very conservative theoretical worst-case complexity analysis.

Today there are efficient "long-step" policies of tracing the paths, policies which are both fine theoretically (i.e., satisfy complexity bound (10.3.9)) and very efficient computationally. Extremely surprising phenomenon here is that for "good" long-step path-following methods as applied to convex problems of the most important classes (Linear Programming, Quadratically Constrained Convex Quadratic Programming and some other) it turns out that

> *the actually observed number of Newton iterations required to solve the problem within reasonable accuracy is basically independent of the sizes of the problem and is within 30-50.*

This "empirical fact" (which can be only partly supported by theoretical considerations, not proved completely) is extremely important for applications; it makes polynomial time interior point methods the most attractive (and sometimes - the only appropriate) optimization tool in many important large-scale applications.

I should add that the efficient "long-step" implementations of the path-following scheme are relatively new, and for a long time[5)] the only interior point methods which demonstrated the outlined "data- and size-independent" convergence rate were the so called *potential reduction interior point methods*. In fact, the very first interior point method – the *method of Karmarkar* for LP – which initialized the entire interior point revolution, was a potential reduction algorithm, and what indeed caused the revolution was outstanding practical performance of this method. The method of Karmarkar possesses a very nice (and in fact very simple) geometry and is closely related to the interior penalty scheme; anyhow, time limitations enforce me to skip description of this wonderful, although now a little bit old-fashioned, algorithm.

The concluding remark I would like to do is as follows: all polynomial time implementations of the penalty/barrier scheme known so far are those of the barrier scheme (which is reflected

[5]if I could qualify as "long" a part of the story which – the entire story – started in 1984

in the name of these implementations: "interior point methods"); numerous attempts to do something similar with the penalty approach failed to be successful. It is a pity, due to some attractive properties of the scheme (e.g., here you do not meet with the problem of finding a feasible starting point, which, of course, is needed to start the barrier scheme).

Lecture 11

Augmented Lagrangians

Penalty methods studied in the previous lecture are very natural and general; unfortunately, they suffer from a serious drawback: to approximate well the solution to constrained problem, you have to work with large penalty parameters, and this inevitably makes the problem of unconstrained minimization of the penalized objective very ill-posed. The main advantage of the *augmented Lagrangian methods* is that they allow to approximate well the solution to a constrained problem by solutions of unconstrained (and penalized) auxiliary problems *without pushing the penalty parameter to infinity*; as a result, the auxiliary problems remain reasonably conditioned even when we are seeking for high-accuracy solutions. Let us look how all this works. We shall mainly focus on the case of equality constrained problem

$$\text{(ECP)} \quad f(x) \to \min \mid h(x) = \begin{pmatrix} h_1(x) \\ \dots \\ h_m(x) \end{pmatrix} = 0 \quad [x \in \mathbf{R}^n];$$

in the mean time I shall explain how the results for this case can be straightforwardly extended onto the general one.

11.1 Main ingredients

11.1.1 Local Lagrange Duality

Let x^* be a nondegenerate local solution to (ECP) (see Definition 8.2.2), let

$$L(x, \lambda) = f(x) + \lambda^T h(x) \equiv f(x) + \sum_{i=1}^{m} \lambda_i h_i(x)$$

be the Lagrange function of the problem, and let λ^* be the vector of Lagrange multipliers corresponding to the solution x^*, so that

$$\nabla_x L(x^*, \lambda^*) = 0, \tag{11.1.1}$$

and the matrix $\nabla_x^2 L(x^*, \lambda^*)$ is positive definite along the plane tangent at x^* to the feasible surface of the problem:

$$\nabla h(x^*) d = 0 \Rightarrow d^T \nabla_x^2 L(x^*, \lambda^*) d \geq \alpha |d|^2 \tag{11.1.2}$$

with certain positive $\alpha > 0$.

Assume for a moment that instead of (11.1.2) a stronger condition is satisfied:

(!) *the matrix* $\nabla_x^2 L(x^*, \lambda^*)$ *is positive definite on the entire space*

What would be the consequences of this stronger assumption?

The consequences are immediate: (11.1.1) together with (!) mean that x^* is a *nondegenerate local minimizer* of the Lagrange function $L(\cdot, \lambda^*)$. It follows that if

A. *We are clever enough to guess the vector* λ^* *of Lagrange multipliers,*

and

B. *We are lucky enough to be in the case (!),*

then we are be able to find x^* via unconstrained minimization applied to the function $L(\cdot, \lambda^*)$. Note that this is a fine smooth function which contains no penalty parameters and therefore is convenient for unconstrained minimization.

It remains to be clever and lucky to ensure **A** and **B**. Let us start with becoming clever, provided that *we already are lucky*, i.e., that *(!) indeed takes place.*

Those who have passed through the course Optimization I already know what is Lagrange duality: this is a machinery which allows to associate with constrained optimization problem (called *primal*) certain other optimization problem – the *dual* one; the solutions to the problems are closely related to each other. To enjoy all the advantages of this machinery, we should deal with *convex* primal problem, but something can be obtained also in the "locally convex case" (!).

Namely, one can prove the following

Theorem 11.1.1 *Under assumption (!) there exist*

- *convex neighbourhood* V *of the point* x^* *in* $\mathbf{R}^n$*,*
- *convex neighbourhood* Λ *of the point* λ^* *in* $\mathbf{R}^m$*,*

such that

(i) *The function*

$$L_\lambda(x) = L(x, \lambda)$$

for every $\lambda \in \Lambda$ *is strongly convex in* V *and possesses uniquely defined critical point* $x^*(\lambda)$ *in* V*, the point being the nondegenerate minimizer of* L_λ *in* V*; the mapping* $x^*(\lambda)$ *is at least once continuously differentiable in* Λ*.*

(ii) *The optimal value*

$$\underline{L}(\lambda) = L_\lambda(x^*(\lambda))$$

of the function $L_\lambda(x)$ *in* V *is* <u>concave</u> *twice continuously differentiable function in* Λ *with the gradient*

$$\nabla_\lambda \underline{L}(\lambda) = h(x^*(\lambda)). \tag{11.1.3}$$

The function $\underline{L}$ *attains its maximum over* $\lambda \in \Lambda$ *at the point* λ^**, the maximizer being unique and nondegenerate*[1)]*.*

[1]of course, for <u>maximization</u> problem nondegeneracy of local solution means that the Hessian of the objective at the solution is negative definite

The proof of the theorem is skipped.

Note that the Theorem says that under assumption (!) (x^*, λ^*) is *local saddle point* of the Lagrange function: there exists a neighobouthood of this point (namely, $V \times \Lambda$) such that L, being restricted onto the neighbourhood, attains at (x^*, λ^*) its minimum in x and its maximum in λ.

Let us look at the consequences of the Theorem; they are extremely encouraging. Indeed, we already know that if we were clever enough to guess what is λ^*, we could solve (ECP) by unconstrained minimization of $L_{\lambda^*}(x)$ – x^* is a nondegenerate unconstrained local minimizer of this function. And the Theorem says that there exists something which allows us to identify λ^* – this is the "locally dual" to (ECP) problem

$$(\mathrm{D}) \quad \underline{L}(\lambda) \to \max;$$

this is fine problem of maximizing a smooth concave function, and λ^* is the unique (and non-degenerate) unconstrained maximizer of the function.

Of course, in order to solve (D), we should be able to compute the values and at least the first order derivatives of $\underline{L}$. Here again the above Theorem gives us all necessary tools: in order to compute the value of $\underline{L}(\lambda)$, we should minimize $L_\lambda(x)$ over V; according to item (i) of the Theorem, this latter problem is well-posed and basically unconstrained: the Theorem says that for λ close enough to λ^* L_λ possesses unique critical point $x^*(\lambda)$ in V, and this point is nondegenerate local minimizer. Thus, we can approximate $x^*(\lambda)$ by unconstrained minimization tools. And from (11.1.3) it follows that $x^*(\lambda)$ gives us not only the value of $\underline{L}(\lambda)$ of the dual objective, but also its gradient – this gradient is simply the vector $h(\cdot)$ of constraints at $x^*(\lambda)$.

With these observations, we see that it is possible to approximate λ^* applying to $\underline{L}(\cdot)$ an unconstrained optimization method (Gradient Descent, Quasi-Newton or whatever else); in order to provide the method by the required first-order information at the current dual iterate λ, it suffices to find, again by unconstrained minimization, the minimizer $x^*(\lambda)$ of the function $L_\lambda(x)$. And what is nice in this picture, is that there are no "large parameters" – both the dual problem of maximizing $\underline{L}(\lambda)$ and the auxiliary primal problems of minimizing $L_\lambda(x)$ possess, at least in neighborhoods of their solutions, nice smoothness and nondegeneracy properties, and the values of the parameters responsible for these latter properties are not spoiled from iteration to iteration.

11.1.2 "Penalty convexification"

The considerations of the previous subsection were conditional – "if we are lucky to be in situation (!), then...". What to do if we are not lucky to be in this situation? Well, what follows demonstrates that in order to become lucky it suffices to be smart – we always can enforce (!) to be the case, and what helps us to reach this goal, is the idea of penalization we know from the previous lecture. Namely, let us add to the actual objective f of (ECP) the quadratic penalty term, thus coming to the penalized objective

$$f_\rho(x) = f(x) + \frac{1}{2}\rho \sum_{i=1}^{m} h_i^2(x),$$

but *in contrast to what we did in the pure penalty method, let us keep the constraints as they were*, thus coming to the problem

$$(\mathrm{ECP}_\rho) \quad f_\rho(x) \to \min \mid\ h(x) = 0.$$

Since the penalty term is zero on the feasible surface, the new problem is equivalent to the initial one, and you can ask what is the goal of this strange manipulation. The answer is as follows:

(+) *if x^* is a nondegenerate solution to the original problem (ECP), then it is nondegenerate local solution to (ECP$_\rho$) as well* [which by itself is not that important] *and, moreover, for all large enough ρ this nondegenerate solution to (ECP$_\rho$) satisfies the crucial property (!).*

The immediate consequence of (+) is as follows: passing from (ECP) to equivalent problem (ECP$_\rho$), we may ensure (!) and thus get all nice possibilities of solving (ECP$_\rho$) (or, which is the same, (ECP)) outlined in the previous section.

Even if you believe in (+), you may ask: all this is fine, but we again meet with "large enough" ρ; this large ρ will finally occur in the objectives $L_\lambda(\cdot)$ of the auxiliary primal problems given by the scheme of the previous section, and we know from our discussion of the penalty methods that it would make these auxiliary problems very difficult for numerical solution. This could be a quite reasonable objection, if there were no crucial difference between the situation with penalty methods and our now situation. In the penalty scheme, the value of the penalty parameter is directly linked with the accuracy ϵ to which we are going to solve the problem: as we remember, violations of the constraints at the unconstrained minimizer of penalized objective are of order of $1/\rho$, and to make these violations $\leq \epsilon$ we *must* work with penalty parameter of order of $1/\epsilon$. The role of penalization now is quite different: it should enforce (!), i.e., positive definiteness (on the entire space) of the Hessian of the Lagrangian $L^\rho(x,\lambda)$ of the penalized problem, and (+) says that all large enough values of ρ are appropriate for this purpose. Whenever a given ρ results in positive definite $\nabla_x^2 L^\rho(x^*,\lambda^*)$, we may use this value of the penalty in the scheme of the previous section to solve problem (ECP$_\rho$) (or, which is the same, our original problem – they are equivalent!) to arbitrarily high accuracy ϵ. Thus, in our now situation we are not enforced to push the penalty parameter to infinity and will be satisfied by fixed value of it; whether this value indeed must be large and will cause numerical difficulties when solving auxiliary primal problems, or it is quite moderate and no problems of this type occur, it depends on local properties of the data, but not on the accuracy to which the problem should be solved. The simplest way to see the indicated difference is to look what happens in the convex case (linear constraints, convex objective). It can be easily seen that here *each* positive value of ρ makes $\nabla^2 L^\rho(x^*,\lambda^*)$ positive definite, so that each value of ρ is "large enough"; note that in the penalty scheme we should work with indeed large penalties independently of whether the problem is convex or not.

It is time now to support our claim (+). To see that (+) indeed is valid, let us denote by λ^* the vector of Lagrange multipliers associated with x^* in the original problem (ECP); as we know, this vector is uniquely defined by the KKT condition

$$\nabla_x f(x^*) + \sum_{i=1}^m \lambda_i^* \nabla h_i(x^*) = 0.$$

Now, the gradient of the penalized objective is

$$\nabla_x f_\rho(x) = \nabla_x f(x) + \rho \sum_{i=1}^m h_i(x) \nabla h_i(x),$$

and we see that at the feasible surface (and, in particular, at the point x^*), where $h(x)=0$, the gradients of the actual and the penalized objectives coincide with each other; thus, λ^* serves as the vector of Lagrange multipliers associated with x^* in problem (ECP$_\rho$) as well.

Now let us compute the Hessian with respect to x of the Lagrange function

$$L^\rho(x,\lambda) = f_\rho(x) + \sum_{i=1}^m \lambda_i h_i(x) \equiv f(x) + \sum_{i=1}^m \lambda_i h_i(x) + \frac{1}{2}\rho \sum_{i=1}^m h_i^2(x) \equiv$$

$$\equiv L(x,\lambda) + \frac{1}{2}\rho \sum_{i=1}^m h_i^2(x) \tag{11.1.4}$$

of the modified problem (ECP_ρ) (from now on, $L(x,\lambda)$ means the Lagrangian of the original problem). We have

$$\nabla_x L^\rho(x,\lambda) = \nabla_x L(x,\lambda) + \rho \sum_{i=1}^m h_i(x)\nabla_x h_i(x),$$

whence

$$\nabla_x^2 L^\rho(x,\lambda) = \nabla_x^2 L(x,\lambda) + \rho \sum_{i=1}^m \nabla_x h_i(x)[\nabla_x h_i(x)]^T + \rho \sum_{i=1}^m h_i(x)\nabla_x^2 h_i(x).$$

Introducing notation

$$H(x,\lambda) = \nabla_x^2 L(x,\lambda), \quad A(x) = \nabla_x h(x) \equiv \begin{pmatrix} [\nabla h_1(x)]^T \\ \ldots \\ [\nabla h_m(x)]^T \end{pmatrix}, \quad R(x) = \sum_{i=1}^m h_i(x)\nabla_x^2 h_i(x),$$

we can rewrite the resulting equality as

$$\nabla_x^2 L^\rho(x,\lambda) = H(x,\lambda) + \rho A^T(x)A(x) + \rho B(x). \tag{11.1.5}$$

We would like to prove that the left hand side in this relation at the point $(x = x^*, \lambda = \lambda^*)$ is positive definite, provided that ρ is large enough. Now let us make the following useful observations:

- $H(x^*,\lambda^*)$ is positive definite at the subspace T of directions tangent to the feasible surface at the point x^*; this is the second order part of the Second Order Sufficient Optimality condition which is assumed to be satisfied at x^* (recall that we all the time are speaking about the case when x^* is a nondegenerate solution to (ECP)). As we know, T is exactly the kernel of the matrix $\nabla h(x^*) = A(x^*)$:

$$T = \{d \mid \ A(x^*)d = 0\};$$

- the matrix $A^T(x^*)A(x^*)$ is positive semidefinite (as any matrix of the form B^TB: indeed, $d^TB^TBd = |Bd|^2$); its kernel, as it is immediately seen from the computation in parentheses, is exactly the kernel of $A(x^*)$, i.e., T;

- The matrix $R(\cdot)$ simply vanishes at x^*, due to presence of factors $h_i(x)$ in the expression for the matrix; at the point x^* – this point is feasible for (ECP) – these factors become zero, so that $R(x^*) = 0$.

We conclude that

$$\nabla_x^2 L^\rho(x^*,\lambda^*) = H(x^*,\lambda^*) + \rho A^T(x^*)A(x^*),$$

and the fact that the right hand side is positive definite for large enough ρ is an immediate consequence of the following

Lemma 11.1.1 *Let D be positive semidefinite matrix and C be a symmetric matrix which is positive definite on the kernel of D. Then the matrix $C + \rho D$ is positive definite for all large enough values of ρ.*

The Lemma is very close to Lemma 10.2.1; you may treat its proof as an optional home task.

To derive (+) from above observations, it suffices to apply the Lemma to $D = A^T(x^*)A(x^*)$ and $C = H(x^*, \lambda)$.

11.2 Putting things together: Augmented Lagrangian Scheme

The algorithmic scheme implementing the above ideas is called the Augmented Lagrangian method, or the Multipliers method; the first name comes from the fact that the Lagrange function (11.1.4) of the problem (ECP_ρ) can be treated as certain "augmentation" of the Lagrange function of the original problem (ECP); the second name stresses the leading role of the Lagrange multipliers in the method.

The generic Augmented Lagrangian scheme is as follows. Assume that we already have chosen somehow value of the penalty parameter ρ and thus deal with problem

$$(\mathrm{ECP}_\rho) \quad f(x) + \frac{\rho}{2}|h(x)|^2 \to \min \mid \; h(x) = 0$$

equivalent to the original problem (ECP), and let ρ be large enough to ensure (!) – let the Hessian $\nabla_x^2 L^\rho(x, \lambda)$ of the Lagrange function

$$L^\rho(x, \lambda) = f(x) + \frac{\rho}{2}\sum_{i=1}^m h_i^2(x) + \sum_{i=1}^m \lambda_i h_i(x)$$

taken at the point (x^*, λ^*) be positive definite on the entire space; here x^* is the nondegenerate solution of (ECP) to be approximated and λ^* is the corresponding vector of Lagrange multipliers. To make further considerations clear, let us summarize what is said about the situation by the discussion of Section 11.1.1.

We know that

- to find x^* is the same as to solve the dual problem

$$(\mathrm{D}^\rho) \quad \underline{L}^\rho(\lambda) \equiv \min_x L^\rho(x, \lambda) \to \max$$

  ($\min_x$ here should be taken locally, as was explained in Section 11.1.1);

- the objective in the dual problem is everywhere concave and twice continuously differentiable in a neighbourhood of λ^*, and λ^* is nondegenerate solution to the dual problem; thus, the dual problem is well-suited for numerical solution, *provided that we are able to compute the value and the gradient of the dual objective at a point*

- we do have a possibility to compute the value and the gradient of the dual objective at a point, at least at a point close to λ^*: to this end it suffices to find the solution $x^\rho(\lambda)$ of the auxiliary unconstrained problem

$$(\mathrm{P}_\lambda^\rho) \quad \min_x L^\rho(x, \lambda)$$

  (where $\min_x$ again should be taken locally) and to set

$$\underline{L}^\rho(\lambda) = L^\rho(x^\rho(\lambda), \lambda), \quad \nabla_\lambda \underline{L}^\rho(\lambda) = h(x^\rho(\lambda)). \tag{11.2.1}$$

Now we may run any method for unconstrained maximization of the dual objective $\underline{L}^\rho$ (*"outer iterations"* generating sequence $\lambda_0, \lambda_1, ...$ of iterates converging to λ^*), accompanying every step t of the method by solving the auxiliary problem $(\mathrm{P}^\rho_{\lambda_{t-1}})$ in order to find

$$x_{t-1} = x^\rho(\lambda_{t-1})$$

and thus to get the first order information on the dual objective which is required to update λ_{t-1} into λ_t.

Implementations of the scheme differ from each other mainly by methods we use to maximize the dual objective and to solve the auxiliary primal problems.

11.2.1 Solving auxiliary primal problems $(\mathrm{P}^\rho_\lambda)$

The best choice here is, of course, the Newton method with linesearches or the modified Newton method, when the second order information on the objective and the constraints is available. If it is not the case, we could use Quasi-Newton methods, Conjugate Gradient, etc. When solving the auxiliary problems, we should use reasonable *safeguard policy* in order to ensure that our minimization is indeed local. Since we do not know in advance what is the neighbourhood V introduced in Section 11.1.1, this is not an easy task. The simplest policy here is to use the solution to the previous auxiliary problem as the starting point when solving the current one and to take specific care of linesearches (e.g., to move until the closest minimum of the current auxiliary objective on the current search line). With reasonable tactics of this type we indeed may ensure that if the starting point in the very first auxiliary problem is close enough to x^* and the initial guess for λ^* is close enough to λ^*, then at all steps of the method we shall indeed approximate $x^\rho(\cdot)$ and not other tentative local solutions to the auxiliary primal problems. And of course to use the previous primal iterate as the starting point in the new auxiliary primal problem saves a lot of computational effort.

11.2.2 Solving the dual problem

When solving the dual problem (D^ρ) – this is the upper-level process in our two-level scheme – we are less flexible when choosing the optimization method: since the information on the dual objective is "implicit" – to get its value and gradient, we should solve an auxiliary primal optimization problem, it normally is computationally expensive, and things like line search become very costly. The simplest algorithm we could use in the outer iterates is the following scheme:

$$\lambda_t = \lambda_{t-1} + \rho h(x_{t-1}). \tag{11.2.2}$$

Relation (11.2.2) has a very simple motivation: since x_{t-1} is unconstrained minimizer of $L^\rho(\cdot, \lambda_{t-1})$, we have

$$0 = \nabla_x L^\rho(x_{t-1}, \lambda_{t-1}) = \nabla_x f(x) + \sum_{i=1}^m \rho h_i(x) \nabla_x h_i(x) + \sum_{i=1}^m (\lambda_{t-1})_i \nabla h_i(x) =$$

$$= \nabla_x f(x_{t-1}) + \sum_{i=1}^m [(\lambda_{t-1})_i + \rho h_i(x_{t-1})] \nabla h_i(x).$$

Comparing this equality with the one

$$0 = \nabla_x f(x^*) + \sum_{i=1}^m \lambda_i^* \nabla_x h_i(x^*)$$

defining the Lagrange multipliers λ^*, we see that the right hand side in (11.2.2) can be viewed as a natural approximation to λ^*, the better the closer is x_{t-1} to x^*.

This is only motivation, of course; now let us look what indeed can be said about recurrence (11.2.2). From (11.2.1) we observe that $h(x_{t-1})$ is exactly the gradient of the dual objective at λ_{t-1}, so that (11.2.2) is in fact Gradient Ascent method for the dual problem with constant stepsize ρ (recall that the dual problem is the maximization one, so that a gradient-type method here should perform steps along the gradient rather than along the antigradient, as it is in the minimization case; this is why we are speaking about Gradient *Ascent*).

Dual rate of convergence

To understand whether the stepsize ρ in Gradient Ascent is or is not good for us, let us look at the eigenvalue structure of the Hessian of the dual objective at λ^* – we know from our investigation of the Gradient method that this structure is exactly what governs the asymptotical rate of convergence of the method.

The Hessian of the dual objective can be found as follows (to save notation, I write $x(\lambda)$ instead of $x^\rho(\lambda)$): we already know that

$$\nabla_\lambda \underline{L}^\rho(\lambda) = h(x(\lambda))$$

and $x(\cdot)$ is continuously differentiable in a neighbourhood of $\lambda = \lambda^*$ by Theorem 11.1.1. Differentiating both sides in λ, we get

$$\nabla^2_\lambda \underline{L}^\rho(\lambda) = \nabla_x h(x(\lambda))\nabla_\lambda x(\lambda). \tag{11.2.3}$$

To find $\nabla_\lambda x(\lambda)$, let us differentiate in λ the relation

$$\nabla_x L^\rho(x(\lambda), \lambda) = 0,$$

which defines $x(\lambda)$. We get

$$\nabla^2_x L^\rho(x(\lambda), \lambda)\nabla_\lambda x(\lambda) + \frac{\partial^2}{\partial\lambda\partial x} L^\rho(x(\lambda), \lambda) = 0;$$

as it is immediately seen from the expression for $L\rho$, the second term in the left hand side of the latter equation is $[\nabla_x h(x(\lambda))]^T$, and we get

$$\nabla_\lambda x(\lambda) = -[\Phi^\rho(\lambda)]^{-1}[\nabla_x h(x(\lambda)]^T,$$

Φ^ρ being the Hessian of the augmented Lagrangian L^ρ with respect to x taken at the point $(x(\lambda), \lambda)$. Substituting this expression into (11.2.3), we get

$$\nabla^2_\lambda \underline{L}^\rho(\lambda) = -[\nabla_x h(x(\lambda))][\Phi^\rho(\lambda)]^{-1}[\nabla_x h(x(\lambda)]^T, \tag{11.2.4}$$

Substituting expression for the Hessian of the augmented Lagrangian L^ρ in x (see (11.1.5), we get the following important conclusion:

The Hessian Ψ^ρ of the dual objective $\underline{L}^\rho(\cdot)$ at the point λ^* is given by

$$\Psi^\rho = -A[H + \rho A^T A]^{-1} A^T, \tag{11.2.5}$$

$A = \nabla_x h(x^*)$ and

$$H = \nabla_x^2 f(x^*) + \sum_{i=1}^m \lambda_i \nabla_x^2 h_i(x^*)$$

is the Hessian with respect to x of the Lagrange function of the original problem (ECP), the Hessian being taken at the point (x^*, λ^*).

Relation (11.2.5) allows to make the following crucial observation:

Proposition 11.2.1 *The matrix $-\rho\Phi^\rho$ tends to the unit matrix I as $\rho \to \infty$.*

Proof. Assuming, for the sake of simplicity, that the matrices H and $AH^{-1}A^T$ are nonsingular (this assumption can be easily eliminated by a little bit more complicated reasoning) and applying Sherman-Morrison formula (Exercise 6.6.2), we get

$$[H + \rho A^T A]^{-1} = H^{-1} - H^{-1}A^T[\rho^{-1}I_m + AH^{-1}A^T]^{-1}AH^{-1};$$

denoting $Q = AH^{-1}A^T$ and taking into account (11.2.5), we get

$$-\Psi^\rho = Q - Q[\rho^{-1}I_m + Q]^{-1}Q.$$

Applying the formula[2)]

$$[Q + \rho^{-1}I_m]^{-1} = \left[Q(I_m + \rho^{-1}Q^{-1})\right]^{-1} = [I_m + \rho^{-1}Q^{-1}]^{-1}Q^{-1} =$$

$$= [I_m - \rho^{-1}Q^{-1} + o(\rho^{-1})]Q^{-1} = Q^{-1} - \rho^{-1}Q^{-2} + o(\rho^{-1}),$$

we come to

$$-\Psi^\rho = Q - Q[Q^{-1} - \rho^{-1}Q^{-2} + o(\rho^{-1})]Q = \rho^{-1}I_m + o(\rho^{-1}),$$

and the conclusion follows. ∎

The result stated in the Proposition is indeed crucial for us, due to the following

Corollary 11.2.1 [Dual convergence rate] *Let ρ be large enough, and let the starting point λ_0 in recurrence (11.2.2) be close enough to λ^*. Then recurrence (11.2.2) converges to λ^* linearly with the convergence ratio $\kappa(\rho)$ which tends to 0 as $\rho \to \infty$.*

The statement given by the corollary is indeed very important: it establishes (local) linear convergence of simple and easily implementable recurrence (11.2.2), provided that ρ is large enough; moreover, the convergence can be done arbitrarily fast, if ρ is large enough. Of course, this observation does not say that in actual computations you should set ρ to a huge value and get the solution in one step: this "one step" is one step of outer iteration in λ, and to perform this step, you should solve the auxiliary primal problem with huge value of penalty, which, as we already know, is very difficult (note also that we know without any dual considerations that solving the auxiliary primal problem with huge ρ we get good approximate solution to (ECP) – this is the penalty method). The actual meaning of the statement is, of course, that we may hope to deal with "large enough" value of the penalty which will ensure reasonable rate

[2)] coming from the standard expansion

$$(I_m + \epsilon S)^{-1} = I_m - \epsilon S + \epsilon^2 S^2 - \epsilon^3 S^3 + ...$$

which is valid, for all small in absolute value ϵ, also in the matrix case

of convergence of (11.2.2) and at the same time will be not "too large" to make the auxiliary primal problems very difficult.

It is time now to prove the Corollary. I shall give the model of the proof only. Namely, let us believe that the behaviour of recurrence (11.2.2) started close enough to λ^* is basically the same as if the dual objective was the quadratic function

$$\psi(\lambda) = \frac{1}{2}(\lambda - \lambda^*)^T \Psi^\rho (\lambda - \lambda^*);$$

it indeed is so, but let us skip the required dull (although simple) justification. With our quadratic model of the dual objective, the result can be obtained in one line. Recall that (11.2.2) is Gradient Ascent with stepsize ρ, so that

$$\lambda_t - \lambda^* = \lambda_{t-1} - \lambda^* + \rho \nabla_\lambda \psi(\lambda_{t-1}) = (I_m + \rho \Psi^\rho)(\lambda_{t-1} - \lambda^*);$$

thus, the residual vector $\lambda_t - \lambda^*$ is multiplied at each step by the matrix $I_m + \rho\Psi^\rho$, and this matrix tends to 0 as $\rho \to \infty$ by Proposition 11.2.1 and therefore is close to 0 for large ρ. ∎

One may ask whether indeed simple recurrence (11.2.2) is the best we can use. The question is quite reasonable: for fixed ρ – and our main motivation of the entire method was the desire to deal with perhaps large, but fixed value of the penalty parameter – the rate of convergence of this recurrence is linear. It is easily seen that the distance between x_t and the exact solution x^* is of order of distance between λ_t and λ^*, so that with recurrence (11.2.2) we get only linear convergence of both primal and dual variables. Could we get something better? Of course we could: relation (11.2.4) expresses the Hessian of the dual objective via the data which are available after $x(\lambda) = x^\rho(\lambda)$ is found from the solution of the auxiliary primal problem; thus, we could solve the dual problem by the Newton method, thus ensuring quadratic rate of local convergence of the scheme.

11.2.3 Adjusting penalty parameter

There is one more important issue to be discussed: how to choose ρ? For the time being, we simply have assumed that the penalty parameter is chosen in a way which ensures (!). What we know is that *all* large enough values of the penalty fit this requirement, but how to choose an appropriate ρ in practice? Advice like "set ρ to something huge" definitely is of no interest: with this approach, we convert the method into the penalty one and loose all computational advantages associated with possibility to deal with moderate values of the penalty.

The simplest way to *tune* the penalty is as follows. As we know from the previous analysis, when ρ is "appropriately large" to ensure linear rate of convergence of (11.2.2) with reasonable convergence ratio, then the norm of the gradient of the dual objective also should decrease linearly with reasonable convergence ratio. We know what is this gradient – this is the vector of constraints at the solution to the corresponding auxiliary primal problem; thus, for appropriately large ρ the quantities $|h(x_t)|$ decrease with t in geometric progression with the ratio which, in principle, can be done arbitrarily close to 0, provided that ρ is large enough. Let us use this observation as the driving force in our policy for adjusting the penalty. Namely, let us start the procedure with certain ad hoc value of ρ and look whether the residuals $|h(x_t)|$ indeed reasonably decrease with t, say, at least by factor 0.25 each time. Until it is so, we keep the penalty constant; and if at certain step t the current value of $|h(x_t)|$ is too large – exceeds $0.25|h(x_{t-1}|$ – we interpret this as a signal to increase the penalty by certain factor, say, 10. Then we resolve the auxiliary primal problem with this new value of the penalty and the same

as before value of λ, again look whether we got required progress in constraint violation, and if not – again increase the penalty, and so on.

Of course, we could use another "driving forces" in penalty updating policy, but the progress in constraint violation is the most natural one: all we are interested in is in fact to make the constraint violation small. Indeed, at the solution x_t to the auxiliary primal problem the gradient of the objective for sure is linear combination of the gradients of constraints (write down the Fermat rule for the auxiliary problem), so that the violations of the constraints form the only obstacle for x_t to be KKT point of (ECP); thus, it is very natural to update the penalty on the basis of progress in diminishing this obstacle.

Let me also note that the indicated policy in fact incorporates both the penalty and the Lagrange Multiplier technique. Indeed, this policy increases the penalty until the solution to the current auxiliary primal problem results in prescribed progress in constraint violation; independently of what we do with the Lagrange multipliers and how high is the rate of dual convergence, we will achieve eventually this goal via the penalty mechanism.

11.3 Incorporating Inequality Constraints

For the moment, we know only how to apply the Augmented Lagrangian Scheme to an equality constrained problem. In fact there is no difficulty to incorporate into the Augmented Lagrangian Scheme inequality constraints, and to this end it suffices to use the reduction of a general constrained problem to an equality constrained one (Lecture 8). Namely, given a constrained problem

$$\text{(GCP)} \quad f(x) \to \min \mid h_i = 0,\ i = 1, ..., m,\ g_j(x) \le 0,\ j = 1, ..., k,$$

we can transform it equivalently into the equality constrained problem

$$\text{(ECP}^*\text{)} \quad f(x) \to \min \mid h_i(x) = 0,\ i = 1, ..., m,\ g_j(x) + s_j^2 = 0,\ j = 1, ..., k$$

with extended list of variables. Applying the Augmented Lagrangian Scheme to the resulting problem, we come to the necessity to work with the augmented Lagrangian

$$L^\rho(x, s; \lambda, \mu) = f(x) + \sum_{i=1}^m \lambda_i h_i(x) + \sum_{j=1}^k \mu_j [g_j(x) + s_j^2] + \frac{\rho}{2}\left[\sum_{i=1}^m h_i^2(x) + \sum_{j=1}^k [g_j(x) + s_j^2]^2\right];$$

what we should do with this Lagrangian is to find its (local) saddle point, and to this end the scheme in question suggests to maximize with respect to the dual variables λ, μ the function

$$\underline{L}^\rho(\lambda, \mu) = \min_{x,s} L^\rho(x, s; \mu, \lambda).$$

In fact we have no problems with minimization with respect to s_j in the right hand side of the latter relation. Indeed, explicit computation results in

$$L^\rho(x, s; \lambda, \mu) = f(x) + \frac{\rho}{2}\sum_{j=1}^k \left[g_j(x) + s_j^2 + \frac{\mu_j}{\rho}\right]^2 + \sum_{i=1}^m \lambda_i h_i(x) + \frac{\rho}{2}\sum_{i=1}^m h_i^2(x) - \sum_{j=1}^k \frac{\mu_j^2}{2\rho}.$$

It follows, consequently, that minimization of the Lagrange function with respect to the part s of primal variables results in setting s_j^2 equal to the quantity $-(g_j(x) + \mu_j/\rho)$, when this quantity is nonnegative, or to zero, when this quantity is negative. Consequently,

$$\underline{L}^\rho(\lambda, \mu) = \min_x \{f(x) + \frac{\rho}{2}\sum_{j=1}^k \left(g_j(x) + \frac{\mu_j}{\rho}\right)_+^2 + + \sum_{i=1}^m \lambda_i h_i(x) + \frac{\rho}{2}\sum_{i=1}^m h_i(x)^2\} - \sum_{j=1}^k \frac{\mu_j^2}{2\rho}, \quad (11.3.1)$$

where, as in the previous Lecture, $(a)_+$ is the positive part of real a (a itself, if it is positive, and 0 otherwise).

Thus, the auxiliary primal problems arising in the Augmented Lagrangian Scheme in fact are problems in the initial design variables.

From theoretical viewpoint, the following question is important. Validity of the Augmented Lagrangian Scheme for equality constrained problem (i.e., the theoretical local convergence results of the previous sections) required nondegeneracy of the solution x^* to be approximated. Thus, we should understand when a locally optimal solution x^* to (GCP), extended in the evident way (by setting $s_j^* = \sqrt{-g_j(x^*)}$) to solution (x^*, s^*) of (ECP*), results in a nondegenerate solution to the latter problem.

The answer is as it should be:

Proposition 11.3.1 *If x^* is a nondegenerate solution to* (GCP) [3)]*, then (x^*, s^*) is nondegenerate solution to* (ECP*)*.*

Proof of the Proposition is part of the assignment to the lecture.

11.4 Convex case: Augmented Lagrangians; Decomposition

All results of Section 11.1.1 were local; this is quite natural, since there we dealt with general-type nonconvex program. In the case of *convex* program all results can be globalized; let us briefly outline the good things happening in this case. Thus, consider a *convex* optimization program

$$\text{(CnvP)} \quad f(x) \to \min \mid g_j(x) \le 0,\ j = 1, ..., k,$$

where f and g_j are convex and twice continuously differentiable functions on $\mathbf{R}^n$. From now on, let us assume that the problem is *solvable* and *satisfies the Slater condition*: there exists $\bar{x}$ such that all constraints are strictly negative at $\bar{x}$.

Those who have passed through the course "Optimization I" know, and other should believe that under the indicated assumptions one can define the problem *dual* to (CnvP), namely, the problem

$$\text{(CnvD)} \quad \phi^*(\mu) \to \max \mid \mu \ge 0,$$

where $\mu = (\mu_1, ..., \mu_k)$ are the *dual variables* and

$$\phi^*(\mu) = \inf_{x \in \mathbf{R}^n} L(x, \mu),$$

with

$$L(x, \mu) = f(x) + \sum_{j=1}^{k} \mu_j g_j(x)$$

being the classical Lagrange function of problem (CnvP). From the construction of the dual problem it follows that the dual objective is concave function taking values in the extended by $-\infty$ real axis. Due to the concavity of the dual objective, (CnvD) is in fact *convex* optimization problem (this is the same – to maximize a concave function ϕ^* or to minimize its negation $-\phi^*$, which is a convex function).

[3] Definition 8.2.4: x^* is regular for the constraints, satisfies the Second Order Sufficient Optimality conditions and, besides this, Lagrange multipliers related to all inequality constraints active at x^* are strictly positive – this is called *strict* complementary slackness

The links between the primal problem (CnvP) and its dual (CnvD) are given by the *Convex Programming Duality Theorem* which states that

If (CnvP) *is solvable and satisfies the Slater condition, then* (CnvD) *also is solvable, and the optimal values in the problems are equal to each other. In this case a feasible solution x^* to* (CnvP) *and a feasible solution μ^* to* (CnvD) *are optimal for the corresponding problems if and only if (x^*, μ^*) is a saddle point of the Lagrange function $L(x, \mu)$ on $\mathbf{R}^n \times \mathbf{R}^k_+$ (i.e., $L(x, \mu^*)$ attains its minimum over $x \in \mathbf{R}^n$ at $x = x^*$, while $L(x^*, \mu)$ attains its maximum over $\mu \geq 0$ at $\mu = \mu^*$).*

In particular, all optimal solutions to (CnvP) *are among unconstrained minimizers of the function $L(x, \mu^*)$, μ^* being any optimal solution of the dual problem.*

Note that Theorem 11.1.1 is a local version of this Duality Theorem.

The Duality Theorem has a lot of applications, both of theoretical and of computational type. From the *computational* viewpoint, there are two ways to exploit the theorem:

- There are cases when the structure of the data in (CnvP) allows to compute the dual objective analytically (this is the case in Linear, Linearly Constrained Quadratic and Geometric Programming). Whenever this is the case, we may gain a lot solving (CnvD) instead of (CnvP) and then recovering the primal solutions from the dual ones (which sometimes is easy and sometimes is not, see below).

Computational consequences of this type clearly are restricted by our abilities to compute the dual objective analytically; in our today lecture we are not interested in these (in fact very important) "particular cases".

- In the general case we can implement the Lagrange Multipliers scheme, namely, to solve (CnvD) numerically by a first-order method for convex optimization under *linear* inequality constraints (note that the only explicit inequalities in (CnvD) are $\mu_j \geq 0$); thus, we may switch from *nonlinearly* constrained primal problem to a problem with *simple linear* inequalities. Of course, to solve the dual problem, we need possibility to compute its objective and its gradient at a given point; same as in the Augmented Lagrangian Scheme for general-type problems, to this end we can solve numerically the problem of minimizing the Lagrangian in x with $\mu \geq 0$ being fixed. In our now situation, this "auxiliary primal problem" is fine - with smooth convex objective, so that we have good possibilities to approximate the global solution to the auxiliary problem.

11.4.1 Augmented Lagrangians

In this section we focus on the second of the aforementioned issues, on possibility to reduce a general type primal convex program to a linearly constrained dual one.

The outlined scheme is "globalization" of the one developed in the previous sections, but, for the time being, without "augmenting" the Lagrangian. What is fine is that now we are in convex situation, so that all "local minimizations" from Section 11.1.1 can be replaced by global ones, we should not bother much about "good enough" primal and dual starting points, safeguards, etc.

There is, anyhow, a weak point in the classical Lagrange function. Namely, it may happen that

- the dual objective ϕ^* may be nonsmooth and even infinite at some points; whenever it is the case, it is unclear how to solve the dual problem numerically

The indicated difficulty is not something rare: this is exactly what happens in the Linear Programming case. Here the dual objective turns out to be linear in certain polytope and to be $-\infty$ outside this polytope, and (CnvD) (the usual LP dual program) normally is as difficult as the primal problem.

- it may happen that the auxiliary primal problem *associated with the optimal vector μ^* of Lagrange multipliers*, the one being the optimal solution to (CnvD), has many optimal solutions. The Convex Programming Duality Theorem says that all optimal solutions to (CnvP) for sure are optimal for the auxiliary primal problem in question, but the Theorem does not say that the latter problem has no other optimal solutions. Whenever it has "extra" optimal solutions, we meet with nontrivial problem how to select the solutions we actually are interested in, i.e, those optimal for (CnvP), from the minimizers of $L(\cdot, \mu^*)$.

This second difficulty also is not something rare: in the case of Linear Programming the function $L(x, \mu^*)$, μ^* being optimal solution to (CnvD), is constant, so that its optimal set – the entire $\mathbf{R}^n$ – says nothing about the actual optimal set of (CnvP).

In fact both indicated difficulties have the same origin: possible non-existence or non-uniqueness of optimal solution to the auxiliary problem

$$\min_x L(x, \mu) \tag{11.4.1}$$

If we somehow assure existence and uniqueness of the solution $x^*(\lambda)$ in the latter problem and its continuity in μ, then we, first, will get continuously differentiable dual objective. Indeed, it can be proved (cf. Section 11.1.1) that in the case in question

$$\nabla_\mu \phi^*(\mu) = g(x^*(\mu)), \tag{11.4.2}$$

which is the continuous function of μ, provided that $x^*(\mu)$ is continuous. Second, in the case under consideration we clearly shall be able to restore the optimal solution to the primal problem via the optimal solution to the dual.

There is one evident case when problem (11.4.1) has exactly one solution: this is the case when the sum

$$r(x) = f(x) + \sum_{j=1}^{k} g_j(x)$$

of the objective and the constraints is *strongly convex* (with positive definite Hessian at every point) and grows faster than $|x|$ as $|x| \to \infty$:

$$r(x)/|x| \to \infty, \;\; |x| \to \infty.$$

Indeed, in this case it is easily seen that the objective in (11.4.1) for every $\mu > 0$ also is strongly convex and goes to infinity as $|x| \to \infty$; consequently, (11.4.1) has unique solution (which, as it is easily seen, is continuous in $\mu > 0$). Consequently, in the case under consideration application of the Lagrange Multipliers Scheme causes no conceptual difficulties.

What to do if the just mentioned sufficient condition of "well-posedness" of (11.4.1) is not satisfied? A good idea is to pass to *augmented Lagrange function* – basically the same idea we used in the general case of nonconvex problems. Namely, let θ_j be *increasing strongly convex* functions on the axis normalized by the conditions

$$\theta_j(0) = 0, \;\; \theta_j'(0) = 1.$$

The problem

$$(\text{CnvP}^*) \quad f(x) \to \min \mid \theta_j(g_j(x)) \le 0,\ j = 1, ..., k$$

clearly is equivalent to (CnvP) and, as it is easily seen, also is convex. One can straightforwardly verify that, due to the normalization conditions on θ_j, the vector of optimal Lagrange multipliers for the new problem is the same as for the old one (the new dual objective is, anyhow, different from the old one). Under mild regularity assumptions one can prove that the modified problem satisfies the aforementioned sufficient condition for well-posedness of problems (11.4.1); e.g., if all g_j are linear, this sufficient condition is satisfied whenever the feasible domain of (CnvP) is bounded. Thus, by passing from (CnvP) to equivalent problem (CnvP*), or, which is the same, passing from the classical Lagrange function $L(x, \mu)$ of problem (CnvP) the the *augmented Lagrange function*

$$\mathcal{L}(x, \mu) = f(x) + \sum_{j=1}^{k} \mu_j \theta_j(g_j(x))$$

(which is the classical Lagrange function of the equivalent problem), we normally get possibility to solve the constrained problem (CnvP) by the Lagrange Multipliers Scheme (which now becomes augmented).

All this is completely similar to the Augmented Lagrangian Scheme we used in the general case, including possibility to introduce penalty parameter into our augmented scheme for (ConvP); the "penalized" augmented Lagrangian for (CnvP) is

$$\mathcal{L}^\rho(x, \mu) = f(x) + \sum_{j=1}^{k} \mu_j \rho^{-1} \theta_j(\rho g_j(x));$$

this corresponds to replacing $\theta_j(\cdot)$ by the rescaled functions $\bar{\theta}_j(s) = \rho^{-1}\theta_j(\rho s)$, which also satisfy the normalization conditions. In the general (nonconvex) case the role of the penalty parameter is to ensure the crucial property (!); besides this, large penalty, as we remember, enforces the Hessian of the dual objective at λ^* to be almost proportional to the unit matrix and thus assures fast local convergence of simple gradient-type methods for solving the dual problem. In our now case, we are not interested in the first of these two issues – due to the convexity of the original problem, (!) is ensured even with small values of ρ; the second advantage of penalization, anyhow, still is important: the larger is ρ, the better is conditioning of the dual objective associated with the augmented Lagrange function $\mathcal{L}^\rho$ and the better are our abilities for fast maximization of this objective.

I should stress that the presented brief outline of the Augmented Lagrangian Multipliers Scheme in Convex Programming is far from being complete; e.g., we did not speak in details on how to solve the dual problem associated with the augmented Lagrangian (now this problem is constrained, although with simple constraints, and simple gradient projection routines are possible, but in no sense best possible options). More detailed considerations of these issues are beyond the scope of this course.

11.4.2 Lagrange Duality and Decomposition

Although the below considerations are not related to *Augmented* Lagrange Multipliers Scheme, I cannot avoid them when speaking on Lagrange Duality and its computational consequences. Thus, let us speak about advantages of the *classical* Lagrange Multipliers Scheme in the case of *separable* problems. In many real-world situations objective and constraints of a Convex

Programming program (ConvP) are *separable*: one can partition the vector of design variables x into non-overlapping sub-vectors $x_1, ..., x_l$ in such a way that

$$f(x) = \sum_{i=1}^{l} f_i(x_i),\ g_j(x) = \sum_{i=1}^{l} g_{ji}(x_i).$$

Whenever it is the case, the Lagrange scheme becomes especially attractive due to the following evident property: computation of the dual objective

$$\phi^*(\mu) = \min_x \left[f(x) + \sum_{j=1}^{k} \mu_j g_j(x) \right]$$

clearly reduces to solving l independent problems

$$(P_i) \quad \min_{x_i} \left[f(x_i) + \sum_{j=1}^{k} \mu_j g_{ji}(x_i) \right].$$

$i = 1, ..., k$. Note that the computational complexity of an optimization problem normally is superadditive function of the design dimension: it is easier, and usually - much easier, to solve l independent convex problems (P_i) of a given total design dimension n than to solve a single convex program of dimension n. Therefore separability simplifies a lot implementation of the classical Lagrange scheme (note that the main computational effort in this scheme is the one to solve auxiliary primal problems). In any words, Lagrange scheme, in contrast to other Convex Programming techniques, is capable to exploit separability of the objective and the constraints: it allows to *decompose* the original problem into a collection of independent problems which "interact" with each other only via the "coordinating" dual problem (CnvD).

In fact similar decomposition approach can be carried out also in the case when (CnvP) contains, besides separable objective and constraints, a number of constraints depending each of its own part x_i of the variables. Such a problem can be written down as

$$\text{(CnvPS)} \quad \sum_{i=1}^{l} f_i(x_i) \to \min \mid \sum_{i=1}^{l} g_{ji}(x) \le 0,\ j = 1, ..., k,\ x_i \in X_i,$$

where X_i are the convex sets given by those inequalities which depend on x_i solely.

The corresponding version of Lagrange Duality says that the natural dual problem is

$$\text{(CnvDS)} \quad \sum_{i=1}^{l} \phi_i^*(\mu) \to \max,\ \ \mu \ge 0,$$

where

$$\phi_i^*(\mu) = \min_{x_i \in X_i} \left[f_i(x_i) + \sum_{j=1}^{k} \mu_j g_{ji}(x_i) \right];$$

the difference with the standard duality is that now minimization over x_i in the formulae for dual objective is taken over the set X_i given by the constraints depending on x_i solely, not over the entire space.

The meaning of the words "(CnvDS) is natural dual to (CnvPS)" is given by the corresponding version of the Duality Theorem:

(*) *If (CnvPS) is solvable and satisfies the Slater condition, then (CnvDS) (which always is a Convex Programming problem) also is solvable, the optimal values in the problems are equal to each other, and the pair* (x^*, μ^*) *of feasible primal and dual solutions to the problems is comprised of optimal solutions if and only if* (x^*, λ^*) *is saddle point of the classical Lagrange function* $L(x, \mu)$ *of (CnvPS) on the set* $X \equiv X_1 \times ... \times X_l$ *of values of* x *and the set* $\mathbf{R}^k_+ = \{\mu \in \mathbf{R}^k \mid \mu \geq 0\}$ *of values of* μ*, i.e., if* $L(x, \mu^*)$ *attains its minimum over* $x \in X$ *at* $x = x^*$*, while the function* $L(x^*, \mu)$ *attains its maximum over* $\mu \geq 0$ *at* $\mu = \mu^*$*.*

As a corollary, we get that if μ^* is optimal solution to (CnvDS), then the components x_i^* of (any) optimal solution to (CnvPS) are among the minimizers of the Lagrange functions

$$L_i(x_i, \mu^*) = f_i(x) + \sum_{j=1}^{k} \mu_j^* g_{ji}(x_i)$$

over $x_i \in X_i$. Under assumptions of strong convexity similar to those indicated in the previous subsection, this information is sufficient to restore, given μ^*, the components of optimal primal solution x^*. In the general case we again might meet with the difficulty that there are many minimizers of $L_i(x_i, \mu^*)$ over $x_i \in X_i$, and only some of these minimizers can be combined into optimal solution to (CnvPS). How to overcome this difficulty, this is another story which goes beyond the bounds of this lecture; what should be stressed, anyhow, is that the outlined decomposition scheme allows to obtain important, and sometimes complete information on the optimal solution to (CnvPS).

The Decomposition approach is especially attractive when l and the total number of constraints participating in the description of the sets X_i are large, and there is relatively small number of separable "linking constraints" $g_j(x) \leq 0$. Indeed, in this case the dual problem (CnvDS) is of small dimension, which simplifies it computationally. At the same time, to compute the objective of (CnvDS) at a given μ, one has to solve l independent problems of minimizing $L_i(x_i, \mu)$ over $x_i \in X_i$. This latter task can be achieved by parallel computations, and even with serial computations it is, as I have already mentioned, much easier to solve many "small" independent problems than a single "large" one.

The outlined Decomposition scheme has very nice economical interpretation. Imagine that certain company is comprised of basically independent units; the only interconnections between the units are that all the units consume a number of resources like money, energy, etc., which are controlled by the company. Let us interpret x_i as the action of i-th unit, X_i as the set of actions allowed for the unit by its own technological constraints. Let, further, $g_{ji}(x_i)$ be the amount of resource j which is required by unit i to implement action x_i, and let $-f_i(x_i)$ be the profit the unit will get carrying out this action. The company is interested to maximize the total profit of the units

$$-f(x) \equiv \sum_{i=1}^{l} (-f_i(x_i))$$

under the constraints

$$\sum_{i=1}^{l} g_{ji}(x_i) \leq b_j,\ j = 1, ..., m,$$

on the total amount of the resources consumed by the units, and, of course, the constraints

$$x_i \in X_i,\ i = 1, ..., l.$$

This is certain optimization problem, and a straightforward way to solve it would be: to "lift" to the level of the company all the data (sets X_i, functions g_{ji}, f_i), to solve at this level the resulting problem and to enforce somehow the units to carry out the resulting directives x_i^*. This completely centralized approach is very unreliable, due to the necessity in sending back and forth huge amounts of information, and is very costly computationally. A good alternative could be *decentralized* approach, where the company declares some "internal prices" on the common resources and says to the units:
"I can sell to you as many resources as you wish at such and such prices. Please order the resources and say what will be your profit".

The units will try to maximize, given prices of the resources, their profit

$$-f_i(x_i) - \sum_{j=1}^{k} \mu_j g_{ji}(x_i)$$

over $x_i \in X_i$, i.e., minimize their losses $L_i(x_i, \mu)$; their orders will allow the company to compute the objective of the dual problem (CnvDS), which for the case of nonzero right hand sides in the resource inequality constraints clearly is

$$\phi^*(\mu) = \sum_{j=1}^{k} L_i(x_i^*(\mu), \mu) - \sum_{j=1}^{m} \mu_j b_j.$$

In fact the vectors of the resources ordered by the units allow the company to compute the gradient of the dual objective at the current μ (or a supergradient, if the dual objective is nonsmooth). With this information, the company can update the prices on the resources, applying certain method for concave minimization of the dual objective on the nonnegative orthant, ask the units to update accordingly their desicions, and to repeat this procedure until the dual objective will be maximized, thus solving – in parallel and distributed fashion – the problem of maximizing total profit under given constraints on the common resources. This is exactly what is Decomposition does.

It is time now to explain why in the Decomposition scheme we deal with the classical Lagrange function, not an augmented one. The reason is that the "augmenting" $g_j \mapsto \theta_j(g_j)$ of a separable constraint $g_j(x) \leq 0$ destroys separability. The price for using the classical Lagrangian is typical nonsmoothness of the dual objective, and in fact Decomposition is the main source of *nonsmooth* convex optimization programs.

Assignment # 11 (Lecture 11)

Obligatory problems

Exercise 11.4.1 *Prove Proposition 11.3.1.*

Exercise 11.4.2 *Consider the following simple separable optimization problem:*

$$f(x) = \sum_{i=1}^{n} f_i(x_i) \to \min \mid \; g(x) = \sum_{i=1}^{n} g_i(x_i) \leq 0, \; x_i \in X_i = [a_i, b_i], \; i = 1, ..., n.$$

Here x_i are real variables, f_i and g_i are smooth convex functions, assumed, for the sake of simplicity, strongly convex, $-\infty < a_i < b_i < +\infty$ for all i.

Assume also that the Slater condition holds: $g(x) < 0$ for some feasible x.

1) Write down optimality condition for the problem based on Theorem (*), Section 11.4.2

2) Explain how would you solve the problem via the Decomposition scheme

Optional problems

Exercise 11.4.3 *Prove that the statement mentioned in Section 11.4.1:*
If the sum $r(x)$ of the objective and the constraints in (CnvP) is strongly convex (i.e., with positive definite at every point Hessian) and grows at infinity faster than $|x|$, then the auxiliary primal problem

$$\min_x \{f(x) + \sum_{j=1}^{k} \mu_j g_j(x)\}$$

has, for every $\mu > 0$, exactly one solution, and this solution is continuous function of $\mu > 0$.

Prove that if the objective itself is strongly convex and grows at infinity faster than $|x|$, then "$\mu > 0$" in the statement can be replaced with "$\mu \geq 0$".

Lecture 12

Sequential Quadratic Programming

This last lecture of the course is devoted to the *Sequential Quadratic Programming* methods – the methods which are now thought to be the most efficient for practical solution of general-type (i.e., nonconvex) constrained optimization problems.

The idea underlying the methods is to solve directly the KKT system of the problem under consideration by a Newton-type iterative process. In order to get the idea, let us start with the equality constrained case.

12.1 SQP methods: Equality Constrained case

Consider an equality constrained optimization problem

$$\text{(ECP)} \quad f(x) \to \min \mid h(x) = \begin{pmatrix} h_1(x) \\ ... \\ h_m(x) \end{pmatrix} = 0 \quad [x \in \mathbf{R}^n]$$

and let us write down the corresponding Karush-Kuhn-Tucker system:

$$\begin{array}{ll} \text{(KKT)} & \\ \nabla_x L(x,\lambda) & \equiv \quad \nabla f(x) + [\nabla h(x)]^T \lambda = 0 \ , \\ \nabla_\lambda L(x,\lambda) & \equiv \quad h(x) = 0, \end{array}$$

where

$$L(x,\lambda) = f(x) + \lambda^T h(x) \equiv f(x) + \sum_{i=1}^{m} \lambda_i h_i(x)$$

is the Lagrangian of (ECP). As we remember from Lecture 8, any locally optimal solution x^* of (ECP) which is a regular point of the system of constraints (i.e., is such that the gradients of constraints taken at this point are linearly independent) can be extended by properly chosen $\lambda = \lambda^*$ to a solution of (KKT).

Now, (KKT) is a system of $n + m$ equations with $n + m$ unknowns x, λ, and we can try to solve this system with the Newton method. To the moment we know what is the Newton *minimization* method, not the Newton method for solving systems of equations; let us look what is this latter routine.

12.1.1 Newton method for systems of equations

The method

Given a system of N nonlinear scalar equations with N unknowns

$$\text{(E)} \qquad P(u) \equiv \begin{pmatrix} p_1(u) \\ ... \\ p_N(u) \end{pmatrix} = 0,$$

with continuously differentiable real-valued functions p_i, one can try to solve this system as follows. Given current approximate solution $\bar{u}$, let us linearize the system at this point, i.e., replace the actual nonlinear system by the linear one

$$[P(u) \approx] \quad P(\bar{u}) + P'(\bar{u})(u - \bar{u}) \equiv \begin{pmatrix} p_1(\bar{u}) + [\nabla p_1(\bar{u})]^T(u - \bar{u}) \\ ... \\ p_N(\bar{u}) + [\nabla p_N(\bar{u})]^T(u - \bar{u}) \end{pmatrix} = 0.$$

Assuming the $N \times N$ matrix $P'(\bar{u})$ nonsingular, we can write down the solution to the linearized system:

$$\bar{u}^+ = \bar{u} - [P'(\bar{u})]^{-1} P(\bar{u}).$$

The vector

$$\bar{u}^+ - \bar{u} \equiv -[P'(\bar{u})]^{-1} P(\bar{u})$$

is called the *Newton displacement*.

Now we can look at $\bar{u}^+$ as at our new approximate solution and iterate the indicated updating, thus coming to the *Newton method for solving (E)* – to the recurrence

$$u_t = u_{t-1} - [P'(u_{t-1})]^{-1} P(u_{t-1}). \tag{12.1.1}$$

Note that the basic Newton method for unconstrained minimization of a smooth function f (Lecture 4) is nothing but the above recurrence applied to the Fermat equation[1)]

$$\text{(F)} \qquad P(x) \equiv \nabla f(x) = 0.$$

Local quadratic convergence

Same as in the particular case of equation (F), in the case of general equation (E) the Newton method possesses fast local convergence:

Proposition 12.1.1 [Local superlinear/quadratic convergence of the Newton method for solving systems of equations]
Let u^* be a solution to (E)*, and assume that this solution is nondegenerate, i.e., that the matrix $P'(u^*)$ is nonsingular. Then the Newton method (12.1.1) locally superlinearly converges to u^*: there exists a neighbourhood U of u^* such that the recurrence (12.1.1), being started at a point $u_0 \in U$, is well defined (i.e., the required inverse matrices do exist), keeps the iterates in U and ensures that*

$$|u_t - u^*| \leq C_\kappa \kappa^t, \ t = 0, 1, ...$$

[1)] to save words, I shall use the word "equation" also for systems of scalar equations, interpreting such a system as a single vector equation. It always will be clear which equation – a vector or a scalar one – is meant.

for every $\kappa > 0$.

If, in addition, P *is twice continuously differentiable in a neighbourhood of* u^**, then the indicated convergence is quadratic:*

$$|u_t - u^*| \le C|u_{t-1} - u^*|^2$$

for all t *and some* $C < \infty$.

The proof of this proposition repeats word by word the one of the similar statement from Lecture 4, and you are welcome to prove the statement by yourself.

Now let us look how all this works when (E) is the Karush-Kuhn-Tucker system of the equality constrained problem (ECP).

12.1.2 Solving (KKT) by the Newton method

When applying the Newton method to (KKT), we should answer the following crucial question:

- When a KKT point (x^*, λ^*) of problem (ECP) is a nondegenerate solution to the system (KKT)?

  [when it is so, we may hope to get at least *locally* converging routine; on the other hand, when the corresponding matrix P' is singular at (x^*, λ^*), we have small hope on convergence, not speaking of the unpleasant possibility to meet with singular P' also at a point close to (x^*, λ^*): if this latter point will be our iterate, how could we define the Newton displacement?]

Our local goal is to answer the above question and to present important interpretation of the Newton displacement in terms of certain auxiliary quadratic optimization problem with linear equality constraints. This interpretation will allow us to extend the approach from the case of equality constrained problem (ECP) to the general constrained case.

To get convenient notation, we set

$$P(x,\lambda) = \nabla_{x,\lambda} L(x,\lambda) = \begin{pmatrix} \nabla_x L(x,\lambda) \equiv \nabla f(x) + [\nabla h(x)]^T \lambda \\ \nabla_\lambda L(x,\lambda) \equiv h(x) \end{pmatrix}; \tag{12.1.2}$$

in this notation, system (KKT) is exactly (E). Note that

$$P'(x,\lambda) = \begin{pmatrix} \nabla_x^2 L(x,\lambda) & [\nabla h(x)]^T \\ \nabla h(x) & 0 \end{pmatrix} \tag{12.1.3}$$

Nonsingularity of KKT points

We start with the following important observation:

Proposition 12.1.2 *Let* x^* *be a nondegenerate solution to (ECP)* (Definition 8.2.2)*, and let* λ^* *be the corresponding vector of Lagrange multipliers. Then the matrix* $P'(x^*, \lambda^*)$ *is nonsingular.*

Proof. Denoting, as usual, $H = \nabla_x^2 L(x^*, \lambda^*)$, $A = \nabla h(x^*)$, we have

$$P' = \begin{pmatrix} H & A^T \\ A & 0 \end{pmatrix};$$

we should prove that $P'h = 0$, h being $(n+m)$-dimensional vector, implies $h = 0$. Partitioning h into n-dimensional fragment v and m-dimensional fragment w, we get

$$0 = P' \begin{pmatrix} v \\ w \end{pmatrix} \equiv \begin{pmatrix} Hv + A^T w \\ Av \end{pmatrix},$$

whence

$$Hv + A^T w = 0; \quad Av = 0.$$

The second equation says that v is in the kernel of $A = \nabla h(x^*)$, i.e., is in the plane T tangent to the feasible surface at the point x^*. Multiplying the first equation from the left by v^T and taking into account that $v^T A^T w = (Av)^T w = 0$ by the second equation, we get $v^T H v = 0$. Since x^* is a nondegenerate local solution to (ECP), $H = \nabla_x^2 L(x^*, \lambda^*)$ is positive definite on T; consequently, $v \in T$ and $v^T H v = 0$ imply that $v = 0$.

Since $v = 0$, the first of our two equations implies that $A^T w = 0$; but x^* is regular for the constraints of (ECP), i.e., the columns of the matrix $A^T = [\nabla h(x^*)]^T$ – these columns are exactly the gradients of the constraints at the point x^* – are linearly independent; since the columns of A^T are linearly independent, their combination $A^T w$ can be zero only if the vector w of coefficients in the combination is zero. Thus, both v and w are zero. ∎

Structure and interpretation of the Newton displacement

Let us look what is the Newton system

$$P'(u)d = -P(u)$$

– the system defining the Newton displacement – for the particular case when the system (E) of nonlinear equations is the Karush-Kuhn-Tucker system (KKT) of (ECP). In the case in question P is given by (12.1.2), while P' is given by (12.1.3). Partitioning the vector of unknowns d of the Newton system into x-part dx and λ-part $d\lambda$, we can rewrite the Newton system as

$$\begin{array}{lcl} [\nabla_x^2 L(\bar{x}, \bar{\lambda})]dx + [\nabla h(\bar{x})]^T d\lambda & = & -\nabla f(\bar{x}) - [\nabla h(\bar{x})]^T \lambda \\ [\nabla h(\bar{x})]d\lambda & = & -h(\bar{x}) \end{array},$$

$(\bar{x}, \bar{\lambda})$ being the current iterate to be updated by the Newton step.

It is convenient to pass from the unknowns dx and $d\lambda$ to dx and $\lambda^+ = \bar{\lambda} + d\lambda$ (note that λ^+ is the λ-component of the Newton iterate of $(\bar{x}, \bar{\lambda})$). The new unknowns are given by the system

$$\begin{array}{lcl} [\nabla_x^2 L(\bar{x}, \bar{\lambda})]dx + [\nabla h(\bar{x})]^T \lambda^+ & = & -\nabla f(\bar{x}) \\ [\nabla h(\bar{x})]dx & = & -h(\bar{x}) \end{array} \tag{12.1.4}$$

The resulting form of the Newton system is very instructive. To see this, let us for the time being forget about the Newton method for solving the KKT system of problem (ECP) and look at the situation from a little bit different viewpoint. Let x^* be a nondegenerate solution to (ECP) and λ^* be the corresponding vector of Lagrange multipliers. Then

$$\nabla_x L(x^*, \lambda^*) = 0$$

and the Hessian $\nabla_x^2 L(x^*, \lambda^*)$ of the Lagrange function is positive definite *on the plane T tangent to the feasible surface of (ECP) at the point x^**. In other words, the Lagrange function regarded

as function of x (with λ being set to λ^*) attains at $x = x^*$ its nondegenerate local minimum *on the plane* T. If we knew the tangent plane T (and λ^*) in advance, we could find x^* by applying to $L(x, \lambda^*)$ any method of unconstrained minimization, not in the entire space, of course, but in the plane T. Let us look what would happen if we would use in this scheme the Newton minimization method. How we should act? We should replace our current objective – the function $L(\cdot, \lambda^*)$ – by its quadratic expansion at the current iterate $\bar{x}$, i.e., by the function

$$L(\bar{x}, \lambda^*) + (dx)^T \nabla_x L(\bar{x}, \lambda^*) + \frac{1}{2}(dx)^T \nabla_x^2 L(\bar{x}, \lambda^*) dx \tag{12.1.5}$$

and to minimize this quadratic approximation over displacements dx such that $\bar{x} + dx \in T$; the new iterate would be then $\bar{x} + dx$, where dx is given by the aforementioned minimization.

Now, actually we do not know T and λ^*; all we have is the current approximation $(\bar{x}, \bar{\lambda})$ of (x^*, λ^*). With this approximation, we can imitate the latter construction as follows:

- use, instead of λ^*, its current approximate $\bar{\lambda}$;
- use, instead of T, the plane $\bar{T}$ given by *linearization* of the constraints $h(x) = 0$ at the current iterate $\bar{x}$:
$$\bar{T} = \{y = \bar{x} + dx \mid [\nabla h(\bar{x})]dx + h(\bar{x}) = 0\}.$$

Thus, the above idea – to improve current iterate $\bar{x}$ by applying a single step of the Newton minimization of the Lagrange function over the plane T – could be implemented as follows: we define dx as the solution to the quadratic minimization problem with linear equality constraints:

$$\begin{array}{ll} (*) & \\ & L(\bar{x}, \bar{\lambda}) + (dx)^T \nabla_x L(\bar{x}, \bar{\lambda}) + \frac{1}{2}(dx)^T \nabla_x^2 L(\bar{x}, \bar{\lambda}) dx \quad \rightarrow \quad \min \\ \text{s.t.} & \\ & [\nabla h(\bar{x})]dx \quad = \quad -h(\bar{x}) \end{array}$$

and define the new x-iterate as

$$x^+ = \bar{x} + dx.$$

Note that the problem can be a little bit simplified: recalling the definition of the Lagrange function, we see that the linear part

$$L(\bar{x}, \bar{\lambda}) + (dx)^T \nabla_x L(\bar{x}, \bar{\lambda})$$

of the objective in the problem is equal to

$$f(\bar{x}) + (dx)^T \nabla f(\bar{x}) + \left[\bar{\lambda}^T h(\bar{x}) + \bar{\lambda}^T [\nabla h(\bar{x})]dx\right];$$

the quantity in the brackets is constant on the feasible plane of the problem and therefore can be eliminated from the objective without varying the solution. Thus, problem (*) is equivalent to

$$\begin{array}{ll} (\mathrm{QP}_{(\bar{x},\bar{\lambda})}) & \\ & f(\bar{x}) + (dx)^T \nabla f(\bar{x}) + + \frac{1}{2}(dx)^T \nabla_x^2 L(\bar{x}, \bar{\lambda}) dx \quad \rightarrow \quad \min \\ \text{s.t.} & \\ & [\nabla h(\bar{x})]dx \quad = \quad -h(\bar{x}). \end{array}$$

Problem $(\mathrm{QP}_{(\bar{x},\bar{\lambda})})$ is quite respectable optimization problem, and its solution, if exists, is achieved at a KKT point. Moreover, the solution to the problem does exist, if $(\bar{x}, \bar{\lambda})$ is close

enough to (x^*, λ^*). Indeed, if $\bar{x} = x^*, \bar{\lambda} = \lambda^*$, then the objective of the problem is a quadratic form which is strongly convex at the feasible plane T of the problem (recall that x^* is a nondegenerate solution to (ECP)). By continuity reasons, for $(\bar{x}, \bar{\lambda})$ close enough to (x^*, λ^*) the objective in $(\mathrm{QP}_{(\bar{x},\bar{\lambda})})$ also is strongly convex quadratic form on the feasible plane $\bar{T}$ of the problem and therefore attains its minimum on the plane.

Now let us make the following crucial observation (for the sake of brevity, I write (QP) instead of $(\mathrm{QP}_{(\bar{x},\bar{\lambda})})$):

> *Let the Newton system (12.1.4) be a system with nonsingular matrix (we know that it indeed is the case when $(\bar{x}, \bar{\lambda})$ is close enough to (x^*, λ^*)). Then the Newton displacement dx given by the Newton system is the unique KKT point of the problem (QP), and the vector λ^+ given by the Newton system is the vector of Lagrange multipliers associated with the KKT point dx in the problem (QP).*

Indeed, let z be a KKT point of (QP) and μ be the corresponding vector of Lagrange multipliers. The KKT equations for (QP) are (check it!)

$$\begin{aligned} \nabla f(\bar{x}) + \nabla_x^2 L(\bar{x}, \bar{\lambda}) z + [\nabla h(\bar{x})]^T \mu &= 0, \\ [\nabla h(\bar{x})] z &= -h(\bar{x}); \end{aligned}$$

but this is exactly the Newton system (12.1.4).

Now we understand what the Newton method from the previous section does in the optimization terms. The optimal solution is the minimizer of the Lagrange function on the tangent plane T to the feasible surface of the problem, the Lagrange multipliers being set to their optimal values. The method, at each step, replaces this the tangent plane with the plane $\bar{T}$ given by linearizations of the constraints at the current iterate, replaces "exact" multipliers in the Lagrange function with their current estimate and performs a single Newton step of minimization of the resulting function over $x \in \bar{T}$, thus generating the new x-iterate. As a byproduct of this Newton minimization step (this step solves quadratic minimization problem (QP) with linear equality constraints) we get the vector of Lagrange multipliers for (QP), and this vector is our new estimate of the vector λ^* of Lagrange multipliers of the original problem.

The presented interpretation of the method explains why it is called "Sequential Quadratic Programming"; the name comes from the fact that at each step of the method we in fact solve certain Quadratic Programming problem with linear equality constraints. There is another name of the method – the *Projected Lagrangian Scheme*, and this second name reflects the origin of our Quadratic Programming problems.

What should be stressed is that the linear term in the objective of (QP) comes from our original objective f [2)]; in contrast to this, the *quadratic* term in the objective of (QP) comes from the Lagrange function, not from f. This is quite reasonable: this is the Lagrange function which attains its nondegenerate local minimum on T at x^*, not the actual objective f. For f itself, x^* is nothing but a critical point on T, and it may happen to be the point of maximum, as it is seen from the following example:

$$f(x_1, x_2) = x_2 - 0.1x_1^2, \;\; h(x) = x_2 - x_1^2, \;\; x^* = (0, 0).$$

[2] note, anyhow, that we may deal with the linear term coming from the Lagrange function as well: recall that (QP) is equivalent to (*). The only advantage of (QP) is that the vector of Lagrange multipliers in this problem is exactly the new estimate λ^+ of λ^*, while the vector of Lagrange multipliers in (*) is $\lambda^+ - \bar{\lambda}$

In this example, x^* indeed is a nondegenerate local solution with $\lambda^* = -1$ and $T = \{x \in \mathbf{R}^2 \mid x_2 = 0\}$. The Lagrange function with "correct" value of the Lagrange multiplier is $L(x, \lambda^*) = 0.9x_1^2$, and x^* is its nondegenerate local (and even global) minimizer on T. In contrast to this, x^* is nondegenerate global maximizer of the objective f on T.

12.2 The case of general constrained problems

12.2.1 Basic SQP scheme

Given the above interpretation of the Newton method for solving the KKT system of an equality constrained problem, we now can easily understand how the method should be extended onto the case of a general constrained problem

$$\text{(GCP)} \quad f(x) \to \min \mid h(x) = \begin{pmatrix} h_1(x) \\ ... \\ h_m(x) \end{pmatrix} = 0,\ g(x) = \begin{pmatrix} g_i(x) \\ ... \\ g_k(x) \end{pmatrix} \leq 0.$$

Our starting observation is that if x^* is a nondegenerate solution to (GCP) and (λ^*, μ^*) is the corresponding vector of Lagrange multipliers (λ's are the multipliers for the equality constraints, and μ's - for the inequality ones), then, as it is immediately given by straightforward verification of Definition 8.2.4, x^* is a nondegenerate solution, and λ^*, μ^* are Lagrange multipliers, in the following linearly constrained quadratic programming program:

$$\begin{array}{ll} (\text{QP}^*) & \\ & f(x^*) + (x - x^*)^T \nabla f(x^*) + \frac{1}{2}(x - x^*)^T \nabla_x^2 L(x^*; \lambda^*, \mu^*)(x - x^*) \quad \to \quad \min \\ \text{s.t.} & \\ & \begin{array}{rcl} h_i(x^*) + (x - x^*)^T \nabla h_i(x^*) & = & 0,\ i = 1, ..., m, \\ g_j(x^*) + (x - x^*)^T \nabla g_j(x^*) & \leq & 0,\ j = 1, ..., k, \end{array} \end{array}$$

where

$$L(x; \lambda, \mu) = f(x) + \sum_{i=1}^{m} \lambda_i h_i(x) + \sum_{j=1}^{k} \mu_j g_j(x)$$

is the Lagrange function of (GCP).

This observation motivates the following iterative process: given current approximation (x_t, λ_t, μ_t) to (x^*, λ^*, μ^*), we use this approximation to model (QP*) by the linearly constrained quadratic program

$$\begin{array}{ll} (\text{QP}_t) & \\ & f(x_t) + (dx)^T \nabla f(x_t) + \frac{1}{2}(dx)^T [\nabla_x^2 L(x_t; \lambda_t, \mu_t)] dx \quad \to \quad \min \\ \text{s.t.} & \\ & \begin{array}{rcl} [\nabla h(x_t)] dx & = & -h(x_t), \\ [\nabla g(x_t)] dx & \leq & -g(x_t). \end{array} \end{array}$$

Solving this problem, we get the corresponding displacement dx and, besides this, the vectors λ^+ and μ^+ of the Lagrange multipliers for (QP$_t$):

$$\begin{array}{rcl} \nabla f(x_t) + \nabla_x^2 L(x_t; \lambda_t, \mu_t) dx + \sum_{i=1}^m \lambda_i^+ \nabla h_i(x_t) + \sum_{j=1}^k \mu_j^+ \nabla g_j(x_t) & = & 0, \\ \mu_j^+ [g_j(x_t) + (dx)^T \nabla g_j(x_t)] & = & 0,\ j = 1, ..., k, \\ \mu_j^+ & \geq & 0,\ j = 1, ..., k; \end{array}$$

after dx, λ^+, μ^+ are identified, we take as the new iterate the triple

$$x_{t+1} = x_t + dx, \;\; \lambda_{t+1} = \lambda^+, \;\; \mu_{t+1} = \mu^+$$

and loop.

It can be proved that if x^* is a nondegenerate solution to (GCP) and the outlined recurrence is started close enough to (x^*, λ^*, μ^*), then the recurrence is well-defined and converges quadratically to (x^*, λ^*, μ^*).

12.2.2 Quasi-Newton Approximations

A disadvantage of the SQP scheme as developed so far is that the auxiliary quadratic problem (QP_t) may be "bad" – e.g., the quadratic objective in this problem may be below unbounded at the feasible setof (QP_t). This bad thing for sure does not happen in the *equality* constrained case, provided that x_t, λ_t) is close enough to (x^*, λ^*), since then the matrix $\nabla^2 L_x(x_t, \lambda_t)$ is positive definite along the approximate tangent plane $\bar{T}$; this favourable situation, however, is ensured only in a neighbourhood of the solution. And in the case of general constraints problem (QP_t) can be bad independently of whether the current iterates are or are not close to $(x^*\lambda^*, \mu^*)$. The situation is very similar to the one we met in the Newton method for unconstrained minimization, where the Hessian of the objective was not oblidged to be positive definite at every step, and it caused a lot of troubles.

To overcome the difficulty, in the unconstrained case we simply replaced the current Hessian with its positive definite correction whenever the Hessian was not "well positive definite". The same remedy can be used now: if the objective in (QP_t) turns out to be not strongly convex on the entire space, i.e., if the matrix $\nabla_x^2 L(x_t; \lambda_t, \mu_t)$ is not positive definite, we replace it with its positive definite correction B_t. With this correction, the auxiliary quadratic programming problem which we should solve at a step t of the SQP method becomes

$$\begin{array}{ll} (\mathrm{QP}'_t) & \\ & f(x_t) + (dx)^T \nabla f(x_t) + \frac{1}{2}(dx)^T B_t dx \;\; \rightarrow \;\; \min \\ \text{s.t.} & \\ & [\nabla h(x_t)]dx \;\; = \;\; -h(x_t), \\ & [\nabla g(x_t)]dx \;\; \leq \;\; -g(x_t); \end{array}$$

the x-component of the new iterate is $x_{t+1} = x_t + dx$, while λ_{t+1} and μ_{t+1} are the vectors of Lagrange multipliers of (QP'_t).

The indicated modification (which, as we shall see in a while, still is converging) possesses the following attractive properties:

- problem (QP'_t) does not depend explicitly on the current approximates λ_t and μ_t to the Lagrange multipliers of (GCP) (all the dependence of λ_t, μ_t, if any, is via the choice of the matrix B_t);
- it ensures *global* convergence of a reasonable linesearch version of the SQP scheme (see Section 12.3);
- (QP'_t) becomes a problem with strongly convex quadratic objective and linear constraints; the problem therefore can be efficiently solved in finitely many steps by the active set method (Section 9.4, Lecture 9).

It should be also mentioned that if, as it is the case under reasonable regularity assumptions, the Lagrange multipliers μ_{t+1} for inequality constraints in the problem (QP'_t) converge to the actual Lagrange multipliers μ^* for the inequality constraints in (GCP), then for large t the vector μ_t is close to μ_{t+1}. It means that μ_t *allows to predict which inequalities in* (QP'_t) *are active at the solution to this problem*: for large t, these active inequalities are exactly those which correspond to positive entries of μ_t. It follows that when solving (QP'_t) by the active set method, it makes sense to use this prediction as the initial guess for the active set of (QP'_t). With this policy, eventually (for large enough values of t) (QP'_t) will be solved in *one* step of the active set method (i.e., at the cost of solving a single system of linear equations)! And even for initial values of t, when it may take several steps of the active set method to solve (QP'_t), this prediction used as the initial guess for the actual active set of (QP'_t) typically saves a lot of computations.

- using matrix B_t instead of the Hessain of the Lagrangian, we may apply the Quasi-Newton tactics to approximate this Hessian (more exactly, its positive definite correction) *without explicit usage of the second-order information on the data*

12.3 Linesearch, Merit functions, global convergence

The Sequential Quadratic Programming scheme studied so far possesses nice *local* convergence properties, at least for the case when $B_t = \nabla_x^2 L(x_t; \lambda_t, \mu_t)$: if x^* is a nondegenerate local solution to (GCP) with Lagrange multipliers λ^*, μ^*, and if SQP is initialized by a triple (x_1, λ_1, μ_1) close enough to (x^*, λ^*, μ^*), then the routine quadratically converges to the solution of (GCP). The difficulty, anyhow, is that we have no idea whether the routine possesses *global* convergence, or, better to say, we know for sure that it may fail to converge globally. Indeed, in the simplest case where there is no constraints at all, our routine becomes the Basic Newton method for unconstained minimization from Lecture 4, and we know from this Lecture that the Newton method, even applied to a fine convex objective, may diverge, if the starting point is too far from the optimal solution.

What we did in the unconstrained case in order to overcome this drawback? The remedy was *linesearch*. More specifically, what we did in our now situation would look as follows:

given current iterate $u_t = (x_t, \lambda_t, \mu_t)$ and computing dx, λ^+, μ^+ as a solution to (QP'_t) and the Lagrange multipliers associated with this solution, we treat the triple

$$d_{t+1} = (dx, d\lambda = \lambda^+ - \lambda_t, d\mu = \mu^+ - \mu_t) \tag{12.3.1}$$

not as the actual step we should perform from u_t to get the new iterate u_{t+1}, but as a *search direction* only; the new iterate u_{t+1} is obtained from u_t by certain – not necessarily unit – stepsize γ_{t+1} from u_t in the direction d_t:

$$u_{t+1} = u_t + \gamma_{t+1} d_{t+1}, \tag{12.3.2}$$

and the role of linesearch is to choose a reasonable stepsize γ_{t+1}.

In the case of unconstrained optimization we measured the quality of a candidate stepsize by the progress in the objective, so that the linesearch was aimed to minimize the objective along the search ray $\{u_t + \gamma d_{t+1} \mid \gamma \geq 0\}$. In our now situation, it makes no sense to look at the

objective alone: what we are interested in, is "movement towards the KKT set of (GCP)", and the description of this set involves both the objective and the constraints.

Thus, what we need is certain "auxiliary objective", certain integrated measure of violation of the relations we would like to ensure (in our case these are the KKT conditions). Choosing somehow this measure, we could use the associated with this measure linesearch in order to choose stepsizes which indeed decrease violations of our our target relations and thus move us towards the solution set.

The general scheme we come to – some policy of choosing sequential search directions plus tactics of choosing stepsizes in these directions, tactics based on minimization of a reasonable auxiliary objective over the search rays – is typical for many iterative methods, and the auxiliary objective in these methods has a special name - it is called the *merit function*. Good choice of a merit function is as important for quality of the overall routine as a good policy of choosing search directions, and in fact both these components of a method should fit each other. The crucial requirement which should be satisfied by these components is as follows:

(!) *if current iterate u is not a solution to the problem in question, then the search direction d associated with u should be descent for the merit function $M(\cdot)$:*

$$M(u+\gamma d) < M(u) \quad \text{for all small enough positive } \gamma.$$

The role of property (!) is clear: if it is violated, then we may arrive at an iterate which is *not* a solution to the problem we want to solve and at the same time is such that *the merit function increases along the search ray associated with the iterate*. In this case the linesearch will return zero stepsize, and the next iterate will be the same as the previous one, so that we shall stay at this "bad" iterate forever. On the other hand, if (!) is satisfied, then the linesearch will ensure nonzero progress in the value of the merit function at each step where the current iterate is not an exact solution, and the method will become *descent* with respect to the merit function. In this case we have good chances to use the Global Convergence Theorem from Lecture 1 to prove convergence of the method.

Now let us apply the outlined general considerations to the case we are interested in, i.e., to the case when the problem of interest is (GCE) and the policy of choosing search directions is given by the Sequential Quadratic Programming scheme. What we need is a merit function which, along with this policy, fits (!).

12.3.1 l_1 merit function

It turns out that an appropriate (and the most common) merit function in our case is

$$M(x) = f(x) + \theta \left[\sum_{i=1}^{m} |h_i(x)| + \sum_{j=1}^{k} g_j^+(x) \right], \tag{12.3.3}$$

where $\theta > 0$ is parameter of the function and, as always,

$$g_j^+(x) = \max[g_j(x), 0]$$

is the positive part of the constraint g_j.

The following lemma justifies the indicated choice of the merit function:

Lemma 12.3.1 *Let x_t be current iterate, let B_t be a positive definite matrix used in (QP'_t), let dx be the solution to the latter problem and $\lambda \equiv \lambda_{t+1}$, $\mu \equiv \mu_{t+1}$ be the Lagrange multipliers associated with (QP'_t). Assume that θ is large enough:*

$$\theta \geq \max\{|\lambda_1|, ..., |\lambda_m|, \mu_1, \mu_2, ..., \mu_k\} \tag{12.3.4}$$

Then either $dx = 0$, and then x_t is a KKT point of (GCP), or $dx \neq 0$, and then ds is a descent direction of the merit function M.

Proof. By the origin of dx, λ and μ we have (values and derivatives of f, h, g are taken at x_t, $B = B_t$):

$$\nabla f + Bdx + [\nabla h]^T \lambda + [\nabla g]^T \mu = 0, \tag{12.3.5}$$

$$h + [\nabla h]dx = 0, \tag{12.3.6}$$

$$g + [\nabla g]dx \leq 0, \tag{12.3.7}$$

and, finally,

$$\mu_j \geq 0, \;\; \mu_j[g_j + (dx)^T \nabla g_j] = 0, \; j = 1, ..., k. \tag{12.3.8}$$

Looking at these relations, we see that if $dx = 0$, then x_t clearly is a KKT point of (GCP). Now let $dx \neq 0$. We have

$$(dx)^T \nabla f =$$

[see (12.3.5)]

$$= -(dx)^T Bdx - \sum_{i=1}^{m} \lambda_i (dx)^T \nabla h_i - \sum_{j=1}^{l} \mu_j (dx)^T \nabla g_j =$$

[see (12.3.7) and (12.3.1)]

$$= -(dx)^T Bdx + \sum_{i=1}^{m} \lambda_i h_i + \sum_{j=1}^{k} \mu_j g_j \leq$$

[note that $\mu_j \geq 0$]

$$\leq -(dx)^T Bdx + \sum_{i=1}^{m} |\lambda_i||h_i| + \sum_{j=1}^{k} \mu_j g_j^+ \leq$$

[see (12.3.4)]

$$\leq -(dx)^T Bdx + \theta \left[\sum_{i=1}^{m} |h_i| + \sum_{j=1}^{k} g_j^+ \right]. \tag{12.3.9}$$

Now let $\gamma > 0$. We have

$$M(x_t) - M(x_t + \gamma dx) = [f(x_t) - f(x_t + \gamma dx)] +$$

$$+\theta \left[\sum_{i=1}^{m} [|h_i(x_t)| - |h(x_t + \gamma dx)|] + \sum_{j=1}^{k} [g_j^+(x_t) - g_j^+(x_t + \gamma dx)] \right]. \tag{12.3.10}$$

Denoting by $O_i(\gamma)$ appropriate functions of γ which tend to 0 as $\gamma \to +0$, we have

$$f(x_t) - f(x_t + \gamma dx) \geq -(dx)^T \nabla f + \gamma O_0(\gamma). \tag{12.3.11}$$

Further,

$$h_i(x_t+\gamma dx) = h_i + \gamma (dx)^T \nabla h_i + \gamma O_i(\gamma) =$$

[see (12.3.6)]

$$= (1-\gamma)h_i + \gamma O_i(\gamma),$$

whence

$$|h_i(x_t+\gamma dx)| \le (1-\gamma)|h_i| + \gamma |O_i(\gamma)|,\ i = 1,...,m. \tag{12.3.12}$$

Similarly,

$$g_j(x_t+\gamma dx) = g_j + \gamma (dx)^T \nabla g_j + \gamma O_{m+j}(\gamma) \le$$

[see (12.3.7)]

$$\le (1-\gamma)g_j + \gamma O_{m+j}(\gamma),$$

whence

$$g_j^+(x_t+\gamma dx) \le (1-\gamma)g_j^+ + \gamma |O_{m+j}(\gamma)|,\ j = 1,...,k. \tag{12.3.13}$$

Substituting (12.3.11) - (12.3.13) into (12.3.10), we get

$$M(x_t) - M(x_t+\gamma dx) \ge \gamma (dx)^T \nabla f + \gamma\theta \left[\sum_{i=1}^m |h_i| + \sum_{j=1}^k g_j^+ \right] + \gamma\theta O(\gamma) \ge$$

[see (12.3.9)]

$$\ge \gamma \left[(dx)^T B_t dx + \theta O(\gamma) \right].$$

Since $dx \neq 0$ and B is positive definite, we have $(dx)^T B_t dx > 0$, and since $O(\gamma) \to 0$ as $\gamma \to +0$, we have also

$$(dx)^T B_t dx + \theta O(\gamma) > 0$$

for all small enough positive γ; from the above computation, for these γ the quantity $M(x_t) - M(x_t+\gamma dx)$ is positive, as required. ∎

12.3.2 SQP Algorithm with Merit Function

Our considerations motivate the following

Algorithm 12.3.1 [Generic SQP Algorithm with l_1 Merit Function for (GCP)]
Initialization. *Choose $\theta_1 > 0$, an arbitrary starting point x_1 and set $t = 1$.*
Step t. *Given current iterate x_t, act as follows:*

- *Choose somehow positive definite matrix B_t;*
- *Form linearly constrained quadratic problem (QP'_t) given by x_t and B_t;*
- *Solve (QP'_t), thus obtaining the solution dx and the vectors of Lagrange multipliers $\lambda = \lambda_{t+1}$ and $\mu = \mu_{t+1}$. If $dx = 0$, terminate: x_t is a KKT point of (GCP).*
- *Check whether*

  $$\theta_t \ge \bar\theta_t \equiv \max\{|\lambda_1|,...,|\lambda_m|,\mu_1,...,\mu_k\}.$$

  if this inequality is satisfied, set $\theta_{t+1} = \theta_t$, otherwise set

  $$\theta_{t+1} = \max[\bar\theta_t, 2\theta_t];$$

- *Find the new iterate*

$$x_{t+1} = x_t + \gamma_{t+1} dx$$

by linesearch aimed to minimize the merit function

$$M_{t+1}(x) = f(x) + \theta_{t+1}\left[\sum_{i=1}^{m} |h_i(x)| + \sum_{j=1}^{k} g_j^+(x)\right]$$

on the search ray $\{x_t + \gamma dx \mid \gamma \geq 0\}$. *Replace* t *with* $t+1$ *and loop.*

We can prove the following Main Theorem:

Theorem 12.3.1 [Global Convergence of the SQP Method with l_1 Merit Function]
Let problem (GCP) be solved by Algorithm 12.3.1, and assume that

- *a) there exists a bounded and closed set* $\Omega \subset \mathbf{R}^n$ *such that for* $x \in \Omega$ *the solution set* $D(x)$ *of the system of linear inequality constraints*

$$S(x): \quad [\nabla h(x)]dx = -h(x), \;\; [\nabla g(x)]dx \leq -g(x)$$

with unknowns dx *is nonempty, and each vector* $dx \in D(x)$ *is a regular point of the system* $S(x)$ *(Definition 8.2.3);*

- *b) the trajectory* $\{x_t\}$ *of the algorithm belongs to* Ω *and is infinite (i.e., the method does not terminate with exact KKT point of (GCE));*

- *c) the matrices* B_t *used in the method are uniformly bounded and uniformly positive definite:*

$$cI \leq B_t \leq CI$$

for all t, *with some* $0 < c \leq C < \infty$.

Then all accumulation points of the trajectory of the method are KKT points of (GCP).

Sketch of the Proof.
1) Let (QP(x, B)) denote the auxiliary linearly constrained quadratic problem of the type (QP$'_t$) associated with an iterate $x_t = x$ and a matrix $B_t = B$, and let $\mathcal{B}$ be the set of positive definite matrices such that

$$cI \leq B \leq CI.$$

From a) and c) one can extract (this is easy, although dull) that for $x \in \Omega$ and $B \in \mathcal{B}$ the problem (QP(x, B)) has a unique solution $dx(x, B)$ with uniquely defined Lagrange multipliers $\lambda(x, B), \mu(x, B)$, and the triple $(dx(x, B), \lambda(x, B), \mu(x, B))$ is continuous in $(x, B) \in \Omega \times \mathcal{B}$.
2) From 1) and b) it follows that the Lagrange multipliers generated in course of running the method are uniformly bounded; from the description of the method it therefore follows that the parameter θ_t of the merit function is updated only finitely many times. Thus, forgetting about certain finite initial fragment of the trajectory, we may assume that this parameter simply is constant: $\theta_t \equiv \theta$.
3) To prove the Theorem, we should prove that if $\bar{x}$ is the limit of certain subsequence $\{x_{t_p}\}_{p=1}^{\infty}$ of our (bounded) trajectory, then $\bar{x}$ is a KKT point of (GCE). From c) it follows that the matrices

B_{t_p} belong to the (clearly compact) set $\mathcal{B}$; passing to a subsequence, we therefore may assume that

$$x_{t_p} \to \bar{x}, \;\; B_{t_p} \to \bar{B} \in \mathcal{B} \;\; \text{as} \;\; p \to \infty.$$

If the solution $d^+ = dx(\bar{x}, \bar{B})$ to problem (QP$(\bar{x}, \bar{B})$) is zero, then $\bar{x}$ is a KKT point of (GCE) by Lemma 12.3.1, as required. Thus, all we need is to lead to a contradiction the assumption that $d^+ \neq 0$. Under this assumption from the convergencies $x_{t_p} \to \bar{x}$, $B_{t_p} \to \bar{B}$, $p \to \infty$, and from 1) it follows that $d_p \equiv dx(x_{t_p}, B_{t_p}) \to d^+ \neq 0$. From the indicated convergencies and the relation $d^+ \neq 0$ by the arguments completely similar to those used in the proof of Lemma 12.3.1 it follows that the progresses $\delta_p \equiv M(x_{t_p}) - M(x_{t_p+1})$ in the values of the merit function at the steps t_p, $p = 1, 2, ...$, are bounded away from zero:

$$\delta_p \geq \delta > 0, \; p = 1, 2, ...$$

This is clearly a contradiction: our process at each step decreases the merit function, and since this function clearly is below bounded on Q, the sum of progresses in its values over all steps is finite, and therefore these progresses go to 0 as $t \to \infty$. ∎

12.3.3 Concluding remarks

There are several essential comments to the presented method.

- In the presentation of the Generic SQP algorithm with l_1 merit function we did not specify how the matrices B_t are generated; Theorem 12.3.1 also is not that specific in this respect (all it asks for is uniform boundedness and uniform positive definiteness of the matrices B_t). Our preceding considerations, however, make it clear that in order to get fast algorithm, one should make B_t "close" to the Hessian of the Lagrange function associated with the current iterate x and the current (coming from the previous auxiliary quadratic programming problem) estimates of the Lagrange multipliers. Of course, if this Hessian, being available, is not "well positive definite", one should replace it by its positive definite correction. And if the second order information is not available at all, we can generate B_t via quasi-Newton approximations similar to those from Lecture 7.

- A weak point of the method in its now form is the possibility for auxiliary problems (QP$'_t$) to be unfeasible. This indeed may happen, even in the case of feasible problem (GCP). More exactly, if x^* is a nondegenerate solution to (GCP), then the quadratic problems associated with the iterates close enough to x^* are feasible, but if the current iterate is too far from x^*, the quadratic problem can be infeasible. What to do, if an unfeasible auxiliary quadratic problem is met?

  A popular way to overcome this difficulty is to pass from the auxiliary problem (QP$'_t$) to the "trust region SQP problem" which gives us both the search direction and the stepsize and *always* is feasible and solvable. Namely, given current iterate x_t and positive definite approximation B_t to the Hessian of the Lagrange function, we find the new iterate

$x_{t+1} = x_t + dx$ by solving the optimization problem as follows:

$$\begin{aligned} & [f(x_t) + (dx)^T \nabla f(x_t) + \frac{1}{2}(dx)^T B_t dx] \\ & +\theta[\sum_{i=1}^{m} |h_i(x_t) + (dx)^T \nabla h_i(x_t)| \\ & + \sum_{j=1}^{k} \max[0, g_j(x_t) + (dx)^T \nabla g_j(x_t)]] \quad \rightarrow \quad \min, \\ \text{s.t.} \quad & -\delta_t \leq (dx)_i \leq \delta_t,\ i = 1, ..., n. \end{aligned}$$

This problem can be easily rewritten as a linearly constrained problem with convex quadratic objective. Solving this problem, we take direct care of decreasing the l_1 Merit Function, or, better to say, its analogy obtained when replacing the actual – nonlinear – constraints in the merit function by their linearizations. The "trust region" bounds $-\delta_t \leq (dx)_i \leq \delta_t$ should be tight enough to make the "analogy" sufficiently close to the actual merit function.

It should also be mentioned that if (GCP) is a (feasible) program with *convex* constraints (i.e., the equality constraints h_i are linear, and the inequality ones g_j are convex), then auxiliary quadratic problems (QP$'_t$) for sure are feasible. Indeed, if $\bar{x}$ is an arbitrary point of $\mathbf{R}^n$ and x is a feasible solution to (GCP), then, from linearity of the equality constraints,

$$h(\bar{x}) + [\nabla h(\bar{x})](x - \bar{x}) = h(x) = 0,$$

and from the Gradient inequality[3)]

$$g(\bar{x}) + [\nabla g(\bar{x})](x - \bar{x}) \leq g(x) \leq 0,$$

so that $x - \bar{x}$ is a feasible solution to the quadratic program (QP$'$) associated with $\bar{x}$.

- "Maratos effect". The Merit Function technique ensures global convergence of the SQP scheme at the cost of "spoiling", to some extent, local quadratic convergence of the "pure" SQP scheme from Sections 12.1.2 and 12.2.1. For the sake of definiteness, let us speak about the case of equality constrained problem (ECP). Moreover, let the matrix $H = \nabla_x^2 L(x^*, \lambda^*)$ be the unit matrix (so that H is positive definite on the entire space). Assume that we use the unit matrix also as B_t at all steps of the SQP method; then, as we know from Section 12.1.2, the unit stepsizes γ_t would result in asymptotical quadratic convergence of the routine, provided that it converges at all. And it is easy to demonstrate that, vise versa, "essentially non-unit" stepsizes will prevent the superlinear convergence. Now, what about "asymptotical compatibility" of the unit stepsizes with the linesearch based on the l_1 Merit Function? Is it true that the linesearch will eventually result in nearly unit stepsizes, or is it at least true that the unit stepsizes do not contradict the Merit Function

[3)] which says that for a differentiable convex function $\phi(\cdot)$ one has

$$\phi(u) \geq \phi(v) + (u - v)^T \nabla \phi(u)$$

for all u, v

philosophy, i.e., that close to the solution they decrease the merit function? The answer on both these question is negative. There are examples of problems (ECP) where, arbitrarily close to a nondegenerate solution to the problem, the unit stepsizes in the SQP scheme *increase* the l_1 Merit Function (and in fact all natural merit functions). This phenomenon (it was discovered by N. Maratos in '78 and is called the *Maratos effect*) demonstrates that we cannot expect the SQP scheme with merit function based linesearch to possess superlinear asymptotic rate of convergence. There are some ways to avoid this drawback, but these details are beyond the scope of the course.