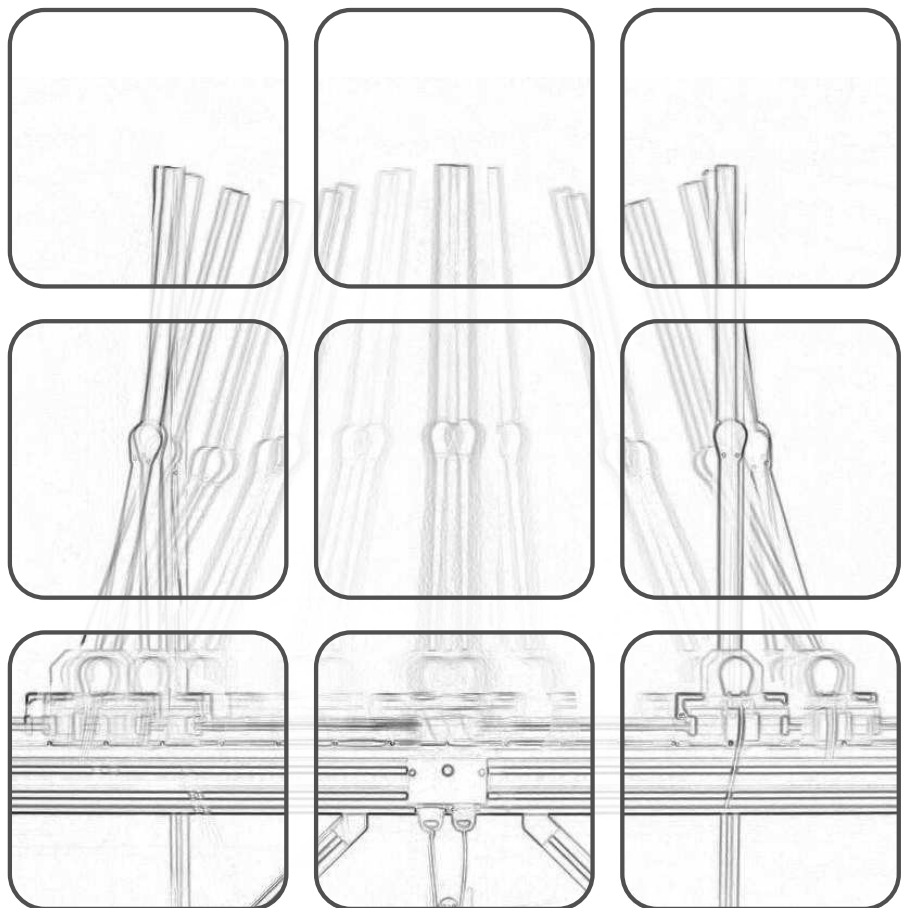


Lecture and Exercise  
WS 2026/2027

Andreas STEINBÖCK

# OPTIMIZATION



## **Optimization**

Lecture and Exercise  
WS 2026/2027

Andreas STEINBÖCK

TU Wien  
Automation and Control Institute  
Complex Dynamical Systems Group

Gußhausstraße 27–29  
1040 Wien  
Phone: +43 1 58801 – 37615  
Internet: <https://www.acin.tuwien.ac.at>

© Automation and Control Institute, TU Wien

# Contents

|          |                                                               |           |
|----------|---------------------------------------------------------------|-----------|
| <b>1</b> | <b>Introduction</b>                                           | <b>2</b>  |
| 1.1      | Static Optimization Problems . . . . .                        | 2         |
| 1.1.1    | Mathematical Formulation . . . . .                            | 2         |
| 1.1.2    | Examples . . . . .                                            | 3         |
| 1.2      | Dynamic Optimization Problems . . . . .                       | 7         |
| 1.2.1    | Mathematical Formulation . . . . .                            | 7         |
| 1.2.2    | Examples . . . . .                                            | 8         |
| 1.3      | Mathematical Foundations . . . . .                            | 12        |
| 1.3.1    | Infimum and Minimum . . . . .                                 | 12        |
| 1.3.2    | Existence of Minima . . . . .                                 | 13        |
| 1.3.3    | Gradient and Hessian . . . . .                                | 14        |
| 1.3.4    | Calculation of Derivatives . . . . .                          | 16        |
| 1.3.5    | Implicit Function Theorem . . . . .                           | 21        |
| 1.3.6    | Convexity . . . . .                                           | 21        |
| 1.3.6.1  | Convex Sets . . . . .                                         | 21        |
| 1.3.6.2  | Convex Functions . . . . .                                    | 22        |
| 1.4      | References . . . . .                                          | 25        |
| <b>2</b> | <b>Static Optimization: Unconstrained Case</b>                | <b>27</b> |
| 2.1      | Optimality Conditions . . . . .                               | 27        |
| 2.2      | Computer-Aided Minimization Methods: Fundamentals . . . . .   | 30        |
| 2.2.1    | Iterative Descend Methods . . . . .                           | 30        |
| 2.2.2    | Convergence Behavior . . . . .                                | 31        |
| 2.2.3    | Termination Criteria . . . . .                                | 32        |
| 2.3      | Line Search Methods . . . . .                                 | 33        |
| 2.3.1    | Choice of Step Size . . . . .                                 | 35        |
| 2.3.1.1  | Interval Bisection Method ('Golden-Section Search') . . . . . | 35        |
| 2.3.1.2  | Quadratic Interpolation . . . . .                             | 36        |
| 2.3.1.3  | Heuristic Choice of Step Size . . . . .                       | 37        |
| 2.3.2    | Choice of Search Direction . . . . .                          | 39        |
| 2.3.2.1  | Gradient Method . . . . .                                     | 39        |
| 2.3.2.2  | Conjugate Gradient Method . . . . .                           | 43        |
| 2.3.2.3  | Newton's Method . . . . .                                     | 49        |
| 2.3.2.4  | Quasi-Newton Method . . . . .                                 | 51        |
| 2.3.2.5  | Gauss-Newton Method . . . . .                                 | 55        |
| 2.4      | Trust Region Method . . . . .                                 | 58        |
| 2.5      | Direct Search Methods . . . . .                               | 61        |
| 2.6      | Example: Rosenbrock's "Banana Function" . . . . .             | 64        |

|     |                      |    |
|-----|----------------------|----|
| 2.7 | References . . . . . | 71 |
|-----|----------------------|----|

# Preface

Substantial parts of these lecture notes were authored by Prof. Dr.-Ing. Knut GRAICHEN and Univ.-Prof. Dr. techn. Andreas KUGL. They deserve thanks for their work. Please, address questions as well as suggestions for corrections and improvements in these lecture notes to Andreas STEINBÖCK at any time.

# 1 Introduction

*Optimization* is generally understood as the search for a point (optimal solution) in a decision space. Usually, constraints have to be taken into account during this search. The optimum point is best in terms of a certain objective. Mathematical formulations and solutions of optimization problems can be used to systematically design such decision-making processes. These lecture notes provide an overview of the mathematical formulation and solution of optimization problems.

We generally differentiate between *static* and *dynamic* optimization problems:

- *Static optimization problem*: Minimization of a function with optimization variables from a finite-dimensional space (e.g., Euclidean space)
- *Dynamic optimization problem*: Minimization of a functional with optimization variables from an infinite-dimensional space (e.g., time functions)

This section illustrates the fundamental difference between static and dynamic optimization problems by means of examples.

## 1.1 Static Optimization Problems

A *static optimization problem* is the minimization of a function  $f(\mathbf{x})$  subject to certain constraints, where the optimization variables  $\mathbf{x}$  come from the Euclidean space  $\mathbb{R}^n$ .

### 1.1.1 Mathematical Formulation

The standard formulation of a static optimization problem is

$$\min_{\mathbf{x} \in \mathbb{R}^n} f(\mathbf{x}) \quad \text{cost function} \quad (1.1a)$$

$$\text{u.B.v. } g_i(\mathbf{x}) = 0, \quad i = 1, \dots, p \quad \text{equality constraints} \quad (1.1b)$$

$$h_i(\mathbf{x}) \leq 0, \quad i = 1, \dots, q \quad \text{inequality constraints.} \quad (1.1c)$$

If an optimization problem is given without equality and inequality constraints (1.1b) and (1.1c), respectively, it is called an *unconstrained optimization problem*. In the general case, i.e., taking into account the constraints (1.1b) and (1.1c), it is a *constrained optimization problem*.

The set  $\mathcal{X} \subset \mathbb{R}^n$  contains points that satisfy the equality and inequality constraints (1.1b) and (1.1c), respectively.

$$\mathcal{X} = \{ \mathbf{x} \in \mathbb{R}^n \mid g_i(\mathbf{x}) = 0, \quad i = 1, \dots, p, \quad h_i(\mathbf{x}) \leq 0, \quad i = 1, \dots, q \} \quad (1.2)$$

This set is called the *admissible region* or *feasible region* and each point  $\mathbf{x} \in \mathcal{X}$  is called an *admissible point* or *feasible point*. Thus, the static optimization problem (1.1) can also be written in the equivalent form

$$\min_{\mathbf{x} \in \mathcal{X}} f(\mathbf{x}) . \quad (1.3)$$

In case of unconstrained problems,  $\mathcal{X} = \mathbb{R}^n$  holds.  $\mathcal{X}$  must not be an empty set because otherwise the optimization problem (1.3) does not have a solution. Another necessary condition for  $\mathcal{X}$  can be derived from the equality constraints (1.1b). The  $p$  algebraic restrictions  $g_i(\mathbf{x}) = 0$  reduce the number of free optimization variables  $\mathbf{x} \in \mathbb{R}^n$  to  $n - p$ . Thus,  $p$ , the number of equality constraints (1.1b), must not exceed  $n$ , the number of optimization variables  $\mathbf{x} \in \mathbb{R}^n$ . Otherwise, the feasible set  $\mathcal{X}$  may be empty.

In the literature, the formulation as a minimization problem (1.1) or (1.3) is most commonly used. Note that a maximization problem can also be written as a minimization problem in the form (1.3):

$$\max_{\mathbf{x} \in \mathcal{X}} f(\mathbf{x}) = - \left( \min_{\mathbf{x} \in \mathcal{X}} -f(\mathbf{x}) \right).$$

In addition to the term static optimization, the terms *mathematical programming* or *finite-dimensional optimization* are frequently used. The term *programming* is to be understood in the sense of *planning* rather than in the sense of coding a computer program.

The following classes of static optimization problems are often distinguished:

- *Linear programming*: The cost function and the constraints are linear (more precisely, affine).
- *Quadratic programming*: The cost function is quadratic, while the constraints are linear (more precisely, affine).
- *Nonlinear programming*: The cost function or at least one constraint is nonlinear.
- *Convex programming*: Convexity is a mathematical property that plays a special role in optimization. An optimization problem is convex if the cost function is a convex function and if the feasible set is a convex set. For convex optimization problems, the necessary first-order optimality conditions are also sufficient for a global optimum.
- *Integer programming*: All optimization variables are discrete.
- *Mixed-integer programming*: Both continuous and discrete optimization variables occur.

### 1.1.2 Examples

*Linear programming* is frequently used for economic problems, such as production planning or investment problems. The following example shows a simple portfolio optimization.

**Example 1.1 (Portfolio optimization).** An investor wants to invest 10,000 euros profitably and has the choice between three equity funds with different profit expectations and risk classifications:

| Fund | Expected profit/year | Risk classification |
|------|----------------------|---------------------|
| A    | 10 %                 | 4                   |
| B    | 7 %                  | 2                   |
| C    | 4 %                  | 1                   |

The investor wants to achieve a profit of at least 600 euros after one year. On the other hand, he wants to invest his money rather conservatively, i.e., he wants to invest at least 4,000 euros in Fund C and minimize the risk according to the risk classification given above. How should the investor allocate the 10,000 euros to meet these criteria?

First, the optimization variables  $x_1, x_2, x_3$  are introduced, which represent the percentage of the invested 10,000 euros in funds A, B, C, respectively. Here,  $x_3$  can be replaced by the relationship

$$x_3 = 1 - x_1 - x_2 . \quad (1.4)$$

The required minimum profit of 600 euros can be expressed by the constraint

$$10,000(0.1x_1 + 0.07x_2 + 0.04(1 - x_1 - x_2)) \geq 600 \quad \Rightarrow \quad 6x_1 + 3x_2 \geq 2 . \quad (1.5)$$

The minimum investment of 4,000 euros in Fund C leads to

$$10,000(1 - x_1 - x_2) \geq 4,000 \quad \Rightarrow \quad x_1 + x_2 \leq 0.6. \quad (1.6)$$

Furthermore,  $x_1 \geq 0$ ,  $x_2 \geq 0$  and  $x_3 \geq 0$  must be satisfied. With these constraints and (1.6) it is also ensured that  $x_1 \leq 1$ ,  $x_2 \leq 1$  and  $x_3 \leq 1$ . The goal is to minimize the investment risk, which can be expressed by the function

$$f(\mathbf{x}) = 4x_1 + 2x_2 + (1 - x_1 - x_2) = 1 + 3x_1 + x_2 . \quad (1.7)$$

Thus, the static optimization problem can be written in the form

$$\min_{\mathbf{x} \in \mathbb{R}^2} \quad f(\mathbf{x}) = 1 + 3x_1 + x_2 \quad (1.8a)$$

$$\text{u.B.v.} \quad 6x_1 + 3x_2 \geq 2 \quad (1.8b)$$

$$x_1 + x_2 \leq 0.6 \quad (1.8c)$$

$$x_1 \geq 0 \quad (1.8d)$$

$$x_2 \geq 0 . \quad (1.8e)$$

Figure 1.1 shows the individual constraints and the feasible set. the contour lines  $f(\mathbf{x}) = \text{const.}$  of the cost function (1.8a), it can be seen that the optimum  $\mathbf{x}^*$  equals

the corner point of the feasible set  $\mathcal{X}$  with the lowest value  $f(\mathbf{x})$ . Thus, the optimal allocation of 10,000 euros to the individual funds is

$$x_1^* = \frac{1}{15}, \quad x_2^* = \frac{8}{15}, \quad x_3^* = \frac{6}{15}. \quad (1.9)$$

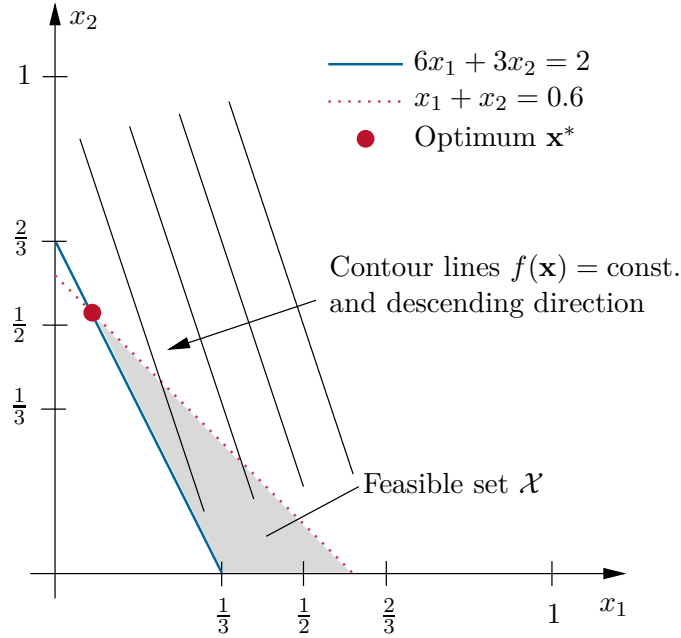


Figure 1.1: Graphical solution of portfolio optimization problem.

The following example of quadratic programming illustrates the influence of constraints on optimal solutions.

*Example 1.2.* Consider the (initially) unconstrained problem

$$\min_{\mathbf{x} \in \mathbb{R}^2} f(\mathbf{x}) = (x_1 - 2)^2 + (x_2 - 1)^2. \quad (1.10)$$

The contour lines  $f(\mathbf{x}) = \text{const.}$  of the function  $f(\mathbf{x})$  are shown in Figure 1.2 as a function of the two optimization variables  $\mathbf{x} = [x_1 \ x_2]^T$ . Clearly, the minimum  $f(\mathbf{x}^*) = 0$  is located at  $\mathbf{x}^* = [2 \ 1]^T$ .

To study the influence of different constraints, we first consider an equality constraint of the form (1.1b)

$$g(\mathbf{x}) = x_2 - 2x_1 = 0. \quad (1.11)$$

This corresponds to an algebraic constraint, implying that only one optimization variable is freely selectable. Geometrically interpreted, this means that a possible solution must lie on the line defined by (1.11) (see Figure 1.3). The optimal solution is the tangential contact point of the line  $g(\mathbf{x}) = 0$  and the contour line  $f(\mathbf{x}) = 1.8$ .

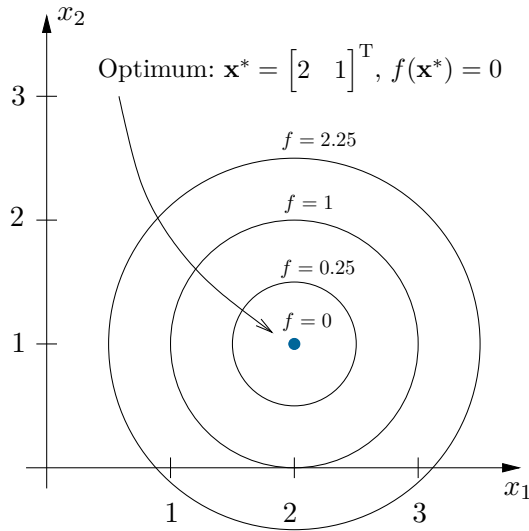


Fig. 1.2: Graphical representation of the unconstrained optimization problem (1.10).

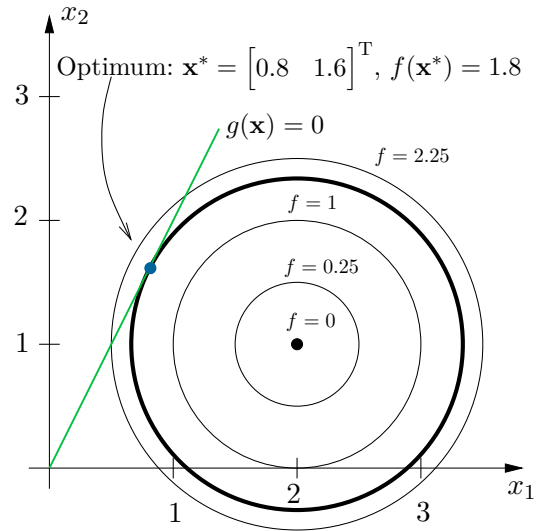


Fig. 1.3: Graphical representation of the constrained optimization problem (1.10), (1.11).

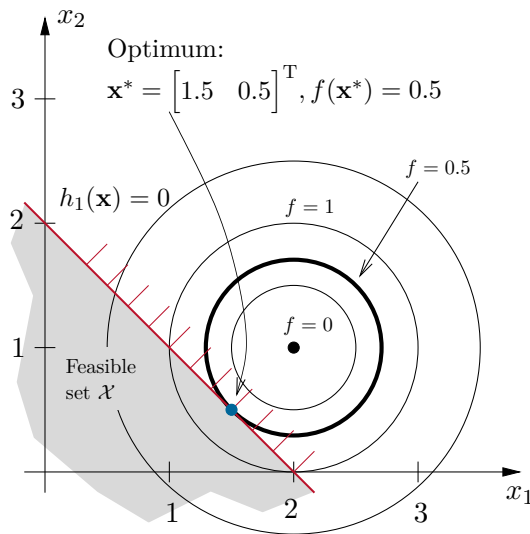


Fig. 1.4: Graphical representation of the constrained optimization problem (1.10), (1.12).

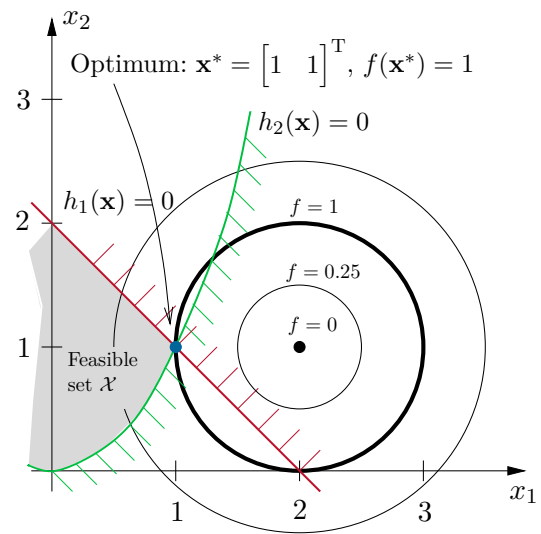


Fig. 1.5: Graphical representation of the constrained optimization problem (1.10), (1.12), (1.13).

Instead of the equality constraint (1.11), we now consider the inequality constraint

$$h_1(\mathbf{x}) = x_1 + x_2 - 2 \leq 0 \quad (1.12)$$

which restricts the set of feasible points  $\mathbf{x} = [x_1 \ x_2]^T$  to the area below the line  $h_1(\mathbf{x}) = 0$  (see Figure 1.4). The optimum  $f(\mathbf{x}^*) = 0.5$  at  $\mathbf{x}^* = [1.5 \ 0.5]^T$  is on the boundary of the feasible set and, as in the previous scenario, lies on a contour line that intersects the line  $h_1(\mathbf{x}) = 0$  tangentially.

In addition to the first inequality constraint (1.12), another inequality of the form

$$h_2(\mathbf{x}) = x_1^2 - x_2 \leq 0 \quad (1.13)$$

is to be considered, which further reduces the set of admissible points (see Figure 1.5). The optimal point  $\mathbf{x}^* = [1 \ 1]^T$  with the minimum  $f(\mathbf{x}^*) = 1$  is now at the intersection of the curves  $h_1(\mathbf{x}) = 0$  and  $h_2(\mathbf{x}) = 0$ , i.e., both constraints (1.12) and (1.13) are active.

The above Example 1.2 illustrates the influence of equality and inequality constraints on the solution (and solvability) of a static optimization problem. The systematic investigation of static optimization problems and methods for their numerical solution will be discussed in later sections.

## 1.2 Dynamic Optimization Problems

In static optimization problems, the optimization variables  $\mathbf{x}$  are elements of a finite-dimensional space, usually the Euclidean space  $\mathbb{R}^n$ . In dynamic optimization, however, an optimum is searched for in a space of functions of an independent variable. Because this independent variable is usually the time  $t$ , the term *dynamic optimization* is often used in this context.

### 1.2.1 Mathematical Formulation

The general structure of a dynamic optimization problem is

$$\min_{\mathbf{u}(\cdot)} \quad J(\mathbf{u}) = \varphi(t_1, \mathbf{x}(t_1)) + \int_{t_0}^{t_1} l(t, \mathbf{x}(t), \mathbf{u}(t)) \, dt \quad \text{cost functional} \quad (1.14a)$$

$$\text{u.B.v.} \quad \dot{\mathbf{x}} = \mathbf{f}(t, \mathbf{x}, \mathbf{u}) \quad \text{system dynamics} \quad (1.14b)$$

$$\mathbf{x}(t_0) = \mathbf{x}_0 \quad \text{initial conditions} \quad (1.14c)$$

$$\psi(t_1, \mathbf{x}(t_1)) = \mathbf{0} \quad \text{terminal constraints} \quad (1.14d)$$

$$h_i(\mathbf{x}, \mathbf{u}) \leq 0, \quad i = 1, \dots, q \quad \text{inequality constraints.} \quad (1.14e)$$

Here,  $\mathbf{u} \in \mathbb{R}^m$  represents the input of the nonlinear system (1.14b) with the state  $\mathbf{x} \in \mathbb{R}^n$ . In addition to the initial conditions (1.14c), terminal constraints of the form (1.14d) are often given, e.g., to reach a desired state  $\mathbf{x}_f$  at the terminal time  $t_1$  (i.e.,  $\psi(t_1, \mathbf{x}(t_1)) = \mathbf{x}(t_1) - \mathbf{x}_f$ ). In practice, inequality constraints (1.14e) often occur, which can represent, for example, the limitation of a control variable or safety limits of a state.

The task of dynamic optimization is to find an input trajectory  $\mathbf{u}(t)$ ,  $t \in [t_0, t_1]$  such that the state trajectory  $\mathbf{x}(t)$ ,  $t \in [t_0, t_1]$  of the dynamic system (1.14b) with the initial conditions (1.14c) satisfies the terminal constraints (1.14d), the constraints (1.14e) are

satisfied, and at the same time the cost functional (1.14a) is minimized. Depending on whether  $t_1$  is given or unknown, the *terminal time*  $t_1$  is either *fixed* or *free*, respectively.

In addition to the term dynamic optimization, the terms *infinite-dimensional optimization*, *optimal control problem*, or *dynamic programming* are frequently used. The following examples illustrate the problem and task of dynamic optimization.

### 1.2.2 Examples

**Example 1.3 (Inverted pendulum).** A classic problem in control engineering is the inverted pendulum, which is hinge-mounted on a cart. As an example problem, we consider the lateral movement of the pendulum

$$\min_{u(\cdot), t_1} J(u) = \int_0^{t_1} 1 + c u^2 \, dt, \quad (1.15a)$$

$$\text{u.B.v.} \quad \begin{bmatrix} 1 & \varepsilon \cos \theta \\ \cos \theta & 1 \end{bmatrix} \begin{bmatrix} \ddot{x} \\ \ddot{\theta} \end{bmatrix} = \begin{bmatrix} \varepsilon \dot{\theta}^2 \sin \theta + u \\ -\sin \theta \end{bmatrix}, \quad \varepsilon = m/(M + m) \quad (1.15b)$$

$$\mathbf{x}(0) = [0 \ 0 \ \pi \ 0]^T, \quad \mathbf{x}(t_1) = [1 \ 0 \ \pi \ 0]^T, \quad (1.15c)$$

$$-1 \leq u \leq 1. \quad (1.15d)$$

The simplified equations of motion (1.15b) for the states  $\mathbf{x} = [x \ \dot{x} \ \theta \ \dot{\theta}]^T$  are normalized. The input  $u$  represents the force acting on the cart and is bounded by (1.15d). The mass of the pendulum is denoted by  $m$ , that of the cart by  $M$ . Figure 1.6 shows an example of the lateral displacement of the pendulum to illustrate the motion.

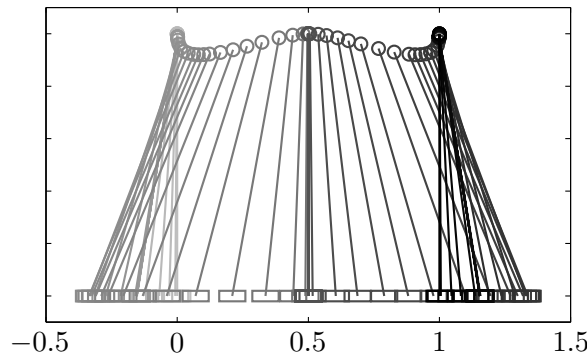


Figure 1.6: Snapshots of the movement of the inverted pendulum.

The cost functional (1.15a) and thus the character of the optimization problem depends on the parameter  $c$ . For  $c = 0$ , the task is to minimize just the terminal

time  $t_1$

$$J(u) = \int_0^{t_1} 1 \, dt = t_1. \quad (1.16)$$

For  $c > 0$ , the input  $u$  (and thus the control effort) is also taken into account in the cost functional.

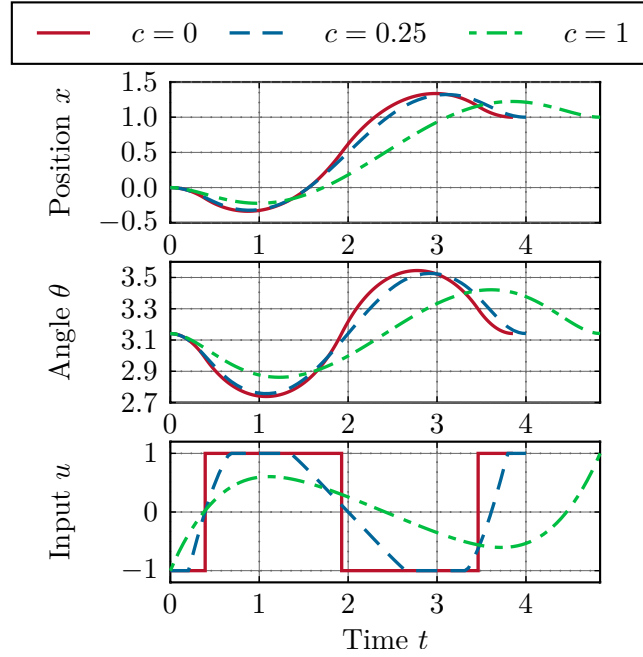


Figure 1.7: Optimal trajectories for the movement of the inverted pendulum.

Figure 1.7 shows the optimal trajectories for the parameter value  $\varepsilon = 0.5$  and for the values  $c = 0$ ,  $c = 0.25$ , and  $c = 1$ . For  $c = 0$ , the input  $u$  exhibits so-called bang-bang behavior, while for  $c > 0$  the control amplitudes become smaller and the required time  $t_1$  increases.

This example illustrates that not every optimization problem has a solution, especially if the terminal time  $t_1$  is not fixed. As can be seen from Figure 1.7, the terminal time  $t_1$  increases as the weighting of  $u^2$  increases compared to the time-optimal part in the cost functional (1.15a). If only the control variable  $u$  was weighted, i.e.,

$$J(u) = \int_0^{t_1} u^2 \, dt, \quad (1.17)$$

the optimization problem would not have a solution because the pendulum motion would happen infinitely slowly, i.e., with  $t_1 \rightarrow \infty$ .

**Example 1.4 (Goddard rocket [1.1, 1.2]).** A classic optimization problem from aerospace engineering is the maximization of the altitude of a rocket subject to air drag and gravity. This problem was posed by Robert H. Goddard in 1919 and

can be written in the normalized form

$$\min_{u(\cdot)} -h(t_1) \quad (1.18a)$$

$$\text{u.B.v.} \quad \dot{h} = v, \quad \dot{v} = \frac{u - D(h, v)}{m} - \frac{1}{h^2}, \quad \dot{m} = -\frac{u}{c}, \quad (1.18b)$$

$$h(0) = 1, \quad v(0) = 0, \quad m(0) = 1, \quad m(t_1) = 0.6, \quad (1.18c)$$

$$0 \leq u \leq 3.5. \quad (1.18d)$$

The state variables are the altitude  $h$ , the velocity  $v$ , and the mass  $m$  of the rocket. The air drag  $D(h, v)$  depends on the states  $h$  and  $v$  in the form

$$D(h, v) = D_0 v^2 e^{\beta(1-h)}. \quad (1.19)$$

The boundary conditions in (1.18c) include the normalized initial conditions and a terminal constraint for  $m(t_1)$ , which corresponds to the empty weight of the rocket without fuel. The input of the system is the thrust  $u$ , which must satisfy the bounds (1.18d).

Figure 1.8 shows the optimal trajectories of the Goddard rocket. The parameter values used are  $c = 0.5$ ,  $D_0 = 310$ , and  $\beta = 500$ . The thrust  $u(t)$  is initially maximal and then has a parabolic profile before the fuel is consumed. This behavior is caused by the air resistance  $D(h, v)$ , which decreases with increasing altitude. In the case of high air resistance, it is therefore *not optimal* to fly permanently with full thrust.

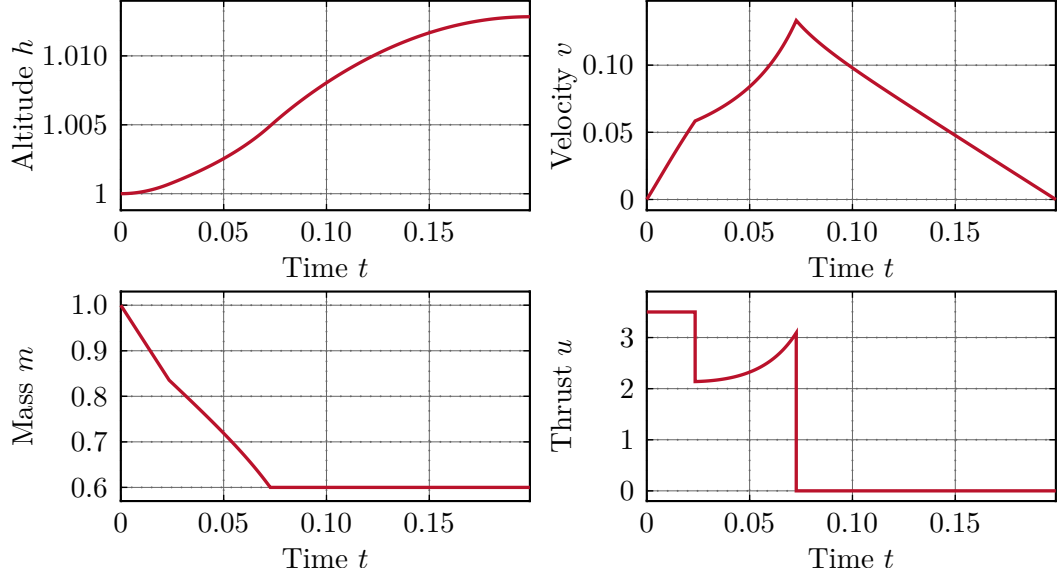


Figure 1.8: Trajectories for the Goddard rocket.

*Example 1.5 (Economic model [1.3, 1.4]).* Dynamic optimization can also be applied to economic processes. The following example describes the behavior of a person who wants to optimize consumption, leisure, and education over the lifespan. The educational level  $B$  and the capital  $K$  of the person are described by the following model.

$$\dot{B} = \overbrace{B^\varepsilon u_2 u_3}^{\text{Further training}} - \overbrace{\delta B}^{\text{Forgetting}}, \quad B(0) = B_0 \quad (1.20a)$$

$$\dot{K} = \underbrace{iK}_{\text{Interest}} + \underbrace{B u_2 g(u_3)}_{\text{Income}} - \underbrace{u_1}_{\text{Consumption}}, \quad K(0) = K_0 \quad (1.20b)$$

The input variables are consumption  $u_1$ , the ratio of working time to total time  $u_2$ , and the ratio of training time to working time  $u_3$ . The inputs are subject to the constraints

$$u_1 > 0, \quad 0 \leq u_2 \leq 1, \quad 0 \leq u_3 < 1. \quad (1.21)$$

The person's objective is to maximize consumption, leisure, and education over a lifespan of  $t_1 = 75$  years, which is expressed in the objective functional

$$J(\mathbf{u}) = -K^\kappa(t_1) - \int_{t_0}^{t_1} U(t, u_1, u_2, B) e^{-\rho t} dt, \quad (1.22)$$

which is to be minimized. The utility function

$$U(t, u_1, u_2, B) = \alpha_0 u_1^\alpha + \beta_0 (1 - u_2)^\beta + \gamma_0 t B^\gamma \quad (1.23)$$

weights the consumption  $u_1$ , leisure  $1 - u_2$ , and educational level  $B$ , while the terminal value  $-K^\kappa(t_1)$  in (1.22) additionally takes into account the inheritable capital. The function  $g(u_3)$  in (1.20b) is given by the parabola  $g(u_3) = 1 - (1 - a)u_3 - a u_3^2$ . The parameter values used are  $a = 0.3$ ,  $\alpha = -1$ ,  $\alpha_0 = -1$ ,  $\beta = -0.5$ ,  $\beta_0 = -1$ ,  $\gamma = 0.2$ ,  $\gamma_0 = 5$ ,  $\kappa = 0.2$ ,  $\rho = 0.01$ ,  $\varepsilon = 0.35$ ,  $\delta = 0.01$ ,  $i = 0.04$ ,  $B_0 = 1$ , and  $K_0 = 30$ .

The optimal trajectories of the educational level  $B(t)$  and the capital  $K(t)$  are shown in Figure 1.9. The first 17 years are a learning phase (i.e.,  $u_3 = 1$ ). This is followed by a long working phase of 34 years with a high level of further training, before the next 10 years (years 52 to 61) are a pure working phase with reduced working time  $u_2$ . Retirement begins at the age of 62. The educational level  $B$  is particularly high between the ages of 30 and 60. The capital  $K$  is negative during the first half of life, which corresponds to taking out a loan. In the course of life, however, this is compensated for by the rising income.

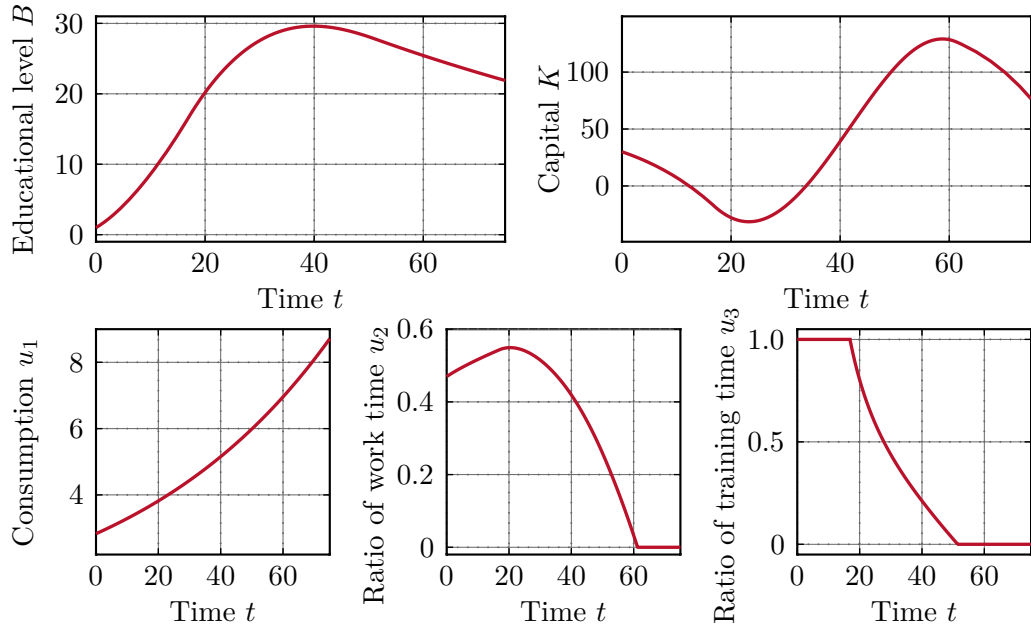


Figure 1.9: Optimal trajectories for consumer behavior.

## 1.3 Mathematical Foundations

This section briefly explains some mathematical terms and basic concepts that will facilitate the understanding of the following chapters.

### 1.3.1 Infimum and Minimum

**Definition 1.1 (Infimum).** Let  $\mathcal{Y} \subset \mathbb{R}$  be a non-empty set. The infimum of  $\mathcal{Y}$ , written as  $\inf \mathcal{Y}$ , denotes the greatest lower bound of  $\mathcal{Y}$ , i.e., there exists a number  $\alpha = \inf \mathcal{Y}$  such that

- (a)  $x \geq \alpha$  for all  $x \in \mathcal{Y}$
- (b) for all  $\bar{\alpha} > \alpha$  there exists an  $x \in \mathcal{Y}$  such that  $x < \bar{\alpha}$ .

If an infimum exists for a non-empty set  $\mathcal{Y}$ , it does not have to be contained in  $\mathcal{Y}$ . As an example, consider the set  $\mathcal{Y} = (0, +\infty)$ . In this case, obviously  $\inf \mathcal{Y} = 0 \notin \mathcal{Y}$ .

For the following definition, it is assumed that  $\mathcal{X} \subset \mathbb{R}^n$  defined in (1.2) denotes the admissible set of an optimization problem.

**Definition 1.2 (Global and local minima).** At the point  $\mathbf{x}^*$ , the function  $f(\mathbf{x})$  has

- (a) a *local minimum* if for some norm  $\|\cdot\|$  there exists an  $\varepsilon > 0$  such that  $f(\mathbf{x}^*) \leq f(\mathbf{x})$  for all  $\mathbf{x} \in \{\mathbf{x} \in \mathcal{X} \mid \|\mathbf{x} - \mathbf{x}^*\| < \varepsilon\}$ ,
- (b) a *strict local minimum* if for some norm  $\|\cdot\|$  there exists an  $\varepsilon > 0$  such that

- $f(\mathbf{x}^*) < f(\mathbf{x})$  for all  $\mathbf{x} \in \{\mathbf{x} \in \mathcal{X} \mid \|\mathbf{x} - \mathbf{x}^*\| < \varepsilon, \mathbf{x} \neq \mathbf{x}^*\}$ ,
- (c) a *global minimum* if  $f(\mathbf{x}^*) \leq f(\mathbf{x})$  for all  $\mathbf{x} \in \mathcal{X}$ , and
- (d) a *strict global minimum* if  $f(\mathbf{x}^*) < f(\mathbf{x})$  for all  $\mathbf{x} \in \mathcal{X} \setminus \{\mathbf{x}^*\}$ .

Instead of the term *strict*, the term *unique* is occasionally used. Note that both  $\mathbf{x}^*$  and  $f(\mathbf{x}^*)$  are often referred to as minimum. Figure 1.10 shows different types of minima for a function with a scalar argument.

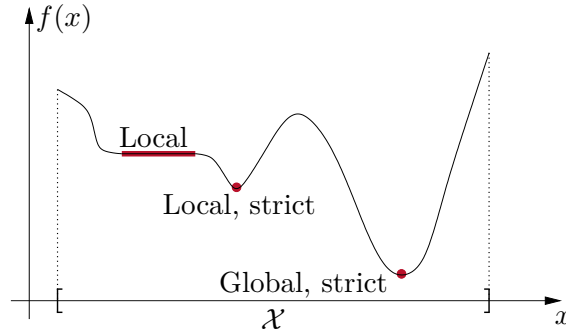


Figure 1.10: Different minima of a function  $f(x)$  with  $x \in \mathcal{X} \subset \mathbb{R}$ .

It is emphasized that a point  $\mathbf{x}^*$  which minimizes the function  $f(\mathbf{x})$  in the set  $\mathcal{X}$  must be contained in  $\mathcal{X}$ . However, a point  $\mathbf{x}$ , whose function value  $f(\mathbf{x})$  equals the infimum  $\inf\{f(\mathbf{x}) \mid \mathbf{x} \in \mathcal{X}\}$  does not have to exist, not even outside of  $\mathcal{X}$ .

The set of all minima (strictly speaking the set of all points where  $f(\mathbf{x})$  attains a minimum) is often written in the form

$$\mathcal{G} = \arg \min\{f(\mathbf{x}) \mid \mathbf{x} \in \mathcal{X}\} \quad (1.24)$$

where naturally  $\mathcal{G} \subseteq \mathcal{X}$  holds true and  $\mathcal{G}$  can either be empty or consist of a finite or infinite number of points. In the case of a strict global minimum in  $\mathcal{X}$ , the expression  $\bar{\mathbf{x}} = \arg \min\{f(\mathbf{x}) \mid \mathbf{x} \in \mathcal{X}\}$  usually refers to the function that returns the point  $\bar{\mathbf{x}} \in \mathcal{X}$  that globally minimizes the function  $f(\mathbf{x})$ .

### 1.3.2 Existence of Minima

Figure 1.11 shows three cases where a minimum does not exist. In Figure 1.11(a), the infimum of  $f(x)$  in the set  $\mathcal{X} = (\alpha, \beta)$  is given by  $f(\beta)$ . However, because  $\mathcal{X}$  is not closed and thus  $\beta \notin \mathcal{X}$ , there is no minimum in this case. In Figure 1.11(b), the left-sided limit  $\lim_{x \rightarrow \gamma^-} f(x)$  is the infimum of  $f(x)$  in the set  $\mathcal{X} = [\alpha, \beta]$ . In this case, a minimum does not exist due to the discontinuity of  $f(x)$ . In the last case, Figure 1.11(c), a minimum does not exist because  $f(x)$  is unbounded below in the unconstrained set  $\mathcal{X} = \{x \in \mathbb{R} \mid x \geq \alpha\}$ .

The following theorem states sufficient conditions for the existence of a solution of an optimization problem.

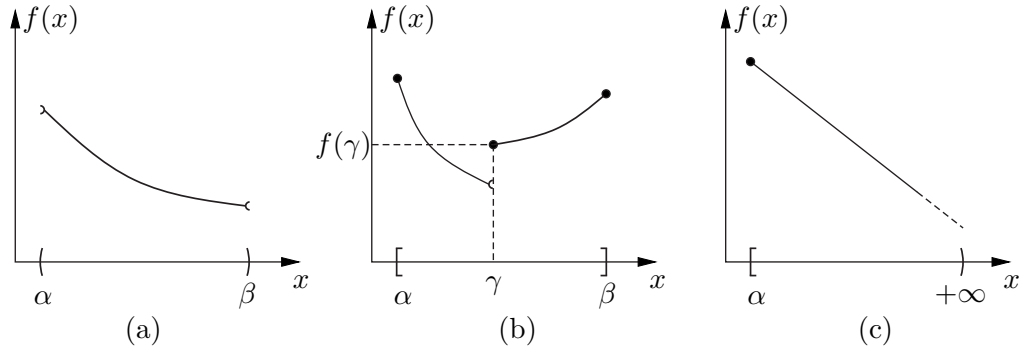


Figure 1.11: Non-existence of minima.

**Theorem 1.1 (Weierstrass extreme value theorem).** Let  $\mathcal{X}$  be a non-empty and compact (closed and bounded) set and  $f : \mathcal{X} \rightarrow \mathbb{R}$  be continuous on  $\mathcal{X}$ . Then the set of all minima  $\mathcal{G} = \arg \min\{f(\mathbf{x}) \mid \mathbf{x} \in \mathcal{X}\}$  is non-empty and compact.

The proof of this theorem can be found, for example, in [1.5, 1.6]. Theorem 1.1 describes a sufficient (but not necessary) condition for the existence of an optimal solution. To see that this condition is not necessary, consider the minimization problem  $\min_{x \in (-1,1)} x^2$ , where a minimum exists at  $x = 0$  although the set  $\mathcal{X} = (-1, 1)$  is open and thus not compact.

### 1.3.3 Gradient and Hessian

The calculation of first- and second-order derivatives of a cost function  $f(\mathbf{x})$  is of fundamental importance in optimization. To avoid problems that can arise with discontinuous functions or discontinuous derivatives (both numerical and theoretical), it is often assumed that all functions of an optimization problem are continuous and sufficiently often differentiable. Unless otherwise stated, this also applies to this course. The gradient and the Hessian play an important role in the context of optimization algorithms.

**Definition 1.3 (Gradient).** Let  $f : \mathcal{X} \rightarrow \mathbb{R}$  be a continuously differentiable function, i.e.,  $f \in C^1$ . Then

$$(\nabla f)(\mathbf{x}) = \left( \frac{\partial f}{\partial \mathbf{x}} \right)^T = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \vdots \\ \frac{\partial f}{\partial x_n} \end{bmatrix} \quad (1.25)$$

denotes the gradient (i.e., the first partial derivative) of  $f(\mathbf{x})$  at the point  $\mathbf{x} = [x_1 \ \dots \ x_n]^T$ .

**Definition 1.4 (Hessian).** Let  $f : \mathcal{X} \rightarrow \mathbb{R}$  be a twice continuously differentiable function, i.e.,  $f \in C^2$ . Then

$$(\nabla^2 f)(\mathbf{x}) = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \vdots & & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix} \quad (1.26)$$

denotes the Hessian or Hessian matrix (i.e., the second partial derivative) of  $f(\mathbf{x})$  at the point  $\mathbf{x} = [x_1 \ \dots \ x_n]^T$ .

In case of a function  $f(x)$  with only one scalar argument, the  $\nabla$  notation is often replaced by  $f'(x)$  and  $f''(x)$ .

Continuity of the second partial derivatives implies commutativity, i.e.,

$$\frac{\partial^2 f}{\partial x_i \partial x_j} = \frac{\partial^2 f}{\partial x_j \partial x_i} \quad (1.27)$$

or  $(\nabla^2 f)(\mathbf{x}) = (\nabla^2 f)^T(\mathbf{x})$ , meaning that the Hessian is symmetric. Consequently, the Hessian has purely real eigenvalues. In optimization, it is often important whether the Hessian is positive (semi-)definite. This property can be examined as follows.

**Theorem 1.2 (Definiteness of Matrices).** The definiteness of a symmetric matrix  $\mathbf{A} \in \mathbb{R}^{n \times n}$  can be characterized by the following conditions:

| Matrix $\mathbf{A}$ is | (a)<br>for all $\mathbf{p} \in \mathbb{R}^n$<br>with $\mathbf{p} \neq \mathbf{0}$ holds | (b)<br>all $n$ eigen-<br>values $\lambda_i$ are | (c)<br>for all $n$ leading principal<br>minors $D_i$ holds |
|------------------------|-----------------------------------------------------------------------------------------|-------------------------------------------------|------------------------------------------------------------|
| positive semidefinite: | $\mathbf{p}^T \mathbf{A} \mathbf{p} \geq 0$                                             | $\geq 0$                                        | -                                                          |
| positive definite:     | $\mathbf{p}^T \mathbf{A} \mathbf{p} > 0$                                                | $> 0$                                           | $D_i > 0$                                                  |
| negative semidefinite: | $\mathbf{p}^T \mathbf{A} \mathbf{p} \leq 0$                                             | $\leq 0$                                        | -                                                          |
| negative definite:     | $\mathbf{p}^T \mathbf{A} \mathbf{p} < 0$                                                | $< 0$                                           | $(-1)^{i+1} D_i < 0$                                       |

The eigenvalues  $\lambda_i$ ,  $i = 1, \dots, n$  of the matrix  $\mathbf{A}$  are the solutions of the equation

$$\det(\lambda \mathbf{E} - \mathbf{A}) = 0, \quad (1.28)$$

where  $\mathbf{E}$  represents the identity matrix of dimension  $n$ . The leading principal minors  $D_i$  are the determinants of the upper left submatrices of  $\mathbf{A}$ ,

$$D_1 = a_{11}, \quad D_2 = \det \begin{pmatrix} a_{11} & a_{12} \\ a_{12} & a_{22} \end{pmatrix}, \quad \dots, \quad D_n = \det \begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{1n} & \cdots & a_{nn} \end{pmatrix}, \quad (1.29)$$

where  $a_{ij}$  denotes the element row  $i$  and column  $j$  of  $\mathbf{A}$ , i.e.,  $\mathbf{A} = [a_{ij}]_{i,j=1,\dots,n}$ . To determine the definiteness of a symmetric matrix  $\mathbf{A}$ , only one of the three conditions

(a)–(c) in Theorem 1.2 needs to be evaluated. Each of them is necessary and sufficient. Criterion (c) is also called the *Sylvester criterion* and cannot be used for semidefinite matrices.

The positive definiteness of a symmetric matrix  $\mathbf{A}$  can also be checked using the *Cholesky factorization* as an alternative to the conditions (a)–(c) from Theorem 1.2. According to this method, a symmetric matrix  $\mathbf{A}$  is positive definite if and only if it can be factorized in the form  $\mathbf{A} = \mathbf{G}\mathbf{G}^T$ , where  $\mathbf{G}$  is a lower triangular matrix with positive diagonal entries.

The gradient and the Hessian are often used in the context of the *mean value theorem* (*Taylor's theorem*) to approximate functions.

**Theorem 1.3 (Mean value theorem, Taylor's theorem).** *Let  $f(\mathbf{x})$  be a continuously differentiable function, i.e.,  $f \in C^1$ , on a set  $\mathcal{X}$  containing the line segment  $[\mathbf{x}_1, \mathbf{x}_2]$ , then there exists a real number  $\alpha$  with  $0 \leq \alpha \leq 1$  such that*

$$f(\mathbf{x}_2) = f(\mathbf{x}_1) + (\mathbf{x}_2 - \mathbf{x}_1)^T (\nabla f)(\mathbf{x}_1 + \alpha(\mathbf{x}_2 - \mathbf{x}_1)) . \quad (1.30)$$

*If the function  $f(\mathbf{x})$  is twice continuously differentiable, i.e.,  $f \in C^2$ , then there exists a real number  $\alpha$  with  $0 \leq \alpha \leq 1$  such that*

$$f(\mathbf{x}_2) = f(\mathbf{x}_1) + (\mathbf{x}_2 - \mathbf{x}_1)^T (\nabla f)(\mathbf{x}_1) + \frac{1}{2} (\mathbf{x}_2 - \mathbf{x}_1)^T (\nabla^2 f)(\mathbf{x}_1 + \alpha(\mathbf{x}_2 - \mathbf{x}_1)) (\mathbf{x}_2 - \mathbf{x}_1) . \quad (1.31)$$

### 1.3.4 Calculation of Derivatives

Various methods can be used to actually calculate derivatives.

#### Analytic differentiation

If  $f(\mathbf{x})$  is given as an analytic expression, derivatives can be calculated directly using analytic differentiation.

#### Algorithmic differentiation

In optimization, the cost function  $f(\mathbf{x})$  is often not available as a closed analytic expression but in the form of functions (algorithms) implemented in a computer program. In this case, *algorithmic differentiation*, sometimes also called *automatic differentiation*, offers a convenient way of calculating derivatives. The method uses the rules of analytic differentiation (differentiation rules for elementary functions, chain rule, product rule, sum rule, quotient rule, differentiation rule for inverse functions, etc.) to create a new computer program that calculates the desired derivatives of  $f(\mathbf{x})$ . The program steps are organized in such a way that an efficient and at the same time accurate calculation of the derivatives is achieved. More information about algorithmic differentiation can be found, for example, at <http://www.autodiff.org> or in [1.7].

### Derivative calculation with difference quotients

For an approximate numerical calculation of derivatives, difference quotients can be computed [1.8]. For a general, sufficiently often differentiable function  $f(\mathbf{x})$ , examples for difference quotients are given in Table 1.1. Here,  $h$  is the step size and  $\mathbf{e}_i$  is the unit vector with entry 1 at position  $i$ . The table also contains the orders of the errors that can occur in this approximate calculation of derivatives. Possible errors are the *truncation error* and the *round-off error*, where  $e_r$  is the maximum relative error due to rounding operations in floating-point arithmetic.

| Derivative,<br>direction                | Formula                                                                                                                                                                                                                                                     | Truncation<br>error | Round-<br>off error       |
|-----------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------|---------------------------|
| 1 <sup>st</sup> derivative,<br>forward  | $(\nabla f)(\mathbf{x}) \approx \left[ \frac{f(\mathbf{x} + h\mathbf{e}_i) - f(\mathbf{x})}{h} \right]_{i=1,\dots,n}$                                                                                                                                       | $\mathcal{O}(h)$    | $\mathcal{O}(e_r h^{-1})$ |
| 1 <sup>st</sup> derivative,<br>backward | $(\nabla f)(\mathbf{x}) \approx \left[ \frac{f(\mathbf{x}) - f(\mathbf{x} - h\mathbf{e}_i)}{h} \right]_{i=1,\dots,n}$                                                                                                                                       | $\mathcal{O}(h)$    | $\mathcal{O}(e_r h^{-1})$ |
| 1 <sup>st</sup> derivative,<br>central  | $(\nabla f)(\mathbf{x}) \approx \left[ \frac{f(\mathbf{x} + h\mathbf{e}_i) - f(\mathbf{x} - h\mathbf{e}_i)}{2h} \right]_{i=1,\dots,n}$                                                                                                                      | $\mathcal{O}(h^2)$  | $\mathcal{O}(e_r h^{-1})$ |
| 2 <sup>nd</sup> derivative,<br>forward  | $(\nabla^2 f)(\mathbf{x}) \approx [f(\mathbf{x} + h\mathbf{e}_i + h\mathbf{e}_j) - f(\mathbf{x} + h\mathbf{e}_i) - f(\mathbf{x} + h\mathbf{e}_j) + f(\mathbf{x})]_{i,j=1,\dots,n} / h^2$                                                                    | $\mathcal{O}(h)$    | $\mathcal{O}(e_r h^{-2})$ |
| 2 <sup>nd</sup> derivative,<br>central  | $(\nabla^2 f)(\mathbf{x}) \approx [f(\mathbf{x} + h\mathbf{e}_i + h\mathbf{e}_j) - f(\mathbf{x} + h\mathbf{e}_i - h\mathbf{e}_j) - f(\mathbf{x} - h\mathbf{e}_i + h\mathbf{e}_j) + f(\mathbf{x} - h\mathbf{e}_i - h\mathbf{e}_j)]_{i,j=1,\dots,n} / (4h^2)$ | $\mathcal{O}(h^2)$  | $\mathcal{O}(e_r h^{-2})$ |

Table 1.1: Difference quotients.

The deduction of difference quotients and the calculation of the associated truncation errors can be done using Taylor series expansion at the point  $\mathbf{x}$  or the mean value theorem 1.3 [1.9]. The truncation error occurs because the Taylor series expansion is truncated at a certain order and a finite step size  $h > 0$  must be used. Thus, the truncation error is inherent to the method and occurs even with an exact calculation of the function value  $f(\mathbf{x})$ . The round-off error is due to the practically non-exact numerical calculation of  $f(\mathbf{x})$ ,  $f(\mathbf{x} \pm h\mathbf{e}_i)$  and  $f(\mathbf{x} \pm h\mathbf{e}_i \pm h\mathbf{e}_j)$  when using floating-point arithmetic. As the following example shows, the round-off error can grow unbounded for  $h \rightarrow 0$ .

*Example 1.6.* As an example, we now derive and calculate the error of the forward difference quotient to approximate the gradient (first row in Table 1.1). taking into account the mean value theorem 1.3, a Taylor series expansion of  $f(\mathbf{x})$  yields

$$f(\mathbf{x} + h\mathbf{e}_i) = f(\mathbf{x}) + h \frac{\partial f}{\partial x_i} \Big|_{\mathbf{x}} + \frac{h^2}{2} \frac{\partial^2 f}{\partial x_i^2} \Big|_{\mathbf{x} + \alpha h \mathbf{e}_i} \quad (1.32)$$

with  $\alpha \in (0, 1)$ . This implies

$$\frac{\partial f}{\partial x_i} \Big|_{\mathbf{x}} = \frac{f(\mathbf{x} + h\mathbf{e}_i) - f(\mathbf{x})}{h} - \underbrace{\frac{h}{2} \frac{\partial^2 f}{\partial x_i^2} \Big|_{\mathbf{x} + \alpha h \mathbf{e}_i}}_{\text{Truncation error, } \mathcal{O}(h)}. \quad (1.33)$$

Now, assuming that  $e_r$  is the maximum relative error due to the rounding operations required in floating-point arithmetic when evaluating the functions  $f(\mathbf{x} + h\mathbf{e}_i)$  and  $f(\mathbf{x})$ , the calculated value for the approximation of  $\partial f / \partial x_i|_{\mathbf{x}}$  is

$$\begin{aligned} & \frac{f(\mathbf{x} + h\mathbf{e}_i)(1 + e_r) - f(\mathbf{x})(1 - e_r)}{h} \\ &= \frac{f(\mathbf{x} + h\mathbf{e}_i) - f(\mathbf{x})}{h} + \underbrace{\frac{(f(\mathbf{x} + h\mathbf{e}_i) + f(\mathbf{x}))e_r}{h}}_{\text{Round-off error, } \mathcal{O}(e_r h^{-1})} \end{aligned} \quad (1.34)$$

in the worst case. The total error is therefore

$$\begin{aligned} & \underbrace{\frac{f(\mathbf{x} + h\mathbf{e}_i) - f(\mathbf{x})}{h}}_{\text{Calculated value}} + \underbrace{\frac{(f(\mathbf{x} + h\mathbf{e}_i) + f(\mathbf{x}))e_r}{h}}_{\text{Round-off error}} - \underbrace{\frac{\partial f}{\partial x_i} \Big|_{\mathbf{x}}}_{\text{Exact value}} \\ &= \frac{h}{2} \frac{\partial^2 f}{\partial x_i^2} \Big|_{\mathbf{x} + \alpha h \mathbf{e}_i} + \frac{(f(\mathbf{x} + h\mathbf{e}_i) + f(\mathbf{x}))e_r}{h}. \end{aligned} \quad (1.35)$$

Figure 1.12 shows for this case how the absolute values of truncation, round-off, and total error depend on  $h$ . So there is an optimal step size  $h$  to minimize the total error.

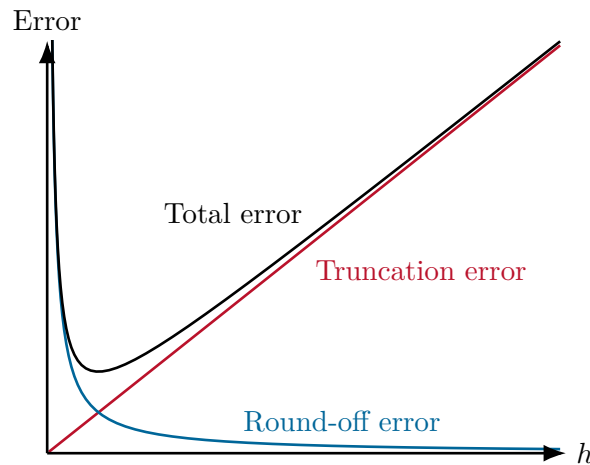


Figure 1.12: Absolute values of the errors in the approximation of the gradient by the forward difference quotient as a function of the step size  $h$ .

In Table 1.1 and Example 1.6, only the round-off error due to floating-point arithmetic in the evaluation of the function  $f$  was considered. In fact, such rounding errors also occur in the calculation of the difference quotients themselves, i.e., for example, in the computation of

$$\frac{f(\mathbf{x} + h\mathbf{e}_i) - f(\mathbf{x})}{h}. \quad (1.36)$$

When calculating the difference in the numerator of this expression, a considerable loss of accuracy due to *cancellation errors* can occur [1.10]. When calculating with floating-point arithmetic, all input quantities, intermediate results of involved elementary mathematical operations, and final results must be rounded to the number of mantissa digits. *Cancellation* refers to the annihilation of leading mantissa digits when subtracting two (similar) numbers [1.11]. Assuming that calculations are performed with  $M$  mantissa digits and two numbers to be subtracted match in  $m$  leading mantissa digits, then only  $M - m$  valid non-zero mantissa digits remain in the result, which directly limits the achievable accuracy. Thus, the more similar two numbers to be subtracted are, the smaller  $M - m$  becomes, which can lead to useless results when calculating difference quotients in the case  $h \rightarrow 0$ .

*Example 1.7 (Cancellation error).* Consider the calculation

$$0.123\,456 - 0.123\,455 = 0.000\,001. \quad (1.37)$$

If only the last digit in the two input quantities is uncertain due to previous rounding errors, the result 0.000 001 can be completely wrong, i.e., it may not have a single valid non-zero digit.

In addition to the error orders, rounding errors, and cancellation errors, the respective computational effort also plays a role in the choice of a difference quotient. For the approximate calculation of  $(\nabla f)(\mathbf{x})$ , one-sided difference quotients require  $n + 1$  evaluations of the function  $f$  and the central difference quotient requires  $2n$  such evaluations. In other

words, for the approximate calculation of  $(\nabla f)(\mathbf{x})$ , the computational effort increases with the order  $\mathcal{O}(n)$ . For the approximate calculation of  $(\nabla^2 f)(\mathbf{x})$ , the computational effort increases with the order  $\mathcal{O}(n^2)$ .

### Derivative calculation using complex function evaluation

The following method for approximate derivative calculation is known as the *complex step derivative* [1.12, 1.13]. This numerical method is only applicable to *holomorphic* functions, which are also called *complex differentiable*.

**Definition 1.5 (Holomorphic function).** A function  $f : \mathcal{X} \rightarrow \mathbb{C}$  with  $\mathcal{X} \subseteq \mathbb{C}$  is called *holomorphic* if the limit

$$\lim_{h \rightarrow 0} \frac{f(z+h) - f(z)}{h} \quad \forall z \in \mathcal{X} \quad (1.38)$$

with  $h \in \mathbb{C}$  exists.

A function  $f : \mathcal{X} \rightarrow \mathbb{C}$  of several variables, i.e.,  $\mathcal{X} \subseteq \mathbb{C}^n$ , is holomorphic if and only if it is holomorphic with respect to each single variable while the other variables are held constant.

The first derivative of a holomorphic real function  $f(\mathbf{x})$ , i.e.,  $f : \mathbb{R}^n \rightarrow \mathbb{R}$ , can be approximately calculated with the formula

$$(\nabla f)(\mathbf{x}) \approx \left[ \frac{\operatorname{Im}(f(\mathbf{x} + h\mathbf{I}e_i))}{h} \right]_{i=1, \dots, n}, \quad (1.39)$$

where  $h > 0$  is a small real step size. To derive (1.39),  $f(\mathbf{x})$  is first expanded into a Taylor series

$$f(\mathbf{x} + h\mathbf{I}e_i) = f(\mathbf{x}) + h\mathbf{I} \left. \frac{\partial f}{\partial x_i} \right|_{\mathbf{x}} - \frac{h^2}{2} \left. \frac{\partial^2 f}{\partial x_i^2} \right|_{\mathbf{x}} - \frac{h^3}{6} \left. \frac{\partial^3 f}{\partial x_i^3} \right|_{\mathbf{x} + \alpha h\mathbf{I}e_i} \quad (1.40)$$

with  $\alpha \in (0, 1)$ . Dividing the imaginary part of (1.40) by  $h$ , we obtain

$$\left. \frac{\partial f}{\partial x_i} \right|_{\mathbf{x}} = \frac{\operatorname{Im}(f(\mathbf{x} + h\mathbf{I}e_i))}{h} + \underbrace{\frac{h^2}{6} \left. \frac{\partial^3 f}{\partial x_i^3} \right|_{\mathbf{x} + \alpha h\mathbf{I}e_i}}_{\text{Truncation error, } \mathcal{O}(h^2)} \quad (1.41)$$

and thus directly the components of (1.39). In [1.14], similar formulas for the approximate calculation of second derivatives using complex function evaluation are derived.

Thus,  $(\nabla f)(\mathbf{x})$  can be approximately calculated with only  $n$  complex evaluations of the function  $f$ , whereby a truncation error of order  $\mathcal{O}(h^2)$  is achieved. The round-off error is of order  $\mathcal{O}(e_r h^{-1})$ . A key advantage of this method over derivative calculation with difference quotients is the absence of a cancellation error when calculating the first derivative (no subtraction of similar numbers required). Hence,  $h$  can be chosen very small. This is generally not the case for higher derivatives. The disadvantage of this method is the slightly higher numerical effort because complex numbers have to be used. In [1.13], the relationship between this method and algorithmic differentiation is discussed.

From the real part of (1.40),

$$f(\mathbf{x}) = \operatorname{Re}(f(\mathbf{x} + h\mathbf{e}_i)) + \mathcal{O}(h^2) \quad (1.42)$$

can be obtained. So if a derivative  $\partial f / \partial x_i$  is calculated, we also get an approximation for the value  $f(\mathbf{x})$  without additional function evaluations.

### 1.3.5 Implicit Function Theorem

**Theorem 1.4 (Implicit function theorem).** Let  $\mathbf{f} : \mathbb{R}^n \times \mathbb{R}^m \rightarrow \mathbb{R}^m$  be a continuously differentiable function. If at a point  $(\bar{\mathbf{x}}, \bar{\mathbf{y}}) \in \mathbb{R}^n \times \mathbb{R}^m$

$$\mathbf{f}(\bar{\mathbf{x}}, \bar{\mathbf{y}}) = \mathbf{0} \quad (1.43)$$

and

$$\operatorname{rank} \left( \left. \frac{\partial \mathbf{f}(\bar{\mathbf{x}}, \mathbf{y})}{\partial \mathbf{y}} \right|_{\mathbf{y}=\bar{\mathbf{y}}} \right) = m \quad (1.44)$$

are satisfied, there exists an open neighborhood  $\mathcal{X} \subseteq \mathbb{R}^n$  of  $\bar{\mathbf{x}}$ , an open neighborhood  $\mathcal{Y} \subseteq \mathbb{R}^m$  of  $\bar{\mathbf{y}}$ , and a continuously differentiable function  $\mathbf{g} : \mathcal{X} \rightarrow \mathcal{Y}$  such that

$$\mathbf{f}(\mathbf{x}, \mathbf{y}) = \mathbf{0} \Leftrightarrow \mathbf{y} = \mathbf{g}(\mathbf{x}) \quad \forall (\mathbf{x}, \mathbf{y}) \in \mathcal{X} \times \mathcal{Y} \quad (1.45)$$

holds true.

Therefore,  $\mathbf{f}(\mathbf{x}, \mathbf{g}(\mathbf{x})) = \mathbf{0}$  is satisfied for all  $\mathbf{x} \in \mathcal{X}$ . For a given point  $\mathbf{x}$ , the unique value  $\mathbf{g}(\mathbf{x})$  (without explicit knowledge of the function  $\mathbf{g}(\mathbf{x})$ ) can be computed by numerically solving the equation

$$\mathbf{f}(\mathbf{x}, \mathbf{y}) = \mathbf{0} \quad (1.46)$$

for  $\mathbf{y}$ . The Jacobian matrix  $(\nabla \mathbf{g})(\mathbf{x}) \in \mathbb{R}^{n \times m}$  can be computed (without explicit knowledge of the function  $\mathbf{g}(\mathbf{x})$ ) by *implicit differentiation* in the form

$$(\nabla \mathbf{g})(\mathbf{x}) = - \left( \frac{\partial \mathbf{f}(\mathbf{x}, \mathbf{g}(\mathbf{x}))}{\partial \mathbf{x}} \right)^T \left( \left. \frac{\partial \mathbf{f}(\mathbf{x}, \mathbf{y})}{\partial \mathbf{y}} \right|_{\mathbf{y}=\mathbf{g}(\mathbf{x})} \right)^{-T}. \quad (1.47)$$

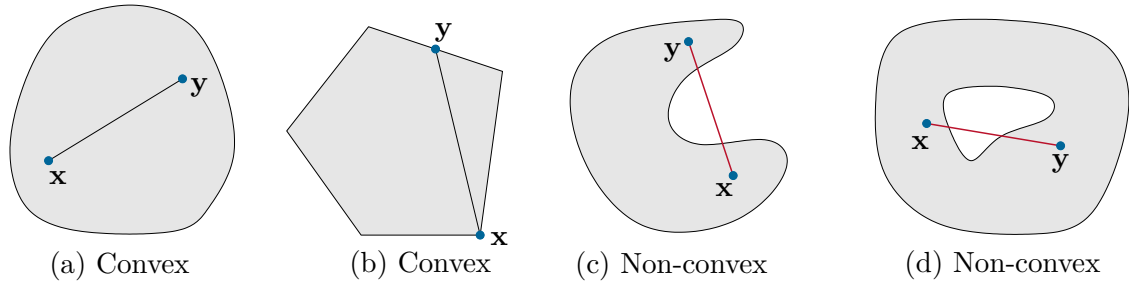
### 1.3.6 Convexity

The property of convexity is of great importance in optimization and often allows for a simple (numerical) solution of an optimization problem. The term *convex* can be applied to both sets and functions.

#### 1.3.6.1 Convex Sets

**Definition 1.6 (Convex set).** A set  $\mathcal{X} \subseteq \mathbb{R}^n$  is called *convex* if for all  $\mathbf{x}, \mathbf{y} \in \mathcal{X}$  and all real numbers  $\alpha$  with  $0 < \alpha < 1$  the following holds

$$(\alpha \mathbf{x} + (1 - \alpha) \mathbf{y}) \in \mathcal{X}. \quad (1.48)$$

Figure 1.13: Examples of convex and non-convex sets in  $\mathbb{R}^2$ .

A geometric interpretation of this definition is that a set  $\mathcal{X} \subseteq \mathbb{R}^n$  is convex if and only if the connecting line between any two points  $\mathbf{x}, \mathbf{y} \in \mathcal{X}$  is completely contained in  $\mathcal{X}$ . Figure 1.13 shows some examples of convex and non-convex sets for the space  $\mathbb{R}^2$ .

Convex sets have the following properties:

- (a) The *intersection* of convex sets is a convex set.
- (b) If  $\mathcal{X} \subseteq \mathbb{R}^n$  is a convex set,  $\mathbf{A} \in \mathbb{R}^{m \times n}$  is a fixed matrix and  $\mathbf{b} \in \mathbb{R}^m$  is a fixed vector, then the set

$$\{\mathbf{Ax} + \mathbf{b} \mid \mathbf{x} \in \mathcal{X}\} \subseteq \mathbb{R}^m \quad (1.49)$$

is also convex. In other words, the image of a convex set under an *affine transformation* is convex.

- (c) If  $\mathcal{X}$  and  $\mathcal{Y}$  are convex sets, then the set

$$\{\mathbf{x} + \mathbf{y} \mid \mathbf{x} \in \mathcal{X}, \mathbf{y} \in \mathcal{Y}\} \quad (1.50)$$

is also convex.

These properties are important for characterizing the convexity of the feasible set  $\mathcal{X}$  of an optimization problem.

### 1.3.6.2 Convex Functions

**Definition 1.7 (Convex and concave functions).** Let  $\mathcal{X} \subseteq \mathbb{R}^n$  be a convex set. The function  $f : \mathcal{X} \rightarrow \mathbb{R}$  is called *convex* on  $\mathcal{X}$  if for all  $\mathbf{x}, \mathbf{y} \in \mathcal{X}$  and all real numbers  $\alpha$  with  $0 \leq \alpha \leq 1$

$$f(\alpha \mathbf{x} + (1 - \alpha) \mathbf{y}) \leq \alpha f(\mathbf{x}) + (1 - \alpha) f(\mathbf{y}) \quad (1.51)$$

holds true. The function  $f$  is called *strictly convex* if for all  $\alpha$  with  $0 < \alpha < 1$  and  $\mathbf{x} \neq \mathbf{y}$

$$f(\alpha \mathbf{x} + (1 - \alpha) \mathbf{y}) < \alpha f(\mathbf{x}) + (1 - \alpha) f(\mathbf{y}) \quad (1.52)$$

holds true. The function  $f$  is called (*strictly*) *concave* if  $-f$  is (strictly) convex.

Definition 1.7 can also be geometrically interpreted: A function  $f$  is convex (concave) if for all  $\mathbf{x} \in \mathcal{X}$ ,  $\mathbf{y} \in \mathcal{X}$  and  $0 < \alpha < 1$  all function values  $f(\alpha \mathbf{x} + (1 - \alpha) \mathbf{y})$  along the axis  $f$  are below (above) or on the connecting line between  $(\mathbf{x}, f(\mathbf{x}))$  and  $(\mathbf{y}, f(\mathbf{y}))$ . Figure 1.14

shows some examples of convex and concave functions for the scalar case. It thus follows that affine functions are both concave and convex.

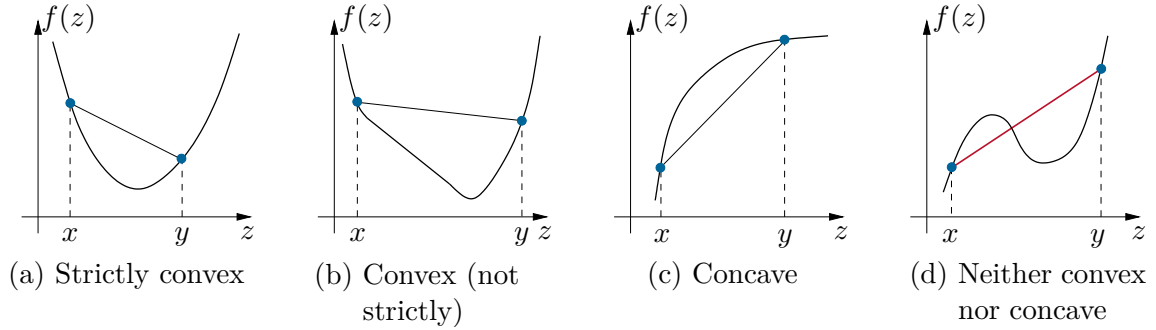


Figure 1.14: Examples of convex and concave functions.

Convex functions have the following properties:

- (a) The sum

$$f(\mathbf{x}) = \sum_{i=1}^k a_i f_i(\mathbf{x}) \quad (1.53)$$

of convex functions  $f_i(\mathbf{x})$ ,  $i = 1, \dots, k$  on the convex set  $\mathcal{X}$  with real coefficients  $a_i \geq 0$ ,  $i = 1, \dots, k$  is also convex on  $\mathcal{X}$ .

- (b) If the function  $f(\mathbf{y})$  is convex on the convex set  $\mathcal{Y} \subseteq \mathbb{R}^m$  and there exist a convex set  $\mathcal{X} \subseteq \mathbb{R}^n$ , a fixed matrix  $\mathbf{A} \in \mathbb{R}^{m \times n}$ , and a fixed vector  $\mathbf{b} \in \mathbb{R}^m$  such that

$$\{\mathbf{Ax} + \mathbf{b} \mid \mathbf{x} \in \mathcal{X}\} \subseteq \mathcal{Y}, \quad (1.54)$$

then the function

$$\tilde{f}(\mathbf{x}) = f(\mathbf{Ax} + \mathbf{b}) \quad (1.55)$$

is convex on  $\mathcal{X}$ .

- (c) If the function  $f(\mathbf{x})$  is convex on the convex set  $\mathcal{X}$ , the set

$$\{\mathbf{x} \in \mathcal{X} \mid f(\mathbf{x}) \leq c\} \quad (1.56)$$

is convex for all values  $c \in \mathbb{R}$ , see Figure 1.15.

- (d) A continuously differentiable function  $f \in C^1$  is convex on the convex set  $\mathcal{X}$  if and only if the inequality

$$f(\mathbf{y}) \geq f(\mathbf{x}) + (\mathbf{y} - \mathbf{x})^T (\nabla f)(\mathbf{x}) \quad (1.57)$$

holds true for all  $\mathbf{x}, \mathbf{y} \in \mathcal{X}$ . The geometric interpretation of the inequality (1.57) is that at each point  $\mathbf{x}$  of a convex function  $f(\mathbf{x})$  there exists a so-called *supporting hyperplane* (scalar case: *supporting tangent line*), above or on which  $f(\mathbf{x})$  passes. This is illustrated in Figure 1.16.

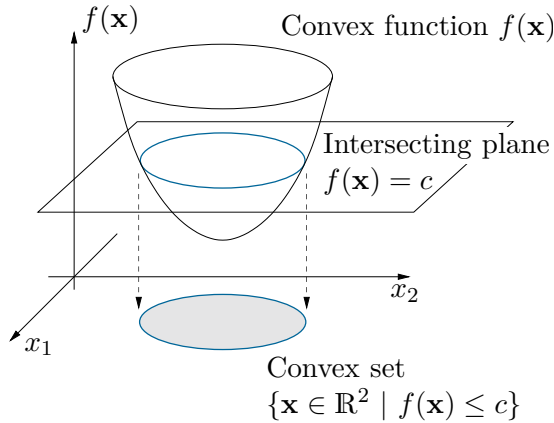


Fig. 1.15: Convex set resulting from the intersection of a convex function  $f(\mathbf{x})$  with the plane  $f(\mathbf{x}) = \text{const.}$

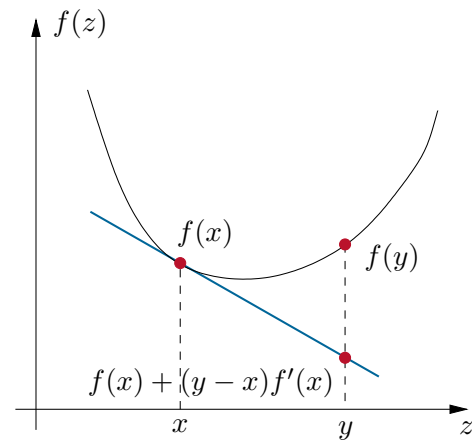


Fig. 1.16: Supporting tangent line of a convex function  $f(z)$ .

- (e) A twice continuously differentiable function  $f \in C^2$  is convex on the convex set  $\mathcal{X}$  if and only if the Hessian  $(\nabla^2 f)(\mathbf{x})$  is positive semidefinite for all  $\mathbf{x} \in \mathcal{X}$ . If the Hessian  $(\nabla^2 f)(\mathbf{x})$  is positive definite, it follows that the function  $f(\mathbf{x})$  is strictly convex. However, the converse of this statement is invalid, which can be inferred from the function  $f(x) = x^4$ . This function is strictly convex but its Hessian at  $x = 0$  is zero.

**Exercise 1.1.** Prove the properties (a)–(e) of convex functions. For the proof of property (e), use the mean value theorem, see Theorem 1.3, in particular (1.31).

**Exercise 1.2.** Show that the function  $f(\mathbf{x}) = x_1^4 + x_1^2 - 2x_1x_2 + x_2^2$  with  $\mathbf{x} = [x_1 \ x_2]^T \in \mathbb{R}^2$  is convex on the entire domain  $\mathbb{R}^2$ .

## 1.4 References

- [1.1] A. E. Bryson, Jr., *Dynamic Optimization*. Addison-Wesley, 1999.
- [1.2] R. H. Goddard, “A method of reaching extreme altitudes,” *Smithsonian Miscellaneous Collections*, vol. 71, no. 2, 1919.
- [1.3] K. Pohmer, *Mikroökonomische Theorie der personellen Einkommens- und Vermögensverteilung* (Studies in Contemporary Economics). Springer, 1985, vol. 16.
- [1.4] H. J. Oberle and R. Rosendahl, “Numerical computation of a singular-state subarc in an economic optimal control model,” *Optimal Control Applications and Methods*, vol. 27, no. 4, pp. 211–235, 2006.
- [1.5] M. Bazaraa, H. Sherali, and C. Shetty, *Nonlinear Programming: Theory and Algorithms*, 3rd ed. John Wiley & Sons, 2006.
- [1.6] I. Griva, S. Nash, and A. Sofer, *Linear and Nonlinear Optimization*, 2nd ed. Society for Industrial and Applied Mathematics, 2009.
- [1.7] A. Griewank and A. Walther, *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation* (Other Titles in Applied Mathematics), 2nd ed. Society for Industrial and Applied Mathematics, 2008.
- [1.8] D. Lynch, *Numerical Partial Differential Equations for Environmental Scientists and Engineers - A First Practical Course*. New York: Springer, 2005.
- [1.9] M. Hanke-Bourgeois, *Grundlagen der Numerischen Mathematik und des Wissenschaftlichen Rechnens*, 3rd ed. Vieweg+Teubner, 2009.
- [1.10] P. E. Gill, W. Murray, and M. H. Wright, *Practical Optimization*. Academic Press, 1981.
- [1.11] H.R. Schwarz and N. Köckler, *Numerische Mathematik*, 8th ed. Wiesbaden: Vieweg+Teubner, 2011.
- [1.12] J. Lyness and C. Moler, “Numerical differentiation of analytic functions,” *SIAM Journal on Numerical Analysis*, vol. 4, no. 2, pp. 202–210, 1967.
- [1.13] J. Martins, P. Sturdza, and J. Alonso, “The complex-step derivative approximation,” *ACM Transactions on Mathematical Software*, vol. 29, no. 3, pp. 245–262, 2003.
- [1.14] R. Abreu, “Complex steps finite differences with applications to seismic problems,” Ph.D. dissertation, University of Granada, Granada, Spanien, 2013.
- [1.15] J. Nocedal and S. J. Wright, *Numerical Optimization* (Springer Series in Operations Research and Financial Engineering), 2nd ed. Springer, 2006.
- [1.16] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004.
- [1.17] M. Papageorgiou, M. Leibold, and M. Buss, *Optimierung: Statische, dynamische, stochastische Verfahren für die Anwendung*, 4th ed. Springer, 2015.
- [1.18] D. P. Bertsekas, *Nonlinear Programming*, 2nd ed. Athena Scientific, 1999.

- [1.19] D. G. Luenberger and Y. Ye, *Linear and Nonlinear Programming* (International Series in Operations Research & Management Science), 3rd ed. Springer, 2008, vol. 116.
- [1.20] B. C. Chachuat, “Nonlinear and Dynamic Optimization: From Theory to Practice,” abrufbar unter <http://infoscience.epfl.ch/record/111939>, Laboratoire d’Automatique, École Polytechnique Fédérale de Lausanne, 2007. Accessed: Sep. 30, 2020.

## 2 Static Optimization: Unconstrained Case

In this chapter, unconstrained static optimization problems of the form

$$\min_{\mathbf{x} \in \mathbb{R}^n} f(\mathbf{x}) \quad (2.1)$$

are considered. Sections 2.2 to 2.6 cover numerical methods for solving such optimization problems. In the following section, optimality conditions are formulated for the problem (2.1).

### 2.1 Optimality Conditions

**Theorem 2.1** (First-order necessary optimality condition). *Let  $f \in C^1$  be a function defined on  $\mathbb{R}^n$ . If  $\mathbf{x}^*$  is a local minimum of  $f$  on  $\mathbb{R}^n$ , then*

$$(\nabla f)(\mathbf{x}^*) = \mathbf{0}. \quad (2.2)$$

*Proof.* Consider the function  $g(\alpha) = f(\mathbf{x}^* + \alpha \mathbf{d})$  with an arbitrary direction  $\mathbf{d} \in \mathbb{R}^n$  and  $\alpha \geq 0$ . For any value  $\alpha \geq 0$ , we have  $\mathbf{x}^* + \alpha \mathbf{d} \in \mathbb{R}^n$ , so the function  $g(\alpha)$  is well-defined. This function must have a local minimum at the point  $\alpha = 0$ . Expanding  $g(\alpha)$  into a Taylor series around the point  $\alpha = 0$  and truncating after the linear term, we get

$$g(\alpha) = g(0) + g'(0)\alpha + \mathcal{O}(\alpha^2) \quad (2.3)$$

with  $g'(0) = \mathbf{d}^T (\nabla f)(\mathbf{x}^*)$ . The remainder  $\mathcal{O}(\alpha^2)$  decays quadratically to zero, i.e., faster than the linear term  $g'(0)\alpha$ . If  $g'(0) < 0$ , then for sufficiently small  $\alpha > 0$ , we would have  $g(\alpha) - g(0) < 0$ , which contradicts the assumption that  $\alpha = 0$  and thus  $\mathbf{x}^*$  is a minimum. Therefore,  $g'(0) = \mathbf{d}^T (\nabla f)(\mathbf{x}^*) \geq 0$  must be satisfied. However, this can only be the case for arbitrary  $\mathbf{d} \in \mathbb{R}^n$  if  $(\nabla f)(\mathbf{x}^*) = \mathbf{0}$ .  $\square$

*Example 2.1.* Consider the optimization problem

$$\min_{\mathbf{x} \in \mathbb{R}^2} f(x_1, x_2) = x_1^2 - x_1 x_2 + x_2^2 - 3x_2. \quad (2.4)$$

From the first-order necessary optimality condition according to (2.2)

$$2x_1 - x_2 = 0 \quad (2.5a)$$

$$-x_1 + 2x_2 = 3, \quad (2.5b)$$

follows the solution  $\mathbf{x}^* = [1 \ 2]^T$ . It can be shown with the help of the following Theorem 2.3 that  $\mathbf{x}^*$  in this case is even a strict global minimum.

The optimality condition according to Theorem 2.1 is necessary but not sufficient. The condition only states that the point in question is an *extremal point* (also called a *stationary point*) and is satisfied equally by a minimum, maximum, or saddle point, see Figure 2.1.

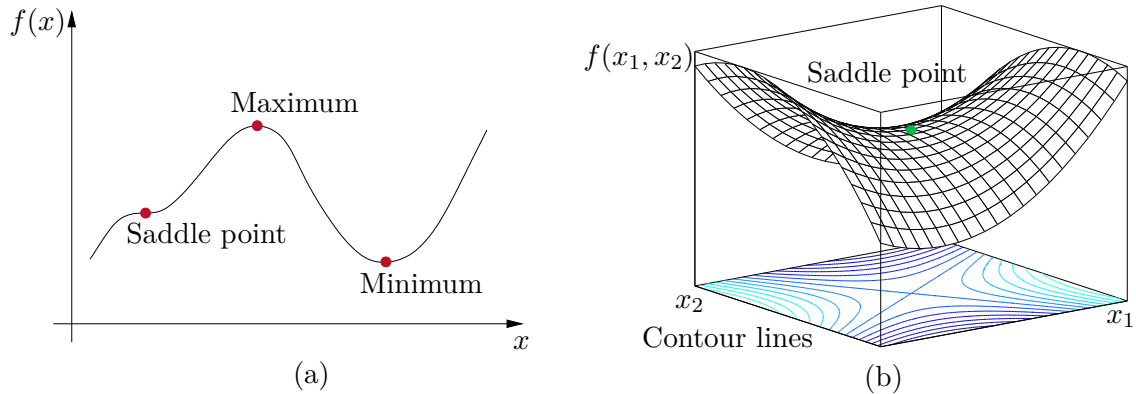


Figure 2.1: Examples of stationary points in one and two dimensions.

**Theorem 2.2 (Second-order necessary optimality conditions).** Let  $f \in C^2$  be a function defined on  $\mathbb{R}^n$ . If  $\mathbf{x}^*$  is a local minimum of  $f$  on  $\mathbb{R}^n$ , the conditions

$$(a) \quad (\nabla f)(\mathbf{x}^*) = \mathbf{0} \quad (2.6a)$$

$$(b) \quad (\nabla^2 f)(\mathbf{x}^*) \text{ is positive semi-definite} \quad (2.6b)$$

hold true.

**Exercise 2.1.** Prove Theorem 2.2. **Hint:** Use the proof of Theorem 2.1 as a guide.

Also Theorem 2.2 only describes necessary optimality conditions, as can be easily seen by considering the function  $f(x) = x^3$ . This function has an extremal point at  $x^* = 0$  ( $f'(x^*) = 3(x^*)^2 = 0$ ) and although the second derivative  $f''(x^*) = 6x^* = 0$  is positive semi-definite,  $x^* = 0$  is not a minimum. The function has a saddle point at  $x^* = 0$ .

We can now state the following sufficient optimality conditions.

**Theorem 2.3 (Second-order sufficient optimality conditions).** Let  $f \in C^2$  be a function defined on  $\mathbb{R}^n$ . If the conditions

$$(a) \quad (\nabla f)(\mathbf{x}^*) = \mathbf{0} \quad (2.7a)$$

$$(b) \quad (\nabla^2 f)(\mathbf{x}^*) \text{ is positive definite} \quad (2.7b)$$

are satisfied,  $\mathbf{x}^*$  is a strict local minimum of  $f$  on  $\mathbb{R}^n$ .

**Exercise 2.2.** Prove Theorem 2.3.

**Example 2.2.** For the optimization problem

$$\min_{\mathbf{x} \in \mathbb{R}^2} f(\mathbf{x}) = x_1^2 + ax_2^2 - x_1x_2 \quad (2.8)$$

we want to characterize the stationary values  $\mathbf{x}^*$  as a function of the parameter  $a \neq \frac{1}{4}$ . The gradient and the Hessian of  $f(\mathbf{x})$  are

$$(\nabla f)(\mathbf{x}) = \begin{bmatrix} 2x_1 - x_2 \\ 2ax_2 - x_1 \end{bmatrix}, \quad (\nabla^2 f)(\mathbf{x}) = \begin{bmatrix} 2 & -1 \\ -1 & 2a \end{bmatrix}. \quad (2.9)$$

From  $(\nabla f)(\mathbf{x}^*) = \mathbf{0}$ ,  $\mathbf{x}^* = [0 \ 0]^T$  follows as the only stationary point. The definiteness of the Hessian  $(\nabla^2 f)(\mathbf{x})$  at  $\mathbf{x}^*$  can be investigated with the help of the principal minors (Sylvester's criterion, see (c) in Theorem 1.2)

$$D_1 = 2, \quad D_2 = 4a - 1. \quad (2.10)$$

Thus,  $(\nabla^2 f)(\mathbf{x}^*)$  is positive definite for  $a > \frac{1}{4}$  and  $\mathbf{x}^* = [0 \ 0]^T$  represents a strict minimum. For  $a < \frac{1}{4}$  we have  $D_1 > 0$  and  $D_2 < 0$  and  $(\nabla^2 f)(\mathbf{x})$  is thus *indefinite*. In this case,  $\mathbf{x}^* = [0 \ 0]^T$  is a *saddle point*, similar to the one shown in Figure 2.1(b) for  $a = -1$ .

The theorems 2.1 to 2.3 only provide statements about local minima. If the function  $f(\mathbf{x})$  is convex or strictly convex, then the following simplified optimality conditions can be formulated.

**Theorem 2.4 (Sufficient optimality condition for convex function).** Let  $f \in C^1$  be a convex function defined on  $\mathbb{R}^n$ . If the condition

$$(\nabla f)(\mathbf{x}^*) = \mathbf{0} \quad (2.11)$$

is satisfied,  $\mathbf{x}^*$  is a global minimum of  $f$  on  $\mathbb{R}^n$ .

*Proof.* For any  $\mathbf{y} \in \mathbb{R}^n$ , because of the convexity of  $f(\mathbf{x})$  with the minimum  $\mathbf{x}^*$ , the inequality

$$f(\mathbf{y}) \geq f(\mathbf{x}^*) + (\mathbf{y} - \mathbf{x}^*)^T \underbrace{(\nabla f)(\mathbf{x}^*)}_{=0} = f(\mathbf{x}^*) \quad (2.12)$$

is satisfied. □

Consequently, for convex functions  $f(\mathbf{x})$  the first-order necessary optimality condition according to Theorem 2.1 is also *sufficient*.

**Theorem 2.5 (Global minima of a convex function).** Let  $f(\mathbf{x})$  be a convex function on  $\mathbb{R}^n$ . The set of all minima  $\mathcal{G} = \arg \min\{f(\mathbf{x}) \mid \mathbf{x} \in \mathbb{R}^n\}$  is convex. Every local

minimum  $\mathbf{x}^* \in \mathcal{G}$  of  $f$  is also a global minimum. If  $f(\mathbf{x})$  is strictly convex, then  $\mathbf{x}^*$  is a strict global minimum.

*Proof.* Assume that  $c$  describes the minimum value of  $f$ . Then the set  $\mathcal{G} = \{\mathbf{x} \mid \mathbf{x} \in \mathbb{R}^n, f(\mathbf{x}) \leq c\}$  is convex according to (1.56), which proves the first part of the theorem.

The second part of the theorem can be shown by contradiction. Assume that  $\mathbf{x}^*$  is a local minimum but not a global minimum of  $f$  on  $\mathbb{R}^n$ . Then there exists a point  $\mathbf{y} \in \mathbb{R}^n$  that satisfies the inequality  $f(\mathbf{y}) < f(\mathbf{x}^*)$ . According to Definition 1.7 for convex functions, the following holds for all  $\alpha \in [0, 1]$

$$f(\alpha\mathbf{x}^* + (1-\alpha)\mathbf{y}) \leq \alpha f(\mathbf{x}^*) + (1-\alpha)f(\mathbf{y}) < \alpha f(\mathbf{x}^*) + (1-\alpha)f(\mathbf{x}^*) = f(\mathbf{x}^*). \quad (2.13)$$

This shows that with  $\alpha \rightarrow 1$  we would find points  $\alpha\mathbf{x}^* + (1-\alpha)\mathbf{y}$  that are arbitrarily close to  $\mathbf{x}^*$ , but whose function values  $f(\alpha\mathbf{x}^* + (1-\alpha)\mathbf{y})$  are strictly smaller than  $f(\mathbf{x}^*)$ . This violates the definition of a local minimum at  $\mathbf{x}^*$  and therefore contradicts the assumption. Consequently, a point  $\mathbf{y} \in \mathbb{R}^n$  that satisfies the inequality  $f(\mathbf{y}) < f(\mathbf{x}^*)$  cannot exist and  $\mathbf{x}^*$  is a global minimum.

The third part of the Theorem can be shown in a similar way. The assumption that  $\mathbf{x}^*$  is not a strict global minimum, i.e., that there exists a point  $\mathbf{y} \in \mathbb{R}^n$  with  $\mathbf{y} \neq \mathbf{x}^*$  which satisfies  $f(\mathbf{y}) = f(\mathbf{x}^*)$  leads to a contradiction for a strictly convex function  $f(\mathbf{x})$ .  $\square$

## 2.2 Computer-Aided Minimization Methods: Fundamentals

### 2.2.1 Iterative Descend Methods

Because determining a (locally) optimal point  $\mathbf{x}^*$  of (2.1) by analytically solving the stationarity condition  $(\nabla f)(\mathbf{x}^*) = \mathbf{0}$  from (2.7a) ( $n$  nonlinear equations in  $\mathbf{x}^*$ ) is only possible in rare cases, we generally have to resort to *numerical methods* when searching for  $\mathbf{x}^*$ . Many of the algorithms discussed in this lecture do not find the exact point  $\mathbf{x}^*$  directly or within a finite number of computational steps. They rather generate a sequence  $\{\mathbf{x}_k\}$  along which the function  $f(\mathbf{x})$  to be minimized decreases, i. e.,

$$f(\mathbf{x}_{k+1}) < f(\mathbf{x}_k), \quad k = 0, 1, 2, \dots, \quad (2.14)$$

and that should converge to  $\mathbf{x}^*$  at least for  $k \rightarrow \infty$ , i. e.,

$$\lim_{k \rightarrow \infty} \mathbf{x}_k = \mathbf{x}^*. \quad (2.15)$$

Such algorithms are also called *iterative descent methods*, and (2.14) is referred to as the *descent condition*. In addition to the question whether an algorithm ensures convergence to the correct solution  $\mathbf{x}^*$ , which can be answered by (2.15), we are interested in how quickly it converges. This question is addressed in the following section.

### 2.2.2 Convergence Behavior

We analyze the convergence behavior of iterative descend methods, which generate a sequence  $\{\mathbf{x}_k\}$ . If this sequence reaches the optimal point  $\mathbf{x}^*$  already after a finite number of computational steps, the method possesses *finite convergence*. If this sequence reaches the optimal point  $\mathbf{x}^*$  only for  $k \rightarrow \infty$ , the method possesses *asymptotic convergence*, which is primarily discussed below. Even for methods with formally finite convergence, an analysis of the asymptotic convergence behavior can be of interest because computational inaccuracies may prevent that the optimal point  $\mathbf{x}^*$  is exactly reached after a finite number of computational steps.

Usually, an analysis of the convergence behavior is carried out based on an *error function*  $e : \mathbb{R}^n \rightarrow \mathbb{R}$ , which satisfies  $e(\mathbf{x}) \geq 0$  for all  $\mathbf{x} \in \mathbb{R}^n$  and  $e(\mathbf{x}^*) = 0$ . The error function can be, for example, the distance

$$e(\mathbf{x}) = \|\mathbf{x} - \mathbf{x}^*\| \geq 0 \quad (2.16a)$$

in the sense of a norm  $\|\cdot\|$  or the objective function difference

$$e(\mathbf{x}) = f(\mathbf{x}) - f(\mathbf{x}^*) \geq 0 \quad (2.16b)$$

[2.1]. The asymptotic convergence behavior of an algorithm can be analyzed using the sequence  $\{e_k\}$  associated with  $\{\mathbf{x}_k\}$  with  $e_k = e(\mathbf{x}_k)$ . First, we define the terms *order of convergence* and *rate of convergence* of a sequence of scalars.

**Definition 2.1** (Order of convergence, rate of convergence). Let  $\{e_k\}$  be a sequence of scalars, which asymptotically converges to the limit 0. The *order of convergence* of the sequence  $\{e_k\}$  is the least upper bound (supremum) for the set of nonnegative numbers  $p$  which satisfy

$$0 \leq \lim_{k \rightarrow \infty} \frac{|e_{k+1}|}{|e_k|^p} = \mu < \infty. \quad (2.17)$$

The number  $\mu$  is called *rate of convergence*. We distinguish between the following cases:

- The case  $p = 1$  and  $\mu \in (0, 1)$  is referred to as *linear* convergence.
- The case  $p > 1$  with  $\mu > 0$  or  $p = 1$  with  $\mu = 0$  is referred to as *superlinear* convergence.
- The case  $p = 2$  with  $\mu > 0$  is referred to as *quadratic* convergence.
- The case  $p = 3$  with  $\mu > 0$  is referred to as *cubic* convergence.

Following this definition, the order of convergence and the rate of convergence describe the (local) behavior of a sequence for  $k \rightarrow \infty$ . Larger values of the order of convergence  $p$  imply that the sequence converges faster because the sequence elements  $e_k$  (at least for large values of  $k$ ) decrease with the  $p$ -th power. The same applies to smaller values of the rate of convergence  $\mu$ .

**Example 2.3.** The sequence  $\{a^k\}$  with  $0 < a < 1$  converges to zero with the order of convergence  $p = 1$  and the rate of convergence  $\mu = a$ . Note that the condition

$$\lim_{k \rightarrow \infty} \frac{a^{k+1}}{a^{kp}} = \lim_{k \rightarrow \infty} a^{1+k(1-p)} < \infty \quad (2.18)$$

is only fulfilled for  $p \leq 1$ . For  $p = 1$  we then get

$$\lim_{k \rightarrow \infty} \frac{a^{k+1}}{a^k} = a = \mu \quad (2.19)$$

for the rate of convergence  $\mu$ .

**Exercise 2.3.** Show that the sequence  $\{a^{2^k}\}$  with  $0 < a < 1$  converges to 0 with the order of convergence 2 and the rate of convergence 1.

**Example 2.4.** The sequence  $\{\frac{1}{k^k}\}$  has the order of convergence 1 because the condition

$$\lim_{k \rightarrow \infty} \frac{k^{kp}}{(k+1)^{k+1}} = \lim_{k \rightarrow \infty} \frac{k}{k+1} \left( \frac{k}{k+1} \right)^k k^{k(p-1)-1} = \lim_{k \rightarrow \infty} \frac{1}{e} k^{k(p-1)-1} < \infty \quad (2.20)$$

is only fulfilled for  $p \leq 1$ . For  $p = 1$  we then obtain

$$\lim_{k \rightarrow \infty} \frac{k^k}{(k+1)^{k+1}} = \lim_{k \rightarrow \infty} \frac{1}{k+1} \left( \frac{k}{k+1} \right)^k = 0 \frac{1}{e} = \mu = 0. \quad (2.21)$$

Consequently, the sequence  $\{\frac{1}{k^k}\}$  converges superlinearly to zero.

Finally, the question arises whether the observed asymptotic convergence behavior of an optimization algorithm depends on the chosen error function  $e(\mathbf{x})$ . It can be shown (cf. [2.2]) that the order of convergence of an optimization algorithm is largely independent of the choice of the error function  $e(\mathbf{x})$ . This is not the case for the rate of convergence.

### 2.2.3 Termination Criteria

When applying an iterative descend method, it must be decided when an obtained solution is sufficiently accurate so that the iteration can be terminated. The algorithms make this decision based on *termination criteria* [2.1, 2.3, 2.4]. Some examples for such criteria are given in the following. Typically the first satisfaction of one criterion is sufficient to terminate the algorithm.

- Change of the cost function value

$$f(\mathbf{x}_{k-1}) - f(\mathbf{x}_k) \leq \varepsilon_f^{\text{abs}} + \varepsilon_f^{\text{rel}} |f(\mathbf{x}_{k-1})| \quad (2.22)$$

- Change of the current point

$$\|\mathbf{x}_k - \mathbf{x}_{k-1}\| \leq \varepsilon_x^{\text{abs}} + \varepsilon_x^{\text{rel}} \|\mathbf{x}_{k-1}\| \quad (2.23)$$

- Value of the gradient (see also the first-order necessary optimality condition in Theorem 2.1)

$$\|(\nabla f)(\mathbf{x}_k)\| \leq \varepsilon_{\nabla f}^{\text{abs}} + \varepsilon_{\nabla f}^{\text{rel}} \|(\nabla f)(\mathbf{x}_0)\| \quad (2.24)$$

- Number of iterations

$$k \geq k_{\max} \quad (2.25)$$

- Number of function calls

$$\text{Number of evaluations of } f(\mathbf{x}) \geq n_{\max} \quad (2.26)$$

- Required computation time

$$t_{\text{CPU}} \geq t_{\text{CPU},\max} \quad (2.27)$$

The last three criteria cause an early, unsuccessful termination of the algorithm. The scalar parameters  $\varepsilon_f^{\text{abs}}, \varepsilon_x^{\text{abs}}, \varepsilon_{\nabla f}^{\text{abs}} > 0$ ,  $\varepsilon_f^{\text{rel}}, \varepsilon_x^{\text{rel}}, \varepsilon_{\nabla f}^{\text{rel}} \geq 0$  and  $k_{\max}, n_{\max}, t_{\text{CPU},\max} \gg 0$  must be reasonably chosen.

The most commonly used numerical methods for solving the unconstrained static optimization problem (2.1) are the so-called *line search methods*. The following section gives a brief overview of common line search methods. This is followed by a presentation of two alternative solution methods for unconstrained static optimization problems, the *trust region method* and the *direct search method*.

## 2.3 Line Search Methods

---

**Initialization:**  $\mathbf{x}_0$  (initial solution)  
 $k = 0$  (initial index)

**while**  $\mathbf{x}_k$  is not optimal

Choose a suitable search direction  $\mathbf{s}_k$

Choose an optimal step size according to

$\alpha_k = \arg \min_{\alpha > 0} f(\mathbf{x}_k + \alpha \mathbf{s}_k)$

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

$k \leftarrow k + 1$

**end**

---

Table 2.1: General procedure of a line search method.

Table 2.1 shows the basic algorithmic structure of a line search method. In iteration step  $k$ , a suitable *search direction* or *descent direction*  $\mathbf{s}_k$  is first determined. It should be chosen in such a way that for a sufficiently little move away from the point  $\mathbf{x}_k$  along this direction, i.e.,

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{s}_k \quad (2.28)$$

with a suitable *step size*  $\alpha_k > 0$ , the descent condition (2.14), i.e.,

$$f(\mathbf{x}_{k+1}) = f(\mathbf{x}_k + \alpha_k \mathbf{s}_k) < f(\mathbf{x}_k) \quad (2.29)$$

is fulfilled. The optimal step size  $\alpha_k > 0$  is then determined by solving the *scalar optimization problem*

$$\min_{\alpha_k > 0} g(\alpha_k) = f(\mathbf{x}_k + \alpha_k \mathbf{s}_k) . \quad (2.30)$$

The iteration is repeated until a termination criterion is met.

Figure 2.2 illustrates the principle of line search for one iteration step, a (non-convex) objective function  $f(\mathbf{x})$  with  $\mathbf{x} \in \mathbb{R}^2$ , and a given search direction  $\mathbf{s}_k$ . In this context, the name *line search method* becomes clear because for a given search direction  $\mathbf{s}_k$ , the optimization task, i.e., choosing the step size  $\alpha_k$ , is reduced to finding a minimum along a line.

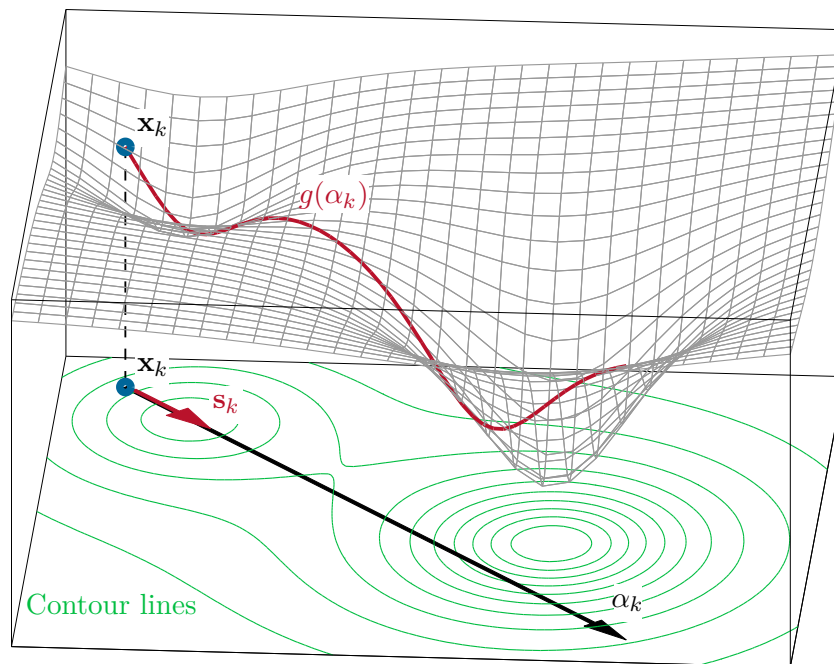


Figure 2.2: Illustration of choosing the step size according to (2.30).

### 2.3.1 Choice of Step Size

#### 2.3.1.1 Interval Bisection Method ('Golden-Section Search')

The *interval bisection method* generates a convergent sequence of interval bisections for the scalar optimization problem (2.30) to (approximately) find the minimum of  $g(\alpha_k)$ .

First, an interval  $[l_0, r_0]$  must be found in which the function  $g(\alpha_k)$  has exactly one minimum, see Figure 2.3. This can be achieved, e.g., by starting with a sufficiently small  $l_0$  and successively increasing  $r_0$  (starting from  $l_0$ ) until the function value  $g(r_0)$  starts to increase. In the following, it is assumed that the function  $g(\alpha_k)$  is continuous and has exactly one unique minimum (one strict minimum and no other local minima) in the interval  $[l_0, r_0]$ .

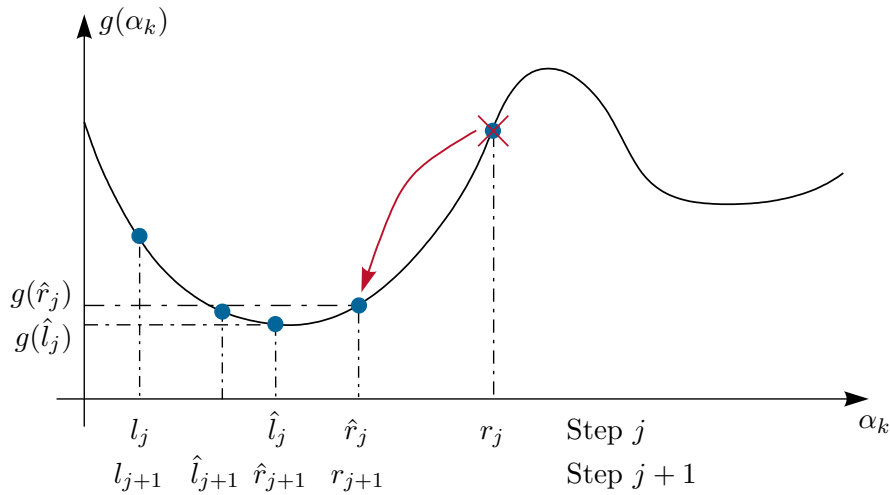


Figure 2.3: Illustration of the interval bisection method according to the principle of the 'golden-section search'.

At iteration step  $j$ , let the interval  $[l_j, r_j]$  be given, which contains the value  $\alpha_k^*$  that minimizes the function  $g(\alpha_k)$ . Now two new points  $\hat{l}_j$  and  $\hat{r}_j$ ,  $l_j < \hat{l}_j < \hat{r}_j < r_j$ , are selected based on

$$\hat{l}_j = l_j + (1 - a)(r_j - l_j) = al_j + (1 - a)r_j \quad (2.31a)$$

$$\hat{r}_j = l_j + a(r_j - l_j) = (1 - a)l_j + ar_j \quad (2.31b)$$

with the parameter  $a \in (\frac{1}{2}, 1)$ . Consider the following theorem.

**Theorem 2.6 (Interval bisection).** Let  $l_j < \hat{l}_j < \hat{r}_j < r_j$  and let  $g(\alpha_k)$  be a continuous function with one unique minimum in the interval  $[l_j, r_j]$ . If we denote this minimum by  $\alpha_k^* \in (l_j, r_j)$ , then  $\alpha_k^* \in [l_j, \hat{r}_j]$  if  $g(\hat{l}_j) \leq g(\hat{r}_j)$  and  $\alpha_k^* \in [\hat{l}_j, r_j]$  if  $g(\hat{l}_j) \geq g(\hat{r}_j)$ .

*Proof.* Consider the case  $g(\hat{l}_j) \leq g(\hat{r}_j)$ . Assume that  $\alpha_k^* > \hat{r}_j$ , then  $\hat{l}_j < \alpha_k^*$ . Since  $g(\hat{l}_j) \leq g(\hat{r}_j)$  and  $g(\alpha_k^*) \leq g(\hat{r}_j)$ , there must exist a point  $\bar{\alpha}_k \in (\hat{l}_j, \alpha_k^*)$  such that

$g(\bar{\alpha}_k) = \max_{\alpha_k \in [\hat{l}_j, \alpha_k^*]} g(\alpha_k)$ , which means that  $\bar{\alpha}_k$  describes a local maximum of  $g(\alpha_k)$  in the interval  $[l_j, r_j]$ . However, the existence of a local maximum is impossible because of the requirement of having one unique minimum of  $g(\alpha_k)$ . For  $g(\hat{l}_j) \geq g(\hat{r}_j)$ , the proof follows in an analogous way.  $\square$

According to Theorem 2.6, for the next iteration step  $j + 1$  in case  $g(\hat{l}_j) \leq g(\hat{r}_j)$ , the outer point  $r_j$  is discarded and the interval thus becomes  $[l_{j+1}, r_{j+1}]$  with  $l_{j+1} = l_j$ ,  $r_{j+1} = \hat{r}_j$ , see Figure 2.3. For  $g(\hat{l}_j) \geq g(\hat{r}_j)$ , the interval at iteration step  $j + 1$  becomes  $[l_{j+1}, r_{j+1}]$  with  $l_{j+1} = \hat{l}_j$ ,  $r_{j+1} = r_j$ .

For the further consideration, assume that, as shown in Figure 2.3,  $g(\hat{l}_j) \leq g(\hat{r}_j)$ . The following steps can be directly transferred to the other case. First, with  $l_{j+1} = l_j$  and  $r_{j+1} = \hat{r}_j = (1 - a)l_j + ar_j$ , a further iteration is performed to calculate the intermediate points according to (2.31) in the form

$$\hat{l}_{j+1} = al_{j+1} + (1 - a)r_{j+1} = (1 - a + a^2)l_j + (1 - a)ar_j \quad (2.32a)$$

$$\hat{r}_{j+1} = (1 - a)l_{j+1} + ar_{j+1} = (1 - a^2)l_j + a^2r_j. \quad (2.32b)$$

Comparing the coefficients of (2.31a) and (2.32b), it follows that  $\hat{r}_{j+1} = \hat{l}_j$  is achieved exactly if  $a^2 = 1 - a$  is satisfied, i.e., when

$$a = \frac{\sqrt{5} - 1}{2} \approx 0.618. \quad (2.33)$$

This choice of  $a$  has the advantage that only one new intermediate point must be calculated per iteration. Note that the calculation of each intermediate point is associated with an evaluation of the cost function  $g(\alpha_k)$ . The number  $1/a = 1 + a \approx 1.618$  is known as the ratio of the 'golden ratio'. Table 2.2 summarizes the algorithm.

Finally, the optimal value  $\alpha_k^*$  can be obtained either by *averaging* the last interval boundaries  $\alpha_k^* = (l_j + r_j)/2$  or by *quadratic interpolation* (see the following section) between the smallest three function values. The interval bisection method is a *simple* and *robust* method but it usually requires more iterations compared to other methods.

### 2.3.1.2 Quadratic Interpolation

A very efficient method for solving the scalar optimization problem (2.30) consists of fitting a quadratic interpolation function through three points. This approach is sometimes also referred to as Newton's method. It is assumed that the pairwise distinct points  $\alpha_{k,1}$ ,  $\alpha_{k,2}$  and  $\alpha_{k,3}$  and their function values  $g_1 = g(\alpha_{k,1})$ ,  $g_2 = g(\alpha_{k,2})$  and  $g_3 = g(\alpha_{k,3})$  are known. According to Lagrange's interpolation formula, the quadratic interpolation function  $q(\alpha_k)$  through these points is then

$$q(\alpha_k) = \sum_{i=1}^3 \left( g_i \prod_{j \neq i} \frac{\alpha_k - \alpha_{k,j}}{\alpha_{k,i} - \alpha_{k,j}} \right) \quad (2.34)$$

and the optimal value  $\alpha_k^*$  follows in the form

---

**Initialization:**  $l_0, r_0$  (initial interval containing the minimum)  
 $j = 0$  (initial index)  
 $a = 0.618$  (golden-ratio parameter)  
 $\varepsilon_{lr}$  (termination tolerance)  
 $\hat{l}_0 \leftarrow al_0 + (1 - a)r_0$  (interior points)  
 $\hat{r}_0 \leftarrow (1 - a)l_0 + ar_0$

**repeat**

**if**  $g(\hat{l}_j) > g(\hat{r}_j)$

$l_{j+1} \leftarrow \hat{l}_j$   
 $r_{j+1} \leftarrow r_j$   
 $\hat{l}_{j+1} \leftarrow \hat{r}_j$   
 $\hat{r}_{j+1} \leftarrow (1 - a)l_{j+1} + ar_{j+1}$

**else** (d. h.  $g(\hat{l}_j) \leq g(\hat{r}_j)$ )

$r_{j+1} \leftarrow \hat{r}_j$   
 $l_{j+1} \leftarrow l_j$   
 $\hat{r}_{j+1} \leftarrow \hat{l}_j$   
 $\hat{l}_{j+1} \leftarrow al_{j+1} + (1 - a)r_{j+1}$

**end if**

$j \leftarrow j + 1$

**until**  $|r_j - l_j| \leq \varepsilon_{lr}$

$\alpha_k^* = (l_j + r_j)/2$

---

Table 2.2: Interval bisection method ('golden-section search').

$$\alpha_k^* = \frac{1}{2} \frac{g_1(\alpha_{k,2}^2 - \alpha_{k,3}^2) + g_2(\alpha_{k,3}^2 - \alpha_{k,1}^2) + g_3(\alpha_{k,1}^2 - \alpha_{k,2}^2)}{g_1(\alpha_{k,2} - \alpha_{k,3}) + g_2(\alpha_{k,3} - \alpha_{k,1}) + g_3(\alpha_{k,1} - \alpha_{k,2})}. \quad (2.35)$$

The value  $\alpha_k^*$  calculated in this way should only be accepted as the optimal step size if the interpolation function  $q(\alpha_k)$  is strictly convex (only then  $\alpha_k^*$  is actually a minimum of  $q(\alpha_k)$ ) and if the conditions  $\alpha_k^* > 0$ ,  $g(\alpha_k^*) \leq g_1$ ,  $g(\alpha_k^*) \leq g_2$ ,  $g(\alpha_k^*) \leq g_3$ , and  $g(\alpha_k^*) \leq g(0)$  are satisfied.

**Exercise 2.4.** Show that (2.35) holds true.

### 2.3.1.3 Heuristic Choice of Step Size

In practice, when choosing the step size, a trade-off between computational effort and achieved accuracy often has to be made. Inaccuracies can occur, for example, when an iterative algorithm for determining the optimal step size is terminated prematurely. There are various *heuristic termination criteria* that will be briefly explained below. For the

subsequent considerations, the *scalar optimization problem*

$$\min_{\alpha_k > 0} g(\alpha_k) = f(\mathbf{x}_k + \alpha_k \mathbf{s}_k) , \quad (2.36)$$

see (2.30), is assumed.

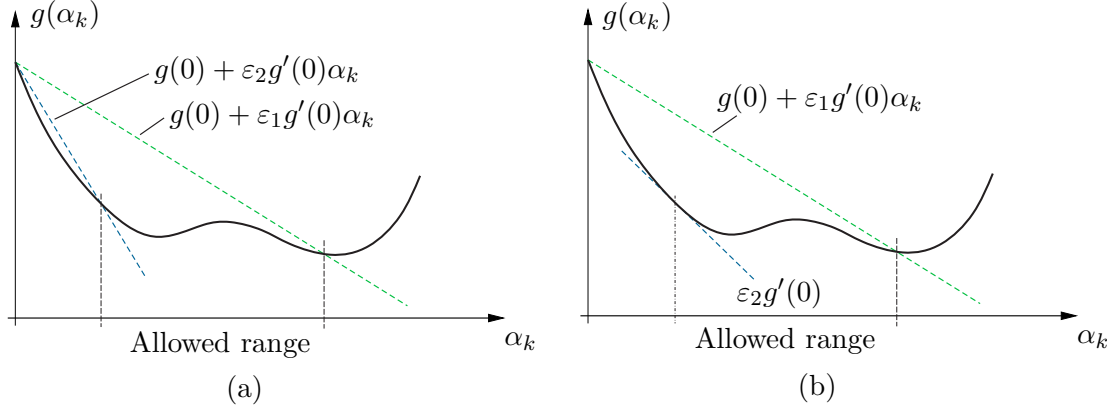


Figure 2.4: Illustration of the Armijo-Goldstein condition (a) and Wolfe condition (b).

**Armijo-Goldstein condition:** Expanding  $g(\alpha_k)$  into a Taylor series around  $\alpha_k = 0$  and truncating after the linear term yields

$$g(\alpha_k) \approx g(0) + g'(0) \alpha_k . \quad (2.37)$$

The *Armijo-Goldstein condition* chooses the step size  $\alpha_k$  such that for a fixed  $\varepsilon_1$ ,  $0 < \varepsilon_1 < 1$ , the inequality

$$g(\alpha_k) \leq g(0) + \varepsilon_1 g'(0) \alpha_k \quad (2.38)$$

is satisfied. This guarantees that the step size  $\alpha_k$  is bounded from above. To ensure that the step size does not become too small, the *Armijo-Goldstein condition* introduces an additional parameter  $\varepsilon_2$  with  $\varepsilon_1 < \varepsilon_2 < 1$  and requires that the step size  $\alpha_k$  satisfies the inequality

$$g(\alpha_k) > g(0) + \varepsilon_2 g'(0) \alpha_k . \quad (2.39)$$

Figure 2.4(a) illustrates the Armijo-Goldstein condition. In practice, the application of this condition often starts by choosing an (arbitrary) initial value for  $\alpha_k$ . If the inequality (2.38) is satisfied for this  $\alpha_k$ , then  $\alpha_k$  is successively multiplied by the factor  $\varepsilon_2/\varepsilon_1$  until the inequality (2.38) is violated for the first time. The penultimate value of  $\alpha_k$  is then chosen as a suitable step size. Conversely, if the initial value of  $\alpha_k$  does not satisfy the inequality (2.38), then  $\alpha_k$  is successively divided by the factor  $\varepsilon_2/\varepsilon_1$  until the inequality (2.38) is satisfied the first time. Typical parameter values are  $\varepsilon_1 = 0.1$  and  $\varepsilon_2 = 0.2$ . However, if  $\varepsilon_1$  is too large, the descent condition becomes too restrictive, i. e. the step size becomes too small.

**Wolfe condition:** If the derivatives of the cost function  $g(\alpha_k)$  can be easily calculated, the so-called *Wolfe condition* is a suitable alternative to the Armijo-Goldstein condition.

An additional parameter  $\varepsilon_2$  with  $0 < \varepsilon_1 < \varepsilon_2 < 1$  is introduced, and the step size  $\alpha_k$  is required to satisfy the inequalities (2.38) and

$$g'(\alpha_k) \geq \varepsilon_2 g'(0) . \quad (2.40)$$

Figure 2.4(b) illustrates this situation. Typical values for  $\varepsilon_2$  are 0.9 when the search direction  $\mathbf{s}_k$  is determined by Newton's method or the quasi-Newton method, and 0.1 when  $\mathbf{s}_k$  is determined by the conjugate gradient method.

## 2.3.2 Choice of Search Direction

### 2.3.2.1 Gradient Method

In the *gradient method*, also called the *steepest descent method*, the search direction  $\mathbf{s}_k$  in (2.28) is chosen as the *negative gradient* at the point  $\mathbf{x}_k$ , i. e., the descent direction

$$\mathbf{s}_k = -(\nabla f)(\mathbf{x}_k) . \quad (2.41)$$

If  $g(\alpha_k) = f(\mathbf{x}_k + \alpha_k \mathbf{s}_k)$  is expanded into a Taylor series around the point  $\alpha_k = 0$  with  $\mathbf{s}_k$  according to (2.41),

$$g(\alpha_k) = f(\mathbf{x}_k + \alpha_k \mathbf{s}_k) = f(\mathbf{x}_{k+1}) = f(\mathbf{x}_k) - \alpha_k \|(\nabla f)(\mathbf{x}_k)\|_2^2 + \mathcal{O}(\alpha_k^2) , \quad (2.42)$$

where  $\mathcal{O}(\alpha_k^2)$  denotes the remainder of quadratic order, it follows that for sufficiently small  $\alpha_k$  the descent condition (2.14) is satisfied as long as  $(\nabla f)(\mathbf{x}_k) \neq \mathbf{0}$ .

To investigate the convergence properties of the gradient method in more detail, consider the quadratic optimization problem

$$\min_{\mathbf{x} \in \mathbb{R}^n} f(\mathbf{x}) = \frac{1}{2} \mathbf{x}^T \mathbf{Q} \mathbf{x} - \mathbf{x}^T \mathbf{b} \quad (2.43)$$

with the symmetric positive definite matrix  $\mathbf{Q} \in \mathbb{R}^{n \times n}$ . Because the Hessian  $(\nabla^2 f)(\mathbf{x}) = \mathbf{Q}$  of  $f(\mathbf{x})$  is positive definite, it follows from property (e) of convex functions from Section 1.3.6.2 that  $f(\mathbf{x})$  is strictly convex. Therefore, based on Theorems 2.1 and 2.5, the strict global minimum  $\mathbf{x}^*$  of  $f(\mathbf{x})$  is obtained from the relation

$$(\nabla f)(\mathbf{x}^*) = \mathbf{g}(\mathbf{x}^*) = \mathbf{Q} \mathbf{x}^* - \mathbf{b} = \mathbf{0} \quad (2.44)$$

in the form

$$\mathbf{x}^* = \mathbf{Q}^{-1} \mathbf{b} . \quad (2.45)$$

In this case, the iteration formula according to the gradient method is

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k \mathbf{g}_k \quad \text{with} \quad \mathbf{g}_k = \mathbf{g}(\mathbf{x}_k) = \mathbf{Q} \mathbf{x}_k - \mathbf{b} , \quad (2.46)$$

see (2.28). The optimal step size  $\alpha_k^*$  can be calculated by explicitly solving the optimization problem (2.30), i. e.,

$$\min_{\alpha_k > 0} f(\mathbf{x}_k + \alpha_k \mathbf{s}_k) = \frac{1}{2} (\mathbf{x}_k - \alpha_k \mathbf{g}_k)^T \mathbf{Q} (\mathbf{x}_k - \alpha_k \mathbf{g}_k) - (\mathbf{x}_k - \alpha_k \mathbf{g}_k)^T \mathbf{b} , \quad (2.47)$$

which yields

$$\alpha_k^* = \frac{\mathbf{g}_k^T \mathbf{g}_k}{\mathbf{g}_k^T \mathbf{Q} \mathbf{g}_k} . \quad (2.48)$$

**Exercise 2.5.** Show that (2.48) holds.

In summary, the gradient method for the quadratic cost function (2.43) can be written in the form

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \frac{\mathbf{g}_k^T \mathbf{g}_k}{\mathbf{g}_k^T \mathbf{Q} \mathbf{g}_k} \mathbf{g}_k , \quad \mathbf{g}_k = \mathbf{Q} \mathbf{x}_k - \mathbf{b} . \quad (2.49)$$

To study the convergence behavior of the gradient method, it is advantageous to use (2.45) and consider the error function

$$\begin{aligned} e(\mathbf{x}) &= \|\mathbf{x} - \mathbf{x}^*\|_{\mathbf{Q}} = \sqrt{(\mathbf{x} - \mathbf{x}^*)^T \mathbf{Q} (\mathbf{x} - \mathbf{x}^*)} \\ &= \sqrt{\mathbf{x}^T \mathbf{Q} \mathbf{x} - 2\mathbf{x}^T \mathbf{Q} \mathbf{x}^* + (\mathbf{x}^*)^T \mathbf{Q} \mathbf{x}^*} = \sqrt{2f(\mathbf{x}) + (\mathbf{x}^*)^T \mathbf{Q} \mathbf{x}^*} \end{aligned} \quad (2.50)$$

instead of  $f(\mathbf{x})$ . The minima  $\mathbf{x}^*$  of the functions  $f(\mathbf{x})$  and  $e(\mathbf{x})$  are identical. The vector norm  $\|\cdot\|_{\mathbf{Q}}$  with a symmetric positive definite matrix  $\mathbf{Q}$  is also called *energy norm*.

**Lemma 2.1 (On the convergence rate of the gradient method).** *With the iteration formula (2.49) of the gradient method, the error function  $e(\mathbf{x})$  evolves according to*

$$e(\mathbf{x}_{k+1}) = \left( 1 - \frac{(\mathbf{g}_k^T \mathbf{g}_k)^2}{\mathbf{g}_k^T \mathbf{Q} \mathbf{g}_k \mathbf{g}_k^T \mathbf{Q}^{-1} \mathbf{g}_k} \right)^{\frac{1}{2}} e(\mathbf{x}_k) . \quad (2.51)$$

*Proof.* From (2.45) and (2.49), it follows that

$$\mathbf{g}_k = \mathbf{Q} \mathbf{x}_k - \mathbf{b} = \mathbf{Q}(\mathbf{x}_k - \mathbf{x}^*) \quad (2.52)$$

and consequently

$$e(\mathbf{x}_k) = \sqrt{\mathbf{g}_k^T \mathbf{Q}^{-1} \mathbf{g}_k} . \quad (2.53)$$

Using these relations and the iteration formula (2.49), (2.51) can now be directly calculated.

$$\begin{aligned} e(\mathbf{x}_{k+1}) &= \left( \left( \mathbf{x}_k - \frac{\mathbf{g}_k^T \mathbf{g}_k}{\mathbf{g}_k^T \mathbf{Q} \mathbf{g}_k} \mathbf{g}_k - \mathbf{x}^* \right)^T \mathbf{Q} \left( \mathbf{x}_k - \frac{\mathbf{g}_k^T \mathbf{g}_k}{\mathbf{g}_k^T \mathbf{Q} \mathbf{g}_k} \mathbf{g}_k - \mathbf{x}^* \right) \right)^{\frac{1}{2}} \\ &= \left( (\mathbf{x}_k - \mathbf{x}^*)^T \mathbf{Q} (\mathbf{x}_k - \mathbf{x}^*) - \frac{(\mathbf{g}_k^T \mathbf{g}_k)^2}{\mathbf{g}_k^T \mathbf{Q} \mathbf{g}_k} \right)^{\frac{1}{2}} \\ &= \left( 1 - \frac{(\mathbf{g}_k^T \mathbf{g}_k)^2}{\mathbf{g}_k^T \mathbf{Q} \mathbf{g}_k \mathbf{g}_k^T \mathbf{Q}^{-1} \mathbf{g}_k} \right)^{\frac{1}{2}} e(\mathbf{x}_k) \end{aligned} \quad (2.54)$$

□

To actually obtain the rate of convergence of the gradient method using the error function  $e(\mathbf{x})$ , the following lemma is also needed.

**Lemma 2.2 (Kantorovich inequality).** *Let  $\mathbf{Q} \in \mathbb{R}^{n \times n}$  be a symmetric positive definite matrix. For every vector  $\mathbf{x} \in \mathbb{R}^n$  with  $\mathbf{x} \neq \mathbf{0}$ , the inequality*

$$\frac{(\mathbf{x}^T \mathbf{x})^2}{(\mathbf{x}^T \mathbf{Q} \mathbf{x})(\mathbf{x}^T \mathbf{Q}^{-1} \mathbf{x})} \geq \frac{4\lambda_{\min} \lambda_{\max}}{(\lambda_{\min} + \lambda_{\max})^2} \quad (2.55)$$

*holds, where  $\lambda_{\min}$  and  $\lambda_{\max}$  denote the smallest and largest (real and positive) eigenvalues of the matrix  $\mathbf{Q}$ .*

The proof of Lemma 2.2 is sketched, e. g., in [2.2].

**Theorem 2.7 (Convergence of the gradient method for quadratic cost functions).** *For any initial value  $\mathbf{x}_0 \in \mathbb{R}^n$ , the iteration formula (2.49) of the gradient method converges to the strict global minimum  $\mathbf{x}^*$  of the cost function  $f(\mathbf{x})$  according to (2.43). The error function  $e(\mathbf{x})$  according to (2.50) converges linearly to 0 with the convergence rate*

$$e(\mathbf{x}_{k+1}) \leq \frac{\kappa - 1}{\kappa + 1} e(\mathbf{x}_k) . \quad (2.56)$$

*Here,  $\kappa = \lambda_{\max}/\lambda_{\min}$  denotes the spectral condition number of the matrix  $\mathbf{Q}$ , i. e., the ratio of the largest and the smallest (real and positive) eigenvalues  $\lambda_{\max}$  and  $\lambda_{\min}$ , respectively, of the matrix  $\mathbf{Q}$ .*

*Proof.* From Lemmas 2.1 and 2.2, it follows immediately that

$$e(\mathbf{x}_{k+1}) \leq \left(1 - \frac{4\lambda_{\min} \lambda_{\max}}{(\lambda_{\min} + \lambda_{\max})^2}\right)^{\frac{1}{2}} e(\mathbf{x}_k) = \frac{\lambda_{\max} - \lambda_{\min}}{\lambda_{\max} + \lambda_{\min}} e(\mathbf{x}_k) . \quad (2.57)$$

□

Theorem 2.7 can be interpreted as follows. Due to the positive definiteness of the matrix  $\mathbf{Q}$ , the contour lines ( $f(\mathbf{x}) = \text{konst.}$ ) of the cost function (2.43) are  $n$ -dimensional ellipsoids whose axes coincide with the directions of the  $n$  pairwise orthogonal eigenvectors of the matrix  $\mathbf{Q}$  and whose lengths are inversely proportional to the respective (positive real) eigenvalues. The gradient  $(\nabla f)(\mathbf{x}_k)$  is orthogonal to the contour line through the point  $\mathbf{x}_k$ , see Figures 2.5 and 2.6 for examples with  $\mathbf{x} \in \mathbb{R}^2$ . If all eigenvalues of  $\mathbf{Q}$  from (2.43) are of the same order of magnitude, the gradient method shows good convergence behavior. In case of  $\lambda_{\min} = \lambda_{\max}$ , meaning that  $\kappa = 1$ , the method even converges within a single step, see Figure 2.5. For poorly conditioned problems ( $\kappa$  very large), the gradient method can converge very slowly. This behavior is shown in Figure 2.6 and is called *zigzagging*.

The gradient method can of course also be applied to non-quadratic cost functions. The following theorem describes the convergence behavior of the gradient method for these

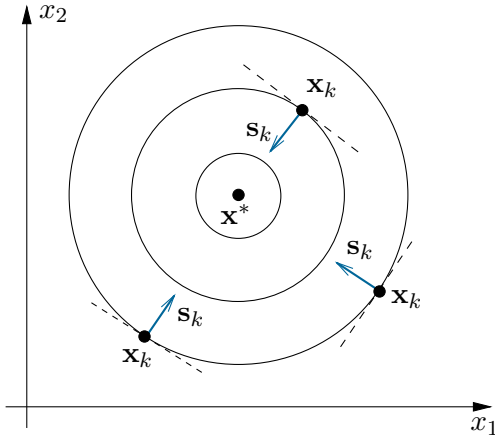


Fig. 2.5: Example of a well-conditioned problem for the gradient method.

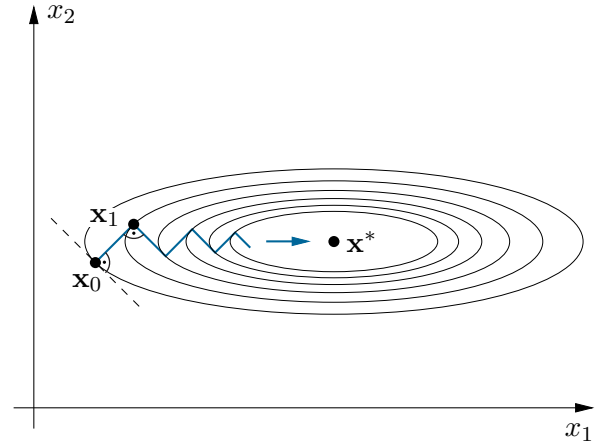


Fig. 2.6: Example of a poorly conditioned problem for the gradient method.

cases. The proof of this theorem can be found, e. g., in [2.2].

**Theorem 2.8** (Convergence of the gradient method for general cost functions). *Let the cost function  $f \in C^2$  be defined in  $\mathbb{R}^n$  with  $\mathbf{x}^*$  as a local minimum. Assume that the Hessian  $(\nabla^2 f)(\mathbf{x}^*)$  has the smallest and largest eigenvalues  $\lambda_{\min} > 0$  and  $\lambda_{\max} > 0$ , respectively, and the spectral condition number  $\kappa = \lambda_{\max}/\lambda_{\min}$ . If the sequence  $\{\mathbf{x}_k\}$  generated by the gradient method*

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k (\nabla f)(\mathbf{x}_k) \quad (2.58)$$

*with a suitable step size  $\alpha_k$  converges to the local minimum  $\mathbf{x}^*$ , then the sequence  $\{e(\mathbf{x}_k)\}$  with  $e(\mathbf{x}) = f(\mathbf{x}) - f(\mathbf{x}^*)$  converges linearly to 0 with a rate of convergence that is  $\left(\frac{\kappa-1}{\kappa+1}\right)^2$  in the best case.*

Problems that are poorly conditioned for the gradient method can sometimes be improved by appropriate *scaling* or *transformation*. The idea is based on the fact that the task of finding a minimum of the cost function  $f(\mathbf{x})$  is equivalent to searching for a minimum of the function  $h(\mathbf{y}) = f(\mathbf{T}\mathbf{y})$  with  $\mathbf{x} = \mathbf{T}\mathbf{y}$  and a regular matrix  $\mathbf{T} \in \mathbb{R}^{n \times n}$ . Expanding the function  $h(\mathbf{y})$  around the optimal point  $\mathbf{y}^* = \mathbf{T}^{-1}\mathbf{x}^*$  into a Taylor series,

$$\begin{aligned} h(\mathbf{y}) &= h(\mathbf{y}^*) + (\nabla h)^T(\mathbf{y}^*)(\mathbf{y} - \mathbf{y}^*) + \frac{1}{2}(\mathbf{y} - \mathbf{y}^*)^T (\nabla^2 h)(\mathbf{y}^*)(\mathbf{y} - \mathbf{y}^*) + \dots \\ &= h(\mathbf{y}^*) + (\nabla f)^T(\mathbf{x}^*)\mathbf{T}(\mathbf{y} - \mathbf{y}^*) \\ &\quad + \frac{1}{2}(\mathbf{y} - \mathbf{y}^*)^T \mathbf{T}^T (\nabla^2 f)(\mathbf{x}^*)\mathbf{T}(\mathbf{y} - \mathbf{y}^*) + \dots, \end{aligned} \quad (2.59)$$

shows that by appropriately choosing  $\mathbf{T}$  the distribution of the eigenvalues of the Hessian

$$(\nabla^2 h)(\mathbf{y}^*) = \mathbf{T}^T (\nabla^2 f)(\mathbf{x}^*) \mathbf{T} \quad (2.60)$$

can be improved compared to the distribution of the eigenvalues of  $(\nabla^2 f)(\mathbf{x}^*)$ . From (2.60), it follows that with the ideal choice  $\mathbf{T} = (\nabla^2 f)^{-\frac{1}{2}}(\mathbf{x}^*)$ , the Hessian  $(\nabla^2 h)(\mathbf{y}^*) = \mathbf{E}$  with the identity matrix  $\mathbf{E} \in \mathbb{R}^{n \times n}$  would follow and the gradient method would converge after one step for quadratic optimization problems (cf. Figure 2.5). In practice, this approach is very similar to Newton's method, which will be described later. However, this approach has the disadvantage that the entire Hessian has to be calculated explicitly, which is often a laborious task.

The advantages and disadvantages of the gradient method can be summarized as follows:

- + simple method
- + Hessian  $(\nabla^2 f)(\mathbf{x}_k)$  (computational effort  $\mathcal{O}(n^2)$ ) not required, only the gradient  $(\nabla f)(\mathbf{x}_k)$  (computational effort  $\mathcal{O}(n)$ ) is needed
- + convergence even for initial values that are further away from the minimum
- slow convergence for poorly conditioned and poorly scaled problems
- only linear order of convergence

### 2.3.2.2 Conjugate Gradient Method

The conjugate gradient method (abbreviated as CG method) tries to achieve faster convergence than the gradient method with only slightly increased computational load and memory requirements. More information on this method can be found, e. g., in [2.2, 2.4–2.6]. Originally, the method was developed for solving high-dimensional quadratic problems of the form (see also (2.43))

$$\min_{\mathbf{x} \in \mathbb{R}^n} f(\mathbf{x}) = \frac{1}{2} \mathbf{x}^T \mathbf{Q} \mathbf{x} - \mathbf{x}^T \mathbf{b} \quad (2.61)$$

and high-dimensional linear systems of equations of the form

$$\mathbf{0} = \mathbf{Q} \mathbf{x} - \mathbf{b} \quad (2.62)$$

each with a positive definite matrix  $\mathbf{Q} \in \mathbb{R}^{n \times n}$  and an arbitrary vector  $\mathbf{b} \in \mathbb{R}^n$ . Because the cost function (2.61) is strictly convex, (2.62) represents a sufficient first-order optimality criterion for a strict global minimum. If a linear system of equations

$$\mathbf{0} = \mathbf{A} \mathbf{x} - \mathbf{c} \quad (2.63)$$

with a regular but not positive definite matrix  $\mathbf{A} \in \mathbb{R}^{n \times n}$  and an arbitrary vector  $\mathbf{c} \in \mathbb{R}^n$  is to be solved, it can be transformed into

$$\mathbf{0} = \mathbf{A}^T \mathbf{A} \mathbf{x} - \mathbf{A}^T \mathbf{c} \quad (2.64)$$

by left multiplication with  $\mathbf{A}^T$ . Because  $\mathbf{A}^T \mathbf{A}$  is positive definite, (2.64) can also be solved using the conjugate gradient method.

Before explaining the conjugate gradient method in more detail, some basic principles are discussed.

**Definition 2.2 (Q-orthogonality).** Non-trivial vectors  $\mathbf{d}_j$ ,  $j = 0, \dots, k$  are called *conjugate with respect to a positive definite matrix  $\mathbf{Q}$*  or *Q-orthogonal*, if  $\mathbf{d}_i^T \mathbf{Q} \mathbf{d}_j = 0 \forall i \neq j$ . The vectors  $\mathbf{d}_j$ ,  $j = 0, \dots, k$  are then linearly independent.

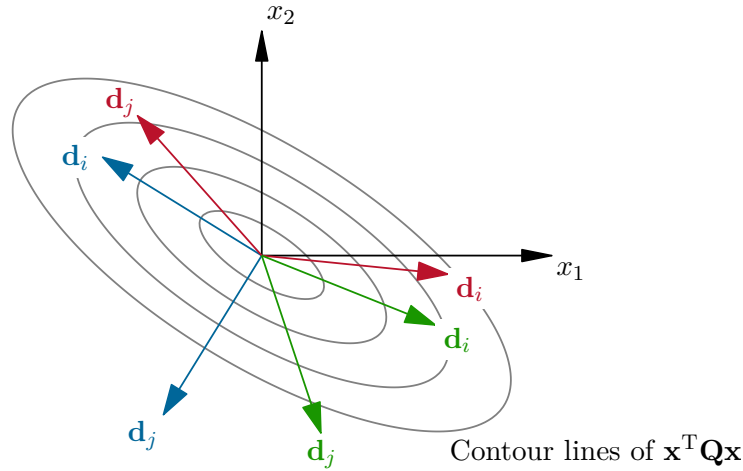


Figure 2.7: Examples of  $\mathbf{Q}$ -orthogonal vectors for  $n = 2$ . Equally colored vectors are  $\mathbf{Q}$ -orthogonal.

For  $\mathbf{Q} = \mathbf{E}$ , the concept of  $\mathbf{Q}$ -orthogonality thus coincides with the classical concept of orthogonality. In Figure 2.7, equally colored vectors are  $\mathbf{Q}$ -orthogonal.

**Exercise 2.6.** Prove that  $\mathbf{Q}$ -orthogonal vectors  $\mathbf{d}_j$ ,  $j = 0, \dots, k$  are linearly independent.

In the conjugate gradient method, starting from an initial solution  $\mathbf{x}_0$ , the optimal solution  $\mathbf{x}^*$  of the optimization problem (2.61) is iteratively searched for along  $\mathbf{Q}$ -orthogonal search directions  $\mathbf{d}_j$ . In each iteration step  $k$ , a new point

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{d}_k \quad (2.65)$$

with the optimal step size  $\alpha_k$  is computed. This yields

$$\mathbf{x}_{k+1} - \mathbf{x}_0 = \sum_{j=0}^k \alpha_j \mathbf{d}_j. \quad (2.66)$$

For the gradient of (2.61), the abbreviation

$$\mathbf{g}_k = (\nabla f)(\mathbf{x}_k) = \mathbf{Q} \mathbf{x}_k - \mathbf{b} \quad (2.67)$$

is introduced. It will now be shown that the optimal step sizes  $\alpha_j$  can be calculated with low mathematical effort. Based on (2.61), (2.66), and the  $\mathbf{Q}$ -orthogonality of the vectors

$\mathbf{d}_j$ , it follows that

$$\begin{aligned} f(\mathbf{x}_k) &= \frac{1}{2} \left( \mathbf{x}_0 + \sum_{j=0}^{k-1} \alpha_j \mathbf{d}_j \right)^T \mathbf{Q} \left( \mathbf{x}_0 + \sum_{j=0}^{k-1} \alpha_j \mathbf{d}_j \right) - \left( \mathbf{x}_0 + \sum_{j=0}^{k-1} \alpha_j \mathbf{d}_j \right)^T \mathbf{b} \\ &= \frac{1}{2} \mathbf{x}_0^T \mathbf{Q} \mathbf{x}_0 - \mathbf{x}_0^T \mathbf{b} + \sum_{j=0}^{k-1} \left( \frac{1}{2} \alpha_j^2 \mathbf{d}_j^T \mathbf{Q} \mathbf{d}_j + \alpha_j \mathbf{g}_0^T \mathbf{d}_j \right). \end{aligned} \quad (2.68)$$

This shows that the problem decomposes into independent scalar optimization problems for the step sizes  $\alpha_j$ . Their solution is

$$\alpha_j = -\frac{\mathbf{g}_0^T \mathbf{d}_j}{\mathbf{d}_j^T \mathbf{Q} \mathbf{d}_j} = -\frac{\mathbf{g}_j^T \mathbf{d}_j}{\mathbf{d}_j^T \mathbf{Q} \mathbf{d}_j}, \quad (2.69)$$

where the identity

$$\mathbf{g}_j^T \mathbf{d}_j = \left( \mathbf{Q} \left( \mathbf{x}_0 + \sum_{i=0}^{j-1} \alpha_i \mathbf{d}_i \right) - \mathbf{b} \right)^T \mathbf{d}_j = \mathbf{g}_0^T \mathbf{d}_j \quad (2.70)$$

was used. Thus, if in each iteration step (2.65), the step size  $\alpha_k$  is chosen according to (2.69), then  $\mathbf{x}_{k+1}$  minimizes the cost function  $f(\mathbf{x}_{k+1})$  in the subspace of  $\mathbb{R}^n$  spanned by the vectors  $\mathbf{d}_j$ ,  $j = 0, \dots, k$ . Because the vectors  $\mathbf{d}_j$ ,  $j = 0, \dots, k$  are linearly independent (see Definition 2.2), the optimal solution  $\mathbf{x}^* = \mathbf{x}_{k+1}$  of the quadratic optimization problem (2.61) is obtained after a maximum of  $k + 1 = n$  iteration steps (finite convergence). The algorithm of the conjugate gradient method described below shows how the still unknown vectors  $\mathbf{d}_j$  are chosen.

**Theorem 2.9 (Conjugate gradient method).** *For any initial value  $\mathbf{x}_0$ , the sequence*

$$\alpha_k = \frac{\mathbf{g}_k^T \mathbf{g}_k}{\mathbf{d}_k^T \mathbf{Q} \mathbf{d}_k} \quad (2.71a)$$

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{d}_k \quad (2.71b)$$

$$\beta_k = \frac{\mathbf{g}_{k+1}^T \mathbf{g}_{k+1}}{\mathbf{g}_k^T \mathbf{g}_k} \quad (2.71c)$$

$$\mathbf{d}_{k+1} = -\mathbf{g}_{k+1} + \beta_k \mathbf{d}_k \quad (2.71d)$$

with  $\mathbf{g}_k = \mathbf{Q} \mathbf{x}_k - \mathbf{b}$  and the initial value

$$\mathbf{d}_0 = -\mathbf{g}_0 = \mathbf{b} - \mathbf{Q} \mathbf{x}_0 \quad (2.72)$$

converges to the unique solution  $\mathbf{x}^*$  of the quadratic optimization problem (2.61) within  $n$  or less iteration steps. The search directions and gradients constructed in this iterative manner have the properties

$$\mathbf{g}_i^T \mathbf{g}_j = 0 \quad \forall i \neq j \quad (2.73a)$$

$$\mathbf{g}_i^T \mathbf{d}_j = 0 \quad \forall i > j \quad (2.73b)$$

$$\mathbf{d}_i^T \mathbf{Q} \mathbf{d}_j = 0 \quad \forall i \neq j . \quad (2.73c)$$

That is, the gradients are mutually orthogonal, the gradients are orthogonal to all previous search directions, and the search directions are  $\mathbf{Q}$ -orthogonal.

*Proof.* According to the iteration formula (2.71b), the following relation holds for the gradients.

$$\mathbf{g}_{k+1} = \mathbf{Q} \mathbf{x}_{k+1} - \mathbf{b} = \mathbf{Q} \mathbf{x}_k - \mathbf{b} + \alpha_k \mathbf{Q} \mathbf{d}_k = \mathbf{g}_k + \alpha_k \mathbf{Q} \mathbf{d}_k \quad (2.74)$$

In the following, the relations (2.71) and (2.72) are used continuously without explicitly mentioning them. First, we will show the properties (2.73) by complete induction. For the base case, using (2.74), we have

$$\mathbf{g}_0^T \mathbf{g}_1 = \mathbf{g}_0^T (\mathbf{g}_0 + \alpha_0 \mathbf{Q} \mathbf{d}_0) = \mathbf{g}_0^T \mathbf{g}_0 - \mathbf{d}_0^T \alpha_0 \mathbf{Q} \mathbf{d}_0 = 0 \quad (2.75a)$$

for (2.73a) and therefore

$$\mathbf{d}_0^T \mathbf{g}_1 = 0 \quad (2.75b)$$

for (2.73b). From (2.74) it follows that  $\mathbf{Q} \mathbf{d}_0 = (\mathbf{g}_1 - \mathbf{g}_0)/\alpha_0$ . Using this relation and (2.75a), the validity of (2.73c) for the base case follows in the form

$$\mathbf{d}_0^T \mathbf{Q} \mathbf{d}_1 = \mathbf{d}_0^T \mathbf{Q} (-\mathbf{g}_1 + \beta_0 \mathbf{d}_0) = \frac{1}{\alpha_0} (\mathbf{g}_1 - \mathbf{g}_0)^T \left( -\mathbf{g}_1 - \frac{\mathbf{g}_1^T \mathbf{g}_1}{\mathbf{g}_0^T \mathbf{g}_0} \mathbf{g}_0 \right) = 0 . \quad (2.75c)$$

Based on the induction hypothesis that (2.73) holds for a fixed index  $i$  for all  $j < i$ , we now show that (2.73) also holds for all  $j < i + 1$ . Here, (2.74) is used several times. From

$$\mathbf{g}_{i+1}^T \mathbf{g}_j = (\mathbf{g}_i + \alpha_i \mathbf{Q} \mathbf{d}_i)^T \mathbf{g}_j = \mathbf{g}_i^T \mathbf{g}_j + \alpha_i \mathbf{d}_i^T \mathbf{Q} (-\mathbf{d}_j + \beta_{j-1} \mathbf{d}_{j-1}) = 0 , \quad (2.76a)$$

it follows in the first step that (2.73a) holds for all  $j < i + 1$ . Using (2.76a) in

$$\begin{aligned} \mathbf{g}_{i+1}^T \mathbf{d}_j &= \mathbf{g}_{i+1}^T (-\mathbf{g}_j + \beta_{j-1} \mathbf{d}_{j-1}) \\ &= \mathbf{g}_{i+1}^T \beta_{j-1} \mathbf{d}_{j-1} = (\mathbf{g}_i + \alpha_i \mathbf{Q} \mathbf{d}_i)^T \beta_{j-1} \mathbf{d}_{j-1} = 0 \end{aligned} \quad (2.76b)$$

shows in the second step that (2.73b) holds for all  $j < i + 1$ . Using (2.76a) in

$$\begin{aligned} \mathbf{d}_{i+1}^T \mathbf{Q} \mathbf{d}_j &= (-\mathbf{g}_{i+1} + \beta_i \mathbf{d}_i)^T \mathbf{Q} \mathbf{d}_j = -\mathbf{g}_{i+1}^T \frac{1}{\alpha_j} (\mathbf{g}_{j+1} - \mathbf{g}_j) + \beta_i \mathbf{d}_i^T \mathbf{Q} \mathbf{d}_j \\ &= -\mathbf{g}_{i+1}^T \mathbf{g}_{j+1} \frac{\mathbf{d}_j^T \mathbf{Q} \mathbf{d}_j}{\mathbf{g}_j^T \mathbf{g}_j} + \beta_i \mathbf{d}_i^T \mathbf{Q} \mathbf{d}_j = 0 \end{aligned} \quad (2.76c)$$

shows in the third step that (2.73c) holds for all  $j < i + 1$ . This proves (2.73).

Because

$$\alpha_k = -\frac{\mathbf{g}_k^T \mathbf{d}_k}{\mathbf{d}_k^T \mathbf{Q} \mathbf{d}_k} = \frac{\mathbf{g}_k^T (\mathbf{g}_k - \beta_{k-1} \mathbf{d}_{k-1})}{\mathbf{d}_k^T \mathbf{Q} \mathbf{d}_k} = \frac{\mathbf{g}_k^T \mathbf{g}_k}{\mathbf{d}_k^T \mathbf{Q} \mathbf{d}_k} \quad (2.77)$$

(2.71a) yields the optimal step size according to (2.69). The fact that the conjugate gradient method converges to the optimal solution  $\mathbf{x}^*$  after a maximum of  $n$  iteration steps, given that the step sizes  $\alpha_k$  are chosen optimally and the vectors  $\mathbf{d}_k$ ,  $k = 0, \dots, n-1$  are  $\mathbf{Q}$ -orthogonal, already follows from the considerations preceding Theorem 2.9.  $\square$

**Exercise 2.7.** Show that the search directions  $\mathbf{d}_k$  of the conjugate gradient method according to Theorem 2.9 are descent directions.

Contrary to what the name *conjugate gradient method* suggests, it is not the local gradients  $\mathbf{g}_k$  of the cost function that are  $\mathbf{Q}$ -orthogonal in the sense of Definition 2.2 but rather the search directions  $\mathbf{d}_k$ . According to (2.71d), the vector  $\mathbf{d}_{k+1}$  forms a linear combination of the direction of steepest descent (negative gradient  $-\mathbf{g}_{k+1}$ ) and the previous search direction  $\mathbf{d}_k$ . The factor  $\beta_k$  is chosen according to (2.71c) in such a way that the  $\mathbf{Q}$ -orthogonality between  $\mathbf{d}_{k+1}$  and all previous search directions is ensured. Equation (2.71c) is also called the *Fletcher-Reeves formula*. Alternatively, the so-called *Hestenes-Stiefel formula*

$$\beta_k = \frac{(\mathbf{g}_{k+1} - \mathbf{g}_k)^T \mathbf{g}_{k+1}}{(\mathbf{g}_{k+1} - \mathbf{g}_k)^T \mathbf{d}_k} \quad (2.78)$$

can be used.

**Exercise 2.8.** Show the equivalence of (2.71c) and (2.78).

The algorithm of the conjugate gradient method given in Theorem 2.9 entails only small numerical effort. Specifically, only the matrix-vector product  $\mathbf{Q} \mathbf{d}_k$  needs to be calculated once in each iteration. The effort of this operation is significantly reduced if  $\mathbf{Q}$  is sparse. All other computational steps of the conjugate gradient method are elementary vector operations (inner product of two vectors, scaling of a vector, and sum of vectors). Moreover, except for the matrix  $\mathbf{Q}$ , only vectors and scalars need to be stored. As an alternative to (2.71a), the step size  $\alpha_k$  can also be determined using one of the methods described in Section 2.3.1, provided that this method actually yields the optimal step size. The explicit calculation of the gradient  $\mathbf{g}_k$  according to (2.67) can be replaced by an iterative calculation according to (2.74). Variations of the algorithm given in Theorem 2.9 can be found, e. g., in [2.4, 2.6, 2.7].

The following theorem (see [2.5, 2.8]) deals with the asymptotic convergence behavior of the conjugate gradient method.

**Theorem 2.10** (Convergence of the conjugate gradient method — quadratic cost function). *For any initial value  $\mathbf{x}_0 \in \mathbb{R}^n$ , the conjugate gradient method according to Theorem 2.9 converges to the strict global minimum  $\mathbf{x}^*$  of the quadratic cost function  $f(\mathbf{x})$  from (2.61). The error function  $e(\mathbf{x})$  according to (2.50) converges linearly to 0*

with the convergence rate

$$e(\mathbf{x}_{k+1}) \leq \frac{\sqrt{\kappa} - 1}{\sqrt{\kappa} + 1} e(\mathbf{x}_k) , \quad (2.79)$$

where  $\kappa = \lambda_{\max}/\lambda_{\min}$  again denotes the spectral condition number of the matrix  $\mathbf{Q}$ .

The conjugate gradient method is a so-called *Krylov subspace method*. To see this, the concept of a Krylov subspace is first defined.

**Definition 2.3 (Krylov subspace).** For a square matrix  $\mathbf{Q} \in \mathbb{R}^{n \times n}$  and a vector  $\mathbf{g} \in \mathbb{R}^n$ ,

$$\mathcal{K}_m(\mathbf{Q}, \mathbf{g}) = \text{span}\{\mathbf{g}, \mathbf{Q}\mathbf{g}, \dots, \mathbf{Q}^{m-1}\mathbf{g}\} \quad (2.80)$$

with  $m \leq n$  is called the Krylov subspace of order  $m$ .

A Krylov subspace method solves the linear system of equations  $\mathbf{0} = \mathbf{Q}\mathbf{x} - \mathbf{b}$  iteratively, starting from an initial solution  $\mathbf{x}_0$ , such that  $\mathbf{x}_k - \mathbf{x}_0$  lies in the Krylov subspace  $\mathcal{K}_k(\mathbf{Q}, \mathbf{g}_0)$  with  $\mathbf{g}_0 = \mathbf{Q}\mathbf{x}_0 - \mathbf{b}$  and that a norm of the residual  $\mathbf{g}_k = \mathbf{Q}\mathbf{x}_k - \mathbf{b}$  is minimized. In each iteration step, the dimension of the Krylov subspace increases by one.

**Theorem 2.11 (Conjugate gradient method as a Krylov subspace method).** The conjugate gradient method according to Theorem 2.9 is a Krylov subspace method where, starting from an initial solution  $\mathbf{x}_0$ , in each iteration step  $k > 0$ , the optimization problem

$$\min_{\mathbf{x}_k \in \mathbf{x}_0 + \mathcal{K}_k(\mathbf{Q}, \mathbf{g}_0)} \frac{1}{2} \mathbf{x}_k^T \mathbf{Q} \mathbf{x}_k - \mathbf{x}_k^T \mathbf{b} \quad (2.81)$$

with a symmetric positive definite matrix  $\mathbf{Q}$ , the Krylov subspace

$$\begin{aligned} \mathcal{K}_k(\mathbf{Q}, \mathbf{g}_0) &= \text{span}\{\mathbf{g}_0, \mathbf{Q}\mathbf{g}_0, \dots, \mathbf{Q}^{k-1}\mathbf{g}_0\} \\ &= \text{span}\{\mathbf{g}_0, \mathbf{g}_1, \dots, \mathbf{g}_{k-1}\} = \text{span}\{\mathbf{d}_0, \mathbf{d}_1, \dots, \mathbf{d}_{k-1}\} , \end{aligned} \quad (2.82)$$

and  $\mathbf{g}_0 = \mathbf{Q}\mathbf{x}_0 - \mathbf{b}$  is solved.

**Exercise 2.9.** Prove (2.82). This is particularly easy using complete induction. The rest of the proof of Theorem 2.11 follows from the considerations preceding Theorem 2.9.

An occasionally used extension of the algorithm according to Theorem 2.9 is the so-called *partial conjugate gradient method*. In this case, only  $m + 1 < n$  iteration steps are performed before the method is reinitialized at the point  $\mathbf{x}_{m+1}$  obtained in this way, i. e., the routine is restarted with the initial values  $\mathbf{x}_0 \leftarrow \mathbf{x}_{m+1}$  and  $\mathbf{d}_0 = \mathbf{b} - \mathbf{Q}\mathbf{x}_0$ . The following theorem, which is proven, e. g., in [2.2], describes the asymptotic convergence behavior of this method.

**Theorem 2.12 (Partial conjugate gradient method).** Consider the quadratic optimization problem (2.61). If the positive definite matrix  $\mathbf{Q}$  has  $n - m$  eigenvalues in

the interval  $[l, r]$  ( $l > 0$ ) and  $m$  eigenvalues that are greater than  $r$ , then the partial conjugate gradient method, which is restarted after every  $m + 1$  steps, shows the following convergence behavior for the error function  $e(\mathbf{x})$  according to (2.50).

$$e(\mathbf{x}_{k+1}) \leq \frac{r-l}{r+l} e(\mathbf{x}_k) \quad (2.83)$$

In the sense of this theorem, the point  $\mathbf{x}_{k+1}$  is reached from the initial point  $\mathbf{x}_k$  after  $m + 1$  intermediate iterations according to Theorem 2.9. For  $m = 0$  (pure gradient method), Theorem 2.12 yields the same statements as Theorem 2.7.

If the conjugate gradient method is to be used for *non-quadratic cost functions*  $f(\mathbf{x})$ , then only the substitutions

$$\mathbf{g}_k \leftrightarrow (\nabla f)(\mathbf{x}_k), \quad \mathbf{Q} \leftrightarrow (\nabla^2 f)(\mathbf{x}_k) \quad (2.84)$$

have to be made in Theorem 2.9. In this case, the algorithm will generally not terminate after a maximum of  $n$  steps. For non-quadratic cost functions  $f(\mathbf{x})$ , the application of the partial conjugate gradient method is often beneficial because the property of  $\mathbf{Q}$ -orthogonality of the search directions can be increasingly violated as the number of iterations grows. To avoid the computationally expensive calculation of the Hessian  $(\nabla^2 f)(\mathbf{x}_k)$ , the step size  $\alpha_k$  can alternatively be determined using one of the methods described in Section 2.3.1.

The advantages and disadvantages of the conjugate gradient method can be summarized as follows:

- + simple method, low computational cost and memory requirements, suitable for large optimization problems
- + only the gradient  $(\nabla f)(\mathbf{x}_k)$  (computational cost  $\mathcal{O}(n)$ ) is required
- + converges after a maximum of  $n$  iteration steps for quadratic optimization problems
- convergence behavior varies depending on the problem, convergence behavior is better than that of the gradient method

### 2.3.2.3 Newton's Method

The idea of Newton's method is to locally approximate the general cost function  $f(\mathbf{x})$  by a quadratic function and to minimize it. Expanding  $f(\mathbf{x}) = f(\mathbf{x}_k + \mathbf{s}_k)$  into a Taylor series around the iteration point  $\mathbf{x}_k$  and truncating the series after the quadratic term yields

$$f(\mathbf{x}) \approx f(\mathbf{x}_k) + \mathbf{s}_k^T (\nabla f)(\mathbf{x}_k) + \frac{1}{2} \mathbf{s}_k^T (\nabla^2 f)(\mathbf{x}_k) \mathbf{s}_k, \quad (2.85)$$

see Figure 2.8. The so-called *Newton direction*  $\mathbf{s}_k$  is obtained by minimizing the right-hand side of (2.85) with respect to  $\mathbf{s}_k$  in the form

$$\mathbf{s}_k = -(\nabla^2 f)^{-1}(\mathbf{x}_k) (\nabla f)(\mathbf{x}_k). \quad (2.86)$$

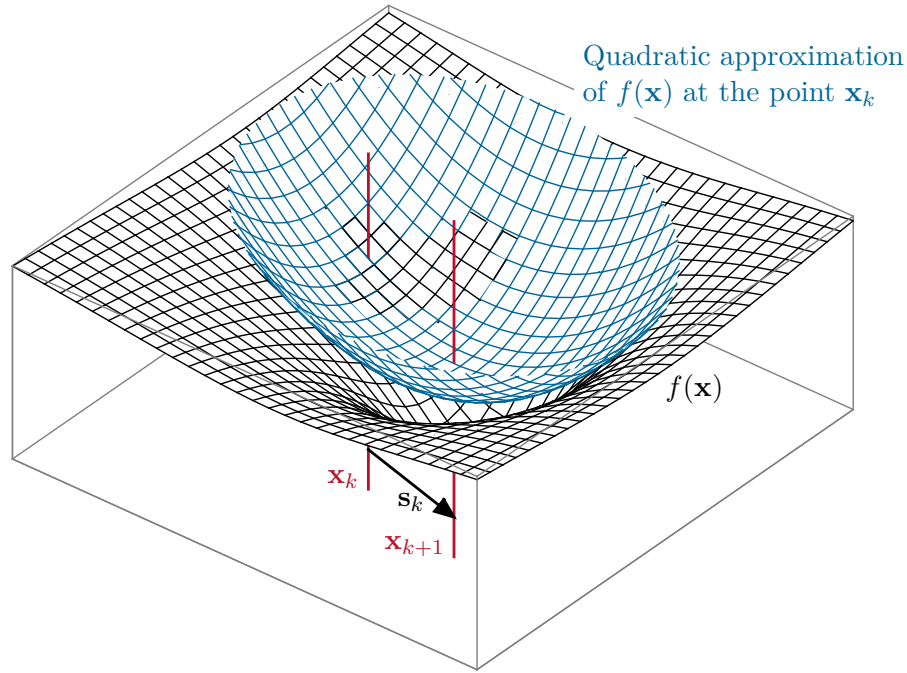


Figure 2.8: Local quadratic approximation of the cost function during a Newton iteration step (with step size  $\alpha_k = 1$ ).

If the Hessian  $(\nabla^2 f)(\mathbf{x}^*)$  at the minimum is positive definite, the inverse  $(\nabla^2 f)^{-1}(\mathbf{x}_k)$  exists in a neighborhood around the minimum and the method is well-defined. The calculation of  $\mathbf{s}_k$  according to (2.86) does not require an actual inversion of  $(\nabla^2 f)(\mathbf{x}_k)$ . The following theorem states the order of convergence of Newton's method. Its proof can be found, e. g., in [2.2, 2.5, 2.9].

**Theorem 2.13** (Order of convergence of Newton's method). *Let the cost function  $f \in C^3$  be defined in  $\mathbb{R}^n$  with the local minimum  $\mathbf{x}^*$ . If the Hessian  $(\nabla^2 f)(\mathbf{x}^*)$  is positive definite and the initial value  $\mathbf{x}_0$  lies in a sufficiently small neighborhood of the minimum, then the Newton iteration*

$$\mathbf{x}_{k+1} = \mathbf{x}_k - (\nabla^2 f)^{-1}(\mathbf{x}_k)(\nabla f)(\mathbf{x}_k) \quad (2.87)$$

*converges to the minimum  $\mathbf{x}^*$  with the order of convergence 2.*

**Exercise 2.10.** Show that a Newton iteration step according to (2.87) corresponds exactly to an iteration step of the Newton-Raphson method for solving the equation

$$(\nabla f)(\mathbf{x}^*) = \mathbf{0} , \quad (2.88)$$

i.e., the necessary first-order optimality condition.

For the practical application of the method, a suitable step size  $\alpha_k$  is introduced, so

that the iteration step is

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k \left( \nabla^2 f \right)^{-1}(\mathbf{x}_k) (\nabla f)(\mathbf{x}_k) \quad (2.89)$$

and the descent condition (2.29) is fulfilled. The method is then often called the *damped Newton's method* and  $\alpha_k$  is called the *damping parameter*. Sometimes the restriction  $\alpha_k \leq 1$  is used. It is expected that near the minimum, the optimal step size is  $\alpha_k \approx 1$ , which is why iterative step size searches typically start with the value  $\alpha_k = 1$ . Strategies for determining the step size  $\alpha_k$  have already been explained in Section 2.3.1.

A problem that occasionally arises in this context is the loss of positive definiteness of  $(\nabla^2 f)(\mathbf{x}_k)$  when  $\mathbf{x}_k$  is too far away from the minimum. Then the right-hand side of (2.85) has no or no strict minimum and the invertibility of  $(\nabla^2 f)(\mathbf{x}_k)$  may be lost. If  $(\nabla^2 f)(\mathbf{x}_k)$  is not positive definite, then instead of (2.89), the iteration step

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k \mathbf{M}_k (\nabla f)(\mathbf{x}_k), \quad \mathbf{M}_k = \left( (\nabla^2 f)(\mathbf{x}_k) + \varepsilon_k \mathbf{E} \right)^{-1} \quad (2.90)$$

with a suitable positive parameter  $\varepsilon_k$  can be used. For  $\varepsilon_k = 0$ , the iteration formula (2.90) reduces to Newton's method according to (2.87) and, for very large  $\varepsilon_k$ , it reduces to the gradient method according to (2.58). However, choosing a suitable value for  $\varepsilon_k$  may not be straightforward. Typically,  $\varepsilon_k$  is successively increased, starting from an initial value  $\varepsilon_k > 0$ , until the matrix  $(\nabla^2 f)(\mathbf{x}_k) + \varepsilon_k \mathbf{E}$  is positive definite.

**Exercise 2.11.** Show that Newton's method converges within a single iteration step for quadratic cost functions

$$\min_{\mathbf{x} \in \mathbb{R}^n} \frac{1}{2} \mathbf{x}^T \mathbf{Q} \mathbf{x} - \mathbf{b}^T \mathbf{x} \quad (2.91)$$

with the positive definite matrix  $\mathbf{Q}$ , regardless of the starting point  $\mathbf{x}_0$ .

The advantages and disadvantages of Newton's method can be summarized as follows:

- + order of convergence of 2 if the Hessian  $(\nabla^2 f)(\mathbf{x}_k)$  is positive definite, which is usually the case at least near the minimum  $\mathbf{x}^*$
- + convergence behavior better than that of the conjugate gradient method
- poorly suited in regions where  $(\nabla^2 f)(\mathbf{x}_k)$  is not positive definite
- computational effort  $\mathcal{O}(n^2)$  for the required Hessian  $(\nabla^2 f)(\mathbf{x}_k)$ , computational effort  $\mathcal{O}(n^3)$  for the search direction  $\mathbf{s}_k$

#### 2.3.2.4 Quasi-Newton Method

One of the main disadvantages of Newton's method is the computationally expensive calculation of the Hessian  $(\nabla^2 f)(\mathbf{x}_k)$ . To avoid this, the quasi-Newton method tries to iteratively determine the *inverse* Hessian. For the following, assume that the cost function  $f \in C^2$  and the abbreviation  $\mathbf{g}_k = (\nabla f)(\mathbf{x}_k)$  are used. Then, a Taylor series expansion yields the approximation

$$\mathbf{g}_{k+1} - \mathbf{g}_k \approx (\nabla^2 f)(\mathbf{x}_k) \mathbf{p}_k \quad (2.92)$$

with  $\mathbf{p}_k = \mathbf{x}_{k+1} - \mathbf{x}_k$ . Assuming that the Hessian  $(\nabla^2 f)(\mathbf{x}_k) = \mathbf{K}$  is constant, this gives

$$\mathbf{q}_k = \mathbf{g}_{k+1} - \mathbf{g}_k = \mathbf{K} \mathbf{p}_k . \quad (2.93)$$

If  $n$  linearly independent vectors  $\mathbf{p}_0, \mathbf{p}_1, \dots, \mathbf{p}_{n-1}$  with the corresponding vectors  $\mathbf{q}_0, \mathbf{q}_1, \dots, \mathbf{q}_{n-1}$  are available, the Hessian can be calculated in the form

$$\mathbf{K} = \begin{bmatrix} \mathbf{q}_0 & \mathbf{q}_1 & \dots & \mathbf{q}_{n-1} \end{bmatrix} \begin{bmatrix} \mathbf{p}_0 & \mathbf{p}_1 & \dots & \mathbf{p}_{n-1} \end{bmatrix}^{-1} . \quad (2.94)$$

The goal is to iteratively compute the inverse Hessian  $\mathbf{K}^{-1}$  in  $n$  iteration steps assuming a constant Hessian  $\mathbf{K}$ . In each step,

$$\mathbf{H}_{k+1} \mathbf{q}_j = \mathbf{p}_j, \quad j = 0, \dots, k \quad (2.95)$$

should be satisfied, which implies  $\mathbf{H}_n = \mathbf{K}^{-1}$ . This iterative computation can be done in different ways, one possibility is described below. Because the Hessian and its inverse are symmetric, it is natural to use a recursion formula that preserves this symmetry, e.g.,

$$\mathbf{H}_{k+1} = \mathbf{H}_k + \mathbf{z}_k \mathbf{z}_k^T . \quad (2.96)$$

The dyadic product  $\mathbf{z}_k \mathbf{z}_k^T$  has rank 1 (for  $\mathbf{z}_k \neq \mathbf{0}$ ), which is why this correction is also called a *rank 1 correction*. Substitution of (2.96) into (2.95) gives for  $j = k$

$$\mathbf{p}_k = \mathbf{H}_{k+1} \mathbf{q}_k = \mathbf{H}_k \mathbf{q}_k + \mathbf{z}_k \mathbf{z}_k^T \mathbf{q}_k \quad (2.97)$$

and after multiplication with  $\mathbf{q}_k$  the scalar product

$$\mathbf{q}_k^T \mathbf{p}_k = \mathbf{q}_k^T \mathbf{H}_k \mathbf{q}_k + (\mathbf{z}_k^T \mathbf{q}_k)^2 . \quad (2.98)$$

From (2.97) and (2.98), it follows that

$$\begin{aligned} (\mathbf{p}_k - \mathbf{H}_k \mathbf{q}_k)(\mathbf{p}_k - \mathbf{H}_k \mathbf{q}_k)^T &= \mathbf{z}_k \mathbf{z}_k^T \mathbf{q}_k (\mathbf{z}_k \mathbf{z}_k^T \mathbf{q}_k)^T = \mathbf{z}_k \mathbf{z}_k^T \mathbf{q}_k \mathbf{q}_k^T \mathbf{z}_k \mathbf{z}_k^T \\ &= \mathbf{z}_k \mathbf{z}_k^T (\mathbf{z}_k^T \mathbf{q}_k)^2 = \mathbf{z}_k \mathbf{z}_k^T \mathbf{q}_k^T (\mathbf{p}_k - \mathbf{H}_k \mathbf{q}_k) . \end{aligned} \quad (2.99)$$

Substituting the resulting expression for  $\mathbf{z}_k \mathbf{z}_k^T$  into (2.96) yields

$$\mathbf{H}_{k+1} = \mathbf{H}_k + \frac{(\mathbf{p}_k - \mathbf{H}_k \mathbf{q}_k)(\mathbf{p}_k - \mathbf{H}_k \mathbf{q}_k)^T}{\mathbf{q}_k^T (\mathbf{p}_k - \mathbf{H}_k \mathbf{q}_k)} . \quad (2.100)$$

This allows the formulation of the following theorem.

**Theorem 2.14** (Quasi-Newton method with rank 1 correction). Assume  $\mathbf{K}$  is a constant symmetric matrix and  $\mathbf{p}_0, \mathbf{p}_1, \dots, \mathbf{p}_k$  are linearly independent vectors. With  $\mathbf{q}_j = \mathbf{K} \mathbf{p}_j$ ,  $j = 0, \dots, k$ , for any symmetric initial matrix  $\mathbf{H}_0$  and the iteration formula

$$\mathbf{H}_{j+1} = \mathbf{H}_j + \frac{(\mathbf{p}_j - \mathbf{H}_j \mathbf{q}_j)(\mathbf{p}_j - \mathbf{H}_j \mathbf{q}_j)^T}{\mathbf{q}_j^T (\mathbf{p}_j - \mathbf{H}_j \mathbf{q}_j)} , \quad (2.101)$$

the relation

$$\mathbf{p}_j = \mathbf{H}_{k+1}\mathbf{q}_j, \quad j = 0, \dots, k \quad (2.102)$$

holds.

**Exercise 2.12.** Prove Theorem 2.14 by complete induction. **Hint:** This proof is sketched, e.g., in [2.9].

A major disadvantage of the rank 1 correction is that the positive definiteness of  $\mathbf{H}_{k+1}$  is only guaranteed if  $\mathbf{q}_k^T(\mathbf{p}_k - \mathbf{H}_k\mathbf{q}_k) > 0$  holds. For this reason, further iterative correction formulas for  $\mathbf{H}_k$  have been developed. Examples are the Davidon-Fletcher-Powell (DFP) correction formula

$$\mathbf{H}_{k+1} = \left( \mathbf{E} - \frac{\mathbf{H}_k\mathbf{q}_k\mathbf{q}_k^T}{\mathbf{q}_k^T\mathbf{H}_k\mathbf{q}_k} \right) \mathbf{H}_k \left( \mathbf{E} - \frac{\mathbf{q}_k\mathbf{q}_k^T\mathbf{H}_k}{\mathbf{q}_k^T\mathbf{H}_k\mathbf{q}_k} \right) + \frac{\mathbf{p}_k\mathbf{p}_k^T}{\mathbf{q}_k^T\mathbf{p}_k} \quad (2.103)$$

and the slightly more frequently used Broyden-Fletcher-Goldfarb-Shanno (BFGS) correction formula

$$\mathbf{H}_{k+1} = \left( \mathbf{E} - \frac{\mathbf{p}_k\mathbf{q}_k^T}{\mathbf{q}_k^T\mathbf{p}_k} \right) \mathbf{H}_k \left( \mathbf{E} - \frac{\mathbf{q}_k\mathbf{p}_k^T}{\mathbf{q}_k^T\mathbf{p}_k} \right) + \frac{\mathbf{p}_k\mathbf{p}_k^T}{\mathbf{q}_k^T\mathbf{p}_k}. \quad (2.104)$$

Both are called *rank 2 correction formulas* because the current approximation of the inverse Hessian is corrected by adding a matrix of rank 2.

**Exercise 2.13.** Show that the three iteration formulas (2.101), (2.103), and (2.104) yield  $\mathbf{H}_{k+1} = \mathbf{H}_k$  in the case  $\mathbf{p}_k = \mathbf{H}_k\mathbf{q}_k$  (i.e.,  $\mathbf{H}_k$  already fulfills the requirement (2.95) for  $\mathbf{H}_{k+1}$  with  $j = k$ ).

Linear combinations of the two correction formulas (2.103) and (2.104) in the form  $\mathbf{H}_{k+1} = \phi\mathbf{H}_{k+1}^{\text{DFP}} + (1 - \phi)\mathbf{H}_{k+1}^{\text{BFGS}}$  with  $\phi \in [0, 1]$  can also be used. All rank 2 correction formulas obtained in this way belong to the so-called *Broyden family*. Obviously, these correction formulas preserve the symmetry of  $\mathbf{H}_k$ .

Based on the current estimate  $\mathbf{H}_k$  of the inverse Hessian, the quasi-Newton method uses a search direction

$$\mathbf{s}_k = -\mathbf{H}_k(\nabla f)(\mathbf{x}_k). \quad (2.105)$$

Unlike (2.86) for Newton's method, (2.105) only requires knowledge of the gradient  $(\nabla f)(\mathbf{x}_k)$  and a matrix multiplication. The algorithm of the quasi-Newton method is summarized in Table 2.3 using the BFGS correction formula.

It can be shown (see [2.1, 2.4, 2.5]) that the correction formulas (2.103) and (2.104) preserve the positive definiteness of  $\mathbf{H}_k$  if

$$\mathbf{q}_k^T\mathbf{p}_k > 0 \quad (2.106)$$

holds. Fulfilling this condition can be ensured by appropriately choosing the step size  $\alpha_k > 0$  in  $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k\mathbf{s}_k$  or  $\mathbf{p}_k = \mathbf{x}_{k+1} - \mathbf{x}_k = \alpha_k\mathbf{s}_k$ . For example, if the chosen step size  $\alpha_k$  satisfies the Wolfe condition according to Section 2.3.1.3, then (2.106) is

---

**Initialization:**  $\mathbf{H}_0$  (initial value, positive definite matrix)  
 $k = 0$  (initial index)  
 $\mathbf{x}_0$  (initial solution)  
 $\mathbf{g}_0 = (\nabla f)(\mathbf{x}_0)$  (gradient at  $\mathbf{x}_0$ )  
 $\varepsilon_x$  (threshold for termination criterion)

**repeat**

Step 1: Calculate the search direction  $\mathbf{s}_k = -\mathbf{H}_k \mathbf{g}_k$

Step 2: Solve the minimization problem  $\min_{\alpha_k \geq 0} f(\mathbf{x}_k + \alpha_k \mathbf{s}_k)$

Step 3:  $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{s}_k$

$$\mathbf{p}_k = \mathbf{x}_{k+1} - \mathbf{x}_k = \alpha_k \mathbf{s}_k$$

$$\mathbf{g}_{k+1} = (\nabla f)(\mathbf{x}_{k+1})$$

$$\mathbf{q}_k = \mathbf{g}_{k+1} - \mathbf{g}_k$$

Step 4: BFGS correction formula

$$\mathbf{H}_{k+1} = \left( \mathbf{E} - \frac{\mathbf{p}_k \mathbf{q}_k^T}{\mathbf{q}_k^T \mathbf{p}_k} \right) \mathbf{H}_k \left( \mathbf{E} - \frac{\mathbf{q}_k \mathbf{p}_k^T}{\mathbf{q}_k^T \mathbf{p}_k} \right) + \frac{\mathbf{p}_k \mathbf{p}_k^T}{\mathbf{q}_k^T \mathbf{p}_k}$$

$$k \leftarrow k + 1$$

**until**  $\|\mathbf{x}_{k+1} - \mathbf{x}_k\| \leq \varepsilon_x$

---

Table 2.3: Quasi-Newton method with the BFGS correction formula.

automatically fulfilled. To see this, note that

$$g'(\alpha_k) = \mathbf{g}_{k+1}^T \mathbf{s}_k \geq \varepsilon_2 g'(0) = \varepsilon_2 \mathbf{g}_k^T \mathbf{s}_k \quad (2.107)$$

with  $0 < \varepsilon_2 < 1$  follows from (2.40). Based on this,

$$\mathbf{q}_k^T \mathbf{p}_k = \alpha_k \mathbf{q}_k^T \mathbf{s}_k = \alpha_k (\mathbf{g}_{k+1} - \mathbf{g}_k)^T \mathbf{s}_k \geq \underbrace{\alpha_k (\varepsilon_2 - 1) \mathbf{g}_k^T \mathbf{s}_k}_{> 0} \quad (2.108)$$

is obtained. The right-hand side of this inequality must be strictly positive (away from the optimal point  $\mathbf{x}^*$ ) because  $\mathbf{s}_k$  is a descent direction. Thus, (2.106) is fulfilled.

For unconstrained nonlinear optimization problems with a convex cost function  $f(\mathbf{x})$ , the quasi-Newton method converges with superlinear order of convergence. For the unconstrained quadratic optimization problem (2.61), the quasi-Newton method converges after a maximum of  $n$  iteration steps. If the method converges exactly after  $n$  iteration steps in this case, it can be shown (see [2.1]) that  $\mathbf{H}_n = (\nabla^2 f)^{-1}(\mathbf{x}^*) = \mathbf{Q}^{-1}$ , i. e., the algorithm yields the exact inverse Hessian.

The advantages and disadvantages of the quasi-Newton method can be summarized as follows:

+ simple method with moderate computational effort

- + only the gradient  $(\nabla f)(\mathbf{x}_k)$  (computational cost  $\mathcal{O}(n)$ ) is needed
- + converges after a maximum of  $n$  iteration steps for quadratic optimization problems
- + usually superlinear convergence behavior
- matrix  $\mathbf{H}_k$  must be stored (memory requirement  $\mathcal{O}(n^2)$ )

### 2.3.2.5 Gauss-Newton Method

In the Gauss-Newton method, the computationally complex calculation of the Hessian  $(\nabla^2 f)(\mathbf{x}_k)$  is replaced by a less computationally expensive approximation. This optimization method is only applicable if the cost function  $f(\mathbf{x})$  has the structure

$$f(\mathbf{x}) = \frac{1}{2} \|\mathbf{r}(\mathbf{x})\|_2^2 = \frac{1}{2} \sum_{i=1}^m r_i^2(\mathbf{x}) \quad (2.109)$$

with an arbitrary function  $\mathbf{r} : \mathbb{R}^n \rightarrow \mathbb{R}^m$ . The components  $r_i(\mathbf{x})$  of  $\mathbf{r}(\mathbf{x})$  are assumed to satisfy  $r_i \in C^2$ . In this case, the exact Hessian of  $f(\mathbf{x})$  is

$$(\nabla^2 f)(\mathbf{x}) = (\nabla \mathbf{r})(\mathbf{x})(\nabla \mathbf{r})^T(\mathbf{x}) + \sum_{i=1}^m r_i(\mathbf{x}) (\nabla^2 r_i)(\mathbf{x}), \quad (2.110)$$

where the columns of the Jacobian matrix  $(\nabla \mathbf{r})(\mathbf{x}) \in \mathbb{R}^{n \times m}$  contain the gradients  $(\nabla r_i)(\mathbf{x})$ .

**Exercise 2.14.** Verify (2.110) starting from (2.109).

In the Gauss-Newton method, the last term in (2.110) is neglected, resulting in the approximation

$$(\nabla^2 f)(\mathbf{x}) \approx (\nabla \mathbf{r})(\mathbf{x})(\nabla \mathbf{r})^T(\mathbf{x}). \quad (2.111)$$

The entailed approximation error is thus small if  $r_i(\mathbf{x})$  or  $(\nabla^2 r_i)(\mathbf{x}) \forall i = 1, \dots, m$  contain only values close to zero. For example, if it is known that  $f(\mathbf{x}^*) = 0$ , then it follows that  $r_i(\mathbf{x}^*) = 0 \forall i = 1, \dots, m$ . The case  $(\nabla^2 r_i)(\mathbf{x}) = \mathbf{0} \forall i = 1, \dots, m$  occurs if  $\mathbf{r}(\mathbf{x})$  is affine in  $\mathbf{x}$  (cf. Exercise 2.15).

Using the gradient  $(\nabla f)(\mathbf{x}) = (\nabla \mathbf{r})(\mathbf{x})\mathbf{r}(\mathbf{x})$  and the approximation (2.111), the Gauss-Newton step is

$$\begin{aligned} \mathbf{s}_k &= -((\nabla \mathbf{r})(\mathbf{x}_k)(\nabla \mathbf{r})^T(\mathbf{x}_k))^{-1}(\nabla f)(\mathbf{x}_k) \\ &= -((\nabla \mathbf{r})(\mathbf{x}_k)(\nabla \mathbf{r})^T(\mathbf{x}_k))^{-1}(\nabla \mathbf{r})(\mathbf{x}_k)\mathbf{r}(\mathbf{x}_k) \end{aligned} \quad (2.112)$$

and the iteration formula of the Gauss-Newton method is

$$\mathbf{x}_{k+1} = \mathbf{x}_k - ((\nabla \mathbf{r})(\mathbf{x}_k)(\nabla \mathbf{r})^T(\mathbf{x}_k))^{-1}(\nabla \mathbf{r})(\mathbf{x}_k)\mathbf{r}(\mathbf{x}_k). \quad (2.113)$$

In the iteration step  $k$ , the Gauss-Newton method thus exactly minimizes the local quadratic approximation

$$f(\mathbf{x}_k) + (\nabla f)^T(\mathbf{x}_k)(\mathbf{x} - \mathbf{x}_k) + \frac{1}{2}(\mathbf{x} - \mathbf{x}_k)^T(\nabla \mathbf{r})(\mathbf{x}_k)(\nabla \mathbf{r})^T(\mathbf{x}_k)(\mathbf{x} - \mathbf{x}_k) \quad (2.114)$$

of  $f(\mathbf{x})$  at the point  $\mathbf{x}_k$  with respect to  $\mathbf{x}$ . For the practical application of the method, a suitable step size  $\alpha_k$  can be introduced in (2.113), so that the iteration formula is

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k ((\nabla \mathbf{r})(\mathbf{x}_k)(\nabla \mathbf{r})^T(\mathbf{x}_k))^{-1} (\nabla \mathbf{r})(\mathbf{x}_k) \mathbf{r}(\mathbf{x}_k) \quad (2.115)$$

and the descent condition (2.29) is fulfilled. The method is then often called *damped Gauss-Newton method* and  $\alpha_k$  is called *damping parameter*. Sometimes the restriction  $\alpha_k \leq 1$  is used.

Note that the calculation of  $\mathbf{s}_k$  according to (2.112) does not actually require an inversion of  $(\nabla \mathbf{r})(\mathbf{x}_k)(\nabla \mathbf{r})^T(\mathbf{x}_k)$ . Choosing the search direction according to (2.112) only makes sense if the  $n \times n$  matrix  $(\nabla \mathbf{r})(\mathbf{x})(\nabla \mathbf{r})^T(\mathbf{x})$  is positive definite. If it is not positive definite (or poorly conditioned), similar to Newton's method, the alternative iteration formula

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k \mathbf{M}_k (\nabla f)(\mathbf{x}_k), \quad \mathbf{M}_k = \left( (\nabla \mathbf{r})(\mathbf{x}_k)(\nabla \mathbf{r})^T(\mathbf{x}_k) + \varepsilon_k \mathbf{E} \right)^{-1} \quad (2.116)$$

with a suitable positive parameter  $\varepsilon_k \geq 0$  can be used. The method is then called (for the case  $\alpha_k = 1$ ) *Levenberg-Marquardt method*. For  $\varepsilon_k \rightarrow 0$ , the iteration formula (2.116) reduces to the Gauss-Newton method according to (2.115) and for very large  $\varepsilon_k$  it reduces to the gradient method according to (2.58). Thus, the Levenberg-Marquardt method combines the behavior of the Gauss-Newton method with that of the gradient method. Often, the parameter  $\varepsilon_k$  is heuristically adapted, starting from a high initial value. For this purpose, the actual change  $f(\mathbf{x}_{k+1}) - f(\mathbf{x}_k)$  of the cost function is considered. If it is positive (no descent), the step is rejected and  $\varepsilon_k$  is increased (behavior becomes more similar to the gradient method). If it is negative, the step is accepted. If it also agrees well with the predicted (negative) change  $(\nabla f)^T(\mathbf{x}_k)(\mathbf{x}_{k+1} - \mathbf{x}_k) + \frac{1}{2}(\mathbf{x}_{k+1} - \mathbf{x}_k)^T (\nabla \mathbf{r})(\mathbf{x}_k)(\nabla \mathbf{r})^T(\mathbf{x}_k)(\mathbf{x}_{k+1} - \mathbf{x}_k)$  of the cost function according to (2.114),  $\varepsilon_k$  is decreased (behavior becomes more similar to the Gauss-Newton method). A similar heuristic, presented in more detail in Section 2.4.

The following theorem provides a statement about the order of convergence of the Gauss-Newton method. Its proof can be found, e. g., in [2.10].

**Theorem 2.15 (Order of convergence of the Gauss-Newton method).** *Let the cost function  $f(\mathbf{x}) = \frac{1}{2} \|\mathbf{r}(\mathbf{x})\|_2^2$  be given with  $\mathbf{r} \in C^2$  defined in  $\mathbb{R}^n$  and the local minimum  $\mathbf{x}^*$ . If the Jacobian matrix  $(\nabla \mathbf{r})(\mathbf{x}^*)$  has full row rank,  $\sum_{i=1}^m r_i(\mathbf{x}^*)(\nabla^2 r_i)(\mathbf{x}^*) = \mathbf{0}$  holds, and the initial point  $\mathbf{x}_0$  lies in a sufficiently small neighborhood of the minimum, the Gauss-Newton iteration*

$$\mathbf{x}_{k+1} = \mathbf{x}_k - ((\nabla \mathbf{r})(\mathbf{x}_k)(\nabla \mathbf{r})^T(\mathbf{x}_k))^{-1} (\nabla \mathbf{r})(\mathbf{x}_k) \mathbf{r}(\mathbf{x}_k) \quad (2.117)$$

*converges to the minimum  $\mathbf{x}^*$  with the order of convergence 2.*

If the term  $\sum_{i=1}^m r_i(\mathbf{x}^*)(\nabla^2 r_i)(\mathbf{x}^*)$  does not equal zero but is small compared to  $(\nabla \mathbf{r})(\mathbf{x}_k)(\nabla \mathbf{r})^T(\mathbf{x}_k)$ , then at least linear convergence behavior can be proven. However, if  $\sum_{i=1}^m r_i(\mathbf{x}^*)(\nabla^2 r_i)(\mathbf{x}^*)$  contains large absolute values, the Gauss-Newton method may not converge [2.10].

**Exercise 2.15.** Show that the Gauss-Newton method applied to the optimization problem

$$\min_{\mathbf{x} \in \mathbb{R}^n} \frac{1}{2} \|\mathbf{r}(\mathbf{x})\|_2^2 \quad \text{with} \quad \mathbf{r}(\mathbf{x}) = \mathbf{A}\mathbf{x} + \mathbf{b} \quad (2.118)$$

and a full column rank matrix  $\mathbf{A}$  (this implies  $m \geq n$ ) converges to the optimal point  $\mathbf{x}^* = -(\mathbf{A}^T \mathbf{A})^{-1} \mathbf{A}^T \mathbf{b}$  within a single iteration step (finite convergence), regardless of the starting point  $\mathbf{x}_0$ .

Cost functions as in (2.118) occur, for example, in the least squares method for a parametrically linear model (linear regression analysis). Cost functions of the form (2.109) occur, for example, in the least squares method for a parametrically nonlinear model (nonlinear regression analysis) [2.11].

**Exercise 2.16.** Show that the iterative solution of the nonlinear equation

$$\mathbf{r}(\mathbf{x}) = \mathbf{0} \quad (2.119)$$

with  $\mathbf{r} : \mathbb{R}^n \rightarrow \mathbb{R}^n$  and  $\mathbf{r} \in C^1$  using the Newton-Raphson method corresponds exactly to the Gauss-Newton method applied to the optimization problem

$$\min_{\mathbf{x} \in \mathbb{R}^n} \frac{1}{2} \|\mathbf{r}(\mathbf{x})\|_2^2. \quad (2.120)$$

**Solution of exercise 2.16.** For (2.119) to have a unique solution  $\mathbf{x}^*$ ,  $(\nabla \mathbf{r})^T(\mathbf{x}^*)$  must be regular. Because of  $\mathbf{r} \in C^1$ , this must also hold true for  $(\nabla \mathbf{r})^T(\mathbf{x}_k)$  in a neighborhood of  $\mathbf{x}^*$ . The iteration formula of the Newton-Raphson method is

$$\mathbf{x}_{k+1} = \mathbf{x}_k - ((\nabla \mathbf{r})^T(\mathbf{x}_k))^{-1} \mathbf{r}(\mathbf{x}_k). \quad (2.121)$$

The iteration formula (2.113) of the Gauss-Newton method simplifies to

$$\begin{aligned} \mathbf{x}_{k+1} &= \mathbf{x}_k - ((\nabla \mathbf{r})(\mathbf{x}_k)(\nabla \mathbf{r})^T(\mathbf{x}_k))^{-1} (\nabla \mathbf{r})(\mathbf{x}_k) \mathbf{r}(\mathbf{x}_k) \\ &= \mathbf{x}_k - ((\nabla \mathbf{r})^T(\mathbf{x}_k))^{-1} \mathbf{r}(\mathbf{x}_k) \end{aligned} \quad (2.122)$$

given that  $(\nabla \mathbf{r})^T(\mathbf{x}_k)$  is regular.

The advantages and disadvantages of the Gauss-Newton method can be summarized as follows:

- + no second derivatives needed, only the Jacobian matrix  $(\nabla \mathbf{r})(\mathbf{x}_k)$  is required
- + converges in the first iteration step if  $\mathbf{r}(\mathbf{x})$  is affine in  $\mathbf{x}$
- + under certain conditions, quadratic convergence behavior can be achieved
- convergence depends on the particular problem and is not guaranteed
- computational effort  $\mathcal{O}(mn)$  for the Jacobian matrix  $(\nabla \mathbf{r})(\mathbf{x}_k)$ , computational effort  $\mathcal{O}(n^3)$  for the search direction  $\mathbf{s}_k$

## 2.4 Trust Region Method

In line search methods, a suitable descent direction (search direction)  $\mathbf{s}_k$  (e.g., the *negative gradient* at the point  $\mathbf{x}_k$  according to (2.41) in the gradient method or the *Newton direction* according to (2.86) in Newton's method) is chosen and then the (optimal) step size  $\alpha_k > 0$  along this descent direction is determined by the scalar optimization problem (2.30). In the trust region method, the cost function  $f(\mathbf{x})$  to be minimized is approximated in the neighborhood of  $\mathbf{x}_k$  by a quadratic function  $m_k$ , i. e.,

$$f(\mathbf{x}_k + \mathbf{s}_k) \approx m_k(\mathbf{s}_k) = f(\mathbf{x}_k) + \mathbf{s}_k^T (\nabla f)(\mathbf{x}_k) + \frac{1}{2} \mathbf{s}_k^T \mathbf{B}_k \mathbf{s}_k \quad (2.123)$$

with a suitable symmetric matrix  $\mathbf{B}_k$ . Generally,  $\mathbf{B}_k$  should be an approximation of the Hessian  $(\nabla^2 f)(\mathbf{x}_k)$  that can be calculated with little effort. The approximation error associated with  $m_k(\mathbf{s}_k)$  is of the order  $\|\mathbf{s}_k\|^2$  or even  $\|\mathbf{s}_k\|^3$  if  $\mathbf{B}_k$  equals the Hessian  $(\nabla^2 f)(\mathbf{x}_k)$ . The *trust region* is characterized by  $\|\mathbf{s}_k\|_2 \leq \Delta_k$  with a scalar parameter  $\Delta_k$  and defines a neighborhood around the point  $\mathbf{x}_k$  in which the cost function  $f(\mathbf{x}_k + \mathbf{s}_k)$  is described by the quadratic function  $m_k(\mathbf{s}_k)$  with sufficient accuracy. In each iteration step, the *constrained* optimization problem

$$\min_{\mathbf{s}_k \in \mathbb{R}^n} \quad m_k(\mathbf{s}_k) = f(\mathbf{x}_k) + \mathbf{s}_k^T (\nabla f)(\mathbf{x}_k) + \frac{1}{2} \mathbf{s}_k^T \mathbf{B}_k \mathbf{s}_k \quad (2.124a)$$

$$\text{u.B.v.} \quad \|\mathbf{s}_k\|_2 \leq \Delta_k \quad (2.124b)$$

is solved for a suitable value  $\Delta_k > 0$  and the update  $\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{s}_k$  is used. Note that (in contrast to most line search methods) the descent direction and the step size are determined *simultaneously*.

A major degree of freedom in the design of this method is the choice of  $\Delta_k$ . For this purpose, in each iteration step, *the agreement of the quadratic function  $m_k$  with the cost function  $f$*  is assessed based on the ratio

$$\rho_k(\mathbf{s}_k) = \frac{f(\mathbf{x}_k) - f(\mathbf{x}_k + \mathbf{s}_k)}{m_k(\mathbf{0}) - m_k(\mathbf{s}_k)}. \quad (2.125)$$

The numerator term in (2.125) describes the *actual reduction* of the cost function, while the denominator term represents the *predicted reduction*. The denominator is always greater than or equal to zero because  $\mathbf{s}_k$  minimizes the function  $m_k$  according to (2.124) within the trust region. If  $\rho_k(\mathbf{s}_k) < 0$ , the value of the cost function at the next iteration point  $f(\mathbf{x}_k + \mathbf{s}_k)$  would be greater than at the previous iteration point  $f(\mathbf{x}_k)$ , which is why this iteration step is rejected and the trust region is made smaller. If however  $\rho_k(\mathbf{s}_k) \approx 1$ , the trust region can be enlarged because the cost function  $f(\mathbf{x}_k)$  is well described by the quadratic function in this case. If  $\rho_k(\mathbf{s}_k)$  is positive and significantly smaller than 1, the trust region is made smaller in the next step.

An algorithm for the trust region method is listed in Table 2.4. Note that  $\bar{\Delta}$  represents an upper bound for  $\Delta_k$  and that the trust region is only enlarged in the next iteration step if  $\mathbf{s}_k$  was limited by the boundary of the trust region (condition  $\|\mathbf{s}_k\|_2 = \Delta_k$ ).

Suggestions for the iteration formula of the matrix  $\mathbf{B}_k$ , which is not shown in Table 2.4, are described, e. g., in [2.1, 2.5, 2.9]. If the cost function  $f(\mathbf{x}_k)$  can be represented in

---

|                        |                                                                                  |                                               |
|------------------------|----------------------------------------------------------------------------------|-----------------------------------------------|
| <b>Initialization:</b> | $\bar{\Delta}, \Delta_0 \in (0, \bar{\Delta})$                                   | (trust region: maximum & initial value)       |
|                        | $\eta \in [0, \frac{1}{4})$                                                      | (parameter)                                   |
|                        | $k \leftarrow 0$                                                                 | (initial index)                               |
|                        | $\mathbf{x}_0$                                                                   | (initial solution)                            |
|                        | $\varepsilon_x$                                                                  | (threshold for termination criterion)         |
| <b>repeat</b>          |                                                                                  |                                               |
|                        | $m_k(\mathbf{s}_k)$ according to (2.123)                                         | (model)                                       |
|                        | $\mathbf{s}_k$ solution of (2.124)                                               | (maybe approximately solved)                  |
|                        | $\rho_k$ according to (2.125)                                                    | (quality of the model)                        |
|                        | <b>if</b> $\rho_k < \frac{1}{4}$                                                 |                                               |
|                        | $\Delta_{k+1} \leftarrow \frac{1}{4} \Delta_k$                                   | (reduction)                                   |
|                        | <b>else if</b> $\rho_k > \frac{3}{4}$ <b>and</b> $\ \mathbf{s}_k\ _2 = \Delta_k$ |                                               |
|                        | $\Delta_{k+1} \leftarrow \min\{2\Delta_k, \bar{\Delta}\}$                        | (enlargement)                                 |
|                        | <b>else</b>                                                                      |                                               |
|                        | $\Delta_{k+1} \leftarrow \Delta_k$                                               |                                               |
|                        | <b>end if</b>                                                                    |                                               |
|                        | <b>if</b> $\rho_k > \eta$                                                        |                                               |
|                        | $\mathbf{x}_{k+1} \leftarrow \mathbf{x}_k + \mathbf{s}_k$                        | (next step)                                   |
|                        | $\mathbf{B}_{k+1} \leftarrow \mathbf{B}_k + \dots$                               | (approximation of the Hessian)                |
|                        | <b>else</b>                                                                      |                                               |
|                        | $\mathbf{x}_{k+1} \leftarrow \mathbf{x}_k$                                       | (repeat step with $\Delta_{k+1} < \Delta_k$ ) |
|                        | <b>end if</b>                                                                    |                                               |
|                        | $k \leftarrow k + 1$                                                             |                                               |
| <b>until</b>           | $\ \mathbf{x}_k - \mathbf{x}_{k-1}\ _2 \leq \varepsilon_x$                       |                                               |

---

Table 2.4: Trust region method.

the form (2.109), i. e.,  $f(\mathbf{x}) = \frac{1}{2} \|\mathbf{r}(\mathbf{x})\|_2^2$ , then  $\mathbf{B}_k$  can be calculated as defined in (2.111) for the Gauss-Newton method, i. e.,  $\mathbf{B}_k = (\nabla \mathbf{r})(\mathbf{x}_k)(\nabla \mathbf{r})^T(\mathbf{x}_k)$ . With this choice, the trust region method is exactly equivalent to the Levenberg-Marquardt method [2.10]. To show this, the inequality constraint in (2.124) is equivalently rewritten, resulting in the optimization problem

$$\min_{\mathbf{s}_k \in \mathbb{R}^n} \quad m_k(\mathbf{s}_k) = f(\mathbf{x}_k) + \mathbf{s}_k^T (\nabla f)(\mathbf{x}_k) + \frac{1}{2} \mathbf{s}_k^T (\nabla \mathbf{r})(\mathbf{x}_k) (\nabla \mathbf{r})^T(\mathbf{x}_k) \mathbf{s}_k \quad (2.126a)$$

$$\text{u.B.v.} \quad h(\mathbf{s}_k) = \frac{1}{2} (\mathbf{s}_k^T \mathbf{s}_k - \Delta_k^2) \leq 0. \quad (2.126b)$$

Based on Theorem ??, necessary optimality conditions for the *constrained* optimization problem (2.126) are

$$(\nabla f)(\mathbf{x}_k) + (\nabla \mathbf{r})(\mathbf{x}_k)(\nabla \mathbf{r})^T(\mathbf{x}_k)\mathbf{s}_k^* + \varepsilon_k^*\mathbf{s}_k^* = \mathbf{0} \quad (2.127a)$$

$$\varepsilon_k^* \geq 0 \quad (2.127b)$$

$$\varepsilon_k^* h(\mathbf{s}_k^*) = 0 . \quad (2.127c)$$

with a multiplier  $\varepsilon_k^*$  still to be determined. Using (2.126b) and (2.127),  $\mathbf{s}_k^*$  and  $\varepsilon_k^*$  can be uniquely calculated. This yields

$$\mathbf{s}_k^* = -\left((\nabla \mathbf{r})(\mathbf{x}_k)(\nabla \mathbf{r})^T(\mathbf{x}_k) + \varepsilon_k^*\mathbf{E}\right)^{-1}(\nabla f)(\mathbf{x}_k) , \quad (2.128)$$

which exactly corresponds to the Levenberg-Marquardt method (cf. (2.116) with  $\alpha_k = 1$ ). In the trust region method, the parameter  $\Delta_k$  is chosen. In the Levenberg-Marquardt method, the parameter  $\varepsilon_k$  is chosen. These parameters can be converted into each other using (2.126b) and (2.127). It holds that  $\varepsilon_k = 0$  for  $\Delta_k \rightarrow \infty$  and  $\varepsilon_k \rightarrow \infty$  for  $\Delta_k \rightarrow 0$ .

The so-called *dogleg method*, summarized in Table 2.5, calculates an approximate solution of the optimization problem (2.124). It combines the behavior of Newton's method with that of the gradient method. If an unconstrained gradient step would cause an exit from the trust region,  $\mathbf{s}_k$  is chosen by restricting the gradient step to the trust region. If an unconstrained Newton step does not cause an exit from the trust region,  $\mathbf{s}_k$  is chosen according to this step. Otherwise, a linear interpolation between the unconstrained gradient step and the unconstrained Newton step is computed such that  $\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{s}_k$  lies on the boundary of the trust region.

---

|                                                                                                                                                                         |                               |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------|
| $m_k(\mathbf{s}_k)$ according to (2.123)                                                                                                                                | (model)                       |
| $\mathbf{s}_{G,k} = -\frac{(\nabla f)(\mathbf{x}_k)^T(\nabla f)(\mathbf{x}_k)}{(\nabla f)(\mathbf{x}_k)^T\mathbf{B}_k(\nabla f)(\mathbf{x}_k)}(\nabla f)(\mathbf{x}_k)$ | (unconstrained gradient step) |
| <b>if</b> $\ \mathbf{s}_{G,k}\ _2 \geq \Delta_k$                                                                                                                        |                               |
| $\mathbf{s}_k = \Delta_k \frac{\mathbf{s}_{G,k}}{\ \mathbf{s}_{G,k}\ _2}$                                                                                               |                               |
| <b>else</b>                                                                                                                                                             |                               |
| $\mathbf{s}_{N,k} = -\mathbf{B}_k^{-1}(\nabla f)(\mathbf{x}_k)$                                                                                                         | (unconstrained Newton step)   |
| <b>if</b> $\ \mathbf{s}_{N,k}\ _2 \leq \Delta_k$                                                                                                                        |                               |
| $\mathbf{s}_k = \mathbf{s}_{N,k}$                                                                                                                                       |                               |
| <b>else</b>                                                                                                                                                             |                               |
| $\mathbf{s}_k = \beta_k \mathbf{s}_{G,k} + (1 - \beta_k) \mathbf{s}_{N,k}$                                                                                              | (interpolation)               |
| with $\beta_k \in (0, 1)$ such that $\ \mathbf{s}_k\ _2 = \Delta_k$                                                                                                     |                               |
| <b>end if</b>                                                                                                                                                           |                               |
| <b>end if</b>                                                                                                                                                           |                               |

---

Table 2.5: Approximate solution of (2.124) using the dogleg method.

For an example with  $n = 2$ , Figure 2.9 shows the contour lines of a local quadratic function  $m_k(\mathbf{s}_k)$  in a neighborhood of  $\mathbf{x}_k$ . The green line contains the approximate

solutions  $\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{s}_k$  of the dogleg method for different values of  $\Delta_k \in (0, \infty)$ . For comparison, the red line shows the exact solutions of the optimization problem (2.124) for different values of  $\Delta_k \in (0, \infty)$ . In the case  $\mathbf{B}_k = (\nabla \mathbf{r})(\mathbf{x}_k)(\nabla \mathbf{r})^T(\mathbf{x}_k)$ , the red line naturally also coincides with the solutions of the Levenberg-Marquardt method.

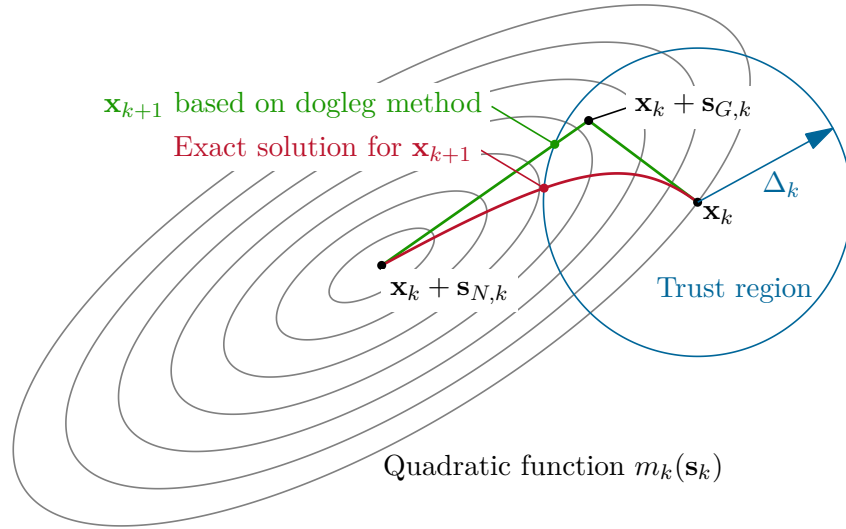


Figure 2.9: Solution of the optimization problem (2.124) during an iteration of the trust region method, comparison between the exact solution (red) and the dogleg method (green).

**Note 2.1.** The solutions of a nonlinear equation

$$\mathbf{r}(\mathbf{x}) = \mathbf{0} \quad (2.129)$$

with  $\mathbf{r} : \mathbb{R}^n \rightarrow \mathbb{R}^n$  can be calculated by solving the optimization problem

$$\min_{\mathbf{x} \in \mathbb{R}^n} \frac{1}{2} \|\mathbf{r}(\mathbf{x})\|_2^2 \quad (2.130)$$

(see also Exercise 2.16). This possibility for solving nonlinear equations is used by the MATLAB function `fsolve` from the Optimization Toolbox. The function `fsolve` uses the trust region method by default, and it can be chosen between the basic variant, the dogleg method, and the Levenberg-Marquardt method (see the documentation of the function in MATLAB).

## 2.5 Direct Search Methods

The so-called *derivative-based* solution methods, which were considered so far, use the gradient  $(\nabla f)(\mathbf{x}_k)$  (and sometimes the Hessian  $(\nabla^2 f)(\mathbf{x}_k)$ ) to determine a new point  $\mathbf{x}_{k+1}$  using a suitable iteration formula. The goal is to achieve a sufficiently large reduction of the cost function  $f(\mathbf{x}_{k+1}) < f(\mathbf{x}_k)$ .

However, in some practical cases, the derivatives required for this approach are not available or cannot be calculated with reasonable effort because the problem under consideration is *too complex* or *not continuously differentiable*. In this case, so-called *direct* or *derivative-free search methods* can be used, which compute samples of function values at a series of points and use them to determine a new iteration point  $\mathbf{x}_{k+1}$ .

The *simplex method* according to *Nelder and Mead* is a well-known and simple heuristic method in nonlinear optimization. It is fundamentally different from the simplex algorithm in *linear programming* and should not be confused with it.

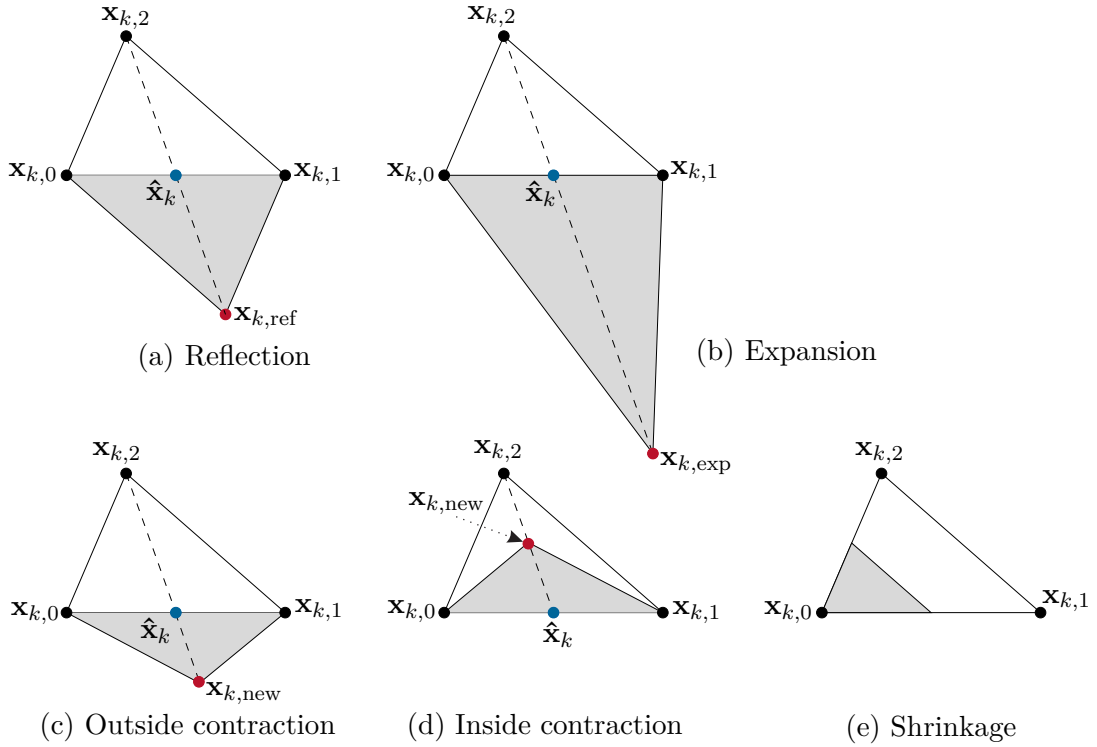


Figure 2.10: Steps of the Nelder-Mead simplex method for  $\beta = 1$ ,  $\gamma = 1$ , and  $\theta = 1/2$ .

The algorithm is based on the progression of a so-called *simplex* in the  $n$ -dimensional space of optimization variables. In this context, a simplex is defined as the convex hull spanned by  $n + 1$  points  $\mathbf{x}_{k,i}$ ,  $i = 0, \dots, n$  in the  $n$ -dimensional search space (for  $n = 1$  this is a line, for  $n = 2$  a triangle, etc.). The index  $k$  refers to the iteration step. In the following,  $\mathbf{x}_{k,\min}$  and  $\mathbf{x}_{k,\max}$  denote those vertices of the simplex which have the lowest and highest cost function value  $f$ , i.e.,

$$\begin{aligned}\mathbf{x}_{k,\min} &= \arg \min_{i=0,\dots,n} f(\mathbf{x}_{k,i}) \\ \mathbf{x}_{k,\max} &= \arg \max_{i=0,\dots,n} f(\mathbf{x}_{k,i}) .\end{aligned}\tag{2.131}$$

The *centroid*  $\hat{\mathbf{x}}_k$  of all vertices of the simplex excluding  $\mathbf{x}_{k,\max}$  is calculated as

$$\hat{\mathbf{x}}_k = \frac{1}{n} \left( \sum_{i=0}^n \mathbf{x}_{k,i} - \mathbf{x}_{k,\max} \right) . \quad (2.132)$$

The algorithm is based on the idea of replacing the point  $\mathbf{x}_{k,\max}$  in the simplex with another point having a lower cost function value. An important first operation is the calculation of the *reflection point*

$$\mathbf{x}_{k,\text{ref}} = \hat{\mathbf{x}}_k + (\hat{\mathbf{x}}_k - \mathbf{x}_{k,\max}) , \quad (2.133)$$

which lies on a line through the points  $\mathbf{x}_{k,\max}$  and  $\hat{\mathbf{x}}_k$  such that  $\hat{\mathbf{x}}_k$  is halfway between  $\mathbf{x}_{k,\text{ref}}$  and  $\mathbf{x}_{k,\max}$ , see Figure 2.10(a). Depending on  $f(\mathbf{x}_{k,\text{ref}})$  compared to the cost function values of the other points of the simplex, a new point  $\mathbf{x}_{k,\text{new}}$  is constructed which replaces the point  $\mathbf{x}_{k,\max}$  in the next iteration step. A basic version of the algorithm is listed in Table 2.6 and the different steps are visualized in Figure 2.10. Note that the shrinking in the algorithm according to Table 2.6 is always performed with respect to the vertex  $\mathbf{x}_{k,\min}$  with the smallest cost function value, i. e., in Figure 2.10(e),  $\mathbf{x}_{k,0} = \mathbf{x}_{k,\min}$ . During the iteration, the simplex moves towards the optimum and successively contracts. However, convergence to an optimal point is generally not guaranteed and the simplex method may converge to a *non-optimal point*. However, in practice, the simplex method often leads to good results and acceptable convergence behavior.

---

|                        |                                                                                                                                   |                                                                             |
|------------------------|-----------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------|
| <b>Initialization:</b> | $\mathbf{x}_{0,i}, i = 0, \dots, n$                                                                                               | (initial simplex)                                                           |
|                        | $k \leftarrow 0$                                                                                                                  | (iteration index)                                                           |
|                        | $\beta > 0$                                                                                                                       | (reflection coefficient, typically $\beta = 1$ )                            |
|                        | $\gamma > 0$                                                                                                                      | (expansion coefficient, typically $\gamma = 1$ )                            |
|                        | $\theta \in (0, 1)$                                                                                                               | (contraction coefficient, typically $\theta = 1/2$ )                        |
|                        | $\varepsilon_x$                                                                                                                   | (threshold for termination criterion)                                       |
| <b>repeat</b>          |                                                                                                                                   |                                                                             |
|                        | $\mathbf{x}_{k,\min}, \mathbf{x}_{k,\max}$                                                                                        | according to (2.131) (points with minimum and maximum cost function values) |
|                        | $\hat{\mathbf{x}}_k$                                                                                                              | according to (2.132) (centroid)                                             |
|                        | $\mathbf{x}_{k,\text{ref}} = \hat{\mathbf{x}}_k + \beta(\hat{\mathbf{x}}_k - \mathbf{x}_{k,\max})$                                | (reflection step, Fig. 2.10(a))                                             |
|                        | <b>if</b> $f(\mathbf{x}_{k,\text{ref}}) < f(\mathbf{x}_{k,\min})$                                                                 |                                                                             |
|                        | $\mathbf{x}_{k,\text{exp}} = \mathbf{x}_{k,\text{ref}} + \gamma(\mathbf{x}_{k,\text{ref}} - \hat{\mathbf{x}}_k)$                  | (expansion step, Fig. 2.10(b))                                              |
|                        | <b>if</b> $f(\mathbf{x}_{k,\text{exp}}) < f(\mathbf{x}_{k,\text{ref}})$                                                           |                                                                             |
|                        | $\mathbf{x}_{k,\text{new}} = \mathbf{x}_{k,\text{exp}}$                                                                           |                                                                             |
|                        | <b>else</b>                                                                                                                       |                                                                             |
|                        | $\mathbf{x}_{k,\text{new}} = \mathbf{x}_{k,\text{ref}}$                                                                           | (keep reflection point)                                                     |
|                        | <b>end if</b>                                                                                                                     |                                                                             |
|                        | <b>else if</b> $f(\mathbf{x}_{k,\text{ref}}) > \max_{\mathbf{x}_{k,i} \neq \mathbf{x}_{k,\max}, i=0,\dots,n} f(\mathbf{x}_{k,i})$ |                                                                             |
|                        | <b>if</b> $f(\mathbf{x}_{k,\max}) \leq f(\mathbf{x}_{k,\text{ref}})$                                                              |                                                                             |
|                        | $\mathbf{x}_{k,\text{new}} = \theta \mathbf{x}_{k,\max} + (1 - \theta) \hat{\mathbf{x}}_k$                                        | (inside contraction, Fig. 2.10(d))                                          |
|                        | <b>else</b>                                                                                                                       |                                                                             |
|                        | $\mathbf{x}_{k,\text{new}} = \theta \mathbf{x}_{k,\text{ref}} + (1 - \theta) \hat{\mathbf{x}}_k$                                  | (outside contraction, Fig. 2.10(c))                                         |
|                        | <b>end if</b>                                                                                                                     |                                                                             |
|                        | <b>else</b>                                                                                                                       |                                                                             |
|                        | $\mathbf{x}_{k,\text{new}} = \mathbf{x}_{k,\text{ref}}$                                                                           | (keep reflection point)                                                     |
|                        | <b>end if</b>                                                                                                                     |                                                                             |
|                        | <b>if</b> $f(\mathbf{x}_{k,\text{new}}) \geq f(\mathbf{x}_{k,\max})$                                                              | (poss. for non-conv. cost functions)                                        |
|                        | $\mathbf{x}_{k+1,i} \leftarrow \frac{1}{2}(\mathbf{x}_{k,i} + \mathbf{x}_{k,\min}), i = 0, \dots, n$                              | (shrinkage, Fig. 2.10(e))                                                   |
|                        | <b>else</b>                                                                                                                       |                                                                             |
|                        | $\mathbf{x}_{k,\max} \leftarrow \mathbf{x}_{k,\text{new}}$                                                                        |                                                                             |
|                        | $\mathbf{x}_{k+1,i} \leftarrow \mathbf{x}_{k,i}, i = 0, \dots, n$                                                                 |                                                                             |
|                        | <b>end if</b>                                                                                                                     |                                                                             |
|                        | $k \leftarrow k + 1$                                                                                                              |                                                                             |
| <b>until</b>           | $\max_{i=0,\dots,n} \ \mathbf{x}_{k,i} - \mathbf{x}_{k,\min}\ _\infty \leq \varepsilon_x$                                         |                                                                             |

---

Table 2.6: Nelder-Mead simplex method.

## 2.6 Example: Rosenbrock's "Banana Function"

The *Rosenbrock* problem

$$\min_{\mathbf{x} \in \mathbb{R}^2} f(\mathbf{x}) \quad \text{with} \quad f(\mathbf{x}) = 100(x_2 - x_1^2)^2 + (x_1 - 1)^2 \quad (2.134)$$

is a well-known tutorial optimization problem. Figure 2.11 shows the shape and the contour lines of the function, which is also called the *banana function*. The Rosenbrock problem will be used as an example to numerically analyze the *convergence properties of the discussed methods* using MATLAB.

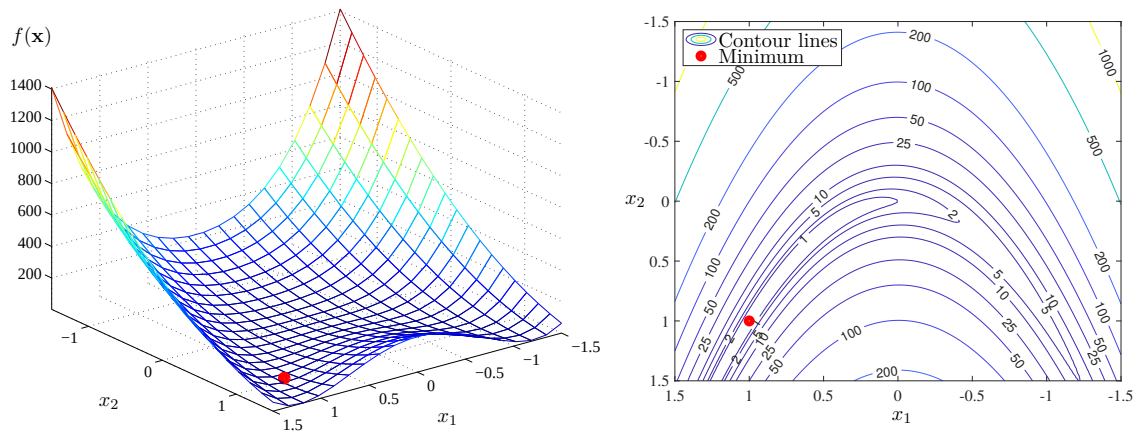


Figure 2.11: Shape and contour lines of Rosenbrock's banana function.

**Exercise 2.17.** Verify that the point  $\mathbf{x}^* = [1 \ 1]^T$  represents a minimum. Is  $\mathbf{x}^*$  a global and strict minimum? Are there other local minima? Are the function  $f(\mathbf{x})$  and the optimization problem (2.134) convex?

For solving unconstrained optimization problems, the *Optimization Toolbox* of MATLAB provides the following functions:

- **fminunc:** Line search: gradient method, quasi-Newton method  
Trust region method: Newton's method
- **fminsearch:** Nelder-Mead simplex method.

A recommendable alternative is the freely available MATLAB function

- **minFunc:** Line search: gradient method, conjugate gradient method, quasi-Newton method, Newton's method

Table 2.7 shows a comparison of numerical solutions of the Rosenbrock problem (starting from the initial value  $\mathbf{x}_0 = [-1 \ -1]^T$ ) computed using **fminunc**, **fminsearch**, and **minFunc**. Moreover, Figure 2.13 shows the iteration histories for some of the methods. In the code listing 2.1 at the end of this section, the MATLAB code for the Rosenbrock problem (2.134) is given. It demonstrates how the individual optimization methods can be called with the commands **fminunc**, **fminsearch**, and **minFunc**.

The *gradient method* shows *slow convergence*. Even after reaching the maximum number of function evaluations of 200, the minimum is still not found. In Figure 2.12, the iteration history of the gradient method is shown over the contour lines of the Rosenbrock function (2.134). The gradient method uses the direction of steepest descent, which is

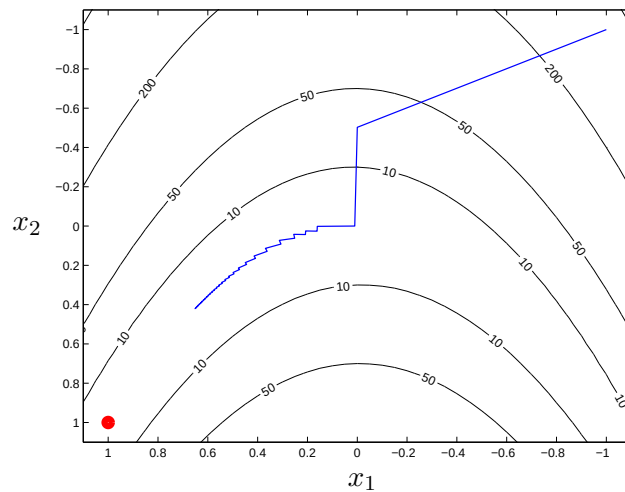


Figure 2.12: Iterations of the gradient method.

| Method                     | Function   | Iter. | $f(\mathbf{x}^*)$    | $\ (\nabla f)(\mathbf{x}^*)\ $ | Function calls  |                          |                            |
|----------------------------|------------|-------|----------------------|--------------------------------|-----------------|--------------------------|----------------------------|
|                            |            |       |                      |                                | $f(\mathbf{x})$ | $(\nabla f)(\mathbf{x})$ | $(\nabla^2 f)(\mathbf{x})$ |
| LS: gradient method        | fminunc    | 57    | 0.1232               | 1.1978                         | 200             | 200                      | –                          |
| LS: conj. gradient method  | minFunc    | 28    | $7.4 \cdot 10^{-21}$ | $7.7 \cdot 10^{-11}$           | 65              | 65                       | –                          |
| LS: Newton's method        | minFunc    | 20    | $3.8 \cdot 10^{-16}$ | $7.3 \cdot 10^{-7}$            | 32              | 32                       | 26                         |
| LS: quasi-Newton (BFGS)    | fminunc    | 23    | $5.4 \cdot 10^{-12}$ | $9.2 \cdot 10^{-6}$            | 29              | 29                       | –                          |
| LS: Gauss-Newton method    | minFunc    | 4     | $1.5 \cdot 10^{-29}$ | $7.8 \cdot 10^{-15}$           | 9               | 9                        | –                          |
| TR: Newton's method        | fminunc    | 25    | $2.2 \cdot 10^{-18}$ | $2.1 \cdot 10^{-8}$            | 26              | 26                       | 26                         |
| Nelder-Mead simplex method | fminsearch | 67    | $5.3 \cdot 10^{-10}$ | –                              | 125             | –                        | –                          |

Table 2.7: Comparison of numerical methods applied to the Rosenbrock problem (LS=line search, TR=trust region method).

orthogonal to the respective contour line. The length of the iteration steps decreases towards the minimum. The low convergence speed has already been illustrated in Figure 2.6 and will be investigated in more detail in the following exercise.

**Exercise 2.18.** Calculate the rate of convergence of the gradient method according to Theorem 2.8 at the minimum  $\mathbf{x}^* = [1 \ 1]^T$  of the Rosenbrock problem (2.134).

The convergence behavior of the *quasi-Newton method* in Figure 2.13 is significantly better than that of the gradient method. The *Newton's method (trust region method)* in Figure 2.13 starts initially in the 'wrong' direction, which can be explained by the *quadratic approximation* (2.123) at the starting point  $\mathbf{x}_0 = [-1 \ -1]^T$ , whose minimum is near

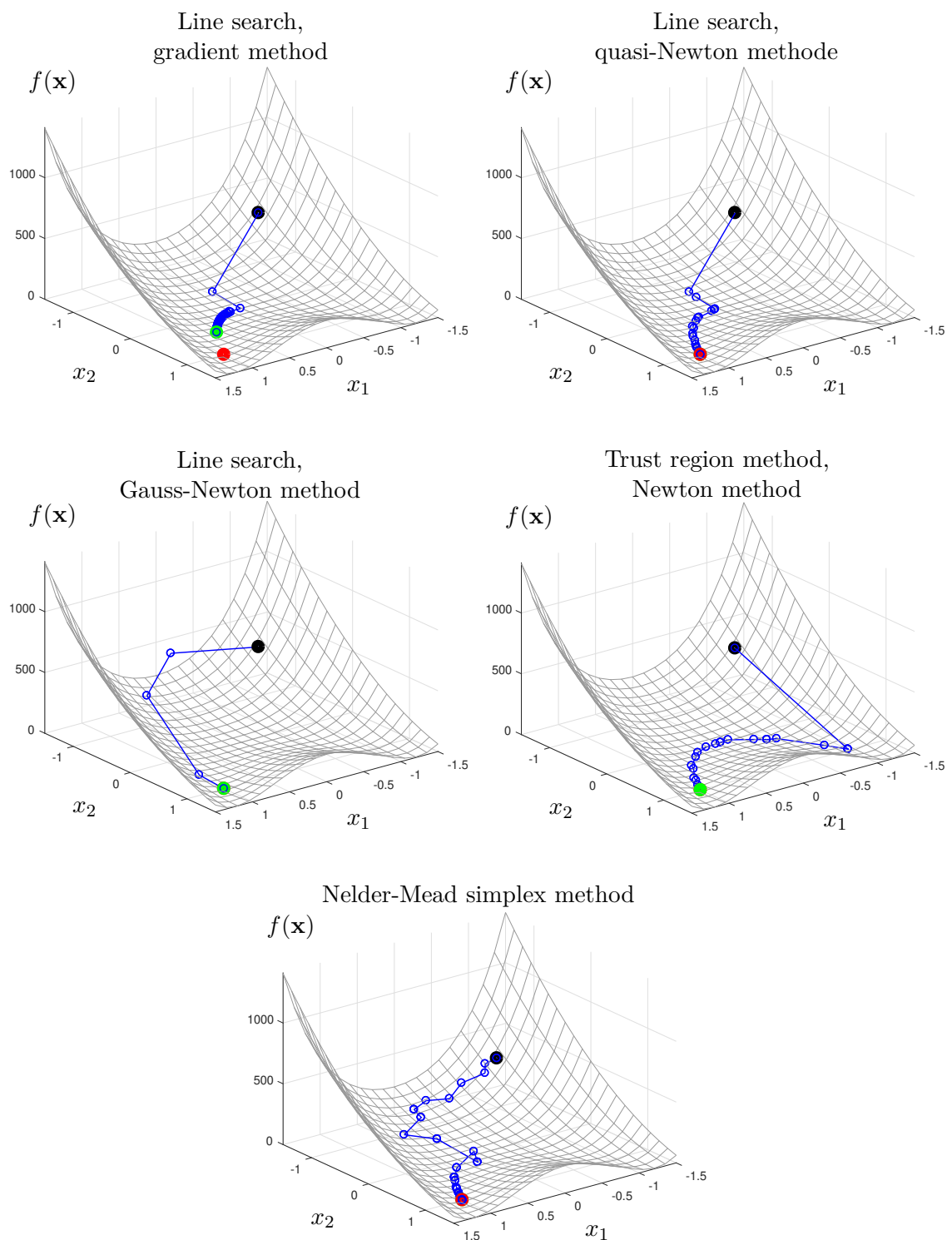


Figure 2.13: Rosenbrock's banana function: comparison of numerical methods with `fminunc`, `fminsearch`, and `minFunc`.

$\mathbf{x} \approx [-1 \ 1]^T$ . However, the individual steps along the valley of the Rosenbrock function are significantly larger because Newton's method *explicitly uses the Hessian* and does not rely on an approximation as in the case of the quasi-Newton method. The *Gauss-Newton method* shows a very good convergence behavior for the Rosenbrock function. Despite the approximation of the Hessian, the minimum can be found with a large step size even in the valley of the cost function. This particularly good behavior is due to the form of the cost function, which contains dominant terms that are affine in  $\mathbf{x}$  (cf. Exercise 2.15).

Table 2.7 and Figure 2.13 also show the results of the *Nelder-Mead simplex method*, which is implemented in the MATLAB function `fminsearch`. The graphical output of `fminsearch` however does not allow to display the individual simplices. The following exercise investigates the simplex method in more detail to get an impression of the simplex operations and the robustness of the method.

**Exercise 2.19.** Write a MATLAB function that numerically solves the Rosenbrock problem (2.134) using the Nelder-Mead simplex method (see the algorithm in Table 2.6). For first simplex, use the vertices

$$\mathbf{x}_{0,1} = \begin{bmatrix} -1 \\ -1 \end{bmatrix}, \quad \mathbf{x}_{0,2} = \mathbf{x}_{0,1} + s \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad \mathbf{x}_{0,3} = \mathbf{x}_{0,1} + s \begin{bmatrix} 0 \\ 1 \end{bmatrix} \quad (2.135)$$

with the side length  $s = 1$ . Use the termination condition in Table 2.6 with the tolerance  $\varepsilon_f = 10^{-9}$  and compare your results with those of `fminsearch` in Table 2.7. Graphically display the simplices obtained in the individual iterations. Investigate the robustness and convergence behavior of the simplex method for different side lengths  $s$  of the initial simplex and different starting points  $\mathbf{x}_{0,1}$ .

**Exercise 2.20.** Write a MATLAB function that numerically solves the Rosenbrock problem (2.134) using Newton's method (line search), see Section 2.3.2.3. Use the heuristic conditions from Section 2.3.1.3 to choose the step size  $\alpha_k$ . Compare the observed performance with the values given in Table 2.7. Graphically display the individual iterations and vary the starting points  $\mathbf{x}_0$ .

Code Listing 2.1: MATLAB code for the Rosenbrock problem (`fminunc`, `fminsearch`, `minFunc`).

```
function Xopt = rosenbrock_problem(Xinit,methodQ)
% Xinit: initial point
% methodQ: 1 - fminunc: line search, gradient method
%           2 - fminunc: line search, quasi-Newton method
%           3 - fminunc: trust region method, Newton method
%           4 - fminsearch: direct search, Nelder-Mead simplex method
%           5 - minFunc: line search, conjugate gradient method
%           6 - minFunc: line search, Newton method
%           7 - minFunc: line search, Gauss-Newton method
global old
old = [Xinit; rosenbrock(Xinit)];

% options for fminunc:
opt_fminu = optimoptions('fminunc','Display','iter','PlotFcn',@plot_iterates);
% options for fminsearch:
```

```

opt_fmins = optimset('Display','iter','PlotFcn',@plot_iterates);
% options for minFunc:
opt=struct('display','iter','outputFcn',@(x,state,varargin) plot_iterates(x,[],state));

% Info: 'SpecifyObjectiveGradient' defines that the function which computes the objective value
% also returns the gradient. If the property 'HessianFcn' is set to 'objective', the Hessian
% must also be returned.
switch methodQ
case 1 % fminunc: line search, gradient method
    opt_fminu = optimoptions(opt_fminu,'Algorithm','quasi-newton','HessUpdate','steepdesc',...
        'SpecifyObjectiveGradient',true);
    [Xopt,fopt,exitflag,output] = fminunc(@rosenbrock,Xinit,opt_fminu);
case 2 % fminunc: line search, quasi-Newton method
    opt_fminu = optimoptions(opt_fminu,'Algorithm','quasi-newton','HessUpdate','bfgs',...
        'SpecifyObjectiveGradient',true);
    [Xopt,fopt,exitflag,output] = fminunc(@rosenbrock,Xinit,opt_fminu);
case 3 % fminunc: trust region method, Newton method
    opt_fminu = optimoptions(opt_fminu,'Algorithm','trust-region','HessianFcn','objective',...
        'SpecifyObjectiveGradient',true);
    [Xopt,fopt,exitflag,output] = fminunc(@rosenbrock,Xinit,opt_fminu);
case 4 % fminsearch: direct search, Nelder-Mead simplex method
    [Xopt,fopt,exitflag,output] = fminsearch(@rosenbrock,Xinit,opt_fmins);
case 5 % minFunc: line search, conjugate gradient method
    figure
    opt.Method = 'cg';
    [Xopt,fopt,exitflag,output] = minFunc(@rosenbrock,Xinit,opt);
case 6 % minFunc: line search, Newton method
    figure
    opt.Method = 'newton';
    [Xopt,fopt,exitflag,output] = minFunc(@rosenbrock,Xinit,opt);
case 7 % minFunc: line search, Gauss-Newton method
    figure
    opt.Method = 'newton';
    [Xopt,fopt,exitflag,output] = minFunc(@rosenbrock_gauss_newton,Xinit,opt);
end
end

function [f, grad, H] = rosenbrock(x)
    grad = [];
    H = [];
    f = 100*(x(2)-x(1)^2)^2 + (x(1)-1)^2; % Rosenbrock cost function
    if nargin>1 % if gradient is requested
        grad = [ -400*(x(2)-x(1)^2)*x(1)+2*(x(1)-1);
            200*(x(2)-x(1)^2) ];
    end
    if nargin>2 % if Hessian is requested
        H = [ -400*(x(2)-3*x(1)^2)+2, -400*x(1);
            -400*x(1), 200 ];
    end
end

function [f, grad, H]=rosenbrock_gauss_newton(x)
% Rosenbrock cost function computed as squared norm of a vector function r(x):
% f(x)=100*(x2-x1^2)^2+(x1-1)^2 = 1/2*norm(r(x))^2
% with
% r1(x)=sqrt(200)*(x2-x1^2)
% r2(x)=sqrt(2)*(x1-1)
r=[sqrt(200)*(x(2)-x(1)^2);
    sqrt(2)*(x(1)-1)];
f=1/2*norm(r)^2;
if nargin>1
    grad_r=[-sqrt(200)*2*x(1),sqrt(2);
        sqrt(200),0];

```

```

        grad=grad_r*r;
    end
    if nargout>2
        H=grad_r*grad_r';
    end
end

function stop = plot_iterates(x,info,state)
    global old
    f = rosenbrock(x);
    switch state
        case 'init'
            plot_surface(x,f);
        case 'iter'
            plot3([old(1),x(1)],[old(2),x(2)],[old(3),f],'b-o','LineWidth',1);
        case 'done'
            plot3(x(1),x(2),f,'go','LineWidth',5);
    end
    stop = false;
    old = [x;f];
end

function plot_surface(x,f)
    [X1,X2] = meshgrid(-1.5:0.15:1.5);
    F = 100*(X2-X1.^2).^2 + (X1-1).^2;
    h = surf(X1,X2,F,'EdgeColor',0.6*[1,1,1],'FaceColor','none');
    hold on; axis tight;
    plot3(x(1),x(2),f,'ko','LineWidth',5);
    plot3(1,1,0,'ro','LineWidth',5);
    xlabel('x_1'); ylabel('x_2'); zlabel('f')
    set(gcf,'ToolBar','figure');
    set(gca,'Xdir','reverse','Ydir','reverse');
    set(gca,'clipping','off');
end

```

## 2.7 References

- [2.1] D. P. Bertsekas, *Nonlinear Programming*, 2nd ed. Athena Scientific, 1999.
- [2.2] D. G. Luenberger and Y. Ye, *Linear and Nonlinear Programming* (International Series in Operations Research & Management Science), 3rd ed. Springer, 2008, vol. 116.
- [2.3] P. E. Gill, W. Murray, and M. H. Wright, *Practical Optimization*. Academic Press, 1981.
- [2.4] R. Fletcher, *Practical methods of optimization*, 2nd ed. John Wiley & Sons, 1987.
- [2.5] I. Griva, S. Nash, and A. Sofer, *Linear and Nonlinear Optimization*, 2nd ed. Society for Industrial and Applied Mathematics, 2009.
- [2.6] E. Chong and S. Żak, *An Introduction to Optimization* (Wiley Series in Discrete Mathematics and Optimization), 4th ed. Wiley, 2013.
- [2.7] M. Hestenes and E. Stiefel, “Methods of conjugate gradients for solving linear systems,” *Journal of Research of the National Bureau of Standards*, vol. 49, no. 6, pp. 409–436, 1952.
- [2.8] J. Shewchuk, *An introduction to the conjugate gradient method without the agonizing pain*, Technical Report CMU-CS-94-125, Pittsburgh, PA: School of Computer Science, Carnegie Mellon University, 1994.
- [2.9] J. Nocedal and S. J. Wright, *Numerical Optimization* (Springer Series in Operations Research and Financial Engineering), 2nd ed. Springer, 2006.
- [2.10] J. Dennis and R. Schnabel, *Numerical Methods for Unconstrained Optimization and Nonlinear Equations* (Classics in Applied Mathematics). Society for Industrial and Applied Mathematics, 1996.
- [2.11] M. Papageorgiou, M. Leibold, and M. Buss, *Optimierung: Statische, dynamische, stochastische Verfahren für die Anwendung*, 4th ed. Springer, 2015.
- [2.12] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004.
- [2.13] C. T. Kelley, *Iterative Methods for Optimization*. Society for Industrial and Applied Mathematics, 1999.
- [2.14] B. C. Chachuat, “Nonlinear and Dynamic Optimization: From Theory to Practice,” abrufbar unter <http://infoscience.epfl.ch/record/111939>, Laboratoire d’Automatique, École Polytechnique Fédérale de Lausanne, 2007. Accessed: Sep. 30, 2020.
- [2.15] H. T. Jongen, K. Meer, and E. Triesch, *Optimization Theory*. Kluwer Academic Publishers, 2004.