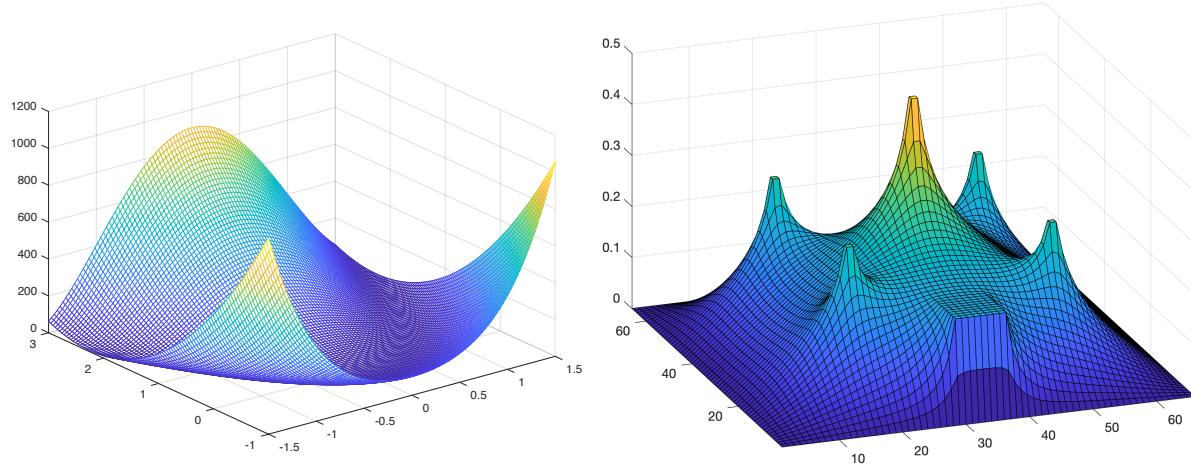

Stefan Funken, Karsten Urban

Numerical optimization (Numerische Optimierung)



LECTURE NOTES, ULM UNIVERSITY, WINTER TERM 2024/2025

Foreword. The present lecture notes are a fusion of the manuscript of several versions of our lectures. It is designed as background material for the lecture and can by no means replace the visit, preparation and work over of the lecture itself. Caused by the different focus of the lecturers, the script is a *superset* of the discussed subjects.

It should be noted that the module description has been changed for the winter term 2019/20. In particular, the topic ‘Numerical computation of eigenvalues and eigenvectors’ has been shifted towards the course ‘Numerical Linear Algebra’. This also caused a restructuring of the present course. Thus, older manuscripts do not necessarily cover all topics,

The reader is encouraged to report bugs, errors and inaccuracies in print, language and also content to us. Please do this by sending an email to {stefan.funken,karsten.urban}@uni-ulm.de.

Copyright. All rights, especially the right to reproduce, spread or translate the script are reserved. It is not allowed to distribute or reproduce the notes using electronic systems or other media without written consent of the authors.

State. Ulm, September 2024.

Contents

1	Nonlinear least squares problems	1
1.1	Some examples	1
1.2	Gauß-Newton Method	5
1.3	Levenberg-Marquardt Method	24
2	Unconstrained optimization	33
2.1	Optimality conditions and convexity	35
2.2	Direct search methods	38
2.3	Gradient descent methods	43
2.4	Trust-Region Methods	58
2.5	Numerical experiments with Matlab-internal routines	61
3	Nonlinear constrained optimization	65
3.1	Optimality conditions	66
3.2	Duality	73
3.3	Slack variables	80
3.4	Penalty method	81
3.5	Penalty-Lagrange Method	87
3.6	Linear-quadratic problems	89
3.7	Nonlinear objective functions	102
4	Interior point methods	109
4.1	Linear-quadratic problems	109
4.2	The barrier method	110
4.3	Solving the primal-dual system	111
4.4	Nonlinear optimization	112
4.5	Linear optimization problems	113
5	Non-smooth Optimization	119
5.1	Examples	119
5.2	Non-smooth, convex problems	120
5.3	The subgradient method	121
5.4	The Cutting plane method	137
5.5	Semismooth Newton methods	141
5.6	The Primal-dual active set strategy	146
Glossary English–Deutsch		153

1 NONLINEAR LEAST SQUARES PROBLEMS

We already discussed the topic of *linear* least squares problems in the lecture *Numerical Linear Algebra*. Recall, that these problems took the following form: For given “measurements” $b \in \mathbb{R}^m$ and a given matrix $A \in \mathbb{R}^{m \times n}$, $m > n$, find $x \in \mathbb{R}^n$ such that

$$f(x) := \|Ax - b\|_2^2 \rightarrow \min! \quad (1.0.1)$$

Doing so, we assume that there is a *linear* dependency between the measurements $b \in \mathbb{R}^m$ and the “parameters” $x \in \mathbb{R}^n$. Moreover, the number of measurements is assumed to be larger than the number of parameters, i.e., we are facing an over-determined system which allows us to “balance” (or optimize) the parameters in a least-squares manner. Hence, a least squares fit is an optimization problem.

It is well-known that such linear problems can be solved (in principle) by the Gaussian¹ normal equations, namely a system of linear equations

$$A^T Ax = A^T b, \quad (1.0.2)$$

or –usually better– by computing the *QR* or singular value decomposition (SVD). It is also known that $f'(x) = \nabla f(x) = 2A^T Ax - A^T b$ and $f''(x) = 2A^T A$. Hence, the first order optimality condition reads $0 = f'(x^*) = 2A^T Ax^* - A^T b$, which corresponds to the normal equations (1.0.2). If A has full rank, then $f''(x) = \nabla^2 f(x)$ is symmetric positive definite so that the first order necessary optimality condition is also sufficient so that $x^* = \arg \min_{x \in \mathbb{R}^n} f(x)$ if and only if x^* solves (1.0.2).

Apparently, we can use numerical *linear* algebra for solving the optimization problem (1.0.1). This is no longer the case, if the parameter-dependency of the measurements is no longer linear. Then, we are facing a **nonlinear** least squares problem of the form

$$f(x) := \|F(x)\|_2^2 \rightarrow \min! \quad \text{for } F : \mathbb{R}^n \rightarrow \mathbb{R}^m, \quad (1.0.3)$$

which we will be solving by methods from numerical optimization. Obviously, $F(x) = Ax - b$ is a special case. However, we note that (1.0.3) is an optimization problem and that the “cost function” f is a quadratic function, which ensures a unique minimum and allows for specific solution algorithms. Apparently, we have

$$f(x) = \|F(x)\|_2^2 \rightarrow \min! \iff g(x) := \frac{1}{2}f(x) = \frac{1}{2}F(x)^T F(x) \rightarrow \min!$$

with a (more or less general) nonlinear (smooth) function $F : \mathbb{R}^n \rightarrow \mathbb{R}^m$.

1.1 SOME EXAMPLES

A typical example of a *linear* least squares problem occurs when using linear regression of a number of measurements, for example a regression line in 2d. The following nonlinear examples are similar in the sense that they are regressions, but the dependency of the parameters in terms of the measurements is no longer linear.

Example 1.1.1 (regression circle) Suppose that we are given scattered data (s_i, t_i) , $i = 1, \dots, m$, and the model assumption that all these points should (at least approximately) be on circle, whose center and radius are unknown. Let us denote the unknown center of the circle by $M = (\mu_s, \mu_t) \in \mathbb{R}^2$ and the radius by $r > 0$. If the data points would be exact, i.e., they would all be on the circle, we would have that

$$\sqrt{(s_i - \mu_s)^2 + (t_i - \mu_t)^2} = r, \quad i = 1, \dots, m.$$

¹Johann Carl Friedrich Gauß (1777-1855)

Due to errors (measurement errors, model errors, etc.) we cannot expect the latter equation to hold exactly, see Figure 1.1. Thus, we are left determining the center M and the radius r of the circle by minimizing the distance of the data to the circle. This is done in a least squares manner by minimizing the sum of the squares of the errors given as $x := (\mu_s, \mu_t, r) \in \mathbb{R}^3$, $F : \mathbb{R}^3 \rightarrow \mathbb{R}^m$, $F(x) := (F_1(x), \dots, F_m(x))^T$ and

$$F_i(x) \equiv F_i(\mu_s, \mu_t, r) := \sqrt{(s_i - \mu_s)^2 + (t_i - \mu_t)^2} - r, \quad i = 1, \dots, m, \quad (1.1.1)$$

which results in the nonlinear least squares problem for finding $x := (\mu_s, \mu_t, r)^T \in \mathbb{R}^3$

$$f(x) = \|F(x)\|_2^2 \rightarrow \min! \Leftrightarrow f(x) := \sum_{i=1}^m \left(\sqrt{(s_i - \mu_s)^2 + (t_i - \mu_t)^2} - r \right)^2 \rightarrow \min! \quad (1.1.2)$$

In this case, there is no linear relation between the parameters $x = (\mu_s, \mu_t, r)^T \in \mathbb{R}^3$ and the data points $(s_i, t_i) \in \mathbb{R}^2$, $i = 1, \dots, m$. \diamond

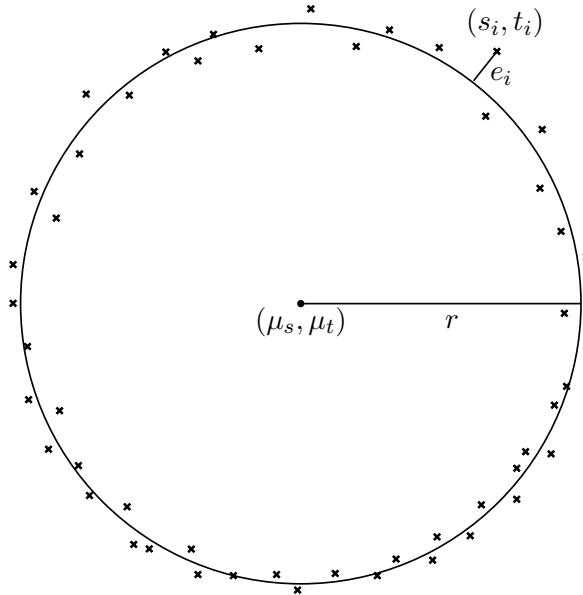


Figure 1.1: Regression of a circle.

Example 1.1.2 (Where have I been?) Assume that you took a picture in the mountains, you are able to identify the summits on a map and now you want to know from where the picture was taken, namely where you have been.² Of course, if you used your smartphone with GPS, most likely, the position where the picture was taken, is stored in the jpeg-format. However, let us assume that you do not know the location and would like to recover it. This problem can be formulated as follows: Determine the location of the photographer given data of the summits on a picture and a map. We are considering in data given in Table 1.1 and Figure 1.2.

The mathematical modeling of this problem is a bit challenging (and interesting as well).

- We are given the cartesian coordinates of the summits, i.e., $P_i \in \mathbb{R}^3$, $i = 1, \dots, 3$ as displayed in Table 1.1 and the right part of Figure 1.1 as well as the locations $M_i \in \mathbb{R}^2$ of the summits on the picture as shown in left part of Figure 1.1.

²The idea for this exercise was given by Ernst Hairer (1949-; father of the Fields medalist Martin Hairer), Gerhard Wanner (1942-) and Stephanie Cirilli from the University of Geneva.

No.	Name of mountain	Height	Position on the map in meters	
			x-coord.	y-coord.
1	Alpspitze	2627,6 m	3700	4125
2	Äußere Hölltalspitze	2720,0 m	2200	3175
3	Vorderer Waxenstein	2136,0 m	2350	6000
4	Mittagscharte	2072,0 m	2175	5885
5	Großer Waxenstein	2276,5 m	1850	5950

Table 1.1: Data for the summits.

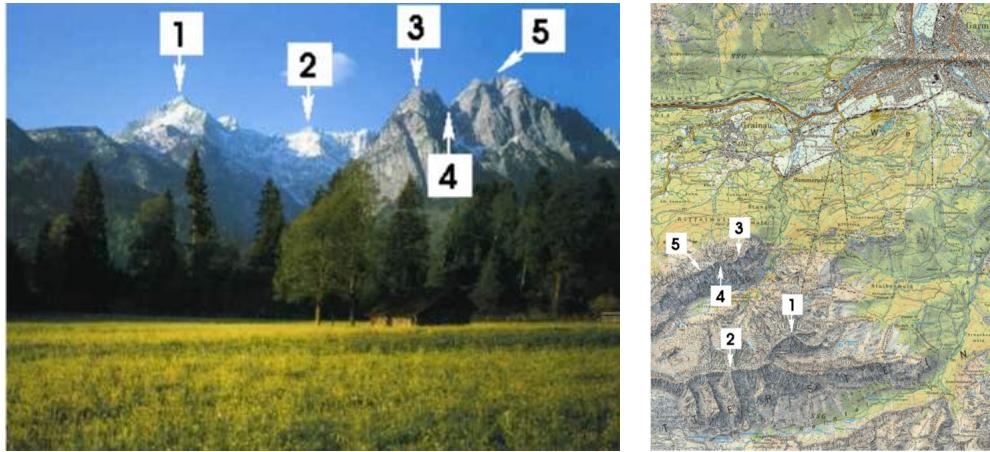


Figure 1.2: Picture (left) and map (right) of the summits.

- First, we need to embed the 2d-picture into the 3d-space. This is done by a rotation angle θ (indicating the orientation of camera) and to view the rotated picture as a plane in the 3-dimensional space, see Figure 1.3. The plane is characterized by a normal vector $d \in \mathbb{R}^3$.
- In order to obtain a coordinate system in which we can express the picture as well as the unknown position $L \in \mathbb{R}^3$ of the lens, we complete d to an orthonormal system (d, e, f) in \mathbb{R}^3 as follows

$$\tilde{e} := \begin{pmatrix} d_2 \\ -d_1 \\ 0 \end{pmatrix}, \quad e = e(d) := \frac{\tilde{e}}{\|\tilde{e}\|}, \quad \tilde{f} := d \times \tilde{e} = \begin{pmatrix} -d_1 d_3 \\ -d_2 d_3 \\ d_1^2 + d_2^2 \end{pmatrix}, \quad f = f(d) := \frac{\tilde{f}}{\|\tilde{f}\|},$$

where $a \times b$ denotes the cross product, i.e., the vector product.

- Next, referring again to Figure 1.3, we determine the location of the summits in the (d, e, f) -coordinates. First, we obtain the location S_i of the summits on the rotated picture by rotating the data M_i by the angle ϑ , i.e.,

$$S_i = S_i(\vartheta) := \begin{pmatrix} \cos \vartheta & -\sin \vartheta \\ \sin \vartheta & \cos \vartheta \end{pmatrix} M_i.$$

The (d, e, f) -coordinates (corresponding to the vector w_i pointing from the summit to the lens L , which is the origin of the (d, e, f) -coordinate system) then read

$$w_i = w_i(d, \vartheta) = (d, e, f) \cdot \begin{pmatrix} 1 \\ S_i(\vartheta) \end{pmatrix} = d + (e(d), f(d)) \cdot S_i(\vartheta).$$

- We compare the vector w_i with the vector $L - P_i$ pointing from the map data P_i of summit i to the lens. If the model is perfect, these two vectors should be collinear, i.e., $w_i \times (L - P_i) = 0$.

To summarize the above considerations, the unknown is given by the position L of the lens, the direction d of the camera and the orientation ϑ of the picture, i.e.,

$$x = (L, d, \vartheta)^T \in \mathbb{R}^7.$$

The function $F : \mathbb{R}^7 \rightarrow \mathbb{R}^{15}$ reads

$$F_i(x) = F_i(L, d, \vartheta) := w_i(L, d, \vartheta) \times (L - P_i) \in \mathbb{R}^3, \quad i = 1, \dots, 5,$$

and we obtain a nonlinear least squares problem of the form (1.0.3). As we see by this example, the mathematical modeling of a given problem might be a challenge as its own. \diamond

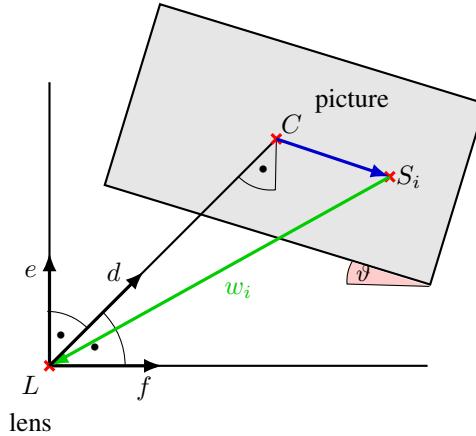


Figure 1.3: Orthogonal system (d, e, f) , rotation angle θ and vector w_i from the lens to the summit S_i .

Example 1.1.3 (Density of the Gamma-distribution) The Gamma-distribution $\Gamma(\lambda, p)$, $\lambda > 0, p > 0$ is well-known from Stochastic. The absolutely continuous distribution is given by its density as

$$\varrho(t; \lambda, p) := \begin{cases} \frac{\lambda^p t^{p-1}}{\Gamma(p)} e^{-\lambda t}, & t \geq 0, \\ 0, & t < 0, \end{cases}$$

where $\Gamma(p)$ denotes the Gamma-function, i.e.,

$$\Gamma(p) := \int_0^\infty t^{p-1} e^{-t} dt, \quad p > 0.$$

Let us assume that we are given certain values ϱ_i of ϱ for some $t_i \geq 0$, $i = 1, \dots, m$, and we assume that these are “good” approximations of the density in the sense that $\varrho_i \approx \varrho(t_i; \lambda, p)$ for suitable (but unknown) parameters $\lambda > 0$ and $p > 0$, see Figure 1.4. In a least squares sense, the unknowns are $x := (\lambda, p) \in \mathbb{R}^2$, $F : \mathbb{R}^2 \rightarrow \mathbb{R}^m$ and

$$F_i(x) = F_i(\lambda, p) := \varrho(t_i; \lambda, p) - \varrho_i, \quad i = 1, \dots, m, \quad (1.1.3)$$

and the nonlinear least squares problem again takes the form (1.0.3). \diamond

Example 1.1.4 (Damped oscillation) This example taken from [5] describes a model for the displacement of a damped oscillation depending on time given by

$$u(t) = u(t; u_0, \delta, \varphi_0, \omega) = u_0 e^{-\delta t} \sin(\omega t + \varphi_0),$$

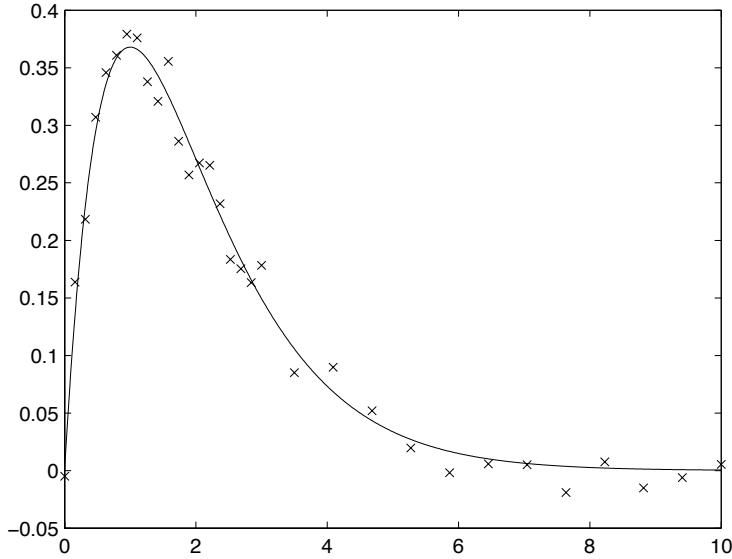


Figure 1.4: Density of the Gamma-distribution for $\lambda = 1, p = 2$ approximated by some scattered data.

with the initial amplitude u_0 at the initial time $t = 0$, the damping rate δ , the initial phase φ_0 and the frequency ω of the oscillation. One can try to find an approximation of the unknown parameters u_0, δ, ω and φ_0 from measurements (t_i, u_i) , $i = 1, \dots, m$, of the displacement. Hence, $x = (u_0, \delta, \omega, \varphi_0) \in \mathbb{R}^4$ is the unknown and the function $F : \mathbb{R}^4 \rightarrow \mathbb{R}^m$ is given by

$$F_i(x) = F_i(u_0, \delta, \omega, \varphi_0) := u(t; u_0, \delta, \varphi_0, \omega) - u_i = u_0 e^{-\delta t_i} \sin(\omega t_i + \varphi_0) - u_i, \quad i = 1, \dots, m,$$

yielding the nonlinear least squares problem again of the form (1.0.3).

A different approach might be to express the displacement $u = u(m, b, D)$ of a string in terms of the mass m of the string, its spring rate D and the friction b in terms of a second order linear ordinary differential equation

$$\ddot{u}(t) + \frac{b}{m} \dot{u}(t) + \frac{D}{m} u(t) = 0, \quad t > 0,$$

with initial values u_0 and v_0 for $u(0)$ and $\dot{u}(0)$, respectively. Hence, one could also use the above differential equation for fitting the parameters to the data. This leads to a field called *PDE-constrained optimization*. \diamond

1.2 GAUSS-NEWTON METHOD

Maybe the most straightforward way to solve a nonlinear least squares problem of the form (1.0.3) consists of a modification of the well-known Newton's method for solving systems of nonlinear equations. We use the following framework: Assume that

$$F := (F_1, \dots, F_m)^T : \Omega \rightarrow \mathbb{R}^m$$

is twice continuously differentiable on some open set $\Omega \subset \mathbb{R}^n$ and $m > n$ (i.e., we only consider the over-determined case), i.e.,

$$F \in C^2(\Omega; \mathbb{R}^m).$$

The least squares framework then amounts solving

$$g(x) := \frac{1}{2} \|F(x)\|_2^2 = \frac{1}{2} F(x)^T F(x) \rightarrow \min!,$$

where we consider the square again for convenience (as in the linear case) and $\|\cdot\|_2$ denotes the Euclidean norm of a vector.³ We are interested in finding local (interior)⁴ minima $x^* \in \Omega$ of g . Thus, we obtain the well-known second order necessary and sufficient optimality conditions

$$\nabla g(x^*) = 0 \quad \text{and} \quad g''(x^*) \text{ is positive definite.} \quad (1.2.1)$$

Let us detail the first order necessary condition. If

$$F'(x) = \nabla F(x) = \left(\frac{\partial}{\partial x_j} F_i(x) \right)_{i=1,\dots,m, j=1,\dots,n} \in \mathbb{R}^{m \times n}$$

denotes the Jacobian matrix of F and $F''_i(x) = \nabla^2 F_i(x)$ the Hessian matrix of the component F_i at the point x , i.e.,

$$F''_i(x) := \left(\frac{\partial^2 F_i(x)}{\partial x_j \partial x_k} \right)_{1 \leq j, k \leq n} \in \mathbb{R}^{n \times n}, \quad i = 1, \dots, m,$$

then we get

$$\nabla g(x) = F'(x)^T F(x), \quad (1.2.2)$$

$$g''(x) = \nabla^2 g(x) = F'(x)^T F'(x) + \sum_{i=1}^m F_i(x) F''_i(x) = F'(x)^T F'(x) + F''(x)^T F(x). \quad (1.2.3)$$

To minimize the function g , we thus have to solve the system of nonlinear equations

$$\nabla g(x) = F'(x)^T F(x) \stackrel{!}{=} 0, \quad (1.2.4)$$

namely, the first order necessary condition to obtain an optimizer x^* . In order to prove that (1.2.4) is also necessary, one would need to show that $\nabla^2 g(x^*)$ is symmetric positive definite. However, due to the specific structure of g , we can also use a different reasoning: In fact, g is convex, so that g has a unique minimum which is characterized by the root of the first derivative. Hence, (1.2.4) is also a sufficient optimality condition.

1.2.1 The Gauß-Newton algorithm

The nonlinear problem (1.2.4) can be solved e.g. by Newton's method⁵. Since we assumed that $F \in C^2(\Omega; \mathbb{R}^m)$, we may consider the following iteration for that:

$$\begin{aligned} g''(x^{(k)}) s^{(k)} &= -\nabla g(x^{(k)}), \quad \text{i.e., a linear system with solution } s^{(k)} \text{ (Newton correction)} \\ x^{(k+1)} &= x^{(k)} + s^{(k)} \\ &= x^{(k)} - [g''(x^{(k)})]^{-1} \nabla g(x^{(k)}) \\ &= x^{(k)} - \left[F'(x^{(k)})^T F'(x^{(k)}) + \sum_{i=1}^m F_i(x^{(k)}) F''_i(x^{(k)}) \right]^{-1} F'(x^{(k)})^T F(x^{(k)}). \end{aligned} \quad (1.2.5)$$

As in the case of Newton's method, we call (1.2.5) the *Newton update*.

³If one wants to replace the Euclidean norm by a different norm such as $\|\cdot\|_1$ or $\|\cdot\|_\infty$, different algorithms are needed. This goes beyond the scope of this lecture.

⁴Recall, that Ω is an *open* set.

⁵Sir Isaac Newton (1642-1729)

Recall, that due to the convexity of g , the Hessian matrix g'' is positive definite on a neighborhood of the critical point x^* and therefore g'' is invertible, at least locally. If we would have data that is fully compatible with the model, i.e., $F(x^*) = 0$, we would get

$$g''(x^*) = F''(x^*)^T F'(x^*) \quad (\text{if } F(x^*) = 0).$$

In this specific case, the Hessian matrix $g''(x^*)$ is positive definite if and only if the Jacobian matrix $F'(x^*)$ has full rank. For (almost) compatible data and for a x in a neighborhood $\mathcal{U}(x^*) \subset \Omega$ of x^* , one can approximate the exact Hessian matrix

$$g''(x) \approx F'(x)^T F'(x), \quad x \in \mathcal{U}(x^*),$$

similar to the *Quasi-Newton method* known from the lecture on *Numerical Analysis*. Consequently, the computation of the second derivative $F''(x)$ in (1.2.5) can be avoided. This procedure leads to the **Gauß-Newton method** shown in Algorithm 1.2.1.

Algorithm 1.2.1: Gauß-Newton algorithm

Input: initial guess $x^{(0)}$, function F with derivative F'

- 1: **for** $k = 0, 1, \dots$ **do**
- 2: compute $F(x^{(k)})$ and $F'(x^{(k)})$
- 3: determine the Newton correction $s^{(k)}$ by solving the linear system

$$F'(x^{(k)})^T F'(x^{(k)}) s^{(k)} = -F'(x^{(k)})^T F(x^{(k)}). \quad (1.2.6)$$

- 4: set $x^{(k+1)} = x^{(k)} + s^{(k)}$ (Newton update)
- 5: **end for**

Remark 1.2.1

- i) Obviously, (1.2.6) is the normal equation of the linear least squares problem

$$\left\| \underbrace{F'(x^{(k)})}_{=:A^{(k)}} s^{(k)} + \underbrace{F(x^{(k)})}_{=:b^{(k)}} \right\|_2 \rightarrow \min!. \quad (1.2.7)$$

Thus, the Gauß-Newton method reduces a nonlinear least squares problem to a sequence of linear least squares problems, quite similar to Newton's method reducing solving a nonlinear problem to a sequence of linear ones.

- ii) Similar to the derivation of Newton's method, one could also derive Algorithm 1.2.1 by Taylor's expansion with a second order remainder term. This is the reason why the scheme is called Gauß-Newton algorithm.
- iii) Recall that Newton's method has locally quadratic order of convergence. However, we cannot expect that the Gauß-Newton method inherits this since we use the approximation $g''(x) \approx F'(x)^T F'(x)$ for the computation of the Newton correction. One could say that this is like an inexact Newton scheme. In fact, we will show later that the Gauß-Newton method is linearly convergent, see also Section 1.2.2 below (Theorem 1.2.3). .

The following code is one possible realization of the Gauß-Newton algorithm in Matlab.

MATLAB-function: gauss_newton.m

```

1 function [x, iter] = gauss_newton(F,DF,x0,maxit,tol,flag)
2
3     if ~exist('flag','var')
4         flag = 0;
5     end
6     iter=zeros(maxit,4);
7     k=0;
8     x=x0;
9     errorold = norm(DF(x0)'*F(x0));
10    s=-(DF(x0)'*DF(x0))\ (DF(x0)'*F(x0));
11    while norm(s)>tol && k<maxit
12        k=k+1;
13        x=x+s;
14        fx=F(x);
15        s=-(DF(x)'*DF(x))\ (DF(x)'*fx);
16        errornew=norm(DF(x)'*fx);
17        iter(k,1)=norm(fx);
18        iter(k,2)=norm(s);
19        iter(k,3)=norm(errornew);
20        iter(k,4)=errornew/errorold;
21        errorold=errornew;
22        if (flag~=0)
23            fprintf('%i\t %12.8f\t %12.8f\t %12.8f\t %12.8f\n',...
24            k,iter(k,1),iter(k,2),iter(k,3),iter(k,4));
25        end
26    end
27    iter=iter(1:k,:);
28 end

```

Since the Gauß-Newton method is rather standard, we can also ask ChatGPT about it. Here is the result of the query.



Please give me a Matlab code for the Gauss-Newton method



The Gauss-Newton method is an iterative optimization technique used for solving non-linear least squares problems. Below is an example of MATLAB code implementing the Gauss-Newton method. This example will fit a non-linear model $y = a \cdot \exp(b \cdot x)$ to some data points (x_i, y_i) .



Please a code for a general input.



Sure, here is a more general MATLAB code for the Gauss-Newton method that allows for any non-linear function and corresponding Jacobian to be provided as input. This version takes a function handle for the model, its Jacobian, initial parameter estimates, data points, and various parameters for the iterative method.

MATLAB-Funktion per ChatGPT: gauss_newton_general.m

```

1 function [params, iter] = gauss_newton_general(model, jacobian, x, y,
2     p0, max_iter, tol)
3     % Inputs:
4     % model - function handle for the model y = f(x, p)
5     % jacobian - function handle for the Jacobian of the model
6     % x - vector of independent variable data points

```

```

6      % y - vector of dependent variable data points
7      % p0 - initial guess for the parameters
8      % max_iter - maximum number of iterations
9      % tol - convergence tolerance
10
11     % Initialize parameters
12     p = p0;
13
14     % Gauss-Newton iterations
15     for iter = 1:max_iter
16         % Compute the Jacobian matrix J and the residual vector r
17         J = jacobian(x, p);
18         r = y - model(x, p);
19
20         % Update the parameters
21         delta_p = (J' * J) \ (J' * r);
22         p = p + delta_p;
23
24         % Check for convergence
25         if norm(delta_p) < tol
26             break;
27         end
28     end
29
30     % Output the estimated parameters and number of iterations
31     params = p;
32 end

```

1.2.2 Convergence of the Gauß-Newton method

To analyze the Gauß-Newton method, we recall that the above assumptions ensure that there is a locally unique critical point $x^* \in \Omega$ of g , i.e., $\nabla g(x^*) = 0$ and there is a neighborhood $\mathcal{U}(x^*)$ in which x^* is the only root of ∇g . Furthermore, we have that

$$\text{rank}(F'(x)) = n \quad \text{for all } x \in \mathcal{U}(x^*).$$

Thus, given some $x^{(k)} \in \mathcal{U}(x^*)$, the normal equation (1.2.6) has a unique solution, which reads

$$s^{(k)} = -[F'(x^{(k)})^T F'(x^{(k)})]^{-1} F'(x^{(k)})^T F(x^{(k)}).$$

As a consequence, the Gauß-Newton iteration can be written as:

$$x^{(k+1)} = x^{(k)} - [F'(x^{(k)})^T F'(x^{(k)})]^{-1} F'(x^{(k)})^T F(x^{(k)}) = \Phi(x^{(k)}),$$

where we set

$$\Phi(x) := x - [F'(x)^T F'(x)]^{-1} F'(x)^T F(x). \quad (1.2.8)$$

This shows that the Gauß-Newton algorithm is as a fixed-point iteration with the fixed-point function $\Phi : \mathbb{R}^n \rightarrow \mathbb{R}^n$ and we can use standard fixed-point theorems, e.g. Banach's fixed-point theorem, to analyze its convergence.

It is well-known that a sufficient condition for the local linear convergence of the fixed-point iteration

$$x^{(k+1)} = \Phi(x^{(k)})$$

is that the Jacobian matrix is a contraction, i.e.,

$$\|\Phi'(x^*)\| < 1$$

for some arbitrary operator norm $\|\cdot\|$. In order to obtain locally second order convergence, one would need as a necessary condition that $\Phi'(x^*) = 0$. We are going to use these conditions in order to analyze the convergence of the Gauß-Newton scheme.

We assume that $\text{rank}(F'(x^*)) = n$, so that the matrix $F'(x^*)^T F'(x^*)$ is symmetric and positive definite. Thus, there exists a (unique) symmetric, positive definite matrix $A = A(x^*) \in \mathbb{R}^{n \times n}$, such that

$$A^2 = F'(x^*)^T F'(x^*) \quad (\text{i.e., } A = (F'(x^*)^T F'(x^*))^{1/2})$$

holds.⁶

For the subsequent convergence analysis, we define a matrix $K \in \mathbb{R}^{n \times n}$ by

$$K = K(x^*) := -A^{-1} \left(\sum_{i=1}^m \frac{F_i(x^*)}{\|F(x^*)\|_2} F_i''(x^*) \right) A^{-1}.$$

Note that both A and K depend on x^* but not on the iterates $x^{(k)}$. Since K is also symmetric by construction, all eigenvalues of K are real, i.e., $\sigma(A) \subset \mathbb{R}$ for the spectrum $\sigma(A)$ of A . We express the Hessian $g''(x^*)$ and the Jacobian $\Phi'(x^*)$ in terms of the matrices A and K .

Lemma 1.2.2 *Let $F : \mathbb{R}^n \rightarrow \mathbb{R}^m$, $g(x) := \frac{1}{2}F(x)^T F(x)$ and $\nabla g(x^*) = 0$ for some $x^* \in \mathbb{R}^n$ such that $F'(x^*)$ has full rank n . Then,*

$$g''(x^*) = A(I - \|F(x^*)\|_2 K)A, \quad (1.2.9)$$

$$\Phi'(x^*) = \|F(x^*)\|_2 A^{-1} K A. \quad (1.2.10)$$

If x^* is a local maximum or a saddle point of g , then

$$\varrho(K) \|F(x^*)\|_2 \geq 1 \quad (1.2.11)$$

holds, where $\varrho(K)$ denotes the spectral radius of K (the largest eigenvalue of K in absolute values).

Proof. We start by (1.2.3) for $x = x^*$, i.e.,

$$\begin{aligned} g''(x^*) &= F'(x^*)^T F'(x^*) + \sum_{i=1}^m F_i(x^*) F_i''(x^*) = A^2 + \|F(x^*)\|_2 \sum_{i=1}^m \frac{F_i(x^*)}{\|F(x^*)\|_2} F_i''(x^*) \\ &= A^2 - \|F(x^*)\|_2 A K A = A(I - \|F(x^*)\|_2 K)A, \end{aligned}$$

i.e., (1.2.9), where we used the definitions of A and K . W.r.t. (1.2.10), we have

$$\Phi'(x) = I - [F'(x)^T F'(x)]^{-2} \left[F'(x)^T F'(x) g''(x) - \frac{d}{dx} [F'(x)^T F'(x)] g'(x) \right].$$

Inserting $x = x^*$ and using $g'(x^*) = 0$ as well as (1.2.9) yields

$$\begin{aligned} \Phi'(x^*) &= I - [F'(x^*)^T F'(x^*)]^{-2} \left[F'(x^*)^T F'(x^*) g''(x^*) - \frac{d}{dx} [F'(x^*)^T F'(x^*)] \underbrace{g'(x^*)}_{=0} \right] \\ &= I - A^{-4} [A^2 A(I - \|F(x^*)\|_2 K)A] \\ &= I - A^{-2} (A^2 - \|F(x^*)\|_2 A K A) = \|F(x^*)\|_2 A^{-1} K A, \end{aligned}$$

⁶Recall the definition of the root of a symmetric positive definite matrix: Let $A = X^T \Lambda X$ be the Schur decomposition of A with an orthogonal matrix X and a diagonal matrix $\Lambda := \text{diag}(\lambda_i)$ with the eigenvalues $\lambda_i > 0$ of A . Then $X^{1/2} := X^T \Lambda^{1/2} X$, where $\Lambda := \text{diag}(\sqrt{\lambda_i})$. A more general definition uses the SVD of a symmetric matrix: If $A = U \Sigma U^T$, then $A^{1/2} = U \Sigma^{1/2} U^T$.

namely (1.2.10).

In order to show (1.3.6), note that if x^* is a local maximum or a saddle point of g , the Hessian matrix $g''(x^*)$ cannot be positive definite. In view of (1.2.9), this means that the matrix $I - \|F(x^*)\|_2 K$ must have non-positive eigenvalues. This in turn implies that there is an eigenvalue λ of $\|F(x^*)\|_2 K$ with $\lambda \geq 1$, which implies (1.3.6). \square

As another tool, we recall the energy-type norm defined for any symmetric positive definite matrix $A \in \mathbb{R}^{n \times n}$ by

$$\|x\|_A := \|Ax\|_2, \quad x \in \mathbb{R}^n,$$

as well as the induced matrix norm (operator norm) for $B \in \mathbb{R}^{n \times n}$ by

$$\|B\|_A := \max_{\|x\|_A=1} \|Bx\|_A = \max_{x \neq 0} \frac{\|Bx\|_A}{\|x\|_A} = \max_{x \neq 0} \frac{\|ABA^{-1}y\|_2}{\|Ay\|_2} = \max_{y \neq 0} \frac{\|ABA^{-1}y\|_2}{\|y\|_2} = \|ABA^{-1}\|_2.$$

Now we are in position to verify the conditions of Banach's fixed-point theorem.

Theorem 1.2.3 *Under the assumptions of Lemma 1.2.2, the iteration function Φ of (1.2.8) of the Gauß-Newton method satisfies*

$$\begin{aligned} \|\Phi'(x^*)\|_A &= \varrho(K) \|F(x^*)\|_2, \quad \text{as well as} \\ \|\Phi'(x^*)\| &\geq \varrho(K) \|F(x^*)\|_2 \quad \text{for any operator norm } \|\cdot\|. \end{aligned}$$

Proof. Lemma 1.2.2 yields

$$\|\Phi'(x^*)\|_A = \|A\Phi'(x^*)A^{-1}\|_2 = \|F(x^*)\|_2 \|K\|_2 = \|F(x^*)\|_2 \varrho(K),$$

where we used that $\varrho(K) = \|K\|_2$ for a symmetric matrix K . Next, recall that $\|B\| \geq \varrho(B)$ holds for all operator norms $\|\cdot\|$ and all $B \in \mathbb{R}^{n \times n}$, which implies

$$\|\Phi'(x^*)\| \geq \varrho(\Phi'(x^*)) = \|F(x^*)\|_2 \varrho(A^{-1}KA) = \|F(x^*)\|_2 \varrho(K),$$

and proofs the claim. \square

The following statements concerning the performance of the Gauß-Newton scheme are now immediate consequences of Theorem 1.2.3.

Corollary 1.2.4 *Under the assumptions of Lemma 1.2.2, we have:*

- i) *If $F(x^*) \neq 0$ (non-perfect data) and $\varrho(K) \|F(x^*)\|_2 < 1$, the Gauß-Newton method is locally linear convergent.*
- ii) *If the critical point x^* is a local maximum or a saddle point, we obtain $\varrho(K) \|F(x^*)\|_2 \geq 1$ and $\|\Phi'(x^*)\| \geq 1$ for any operator norm $\|\cdot\|$.*
- iii) *On the other hand, if x^* is a local minimum of g , such that $\varrho(K) \|F(x^*)\|_2 > 1$, we get $\|\Phi'(x^*)\| > 1$ for any operator norm $\|\cdot\|$, i.e., we cannot guarantee convergence.* \square

We collect some facts that we can deduce from the above findings. First, note that we should have $F(x^*) \neq 0$ since otherwise the data would perfectly fit the nonlinear model without any (measurement) errors. In that case, the condition

$$\varrho(K) \|F(x^*)\|_2 < 1,$$

is crucial for the (local) convergence – we cannot expect that the Gauß-Newton method always converges. On the other hand, if x^* is a local minimum of g , such that $\varrho(K) \|F(x^*)\|_2 > 1$, we cannot expect convergence at all.



- The local convergence of the Gauß-Newton algorithm is ensured for a local minimum x^* such that $\varrho(K)\|F(x^*)\|_2 < 1$.
- If the Gauß-Newton iteration converges, the convergence is in general not faster than linear.
- A local minimum of g can be a rejecting point of the Gauß-Newton scheme.
- Local maxima and saddle points are rejecting points of the Gauß-Newton algorithm. This is a positive property, as we are searching local *minima*.

1.2.3 Examples – continued

We come back to the examples described in Section 1.1 and detail the solution of these examples by means of the Gauß-Newton method.

Example 1.2.5 (Regression Circle – Continued) Reconsider Example 1.1.1 of determining a regression circle. We can easily determine the partial derivatives of the function F given in (1.1.1), in fact

$$\begin{aligned}\frac{\partial}{\partial \mu_s} F_i(\mu_s, \mu_t, r) &= \frac{\mu_s - s_i}{\sqrt{(s_i - \mu_s)^2 + (t_i - \mu_t)^2}}, \\ \frac{\partial}{\partial \mu_t} F_i(\mu_s, \mu_t, r) &= \frac{\mu_t - t_i}{\sqrt{(s_i - \mu_s)^2 + (t_i - \mu_t)^2}}, \\ \frac{\partial}{\partial r} F_i(\mu_s, \mu_t, r) &= -1.\end{aligned}$$

The realization of this example in Matlab can be seen in the following code:

MATLAB-function: RegressionCircle_GaussNewton.m

```

1 clear; close all;
2 %---prescribe the data -----
3 N=100; % no of points
4 dr = 0.025; % max deviation from radius 2
5 % severely impacts convergence
6 rad_L=(2*dr).*randn(N,1)+2-dr; % radius with random distortions
7 zx_L=3; % 'exact' center coordinate
8 zy_L=2; %
9 x_L=zx_L*ones(N,1) + cos(linspace(0,2*pi,N)').*rad_L;
10 y_L=zy_L*ones(N,1) + sin(linspace(0,2*pi,N)').*rad_L;
11 %-----
12
13 x=x_L;
14 y=y_L;
15 m=length(x);
16
17 %--- provide F and Jacobian as functions -----
18 F=@(p) sqrt((x-p(1)).^2+(y-p(2)).^2)-p(3);
19 DF=@(p) [(p(1)-x)./sqrt((x-p(1)).^2+(y-p(2)).^2), ...
20 (p(2)-y)./sqrt((x-p(1)).^2+(y-p(2)).^2), ...
21 -ones(m,1)];
22 %
23
24 %-- initial value - might severely influence convergence
25 p0=[1;1;1];
26
27 maxit=100;
28 tol=1e-10;
```

```

9
0 [xGN, itsGN]=gauss_newton(F,DF,p0,maxit,tol,1);
1
2 fprintf('Gauss-Newton: center = (%12.8f,%12.8f), radius=%12.8f\n',...
3 xGN(1),xGN(2),xGN(3));
4
5 figure(1)
6 hold on
7 plot(x,y,'x','Linewidth',1,'MarkerEdgeColor','k');
8 fplot(@(t) xGN(3)*sin(t)+xGN(1), @(t) xGN(3)*cos(t)+xGN(2),'k',...
9 'Linewidth',1);
hold off
0
1 figure(2);
2 semilogy(itsGN(:,3),'Linewidth',1.5);
3 grid on;
4
5 maxk=size(itsGN,1);
6 fileID = fopen('..../Data/RegCircleGNData.txt','w');
7 for (k=1:maxk)
8     fprintf(fileID,'%i & %12.8f & %12.8f & %12.8f & %12.8f\\ \\\n',...
9         k,itsGN(k,1),itsGN(k,2),itsGN(k,3),itsGN(k,4));
0 end
1 fclose(fileID);

```

Let us add some comments on the code and this example whose results are shown in Figure 1.5.

- We construct randomly distorted data for a circle around $(\mu_s, \mu_t) = (3, 2)$ with radius $r = 2$ (Figure 1.1) in the sense that we pick N (here $N = 100$) points around the circle with a random error in the radius. These errors are normally distributed in the interval $[-\Delta_r, \Delta_r]$, where here $\Delta_r = 0.025$ is chosen.
- We choose the initial guess as $(1, 1, 1)$, fix the maximal number `maxit` of iterations to 100 and the tolerance for stopping the iteration if the Gauß-Newton-updates are too small to `tol` = 0.1.

We obtain a number of observations:

- The Gauß-Newton scheme may converge or not – even if the same code is started several times. The reason is that the data is chosen randomly and this has significant influence on the convergence condition $\varrho(K) \|F(x^*)\|_2 < 1$ even using the same initial guess for the iteration.
- The choice of N and Δ_r severely impacts the convergence or non-convergence. Of course, also the choice of the initial guess does. Try it yourself, what do you observe?

In Figure 1.5, we have shown the results of one loop that did in fact converge. On the left, the data and the computed regression circle is shown. The convergence history is shown in Table 1.2, where we monitor the norm $\|F(x^{(k)})\|_2$ of the cost function, the norm $\|s^{(k)}\|_2$ of the Gauß-Newton update as well as

$$\|\nabla g(x^{(k)})\|_2 = \|F'(x^{(k)})^T F(x^{(k)})\|_2$$

as a measure for the error (recall that $\nabla g(x^*) = 0$) and

$$\varrho_k := \frac{\|\nabla g(x^{(k)})\|_2}{\|\nabla g(x^{(k-1)})\|_2}$$

as a convergence (or error reduction) factor. In this case, we have extremely fast convergence after $k = 6$ iterations. In this case, the scheme stoped after 11 iterations. In the right graph of Figure 1.5, we show $\|\nabla g(x^{(k)})\|_2$ over k . We clearly see that we do not have monotone convergence w.r.t. this quantity, at least for the first 5 iterations. Once a good guess has been found (which here seems to be case for $k = 5$),

the scheme converges very fast towards a good approximation. However, it might well happen that the iteration does not reach any good guess. Both non-convergence and blow-up has been observed. We conclude that even in this quite simple example, the behavior of the Gauß-Newton method might be a very delicate issue. \diamond

k	$\ F(x^{(k)})\ _2$	$\ s^{(k)}\ _2$	$\ \nabla g(x^{(k)})\ _2$	ϱ_k
1	50.29866982	9.41886442	653.30987051	2.87602397
2	104.98135385	19.26158465	1450.21797093	2.21980110
3	238.01805394	33.72637401	3349.35572309	2.30955332
4	405.93895073	85.49496447	5730.74931573	1.71100050
5	1135.67716213	80.34219478	16057.27388149	2.80195015
6	38.57277927	3.46195221	428.46714512	0.02668368
7	4.24465722	0.43640230	41.43369887	0.09670216
8	0.48568419	0.00468429	0.46659789	0.01126131
9	0.48343196	0.00000123	0.00006210	0.00013310
10	0.48343196	0.00000000	0.00000018	0.00290444
11	0.48343196	0.00000000	0.00000000	0.00299566

Table 1.2: Convergence history of the Gauß-Newton method for the regression circle problem in Example 1.1.1.

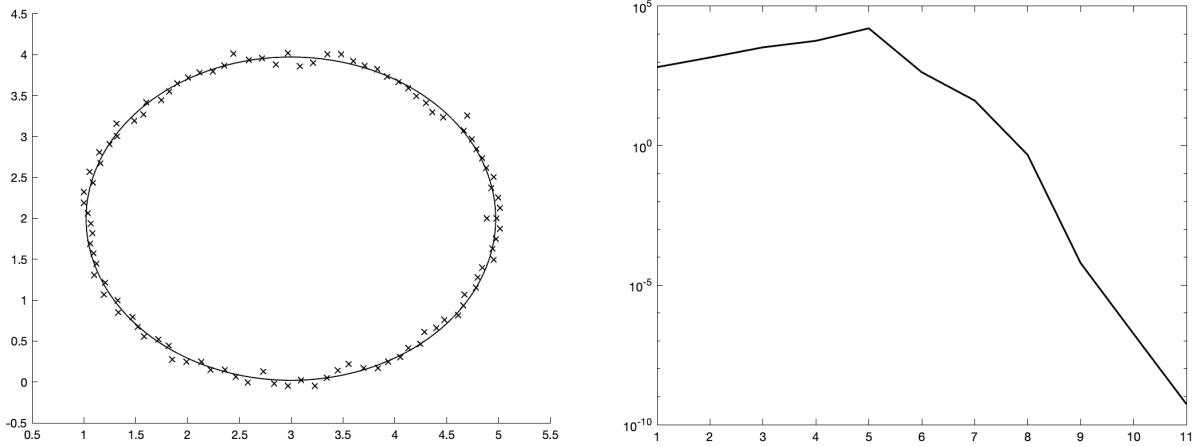


Figure 1.5: Result (left) and convergence history (right) of the Gauß-Newton method for Example 1.1.1.

Example 1.2.6 (Density of the Gamma-distribution – Continued) Now, we reconsider Example 1.1.3. In this case it is at least not straightforward (if not impossible) to derive a formula for the Jacobian of the function F given in (1.1.3). Hence we use a Finite Difference approximation of the derivatives as follows ($F_i(x) = F_i(\lambda, p)$)

$$\frac{\partial}{\partial \lambda} F_i(\lambda, p) \approx \frac{\varrho(t_i; \lambda + \Delta\lambda, p) - \varrho(t_i; \lambda, p)}{\Delta\lambda}, \quad \frac{\partial}{\partial p} F_i(\lambda, p) \approx \frac{\varrho(t_i; \lambda, p + \Delta p) - \varrho(t_i; \lambda, p)}{\Delta p}$$

for small step sizes $\Delta\lambda > 0$ and $\Delta p > 0$. If ϱ is sufficiently smooth, these Finite Difference approximations converge towards the desired partial derivatives. However, there are stability and accuracy issues, so that is interesting to vary the values of $\Delta\lambda$ and Δp in the subsequent Matlab code. The data are produced from random distortions of a given Γ -distribution for $\lambda = 1.5$ and $p = 2$ with normally distributed distortions in the interval $[-\sigma, \sigma]$ with $\sigma > 0$ to be chosen (here $\sigma = 0.015$).

The result is shown in Figure 1.6. On the left, we also show the comparison with the original Γ -distribution. The convergence history is shown on the right in Figure 1.6 as well as in Table 1.3. For $k \geq 10$, we observe linear convergence with convergence factor $\varrho_k \approx \frac{1}{3}$. We suggest the reader to do experiments with different values of $\Delta\lambda$, Δp , N_{data} , σ and the initial value. What do you observe? \diamond

k	$\ F(x^{(k)})\ _2$	$\ s^{(k)}\ _2$	$\ \nabla g(x^{(k)})\ _2$	ϱ_k
1	0.87126570	2.08827062	1.74121642	0.74469083
2	0.35071883	3.79192717	0.23906621	0.13729839
3	0.48288202	0.82760123	0.33126167	1.38564821
4	0.23615907	0.44681318	0.08038685	0.24266875
5	0.17869345	0.03344688	0.00275478	0.03426909
6	0.17863205	0.01713905	0.00019954	0.07243322
7	0.17869032	0.00203917	0.00012882	0.64558850
8	0.17868181	0.00151075	0.00002231	0.17317766
9	0.17868772	0.00046709	0.00001244	0.55749814
10	0.17868585	0.00019661	0.00000416	0.33444282
11	0.17868663	0.00007287	0.00000169	0.40624399
12	0.17868634	0.00002838	0.00000063	0.37561421
13	0.17868645	0.00001084	0.00000025	0.38737527
14	0.17868641	0.00000417	0.00000009	0.38266484
15	0.17868642	0.00000160	0.00000004	0.38451900
16	0.17868642	0.00000061	0.00000001	0.38378468
17	0.17868642	0.00000024	0.00000001	0.38407457
18	0.17868642	0.00000009	0.00000000	0.38396003
19	0.17868642	0.00000003	0.00000000	0.38400517
20	0.17868642	0.00000001	0.00000000	0.38398717
21	0.17868642	0.00000001	0.00000000	0.38399378
22	0.17868642	0.00000000	0.00000000	0.38399208
23	0.17868642	0.00000000	0.00000000	0.38399833
24	0.17868642	0.00000000	0.00000000	0.38399493
25	0.17868642	0.00000000	0.00000000	0.38397182
26	0.17868642	0.00000000	0.00000000	0.38400845

Table 1.3: Convergence history of the Gauß-Newton method for the regression circle problem in Example 1.1.3.

MATLAB-function: Gamma_GaussNewton.m

```

1 close all; clear;
2 %-- requires Matlab Statistics and Machine Learning Toolbox
3
4
5 %-- data is produced by distorting an exact Gamma-distribution
6 lambda_ex = 1.5;
7 p_ex = 2;
8
9 %-- step sizes for Finite Difference approximation
0 dp=0.1;
1 dl=0.1;

```

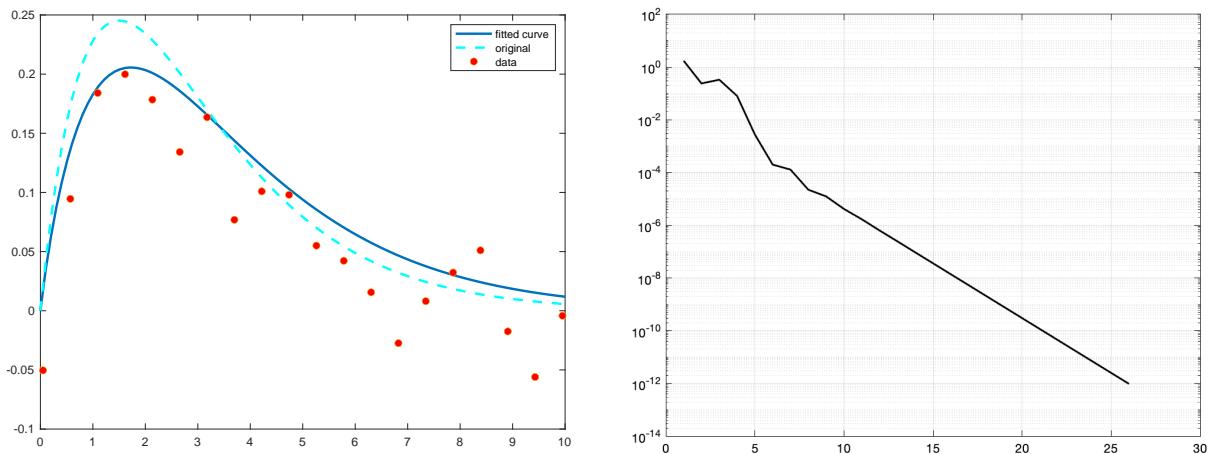


Figure 1.6: Result (left) and convergence history (right) of the Gauß-Newton method for Example 1.1.3.

```

12
13 %-- data
14 N_data=20; %- no of data points
15 sigma=0.015; %- distortions are in [-sigma, sigma]
16 t_data=linspace(0.05,9.95,N_data);
17 dy=(2*sigma).*randn(N_data,1)-sigma;
18 y_data = (gampdf(t_data,p_ex,lambda_ex)' +dy)';
19 x=t_data; y=y_data;
20
21 %-- parameters for Gauss-Newton iteration
22 p0=[1;1]; %- initial value
23 maxit=100; %- max number of iterations
24 tol=1e-10; %- tolerance
25
26 %--- provide F and Jacobian as functions -----
27 F=@(p) [gampdf(x,p(1),p(2))-y]';
28 DF=@(p) [(gampdf(x,p(1)+dp,p(2))-gampdf(x,p(1),p(2)))'./dp, ...
29             (gampdf(x,p(1),p(2)+dl)-gampdf(x,p(1),p(2)))'./dl];
30 %
31
32 [xGN,itsGN]=gauss_newton(F,DF,p0,maxit,tol,1);
33 fprintf('Gauss-Newton: p = %12.8f, lambda=%12.8f\n',xGN(1),xGN(2));
34
35 %
36 %- compare with 'exact' Gamma distribution
37 N=100;
38 t=linspace(0,10,N);
39 gamma_t = gampdf(t,p_ex,lambda_ex);
40
41 figure(1)
42 plot(t,gampdf(t,xGN(1),xGN(2)),'linewidth',2);
43 hold on
44 plot(t, gamma_t,'c--','linewidth',2);
45 plot(t_data,y_data,'o','MarkerSize',6,'MarkerFaceColor','red');
46 hold off
47 legend('fitted curve','original','data');
48
49 %- convergence histoty

```

```

0 figure(2);
1 semilogy(itsGN(:,3),'Linewidth',1.5);
2 grid on;
3
4 maxk=size(itsGN,1);
5 fileID = fopen('..../Data/GammaGNDATA.txt','w');
6 for (k=1:maxk)
7   fprintf(fileID,['%i & %12.8f & %12.8f & %12.8f & %12.8f...' ...
8     '\\\\ \\hline\\n'], k,itsGN(k,1),itsGN(k,2),...
9     itsGN(k,3),itsGN(k,4));
10 end
11 fclose(fileID);

```

Example 1.2.7 (Damped oscillation – Continued) Next, we reconsider Example 1.1.4. The derivatives can easily be determined, namely for $x = (u_0, \delta, \omega, \varphi_0)$ and $i = 1, \dots, m$, we have

$$F'_i(x) = e^{-\delta t_i} \begin{pmatrix} \sin(\omega t_i + \varphi_0) \\ -t_i u_0 \sin(\omega t_i + \varphi_0) \\ u_0 \cos(\omega t_i + \varphi_0) \\ t_i u_0 \cos(\omega t_i + \varphi_0) \end{pmatrix}.$$

The subsequent Matlab file (including the data for t_i and u_i) is basically taken from [5]. The result is shown in Figure 1.7. The convergence history is shown in Table 1.4 and in the right part of Figure 1.7. For $k \geq 6$, we obtain a linear convergence with an error reduction of about $\frac{1}{3}$. \diamond

k	$\ F(x^{(k)})\ _2$	$\ s^{(k)}\ _2$	$\ \nabla g(x^{(k)})\ _2$	ϱ_k
1	0.34259563	1.37305616	0.13517833	0.92961234
2	0.22267010	1.18448860	0.04873647	0.36053465
3	0.17049095	0.24020896	0.10401256	2.13418328
4	0.09214175	0.04973134	0.01861028	0.17892336
5	0.08911345	0.01520798	0.00121238	0.06514590
6	0.08904164	0.00440781	0.00039385	0.32485657
7	0.08903603	0.00158256	0.00012024	0.30529939
8	0.08903545	0.00053168	0.00004242	0.35281727
9	0.08903538	0.00018786	0.00001438	0.33905700
10	0.08903538	0.00006484	0.00000506	0.35147259
11	0.08903538	0.00002268	0.00000175	0.34598169
12	0.08903538	0.00000788	0.00000061	0.34934342
13	0.08903538	0.00000275	0.00000021	0.34763903
14	0.08903538	0.00000096	0.00000007	0.34858718

Table 1.4: Convergence history of the Gauß-Newton method for the damped oscillation problem in Example 1.1.4.

MATLAB-function: Oscillation_GaussNewton.m

```

1 %-- This file is adapted from W. Dahmen, A. Reusken
2 %-- Numerik fuer Ingenieure und Naturwissenschaftler, 3. Aufl.
3 %-- Springer, 2022

```

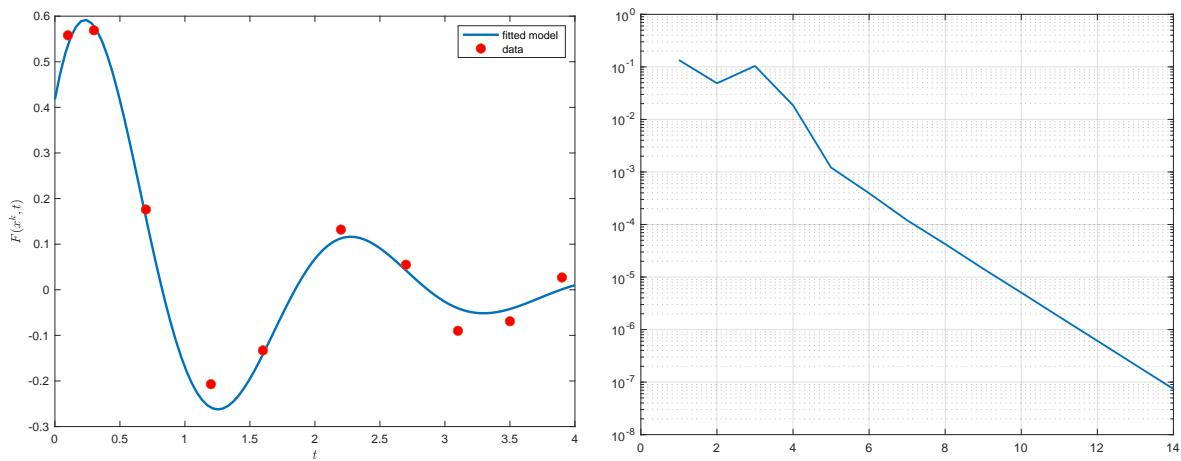


Figure 1.7: Result (left) and convergence history (right) of the damped oscillation Example 1.1.4.

```

4 %-- https://www.igpm.rwth-aachen.de/DahmenReusken/demodownload
5
6 %-- data
7 ti=[0.1 0.3 0.7 1.2 1.6 2.2 2.7 3.1 3.5 3.9]';
8 bi=[0.558 0.569 0.176 -0.207 -0.133 0.132 0.055 -0.090 -0.069 0.027]';
9
10 x=ti; y=bi;
11 p0=[1, 2, 2, 1]';
12
13 %--- provide F and Jacobian as functions -----
14 F=@(p) p(1).*exp(-p(2).*x).*sin(p(3).*x+p(4))-y;
15 DF=@(p) [exp(-p(2).*x).*sin(p(3).*x+p(4)), ...
16 -p(1).*x.*exp(-p(2).*x).*sin(p(3).*x+p(4)), ...
17 p(1).*x.*exp(-p(2).*x).*cos(p(3).*x+p(4)), ...
18 p(1).*exp(-p(2).*x).*cos(p(3).*x+p(4))];
19 %-----
20
21 maxit=100;
22 tol=1e-6;
23
24 [xGN,itsGN]=gauss_newton(F,DF,p0,maxit,tol,1);
25
26 fprintf('Gauss-Newton: u= %12.8f, delta=%12.8f, omega=%12.8f, phi=%12.8f
27 \n',...
28 xGN(1),xGN(2),xGN(3),xGN(4));
29
30 figure(1)
31 t=linspace(0,4);
32 hold on
33 plot(t,xGN(1)*exp(-xGN(2).*t).*sin(xGN(3).*t+xGN(4)),'linewidth',2)
34 plot(ti,bi,'o','MarkerSize',8,'MarkerFaceColor','red');
35 hold off
36 xlabel('$t$', 'Interpreter', 'latex');
37 ylabel('$F(x^k, t)$', 'Interpreter', 'latex')
38 legend('fitted model','data')
39 figure(2);
40 semilogy(itsGN(:,3),'Linewidth',1.5);
41 grid on;

```

```

2
3 maxk=size(itsGN,1);
4 fileID = fopen('..../Data/OscillationGNDATA.txt','w');
5 for (k=1:maxk)
6   fprintf(fileID,'%i & %12.8f & %12.8f & %12.8f & %12.8f\\\'\\\'\n',
7     k,itsGN(k,1),itsGN(k,2),itsGN(k,3),itsGN(k,4));
8 end
9 fclose(fileID);

```

Example 1.2.8 (Where have I been? – Continued) Finally, we continue Example 1.1.2 and describe the realization of the Gauß-Newton method for his example. The derivation of the derivatives of $F_i(x) = F_i(L, d, \vartheta) := w_i(L, d, \vartheta) \times (L - P_i) \in \mathbb{R}^3$ w.r.t. L, d and ϑ turns out to be a bit involved. Hence, we detail the partial derivatives of all involved quantities. Let $d = (d_1, d_2, d_3)^T$, then

$$\frac{\partial \tilde{e}}{\partial d_1} = \begin{pmatrix} 0 \\ -1 \\ 0 \end{pmatrix}, \quad \frac{\partial \tilde{e}}{\partial d_2} = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}, \quad \frac{\partial \tilde{e}}{\partial d_3} = 0, \quad \frac{\partial e}{\partial d_i} = \frac{(\frac{\partial \tilde{e}}{\partial d_i})\|\tilde{e}\| - \tilde{e}\frac{\partial \|\tilde{e}\|}{\partial d_i}}{\|\tilde{e}\|^2},$$

(by the quotient rule),

$$\begin{aligned} \frac{\partial \|\tilde{e}\|}{\partial d_1} &= \frac{\partial}{\partial d_1} (d_1^2 + d_2^2)^{1/2} = \frac{d_1}{\|\tilde{e}\|}, & \text{so that} & \tilde{e} \frac{\partial \|\tilde{e}\|}{\partial d_1} = \frac{d_1 e}{\|\tilde{e}\|} = d_1 e, \\ \frac{\partial \|\tilde{e}\|}{\partial d_2} &= \frac{\partial}{\partial d_2} (d_1^2 + d_2^2)^{1/2} = \frac{d_2}{\|\tilde{e}\|}, & \text{so that} & \tilde{e} \frac{\partial \|\tilde{e}\|}{\partial d_2} = \frac{d_2 e}{\|\tilde{e}\|} = d_2 e \end{aligned}$$

and finally

$$\frac{\partial e}{\partial d_1} = \begin{pmatrix} 0 \\ -1 \\ 0 \end{pmatrix} / \|\tilde{e}\| + d_1 \frac{e}{\|\tilde{e}\|^2}, \quad \frac{\partial e}{\partial d_2} = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} / \|\tilde{e}\| - d_2 \frac{e}{\|\tilde{e}\|^2}, \quad \frac{\partial e}{\partial d_3} = 0.$$

The derivative of f w.r.t. is a bit more involved, but still straightforward. Indeed,

$$\frac{\partial \tilde{f}}{\partial d_1} = \begin{pmatrix} -d_3 \\ 0 \\ 2d_1 \end{pmatrix}, \quad \frac{\partial \tilde{f}}{\partial d_2} = \begin{pmatrix} 0 \\ -d_3 \\ 2d_2 \end{pmatrix}, \quad \frac{\partial \tilde{f}}{\partial d_3} = \begin{pmatrix} -d_1 \\ -d_2 \\ 0 \end{pmatrix},$$

and by using the quotient rule, we get with the common factor $\gamma := \frac{1}{\|\tilde{e}\|^2} + \frac{1}{\|d\|^2}$

$$\frac{\partial f}{\partial d_1} = \frac{1}{\|\tilde{f}\|} \frac{\partial \tilde{f}}{\partial d_1} - \gamma d_1 f, \quad \frac{\partial f}{\partial d_2} = \frac{1}{\|\tilde{f}\|} \frac{\partial \tilde{f}}{\partial d_2} - \gamma d_2 f, \quad \frac{\partial f}{\partial d_3} = \frac{1}{\|\tilde{f}\|} \frac{\partial \tilde{f}}{\partial d_3} - \gamma d_3 f,$$

so that

$$\frac{\partial}{\partial d_j} w_i(d, \vartheta) = \left[\delta_j, \frac{\partial e}{\partial d_j}, \frac{\partial f}{\partial d_j} \right] \cdot \left[\begin{array}{c} 1 \\ S_i \end{array} \right] = \delta_j + S_{i,1} \frac{\partial e}{\partial d_j} + S_{i,2} \frac{\partial f}{\partial d_j}, \quad (1.2.12)$$

where $\delta_j = (\delta_{i,j})_{i=1,2,3}^T$ denotes the unit vector in \mathbb{R}^3 . Next,

$$\frac{\partial}{\partial \vartheta} S_i(\vartheta) = \begin{pmatrix} -\sin \vartheta & -\cos \vartheta \\ \cos \vartheta & -\sin \vartheta \end{pmatrix} M_i,$$

which yields

$$\frac{\partial}{\partial \vartheta} w_i(d, \vartheta) = (e, f) \cdot \frac{\partial}{\partial \vartheta} S_i(\vartheta). \quad (1.2.13)$$

Now, we have everything together to detail $F'_i(x)$ as follows

$$\begin{aligned}\frac{\partial}{\partial L_j} F_i(x) &= w_i(d, \vartheta) \times \delta_j, & j = 1, 2, 3, \\ \frac{\partial}{\partial d_j} F_i(x) &= \frac{\partial}{\partial d_j} w_i(d, \vartheta) \times (L - P_i), & j = 1, 2, 3, \\ \frac{\partial}{\partial \vartheta} F_i(x) &= \frac{\partial}{\partial \vartheta} w_i(d, \vartheta) \times (L - P_i).\end{aligned}$$

We used a weighted norm to measure the residual, namely $\|x\|_\omega := \max\{\omega\|L\|_2, \|d\|, \omega|\vartheta|\}$ with $\omega = 10.000$, which accounts for the different scaling of the variables. Since the computation of $F_i(x)$ and $F'_i(x)$ is more involved as in the previous three examples, we decided not to use our generic implementation of the Gauß-Newton method in `gauss_newton.m`, but to include this into the Matlab file directly.

We show the results in Figure 1.8. For the initial guess $x^{(0)} = (L^{(0)}, d^{(0)}, \vartheta^{(0)})$ with $L^{(0)} = (6000, 9000, 600)^T$, $d = (0, -0.2, 0)^T$ and $\vartheta^{(0)} = 0$, we need 6 iterations to reach $\|s^{(k)}\|_\omega^2 < 0.1$, see Table 1.5. In fact, the convergence is significantly faster than in all previous examples. \diamond

k	$\ F(x^{(k)})\ _2$	$\ s^{(k)}\ _2$	$\ \nabla g(x^{(k)})\ _2$	ϱ_k
1	1578.34905248	1299.87447014	21512241.39991386	0.07348377
2	140.40467946	2289.78159043	1580800.52108272	0.25382648
3	55.84454690	397.96851721	401249.03577339	0.04872505
4	3.18434754	56.75331920	19550.88030125	0.02799033
5	1.69204080	0.67571208	547.23562328	0.00012634
6	1.69039704	0.01457665	0.06913512	0.01807818

Table 1.5: Convergence history of the Gauß-Newton method for Example 1.1.2.

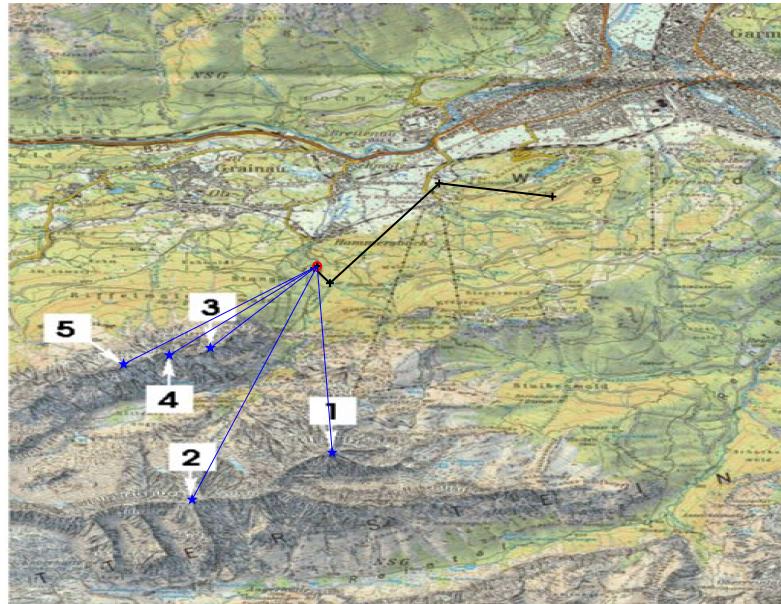


Figure 1.8: Iterations of the Gauß-Newton method for Example 1.1.2. Iterations are marked (and connected) in black, the final approximation in red and the perspective of the lens towards the summits in blue.

MATLAB-function: WhereAmI.m

```

1
2 function WhereAmI
3 clear all
4 close all
5 %-----
6 % data: cols 1:2 position on map
7 % col 3 sea level height
8 % cols 4:5 position on picture
9 dat = [ 3700, 4125, 2627.6, -0.0280, 0.0340;
0 2200, 3175, 2720.0, 0.0025, 0.0265;
1 2350, 6000, 2136.0, 0.0325, 0.0355;
2 2175, 5885, 2072.0, 0.0370, 0.0305;
3 1850, 5950, 2276.5, 0.0505, 0.0390];
4
5 %-- these are the pixel data of the image of the map
6 MapData = [ 215, 498;
7 123, 550;
8 135, 382;
9 108, 390;
0 78, 400];
1
2 %-- we computed a weighted 2-norm
3 weight = 10000;
4
5 %-----
6 %-- Gauss-Newton method for solving the nonlinear least squares
7 %-- problem ||F(x)||_2 = min!
8 %-- initial data
9 L = [ 6000; 9000; 600];
0 d = [0; -0.2; 0];
1 theta = 0;
2 x = [d; L; theta];
3 k=0;
4
5 x_all = x; %-- store all iterations
6 fileID = fopen('..../Data/WhereAmIGNData.txt','w+');
7
8 [Fx, JFx] = F(x, dat);
9 errorold = norm(JFx'*Fx);
0
1 while 1
2 k=k+1;
3 % determine F(x) and JF(x)
4 [Fx, JFx] = F(x, dat);
5 error = norm(JFx'*Fx);
6
7 % linear least squares probl -> update dx
8 dx = JFx\Rx;
9 x = x - dx;
0
1 x_all = [x_all, x];
2 % weighted norm

```

```

53     dxnorm = max([weight*norm(dx(1:3)), norm(dx(4:6)), weight*abs(dx(7))
54         ]);
55     fxnorm = norm(Fx);
56     [Fx, JFx] = F(x, dat);
57     errornew = norm(JFx'*Fx);
58     rho = errornew/errorold;
59     errorold=errornew;
60
61     fprintf('||F(x)|| = %14.8f, ||dx|| = %12.8f, err = %12.8f\n',...
62         fxnorm,dxnorm,error);
63     fprintf(fileID,'%i & %12.8f & %12.8f & %12.8f\\\\ \\hline\n'
64         ...
65         k,fxnorm,dxnorm,error,rho);
66
67     % stop if update is small
68     if dxnorm < 0.1
69         break;
70     end
71 end
72 fclose(fileID);
73
74 %-----
75 %-- display the results
76 %-- show the iterations for the lenses on the map
77 L = x_all(4:5,:);
78 LP = TransformMapImage(dat(1:3,1:2)', MapData(1:3,:)', L);
79
80 mf=figure(1);
81 Image=imread([pwd,'/MountainMap.jpg']);
82 image(Image);
83
84 hold on
85 orig = scatter(MapData(:,1),MapData(:,2),'filled','blue','pentagram');
86 orig.SizeType=150;
87 plot(LP(1,:),LP(2,:)','k+-','linewidth',2,'markersize',8);
88 plot(LP(1,end),LP(2,end)','ro','linewidth',2,'markersize',8);
89 for i=1:5
90     plot([LP(1,end),MapData(i,1)],[LP(2,end),MapData(i,2)],...
91         'b-','linewidth',1);
92 end
93 axis off
94
95 set(mf, 'PaperOrientation', 'landscape');
96 set(mf, 'PaperUnits', 'inches');
97 set(mf, 'PaperSize', [11 8.5]);
98 set(mf, 'PaperPositionMode', 'manual');
99 set(mf, 'PaperPosition', [0 0 11 8.5]);
100 set(gca, 'Position', [0 0 1 1]); % image fills the entire figure
101 print('..../Data/ThereAmI.pdf', '-dpdf', '-r0');
102
103 %-----
104 % determine F(x) and jacobian JF(x)
105 function [Fx, JFx] = F(x, dat)
106     % dimensions
107     m = 3*size(dat,1);
108     n = 7;
109     Fx = zeros(m, 1);

```

```

9      JFx = zeros(m, n);
0
1      % F: first 3 rows
2      for i = 1:size(dat,1)
3          i0 = 3*i-2;
4          [Fx(i0:i0+2), JFx(i0:i0+2,:)] = Fi(x(1:3), x(4:6), x(7), ...
5                                         dat(i,1:3)', dat(i,4:5)');
6      end
7 %-----
8
9 %-----
0 % F(x) and JF(x) for given data
1 function [Fvalue, JFvalue] = Fi(d, L, theta, Q, M)
2 %-- determine ONB (d,e,f) with d orth. to picture (note the sign!)
3 dnorm=norm(d);
4 %--
5 e = [d(2); -d(1); 0];
6 enorm = norm(e);
7 e = e/enorm;
8 %-- e=h*d
9 f = [-d(1)*d(3); -d(2)*d(3); d(1)^2+d(2)^2];
0 fnorm = norm(f);
1 f = f/fnorm;
2
3 %-- position of the current summit after rotation
4 S = [cos(theta), sin(theta); -sin(theta), cos(theta)]*M;
5 %-- vector w from 3d picture position to center of gravity
6 %-- in (d,e,f)-coordinates
7 %-- the first 1 turns outward of the picture
8 w = [d, e, f]*[1; S];
9
0 %-- w should be collinear to Q-L, this is F(x)
1 Fvalue = cross(w, Q-L);
2
3 %-- now JF_i
4 JFvalue = zeros(3, 7);
5
6 %-- derivative w.r.t. L
7 JFvalue(:,4:6) = -[cross(w, [1;0;0]), cross(w, [0;1;0]), ...
8                      cross(w, [0;0;1])];
9
0 %-- derivative w.r.t. theta
1 du_dtheta = [-sin(theta), cos(theta); -cos(theta), -sin(theta)]*M;
2 dw_dtheta = [e, f]*[du_dtheta];
3 JFvalue(:,7) = cross(dw_dtheta, Q-L);
4
5 %-- derivative w.r.t. d = (a;b;c) = (d_1;d_2;d_3)
6 dh_da = [0; -1; 0]/enorm - d(1)*e/enorm^2;
7 dh_db = [1; 0; 0]/enorm - d(2)*e/enorm^2;
8 dh_dc = [0;0;0];
9 dg_da = [-d(3); 0; 2*d(1)]/fnorm - f*d(1)*(1/(enorm^2) ...
10                                + 1/(dnorm^2));
11 dg_db = [0; -d(3); 2*d(2)]/fnorm - f*d(2)*(1/(enorm^2) ...
12                                + 1/(dnorm^2));
13 dg_dc = [-d(1); -d(2); 0]/fnorm - f*d(3)*(1/(dnorm^2));
14 dw_da = [[1;0;0], dh_da, dg_da]*[1; S];
15 dw_db = [[0;1;0], dh_db, dg_db]*[1; S];
16 dw_dc = [[0;0;1], dh_dc, dg_dc]*[1; S];

```

```

167
168      JFvalue(:,1) = cross(dw_da, Q-L);
169      JFvalue(:,2) = cross(dw_db, Q-L);
170      JFvalue(:,3) = cross(dw_dc, Q-L);
171  %-----
172
173  %-----
174  % transformation of points from the map to the image
175  function ImageData = TransformMapImage(MapBase, ImageBase, MapData)
176      % base vectors on image
177      B = ImageBase(:,2:3) - [ImageBase(:,1), ImageBase(:,1)];
178      % base vectors on map
179      C = MapBase(:,2:3) - [MapBase(:,1), MapBase(:,1)];
180      % matrix of the change of basis
181      A = B/C;
182      % transform center of gravity
183      b = (1/3)*ImageBase*ones(3, 1) - A*(1/3)*MapBase*ones(3, 1);
184      % transform x -> Ax+b
185      ImageData = A*MapData + b*ones(1, size(MapData, 2));
186  %-----

```

1.3 LEVENBERG-MARQUARDT METHOD

An obvious problem of the Gauß-Newton scheme is the fact that the linearization of $F(x)$ at $x = x^{(k)}$ is only a decent approximation in a (probably small) neighborhood of $x^{(k)}$, i.e., for “small” steps $s^{(k)}$. This is particularly true if the distance of the current approximation $x^{(k)}$ to the desired minimum is large. We have also clearly seen in this in our above numerical experiments. The following approach tries to overcome that deficiency by introducing a penalty for the length of the Newton correction $s^{(k)}$.

1.3.1 Levenberg-Marquardt algorithm

In fact, the Levenberg-Marquardt algorithm⁷ replaces the Newton correction in the original least problem (1.2.7) by a slightly modified minimization problem, namely

$$\|F'(x^{(k)}) s^{(k)} + F(x^{(k)})\|_2^2 + \mu^2 \|s^{(k)}\|_2^2 \rightarrow \min!, \quad (1.3.1)$$

where $\mu > 0$ is some chosen penalty parameter. Given a solution $s^{(k)}$ of (1.3.1), the Newton update reads the same, namely

$$x^{(k+1)} = x^{(k)} + s^{(k)}.$$

For $\mu = 0$, we obtain the Gauß-Newton method. Hence, an appropriate choice of μ will be important, we might also choose it different in each iteration, i.e., $\mu^{(k)}$.

We are now going to analyze this scheme (here for fixed μ). First, note, that

$$\left\| \begin{pmatrix} F'(x^{(k)}) \\ \mu I \end{pmatrix} s^{(k)} + \begin{pmatrix} F(x^{(k)}) \\ 0 \end{pmatrix} \right\|_2^2 = \|F'(x^{(k)}) s^{(k)} + F(x^{(k)})\|_2^2 + \mu^2 \|s^{(k)}\|_2^2,$$

⁷Named after Kenneth Levenberg (1919-1973) and Donald W. Marquardt (1929-1997). The American statistician Levenberg is the original author of the widely used nonlinear least squares fitting algorithm, which was later improved by Donald Marquardt and hence termed Levenberg-Marquardt algorithm.

which implies that the minimization problem (1.3.1) can equivalently be written as

$$\left\| \begin{pmatrix} F'(x^{(k)}) \\ \mu I \end{pmatrix} s^{(k)} + \begin{pmatrix} F(x^{(k)}) \\ 0 \end{pmatrix} \right\|_2^2 \rightarrow \min!. \quad (1.3.2)$$

Compared to the least squares problem (1.2.7) to be solved within the Gauß-Newton scheme, (1.3.2) always has a unique solution, since the matrix $\begin{pmatrix} F'(x^{(k)}) \\ \mu I \end{pmatrix}$ has full rank. The obtained Newton correction $s^{(k)} \neq 0$ satisfies the following inequalities

$$\begin{aligned} \mu^2 \|s^{(k)}\|_2^2 &\leq \|F'(x^{(k)}) s^{(k)} + F(x^{(k)})\|_2^2 + \mu^2 \|s^{(k)}\|_2^2 \\ &= \min_{s \in \mathbb{R}^n} \left\{ \|F'(x^{(k)}) s + F(x^{(k)})\|_2^2 + \mu^2 \|s\|_2^2 \right\} \\ &\leq \|F(x^{(k)})\|_2^2, \end{aligned}$$

so that

$$\|s^{(k)}\|_2 \leq \frac{\|F(x^{(k)})\|_2}{\mu}. \quad (1.3.3)$$

This shows that the parameter $\mu > 0$ is a damping of the Newton correction $s^{(k)}$ and should avoid that the size of the correction becomes “too large” (which would yield a poor approximation as explained and numerically observed above).

Under certain properties of F , one can rigorously prove the convergence (rate) of the Levenberg-Marquardt scheme, usually requiring that μ is “sufficiently large”. As a consequence, the parameter μ needs to be sufficiently large in order to guarantee convergence. On the other hand, however, choosing μ large results in a small Newton correction $\|s^{(k)}\|_2$ and therefore in many steps and thus a possibly very slow convergence. We obviously have conflicting goals.

Choice of the damping parameter. We are now going to describe a possible choice of the parameter $\mu = \mu^{(k)}$ in each step along with an update strategy for the next iteration. To this end, let $x^{(k)} \in \mathbb{R}^n$ be the current iteration and $s^{(k)} = s^{(k)}(\mu)$ the Newton correction in (1.3.1). Then, we set

$$\begin{aligned} \varepsilon_\mu &:= \frac{\|F(x^{(k)})\|_2^2 - \|F(x^{(k)} + s^{(k)})\|_2^2}{\|F(x^{(k)})\|_2^2 - \|F(x^{(k)}) + F'(x^{(k)}) s^{(k)}\|_2^2} = \frac{\|F(x^{(k)})\|_2^2 - \|F(x^{(k+1)})\|_2^2}{\|F(x^{(k)})\|_2^2 - \|F(x^{(k)}) + F'(x^{(k)}) s^{(k)}\|_2^2} \\ &=: \frac{\Delta R(x^{(k)}, s^{(k)})}{\Delta \tilde{R}(x^{(k)}, s^{(k)})} \quad (\text{i.e., the residual of the iteration over the residual of the linear model}), \end{aligned}$$

where we assume that $\Delta \tilde{R}(x^{(k)}, s^{(k)}) \neq 0$ holds (otherwise, either the Newton correction is zero and the iteration stops, or $F'(x^{(k)}) = 0$, i.e., $x^{(k)}$ would be a critical point).

Thus, ε_μ describes the change of the true residuum $\Delta R(x^{(k)}, s^{(k)})$ in relation to the change of the residuum $\Delta \tilde{R}(x^{(k)}, s^{(k)})$ of the linear model (i.e., the Gauß-Newton model). From (1.3.1) we immediately get that $\Delta \tilde{R}(x^{(k)}, s^{(k)}) \geq 0$.

Moreover, if $s^{(k)}$ is a reasonable Newton correction, it should give rise to an improvement, i.e., $\|F(x^{(k)})\|_2^2 > \|F(x^{(k+1)})\|_2^2$, so that $\Delta R(x^{(k)}, s^{(k)}) > 0$ and therefore $\varepsilon_\mu > 0$. If $x^{(k)} \rightarrow x^*$ for $k \rightarrow \infty$, then necessarily $\varepsilon_\mu \rightarrow 1$ for $\mu \rightarrow \infty$.

A concrete and implementable strategy is as follows: one chooses some values β_0, β_1 with $0 < \beta_0 < \beta_1 < 1$ (e.g. $\beta_0 = 0.3, \beta_1 = 0.9$). Then, the parameters can be verified as follows:

- $\varepsilon_\mu \leq \beta_0$: $s^{(k)}$ is not be accepted; μ is increased, e.g. $\mu \mapsto 2\mu$, and a new associated Newton correction $s^{(k)}$ is determined (shall *guarantee of convergence*).
- $\beta_0 < \varepsilon_\mu < \beta_1$: $s^{(k)}$ is accepted; use the same μ as initial condition for the computation of the next Newton correction $s^{(k+1)}$.
- $\varepsilon_\mu \geq \beta_1$: $s^{(k)}$ is accepted; a smaller value for μ (e.g. $\frac{\mu}{2}$) is used as initial condition for the computation of $s^{(k+1)}$ (shall yield *efficiency*).

Putting all pieces together yields the algorithm of the Levenberg-Marquardt method in Algorithm 1.3.1,

Algorithm 1.3.1: Levenberg-Marquardt method

Input: $x_0, F, F', \mu, \beta_0, \beta_1$

- 1: **for** $k = 0, 1, \dots$ **do**
 - 2: compute $F(x^{(k)})$, $F'(x^{(k)})$
 - 3: compute the the Newton correction $s^{(k)}$ by solving
- $$[F'(x^{(k)})^T F'(x^{(k)}) + \mu^2 I] s^{(k)} = -F'(x^{(k)})^T F(x^{(k)}).$$
- 4: **if** $\varepsilon_\mu \leq \beta_0$ **then**
 - 5: set $\mu := 2\mu$ and compute new $s^{(k)}$ according to line 3
 - 6: **else if** $\varepsilon_\mu \geq \beta_1$ **then**
 - 7: set $\mu := \frac{\mu}{2}$ and keep $s^{(k)}$
 - 8: **end if** ▷ check acceptability of the correction $s^{(k)}$
 - 9: set $x^{(k+1)} := x^{(k)} + s^{(k)}$
 - 10: **end for**
-

A realization of Algorithm 1.3.1 in Matlab can be seen in the following code:

MATLAB-function: levenberg_marquardt.m

```

1 function [x, iter] = levenberg_marquardt(F,DF,x0,mu0,beta0,beta1,maxit,
2 tol,flag)
3   if ~exist('flag','var')
4     flag = 0;
5   end
6   iter=zeros(maxit,5);
7   n=length(x0);
8   k=0;
9   mu=mu0;
10  x=x0;
11  errorold = norm(DF(x0)'*F(x0));
12  s=-(DF(x0)'*DF(x0)+mu^2*eye(n))\ (DF(x0)'*F(x0));
13  while norm(s)>tol && k<maxit
14    [s,mu] = korrektur(F,DF,x,mu,beta0,beta1);
15    x=x+s;
16    k=k+1;
17    fx=F(x);
18    errornew=norm(DF(x)'*fx);
19    iter(k,1)=norm(fx);
20    iter(k,2)=norm(s);
21    iter(k,3)=norm(errornew);
22    iter(k,4)=errornew/errorold;
23    iter(k,5)=mu;
24    errorold=errornew;
25    if (flag~=0)
26      fprintf(' %i\t %12.8f\t %12.8f\t %12.8f\t %12.8f\t %12.8f\n'
27      ,...
28      k,iter(k,1),iter(k,2),iter(k,3),iter(k,4),iter(k,5));
29  end

```

```

8     end
9     iter=iter(1:k,:) ;
0   end
1
2 function [s,mu] = korrektur(F,DF,x,mu,beta0,beta1)
3     n=length(x) ;
4     s=- (DF(x)' *DF(x)+mu^2*eye(n)) \ (DF(x)' *F(x)) ;
5     eps_mu=(F(x)' *F(x)-F(x+s)' *F(x+s))/...
6         (F(x)' *F(x)-(F(x)+DF(x)*s)' *(F(x)+DF(x)*s)) ;
7     if eps_mu <= beta0
8         [s,mu]=korrektur(F,DF,x,2*mu,beta0,beta1);
9     elseif eps_mu >= beta1
0         mu=mu/2;
1     end
2 end

```

1.3.2 Convergence analysis

The analysis basically follows the steps in Section 1.2.2. We shall assume that μ is constant for each iteration; the analysis for $\mu^{(k)}$ is similar, but technically a bit more involved. Similar to (1.2.8), we set

$$x^{(k+1)} = x^{(k)} - [F'(x^{(k)})^T F'(x^{(k)}) + \mu^2 I]^{-1} F'(x^{(k)})^T F(x^{(k)}) = \Phi_\mu(x^{(k)}),$$

where we set

$$\Phi_\mu(x) := x - [F'(x)^T F'(x) + \mu^2 I]^{-1} \underbrace{F'(x)^T F(x)}_{=\nabla g(x)}. \quad (1.3.4)$$

The matrix $F'(x)^T F'(x) + \mu I$ is symmetric and positive definite for $\mu \neq 0$ ⁸. Thus, there exists a (unique) symmetric, positive definite matrix $A_\mu = A_\mu(x^*) \in \mathbb{R}^{n \times n}$, such that

$$A_\mu^2 = F'(x^*)^T F'(x^*) + \mu^2 I \quad (\text{i.e., } A_\mu = (F'(x^*)^T F'(x^*) + \mu^2 I)^{1/2}),$$

and we define $K_\mu \in \mathbb{R}^{n \times n}$ by

$$K_\mu = K_\mu(x^*) := -A_\mu^{-1} \left(\sum_{i=1}^m \frac{F_i(x^*)}{\|F(x^*)\|_2} F_i''(x^*) \right) A_\mu^{-1}.$$

Then, the following analogue to Lemma 1.2.2 can be proven basically following the lines of the proof of Lemma 1.2.2.

Lemma 1.3.1 *Let $F : \mathbb{R}^n \rightarrow \mathbb{R}^m$, $g(x) := \frac{1}{2}F(x)^T F(x)$ and $\nabla g(x^*) = 0$ for some $x^* \in \mathbb{R}^n$ as well as $\mu \neq 0$. Then,*

$$g''(x^*) = A_\mu(I - (\mu^2 A_\mu^{-2} + \|F(x^*)\|_2 K_\mu)) A_\mu, \quad (1.3.5)$$

$$\Phi'_\mu(x^*) = A_\mu^{-1}(\mu^2 A_\mu^{-2} + \|F(x^*)\|_2 K_\mu) A_\mu. \quad (1.3.6)$$

If x^* is a local maximum or a saddle point of g , then $\|\Phi'_\mu(x^*)\| \geq 1$ for any operator norm. \square

We can draw similar conclusions as in Corollary 1.3.2.

Corollary 1.3.2 *Under the assumptions of Lemma 1.3.1, it holds:*

⁸This is a major advantage as compared to the Gauß-Newton method, where we had to assume that $\text{rank}(F'(x^*)) = n$.

- i) For non-perfect data (i.e., $F(x^*) \neq 0$) we have the following: If the Levenberg-Marquardt method converges, the rate of convergence is in general not more than linear.
- ii) If the critical point x^* is a local maximum or a saddle point, the scheme is repelling.

Proof. If $F(x^*) \neq 0$, we get that $\Phi'_\mu(x^*) \neq 0$, which implies that the convergence is in general at most linear. \square

In analogy to Theorem 1.2.3 we use the energy-type norm $\|x\|_{A_\mu} := \|A_\mu x\|_2$ and the induced operator norm. We leave the proof of the next statement as an exercise and refer to [5].

Theorem 1.3.3 *Under the assumptions of Lemma 1.3.1, the iteration function Φ_μ in (1.3.4) of the Levenberg-Marquardt method satisfies*

$$\|\Phi'(x^*)\|_{A_\mu} = \varrho(I - C_\mu), \quad \text{where } C_\mu := A_\mu^{-1} g''(x^*) A_\mu^{-1}.$$

Moreover, $\|\Phi'(x^*)\|_{A_\mu} < 1$ provided that $\mu > \frac{1}{\sqrt{2}} \|g''(x^*)\|_2^{1/2}$. \square

This latter theorem states that the Levenberg-Marquardt scheme converges towards a local minimum if the damping parameter μ is chosen sufficiently large. Finally, we give a global (monotone) convergence statement.

Lemma 1.3.4 *Let $(x^{(k)})_{k \in \mathbb{N}_0} \subset \mathbb{R}^n$ such that $\nabla g(x^{(k)}) \neq 0$ and $g \in C^2(\mathcal{U}^{(k)})$ for all k , where $\mathcal{U}^{(k)}$ is a neighborhood of $x^{(k)}$. Let $s^{(k)}$ be the Levenberg-Marquardt correction. Then, there is a $\mu_0 > 0$ such that*

$$g(x^{(k)} + s^{(k)}) < g(x^{(k)})$$

for all $\mu \geq \mu_0$.

Proof. Due to (1.3.3), we can choose some $\tilde{\mu}_0 > 0$ such that the line segment $\{x^{(k)} + ts^{(k)} : t \in [0, 1]\} \subset \mathcal{U}^{(k)}$ for all $\mu \geq \tilde{\mu}_0$. Then, the Taylor expansion yields

$$g(x^{(k)} + s^{(k)}) = g(x^{(k)}) + \nabla g(x^{(k)})^T s^{(k)} + \mathcal{O}(\|s^{(k)}\|_2^2).$$

Next, we can easily derive the normal equation of (1.3.1), i.e.,

$$\underbrace{(F'(x^{(k)})^T F'(x^{(k)}) + \mu^2 I)}_{=: B_\mu^{-1}} s^{(k)} = -\nabla g(x^{(k)}),$$

so that

$$g(x^{(k)} + s^{(k)}) - g(x^{(k)}) = -[\nabla g(x^{(k)})^T B_\mu \nabla g(x^{(k)})] + \mathcal{O}(\|s^{(k)}\|_2^2).$$

The matrix B_μ is symmetric and positive definite, and for each $\lambda \in \sigma(B_\mu^{-1})$ we have that

$$\mu^2 \leq \lambda \leq \mu^2 + \|F'(x^{(k)})\|_2^2.$$

This implies for all $z \in \mathbb{R}^n$ that

$$(\mu^2 + \|F'(x^{(k)})\|_2^2)^{-1} \|z\|_2^2 \leq z^T B_\mu z \leq \mu^{-2} z^T z,$$

so that (recall that B_μ is s.p.d.) $\|B_\mu^{1/2} z\|_2^2 \leq \mu^{-2} \|z\|_2^2$ and $\|B_\mu z\|_2^2 \leq \mu^{-2} \|B_\mu^{1/2} z\|_2^2 \leq \mu^{-4} \|z\|_2^2$, which then implies

$$\|s^{(k)}\|_2^2 = \|B_\mu \nabla g(x^{(k)})\|_2^2 \leq \frac{1}{\mu^4} \|\nabla g(x^{(k)})\|_2^2.$$

Then, we obtain for some constant $0 < C < \infty$

$$\begin{aligned} \nabla g(x^{(k)})^T B_\mu \nabla g(x^{(k)}) + \mathcal{O}(\|s^{(k)}\|_2^2) &\geq (\mu^2 + \|F'(x^{(k)})\|_2^2)^{-1} \|\nabla g(x^{(k)})\|_2^2 - C \|s^{(k)}\|_2^2 \\ &\geq (\mu^2 + \|F'(x^{(k)})\|_2^2)^{-1} \|\nabla g(x^{(k)})\|_2^2 - \frac{C}{\mu^4} \|\nabla g(x^{(k)})\|_2^2 \\ &\geq \frac{\|\nabla g(x^{(k)})\|_2^2}{\mu^2} \left(\underbrace{\left(1 + \frac{\|F'(x^{(k)})\|_2^2}{\mu^2} \right)^{-1}}_{\substack{\mu \rightarrow \infty \\ \longrightarrow 1}} - \underbrace{\frac{C}{\mu^2}}_{\substack{\mu \rightarrow \infty \\ \longrightarrow 0}} \right) \\ &\geq 0, \end{aligned}$$

for all $\mu \geq \mu_0$, if we choose $\mu_0 \geq \tilde{\mu}_0$ sufficiently large. This concludes the proof. \square

1.3.3 Examples – again

We are now reconsidering the above examples, with specific focus on those that did not work well with the Gauß-Newton method.

Example 1.3.5 (Regression Circle – Levenberg-Marquardt) We reconsider Example 1.1.1 and Example 1.2.5, respectively. For the random distortions, we now choose $\Delta_r = 0.1$, for which Gauß-Newton almost always diverges. We call the Levenberg-Marquardt routine with $\mu^{(0)} = 1$, $\beta_0 = 0.3$ and $\beta_1 = 0.9$. The results are shown in Figure 1.9 and Table 1.6. We clearly see that the by far larger data error still results in a good regression circle. We also see that the damping parameter is monotonically decreased and we get a linear convergence with an error reduction factor of about 0.02, i.e., extremely fast convergence. We recommend to modify also other parameters in order to investigate the robustness of the method. \diamond

k	$\ F(x^{(k)})\ _2$	$\ s^{(k)}\ _2$	$\ \nabla g(x^{(k)})\ _2$	ϱ_k	$\mu^{(k)}$
1	9.46976717	2.20803151	92.16849948	0.41024519	1.00000000
2	2.18693870	0.91203981	0.92547270	0.01004110	0.50000000
3	2.18497646	0.00925315	0.00259026	0.00279886	0.25000000
4	2.18497644	0.00003238	0.00002494	0.00962898	0.12500000
5	2.18497644	0.00000050	0.00000052	0.02078017	0.06250000
6	2.18497644	0.00000001	0.00000001	0.02059675	0.03125000
7	2.18497644	0.00000000	0.00000000	0.02053921	0.03125000
8	2.18497644	0.00000000	0.00000000	0.02058798	0.06250000

Table 1.6: Convergence history of the Levenberg-Marquardt method for the regression circle problem in Example 1.1.1.

Example 1.3.6 (Density of the Gamma-distribution – Levenberg-Marquardt) Next, we reconsider Example 1.1.3 and Example 1.2.6, respectively. We choose a larger distortion, namely $\sigma = 0.02$ and much more points, i.e., $N_{\text{data}} = 50$ (as opposed to $\sigma = 0.015$ and $N_{\text{data}} = 20$ in Example 1.2.6 with Gauß-Newton), which means that we have much more redundancy in the data and a significantly larger model error. We choose the same values $\mu^{(0)} = 1$, $\beta_0 = 0.3$ and $\beta_1 = 0.9$ as before. The results are shown in Figure 1.10 and Table 1.7. We observe linear convergence with an error reduction factor of about 0.11. We also see that the Levenberg-Marquardt solution is an almost perfect fit to the original data. \diamond

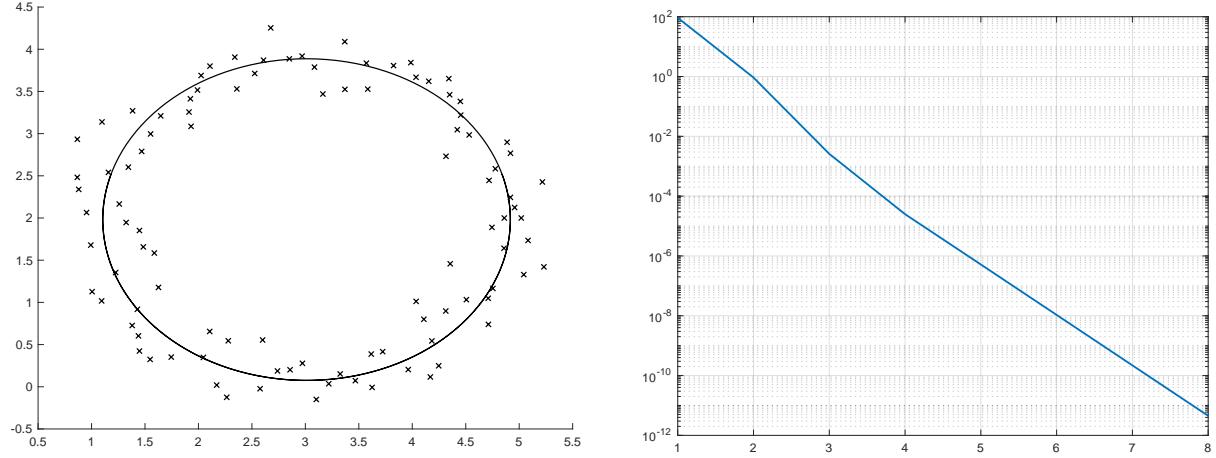


Figure 1.9: Result (left) and convergence history (right) of the Levenberg–Marquardt method for Example 1.1.1.

k	$\ F(x^{(k)})\ _2$	$\ s^{(k)}\ _2$	$\ \nabla g(x^{(k)})\ _2$	ϱ_k	$\mu^{(k)}$
1	0.69773093	0.47522936	0.78417043	0.26361508	0.50000000
2	0.39413101	0.39946039	0.13666776	0.17428324	0.25000000
3	0.36037467	0.21036022	0.00890274	0.06514147	0.12500000
4	0.35624087	0.18763825	0.00526465	0.59135220	0.06250000
5	0.35519396	0.10195680	0.00151564	0.28788927	0.03125000
6	0.35510428	0.02010522	0.00006013	0.03967196	0.01562500
7	0.35509899	0.00157473	0.00000587	0.09763322	0.00781250
8	0.35509844	0.00016631	0.00000067	0.11485188	0.00390625
9	0.35509838	0.00001913	0.00000008	0.11208663	0.00195312
10	0.35509837	0.00000215	0.00000001	0.11236262	0.00097656
11	0.35509837	0.00000024	0.00000000	0.11218452	0.00048828
12	0.35509837	0.00000003	0.00000000	0.11219317	0.00024414
13	0.35509837	0.00000000	0.00000000	0.11217984	0.00012207
14	0.35509837	0.00000000	0.00000000	0.11220367	0.00006104
15	0.35509837	0.00000000	0.00000000	0.11254809	0.00195312

Table 1.7: Convergence history of the Levenberg–Marquardt method for Example 1.1.3.

Example 1.3.7 (Damped oscillation – Levenberg–Marquardt) Finally, we reconsider Example 1.1.4 and Example 1.2.7, respectively. We choose exactly the same data as in Example 1.2.7 and again $\mu^{(0)} = 1$, $\beta_0 = 0.3$ and $\beta_1 = 0.9$ as before. The results are shown in Figure 1.11 and Table 1.8. We observe a (slightly) slower convergence than for Gauß–Newton (17 instead of 14 iterations, but also with a smaller residual). Hence, if Gauß–Newton converges, it might be faster than Levenberg–Marquardt. \diamond

Example 1.2.8 has shown very good results with the Gauß–Newton method, so that there is no real need to compare with Levenberg–Marquardt.

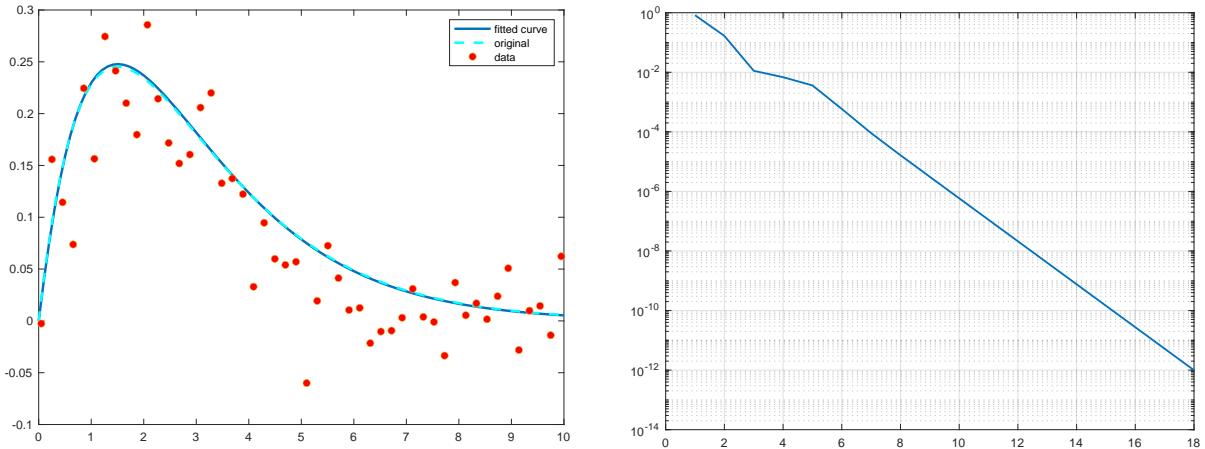


Figure 1.10: Result (left) and convergence history (right) of the Levenberg-Marquardt method for Example 1.1.3.

k	$\ F(x^{(k)})\ _2$	$\ s^{(k)}\ _2$	$\ \nabla g(x^{(k)})\ _2$	ϱ_k	$\mu^{(k)}$
1	0.32525446	0.07776144	0.07640683	0.52544464	0.50000000
2	0.30808822	0.12268242	0.02983838	0.39051987	0.25000000
3	0.27195682	0.36131722	0.03306729	1.10821323	0.12500000
4	0.15788822	0.84368651	0.04314783	1.30484917	0.06250000
5	0.09172928	0.44177728	0.01465839	0.33972483	0.03125000
6	0.08923680	0.08061464	0.00215650	0.14711723	0.03125000
7	0.08905771	0.02783070	0.00076669	0.35552375	0.03125000
8	0.08903774	0.00892797	0.00023599	0.30780758	0.03125000
9	0.08903563	0.00298250	0.00007859	0.33302054	0.03125000
10	0.08903540	0.00096699	0.00002544	0.32375509	0.03125000
11	0.08903538	0.00031863	0.000000838	0.32917071	0.03125000
12	0.08903538	0.00010395	0.00000273	0.32643606	0.03125000
13	0.08903538	0.00003411	0.00000090	0.32801355	0.03125000
14	0.08903538	0.00001116	0.00000029	0.32713724	0.03125000
15	0.08903538	0.00000366	0.00000010	0.32764722	0.03125000
16	0.08903538	0.00000120	0.00000003	0.32735426	0.03125000
17	0.08903538	0.00000039	0.00000001	0.32752527	0.03125000

Table 1.8: Convergence history of the Levenberg-Marquardt method for Example 1.1.3.

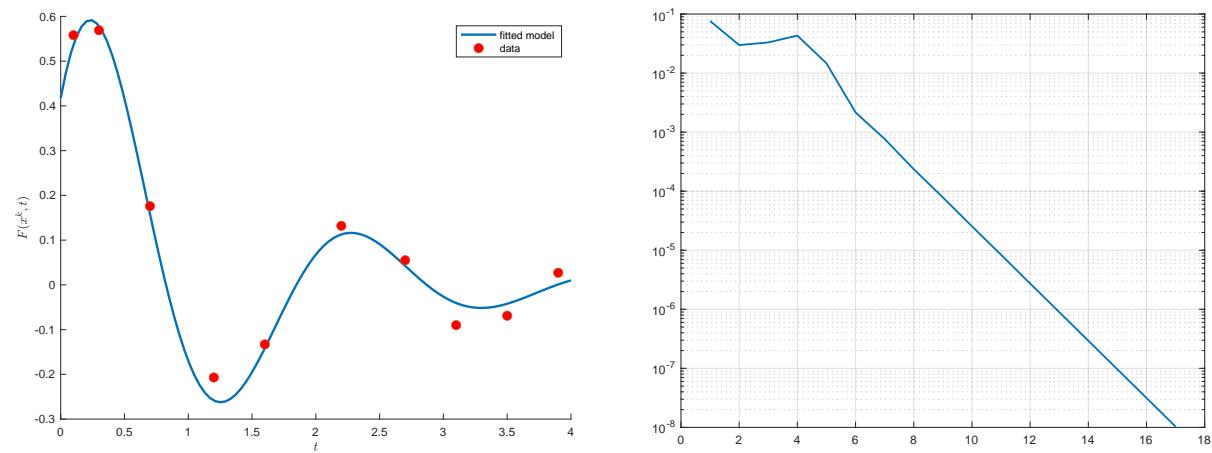


Figure 1.11: Result (left) and convergence history (right) of the Levenberg-Marquardt method for Example 1.1.4.

2 UNCONSTRAINED OPTIMIZATION

We are now going to discuss minimization problems of the form

$$\text{minimize } f(x) \text{ w.r.t. } x \in \mathbb{R}^n, \quad (2.0.1)$$

where $f : \mathbb{R}^n \rightarrow \mathbb{R}$ is a given so-called *objective function* (or *cost function*). This problem is called *unconstrained optimization problem* since all $x \in \mathbb{R}^n$ are allowed, i.e., the argument is not restricted by some constraints. Such a problem is also known as *nonlinear optimization* since f is typically a nonlinear function, the special case of a linear objective function yielding to *linear optimization* is treated by different methods.

We could think of the determination of the optimal distribution of n competitive resources x_1, \dots, x_n that are available in an infinite amount and do not obey any special properties. If those resources would not be infinite, than the minimum of the objective function f is only allowed in a subset $\Omega \subset \mathbb{R}^n$, which means that we are facing the problem

$$\text{minimize } f(x) \text{ w.r.t. } x \in \Omega \subsetneq \mathbb{R}^n. \quad (2.0.2)$$

A problem of the form (2.0.2) is called a *constrained optimization problem*. Such problems will be discussed in Chapter 3 below.

Example 2.0.1 (Nonlinear least squares) With $f(x) := \|F(x)\|_2^2$ for some $F : \mathbb{R}^n \rightarrow \mathbb{R}^m$, we obtain a nonlinear least squares problem as a special case of an unconstrained optimization problem. \diamond

Let us introduce a classical (quite challenging) example of an unconstrained minimization problem, which is also on the cover of this lecture notes.

Example 2.0.2 (Rosenbrock function) The *Rosenbrock function*¹ $f : \mathbb{R}^2 \rightarrow \mathbb{R}$ is defined as

$$f(x) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2, \quad x = (x_1, x_2)^T \in \mathbb{R}^2,$$

see Figure 2.1. It has a unique minimum at $x^* = (1, 1)^T$ with value $f(x^*) = 0$ and is often used as a benchmark to test numerical methods for minimization problems since it has steep gradients in one direction and very flat ones on other directions. Moreover, these directions are not aligned with the coordinate axes, but ‘banana-shaped’. \diamond

Example 2.0.3 (Himmelblau function) The *Himmelblau function*² $f : \mathbb{R}^2 \rightarrow \mathbb{R}$ is defined as

$$f(x) = (x_1^2 + x_2 - 11)^2 + (x_1 + x_2^2 - 7)^2, \quad x = (x_1, x_2)^T \in \mathbb{R}^2,$$

see Figure 2.2. This is a 4th order polynomial with four local minima which are also global minima with minimal value 0. Moreover, the function has 4 saddle points and one local maximum. The specific challenge for numerical methods lies in the appearance of the local minima. \diamond

Example 2.0.4 (Bazaraa-Shetty function) The *Bazaraa-Shetty function*³ $f : \mathbb{R}^2 \rightarrow \mathbb{R}$ is defined as

$$f(x) = (x_1 - 2)^4 + (x_1 - 2x_2)^2, \quad x = (x_1, x_2)^T \in \mathbb{R}^2,$$

see Figure 2.3. This is a 4th order polynomial with a single global minimum at $x^* = (2, 1)^T$. However, the Hessian matrix is singular, which is a challenge for numerical schemes assuming that the Hessian is symmetric positive definite at x^* . \diamond

¹Howard Harry Rosenbrock (1920-2010)

²David Mautner Himmelblau (1924-2011)

³Mokhtar S. Bazaraa (?-), Marakada “Mike” Chitharanjan Shetty (?-2012)

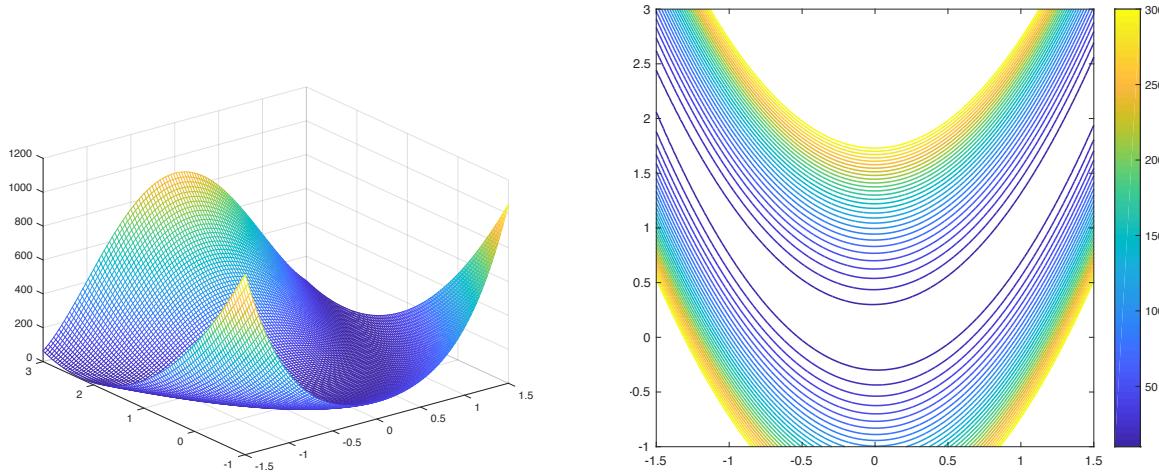


Figure 2.1: Rosenbrock function: 3D view (left), isolines (right).

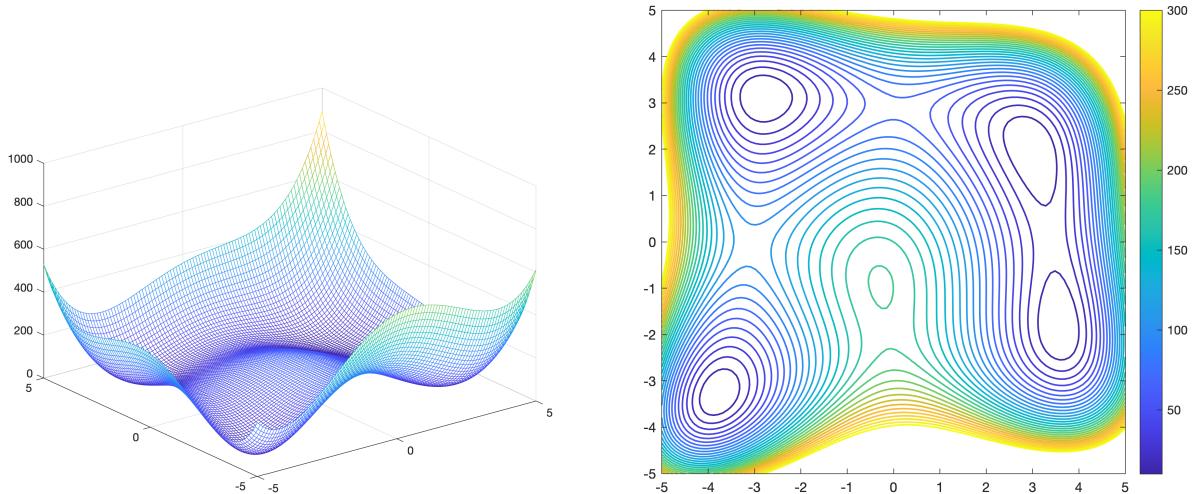


Figure 2.2: Himmelblau function: 3D view (left), isolines (right).

Example 2.0.5 The function

$$f(x_1, \dots, x_5) := 2x_1^2 + 2x_2^2 + x_3^2 + x_4^2 + 0.5x_5^2 - 4x_1 - 4x_2 - 2x_3 - 2x_4 - x_5 + 6.5$$

is a quadratic polynomial and can be written as $f(x) = \frac{1}{2}x^T Ax + a^T x$, $x \in \mathbb{R}^5$, with $A \in \mathbb{R}^{5 \times 5}$ being symmetric and positive definite. The global minimum $x^* = (1, 1, 1, 1, 1)^T$ satisfies $f(x^*) = 0$. \diamond

Example 2.0.6 (Dixon-Price function) The *Dixon-Price function*⁴ $f : \mathbb{R}^d \rightarrow \mathbb{R}$ is defined as

$$f(x) = (x_1 - 2)^2 + \sum_{i=2}^d i(2x_i^2 - x_{i-1})^2, \quad x = (x_1, \dots, x_d)^T \in \mathbb{R}^d,$$

is a 4th polynomial with global minimum $x_i^* = 2^{-\frac{2^{i-2}}{2^i}}$, $i = 1, \dots, d$, and $f(x^*) = 0$. \diamond

One may find much more examples e.g. on the website <https://www.sfu.ca/~ssurjano/optimization.html>.

⁴Laurence C.W. Dixon (1934-2011), Richard C. Price (?)

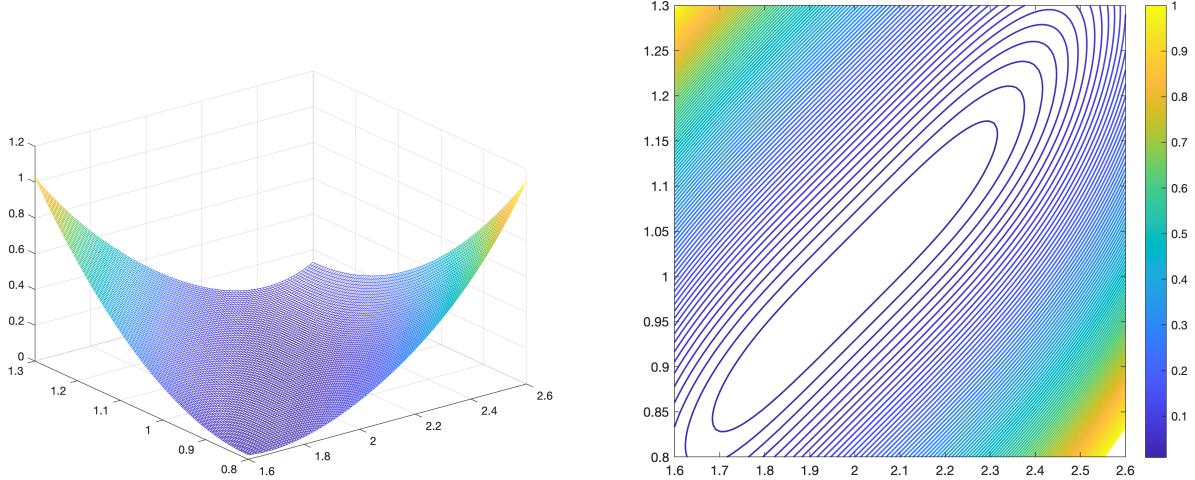


Figure 2.3: Bazaraa-Shetty function: 3D view (left), isolines (right).

Remark 2.0.7 It is clear that maximization problems can easily be rewritten as minimization problems of the form (2.0.1) just by switching the sign of the objective function.

2.1 OPTIMALITY CONDITIONS AND CONVEXITY

We describe necessary and sufficient optimality conditions for minimizing functions $f : \mathbb{R}^n \rightarrow \mathbb{R}$. We mainly follow [25].

Theorem 2.1.1 (First order necessary optimality condition) Let $f \in C^1(\mathcal{U})$ for some open set $\mathcal{U} \subset \mathbb{R}^n$ and let $x^* \in \mathcal{U}$ be a local minimum of f . Then, $\nabla f(x^*) = 0$.

Proof. For $d \in \mathbb{R}^n$ and $h > 0$ consider the differential quotient

$$\Delta_h(f; d) := \frac{1}{h} (f(x^* + hd) - f(x^*)).$$

By assumption, we have that $\Delta_h(f; d) \geq 0$ for all $d \in \mathbb{R}^n$ and $h > 0$ sufficiently small. Taking the limit yields

$$\nabla f(x^*)^T d = \lim_{h \rightarrow 0} \Delta_h(f; d) \geq 0.$$

Choosing $d := -\nabla f(x^*)$ yields $0 \leq -\nabla f(x^*)^T \nabla f(x^*) = -\|\nabla f(x^*)\|^2 \leq 0$, thus $\nabla f(x^*) = 0$. \square

Definition 2.1.2 Let $f \in C^1(\mathcal{U})$ for some open set $\mathcal{U} \subset \mathbb{R}^n$. We call $x^* \in \mathcal{U}$ a stationary point (or critical point) of f if $\nabla f(x^*) = 0$.

Definition 2.1.3 A stationary point of f which is neither a local maximum nor a local minimum is called saddle point of f .

Theorem 2.1.4 (Second order necessary optimality conditions) Let $f \in C^2(\mathcal{U})$ on some open set $\mathcal{U} \subset \mathbb{R}^n$ and let x^* be a local minimum of f . Then,

- (i) $\nabla f(x^*) = 0$, i.e., x^* is a stationary point of f ;
- (ii) The Hessian matrix $\nabla^2 f(x^*) = f''(x^*)$ is positive semidefinite, i.e., $d^T \nabla^2 f(x^*) d \geq 0$ for all $d \in \mathbb{R}^n$.

Proof. (i) is just Theorem 2.1.1. For the proof of (ii) let $d \in \mathbb{R}^n \setminus \{0\}$ be arbitrary. For $\tau = \tau(d) > 0$ sufficiently small (in particular small enough that x^* is the only minimum of f in $B_\tau(x^*) := \{x \in \mathbb{R}^n : \|x - x^*\|_2 \leq \tau\}$), we consider Taylor's expansion⁵ for $t \in (0, \tau)$

$$\begin{aligned} 0 \leq f(x^* + td) - f(x^*) &= t \nabla f(x^*)^T d + \frac{t^2}{2} d^T \nabla^2 f(x^*) d + \varrho(t) \\ &= \frac{t^2}{2} d^T \nabla^2 f(x^*) d + \varrho(t), \end{aligned}$$

since $\nabla f(x^*) = 0$, where the remainder term satisfies $\varrho(t) = o(t^2)$. In view of (i) this yields

$$d^T \nabla^2 f(x^*) d \geq (-2) \frac{\varrho(t)}{t^2} \xrightarrow{t \rightarrow 0} 0,$$

which proves the claim. \square

Hence, so far we have seen *necessary* conditions. We are now considering a *sufficient* condition, for which we need second order derivatives.

Theorem 2.1.5 (Second order sufficient optimality conditions) *Let $f \in C^2(\mathcal{U})$ for some open set $\mathcal{U} \subset \mathbb{R}^n$ and let $x^* \in \mathcal{U}$ be such that*

- (i) x^* is a stationary point of f , i.e., $\nabla f(x^*) = 0$, and
- (ii) the Hessian matrix $\nabla^2 f(x^*)$ is positive definite, i.e.,

$$d^T \nabla^2 f(x^*) d > 0 \quad \text{for all } d \in \mathbb{R}^n \setminus \{0\}.$$

Then, x^* is a strict local minimum of f , i.e., $f(x^*) < f(x)$ for all $x \in \mathcal{U} \setminus \{x^*\}$.

Proof. Due to assumption (ii), there exists some $\mu > 0$ such that $d^T \nabla^2 f(x^*) d \geq \mu \|d\|^2$ for all $d \in \mathbb{R}^n$. Next, we use Taylor's expansion and (i) to obtain

$$\begin{aligned} f(x^* + d) - f(x^*) &= d^T \nabla f(x^*) + \frac{1}{2} d^T \nabla^2 f(x^*) d + o(\|d\|^2) \\ &= \frac{1}{2} d^T \nabla^2 f(x^*) d + o(\|d\|^2) \end{aligned}$$

for $d \in B_\varepsilon(0)$ and $\varepsilon > 0$ sufficiently small. Furthermore,

$$f(x^* + d) - f(x^*) \geq \frac{\mu}{2} \|d\|^2 + o(\|d\|^2) \geq \frac{\mu}{2} \|d\|^2 > 0$$

for $d \neq 0$. Hence, x^* is a strict local minimum. \square

We will now consider a specific class of functions for which we can show the above optimality conditions.

Definition 2.1.6 *Let $X \subset \mathbb{R}^n$.*

- (a) *The set X is called convex if $(1 - \lambda)x + \lambda y \in X$ for all $x, y \in X$ and all $\lambda \in [0, 1]$.*
- (b) *The set*

$$\text{conv}(X) := \left\{ \sum_{i=1}^k \lambda_i x_i : k \in \mathbb{N}, x_i \in X, \lambda_i \geq 0, \sum_{i=1}^k \lambda_i = 1 \right\}$$

is called the convex hull of X .

Definition 2.1.7 *Let $X \subset \mathbb{R}^n$ be convex. Then, the function $f : X \rightarrow \mathbb{R}$ is called*

⁵Brook Taylor (1685-1731)

(a) convex if

$$f((1-\lambda)x + \lambda y) \leq (1-\lambda)f(x) + \lambda f(y)$$

for all $x, y \in X$ and all $\lambda \in [0, 1]$;

(b) strictly convex if

$$f((1-\lambda)x + \lambda y) < (1-\lambda)f(x) + \lambda f(y)$$

for all $x, y \in X$, $x \neq y$ and all $\lambda \in [0, 1]$;

(c) uniformly convex if there is some $\mu > 0$ such that

$$f((1-\lambda)x + \lambda y) + \mu \lambda(1-\lambda) \|y - x\|^2 \leq (1-\lambda)f(x) + \lambda f(y)$$

for all $x, y \in X$ and all $\lambda \in [0, 1]$.

We start by first-order characterizations of convex functions.

Theorem 2.1.8 Let $f \in C^1(X)$ for some open, convex $X \subset \mathbb{R}^n$. Then, f is

(i) convex if and only if

$$\nabla f(x)^T (y - x) \leq f(y) - f(x) \quad \text{for all } x, y \in X; \quad (2.1.1)$$

(ii) strictly convex if and only if

$$\nabla f(x)^T (y - x) < f(y) - f(x) \quad \text{for all } x, y \in X \text{ with } x \neq y;$$

(iii) uniformly convex if and only if there is some $\mu > 0$ such that

$$\nabla f(x)^T (y - x) + \mu \|y - x\|^2 \leq f(y) - f(x) \quad \text{for all } x, y \in X.$$

Proof. We will prove only (i), for the other two statements, we refer to [25, Satz 6.3]. Let f be convex and $x, y \in X$. Then, $x + \lambda(y - x) \in X$ for all $\lambda \in [0, 1]$ and

$$\frac{1}{\lambda}(f(x + \lambda(y - x)) - f(x)) \leq \frac{1}{\lambda}((1-\lambda)f(x) + \lambda f(y) - f(x)) = f(y) - f(x).$$

Taking the limit $\lambda \rightarrow 0+$ yields

$$\nabla f(x)^T (y - x) = \lim_{\lambda \rightarrow 0+} \frac{1}{\lambda}(f(x + \lambda(y - x)) - f(x)) \leq f(y) - f(x),$$

i.e., (2.1.1). On the other hand, let us now assume (2.1.1). Let $x, y \in X$ and $\lambda \in [0, 1]$ be arbitrary and set $x_\lambda := (1-\lambda)x + \lambda y$. Then, using (2.1.1),

$$\begin{aligned} (1-\lambda)f(x) + \lambda f(y) - f(x_\lambda) &= (1-\lambda)(f(x) - f(x_\lambda)) + \lambda(f(y) - f(x_\lambda)) \\ &\geq (1-\lambda)\nabla f(x_\lambda)^T(x - x_\lambda) + \lambda \nabla f(x_\lambda)^T(y - x_\lambda) \\ &= \nabla f(x_\lambda)^T((1-\lambda)(x - x_\lambda) + \lambda(y - x_\lambda)) \\ &= \nabla f(x_\lambda)^T((1-\lambda)x + \lambda y - x_\lambda) = 0, \end{aligned}$$

which means that f is convex, so that (i) is proven. \square

For completeness, we also quote second-order characterizations, but omit the proof.

Theorem 2.1.9 Let $f \in C^2(X)$ for some open, convex $X \subset \mathbb{R}^n$. Then, f is

(i) convex if and only if $\nabla^2 f(x)$ is positive semidefinite for all $x \in X$;

(ii) strictly convex if $\nabla^2 f(x)$ is positive definite for all $x \in X$;

(iii) uniformly convex if and only if $\nabla^2 f(x)$ is uniformly positive definite for all $x \in X$, i.e., there is some $\mu > 0$ such that $d^T \nabla^2 f(x) d > \mu \|d\|^2$ for all $d \in \mathbb{R}^n$.

Proof. See [25, Satz 6.4]. □

Now, we are in position to show the optimality conditions for convex functions.

Theorem 2.1.10 *Let $f : X \rightarrow \mathbb{R}$ be convex on some convex set $X \subset \mathbb{R}^n$. Then,*

- (i) *every local minimum of f in X is a global one;*
- (ii) *if f is strictly convex, then there is at most one local minimum of f in X which is a strict global minimum of f in X – in case of existence;*
- (iii) *if X is open, $f \in C^1(X)$ and $x^* \in X$ is a stationary point, then x^* is a global minimum of f in X .*

Proof. (i) Let $x^* \in X$ be a local minimum and assume that there is some $x \in X$ with $f(x) < f(x^*)$. Then, we have by convexity for all $t \in (0, 1]$ that

$$f(x^* + t(x - x^*)) \leq (1 - t)f(x^*) + t f(x) < (1 - t)f(x^*) + t f(x^*) = f(x^*),$$

which contradicts x^* being a local minimum.

- (ii) Assume that $x^* \neq y^*$ with $x^*, y^* \in X$ are local minima. By (i), both x^* and y^* are also global minima, i.e., $f(x) \geq f(x^*) = f(y^*)$ for all $x \in X$. Choose $x := \frac{1}{2}(x^* + y^*) \in X$ and the strict convexity yields $f(x) = f(\frac{1}{2}x^* + \frac{1}{2}y^*) < \frac{1}{2}(f(x^*) + f(y^*)) = f(x^*) \leq f(x)$, which is a contradiction.
- (iii) By Theorem 2.1.8 (i), we have for all $x \in X$ that $f(x) - f(x^*) \geq \nabla f(x^*)^T (x - x^*) = 0$, i.e., x^* is a global minimum of f in X . □

Having the main theoretical background at hand, we can now turn towards numerical methods for solving such nonlinear unconstrained optimization problems.

2.2 DIRECT SEARCH METHODS

We start by presenting schemes that only rely on evaluations of the objective function at certain points – and no further assumptions such as regularity or evaluation of derivatives. Those methods are thus also called *direct search* or *derivative-free methods*. So-called *gradient descent algorithms* requiring the evaluation of the derivative(s) of f (the gradient) will be considered later. In general, gradient-based methods show better convergence as compared to direct search methods.

Obviously, direct search algorithms are particularly useful for functions that are not differentiable or functions whose derivatives are either not known or at least difficult to compute. Furthermore, derivative-free methods are appropriate for oscillating objective functions f (e.g. random functions), where derivatives might be extremely rough functions.

2.2.1 Hooke-Jeeves method

The algorithm invented by Hooke and Jeeves (1961,[11])⁶, also known as *pattern search* is an iterative scheme to approximate the desired minimum $x^* \in \mathbb{R}^n$ of f by searching along the coordinate axes. Starting from some approximation $x^{(k)} \in \mathbb{R}^n$ of x^* one computes the next (and hopefully better) approximation $x^{(k+1)}$ by two steps, namely

1. the exploration step;
2. the progression step.

We shall describe these steps in more detail.

⁶Robert Hooke (1635-1703), Thomas A. Jeeves (?)

Exploration Step. In the *exploration step* one checks along the orthogonal coordinate directions starting from $x^{(k)}$ in which coordinate direction the objective function f decreases most: Let $e_1, \dots, e_n \in \mathbb{R}^n$ denote the canonical basis vectors of \mathbb{R}^n , $h = (h_1, \dots, h_n)^T \in \mathbb{R}_+^n$ the vector of the step sizes $h_i > 0$ and $x^{(k)}$ some approximation to x^* . Then, based upon $x^{(k)}$, we check for all coordinate directions if

$$f(x^{(k)} + h_j e_j) < f(x^{(k)})$$

holds for $j = 1, \dots, n$. In case of success for the index j , the current approximation $x^{(k)}$ is replaced by vector $x^{(k)} + h_j e_j$. Otherwise, if

$$f(x^{(k)} + h_j e_j) \geq f(x^{(k)}),$$

then one checks the vector $x^{(k)} - h_j e_j$. If

$$f(x^{(k)} - h_j e_j) < f(x^{(k)})$$

holds, $x^{(k)}$ is replaced by $x^{(k)} - h_j e_j$. Then, we continue with the next index $j + 1$. After testing all n directions (which means a total of at most $2n + 1$ function evaluations), we obtain some vector $y^{(k)}$ with $f(y^{(k)}) \leq f(x^{(k)})$. At this point we have two possibilities:

- i) Case 1: $y^{(k)} = x^{(k)}$, i.e., no improvement: In this case the step size h_i is halved, $h \mapsto \frac{1}{2}h$ and we perform a new exploration step starting from $x^{(k)}$ with this new step size.
- ii) Case 2: $y^{(k)} \neq x^{(k)}$, i.e., we actually found a direction $y^{(k)} - x^{(k)}$ in which the objective function decreases. We perform a *progression step* to be explained next.

Progression Step. The *progression step* consists of a search along the line given by $x^{(k)}$ and $y^{(k)}$, in particular beyond $y^{(k)}$, i.e., along $t \mapsto y^{(k)} + t(y^{(k)} - x^{(k)})$ for $t > 0$. A typical choice is $t = 1$ and we set

$$w^{(k)} := 2y^{(k)} - x^{(k)} = x^{(k)} + 2(y^{(k)} - x^{(k)}).$$

From $w^{(k)}$, we perform another exploration step with the current step size. If this yields an additional improvement, namely some $z^{(k)}$ with $f(z^{(k)}) < f(y^{(k)})$, we set $x^{(k+1)} := z^{(k)}$, otherwise we keep $x^{(k+1)} := y^{(k)}$.

Stopping criterium. A standard stopping criterium is

$$\|x^{(k+1)} - x^{(k)}\|_\infty \leq \varepsilon$$

for a given tolerance $\varepsilon > 0$.

This concludes the description of the Hooke-Jeeves method. The corresponding algorithm is shown in Algorithm 2.2.1.

Algorithm 2.2.1: Hooke-Jeeves method

Input: objective function $f \in C(\mathbb{R}^n)$ and

- initial vector $x^{(0)} \in \mathbb{R}^n$
- vector of initial step sizes $h = (h_1, \dots, h_n)^T \in \mathbb{R}_+^n$
- tolerance $\varepsilon > 0$

- 1: $k = 0$
- 2: **while** $\|h\|_\infty > \varepsilon$ **do**
- 3: $y^{(k)} = \text{Exploration}(f, x^{(k)}, h)$

```

4:   if  $y^{(k)} = x^{(k)}$  then
5:      $h \rightarrow \frac{1}{2}h$ 
6:   else
7:      $w^{(k)} = 2y^{(k)} - x^{(k)}$ 
8:      $z^{(k)} = \text{Exploration}(f, w^{(k)}, h)$ 
9:     if  $f(z^{(k)}) < f(y^{(k)})$  then
10:       $x^{(k+1)} = z^{(k)}$ 
11:    else
12:       $x^{(k+1)} = y^{(k)}$ 
13:    end if
14:     $k \rightarrow k + 1$ 
15:  end if
16: end while

```

The procedure **Exploration** called in line 8 of Algorithm 2.2.1 is detailed in Algorithm 2.2.2.

Algorithm 2.2.2: Exploration(f, x, h)

Input: $f \in C(\mathbb{R}^n)$, $x \in \mathbb{R}^n$, $h \in \mathbb{R}_+^n$

```

1: for  $j = 1 : n$  do
2:   if  $f(x + h_j e_j) < f(x)$  then
3:      $x = x + h_j e_j$ 
4:   else if  $f(x - h_j e_j) < f(x)$  then
5:      $x = x - h_j e_j$ 
6:   end if
7: end for

```

Remark 2.2.1 The initial step sizes $h_j > 0$ should be chosen in such a way, that the values

$$|f(x^{(k)} \pm h_j e_j) - f(x^{(k)})|, j \in \{1, \dots, n\}$$

have similar magnitudes in order to get a desired accuracy. Since the shape of f is typically unknown, this might of course be a delicate task.

Example 2.2.2 (Rosenbrock function) We consider Example 2.0.2 and apply the Hooke-Jeeves method to it. We choose 10^{-5} as minimal step size, $\varepsilon = 10^{-7}$ as tolerance and $\text{MaxIt} = 1000$ iterations. For the initial step size $h := 0.1$, we compare different initial values $x^{(0)}$. The results are shown in Table 2.1. We see that the method reaches the minimum x^* only if we start from it. This shows how though the Rosenbrock optimization problem is and also how sensitive the Hooke-Jeeves method depends on the choice of the data.

Example 2.2.3 In the accompanying material you should find also Matlab-codes for the other examples. We do not display all the results in order to keep the size of these notes moderate. We just comment on the results.

- (a) Himmelblau function: The Hooke-Jeeves method is not able to find accurate approximations of the local minima. Moreover, the behavior strongly depends on the chosen initial value.
- (b) Bazaraa-Shetty function: For all tested initial data, the Hooke-Jeeves method converges near the global minimum. However, the accuracy is limited.
- (c) Dixon-Price: For “good” initial guesses, the Hooke-Jeeves method converges to the global minimum. Do these tests for higher space dimensions d as an exercise. \diamond

$x^{(0)}$	$x^{(k)}$	$f(x^{(k)})$	k
(-1.00,-1.00)	(0.00497301, -1.00000000)	100.99502494	17
(0.00,0.00)	(0.16126202, 0.00000000)	0.77110969	20
(0.50,0.50)	(0.70855950, 0.50000000)	0.08536051	20
(0.75,0.75)	(0.86646747, 0.75000000)	0.01788959	17
(0.95,0.95)	(0.97474576, 0.95000000)	0.00063945	17
(0.99,0.99)	(0.99500006, 0.99000000)	0.00002506	17
(1.00,1.00)	(1.00000000, 1.00000000)	0.00000000	17

Table 2.1: Optimization of the Rosenbrock function by the Hooke-Jeeves method for different initial values $x^{(0)}$: computed approximation $x^{(k)}$, value $f(x^{(k)})$ of the objective function and number of iterations until break.

2.2.2 Nelder-Mead method

J. Nelder and R. Mead⁷ presented a derivative-free minimization scheme (1965, [13]), which is widely used also today. The scheme is realized in Matlab's function `fminsearch`. The main idea can be described as follows: Let $f : \mathbb{R}^n \rightarrow \mathbb{R}$ be the objective function. A *simplex* is the convex hull of $n + 1$ points (called *edges*) $\{x_1, \dots, x_{n+1}\}$ (e.g. a triangle for $n = 2$ formed by 3 points). The Nelder-Mead scheme, also called ‘simplex algorithm’⁸ determines a sequence of such simplices aiming at including the minimal point x^* in a simplex and iteratively decreasing the diameter of the simplex.

Let us denote by

$$S^{(k)} := \{x_1^{(k)}, \dots, x_{n+1}^{(k)}\}$$

the simplex in the k -th iteration of the scheme. The goal is to replace an edge of $S^{(k)}$ having the largest value of the objective function (which might not be unique). The new edge is determined by reflecting, dilating and compressing $S^{(k)}$.

Without loss of generality (w.l.o.g.) we may assume that the edges are enumerated in such a way that

$$f(x_1^{(k)}) \leq \dots \leq f(x_{n+1}^{(k)}),$$

i.e., $x_{n+1}^{(k)}$ is the ‘worst’ edge to be replaced. The center of the best n edges is denoted by

$$\bar{x} := \frac{1}{n} \sum_{i=1}^n x_i,$$

i.e., the average of the coordinates of $x_1^{(k)}, \dots, x_n^{(k)}$. Next we consider the straight line through \bar{x} and x_{n+1} , namely

$$\bar{x}(\alpha) = \bar{x} + \alpha(x_{n+1} - \bar{x}), \quad \alpha \in \mathbb{R}. \quad (2.2.1)$$

We check for some α , if this reflection yields an improvement. Dilation and compression of the simplex are mainly straightforward. The full algorithm is detailed in Algorithm 2.2.3.

Algorithm 2.2.3: Nelder-Mead iteration

Input: Parameters for:

- reflection: $\alpha \geq 0$,
- dilation: $\beta > 1$,

⁷John Ashworth Nelder (1924-2010), Roger Mead (1938-2015)

⁸To be distinguished from the simplex method in discrete optimization.

- compression: $\gamma \in (0, 1)$

```

1: Determine  $\bar{x}(-\alpha)$  by (2.2.1) (reflection); set  $f_{-\alpha} := f(\bar{x}(-\alpha))$ 
2: if  $f(x_1) \leq f_{-\alpha} \leq f(x_n)$  then                                 $\triangleright$  the new point is neither the best nor the worst one
3:    $x_{n+1} \rightarrow \bar{x}(-\alpha)$ 
4: else if  $f_{-\alpha} < f(x_1)$  then                       $\triangleright$  new point is the best; proceed and look outside simplex
5:   Determine  $\bar{x}(-\alpha\beta)$  and  $f_{-\alpha\beta} = f(\bar{x}(-\alpha\beta))$ 
6:   if  $f_{-\alpha\beta} < f(x_1)$  then
7:      $x_{n+1} \rightarrow \bar{x}(-\alpha\beta)$                                  $\triangleright$  increase simplex
8:   else
9:      $x_{n+1} \rightarrow \bar{x}(-\alpha)$                                  $\triangleright$  no expansion
10:  end if
11: else                                                  $\triangleright f_{-\alpha} \geq f(x_n)$ 
12:   if  $f(x_n) < f_{-\alpha} < f(x_{n+1})$  then                                 $\triangleright$  outer contraction
13:     Determine  $\bar{x}(-\alpha\gamma)$  and  $f_{-\alpha\gamma} = f(\bar{x}(-\alpha\gamma))$ 
14:     if  $f_{-\alpha\gamma} \leq f_{-\alpha}$  then
15:        $x_{n+1} \rightarrow \bar{x}(-\alpha\gamma)$ 
16:     else
17:       set  $x_i = \frac{1}{2}(x_1 + x_i)$ ,  $i = 2, \dots, n+1$            $\triangleright$  contract simplex around  $x_1$ 
18:     end if
19:   else
20:     Determine  $\bar{x}(\gamma)$  and  $f_\gamma = f(\bar{x}(\gamma))$                                  $\triangleright$  inner contraction
21:     if  $f_\gamma < f(x_{n+1})$  then
22:        $x_{n+1} \rightarrow \bar{x}(\gamma)$ 
23:     else
24:       set  $x_i = \frac{1}{2}(x_1 + x_i)$ ,  $i = 2, \dots, n+1$            $\triangleright$  contract simplex around best point
25:     end if
26:   end if
27: end if

```

We illustrate the points $\bar{x}(\gamma)$, \bar{x} and the other ones involved in Algorithm 2.2.3 ones in Figure 2.4.

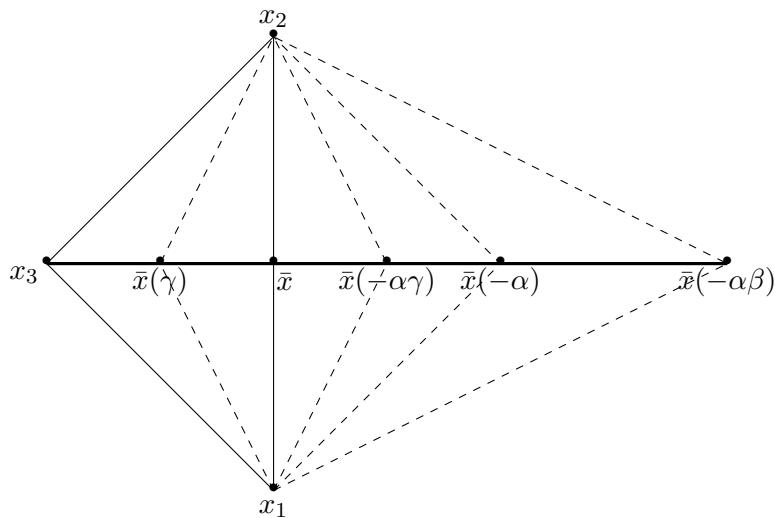


Figure 2.4: Illustration of the points and possibly new simplices involved in the Nelder-Mead iteration.

Some comments on the Nelder-Mead scheme are in order.

Remarks 2.2.4 i) For the parameters α , β and γ , often empirical values $\alpha = 1$, $\beta = 2$ and $\gamma = 1/2$

are used.

- ii) A typical stopping criterion is the claim that the standard deviation of the average of $f(x_1), \dots, f(x_{n+1})$, i.e.,

$$\bar{f} := \frac{1}{n+1} \sum_{i=1}^{n+1} f(x_i)$$

is smaller than a fixed tolerance $\varepsilon > 0$, i.e.

$$\frac{1}{n} \sum_{i=1}^{n+1} (f(x_i) - \bar{f})^2 < \varepsilon.$$

Except from some special theoretical cases, direct search algorithms (both Hooke-Jeeves and Nelder-Mead, but also other direct search schemes) do not allow any rigorous statements on convergence or convergence rate.

Example 2.2.5 (Rosenbrock function) We reconsider Example 2.0.2 and now apply the Nelder-Mead method using `fminsearch` to it. We use the same initial values as in Example 2.2.2 for the Hooke-Jeeves method. Matlab chooses a standard tolerance of 10^{-4} . We observe that the Nelder-Mead method always converges towards the unique minimum. The only difference is the number of required iterations. We visualize the iteration in Figure 2.5.

$x^{(0)}$	$x^{(k)}$	$f(x^{(k)})$	k
(-1.00,-1.00)	(0.99999886, 0.99999542)	0.00000000	103
(0.00,0.00)	(1.00000439, 1.00001064)	0.00000000	117
(0.50,0.50)	(1.00000850, 1.00001872)	0.00000000	67
(0.75,0.75)	(1.00000947, 1.00002044)	0.00000000	56
(0.95,0.95)	(0.99998705, 0.99997434)	0.00000000	37
(0.99,0.99)	(0.99000000, 0.99000000)	0.00000000	36
(1.00,1.00)	(1.00000000, 1.00000000)	0.00000000	30

Table 2.2: Optimization of the Rosenbrock function by the Nelder-Mead method for different initial values $x^{(0)}$: computed approximation $x^{(k)}$, value $f(x^{(k)})$ of the objective function and number of iterations until break.

Example 2.2.6 Again, we just briefly comment on the results for the other mentioned examples.

- (a) Himmelblau function: By increasing the maximal number of iterations, the Nelder-Mead method converges.
- (b) Bazaraa-Shetty function: Same behavior as for the Himmelblau function but with even more iterations.
- (c) Dixon-Price: Same behavior as for the Himmelblau function. \diamond

2.3 GRADIENT DESCENT METHODS

As opposed to direct search methods, we now assume that the objective function f in (2.0.1) is differentiable, so that we can use the gradient within the algorithm in order to obtain a faster convergence as for the direct search schemes.

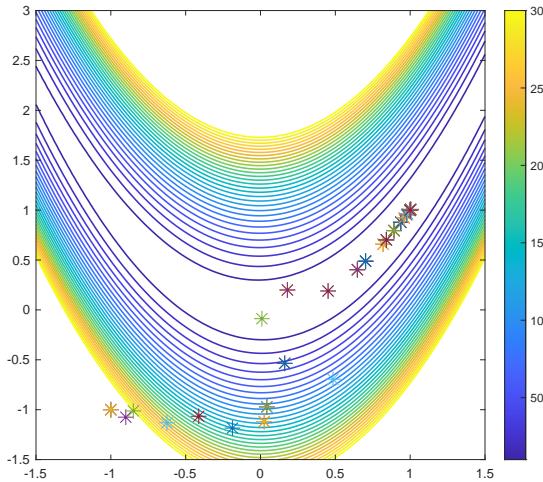


Figure 2.5: Iterations of the Nelder-Mead method with initial value $(-1, -1)$ for the Rosenbrock function.

2.3.1 Core algorithm

Gradient descent methods are a whole class of schemes that use gradient information of the objective function in one way or the other. All types of gradient descent methods consist of two steps:

1. determine a *search direction* $p^{(k)} \in \mathbb{R}^n$,
2. determine a *step size* $\alpha^{(k)} > 0$.

Then, the iteration is done by the update

$$x^{(k+1)} := x^{(k)} + \alpha^{(k)} p^{(k)}, \quad k \in \mathbb{N}_0. \quad (2.3.1)$$

Of course, both the type and also the performance of a gradient descent method depends on the choice of the search direction $p^{(k)}$ and the step size $\alpha^{(k)}$. For later reference, we write this -so far abstract- procedure in form of Algorithm 2.3.1.

Algorithm 2.3.1: Descent method

Input: $f \in C^1(\mathbb{R}^n)$, initial vector $x^{(0)} \in \mathbb{R}^n$

- ```

1: for $k = 0, 1, 2, \dots$ do
2: if $\nabla f(x^{(k)}) = 0$ then STOP
3: determine descent direction $p^{(k)} \in \mathbb{R}^n$
4: determine step size $\alpha^{(k)} > 0$
5: set $x^{(k+1)} := x^{(k)} + \alpha^{(k)} p^{(k)}$
6: end for

```
- 

**Search direction: How to guarantee a descent?** Assuming that the objective function  $f$  is sufficiently smooth, its Taylor expansion yields

$$f(x + \alpha p) = f(x) + \alpha \nabla f(x)^T p + \varepsilon, \quad \varepsilon = o(\alpha),$$

with the remainder term  $\varepsilon \rightarrow 0$  as  $\alpha \rightarrow 0$ . This motivates that we require for a search direction  $p^{(k)}$  that

$$\begin{aligned} \nabla f(x^{(k)})^T p^{(k)} &< 0, & \text{if } \nabla f(x^{(k)}) \neq 0, \\ p^{(k)} &= 0, & \text{if } \nabla f(x^{(k)}) = 0. \end{aligned}$$

**Definition 2.3.1** A vector  $p \in \mathbb{R}^n \setminus \{0\}$  is called descent direction for  $f \in C^1(\mathbb{R}^n)$  at a point  $x \in \mathbb{R}^n$  if  $\nabla f(x)^T p < 0$ .

The name *descent direction* is in fact reasonable, since for  $p^{(k)} \neq 0$  satisfying  $\nabla f(x^{(k)})^T p^{(k)} < 0$  there is a sufficiently small step size  $\alpha^{(k)} > 0$  such that, by Taylor's expansion,

$$f(x^{(k)} + \alpha^{(k)} p^{(k)}) = f(x^{(k)}) + \alpha^{(k)} \nabla f(x^{(k)})^T p^{(k)} + o(\|\alpha^{(k)} p^{(k)}\|) < f(x^{(k)}).$$

**Definition 2.3.2** A sequence  $\{p^{(k)}\}_{k \in \mathbb{N}}$  of search directions is called feasible if each  $p^{(k)}$  is a descent direction in the sense of Definition 2.3.1 and

$$\frac{\nabla f(x^{(k)})^T p^{(k)}}{\|p^{(k)}\|} \xrightarrow{k \rightarrow \infty} 0 \quad \text{implies that} \quad \nabla f(x^{(k)}) \xrightarrow{k \rightarrow \infty} 0.$$

Of course, such a descent direction is not unique. Different constructions or choices of  $p^{(k)}$  lead to different variants of descent methods. The following definition

$$p^{(k)} := -B^{(k)} \nabla f(x^{(k)})$$

is widely used, where  $B^{(k)} \in \mathbb{R}^{n \times n}$  is a “suitable” symmetric and positive definite matrix. Typical options for  $B^{(k)}$  respectively  $p^{(k)}$  are the following:

- $B^{(k)} = I_n$ : Then,  $p^{(k)} = -\nabla f(x^{(k)})$ , which is the classical gradient steepest descent scheme.
- $(B^{(k)})^{-1} = D^2 f(x^{(k)})$ : This choice corresponds to Newton's scheme: Note, that the Hessian matrix  $D^2 f(x)$  is positive definite for  $f \in C^2$ , at least within a suitable small neighborhood of the minimum  $x^*$ .
- $(B^{(k)})^{-1} \approx D^2 f(x^{(k)})$ , i.e., some approximation of the Hessian: This is a Quasi-Newton's scheme where the Hessian matrix is approximated by  $(B^{(k)})^{-1}$ . As an example, we mention Broyden's scheme (lecture on *Numerical Analysis*).
- $p^{(k)} := -\nabla f(x^{(k)}) + \beta^{(k)} p^{(k-1)}$ , where the coefficients  $\beta^{(k)}$  are chosen in such a way that the search directions  $p^{(0)}, p^{(1)}, \dots$  are mutually orthogonal (w.r.t. to some appropriate inner product), which gives rise to the *conjugate gradient* (*cg*) method, which we know from *Numerical Linear Algebra* for linear systems with symmetric positive definite matrices.

### 2.3.2 Choice of the step size

As mentioned above, the construction of a suitable search direction is only the first step of the full scheme. The second one consists of the **choice (or determination) of a suitable step size**  $\alpha^{(k)}$ . Of course, the optimal choice would be to compute the global minimum of  $f$  along the direction  $p^{(k)}$  starting from  $x^{(k)}$ :

$$\Phi^{(k)}(\alpha) := f(x^{(k)} + \alpha p^{(k)}) \xrightarrow{\alpha \in \mathbb{R}} \min!. \quad (2.3.2)$$

However, from a numerical point of view, this is in general way to expensive (with the exception of linear systems with symmetric positive matrices, where we have shown in *Numerical Linear Algebra* how to compute the optimal step size exactly, see below). Moreover, without further assumptions,  $\Phi^{(k)}$  may not have a unique global minimum at all. Again, quadratic functions  $f$  (where the derivative is linear) are examples for that, see below.

**Example 2.3.3** Let  $A \in \mathbb{R}^{n \times n}$  be a symmetric, positive definite matrix and  $b \in \mathbb{R}^n$  be some vector. Then, for the quadratic objective function

$$f : \mathbb{R}^n \rightarrow \mathbb{R}, \quad f(x) = \frac{1}{2}x^T Ax - b^T x,$$

we can easily show that  $\nabla f(x) = Ax - b$ , which implies that

$$\alpha^{(k)} = -\frac{(p^{(k)})^T (Ax^{(k)} - b)}{(p^{(k)})^T Ap^{(k)}} = -\frac{(p^{(k)})^T \nabla f(x^{(k)})}{(p^{(k)})^T Ap^{(k)}}$$

minimizes the function  $\Phi^{(k)}(\alpha) = f(x^{(k)}) + \alpha p^{(k)}$ .  $\diamond$

**Definition 2.3.4** Let  $f \in C^1(\mathbb{R}^n)$  and  $x \in \mathbb{R}^n$  with  $\nabla f(x) \neq 0$ . Furthermore, let  $d \in \mathbb{R}^n$  be the solution of the optimization problem

$$\min_{\|d\|=1} \nabla f(x)^T d. \quad (2.3.3)$$

Then, any vector of the form  $p := \lambda d$ ,  $\lambda > 0$  is called a steepest descent direction of  $f$  in  $x$ .

**Theorem 2.3.5** Let  $f \in C^1(\mathbb{R}^n)$  and  $x \in \mathbb{R}^n$  with  $\nabla f(x) \neq 0$ . Then, (2.3.3) has the unique solution

$$d := -\frac{\nabla f(x)}{\|\nabla f(x)\|}. \quad (2.3.4)$$

In particular,  $p \in \mathbb{R}^n$  is a steepest descent direction if and only if  $p = -\lambda \nabla f(x)$  for some  $\lambda > 0$ .

*Proof.* Let  $v, w \in \mathbb{R}^n$ . Then, we have by the Cauchy-Schwarz inequality that  $|v^T w| \leq \|v\| \|w\|$ . For  $d \in \mathbb{R}^n$  with  $\|d\| = 1$  we thus obtain that

$$\nabla f(x)^T d \geq -\|\nabla f(x)\| \|d\| = -\|\nabla f(x)\|$$

with equality if and only if  $d$  takes the form (2.3.4), which proves the first statement, The second one follows by Definition 2.3.4.  $\square$

Hereafter, we will consider the question how to choose  $\alpha^{(k)}$  appropriately if  $f$  is a general non-linear (i.e., non-quadratic) function.

### Step size control

If we cannot compute or at least approximate an optimal step size in an efficient manner, we aim at setting upper and lower bounds for a *suitable* step size  $\alpha$ . By doing so, we need to clarify what is meant by “suitable”.

- On one hand,  $\alpha$  should not be too large to ensure some descent of the objective function, i.e.,

$$f(x^{(k)} + \alpha^{(k)} p^{(k)}) < f(x^{(k)}). \quad (2.3.5)$$

- On the other hand,  $\alpha$  should be sufficiently large in order to prevent too small step sizes and thus too many iterations. This would cause the algorithm to become inefficient.

As shown in Example 2.3.6 below, the condition (2.3.5) is in general not enough to guarantee that a resulting sequence  $(x^{(k)})_{k \in \mathbb{N}_0}$  converges to the minimum  $x^*$ . We are aiming at constructing sufficient conditions for the step size which yield an interval of acceptable values for  $\alpha^{(k)}$  in order to ensure *convergence* and *efficiency*.

**Example 2.3.6** Let  $f : \mathbb{R} \rightarrow \mathbb{R}$  be the convex function defined as

$$f(x) := (x - 4)^2 - 1.$$

It is easily seen that  $f$  has a unique minimum at  $x^* = 4$  with  $f(x^*) = -1$ . If  $(x^{(k)})_{k \in \mathbb{N}_0}$  is a sequence which fulfills

$$f(x^{(k)}) = \frac{1}{k}, \quad \text{for example } x^{(k)} := 4 + \sqrt{1 + \frac{1}{k}},$$

the function value of  $f$  in fact decreases in each iteration, but the sequence  $(f(x^{(k)}))_{k \in \mathbb{N}_0}$  converges to 0 and not to the minimum  $-1$ . Moreover,  $x^{(k)}$  also does not converge to  $x^*$ .  $\diamond$

**Armijo condition.** A very well-known condition in order to ensure some decay of the objective function  $f$  is the so-called *Armijo condition*<sup>9</sup>, which reads as follows

$$f(x^{(k)} + \alpha^{(k)} p^{(k)}) \leq f(x^{(k)}) + c_1 \alpha^{(k)} \nabla f(x^{(k)})^T p^{(k)} \quad (2.3.6)$$

with some constant  $c_1 \in (0, 1)$  to be chosen. The background of the condition is Taylor's expansion

$$f(x^{(k)} + \alpha^{(k)} p^{(k)}) \approx f(x^{(k)}) + \underbrace{\alpha^{(k)} \nabla f(x^{(k)})^T p^{(k)}}_{\text{chosen to be } < 0}$$

and can be rewritten as

$$\Phi^{(k)}(\alpha^{(k)}) \leq \Phi^{(k)}(0) + c_1 \alpha^{(k)} (\Phi^{(k)})'(0)$$

using the function  $\Phi^{(k)}$  defined in (2.3.2), see also Figure 2.7.

**Theorem 2.3.7 (see [25, Satz 7.7])** Let  $f \in C^1(\mathbb{R}^n)$  and assume that  $\{x^{(k)}\}_{k \in \mathbb{N}}$  is determined by Algorithm 2.3.1 with the Armijo condition (2.3.6). Then, either Algorithm 2.3.1 terminates after finitely many steps with a stationary point of  $f$  or the algorithm produces an infinite sequence such that  $f(x^{(k+1)}) < f(x^{(k)})$  for all  $k \in \mathbb{N}$  and each accumulation point of  $\{x^{(k)}\}_{k \in \mathbb{N}}$  is a stationary point of  $f$ .  $\square$

This latter statement in fact gives a convergence result. We already know from *Numerical Linear Algebra* that the convergence speed of the gradient scheme may be poor (recall the zig-zag behavior for ill-conditioned matrices). Thus, we cannot expect that the convergence speed improves in the general setting, in particular Armijo's rule will in general not use an optimal step size. In fact, since  $c_1 \in (0, 1)$  and  $\nabla f(x^{(k)})^T p^{(k)} < 0$  (recall that  $p^{(k)}$  is a descent direction), the condition (2.3.6) in fact holds for all  $\alpha^{(k)} > 0$  sufficiently small. As a consequence, the Armijo condition alone is not sufficient to ensure the efficiency of the descent algorithm.

**Example 2.3.8 (Rosenbrock function – with Armijo step size control)** We reconsider the Rosenbrock function from Example 2.0.2 and solve it by the steepest descent and the Newton method satisfying the Armijo rule with initial value  $x^{(0)} = (-1, -1)$  and stopping criterium  $\|\nabla f(x^{(k)})\| < \text{tol} = 10^{-6}$ . The results are shown in Table 2.3 and Figure 2.6. We see a very slow convergence for the steepest descent method, in particular within the banana-shape valley. However, the main reason for this is not the Armijo rule, but the choice of the gradients as search directions as we can see from the very fast Newton convergence.

---

<sup>9</sup>Lorenzo “Larry” Armijo (1918-2010)

| $k$   | $\#f(x)$ | $\ \nabla f(x^{(k)})\ $ |
|-------|----------|-------------------------|
| 1     | 20       | 2.97789e+01             |
| 1001  | 9428     | 5.49188e-03             |
| 2001  | 19254    | 7.60275e-04             |
| 3001  | 29162    | 1.31428e-04             |
| 4001  | 39100    | 2.45384e-05             |
| 5001  | 49051    | 4.73211e-06             |
| 6001  | 59008    | 9.24143e-07             |
| 7001  | 68966    | 1.81689e-07             |
| 8001  | 78926    | 3.57892e-08             |
| 9001  | 88886    | 7.06190e-09             |
| 10001 | 98847    | 1.39374e-09             |
| 11001 | 108807   | 2.75152e-10             |
| 12001 | 118768   | 5.43289e-11             |
| 13001 | 128728   | 1.07283e-11             |
| 14001 | 138689   | 2.11812e-12             |

| $k$ | $\#f(x)$ | $\ \nabla f(x^{(k)})\ $ |
|-----|----------|-------------------------|
| 0   | 2        | 4.04000e+02             |
| 1   | 6        | 3.98007e+00             |
| 2   | 7        | 3.43097e+00             |
| 3   | 9        | 2.64013e+00             |
| 4   | 10       | 2.22963e+00             |
| 5   | 11       | 1.69463e+00             |
| 6   | 12       | 1.36776e+00             |
| 7   | 14       | 9.30999e-01             |
| 8   | 15       | 7.36848e-01             |
| 9   | 16       | 4.84290e-01             |
| 10  | 17       | 4.54572e-01             |
| 11  | 19       | 1.79921e-01             |
| 12  | 20       | 1.33132e-01             |
| 13  | 22       | 5.08196e-02             |
| 14  | 23       | 2.64649e-02             |
| 15  | 24       | 8.66419e-03             |
| 16  | 25       | 1.92511e-03             |
| 17  | 26       | 1.75898e-04             |
| 18  | 27       | 2.82644e-06             |
| 19  | 28       | 9.46081e-10             |

Table 2.3: Optimization of the Rosenbrock function by the gradient descent (left) and Newton's (right) method with Armijo stepsize rule ( $\#f(x)$  indicates the number of function evaluations). For the gradient descent method, we only show each 1000th iteration.

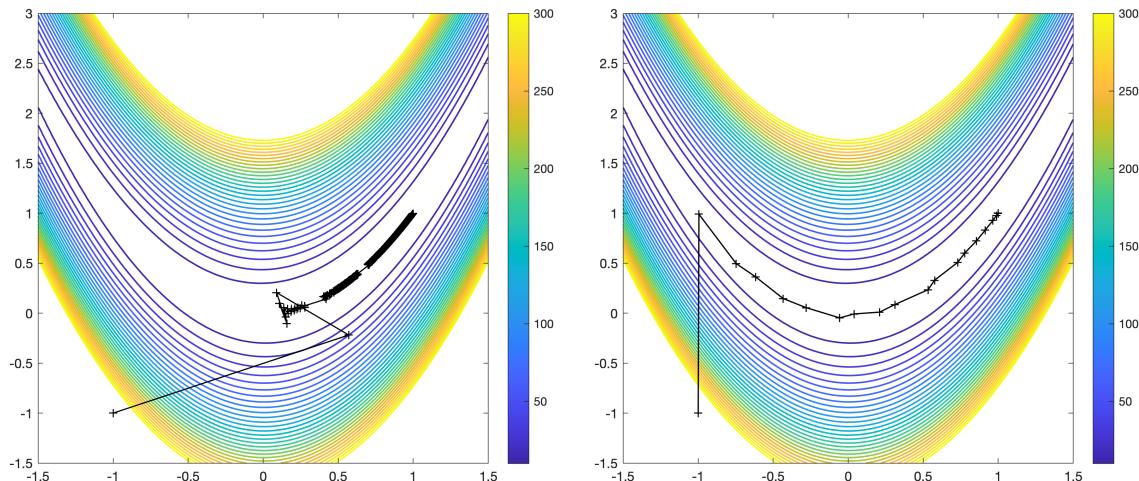


Figure 2.6: Iterations of the gradient descent (left) and Newton's (right) method using the Armijo step size rule with initial value  $(-1, -1)$  for the Rosenbrock function.

**Curvature condition.** Thus, in order to avoid unacceptable small step sizes, a second condition for  $\alpha^{(k)}$  is necessary. An example for this is the so-called *curvature condition*, which reads as follows

$$\nabla f(x^{(k)} + \alpha^{(k)} p^{(k)})^T p^{(k)} \geq c_2 \nabla f(x^{(k)})^T p^{(k)} \quad (2.3.7)$$

with some constant  $c_2 \in (c_1, 1)$  and  $c_1$  being the constant in condition (2.3.6). Again, this inequality can be formulated as a condition on  $\Phi^{(k)}$ , namely

$$(\Phi^{(k)})'(\alpha^{(k)}) \geq c_2 (\Phi^{(k)})'(0).$$

Therefore, the curvature condition ensures that the slope of  $\Phi^{(k)}$  at  $\alpha^{(k)}$  is larger than a factor  $c_2$  times the original slope  $(\Phi^{(k)})'(0)$ . This ensures that the value of the objective function  $f(x^{(k+1)})$  becomes smaller for increasing values of  $\alpha^{(k)}$ , as long as the slope  $(\Phi^{(k)})'(\alpha^{(k)})$  is negative.

**Remark 2.3.9** Typical choices are that  $c_1$  is chosen relatively small, likely in the order of  $c_1 = 10^{-4}$ . A common value for  $c_2$  is 0.9.  $\diamond$

**Wolfe's condition.** The combination of the two conditions, namely the Armijo condition (2.3.6) and the curvature condition (2.3.7), is also known as *Wolfe conditions*.<sup>10</sup> We show an example in Figure 2.7.

$$\Phi^{(k)}(\alpha) = f(x^{(k)} + \alpha p^{(k)})$$

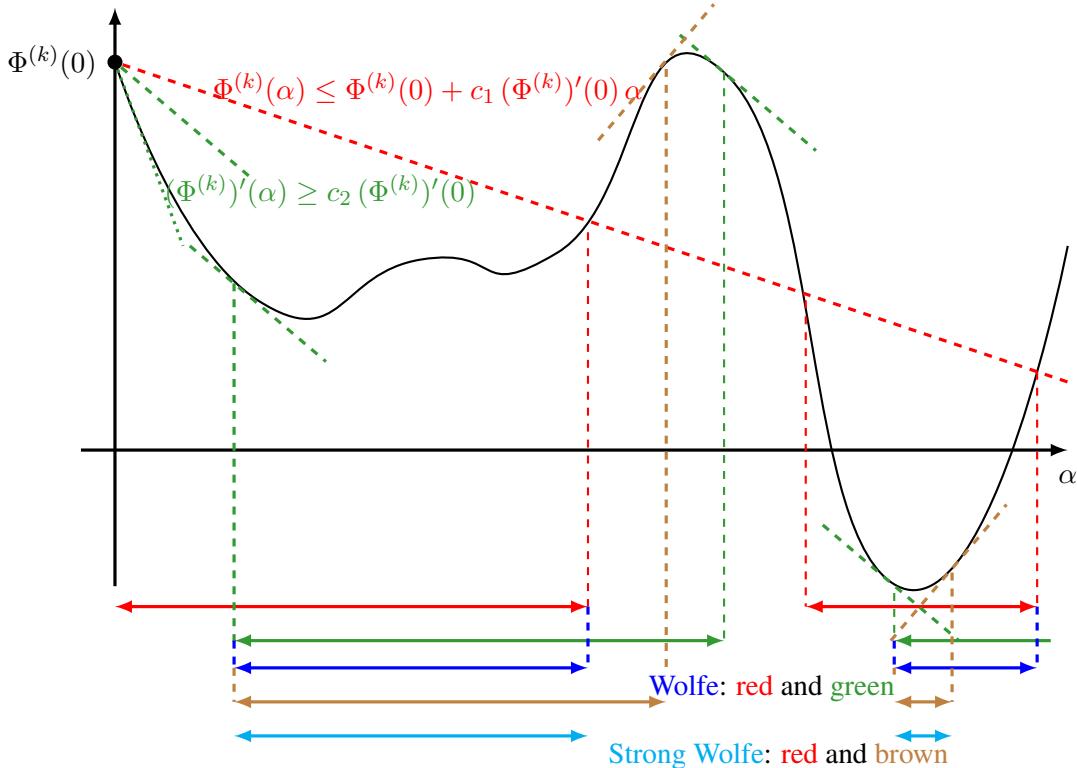


Figure 2.7: Wolfe condition (here depicted for  $0.13 = c_1 < c_2 = 0.44$ ): Feasible step size range according to the first (red) and second (green) condition. The blue part is the range of feasible step sizes. In addition, we depict the strong Wolfe condition. The second condition (green) is replaced by the brown lines, the resulting range of feasible step sizes is shown in cyan.

**Strong Wolfe's condition.** However, those Wolfe's conditions are still relatively weak and allow values for  $\alpha^{(k)}$  so that  $x^{(k+1)}$  is still far away from the desired minimum point  $x^*$ . To avoid this, one can exclude those values for  $\alpha^{(k)}$ , where the derivative of  $\Phi^{(k)}$  is ‘too’ large. The reasoning is that such values would

<sup>10</sup>Philip Starr Wolfe (1927-2016)

be far away from the local and global minimum. Thus, one substitutes the curvature condition (2.3.7) by the following condition, namely

$$|\nabla f(x^{(k)} + \alpha^{(k)} p^{(k)})^T p^{(k)}| \leq c_2 |\nabla f(x^{(k)})^T p^{(k)}|, \quad (2.3.8)$$

reformulated as  $|(\Phi^{(k)})'(\alpha^{(k)})| \leq c_2 |(\Phi^{(k)})'(0)|$ .

Armijo's condition (2.3.6) in combination with the condition (2.3.8) for  $0 < c_1 < c_2 < 1$  are also known as as *strong Wolfe's conditions*.

It's not hard to prove that step sizes exist which suffice the Wolfe and strong Wolfe conditions, respectively (see e.g. [14, Lemma 3.1]):

**Lemma 2.3.10** *Let  $f : \mathbb{R}^n \rightarrow \mathbb{R}$  be continuously differentiable and let  $p^{(k)} \in \mathbb{R}^n$  be a descent direction such that*

$$\nabla f(x^{(k)})^T p^{(k)} < 0$$

*holds. Furthermore, let  $f$  be bounded from below on the set  $\{x^{(k)} + \alpha p^{(k)} \mid \alpha > 0\}$ . Then, choosing  $0 < c_1 < c_2 < 1$ , there exist intervals for  $\alpha^{(k)}$ , so that the Wolfe conditions (2.3.6) and (2.3.7) and the strong Wolfe conditions (2.3.6) and (2.3.8), respectively, are fulfilled.*  $\square$

**Example 2.3.11 (Rosenbrock function – steepest descent with Wolfe conditions)** We reconsider the Rosenbrock function from Example 2.0.2 and solve it by the steepest descent and the Newton method satisfying the Wolfe conditions with  $c_1 = 10^{-4}$ ,  $c_2 = 0.9$ , initial value  $x^{(0)} = (-1, -1)$  and stopping criterium  $\|\nabla f(x^{(k)})\| < \text{tol} = 10^{-6}$ . The results are shown in Table 2.4 and Figure 2.8. As above, we see a very slow convergence and the extremely large number of function evaluations for the steepest descent – again mainly due to the poor choice of the search directions as we can deduce from the fast convergence of Newton's method. Comparing Newton's method with Armijo and Wolfe's condition, we see that the number of iterations is smaller with the Wolfe conditions but at the cost of more function evaluations.  $\diamond$

| $k$   | $\#f(x)$ | $\ \nabla f(x^{(k)})\ $ | $k$ | $\#f(x)$ | $\ \nabla f(x^{(k)})\ $ |
|-------|----------|-------------------------|-----|----------|-------------------------|
| 1     | 23       | 2.97789e+01             | 1   | 17       | 3.98007e+00             |
| 1001  | 23210    | 8.15793e-02             | 2   | 27       | 3.43864e+00             |
| 2001  | 45822    | 2.93571e-02             | 3   | 37       | 1.86150e+00             |
| 3001  | 67386    | 2.81108e-03             | 4   | 41       | 1.40953e+00             |
| 4001  | 88932    | 6.23329e-05             | 5   | 48       | 8.08977e-01             |
| 5001  | 110884   | 1.27935e-05             | 6   | 55       | 4.10538e-01             |
| 6001  | 132820   | 2.56999e-06             | 7   | 62       | 1.68745e-01             |
| 7001  | 154746   | 5.12296e-07             | 8   | 70       | 5.47417e-02             |
| 8001  | 176670   | 1.01498e-07             | 9   | 78       | 2.41024e-03             |
| 9001  | 198594   | 2.00430e-08             | 10  | 83       | 3.80595e-05             |
| 10001 | 220516   | 3.96079e-09             | 11  | 90       | 5.22039e-08             |
| 11001 | 242436   | 7.82602e-10             | 12  | 94       | 2.82282e-13             |
| 12001 | 264358   | 1.54490e-10             | 13  | 115      | 1.69128e-24             |
| 13001 | 286278   | 3.05425e-11             |     |          |                         |
| 14001 | 308200   | 6.02477e-12             |     |          |                         |
| 15001 | 330122   | 1.18968e-12             |     |          |                         |

Table 2.4: Optimization of the Rosenbrock function by the gradient descent (left) and Newton's (right) method satisfying the Wolfe conditions ( $\#f(x)$  indicates the number of function evaluations). For the gradient descent method, we only show each 1000th iteration.

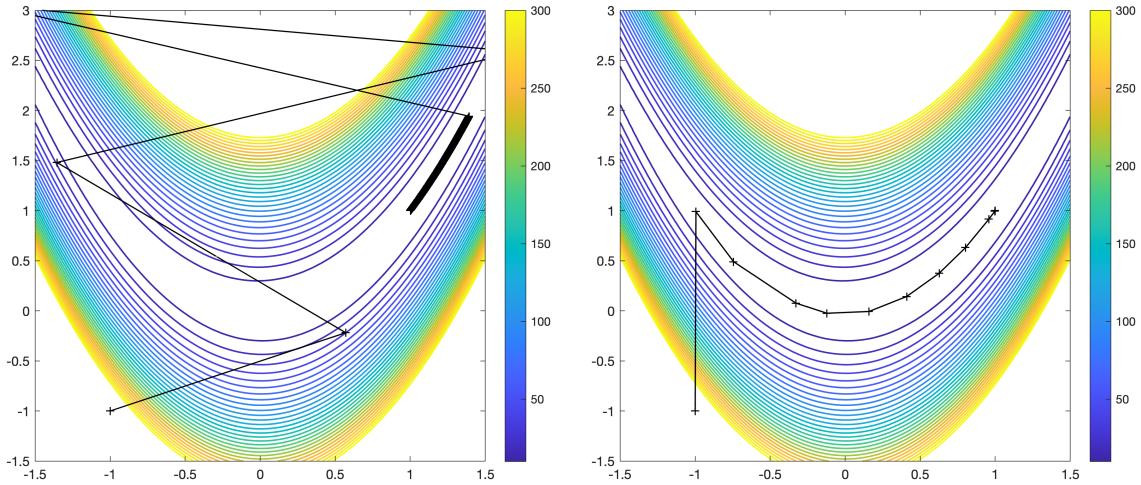


Figure 2.8: Iterations of the gradient descent (left) and Newton's method using the Wolfe condition and initial value  $(-1, -1)$  for the Rosenbrock function.

**Example 2.3.12** The corresponding numerical experiments for the Himmelblau, the Bazaraa-Shetty and the Dixon-Price can be produced by the Matlab files in the supplementary material available at the Moodle site of the course.  $\diamond$

**Further conditions.** Besides Wolfe's conditions, there are also other methods to determine the step size  $\alpha^{(k)}$ . We mention just two more conditions:

- i) *Goldstein conditions*<sup>11</sup> (consisting of two inequalities) read:

$$f(x^{(k)}) + (1 - c)\alpha^{(k)}\nabla f(x^{(k)})^T p^{(k)} \leq f(x^{(k)} + \alpha^{(k)}p^{(k)}) \leq f(x^{(k)}) + c\alpha^{(k)}\nabla f(x^{(k)})^T p^{(k)} \quad (2.3.9)$$

with a constant  $c \in (0, 1/2)$ , or, in terms of  $\Phi^{(k)}$ ,

$$\Phi^{(k)}(0) + (1 - c)\alpha^{(k)}(\Phi^{(k)})'(0) \leq \Phi^{(k)}(\alpha^{(k)}) \leq \Phi^{(k)}(0) + c\alpha^{(k)}(\Phi^{(k)})'(0).$$

In fact, the second inequality coincides with Armijo's condition (2.3.6). The first one is posed in order to ensure that the step size does not become too small. However, the problem with this condition is that minimum points can in fact be excluded.

- ii) *Resetting algorithm*: This algorithm starts with a relatively large step size  $\alpha_0^{(k)}$  and reduces the step size until Armijo's condition (2.3.6) is satisfied. This is similar to the bisection method introduced in the lecture *Numerical Analysis*.

It turned out that the resetting algorithm is particularly suitable for Newton's method, but *not* for the Quasi-Newton method. Often, the factor  $\rho \in (0, 1)$  is adjusted in each iteration  $k \in \mathbb{N}_0$ . The method is shown in Algorithm 2.3.2

---

#### Algorithm 2.3.2: Resetting algorithm to control the step size

**Input:**  $\alpha_0^{(k)} > 0$ ,  $\rho \in (0, 1)$ ,  $c \in (0, 1)$   
 1: set  $\alpha^{(k)} = \alpha_0^{(k)}$

<sup>11</sup>Allen A. Goldstein (1925-2022)

---

```

2: while $f(x^{(k)} + \alpha^{(k)} p^{(k)}) > f(x^{(k)}) + c\alpha^{(k)} \nabla f(x^{(k)})^T p^{(k)}$ do
3: set $\alpha^{(k)} = \rho\alpha^{(k)}$
4: end while

```

---

**Convergence.** For the gradient descent method with step size control via Wolfe's conditions (2.3.6) and (2.3.7), the following statement ensures convergence.

**Theorem 2.3.13 (Zoutendijk, [28])** (i) Let  $(x^{(k)})_{k \in \mathbb{N}_0} \subset \mathbb{R}^n$  be a sequence of the form

$$x^{(k+1)} = x^{(k)} + \alpha^{(k)} p^{(k)}, \quad k \in \mathbb{N}_0,$$

with  $p^{(k)}$  being descent directions, i.e.  $\nabla f(x^{(k)})^T p^{(k)} < 0$  for all  $k \in \mathbb{N}_0$ , and the step sizes  $\alpha^{(k)}$  are chosen fulfilling the Wolfe conditions (2.3.6) and (2.3.7).

(ii) Let  $f : \mathbb{R}^n \rightarrow \mathbb{R}$  be bounded from below on all of  $\mathbb{R}^n$  and continuously differentiable on an open set  $\mathcal{N}$ ,  $f \in C^1(\mathcal{N})$ , including the level set  $\mathcal{L} := \{x \in \mathbb{R}^n | f(x) \leq f(x^{(0)})\}$ , where  $x^{(0)}$  denotes the initial vector.

(iii) In addition, let the gradient  $\nabla f$  be Lipschitz continuous on  $\mathcal{N}$ , which means that there exists a constant  $L > 0$ , so that  $\|\nabla f(x) - \nabla f(y)\| \leq L\|x - y\|$  for all  $x, y \in \mathcal{N}$ .

Then, the Zoutendijk condition

$$\sum_{k=0}^{\infty} (\cos \theta^{(k)})^2 \|\nabla f(x^{(k)})\|^2 < \infty \quad (2.3.10)$$

holds, where

$$\cos \theta^{(k)} := \frac{-\nabla f(x^{(k)})^T p^{(k)}}{\|\nabla f(x^{(k)})\| \|p^{(k)}\|}.$$

*Proof.* See e.g. [14, Theorem 3.2] and also [28]. □

To see that this statement in fact ensures convergence, note, that Zoutendijk's condition<sup>12</sup> (2.3.10) implies that

$$(\cos \theta^{(k)})^2 \|\nabla f(x^{(k)})\|^2 \longrightarrow 0 \quad \text{for } k \rightarrow \infty.$$

This yields the desired convergence result, which means that  $\|\nabla f(x^{(k)})\| \rightarrow 0$  as  $k \rightarrow \infty$ , if the angle  $\theta^{(k)}$  between the search direction  $p^{(k)}$  and the direction of the steepest decent  $-\nabla f(x^{(k)})$  is uniformly bounded away from  $90^\circ$ , i.e., there exists a  $\delta > 0$  such that

$$\cos \theta^{(k)} \geq \delta > 0 \quad \forall k \in \mathbb{N}_0.$$

We note this fact in the following statement.

**Theorem 2.3.14** Let  $f \in C^1(\mathbb{R}^n)$  and let  $\{p^{(k)}\}_{k \in \mathbb{N}}$  be the sequence of search directions. If Zoutendijk's condition (2.3.10) holds, the sequence  $\{p^{(k)}\}_{k \in \mathbb{N}}$  is feasible in the sense of Definition 2.3.2. □

**Remark 2.3.15** Similar to Theorem 2.3.13 one can show that Zoutendijk's condition (2.3.10) is also fulfilled if the step sizes  $(\alpha^{(k)})_{k \in \mathbb{N}_0}$  do not satisfy Wolfe's conditions (2.3.6) and (2.3.7), but the strong Wolfe conditions (2.3.6) and (2.3.8) or the Goldstein conditions (2.3.9). ◊

Finally, we formulate a more general convergence result, compare Definition 2.3.2 of a feasible search direction.

---

<sup>12</sup>Guus Zoutendijk (1929-2005)

**Definition 2.3.16** A sequence  $\{\alpha^{(k)}\}_{k \in \mathbb{N}}$  is called feasible if  $f(x^{(k)} + \alpha p^{(k)}) \leq f(x^{(k)})$  for all  $k \in \mathbb{N}$  and

$$f(x^{(k)} + \alpha p^{(k)}) - f(x^{(k)}) \xrightarrow{k \rightarrow \infty} 0 \quad \text{implies that} \quad \left\{ \frac{\nabla f(x^{(k)})^T p^{(k)}}{\|p^{(k)}\|} \right\}_{k \in \mathbb{N}} \xrightarrow{k \rightarrow \infty} 0.$$

**Theorem 2.3.17 (see [25, Satz 8.7])** Let  $f \in C^1(\mathbb{R}^n)$  and assume that Algorithm 2.3.1 does not terminate after finitely many steps. Then, there exists some  $\mathbb{K} \subset \mathbb{N}$  and a convergent subsequence  $\{x^{(k)}\}_{k \in \mathbb{K}}$  with limit  $x^*$  such that  $\{p^{(k)}\}_{k \in \mathbb{K}}$  and  $\{\alpha^{(k)}\}_{k \in \mathbb{K}}$  are feasible. Then,  $x^*$  is a stationary point of  $f$ .  $\square$

### Algorithms for selecting the step size

We are now going to present some basic algorithms in order to realize the step size selection. This is typically done in two steps:

1. *Initial phase*: determine an interval of acceptable values,

2. *Backtracking phase*: find a good candidate for the step size.

Recalling the function  $\Phi^{(k)}$  defined in (2.3.2), we start by some initial step size  $\alpha_0^{(k)}$ . One checks if Armijo's condition (2.3.6) is fulfilled. If this is the case, the search is stopped and  $\alpha^{(k)} := \alpha_0^{(k)}$  is chosen as step size. Otherwise, we conclude that there is some acceptable step size in the interval  $[0, \alpha_0^{(k)}]$ .

Still, it remains to evaluate  $\Phi^{(k)}$  and possibly its derivative, which might be numerically costly. Thus, the idea of the following algorithm is to approximate  $\Phi^{(k)}$  by polynomials and to search for the minimum of the polynomial (which is –hopefully– more efficient) instead of minimizing  $\Phi^{(k)}$ . To this end, we only want to use  $(\Phi^{(k)})'(0) = \nabla f(x^{(k)})^T p^{(k)}$  and function values of  $\Phi^{(k)}$ . Hence, using  $\Phi^{(k)}(0)$ ,  $(\Phi^{(k)})'(0)$  and  $\Phi^{(k)}(\alpha_0^{(k)})$ , we can determine the quadratic polynomial  $\Phi_1^{(k)}$  fulfilling the interpolation conditions

$$\Phi_1^{(k)}(0) = \Phi^{(k)}(0), \quad (\Phi_1^{(k)})'(0) = (\Phi^{(k)})'(0), \quad \Phi_1^{(k)}(\alpha_0^{(k)}) = \Phi^{(k)}(\alpha_0^{(k)}).$$

Then, we determine the minimum  $\alpha_1^{(k)}$  of that quadratic function  $\Phi_1^{(k)}$ , which is easily done. If  $\alpha_1^{(k)}$  satisfies the Armijo condition (2.3.6), we stop the search and set  $\alpha^{(k)} := \alpha_1^{(k)}$ . Otherwise, one determines a cubic interpolation polynomial  $\Phi_2^{(k)}$  by the data  $\Phi^{(k)}(0)$ ,  $(\Phi^{(k)})'(0)$ ,  $\Phi^{(k)}(\alpha_0^{(k)})$  and  $\Phi^{(k)}(\alpha_1^{(k)})$  using the interpolation conditions

$$\begin{aligned} \Phi_2^{(k)}(0) &= \Phi^{(k)}(0), & (\Phi_2^{(k)})'(0) &= (\Phi^{(k)})'(0), \\ \Phi_2^{(k)}(\alpha_0^{(k)}) &= \Phi^{(k)}(\alpha_0^{(k)}), & \Phi_2^{(k)}(\alpha_1^{(k)}) &= \Phi^{(k)}(\alpha_1^{(k)}). \end{aligned}$$

If the minimum  $\alpha_2^{(k)}$  of the cubic function  $\Phi_2^{(k)}$  fulfills Armijo's condition (2.3.6), one sets  $\alpha^{(k)} := \alpha_2^{(k)}$  and stops. Otherwise, one performs another cubic interpolation using the interpolation data  $\Phi^{(k)}(0)$ ,  $(\Phi^{(k)})'(0)$ ,  $\Phi^{(k)}(\alpha_1^{(k)})$  and  $\Phi^{(k)}(\alpha_2^{(k)})$  and so on. If  $\alpha_j^{(k)}$  and  $\alpha_{j-1}^{(k)}$  are too close to each other, or the difference of the two is too large, one substitutes  $\alpha_j^{(k)}$  by  $\alpha_j^{(k)} = \frac{\alpha_{j-1}^{(k)}}{2}$ . Then, the full algorithm looks as in Algorithm 2.3.3.

### Algorithm 2.3.3: Armijo step size determination

**Input:**  $\alpha_0^{(k)} > 0$ ,  $\Phi^{(k)}$ ,  $\delta_{\min}, \delta_{\max} > 0$

- 1: Compute  $\Phi^{(k)}(0)$ ,  $(\Phi^{(k)})'(0)$ ,  $\Phi^{(k)}(\alpha_0^{(k)})$
- 2: **if**  $\Phi^{(k)}(\alpha_0^{(k)}) \leq \Phi^{(k)}(0) + c_1 \alpha_0^{(k)} (\Phi^{(k)})'(0)$  **then**  $\triangleright$  Armijo's condition
- 3:      $\alpha^{(k)} := \alpha_0^{(k)}$ ; **return**
- 4: **else**  $\triangleright$  Identify quadratic int. polynomial and its minimum points
- 5:     
$$\Phi_1^{(k)}(\alpha) = \left( \frac{\Phi^{(k)}(\alpha_0^{(k)}) - \Phi^{(k)}(0) - \alpha_0^{(k)} (\Phi^{(k)})'(0)}{(\alpha_0^{(k)})^2} \right) \alpha^2 + \alpha (\Phi^{(k)})'(0) + \Phi^{(k)}(0)$$

```

6: $\alpha_1^{(k)} = -\frac{(\Phi^{(k)})'(0)(\alpha_0^{(1)})^2}{2(\Phi^{(k)}(\alpha_0^{(1)})-\Phi^{(k)}(0)-(\Phi^{(k)})'(0)\alpha_0^{(k)})} \in [0, \alpha_0^{(k)}]$
7: Compute $\Phi^{(k)}(\alpha_1^{(k)})$
8: end if
9: if $\alpha_0^{(k)} - \alpha_1^{(k)} < \delta_{\min}$ or $\alpha_0^{(k)} - \alpha_1^{(k)} > \delta_{\max}$ then ▷ Step size difference too small?
10: $\alpha_1^{(k)} := \frac{\alpha_1^{(k)}}{2}$
11: end if
12: if $\Phi^{(k)}(\alpha_1^{(k)}) \leq \Phi^{(k)}(0) + c_1 \alpha_1^{(k)} (\Phi^{(k)})'(0)$ then ▷ Armijo's condition
13: $\alpha^{(k)} = \alpha_1^{(k)}$; return
14: else
15: for $j = 1, 2, \dots$ do ▷ Cubic int. polynomial & minimum
16: $\gamma_{j+1} = \frac{1}{(\alpha_{j-1}^{(k)} \alpha_j^{(k)})^2 (\alpha_j^{(k)} - \alpha_{j-1}^{(k)})}$
17: $\begin{pmatrix} a_{j+1} \\ b_{j+1} \end{pmatrix} = \gamma_{j+1} \begin{pmatrix} (\alpha_{j-1}^{(k)})^2 & -(\alpha_j^{(k)})^2 \\ -(\alpha_{j-1}^{(k)})^3 & (\alpha_j^{(k)})^3 \end{pmatrix} \begin{pmatrix} \Phi^{(k)}(\alpha_j^{(k)}) - \Phi^{(k)}(0) - (\Phi^{(k)})'(0)\alpha_j^{(k)} \\ \Phi^{(k)}(\alpha_{j-1}^{(k)}) - \Phi^{(k)}(0) - (\Phi^{(k)})'(0)\alpha_{j-1}^{(k)} \end{pmatrix}$
18: $\Phi_{j+1}^{(k)}(\alpha) = a_{j+1}\alpha^3 + b_{j+1}\alpha^2 + \alpha(\Phi^{(k)})'(0) + \Phi^{(k)}(0)$
19: $\alpha_{j+1}^{(k)} = \frac{-b_{j+1} + \sqrt{b_{j+1}^2 - 3a_{j+1}(\Phi^{(k)})'(0)}}{3a_{j+1}}$
20: Compute $\Phi^{(k)}(\alpha_{j+1}^{(k)})$
21: if $\alpha_j^{(k)} - \alpha_{j+1}^{(k)} < \delta_{\min}$ or $\alpha_j^{(k)} - \alpha_{j+1}^{(k)} > \delta_{\max}$ then
22: $\alpha_{j+1}^{(k)} = \frac{\alpha_j^{(k)}}{2}$
23: end if
24: if $\Phi^{(k)}(\alpha_{j+1}^{(k)}) \leq \Phi^{(k)}(0) + c_1 \alpha_{j+1}^{(k)} (\Phi^{(k)})'(0)$ then ▷ Armijo's condition
25: $\alpha^{(k)} = \alpha_{j+1}^{(k)}$; return
26: end if
27: end for
28: end if

```

**Remarks 2.3.18** i) Algorithm 2.3.3 only checks Armijo's condition (2.3.6) in each iteration. Therefore, only  $(\Phi^{(k)})'(0)$  (and no further values of the derivative) needs to be computed, which is clearly a benefit in terms of efficiency. However, one does not know if Wolfe's or the strong Wolfe conditions might be fulfilled.

ii) Typically, the initial step size in Newton's or Quasi-Newton's method is chosen to be 1. For the gradient or the conjugate gradient algorithm, a typical procedure for the determination of the initial step size  $\alpha_0^{(k)}$  is obtained under the assumption that the change of the (first order term w.r.t.  $\alpha^{(k)}$  respectively  $\alpha^{(k-1)}$ ) function value  $f(x^{(k)})$  as compared to  $f(x^{(k-1)})$  is approximately equal, which means that

$$\alpha^{(k)} \nabla f(x^{(k)})^T p^{(k)} \approx \alpha^{(k-1)} \nabla f(x^{(k-1)})^T p^{(k-1)}.$$

Hence, one sets

$$\alpha_0^{(k)} := \alpha^{(k-1)} \frac{\nabla f(x^{(k-1)})^T p^{(k-1)}}{\nabla f(x^{(k)})^T p^{(k)}}$$

as the initial step size. ◇

As mentioned above, Algorithm 2.3.3 does not ensure the (strong) Wolfe conditions. That is why we introduce a second algorithm that returns step sizes who actually satisfy the strong Wolfe conditions. This algorithm utilizes the fact that there are step sizes in the interval  $[\alpha_{j-1}^{(k)}, \alpha_j^{(k)}]$ , which satisfy the strong Wolfe conditions provided one of the following three cases are fulfilled:

1.  $\alpha_j^{(k)}$  violates Armijo's condition (2.3.6),
2.  $\Phi^{(k)}(\alpha_j^{(k)}) \geq \Phi^{(k)}(\alpha_{j-1}^{(k)})$ ,
3.  $(\Phi^{(k)})'(\alpha_j^{(k)}) \geq 0$ .

A suitable order for the testing of those conditions (besides checking the strong Wolfe conditions), would be to test those conditions involving derivatives of  $\Phi^{(k)}$  at last. That gives the algorithm detailed in Algorithm 2.3.4.

---

**Algorithm 2.3.4: Determination of the step size via strong Wolfe conditions**

**Input:**  $\alpha_{\max} > 0$ ,  $\alpha_1^{(k)} \in (0, \alpha_{\max})$

```

1: Set $\alpha_0^{(k)} = 0$
2: for $j = 1, 2, \dots$ do
3: Compute $\Phi^{(k)}(\alpha_1^{(k)})$
4: if $\Phi^{(k)}(\alpha_j^{(k)}) > \Phi^{(k)}(0) + c_1 \alpha_j^{(k)} (\Phi^{(k)})'(0)$ or [$j > 1$ and $\Phi^{(k)}(\alpha_j^{(k)}) \geq \Phi^{(k)}(\alpha_{j-1}^{(k)})$] then
5: $\alpha^{(k)} = \text{Interpolation}(\alpha_{j-1}^{(k)}, \alpha_j^{(k)})$
6: return
7: end if
8: Compute $(\Phi^{(k)})'(\alpha_j^{(k)})$
9: if $|(\Phi^{(k)})'(\alpha_j^{(k)})| \leq -c_2 (\Phi^{(k)})'(0)$ then
10: $\alpha^{(k)} = \alpha_j^{(k)}$; return
11: end if
12: if $(\Phi^{(k)})'(\alpha_j^{(k)}) \geq 0$ then
13: $\alpha^{(k)} = \text{Interpolation}(\alpha_j^{(k)}, \alpha_{j-1}^{(k)})$
14: return
15: end if
16: Choose $\alpha_{j+1}^{(k)} \in (\alpha_j^{(k)}, \alpha_{\max})$
17: end for

```

---

The procedure **Interpolation** called within Algorithm 2.3.4 determines a step size  $\alpha^{(k)}$  via interpolation or bisection until a step size was found that satisfies the strong Wolfe condition. For this, we need to take into account that in general the inequality  $\alpha_{\text{low}}^{(k)} \leq \alpha_{\text{high}}^{(k)}$  is *not* ensured within the function call **Interpolation**( $\alpha_{\text{low}}^{(k)}, \alpha_{\text{high}}^{(k)}$ ), but  $\alpha_{\text{low}}^{(k)}$  and  $\alpha_{\text{high}}^{(k)}$  are rather characterized by the following properties:

1. The interval which is bounded by the values  $\alpha_{\text{low}}^{(k)}$  and  $\alpha_{\text{high}}^{(k)}$  contains step sizes that satisfy the strong Wolfe conditions;
2.  $\alpha_{\text{low}}^{(k)}$  is the smallest function value with respect to  $\Phi^{(k)}$  according to all so far generated step sizes fulfilling Armijo's condition (2.3.6);
3.  $\alpha_{\text{high}}^{(k)}$  is chosen so that  $(\Phi^{(k)})'(\alpha_{\text{low}}^{(k)})(\alpha_{\text{high}}^{(k)} - \alpha_{\text{low}}^{(k)}) < 0$  holds.

We detail this proceeding in Algorithm 2.3.5.

---

**Algorithm 2.3.5: Interpolation**

**Input:**  $\alpha_{\text{low}}^{(k)}, \alpha_{\text{high}}^{(k)} > 0$ .

```

1: for $i = 1, 2, \dots$ do
2: Identify a $\alpha_i^{(k)} \in [\alpha_{\text{low}}^{(k)}, \alpha_{\text{high}}^{(k)}]$ by quadratic/cubic interpolation/bisection
3: Compute $\Phi^{(k)}(\alpha_i^{(k)})$
4: if $\Phi^{(k)}(\alpha_i^{(k)}) > \Phi^{(k)}(0) + c_1 \alpha_i^{(k)} (\Phi^{(k)})'(0)$ or $\Phi^{(k)}(\alpha_i^{(k)}) \geq \Phi^{(k)}(\alpha_{\text{low}}^{(k)})$ then

```

```

5: $\alpha_{\text{high}}^{(k)} = \alpha_i^{(k)}$
6: else
7: Compute $(\Phi^{(k)})'(\alpha_i^{(k)})$
8: if $|(\Phi^{(k)})'(\alpha_i^{(k)})| \leq -c_2(\Phi^{(k)})'(0)$ then
9: $\alpha^{(k)} = \alpha_i^{(k)}$; return
10: end if
11: if $(\Phi^{(k)})'(\alpha_i^{(k)}) (\alpha_{\text{high}}^{(k)} - \alpha_{\text{low}}^{(k)}) \geq 0$ then
12: $\alpha_{\text{low}}^{(k)} = \alpha_i^{(k)}$
13: end if
14: $\alpha_{\text{low}}^{(k)} = \alpha_i^{(k)}$
15: end if
16: end for

```

### 2.3.3 Non-linear conjugate gradient (cg) method

After addressing the determination of the step size  $\alpha^{(k)}$ , we will now focus on the determination of the search direction  $p^{(k)}$ . In particular, we describe the non-linear variant of the cg-algorithm. We recall from *Numerical Linear Algebra* the (linear) cg-algorithm for the iterative numerical approximation the solution of a linear system  $Ax = a$ <sup>13</sup> with a positive definite matrix  $A$ . In that case, one writes the solution  $x^* = A^{-1}a$  as the minimum of the quadratic function  $\frac{1}{2}x^T Ax - a^T x$ . In order to point out similarities and differences of the non-linear cg-algorithm and the linear counterpart, we repeat the cg-algorithm for the linear case, see Algorithm 2.3.6. This will be done without further descriptions.

#### Algorithm 2.3.6: Linear cg-algorithm

**Input:**  $x^{(0)} \in \mathbb{R}^n$ ,  $A \in \mathbb{R}^{n \times n}$ ,  $a \in \mathbb{R}^n$ .

```

1: $r^{(0)} = Ax^{(0)} - a$
2: $p^{(0)} = -r^{(0)}$
3: $k = 0$
4: while $\|r^{(k)}\| \neq 0$ do
5: $\alpha^{(k)} = \frac{(r^{(k)})^T r^{(k)}}{(p^{(k)})^T A p^{(k)}}$
6: $x^{(k+1)} = x^{(k)} + \alpha^{(k)} p^{(k)}$
7: $r^{(k+1)} = Ax^{(k+1)} - a = r^{(k)} + \alpha^{(k)} A p^{(k)}$
8: $\beta^{(k+1)} = \frac{(r^{(k+1)})^T r^{(k+1)}}{(r^{(k)})^T r^{(k)}}$
9: $p^{(k+1)} = -r^{(k+1)} + \beta^{(k+1)} p^{(k)}$
10: $k \rightarrow k + 1$
11: end while

```

Now, we are going to generalize Algorithm 2.3.6 from the specific case of a quadratic function  $f(x) = \frac{1}{2}x^T Ax - a^T x$  to general smooth functions  $f : \mathbb{R}^n \rightarrow \mathbb{R}$ . In order to do so, we need to take the following issues into account:

1. In the general case, the optimal step size  $\alpha^{(k)}$  cannot be computed exactly and thus will be determined by one of the algorithms for the step size control introduced above, namely Algorithm 2.3.2, 2.3.3 or 2.3.4.
2. In the quadratic case, the residual  $r^{(k)}$  coincides with the gradient of  $f$  and can thus be evaluated exactly at the point  $x^{(k)}$ . In the general case, we need the gradient of the non-linear function  $f$  at  $x^{(k)}$ , or some appropriate approximation.

<sup>13</sup>The notation  $a$  instead of  $b$  for the right-hand side will turn out to be convenient later for constraint optimization problems.

Algorithm 2.3.7 shows the *non-linear cg-algorithm* after Fletcher-Reeves.<sup>14</sup>

---

**Algorithm 2.3.7: Non-linear cg-algorithm after Fletcher-Reeves**

**Input:**  $x^{(0)} \in \mathbb{R}^n$ ,  $f : \mathbb{R}^n \rightarrow \mathbb{R}$ .

- 1: Compute  $\nabla f^{(0)} = \nabla f(x^{(0)})$
- 2:  $p^{(0)} = -\nabla f^{(0)}$
- 3:  $k = 0$
- 4: **while**  $\|\nabla f^{(k)}\| \neq 0$  **do**
- 5:     Compute step size  $\alpha^{(k)}$  with a suitable algorithm
- 6:      $x^{(k+1)} = x^{(k)} + \alpha^{(k)} p^{(k)}$
- 7:     Compute  $\nabla f^{(k+1)} = \nabla f(x^{(k+1)})$
- 8:      $\beta_{\text{FR}}^{(k+1)} = \frac{(\nabla f^{(k+1)})^T \nabla f^{(k+1)}}{(\nabla f^{(k)})^T \nabla f^{(k)}}$
- 9:      $p^{(k+1)} = -\nabla f^{(k+1)} + \beta_{\text{FR}}^{(k+1)} p^{(k)}$
- 10:     $k \rightarrow k + 1$
- 11: **end while**

---

**Remarks 2.3.19** i) It is easily seen that Algorithm 2.3.7 coincides with Algorithm 2.3.6 for the special case  $f(x) = \frac{1}{2}x^T Ax - a^T x$ .

ii) There are also further versions of the non-linear cg-algorithm. Those essentially differ in the choice of  $\beta^{(k+1)}$  in line 8. We only mention two variants:

- Variant of Polak-Ribiére,<sup>15</sup> where

$$\beta_{\text{PR}}^{(k+1)} := \frac{\nabla(f^{(k+1)})^T (\nabla f^{(k+1)} - \nabla f^{(k)})}{\|\nabla f^{(k)}\|^2}$$

- Variant of Hestenes-Stiefel<sup>16</sup> with

$$\beta_{\text{HS}}^{(k+1)} := \frac{\nabla(f^{(k+1)})^T (\nabla f^{(k+1)} - \nabla f^{(k)})}{(\nabla f^{(k+1)} - \nabla f^{(k)})^T p^{(k)}}$$

replaces  $\beta_{\text{FR}}^{(k+1)}$  in line 8.

**Example 2.3.20 (Rosenbrock function – nonlinear cg with Armijo)** We reconsider the Rosenbrock function from Example 2.0.2 and solve it by the nonlinear cg-method with different variants for defining  $\beta^{(k+1)}$ . We use the Armijo conditions as before, the initial value  $x^{(0)} = (-1, -1)$  and stopping criterium  $\|\nabla f(x^{(k)})\| < \text{tol} = 10^{-6}$ . The results are shown in Table 2.5 and Figure 2.9. We see that the Polak-Ribiére variant is the most efficient one among the nonlinear cg-methods. On the other hand, of course, all nonlinear cg-methods cannot compete with Newton's method. This is to be expected since the Newton method uses the Hessian, whereas the nonlinear cg-method only requires the gradient. ◇

**Example 2.3.21** The corresponding numerical experiments for the Himmelblau and the Bazaraa-Shetty can be produced by the Matlab files in the supplementary material available at the Moodle site of the course. We note that Newton's method does not converge for the Bazaraa-Shetty method even if we use initial points very close to the minimum. The reason is that the Hessian matrix is singular. We leave the Dixon-Price function (in particular for higher space dimensions) as a coding exercise. ◇

<sup>14</sup>Roger Fletcher (1939-2016), Colin M. Reeves (1944-)

<sup>15</sup>Elijah Polak (1931-), Gerard Ribiére (?)

<sup>16</sup>Magnus Hestenes (1906-1991), Eduard Ludwig Stiefel (1909-1978)

| $k$ | Fletcher-Reeves |                         | Polak-Ribi  re |                         | Hestenes-Stiefel |                         |
|-----|-----------------|-------------------------|----------------|-------------------------|------------------|-------------------------|
|     | # $f(x)$        | $\ \nabla f(x^{(k)})\ $ | # $f(x)$       | $\ \nabla f(x^{(k)})\ $ | # $f(x)$         | $\ \nabla f(x^{(k)})\ $ |
| 1   | 20              | 5.53018e+00             | 20             | 4.25703e+00             | 20               | 4.53419e+00             |
| 11  | 208             | 3.61151e-01             | 184            | 5.27449e-05             | 264              | 6.00241e-05             |
| 21  | 306             | 5.19552e-04             | 311            | 1.95177e-07             | 365              | 3.64770e-07             |
| 31  | 393             | 5.17359e-06             | 465            | 9.21449e-11             | 539              | 1.66741e-08             |
| 41  | 498             | 1.95602e-06             | 587            | 1.69033e-12             | 735              | 7.90686e-14             |
| 51  | 603             | 4.86210e-07             |                |                         |                  |                         |
| 61  | 705             | 4.95899e-08             |                |                         |                  |                         |
| 71  | 795             | 7.79536e-09             |                |                         |                  |                         |
| 81  | 904             | 3.57380e-09             |                |                         |                  |                         |
| 91  | 1026            | 3.26520e-09             |                |                         |                  |                         |
| 101 | 1149            | 1.55238e-09             |                |                         |                  |                         |
| 111 | 1267            | 4.10391e-10             |                |                         |                  |                         |
| 121 | 1376            | 2.06089e-10             |                |                         |                  |                         |
| 131 | 1474            | 2.80076e-11             |                |                         |                  |                         |
| 141 | 1555            | 1.58642e-14             |                |                         |                  |                         |

Table 2.5: Optimization of the Rosenbrock function by the nonlinear cg-method with the Fletcher-Reeves, Polak-Ribi  re and Hestenes-Stiefel variant satisfying the Wolfe conditions (# $f(x)$  indicates the number of function evaluations). We only show each 10th iteration.

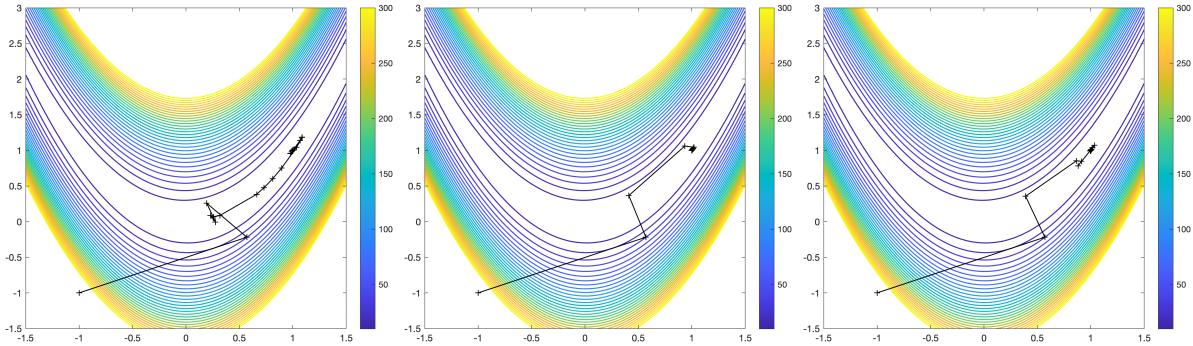


Figure 2.9: Iterations of the nonlinear cg-method using the Armijo condition and initial value  $(-1, -1)$  for the Rosenbrock function and Fletcher-Reeves (left), Polak-Ribi  re (center) and Hestenes-Stiefel (right) variant.

## 2.4 TRUST-REGION METHODS

Developed in the 1970's, trust-region methods are nowadays among the most powerful global optimization schemes. Let us describe the background motivation for an objective function  $f \in C^2(\mathbb{R}^n)$  following [25]. The iteration  $x^{(k)} \rightarrow x^{(k+1)} = x^{(k)} + p^{(k)}$  is done as follows:

1. Using Taylor's expansion of  $f(x^{(k)} + p)$  around  $p = 0$  yields a *quadratic model*

$$q^{(k)}(p) := f^{(k)} + (g^{(k)})^T p + \frac{1}{2} p^T H^{(k)} p \quad (2.4.1)$$

for –in the smooth case–  $f^{(k)} := f(x^{(k)})$ ,  $g^{(k)} := \nabla f(x^{(k)})$  and  $H^{(k)} := \nabla^2 f(x^{(k)})$  (later, we might use approximations instead of the exact quantities, in particular, if  $f \notin C^2(\mathbb{R}^n)$ ). The idea is to obtain  $p^{(k)}$  by solving a (constrained) optimization with the quadratic function  $q^{(k)} : \mathbb{R}^n \rightarrow \mathbb{R}$ .

2. The quadratic model  $q^{(k)}(p)$  is an  $o(\|p\|^2)$ -approximation of  $f(x^{(k)} + p)$  by Taylor's expansion. This means that the approximation is only reasonable in a neighborhood of  $x^{(k)}$ . Thus, we introduce some *trust-region radius*  $\Delta^{(k)} > 0$  and the *trust-region*

$$R^{(k)} := B_{\Delta^{(k)}}(0) = \{p \in \mathbb{R}^n : \|p\| \leq \Delta^{(k)}\},$$

in which we 'trust' the quadratic approximation. Given  $\Delta^{(k)}$ , we solve the (constrained) *trust-region-subproblem*

$$p^{(k)} := \arg \min_{p \in \mathbb{R}^n} \{q^{(k)}(p) : \|p\| \leq \Delta^{(k)}\} = \arg \min_{p \in R^{(k)}} q^{(k)}(p). \quad (2.4.2)$$

3. It remains to determine  $\Delta^{(k)}$  and  $R^{(k)}$ . To this end, we consider

- the *predicted reduction*  $\text{pr}^{(k)} := q^{(k)}(0) - q^{(k)}(p^{(k)}) = f^{(k)} - q^{(k)}(p^{(k)})$  and
- the *actual reduction*  $\text{ac}^{(k)} := f^{(k)} - f(x^{(k)} + p^{(k)})$

as well as the ratio

$$\varrho^{(k)} := \frac{\text{ac}^{(k)}}{\text{pr}^{(k)}}, \quad (2.4.3)$$

which is in fact computable once  $f^{(k)}$ ,  $g^{(k)}$ ,  $H^{(k)}$ ,  $x^{(k)}$  and  $p^{(k)}$  are known. We now choose a parameter  $\eta_1 \in (0, 1)$  (e.g.  $\eta_1 = 0.1$ ) and proceed as follows:

- (i) If  $\varrho^{(k)} \leq \eta_1$ , the actual reduction is too small as compared to the predicted one, which means that the current trust-region is too large. We reject the step  $p^{(k)}$  and choose

$$x^{(k+1)} := x^{(k)}, \quad \Delta^{(k+1)} < \Delta^{(k)}.$$

- (ii) Otherwise, i.e., if  $\varrho^{(k)} > \eta_1$ , the actual reduction is satisfactory, we accept the step and set

$$x^{(k+1)} := x^{(k)} + p^{(k)}.$$

The new trust-region radius is chosen as follows:

$$\tilde{\Delta}^{(k+1)} \begin{cases} \geq \Delta^{(k)}, & \text{if } \varrho^{(k)} \text{ is close to 1,} \\ \leq \Delta^{(k)}, & \text{else,} \end{cases}$$

and then  $\Delta^{(k+1)} := \max\{\tilde{\Delta}^{(k+1)}, \Delta_{\min}\}$ , where we choose  $\Delta_{\min} \geq 0$  as some minimal radius (for efficiency reasons).

We can generalize the above approach, e.g. concerning  $H^{(k)} = \nabla^2 f(x^{(k)})$  (which is not appropriate if  $f \notin C^2(\mathbb{R}^n)$ ) and the need for an exact solution of the trust-region-subproblem (which is often also not appropriate in practice). To this end, we formulate two possible assumptions.

- Assumption 2.4.1** (a) Let  $f \in C^1(\mathbb{R}^n)$  be *bounded from below*, i.e., there exists some  $c > -\infty$  such that  $f(x) \geq c$  for all  $x \in \mathbb{R}^n$ .  
(b) Let  $H^{(k)} \in \mathbb{R}^{n \times n}$  be *uniformly bounded*, i.e., there is some constant  $0 < c_H < \infty$  such that  $\|H^{(k)}\| \leq c_H$  for all  $k \in \mathbb{N}$ .

The determination of solutions of the trust-region subproblems can often be done in a quite efficient manner, but is still costly for large-scale optimization problems. It can be shown that the trust-region method is already convergent under a milder condition on the computed increments  $p^{(k)}$ .

- Assumption 2.4.2 (Fraction of Cauchy decrease)** The updates  $p^{(k)}$  are said to satisfy the *Cauchy descent condition*, if there exists constants  $\alpha \in (0, 1]$  and  $\beta \geq 1$  such that

$$\|p^{(k)}\| \leq \beta \Delta^{(k)}, \quad \text{pr}^{(k)} = f^{(k)} - q^{(k)}(p^{(k)}) \geq \alpha [f^{(k)} - q^{(k)}(p_C^{(k)})], \quad (2.4.4)$$

where  $p_C^{(k)}$  denotes the *Cauchy step*, i.e., the unique solution of the univariate constrained minimization problem

$$q^{(k)}(p) \longrightarrow \min! \quad \text{s.t. } p = -t g^{(k)}, \quad t \geq 0, \quad \|p\| \leq \Delta^{(k)}.$$

Now, we consider the general version of the trust-region method as in Algorithm 2.4.1.

**Algorithm 2.4.1: Trust-region method****Input:** Choose

- parameters  $\alpha \in (0, 1]$ ,  $\beta \geq 1$ ,  $0 < \eta_1 < \eta_2 < 1$ ,  $0 < \gamma_0 < \gamma_1 < 1 < \gamma_2$ ,  $\Delta_{\min} \geq 0$
- initial vector  $x^{(0)} \in \mathbb{R}^n$ ,  $\Delta^{(0)} \geq \Delta_{\min}$

```

1: for $k = 0, 1, 2, \dots$ do
2: if $g^{(k)} = 0$ then
3: Stop, return $x^{(k)}$
4: end if
5: Choose $H^{(k)} \in \mathbb{R}^{n \times n}$ symmetric
6: Determine $p^{(k)}$ satisfying (2.4.4)
7: Compute $\varrho^{(k)}$ according to (2.4.3)
8: if $\varrho^{(k)} \geq \eta_1$ then
9: $x^{(k+1)} := x^{(k)} + p^{(k)}$ ▷ accept the step $p^{(k)}$
10: else
11: $x^{(k+1)} := x^{(k)}$
12: end if
13: Determine $\Delta^{(k+1)}$ by Algorithm 2.4.2
14: end for

```

**Algorithm 2.4.2: Trust-region-radius update****Input:** As in Algorithm 2.4.1

- $\eta_1, \eta_2, \gamma_0, \gamma_1, \gamma_2$
  - $\Delta^{(k)}, \Delta_{\min}, \varrho^{(k)}$
- ```

1: if  $\varrho^{(k)} \leq \eta_1$  then
2:   choose  $\Delta^{(k+1)} \in [\gamma_0 \Delta^{(k)}, \gamma_1 \Delta^{(k)}]$ 
3: else if  $\varrho^{(k)} \leq \eta_2$  then
4:   choose  $\Delta^{(k+1)} \in [\max\{\Delta_{\min}, \gamma_1 \Delta^{(k)}\}, \max\{\Delta_{\min}, \Delta^{(k)}\}]$ 
5: else
6:   choose  $\Delta^{(k+1)} \in [\max\{\Delta_{\min}, \Delta^{(k)}\}, \max\{\Delta_{\min}, \gamma_2 \Delta^{(k)}\}]$ 
7: end if

```

Definition 2.4.3 A step $p^{(k)}$ is called successful if $\varrho^{(k)} > \eta_1$ (and thus $x^{(k+1)} = x^{(k)} + p^{(k)}$). We denote by $\mathbb{P} \subset \mathbb{N}$ the set of indices of all successful steps.

Lemma 2.4.4 (See [25, Lemma 14.9].) Consider Algorithm 2.4.1 under Assumptions 2.4.1 and 2.4.2. If $\mathbb{K} \subset \mathbb{P}$ an infinite set with $\|f^{(k)}\| \geq \varepsilon > 0$ for all $k \in \mathbb{K}$, then

$$\sum_{k \in \mathbb{K}} \Delta^{(k)} < \infty.$$

We report a global convergence result, which is based upon the previous lemma. We recall the quadratic model from (2.4.1), in particular that $g^{(k)} = \nabla f(x^{(k)})$ in the smooth case, otherwise some suitable approximation. For the proof, we refer to [25, §14].

Theorem 2.4.5 Assume that Assumption 2.4.2 holds. Then, either Algorithm 2.4.1 terminates with the output of a stationary point or it generates an infinite sequence $\{x^{(k)}\}_{k \in \mathbb{N}}$ such that

$$\liminf_{k \rightarrow \infty} \|g^{(k)}\| = 0. \quad (2.4.5)$$

Furthermore, if ∇f is uniformly continuous on some $\Omega \subset \mathbb{R}^n$ and $\{x^{(k)}\}_{k \in \mathbb{N}} \subset \Omega$, then

$$\lim_{k \rightarrow \infty} \|g^{(k)}\| = 0. \quad (2.4.6)$$

- Remark 2.4.6**
- (a) There are necessary and sufficient conditions for the global solution of the trust-region-subproblem, [25, §14.2], which we will discuss in general within the scope of constrained optimization problems.
 - (b) The algorithm with the choice $H^{(k)} = \nabla^2 f(x^{(k)})$, $f \in C^2(\mathbb{R}^n)$, is called Trust-Region-Newton scheme, which can be shown to converge super-linearly under appropriate conditions, [25, §14.3].

2.5 NUMERICAL EXPERIMENTS WITH MATLAB-INTERNAL ROUTINES

Again, we reconsider the Rosenbrock function from Example 2.0.2. We use the Matlab internal function `fminunc` (“find minimum of unconstrained multivariable function”) that realizes the gradient descent method offering various options. We start by using a Quasi-Newton method. Without providing the gradient, `fminunc` uses an approximate gradient. We test the same initial values as for Hooke-Jeeves and report the results in Table 2.6. First, we observe that the scheme converges for all choices of the initial value. Second, the number of iterations is significantly smaller the closer we start to x^* . The output is always

Optimization completed because the size of the gradient is less than the value of the optimality tolerance, whose default value is 10^{-6} .

$x^{(0)}$	$x^{(k)}$	$f(x^{(k)})$	k
(-1.00,-1.00)	(0.99999538, 0.99999070)	0.00000000	24
(0.00,0.00)	(0.99999559, 0.99999117)	0.00000000	20
(0.50,0.50)	(0.99999552, 0.99999104)	0.00000000	20
(0.75,0.75)	(0.99998458, 0.99996912)	0.00000000	13
(0.95,0.95)	(0.99999564, 0.99999124)	0.00000000	13
(0.99,0.99)	(0.99999551, 0.99999101)	0.00000000	5
(1.00,1.00)	(1.00000000, 1.00000000)	0.00000000	1

Table 2.6: Optimization of the Rosenbrock function by the gradient descent method `fminunc` using Quasi-Newton with approximate gradient for different initial values $x^{(0)}$: computed approximation $x^{(k)}$, value $f(x^{(k)})$ of the objective function and number of iterations until break.

Next, we provide `fminunc` with the gradient of the Rosenbrock function and use $x^{(0)} = (-1, -1)$ as initial point. The scheme terminates after 24 iterations towards $x^{(24)} = (0.99999538, 0.99999070)$ with $f(x^{(24)}) = 0.00000000$. We display the iteration history in Table 2.7. As we see, the provided Jacobian does not give rise to a speed-up. Our interpretation of this fact is that the Matlab-internal approximation of the first order derivative is already good, which is not too surprising as f is smooth. We also see that the step size is different from 1 only in some rare cases. In Figure 2.11, we show the iteration history.

Finally, we compare three variants of the gradient descent method, namely

k	# eval. f	$f(x^{(k)})$	$\alpha^{(k)}$	$\nabla f(x^{(k)})$
0	3	404		804
1	6	26.2494	0.00124378	100
2	9	13.9789	1	72.10
3	12	0.9284290	1	1.89
4	15	0.9231300	1	1.93
5	18	0.8989750	1	2.07
6	21	0.8316180	1	3.40
7	30	0.4714800	0.313262	2.50
8	36	0.4657710	0.153144	3.85
9	42	0.3511660	10	2.07
10	45	0.2875810	1	5.85
11	48	0.1858730	1	0.77
12	51	0.1229000	1	2.92
13	54	0.0927551	1	3.32
14	57	0.0523518	1	2.74
15	63	0.0237227	0.5	1.36
16	66	0.0196576	1	3.04
17	69	0.0103558	1	1.34
18	72	0.0032830	1	0.902
19	75	0.0008774	1	0.199
20	78	0.0003816	1	0.760
21	81	2.99146e-05	1	0.197
22	84	1.53773e-07	1	0.00181
23	87	4.02863e-09	1	0.00138
24	90	2.17305e-11	1	2.23e-05

Table 2.7: Optimization of the Rosenbrock function by the gradient descent method `fminunc` using Quasi-Newton with approximate gradient for the initial value $x^{(0)} = (-1, -1)$: iteration history. We also display the number of function evaluations as a measure for the computational complexity.

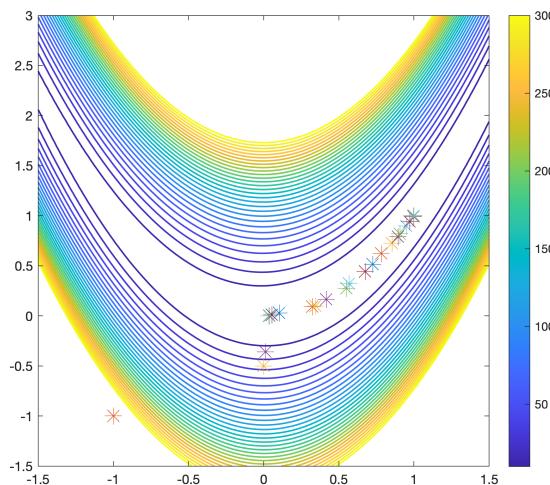


Figure 2.10: Iterations of the gradient descent method using the first-order derivative Quasi-Newton scheme with initial value $(-1, -1)$ for the Rosenbrock function.

- (i) Quasi-Newton with approximate derivative,
- (ii) Quasi-Newton with provided derivative,
- (iii) Newton with Jacobian and Hessian. The latter one is realized ion `fminunc` by choosing the “Trust-Region Method” from Section 2.4.

We use the same initial point $x^{(0)} = (0, 0)$ and display the results in Table 2.8. As we see, providing the Jacobian results in fewer iterations, whereas providing the Hessian does not yield a further improvement. Again, we conjecture that the internal Matlab approximation of the Hessian is working very well due to the smoothness of f , in particular once the iteration is the “banana-shape valley”. However, we also see that our own Newton iteration required less iterations than `fminunc`, see Table 2.4.

Method	$x^{(k)}$	$f(x^{(k)})$	k
(i) Quasi-Newton w/o deriv.	(0.99999559, 0.99999117)	0.00000000	24
(ii) Quasi-Newton with deriv.	(0.99999989, 0.99999976)	0.00000000	20
(iii) Newton	(0.99999989, 0.99999976)	0.00000000	20

Table 2.8: Optimization of the Rosenbrock function by the gradient descent method `fminunc` using different schemes for the same initial point $x^{(0,0)}$.

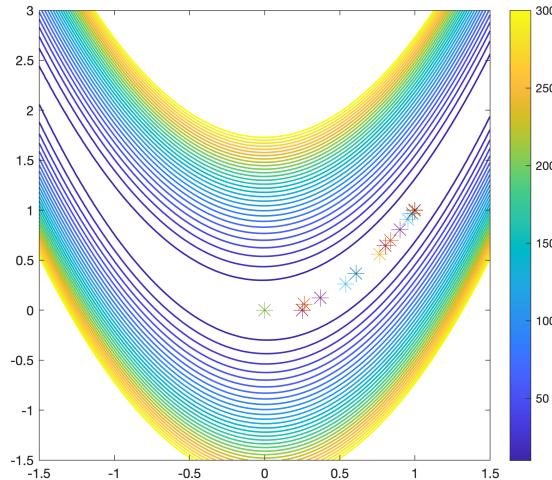


Figure 2.11: Iterations of the gradient descent method using Newtons scheme with initial value $(0, 0)$ for the Rosenbrock function.

The Matlab files for producing all results in this chapter are provided on the Moodle page of the course.

Example 2.5.1 The corresponding numerical experiments for the Himmelblau, the Bazaraa-Shetty and the Dixon-Prize can be produced by the Matlab files in the supplementary material available at the Moodle site of the course. \diamond

3 NONLINEAR CONSTRAINED OPTIMIZATION

In this chapter we will focus on the question how to proceed, if the minimum of the objective function $f : \mathbb{R}^n \rightarrow \mathbb{R}$ is not searched in all of \mathbb{R}^n , but only in a subset $\mathcal{F} \subset \mathbb{R}^n$. Therefore, we are facing a *constrained optimization* problem. Typically, we consider the case where \mathcal{F} is a set of the form

$$\mathcal{F} = \{x \in \mathbb{R}^n : h(x) = 0, g(x) \leq 0\}, \quad (3.0.1)$$

for given functions $h : \mathbb{R}^n \rightarrow \mathbb{R}^m$ and $g : \mathbb{R}^n \rightarrow \mathbb{R}^p$. Then, the optimization problem reads

$$\text{minimize } f(x), x \in \mathbb{R}^n \text{ subject to the constraints (s.t.)} \quad (3.0.2a)$$

$$h(x) = 0, \quad h : \mathbb{R}^n \rightarrow \mathbb{R}^m, \quad (3.0.2b)$$

$$g(x) \leq 0, \quad g : \mathbb{R}^n \rightarrow \mathbb{R}^p, \quad (3.0.2c)$$

where the number of constraints must be smaller than the number of variables, i.e., $m + p \leq n$. We call \mathcal{F} the *feasible region* of (3.0.2) and some $x \in \mathcal{F}$ is called a *feasible point*.

We start by considering some examples.

Example 3.0.1 Let the following optimization problem be given:

$$\begin{aligned} & \text{minimize } f(x_1, x_2) := (x_1 - 2)^2 + (x_2 - 2)^2 \text{ in } \mathbb{R}^2 \\ & \text{s.t. } g_1(x_1, x_2) := x_1^2 - x_2 \leq 0, \\ & \quad g_2(x_1, x_2) := x_1 + x_2 - 2 \leq 0. \end{aligned}$$

Then, the objective function is defined for $x = (x_1, x_2)^T \in \mathbb{R}^2$ as above and the inequality constraints are summarized to

$$g(x) := \begin{pmatrix} g_1(x) \\ g_2(x) \end{pmatrix} = \begin{pmatrix} x_1^2 - x_2 \\ x_1 + x_2 - 2 \end{pmatrix}.$$

As we do not have equality constraints, we have $m = 2$ and $p = 0$. The exact solution is $x^* = (1, 1)$ with $f(x^*) = 2$. Note that, $x^* \in \partial\mathcal{F}$, i.e., on the boundary of \mathcal{F} , i.e., $g(x^*) = 0$, the inequality constraints are active. \diamond

Example 3.0.2 Again, we consider the Rosenbrock function defined in Example 2.0.2, i.e., $f(x) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2$ for $x = (x_1, x_2)^T \in \mathbb{R}^2$. Then, we consider the constrained minimization problem with inequality resp. equality constraints

$$\begin{aligned} & \text{minimize } f(x), x \in \mathbb{R}^2 && \text{minimize } f(x), x \in \mathbb{R}^2 \\ & \text{s.t. } g(x) := x_1^2 + x_2^2 - 1.5 \leq 0, && \text{s.t. } h(x) := x_1^2 + x_2^2 - 1.5 = 0, \end{aligned}$$

which is visualized in Figure 3.1. Apparently, both problems admit the same exact solution $x^* \approx (0.907, 0.823)$ with $f(x^*) \approx 0.0087$. Again, $x^* \in \partial\mathcal{F}$. \diamond

Example 3.0.3 Of course, one can perform analogous numerical experiments using the Himmelblau, Bazaraa-Shetty or the Dixon-Price function (in particular for higher space dimensions) introducing constraints. We leave this as coding exercises. \diamond

Before we present theory and algorithms for constrained optimization problems, we remark that inequality constraints of the form $g(x) \leq 0$ can be rewritten as equality constraints $\tilde{g}(x) = 0$ by setting

$$\tilde{g}(x) := \max\{0, g(x)\}. \quad (3.0.3)$$

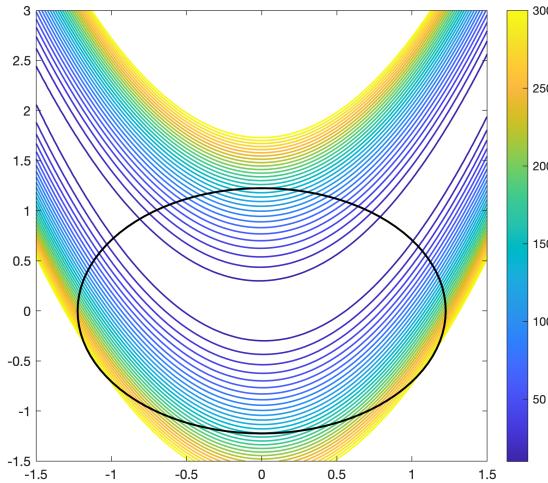


Figure 3.1: Constrained optimization problem with the Rosenbrock function. The desired solution is constrained to be in the black circle.

This is the reason, why –at least in some parts– we consider only constrained problems of the form

$$\begin{cases} \text{minimize } f(x) \text{ in } \mathbb{R}^n, \\ \text{s.t. } h(x) = 0, \end{cases} \quad (3.0.4)$$

with $f : \mathbb{R}^n \rightarrow \mathbb{R}$, $h : \mathbb{R}^n \rightarrow \mathbb{R}^m$ and $m \leq n$. However, the function \tilde{g} in (3.0.3) is in general not smooth, which yields some drawbacks of this seemingly simple approach. For that reason, we will also present alternative methods to reduce inequality constraints to equality constraints.

3.1 OPTIMALITY CONDITIONS

This section is devoted to the development and analysis of necessary and sufficient optimality conditions for the constrained optimization problem (3.0.2). In order to formulate those, we need some notation.

Definition 3.1.1 Let $x \in \mathcal{F}$ be a feasible point of (3.0.2). Then, we call

$$\mathcal{A}(x) := \{j = 1, \dots, p : g_j(x) = 0\}$$

the index set of active inequality constraints and

$$\mathcal{I}(x) := \{1, \dots, p\} \setminus \mathcal{A}(x) = \{j = 1, \dots, p : g_j(x) < 0\}$$

the index set of inactive inequality constraints.

It is obvious that there are constraints which make it impossible to find a solution. Hence, there must be conditions on the constraints in order to ensure well-posedness. Such conditions are termed “Constraint Qualifications (CQ)”. It is also natural, that there are several concepts of such (CQ). A complete description would go far beyond the scope of such a lecture. Hence, we only describe the most common ones, which also turned out to be particularly useful for practical applications. For the formulation of the (CQ), we need some preparation.

Definition 3.1.2 (a) A set $\mathcal{C} \subset \mathbb{R}^n$ is called cone if $\lambda x \in \mathcal{C}$ for all $x \in \mathcal{C}$ and all $\lambda > 0$.¹

¹In the literature, one also finds “ $\lambda \geq 0$ ” here.

- (b) A set $\mathcal{C} \subset \mathbb{R}^n$ is called convex cone, if it is convex and a cone, i.e., $\lambda_1 x_1 + \lambda_2 x_2 \in \mathcal{C}$ for all $x_1, x_2 \in \mathcal{C}$ and all $\lambda_1, \lambda_2 \geq 0$.
- (c) The conic hull of a set \mathcal{C} is defined as

$$\text{cone}(\mathcal{C}) := \left\{ \sum_{i=1}^k \lambda_i x_i : k \in \mathbb{N}, x_i \in \mathcal{C}, \lambda_i \geq 0 \right\}.$$

Let $x \in \mathcal{C}$, then $\mathcal{K}(\mathcal{C}, x) := \text{cone}(X - x) = \{\lambda(y - x) : y \in \mathcal{C}\}$ ² is called conic hull of \mathcal{C} at x .

- (d) Let $x_1, \dots, x_k \in \mathcal{C}$, $\lambda_i \geq 0$, $i = 1, \dots, k$, $k \in \mathbb{N}$. A linear combination $\sum_{i=1}^k \lambda_i x_i$ is called conical combination of x_1, \dots, x_k . If in addition $\sum_{i=1}^k \lambda_i = 1$, then the conical combination is called a convex combination of x_1, \dots, x_k .

By definition, if \mathcal{C} is a cone, either $0 \in \mathcal{C}$ or at least $0 \in \partial\mathcal{C}$; a convex cone contains the origin. The conic hull is the smallest convex cone containing \mathcal{C} , see Figure 3.2. A cone is convex if and only if it contains all conical combinations of its elements. If \mathcal{C} is a convex set, then $\mathcal{K}(\mathcal{C}, x)$ is a convex cone.

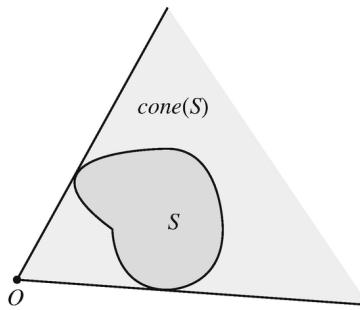


Figure 3.2: Conic hull cone(S) of a set S (taken from [4]).

- Example 3.1.3** (a) The sets $\{x \in \mathbb{R}^n : x > 0\}$ and $\{x \in \mathbb{R}^n : x \geq 0\}$ are convex cones.
- (b) The set $\{x = (x_1, x_2)^T \in \mathbb{R}^2 : x_1 \geq 0 \text{ and } x_2 = 0 \text{ or } x_1 = 0 \text{ and } x_2 \geq 0\}$ is a cone, but not convex.
- (c) The set $\{x \in \mathbb{R}^n : x \geq 0\} \cup \{x \in \mathbb{R}^n : x \leq 0\}$ is also a cone, but not convex.
- (d) The set $\{x = (x_1, x_2)^T \in \mathbb{R}^2 : x_1 \geq 0, 0 < x_2 \leq x_1\}$ is a convex cone, see Figure 3.3.

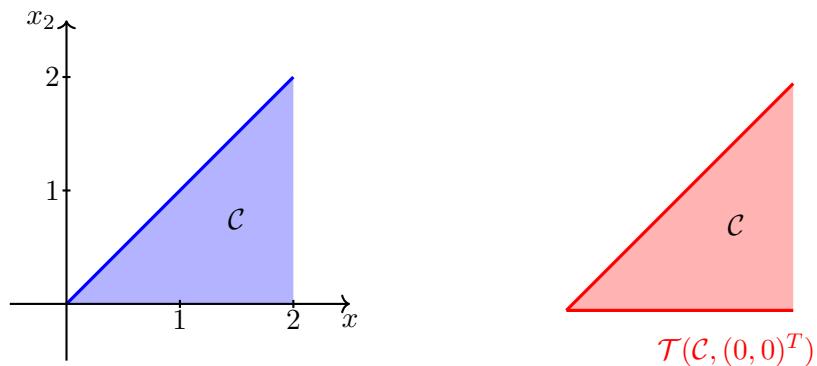


Figure 3.3: Convex cone $\mathcal{C} := \{x = (x_1, x_2)^T \in \mathbb{R}^2 : x_1 \geq 0, 0 < x_2 \leq x_1\}$ (left) and tangential cone $\mathcal{T}(\mathcal{C}, (0, 0)^T)$ (right).

²This last representation requires an (easy) proof.

Lemma 3.1.4 A cone $\mathcal{C} \subset \mathbb{R}^n$ is convex if and only if $\mathcal{C} + \mathcal{C} \subset \mathcal{C}$, where $\alpha\mathcal{A} + \beta\mathcal{B} := \{\alpha a + \beta b : a \in \mathcal{A}, b \in \mathcal{B}\}$. \square

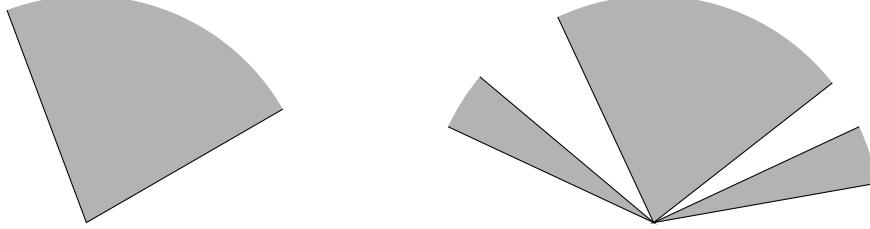


Figure 3.4: Convex (left) and non-convex cone (right).

Definition 3.1.5 Let $\emptyset \neq \mathcal{C} \subset \mathbb{R}^n$. The set

$$\begin{aligned} \mathcal{T}(\mathcal{C}, x) := & \left\{ d \in \mathbb{R}^n : \exists (\eta^{(\ell)})_{\ell \in \mathbb{N}} \subset \mathbb{R}^+, (x^{(\ell)})_{\ell \in \mathbb{N}} \subset \mathcal{C} : \right. \\ & \left. \lim_{\ell \rightarrow \infty} x^{(\ell)} = x, \lim_{\ell \rightarrow \infty} \eta^{(\ell)}(x^{(\ell)} - x) = d \right\} \end{aligned}$$

is called tangential cone of \mathcal{C} in $x \in \mathcal{C}$.

Example 3.1.6 (a) Consider the convex cone \mathcal{C} introduced in Example 3.1.3 (d) and $x = (0, 0)^T$. Then, $\mathcal{T}(\mathcal{C}, (0, 0)^T) = \{(d_1, 0)^T : d_1 \geq 0\}$, which can be proven as an exercise; see the right graph in Figure 3.3.

(b) Consider the non-convex cone \mathcal{C} defined in Example 3.1.3 (b) and $x = (0, 0)^T$. Then, $\mathcal{T}(\mathcal{C}, (0, 0)^T) = \mathcal{C}$ is not convex.

We note (without proof) some properties of the tangential cone, which we leave as exercises.

Lemma 3.1.7 Then, we have:

- (i) Let $\mathcal{S} \subset \mathbb{R}^n$ be a set and $x \in \mathcal{S}$. Then, the tangential cone $\mathcal{T}(\mathcal{S}, x)$ is closed.
- (ii) Let $\mathcal{S} \subset \mathbb{R}^n$ be convex and $x \in \mathcal{S}$, then $\mathcal{T}(\mathcal{S}, x) = \overline{\mathcal{K}(\mathcal{S}, x)}$ (the tangential cone is the closure of the conic hull). \square

Theorem 3.1.8 Let x^* be a local solution of (3.0.2). Then,

- (i) $x^* \in \mathcal{F}$ (recall $\mathcal{F} = \{x \in \mathbb{R}^n : g(x) \leq 0, h(x) = 0\}$);
- (ii) $\nabla f(x^*)^T d \geq 0$ for all $d \in \mathcal{T}(\mathcal{F}, x^*)$.

Proof. (i) is obvious. As for (ii), let $d \in \mathcal{T}(\mathcal{F}, x^*)$ be arbitrary and let $(\eta^{(\ell)})_{\ell \in \mathbb{N}} \subset \mathbb{R}^+$ and $(x^{(\ell)})_{\ell \in \mathbb{N}} \subset \mathcal{F}$ be such that $x^{(\ell)} \rightarrow x^*$ as well as $d^{(\ell)} := \eta^{(\ell)}(x^{(\ell)} - x^*) \rightarrow d$ as $\ell \rightarrow \infty$. Since x^* is a local minimum, we have $f(x^{(\ell)}) \geq f(x^*)$ for ℓ sufficiently large. Then, Taylor series expansion yields

$$\begin{aligned} 0 \leq \eta^{(\ell)}(f(x^{(\ell)}) - f(x^*)) &= \eta^{(\ell)} \nabla f(x^{(\ell)})^T (x^{(\ell)} - x^*) + \eta^{(\ell)} o(\|x^{(\ell)} - x^*\|) \\ &= \nabla f(x^{(\ell)})^T d^{(\ell)} + \|d^{(\ell)}\| \frac{o(\|x^{(\ell)} - x^*\|)}{\|x^{(\ell)} - x^*\|} \xrightarrow{\ell \rightarrow \infty} \nabla f(x^*)^T d, \end{aligned}$$

which proves the claim. \square

If we linearize the constraints in (3.0.2b) and (3.0.2c), we obtain the linearized tangential cone of the feasible region \mathcal{F} to be defined next.

Definition 3.1.9 The set

$$\mathcal{T}_{lin}(\mathcal{F}, x) \equiv \mathcal{T}_{lin}(g, h, x) := \{d \in \mathbb{R}^n : \nabla g_j(x)^T d \leq 0, j \in \mathcal{A}(x); \nabla h(x)^T d = 0\}$$

is called the linearized tangential cone at $x \in \mathcal{F}$ with \mathcal{F} in (3.0.1).

Lemma 3.1.10 For all $x \in \mathcal{F}$ we have that $\mathcal{T}(\mathcal{F}, x) \subseteq \mathcal{T}_{lin}(\mathcal{F}, x)$.

Proof. Let $d \in \mathcal{T}(\mathcal{F}, x)$ and let $(\eta^{(\ell)})_{\ell \in \mathbb{N}} \subset \mathbb{R}^+$ and $(x^{(\ell)})_{\ell \in \mathbb{N}} \subset \mathcal{F}$ be such that $x^{(\ell)} \rightarrow x$ and $d^{(\ell)} := \eta^{(\ell)}(x^{(\ell)} - x) \rightarrow d$ as $\ell \rightarrow \infty$. Then, since $g_j(x) = 0$, $g_j(x^{(\ell)}) \leq 0$, $j \in \mathcal{A}(x)$ and $x^{(\ell)} \in \mathcal{F}$,

$$0 \geq \eta^{(\ell)}(g_j(x^{(\ell)}) - g_j(x)) = \nabla g_j(x)^T d^{(\ell)} + \eta^{(\ell)} o(\|x^{(\ell)} - x\|), \quad j \in \mathcal{A}(x),$$

and

$$0 = \eta^{(\ell)}(h_i(x^{(\ell)}) - h_i(x)) = \nabla h_i(x)^T d^{(\ell)} + \eta^{(\ell)} o(\|x^{(\ell)} - x\|) \quad i = 1, \dots, m,$$

we obtain for $\ell \rightarrow \infty$ that $\nabla g_j(x)^T d \leq 0$ for $j \in \mathcal{A}(x)$ as well as $\nabla h(x)^T d = 0$, which means that $d \in \mathcal{T}_{lin}(\mathcal{F}, x)$. \square

The reverse inclusion is in general false. Hence, the following is in general an assumption and our first (CQ).

Definition 3.1.11 The condition

$$\mathcal{T}_{lin}(\mathcal{F}, x) = \mathcal{T}(\mathcal{F}, x) \tag{ACQ}$$

is called Abadie Constraint Qualification³ for $x \in \mathcal{F}$.

From Theorem 3.1.8, we immediately deduce the following result.

Corollary 3.1.12 Let x^* be a local solution of (3.0.2) such that (ACQ) holds for x^* . Then,

- (i) $x^* \in \mathcal{F}$;
- (ii) $\nabla f(x^*)^T d \geq 0$ for all $d \in \mathcal{T}_{lin}(\mathcal{F}, x^*)$.

\square

Next, we will slightly weaken (ACQ). To this end, we need one more notation.

Definition 3.1.13 (a) Let $\emptyset \neq \mathcal{C} \subset \mathbb{R}^n$ be a cone. Then,

$$\mathcal{C}^\circ := \{v \in \mathbb{R}^n : v^T d \leq 0 \quad \forall d \in \mathcal{C}\}$$

is called polar cone of \mathcal{C} , see Figure 3.5

(b) The condition

$$\mathcal{T}_{lin}(\mathcal{F}, x)^\circ = \mathcal{T}(\mathcal{F}, x)^\circ \tag{GCQ}$$

is called Guignard Constraint Qualification⁴ for $x \in \mathcal{F}$.

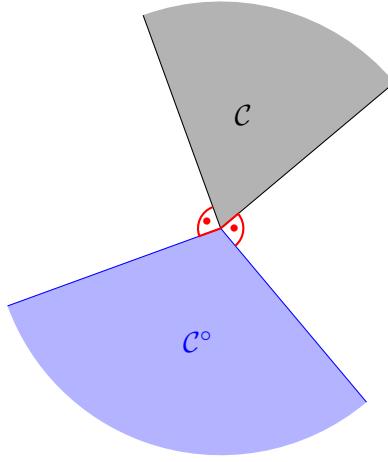
It is obvious that for some $x \in \mathcal{F}$ with (ACQ) also (GCQ) holds. Moreover, the statements of Corollary 3.1.12 also hold if (ACQ) is replaced by (GCQ).

Definition 3.1.14 Any condition for $x \in \mathcal{F}$ that implies (GCQ) is called Constraint Qualification (CQ) for x .

We need one more preparation.

³Jean M. Abadie (1919-2014)

⁴Monique Guignard-Spielberg (1947-)

Figure 3.5: Polar cone \mathcal{C}° of \mathcal{C} .

Lemma 3.1.15 (Farkas (1912)⁵) Let $B \in \mathbb{R}^{m \times n}$, $C \in \mathbb{R}^{p \times n}$, $g \in \mathbb{R}^n$ and define the cone

$$\mathcal{C} := \{B^T u + C^T v : \mathbb{R}^p \ni v \geq 0, u \in \mathbb{R}^m\} \subset \mathbb{R}^n.$$

Given any $g \in \mathbb{R}^n$, the following statements are equivalent:

- (i) For all $d \in \mathbb{R}^n$ with $Cd \leq 0$ and $Bd = 0$ it holds that $g^T d \leq 0$;
- (ii) $g \in \mathcal{C}$, i.e., there exist $v \in \mathbb{R}^p$, $v \geq 0$ and $u \in \mathbb{R}^m$ such that $g = B^T u + C^T v$.

Proof. Assume that (ii) is valid, i.e., $g = B^T u + C^T v \in \mathcal{C}$ for $0 \leq v \in \mathbb{R}^p$ and $u \in \mathbb{R}^m$. Now, let $d \in \mathbb{R}^n$ with $Cd \leq 0$ and $Bd = 0$. Then,

$$g^T d = (B^T u + C^T v)^T d = u^T \underbrace{Bd}_{=0} + \underbrace{v^T}_{\geq 0} \underbrace{Cd}_{\leq 0} \leq 0,$$

i.e., (i). Now assume that $g \notin \mathcal{C}$, which is a closed set (even though seemingly obvious, this needs to be proven, see [14, Lemma 12.15]). Let $\hat{s} \in \mathcal{C}$ be such that

$$\hat{s} = \arg \text{dist}(g; \mathcal{C}), \quad \text{where } \text{dist}(g; \mathcal{C}) := \min_{c \in \mathcal{C}} \|g - c\|_2 > 0,$$

and $\|\cdot\|_2$ denotes the Euclidean norm. Since \mathcal{C} is a cone, we have $\alpha \hat{s} \in \mathcal{C}$ for all $\alpha \in \mathbb{R}^+$. The function

$$\phi : \mathbb{R}^+ \rightarrow \mathbb{R} \quad \text{defined by } \phi(\alpha) := \|\alpha \hat{s} - g\|_2^2$$

takes its minimum at $\alpha^* = 1$, hence

$$0 = \phi'(1) = 2(\hat{s} - g, \hat{s})_2 \quad \text{so that } \hat{s}^T (\hat{s} - g) = 0. \tag{3.1.1}$$

Now, let $s \in \mathcal{C}$ be arbitrary. Hence, by convexity of \mathcal{C} and the minimality of \hat{s}

$$\|\hat{s} + \theta(s - \hat{s}) - g\|_2^2 \geq \|\hat{s} - g\|_2^2 \quad \text{for all } \theta \in [0, 1].$$

This yields

$$\begin{aligned} 0 &\leq \|\hat{s} + \theta(s - \hat{s}) - g\|_2^2 - \|\hat{s} - g\|_2^2 \\ &= \|\hat{s} - g\|_2^2 + 2\theta(\hat{s} - g)^T (s - \hat{s}) + \theta^2 \|s - \hat{s}\|_2^2 - \|\hat{s} - g\|_2^2 \\ &= 2\theta(\hat{s} - g)^T (s - \hat{s}) + \theta^2 \|s - \hat{s}\|_2^2. \end{aligned}$$

⁵Gyula Farkas (1847-1930)

Dividing this inequality by θ and taking the limit as $\theta \rightarrow 0$ yields by (3.1.1) the inequality $0 \leq (\hat{s} - g)^T(s - \hat{s}) = s^T(\hat{s} - g)$. Defining $d := g - \hat{s}$ implies that

$$d^T s \leq 0 \quad \text{for all } s \in \mathcal{C}. \quad (3.1.2)$$

From (3.1.2) we deduce that $d^T(B^T u + C^T v) \leq 0$ for all $\mathbb{R}^p \ni v \geq 0$ and all $u \in \mathbb{R}^m$. Choosing $v = 0$ yields $u^T(Bd) \leq 0$ for all $u \in \mathbb{R}^m$, which implies that $Bd = 0$. Fixing $u = 0$ gives $v^T Cd \leq 0$ for all $\mathbb{R}^p \ni v \geq 0$, which shows that $Cd \leq 0$. Hence, d satisfies the conditions of (i). However, since $g \notin \mathcal{C}$ and $\hat{s} \in \mathcal{C}$, we have $d \neq 0$ and

$$d^T g = d^T(d + \hat{s}) = \|d\|_2^2 + d^T \hat{s} = \|d\|_2^2 + \underbrace{\hat{s}^T(g - \hat{s})}_{=0 \text{ by (3.1.1)}} = \|d\|_2^2 > 0,$$

so that (i) would be wrong and that we obtain the claimed equivalence. \square

Remark 3.1.16 The statement of Farkas' lemma can also be expressed as follows: It holds

- (a) either $g \in \mathcal{C}$
 - (b) or there exists some $d \in \mathbb{R}^n$ satisfying $g^T d < 0$, $Cd \geq 0$, $Bd = 0$,
- but not both, see Figure 3.6 (this statement follows directly by the above proof). This is one instance of a separating hyperplane theorem, see lectures on optimization. Moreover, this explains why the statement of also known as Farkas' lemma of alternative.

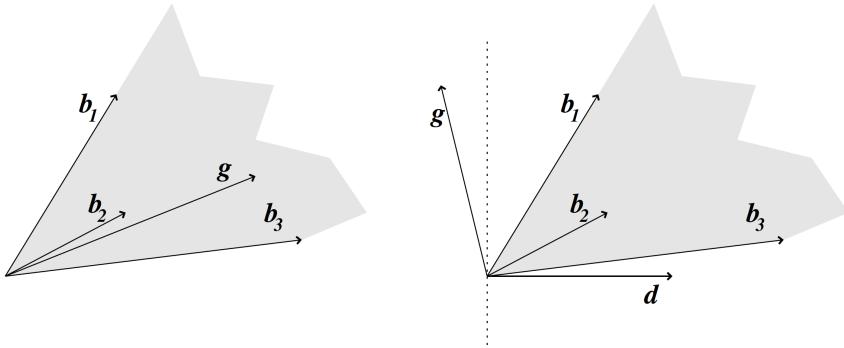


Figure 3.6: Farkas' lemma states the following: Either $g \in \mathcal{C}$ (left) or there is a separating hyperplane (right). Figure taken from [14, p. 327].

Now we are in position to prove the famous necessary first-order optimality conditions which are also known as Karush-Kuhn-Tucker or short *KKT conditions*.⁶

Theorem 3.1.17 (KKT first-order optimality conditions) Let x^* be a local solution of (3.0.2) such that a (CQ) holds for x^* . Then, the KKT conditions hold, namely there exist Lagrange multipliers $\lambda^* \in \mathbb{R}^m$ and $\mu^* \in \mathbb{R}^p$ such that

$$\nabla f(x^*) + (\lambda^*)^T \nabla h(x^*) + (\mu^*)^T \nabla g(x^*) = 0 \quad (\text{multiplier rule}), \quad (3.1.3a)$$

$$h(x^*) = 0 \quad (\text{equality constraints}), \quad (3.1.3b)$$

$$\mu^* \geq 0, \quad g(x^*) \leq 0, \quad (\mu^*)^T g(x^*) = 0 \quad (\text{complementarity condition}). \quad (3.1.3c)$$

Proof. Let x^* be a local solution of (3.0.2) such that a (CQ) holds for x^* . By Corollary 3.1.12 (i) we then have $x^* \in \mathcal{F}$, in particular (3.1.3b) and also $g(x^*) \leq 0$ (the second complementarity condition).

⁶William Karush (1917-1997), Harold W. Kuhn (1925-2014), Albert W. Tucker (1905-1995)

Moreover, from Corollary 3.1.12 (ii), we obtain $-\nabla f(x^*)^T d \leq 0$ for all $d \in \mathcal{T}_{\text{lin}}(\mathcal{F}, x^*)$, i.e., for all $d \in \mathbb{R}^n$ with $\nabla g_j(x^*)^T d \leq 0$, $j \in \mathcal{A}(x^*)$ and $\nabla h(x^*)^T d = 0$. Now, we can use Farkas' lemma (Lemma 3.1.15) for $C = (\nabla g_j(x^*)^T)_{j \in \mathcal{A}(x^*)}$, $B = \nabla h(x^*)^T$ and $g = -\nabla f(x^*)$, i.e., $g^T d \leq 0$ and $Bd = 0$. Thus, by Lemma 3.1.15, there exist $v \in \mathbb{R}^{\#\mathcal{A}(x^*)}$, $v \geq 0$ and $u \in \mathbb{R}^m$ with $g = C^T v + B^T u$, i.e.,

$$-\nabla f(x^*) = (\nabla g_j(x^*))_{j \in \mathcal{A}(x^*)}^T v + \nabla h(x^*)^T u.$$

Now we define $\mu^* \in \mathbb{R}^p$ by $(\mu^*)_{j \in \mathcal{A}(x^*)} := v$ and $(\mu^*)_{j \in \mathcal{I}(x^*)} := 0$ as well as $\lambda^* := u$, so that we obtain the multiplier rule in (3.1.3a). Finally, $\mu^* \geq 0$ by definition and

$$(\mu^*)^T g(x^*) = (\mu^*)_{j \in \mathcal{A}(x^*)}^T \underbrace{(g_j(x^*))_{j \in \mathcal{A}(x^*)}}_{=0} = 0,$$

by definition of $\mathcal{A}(x^*)$, which proves complementarity condition in (3.1.3c) and completes the proof. \square

Definition 3.1.18 The function $\mathcal{L} : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^p \rightarrow \mathbb{R}$ defined by

$$\mathcal{L}(x, \lambda, \mu) := f(x) + \lambda^T h(x) + \mu^T g(x) \quad (3.1.4)$$

is called Lagrange function of (3.0.2), λ and μ are called Lagrange multipliers for equality and inequality constraints, respectively.

Definition 3.1.19 We call the constrained optimization problem (3.0.2) convex if the functions f as well as g_j , $j = 1, \dots, p$, are convex and the function h is affine linear.

Then, the feasible region is a convex set and we get the following statement.

Theorem 3.1.20 (KKT for convex problems) Let (3.0.2) be convex, then

- (a) any local solution of (3.0.2) is also a global solution;
- (b) if a local solution x^* of (3.0.2) satisfies a (CQ), then the KKT conditions (3.1.3) hold;
- (c) on the other hand, if x^* satisfies the KKT conditions (3.1.3), then x^* is a global solution of (3.0.2).

Proof. (a) Let x^* be a local solution of (3.0.2) and $x \in \mathcal{F}$ be arbitrary. Set $d := x - x^*$. For any $t \in [0, 1]$ we have that $x^* + td \in \mathcal{F}$ and we get for $t \in [0, 1]$ sufficiently small by convexity

$$0 \leq f(x^* + td) - f(x^*) \leq (1-t)f(x^*) + t f(x) - f(x^*) = t(f(x) - f(x^*)),$$

which shows that x^* is a global solution of (3.0.2), i.e., (a). Next, (b) follows by Theorem 3.1.17.

As for (c) assume that x^* satisfies the KKT conditions (3.1.3) and let $x \in \mathcal{F}$ be arbitrary. Set $d := x - x^*$. For $j = 1, \dots, p$, we get by (3.1.3c), $x \in \mathcal{F}$ and convexity that

$$\mu_j^* \nabla g_j(x^*)^T d \leq \underbrace{\mu_j^*}_{\geq 0} \underbrace{(g_j(x) - g_j(x^*))}_{\leq 0} \leq \mu_j^* g_j(x^*)$$

using Theorem 2.1.8. Hence, $(\mu^*)^T \nabla g(x^*)^T d \leq -(\mu^*)^T g(x^*) = 0$ by (3.1.3c). Furthermore, since h is affine, we have $\nabla h(x^*)^T d = h(x) - h(x^*) = 0$. Finally, convexity and (3.1.3) yield

$$f(x) - f(x^*) \geq \nabla f(x^*)^T d = -(\lambda^*)^T \underbrace{\nabla h(x^*)^T d}_{=0} - \underbrace{(\mu^*)^T \nabla g(x^*)^T d}_{\leq 0} \geq 0,$$

which proves (c). \square

For completeness, we mention that there are also necessary and sufficient second-order optimality conditions involving the Hessian $\nabla^2 f$ of the objective function and a second-order constraint qualification (CQ2). It turns out that the KKT conditions (3.1.3) are also necessary but in addition the Hessian w.r.t. x of the Langrangian

$$\nabla_{xx}^2 \mathcal{L}(x^*, \lambda^*, \mu^*)$$

needs to be positive definite on a certain cone defined by the constraints. Since we want to concentrate on describing *numerical* methods for solving constrained optimization problems, we omit a detailed description of such optimality conditions here and refer to [25, §16.4] instead.

3.2 DUALITY

The aim of this section is to associate a dual problem to (3.0.2) (which is then called *primal* problem), which is often equivalent to (3.0.2) (and sometimes easier to solve) and also yields lower bounds for the optimality value. Dual problems can also be used for sensitivity analysis and the development of adaptive numerical methods for partial differential equations (PDEs) and PDE-constrained optimization.

3.2.1 The dual problem

Let us rephrase the notation for (3.0.2) in order to distinguish between primal and dual problems. Again, let $f : \mathbb{R}^n \rightarrow \mathbb{R}$, $h : \mathbb{R}^n \rightarrow \mathbb{R}^m$, $g : \mathbb{R}^n \rightarrow \mathbb{R}^p$ with $m, n, p \in \mathbb{N}$, $m + p \leq n$, and denote (similar to (3.0.1)) by

$$\mathcal{F}_P := \{x \in \mathbb{R}^n : h(x) = 0, g(x) \leq 0\} \quad (3.2.1)$$

the (*primal*) *feasible region*. Then, (3.0.2) becomes the *primal problem*, namely

$$(P) \quad \begin{cases} f(x) \rightarrow \min, & x \in \mathbb{R}^n \\ \text{s.t. } x \in \mathcal{F}_P. \end{cases} \quad (3.2.2)$$

According Definition 3.1.18, we define the (*primal*) Lagrange function for (3.2.2) depending on the variables $\lambda \in \mathbb{R}^m$ (the equality Lagrange multiplier) and $\mu \in \mathbb{R}^p$ (the inequality Lagrange multiplier) as follows

$$\begin{aligned} \mathcal{L}(x, \lambda, \mu) &:= f(x) + \langle \lambda, h(x) \rangle + \langle \mu, g(x) \rangle = f(x) + \lambda^T h(x) + \mu^T g(x) \\ &= f(x) + \sum_{i=1}^m \lambda_i h_i(x) + \sum_{j=1}^p \mu_j g_j(x). \end{aligned} \quad (3.2.3)$$

Now, observe that

$$p(x) := \sup_{\lambda \in \mathbb{R}^m, \mu \in \mathbb{R}_+^p} \mathcal{L}(x, \lambda, \mu) = \begin{cases} f(x), & \text{if } x \in \mathcal{F}_P, \\ \infty, & \text{else.} \end{cases}$$

Hence, the primal problem (3.2.2) is equivalent to

$$\inf_{x \in \mathbb{R}^n} \sup_{\lambda \in \mathbb{R}^m, \mu \in \mathbb{R}_+^p} \mathcal{L}(x, \lambda, \mu)$$

where we may also replace the inf by a min. Hence, it seems natural to exchange inf and sup. Thus, we define the *dual function* $\varphi : \mathbb{R}^m \times \mathbb{R}^p \rightarrow \mathbb{R}$ by

$$\varphi(\lambda, \mu) := \inf_{x \in \mathbb{R}^n} \mathcal{L}(x, \lambda, \mu) \quad (3.2.4)$$

and the *dual problem* as

$$(D) \quad \begin{cases} \varphi(\lambda, \mu) \longrightarrow \max, & (\lambda, \mu) \in \mathbb{R}^m \times \mathbb{R}^p, \\ \text{s.t. } (\lambda, \mu) \in \mathcal{F}_D, \end{cases} \quad (3.2.5)$$

with the corresponding *dual feasible region*

$$\mathcal{F}_D := \{(\lambda, \mu) \in \mathbb{R}^m \times \mathbb{R}_+^p : \varphi(\lambda, \mu) = \inf_{x \in \mathbb{R}^n} \mathcal{L}(x, \lambda, \mu) > -\infty\}. \quad (3.2.6)$$

Note, that the dual problem is a *maximization* problem.

Remark 3.2.1 *The set \mathcal{F}_D is convex and the function φ is concave on \mathcal{F}_D .*

Proof. Let $x \in \mathbb{R}^n$, $\alpha \in [0, 1]$ and $(\lambda, \mu), (\tilde{\lambda}, \tilde{\mu}) \in \mathcal{F}_D$. Then,

$$\begin{aligned} \mathcal{L}(x, \alpha(\lambda, \mu) + (1 - \alpha)(\tilde{\lambda}, \tilde{\mu})) &= \alpha f(x) + (1 - \alpha)f(x) \\ &\quad + \alpha \langle \lambda, h(x) \rangle + (1 - \alpha) \langle \tilde{\lambda}, h(x) \rangle + \alpha \langle \mu, g(x) \rangle + (1 - \alpha) \langle \tilde{\mu}, g(x) \rangle \\ &= \alpha \mathcal{L}(x, \lambda, \mu) + (1 - \alpha) \mathcal{L}(x, \tilde{\lambda}, \tilde{\mu}) \\ &\geq \alpha \varphi(\lambda, \mu) + (1 - \alpha) \varphi(\tilde{\lambda}, \tilde{\mu}) \\ &> -\infty. \end{aligned}$$

From that we deduce $\alpha(\lambda, \mu) + (1 - \alpha)(\tilde{\lambda}, \tilde{\mu}) \in \mathcal{F}_D$ as well as

$$\begin{aligned} \varphi(\alpha(\lambda, \mu) + (1 - \alpha)(\tilde{\lambda}, \tilde{\mu})) &= \inf_{x \in \mathbb{R}^n} \mathcal{L}(x, \alpha(\lambda, \mu) + (1 - \alpha)(\tilde{\lambda}, \tilde{\mu})) \\ &\geq \alpha \inf_{x \in \mathbb{R}^n} \mathcal{L}(x, \lambda, \mu) + (1 - \alpha) \inf_{x \in \mathbb{R}^n} \mathcal{L}(x, \tilde{\lambda}, \tilde{\mu}) \\ &= \alpha \varphi(\lambda, \mu) + (1 - \alpha) \varphi(\tilde{\lambda}, \tilde{\mu}), \end{aligned}$$

which completes the proof. \square

Next we define the primal and dual *optimality values* as

$$p^* := \inf_{x \in \mathcal{F}_P} f(x), \quad (3.2.7)$$

$$d^* := \sup_{(\lambda, \mu) \in \mathcal{F}_D} \varphi(\lambda, \mu). \quad (3.2.8)$$

3.2.2 Weak duality theorem and the saddle point property

The following statement shows the connection of the primal and the dual problem.

Theorem 3.2.2 (Weak duality theorem) *Let $x \in \mathcal{F}_P$ and $(\lambda, \mu) \in \mathcal{F}_D$. Then,*

$$\varphi(\lambda, \mu) \leq f(x) = p(x), \quad (3.2.9)$$

in particular

$$d^* \leq p^*. \quad (3.2.10)$$

Proof. Let $x \in \mathcal{F}_P$ and $(\lambda, \mu) \in \mathcal{F}_D$. Then, $h(x) = 0$, $g(x) \leq 0$ and $\mu \geq 0$, which implies that

$$\varphi(\lambda, \mu) = \inf_{\tilde{x} \in \mathbb{R}^n} \mathcal{L}(\tilde{x}, \lambda, \mu) \leq \mathcal{L}(x, \lambda, \mu) = f(x) + \underbrace{\langle \lambda, h(x) \rangle}_{=0} + \underbrace{\langle \mu, g(x) \rangle}_{\geq 0} \leq f(x).$$

Thus, (3.2.9) and (3.2.10) immediately follow. \square

Theorem 3.2.3 (Saddle point property) Let $x^* \in \mathbb{R}^n$ and $(\lambda^*, \mu^*) \in \mathbb{R}^m \times \mathbb{R}_+^p$. Then, the following statements are equivalent:

(a) The pair (x^*, λ^*, μ^*) is a saddle point of \mathcal{L} , i.e.,

$$\mathcal{L}(x^*, \lambda, \mu) \leq \mathcal{L}(x^*, \lambda^*, \mu^*) \leq \mathcal{L}(x, \lambda^*, \mu^*) \quad \text{for all } (x, \lambda, \mu) \in \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}_+^p.$$

(b) The point x^* is a minimizer of the primal problem (P) in (3.2.2) and (λ^*, μ^*) is a maximizer of the dual problem (D) in (3.2.5) with

$$f(x^*) = \mathcal{L}(x^*, \lambda^*, \mu^*) = \varphi(\lambda^*, \mu^*).$$

Proof. “(a) \Rightarrow (b)”: It holds by (a) that

$$\begin{aligned} \mathcal{L}(x^*, \lambda^*, \mu^*) &\leq \inf_{x \in \mathbb{R}^n} \mathcal{L}(x, \lambda^*, \mu^*) \leq \sup_{(\lambda, \mu) \in \mathbb{R}^m \times \mathbb{R}_+^p} \inf_{x \in \mathbb{R}^n} \mathcal{L}(x, \lambda, \mu) \\ &\leq \inf_{x \in \mathbb{R}^n} \sup_{(\lambda, \mu) \in \mathbb{R}^m \times \mathbb{R}_+^p} \mathcal{L}(x, \lambda, \mu), \end{aligned}$$

where the last step can be seen as follows: For all $\tilde{x} \in \mathbb{R}^n$ it holds that

$$\sup_{(\lambda, \mu) \in \mathbb{R}^m \times \mathbb{R}_+^p} \inf_{x \in \mathbb{R}^n} \mathcal{L}(x, \lambda, \mu) \leq \sup_{(\lambda, \mu) \in \mathbb{R}^m \times \mathbb{R}_+^p} \mathcal{L}(\tilde{x}, \lambda, \mu).$$

Since \tilde{x} can be chosen arbitrarily, we conclude that

$$\sup_{(\lambda, \mu) \in \mathbb{R}^m \times \mathbb{R}_+^p} \inf_{x \in \mathbb{R}^n} \mathcal{L}(x, \lambda, \mu) \leq \inf_{x \in \mathbb{R}^n} \sup_{(\lambda, \mu) \in \mathbb{R}^m \times \mathbb{R}_+^p} \mathcal{L}(x, \lambda, \mu),$$

which is used in the above inequality. Furthermore, due to (a),

$$\mathcal{L}(x^*, \lambda^*, \mu^*) \leq \sup_{(\lambda, \mu) \in \mathbb{R}^m \times \mathbb{R}_+^p} \mathcal{L}(x^*, \lambda, \mu) \leq \mathcal{L}(x^*, \lambda^*, \mu^*),$$

so that in fact equality holds and $-\infty < \varphi(\lambda^*, \mu^*) = \mathcal{L}(x^*, \lambda^*, \mu^*) = f(x^*)$. Thus, x^* is a minimizer of (P) and (λ^*, μ^*) is a maximizer of (D).

“(b) \Rightarrow (a)”: In view of (b), it holds that

$$\begin{aligned} \varphi(\lambda^*, \mu^*) &= \inf_{x \in \mathbb{R}^n} \mathcal{L}(x, \lambda^*, \mu^*) \leq \mathcal{L}(x^*, \lambda^*, \mu^*) \\ &= f(x^*) + \underbrace{\langle \lambda^*, h(x^*) \rangle}_{=0} + \underbrace{\langle \mu^*, g(x^*) \rangle}_{\leq 0} \leq f(x^*). \end{aligned} \tag{3.2.11}$$

Since $f(x^*) = \varphi(\lambda^*, \mu^*)$, we have equality in (3.2.11), in particular

$$\langle \mu^*, g(x^*) \rangle = 0$$

and $\mathcal{L}(x^*, \lambda^*, \mu^*) = \inf_{x \in \mathbb{R}^n} \mathcal{L}(x, \lambda^*, \mu^*)$ since $x^* \in \mathcal{F}_P$. With that we have $\mathcal{L}(x^*, \lambda^*, \mu^*) = f(x^*) \leq \mathcal{L}(x, \lambda^*, \mu^*)$ for all $x \in \mathbb{R}^n$ and

$$\mathcal{L}(x^*, \lambda, \mu) = f(x^*) + \underbrace{\langle \lambda, h(x^*) \rangle}_{=0} + \underbrace{\langle \mu, g(x^*) \rangle}_{\leq 0} \leq f(x^*) = \mathcal{L}(x^*, \lambda^*, \mu^*)$$

for all $(\lambda, \mu) \in \mathbb{R}^m \times \mathbb{R}_+^p$ and therefore the saddle point property holds, i.e., $\mathcal{L}(x^*, \lambda, \mu) \leq \mathcal{L}(x^*, \lambda^*, \mu^*) \leq \mathcal{L}(x, \lambda^*, \mu^*)$. \square

Now we are ready to formulate and prove the optimality condition (recalling Theorem 2.1.8 as well as Theorem 3.1.17).

Theorem 3.2.4 Let $f \in C^1(\mathbb{R}^n)$, $g \in C^1(\mathbb{R}^n; \mathbb{R}^p)$, $h \in \mathbb{C}^1(\mathbb{R}^n, \mathbb{R}^m)$. Furthermore, assume that the primal problem (P) is convex.⁷ Let $x^* \in \mathcal{F}_P$. If there exist Lagrange multipliers $\lambda^* \in \mathbb{R}^m$, $\mu^* \in \mathbb{R}_+^p$ such that

$$\begin{cases} \nabla f(x^*) + \langle \lambda^*, \nabla h(x^*) \rangle + \langle \mu^*, \nabla g(x^*) \rangle = 0, \\ \mu_i^* g_i(x^*) = 0 \quad \forall 1 \leq i \leq p, \end{cases} \quad (3.2.12)$$

then x^* minimizes (P) .

Proof. Let $x \in \mathcal{F}_P$. Then, Theorem 2.1.8 (i) and the KKT condition (3.2.12) yield

$$f(x) - f(x^*) \geq \nabla f(x^*)^T (x - x^*) \stackrel{(3.2.12)}{=} -\langle \mu^*, \nabla g(x^*) \rangle (x - x^*) - \sum_{j=1}^m \lambda_j^* \nabla h_j(x^*) (x - x^*).$$

Next, since the function h is assumed to be affine linear, we have that

$$\nabla h_j(x^*)(x - x^*) = h_j(x) - h_j(x^*) = 0,$$

since $x, x^* \in \mathcal{F}_P$. As the function g is assumed to be convex, it follows

$$\begin{aligned} -\langle \mu^*, \nabla g(x^*) \rangle (x - x^*) &= -\sum_{i=1}^p \mu_i^* \nabla g_i(x^*) (x - x^*) \\ &\geq -\sum_{i=1}^p \mu_i^* (g_i(x) - g_i(x^*)) \stackrel{(3.2.12)}{=} -\sum_{i=1}^p \mu_i^* g_i(x). \end{aligned}$$

Thus, since $x \in \mathcal{F}_P$, we obtain

$$f(x) - f(x^*) \geq -\sum_{i=1}^p \mu_i^* g_i(x) \geq 0$$

with “= 0” for $x = x^*$, which shows that x^* is an optimal solution of (P) . \square

Remark 3.2.5 Recall Theorem 3.1.17 concerning first-order necessary optimality conditions for general optimization problems (3.0.2) satisfying a constraint qualification (CQ). In this general setting, the KKT conditions are necessary but not sufficient – we needed second-order conditions for that. For convex problems, Theorem 3.2.4 shows that the KKT conditions are also sufficient in the sense that x^* is a primal minimizer provided that (3.2.12) holds.

3.2.3 Examples

We describe some examples for which we derive the dual problem.

Example 3.2.6 (Linear-quadratic problem with equality constraints) We start by considering linear-quadratic problems with equality constraints, namely

$$(QP) \quad \begin{cases} \frac{1}{2} \langle x, Ax \rangle + \langle a, x \rangle \longrightarrow \min! \text{ w.r.t. } x \in \mathbb{R}^n, \\ \text{s.t. } Bx = b, \end{cases} \quad (3.2.13)$$

where $A \in \mathbb{R}^{n \times n}$ is assumed to be symmetric and positive definite and $B \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$ as well as $a \in \mathbb{R}^n$ are given. The Lagrange function corresponding to (3.2.13) formally reads

$$\mathcal{L}(x, \lambda) := \frac{1}{2} \langle x, Ax \rangle + \langle a, x \rangle + \langle \lambda, Bx - b \rangle, \quad \lambda \in \mathbb{R}^m.$$

⁷According to Definition 3.1.19 this means that f , g are convex and h is (affine) linear.

To derive the dual problem, we have

$$\begin{aligned}\varphi(\lambda) &= \inf_{x \in \mathbb{R}^n} \mathcal{L}(x, \lambda) = \inf_{x \in \mathbb{R}^n} \left\{ \frac{1}{2}x^T Ax + a^T x + \langle \lambda, Bx \rangle \right\} - \langle \lambda, b \rangle \\ &= \frac{1}{2}(x^*)^T Ax^* + \langle B^T \lambda + a, x^* \rangle - \langle \lambda, b \rangle,\end{aligned}$$

where the optimal infimizer x^* is given by the first-order optimality conditions, namely

$$Ax^* + a + B^T \lambda = 0, \quad \text{i.e.,} \quad x^* = -A^{-1}(a + B^T \lambda),$$

so that

$$\begin{aligned}\varphi(\lambda) &= \frac{1}{2}\langle a + B^T \lambda, A^{-1}(a + B^T \lambda) \rangle - \langle B^T \lambda + a, A^{-1}(a + B^T \lambda) \rangle - \langle \lambda, b \rangle \\ &= -\frac{1}{2}\langle a + B^T \lambda, A^{-1}(a + B^T \lambda) \rangle - \langle \lambda, b \rangle \\ &= -\frac{1}{2}\langle \lambda, BA^{-1}B^T \lambda \rangle - \langle \lambda, BA^{-1}a + b \rangle + \frac{1}{2}\langle a, A^{-1}b \rangle,\end{aligned}$$

Hence, the dual problem can be written as

$$(D) \quad \frac{1}{2}\langle \lambda, BA^{-1}B^T \lambda \rangle + \langle \lambda, BA^{-1}a + b \rangle \longrightarrow \min, \quad \lambda \in \mathbb{R}^m, \quad (3.2.14)$$

which is an unconstrained problem, i.e., $\mathcal{F}_D = \mathbb{R}^m$ with the optimal solution $\lambda^* = -(BA^{-1}B^T)^{-1}(BA^{-1}a + b)$. The matrix $BA^{-1}B^T$ is also known as the *Schur complement*⁸ of the optimality system

$$\begin{pmatrix} A & B^T \\ B & 0 \end{pmatrix} \begin{pmatrix} x \\ \lambda \end{pmatrix} = \begin{pmatrix} -a \\ b \end{pmatrix},$$

which is a saddle point problem, also termed *primal-dual optimality system* as it incorporates the primal and the dual optimality conditions. \diamond

Example 3.2.7 (Linear-quadratic problem with inequality constraints) Next, we focus on linear-quadratic problems with inequality constraints, namely

$$(QP) \quad \begin{cases} \frac{1}{2}\langle x, Ax \rangle + \langle a, x \rangle \longrightarrow \min! \text{ w.r.t. } x \in \mathbb{R}^n, \\ \text{s.t. } Cx \leq c, \end{cases} \quad (3.2.15)$$

where again $A \in \mathbb{R}^{n \times n}$ is assumed to be symmetric and positive definite and $C \in \mathbb{R}^{p \times n}$, $c \in \mathbb{R}^p$ as well as $a \in \mathbb{R}^n$ are given. The Lagrange function corresponding to (3.2.15) formally reads

$$\mathcal{L}(x, \mu) := \frac{1}{2}\langle x, Ax \rangle + \langle a, x \rangle + \langle \mu, c - Cx \rangle, \quad \mu \in \mathbb{R}_+^p.$$

To derive the dual problem, we follow the same path as before, namely

$$\begin{aligned}\varphi(\mu) &= \inf_{x \in \mathbb{R}^n} \mathcal{L}(x, \mu) = \inf_{x \in \mathbb{R}^n} \left\{ \frac{1}{2}x^T Ax + a^T x - \langle \mu, Cx \rangle \right\} + \langle \mu, c \rangle \\ &= \frac{1}{2}(x^*)^T Ax^* + \langle a - C^T \mu, x^* \rangle + \langle \mu, c \rangle,\end{aligned}$$

where again x^* is given by the first-order optimality conditions, namely

$$Ax^* + a - C^T \mu = 0, \quad \text{i.e.,} \quad x^* = A^{-1}(C^T \mu - a),$$

so that

$$\begin{aligned}\varphi(\mu) &= \frac{1}{2}\langle C^T \mu - a, A^{-1}(C^T \mu - a) \rangle - \langle C^T \mu - a, A^{-1}(C^T \mu - a) \rangle + \langle \mu, c \rangle \\ &= -\frac{1}{2}\langle C^T \mu - a, A^{-1}(C^T \mu - a) \rangle + \langle \mu, c \rangle \\ &= -\frac{1}{2}\langle \mu, CA^{-1}C^T \mu \rangle - \langle \mu, CA^{-1}a + c \rangle + \frac{1}{2}\langle a, A^{-1}a \rangle,\end{aligned}$$

⁸Issai Schur (1875-1941)

Hence, the dual problem can be written as

$$(D) \quad \begin{cases} \frac{1}{2}\langle\mu, CA^{-1}C^T\mu\rangle + \langle\mu, c - CA^{-1}a\rangle \longrightarrow \min, & \mu \in \mathbb{R}^p, \\ \text{s.t. } \mu \geq 0 \end{cases} \quad (3.2.16)$$

i.e., $\mathcal{F}_D = \mathbb{R}_+^p$. \diamond

Example 3.2.8 (Optimal control) Let $x : [0, T] \rightarrow \mathbb{R}^n$ be a function of time describing the *state* of some process for a time interval $[0, T]$, $T > 0$. The aim is to control the state in such a way that a given *cost function*⁹ is minimized. The control is modeled as a *function* $u : [0, T] \rightarrow \mathbb{R}^p$, which means that x and u are searched in infinite-dimensional function spaces, which, however, are typically discretized (yielding high-dimensional problems, though). The cost function to be minimized is written in an abstract form as

$$\mathcal{J}(x, u) := \int_0^T L(x(t), u(t)) dt + E(x(T)),$$

where $L : \mathbb{R}^n \times \mathbb{R}^p \rightarrow \mathbb{R}^+$ and $E : \mathbb{R}^n \rightarrow \mathbb{R}^+$ are so-called *distributed cost functions* and *terminal time control*. One could think of the second term as a deviation from a desired terminal time state and the first one as the required energy that is needed to apply the control over the time interval.

The evolution of the state over time is described by an initial value problem of an ordinary differential equation (ODE) of the form

$$\dot{x}(t) = f(x(t), u(t)), \quad t \in (0, T], \quad x(0) = x_0, \quad (3.2.17)$$

where $f : \mathbb{R}^n \times \mathbb{R}^p \rightarrow \mathbb{R}^n$ is a given function (allowing for well-posedness of the ODE-initial value problem by the Picard-Lindelöf theorem¹⁰) and $x_0 \in \mathbb{R}^n$ some given initial state. Then, we consider the *optimal control problem*

$$\begin{cases} \text{minimize } \mathcal{J}(x, u) \text{ for } x : [0, T] \rightarrow \mathbb{R}^n, u : [0, T] \rightarrow \mathbb{R}^p, \\ \text{s.t. (3.2.17).} \end{cases} \quad (3.2.18)$$

The Lagrange function reads

$$\begin{aligned} \mathcal{L}(x, u, \lambda) &:= \mathcal{J}(x, u) - \int_0^T \lambda(t)^T (\dot{x}(t) - f(x(t), u(t))) dt \\ &= \int_0^T (H(x(t), u(t), \lambda(t)) - \lambda(t)^T \dot{x}(t)) dt + E(x(T)), \end{aligned}$$

with the *Hamilton function*¹¹

$$H(x(t), u(t), \lambda(t)) := L(x(t), u(t)) + \lambda(t)^T f(x(t), u(t)),$$

which collects the control-dependent part. We determine the partial (Fréchet¹²) derivatives of the Lagrange function as follows: Let $\delta x : [0, T] \rightarrow \mathbb{R}$ with $\delta x(0) = 0$ be some variation, then

$$\begin{aligned} \mathcal{L}_x(x, u, \lambda)[\delta x] &= \int_0^T \{ H_x(x(t), u(t), \lambda(t)) \delta x(t) - \lambda(t)^T \partial_t \delta x(t) \} dt + \partial_x E(x(T)) \delta x(T) \\ &= \int_0^T \{ (\nabla_x H(x(t), u(t), \lambda(t)) + \dot{\lambda}(t))^T \delta x(t) - \partial_t (\lambda(t)^T \delta x(t)) \} dt + \partial_x E(x(T)) \delta x(T) \\ &= \int_0^T \{ (\nabla_x H(x(t), u(t), \lambda(t)) + \dot{\lambda}(t))^T \delta x(t) \} dt + (\nabla_x E(x(T)) - \lambda(T))^T \delta x(T) \end{aligned}$$

⁹Sometimes also called *objective* or *merit* function.

¹⁰Charles Émile Picard (1856-1941), Ernst Leonard Lindelöf (1870-1946).

¹¹Sir William Rowan Hamilton (1805-1865)

¹²René Maurice Fréchet (1878-1973)

by the chain rule and the fundamental theorem of calculus (or, alternatively by integration by parts). Hence, if $\nabla \mathcal{L}(x^*, u^*, \lambda^*) = 0$, then the optimal Lagrange multiplier λ^* is determined by the *terminal value problem* of an ODE

$$\begin{aligned}\dot{\lambda}(t) &= -\nabla_x H(x^*(t), u^*(t), \lambda(t)), & t \in (0, T), \\ \lambda(T) &= \nabla_x E(x^*(T)).\end{aligned}$$

Next, for a variation $\delta u : [0, T] \rightarrow \mathbb{R}$, we get that

$$\begin{aligned}\mathcal{L}_u(x, u, \lambda)[\delta u] &= \int_0^T \left\{ H_u(x(t), u(t), \lambda(t)) \delta x(t) - \lambda(t)^T \partial_t \delta x(t) \right\} dt \\ &= \int_0^T (L_u(x(t), u(t)) \delta u(t) + \lambda(t)^T f_u(x(t), u(t))[\delta u]) dt.\end{aligned}$$

Collecting these findings, we obtain necessary (not sufficient) first order optimality conditions as follows

$$\begin{aligned}\partial_t x^*(t) &= f(x^*(t), u^*(t)), & t \in (0, T), & \text{(ODE model)} \\ x^*(0) &= x_0 & & \text{(initial state)} \\ \partial_t \lambda^*(t) &= -H_x(x^*(t), u^*(t), \lambda^*(t))^T, & t \in (0, T), & \text{(adjoint equations)} \\ \lambda^*(T) &= \nabla_x E(x^*(T)) & & \text{(adjoint terminal state)} \\ u^*(t) &= \arg \min_u H(x^*(t), u(t), \lambda^*(t)), & t \in (0, T). & \text{(minimum principle)}\end{aligned}$$

There are several observations from these equations. First, of all, we obtain a boundary value problem for the coupled system and observe that the adjoint ODE is backwards in time (i.e., a terminal value problem). The minimum principle is historically also known as *Pontryagin's Maximum Principle*¹³ and is closely related to the *Hamilton-Jacobi-Bellman equation*¹⁴ $-\partial_t \mathcal{J}(t, x) = \min_u H(x, \nabla \mathcal{J}(t, x), u)$ for some cost function \mathcal{J} of time and state under a control u , see e.g. [22]. \diamond

As the previous one, also the following example relies on *infinite-dimensional* spaces instead of \mathbb{R}^n , \mathbb{R}^m and \mathbb{R}^p . It shows that the above framework is also useful in this more general setting, which is needed for PDE¹⁵-constrained optimization. A detailed analysis of the arising problems would require more sophisticated functional analysis and goes beyond the scope of this lecture.

Example 3.2.9 (Numerical solution of (partial) differential equations) Let V be a linear space, possibly of infinite dimension (e.g. a Sobolev¹⁶ space, which is appropriate as a solution space of elliptic partial differential equation). We view V as a so-called *state space*, i.e., the space which is appropriate for describing the state of a physical variable. As an example, we think of fluid dynamic and $u \in V$ would describe the velocity. A partial differential equation (PDE) can often be written in the following form:

$$\text{determine } u \in V \text{ such that } a(u, v) = f(v) \text{ for all } v \in V,$$

where $a : V \times V \rightarrow \mathbb{R}$ is bilinear and $f : V \rightarrow \mathbb{R}$ is linear.

Often, one is not interested in the full state, but on certain output quantities such as drag or lift coefficient of an airfoil. This can be modeled in terms of an *output function* $J : V \rightarrow \mathbb{R}$.

PDE-constrained optimization would e.g. amount optimizing the lift of an airfoil. The lift depends on the velocity, which in turn is determined as the solution of a PDE. Hence, the primal problem for the minimization of the output quantity takes the form

$$(P) \left\{ \begin{array}{l} J(u) \longrightarrow \min!, \quad u \in V \\ \text{s.t. } a(u, v) = f(v) \quad \forall v \in V. \end{array} \right.$$

¹³Lev Semyonovich Pontryagin (1908-1988)

¹⁴Sir William Rowan Hamilton (1805-1865), Carl Gustav Jacob Jacobi (1804-1851), Richard Ernest Bellman (1920-1984)

¹⁵PDE: partial differential equation

¹⁶Sergei Lvovich Sobolev (1908-1989)

Of course, (P) is an infinite-dimensional problem and cannot be solved numerically in a straightforward way. However, we can detail the dual framework also in the infinite-dimensional setting. In fact, with the Lagrange function the dual problem is computable. We obtain

$$\begin{aligned}\mathcal{L}(u, \mu) &:= J(u) + a(u, \lambda) - f(\mu), \quad u, \mu \in V \\ \varphi(\mu) &:= \inf_{u \in V} \mathcal{L}(u, \lambda) = \inf_{u \in V} \{J(u) + a(u, \lambda) - f(\lambda)\} = \inf_{u \in V} \{J(u) + a(u, \lambda)\} - f(\lambda)\end{aligned}$$

Hence, we get that

$$\varphi(\lambda) := \begin{cases} -f(\lambda) & \text{if } a(v, \lambda) = -J(v) \text{ for all } v \in V, \\ -\infty & \text{else.} \end{cases}$$

That implies that the dual problem reads

$$(D) \begin{cases} -f(\lambda) \longrightarrow \max! \\ \text{s.t. } \lambda \in \mathcal{F}_D = \{\lambda \in V : \inf_{u \in V} \mathcal{L}(u, \lambda) > -\infty\} = \{\lambda \in V : a(v, \lambda) = -J(v) \text{ for all } v \in V\}. \end{cases}$$

This shows that $\mathcal{F}_D = \{\lambda^*\}$ consists of only one element, namely the unique solution λ^* of the adjoint problem!

$$\text{find } \mu \in V : \quad a(v, \lambda) = -J(v) \text{ for all } v \in V$$

and thus we get that $d^* = -f(\lambda^*)$ – the optimal value can be determined by the solution of a linear problem! For more details, we refer e.g. to [10]. \diamond

3.3 SLACK VARIABLES

It turns out to be convenient to rewrite inequality constraints by using so-called *slack variable*. In fact, the problem given $f : \mathbb{R}^n \rightarrow \mathbb{R}$, $g : \mathbb{R}^n \rightarrow \mathbb{R}^p$, $h : \mathbb{R}^n \rightarrow \mathbb{R}^m$

$$\begin{cases} \text{minimize} & f(x), \quad x \in \mathbb{R}^n, \\ \text{s.t.} & h(x) = 0, \\ & g(x) \leq 0, \end{cases}$$

can be rewritten as

$$\begin{cases} \text{minimize} & f(x), \quad x \in \mathbb{R}^n, s \in \mathbb{R}^p \\ \text{s.t.} & h(x) = 0, \\ & g(x) + s = 0, \\ & s \geq 0. \end{cases} \quad (3.3.1)$$

Obviously, the dimension of the problem is increased from n to $n + p$ by introducing the slack variable. On the other hand, the inequality constraint is significantly easier, as it is only a non-negativity bound on a variable.

Example 3.3.1 We continue Example 3.2.7 and rewrite (3.2.15) by inserting a slack variable, i.e.,

$$(QP_1) \quad \begin{cases} \frac{1}{2} \langle x, Ax \rangle + \langle a, x \rangle \longrightarrow \min! \text{ w.r.t. } x \in \mathbb{R}^n, \\ \text{s.t. } Cx + s = c \\ \quad s \geq 0. \end{cases} \quad (3.3.2)$$

The Lagrange function reads for $x \in \mathbb{R}^n$, $\lambda \in \mathbb{R}^p$ and $s, \mu \in \mathbb{R}_+^p$

$$\mathcal{L}(x, s; \lambda, \mu) = \frac{1}{2}\langle x, Ax \rangle + \langle a, x \rangle + \langle \lambda, Cx + s - c \rangle - \langle \mu, s \rangle,$$

so that

$$\begin{aligned} \varphi(\lambda, \mu) &= \inf_{x \in \mathbb{R}^n, s \in \mathbb{R}_+^p} \mathcal{L}(x, s; \lambda, \mu) = \inf_{x \in \mathbb{R}^n, s \in \mathbb{R}_+^p} \underbrace{\left\{ \frac{1}{2}\langle x, Ax \rangle + \langle a + C^T \lambda, x \rangle + \langle \lambda - \mu, s \rangle \right\}}_{=: F(x, s)} - \langle \lambda, c \rangle \\ &= \dots = -\frac{1}{2}\langle a + C^T \lambda, A^{-1}(a + C^T \lambda) \rangle - \langle \lambda, c \rangle, \quad \lambda \geq 0, \end{aligned}$$

which can be seen as follows: in order to determine $\inf_{x \in \mathbb{R}^n, s \in \mathbb{R}_+^p} F(x, s)$, we determine the partial derivatives:

$$\frac{\partial F(x, s)}{\partial x} = Ax + a + C^T \lambda, \quad \frac{\partial F(x, s)}{\partial s} = \lambda - \mu.$$

Setting this to zero yields the conditions $x = -A^{-1}(a + C^T \lambda)$ and $\mu = \lambda$ (which implies that $\lambda \geq 0$). Inserting this into $F(x, s)$ yields

$$\begin{aligned} \inf_{x \in \mathbb{R}^n, s \in \mathbb{R}_+^p} F(x, s) &= \frac{1}{2}\langle -A^{-1}(a + C^T \lambda), -(a + C^T \lambda) \rangle + \langle a + C^T \lambda, -A^{-1}(a + C^T \lambda) \rangle + \underbrace{\langle \lambda - \mu, s \rangle}_{=0} \\ &= -\frac{1}{2}\langle a + C^T \lambda, A^{-1}(a + C^T \lambda) \rangle - \langle \lambda, c \rangle, \quad \lambda \geq 0. \end{aligned}$$

Hence, the dual problem reads

$$(DQP_1) \quad \begin{cases} \frac{1}{2}\langle a + C^T \lambda, A^{-1}(a + C^T \lambda) \rangle + \langle \lambda, c \rangle \longrightarrow \min! \text{ w.r.t. } \lambda, \mu \in \mathbb{R}^p, \\ \text{s.t. } \lambda \geq 0, \end{cases}$$

which results in the following primal-dual optimality system

$$Ax + C^T \lambda + a = 0, \tag{3.3.3a}$$

$$Cx + s - c = 0, \tag{3.3.3b}$$

$$s_i \lambda_i = 0, \quad i = 1, \dots, p, \tag{3.3.3c}$$

$$s \geq 0, \lambda \geq 0, \tag{3.3.3d}$$

which we will reconsider later. \diamond

3.4 PENALTY METHOD

The basic idea of the *penalty method* is to transform the constrained problem (3.0.4) with equality constraints only into a sequence of unconstrained optimization problems, whose minimum points x_k^* converge towards the solution x^* of (3.0.4). For this purpose, we define the *penalty Lagrange function* $\mathcal{L}_\tau : \mathbb{R}^n \rightarrow \mathbb{R}$ via

$$\mathcal{L}_\tau(x) := f(x) + \frac{1}{2}\tau\|h(x)\|^2.$$

The parameter $\tau > 0$ is called *penalty parameter*. Due to the specific form of the penalty function, this approach is also known as *quadratic penalty method*, [14, §17.1].

Apparently, if all constraints are exactly fulfilled, the minimization of f is equivalent to the minimization of the penalty Lagrange function \mathcal{L}_τ on $\mathcal{F} = \{x \in \mathbb{R}^n : h(x) = 0\}$. However, considering the sequence of unconstrained optimization problems

$$\text{minimize } \mathcal{L}_{\tau_\ell}(x) \text{ in } \mathbb{R}^n, \tag{3.4.1}$$

the minimum points x_ℓ^* in general of course do not coincide with x^* . Choosing a sequence of penalty parameters $(\tau_\ell)_{\ell \in \mathbb{N}} \subset \mathbb{R}^+$ as a monotonically increasing sequence of positive penalty parameters, so that

$$\tau_\ell \longrightarrow \infty \quad \text{as } \ell \rightarrow \infty,$$

then, the objective function $\mathcal{L}_{\tau_\ell}(x)$ tends to infinity, except for the case that the constraints are fulfilled, i.e., $h(x) = 0$. Hence, we can at least hope that “under certain conditions” (to be described in detail below) the sequence $(x_\ell^*)_{\ell \in \mathbb{N}}$ of solutions of (3.4.1) converges towards the desired minimum x^* of f in \mathcal{F} . More precisely, the following statement holds.

Theorem 3.4.1 (Convergence of the penalty method) *Let $f : \mathbb{R}^n \rightarrow \mathbb{R}$ and $h : \mathbb{R}^n \rightarrow \mathbb{R}^m$ with $m \leq n$ be continuous functions on a closed set $K \subset \mathbb{R}^n$ and let the sequence of penalty parameters $\tau_\ell > 0$ be monotonically divergent, i.e.,*

$$\tau_\ell \longrightarrow \infty \text{ as } \ell \rightarrow \infty.$$

Let x_ℓ^ be a global minimum of the unconstrained problem ($\ell \in \mathbb{N}$)*

$$\text{minimize } \mathcal{L}_{\tau_\ell}(x) \text{ in } K \subset \mathbb{R}^n.$$

Then,

$$x_\ell^* \longrightarrow x^* \quad \text{as } \ell \rightarrow \infty,$$

where x^ is a global minimum of f in K under the constraint $h(x^*) = 0$.*

Proof. Let \bar{x} be a global minimum of f in K under the constraint $h(\bar{x}) = 0$, i.e., $f(\bar{x}) \leq f(x)$ for all $x \in K$ with $h(x) = 0$. Since x_ℓ^* minimizes each \mathcal{L}_{τ_ℓ} , we have that $\mathcal{L}_{\tau_\ell}(x_\ell^*) \leq \mathcal{L}_{\tau_\ell}(\bar{x})$, so that

$$f(x_\ell^*) + \frac{\tau_\ell}{2} \|h(x_\ell^*)\|^2 = \mathcal{L}_{\tau_\ell}(x_\ell^*) \leq \mathcal{L}_{\tau_\ell}(\bar{x}) = f(\bar{x}) + \frac{\tau_\ell}{2} \|h(\bar{x})\|^2 = f(\bar{x}), \quad (3.4.2)$$

which implies that

$$\|h(x_\ell^*)\|^2 \leq \frac{2}{\tau_\ell} [f(\bar{x}) - f(x_\ell^*)]. \quad (3.4.3)$$

Suppose that x^* is a limit point of $(x_\ell^*)_{\ell \in \mathbb{N}}$, then there is a subsequence converging towards x^* ; we shall denote this subsequence again by $(x_\ell^*)_{\ell \in \mathbb{N}}$. Then, taking the limit in (3.4.3) yields

$$\|h(x^*)\|^2 = \lim_{\ell \rightarrow \infty} \|h(x_\ell^*)\|^2 \leq \lim_{\ell \rightarrow \infty} \frac{2}{\tau_\ell} [f(\bar{x}) - f(x_\ell^*)] = 0,$$

since $\tau_\ell \rightarrow \infty$ as $\ell \rightarrow \infty$. Hence $h(x^*) = 0$. Next, since $\tau_\ell > 0$, $\|h(x_\ell^*)\| \geq 0$ and taking the limit in (3.4.2), again, possibly w.r.t. a subsequence, yields

$$f(x^*) \leq f(x^*) + \lim_{\ell \rightarrow \infty} \frac{\tau_\ell}{2} \|h(x_\ell^*)\|^2 \leq f(\bar{x}).$$

Since $h(x^*) = 0$ and $f(x^*) \leq f(\bar{x})$, the limit x^* is a global minimum. \square

Remarks 3.4.2 (i) For $\tau_\ell \rightarrow \infty$ the condition number of the unconstrained optimization problems (3.4.1) becomes worse and worse, which is of course a numerical challenge. Since the condition number depends on the penalty τ_ℓ , the sequence must not increase too fast. On the other hand, however, the sequence $(\tau_\ell)_{\ell \in \mathbb{N}}$ must also not increase too slow as well, because this would yield slow convergence.

To reduce the effort for the numerical solution of (3.4.1), one tries to choose the initial vectors for the iterative method to compute x_ℓ^* as good as possible. A possible choice is to use $x_{\ell-1}^*$ (i.e., the approximation of the minimum in the $(\ell-1)$ -st iteration) as initial vector for the ℓ -th iteration.

(ii) A possible strategy for the choice of the penalty parameter τ_ℓ is to start with a moderate τ_1 as initial value and then set $\tau_{\ell+1} := \rho \tau_\ell$ for $\ell \in \mathbb{N}$, where ρ is in practice some factor in the range of 4 to 10.

- (iii) We stress the fact that x_ℓ^* is required to be a global minimum of the unconstrained problem. We shall come back to this issue in the subsequent example.

Example 3.4.3 We consider the equality-constrained Rosenbrock problem introduced in Example 3.0.2 and use the penalty method to solve it. The results for the initial value of $x^{(0)} = (-1, -1)^T$ are shown in Table 3.1 and Figure 3.7. We see the crucial dependence on the penalty parameter τ . For small values of τ , the scheme converges close to the optimum. Then, we reach a critical value, where the scheme does not converge (we set the maximum number of iterations to 1000, so that this number in the last column means that the scheme did not converge within 1000 iterations). Beyond those critical values of τ , the scheme again converges, but not to the minimum of f under the constraint $h(x) = 0$, which would be on the right-hand side of the red circle in Figure 3.7.

τ	$h(x^{(k)})$	$\ \mathcal{L}'_\tau(x^{(k)})\ $	$\mathcal{L}_\tau(x^{(k)})$	k
0.1	0.20053625	0.00000011	0.00482732	18
0.5	0.06422977	0.00001169	0.00737766	20
1.0	0.03501234	0.00000002	0.00793956	21
1.4	0.02568807	0.00000032	0.00811943	26
1.45	0.02486157	0.00001791	0.00813540	31
1.46	0.02470180	0.00000005	0.00813847	27
1.47	-1.08872236	1.73093944	3.30391144	1000
1.5	-0.98057184	0.47396288	3.32359425	1000
1.55	-0.83567259	0.00000016	3.34476298	9
2.0	-0.51093435	0.00000000	3.43629636	7
5	-0.17095525	0.00000199	3.56512083	5
10	-0.08218609	0.00000537	3.60022130	5
50	-0.82095745	262.69469264	100.46233421	1000
1000	-0.10000434	0.00030975	145.99273487	7

Table 3.1: Equality-constrained Rosenbrock problem for $x^{(0)} = (-1, -1)^T$.

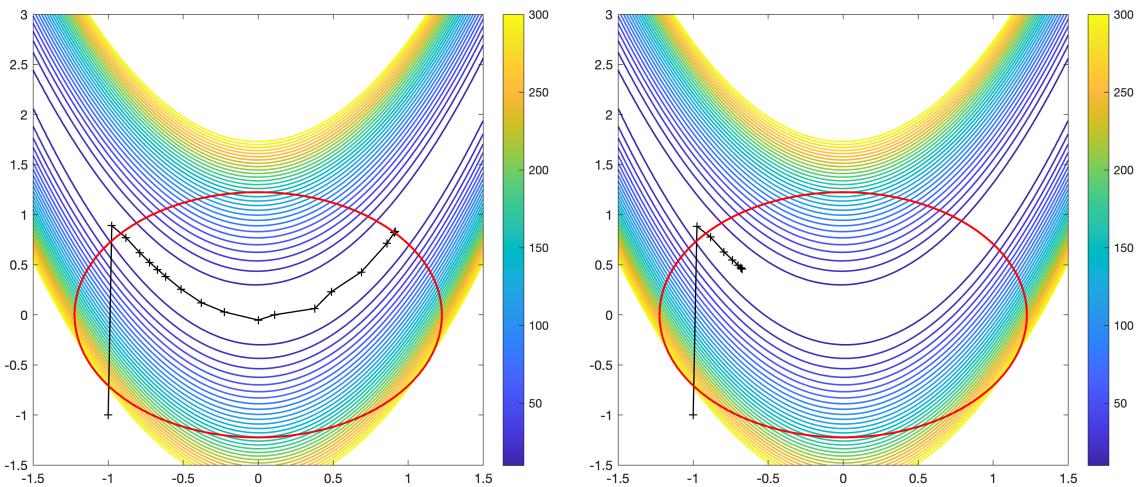


Figure 3.7: Iteration history for the equality-constrained Rosenbrock problem and $\tau = 1.46$ (left) and $\tau = 1.55$ (right) for $x^{(0)} = (-1, -1)$.

At a first glance, these observations seem to contradict the convergence statement of Theorem 3.4.1. However, Figure 3.7 already gives some indication what is going wrong here. In fact, for small values

of τ , the iteration converges at least close to the exact one (on the right-hand side of the red circle close to the global minimum of the Rosenbrock function), see the left graph in Figure 3.7. However, once we increase τ (and keep the same initial value $x^{(0)}$), the iteration goes towards points on the left-hand side of the red circle, far away from the exact minimum.

Let us now consider Figure 3.8, where we depict the level lines of \mathcal{L}_{10} . We clearly see that this function has *two* local minima. Apparently, using $x^{(0)} = (-1, -1)^T$ forces the internal Newton iteration to converge towards the left local minimum, which, however, is *not the global minimum*, but which was required in Theorem 3.4.1.

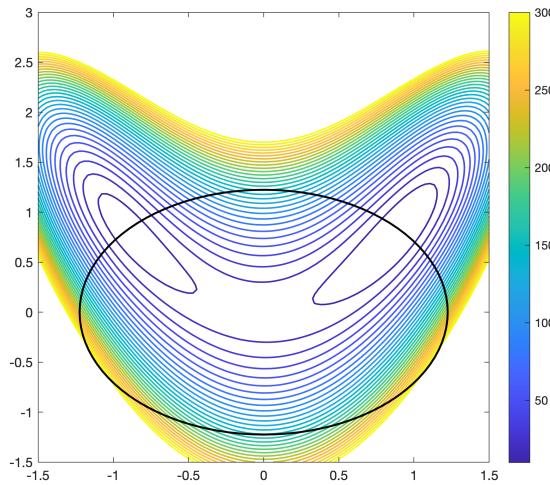


Figure 3.8: Penalty function \mathcal{L}_τ for the constrained Rosenbrock problem with $\tau = 10$. We clearly see the existence of two local minima.

To investigate this a bit more in detail, we use $x^{(0)} = (1, -1)^T$ as initial value. The results are shown in Table 3.2, where we show the iteration history for $\tau = 1$ converging towards the exact minimum. We see that things go nice until $\tau = 10$ and for larger values, we again obtain non-convergence of the iteration. Again, the reason is that the Newton method for larger values of τ does not converge towards the *global* minimum of \mathcal{L}_τ .

To overcome this problem, we use the same sequence of values for τ , but use the approximation determined by the current value of τ as initial value for the next value of τ . These results are shown in the last two lines of Table 3.2. As we see from the values of $h(x^{(k)})$, we come closer to the circle, i.e., closer to the exact constrained minimum. This example shows how sensitive the method is w.r.t. the choice of the initial value and also indicates a good strategy for choosing initial values.

Finally, we test the strategy to increase the value of τ iteratively and use the previous approximation as initial value for the next. We choose $\tau_1 = 1$ as initial value and $\tau_{\ell+1} := \rho\tau_\ell$ with $\rho = 5$ as in Example 3.4.2. In Figure 3.9, we display the iteration histories for $x^{(0)} = (-1, -1)$ (left) and $x^{(0)} = (1, -1)$ (right), both showing nice convergence towards the global minimum, see also Table 3.3. \diamond

Example 3.4.4 As before, one can of course perform analogous numerical experiments using the Himmelblau, Bazaraa-Shetty or the Dixon-Price function (in particular for higher space dimensions) introducing constraints. We leave this as coding exercises. \diamond

τ	$h(x^{(k)})$	$\ \mathcal{L}'_\tau(x^{(k)})\ $	$\mathcal{L}_\tau(x^{(k)})$	k
0.1	0.20053637	0.00000034	0.00482732	5
0.5	0.06422853	0.00000286	0.00737766	5
1.0	0.03501240	0.00000125	0.00793956	5
1.5	0.02408537	0.00000050	0.00815037	5
2.0	0.01835988	0.00000021	0.00826092	5
5	0.00756968	0.00000001	0.00846937	5
10	0.00382450	0.00000304	0.00854174	5
50	-0.81841295	261.01987347	97.45005947	1000
1000	-0.10000438	0.00021960	145.99273487	7
50*	0.00077138	0.00000000	0.00860074	2
1000*	0.00003895	0.00070030	0.00861490	1

Table 3.2: Equality-constrained Rosenbrock problem for $x^{(0)} = (1, -1)^T$. The last two lines use the approximation of the previous τ as initial value, i.e., the initial value for $\tau = 50$ is the approximation computed with $\tau = 10$.

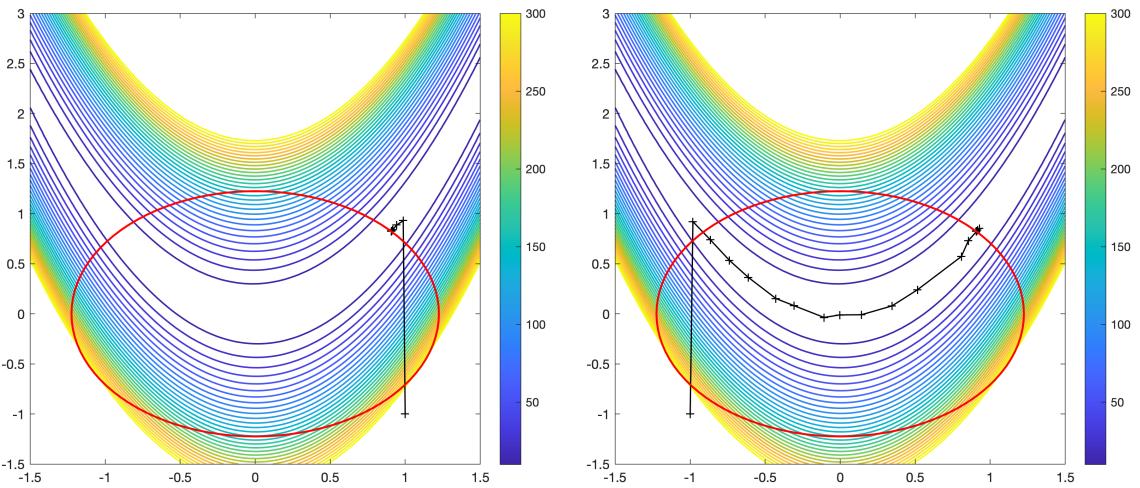


Figure 3.9: Iteration history for the penalty method for solving the equality-constrained Rosenbrock problem with increasing sequence of τ and new initial values for $x^{(0)} = (1, -1)$ (left) and $x^{(0)} = (-1, -1)$ (right).

τ	$f(x^{(k)})$	$\ h(x^{(k)})\ $
1	4.00000e+02	5.00000e-01
1	1.91059e-01	3.38596e-01
1	4.35387e-03	1.74714e-01
1	6.91818e-03	4.83460e-02
1	7.31856e-03	3.52433e-02
5	7.32663e-03	3.50124e-02
5	8.31125e-03	7.96488e-03
25	8.32611e-03	7.56981e-03
25	8.55550e-03	1.55954e-03
125	8.55627e-03	1.53948e-03
625	8.60368e-03	3.09788e-04
3125	8.61326e-03	6.18653e-05
15625	8.61517e-03	1.23693e-05
390625	8.61556e-03	2.47370e-06

τ	$f(x^{(k)})$	$\ h(x^{(k)})\ $
1	4.04000e+02	5.00000e-01
1	4.11861e+00	3.10896e-01
1	3.49164e+00	2.05307e-01
1	3.03288e+00	6.71518e-01
1	2.61324e+00	9.92220e-01
1	2.13662e+00	1.29330e+00
1	1.72768e+00	1.39875e+00
1	1.45411e+00	1.48716e+00
1	1.01768e+00	1.49987e+00
1	8.21935e-01	1.47981e+00
1	5.91038e-01	1.37433e+00
1	3.00034e-01	1.17800e+00
1	6.80882e-01	5.20818e-01
1	2.10842e-02	2.35280e-01
1	8.25740e-03	8.72020e-02
1	7.14889e-03	4.02028e-02
1	7.32573e-03	3.50382e-02
5	7.32663e-03	3.50123e-02
5	8.31125e-03	7.96488e-03
25	8.32611e-03	7.56981e-03
25	8.55550e-03	1.55954e-03
125	8.55627e-03	1.53948e-03
625	8.60368e-03	3.09788e-04
3125	8.61326e-03	6.18653e-05
15625	8.61517e-03	1.23693e-05
390625	8.61556e-03	2.47370e-06

Table 3.3: Equality-constrained Rosenbrock problem for $x^{(0)} = (1, -1)^T$ (left) and $x^{(0)} = (-1, -1)^T$ (right) for increasing sequences of the penalty parameter. Apparently, both converge towards the global minimum.

3.5 PENALTY-LAGRANGE METHOD

The following approach is a slight modification of the penalty method and uses the well-known Lagrange multipliers¹⁷. The idea is to avoid the possibly poor stability of minimizing the penalty \mathcal{L}_τ for large τ , by considering the Penalty-Lagrange function $\mathcal{G}_\tau : \mathbb{R}^n \times \mathbb{R}^m \rightarrow \mathbb{R}$ defined by

$$\mathcal{G}_\tau(x, \lambda) := f(x) + \lambda^T h(x) + \frac{1}{2}\tau \|h(x)\|^2, \quad (3.5.1)$$

where $\lambda \in \mathbb{R}^m$ is the *Lagrange multiplier* for the equality constraints. This corresponds to the penalty method, but now applied to the *Lagrange function* defined as $\mathcal{L}(x, \lambda) := f(x) + \lambda^T h(x)$. Then, using the above penalty scheme, one is left with a sequence of optimization problems for $\ell \in \mathbb{N}$

$$\text{minimize } \mathcal{G}_{\tau_\ell}(x, \lambda_\ell) \text{ for } x \in \mathbb{R}^n,$$

where $(\lambda_\ell)_{\ell \in \mathbb{N}}$ is a *bounded* sequence of vectors in \mathbb{R}^m and $\tau_\ell > 0$ are penalty parameters as described in §3.4 above. Since the sequence of Lagrange multipliers is assumed to be bounded, Theorem 3.4.1 can in fact be applied to this method. Often, the multipliers are chosen according to

$$\lambda_{\ell+1} := \lambda_\ell + \tau_\ell h(x_\ell^*)$$

with some given initial vector $\lambda_0 \in \mathbb{R}^m$.

- Remarks 3.5.1** (i) One disadvantage of this approach is the fact that the existence of a minimum of (3.5.1) is not guaranteed, not even in the case that f has a unique global minimum!
(ii) On the other hand, an advantage of this algorithm as compared to the penalty method is that the sequence $(\tau_\ell)_{\ell \in \mathbb{N}}$ can even be chosen constant and does not need to diverge to infinity. Therefore, the arising optimization problems in the course of the method are in general better conditioned than in the penalty scheme. Moreover, this method typically shows considerably better convergence properties than the original penalty method. Under certain conditions, the convergence rate is even superlinear, see [16, Vol. 1, §7.3.3].

Example 3.5.2 We continue with the constrained Rosenbrock minimization in Example 3.4.3 and use the same setting as in Figure 3.9 and Table 3.3. The results are displayed in Figure 3.10 and Table 3.4. We see again that the method converges for both choices of the initial value, again with more iterations for $x^{(0)} = (-1, -1)$. Comparing with the results of the penalty method, we observe significantly smaller values of τ , i.e., more stability. On the other hand, the iteration tends to “overshoot” in the sense that it crosses the boundary of the red circle and then returns. ◇

Example 3.5.3 Again, one can perform analogous numerical experiments using the Himmelblau, Bazaraa-Shetty or the Dixon-Price function (in particular for higher space dimensions) introducing constraints. We leave this as coding exercises. ◇

¹⁷Joseph-Louis de Lagrange (1736 as Giuseppe Lodovico Lagrangia-1813)

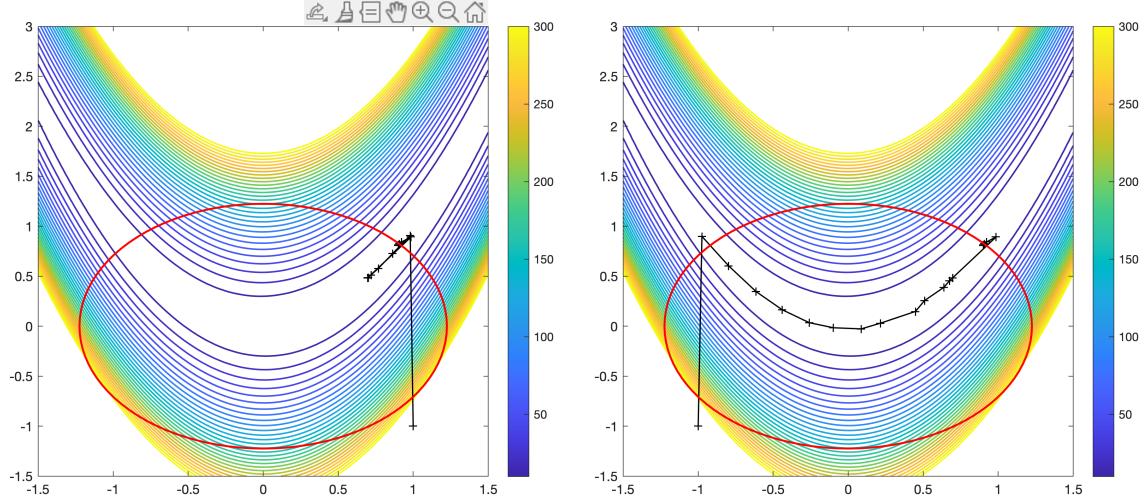


Figure 3.10: Iteration history for the Penalty-Lagrange method for solving the equality-constrained Rosenbrock problem with increasing sequence of τ and new initial values for $x^{(0)} = (1, -1)$ (left) and $x^{(0)} = (-1, -1)$ (right).

τ	λ	$f(x^{(k)})$	$\ h(x^{(k)})\ $
1	1.0000	4.00000e+02	5.00000e-01
1	1.0000	2.76798e-01	2.84157e-01
1	1.0000	5.18457e-02	2.31496e-01
1	1.0000	6.48811e-02	5.74042e-01
1	1.0000	8.06295e-02	7.20997e-01
1	1.0000	9.04500e-02	7.72423e-01
1	1.0000	9.19029e-02	7.79126e-01
5	0.2207	9.19312e-02	7.79254e-01
5	0.2207	6.28571e-01	2.67287e-01
5	0.2207	1.18886e-02	5.98479e-02
5	0.2207	9.80721e-03	2.89857e-02
5	0.2207	1.00605e-02	3.56020e-02
25	0.0426	1.00620e-02	3.56386e-02
25	0.0426	8.59355e-03	5.75263e-04
125	0.0387	8.62165e-03	1.55101e-04

τ	λ	$f(x^{(k)})$	$\ h(x^{(k)})\ $
1	1.0000	4.04000e+02	5.00000e-01
1	1.0000	4.17707e+00	2.57445e-01
1	1.0000	3.32688e+00	5.02599e-01
1	1.0000	2.70442e+00	1.00145e+00
1	1.0000	2.16041e+00	1.27891e+00
1	1.0000	1.69062e+00	1.43155e+00
1	1.0000	1.27417e+00	1.48989e+00
1	1.0000	9.54898e-01	1.49170e+00
1	1.0000	6.42651e-01	1.45288e+00
1	1.0000	5.94265e-01	1.27724e+00
1	1.0000	2.42552e-01	1.17623e+00
1	1.0000	1.64604e-01	9.42821e-01
1	1.0000	1.05609e-01	8.36871e-01
1	1.0000	9.18668e-02	7.78873e-01
1	1.0000	9.19284e-02	7.79241e-01
5	0.2207	9.19312e-02	7.79254e-01
5	0.2207	6.28572e-01	2.67287e-01
5	0.2207	1.18886e-02	5.98481e-02
5	0.2207	9.80721e-03	2.89857e-02
5	0.2207	1.00605e-02	3.56020e-02
25	0.0426	1.00620e-02	3.56386e-02
25	0.0426	8.59355e-03	5.75263e-04
125	0.0387	8.62165e-03	1.55101e-04

Table 3.4: Equality-constrained Rosenbrock problem for $x^{(0)} = (1, -1)^T$ (left) and $x^{(0)} = (-1, -1)^T$ (right) for the Penalty-Lagrange method. Apparently, both converge towards the global minimum.

3.6 LINEAR-QUADRATIC PROBLEMS

Optimization problems with a quadratic objective function and linear constraints form the core of more general methods, namely *sequential quadratic programming* (SQP), which will be described later. Roughly speaking, SQP methods reduce a more general constrained optimization problem to a sequence of linear-quadratic problems. That is the reason, why we start by investigating such simpler problems and consider the following linear-quadratic problem (in the optimization literature also known as linear-quadratic *program*)

$$\begin{cases} \text{minimize} & f(x) := \frac{1}{2}x^T Ax + a^T x, \quad x \in \mathbb{R}^n, \\ \text{s.t.} & Bx = b, \end{cases} \quad (3.6.1)$$

where $A \in \mathbb{R}^{n \times n}$ is a symmetric but not necessarily positive definite matrix, $a \in \mathbb{R}^n$, $B \in \mathbb{R}^{m \times n}$ with $m \leq n$, $b \in \mathbb{R}^m$ are given and the feasible region (i.e., the set of all arguments satisfying the constraint) reads

$$\mathcal{F} := \{x \in \mathbb{R}^n : Bx = b\}. \quad (3.6.2)$$

We shall typically assume that B has full rank m . Sometimes, the feasible region is also denoted by Ω . However, in optimization, the notation \mathcal{F} is preferred in order to stress the notion of a *feasible* set or range ($f \leftrightarrow \mathcal{F}$).

It turns out that the treatment of linear-quadratic problems is based upon several methods known from numerical linear algebra.

Example 3.6.1 The “Ulmer Zelt” (`ulmerzelt.de`) is an institution in Ulm since 1987. In this example, we aim at computing its shape.¹⁸ Assume that the tent is located in $\Omega = [0, 1]^2$ and that its boundary is fixed at the ground. We are given five poles, one with height 0.5 in the center and four with height 0.3 as indicated in Figure 3.11 on the picture on the left. On those poles, the tent is fixed. The tent itself consists of elastic, heavy material. In order to be able also to consider inequality constraints, we introduce an obstacle with height 0.2 at the front center of the tent. The corresponding solutions without and with obstacle are shown in Figure 3.11.

We model this problem as a linear-quadratic problem as follows. The shape of the tent is determined by minimizing its total energy E_{total} , which is the sum of the potential and deformation energy: $E_{\text{total}} = E_{\text{pot}} + E_{\text{def}}$. For the position $p = (p_1, p_2)^T \in \Omega$, we define $x(p)$ as the height of the tent. Then, $E_{\text{pot}}(p) = \rho x(p)$ for some constant, which we set to $\rho = 1/30000$ here. The deformation energy is approximately proportional to the product of the height and the second derivative of the height, i.e.,

$$\begin{aligned} E_{\text{def}}(x) &\approx x(p)\Delta x(p) \\ &\approx x(p)\Delta_h^2 x(p) = x(p) \frac{1}{h^2} [4x(p) - x(p + \delta_1) - x(p - \delta_1) - x(p + \delta_2) - x(p + \delta_2)], \end{aligned}$$

where δ_i denotes the i -th canonical vector. This approximation corresponds to the central difference, i.e., the 5-point stencil. Next, we introduce a discretization of Ω , here by equidistant points in both directions, i.e., $p_{i,j} = (ih, jh)^T$ and the unknown is a vector $x = (x(p_{i,j}))_{i,j}$. Then, we get $f(x) = \frac{1}{2}x^T Ax + a^T x$, where $a = \rho(1, \dots, 1)^T$, A is the block-tridiagonal matrix arising from the 5-point stencil (multiplied by 2 in order to match the form of f). The matrix B and the vector b are modeled by fixing the tent on the poles. Later, we will also include inequality constraints of the form $Cx \leq c$, where C and c are derived from the constraint $x(p) \geq 0.2$ for all points p in the ground of the obstacle. The corresponding Matlab-codes are named `UlmerZelt.m` and `UlmerZeltObstacle.m`, respectively. \diamond

¹⁸This example is taken in part from <https://de.mathworks.com/help/optim/ug/large-scale-problem-based-quadratic-programming.html>.

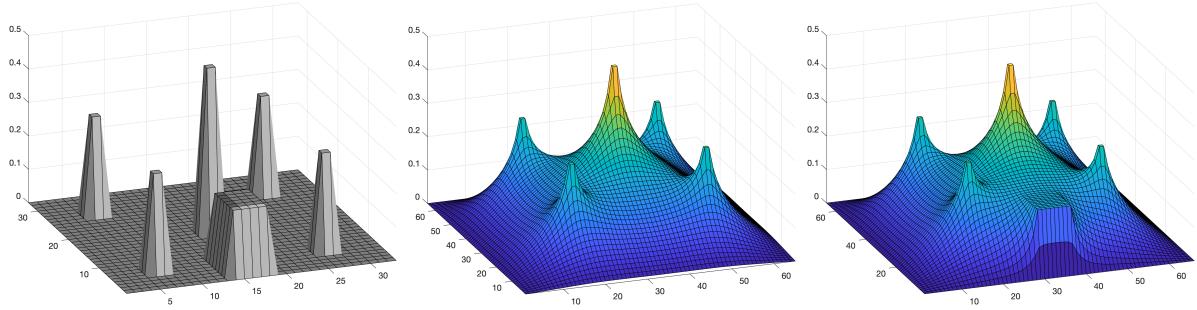


Figure 3.11: Ulmer Zelt problem: Left: Poles and possible obstacle; center: solution without obstacle; right: solution with obstacle.

3.6.1 KKT conditions

A straightforward approach to treat linear-quadratic problems is by solving the first-order necessary optimality conditions. The *Lagrange function* for (3.6.1) reads

$$\mathcal{L}(x, \lambda) := f(x) + \lambda^T(Bx - b),$$

with the *Lagrange multiplier* $\lambda \in \mathbb{R}^m$. Then, the gradient and the Hessian matrix of the Lagrange function read

$$\nabla \mathcal{L}(x, \lambda) = \begin{pmatrix} Ax + a + B^T\lambda \\ Bx - b \end{pmatrix} \quad \text{and} \quad \nabla^2 \mathcal{L}(x, \lambda) = \begin{pmatrix} A & B^T \\ B & 0 \end{pmatrix}, \quad \text{respectively.}$$

As usual, a root (x^*, λ^*) of $\nabla \mathcal{L}$ is called *critical point* similar to what is known from first-order necessary optimality or KKT conditions. As a consequence, the condition $\nabla \mathcal{L}(x^*, \lambda^*) = 0$ yields a *saddle point problem* with the Hessian matrix of the form

$$\nabla^2 \mathcal{L}(x, \lambda) \begin{pmatrix} x \\ \lambda \end{pmatrix} = \begin{pmatrix} A & B^T \\ B & 0 \end{pmatrix} \begin{pmatrix} x \\ \lambda \end{pmatrix} = \begin{pmatrix} -a \\ b \end{pmatrix}. \quad (3.6.3)$$

Note, that the matrix $\nabla^2 \mathcal{L}$ is independent of x and λ , symmetric but *indefinite*. Hence, we need some conditions in order to ensure that the first-order optimality conditions and also the linear-quadratic optimization problem admits a unique solution. In fact, if

- (1) the constraint matrix B has full rank and
- (2) the matrix A is positive definite on the *kernel* of B , i.e., on $\ker(B) := \{x \in \mathbb{R}^n : Bx = 0\}$, i.e.,

$$d^T Ad \geq \alpha \|d\|^2 \text{ for all } d \in \ker(B) \quad \text{and} \quad d^T Ad = 0 \iff d = 0 \text{ for } d \in \ker(B),$$

then the problem (3.6.1) has a unique solution x^* . In that case, for the critical point x^* , there exists a unique multiplier λ^* so that (x^*, λ^*) is a solution of the saddle point problem (3.6.3).

3.6.2 The null-space method

This is a (quite simple) method whose idea is to reduce the linear-quadratic problem (3.6.1) to an unconstrained optimization problem on a lower-dimensional subset. The scheme consists of three steps:

1. Computation of a so-called *null-space matrix* Z , where

$$\ker(B) = \text{im}(Z).$$

This means that the columns of Z form a basis for the null-space of B .

2. Reduction to an unconstrained optimization problem of the dimension $n - m$ corresponding to the column rank of Z .
 3. Computation of the corresponding Lagrange multiplier.
- We will now describe these steps in more detail.

Step 1: Computation of a null-space matrix

- 1.1) Determine a QR decomposition of $\mathbb{R}^{n \times m} \ni B^T = Q\tilde{R}$ by computing an orthogonal matrix Q and an upper-right regular triangular matrix R such that

$$Q^T B^T = \begin{pmatrix} R \\ 0 \end{pmatrix} = \tilde{R}, \quad R \in \mathbb{R}^{m \times m}.$$

Next, we split the orthogonal matrix $Q \in \mathbb{R}^{n \times n}$ according to

$$Q = (Y, Z) \in \mathbb{R}^{n \times n},$$

where the two sub-matrices $Y \in \mathbb{R}^{n \times m}$ and $Z \in \mathbb{R}^{n \times (n-m)}$ obviously both have full rank due of the orthogonality of Q .

- 1.2) Given this split, we have that each $x \in \mathbb{R}^n$ has a unique decomposition into $x_Y \in \mathbb{R}^m$ and $x_Z \in \mathbb{R}^{n-m}$ with

$$x = Yx_Y + Zx_Z = Q \begin{pmatrix} x_Y \\ x_Z \end{pmatrix}.$$

- 1.3) Now, consider an arbitrary vector $d \in \ker(B) \subset \mathbb{R}^n$. With the above splitting, we get that $d = Yd_Y + Zd_Z$ and

$$0 = Bd = B(Yd_Y + Zd_Z) = BQ \begin{pmatrix} d_Y \\ d_Z \end{pmatrix}.$$

Next, invoking 1.1) yields

$$\begin{pmatrix} R \\ 0 \end{pmatrix} = Q^T B^T = \begin{pmatrix} Y^T \\ Z^T \end{pmatrix} B^T = \begin{pmatrix} Y^T B^T \\ Z^T B^T \end{pmatrix},$$

so that $BQ = (R^T, 0)$ and therefore

$$0 = Bd = BQ \begin{pmatrix} d_Y \\ d_Z \end{pmatrix} = (R^T, 0) \begin{pmatrix} d_Y \\ d_Z \end{pmatrix} = R^T d_Y.$$

By assumption and 1.1), the matrix R has full rank, so that

$$d \in \ker(B) \iff d_Y = 0 \iff d = Zd_Z \iff d \in \text{im}(Z).$$

Consequently, the matrix Z is the desired *null-space matrix* of the constraint-matrix B .

Step 2: Reduction to an unconstrained optimization problem

- 2.1) Let x^* be the solution of (3.6.1) with its unique decomposition $x^* = Yx_Y^* + Zx_Z^*$ according to 1.2). In view of the constraint and since $Zx_Z^* \in \text{im}(Z) = \ker(B)$, we obtain by 1.3) that

$$b = Bx^* = BYx_Y^* + \underbrace{BZx_Z^*}_{=0} = BYx_Y^* = R^T x_Y^*. \quad (3.6.4)$$

Since R is an upper-right triangular matrix, one can easily determine x_Y^* by backward substitution.

- 2.2) Next, we need a feasible point $w \in \mathcal{F}$. To that end, define $w := Yx_Y^*$, then $Bw = BYx_Y^* = Bx^* = b$, so that we get $w \in \mathcal{F}$.

- 2.3) Reduction to an unconstrained problem: Combing the previous steps yields:

$$\min_{z \in \mathbb{R}^{n-m}} f(w + Zz).$$

Next, in order to get an efficient reduction of the dimension, we perform a transformation of variables. In order to do so, note, that

$$\begin{aligned} f(w + Zz) &= \frac{1}{2}(w + Zz)^T A(w + Zz) + a^T(w + Zz) \\ &= \frac{1}{2}z^T Z^T AZz + w^T AZz + \frac{1}{2}w^T Aw + a^T(w + Zz) \\ &= \frac{1}{2}z^T Z^T AZz + (Z^T a + Z^T Aw)^T z + a^T w + \frac{1}{2}w^T Aw, \end{aligned}$$

and thus (3.6.1) is equivalent to the following transformed problem

$$\text{minimize } F(z) := \frac{1}{2}z^T \tilde{A}z + \tilde{a}^T z, \quad z \in \mathbb{R}^{n-m}, \quad (3.6.5)$$

with the transformed quantities

$$\tilde{A} := Z^T AZ \quad \text{and} \quad \tilde{a} := Z^T a + Z^T Aw.$$

Obviously, the new objective function F is again quadratic and convex. As a consequence, the transformed problem (3.6.5) has a unique solution z^* with

$$0 = \nabla F(z^*) = \tilde{A}z^* + \tilde{a},$$

i.e. $x_Z^* = -\tilde{A}^{-1}\tilde{a}$. We can detail the linear system as

$$Z^T AZx_Z^* = \tilde{A}x_Z^* = -\tilde{a} = -Z^T a - Z^T Aw = -Z^T(a + AYx_Y^*). \quad (3.6.6)$$

With that, $x^* = Yx_Y^* + Zx_Z^*$ is given by (3.6.4) and (3.6.6).

For stability reasons, one does the setup of the linear system of equations (3.6.6) as follows: As above, with $Q = (Y, Z)$ set

$$\begin{aligned} h &:= -Q^T a = \begin{pmatrix} h_1 \\ h_2 \end{pmatrix}, \quad h_1 \in \mathbb{R}^m, h_2 \in \mathbb{R}^{n-m}, \\ \tilde{B} &:= Q^T A Q = \begin{pmatrix} \tilde{B}_{11} & \tilde{B}_{12} \\ \tilde{B}_{21} & \tilde{B}_{22} \end{pmatrix}, \quad \tilde{B}_{11} \in \mathbb{R}^{m \times m}. \end{aligned}$$

Then, we have

$$\begin{aligned} \tilde{B}_{11} &= Y^T AY, \quad \tilde{B}_{12} = Y^T AZ, \quad h_1 = -Y^T b \\ \tilde{B}_{21} &= Z^T AY, \quad \tilde{B}_{22} = Z^T AZ = \tilde{A}, \quad h_2 = -Z^T b. \end{aligned}$$

Now, x_Y^* is known from (3.6.4), which means that (3.6.6) is equivalent to

$$\tilde{B}_{22}x_Z^* = h_2 - \tilde{B}_{21}x_Y^* \quad (3.6.7)$$

and the matrix \tilde{B}_{22} is symmetric and positive definite. Therefore, this linear system of equations can efficiently be solved and is stable w.r.t. errors.

Step 3: Computation of the Lagrange multiplier

Considering the first row of the saddle point problem (3.6.3) yields a linear system for λ^* , namely

$$B^T \lambda^* = -a - Ax^*, \quad (3.6.8)$$

which has a unique solution, since B has full rank. For an efficient solution, we use the splitting introduced above and get

$$x^* = Q \begin{pmatrix} x_Y^* \\ x_Z^* \end{pmatrix} = Yx_Y^* + Zx_Z^*.$$

Then, we get by (3.6.8)

$$Q^T A Q \begin{pmatrix} x_Y^* \\ x_Z^* \end{pmatrix} + Q^T B^T \lambda^* = -Q^T a,$$

which is equivalent to

$$\begin{pmatrix} \tilde{B}_{11} & \tilde{B}_{12} \\ \tilde{B}_{21} & \tilde{B}_{22} \end{pmatrix} \begin{pmatrix} x_Y^* \\ x_Z^* \end{pmatrix} + \begin{pmatrix} R \\ 0 \end{pmatrix} \lambda^* = \begin{pmatrix} h_1 \\ h_2 \end{pmatrix}.$$

Therefore, the desired Lagrange multiplier λ^* is determined by the solution of the triangular linear system of equations

$$R\lambda^* = h_1 - \tilde{B}_{11}x_Y^* - \tilde{B}_{12}x_Z^*, \quad (3.6.9)$$

which can easily be solved by backward substitution.

We summarize all findings described above in Algorithm 3.6.1.

Algorithm 3.6.1: Null-space method

Input: matrices $A \in \mathbb{R}^{n \times n}$, $B \in \mathbb{R}^{m \times n}$, vectors $a \in \mathbb{R}^n$, $b \in \mathbb{R}^m$

- 1: Compute the QR decomposition of $B^T = Q \begin{pmatrix} R \\ 0 \end{pmatrix}$ and split $Q = (Y, Z)$
- 2: Compute x_Y^* by (3.6.4), i.e., solve $R^T x_Y^* = b$
- 3: Compute x_Z^* by (3.6.6), i.e., $Z^T A Z x_Z^* = -Z^T(a + A Y x_Y^*)$; set $x^* := Q \begin{pmatrix} x_Y^* \\ x_Z^* \end{pmatrix}$
- 4: Set $\tilde{B}_{11} = Y^T A Y$, $\tilde{B}_{12} = Y^T A Z$, $h_1 = -Y^T a$ and
compute λ^* by (3.6.9), i.e., solve $R\lambda^* = h_1 - \tilde{B}_{11}x_Y^* - \tilde{B}_{12}x_Z^*$

A Matlab realization reads as follows.

MATLAB-function: nullspace_method.m

```

1 function [x, lambda] = nullspace_method(A, a, B, b)
2     n=size(A, 1);
3     m=size(B, 1);
4
5     %-- (1) QR-decomposition and splitting
6     [Q, R]=qr(B');
7     R = R(1:m, 1:m);
8     Y = Q(1:n, 1:m);
9     Z = Q(1:n, m+1:n);
0
1     %-- (2) determine x_Y
2     x_Y = (R')\b;
3
4     %-- (3) determine x_Z and set x
5     tildB22 = Z'*A*Z;
6     tildB = (-1)*Z'*(a+A*Y*x_Y);
7     x_Z = tildB22\tildB;
8     x = Q*[x_Y' x_Z']';
9

```

```

20      %-- (4) determine lambda
21      tildeB11 = Y'*A*Y;
22      tildeB12 = Y'*A*Z;
23      h1       = (-1)*Y'*a;
24      tildeR   = (h1 - tildeB11*x_Y - tildeB12*x_Z);
25      lambda   = R\tildeR;
26  end

```

Example 3.6.2 We consider the equality-constrained Ulmer Zelt problem in Example 3.6.1. The corresponding Matlab code is listed below. We obtain the solution depicted in Figure 3.11. The reader can modify the dimension; what do you observe? \diamond

MATLAB-function: UlmerZelt_nullspace_method.m

```

1 close all; clear
2 dim=64;
3 [A,a,B,b] = UlmerZelt(dim);
4 [x,lambda] = nullspace_method(A,a,B,b);
5
6 %-- pad x by zeros to enforce zero boundary conditions
7 u = zeros(dim+2,dim+2);
8 for i=1:dim
9     u(i+1,2:dim+1) = x((i-1)*dim+1:i*dim);
10 end
11
12 %-- export solution
13 surf(u); axis tight; view([-20,30]);
14 exportgraphics(gca,'../../Data/UlmerZelt-NullSpace.png','Resolution'
15 ,300)

```

3.6.3 Active set method

We continue with the linear-quadratic problem (3.6.1) and add inequality constraints, i.e., we consider

$$\begin{cases} \text{minimize} & f(x) = \frac{1}{2}x^T Ax + a^T x \\ \text{s.t.} & Bx = b, Cx \leq c \end{cases} \quad (3.6.10)$$

with given $C \in \mathbb{R}^{p \times n}$ and $c \in \mathbb{R}^p$. Then, the *feasible region* reads

$$\mathcal{F} = \{x \in \mathbb{R}^n, Bx = b, Cx \leq c\}. \quad (3.6.11)$$

The *active set method* (also called the method of active (index-)sets) is a numerical algorithm that transforms an optimization problem with inequality constraints into a sequence of optimization problems of the form (3.6.1), i.e., a linear-quadratic optimization problem involving only equality constraints. The key idea is to (iteratively) identify those components of the inequality $Cx \leq c$ (both sides are vectors and the inequality is to be understood component-wise), where indeed equality holds, like in Definition 3.1.1

$$\mathcal{A}(x) = \{j = 1, \dots, p : (Cx - c)_j = 0\}. \quad (3.6.12)$$

The indices of those inequalities is the set of “active” inequalities and the corresponding rows of Cx and c are added to the equality constraints $Bx = b$. The resulting linear-quadratic problem is of the form

(3.6.1) and can be solved e.g. by the null-space method in Algorithm 3.6.1. This means, we obtain an iterative method using the null-space method in each step.

We shall denote by $C_j \in \mathbb{R}^n$ the j -th row of C , i.e.,

$$C = \begin{pmatrix} C_1 \\ \vdots \\ C_p \end{pmatrix}. \quad (3.6.13)$$

We start to collect some notations needed for the definition of a set of active indices.

Definition 3.6.3 (a) Let $S \subset \mathbb{R}^n$ and $x \in S$. Then, the set

$$\mathcal{C}(S, x) := \{\alpha(s - x) : s \in S, \alpha > 0\} \quad (3.6.14)$$

is called the cone generated by $S - x$, or, in short, the generated cone.

(b) Let \mathcal{F} be a convex feasible region and let $\tilde{x} \in \mathcal{F}$. Then, we call a vector $d \in \mathbb{R}^n$ feasible direction in \mathcal{F} at \tilde{x} , if $\tilde{x} + td \in \mathcal{F}$ is suitable for all $t > 0$ sufficiently small.

Remark 3.6.4 Let \mathcal{F} be convex and $\tilde{x} \in \mathcal{F}$. Then, each $d \in \mathcal{C}(\mathcal{F}, \tilde{x})$ is a feasible direction in \mathcal{F} at \tilde{x} and $\mathcal{C}(\mathcal{F}, \tilde{x})$ is called cone of feasible directions at \tilde{x} .

Proof. Let $d \in \mathcal{C}(\mathcal{F}, \tilde{x})$, such that $d = \alpha(x - \tilde{x})$ with $x \in \mathcal{F}$. Since \mathcal{F} is convex, the connection of x and \tilde{x} is in \mathcal{F} , i.e., $\{\tilde{x} + t(x - \tilde{x}) : t \in [0, 1]\} \subset \mathcal{F}$. Hence, $\tilde{x} + s\alpha(x - \tilde{x}) = \tilde{x} + sd \in \mathcal{F}$ for $s > 0$ sufficiently small. Thus, d is a feasible direction at \tilde{x} . On the other hand, let d be an arbitrary feasible direction at \tilde{x} . Then, by definition, there exists some $t > 0$ such that $\tilde{x} + td \in \mathcal{F}$. Therefore, $td \in \mathcal{F} - \tilde{x}$ is feasible and hence $d \in \mathcal{C}(\mathcal{F}, \tilde{x})$. \square

Now, we can characterize the set of active indices.

Lemma 3.6.5 Let $\tilde{x} \in \mathcal{F}$. Then, we have

$$\mathcal{C}(\mathcal{F}, \tilde{x}) = \mathcal{L}(\mathcal{F}, \tilde{x}) := \{d \in \mathbb{R}^n : Bd = 0, C_j d \leq 0, j \in \mathcal{A}(\tilde{x})\}.$$

Note, that the definition of $\mathcal{L}(\mathcal{F}, \tilde{x})$ involves $C_j d \leq 0$ — and not c_j on the right-hand side of the inequality constraint!

Proof. We show that the spaces mentioned in the claim are contained in each other.

1) “ $\mathcal{C}(\mathcal{F}, \tilde{x}) \subseteq \mathcal{L}(\mathcal{F}, \tilde{x})$ ”: Let $d \in \mathcal{C}(\mathcal{F}, \tilde{x})$ be arbitrary. Then, there exists a $x \in \mathcal{F}$ and a $\alpha \geq 0$ with $d = \alpha(x - \tilde{x})$. Because $x, \tilde{x} \in \mathcal{F}$ and $Bx = B\tilde{x} = b$, we get that

$$Bd = \alpha B(x - \tilde{x}) = 0.$$

Furthermore, since $x \in \mathcal{F}$ and due to the definition of $\mathcal{A}(\tilde{x})$ we have that $C_j x \leq c_j = C_j \tilde{x}$ for $j \in \mathcal{A}(\tilde{x})$. Hence, it follows that $C_j d = \alpha C_j(x - \tilde{x}) \leq 0$ for $j \in \mathcal{A}(\tilde{x})$ from which we obtain that $d \in \mathcal{L}(\mathcal{F}, \tilde{x})$.

2) “ $\mathcal{L}(\mathcal{F}, \tilde{x}) \subseteq \mathcal{C}(\mathcal{F}, \tilde{x})$ ”: Now let $d \in \mathcal{L}(\mathcal{F}, \tilde{x})$ be arbitrary. Because of $\tilde{x} \in \mathcal{F}$ and in view of the definition of $\mathcal{A}(\tilde{x})$, we get that $C_j \tilde{x} < c_j$ for all $j \notin \mathcal{A}(\tilde{x})$. Thus, there exists a $\bar{t}_j > 0$ with $C_j(\tilde{x} + td) < c_j$ for all $t \in [0, \bar{t}_j]$. Now set

$$\bar{t} := \begin{cases} \min_{j \notin \mathcal{A}(\tilde{x})} \bar{t}_j, & \text{if } \mathcal{A}(\tilde{x}) \subsetneq \{1, \dots, p\}, \\ > 0 \text{ arbitrary,} & \text{if } \mathcal{A}(\tilde{x}) = \{1, \dots, p\}. \end{cases}$$

In particular, we have $\bar{t} > 0$. Then,

$$C_j(\tilde{x} + \bar{t}d) < c_j \quad \forall j \notin \mathcal{A}(\tilde{x}). \quad (3.6.15)$$

Since $d \in \mathcal{L}(\mathcal{F}, \tilde{x})$ we get that $Bd = 0$ as well as $C_j d \leq 0$ for all $j \in \mathcal{A}(\tilde{x})$. Hence,

$$B(\tilde{x} + \bar{t}d) = B\tilde{x} + \bar{t}Bd = B\tilde{x} = b, \quad (3.6.16)$$

since $\tilde{x} \in \mathcal{F}$. For $j \in \mathcal{A}(\tilde{x})$, we conclude that

$$C_j(\tilde{x} + \bar{t}d) = C_j\tilde{x} + \bar{t}C_jd \leq C_j\tilde{x} = c_j, \quad (3.6.17)$$

because $C_j d \leq 0$ and $\bar{t} > 0$. By (3.6.15)-(3.6.17) it follows that $\tilde{x} + \bar{t}d \in \mathcal{F}$, so that $\bar{t}d \in \mathcal{F} - \tilde{x}$ and we finally get that $d \in \mathcal{C}(\mathcal{F}, \tilde{x})$. \square

Remark 3.6.6 (a) For the quadratic objective function f in (3.6.10) it holds that $\nabla f(x) = Ax + a$ and thus a vector $d \neq 0$ is a descent direction, if

$$\nabla f(x)^T d = (Ax + a)^T d < 0. \quad (3.6.18)$$

Thus, $d \in \mathcal{C}(\mathcal{F}, x)$ with (3.6.18) is called a feasible descent direction at the point x .

(b) If x^* is a local minimum of the function f , then $\nabla f(x^*)^T d \geq 0$ for all $d \in \mathcal{C}(\mathcal{F}, x)$. Hence, if $\nabla f(x^*)^T d < 0$ holds, then x^* is not a local minimum and there exists a feasible descent direction by (a).

Next, we introduce some notation and abbreviations required for the computation of such directions.

Definition 3.6.7 We define for the problem (3.6.10) and $x \in \mathbb{R}^n$:

(a) The number of active inequalities in x is denoted by

$$p(x) := \#\mathcal{A}(x).$$

(b) The matrix consisting of those row vectors of C corresponding to the active inequalities reads

$$C(x) := (C_j)_{j \in \mathcal{A}(x)}.$$

(c) The matrix of the (equality) constraints is denoted by

$$D(x) := \begin{pmatrix} B \\ C(x) \end{pmatrix}.$$

(d) Let $(x^{(k)})_{k \in \mathbb{N}_0}$ be a sequence (determined through some iteration for example). Then, we use the abbreviations

$$\mathcal{A}^{(k)} := \mathcal{A}(x^{(k)}), \quad p^{(k)} := p(x^{(k)}), \quad C^{(k)} := C(x^{(k)}) \quad \text{and} \quad D^{(k)} := D(x^{(k)}).$$

In order to describe an iteration for solving (3.6.10), let $x^{(k)}$ be given and we want to determine a feasible descent direction $d^{(k)}$. For its computation, we consider the following problem and will show later that the solution in fact yields a feasible descent direction.

$$\begin{cases} \min_{d \in \mathbb{R}^n} \frac{1}{2} d^T A d + d^T (Ax^{(k)} + a) \\ \text{s.t. } D^{(k)} d = 0. \end{cases} \quad (3.6.19)$$

Obviously (3.6.19) is a constrained optimization problem with equality constraints and feasible region

$$\mathcal{F}^{(k)} := \{d \in \mathbb{R}^n : Bd = 0, C^{(k)}d = 0\}.$$

We can solve (3.6.19) for example by the Penalty(-Lagrange), by the null-space method or by solving the KKT first-order condition as described in §3.6.1. Then, we get $d^{(k)}$ as well as the multipliers $\lambda^{(k)} \in \mathbb{R}^m$ (related to the equality constraints in terms of the matrix B) and $\mu^{(k)} \in \mathbb{R}^{p^{(k)}}$ (related to the active inequalities in $C^{(k)}$).

Remark 3.6.8 (a) If $x^{(k)}$ is an exact solution of the original constrained optimization problem (3.6.10) (with inequality and inequality constraints), then the solution $(d^{(k)}, \lambda^{(k)}, \mu^{(k)})$ of (3.6.19) satisfies

$$d^{(k)} = 0 \text{ and } \mu^{(k)} \geq 0. \quad (3.6.20)$$

On the other hand, if (3.6.20) does not hold, one can in fact compute a feasible descent direction. Thus, as long as the algorithm does not terminate with the exact solution, one can compute a feasible descent direction.

(b) In order to apply the null-space method in Algorithm 3.6.1, we need to verify the following requirements:

$$D^{(k)} \text{ needs to have full rank,} \quad (3.6.21)$$

$$A \text{ needs to be s.p.d. on } \ker(D^{(k)}). \quad (3.6.22)$$

In many cases, the matrix A is symmetric positive definite even on the larger space $\ker(B) \supseteq \ker(D^{(k)})$. In that case, of course (3.6.22) is fulfilled and one only needs to verify (3.6.21).

(c) By Lemma 3.6.5 we have that

$$\mathcal{F}^{(k)} \subset \mathcal{L}(\mathcal{F}, x^{(k)}) = \mathcal{C}(\mathcal{F}, x^{(k)}).$$

Therefore, each element $d^{(k)} \in \mathcal{F}^{(k)}$ is also a feasible direction.

(d) A necessary and sufficient optimality condition for $d^{(k)}$ as solution of (3.6.19) reads

$$Ad^{(k)} + Ax^{(k)} + a + B^T \lambda^{(k)} + (C^{(k)})^T \mu^{(k)} = 0, \quad (3.6.23)$$

which can easily be verified. \diamond

Therefore, we are left with three possibilities:

- 1) **Case 1:** $d^{(k)} = 0, \mu^{(k)} \geq 0$;
- 2) **Case 2:** $d^{(k)} = 0, \mu_j^{(k)} < 0$ for at least one $j \in \mathcal{A}^{(k)}$;
- 3) **Case 3:** $d^{(k)} \neq 0$ is a feasible direction.

We examine those three cases more detailed and present the algorithmic proceeding.

- 1) **Case 1:** $d^{(k)} = 0, \mu^{(k)} \geq 0$.

In view of Remark 3.6.8 (a), the vector $x^{(k)}$ is a solution of (3.6.10). The algorithm terminates.

- 2) **Case 2:** $d^{(k)} = 0, \mu_j^{(k)} < 0$ for at least one $j \in \mathcal{A}^{(k)}$.

In that case, the vector $d^{(k)}$ is *not* a feasible descent direction in $x^{(k)}$. We have

$$\nabla f(x^{(k)}) = Ax^{(k)} + a = -Ad^{(k)} - B^T \lambda^{(k)} - (C^{(k)})^T \mu^{(k)} \quad (3.6.24)$$

with (3.6.23). Now, let $d \in \mathcal{C}(\mathcal{F}, x^{(k)})$, then $Bd = 0$. Thus we have for this d and $d^{(k)} = 0$ that

$$\nabla f(x^{(k)})^T d = -(d^{(k)})^T Ad - (\lambda^{(k)})^T Bd - (\mu^{(k)})^T C^{(k)}d = - \sum_{j \in \mathcal{A}^{(k)}} \mu_j^{(k)} C_j d.$$

If $\nabla f(x^{(k)})^T d < 0$ would be valid, we would need to have that $C_j d < 0$ for at least one $j \in \mathcal{A}^{(k)}$ with $\mu_j^{(k)} < 0$. But then one would be able to eliminate the related constraint $C_j d \leq 0$ in (3.6.19) as this would automatically be fulfilled. The corresponding index would thus become inactive. In order to realize this, one performs the following steps:

Inactivation step:

- i) Choose an index $j \in \mathcal{A}^{(k)}$ such that $\mu_j^{(k)} = \min\{\mu_i^{(k)} : i \in \mathcal{A}^{(k)}\}$. This index does not need to be uniquely determined (non-uniqueness is not a problem here).
- ii) Cancel this (in-)equality constraint from (3.6.19), which means that one sets $\tilde{\mathcal{A}}^{(k)} := \mathcal{A}^{(k)} \setminus \{j\}$ and $\tilde{p}^{(k)} := p^{(k)} - 1$.
- iii) Solve the constrained optimization problem (3.6.19) involving only equality constraints with $\tilde{\mathcal{A}}^{(k)}$, $\tilde{D}^{(k)}$ and obtain $\tilde{d}^{(k)}$ as well as the Lagrange multipliers $\tilde{\lambda}^{(k)}$ and $\tilde{\mu}^{(k)}$. For the obtained solution we have that $B\tilde{d}^{(k)} = 0$ and $\tilde{C}^{(k)}\tilde{d}^{(k)} = 0$ (possibly after iterating).

Remark 3.6.9 After performing the inactivation step one has $\tilde{d}^{(k)} \neq 0$. One can furthermore show (see e.g. [1, S. 232]) that $\tilde{d}^{(k)} \neq 0$ is a feasible direction in the point $x^{(k)}$. Then, in the next iteration, one arrives in **Case 3** to be described next.

- 3) **Case 3:** $d^{(k)} \neq 0$ is a feasible direction. By (3.6.24), we have that $\nabla f(x^{(k)}) = -Ad^{(k)} - B^T\lambda^{(k)} - (C^{(k)})^T\mu^{(k)}$. Due to $D^{(k)}d^{(k)} = 0$ (recall, that $D^{(k)} = (B, C^{(k)})^T$), we obtain:

$$\nabla f(x^{(k)})^T d^{(k)} = -(d^{(k)})^T Ad^{(k)} - (\lambda^{(k)}, \mu^{(k)})^T D^{(k)}d^{(k)} = -(d^{(k)})^T Ad^{(k)} < 0, \quad (3.6.25)$$

since A is symmetric positive definite by assumption. Thus, $d^{(k)}$ is a descent direction.

Step size determination. In order to determine a corresponding step size $\sigma^{(k)}$, we have that

$$\begin{aligned} f(x^{(k)} + td^{(k)}) &= \frac{1}{2}(x^{(k)} + td^{(k)})^T A(x^{(k)} + td^{(k)}) + (x^{(k)} + td^{(k)})^T a \\ &= f(x^{(k)}) + t(d^{(k)})^T (Ax^{(k)} + a) + \frac{1}{2}t^2(d^{(k)})^T Ad^{(k)} \\ &= f(x^{(k)}) + t \underbrace{\nabla f(x^{(k)})^T d^{(k)}}_{\stackrel{(3.6.25)}{=} -(d^{(k)})^T Ad^{(k)}} + \frac{1}{2}t^2(d^{(k)})^T Ad^{(k)} \\ &= f(x^{(k)}) + \left(\frac{1}{2}t^2 - t\right)(d^{(k)})^T Ad^{(k)}. \end{aligned}$$

In order to determine an optimal step size t^* , we aim at minimizing the right-hand side. To this end, note, that the only t -dependent term on the right-hand side $\frac{1}{2}t^2 - t$ becomes minimal for $t^* = 1$ and attains the value $-\frac{1}{2}$. So, $t^* = 1$ would be optimal.

However, we need to restrict the values for t to prevent a violation of the constraints. Therefore, one identifies a maximal step size $\tau^{(k)} > 0$ such that $x^{(k)} + \tau^{(k)}d^{(k)} \in \mathcal{F}$. Because of $Bx^{(k)} = b$ and $Bd^{(k)} = 0$, we get that $B(x^{(k)} + \tau d^{(k)}) = Bx^{(k)} + \tau Bd^{(k)} = b$ for any $\tau \in \mathbb{R}$. Hence, the condition is always fulfilled. We are left with the requirement

$$C_j(x^{(k)} + \tau d^{(k)}) = \underbrace{C_j x^{(k)}}_{\leq c_j} + \tau C_j d^{(k)} \stackrel{!}{\leq} c_j. \quad (3.6.26)$$

For this, we have

- If $C_j d^{(k)} \leq 0$, then (3.6.26) holds, because of $Cx^{(k)} \leq c$ (component-wise). In fact, for $j \in \mathcal{A}^{(k)}$ we even have “=” in this case.
- On the other hand, if $C_j d^{(k)} > 0$ holds for some index $j \in \{1, \dots, p\}$ with $j \notin \mathcal{A}^{(k)}$, then we get that the restriction

$$\tau^{(k)} \leq \frac{c_j - C_j x^{(k)}}{C_j d^{(k)}}$$

should hold in order to fulfill (3.6.26). With the index set

$$\mathcal{I}_k := \{j \in \{1, \dots, p\} : C_j d^{(k)} > 0\}$$

of the inactive indices, we set

$$\tau^{(k)} := \begin{cases} \min \left\{ \frac{c_j - C_j x^{(k)}}{C_j d^{(k)}} : j \in \mathcal{I}_k \right\}, & \text{if } \mathcal{I}_k \neq \emptyset, \\ \infty, & \text{else.} \end{cases} \quad (3.6.27)$$

Finally, we define a feasible step size according to

$$\sigma^{(k)} := \min\{1, \tau^{(k)}\}. \quad (3.6.28)$$

Now, we have all ingredients at hand and collect the full scheme in Algorithm 3.6.2 below.

Algorithm 3.6.2: Active set method

Input: Feasible initial value $x^{(0)} \in \mathcal{F}$, problem (3.6.10).

```

1: for  $k = 0, 1, 2, \dots$  do
2:   Determine  $\mathcal{A}(x^{(k)}) = \{j = 1, \dots, p : (Cx^{(k)} - r)_j = 0\}$ ,  $p^{(k)} := \#\mathcal{A}(x^{(k)})$ .
3:   Set  $C^{(k)} = (C_j)_{j \in \mathcal{A}(x^{(k)})}$ ,  $D^{(k)} := (B^T, (C^{(k)})^T)^T$ .
4:   Solve (3.6.19) and get  $(d^{(k)}, \lambda^{(k)}, \mu^{(k)})$  with  $\lambda^{(k)} \in \mathbb{R}^m$ ,  $\mu^{(k)} \in \mathbb{R}^{p^{(k)}}$ .
5:   if  $d^{(k)} = 0$  then
6:     if  $\mu^{(k)} \geq 0$  then
7:       Stop, output:  $(x^{(k)}, \lambda^{(k)}, \mu^{(k)})$  ▷ Case 1
8:     else ▷ Case 2:  $\mu^{(k)} < 0$  for at least one  $j \in \mathcal{A}^{(k)}$ 
9:       Determine  $j$  with  $\mu_j^{(k)} = \min\{\mu_i^{(k)} : i \in \mathcal{A}^{(k)}\}$ 
10:      Set  $\mathcal{A}^{(k)} := \mathcal{A}^{(k)} \setminus \{j\}$ ,  $p^{(k)} := p^{(k)} - 1$ , determine  $C^{(k)}$  and  $D^{(k)}$ 
11:    end if
12:   else ▷ Case 3:  $d^{(k)} \neq 0$  is a feasible direction
13:     Set  $\mathcal{I}_k := \{j \in \{1, \dots, p\} : C_j d^{(k)} > 0\}$ 
14:     Compute
        
$$\tau^{(k)} := \begin{cases} \min \left\{ \frac{c_j - C_j x^{(k)}}{C_j d^{(k)}} : j \in \mathcal{I}_k \right\}, & \text{if } \mathcal{I}_k \neq \emptyset, \\ \infty, & \text{else.} \end{cases}$$

15:     Set  $\sigma^{(k)} = \min\{1, \tau^{(k)}\}$ 
16:     Set  $x^{(k+1)} := x^{(k)} + \sigma^{(k)} d^{(k)}$ 
17:   end if
18: end for

```

A Matlab realization reads as follows. In this code, the feasible search direction is determined by solving the first order KKT conditions. Of course, other approaches can also be used instead.

MATLAB-function: activeset.m

```

1 function [x_opt, lambda, mu, fval, exitflag, its] = activeset(A, a, B, b, C, c, x0
2 ,tol,maxIter)
3 % acitve set method for quadratic problems
4 % Minimize: 0.5*x'*A*x + b'*x
5 % Subject to: B * x = g, C * x <= r
6
7 if nargin<9
8   maxIter = 1000;
9 end
10
11 %-- No Inequality Constraints
12 if size(C,1) == 0
13   [x,lambda] = nullspace_method(A,a,B,b); mu = zeros(0,1);
14   return

```

```

14     end
15
16     %-- Initialization
17     x = x0;
18     m           = size(B, 1);
19     lambda      = zeros(m, 1);
20     mu          = zeros(size(C, 1), 1);
21     activeSet   = [];
22     inactiveSet = 1:size(C, 1);
23
24     for iter = 1:maxIter
25         %-- solve KKT for feasible descent direction
26         %-- could also be done e.g. by nullspace method
27         ghelp = A*x + a;
28         Bhelp = [B; C(activeSet, :)];
29         bhelp = [b; c(activeSet)];
30
31         if isempty(Bhelp)
32             d = -A \ ghelp;
33         else
34             KKT = [A, Bhelp'; Bhelp, zeros(size(Bhelp, 1))];
35             rhs = [-ghelp; zeros(size(Bhelp, 1), 1)];
36             sol = KKT \ rhs;
37             d = sol(1:length(x));
38             lambda = sol(length(x) + 1:length(x) + m);
39             if isempty(activeSet)
40                 mu = [];
41             else
42                 mu(activeSet) = sol(length(x) + 1 + m:end);
43             end
44         end
45         %-----
46         if norm(d) < tol
47             if all(mu >= 0) %--case 1
48                 exitflag = 1;
49                 break;
50             else %-- case 2: inactivation
51                 [~, idx] = min(mu(activeSet));
52                 activeSet(idx) = [];
53                 continue;
54             end
55         end
56
57         %-- case 3
58         sigma = 1; %-- step size determination
59         for i = inactiveSet
60             a_i = C(i, :);
61             b_i = c(i);
62             if a_i * d > 0
63                 sigma = min(sigma, (b_i - a_i * x) / (a_i * d));
64             end
65         end
66
67         x = x + sigma * d;
68         if sigma < 1
69             idx = find(abs(C * x - c) < tol);
70             activeSet = unique([activeSet; idx]);
71             inactiveSet = setdiff(1:size(C, 1), activeSet);

```

```

2      end
3  end
4
5  x_opt = x;
6  fval = 0.5 * x' * A * x + a' * x;
7  its = iter;
8  if iter == maxIter
9    exitflag = 0;
10   end
11 end

```

The convergence of Algorithm 3.6.2 basically follows from by the above derivation. In fact, after a finite number of steps, no inequality constraints remain, i.e., $\mathcal{A}^{(k_{\text{final}})} = \emptyset$. We summarize this observation:

Remark 3.6.10 Let A be symmetric and positive definite on $\ker(B)$ and assume that $D(x)$ has full rank for all $x \in \mathcal{F}$. Then, Algorithm 3.6.2 terminates after a finite number of steps with the unique solution of (3.6.10).

Example 3.6.11 Let us consider the in inequality-constrained Ulmer Zelt problem (including the obstacle) in Example 3.6.1. The corresponding Matlab code is listed below. We obtain the solution depicted on the right-hand side in Figure 3.12. For the active set method, we need a feasible initial iteration $x^{(0)} \in \mathcal{F}$. For our experiment, we set the initial value as equality constraints both on the poles and on the obstacle. The corresponding function is shown on the left-hand side in Figure 3.12. Try what happens when choosing a different $x^{(0)}$, also a non-feasible one. In our case, the active set method terminated after one iteration. \diamond

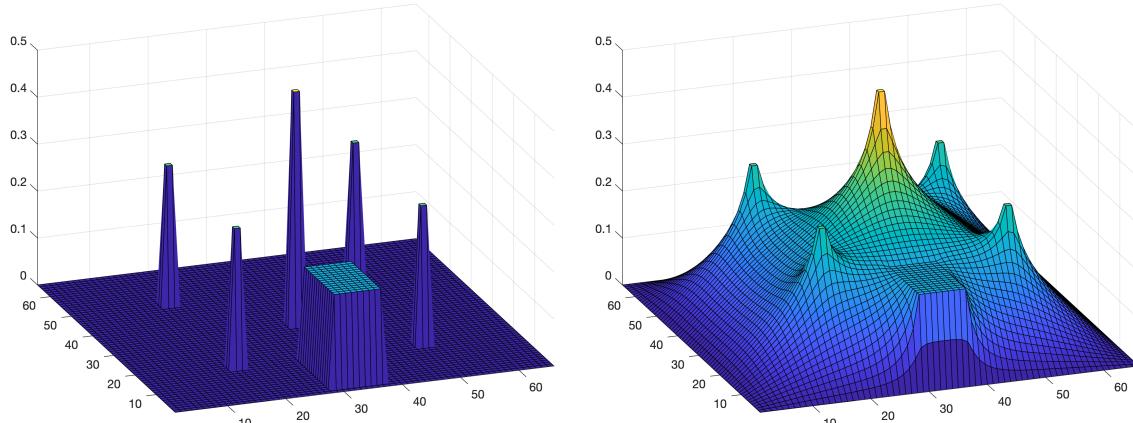


Figure 3.12: Ulmer Zelt problem with obstacle: feasible initial iteration (left) and solution by the active set method (right).

MATLAB-function: UlmerZeltObstacle_activeset.m

```

1 close all; clear
2 %-- computes optimization problem for the tent
3 %-- using quadratic programming
4
5 n=64;
6 tol=1e-6;
7 [A,a,B,b,C,c,x0]=UlmerZeltObstacle(n);

```

```

8 [x, lambda, mu, fval, exitflag, its] = activeset(A,a,B,b,C,c,x0,tol);
9
10 fprintf('Active set terminated after %i iterations.\n',its);
11 fprintf('Function value: %12.8f\n',fval);
12 fprintf('Optimality residual: %12.8f\n',norm(A*x-a));
13 fprintf('Inequality residual: %12.8f\n',norm(C*x-c));
14 fprintf('Equality residual: %12.8f\n',norm(B*x-b));
15
16 u = zeros(n+2,n+2);
17 u0 = zeros(n+2,n+2);
18 for i=1:n
19     u(i+1,2:n+1) = x((i-1)*n+1:i*n);
20     u0(i+1,2:n+1) = x0((i-1)*n+1:i*n);
21 end
22
23 figure(1)
24 surf(u);axis tight;view([-20,30]);
25 exportgraphics(gca,'../../Data/UlmerZeltObstacle_ActiveSet.png','
    Resolution',300)
26
27 figure(2)
28 surf(u0);axis tight;view([-20,30]);
29 exportgraphics(gca,'../../Data/UlmerZeltObstacle_Initial.png','
    Resolution',300)

```

3.7 NONLINEAR OBJECTIVE FUNCTIONS

So far, we have only considered quadratic objective functions. Now, we will omit this strong requirement and consider general nonlinear objective functions f . The constraints are still assumed to be linear – we may consider them as being linearized before.

Example 3.7.1 The following example of determining a chemical equilibrium was taken from [3, p. 46–49] and has served as benchmark problem for a nonlinear optimization problem with linear constraints. We consider a mixture of m chemical elements, which can combine chemically to produce n compounds. The variables $x_j \geq 0$, $j = 1, \dots, n$, denote the number of moles of compound j in the mixture. Let $b_{i,j}$ denote the the number of atoms of element $i = 1, \dots, m$ in a molecule of compound j and g_i the number of atomic weights of element i in the mixture. The mass balance relationships that must hold for the m elements then takes the form of an equality constraint

$$Bx = b,$$

where $B = (b_{i,j})_{i=1,\dots,m;j=1,\dots,n} \in \mathbb{R}^{m \times n}$, $x = (x_j)_{j=1,\dots,n} \in \mathbb{R}^n$ and $b = (b_i)_{i=1,\dots,m} \in \mathbb{R}^m$. Determination of the composition of the mixture at equilibrium requires to minimize the total free energy of the mixture, which is given by

$$f(x) := \sum_{j=1}^n x_j \left[\rho_j + \ln \left(\frac{x_j}{\bar{x}} \right) \right], \text{ where } \bar{x} := \sum_{j=1}^n x_j,$$

and $\rho_j := \left(\frac{F^0}{RT} \right)_j + \ln P$ is a constant involving the modal standard (Gibbs) free energy function $\frac{F^0}{RT}$ and the total pressure P in the atmosphere. The corresponding values can be taken from standard tables, see Table 3.5. Hence, we obtain an optimization problem of the form

$$\text{minimize } f(x), x \in \mathbb{R}^n \text{ s.t. } Bx = b, x \geq 0.$$

j	compound	$\left(\frac{F^0}{RT}\right)_j$	ρ_j	$b_{i,j}$			x_j
				H $i = 1$	N $i = 2$	O $i = 3$	
1	H	-10.021	-6.089	1			0.0407
2	H_2	-21.096	-17.164	2			0.1477
3	H_2O	-37.986	-34.054	2		1	0.7831
4	N	-9.846	-5.914		1		0.0014
5	N_2	-28.653	-24.721		2		0.4853
6	NH	-18.918	-14.986	1	1		0.0007
7	NO	-28.032	-24.100		1	1	0.0274
8	O	-14.640	-10.708			1	0.0180
9	O_2	-30.594	-26.662			2	0.0373
10	OH	-26.111	-22.179	1		1	0.0969
				b_i	2	1	1

Table 3.5: Values for a mixture consisting of the elements H, N and O ($m = 3$) and $n = 10$ compounds and optimal solution from [3].

Straightforward calculations show that

$$\frac{\partial}{\partial x_j} f(x) = \rho_j + \ln\left(\frac{x_j}{\bar{x}}\right), \quad \frac{\partial^2}{\partial x_j \partial x_k} f(x) = \frac{\delta_{j,k}}{x_j} - \frac{1}{\bar{x}},$$

so that the Hessian matrix is symmetric and positive definite. \diamond

3.7.1 Equality constraints

We start by a short remark concerning equality constrained nonlinear optimization problems of the form

$$\begin{cases} \text{minimize } f(x) \\ \text{s.t. } Bx = b. \end{cases} \quad (3.7.1)$$

A short consideration shows that we indeed already have all required numerical methods for solving (3.7.1) at hand. In fact, using the null-space method applied to the matrix B allows us to reduce (3.7.1) to an unconstrained optimization problem with a nonlinear objective function. The constraint matrix B only affects the constraint and is fully independent of the objective function f . The reduced problem can then be solved e.g. by the Quasi-Newton method or one of the other schemes for unconstrained optimization problems.

3.7.2 Inequality constraints – the Newton-SQP method

Now to the more general situation of an optimization problem with linear equality- and inequality constraints, i.e.,

$$\begin{cases} \text{minimize } f(x), x \in \mathbb{R}^n \\ \text{s.t. } Bx = b, Cx \leq c. \end{cases} \quad (3.7.2)$$

As already explained earlier, we will reduce (3.7.2) to a sequence of linear-quadratic problems. The key idea is thus quite similar to Newton's or the Gauß-Newton method, which reduce a nonlinear problem or nonlinear least-squares problem to a sequence of corresponding linear problems. Finally, all those algorithms are based upon the Taylor expansion.

The resulting method for nonlinear constrained optimization problems is called *SQP method* (SQP stands for *sequential quadratic programming*) and is probably among the most common methods.

As above, we use Taylor's expansion of $\nabla f(x)$ around the current iteration $x^{(k)}$ (with the Hessian matrix $\nabla^2 f(x^{(k)})$)

$$\nabla f(x) = \nabla f(x^{(k)}) + \nabla^2 f(x^{(k)})(x - x^{(k)}) + \mathcal{O}(\|x - x^{(k)}\|^2).$$

By doing so, we have implicitly assumed that $f \in C^2(\mathcal{U}(x^{(k)}))$ at least in a neighborhood $\mathcal{U}(x^{(k)})$ of $x^{(k)}$.

Let $(x^{(k)})_{k \in \mathbb{N}_0}$ be a sequence generated by the iteration, then we substitute the original optimization problem (3.7.2) by the linear-quadratic problem of the form ($x^{(k)}$ given) for finding an increment $d^{(k)}$

$$\begin{cases} \text{minimize} & \frac{1}{2} d^T \nabla^2 f(x^{(k)}) d + (\nabla f(x^{(k)}))^T d, \\ \text{s.t.} & Bd = 0, Cx^{(k)} + Cd \leq c. \end{cases} \quad (3.7.3)$$

Obviously, this a linear-quadratic optimization problem with equality and inequality constraint exactly of the form (3.6.10). Therefore, we can solve (3.7.3) e.g. with the active set method. Hence, we have already all components for the SQP method at hand. The scheme is described in Algorithm 3.7.1.

Algorithm 3.7.1: SQP algorithm

Require: $f, \nabla f, \nabla^2 f$

- 1: Compute a feasible initial value $x^{(0)} \in \mathcal{F}, k := 0$
 - 2: Solve the linear-quadratic problem (3.7.3) and get a feasible search direction $d^{(k)}$ as well as the multipliers $\lambda^{(k+1)}$ and $\mu^{(k+1)}$
 - 3: $x^{(k+1)} := x^{(k)} + d^{(k)}$
 - 4: Set $k := k + 1$, go to line 2
-

Algorithm 3.7.1 is also called *Newton-SQP method*, because the quadratic sub-problems are obtained by using Taylor's expansion. To analyze the SQP method we will use the following notation:

$$\mathcal{F}^{(k)} := \{d \in \mathbb{R}^n : Bd = 0, Cx^{(k)} + Cd \leq c\}. \quad (3.7.4)$$

Recalling that $C_j \in \mathbb{R}^n$ the j -th row of C , see (3.6.13), we will often use the notation $C_j x$ for the product of the n -dimensional row vector C_j and a column vector $x \in \mathbb{R}^n$.

Theorem 3.7.2 (Convergence of the Newton-SQP method) *Let x^* be the unique optimality solution of (3.7.2) and assume that the following condition hold:*

- (i) $f \in C^2(B_{\delta_1}(x^*))$ for a $\delta_1 > 0$ with $B_\delta(x^*) := \{x \in \mathbb{R}^n : \|x - x^*\| \leq \delta\}$.
- (ii) The Hessian $\nabla^2 f$ is Lipschitz continuous on $B_{\delta_1}(x^*)$, i.e., there exists an $L > 0$ with

$$\|\nabla^2 f(x) - \nabla^2 f(y)\| \leq L\|x - y\| \text{ for all } x, y \in B_{\delta_1}(x^*).$$

- (iii) The matrix $D(x) := (B^T, C(x)^T)^T$ with $C(x) := (C_j)_{j \in \mathcal{A}(x)}$ has full rank at least for $x \in \{x^*, x^{(k)} : k \in \mathbb{N}_0\}$.
- (iv) The Hessian $\nabla^2 f(x^*)$ is positive definite on $\ker(B) \cap \ker(C(x^*))$, i.e., there exists a constant $\alpha > 0$ such that

$$d^T \nabla^2 f(x^*) d \geq \alpha \|d\|^2 \text{ for all } d \in \mathbb{R}^n \text{ such that } Bd = 0 \text{ and } C(x^*)d = 0$$

(i.e., all vectors satisfying homogeneous equality constraints).

- (v) Strict complementarity: Denoting the Lagrange multipliers of $x^* \in \mathcal{F}$ by $\lambda^* \in \mathbb{R}^m$ and $\mu^* \in \mathbb{R}^p$, we have $\mu_j^* > 0$ for $j \in \mathcal{A}(x^*)$.

Then, the Newton-SQP method is locally second order convergent.

Proof. Let x^* be an optimality solution of (3.7.2) and let $\lambda^* \in \mathbb{R}^m$ and $\mu^* \in \mathbb{R}^p$ be the corresponding Lagrange multipliers. Then, we have by the KKT conditions (Theorem 3.1.17)

$$\nabla f(x^*) + B^T \lambda^* + C^T \mu^* = 0, \quad (3.7.5)$$

$$\mu_j^* \geq 0, \quad \mu_j^* C_j x^* - c_j = 0, \quad j = 1, \dots, p. \quad (3.7.6)$$

Combined with the constraints for the exact solution, i.e., $Bx^* = b$ and $Cx^* \leq c$, we obtain a system of constraints. In the next step, we reduce this system to the active inequalities:

$$\begin{cases} \nabla f(x^*) + B^T \lambda^* + C(x^*)^T \nu^* = 0, \\ Bx^* - b = 0, \\ C_j x^* - c_j = 0, \quad j \in \mathcal{A}(x^*), \end{cases} \quad (3.7.7)$$

with the Lagrange multipliers $\nu^* \in \mathbb{R}^{p(x^*)}$ and $\nu^* = (\mu_j^*)_{j \in \mathcal{A}(x^*)}$. Doing so, we define a nonlinear function $F : \mathbb{R}^{n+m+p(x^*)} \rightarrow \mathbb{R}^{n+m+p(x^*)}$ by

$$F(z) := F(x, \lambda, \nu) := \begin{pmatrix} \nabla f(x) + B^T \lambda + C(x^*)^T \nu \\ Bx - b \\ (C_j x - c_j)_{j \in \mathcal{A}(x^*)} \end{pmatrix}.$$

Accordingly, we are looking for a root z^* of F . For that, we apply Newton's method to the nonlinear system of equations $F(z) = 0$, i.e., for some $z^{(0)}$, we compute for $k = 1, 2, \dots$

$$F'(z^{(k)}) \delta z^{(k)} = -F(z^{(k)}), \quad \delta z^{(k)} = (d^{(k)}, \delta \lambda^{(k)}, \delta \nu^{(k)})^T, \quad z^{(k+1)} = z^{(k)} + \delta z^{(k)},$$

i.e., $d^{(k)} = \delta x^{(k)}$. Due to assumptions (i)-(iv), Newton's method is locally second order convergent with limit z^* satisfying $F(z^*) = 0$. Note, that

$$F'(z) = \begin{pmatrix} \nabla^2 f(x) & B^T & C(x^*)^T \\ B & 0 & 0 \\ C(x^*) & 0 & 0 \end{pmatrix},$$

so that

$$F'(z) \begin{pmatrix} d \\ \delta \lambda \\ \delta \nu \end{pmatrix} = \begin{pmatrix} \nabla^2 f(x)d + B^T \delta \lambda + C(x^*)^T \delta \nu \\ Bd \\ C(x^*)d \end{pmatrix} \stackrel{!}{=} -F(z) = \begin{pmatrix} -\nabla f(x) - B^T \lambda - C(x^*)^T \nu \\ -Bx + b \\ (-C_j x + c_j)_{j \in \mathcal{A}(x^*)} \end{pmatrix}.$$

Let $z^{(k)} = (x^{(k)}, \lambda^{(k)}, \nu^{(k)})$ denote the k -th iteration of this Newton's method for F . Hence, $x^{(k)} \in \mathcal{F}$ is feasible (in particular $Bx^{(k)} = b$) and we get that the right-hand side of the Newton update equation reads

$$-F(z^{(k)}) = -F(x^{(k)}, \lambda^{(k)}, \nu^{(k)}) = \begin{pmatrix} -\nabla f(x^{(k)}) - B^T \lambda^{(k)} - C(x^*)^T \nu^{(k)} \\ 0 \\ (-C_j x^{(k)} + c_j)_{j \in \mathcal{A}(x^*)} \end{pmatrix}.$$

Then, the next iteration of Newton's method $z^{(k+1)} = z^{(k)} + \delta z^{(k)}$ satisfies the system of equations

$$\nabla^2 f(x^{(k)})d^{(k)} + \nabla f(x^{(k)}) + B^T \lambda^{(k+1)} + C(x^*)^T \nu^{(k+1)} = 0,$$

$$Bd^{(k)} = 0,$$

$$C_j(x^{(k)} + d^{(k)}) = c_j, \quad j \in \mathcal{A}(x^*).$$

These are the necessary optimality conditions for the solution $(d^{(k)}, \lambda^{(k+1)}, \nu^{(k+1)})$ of the linear-quadratic problem with the equality constraints:

$$\begin{cases} \text{minimize} & \frac{1}{2}d^T \nabla^2 f(x^{(k)})d + \nabla f(x^{(k)})^T d \\ \text{s.t.} & Bd = 0 \text{ and } C_j(x^{(k)} + d) = c_j, j \in \mathcal{A}(x^*). \end{cases} \quad (3.7.8)$$

To complete the proof, it remains to show that the solution of the latter problem (3.7.8) is equivalent to the solution of the SQP problem (3.7.3). For the solution z^* of the nonlinear system of equations $F(z) = 0$ we have that $z^{(k)} \rightarrow z^*$ as $k \rightarrow \infty$ due to the convergence of Newton's method. Consequently, it follows that $\|d^{(k)}\| \rightarrow 0$ and therefore also $x^{(k)} \rightarrow x^*$ as $k \rightarrow \infty$. Hence, $C_j(x^{(k)} + d^{(k)}) < c_j$ for all $j \notin \mathcal{A}(x^*)$ for sufficiently large k and $C_j(x^{(k)} + d^{(k)}) = c_j$ for $j \in \mathcal{A}(x^*)$ by (3.7.8), which means that $d^{(k)}$ is feasible for the SQP problem (3.7.3).

Finally, we use the strict complementarity in (v) and conclude that

$$\mu_j^{(k+1)} = \nu_j^{(k+1)} \rightarrow \mu_j^* > 0, \quad \text{for } j \in \mathcal{A}(x^*) \text{ as } k \rightarrow \infty.$$

which implies that $\mu^{(k+1)} \geq 0$ for sufficiently large k . Thus, $(x^{(k+1)}, \lambda^{(k+1)}, \mu^{(k+1)})$ is a solution of (3.7.3) (it satisfies the necessary and sufficient optimality conditions) and because of (iv) it is a local unique solution, which concludes the proof. \square

Remark 3.7.3 It is not astonishing that the Newton-SQP method is only locally convergent as the proof shows that the scheme relies on Newton's method for the first-order optimality condition. Hence, the locality of Newton-SQO is inherited from the corresponding property of Newton's method. However, for nonlinear optimization problems, this is a much more severe restriction as we need a feasible initial value which is close enough to the desired minimum. This is the reason why there are globalized SQP-methods described for example in [14, 25]. Those schemes are based upon Quasi-Newton methods and/or penalty terms.

A Matlab realization of the Newton-SQP methods reads as follows.

MATLAB-function: sqp.m

```

1 function [x,lambda,mu,its] = sqp(f,Jf,Hf,B,b,C,c,x0,tol)
2     n = size(B,2);
3     m = size(B,1);
4     p = size(C,1);
5     x = x0;
6     its=0;
7
8     reseq=norm(B*x0-b);
9     resie=max(C*x0-c>0);
10
11    if (reseq>tol) || (resie==1)
12        fprintf(' *** Warning: x0 in sqp is infeasible: ');
13        fprintf(' ||Bx-b||=%12.8f; Cx-r>0; %i\n',reseq,resie);
14    end
15
16    fileID = fopen('..../Data/SQP_output.txt','w');
17
18    noready = 1;
19    fprintf(fileID,'$\\texttt{its}$ & $\\|d\\|\\$ & $f(x)$ & $\\|Jf(x)\\|\\$');
20    fprintf(fileID,' & $\\|Bx-b\\|\\$ & $\\max(Cx-c) !<0\\$ \\\\hline\\n');
;
```

```

1   fprintf(fileID,'%i & %12.8f & %12.8f & %12.8f & %12.8f & %12.8f\\\\\\
2     \\\hline\n',...
3       its,0,f(x),norm(Jf(x)),norm(B*x-b),max(C*x-c));
4
5   while noready
6     Ak=Hf(x);
7     bk=Jf(x);
8     gk=zeros(m,1);
9     rk=c-C*x;
10    d0=zeros(n,1);      %-- initial value for direction
11    [d,lambda,mu] = activeset(Ak,bk,B,gk,C,rk,d0,1e-6);
12    if norm(d)<tol
13      noready=0; break
14    end
15    x = x+d;
16    its=its+1;
17    fprintf(fileID,'%i & %12.8f & %12.8f & %12.8f & %12.8f & %12.8f
18      \\\hline\n',...
19      its,norm(d),f(x),norm(Jf(x)),norm(B*x-b),max(C*x-c));
20  end
21  fclose(fileID);
22  return
23 end

```

Example 3.7.4 We continue Example 3.7.1 of the chemical equilibrium. We determine “by hands” a feasible initial value $x^{(0)}$ and consider the iteration history displayed in Table 3.6. We obtain convergence after 11 iterations towards the same optimal point as reported in the literature. One might experiment with other initial values $x^{(0)}$ in order to get a better feeling for the *local* convergence of the Newton-SQP-method. ◇

its	$\ d\ $	$f(x)$	$\ Jf(x)\ $	$\ Bx - b\ $	$\max(Cx - c)$
0	0.00000000	-37.43893863	75.77741209	0.00000000	-0.04999900
1	1.96495230	-46.89034632	77.65047684	0.00000000	0.00000000
2	0.64576458	-47.65663785	84.48102392	0.00000000	0.00000000
3	0.06759063	-47.70062049	81.42668136	0.00000000	-0.00000648
4	0.04108302	-47.71417086	79.09638349	0.00000000	-0.00004000
5	0.01100588	-47.72344055	77.33360166	0.00000000	-0.00015419
6	0.00776624	-47.73860583	76.10365975	0.00000000	-0.00038127
7	0.01341495	-47.75335440	75.39418803	0.00000000	-0.00060167
8	0.01434206	-47.76015899	75.09711451	0.00000000	-0.00068252
9	0.00739617	-47.76107813	75.01907169	0.00000000	-0.00069166
10	0.00100359	-47.76109086	75.01066015	0.00000000	-0.00069217
11	0.00001363	-47.76109086	75.01054813	0.00000000	-0.00069217

Table 3.6: Results of the Newton-SQP method for the chemical equilibrium example using the feasible initial value $x^{(0)} = (\frac{8}{5}, \frac{1}{20}, \frac{1}{20}, \frac{7}{10}, \frac{1}{20}, \frac{1}{10}, \frac{1}{10}, \frac{13}{20}, \frac{1}{20}, \frac{1}{10})^T$.

4 INTERIOR POINT METHODS

Interior point methods are a *class* of efficient methods that are nowadays widely used for solving optimization problems with inequality constraints. Together with active-set SQP methods they are currently considered the most powerful algorithms for large-scale nonlinear optimization problems. Interior point methods are heavily used in particular in academia as well as in industrial research and applications. However, their appropriate application requires some background knowledge. There is a public domain open source software IPOpt: <https://www.coin-or.org/download/source/Ipopt> and a documentation in [26]. Moreover, interior point methods are meanwhile also included in Matlab, e.g. within `fmincon`.

The main idea can be described as follows: We have seen that derivative-based methods allow for a significantly faster rate of convergence as compared with direct search methods. However, facing inequality constraints, the optimal solution is often located on the boundary of the feasible set. Since derivatives can only be used in an open domain, interior point methods aim at producing a sequence of approximations being located in the interior of the feasible set and allowing to use derivatives for descent directions.

4.1 LINEAR-QUADRATIC PROBLEMS

We reconsider the linear-quadratic problem with inequality constraints of the form (3.3.2) and recall the KKT conditions (3.1.3) as well as the optimality system (3.3.3), which provides us with necessary and sufficient conditions to determine the optimal solution (x^*, λ^*, s^*) . Note, that (3.3.3c) is nonlinear, so that the optimality system (3.3.3) requires to solve a nonlinear problem. We want to do this by a Newton method.

However, in order to do so, we need to modify the optimality system such that the Newton update involving derivatives is well-posed. Given an iteration (x, λ, s) with $\lambda > 0$ and $s > 0$, we define a *complementarity measure* μ^1 by

$$\mu := \frac{\lambda^T s}{p}. \quad (4.1.1)$$

With that, instead of (3.3.3), we consider *perturbed* primal-dual optimality system

$$F_\mu(x, \lambda, s; \sigma) := \begin{pmatrix} Ax + C^T \lambda + a \\ Cx + s - c \\ (S\Lambda - \sigma \mu)e \end{pmatrix} = 0, \quad (4.1.2)$$

where

$$S := \text{diag}(s_1, \dots, s_p), \quad \Lambda := \text{diag}(\lambda_1, \dots, \lambda_p), \quad e := (1, \dots, 1)^T \in \mathbb{R}^p,$$

and $\sigma \in [0, 1]$. For $\sigma = 0$, we recover the original problem (3.3.3). The solutions of (4.1.2) for all positive values of σ and μ define the *central path*. This path lies in the interior of the feasible region and converges for $\sigma\mu \rightarrow 0$ to the solution of the optimization problem (3.3.2). For a fixed value of μ we apply Newton's method to (4.1.2) yielding the Newton correction

$$\nabla F_\mu(x, \lambda, s; \sigma) \begin{pmatrix} \Delta x \\ \Delta \lambda \\ \Delta s \end{pmatrix} = \begin{pmatrix} A & C^T & 0 \\ C & 0 & I \\ 0 & S & \Lambda \end{pmatrix} \begin{pmatrix} \Delta x \\ \Delta \lambda \\ \Delta s \end{pmatrix} = -F_\mu(x, \lambda, s; \sigma) = \begin{pmatrix} -r_x \\ -r_\lambda \\ (-S\Lambda + \sigma \mu)e \end{pmatrix}, \quad (4.1.3)$$

¹Not to be confused with the Lagrange multiplier μ for inequality constraints. We decided to use the notation which is common for interior point methods even though it might cause some slight confusion.

where $r_x := Ax + C^T \lambda - a$, $r_\lambda := Cx + \lambda - c$ and the Newton update then reads

$$(x^+, \lambda^+, s^+) = (x, \lambda, s) + \alpha (\Delta x, \Delta \lambda, \Delta s),$$

where the step size α is chosen in such a way to ensure that $\lambda^+, s^+ > 0$ and possibly other feasibility or descent conditions.

Lemma 4.1.1 *The Jacobi matrix*

$$\nabla F_\mu(x, \lambda, s; \sigma) = \begin{pmatrix} A & C^T & 0 \\ C & 0 & I \\ 0 & S & \Lambda \end{pmatrix},$$

is regular for all $x > 0$, $s > 0$ and $\sigma \geq 0$.

Proof. Let $(u, v, w)^T \in \mathbb{R}^n \times \mathbb{R}^p \times \mathbb{R}^p$ be with $\nabla F_\mu(x, \lambda, s; \sigma)(u, v, w)^T = 0$, i.e.

$$\begin{aligned} Au + C^T v &= 0, \\ Cu + w &= 0, \\ Sv + \Lambda w &= 0. \end{aligned}$$

We first eliminate $w = -\Lambda^{-1}Sv$ by the last equation, so that the second one reads $Cu - \Lambda^{-1}Sv = 0$. Now, using the first equation, we get $0 = -(CA^{-1}C^T + \Lambda^{-1}S)v$. Since A is s.p.d. and C is assumed to have full rank, the matrix $CA^{-1}C^T + \Lambda^{-1}S$ is s.p.d., so that we obtain $v = 0$. The first equation then implies $u = 0$ and the second or third also $w = 0$. This proves our claim. \square

4.2 THE BARRIER METHOD

A second path to derive interior point methods is via so-called *barrier methods*, which we shall briefly introduce for the linear-quadratic problem with inequality constraints of the form (3.3.2). We define the *logarithmic barrier* as

$$\Psi_\mu(x, s) := \frac{1}{2}\langle x, Ax \rangle + \langle a, x \rangle - \mu \sum_{i=1}^p \log(s_i), \quad x \in \mathbb{R}^n, s \in \mathbb{R}_+^p, \mu \geq 0. \quad (4.2.1)$$

Theorem 4.2.1 *The barrier problem*

$$\min_{x \in \mathbb{R}^n, s \in \mathbb{R}^p} \Psi_\mu(x, s), \quad (4.2.2a)$$

$$\text{s.t. } Cx + s = c, \quad (4.2.2b)$$

is equivalent to (3.3.2),

Proof. The Lagrange function for (4.2.2) reads

$$\mathcal{L}_\mu(x, s, \lambda) := \frac{1}{2}\langle x, Ax \rangle + \langle a, x \rangle - \mu \sum_{i=1}^n \log(s_i) + \langle \lambda, Cx + s - c \rangle,$$

and the corresponding KKT conditions read

$$\begin{aligned} 0 &= \nabla_x \mathcal{L}_\mu(x, s, \lambda) = Ax + C^T \lambda + a, \\ 0 &= \nabla_\lambda \mathcal{L}_\mu(x, s, \lambda) = Cx + s - c, \\ 0 &= \nabla_{s_i} \mathcal{L}_\mu(x, s, \lambda) = -\mu s_i^{-1} + \lambda_i, \quad i = 1, \dots, p. \end{aligned}$$

The first two equations coincide with the optimality system (3.3.3) of (3.3.2) and the third equation is equivalent to $S\Lambda - \mu e = 0$, which is the third equation of (3.3.3). \square

Remark 4.2.2 Note, that (4.2.2) does not involve a constraint for s , as the logarithmic part in Ψ_μ automatically implies that $s^* > 0$. \diamond

4.3 SOLVING THE PRIMAL-DUAL SYSTEM

In order to solve the Newton correction (4.1.3) for the primal-dual system (4.1.2) efficiently, it is important to exploit the structure of the system by choosing a suitable direct factorization algorithm, or by choosing an appropriate preconditioner for an iterative solver. First, we can rewrite the system in a more compact form, i.e.,

$$\begin{pmatrix} A & C^T \\ C & -S^{-1}\Lambda \end{pmatrix} \begin{pmatrix} \Delta x \\ \Delta s \end{pmatrix} = \begin{pmatrix} -r_x \\ -r_\lambda + (\Lambda - \sigma \mu S^{-1})e \end{pmatrix}, \quad (4.3.1)$$

since by the third equation

$$S \Delta \lambda = (-S\Lambda + \sigma \mu)e - \Lambda \Delta s, \quad \text{i.e.,} \quad \Delta \lambda = (-\Lambda + \sigma \mu S^{-1})e - S^{-1}\Lambda \Delta s.$$

Inserting this into the second one gives

$$-r_\lambda = -Cx - \lambda + r = C \Delta x + \Delta \lambda = C \Delta x + (-\Lambda + \sigma \mu S^{-1})e - S^{-1}\Lambda \Delta s,$$

which shows (4.3.1), also known as “augmented system”. This system can be further transformed by eliminating Δs using the second equation and inserting this into the first one to obtain

$$(A + C^T \Lambda^{-1} S C) \Delta x = -r_x + C^T \Lambda^{-1} S [-r_\lambda + (\Lambda - \sigma \mu \Lambda^{-1})e],$$

which can be solved e.g. by a sparse Cholesky factorization. This is efficient if $C^T \Lambda^{-1} S C$ is “almost as sparse” as A . The advantage over (4.3.1) or even (4.1.3) is the smaller dimension of the linear system. Another possibility is a reformulation of (4.1.3) as follows

$$\begin{pmatrix} A & 0 & C^T \\ 0 & \Lambda^{-1} S & I \\ C & -I & \Lambda \end{pmatrix} \begin{pmatrix} \Delta x \\ \Delta \lambda \\ \Delta s \end{pmatrix} = \begin{pmatrix} -r_x \\ (-S + \sigma \mu \Lambda^{-1})e \\ -r_\lambda \end{pmatrix}, \quad (4.3.2)$$

which can be solved by a projected cg method as described in [14, Alg. 16.2]. This approach may be useful for problems in which the Cholesky factorization cannot efficiently be performed due to excessive memory demands. The projected CG method only requires a matrix-vector product with A , no factorization.

Now, we have all ingredients for the primal-dual interior point method at hand and formulate it in Algorithm 4.3.1.

Algorithm 4.3.1: Interior point method for linear-quadratic problems

- 1: Determine a feasible initial vector $(x^{(0)}, \lambda^{(0)}, s^{(0)}) \in \mathcal{F}_D^0 := \mathcal{F}_D \cap \{x > 0, s > 0\}$
 - 2: Choose a tolerance $\varepsilon > 0$
 - 3: Choose a centralizing parameter $\sigma \in [0, 1]$
 - 4: Set $k := 0$
 - 5: **do**
 - 6: Set the “duality measure” $\tau_\ell := \frac{1}{n} \langle x^{(k)}, s^{(k)} \rangle$
 - 7: Set $\mu_k := \sigma \tau_\ell$ and solve (4.1.3) for $\mu = \mu_k$; obtain $(\Delta x, \Delta \lambda, \Delta s)$
 - 8: Set $(x^{(k+1)}, \lambda^{(k+1)}, s^{(k+1)}) := (x^{(k)}, \lambda^{(k)}, s^{(k)}) - \alpha(\Delta x, \Delta \lambda, \Delta s)$
with $\alpha \in (0, 1]$ such that $(x^{(k+1)}, s^{(k+1)}) > 0$
 - 9: **while** $\tau_\ell > \varepsilon$
-

The method in Algorithm 4.3.1 is also called *primal-dual interior point method* since the primal variable x as well as the dual variable y are involved.

4.4 NONLINEAR OPTIMIZATION

Finally, we describe the (more or less straightforward) extension of the above described interior point methods for general nonlinear optimization problems of the form

$$\begin{cases} f(x) \rightarrow \min!, & \text{w.r.t. } x \in \mathbb{R}^n \\ \text{s.t. } h(x) = 0, \\ g(x) \leq 0, \end{cases} \quad (4.4.1)$$

where $f : \mathbb{R}^n \rightarrow \mathbb{R}$ is convex and $h : \mathbb{R}^n \rightarrow \mathbb{R}^m$, $g : \mathbb{R}^n \rightarrow \mathbb{R}^p$ having convex components $g_i : \mathbb{R}^n \mapsto \mathbb{R}$ and $g_i \in C^2(\mathbb{R}^n)$ for $i = 1, \dots, p$, i.e., the functions g_i , $i = 1, \dots, p$, are *nonlinear* inequality constraints. In order to proceed, we define the *domain* of a function $f : \mathbb{R}^n \rightarrow \mathbb{R}^n$ by

$$\text{dom}(f) := \{x \in \mathbb{R}^n : \exists y \in \mathbb{R}^n \text{ with } y = f(x)\}. \quad (4.4.2)$$

With that, the *strictly feasible region* of (4.4.1) reads

$$\mathcal{F}^0 := \{x \in \text{dom}(f) : h(x) = 0, g(x) < 0\},$$

where we always assume $\mathcal{F}^0 \neq \emptyset$. As before we define the logarithmic barrier as

$$\Phi(x) := - \sum_{i=1}^p \log(-g_i(x)).$$

Obviously $\Phi \in C^2(\mathbb{R}^n)$ is a convex function with $\text{dom } \Phi = \{x \in \mathbb{R}^n : g(x) < 0\}$. Then, we consider

$$\Psi_\mu(x) := f(x) + \frac{1}{\mu} \Phi(x), \quad x \in \mathcal{F}^0. \quad (4.4.3)$$

As above, we use the barrier function to substitute the general problem (4.4.1) by a problem that only involves equality constraints. The arising convex optimization problem then reads:

$$\begin{cases} \Psi_\mu(x) \rightarrow \min! \\ \text{s.t. } h(x) = 0. \end{cases} \quad (4.4.4)$$

Under the above assumptions, the problem (4.4.4) has a unique solution, which (of course) depends on the parameter μ . We denote this solution by $x^*(\mu)$. Obviously, one can solve (4.4.4) for any fixed parameter $\mu > 0$ for example by the SQP method or Newton's method. One also calls the arising solution the *centering step*, since it determines an element in the central path. The resulting scheme is called *barrier method* and is detailed in Algorithm 4.4.1.

Algorithm 4.4.1: Barrier method

- 1: Choose a feasible initial value $x^{(0)} \in \mathcal{F}^0$
- 2: Choose a centralizing parameter $\sigma > 1$
- 3: Choose $\mu^{(0)} > 0$, the tolerance $\varepsilon > 0$
- 4: **for** $k = 0, 1, 2, \dots$ **do**
- 5: *Centering step* : Solve (4.4.4) with $\mu = \mu^{(k)}$
- 6: Update $x^{(k+1)} = x^*(\mu^{(k)})$
- 7: **if** $\frac{p}{\mu^{(k)}} < \varepsilon$ **then**
- 8: **Stop** $\triangleright p \hat{=} \# \text{inequality constraints}$
- 9: **end if**
- 10: $\mu^{(k+1)} := \sigma \mu^{(k)}$
- 11: **end for**

Obviously, if one has all the involved subroutines at hand, the realization of Algorithm 4.4.1 is relatively easy.

Remark 4.4.1 (a) The solution of (4.4.4) can e.g. be done by linearizing the constraint $h(x) = 0$.

(b) Let \tilde{x}^* be the output of of Algorithm 4.4.1 and $\rho^* := f(\tilde{x}^*)$, then

$$|f(x^*) - \rho^*| \leq \varepsilon.$$

(c) The centering step us usually done by Newton's method.

(d) One often assumes that $\sigma > 1$. If one chooses the parameter σ large, this results in fewer outer iterations, but more inner (Newton-) iterations. Usually, the parameter σ is chosen in the order of about $20 - 30$.

(e) There are many heuristics for the choice of the initial parameter $\mu^{(0)}$, which of course influences the quantitative convergence.

(f) The algorithm is of polynomial complexity.

(g) Several details concerning various related algorithms are described in [14, §19]. \diamond

Example 4.4.2 One more time we consider the constrained Rosenbrock problem in Example 3.4.3. We use the interior-point realization within fmincon. The initial guess $x^{(0)} = (-1.9, 2)^T$, which we also used earlier, is infeasible, but the scheme determines a feasible point after one iteration. The scheme needs 19 iterations with 61 function evaluations to reach a first-order optimality error of $8.0e - 8$ and a feasibility error of 0.

As a second example, we add inequality constraints $Cx \leq c$, with $C = (2, 3)$ and $c = 1$ Indicated by the line in the right graph in Figure 4.1). Again, we start with the infeasible initial value $x^{(0)} = (-1.9, 2)^T$. The scheme needs 36 iterations with 121 function evaluations to reach a first-order optimality error of $3.2e - 7$ and a feasibility error of 0. The history is plotted in Figure 4.1. \diamond

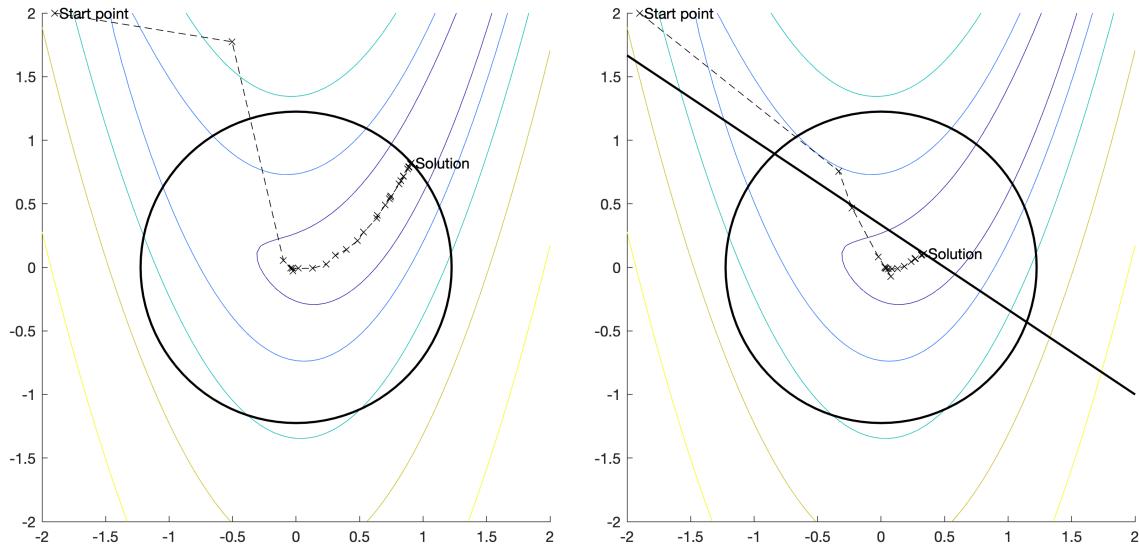


Figure 4.1: Iteration history for the interior point method for the Rosenbrock function with equality constraints only (left) and both inequality and equality constraints (right).

4.5 LINEAR OPTIMIZATION PROBLEMS

Linear optimization problems are widely spread in science, economy and industry. They are typically covered in lectures on optimization, not in numerical optimization. However, due to their relevance

and since interior point methods can be used for linear and nonlinear optimization, we describe the use of interior point methods for the solution of linear optimization problems. In fact, in many books on numerical optimization, interior point methods are introduced via linear optimization problems.

For a given vector $f \in \mathbb{R}^n$, we consider the problem²

$$(P) \quad \begin{cases} f(x) := \langle f, x \rangle \rightarrow \min, & x \in \mathbb{R}^n, \\ \text{s.t. } Bx = b, & x \geq 0, \end{cases} \quad (4.5.1)$$

i.e., the objective function is linear. Within our previous general framework, we can write the constraints as $h(x) := b - Bx : \mathbb{R}^n \rightarrow \mathbb{R}^m$ for the equality and as $g(x) := -x : \mathbb{R}^n \rightarrow \mathbb{R}^n$, i.e., $p = n$ for the inequality constraints. As before, we determine the Lagrange function and compute the dual problem, i.e.,

$$\begin{aligned} \mathcal{L}(x, \lambda, s) &:= f(x) + \langle \lambda, b - Bx \rangle + \langle s, -x \rangle \\ \varphi(\lambda, s) &:= \inf_{x \in \mathbb{R}^n} \mathcal{L}(x, \lambda, s) = \inf_{x \in \mathbb{R}^n} \{ \langle f, x \rangle - \langle B^T \lambda, x \rangle + \langle \lambda, b \rangle - \langle s, x \rangle \} \\ &= \langle \lambda, b \rangle + \underbrace{\inf_{x \in \mathbb{R}^n} \{ \langle x, f - B^T \lambda - s \rangle \}}_{=: \psi(x)}. \end{aligned}$$

Thus, $\psi : \mathbb{R}^n \rightarrow \mathbb{R}$ with $\psi'(x) = f - B^T \lambda - s$ and therefore

$$\varphi(\lambda, s) = \begin{cases} \langle \lambda, b \rangle, & \text{if } s = f - B^T \lambda, \\ -\infty, & \text{else.} \end{cases}$$

Note, that

$$s \in \mathbb{R}_+^n \iff f - B^T \lambda \geq 0 \iff B^T \lambda \leq f,$$

and the dual problem can be written as

$$(D) \quad \begin{cases} \langle \lambda, b \rangle \rightarrow \max, & \lambda \in \mathbb{R}^m, s \in \mathbb{R}^n, \\ \text{s.t. } B^T \lambda + s = f, & s \geq 0. \end{cases} \quad (4.5.2)$$

Obviously, the Lagrange multiplier s is the *slack variable* for the inequality constraint, which is the reason for choosing the letter s .

It is easy to show that the dual problem of (4.5.2) coincides with the original problem (4.5.1). In fact, the Lagrange function of the dual problem reads (in this case, the variable x is the Lagrange multiplier)

$$\tilde{\mathcal{L}}(\lambda, s, x) = -\langle b, \lambda \rangle + \langle B^T \lambda + s - f, x \rangle = -\langle f - s, x \rangle + \langle Bx - b, \lambda \rangle.$$

The feasible sets for primal and dual problems, respectively, read (for the dual problem without the slack variable)

$$\mathcal{F}_P := \{x \in \mathbb{R}^n : Bx = b, x \geq 0\}, \quad \mathcal{F}_D := \{\lambda \in \mathbb{R}^m : B^T \lambda \leq f\}.$$

For $\lambda \in \mathcal{F}_D$ and $x \in \mathcal{F}_P$ we get the *weak duality* (see also Theorems 3.2.2 and 3.2.3 for the general case)

$$\langle f, x \rangle \geq \langle B^T \lambda, x \rangle = \langle \lambda, Bx \rangle = \langle \lambda, b \rangle. \quad (4.5.3)$$

With that preparation at hand, we can now easily prove the duality.

Lemma 4.5.1 *Let $x^* \in \mathcal{F}_P$, $\lambda^* \in \mathcal{F}_D$ with $\langle f, x^* \rangle = \langle \lambda^*, b \rangle$. Then, x^* , λ^* are the optimal solutions of (4.5.1) and (4.5.2), respectively.*

²The slight abuse of notation should not cause a problem when distinguishing between the objective function $f(x)$ and the vector $f \in \mathbb{R}^n$.

Proof. With (4.5.3), we get that for all $x \in \mathcal{F}_P$ it holds

$$\langle f, x \rangle \geq \langle \lambda^*, b \rangle = \langle f, x^* \rangle$$

and for all $\lambda \in \mathcal{F}_D$ again by (4.5.3)

$$\langle b, \lambda \rangle \leq \langle f, x^* \rangle = \langle b, \lambda^* \rangle$$

is valid, which implies that λ^* solves (4.5.2) and completes the proof. \square

Using the dual problem and the slack variable s we are now able to prove an optimality condition.

Theorem 4.5.2 (Optimality condition) *The following statements are equivalent:*

- (a) *The primal problem (4.5.1) possess a solution x^* .*
- (b) *The dual problem (4.5.2) possess a solution (λ^*, s^*) .*
- (c) *The following optimality system possess a solution (x^*, λ^*, s^*) ,*

$$Bx = b \tag{4.5.4a}$$

$$B^T \lambda + s = f \tag{4.5.4b}$$

$$x_i s_i = 0, \quad i = 1, \dots, n, \quad x, s \geq 0. \tag{4.5.4c}$$

Proof. “(a) \Leftrightarrow (c)”: The primal problem (4.5.1) is a convex problem with linear constraints. Hence, we know that a solution x^* of (4.5.1) exists, if and only if there exist $\lambda^* \in \mathbb{R}^m$, $s^* \in \mathbb{R}_+^n$, so that the KKT conditions (3.1.3) corresponding to (4.5.1) are fulfilled. The Lagrange function of (4.5.1) is given by:

$$\mathcal{L}_P(x, \lambda, s) = \langle f, x \rangle + \langle \lambda, b - Bx \rangle + \langle s, -x \rangle.$$

Therefore the KKT condition (3.1.3) read

$$\begin{aligned} \nabla_x \mathcal{L}_P(x^*, \lambda^*, s^*) &= f - s^* - B^T \lambda^* = 0, \\ x^*, s^* &\geq 0, \\ Bx^* &= b, \\ \langle x^*, s^* \rangle &= 0, \end{aligned}$$

which obviously are equivalent to (4.5.4).

“(b) \Leftrightarrow (c)”: In a similar way as above, we show that $\lambda^* \in \mathbb{R}^m$ is a solution of (4.5.2) if and only if a vector (or multiplier) x^* exists, such that the pair satisfies the KKT conditions corresponding to (4.5.2). To this end, we again consider the dual Lagrange function (regarding to (4.5.2)):

$$\mathcal{L}_D(\lambda, s, x) = \langle -b, \lambda \rangle + \langle x, B^T \lambda - f \rangle + \langle s, x \rangle.$$

This results in the KKT conditions:

$$\begin{aligned} \nabla_\lambda \mathcal{L}_D(\lambda^*, s^*, x^*) &= Bx^* - b = 0, \\ s^* &\geq 0, x^* \geq 0, \\ B^T \lambda^* + s^* &= f, \\ \langle x^*, s^* \rangle &= 0, \end{aligned}$$

which are equivalent to (4.5.4). This completes the proof. \square

A so-called “dual theorem” clarifies all possibly cases that might occur concerning the existence of solutions, see [8, 246-247]. In order to formulate it, we need some further preparations.

Definition 4.5.3 (a) The set of strictly feasible points is defined as

$$\mathcal{F}_P^0 := \{x \in \mathbb{R}^n : Bx = b, x > 0\}, \quad \mathcal{F}_D^0 := \{\lambda \in \mathbb{R}^m : B^T \lambda < f\}. \quad (4.5.5)$$

- (b) We say that the primal-dual system (4.5.1), (4.5.2) satisfies the interior point condition (IPC), if $\mathcal{F}_P^0 \neq \emptyset$ and $\mathcal{F}_D^0 \neq \emptyset$.
(c) The logarithmic barriers Φ_μ of (4.5.1) and $\tilde{\Phi}_\mu$ of (4.5.2) are defined as

$$\Phi_\mu(x) := \langle f, x \rangle - \mu \sum_{i=1}^n \log(x_i), \quad \tilde{\Phi}_\mu(y) := \langle b, \lambda \rangle + \mu \sum_{i=1}^n \log(f_i - \langle A_i, \lambda \rangle), \quad (4.5.6)$$

with the barrier parameter μ and A_i denoting the columns of A .

Remark 4.5.4 The barrier function Φ_μ is strictly convex on \mathcal{F}_P^0 . The die dual barrier function $\tilde{\Phi}_\mu$ is strictly concave on \mathcal{F}_D^0 , see [8, 248-249].

With that we are able to formulate the announced existence theorem.

Theorem 4.5.5 Let $\mu > 0$. Then the following statements are equivalent:

- (a) $\mathcal{F}_P^0 \neq \emptyset, \mathcal{F}_D^0 \neq \emptyset$.
- (b) There exists a unique minimizer of Φ_μ in \mathcal{F}_P^0 .
- (c) There exists a unique maximizer of $\tilde{\Phi}_\mu$ in \mathcal{F}_D^0 .
- (d) The system $F_\mu(x, \lambda, s) = 0, x \geq 0, s \geq 0$ with

$$F_\mu(x, \lambda, s) := \begin{pmatrix} B^T \lambda + s - f \\ Bx - b \\ Xs - \mu e \end{pmatrix}, \quad e := (1, \dots, 1)^T, \quad X = (\text{diag}(x_i))_i$$

has a unique solution.

If one of the equivalent conditions (a)-(d) holds, the minimizer $x^*(\mu)$ to Φ_μ and $\lambda^*(\mu)$ to $\tilde{\Phi}_\mu$ yield a solution to $F_\mu(x, \lambda, s) = 0, x \geq 0, s \geq 0$, where $s^*(\mu) := f - B^T \lambda^*(\mu)$.

Proof. See [8, 249-261]. □

With the notation above, the optimality condition (4.5.4) can be written as a nonlinear system of equations

$$F_0(x, \lambda, s) = 0. \quad (4.5.7)$$

We want to solve this with Newton's method. One can (easily) show that the Jacobi matrix reads

$$\nabla F_0(x, \lambda, s) = \begin{pmatrix} 0 & B^T & I \\ B & 0 & 0 \\ S & 0 & X \end{pmatrix} \text{ where } S := \text{diag}(s_i)$$

is regular for $x > 0$ and $s > 0$. Since the solution (x^*, λ^*, s^*) of (4.5.7) is on the boundary of the primal-dual feasible region

$$\mathcal{F} := \{(x, \lambda, s) \in \mathbb{R}_+^n \times \mathbb{R}^m \times \mathbb{R}_+^n : Bx = b, B^T \lambda + s = f\}, \quad (4.5.8)$$

one considers the “relaxed nonlinear system” introduced in Theorem 4.5.5 (d), namely

$$F_\mu(x, \lambda, s) = 0, \quad \mu > 0. \quad (4.5.9)$$

Because of $x_i s_i = \mu > 0$ for $i = 1, \dots, n$ we get $x_i > 0, s_i > 0$ in (4.5.9). We apply Newton's method to (4.5.9). By Theorem 4.5.5, the interior point condition (IPC) is independent of μ which implies the following: if a solution exists for some $\mu > 0$, then a solution exists for all μ . This observation leads to the following definition.

Definition 4.5.6 Let $r(\mu) := (x(\mu), \lambda(\mu), s(\mu))$ denote the solutions of (4.5.9) for $\mu > 0$. Then,

$$\mathcal{C} := \{r(\mu) : \mu > 0\} \quad (4.5.10)$$

is called (primal-dual) central path.

One can show that the limits as $\mu \rightarrow 0$ exist and that these limits are the desired solutions, see for example [8].

Theorem 4.5.7 The following limits

$$x^* := \lim_{\mu \rightarrow 0^+} x(\mu), \quad \lambda^* := \lim_{\mu \rightarrow 0^+} \lambda(\mu)$$

exist and x^*, λ^* solve (4.5.1) and (4.5.2), respectively. \square

Now we can formulate the algorithm for the interior point method for linear optimization problems in Algorithm 4.5.1.

Algorithm 4.5.1: Interior point method for linear optimization

- 1: Determine some feasible initial vector $(x^{(0)}, \lambda^{(0)}, s^{(0)}) \in \mathcal{F}^0 := \mathcal{F} \cap \{x > 0, s > 0\}$
- 2: Choose a tolerance $\varepsilon > 0$ and the centralizing parameter $\sigma \in [0, 1]$
- 3: Set $k := 0$
- 4: **do**
- 5: Set the duality measure $\tau_\ell := \frac{1}{n} \langle x^{(k)}, s^{(k)} \rangle$
- 6: Solve the linear system $\nabla F_0(\Delta x, \Delta \lambda, \Delta s)^T = (0, 0, XSe - \sigma \tau_\ell e)^T$
- 7: Set $(x^{(k+1)}, \lambda^{(k+1)}, s^{(k+1)}) := (x^{(k)}, \lambda^{(k)}, s^{(k)}) - \alpha(\Delta x, \Delta \lambda, \Delta s)$
with $\alpha > 0$ such that $(x^{(k+1)}, s^{(k+1)}) > 0$
- 8: **while** $\tau_\ell > \varepsilon$

Remark 4.5.8 For the choice of the damping parameter α , the following approach has turned out to be useful: Choose $\eta \in (0, 1)$ ($\eta \approx 1$ for example $\eta = 0.99$) and set

$$\alpha := \min \left\{ 1, \eta \min_{\Delta x_\nu > 0} \frac{x_\nu}{\Delta x_\nu}, \eta \min_{\Delta s_\nu > 0} \frac{s_\nu}{\Delta s_\nu} \right\} = \eta \cdot \max \left\{ \eta, \max_\nu \frac{\Delta x_\nu}{x_\nu}, \max_\nu \frac{\Delta s_\nu}{s_\nu} \right\}^{-1}.$$

Remark 4.5.9 The algorithm has polynomial complexity $\mathcal{O}(n^\bullet |\ln(\varepsilon)|)$, see [8, 265-6].

Example 4.5.10 A local brewery produces two kind of drinks. The profit for 1 liter of drink 1 (D1) is €4 and €3 for 1 liter of D2. Both drinks consist of two resources, which are limited, i.e.,

$$2x_1 + 3x_2 \leq 8, \quad 3x_1 + 2x_2 \leq 7.$$

Finally, of course, the amounts cannot be negative, i.e., $x_1 \geq 0, x_2 \geq 0$. The following Matlab realization shows how to use the interior point method within `linprog` for the solution of this linear optimization problem. The scheme terminates after 3 iteration with the optimal value $x^* = (0.8, 1.8)$ and maximal profit of €8.60. \diamond

MATLAB-function: Brewery.m

```
1 % coefficients for objective function
2 f = -[4; 3]; % minus sign for maximization
3
4 %- inequality constraints
5 B = [2, 3; 3, 2];
6 b = [7; 6];
7
8 %- variables are non-negative - lower bounds
9 lb = [0; 0];
10 ub = [];
11
12 options = optimoptions('linprog','Algorithm','interior-point',...
13     'Display','iter');
14 [x, fval] = linprog(f, B, b, [], [], lb, ub, options);
15
16 %- output
17 fprintf('Optimal amount of productin for D1 (x1): %.2f\n', x(1));
18 fprintf('Optimal amount of productin for D2 (x2): %.2f\n', x(2));
19 fprintf('Maximal profit: %.2f\n', -fval);
20 %- nagative value due to minimization
```

5 NON-SMOOTH OPTIMIZATION

Except for the direct search algorithms (Hooke-Jeeves and Nelder-Mead), all methods presented so far rely on the assumption of a certain smoothness of the objective function. Unfortunately, for many “real-world” problems this is generally not the case. That is the reason, why we now consider non-smooth problems, i.e., optimization problems with non-smooth objective function. Numerical methods for non-smooth optimization problems have gained particular interest recently also due to the fact that such problems appear in the training of neural networks, where the corresponding loss function is often not smooth.

5.1 EXAMPLES

We start by collecting some examples of non-smooth optimization problems.

Example 5.1.1 (Wolfe function) Let $f : \mathbb{R}^2 \rightarrow \mathbb{R}$ be defined as

$$f(x_1, x_2) := \begin{cases} 5\sqrt{9x_1^2 + 16x_2^2}, & x_1 \geq |x_2|, \\ 9x_1 + 16|x_2|, & 0 < x_1 < |x_2|, \\ 9x_1 + 16|x_2| - x_1^9, & x_1 \leq 0, \end{cases}$$

which is known as *Wolfe function*, see Figure 5.1. It is convex and continuous, but non-differentiable on $M := \{(x_1, x_2)^T \in \mathbb{R}^2 : x_1 \leq 0, x_2 = 0\}$, which is a set of Lebesgue measure zero. The function f has a unique minimum $x^* = (-1, 0)^T \in M$. Moreover, it is known (see, e.g. [2, §1.3]) that the gradient descent method (without modifications which are usually done in Matlab-internal routines) converges towards the non-optimal point $(0, 0)^T$ for all initial points $x^{(0)} \in S := \{(x_1, x_2)^T \in \mathbb{R}^2 : x_1 > |x_2| > (\frac{9}{16})^2|x_1|\}$, which shows the need for specific numerical schemes for non-smooth problems. \diamond

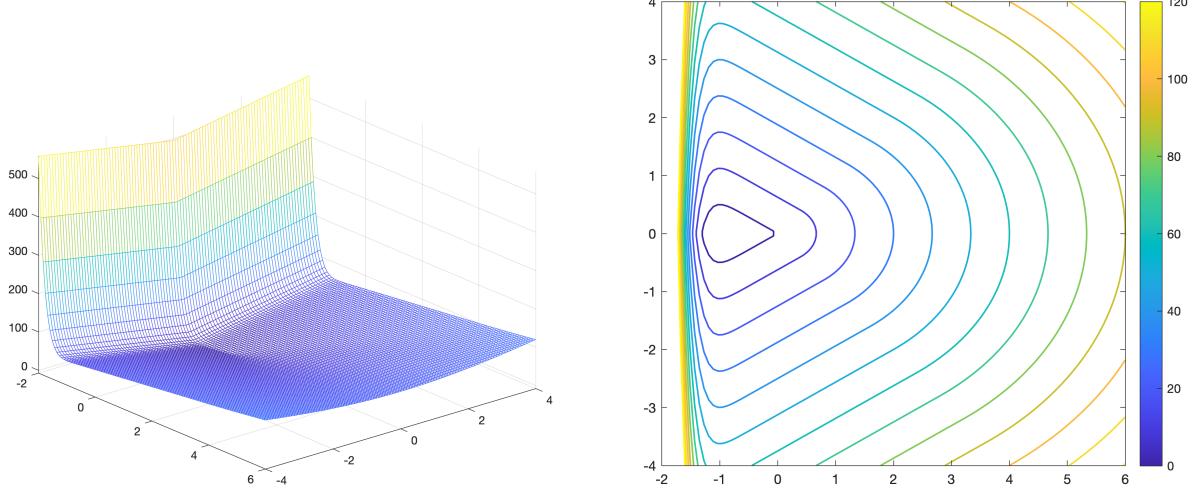


Figure 5.1: Wolfe function: 3D view (left), isolines (right).

Example 5.1.2 (ℓ^1 -minimization) Let $A \in \mathbb{R}^{d \times d}$ and $a \in \mathbb{R}^n$. Then, minimizing $f(x) := \|Ax - a\|_1$ for the ℓ_1 -norm $\|x\|_1 := \sum_{i=1}^n |x_i|$ is known as ℓ^1 -minimization. The function f is not smooth and numerical methods for least-squares problems cannot be used. \diamond

Example 5.1.3 (The max-function) Let $f_i : \mathbb{R}^d \rightarrow \mathbb{R}$, $i = 1, \dots, n$, be (smooth) functions. Then $f(x) := \max\{f_1(x), \dots, f_n(x)\} : \mathbb{R}^d \rightarrow \mathbb{R}$ is in general non smooth. If all f_i are convex, then f is also convex. The special case $f_i(x) = (Ax - a)_i$ leads to the ℓ^∞ -minimization problem $f(x) := \|Ax - a\|_\infty \rightarrow \min$. \diamond

Example 5.1.4 (Location problem) Consider a warehouse chain with k branches at the locations a_1, \dots, a_k and suppliers at a_{k+1}, \dots, a_m . The question to be considered reads: where is the optimal position for a new storehouse?

Therefore, let $f_i(x), f_i : \mathbb{R}^2 \rightarrow \mathbb{R}$ be the cost functions that originate from the transport of the product from a position x to the location a_i . The easiest case would be if the cost is proportional to the distance, i.e., one can choose $f_i(x) := \omega_i \cdot \|x - a_i\|$, $i = 1, \dots, m$, in that case. Then, one gets

$$f(x) := \sum_{i=1}^m \omega_i \|x - a_i\|_2 \longrightarrow \min_{x \in \mathbb{R}^2}! \quad (5.1.1)$$

Note, that here the objective function f is not differentiable at the positions a_i .

As we cannot build a new storehouse everywhere we would like to, the location is also subject to constraints. Thus, the location we searching for has to be in a subset $\mathcal{F} \subset \mathbb{R}^2$. \diamond

5.2 NON-SMOOTH, CONVEX PROBLEMS

We have seen already that convex optimization problems (i.e., convex objective functions and convex feasible set) guaranty a unique (local) solution at least. Therefore, we start by considering the general convex optimization problem

$$\min f(x), \quad x \in \mathcal{F}, \quad (5.2.1)$$

where $f : \mathbb{R}^n \rightarrow \mathbb{R}$ is convex (but in general not differentiable) and $\mathcal{F} \subset \mathbb{R}^n$ is a convex set. The assumed convexity allows us a global treatment because each local minimum of a convex function is automatically a global one, see Theorem 3.1.20.

We start by presenting some simple examples for problems of kind (5.2.1).

Example 5.2.1 For the restricted (primal) optimization problem

$$(P) \begin{cases} \min_{x \in \mathcal{F}} f(x), \\ \text{s.t. } g(x) \leq 0, h(x) = 0, \end{cases} \quad (5.2.2)$$

consider the Lagrange function

$$\mathcal{L}(x, \lambda, \mu) := f(x) + \sum_{i=1}^m \lambda_i g_i(x) + \sum_{j=1}^p \mu_j h_j(x) = f(x) + \langle \lambda, g(x) \rangle + \langle \mu, h(x) \rangle$$

and the arising dual function $\varphi(\lambda, \mu) := \inf_{x \in \mathcal{F}} \mathcal{L}(x, \lambda, \mu)$. Then, the dual problem reads

$$(D) \begin{cases} \max \varphi(\lambda, \mu), \\ \text{s.t. } \lambda \geq 0, \mu \in \mathbb{R}^p, \end{cases} \quad (5.2.3)$$

which means that we get significantly simpler constraints in the dual problem. However, the dual objective function φ is in general not differentiable, even if the original cost function f might be smooth. \diamond

A similar observation holds true for the seemingly simple approach to treat inequality constraints presented in (3.0.3) (page 65). In general, the involved max-function destroys the smoothness, i.e., the modified function for equality constraints is in general non-smooth even in case that the original inequality constraint function might be smooth.

So far, we only know two classes of methods for such kind of problems:

- (1) Direct search algorithms, whose convergence might be unsatisfying so that these methods may be extremely slow in practice.
- (2) Methods that use information on the derivatives in some suitable form. In general, this yields to significant faster methods, but requires smoothness.

From this, we motivate the aim to introduce a concept for optimization schemes for non-smooth objective functions, in which we can use a surrogate instead of the gradient of smooth functions in order to get efficient algorithms. This leads to the notion of the *subgradient*.

5.3 THE SUBGRADIENT METHOD

A seemingly attractive approach is to regularize non-smooth objective function by adding some smooth terms. However, this often destroys the behavior of the original problem and the regularization term has to be chosen very carefully in order to control the overall error. A different approach is to look for a more general definition of a “derivative” which is also well-defined for functions that are not differentiable in a classical sense. One such generalization are subdifferentials or subgradients to be introduced next.

5.3.1 Subgradients

For the motivation, let us start by considering a convex and continuously differentiable function $f : \mathcal{F} \rightarrow \mathbb{R}$ on some open set $\mathcal{F} \subset \mathbb{R}^n$, $f \in C^1(\mathcal{F})$. We start by rewriting the convexity in the following form: Setting $u := y$, $v := x - y$ and $0 < \alpha < 1$, we have

$$\begin{aligned} f(u + \alpha v) &= f(y + \alpha(x - y)) = f(\alpha x + (1 - \alpha)y) \\ &\leq \alpha f(x) + (1 - \alpha)f(y) \\ &= (1 - \alpha)f(u) + \alpha f(u + v). \end{aligned}$$

Now, let $x, x + ts \in \mathcal{F}$ for some $t > 0$, $s \in \mathcal{F}$ and define the *difference quotient* of f at the point x in direction s with the step size t as

$$\Delta_s f(x; t) := \frac{f(x + ts) - f(x)}{t}.$$

For any $0 < \alpha < 1$ it can be estimated by convexity as

$$\begin{aligned} \Delta_s f(x; \alpha t) &= \frac{f(x + \alpha ts) - f(x)}{\alpha t} \\ &\leq \frac{(1 - \alpha)f(x) + \alpha f(x + ts) - f(x)}{\alpha t} = \frac{f(x + ts) - f(x)}{t} = \Delta_s f(x; t). \end{aligned}$$

Then, for arbitrary $x, y \in \mathcal{F}$ and setting $s := y - x$, we obtain

$$\begin{aligned} f(y) - f(x) &= f(x + s) - f(x) = \Delta_s f(x; 1) \\ &\geq \inf_{t>0} \Delta_s f(x; t) = \lim_{t \rightarrow 0+} \Delta_s f(x; t), \end{aligned}$$

since $\Delta_s f(x; t)$ is decreasing as a function of t due to the convexity of f . Since f was assumed to continuously differentiable (thus in particular locally Lipschitz continuous), we obtain that

$$|\Delta_s f(x; t)| = \frac{|f(x + ts) - f(x)|}{t} \leq \frac{L \cdot \|ts\|}{t} = L \cdot \|s\|,$$

so that $\inf_{t>0} \Delta_s f(x; t) \geq -L\|s\|$ holds. This implies that the limit as $t \rightarrow 0$ exists. This framework can be generalized to convex functions without assuming smoothness as we shall see now.

Lemma 5.3.1 *Let $f : \mathbb{R}^n \rightarrow \mathbb{R}$ be convex and $x_0 \in \text{int}(\text{dom}(f))$, see (4.4.2). Then, f is locally Lipschitz continuous, i.e., for any $z \in \text{int}(\text{dom}(f))$ there is a $\delta = \delta(z) > 0$ and some $L = L(z) \geq 0$ such that $|f(x) - f(y)| \leq L\|x - y\|$ for all $x, y \in B_\delta(z)$.*

Proof. See e.g. [2, Satz 2.7.3]. \square

The following is an immediate consequence, which we note for later reference.

Corollary 5.3.2 *Let $f : \mathbb{R}^n \rightarrow \mathbb{R}$ be convex and $x_0 \in \text{int}(\text{dom}(f))$, see (4.4.2). Then, $|\Delta_s f(x; t)| \leq L \cdot \|s\|$ for all $s \in \mathbb{R}^n$. \square*

On the other hand, for smooth f , the limit of the difference quotient is the directional derivative of f at x in direction s , which is the scalar product of the gradient with the direction under consideration, i.e.,

$$\lim_{t \rightarrow 0+} \Delta_s f(x; t) = \frac{\partial}{\partial s} f(x) = \langle \nabla f(x), s \rangle = \nabla f(x)^T (y - x),$$

since $s = y - x$. Hence, it follows

$$f(y) \geq f(x) + \nabla f(x)^T (y - x), \quad (5.3.1)$$

which we already know from Theorem 2.1.8. This inequality can now be used to define a more general notion of a gradient by substituting $\nabla f(x)$ in (5.3.1) by some vector s (or even a set of vectors) that satisfy the same inequality. This yields to the following notion.

Definition 5.3.3 *Let $\mathcal{F} \subset \mathbb{R}^n$ be an open and convex set and let $f : \mathcal{F} \rightarrow \mathbb{R}$ be convex (but not necessarily differentiable). A vector $s \in \mathbb{R}^n$ is called (convex) subgradient of f at the point $x_0 \in \mathcal{F}$, if*

$$f(y) \geq f(x_0) + s^T (y - x_0) \quad \forall y \in \mathcal{F} \quad (5.3.2)$$

holds. The set

$$\partial f(x_0) := \{s \in \mathbb{R}^n : s \text{ is a convex subgradient of } f \text{ in } x_0\} \quad (5.3.3)$$

is called convex subdifferential of f at x_0 .

In the univariate case (i.e., in 1D), one can easily characterize the convex subdifferential.

Example 5.3.4 Let $f : \mathbb{R} \rightarrow \mathbb{R}$, i.e., $n = 1$. Then, the convex subdifferential $\partial f(x_0)$ consists of all hyperplanes across $(x_0, f(x_0))$ which meet the graph of f in x_0 and are below this graph. Hence, $g \in \partial f(x_0)$, if $\ell(x) = f(x_0) + g^T(x - x_0)$ is on or below the graph of f , see Figure 5.2 \diamond

We define a generalized directional derivative by

$$f'(x, s) := \inf_{t>0} \Delta_s f(x; t). \quad (5.3.4)$$

It is an immediate consequence of Corollary 5.3.2 that

$$|f'(x_0, s)| \leq L(x_0)\|s\|$$

for all $s \in \mathbb{R}^n$. Moreover, this implies that $f'(x_0, 0) = 0$. Next, one can show some fundamental properties of convex subdifferentials.

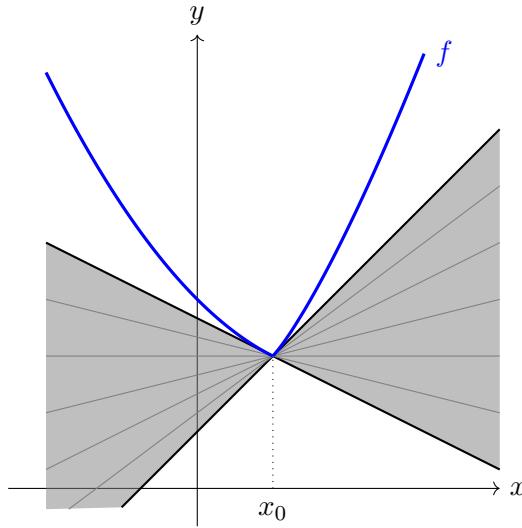


Figure 5.2: Subgradients of f at the point x_0 : all lines in the gray sector going through the point $(x_0, f(x_0))$ form the subdifferential $\partial f(x_0)$.

Theorem 5.3.5 Let $f : \mathbb{R}^n \rightarrow \mathbb{R}$ be convex and $x_0 \in \text{int}(\text{dom}(f))$ ¹. Then,

- (a) $\partial f(x_0) = \{g \in \mathbb{R}^n : g^T s \leq f'(x_0, s) \quad \forall s \in \mathbb{R}^n\}$.
- (b) For all $s \in \mathbb{R}^n$, it holds that $f'(x_0, s) = \max_{g \in \partial f(x_0)} g^T s$.
- (c) The subdifferential $\partial f(x_0)$ is not empty, convex and compact.
- (d) The function f is differentiable in x_0 with $\nabla f(x_0) = g$, if and only if $\partial f(x_0) = \{g\}$ (i.e., the convex subdifferential consists of only one element, namely the gradient).

Proof. (a): “ \subseteq ”: Let $g \in \partial f(x_0)$. Then, by definition $f(x_0 + ts) \geq f(x_0) + t g^T s$ for any $s \in \mathbb{R}^n$, so that

$$\frac{f(x_0 + ts) - f(x_0)}{t} \geq g^T s.$$

By taking the limit, we get $g^T s \leq f'(x_0, s)$.

“ \supseteq ”: Let $g \in \mathbb{R}^n$ with $g^T s \leq f'(x_0, s)$ for all $s \in \mathbb{R}^n$. Then,

$$g^T s \leq f'(x_0, s) = \inf_{\tilde{t} > 0} \Delta_s f(x; \tilde{t}) \leq \frac{f(x_0 + ts) - f(x_0)}{t}$$

for all $t > 0$. If $x + s \in \text{dom}(f)$, then $[x_0, x_0 + s] \subset \text{dom}(f)$ and setting $t = 1$ yields

$$f(x_0 + s) \geq f(x_0) + g^T s. \quad (5.3.5)$$

If, on the other hand, $x + s \notin \text{dom}(f)$, then $f(x + s) = \infty$, which also implies that (5.3.5) holds true. This means that (5.3.5) holds for all $s \in \mathbb{R}^n$. Let $y \in \mathbb{R}^n$ be arbitrary, set $s := y - x_0$ and deduce from (5.3.5) that

$$f(y) = f(x_0 + s) \geq f(x_0) + g^T (y - x_0),$$

which means that $g \in \partial f(x_0)$.

(b): By (a) we have for any $s \in \mathbb{R}^n$ that $g^T s \leq f'(x_0, s)$ for all $g \in \partial f(x_0)$, i.e., $f'(x, s) \geq \sup_{g \in \partial f(x_0)} g^T s$.

Hence, it remains to show that for any $s \in \mathbb{R}^n$ there is a $g_s \in \partial f(x_0)$ such that $g_s^T s = f'(x, s)$. Since $f'(x_0, \cdot)$ is convex on \mathbb{R}^n , for all $s \in \text{int}(\text{dom}(f'(x_0, \cdot))) = \mathbb{R}^n$ there is a $g_s \in \mathbb{R}^n$ such that

$$f'(x_0, y) \geq f'(x_0, s) + g_s^T (y - s) \quad (5.3.6)$$

¹For the definition, see (4.4.2).

for all $y \in \mathbb{R}^n$. Since $f'(x_0, 0)$, we get by inserting $y = 0$ into (5.3.6) that $0 \geq f'(x_0, s) - g_s^T s$, i.e., $g_s^T s \geq f'(x_0, s)$. Since $f'(x_0, 2s) = 2f'(x_0, s)$ we get by inserting $y = 2s$ in (5.3.6) that $2f'(x_0, s) \geq f'(x_0, s) + g_s^T s$, i.e., $g_s^T s \leq f'(x_0, s)$, so that we have proven equality. Then, again by (5.3.6)

$$f'(x_0, y) \geq f'(x_0, s) + g_s^T(y - s) = f'(x_0, s) + g_s^T y - g_s^T s = f'(x_0, s) + g_s^T y - f'(x_0, s) = g_s^T y,$$

which shows that $g_s \in \partial f(x_0)$ by using (a).

(c): First of all $\partial f(x_0) \neq \emptyset$, see e.g. [2, Satz 2.6.3]. Convexity and closedness simply follow from the definition (5.3.3). By (b) we get $g^T s \leq f'(x_0, s) \leq |f'(x_0, s)| \leq L(x_0)\|s\|$ for all $s \in \mathbb{R}^n$. Hence, $\|g\| \leq L(x_0)$ for all $g \in \partial f(x_0)$, which means that $\partial(x_0)$ is bounded.

(d): Let f be differentiable in x_0 , then by definition of the gradient, we have $f'(x_0, s) = s^T \nabla f(x_0)$. By (a) we thus get that

$$\partial f(x_0) = \{g \in \mathbb{R}^n : g^T s \leq s^T \nabla f(x_0) \quad \forall s \in \mathbb{R}^n\}.$$

Hence, for any $g \in \partial f(x_0)$, we have that

$$\langle g - \nabla f(x_0), s \rangle \leq 0 \quad \text{for all } s \in \mathbb{R}^n,$$

i.e., $g = \nabla f(x_0)$. The assertion now follows since $\nabla f(x_0) \in \partial f(x_0)$. \square

We now state some rules for computing with subdifferential. The (simple) proofs are exercises.

Theorem 5.3.6 (Subdifferential calculus) (a) Let $\mathcal{F} \subset \mathbb{R}^n$ be open, convex and let $f_i : \mathcal{F} \rightarrow \mathbb{R}$, $1 \leq i \leq m$, be convex functions. Then, the function $f : \mathcal{F} \rightarrow \mathbb{R}$ defined by

$$f(x) := \max_{1 \leq i \leq m} f_i(x)$$

is convex. With the index set $I(x) := \{1 \leq i \leq m : f_i(x) = f(x)\}$, $x \in \mathcal{F}$, it holds that

$$f'(x, s) = \max_{i \in I(x)} f'_i(x, s) \quad \partial f(x) = \text{conv} \left\{ \bigcup_{i \in I(x)} \partial f_i(x) \right\}. \quad (5.3.7)$$

(b) Let $f : \mathbb{R}^m \rightarrow \mathbb{R}^n$ be convex and $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$. Then, the function $g : \mathbb{R}^n \rightarrow \mathbb{R}$ defined by

$$g(x) := f(Ax + b)$$

is also convex and it holds that

$$\partial g(x) = A^T \partial f(Ax + b).$$

(c) Let the functions $f_1, \dots, f_m : \mathbb{R}^n \rightarrow \mathbb{R}$ as well as $g : \mathbb{R}^m \rightarrow \mathbb{R}$ be convex such that $g(y) \leq g(z)$ for all $y, z \in \mathbb{R}^m$ with $y \leq z$ (component-wise). Then, the composed function

$$h : x \mapsto g(f_1(x), \dots, f_m(x))$$

is also convex and it holds

$$\partial h(x) = \left\{ \sum_{i=1}^m v_i w_i : v \in \partial g(f_1(x), \dots, f_m(x)), w_i \in \partial f_i(x) \right\},$$

i.e., an analogue to the chain rule. \square

Remark 5.3.7 From (c) we get that

$$\partial(f_1 + \dots + f_m)(x) = \partial f_1(x) + \dots + \partial f_m(x)$$

holds. Obviously, this is an important property for numerical methods.

We know that for the differentiable case a critical point (root of the derivative) is a local minimum under appropriate assumptions. We are now able to prove an analogy also for non-smooth functions.

Theorem 5.3.8 (First order optimality conditions) *Let $\mathcal{F} \subset \mathbb{R}^n$ be open and convex, $f : \mathcal{F} \rightarrow \mathbb{R}$ convex, $x^* \in \mathcal{F}$. Then, the following statements are equivalent:*

- (a) x^* is a (local and therefore global) minimum of f on \mathcal{F} .
- (b) It holds that $f'(x^*, s) \geq 0$ for all $s \in \mathbb{R}^n$.
- (c) $0 \in \partial f(x^*)$.

Proof. “(a) \Rightarrow (b)”: Let $s \in \mathbb{R}^n$ be arbitrary. Then, $f(x^* + ts) \geq f(x^*)$, $t > 0$, and therefore

$$f'(x^*, s) = \inf_{t>0} \Delta_s f(x^*; t) = \inf_{t>0} \frac{f(x^* + ts) - f(x^*)}{t} \geq 0,$$

since x^* is a minimum of f on \mathcal{F} , which shows (b).

“(b) \Rightarrow (c)”: We have $0^T s = 0 \leq f'(x^*, s)$ for all $s \in \mathbb{R}^n$ by assumption. Thus, $0 \in \partial f(x^*)$ by Theorem 5.3.5 (a), i.e., (c).

“(c) \Rightarrow (a)”: Since $0 \in \partial f(x^*)$ by assumption, the inequality (5.3.2) holds for $s = 0$ and $x_0 = x^*$, i.e., $f(y) - f(x^*) \geq 0^T(y - x^*) = 0$ for all $y \in \mathcal{F}$. Therefore, x^* is a global minimum of f in \mathcal{F} . \square

Within the standard gradient method, we have used the fact that the negative gradient is the direction of the steepest descent. Something similar can also be shown for the general case.

Definition 5.3.9 *Let $f : \mathbb{R}^n \rightarrow \mathbb{R}$ and $x \in \mathbb{R}^n$. A vector $d \in \mathbb{R}^n$ is called descent direction of f in x , if there is a $t > 0$ such that $f(x + td) < f(x)$.*

Theorem 5.3.10 *Let $f : \mathbb{R}^n \rightarrow \mathbb{R}$ be convex, $x, d \in \mathbb{R}^n$. Then, the following statements are equivalent:*

- (a) d is a descent direction of f in x ;
- (b) $f'(x, d) < 0$;
- (c) $\max_{s \in \partial f(x)} s^T d < 0$.

Proof. “(a) \Rightarrow (b)”: If d is a descent direction of f in x , there is a $t > 0$ such that $f(x + td) < f(x)$, so that by definition of the directional derivative we get

$$f'(x, d) = \lim_{\tilde{t} \rightarrow 0} \frac{f(x + \tilde{t}d) - f(x)}{\tilde{t}} \leq \frac{f(x + td) - f(x)}{t} < 0,$$

which proves (b).

“(b) \Rightarrow (a)”: By $f'(x, d) < 0$, we get

$$0 > f'(x, d) = \lim_{\tilde{t} \rightarrow 0} \frac{f(x + \tilde{t}d) - f(x)}{\tilde{t}},$$

which means that $f(x + td) - f(x) < 0$ must hold for $t > 0$ sufficiently small, i.e., (a).

“(b) \Leftrightarrow (c)” follows from Theorem 5.3.5 (b). \square

Theorem 5.3.11 (Direction of descent) *Let $f : \mathcal{F} \rightarrow \mathbb{R}$ be convex on an open and convex set $\mathcal{F} \subset \mathbb{R}^n$. Furthermore, let $x \in \mathcal{F}$ with $0 \notin \partial f(x)$ and let $s^* \in \partial f(x)$ be the solution of the constrained optimization problem*

$$\begin{cases} \min_{s \in \mathbb{R}^n} \|s\| \\ \text{s.t. } s \in \partial f(x). \end{cases} \quad (5.3.8)$$

Then, $d := -s^$ is a descent direction of f in x (on other words: the negative subgradient of minimal norm is a descent direction) and $f'(x, d) = -\|d\|^2$.*

Proof. First, we note that $s^* \neq 0$ since $0 \notin \partial f(x)$. Then, by Theorem 5.3.10 (b), we have to show that $f'(x, d) = f'(x, -s^*) < 0$.

To this end, consider the function $h : \mathbb{R}^n \rightarrow \mathbb{R}$ defined by $h(z) := \frac{1}{2}\|z\|^2$, $z \in \mathbb{R}^n$, which is differentiable and convex. Obviously, a solution s^* of (5.3.8) also minimizes the function h on the convex set $\partial f(x)$. For any $s \in \partial f(x)$ and all $t \in [0, 1]$, we have that $s^* + t(s - s^*) \in \partial f(x)$ and $h(s^* + t(s - s^*)) \geq h(s^*)$ because s^* is a minimizer. Hence,

$$(s^*)^T(s - s^*) = (\nabla h(s^*))^T(s - s^*) = \lim_{t \rightarrow 0} \frac{h(s^* + t(s - s^*)) - h(s^*)}{t} \geq 0,$$

i.e., $s^T d = -s^T s^* \leq \|s^*\|^2$ for all $s \in \partial f(x)$ with equality for $s = s^*$. By Theorem 5.3.5 (b), we thus get

$$f'(x, d) = \max_{s \in \partial f(x)} s^T d = -\|s^*\|^2 = -\|d\|^2 < 0$$

since $s^* \neq 0$, which proves the assertion. \square

For the computation of a descent direction, one therefore needs the information of the *full subdifferential*. It is needless to say that this is a challenge for the numerical realization and often out of reach. It is remarkable that the solution s^* of (5.3.8) can equivalently be characterized as the projection of $0 \notin \partial f(x)$ onto $\partial f(x)$. In fact, (5.3.8) can also be written as

$$s^* = \arg \min_{s \in \partial f(x)} \|s - 0\|,$$

i.e., s^* is the element in $\partial f(x)$, which is closest to $0 \notin \partial f(x)$ w.r.t. the chosen norm $\|\cdot\|$. With the notation introduced in (5.3.9) below, we have that $s^* = P_{\partial f(x)}(0)$.

5.3.2 The projected subgradient method

Now we have collected all ingredients required for the introduction of the *subgradient method*. As this method uses only one function value (evaluation of the objective function at one point) and one subgradient per step, it is relatively efficient (concerning complexity). However, one needs to be careful, because **not every subgradient g yields a descent direction** by $s = -g$ as the following example shows.

Example 5.3.12 Let $x = (x_1, x_2)^T$ and consider the function $f : \mathbb{R}^2 \rightarrow \mathbb{R}_0^+$ defined by $f(x) := \frac{1}{2}x_1^2 + 2|x_2|$. Obviously, the global minimum is $x^* = (0, 0)$. We consider the point $x_0 := (1, 0)^T$. For this point we have $s = (1, 2)^T \in \partial f(x_0)$, which can be seen as follows. We have $f(x_0) + s^T(y - x_0) = y_1 + 2y_2 - \frac{1}{2}$ so that $f(y) \geq f(x_0) + s^T(y - x_0)$ holds if and only if $\frac{1}{2}y_1^2 + 2|y_2| \geq y_1 + 2y_2 - \frac{1}{2}$ which is equivalent to $(y_1 - 1)^2 \geq 4(y_2 - |y_2|)$. Since the left-hand side is non-negative and the right-hand side is non-positive, this condition is always valid.

On the other hand, however, the estimate

$$f(x_0 - ts) = \frac{1}{2}(1-t)^2 + 2|0 - 2t| = \frac{1}{2}(1-t)^2 + 4t = \frac{1}{2} + \left(3t + \frac{1}{2}t^2\right) > \frac{1}{2} = f(x_0)$$

is valid for all $t > 0$, since $3t + \frac{1}{2}t^2 > 0$ holds for all $t > 0$. With that $f(x_0 - ts) > f(x_0)$ is valid for all $t > 0$, so that $d = -s$ is *not* a descent direction in x_0 . \diamond

Remark 5.3.13 Let $x^* \in \mathbb{R}^n$ be fixed and consider the mapping $h : \mathbb{R}^n \rightarrow \mathbb{R}$ defined by $h(x) := \|x - x^*\|$. If the vector $d = -s$ is a descent direction for the function h (i.e., the distance of the arguments), then $\tilde{x} := x + td$ for some $t > 0$ might be closer to x^* even if $f(\tilde{x}) > f(x)$ holds. In such a case, one might be able to decrease the distance to an optimal point even if the algorithm does not yield a decrease of the function value.

Hence, even if we cannot guarantee that a convex subgradient yields to a decrease of the cost function f , it might yield an approximation which is closer to the optimal point as compared to the previous step of the algorithm.

The function $P_{\mathcal{F}} : \mathbb{R}^n \rightarrow \mathcal{F}$ defined for an open set $\mathcal{F} \subseteq \mathbb{R}^n$ by

$$P_{\mathcal{F}}(z) := \arg \min_{y \in \mathcal{F}} \|y - z\|, \quad z \in \mathbb{R}^n, \quad (5.3.9)$$

is a projection of $z \in \mathbb{R}^n$ onto $\mathcal{F} \subset \mathbb{R}^n$ (i.e., $P_{\mathcal{F}}(x) = x$ for $x \in \mathcal{F}$). One can easily show that the function is *non-expansive*, which means that

$$\|P_{\mathcal{F}}(x) - P_{\mathcal{F}}(y)\| \leq \|x - y\| \quad \forall x, y \in \mathbb{R}^n. \quad (5.3.10)$$

Remark 5.3.14 If \mathcal{F} is a linear subspace of \mathbb{R}^n and if $\|\cdot\|$ is chosen as the Euclidean norm on \mathbb{R}^n , then $P_{\mathcal{F}}$ coincides with the orthogonal projection, i.e., $\langle P_{\mathcal{F}}x - x, f \rangle = 0$ for all $f \in \mathcal{F}$, as it is known that the best approximation in a Hilbert space onto a linear subspace is the orthogonal projection.

This projection can be used for this method in order to “correct” an iteration which may be outside of \mathcal{F} into the feasible region. For a closed set \mathcal{F} , one replaces the projection by a suitable operator for which (5.3.10) holds. With that we get a class of solution methods for the general optimization problem (5.2.1) detailed in Algorithm 5.3.1.

Algorithm 5.3.1: Projected subgradient method

Input: Feasible initial value $x^{(0)} \in \mathcal{F}$

- 1: **for** $k = 0, 1, 2, \dots$ **do**
 - 2: Determine some $s^{(k)} \in \partial f(x^{(k)})$
 - 3: **if** $s^{(k)} = 0$ or another suitable stopping criteria is satisfied **then**
 - 4: **Stop**, output: $x^{(k)}$
 - 5: **end if**
 - 6: Set $d^{(k)} := -s^{(k)} / \|s^{(k)}\|$
 - 7: Choose a suitable step size $\sigma_k > 0$
 - 8: Set $x^{(k+1)} := P_{\mathcal{F}}(x^{(k)} + \sigma_k d^{(k)})$
 - 9: **end for**
-

Now, we are going to analyze the convergence of the projected subgradient method. By Remark 5.3.13 and Example 5.3.12 it is senseless to try to prove “ $f(x^{(k+1)}) < f(x^{(k)})$ ”. Therefore, the goal is to choose a step size $\sigma_k > 0$ in line 7 of Algorithm 5.3.1 so that

$$\|x^{(k+1)} - x^*\| < \|x^{(k)} - x^*\|$$

holds for all $k \in \mathbb{N}_0$. So the iteration regarding the searched minimum point is monotonic at least. Indeed this is possible.

So far, we have not specified the norm $\|\cdot\|$ we want to use. As we recall from Linear Algebra, all norms on \mathbb{R}^n are equivalent. Therefore, **from now on we use the Euclidean norm**, which means $\|x\|^2 := \|x\|_2^2 = x^T x = \langle x, x \rangle$ for $x \in \mathbb{R}^n$. This is convenient for the proof of the following convergence theorem.

Theorem 5.3.15 (Convergence of the projected subgradient method) Let x^* be a solution of (5.2.1), $x \in \mathcal{F}$ and $s \in \partial f(x)$, $s \neq 0$, $d := -\frac{s}{\|s\|}$. Then,

$$\|P_{\mathcal{F}}(x + \sigma d) - x^*\|^2 \leq \|x - x^*\|^2 - \rho(\sigma) \quad (5.3.11)$$

for all step sizes σ with $0 < \sigma < \frac{2}{\|s\|}(f(x) - f(x^*))$ and the reduction factor

$$\rho(\sigma) := 2\sigma \frac{f(x) - f(x^*)}{\|s\|} - \sigma^2 > 0.$$

Proof. Using the properties of Euclidean norm $\|\cdot\| \equiv \|\cdot\|_2$ and (5.3.2) for $s \in \partial f(x)$, we obtain

$$\begin{aligned} \left\| x - \frac{\sigma}{\|s\|} s - x^* \right\|^2 &= \|x - x^*\|^2 - 2 \frac{\sigma}{\|s\|} s^T (x - x^*) + \sigma^2 = \|x - x^*\|^2 + 2 \frac{\sigma}{\|s\|} s^T (x^* - x) + \sigma^2 \\ &\leq \|x - x^*\|^2 + 2\sigma \frac{f(x^*) - f(x)}{\|s\|} + \sigma^2 = \|x - x^*\|^2 - \rho(\sigma). \end{aligned}$$

Note, that $\rho(\sigma) > 0$ is valid for all $0 < \sigma < \frac{2}{\|s\|}(f(x) - f(x^*))$ since x^* is a minimizer of f . Furthermore, with (5.3.10) holds

$$\left\| P_{\mathcal{F}} \left(x - \sigma \frac{s}{\|s\|} \right) - x^* \right\| = \left\| P_{\mathcal{F}} \left(x - \sigma \frac{s}{\|s\|} \right) - P_{\mathcal{F}}(x^*) \right\| \leq \left\| x - \sigma \frac{s}{\|s\|} - x^* \right\|$$

which completes the proof. \square

Choice of the step size

Theorem 5.3.15 only provides us with a *possible range* for the step size. It is needless to say that one tries to determine the step size as good as possible regarding the convergence speed as well as regarding efficiency. For that, we consider the error reduction (5.3.11) in Theorem 5.3.15, which would be optimal for $x^{(k)}$, if

$$\rho_k(\sigma) := 2\sigma \frac{f(x^{(k)}) - f(x^*)}{\|s^{(k)}\|} - \sigma^2 \longrightarrow \text{max!}$$

The derivative w.r.t. σ can easily be determined as

$$\rho'_k(\sigma) = 2 \frac{f(x^{(k)}) - f(x^*)}{\|s^{(k)}\|} - 2\sigma.$$

Hence, the optimal step size is the root of this, namely

$$\sigma_k = \frac{f(x^{(k)}) - f(x^*)}{\|s^{(k)}\|}. \quad (5.3.12)$$

However, for this, the optimal value $f(x^*)$ of the objective function would need to be known.

If the optimal value $f(x^*)$ is *not* known, a number of heuristic methods are known to choose some sub-optimal step size σ_k . To this end, one often assumes that the following two properties are valid:

$$\sigma_k \rightarrow 0, \quad (k \rightarrow \infty), \quad (5.3.13a)$$

$$\sum_{k=0}^{\infty} \sigma_k = \infty. \quad (5.3.13b)$$

At a first glance, the second condition seems surprising as the following consideration shows: Using a telescopic sum yields

$$\|x^{(k)} - x^{(0)}\| \leq \sum_{\ell=0}^{k-1} \|x^{(\ell+1)} - x^{(\ell)}\| \leq \sum_{\ell=0}^{k-1} \sigma_\ell.$$

Hence, if the sequence $(\sigma_k)_{k \in \mathbb{N}_0}$ would be summable (i.e., in $\ell_1(\mathbb{N}_0)$), then $\sum_{\ell=0}^{\infty} \sigma_\ell = M < \infty$ and

$$\|x^{(k)} - x^{(0)}\| \leq M$$

would hold for all $k \in \mathbb{N}_0$. However, this would imply that $x^{(k)} \in B_M(x^{(0)})$, the iteration would not leave a ball around the initial guess and the minimum might not be reachable. In particular this would be the case if $x^* \notin B_M(x^{(0)})$. As a consequence, one would need a quite good initial value. However, as will see, the assumptions in (5.3.13) will allow us to guarantee convergence.

Theorem 5.3.16 Let $\mathcal{U} \subset \mathbb{R}^n$ be open, convex and let the function $f : \mathcal{U} \rightarrow \mathbb{R}$ be convex. Furthermore, let $\mathcal{F} \subset \mathcal{U}$ be closed and convex and let $x^* \in \mathcal{F}$ be a solution of (5.2.1). Moreover, (5.3.13) is assumed for Algorithm 5.3.1 with step sizes σ_k .

Then, either the algorithm terminates after finitely many steps with x^* or it generates a sequence $(x^{(k)})_{k \in \mathbb{N}_0}$ with

1. For the function values holds

$$\liminf_{k \rightarrow \infty} f(x^{(k)}) = f(x^*) \quad (5.3.14)$$

2. If

$$\lim_{\substack{x \in \mathcal{F} \\ \|x\| \rightarrow \infty}} f(x) = \infty,$$

diverges, there exists a subsequence of $(x^{(k)})_{k \in \mathbb{N}_0}$ which converges to the global minimum x^* of f (this condition is equivalent to the compactness of the set of global minima).

Proof. If the method terminates, there is nothing more to prove. Hence, let the sequence of the iterations $(x^{(k)})_{k \in \mathbb{N}_0}$ be infinite.

Let us assume that (5.3.14) does not hold. Then, there exists a $\varepsilon > 0$ and a $K > 0$ such that $f(x^{(k)}) > f(x^*) + \varepsilon$ for all $k \geq K$. The set

$$\mathcal{M} := \{x \in U : f(x) < f(x^*) + \varepsilon\}$$

is open and it holds that $x^{(k)} \notin \mathcal{M}$ for all $k \geq K$. Now, choose a $\delta > 0$ with $\overline{B_\delta(x^*)} \subset \mathcal{M}$ and set $y^{(k)} := x^* - \delta d^{(k)}$. Then, $\|y^{(k)} - x^*\| = \delta \|d^{(k)}\| = \delta$, i.e., $y^{(k)} \in \overline{B_\delta(x^*)}$ holds and for all $k \geq K$ it follows:

$$\begin{aligned} \|x^{(k+1)} - x^*\|^2 - \|x^{(k)} - x^*\|^2 &= \|P_{\mathcal{F}}(x^{(k)} + \sigma_k d^{(k)}) - x^*\|^2 - \|x^{(k)} - x^*\|^2 \\ &= \|P_{\mathcal{F}}(x^{(k)} + \sigma_k d^{(k)}) - P_{\mathcal{F}}(x^*)\|^2 - \|x^{(k)} - x^*\|^2 \\ &\leq \|x^{(k)} - x^* + \sigma_k d^{(k)}\|^2 - \|x^{(k)} - x^*\|^2, \end{aligned}$$

as $x^* = P_{\mathcal{F}}(x^*)$ and by (5.3.10). Now, we use the representation of the Euclidean norm in terms of the scalar product and get

$$\|x^{(k)} - x^* + \sigma_k d^{(k)}\|^2 = \|x^{(k)} - x^*\|^2 + \sigma_k \|d^{(k)}\|^2 + 2\sigma_k (d^{(k)})^T (x^{(k)} - x^*).$$

Since $\|d^{(k)}\| = 1$, $y^{(k)} - x^* = -\delta d^{(k)}$ and $d^{(k)} = -s^{(k)} / \|s^{(k)}\|$ we proceed by adding and subtracting $y^{(k)}$

$$\begin{aligned} \|x^{(k+1)} - x^*\|^2 - \|x^{(k)} - x^*\|^2 &\leq \sigma_k^2 + 2\sigma_k (d^{(k)})^T (x^{(k)} - x^*) \\ &= \sigma_k^2 + 2\sigma_k (d^{(k)})^T (y^{(k)} - x^*) + 2\sigma_k (d^{(k)})^T (x^{(k)} - y^{(k)}) \\ &= \sigma_k^2 - 2\sigma_k \underbrace{\delta (d^{(k)})^T d^{(k)}}_{=1} - 2 \frac{\sigma_k}{\|s^{(k)}\|} (s^{(k)})^T (x^{(k)} - y^{(k)}) \\ &= \sigma_k^2 - 2\delta\sigma_k + 2 \frac{\sigma_k}{\|s^{(k)}\|} (s^{(k)})^T (y^{(k)} - x^{(k)}) \\ &\leq \sigma_k^2 - 2\delta\sigma_k + 2 \frac{\sigma_k}{\|s^{(k)}\|} (f(y^{(k)}) - f(x^{(k)})), \end{aligned}$$

where we used (5.3.2) and the fact that $s^{(k)} \in \partial f(x^{(k)})$. Because $y^{(k)} \in \mathcal{M}$ and $x^{(k)} \notin \mathcal{M}$, we have $f(y^{(k)}) < f(x^*) + \varepsilon$ and $f(x^{(k)}) \geq f(x^*) + \varepsilon$, we therefore get $f(y^{(k)}) - f(x^{(k)}) < 0$ and

$$\|x^{(k+1)} - x^*\|^2 - \|x^{(k)} - x^*\|^2 < \sigma_k^2 - 2\delta\sigma_k.$$

Due to (5.3.13a) we have $\sigma_k \rightarrow 0$ with $k \rightarrow \infty$. Hence, we choose K sufficiently large so that $\sigma_k \leq \delta$ for all $k \geq K$ and thus

$$\sigma_k^2 - 2\delta\sigma_k = \sigma_k(\sigma_k - \delta) - \delta\sigma_k \leq -\delta\sigma_k, \quad \text{for all } k \geq K.$$

With that, we get by (5.3.13b) for all $\ell \geq 0$

$$\begin{aligned} \|x^{(K)} - x^*\|^2 &\geq \|x^{(K+1)} - x^*\|^2 + \delta\sigma_K \geq \|x^{(K+2)} - x^*\|^2 + \delta\sigma_K + \delta\sigma_{K+1} \geq \dots \geq \\ &\geq \underbrace{\|x^{(K+\ell)} - x^*\|^2}_{\geq 0} + \delta \sum_{k=K}^{K+\ell-1} \sigma_k \\ &\geq \delta \sum_{k=K}^{K+\ell-1} \sigma_k \xrightarrow{\ell \rightarrow \infty} \infty. \end{aligned}$$

But this is a contradiction to Theorem 5.3.15 and we have thus proven (5.3.14).

In view of (5.3.14), there is a subsequence $(x^{(k')})_{k' \in \mathbb{N}}$ of $(x^{(k)})_k$ with $f(x^{(k')}) \rightarrow f(x^*)$ as $k' \rightarrow \infty$. Because of the assumptions concerning f and U , this sequence is bounded and therefore contains a convergent subsequence $(x^{(k'')})_{k'' \in \mathbb{N}}$. This means that $x^{(k'')} \rightarrow \tilde{x}$ as $k'' \rightarrow \infty$, with a limit \tilde{x} . As the function f is continuous by assumption, we get that

$$f(\tilde{x}) = \lim_{k'' \rightarrow \infty} f(x^{(k'')}) = f(x^*)$$

which also proves the second part of the statement. \square

Convergence rate

The above statement ensures the convergence of a subsequence. It does not say anything about the convergence *rate*. As we will see now, even in simple cases, the convergence rate is not what we would like to have.

Theorem 5.3.17 *For $\mathcal{F} = \mathbb{R}^n$ (hence $P_{\mathcal{F}} = I$), let the assumption of Theorem 5.3.16 be satisfied and let the sequence $(x^{(k)})_{k \in \mathbb{N}}$ converge towards x^* . Then, for all $\gamma \in (0, 1)$ it holds that*

$$\limsup_{k \rightarrow \infty} \frac{1}{\gamma^k} \|x^{(k)} - x^*\| = \infty. \quad (5.3.15)$$

Obviously, (5.3.15) implies that we cannot even expect linear convergence. Before we are going to prove this theorem, we will repeat some convergence concepts in order to classify the statements of Theorem 5.3.17.

Definition 5.3.18 *A sequence $(x^{(k)})_{k \in \mathbb{N}}$ converges R-linear to x^* , if there exist $C > 0$ and $q \in (0, 1)$ such that*

$$\|x^{(k)} - x^*\| \leq Cq^k$$

for all $k \in \mathbb{N}$.

Hence, the statement in (5.3.15) says that the convergence of the scheme is not even R-linear. Let us compare that with the convergence concept that has been introduced *Numerical Linear Algebra*. There, we defined convergence order $p > 1$ if there exists a constant $C < \infty$ such that

$$\|x^{(k)} - x^*\| \leq C\|x^{(k-1)} - x^*\|^p$$

(and for linear convergence $p = 1$, we require in addition that $C < 1$). This can be rewritten as follows.

Definition 5.3.19 *A sequence $(x^{(k)})_{k \in \mathbb{N}}$ Q-converges to x^* of order $p \geq 1$, if*

$$\limsup_{k \rightarrow \infty} \frac{\|x^{(k+1)} - x^*\|}{\|x^{(k)} - x^*\|^p} < \begin{cases} \infty, & \text{if } p > 1, \\ 1 & \text{if } p = 1. \end{cases}$$

for all $k \in \mathbb{N}$.

The notation “Q” stands for “quotient”, whereas “R” stands for “root”. This makes sense since Definition 5.3.19 can also be written as

$$\limsup_{k \rightarrow \infty} \sqrt[k]{\|x^{(k)} - x^*\|} < 1,$$

i.e., a condition to the k -th root. It can be shown that Q -linear convergence implies R -linear convergence. Therefore the R -convergence is in fact weaker than what we have seen earlier. However, (5.3.15) means that it is not reached by Algorithm 5.3.1. Let us now prove the above theorem.

Proof of Theorem 5.3.17. Let us assume (5.3.15) would be wrong. Then, there exist $K > 0$, $C > 0$ and $\gamma \in (0, 1)$ with $\|x^{(k)} - x^*\| \leq C\gamma^k$ for all $k \geq K$. Hence, for all $k \geq K$ we get by using $P_{\mathcal{F}} = I$ and triangle inequality that

$$\sigma_k = \|x^{(k+1)} - x^{(k)}\| \leq \|x^{(k+1)} - x^*\| + \|x^{(k)} - x^*\| \leq C(\gamma^{k+1} + \gamma^k) \leq 2C\gamma^k,$$

since $\gamma < 1$. With that and taking (5.3.13b) as well as the geometrical series into account we obtain

$$\infty = \sum_{k=K}^{\infty} \sigma_k \leq 2C \sum_{k=K}^{\infty} \gamma^k = 2C \frac{\gamma^K}{1-\gamma} < \infty$$

thus a contradiction. This completes the proof. \square

In both Theorems 5.3.16 and 5.3.17 we have not used any assumption on the optimal value of the objective function $f(x^*)$. However, if one knows this optimal value, then we can in fact derive convergence results. Moreover, in that case, one can in fact identify the optimal step size as above in (5.3.12).

Theorem 5.3.20 *Let the assumptions of Theorem 5.3.16 concerning f, U, \mathcal{F} and x^* be fulfilled. Moreover, let the step sizes σ_k be chosen according to (5.3.12) and we use the stopping criterium*

$$|f(x^{(k)}) - f(x^*)| \leq tol.$$

Then, Algorithm 5.3.1 either terminates after a finite number of steps or the following statements hold

- (a) *the sequence $(\|s^{(k)}\|)_{k \in \mathbb{N}}$ is bounded;*
- (b) $\lim_{k \rightarrow \infty} f(x^{(k)}) = f(x^*)$;
- (c) *the sequence $(x^{(k)})_{k \in \mathbb{N}}$ converges to the solution of (5.2.1).*

Proof. With the choice (5.3.12) of σ_k , we get

$$\rho_k(\sigma_k) = \frac{(f(x^{(k)}) - f(x^*))^2}{\|s^{(k)}\|^2}$$

and with (5.3.11) we obtain

$$\begin{aligned} \|x^{(k+1)} - x^*\|^2 &\leq \|x^{(k)} - x^*\|^2 - \rho_k(\sigma_k) \leq \|x^{(k-1)} - x^*\|^2 - \rho_{k-1}(\sigma_{k-1}) - \rho_k(\sigma_k) \\ &\leq \dots \leq \underbrace{\|x^{(0)} - x^*\|^2}_{=: R^2} - \sum_{\ell=0}^k \rho_\ell(\sigma_\ell) = R^2 - \sum_{\ell=0}^k \rho_\ell(\sigma_\ell) < R^2. \end{aligned}$$

Therefore, $x^{(k)} \in \mathcal{K} := \overline{B_R(x^*)} \cap \mathcal{F}$. As $\mathcal{K} \subset \mathcal{F}$ is compact and f is locally Lipschitz continuous on \mathcal{F} , the function f is Lipschitz continuous on \mathcal{K} as well. By Theorem 5.3.5 we get that $\partial f(x) \subset \overline{B_L(0)^2}$, so that $\|s^{(k)}\| \leq L$ for all $k \in \mathbb{N}$, yielding (a).

²This can be seen as follows: For $s \in \partial f(x)$, we know that $s^T g \leq f'(x, g)$ for all $g \in \mathbb{R}^n$. Moreover, since f is locally Lipschitz continuous, we have that $|f'(x, g)| \leq L\|g\|$, so that $s^T g \leq L\|g\|$. Choosing $g = s$ yields the claim.

Furthermore, it holds

$$0 \leq \|x^{(k+1)} - x^*\|^2 \leq R^2 - \sum_{\ell=0}^k \underbrace{\rho_\ell(\sigma_\ell)}_{\geq 0}. \quad (5.3.16)$$

This means that $(\rho_\ell(\sigma_\ell))_{\ell \in \mathbb{N}}$ needs to be a null sequence. With that, we conclude that

$$0 < f(x^{(k)}) - f(x^*) = \underbrace{\sqrt{\rho_k(\sigma_k)}}_{\substack{\rightarrow 0 \\ k \rightarrow \infty}} \cdot \underbrace{\|s^{(k)}\|}_{\leq L} \xrightarrow{k \rightarrow \infty} 0,$$

which yields (b).

In order to prove (c), note, that $(x^{(k)})_{k \in \mathbb{N}} \subset \mathcal{K}$ and \mathcal{K} is compact. Hence, there exists a subsequence $(x^{(k')})_{k' \in \mathbb{N}}$ with a limit $\tilde{x} \in \mathcal{K}$. As the function f is continuous, we get by (b) that $f(\tilde{x}) = f(x^*)$. Hence, \tilde{x} (and thus each accumulation point of the sequence $(x^{(k)})_{k \in \mathbb{N}}$) is an optimal solution. Moreover, x^* is such an accumulation point as well.

Because of the choice of σ_k , by (5.3.11), the sequence $(\|x^{(k)} - x^*\|)_{k \in \mathbb{N}}$ is monotonically decreasing and bounded from below, therefore there exists a limit

$$0 \leq \alpha := \lim_{k \rightarrow \infty} \|x^{(k)} - x^*\|.$$

Let \tilde{x} be an additional accumulation point, then (recall that x^* is an accumulation point) $\|\tilde{x} - x^*\| = \alpha = \|x^* - x^*\| = 0$. Hence, $\alpha = 0$ as well as $\tilde{x} = x^*$ is valid and therefore (c) holds. \square

Now, we consider the convergence rate in the case of a known optimal value.

Theorem 5.3.21 *Under the assumptions of Theorem 5.3.20, we have*

- (a) $\liminf_{k \rightarrow \infty} \sqrt{k}(f(x^{(k)}) - f(x^*)) = 0$.
- (b) *If there exists a $\gamma > 0$ with $f(x^{(k)}) - f(x^*) \geq \gamma \|x^{(k)} - x^*\|$ for all $k \in \mathbb{N}$, then the sequence $(x^{(k)})_{k \in \mathbb{N}}$ converges (Q -)linear to x^* , more precisely*

$$\|x^{(k+1)} - x^*\| \leq \alpha \|x^{(k)} - x^*\| \quad \text{with } \alpha = \sqrt{1 - \frac{\gamma^2}{C^2}} \text{ and } C = \sup_{k \in \mathbb{N}} \|s^{(k)}\|.$$

Proof. Let us assume that (a) does not hold. Then, there exists a $\varepsilon > 0$ and a $K > 0$ with $\sqrt{k}(f(x^{(k)}) - f(x^*)) \geq \varepsilon$ for all $k \geq K$. Then, with (5.3.16) we have that

$$\begin{aligned} R^2 &\geq \sum_{k=0}^{\infty} \rho_k(\sigma_k) = \sum_{k=0}^{\infty} \frac{(f(x^{(k)}) - f(x^*))^2}{\|s^{(k)}\|^2} \\ &\geq \frac{1}{C^2} \sum_{k=0}^{\infty} (f(x^{(k)}) - f(x^*))^2 \geq \frac{1}{C^2} \sum_{k=K}^{\infty} \frac{\varepsilon^2}{k} = \infty \end{aligned}$$

in view of the harmonic series. Hence, we get a contradiction yielding (a).

Concerning (b): With (5.3.11) it holds that

$$\begin{aligned} \|x^{(k+1)} - x^*\|^2 &\leq \|x^{(k)} - x^*\|^2 - \rho_k(\sigma_k) = \|x^{(k)} - x^*\|^2 - \frac{(f(x^{(k)}) - f(x^*))^2}{\|s^{(k)}\|^2} \\ &\leq \|x^{(k)} - x^*\|^2 - \frac{\gamma^2 \|x^{(k)} - x^*\|^2}{C^2} = \left(1 - \frac{\gamma^2}{C^2}\right) \|x^{(k)} - x^*\|^2, \end{aligned}$$

and therefore (b). \square

MATLAB-function: projected_subgradient_method.m

```

1  %% Projected Subgradient method function
2  function [x, f_val, allx, iter] = projected_subgradient_method(f,
3      subgrad_f, proj, x0, ...
4                      max_iter, step_size_rule, tolerance, ...
5                      outflag)
6
7  x = x0;
8  allx=[x];
9  d = size(x,1);
10
11 for k = 1:max_iter
12     subgrad = subgrad_f(x);
13     g = subgrad/norm(subgrad);
14     alpha = step_size_rule(k);
15
16     % Check for convergence
17     if (norm(subgrad) < tolerance) || (alpha < tolerance)
18         break;
19     end
20
21     x = x - alpha * g;      % update
22     x = proj(x);           % project onto the feasible set
23
24     allx = [allx,x];
25     iter = k;
26     f_val = f(x);
27
28     if (outflag==1)
29         % Display progress (optional)
30         fprintf('Iteration %i: , x=[' , k);
31         for i=1:d-1
32             fprintf('%.4f, ' , x(i));
33         end
34         fprintf('%.4f], ' , x(d));
35         fprintf('f(x) = %.4f, ||Df(x)|| = %.4f, alpha = %.4f\n' , ...
36                  f_val, norm(subgrad), alpha);
37
38     end
39
40 end
41
42 end

```

Example 5.3.22 We consider the simple example $f(x) = |x|$ and choose the step size sequence as $\sigma_k = \frac{2}{k+1}$, which satisfies the above conditions (5.3.13). We choose $x^{(0)} = 3$. The result shown in Figure 5.3 shows that the sequence is not monotonically decreasing, but that there is subsequence which does so. Moreover, the speed of convergence is very slow. More details on this example can be found in [2, Bsp. 4.2.2]. \diamond

Example 5.3.23 Next, we show the (delicate) dependence on the sequence σ_k and also on the tolerance for the stopping criterium. To this end, we consider the Wolfe function introduced in Example 5.1.1. We choose $x^{(0)} = (5, 4)^T \in S$. The following table shows some results:

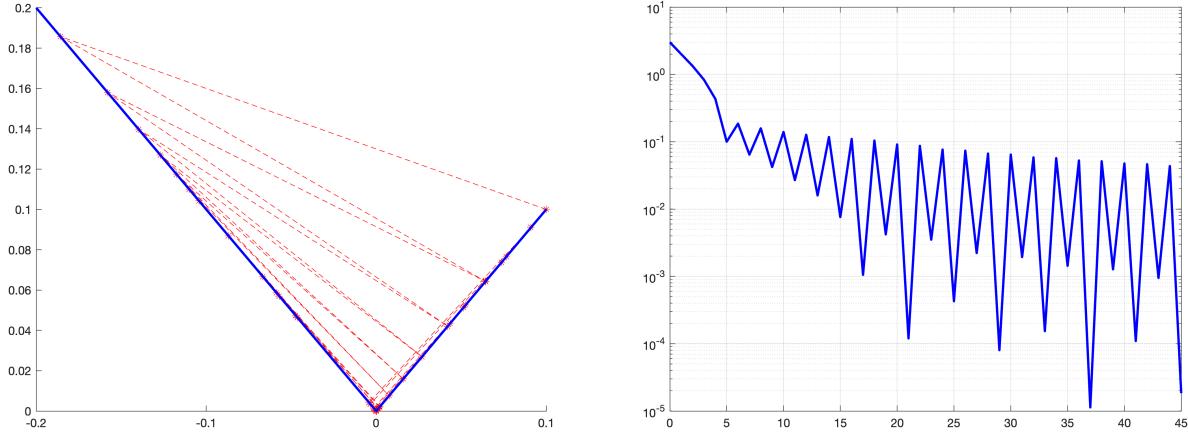


Figure 5.3: Subgradient method for $f(x) = |x|$: iteration history (left; the red crosses mark the iterations connected by red dashed lines) and convergence history (right, function value over iteration in semilogarithmic scale).

σ_k	tol	k_{final}	$x^{(k_{\text{final}})}$	$f(x^{(k_{\text{final}})})$	$\ d^{(k_{\text{final}})}\ $
$\frac{3}{k+1}$	$1e - 1$	30	(-0.8511, 0.0012)	-5.7194	17.2783
$\frac{3}{k+1}$	$1e - 2$	300	(-1.0000, 0.0000)	-7.8405	16.0000
$\frac{3}{k+1}$	$1e - 3$	3000	(-1.0000, 0.0000)	-7.9849	16.0000
$\frac{2}{k+1}$	$1e - 2$	200	(-0.9891, 0.0000)	-7.8357	16.0177
$\frac{1}{k+1}$	$1e - 2$	100	(2.5879, 1.5745)	50.1541	17.1597

In particular, we observe that if σ_k is too small, the scheme does not converge. \diamond

Example 5.3.24 We show the convergence history for the function defined in Example 5.3.12. Again, we see the dependence of the step size sequence σ_k and the non-monotone convergence. \diamond

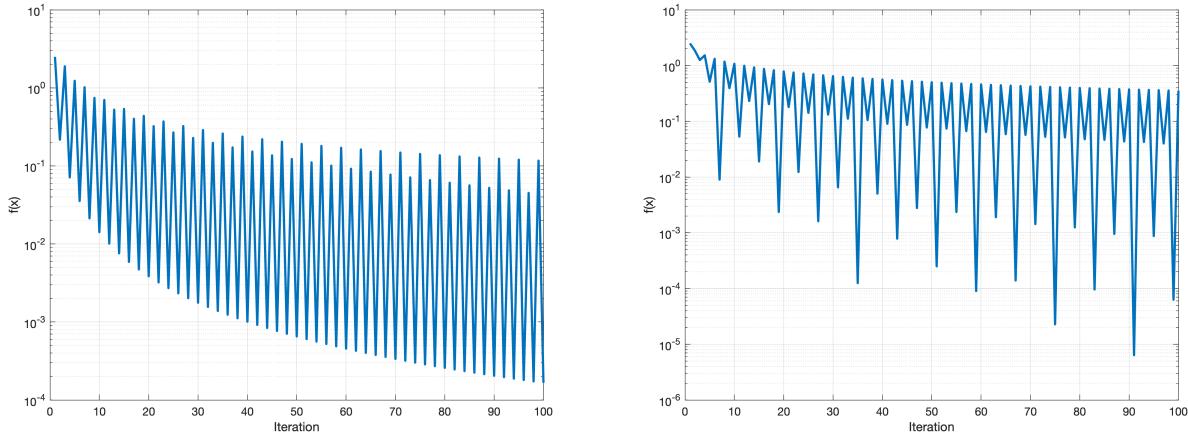


Figure 5.4: Iteration history for the subgradient method for $f(x) := \frac{1}{2}x_1^2 + 2|x_2|$ for $\sigma_k = \frac{4}{k+1}$ (left) and $\sigma_k = \frac{1}{k+1}$ (right) in semilogarithmic scale.

Example 5.3.25 Consider the Wolfe function in Example 5.1.1. Again, we see the dependence of the step size sequence σ_k and the non-monotone convergence. \diamond

Example 5.3.26 Next, we consider the maximum of two convex functions, i.e., $f(x) :=$

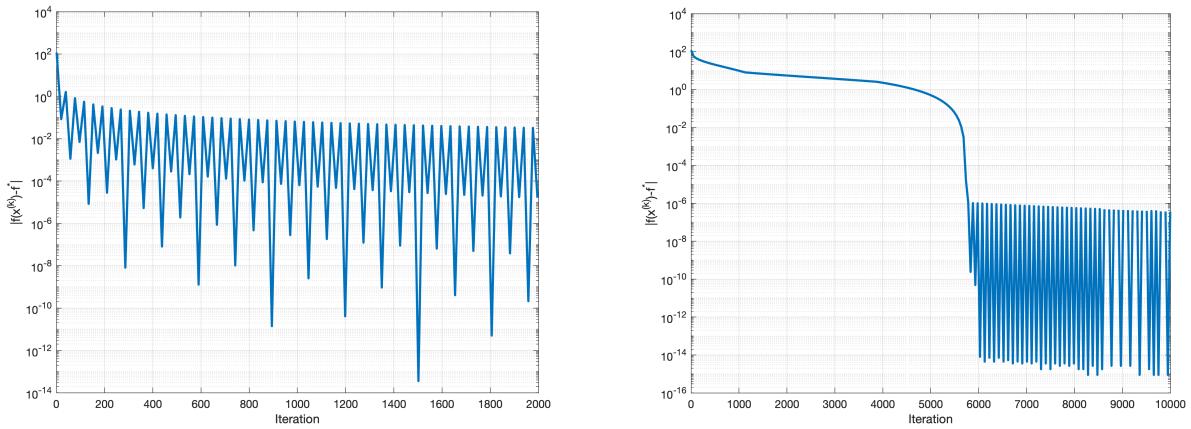


Figure 5.5: Iteration history for the subgradient method for the Wolfe function and $\sigma_k = \frac{4}{k+1}$ (left) and $\sigma_k = \frac{1}{k+1}$ (right) in semilogarithmic scale.

$\max\{f_1(x), f_2(x)\}$, whose subgradient reads

$$\partial f(x) = \begin{cases} f'_1(x) & \text{if } f_1(x) > f_2(x), \\ f'_2(x) & \text{if } f_2(x) > f_1(x), \\ g(x) & \text{if } f_2(x) = f_1(x), \end{cases}$$

where g is any function being a lower bound of f , e.g. $g(x) := \frac{1}{2}(f'_1(x) + f'_2(x))$. We show numerical examples for $f_1(x) = (x-2)^2$ and $f_2(x) = x^3$ yielding a unique minimum at $x^* = 1$ with optimal value $f(x^*) = 1$. We use the initial point $x^{(0)} = 1.2$ and test different strategies for the step size sequence σ_k . The results are shown in Figure 5.6. \diamond

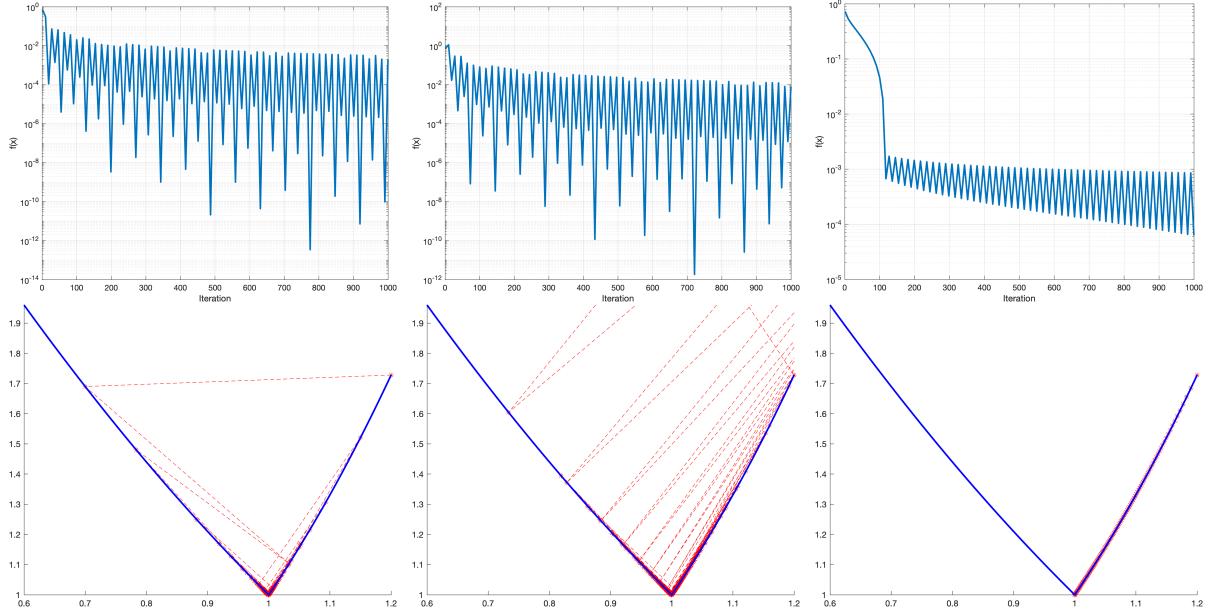


Figure 5.6: Iteration history for the subgradient method for $f(x) := \max\{f_1(x), f_2(x)\}$ and $\sigma_k = \frac{4}{k+1}$ (left), $\sigma_k = \frac{1}{k+1}$ (center) and $\sigma_k = \frac{0.01}{\sqrt{k}}$ (right).

Example 5.3.27 Next, we investigate the influence of the projection – all previous examples were without the projection $P_{\mathcal{F}}$. Recall also Remark 5.3.13. We consider the minimization of $f(x) := \|x - b\|_2^2$

for some given $b \in \mathbb{R}^d$ and also allow for various space dimensions d . We use $\mathcal{F} = B_0(1) = \{x \in \mathbb{R}^d : \|x\|_2 \leq 1\}$, i.e., the unit ball as the feasible set, so that the solution in \mathcal{F} is given by $x^* = P_{\mathcal{F}}(b)$. The results are shown in Figure 5.7, where we clearly see that the projection significantly improves the convergence (note the different ranges on the vertical axis). \diamond

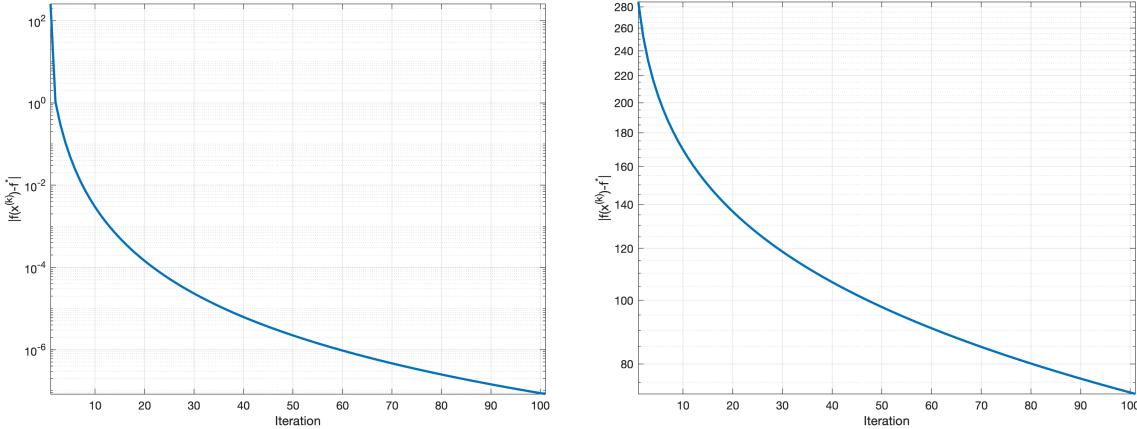


Figure 5.7: Iteration history for the subgradient method for $f(x) := \|x - b\|_2^2$ and $\sigma_k = \frac{2}{k+1}$ in $d = 10$ space dimensions with projection (left) and without (right).

Example 5.3.28 Consider the minimization of the *hinge loss function* used in support vector machines (SVMs):

$$f(w) := \sum_{i=1}^m \max(0, 1 - y_i w^T x_i)$$

subject to $w \in C$, where $C = \{w \in \mathbb{R}^n : \|w\|_2 \leq \lambda\} = B_0(\lambda)$, with λ being a scaling parameter for the weights. Here $X \in \mathbb{R}^{s \times d}$ is a data matrix of $s \in \mathbb{N}$ samples and $d \in \mathbb{N}$ features and $y \in \{0, 1\}^s$ is a binary label vector. The desired unknown $w \in \mathbb{R}^d$ is the weight vector assigning the data to the labels. Apparently, the function is non-smooth. Its subgradient for each data point (x_i, y_i) reads

$$g_i(w) = \begin{cases} -y_i x_i, & \text{if } y_i w^T x_i < 1, \\ 0, & \text{otherwise.} \end{cases}$$

Within the projected subgradient method, after each update, the current feature vector w is projected onto the feasible set C , which is the Euclidean ball with radius λ . This problem is a classic example in machine learning optimization, where we use a subgradient method to optimize a non-smooth loss function like the hinge loss, constrained to a ball C . The reason for using the projected subgradient method is that it rapidly reduces an initial error as we can see in Figure 5.8 on the left. The converge for the following iterations is rather slow (see Figure 5.8 on the right). However, when minimizing a loss function in machine learning, high accuracy is not required due to model and data errors, which explains the common use of the projected subgradient method.

We shall later also consider the regularized version of the hinge loss, i.e., $\tilde{f}(w) := \frac{\tau}{2} \|w\|_2^2 + \mu f(w)$ with regularization (or weightening) factors $\tau, \mu \geq 0$ controlling the tradeoff between the margin and the hinge loss. \diamond

Remark 5.3.29 Similar to Quasi-Newton methods one may also replace an exact subgradient by approximations. We refer to [2, §5] for details.

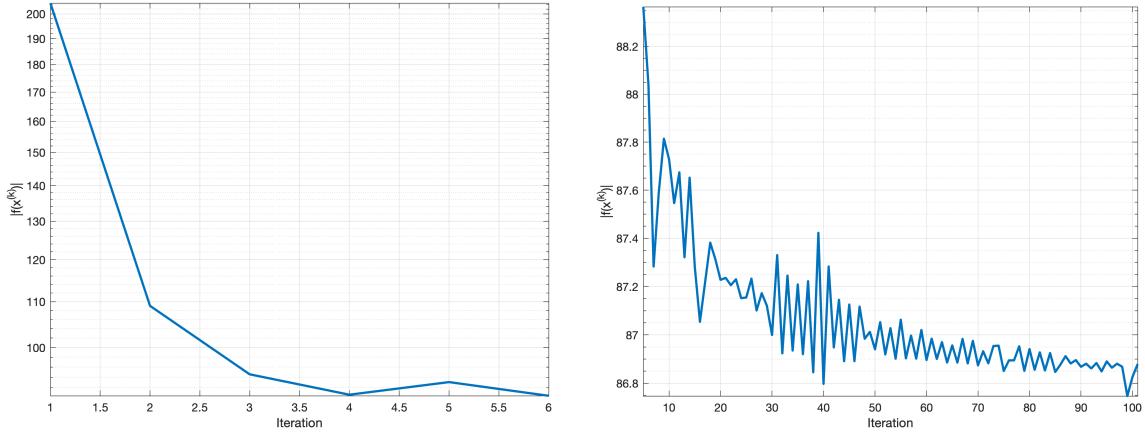


Figure 5.8: Iteration history for the subgradient method for the hinge loss function and $\sigma_k = \frac{1}{\sqrt{k}}$ for $d = 5$ and $s = 100$: iterations 1-5 (left) and iterations 5-100 (right).

5.4 THE CUTTING PLANE METHOD

Apparently, the availability of a formula or software for a subgradient is restricted to academic examples in which we can compute a subgradient by hands or by a computer algebra system. This is in general not realistic for more challenging problems. Hence, we need an approach to determine a good approximation of an element of the subdifferential. This is usually done by constructing a *bundle* of functions.

Another approach towards cutting plane methods is to improve the performance of the subgradient method by replacing one single subgradient by a bundle of them. One example is a bundle of cutting planes, which are formed by subgradients. The arising scheme is also called a *bundle method*.

The idea of the cutting plane method can be summarized in the following three points:

1. For $(x^{(k)}, s^{(k)}) \in \mathcal{F} \times \partial f(x^{(k)})$ we have

$$f(x) \geq \ell_k(x) := f(x^{(k)}) + (s^{(k)})^T(x - x^{(k)}),$$

i.e., we get a lower bound in terms of a linear function ℓ_k . Since $f(x^{(k)}) = \ell_k(x^{(k)})$, the function ℓ_k is called a “cutting plane”.

2. If we use the maximum of the cutting planes with respect to all previous iterations, i.e.,

$$f_k^{\text{CP}}(x) := \max_{0 \leq j \leq k} \ell_j(x) = \max_{0 \leq j \leq k} \{f(x^{(j)}) + (s^{(j)})^T(x - x^{(j)})\},$$

we obtain $f(x) \geq f_k^{\text{CP}}(x)$ and it is obvious that the addition of any new cutting plane to such a bundle improves the model of f .

3. Then, for the next iteration, set

$$x^{(k+1)} := \arg \min_{x \in \mathcal{F}} f_k^{\text{CP}}(x). \quad (5.4.1)$$

Of course, it heavily depends on the structure of \mathcal{F} if the non-smooth problem (5.4.1) is actually easier to solve than to the original problem. This is in general *not* obvious, even less a priori. Therefore, we will transform (5.4.1) to a smooth problem.

Lemma 5.4.1 *Let $\mathcal{F} \subset \mathbb{R}^n$ and $f_i : \mathcal{F} \rightarrow \mathbb{R}$, $1 \leq i \leq m$ and consider the problem*

$$\min_{x \in \mathcal{F}} f(x) \quad \text{where } f(x) := \max_{1 \leq i \leq m} f_i(x) \quad (5.4.2)$$

as well as

$$\begin{cases} \min_{\alpha \in \mathbb{R}} \alpha \\ \text{s.t. } f_i(x) \leq \alpha, \forall 1 \leq i \leq m, \forall x \in \mathcal{F}. \end{cases} \quad (5.4.3)$$

Then,

- (a) If x^* is a global solution of (5.4.2), then $(x^*, f(x^*))$ is a global solution of (5.4.3).
- (b) If (x^*, α^*) is a global solution of (5.4.3), then x^* is a global solution of (5.4.2) and $f(x^*) = \alpha^*$.

Proof. As for (a), let x^* be a global solution of (5.4.2), then set $\alpha^* := f(x^*)$ and the point (x^*, α^*) is feasible for (5.4.3). Now, let (x, α) be an arbitrary feasible point for (5.4.3). Then, by (5.4.2),

$$\alpha^* = f(x^*) \leq f(x) = \max_{1 \leq i \leq m} f_i(x) \leq \alpha.$$

Thus, $(x^*, f(x^*))$ is an optimal solution of (5.4.3).

In order to prove (b), let α^* be a global solution of (5.4.3) and choose x^* as $\alpha^* = f(x^*)$. Now, let $x \in \mathcal{F}$ be arbitrary. Then, since α^* is minimal, we have that $f(x^*) = \alpha^* \leq f(x)$, so that x^* is an optimal solution of (5.4.2). \square

With Lemma 5.4.1 at hand, we transform (5.4.1) to the following problem

$$\begin{cases} \min_{\alpha \in \mathbb{R}} \alpha \\ \text{s.t. } f(x^{(j)}) + (s^{(j)})^T(x - x^{(j)}) - \alpha \leq 0, \forall 0 \leq j \leq k, \forall x \in \mathcal{F}. \end{cases} \quad (5.4.4)$$

Remark 5.4.2 Each set

$$M_j := \{(x, \alpha) \in \mathcal{F} \times \mathbb{R} : f(x^{(j)}) + (s^{(j)})^T(x - x^{(j)}) - \alpha \leq 0\}$$

is an affine half-space in \mathbb{R}^{n+1} , i.e., a cutting plane. For $\mathcal{F} = \mathbb{R}^n$, $\mathcal{F} = \{Ax = b, x \geq 0\}$ or $\mathcal{F} = \{Ax \leq b\}$ the problem (5.4.4) is a relatively simple and easy to solve linear optimization problem.

Now we are able to summarize the above findings in Algorithm 5.4.1.

Algorithm 5.4.1: Cutting plane method

Input: Feasible initial value $x^{(0)} \in \mathcal{F}$.

```

1: for  $k = 0, 1, 2, \dots$  do
2:   Compute a  $s^{(k)} \in \partial f(x^{(k)})$ 
3:   if Suitable stopping criterium satisfied then
4:     Stop
5:   end if
6:   Solve (5.4.1) respectively (5.4.4) and get  $x^{(k+1)}$ .
7: end for

```

Theorem 5.4.3 Let the sequence $\{x^{(k)}\}_{k \in \mathbb{N}}$ of the iterations of Algorithm 5.4.1 be infinite. Then,

- (a) Each accumulation point of $\{x^{(k)}\}_{k \in \mathbb{N}}$ is an optimal solution of (5.2.1).
- (b) Let x^* be an optimal solution of (5.2.1) and let

$$f(x^{(k+1)}) - f_k^{CP}(x^{(k+1)}) \leq \varepsilon \quad (5.4.5)$$

for some $\varepsilon > 0$. Then, $f(x^{(k+1)}) - f(x^*) \leq \varepsilon$.

Proof. Let $\{x^{(k')}\}_{k' \in \mathbb{K}} \subset \{x^{(k)}\}_{k \in \mathbb{N}}$ be a subsequence with $x^{(k')} \rightarrow x^*$ as $k' \rightarrow \infty$. Because \mathcal{F} is closed, we have that $x^* \in \mathcal{F}$. Then, for $x \in \mathcal{F}$ and $0 \leq j < k$ it holds that

$$\begin{aligned} f(x) &\geq f_{k-1}^{\text{CP}}(x) \geq f_{k-1}^{\text{CP}}(x^{(k)}) \geq f(x^{(j)}) + (g^{(j)})^T(x^{(k)} - x^{(j)}) \\ &\xrightarrow{k \in \mathbb{K}, k \rightarrow \infty} f(x^{(j)}) + (g^{(j)})^T(x^* - x^{(j)}) \xrightarrow{j \in \mathbb{K}, j \rightarrow \infty} f(x^*), \end{aligned}$$

since f is continuous and $\|g^{(j)}\| \leq L$ as $j \rightarrow \infty$. With that, x^* is a global solution of (5.2.1) and (a) is proven.

Part (b) immediately follows from (5.4.5), since $f(x^*) \geq f_k^{\text{CP}}(x^*) \geq f_k^{\text{CP}}(x^{(k+1)}) \geq f(x^{(k+1)}) - \varepsilon$. \square

Remark 5.4.4 With these facts at hand, we see that (5.4.5) is in fact a suitable stopping criterium.

Remark 5.4.5 Due to (5.4.1), the iterations $x^{(k+1)}$ solve the problem

$$\min_{x \in \mathcal{F}} f_k^{\text{CP}}(x). \quad (5.4.6)$$

If the previous $x^{(k)}$ already solves the problem (5.4.6), one can of course choose $x^{(k+1)} := x^{(k)}$. The application of Theorem 5.4.3 (b) with $\varepsilon = 0$ yields

$$f(x^{(k+1)}) = f_k^{\text{CP}}(x^{(k+1)}) = f_k^{\text{CP}}(x^{(k)}) = f(x^{(k)}),$$

and therefore $f(x^{(k+1)}) = f(x^*)$, which means that $x^{(k+1)} = x^{(k)}$ is an optimal solution of (5.2.1). On the other hand, as long as $x^{(k)}$ is not an optimal solution, each new cutting plane in fact shrinks the feasible region.

MATLAB-function: cutting_plane_method.m

```

1 function [x, f_vals, iter] = cutting_plane_method(f, grad_f, x0, tol,
2 max_iters)
3 % Cutting plane method for convex optimization
4 %
5 % Inputs:
6 % - f: Function handle for the objective function f(x)
7 % - grad_f: Function handle for the gradient of f(x)
8 % - x0: Initial guess for x
9 % - tol: Convergence tolerance
10 % - max_iters: Maximum number of iterations
11 %
12 % Outputs:
13 % - x: Solution vector
14 % - f_vals: History of the function values at each iteration
15 % - iter: Number of iterations performed
16 %
17 % Initialize variables
18 x = x0;
19 f_vals = zeros(max_iters, 1);
20 cuts = {};  
% List of cuts (linear constraints)
21 %
22 % Parameters
23 epsilon = 1e-6; % Small constant for numerical stability
24 %
25 for iter = 1:max_iters

```

```

25      % Solve the relaxed problem with current cuts
26      [x_new, f_new] = solve_relaxed_problem(f, grad_f, x, cuts,
27          epsilon);
28
29      % Store the function value
30      f_vals(iter) = f_new;
31
32      % Check convergence
33      if norm(x_new - x) < tol && abs(f_new - f(x)) < tol
34          x = x_new;
35          break;
36      end
37
38      % Update the current solution
39      x = x_new;
40
41      % Generate and add new cut
42      new_cut = generate_cut(x, grad_f(x), f(x));
43      cuts = [cuts, {new_cut}];
44
45      % Truncate the f_vals array if convergence was reached early
46      f_vals = f_vals(1:iter);
47  end
48
49 function [x_opt, f_opt] = solve_relaxed_problem(f, grad_f, x, cuts,
50     epsilon)
51
52     % Solve the relaxed problem with current cuts
53     % The relaxed problem is min f(x) subject to all cuts
54     % Define the objective function for the relaxed problem
55     obj_fun = @(x_new) f(x_new);
56
57     % Define constraints
58     A = []; b = [];
59     for i = 1:length(cuts)
60         cut = cuts{i};
61         A = [A; cut.A];
62         b = [b; cut.b];
63     end
64
65     % Solve the linear problem with constraints
66     options = optimoptions('fmincon', 'Display', 'off');
67     x_opt = fmincon(obj_fun, x, A, b, [], [], [], [], options);
68     f_opt = f(x_opt);
69
70 function cut = generate_cut(x, grad_f_x, f_x)
71
72     % Generate a new cut (linear constraint)
73     % The cut is defined as: grad_f_x' * (x - x0) <= f(x) - f(x0)
74     cut.A = -grad_f_x'; % -grad_f_x to form the constraint
75     cut.b = f_x - grad_f_x' * x; % f(x) - grad_f_x' * x
76
77 end

```

Example 5.4.6 We reconsider the ℓ^1 -minimization problem introduced in Example 5.1.2 above. We compare the subgradient and the cutting-plane method for $d = 10$ with the same initial value. The convergence histories are shown in Figure 5.9, where we see that the cutting-plane method terminates

after 2 iterations with (up to the tolerance) the exact solution. \diamond

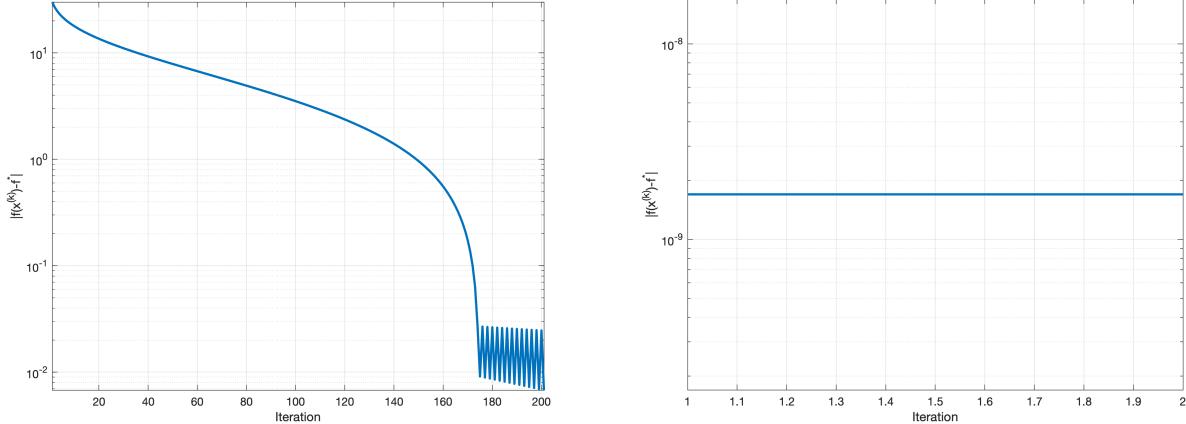


Figure 5.9: ℓ^1 -minimization in $d = 10$ space dimensions: convergence history for the subgradient (left) and the cutting plane method (right).

5.5 SEMISMOOTH NEWTON METHODS

In this section, we give a brief introduction into a class of algorithms which have proven to have great potential for non-smooth optimization problems. Semismooth Newton methods also rely on a concept for a weaker notion of differentiability, similar to the convex subgradient introduced above. However, in what we will describe next, we shall omit the assumption of convexity.

We can only give a *brief* introduction here, in particular since the full power of semismooth Newton method becomes clear in the infinite-dimensional setting, which, in turn, requires background in functional analysis. For more details on semismooth Newton methods, we refer e.g. to [23].

We start by recalling some well-known facts, which we partly have already seen above. Let $\mathcal{F} \subset \mathbb{R}^n$ and $f : \mathcal{F} \rightarrow \mathbb{R}^m$. We set

$$D_f := \{x \in \mathcal{F} : f'(x) \in \mathbb{R}^{m \times n} \text{ exists}\}, \quad (5.5.1)$$

where $f'(x) \equiv \nabla f(x)$ is the usual derivative. Recall, that a function f is called *locally Lipschitz continuous* at $x \in \mathcal{F}$, if there exists a neighborhood $\mathcal{N}(x)$, $x \in \mathcal{N}(x) \subset \mathcal{F}$ and a constant $L = L(x, \mathcal{N}(x)) < \infty$ such that

$$\|f(y) - f(z)\| \leq L \|y - z\| \quad \text{for all } y, z \in \mathcal{N}(x).$$

If f is locally Lipschitz continuous, it is known that the set $\mathcal{N}(x) \setminus D_f$ has Lebesgue measure zero. In other words, a locally Lipschitz continuous function is almost everywhere differentiable (this result is known as the Rademacher³ theorem). This motivates the following weak notion of differentiability.

Definition 5.5.1 Let $\mathcal{F} \subset \mathbb{R}^n$ be open and let $f : \mathcal{F} \rightarrow \mathbb{R}^m$ be locally Lipschitz continuous at some $x \in \mathcal{F}$. The set

$$\partial_B f(x) := \{M \in \mathbb{R}^{m \times n} : \exists (x_k)_{k \in \mathbb{N}} \subset D_f \text{ s.t. } x_k \rightarrow x, f'(x_k) \rightarrow M \text{ as } k \rightarrow \infty\}$$

is called *B-subdifferential (Bouligand⁴)* of f at x . The convex hull of $\partial_B f(x)$, i.e.,

$$\partial_C f(x) := \text{conv}\{\partial_B f(x)\} := \left\{ \sum_{i=1}^k \alpha_i M_i : M_i \in \partial_B f(x), k \in \mathbb{N}, \sum_{i=1}^k \alpha_i = 1, \alpha_i \geq 0 \right\},$$

³Hans Adolph Rademacher, 1892-1969.

⁴Georges Louis Bouligand, 1889-1979.

is called Clarke's⁵ generalized Jacobian of f at x .⁶

We note some basic properties of these quantities, somehow similar to properties of the convex subdifferential.

Proposition 5.5.2 Let $\mathcal{F} \subset \mathbb{R}^n$ be open and $f : \mathcal{F} \rightarrow \mathbb{R}^m$ locally Lipschitz continuous at $x \in \mathcal{F}$. Then,

(a) $\partial_C f(x)$ and $\partial_B f(x)$ are nonempty and compact.

(b) If f is locally differentiable at x , then $\partial_C f(x) = \partial_B f(x) = \{f'(x)\}$.

Proof. See, e.g. [23, Prop. 2.2]. □

In addition, this notion of weak differentiability allows for a calculus, which is quite analogous to what we know from the usual differential calculus, e.g. for $h : \mathbb{R}^\ell \rightarrow \mathbb{R}^m$ and $g : \mathbb{R}^n \rightarrow \mathbb{R}^\ell$, which reads

$$\partial_C(h \circ g)v \subset \text{conv}\{\partial_C h(g(x))\partial_C g(x)v\} = \text{conv}\{M_h M_g v : M_h \in \partial_C h(g(x)), M_g \in \partial_C g(x)\}$$

for all $v \in \mathbb{R}^n$ and if h is differentiable we get $\partial_C(h \circ g) = h'(g(x))\partial_C g(x)$, i.e., a generalization of the chain rule, see [23, Prop. 2.3]. Now we can define semismoothness.

Definition 5.5.3 Let $\emptyset \neq \mathcal{F} \subset \mathbb{R}^n$ be open. The function $f : \mathcal{F} \rightarrow \mathbb{R}^m$ is called semismooth at $x \in \mathcal{F}$, if it is locally Lipschitz continuous at x and if the limit

$$\lim_{\substack{M \in \partial_C f(x+\tau d) \\ d \rightarrow s \\ \tau \rightarrow 0+}} M d$$

exists for all $s \in \mathbb{R}^n$. We call f semismooth (on \mathcal{F}) if it is semismooth at all $x \in \mathcal{F}$.

One can prove the following: if f is continuously differentiable in a neighborhood of x , then f is semismooth at x and $\partial_C f(x) = \partial_B f(x) = \{f'(x)\}$, [23, Prop. 2.8].

Example 5.5.4 The following functions are semismooth:

- (i) $f : \mathbb{R}^n \rightarrow \mathbb{R}$, $x \mapsto \|x\|^2$.
- (ii) The *Fischer-Burmeister*⁷ function: $\phi_{FB} : \mathbb{R}^2 \rightarrow \mathbb{R}$ defined as $x \mapsto x_1 + x_2 - \|x\|$.
- (iii) $\phi_{\max} : \mathbb{R}^2 \rightarrow \mathbb{R}$ defined as $x \mapsto x_1 - \max\{x_1 - cx_2, 0\}$ for arbitrary $c > 0$.
- (iv) The function $f : \mathbb{R}^2 \rightarrow \mathbb{R}_0^+$ defined by $f(x) := \frac{1}{2}x_1^2 + 2|x_2|$ introduced in Example 5.3.12. ◊

Example 5.5.5 (i) The Wolfe function as defined in Example 5.1.1 is *not* semismooth. Although it is continuous and differentiable in each of its regions, the function has sharp transitions at the boundaries between these regions, particularly at $x_1 = |x_2|$ and $x_1 = 0$. These transitions prevent the existence of well-behaved generalized derivatives required for semismoothness. Thus, while the function is continuous and partially smooth, it does not satisfy the requirements for semismoothness.

- (ii) The function $f(x) = |x|$ is also *not* semismooth because the derivatives from the left and right near $x = 0$ do not approach the same value, resulting in a sharp change in the slope at the nondifferentiable point. ◊

Semismoothness admits different equivalent characterizations. We shall introduce some of them.

Definition 5.5.6 Let $\mathcal{F} \subset \mathbb{R}^n$ be open and $f : \mathcal{F} \rightarrow \mathbb{R}^m$.

⁵Clarke, Frank H., 1948-.

⁶Note, that we use a slightly different notation than in [23], where Clarke's subdifferential is denoted by $\partial f(x)$, whereas $\partial_C f(x)$ is used for Qi's subdifferential. Since we already used $\partial f(x)$ for the convex subdifferential, we wanted to avoid a notational clash. Moreover, we do not describe Qi's subdifferential here.

⁷Andreas Fischer, ?; Wolfgang Burmeister, ?-

(a) f is called directionally differentiable at $x \in \mathcal{F}$, if the directional derivative

$$f'(x, s) := \lim_{h \rightarrow 0+} \frac{f(x + hs) - f(x)}{h}$$

exists for all $s \in \mathbb{R}^n$.

(b) f is called B-differentiable at $x \in \mathcal{F}$, if f is directionally differentiable at x and

$$\|f(x + s) - f(x) - f'(x, s)\| = o(\|s\|) \quad \text{as } s \rightarrow 0. \quad (5.5.2)$$

Proposition 5.5.7 Let $\mathcal{F} \subset \mathbb{R}^n$ be open, $f : \mathcal{F} \rightarrow \mathbb{R}^m$ and $x \in \mathcal{F}$. Then, the following statements are equivalent

- (a) f is semismooth at x ;
- (b) f is locally Lipschitz continuous at x , $f'(x, \cdot)$ exists and

$$\sup_{M \in \partial_C f(x+s)} \|Ms - f'(x, s)\| = o(\|s\|) \quad \text{as } s \rightarrow 0;$$

- (c) f is locally Lipschitz continuous at x , $f'(x, \cdot)$ exists and (5.5.2) holds.

Proof. See [23, Prop. 2.7] and [15, Thm. 2.3]. \square

Example 5.5.8 The function $f_{\max} : \mathbb{R}^n \rightarrow \mathbb{R}^n$ defined as $x_i \mapsto \max\{x_i, 0\}$, $i = 1, \dots, n$, is semismooth and the matrix $G_\delta(x)$ is a particular element of $\partial_C f_{\max}(x)$, $x \in \mathbb{R}^n$, for every $\delta \in \mathbb{R}^n$, where

$$G_\delta(x) := \text{diag}(g_1(x_1), \dots, g_n(x_n)), \quad g_i(x_i) := \begin{cases} 0, & \text{if } x_i < 0, \\ 1, & \text{if } x_i > 0, \\ \delta_i, & \text{if } x_i = 0, \end{cases} \quad (5.5.3)$$

i.e., $G_\delta(x) \in \partial_C f_{\max}(x)$ for all $\delta \in \mathbb{R}^n$. We check condition (c) in Proposition 5.5.7. It is readily seen that f_{\max} is locally continuous and that the directional derivatives exist. Next, $\{\|G_\delta(x)\| : x \in \mathbb{R}^n\}$ is bounded. Defining

$$D(x, h) := \|f_{\max}(x + h) - f_{\max}(x) - G_\delta(x + h)h\|,$$

it is readily seen that $D(x, h) = 0$ for all $\|h\|_\infty < \min\{|x_i| : x_i \neq 0\} =: \beta$ which ensures (5.5.2).

Note, that the value of the generalized derivative G_δ of the max-function can be assigned an *arbitrary value* at the coordinates satisfying $x_i = 0$. In fact, we have $D(x, h) = 0$ if $\|h\|_\infty < \beta$. Moreover, for every $\gamma > \beta$ there exists h satisfying $D(x, h) \geq \beta$ and $\|h\|_\infty = \gamma$. Here we assume that $\beta := 0$ whenever $\{i = 1, \dots, n : x_i \neq 0\} = \emptyset$. Consequently, for $\beta > 0$ the mapping $\gamma \mapsto \sup\{\max\{0, x + h\} - \max\{0, x\} - G_\delta(x + h)h\|_\infty : \|h\|_\infty = \gamma\}$ is discontinuous at $\gamma = \beta$ and equals zero for $\gamma \in (0, \beta)$. \diamond

We are now going to describe a Newton-like method for the solution of the equation

$$f(x) = 0, \quad (5.5.4)$$

where $f : \mathcal{F} \rightarrow \mathbb{R}^n$, $\mathcal{F} \subset \mathbb{R}^n$ open (i.e., $m = n$), is semismooth at an isolated solution $x^* \in \mathcal{F}$.⁸

⁸This means that there is a neighborhood of x^* in which there is no further root of f .

Algorithm 5.5.1: Semismooth Newton method

- 1: Choose an initial point $x^{(0)}$ and set $k = 0$.
 - 2: If $f(x^{(k)}) = 0$, then STOP.
 - 3: Choose $M^{(k)} \in \partial_C f(x^{(k)})$ and compute $\delta x^{(k)}$ from $M^{(k)}\delta x^{(k)} = -f(x^{(k)})$.
 - 4: Set $x^{(k+1)} = x^{(k)} + \delta x^{(k)}$, $k \rightarrow k + 1$ and goto step 2.
-

We report some remarkable convergence properties of Algorithm 5.5.1.

Proposition 5.5.9 Let $f : \mathcal{F} \rightarrow \mathbb{R}^n$ be defined on the open set $\mathcal{F} \subset \mathbb{R}^n$ and denote by $x^* \in \mathbb{R}^n$ an isolated solution of (5.5.4). Furthermore, assume

- (i) The estimate (5.5.2) holds for $x = x^*$;
 - (ii) There exists a constant $C > 0$, such that $M^{(k)}$ is nonsingular for all k with $\|(M^{(k)})^{-1}\| \leq C$.⁹
- Then, there exists some $\delta > 0$ such that, for all $x^{(0)} \in B_\delta(x^*)$, Algorithm 5.5.1 either terminates with $x^{(k)} = x^*$ or it generates an infinite sequence $\{x^{(k)}\}_{k \in \mathbb{N}}$ that converges Q-superlinearly to x^* .

Proof. Denote the error by $e^{(k)} := x^{(k)} - x^*$, then we have

$$\begin{aligned} M^{(k)}e^{(k+1)} &= M^{(k)}(x^{(k)} + \delta x^{(k)} - x^*) = M^{(k)}(e^{(k)} + \delta x^{(k)}) = M^{(k)}e^{(k)} - f(x^{(k)}) \\ &= -[f(x^* + e^{(k)}) - f(x^*) - M^{(k)}e^{(k)}]. \end{aligned}$$

Using (5.5.2) yields $\|M^{(k)}e^{(k+1)}\| = o(\|e^{(k)}\|)$ as $\|e^{(k)}\| \rightarrow 0$. This means, for sufficiently small $\delta > 0$, we have $\|M^{(k)}e^{(k+1)}\| \leq \frac{1}{2C}\|e^{(k)}\|$ for $\|e^{(k)}\| \leq \delta$. Next, by (ii), we get that

$$\|e^{(k+1)}\| \leq \|(M^{(k)})^{-1}\| \|M^{(k)}e^{(k+1)}\| \leq \frac{1}{2}\|e^{(k)}\| \leq \dots \leq 2^{-(k+1)}\|e^{(0)}\|.$$

This shows that $x^{(k+1)} \in B_{\delta/2}(x^*)$ and inductively, we deduce that $x^{(k)} \rightarrow x^*$ and the rate of convergence is Q-superlinear. \square

MATLAB-function: semismooth_newton_method.m

```

1 function [allx, f_vals, iter] = semismooth_newton_method(F, J, x0, tol,
2   max_iters)
3   % Semismooth Newton method for solving nonsmooth equations
4   %
5   % Inputs:
6   % - F: Handle to the nonsmooth function, F(x) (returns a vector)
7   % - J: Handle to the generalized Jacobian, J(x) (returns a matrix)
8   % - x0: Initial guess
9   % - tol: Tolerance for convergence
10  % - max_iters: Maximum number of iterations
11  %
12  % Outputs:
13  % - x: Solution vector
14  % - f_vals: History of the norm of F(x) at each iteration
15  %
16  % Initialize variables
17  x = x0;

```

⁹This means that the inverses are uniformly bounded w.r.t. k .

```

8     allx=[x];
9     f_vals = zeros(max_iters, 1);
10
11    for iter = 1:max_iters
12        % Evaluate the function value and Jacobian at the current
13        % iterate
14        Fx = F(x);
15        Jx = J(x);
16
17        % Check convergence
18        f_vals(iter) = norm(Fx, inf);
19        if f_vals(iter) < tol
20            break;
21        end
22
23        % Solve the linear system J(x) * delta_x = -F(x)
24        delta_x = -Jx \ Fx;
25
26        % Update the solution
27        x = x + delta_x;
28        allx = [allx, x];
29
30        % (Optional) Display iteration details
31        fprintf('Iter %d: ||F(x)||_inf = %.4e\n', iter, f_vals(iter));
32    end
33
34    % Truncate the f_vals array if convergence was reached early
35    f_vals = f_vals(1:iter);
36 end

```

Example 5.5.10 We consider the regularized hinge loss function introduced in Example 5.3.28 and compare the projected subgradient with the semismooth Newton method. We choose $d = 5$, $s = 100$, $\tau = 2$, $\mu = 1$ and $\lambda = 1$. The results are shown in Figure 5.10. We clearly see that the semismooth Newton method converges much faster (note, the different scaling of the horizontal axis). However, it requires the generalized Hessian matrix, which only exists for a non-trivial regularization $\tau \neq 0$.

As a conclusion, the subgradient method may take a large number of iterations to converge due to the nature of the non-smooth problem. The convergence rate is slow because it relies on approximations using subgradients, which do not carry much curvature information. On the other hand, the semismooth Newton method will generally converge much faster, especially when close to the optimal solution. This is because the method can take advantage of the piecewise linear structure of the hinge loss and apply Newton-type updates based on generalized derivatives.

In the context of SVM hinge loss minimization or other non-smooth convex optimization problems involving piecewise linear or quadratic functions, the semismooth Newton method often outperforms the subgradient method, especially in terms of convergence speed and computational efficiency. This superiority is especially pronounced in large-scale problems where slow sublinear convergence of the subgradient method can be prohibitive, while the semismooth Newton method uses second-order information and achieves faster convergence. However, w.r.t. to the presented Matlab code, our solver should work well for small to medium-sized datasets. For larger datasets, more sophisticated strategies (e.g., sparse or block matrix operations) may be needed for handling the Hessian efficiently. \diamond

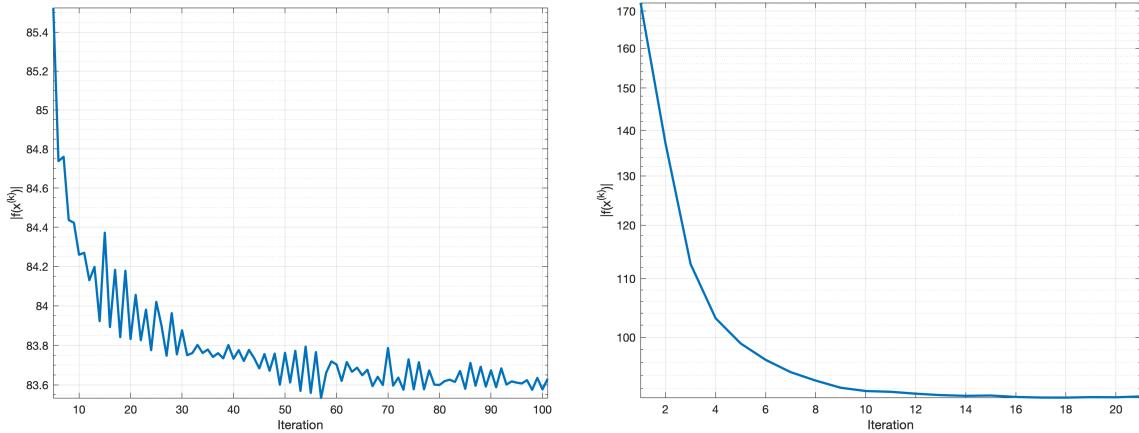


Figure 5.10: Iteration history for the subgradient (left) and the semismooth Newton method (right).

5.6 THE PRIMAL-DUAL ACTIVE SET STRATEGY

We are now going to describe a particular example which shows how semismooth Newton methods can be used (and constructed) within the context of constrained optimization.

Let $A \in \mathbb{R}^{n \times n}$ be symmetric and positive definite¹⁰. Consider the inequality-constrained optimization problem

$$J(y) := \frac{1}{2}\langle y, Ay \rangle - \langle g, y \rangle \rightarrow \min! \quad \text{s.t. } y \leq \psi,$$

for given $g, \psi \in \mathbb{R}^n$. The optimality system reads

$$\begin{cases} Ay + \lambda = g, \\ y \leq \psi, \lambda \geq 0, \langle \lambda, y - \psi \rangle = 0, \end{cases}$$

and can equivalently be rewritten as

$$\begin{cases} Ay + \lambda = g, \\ C(y, \lambda) = 0, \end{cases} \quad (5.6.1)$$

where $C(y, \lambda) := \lambda - \max\{0, \lambda + c(y - \psi)\}$ for any $c > 0$, which we fix from now on. We will show below that applying the semismooth Newton method to (5.6.1) results in Algorithm 5.6.1, which is also known as the *primal-dual active set strategy*. In fact, we will see that the algorithm can in fact be interpreted as a semismooth Newton method, so that Proposition 5.5.9 ensures Q-superlinear convergence.

Algorithm 5.6.1: Primal-dual active set strategy

- 1: Initialize $y^{(0)}, \lambda^{(0)}$, set $k = 0$.
- 2: Set $\mathcal{I}^{(k)} := \{i = 1, \dots, n : \lambda_i^{(k)} + c(y^{(k)} - \psi)_i \leq 0\}$
and $\mathcal{A}^{(k)} := \{i = 1, \dots, n : \lambda_i^{(k)} + c(y^{(k)} - \psi)_i > 0\}$.
- 3: Solve
$$\begin{cases} Ay^{(k+1)} + \lambda^{(k+1)} &= g, \\ y_i^{(k+1)} &= \psi_i \text{ for } i \in \mathcal{A}^{(k)}, \\ \lambda_i^{(k+1)} &= 0 \text{ for } i \in \mathcal{I}^{(k)}. \end{cases}$$

¹⁰This assumption is just for simplicity of exposition, the results are valid also under milder conditions on A .

4: STOP or set $k \rightarrow k + 1$ and return to step 2.

In order to show that Algorithm 5.6.1 can be interpreted as a semismooth Newton method, it is convenient to arrange the coordinates in such a way that inactive and active indices are ordered. Hence, we obtain a block decomposition of the matrix A as follows

$$\begin{aligned} A_{\mathcal{I}^{(k)}, \mathcal{I}^{(k)}} &:= (a_{i,j})_{i,j \in \mathcal{I}^{(k)}}, & A_{\mathcal{A}^{(k)}, \mathcal{A}^{(k)}} &:= (a_{i,j})_{i,j \in \mathcal{A}^{(k)}}, \\ A_{\mathcal{A}^{(k)}, \mathcal{I}^{(k)}} &:= (a_{i,j})_{i \in \mathcal{A}^{(k)}, j \in \mathcal{I}^{(k)}}, & A_{\mathcal{I}^{(k)}, \mathcal{A}^{(k)}} &:= (a_{i,j})_{i \in \mathcal{I}^{(k)}, j \in \mathcal{A}^{(k)}}, \end{aligned}$$

so that A can be written as a block matrix as follows

$$A = \begin{pmatrix} A_{\mathcal{I}^{(k)}, \mathcal{I}^{(k)}} & A_{\mathcal{I}^{(k)}, \mathcal{A}^{(k)}} \\ A_{\mathcal{A}^{(k)}, \mathcal{I}^{(k)}} & A_{\mathcal{A}^{(k)}, \mathcal{A}^{(k)}} \end{pmatrix}.$$

Analogously, we decompose the vector $y = (y_{\mathcal{I}^{(k)}}, y_{\mathcal{A}^{(k)}})$ and similarly for g and ψ .

Let us now detail the Newton correction step for $x := (y, \lambda) \in \mathbb{R}^{2n}$ and $f : \mathbb{R}^{2n} \rightarrow \mathbb{R}^{2n}$ defined by

$$f(y, \lambda) := (Ay + \lambda - g, C(y, \lambda)).$$

Recalling that the matrix $G_0(x)$ defined in (5.5.3) for $\delta = 0$ gives rise to an element of $\partial_C f_{\max}(x)$ (which is needed for $C(y, \lambda)$), a semismooth Newton step for f is given by

$$\underbrace{\begin{pmatrix} A_{\mathcal{I}^{(k)}, \mathcal{I}^{(k)}} & A_{\mathcal{I}^{(k)}, \mathcal{A}^{(k)}} & I_{\mathcal{I}^{(k)}} & 0 \\ A_{\mathcal{A}^{(k)}, \mathcal{I}^{(k)}} & A_{\mathcal{A}^{(k)}, \mathcal{A}^{(k)}} & 0 & I_{\mathcal{A}^{(k)}} \\ 0 & 0 & I_{\mathcal{I}^{(k)}} & 0 \\ 0 & -cI_{\mathcal{A}^{(k)}} & 0 & 0 \end{pmatrix}}_{=:M^{(k)}} \underbrace{\begin{pmatrix} \delta y_{\mathcal{I}^{(k)}}^{(k)} \\ \delta y_{\mathcal{A}^{(k)}}^{(k)} \\ \delta \lambda_{\mathcal{I}^{(k)}}^{(k)} \\ \delta \lambda_{\mathcal{A}^{(k)}}^{(k)} \end{pmatrix}}_{=:x^{(k)}} = -\underbrace{\begin{pmatrix} (Ay^{(k)} + \lambda^{(k)} - g)_{\mathcal{I}^{(k)}} \\ (Ay^{(k)} + \lambda^{(k)} - g)_{\mathcal{A}^{(k)}} \\ \lambda_{\mathcal{I}^{(k)}}^{(k)} \\ -c(y^{(k)} - \psi)_{\mathcal{A}^{(k)}} \end{pmatrix}}_{=-f(x^{(k)})}, \quad (5.6.2)$$

namely $M^{(k)} \in \partial_C f(x^{(k)})$. In fact, we have that

$$C(y, \delta)_i = \begin{cases} \lambda_i, & \text{for } i \in \mathcal{I}^{(k)}, \\ -c(y - \psi)_i, & \text{for } i \in \mathcal{A}^{(k)}, \end{cases}$$

so that (using $G_0(x) \in \partial_C f_{\max}(x)$)

$$\left(\frac{\partial C_j}{\partial y}(y, \lambda) \right)_i = \begin{cases} -c, & \text{for } i = j \in \mathcal{A}^{(k)}, \\ 0, & \text{else,} \end{cases} \quad \text{and} \quad \left(\frac{\partial C_j}{\partial y}(y, \lambda) \right)_i = \begin{cases} 1, & \text{for } i = j \in \mathcal{I}^{(k)}, \\ 0, & \text{else,} \end{cases}$$

which explains the third and fourth row of $M^{(k)}$.

Now, we show that (5.6.2) is equivalent to the KKT system in line 3 of Algorithm 5.6.1. In fact, the third equation in the system (5.6.2) reads $\delta \lambda_{\mathcal{I}^{(k)}}^{(k)} = -\lambda_{\mathcal{I}^{(k)}}^{(k)}$, so that the Newton correction yields

$$\lambda_{\mathcal{I}^{(k)}}^{(k+1)} = \lambda_{\mathcal{I}^{(k)}}^{(k)} + \delta \lambda_{\mathcal{I}^{(k)}}^{(k)} = 0,$$

which is precisely the third condition in line 3 of Algorithm 5.6.1.

The fourth equation in (5.6.2) reads $-c \delta y_{\mathcal{A}^{(k)}}^{(k)} = c(y^{(k)} - \psi)_{\mathcal{A}^{(k)}}$, which yields

$$y_{\mathcal{A}^{(k)}}^{(k+1)} = \psi_{\mathcal{A}^{(k)}},$$

i.e., the second line in step 3 of Algorithm 5.6.1. Finally, since the first two lines of (5.6.2) are equivalent to $Ay^{(k+1)} + \lambda^{(k+1)} = g$, we have shown that line 3 of Algorithm 5.6.1 is in fact a semismooth Newton step for the function f defined above and thus for the optimality system in (5.6.1).

Remark 5.6.1 It is a major advantage of the above described semismooth Newton approach that it can be generalized to the infinite-dimensional case, which allows the treatment of huge classes of applications including optimal control of flow problems, distributed control, etc.

We remark that sequential quadratic programming (SQP) algorithms, which are also quite efficient in practice, require the solution of an *inequality constrained quadratic program* per iteration, which can be significantly more expensive than a semismooth Newton step. Thus, it is also attractive to combine SQP methods with the class of Newton methods we described here, either by using the Newton method for solving subproblems, or by rewriting the complementarity conditions in the KKT system as an operator equation as we have seen above.

MATLAB-function: primal_dual_active_set_ssn.m

```

1 function [x, lambda, iter] = primal_dual_active_set_ssn(A, a, B, b, x0,
2     lambda0, tol, max_iters)
3     % Primal-Dual Active Set Method with Semi-Smooth Newton Method
4     %
5     % Minimize: (1/2) x' A x + a' x
6     % Subject to: B x = b
7     %
8     % Inputs:
9     % - A: Symmetric positive definite matrix in the quadratic term
10    % - a: Linear term in the objective function
11    % - B: Matrix for equality constraints
12    % - b: Vector for equality constraints
13    % - x0: Initial guess for the primal variables
14    % - lambda0: Initial guess for the dual variables (Lagrange
15        multipliers)
16    % - tol: Convergence tolerance
17    % - max_iters: Maximum number of iterations
18    %
19    % Outputs:
20    % - x: Optimal primal variables
21    % - lambda: Optimal dual variables
22    % - iter: Number of iterations performed
23
24    % Initialize variables
25    x = x0;
26    lambda = lambda0;
27    m = size(B, 1); % Number of equality constraints
28    n = length(x0); % Number of variables
29    iter = 0;
30
31    % Parameters for Semi-Smooth Newton Method
32    ssn_tol = 1e-6; % Tolerance for SSN method
33    max_ssn_iters = 100; % Maximum number of SSN iterations
34
35    while iter < max_iters
36        % Formulate the KKT system
37        KKT_matrix = [A, B'; B, zeros(m, m)];
38        rhs = [-a; b];
39
40        % Define function handle for semi-smooth Newton method
41        F = @(x_lambda) kkt_residual(KKT_matrix, rhs, x_lambda, n);
42        J = @(x_lambda) kkt_jacobian(KKT_matrix, n);

```

```

1 % Solve KKT system using Semi-Smooth Newton Method
2 [x_new_lambda, ~] = semismooth_newton_method(F, J, [x; lambda],
3     ssn_tol, max_ssn_iters);
4 x_new = x_new_lambda(1:n);
5 lambda_new = x_new_lambda(n+1:end);
6
7 % Compute primal and dual residuals
8 primal_residual = norm(B * x_new - b);
9 dual_residual = norm(A * x_new + a - B' * lambda_new);
0
1 % Update variables
2 x = x_new;
3 lambda = lambda_new;
4
5 % Check convergence
6 if primal_residual < tol && dual_residual < tol
7     break;
8 end
9
0 % Update iteration count
1 iter = iter + 1;
2 end
3
4 % Display results
5 fprintf('Optimal value of primal variables:\n');
6 disp(x);
7 fprintf('Optimal value of dual variables:\n');
8 disp(lambda);
9 fprintf('Number of iterations: %d\n', iter);
0 end
1
2 function r = kkt_residual(KKT_matrix, rhs, x_lambda, n)
3     % Compute the residual for the KKT system
4     % Residual = KKT_matrix * [x; lambda] - rhs
5     r = KKT_matrix * x_lambda - rhs;
6 end
7
8 function J = kkt_jacobian(KKT_matrix, n)
9     % Compute the Jacobian of the KKT residual
10    % For the linear system, the Jacobian is just the KKT_matrix
11    J = KKT_matrix;
12 end

```

Bibliography

- [1] ALT, W.: *Nichtlineare Optimierung*. Vieweg Studium: Aufbaukurs Mathematik. [Vieweg Studies: Mathematics Course]. Friedr. Vieweg & Sohn, Braunschweig, 2002. Eine Einführung in Theorie, Verfahren und Anwendungen. [An introduction to theory, procedures and applications].
- [2] ALT, W.: *Numerische Verfahren der konvexen, nichtglatten Optimierung* [in German]. Vieweg+Teubner Verlag, 2004.
- [3] BRACKEN, J. and G. P. MCCORMICK: *Selected Applications of Nonlinear Programming*. John Wiley and Sons, New York, 1968.
- [4] BRINKHUIS, J.: *Convex Sets: Basic Properties*, pp. 1–52. Springer International Publishing, Cham, 2020.
- [5] DAHMEN, W. and A. REUSKEN: *Numerik für Ingenieure und Naturwissenschaftler* [in German]. Springer-Lehrbuch. Springer Berlin Heidelberg, 3 ed., 2022.
- [6] DENNIS, JR., J. E. and R. B. SCHNABEL: *Numerical methods for unconstrained optimization and nonlinear equations*, vol. 16 of *Classics in Applied Mathematics*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1996. Corrected reprint of the 1983 original.
- [7] DEUFLHARD, P. and A. HOHMANN: *Numerische Mathematik. 1.* de Gruyter Lehrbuch. [de Gruyter Textbook]. Walter de Gruyter & Co., Berlin, Fourth ed., 2008. Eine algorithmisch orientierte Einführung. [An algorithmically oriented introduction].
- [8] FORST, W. and D. HOFFMANN: *Optimization—theory and practice*. Springer Undergraduate Texts in Mathematics and Technology. Springer, New York, 2010.
- [9] GROSS MANN, C. and J. TERNO: *Numerik der Optimierung*. Teubner Studienbücher Mathematik. [Teubner Mathematical Textbooks]. B. G. Teubner, Stuttgart, 1993.
- [10] HINZE, M., R. PINNAU, M. ULRICH and S. ULRICH: *Optimization with PDE constraints*, vol. 23 of *Mathematical Modelling: Theory and Applications*. Springer, New York, 2009.
- [11] Hooke, R. and T. A. JEEVES: “*Direct Search*” Solution of Numerical and Statistical Problems. J. ACM, 8(2):212–229, 1961.
- [12] KARMARKAR, N.: *A new polynomial-time algorithm for linear programming*. Combinatorica, 4(4):373–395, 1984.
- [13] NELDER, J. A. and R. MEAD: *A simplex method for function minimization*. Comput. J., 7(4):308–313, 1965.
- [14] NOCEDAL, J. and S. J. WRIGHT: *Numerical optimization*. Springer Series in Operations Research and Financial Engineering. Springer, New York, Second ed., 2006.
- [15] QI, L. and J. SUN: *A nonsmooth version of Newton’s method*. Mathematical Programming, 53:353–367, 1993.
- [16] QUARTERONI, A., R. SACCO and F. SALERI: *Numerical mathematics*, vol. 37 of *Texts in Applied Mathematics*. Springer-Verlag, Berlin, Second ed., 2007.

- [17] QUARTERONI, A., F. SALERI and P. GERVASIO: *Scientific computing with MATLAB and Octave*, vol. 2 of *Texts in Computational Science and Engineering*. Springer, Heidelberg, 2014. Fourth edition [of MR2253397].
- [18] REYES, J. C. DE LOS: *Numerical PDE-constrained optimization*. SpringerBriefs in Optimization. Springer, Cham, 2015.
- [19] SPELLUCCI, P.: *Numerische Verfahren der nichtlinearen Optimierung*. Internationale Schriftenreihe zur Numerischen Mathematik. [International Series of Numerical Mathematics]. Birkhäuser Verlag, Basel, 1993.
- [20] STOER, J.: *Numerische Mathematik. 1.* Springer-Lehrbuch. [Springer Textbook]. Springer-Verlag, Berlin, Seventh ed., 1994. Eine Einführung—unter Berücksichtigung von Vorlesungen von F. L. Bauer. [An introduction—based on the lectures of F. L. Bauer].
- [21] STOER, J. and R. BULIRSCH: *Numerische Mathematik. 2.* Springer-Lehrbuch. [Springer Textbook]. Springer-Verlag, Berlin, Third ed., 1990. Eine Einführung—unter Berücksichtigung von Vorlesungen von F. L. Bauer. [An introduction, with reference to lectures by F. L. Bauer].
- [22] TRÖLTZSCH, F.: *Optimal Control of Partial Differential Equations: Theory, Methods, and Applications*. Graduate studies in mathematics. American Mathematical Society, 2010.
- [23] ULRICH, M.: *Semismooth Newton Methods for Variational Inequalities and Constrained Optimization Problems in Function Spaces*, vol. 11 of *MOS-SIAM Series on Optimization*. SIAM, 2011.
- [24] ULRICH, M.: *Semismooth Newton methods for variational inequalities and constrained optimization problems in function spaces*, vol. 11 of *MOS-SIAM Series on Optimization*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA; Mathematical Optimization Society, Philadelphia, PA, 2011.
- [25] ULRICH, M. and S. ULRICH: *Nichtlineare Optimierung [in German]*. Birkhäuser, Springer Basel, 2012.
- [26] WÄCHTER, A. and L. BIEGLER: *On the Implementation of a Primal-Dual Interior Point Filter Line Search Algorithm for Large-Scale Nonlinear Programming*. Mathematical Programming, 106:25–57, 2006.
- [27] WALTER, E.: *Numerical methods and optimization*. Springer, Cham, 2014. A consumer guide.
- [28] ZOUTENDIJK, G.: *Mathematical programming methods*. North-Holland Publishing Co., Amsterdam-New York-Oxford, 1976.

Glossary English–Deutsch

central path zentraler Pfad.

cone Kegel.

constrained optimization Optimierung unter Nebenbedingungen.

curvature condition Wölbungsbedingung.

damped oscillation gedämpfte Schwingung.

damping Dämpfung.

descent Abstieg.

displacement Auslenkung.

exploration step Erkundungsschritt.

feasible zulässig.

feasible region zulässiger Bereich.

First order necessary optimality condition Notwendige Optimalitätsbedingungen erster Ordnung.

Interior point methods Innere-Punkte-Verfahren.

Lagrange multipliers Lagrange Multiplikatoren.

least squares problems Ausgleichsprobleme.

logarithmic barrier logarithmische Barriere.

null-space method Nullraum-Verfahren.

objective function Zielfunktion, Kostenfunktion.

penalty Strafterm.

progression step Fortschreitungsschritt.

Resetting algorithm Rücksetzungs-Algorithmus.

s.t. (subject to) unter den Nebenbedingungen.

Second order sufficient optimality conditions Hinreichende Optimalitätsbedingungen zweiter Ordnung.

slack variable Schlupfvariable.

steepest descent direction Richtung des steilsten Abstiegs.

unconstrained optimization Optimierung ohne Nebenbedingungen.