Chapter 2
Introduction to Nonlinear Programming

2.1 Introduction

In this chapter, we review some important theory on mathematical optimization (also known as mathematical programming) which provides direct motivation for some numerical algorithms. However, highly efficient algorithms must also account for particular properties and structures of the problems. There are many important special cases for which specialized algorithms are available. Figure 2.1 depicts some of them.

The central problem of such mathematical programming problems is that of minimizing or maximizing a given function of a finite number of variables subject to a finite set of equality and/or inequality constraints. In the first part of this chapter we concentrate on nonlinear programming (NLP) which can be viewed as a part of mathematical optimization. Nonlinear programming deals with optimization problems, where the objective function or some of the constraints are nonlinear. This contrasts with: linear programming, which frames algorithms for the solution of optimization problems with linear objectives and constraints; quadratic programming, which frames algorithms for the solution of optimization problems with quadratic objective and linear constraints; and convex programming, which frames algorithms for the solution of optimization problems with convex objective and concave inequality constraints. During the course of this chapter we will see that nonlinear programming also makes use of quadratic programming solution techniques.

In the second part the nonlinear programming problem formulation is extended by a disturbance parameter. The study of the influence of this disturbance parameter on the optimal solution introduces the concept of sensitivity, which motivates some advanced algorithmic extensions.

The last part of this chapter extends the single-objective formulations to multi-objective formulations.

We begin our discussion by introducing some definitions regarding rates of convergence of iterative solvers. These definitions are quite helpful in giving us a metric...
to assess algorithms for solving nonlinear problems. The following results can be found in Ortega and Rheinboldt [62].

**Definition 2.1 (Q-linear Convergence)** A sequence \( \{y_i\} \subset \mathbb{R}^N \) converging to a fix point \( \{y^*\} \) is said to converge Q-linearly if a constant \( 0 < c_{ql} < 1 \) exists such that

\[
\| y_{i+1} - y^* \| \leq c_{ql} \| y_i - y^* \|
\]

holds for all indices \( i \) that are sufficiently large.

\[\triangle\]

**Definition 2.2 (Q-superlinear Convergence)** The sequence is said to converge Q-superlinearly if a sequence of positive factors \( c_{qs,i} \to 0 \) exists such that
\[ \| y_{i+1} - y^* \| \leq c_{qs,i} \| y_i - y^* \| \]

holds for all indices \( i \) that are sufficiently large.

\[ \triangle \]

**Definition 2.3** (*Q*-quadratic Convergence) Finally, the sequence is said to converge \( Q \)-quadratically if a constant \( 0 < c_{qq} < 1 \) exists such that

\[ \| y_{i+1} - y^* \| \leq c_{qq} \| y_i - y^* \|^2 \]

holds for all indices \( i \) that are sufficiently large.

\[ \triangle \]

These convergence rates are called \( Q \)-rates, because the convergence factor \( c_{qx} \) in the definitions above is a quotient. Similarly defined are the \( R \)-rates for which the convergence factor \( c_{rx} \) is a root. If in one of three Definitions 2.1, 2.2, and 2.3, the index \( i + 1 \) on the left-hand side is replaced by \( i + m \) the corresponding convergence rate is denoted as the \( m \)-step convergence rate.

The \( R \)-rates are defined as:

**Definition 2.4** (*R*-linear Convergence) A sequence \( \{y_i\} \subset \mathbb{R}^{N_y} \) converging to a fix point \( \{y^*\} \) is said to converge \( R \)-linearly if a constant \( 0 < c_{rl} < 1 \) exists such that

\[ \sqrt{\| y_i - y^* \|} \leq c_{rl} \]

holds for all indices \( i \) that are sufficiently large.

\[ \triangle \]

**Definition 2.5** (*R*-superlinear Convergence) The sequence is said to converge \( R \)-superlinearly if a sequence of positive factors \( c_{rs,i} \rightarrow 0 \) exists such that

\[ \sqrt[3]{\| y_i - y^* \|} \leq c_{rs,i} \]

holds for all indices \( i \) that are sufficiently large.

\[ \triangle \]

**Definition 2.6** (*R*-quadratic Convergence) The sequence is said to converge \( R \)-quadratically if a constant \( 0 < c_{rq} < 1 \) exists such that

\[ \sqrt[3/2]{\| y_i - y^* \|} \leq c_{rq} \]

holds for all indices \( i \) that are sufficiently large.

\[ \triangle \]
For linear and superlinear convergence the $R$-convergence rates are always smaller than the $Q$-convergence rates. Therefore, the $R$-linear convergence is generally slower than the $Q$-linear convergence; the same applies for superlinear convergence. For quadratic convergence this relation does not hold.

Most convergence analyses of optimization algorithms are concerned with $Q$-convergence.

### 2.2 Unconstrained Nonlinear Optimization

We begin our short introduction to numerical optimization with the fundamentals of unconstrained optimization. These are needed to understand the more complex algorithms for constrained optimization, which are later used to solve the optimal control problems arising from the problem formulations.

The simplest nonlinear programming problem is that of minimizing or maximizing a function $f : \mathbb{R}^{N_y} \rightarrow \mathbb{R}$. We restrict our analysis only to the minimum problem, as the problem $\min f(y)$ is equivalent to the problem $\max (-f(y))$.

**Definition 2.7 (Unconstrained Nonlinear Programming Problem)** An unconstrained nonlinear programming problem (also known as a free mathematical programming problem) is given by

\[
\min_{y \in \mathbb{R}^{N_y}} f(y) \quad (2.1)
\]

where $f : \mathbb{R}^{N_y} \rightarrow \mathbb{R}$ is a smooth and real-valued objective function of the vector $y \in \mathbb{R}^{N_y}$.

The vector

\[
y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_{N_y} \end{bmatrix}
\]

contains all the decision variables of the unconstrained NLP.

**Definition 2.8 (Extremals)** A point $y^* \in \mathbb{R}^{N_y}$ is called a local minimum of (2.1), if a local region $U_\epsilon$ with $\epsilon > 0$ exists such that

\[
f(y^*) \leq f(y) \quad (2.2)
\]

is satisfied $\forall y \in U_\epsilon(y^*)$. If instead of (2.2),
2.2 Unconstrained Nonlinear Optimization

\[ f(y^*) < f(y) \]  \hfill (2.3)

applies, then \( y^* \) is called a \textit{strict local minimum} of (2.1). For the case that (2.2) is satisfied \( \forall y \in \mathbb{R}^N_y, y^* \) is a \textit{global minimum} of (2.1). If (2.3) also holds for a global minimum, then (2.1) has an \textit{unique global minimum}.

\[ \triangle \]

In the remainder of this chapter it is assumed that \( f \) is twice continuously differentiable on \( \mathbb{R}^N_y \) with respect to \( y \), so that the Hessian for \( f(\cdot) \) is defined.

2.2.1 Necessary and Sufficient Conditions for Optimality

Every local minimum \( y^* \) of Problem (2.1) satisfies the first-order necessary condition, which is the vanishing of the gradient of \( f(y^*) \) (Fermat’s condition)

\[ \nabla f(y^*) = 0 \]  \hfill (2.4)

and the second-order necessary condition

\[ v^T \nabla^2 f(y^*) v \geq 0, \quad \forall v \in \mathbb{R}^N_y, v \neq 0, \]  \hfill (2.5)

which means that the Hessian \( \nabla^2 f(y^*) \) must be positive semi-definite. But these necessary conditions can also apply to local maxima or saddle points. So, a point that satisfies these conditions is only guaranteed to be a \textit{stationary or critical point} of Problem (2.1).

If in addition to the necessary conditions the second-order sufficient condition for the point \( y^* \) holds that \( f(y^*) \) has a positive definite Hessian (i.e., \( f(y^*) \) is locally strict convex)

\[ v^T \nabla^2 f(y^*) v > 0, \quad \forall v \in \mathbb{R}^N_y, v \neq 0, \]

then the point \( y^* \) is guaranteed to be a local minimum and is called \textit{strict local minimum}.

In summary, every local minimum satisfies the two necessary conditions, but must not necessarily satisfy the sufficient condition. Vice versa, every point that satisfies the necessary conditions and the sufficient condition is guaranteed to be a local minimum.

2.2.2 Newton–Raphson Method

To find a minimum of (2.1) the system of equations (2.4), which is generally nonlinear, must be solved. A commonly used method for this problem is the Newton–Raphson
Introduction to Nonlinear Programming

method. Treating the current point \( y \) as fixed and introducing a new variable \( d \) as deviation from \( y \), the idea now is to construct an iterative procedure by using a linear Taylor approximation of \( \nabla f(y + d) \) around the point \( y \)

\[
\nabla f(y + d) \approx \nabla f(y) + \nabla^2 f(y)d
\]

(2.6)

where \( d \) denotes the search direction,

\[
\nabla f(y) = \begin{bmatrix}
\frac{\partial f}{\partial y_1} \\
\vdots \\
\frac{\partial f}{\partial y_N}
\end{bmatrix}
\]

is the \( N_y \)-dimensional gradient vector with respect to \( y \), and

\[
\nabla^2 f(y) = \begin{bmatrix}
\frac{\partial^2 f}{\partial y_1^2} & \frac{\partial^2 f}{\partial y_1 \partial y_2} & \cdots & \frac{\partial^2 f}{\partial y_1 \partial y_N} \\
\frac{\partial^2 f}{\partial y_2 \partial y_1} & \frac{\partial^2 f}{\partial y_2^2} & \cdots & \frac{\partial^2 f}{\partial y_2 \partial y_N} \\
\vdots & \vdots & \ddots & \vdots \\
\frac{\partial^2 f}{\partial y_N \partial y_1} & \frac{\partial^2 f}{\partial y_N \partial y_2} & \cdots & \frac{\partial^2 f}{\partial y_N^2}
\end{bmatrix}
\]

is the symmetric \( N_y \times N_y \) Hessian matrix.

By simply setting (2.6) to zero, one obtains the Newton’s search direction with

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

(2.7)

Using the Newton’s search direction (2.7), we can generate a sequence \( \{y_k\}_{k\in\mathbb{N}} \) by applying the update rule

\[
y_{k+1} = y_k + d_k.
\]

(2.8)

For the sake of notational simplicity we introduce the following abbreviations \( \nabla f_k := \nabla f(y_k) \) and \( \nabla^2 f_k := \nabla^2 f(y_k) \).

The proof and conditions for local convergence of the method to a stationary point \( y^* \) were first stated in a general form by Kantorovich [46]. We use an adapted formulation of the Kantorovich’s theorem for the problem (2.1), similar to the statement in Ferreira and Svaiter [25].

**Theorem 2.1 (Kantorovich’s Theorem)** Let \( f(\cdot) \) be twice continuously differentiable on \( \text{int}(C) \) (the interior of \( C \)) with respect to \( y \) where \( C \subseteq \mathbb{R}^N \), \( y_0 \in \text{int}(C) \), \( L > 0 \), \( \omega > 0 \) and suppose that

1. \( \nabla^2 f_0 \) is nonsingular;
2. \( \| (\nabla^2 f_0)^{-1} [\nabla^2 f_1 - \nabla^2 f_2] \| \leq L \| y_1 - y_2 \|, \quad \forall y_1, y_2 \in C; \)
3. \( \| (\nabla^2 f_0)^{-1} \nabla f_0 \| \leq \omega; \) and
4. \( \omega L \leq \frac{1}{2} \)

holds. Define the radii
\[
\begin{align*}
    r_1 &= \frac{1 - \sqrt{1 - 2\omega L}}{L} \\
    r_2 &= \frac{1 + \sqrt{1 - 2\omega L}}{L}.
\end{align*}
\]

If there exists a ball \( B[y_0, r_1] \subset C \), then the sequence \( \{y_k\}_{k \in \mathbb{N}} \) generated by the Newton–Raphson method with the starting point \( y_0 \) is contained in \( B(y_0, r_1) \), converges to a unique zero of \( y^* \) (2.4) in \( B[y_0, r_1] \) and
\[
\| y^* - y_{k+1} \| \leq \frac{1}{2} \| y^* - y_k \|, \quad \forall k \in \mathbb{N}.
\]

If \( \omega L < \frac{1}{2} \) also holds, then the sequence \( \{y_k\}_{k \in \mathbb{N}} \) converges Q-quadratically with
\[
\| y^* - y_{k+1} \| \leq \frac{L}{2\sqrt{1 - 2\omega L}} \| y^* - y_k \|^2, \quad \forall k \in \mathbb{N}
\]

and \( y^* \) is the unique zero of (2.4) in \( B[y_0, \rho] \) for any \( \rho \) such that
\[
r_1 \leq \rho < r_2 \text{ and } B[y_0, \rho] \subset C.
\]

Proof Proofs of this theorem can be found in Kantorovich [46, 47], and Ortega [61].

Under the assumptions (1–3), the theorem of Kantorovich gives sufficient conditions for Q-linear convergence and, with the additional assumption \( \omega L < \frac{1}{2} \), for Q-quadratic convergence to a stationary point \( y^* \); However, it neither guarantees that the method converges to a local minimum nor that the method converges at all if the starting point is not in the neighborhood of a stationary point.

Even though the theorem of Kantorovitch does not explicitly state the radius \( r_c \), for which the sequence \( \{y_k\}_{k \in \mathbb{N}} \) with \( \| y_k - y^* \| \leq r_c \) is guaranteed to converge to \( y^* \), it does guarantee the existence of \( r_c > 0 \). \( r_c \) is denoted as the convergence radius of the Newton–Raphson method and \( B[y^*, r_c] \) as the sphere of convergence.

If the starting point \( y_0 \) lies inside a sphere of convergence and the Hessian \( \nabla^2 f_0 \) is positive definite, the theorem of Kantorovitch guarantees the convergence to a local minimum, because for a positive definite Hessian \( \nabla^2 f \) the inverse is also positive definite. Then, one can state that
\[
\nabla f(y)^T (\nabla^2 f(y))^{-1} \nabla f(y) > 0, \quad (2.9)
\]
34 2 Introduction to Nonlinear Programming

and it follows with Equation (2.7) that

$$d^T \nabla f(y) < 0$$

(2.10)

which is called a descent condition and this guarantees that every step of the Newton–Raphson method decreases the function value of (2.1).

In this case, the Hessian $\nabla^2 f_k$ keeps positive definite for all further iteration steps and therefore the sufficient condition (2.5) for a local minimum is satisfied at the point of convergence $y^*$.

If the starting point lies inside the sphere of convergence and the Hessian is not positive definite the Newton–Raphson method converge to a stationary point, which can be local minimum, a local maximum, or a saddle point. If the method does not converge to a local minimum, the Newton–Raphson iteration must be restarted with another appropriate starting point or a globalization strategy must be used to be able to find a local minimum.

### 2.2.3 Globalization of the Newton–Raphson Method

In the previous section, we have seen that if the starting point does not lie inside the sphere of convergence of a local minimum, one obtains no guarantee of convergence. It is therefore an obvious extension of the Newton–Raphson method to demand a minimum decrease of the function values for every iteration to establish global convergence to a stationary point and it is hoped to a local minimum

$$f(y_{k+1}) < f(y_k).$$

(2.11)

This condition guarantees a monotonously decreasing sequence of function values and aims to reach a sphere of convergence in a finite number of iterations, if the function has a bound on the minimum value.

There are two main types of algorithms that aim at finding a sequence of points that satisfy Condition (2.11) at every iteration: line-search methods and trust-region methods. We will focus on line-search methods only because these methods perform very well for the class of problems discussed in this book.

By using line-search methods we satisfy Condition (2.11) for every iteration by enforcing that the search direction $d_k$ is a descent direction (2.10) and adding a step-size $\alpha_k \in (0, 1]$ to the iteration rule (2.8) which is then called the line-search iteration rule

$$y_{k+1} = y_k + \alpha_k d_k.$$

Inside a sphere of convergence, the Newton–Raphson method guarantees convergence to a stationary point with the step-size $\alpha_k = 1$. Outside a sphere of convergence $\alpha_k$ can be chosen small enough to satisfy Condition (2.11) because $d_k$ is a descent
direction. This modification of the Newton–Raphson method is known as damped Newton–Raphson method.

The next subsections describe how to choose an adequate step-size and how to enforce the search direction $d_k$ to be a descent direction.

### 2.2.3.1 Step-Size Algorithms

An appropriate step-size should at least satisfy Condition (2.11) for the next iterate. Consequently, the descent condition is defined by

$$f(y_k + \alpha_k d_k) < f(y_k).$$  \hfill (2.12)

If only the descent condition is considered for calculating a step-size, it might be possible that the convergence becomes very slow due to small decreases in function values or that the algorithm does not converge at all.

This requires a modification of (2.12) with a factor $\epsilon \in [0, 1)$. We obtain then the condition

$$f(y_k + \alpha_k d_k) < f(y_k) + \epsilon \alpha_k \nabla f^T d_k.$$  \hfill (2.13)

This condition was proposed by Armijo [2] in 1966 to simplify the application of deepest descent algorithms and is up to now a major contribution in determining a proper step-size. Thus, Condition (2.13) is known as the Armijo condition.

If the factor $\epsilon$ is large enough, then the additional term enforces a reduction in $f(\cdot)$ proportional to both the step-size $\alpha_k$ and the directional derivative $g_k^T d_k$. Obviously, for $\epsilon = 0$ the Armijo condition is equal to the descent condition (2.12) and this procedure can still lead to very small step-sizes.

To avoid this drawback Wolfe (cf. [71] and [72]) introduced the additional condition

$$\nabla f(y_k + \alpha_k d_k)^T d_k \geq \rho \cdot \nabla f_k^T d_k$$  \hfill (2.14)

and restricted the parameters to $\epsilon \in [0, 0.5)$ and $\rho \in (\epsilon, 1)$, which limits the decrease of the step-size to an infimum. Conditions (2.13) and (2.14) together are called Wolfe conditions.

But a step-size, which satisfies these conditions, must still not be close to an optimal step-size. Therefore, the condition can be strengthened by the modification

$$|\nabla f(y_k + \alpha_k d_k)^T d_k| \leq \rho \cdot |\nabla f_k^T d_k|,$$

which is then called together with (2.13) strong Wolfe conditions.
It remains to clarify how a step-size, which satisfies either the Armijo condition, the Wolfe conditions, or the strong Wolfe conditions can be calculated. A step-size for the Armijo condition can be calculated in a fairly simple way because it is automatically satisfied if the step-size is small enough. For this task the basic backtracking algorithm can be used.

The basic backtracking algorithm described in Nocedal and Wright \[59\] determines the smallest \( j \in \mathbb{N}_{\geq 0} \) with \( \beta \in (0, 1) \) such that for \( \alpha_k = \beta^j \) Condition (2.12) is satisfied. The usual way is to search for the first index \( j \) that satisfies the descent condition (2.12) beginning with \( j = 0 \).

For the (strong) Wolfe conditions the backtracking algorithm needs not to terminate, because very small step-sizes are not acceptable. Instead, the line-search algorithm described in Moré and Thuente \[57\] can be used.

### 2.2.3.2 Descent Search Direction

If the Hessian is positive definite then the Newton search direction (2.7) is a descent direction (see (2.9) and (2.10)). In general, one can expect a positive definite Hessian only in the neighborhood of a local minimum. From a global scope it is unlikely that the Hessian \( \nabla^2 f_k \) is positive definite for all iterates \( y_k \). In order to enforce a descent search direction one can replace the Hessian \( \nabla^2 f_k \) with any other positive definite matrix \( B_k \) in Condition (2.10) and calculate the search direction thereby

\[
 d_k = -B_k^{-1}\nabla f_k. \tag{2.15}
\]

If one chooses the unity matrix \( B_k = I \), the search direction \( d_k \) is exactly the steepest descent search direction.

This makes the strategy clear, if the iterates of the minimization process have not reached a sphere of convergence with a positive definite Hessian, the search direction (2.15) is determined by applying a modified matrix \( B_k \) and an adequate step-size until a sphere of convergence with a positive definite Hessian is reached. Hereafter, the Newton search direction with a step-size of 1 is used to exploit the fast quadratic convergence from the Newton–Raphson method.

A common method to realize a blending between a positive definite matrix and the exact Hessian is to apply a scaled updating with

\[
 B_k = \kappa_k I + \nabla^2 f_k
\]

where \( \kappa_k \) must be chosen large enough to ensure positive definiteness of \( B_k \). With increasing \( k \) the weighting factor \( \kappa_k \) must tend to zero such that inside a sphere of convergence the exact Hessian can be used. To guarantee \( B_k \) to be positive definite \( \kappa_k \) must be chosen larger than the absolute value of the most negative Eigenvalue of \( \nabla^2 f_k \). Betts \[7\] assumed the choice of

\[
 \kappa_k = \tau_k \cdot (|\sigma_k| + 1)
\]
with the weighting factor $\tau_k \in [0, 1]$ and the \textit{Gershgorin bound} for the most negative Eigenvalue

$$\sigma_k = \min_{1 \leq i \leq N_y} \left\{ h_{ii} - \sum_{i \neq j} |h_{ij}| \right\}.$$  

Here, $h_{ij}$ denotes the element of the $i$-th row and $j$-th column of the Hessian $\nabla^2 f_k$.

The choice of the weighting factor $\tau_k$ essentially affects the performance of the minimization process; in the neighborhood of a local minimum it should be chosen as small as possible. Therefore, one can use the backtracking algorithm to find the smallest $\tau_k$ for which the matrix $B_k$ is still positive definite. An effective algorithm to check the positive definiteness of $B_k$ is presented in Chap. 9.

### 2.2.4 Quasi-Newton Method

For the methods for nonlinear unconstrained optimization previously introduced the calculation of the Hessian in every iteration is needed. Because this calculation is numerically an expensive task and in general not beneficial outside a sphere of convergence, one can try to approximate the Hessian or the inverse of the Hessian with an update rule, which only requires first-order derivatives of the objective function.

The approximation of the Hessian is denoted by $B_k$ and the approximation of the inverse Hessian is denoted by $H_k$ for every iteration number $k$.

Davidon first introduced the \textit{variable metric method} in 1966, which is known today as Quasi-Newton method [17]. Davidon used the \textit{symmetrical rank 1} (SR1) update

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

and the rank 2 update

$$H_{k+1} = H_k + \frac{\delta_k \delta_k^T}{\delta_k^T y_k} - \frac{H_k y_k y_k^T H_k}{y_k^T H_k y_k}$$ \quad(2.16)$$

to approximate the inverse of the Hessian. Herein, $\delta_k = y_{k+1} - y_k$ is the change made in $y$ and $y_k = \nabla f_{k+1} - \nabla f_k$ the change made in the gradient $\nabla f(y)$ at the $k$-th iteration. The advantage of the approximation of the inverse of the Hessian is that no linear system of equations need to be solved to calculate the search direction. Instead, a simple matrix–vector multiplication is used. A proof of convergence of this method is given by Fletcher and Powell [31]. The update process (2.16) is known as the DFP update formula (named by Davidon, Fletcher, and Powell).
The SR1 formula for the approximation of the Hessian is

\[ B_{k+1} = B_k + \frac{(y_k - B_k \delta_k) (y_k - B_k \delta_k)^T}{(y_k - B_k \delta_k)^T \delta_k} \]

and the DFP formula is

\[ B_{k+1} = \left( I - \frac{y_k \delta_k^T}{\delta_k \gamma_k} \right) \cdot B_k \cdot \left( I - \frac{\delta_k y_k^T}{\delta_k \gamma_k} \right) + \frac{y_k y_k^T}{\delta_k \gamma_k}. \]  

(2.17)

The characteristic feature of the SR1 and DFP formulas is the validity of the secant equations

\[ y_k^T H_{k+1} = \delta_k^T \]  

(2.18)

and

\[ B_{k+1} \delta_k = y_k. \]  

(2.19)

Thereby, the SR1 and DFP updates track the curvature of the objective function along the search path and store these informations in the updated matrices. In order to guarantee that the updated matrix keeps positive definite the curvature condition

\[ \delta_k^T y_k > 0 \]  

(2.20)

must be satisfied in every iteration. It can be seen that this condition preserves the positive definiteness, if Equation (2.19) is multiplied from the left-hand side with \( \delta_k^T \). Obviously, the same applies to the secant equation of the inverse Hessian if (2.18) is multiplied from the right-hand side with \( y_k^T \). The SR1 update has the drawback that the denominator can vanish. In this case the update cannot be applied.

A further update formula for the inverse of the Hessian which also fulfills the secant Eq. (2.18) is

\[ H_{k+1} = \left( I - \frac{\delta_k y_k^T}{\delta_k \gamma_k} \right) \cdot H_k \cdot \left( I - \frac{y_k \delta_k^T}{\delta_k \gamma_k} \right) + \frac{\delta_k \delta_k^T}{\delta_k \gamma_k}. \]

This update formula is known as the BFGS update rule (named by Broyden, Fletcher, Goldfarb, and Shanno). The inverse BFGS formula is

\[ B_{k+1} = B_k + \frac{y_k y_k^T}{\delta_k \gamma_k} - \frac{B_k \delta_k \delta_k^T B_k}{\delta_k \gamma_k}. \]  

(2.21)

On the basis of the two update formulas (2.17) and (2.21), a convex class of update rules, known as the Broyden class, can be stated with
2.2 Unconstrained Nonlinear Optimization

\[ B_{k+1} = (1 - \phi_k)B_{k+1}^{BFGS} + \phi_k B_{k+1}^{DFP}. \]

The explicit formula can be stated as

\[ B_{k+1} = B_k + \frac{\gamma_k y_k^T}{\delta_k^T y_k} - \frac{B_k \delta_k \delta_k^T B_k}{\delta_k^T B_k \delta_k} + \phi_k v_k v_k^T \]

where

\[ v_k = \left( \delta_k^T B_k \delta_k \right)^{1/2} \cdot \left( \frac{y_k}{\delta_k^T y_k} - \frac{B_k \delta_k}{\delta_k^T B_k \delta_k} \right). \]

Dennis and Moré [20] showed that a Quasi-Newton method converges Q-superlinearly if the condition

\[ \lim_{k \to \infty} \frac{\| [B_k - \nabla^2 f (y^*)] (y_{k+1} - y_k) \|}{\| y_{k+1} - y_k \|} = 0 \]

is satisfied. They also showed that this condition is always fulfilled, if any of the above-described Quasi-Newton update methods is used and the curvature condition (2.20) is fulfilled. If a damped Quasi-Newton method is used the step-size must become 1 in the neighborhood of the solution \( y^* \) to hold for a Q-superlinear convergence rate.

2.3 Constrained Nonlinear Optimization

This section is about minimizing at least two times continuously differentiable functions subject to constraints. The following definitions and theorems can be found in numerous textbooks including Nocedal and Wright [59], Fletcher [29], McCormick [54], Avriel [4], and Gill et al. [36].

**Definition 2.9 (Constrained Nonlinear Programming Problem)** The constrained nonlinear programming problem with equality and inequality constraints is given by

\[ \min_{y \in \mathbb{R}^{Ny}} f(y) \]

subject to \( g(y) \leq 0 \)

\[ h(y) = 0 \]

where \( f : \mathbb{R}^{Ny} \to \mathbb{R}, g : \mathbb{R}^{Ng} \to \mathbb{R}^{Ng}, \) and \( h : \mathbb{R}^{Nh} \to \mathbb{R}^{Nh} \) are all assumed to be twice continuously differentiable and real-valued. The constraints are defined as...
\[ g(y) := \begin{bmatrix} g_1(y) \\ \vdots \\ g_{N_g}(y) \end{bmatrix} \quad \text{and} \quad h(y) := \begin{bmatrix} h_1(y) \\ \vdots \\ h_{N_h}(y) \end{bmatrix}. \]

For a compact representation the optimization problem (2.22) may be rewritten to
\[ \min_{y \in S} f(y) \quad (2.23) \]
where \( S \) is the feasible set of points.

**Definition 2.10**  *(Feasible Set)* A point \( y \in \mathbb{R}^{N_y} \) is feasible, if all constraints of (2.22) are satisfied. The feasible set contains all feasible points
\[ S := \{ y \in \mathbb{R}^{N_y} \mid g_i(y) \leq 0, \ i = 1, \ldots, N_g \ \& \ h_j(y) = 0, \ j = 1, \ldots, N_h \}. \]

The feasible set \( S \subset \mathbb{R}^{N_y} \) is closed.

**Remark 2.1** The set \( S \) is also known by the terms: the *feasible region* or the *opportunity set*.

**Remark 2.2** We assume that the feasible set is given by the solutions of a finite number of equations. *Abstract constraints* or *variational inequalities* are not considered.

Before we derive the optimality conditions of Problem (2.23) some further definitions are required.

**Definition 2.11** *(Set of Active Indices)* An inequality constraint \( g_i(y) \) is said to be active if \( g_i(y) = 0 \). Then, the set of active indices \( \mathcal{I} \) of the inequality constraints is defined by
\[ \mathcal{I}(y) := \{ i = 1, \ldots, N_g \mid g_i(y) = 0 \}. \quad (2.24) \]

The number of active indices is defined by
\[ N_{\mathcal{I}} := \#\mathcal{I}(y) \]
where the symbol \# means the number of elements.

The set of active indices characterizes the relevance of the inequality constraints to the optimal solution \( y^* \). That means, an inactive inequality constraint has no influence on the optimal solution.
Definition 2.12 (Extremals) A point \( y^* \in \mathbb{R}^{N_y} \) is said to be a local minimum of (2.23), if \( y^* \in S \) is satisfied and \( \epsilon > 0 \) exists, such that
\[
f(y^*) \leq f(y)
\]
is satisfied \( \forall y \in S \cap U_\epsilon(y^*) \). If instead of (2.25),
\[
f(y^*) < f(y)
\]
applies, then \( y^* \) is called a strict local minimum of (2.23). For the case that (2.26) is satisfied \( \forall y \in S \), \( y^* \) is a global minimum of (2.23).

\[\square\]

2.3.1 Necessary and Sufficient Conditions for Optimality

One can find many necessary conditions with different strong assumptions in the literature, but only a few of them are directly applicable to practical optimization problems. The two fundamental results for (2.23), which have an impact on modern nonlinear programming algorithms, are the Fritz John and the Karush–Kuhn–Tucker conditions.

A prerequisite for stating the necessary conditions for the Problem (2.23) is the definition of the Lagrangian function, named after the mathematician Lagrange who treated equality-constrained optimization problems in the second half of the eighteenth century.

Definition 2.13 (Lagrangian Function (Nocedal and Wright [59])) The mapping \( \mathcal{L} : \mathbb{R}^{N_y} \times \mathbb{R} \times \mathbb{R}^{N_g} \times \mathbb{R}^{N_h} \to \mathbb{R} \) is defined by
\[
\mathcal{L}(y, l_0, \lambda, \mu) := l_0 f(y) + \sum_{i=1}^{N_g} \lambda_i g_i(y) + \sum_{i=1}^{N_h} \mu_i h_i(y)
\]
and is called the Lagrangian function for the problem definition (2.23). \( l_0 \) and the components of the vectors
\[
\lambda^T := [\lambda_1, \ldots, \lambda_{N_g}] \quad \text{and} \quad \mu^T := [\mu_1, \ldots, \mu_{N_h}]
\]
are called Lagrange multipliers.

\[\square\]

Theorem 2.2 (First-Order Necessary Conditions, Fritz John Conditions) Suppose that the functions \( f(\cdot), g(\cdot), \) and \( h(\cdot) \) are continuously differentiable with respect to \( y \). Then, the following Fritz John conditions are satisfied at a local minimum \((y^*, \lambda, \mu)\):
1. **sign condition:**

\[ l_0 \geq 0; \quad (2.29) \]

2. **optimality condition:**

\[
\nabla_y \mathcal{L}(y^*, l_0, \lambda, \mu) = l_0 \nabla f(y^*) + \nabla g^T(y^*)\lambda + \nabla h^T(y^*)\mu = 0. \quad (2.30)
\]

where the Jacobian matrices of \( g(\cdot) \) and \( h(\cdot) \) are of size \( \nabla g(y^*) \in \mathbb{R}^{N_g \times N_y} \) and \( \nabla h(y^*) \in \mathbb{R}^{N_h \times N_y} \);

3. **complementarity conditions:**

\[
\lambda^T g(y^*) = 0; \quad (2.31)
\]

and

4. **equality constraints:**

\[
h(y^*) = 0. \quad (2.32)
\]

**Proof** Proofs can be found in John [45], Mangasarian [51], and Bertsekas [6]. □

Each vector \((y^*, l_0, \lambda, \mu) \in \mathbb{R}^{N_y + 1 + N_g + N_h}\) which satisfies the conditions (2.29)–(2.32) is called a Fritz John stationary point of (2.23). It can be shown that for a local minimum \( y^* \) a point \((l_0, \lambda, \mu) \neq 0 \) with \( l_0 > 0 \) only exists if an additional condition, called a **regularity condition** or a **constraint qualification**, is satisfied. If such a constraint qualification holds, \( l_0 = 1 \) can be chosen without loss of generality.

In textbooks on optimization one can find numerous different constraint qualifications for nonconvex problems. In general, it is desirable to use the weakest constraint qualification in order to restrict the allowed constraints as little as possible. But it is a major drawback of the weaker constraint qualifications, e.g., the **quasiregularity condition**, that they cannot be evaluated in a simple manner. Therefore, stronger but much easier evaluable constraint qualifications like the Mangasarian–Fromowitz constraint qualification or the even stronger linear independence constraint qualification are often used. They are defined as follows.

**Definition 2.14** (Mangasarian–Fromowitz Constraint Qualification) Let a point \( y \in \mathcal{S} \) be feasible for (2.23) and the active set \( \mathcal{I}(y) \) be defined by (2.24). Then, the Mangasarian–Fromowitz constraint qualification (MFCQ) holds if the gradients of the equality constraints \( \nabla h_j(y) \), \( j = 1, \ldots, N_h \) are linearly independent (i.e., the constraints in this case are said to be **regular**) and there exists a vector \( b \in \mathbb{R}^{N_y} \) such that

\[
\nabla h_j^T(y)b = 0 \quad (j = 1, \ldots, N_h) \quad \text{and} \quad \nabla g_i^T(y)b < 0 \quad (\forall i \in \mathcal{I}(y)).
\]

△
**Definition 2.15** *(Linear Independence Constraint Qualification)* Let a point \( y \in S \) be feasible for (2.23) and the active set \( \mathcal{I}(y) \) be defined by (2.24). Then, the *linear independence constraint qualification* (LICQ) holds if the gradients \( \nabla g_i(y), \forall i \in \mathcal{I}(y) \) and \( \nabla h_j(y), j = 1, \ldots, N_h \) are linearly independent.

\[ \quad \triangle \]

**Remark 2.3** One can realize that these constraint qualifications are strongly restrictive. One can even show that these constraint qualifications are over restrictive just with a simple test. If an equality constraint \( h_i(y) = 0 \) is replaced by two equivalent inequality constraints, e.g., with \( g_i(y) \leq 0 \) and \( g_i(y) \geq 0 \), neither the MFCQ nor the LICQ holds.

**Theorem 2.3** *(Karush–Kuhn–Tucker Conditions)* Assume that the functions \( f(\cdot), g(\cdot), \) and, \( h(\cdot) \) are continuously differentiable with respect to \( y \) and a constraint qualification holds. Then the following conditions at a local minimum \((y^*, \lambda, \mu)\)

\[
\nabla_y L(y^*, 1, \lambda, \mu) = 0 \\
\nabla_\mu L(y^*, 1, \lambda, \mu) = h(y^*) = 0 \\
\nabla_i L(y^*, 1, \lambda, \mu) = g(y^*) \leq 0 \\
\lambda^T g(y^*) = 0 \\
\lambda \geq 0
\]

are satisfied and are known as *Karush–Kuhn–Tucker* (KKT) conditions.

\[ \quad \triangle \]

**Proof** Proofs of Theorem 2.3 can be found in Karush [48], Kuhn and Tucker [49], Mangasarian [51], Bertsekas [6], Fiacco and McCormick [27] and Fletcher [29]. \( \square \)

**Remark 2.4** The Fritz John conditions with \( l_0 = 1 \) are also called Karush–Kuhn–Tucker conditions.

Analogous to the Fritz John conditions, each vector \((y^*, \lambda, \mu) \in \mathbb{R}^{N_y + N_g + N_h}\) which satisfies the KKT conditions (2.33)–(2.37) is called a stationary KKT point of (2.23).

Condition (2.36) can be tightened with the strict complementarity condition which excludes the case \( \lambda_i = 0 \) and \( g_i = 0 \) for any \( i = 1, \ldots, N_g \). This stronger assumption is needed in the sensitivity analysis.

**Definition 2.16** *(Strict Complementarity Condition)* The *strict complementarity condition* is satisfied, if

\[
\lambda - g(y) > 0
\]

holds. \( \triangle \)
In order to ensure that any stationary KKT point \((\mathbf{y}^*, \lambda, \mu)\) is indeed an optimal solution of problem definition (2.23), second-order sufficient conditions (SOSC) are needed. SOSC also play an important role in the sensitivity analysis discussed in Sect. 2.4.

For the formulation of the SOSC we need the definition of the critical cone.

**Definition 2.17 (Critical Cone)** Suppose \(\mathbf{y}\) is a feasible point, then the critical cone is defined by

\[
T_{\mathbf{y}} := \{ \mathbf{v} \in \mathbb{R}^{N_y} \mid \nabla g_i^T(\mathbf{y})\mathbf{v} \leq 0, \quad i \in \mathcal{I}(\mathbf{y}), \quad \lambda_i = 0, \\
\nabla g_i^T(\mathbf{y})\mathbf{v} = 0, \quad i \in \mathcal{I}(\mathbf{y}), \quad \lambda_i > 0 \\
\n\nabla h_j^T(\mathbf{y})\mathbf{v} = 0, \quad j \in \{1, \ldots, N_h\} \},
\]

\(\Box\)

**Theorem 2.4 (Second-Order Sufficient Conditions)** Let \(\mathbf{y}^* \in S\) be a feasible point for the problem definition (2.23) and let the functions \(f(\cdot), g(\cdot), \text{and } h(\cdot)\) be twice continuously differentiable with respect to \(\mathbf{y}\). Assume that Lagrange vectors \(\lambda\) and \(\mu\) exist, Theorem 2.3 holds and that the Hessian of the Lagrangian is positive definite on the critical cone

\[
\mathbf{v}^T \nabla^2 \mathcal{L}(\mathbf{y}^*, \lambda, \mu)\mathbf{v} > 0, \quad \forall \mathbf{v} \in T_{\mathbf{y}}^*, \mathbf{v} \neq \mathbf{0}.
\]

(2.39)

Then, \(\mathbf{y}^*\) satisfies the SOSC and is a strict local minimum of (2.23).

\(\Box\)

**Proof** The proof can be found in Karush [48], Fletcher [29], and Bertsekas [6]. \(\Box\)

### 2.3.2 Projected Hessian

Because of the critical cone the evaluation of Theorem 2.4 is quite difficult in practice. Therefore, we project the Hessian of the Lagrangian to the kernel of the constraints to obtain numerically evaluable conditions.

Suppose, we found a triple \((\mathbf{y}, \lambda, \mu)\) of (2.23) which is feasible and the strict complementarity condition of the Lagrange multipliers and a constraint qualification holds, i.e., the tangent cone is a vector space. Then, with the Jacobian matrix of all active constraints

\[
\mathbf{A}_\mathbf{y} := (\nabla g_i(\mathbf{y}) \quad \nabla h_j(\mathbf{y}))^T,
\]

where the indices are given by \(i \in \mathcal{I}(\mathbf{y})\) and \(j = 1, \ldots, N_h\), we can restate the critical cone as

\[
T_{\mathbf{y}} = \{ \mathbf{v} \in \mathbb{R}^{N_y} \mid \mathbf{A}_\mathbf{y}\mathbf{v} = \mathbf{0} \} = \ker(\mathbf{A}_\mathbf{y}).
\]
The set \( \ker(A_y) \) denotes the *kernel of the matrix* \( A_y \). It should be noted that the dimension of the matrix depends on the number of active constraints \( A_y \in \mathbb{R}^{N_y \times N} \) with \( N_y := N_L + N_h \).

Using the kernel \( \ker(A_y) \), Condition (2.39) is equivalent to

\[
v^T \nabla^2 y L(y^*, \lambda, \mu) v > 0 \quad \forall v \in \ker(A_y) \setminus \{0\}.
\]

Introducing the matrix \( P \in \mathbb{R}^{N_y \times (N_y - N_s)} \) with full column rank whose columns span the kernel \( \ker(A_y) \) such that

\[
A_y P = 0_{N_y \times (N_y - N_s)},
\]

any vector \( v \in \ker(A_y) \) can be uniquely reexpressed as

\[
v = Pw
\]

for some vector \( w \in \mathbb{R}^{N_y - N_s} \). Then, Condition (2.39) may be restated as

\[
w^T P^T \nabla^2 y L(y^*, \lambda, \mu) Pw > 0, \quad \forall w \in \mathbb{R}^{N_y - N_s}, \ w \neq 0
\]

under the assumption that the strict complementarity condition (2.38) holds and \( P \) forms an orthogonal basis for \( \ker(A_y) \).

The matrix

\[
B_{\text{red}} := P^T \nabla^2 y L(y^*, \lambda, \mu) P \in \mathbb{R}^{(N_y - N_s) \times (N_y - N_s)}
\]

is called the *projected or reduced Hessian*. From (2.40) it can be concluded that the positive definiteness of the reduced Hessian is equivalent to the positive definiteness of the full Hessian \( \nabla^2 y L(y^*, \lambda, \mu) \) on the critical cone. Thus, the check for SOSC is reduced by showing that the projected Hessian is positive definite. The matrix \( P \) can be obtained by a QR-factorization of \( A_y^T \) with

\[
A_y^T = Q R = (Q_1 \ Q_2) \begin{pmatrix} R_1 \\ 0_{(N_y - N_s) \times N_s} \end{pmatrix}
\]

where \( Q \in \mathbb{R}^{N_y \times N_y} \) is an orthogonal matrix and \( R_1 \in \mathbb{R}^{N_s \times N_s} \) is a regular upper triangular matrix. Then \( P := Q_2 \) forms an orthogonal basis for \( \ker(A_y) \), which can be shown if Eq. (2.42) is multiplied from the left-hand side by \( Q^T \)

\[
\begin{pmatrix} Q_1^T \\ Q_2^T \end{pmatrix} A_y(y) = \begin{pmatrix} R_1 \\ 0_{(N_y - N_s) \times N_s} \end{pmatrix} \Rightarrow A_y Q_2 = 0_{N_y \times (N_y - N_s)}. \]
2.3.3 *Sequential Quadratic Programming*

A group of very popular methods to solve the constrained minimization problem (2.23) are *sequential quadratic programming* (SQP) methods. SQP algorithms are iterative and in every iteration they intend to calculate a suitable search direction. Han [43] and Powell [65] showed that such methods converge to a solution (KKT point) of the NLP problem using local curvature information by approximating the Lagrangian as a quadratic function and the nonlinear constraints as linearized constraints. They also showed that the local convergence rate of these methods is Q-superlinear, if an update method from the Broyden class (i.e., BFGS or DFP) is used to obtain a convex approximation of the Lagrangian. A local minimizer is found by solving a sequence of these convex quadratic subproblems.

There are two common variants of the SQP method, *line-search* and *filter SQP* methods. Line-search SQP methods extend the Quasi-Newton methods of unconstrained optimization with line-search globalization strategy for constrained problems. The step-length is calculated for a suitable merit function, which accounts for the object function and the constraints. Filter SQP methods extend the Quasi-Newton methods of unconstrained optimization with trust-region globalization strategy to constrained problems.

As for the unconstrained case we will concentrate only on line-search methods for constrained optimization, because the numerical results for both types of SQP algorithms are nearly identical for the problem set considered in this book. Furthermore, only the Quasi-Newton based SQP methods are discussed.

The construction of line-search SQP methods is analogous to Newton–Raphson methods. The gradient of the Lagrangian function (2.27) is approximated by a first-order Taylor expansion, which is supposed to vanish in a local minimum according to the necessary conditions. We obtain the first-order Taylor expansion of the Lagrangian function as

\[
\nabla_y \mathcal{L}(y + d, \lambda, \mu) \approx \nabla f(y) + \nabla g^T(y)\lambda + \nabla h^T(y)\mu + B(y, \lambda, \mu)d = 0. \tag{2.43}
\]

The matrix \(B(y, \lambda, \mu)\) accounts for second-order derivatives of the object function and the constraints and is updated in every iteration with a suitable Quasi-Newton update strategy.

The constraints \(g(\cdot)\) and \(h(\cdot)\) are also approximated with a first-order Taylor expansion in every iteration step by

\[
g(y + d) \approx g(y) + \nabla g(y)d \leq 0 \tag{2.44}
\]

and

\[
h(y + d) \approx h(y) + \nabla h(y)d = 0. \tag{2.45}
\]
The search direction can be calculated by solving (2.43)–(2.45). If the active set of the inequality constraints is known, only a system of linear equations must be solved to obtain the search direction. In general, however, the active set is not known a priori.

Notice that the solution of the system of Eqs. (2.43) and (2.45) and the inequalities (2.44) is equivalent to the minimization of a convex quadratic subproblem (QSP) with linear constraints.

**Definition 2.18** (Quadratic Subproblem) A QSP with linear constraints is defined by

\[
\min_{\mathbf{d} \in \mathbb{R}^{N_y}} \quad f(y) + \nabla f^T(y)d + \frac{1}{2}d^TBd
\]

subject to

\[
\begin{align*}
g(y) + \nabla g^T(y)d & \leq 0, \\
h(y) + \nabla h^T(y)d & = 0.
\end{align*}
\]

We regard the solution to the quadratic subproblem (2.46) as search direction \(\mathbf{d}_k\) that will be a descent direction for some merit function. Analogous to the globalized Quasi-Newton method, we can generate a sequence \(\{\mathbf{y}_k\}_{k \in \mathbb{N}}\) by applying the update rule

\[
\mathbf{y}_{k+1} = \mathbf{y}_k + \alpha_k \mathbf{d}_k
\]

where \(\alpha_k\) is a positive step-length and \(\mathbf{d}_k\) is the feasible search direction calculated for the \(k\)-th iteration of the SQP algorithm by solving the quadratic subproblem (2.46).

In the following three subsections, some methods are presented for solving the quadratic subproblem to obtain the search direction, to calculate the step-length, and to calculate appropriate Quasi-Newton updates for the Hessian approximation.

### 2.3.3.1 Solution of the Quadratic Subproblem

For the solution of the QSP (2.46) active set (AS) and interior-point (IP) methods are particularly suitable. The theory of solving convex quadratic problems is quite complex and would take a lot of space in this book. Therefore, only the interior-point algorithm in its basic form will be introduced. Interested readers may consult the textbooks of Nocedal and Wright [59], Fletcher [29], and Wright [73] to get more information about this type of problem.

Before reviewing the IP algorithm we start with a QSP, which is only equality constrained, and rename the variables as \(\omega := \mathbf{y}_k, \mathbf{x} := \mathbf{d}_k, \mathbf{y} := \mu_k, \text{ and } \mathbf{z} := \lambda_k\) for a more compact problem formulation.

**Definition 2.19** (Convex Quadratic Subproblem with Equality Constraints) The convex QSP with equality constraints can be compactly stated as
\[
\min_{x \in \mathbb{R}^{N_y}} \quad f + x^T c + \frac{1}{2} x^T G x
\]
subject to \quad Ax = b \tag{2.47}

where \( G := B_0 \) is the symmetric \( N_y \times N_y \) Hessian matrix, \( A := \nabla h^T(\omega) \) is the Jacobian of the equality constraints, \( c := \nabla f(\omega) \) is the gradient of the objective function, \( b := h(\omega) \) are the equality constraints, and \( f := f(\omega) \) is the objective function value—all at the \( k \)-th iteration. Notably, the term to be minimized is quadratic and the constraints are linear.

\[\triangle\]

From (2.27) we know that the Lagrangian for the quadratic problem (2.47) is given by

\[
\mathcal{L}(x, y) = f + x^T c + \frac{1}{2} x^T G x - y^T (A x - b).
\]

The first-order KKT conditions (see Theorem 2.3) of the equality-constrained problem (2.47) can be written as

\[
F(x, y) = \begin{bmatrix}
\nabla_x \mathcal{L}(x, y) \\
\nabla_y \mathcal{L}(x, y)
\end{bmatrix} = \begin{bmatrix}
c + G x - A^T y \\
-A x + b
\end{bmatrix} = 0_{(N_y+N_h) \times 1}. \tag{2.48}
\]

The Jacobian of (2.48) with respect to \( x \) and \( y \) is given by

\[
\frac{\partial F}{\partial (x, y)}(x, y) = \begin{pmatrix}
\nabla^2_x \mathcal{L}(x, y) & \nabla_{xy} \mathcal{L}(x, y) \\
\nabla_{yx} \mathcal{L}(x, y) & \nabla^2_y \mathcal{L}(x, y)
\end{pmatrix} = \begin{pmatrix}
G & -A^T \\
-A & 0_{N_h \times N_h}
\end{pmatrix} \tag{2.49}
\]

and is known as the so-called Karush–Kuhn–Tucker-matrix (KKT matrix for short).

The solution of (2.48) can be calculated by

\[
\begin{bmatrix}
x \\
y
\end{bmatrix} = - \begin{pmatrix}
G & -A^T \\
-A & 0_{N_h \times N_h}
\end{pmatrix}^{-1} \begin{bmatrix}
c \\
b
\end{bmatrix}, \tag{2.50}
\]

which is equivalent to the calculation of Newton’s search direction.

\textbf{Remark 2.5} Note that the sign of equality constraints together with the Lagrange parameters \( y^T (A x - b) \) coupled to the Lagrangian can be chosen arbitrarily.

Having in mind the quadratic problem formulation (2.46) and using the new variables one obtains a general convex quadratic problem formulation.

\textbf{Definition 2.20} (Convex Quadratic Subproblem) In addition to Definition 2.19, the general convex QSP can be stated as
2.3 Constrained Nonlinear Optimization

\[
\begin{align*}
\min_{x \in \mathbb{R}^n} & \quad f + x^T c + \frac{1}{2}x^T Gx \\
\text{subject to} & \quad Cx \geq d, \\
& \quad Ax = b
\end{align*}
\]

where \( C := -\nabla g^T(\omega) \) is the negative Jacobian of the inequality constraints and \( d := g(\omega) \) are the inequality constraints.

Interior-Point Method

We apply Mehrotra’s predictor-corrector IP method described in [55] to problem definition 2.20. We first introduce slack variables \( s \) to convert the inequality constraints to equality constraints with additional box constraints for the slack variables. Then the first-order necessary conditions for this problem formulation can be stated as

\[
\begin{align*}
Gx - A^T y - C^T z &= -c, \\
Ax &= b, \\
Cx - s &= d, \\
z &\geq 0, \quad s \geq 0, \quad z^T s = 0. 
\end{align*}
\]

(2.51)

A feasible starting point for this method can be directly denoted, because every point \((x_0, y_0, z_0, s_0)\) with \((z_0, s_0) > 0\) is feasible. The iterates of the interior-point method are always feasible, because \(z_j > 0\) and \(s_j > 0\) will be guaranteed due to a suitable step-size calculation. In contrast to feasibility, the initial point will usually not satisfy the complementary condition (2.51). Instead of imposing this condition at every iteration, the algorithm aims at decreasing an appropriately chosen residual of this condition. Therefore, the complementarity measure given by

\[
\mu = \frac{z^T s}{N_g}
\]

(2.52)

plays an important role in interior-point methods. Consequently, the violation of condition (2.52) will be forced by the algorithm to become smaller with every iteration.

Given a starting point \((x_0, y_0, z_0, s_0)\) with \((z_0, s_0) > 0\) and a parameter \(\tau \in [2, 4]\), the following steps for \(j = 1, 2, \ldots\) will be repeated until the convergence check is verified and the algorithm ends with an optimal solution.
Algorithm 2.1 QP Interior-point method (cf. Gertz and Wright [33])

1: \( j \leftarrow 1 \)
2: \( \mu_j \leftarrow \frac{z_j^T s_j}{N_g} \)
3: if first-order necessary conditions are satisfied up to a given tolerance then
4:    stop
5: end if
6: Solve for \( (\Delta x_j^{aff}, \Delta y_j^{aff}, \Delta z_j^{aff}, \Delta s_j^{aff}) \):

\[
\begin{pmatrix}
G - A^T - C^T & 0 \\
A & 0 & 0 & 0 \\
C & 0 & 0 & -I \\
0 & 0 & S_j & Z_j
\end{pmatrix}
\begin{bmatrix}
\Delta x_j^{aff} \\
\Delta y_j^{aff} \\
\Delta z_j^{aff} \\
\Delta s_j^{aff}
\end{bmatrix}
= -
\begin{bmatrix}
r_{Gj} \\
r_{Aj} \\
r_{Cj} \\
Z_j S_j e
\end{bmatrix},
\]

where \( S_j = \text{diag}(s_{j1}, s_{j2}, \ldots, s_{jN_g}) \), \( Z_j = \text{diag}(z_{j1}, z_{j2}, \ldots, z_{jN_g}) \),
\( r_{Gj} = Gx_j - A^Ty_j - C^T z_j + c \), \( r_{Aj} = Ax_j - b \), and \( r_{Cj} = Cx_j - s_j - d \)
7: Compute \( \alpha_j^{aff} = \arg \max_{\alpha \in (0,1)} \{ (z_j, s_j) + \alpha (\Delta z_j^{aff}, \Delta s_j^{aff}) \geq 0 \} \)
8: \( \mu_j^{aff} \leftarrow \frac{(z_j + \alpha_j^{aff} \Delta z_j^{aff})^T (s_j + \alpha_j^{aff} \Delta s_j^{aff})}{N_g} \)
9: \( \sigma_j \leftarrow \left( \frac{\mu_j^{aff}}{\mu_j} \right)^\tau \)
10: Solve for \( (\Delta x_j, \Delta y_j, \Delta z_j, \Delta s_j) \):

\[
\begin{pmatrix}
G - A^T - C^T & 0 \\
A & 0 & 0 & 0 \\
C & 0 & 0 & -I \\
0 & 0 & S_j & Z_j
\end{pmatrix}
\begin{bmatrix}
\Delta x_j \\
\Delta y_j \\
\Delta z_j \\
\Delta s_j
\end{bmatrix}
= -
\begin{bmatrix}
r_{Gj} \\
r_{Aj} \\
r_{Cj} \\
Z_j S_j e - \sigma_j \mu_j e + \Delta z_j^{aff} \Delta s_j^{aff} e
\end{bmatrix},
\]

where \( \Delta Z_j^{aff} \) and \( \Delta S_j^{aff} \) are defined in the same way as \( Z_j \) and \( S_j \) from step 6
11: Compute \( \alpha_j^{max} = \arg \max_{\alpha \in (0,1)} \{ (z_j, s_j) + \alpha (\Delta z_j, \Delta s_j) \geq 0 \} \)
12: Choose \( \alpha_j \in (0, \alpha_j^{max}) \) according to Mehrotra’s heuristic
13: Update \( x_{j+1} \leftarrow x_j + \alpha_j \Delta x_j, y_{j+1} \leftarrow y_j + \alpha_j \Delta y_j, z_{j+1} \leftarrow z_j + \alpha_j \Delta z_j, s_{j+1} \leftarrow s_j + \alpha_j \Delta s_j, \) and \( j \leftarrow j + 1 \)
14: Return to step 2.

For details on the implementation of convergence criteria, Mehrotra’s heuristic for step-size determination, simplifications of the linear system of equations, and higher order corrections the interested reader may refer to Mehrotra [55], Gondzio [40], and Gertz and Wright [33].
2.3.3.2 Quasi-Newton Update for Constrained Problems

The Quasi-Newton update formulas from Sect. 2.2.4 can also be used for the SQP method, if one replaces the definition of $\gamma_k$ with

$$\gamma_k = \nabla y L(y_{k+1}, \lambda_{k+1}, \mu_{k+1}) - \nabla y L(y_k, \lambda_k, \mu_k).$$

The proof of the local convergence of this Quasi-Newton update for constrained optimization can be transferred to the unconstrained case, if the projected Hessian (2.41) is used. Powell showed in [65] that

$$\lim_{k \to \infty} \frac{\| P_k^T \cdot [B_k - \nabla^2 L(y^*, \lambda, \mu)] \cdot P_k \cdot (y_{k+1} - y_k) \|}{\| y_{k+1} - y_k \|} = 0$$

implies

$$\frac{y_{k+1} - y^*}{y_{k-1} - y^*} \to 0 \quad (2.53)$$

where $P_k$ is the same projection matrix as in Eqs. (2.41) and (2.53) implies a two-step superlinear convergence rate. It should be reminded that the strict complementary condition must hold for the existence of the projected Hessian and is therefore required for this result about the convergence rate.

A practical drawback of the Quasi-Newton update for constrained optimization is the fact that the curvature condition (2.20) tends to be violated more frequently and therefore the Quasi-Newton update matrix $B_{k+1}$ is no longer positive definite. To avoid this drawback Powell suggests the use of a modified BFGS update with

$$B_{k+1} = B_k + \frac{\eta_k \eta_k^T}{\delta_k^T \eta_k} - \frac{B_k \delta_k \delta_k^T B_k}{\delta_k^T B_k \delta_k} \quad (2.54)$$

where $y_k$ is replaced with $\eta_k$, which is defined as

$$\eta_k = \theta_k y_k + (1 - \theta_k) B_k \delta_k. \quad (2.55)$$

The factor $\theta_k$ is given by

$$\theta_k = \begin{cases} 1, & \delta_k^T y_k \geq 0.2 \delta_k^T B_k \delta_k \\ 0.8 \frac{\delta_k^T B_k \delta_k}{\delta_k^T B_k \delta_k - \delta_k^T y_k}, & \delta_k^T y_k < 0.2 \delta_k^T B_k \delta_k \end{cases}$$

With this modification, the stronger condition

$$\delta_k^T \eta_k > 0.2 \delta_k^T B_k \delta_k$$
is satisfied in every iteration of the SQP method and the BFGS matrix remains positive definite. The update formula (2.54) has proven to perform well in many applications (often even with a Q-superlinear rate of convergence) and Powell proved that the convergence rate is at least R-superlinear if some additional assumptions for the modified BFGS updates hold (see Powell [65]). However, there is, as far as we know, no general proof of local convergence for this modified update formula.

2.3.3.3 Step-Size Calculation with Merit Functions

After the calculation of the search direction $d_k$ a suitable step-size $\alpha_k$ must be determined. To accomplish this task a merit function can be defined, which weights the decrease in the object function and the violation of the constraints. A prerequisite is that the search direction $d_k$ obtained from the QP is also a descent direction of the merit function.

The necessity of a merit function can be best imagined for a given pair of points. If the two points are feasible one can determine which is best by comparing the objective function values. However, this becomes problematic if the points are allowed to be infeasible. Then, it is not apparent which of the two points offers the best approximation to the solution.

In their line-search SQP algorithms, Han and Powell introduced the $l_1$ merit function

$$\Psi(y, \varrho, \nu) = f(y) + \varrho^T \cdot \max(0, g(y)) + \nu^T \cdot |h(y)|,$$

where the maximum must be evaluated element-wise, and Powell suggests updating the penalty parameters $\varrho$ and $\nu$ according to the update rules

$$\varrho_k = \begin{cases} |\lambda_k|, & k = 0 \\ \max(\{|\lambda_k|, \frac{1}{2} (\varrho_{k-1} + |\lambda_k|)\}), & k > 0 \end{cases}$$

(2.56)

and

$$\nu_k = |\mu_k|. \tag{2.57}$$

That means, the merit function assigns a positive penalty for increasing constraint violations. For the penalty parameters (2.56) and (2.57) it can be shown that the search direction $d_k$ is a descent direction of the $l_1$ merit function. Because of the non-differentiability of the $l_1$ merit function Powell recommends using the backtracking algorithm with a modification of the Armijo condition (2.13) as step-size procedure

$$f(y_k + \alpha_k d_k) < f(y_k) + \epsilon \alpha_k (f(y_k + d_k) - f(y_k))$$

with $\epsilon = 0.1$. 

2.3 Constrained Nonlinear Optimization

A further drawback of the $l_1$ merit function is that the iterations can cycle, such that the SQP algorithm does not converge at all or converges only very slowly. This effect is known as the Maratos effect [52].

To avoid the Maratos effect some modifications to the algorithm can be applied. One possible modification to overcome the problem is the Watchdog technique proposed by Chamberlain, Powell, Lemarechal, and Pedersen [14]. Another approach is a second-order correction of the QSP solution as described in Mayne and Polak [53], Fletcher [28], and Coope [16].

The aim of a second-order correction is to replace the update $y_{k+1} = y_k + \alpha_k d_k$ by a corrected update

$$y_{k+1} = y_k + \alpha_k d_k + \alpha_k^2 \tilde{d}_k,$$

where the correction step $\tilde{d}_k$ will be obtained by a modified equality or modified inequality-constrained quadratic subproblem.

**Definition 2.21** *(Equality-Constrained Quadratic Subproblem for the Second-Order Correction (cf. Coope [16]))* The modified equality-constrained quadratic subproblem for obtaining second-order corrections can be formulated as

$$\min_{\tilde{d} \in \mathbb{R}^N_y} \tilde{d}^T (\nabla f(y) + d) + \frac{1}{2} \tilde{d}^T \tilde{d}$$

subject to

$g_{i \in I_y}(y + d) + \nabla g_{i \in I_y}(y) \tilde{d} = 0,$

$h(y + d) + \nabla h^T(y) \tilde{d} = 0.$

$\triangle$

**Definition 2.22** *(Inequality-Constrained Quadratic Subproblem for Fletcher’s Second-Order Correction (cf. Fletcher [28]))* The modified inequality-constrained quadratic subproblem for obtaining second-order corrections can be formulated as

$$\min_{\tilde{d} \in \mathbb{R}^N_y} \tilde{d}^T (\nabla f(y) + Bd) + \frac{1}{2} \tilde{d}^T B \tilde{d}$$

subject to

$g(y + d) + \nabla g^T(y) \tilde{d} \leq 0,$

$h(y + d) + \nabla h^T(y) \tilde{d} = 0$

$\triangle$

The solutions $\tilde{d}$ of these quadratic subproblems account for a second-order approximation of the constraints curvature and can reduce the $l_1$ merit function and the Lagrangian, if $\| \tilde{d} \|$ is sufficiently small.

Motivated by the conditions for applying a correction step described by Mayne [53], Coope [16], and Fletcher [28] we calculate a correction step only if

$$\mathcal{L}(y + d, \lambda_k, \mu_k) < \mathcal{L}(y, \lambda_k, \mu_k)$$
applies and finally use it only if
\[ \mathcal{L}(y + d + \tilde{d}, \lambda_k, \mu_k) < \mathcal{L}(y + d, \lambda_k, \mu_k) \]
applies.

2.4 Sensitivity Analysis

In the last section we studied the local optimal solution \( y^* \) of problem definition 2.9 with fixed boundary parameters. In the case that some of these parameters change, the solution \( y^* \) can no longer be regarded as locally optimal. But we are highly interested in solutions for different parameters to be able to find the best parameter set for a prescribed NLP. The probably most simple and accurate method to deal with such parameter variation problems is the reoptimization of the problem for several parameter combinations selected by a brute force approach (Beltracchi and Gabriele [5] and Bückens [10]). The brute force approach can handle arbitrary large disturbances in the parameters, but the interpretation of the impact of varying parameters on the results can be difficult due to the overwhelming amount of datasets generated, which makes this method not only computationally demanding but also provides no clear systematics in dealing with such problems.

To get a more systematic approach for such problems, we ask in a first step only for solutions of a NLP with little disturbances in the parameter set, which can be expressed as a first-order correction of the nominal solution. Such problems are known as sensitivity problems and widely applied in the optimization theory. We consider this problem class in a systematic fashion using sensitivity methods based on the Kuhn–Tucker conditions (2.33)–(2.37). These methods avoid the computational expense of reoptimization but are only local approximations of the disturbed solutions. In so doing, we introduce an parameter vector \( p = [p_1, p_2, \ldots, p_{N_p}]^T \in \mathcal{P} \) where \( \mathcal{P} \subset \mathbb{R}^{N_p} \) is an open set of finite-dimensional parameters and define a modified problem definition of the standard form of the nonlinear programming problem (2.22).

Definition 2.23 (Parametric Nonlinear Programming Problem (cf. Bückens and Maurer [12])) The parametric nonlinear programming problem (NLP(\( p \))) with equality and inequality constraints is given

\[
\begin{align*}
\min_{y \in \mathbb{R}^{N_y}} & \quad f(y, p) \\
\text{subject to} & \quad g(y, p) \leq 0 \\
& \quad h(y, p) = 0
\end{align*}
\] (2.58)
where \( f : \mathbb{R}^{N_y} \times \mathcal{P} \rightarrow \mathbb{R}, \ g : \mathbb{R}^{N_y} \times \mathcal{P} \rightarrow \mathbb{R}^{N_g}, \) and \( h : \mathbb{R}^{N_y} \times \mathcal{P} \rightarrow \mathbb{R}^{N_h} \) are real-valued and assumed to be twice continuously differentiable w.r.t. \( y \). The constraints are defined as
\[
g(y, p) := \begin{bmatrix} g_1(y, p) \\ \vdots \\ g_{N_y}(y, p) \end{bmatrix} \quad \text{and} \quad h(y, p) := \begin{bmatrix} h_1(y, p) \\ \vdots \\ h_{N_h}(y, p) \end{bmatrix}.
\]

\[\triangle\]

**Remark 2.6** For a reference parameter vector \( p = p_0 \) the problem formulation (2.58) is called an undisturbed or nominal nonlinear programming problem NLP\((p_0)\).

The aim of the sensitivity analysis is the calculation of sensitivities of the parametric problem definition 2.23. The sensitivities are total differentials of the optimal solution dependent on the parameter vector \( p \). Based on the second-order sufficient conditions the main result of the sensitivity analysis is the sensitivity theorem which gives conditions for the existence and the properties of an optimal solution under presence of disturbing parameters. A perturbed optimization problem is given if the parameter vector \( p \) from (2.58) differs from the reference values \( p_0 \) of the nominal optimization problem (2.22).

The cornerstone of the sensitivity analysis is laid by Fiacco [26]. We follow the definitions of this work and present the main results which are used in Chap. 13.

The following assumptions are made: the functions \( f(\cdot), g(\cdot), \) and \( h(\cdot) \) can be either linear or nonlinear functions of the NLP-variables \( y \) and the parameters \( p \) are held fixed during the optimization. Then, based on the problem formulation (2.58), the Lagrangian function is modified to \( \mathcal{L} : \mathbb{R}^{N_y} \times \mathbb{R}^{N_g} \times \mathbb{R}^{N_h} \times \mathcal{P} \rightarrow \mathbb{R} \)
\[
\mathcal{L}(y, \lambda, \mu, p) := f(y, p) + \sum_{i=1}^{N_g} \lambda_i g_i(y, p) + \sum_{i=1}^{N_h} \mu_i h_i(y, p)
\]
where the Lagrange multipliers are defined as (2.28).

**Remark 2.7** The necessary and sufficient conditions derived for the standard nonlinear programming problem (2.22) apply to the modified problem formulation (2.58) as well. Each constantly disturbed NLP\((p)\) can be transformed to a standard nonlinear programming problem.

Analogous to the nominal problem case, the first-order KKT conditions (2.33)–(2.37) for the disturbed problem must hold. Thus, applying the KKT conditions for the problem formulation (2.58) we obtain a system
\[
F(y, \lambda, \mu, p) := \begin{pmatrix} \nabla_y \mathcal{L}(y, \lambda, \mu, p) \\ \Delta \cdot g(y, p) \\ h(y, p) \end{pmatrix} = 0_{(N_y + N_g + N_h) \times 1}
\]
where $\Delta := \text{diag}(\lambda_1, \ldots, \lambda_{N_y})$. At the nominal solution $\mathbf{F}(y_0^*, \lambda_0, \mu_0, p_0)$, the Jacobian of $\mathbf{F}(\cdot)$ with respect to the arguments $y$, $\lambda$, and $\mu$ is given by

$$\frac{\partial \mathbf{F}}{\partial (y, \lambda, \mu)}(y_0^*, \lambda_0, \mu_0, p_0) = \begin{pmatrix}
\nabla_y^2 \mathcal{L}(y_0^*, \lambda_0, \mu_0, p_0) & \nabla_y \mathbf{g}^T(y_0^*, p_0) & \nabla_y \mathbf{h}^T(y_0^*, p_0) \\
\Delta \cdot \nabla_y \mathbf{g}(y_0^*, p_0) & \Gamma & 0_{N_y \times N_x} \\
\nabla_y \mathbf{h}(y_0^*, p_0) & 0_{N_x \times N_y} & 0_{N_y \times N_x}
\end{pmatrix},$$

(2.60)

where the matrix $\Gamma$ is defined by $\Gamma := \text{diag}(g_1(y_0^*, p_0), \ldots, g_{N_y}(y_0^*, p_0))$. The matrix (2.60) is the KKT matrix. Since we assume from SOSC (2.39) that $\nabla_y^2 \mathcal{L}(y_0^*, \lambda_0, \mu_0, p_0)$ is positive definite on a nonvanishing ker($A_y^T(y_0^*, p_0)$) the KKT matrix is regular and therefore invertible. Hence, the classical implicit function theorem can be applied to (2.60) to obtain differentiable functions $y : P \rightarrow \mathbb{R}^{N_y}$, $\lambda : P \rightarrow \mathbb{R}^{N_x}$, and $\mu : P \rightarrow \mathbb{R}^{N_x}$ in a neighborhood of $p_0$ such that $\lim_{p \rightarrow p_0} y(p) = y_0^*$, $\lim_{p \rightarrow p_0} \lambda(p) = \lambda_0$, and $\lim_{p \rightarrow p_0} \mu(p) = \mu_0$ applies.

**Theorem 2.5** (Implicit Function Theorem (Fiacco [26])) Let the pair $(y_0^*, p_0) \in \mathbb{R}^{N_y} \times P$ be given. Suppose the function $\mathbf{K} : \mathbb{R}^{N_y} \times P \rightarrow \mathbb{R}^{N_y}$ with $\mathbf{K}(y_0^*, p_0) = 0$ is a continuously differentiable mapping at the point $(y_0^*, p_0)$ and its Jacobian with respect to $y$, i.e., $\nabla_y \mathbf{K}(y_0^*, p_0)$, is nonsingular and thus invertible. Then, there exists a neighborhood $\mathcal{V} \subset \mathbb{R}^{N_y}$ with $y_0^* \in \mathcal{V}$ and $P \subset \mathbb{R}^{N_p}$ with $p_0 \in P$ and a differentiable and unique function $y : P \rightarrow \mathcal{V}$, which satisfies the condition $\mathbf{K}(y(p), p) = 0$ for all $p \in P$ with the unique solution $y = y(p)$.

Proof The proof can be found in Oliveira [60].

In order to obtain explicit formulae for the sensitivity derivatives of the optimal solutions and Lagrange multipliers let us apply the Theorem 2.5.

**Theorem 2.6** (Differentiability of Optimal Solutions (Büskens [10])) Consider the parametric nonlinear problem (2.58) under the following assumptions:

- the functions $f(\cdot)$, $g(\cdot)$, and $h(\cdot)$ are twice continuously differentiable with respect to $y$ in a neighborhood of $y_0$. Also, let the gradients $\nabla_y f$, $\nabla_y g$, $\nabla_y h$ and the functions $g(\cdot)$ and $h(\cdot)$ be continuously differentiable with respect to $p$ in the neighborhood of $p_0$;
- $y_0^*$ is a feasible solution such that the LICQ holds; and
- the triple $(y_0^*, \lambda_0, \mu_0)$ is an optimal solution for the nominal problem formulation $\text{NLP}(p_0)$ with the reference parameter vector $p = p_0$ which satisfies the SOSC of Theorem 2.4 and the strict complementarity condition (2.38).

Then, the following applies:

- there exists a neighborhood $P \subset \mathbb{R}^{N_p}$ of $p_0$, i.e., $p_0 \in P$, and once continuously differentiable functions $y : P \rightarrow \mathbb{R}^{N_y}$, $\lambda : P \rightarrow \mathbb{R}^{N_x}$, and $\mu : P \rightarrow \mathbb{R}^{N_x}$ with the following properties:
1. \( y(p_0) = y^*_0, \lambda(p_0) = \lambda_0, \mu(p_0) = \mu_0; \)

2. for all \( p \in P \), the triple \( (y(p), \lambda(p), \mu(p)) \) satisfies the SOSC of Theorem 2.4 and the strict complementarity condition (2.38) for the perturbed NLP\((p)\) problem. \( y(p) \) is a unique local minimum of the NLP\((p)\) problem with the Lagrange multipliers \( \lambda(p) \) and \( \mu(p) \).

• for all \( p \in P \) the following holds:

1. the set of active constraints is constant in \( P \), i.e.,

\[ \mathcal{I}(y(p), p) \equiv \mathcal{I}(y^*_0, p_0), \ \forall p \in P; \] and

2. the Jacobian matrix of active constraints

\[ A_y(y(p), p) := \begin{pmatrix} \nabla_y g_i(y(p), p) & \nabla_y h_j(y(p), p) \end{pmatrix}, \]

where the indices are given by \( i \in \mathcal{I}(y^*_0, p) \) and \( j = 1, \ldots, N_h \), has full rank, i.e.,

\[ \text{rank}(A_y(y(p), p)) = N_s, \ \forall p \in P. \]

\( \square \)

Proof A compact proof which exclusively considers the active constraints can be found in Büskens [10]. The original proof with the consideration of all constraints is presented in the work of Fiacco [26].

Remark 2.8 The requirement that the active sets of the nominal and disturbed problems keep unchanged can be quite restrictive. It is sometimes advantageous to remove some constraints, if possible, to enlarge the neighborhood \( P \) of \( p_0 \).

Corollary 2.1 (Sensitivity Differentials) Suppose Theorem 2.5 holds. Then, the sensitivity differentials are obtained by differentiation of the identity \( K(y(p), p) \equiv 0 \) at the nominal parameter \( p_0 \) with respect to \( p \) which yields

\[ \nabla_y K(y^*_0, p_0) \cdot \nabla_p y(p_0) + \nabla_p K(y^*_0, p_0) = 0. \] (2.61)

Rearranging (2.61) yields the explicit formulae

\[ \nabla_p y(p_0) = -(\nabla_y K(y^*_0, p_0))^{-1} \nabla_p K(y^*_0, p_0). \]

\( \square \)

Analogous to Corollary 2.1, we obtain the sensitivity differentials of the parametric nonlinear programming problem (2.58) by differentiation of its optimal solution (2.59) at the nominal parameter \( p_0 \) with respect to all parameters \( p \). This yields the linear equations
\[
\begin{pmatrix}
\nabla^2 y L_0 & 
\nabla y g_0^T & 
\nabla_y h_0^T \\
\Delta \cdot \nabla y g_0 & 
\Gamma & 
0_{N_y \times N_h} \\
\nabla_y h_0 & 
0_{N_y \times N_g} & 
0_{N_y \times N_h}
\end{pmatrix}
\begin{pmatrix}
\frac{dy}{dp}(p_0) \\
\frac{dy}{d\lambda}(p_0) \\
\frac{dy}{d\mu}(p_0)
\end{pmatrix} + 
\begin{pmatrix}
\nabla_{yp} L_0 \\
\Delta \cdot \nabla_{yp} g_0 \\
\nabla_{yp} h_0
\end{pmatrix} = 0_{(N_y+N_g+N_h) \times N_p}
\]

where \(L_0 := L(y^*_0, \lambda_0, \mu_0, p_0), g_0 := g(y^*_0, p_0), \) and \(h_0 := h(y^*_0, p_0)\) are defined for the sake of a better readability.

Hence, the explicit formulae for the sensitivity differentials are obtained by the following corollary.

\textbf{Corollary 2.2} \textbf{(Sensitivity Differentials of the Optimal Solution)} Suppose Theorem 2.6 holds. Then, the sensitivity differentials of the optimal solution (2.59) at the nominal parameter \(p_0\) with respect to all parameters \(p\) are given by

\[
\begin{pmatrix}
\frac{dy}{dp}(p_0) \\
\frac{dy}{d\lambda}(p_0) \\
\frac{dy}{d\mu}(p_0)
\end{pmatrix} = - \begin{pmatrix}
\nabla^2 y L_0 & 
\nabla y g_0^T & 
\nabla_y h_0^T \\
\Delta \cdot \nabla y g_0 & 
\Gamma & 
0_{N_y \times N_h} \\
\nabla_y h_0 & 
0_{N_y \times N_g} & 
0_{N_y \times N_h}
\end{pmatrix}^{-1} \begin{pmatrix}
\nabla_{yp} L_0 \\
\Delta \cdot \nabla_{yp} g_0 \\
\nabla_{yp} h_0
\end{pmatrix}.
\]

One might think that the calculation of the sensitivity differentials is a byproduct of all modern iterative solution algorithms that apply Newton’s method to the KKT system, e.g., SQP methods, and hence the appearance of the Kuhn–Tucker matrix. However, the exact computation of the Hessian of the Lagrangian \(\nabla^2 y L_0\) in every iteration is far too expensive for large-scale problems. In general, the direct approximation of (2.60) by the wealth of Quasi-Newton methods is not possible because they do not converge to the KKT matrix due to their low rank approximation of the Hessian of the Lagrangian. This makes the embedding of the sensitivity analysis in modern solution algorithms a challenging task.

A more straightforward way is proposed by Büskens and Maurer [12] using the calculation of the sensitivity differentials as a \textit{post-optimal} analysis.

\subsection{Sensitivity Analysis of the Objective Function and Constraints}

Beside the sensitivity of the optimal solution \(y^*\) the sensitivity of the objective function and constraints are also interesting. The calculation of these sensitivities can be accomplished by simply extending Corollary 2.2 to \textit{associated functions}. Associated functions can be interpreted as functions which provide information about the solution of disturbed NLP(\(p\)).
**Definition 2.24 (Associated Function)** Let \( \tilde{a} : \mathbb{R}^{N_x} \times \mathbb{R}^{N_s} \times \mathbb{R}^{N_h} \times \mathcal{P} \to \mathbb{R}^{N_c} \) be a continuously differentiable function and let the assumptions of the sensitivity Theorem 2.6 hold. Hence, there exists a neighborhood of \( p_0, \mathcal{P} \subset \mathbb{R}^{N_p} \), to define differentiable functions \( y : \mathcal{P} \to \mathbb{R}^{N_x}, \lambda : \mathcal{P} \to \mathbb{R}^{N_s}, \) and \( \mu : \mathcal{P} \to \mathbb{R}^{N_h} \). Then, the function

\[
a(p) := \tilde{a}(y(p), \lambda(p), \mu(p), p)
\]

is denoted as an associated function to NLP(p).

According to Definition 2.24, if the function \( f(\cdot), g(\cdot), \) and \( h(\cdot) \) are continuously differentiable with respect to the parameter vector \( p \), then the Lagrangian \( \mathcal{L}(\cdot) \) is an associated function. Moreover, the objective function \( f(\cdot) \) and the constraints \( g(\cdot) \) and \( h(\cdot) \) are associated functions too. Using Corollary 2.2 and the chain rules of differentiation we obtain the sensitivity theorem for the associated function.

**Theorem 2.7 (Sensitivity of Associated Functions)** Let \( a : \mathcal{P} \to \mathbb{R}^{N_c} \) be an associated function. Further, let the assumptions of the sensitivity Theorem 2.6 for the disturbed NLP(p) with the optimal solution \( y_0^*, \lambda_0, \mu_0 \) for \( p = p_0 \) hold. Then

\[
\frac{da}{dp}(p_0) = - (\nabla_y a \nabla_\lambda a \nabla_\mu a) \left[ \begin{array}{ccc} \nabla^2_y \mathcal{L}_0 & \nabla_y g_0^T & \nabla_y h_0^T \\ \Delta \cdot \nabla_y g_0 & \Gamma & 0_{N_s \times N_h} \\ \nabla_y h_0 & 0_{N_h \times N_s} & 0_{N_h \times N_h} \end{array} \right]^{-1} \left[ \begin{array}{c} \nabla_{yp} \mathcal{L}_0 \\ \nabla_{yp} g_0 \\ \nabla_{yp} h_0 \end{array} \right] + \nabla_p a
\]

is the sensitivity differential of the associated function.

**Proof** The function \( a(\cdot) \) is a differentiable mapping with respect to the argument \( y(\cdot), \lambda(\cdot), \mu(\cdot), \) and \( p \). Together with Corollary 2.2 one obtains the result of Theorem 2.7 straightforwardly.

With this result it is easy to state the sensitivities of the objective function and the constraints.

**Corollary 2.3 (Sensitivity Differentials of the Constraints)** Suppose Theorem 2.6 holds. Then, the sensitivity differentials of constraints are given by

\[
\frac{dg}{dp}(y_0^*, p_0) = - (\nabla_y g_0 \ 0_{N_s \times N_s} \ 0_{N_s \times N_h}) \left[ \begin{array}{ccc} \nabla^2_y \mathcal{L}_0 & \nabla_y g_0^T & \nabla_y h_0^T \\ \Delta \cdot \nabla_y g_0 & \Gamma & 0_{N_s \times N_h} \\ \nabla_y h_0 & 0_{N_h \times N_s} & 0_{N_h \times N_h} \end{array} \right]^{-1} \left[ \begin{array}{c} \nabla_{yp} \mathcal{L}_0 \\ \nabla_{yp} g_0 \\ \nabla_{yp} h_0 \end{array} \right] + \nabla_p g_0
\]

\[
= \nabla_y g_0 \frac{dy}{dp}(p_0) + \nabla_p g_0 \tag{2.62}
\]

\[
\frac{dh}{dp}(y_0^*, p_0) = - (\nabla_y h_0 \ 0_{N_s \times N_s} \ 0_{N_s \times N_h}) \left[ \begin{array}{ccc} \nabla^2_y \mathcal{L}_0 & \nabla_y g_0^T & \nabla_y h_0^T \\ \Delta \cdot \nabla_y g_0 & \Gamma & 0_{N_s \times N_h} \\ \nabla_y h_0 & 0_{N_h \times N_s} & 0_{N_h \times N_h} \end{array} \right]^{-1} \left[ \begin{array}{c} \nabla_{yp} \mathcal{L}_0 \\ \nabla_{yp} g_0 \\ \nabla_{yp} h_0 \end{array} \right] + \nabla_p h_0
\]

\[
= \nabla_y h_0 \frac{dy}{dp}(p_0) + \nabla_p h_0 = 0. \tag{2.63}
\]
In particular
\[
\frac{dg_0}{dp}(y_0^*, p_0) = 0, \quad \forall i \in I(y_0^*, p_0)
\]
holds.

\[\triangle\]

**Proof** Equations (2.62)–(2.63) are obtained directly from Theorem 2.7. It remains to show that the sensitivity differentials are zero for active constraints. It follows for every \(g_0\) with \(i \in I(y_0^*, p_0)\) from Corollary 2.2 that
\[
\nabla_y g_0 \frac{dy}{dp}(p_0) = -\nabla_p g_0
\]
and therefore
\[
\frac{dg_i}{dp}(y_0^*, p_0) = -\nabla_p g_0 + \nabla_p g_0 = 0_{1 \times N_p}.
\]
The proof can be applied to the equality constraints in the same manner. \[\square\]

**Corollary 2.4** (Sensitivity Differential of the Objective Function) Suppose Theorem 2.6 holds. Additionally, let the function \(f(\cdot)\) be once continuously differentiable with respect to \(p\). Then, the sensitivity differential of the objective function is given by
\[
\frac{df}{dp}(y_0^*, p_0) = -\left(\begin{array}{c}
\nabla_y f_0 \frac{dy}{dp}(p_0) \\
\nabla_p f_0
\end{array}\right)
\]
with \(f_0 := f(y_0^*, p_0)\).

\[\triangle\]

**Proof** Equation (2.64) follows analogously from Corollary 2.3. With the first-order necessary conditions from Theorem 2.3 it follows that
\[
\frac{df}{dp}(y_0^*, p_0) \overset{\text{Thm. 2.3}}{=} -\lambda_0^T \nabla_y g_0 \frac{dy}{dp}(p_0) - \mu_0^T \nabla_y h_0 \frac{dy}{dp}(p_0) + \nabla_p f_0.
\]
Because of the strict complementarity condition (2.38), the inactive constraints can be disregarded. Finally, it follows from Corollary 2.3 and the definition of the Lagrangian 2.13 that
\[
\frac{df}{dp}(y^*, p_0) = -\lambda_0^T \nabla_y g_0 \frac{dy}{dp}(p_0) - \mu_0^T \nabla_y h_0 \frac{dy}{dp}(p_0) + \nabla_p f_0
\]

Coroll. 2.3
\[
\equiv -\lambda_0^T (-\nabla_p g_0) - \mu_0^T (-\nabla_p h_0) + \nabla_p f_0
\]
\[
= \nabla_p \left(f_0 + \lambda_0^T g_0 + \mu_0^T h_0\right)
\]
Def. 2.13
\[
\equiv \nabla_p L_0.
\]

Remark 2.9 One can realize that the sensitivity differential of the objective function using (2.64) needs second-order information to calculate \((dy/dp)(p_0)\). In contrast, (2.65) requires only first-order information. This has the consequence that for the evaluation of the objective function sensitivities the solution of the linear equation system and the differential of second order are not required which allows for a more efficient computation.

A further differentiation of this identity yields the second-order sensitivities of the objective function.

Corollary 2.5 (Second-Order Sensitivity Differential of the Objective Function)
Suppose Theorem 2.6 holds. Additionally, let the functions \(f(\cdot), g(\cdot),\) and \(h(\cdot)\) be twice continuously differentiable with respect to \(p\). Then, the second-order sensitivity differential of the objective function is given by
\[
\frac{d^2 f}{dp^2}(y^*, p_0) = \left(\frac{dy}{dp}\right)^T (p_0) \cdot \nabla_{y,p} L_0 + \left(\frac{d\lambda}{dp}\right)^T (p_0) \cdot \nabla_p g_0 + \left(\frac{d\mu}{dp}\right)^T (p_0) \cdot \nabla_p h_0 + \nabla_p^2 L_0
\]
(2.66)
\[
= 2 \left(\frac{dy}{dp}\right)^T (p_0) \cdot \nabla_{y,p} L_0 + \left(\frac{dy}{dp}\right)^T (p_0) \cdot \nabla_p^2 L_0 \frac{dy}{dp}(p_0) + \nabla_p^2 L_0.
\]
(2.67)

Proof Using \(\nabla_p L_0 = \nabla_p f_0 + \lambda_0^T \nabla_p g_0 + \mu_0^T \nabla_p h_0\). Eq. (2.66) is obtained by differentiation of \((df/dp)(p_0)\). We obtain
\[
\frac{d^2 f}{dp^2}(y^*, p_0) = \frac{d}{dp} \left(\nabla_p f_0 + \lambda_0^T \nabla_p g_0 + \mu_0^T \nabla_p h_0\right)
\]
\[
= \left(\frac{dy}{dp}\right)^T (p_0) \cdot \nabla_{y,p} f_0 + \nabla_p^2 f_0 + \left(\frac{d\lambda}{dp}\right)^T (p_0) \cdot \nabla_p g_0
\]
\[
+ \left(\frac{d\mu}{dp}\right)^T (p_0) \cdot \nabla_p h_0 + \left(\frac{dy}{dp}\right)^T (p_0) \cdot \left[\lambda_0^T , \nabla_p g_0\right]
\]
\[
+ \left(\frac{dy}{dp}\right)^T (p_0) \cdot \left[\mu_0^T , \nabla_p h_0\right] + \left[\lambda_0^T , \nabla_p g_0\right] + \left[\mu_0^T , \nabla_p h_0\right]
\]
\[
= \left(\frac{dy}{dp}\right)^T (p_0) \cdot \nabla_{y,p} L_0 + \left(\frac{d\lambda}{dp}\right)^T (p_0) \cdot \nabla_p g_0 + \left(\frac{d\mu}{dp}\right)^T (p_0) \cdot \nabla_p h_0 + \nabla_p^2 L_0
\]
where the operator \([\mathbf{b}(\mathbf{x}), \mathbf{A}(\mathbf{x})]\) with the arguments \(\mathbf{b}(\mathbf{x}) \in \mathbb{R}^{N_m}\) and \(\mathbf{A}(\mathbf{x}) \in \mathbb{R}^{N_m+N_p}\) is defined by

\[
[b(\mathbf{x}), A(\mathbf{x})] := \left[ \sum_{i=1}^{N_m} b_i(\mathbf{x}) \nabla_x a_{i,1}(\mathbf{x}), \ldots, \sum_{i=1}^{N_m} b_i(\mathbf{x}) \nabla_x a_{i,p}(\mathbf{x}) \right].
\]

where the variable \(\mathbf{x}\) of the operator is a substituted variable for \(\mathbf{y}(\cdot)\) or \(\mathbf{p}\), respectively.

This completes the proof of the first Eq. (2.66) of the second-order sensitivities of the objective function.

Now follows the proof of (2.67). Using \(\nabla_y \mathcal{L}_0 = \nabla_y f_0 + \lambda^T \nabla_y \mathbf{g}_0 + \mu^T \nabla_y \mathbf{h}_0\) and applying Theorem 2.7 yields the following first equation straightforwardly. The second and third equations follow from Corollary 2.3.

\[
\nabla_y^2 \mathcal{L}_0 \frac{d\mathbf{y}}{d\mathbf{p}}(\mathbf{p}_0) + \nabla_y \mathbf{g}_0 \frac{d\lambda}{d\mathbf{p}}(\mathbf{p}_0) + \nabla_y \mathbf{h}_0 \frac{d\mu}{d\mathbf{p}}(\mathbf{p}_0) = -\nabla_{\mathbf{p}} \mathcal{L}_0 \tag{2.68}
\]

\[
\Delta \cdot \nabla_y \mathbf{g}_0 \frac{d\mathbf{y}}{d\mathbf{p}}(\mathbf{p}_0) = -\Delta \cdot \nabla_{\mathbf{p}} \mathbf{g}_0 \tag{2.69}
\]

\[
\nabla_y \mathbf{h}_0 \frac{d\mathbf{y}}{d\mathbf{p}}(\mathbf{p}_0) = -\nabla_{\mathbf{p}} \mathbf{h}_0. \tag{2.70}
\]

Multiplying the first Eq. (2.68) from the right-hand side by \((d\mathbf{y}/d\mathbf{p})(\mathbf{p}_0)\), the second equation (2.69) from the left-hand side by \((d\lambda/d\mathbf{p})^T(\mathbf{p}_0)\), and the third equation (2.70) from the left-hand side by \((d\mu/d\mathbf{p})^T(\mathbf{p}_0)\) and summing them up yields the following relationships with

\[
\left(\frac{d\lambda}{d\mathbf{p}}\right)^T(\mathbf{p}_0) \cdot \nabla_y \mathbf{g}_0 \frac{d\mathbf{y}}{d\mathbf{p}}(\mathbf{p}_0) = -\left(\frac{d\lambda}{d\mathbf{p}}\right)^T(\mathbf{p}_0) \cdot \nabla_{\mathbf{p}} \mathbf{g}_0 \tag{2.71}
\]

\[
\left(\frac{d\mu}{d\mathbf{p}}\right)^T(\mathbf{p}_0) \cdot \nabla_y \mathbf{h}_0 \frac{d\mathbf{y}}{d\mathbf{p}}(\mathbf{p}_0) = -\left(\frac{d\mu}{d\mathbf{p}}\right)^T(\mathbf{p}_0) \cdot \nabla_{\mathbf{p}} \mathbf{h}_0. \tag{2.72}
\]

Putting (2.71)–(2.72) into the first sensitivity Eq. (2.66), then rearranging yields

\[
\frac{d^2 f}{d\mathbf{p}^2}(\mathbf{y}_0^*, \mathbf{p}_0) = \frac{d\mathbf{y}}{d\mathbf{p}}(\mathbf{p}_0) \cdot \nabla_{\mathbf{p}} \mathcal{L}_0 + \left(\frac{d\lambda}{d\mathbf{p}}\right)^T(\mathbf{p}_0) \cdot \nabla_{\mathbf{p}} \mathbf{g}_0 + \left(\frac{d\mu}{d\mathbf{p}}\right)^T(\mathbf{p}_0) \cdot \nabla_{\mathbf{p}} \mathbf{h}_0 + \nabla_{\mathbf{p}}^2 \mathcal{L}_0
\]

\[
= \left(\frac{d\mathbf{y}}{d\mathbf{p}}\right)^T(\mathbf{p}_0) \cdot \nabla_{\mathbf{p}} \mathcal{L}_0 - \left(\frac{d\lambda}{d\mathbf{p}}\right)^T(\mathbf{p}_0) \cdot \nabla_{\mathbf{p}} \mathbf{g}_0 + \left(\frac{d\mu}{d\mathbf{p}}\right)^T(\mathbf{p}_0) \cdot \nabla_{\mathbf{p}} \mathbf{h}_0 \frac{d\mathbf{y}}{d\mathbf{p}}(\mathbf{p}_0) + \nabla_{\mathbf{p}}^2 \mathcal{L}_0
\]

\[
= \left(\frac{d\mathbf{y}}{d\mathbf{p}}\right)^T(\mathbf{p}_0) \cdot \nabla_{\mathbf{p}} \mathcal{L}_0 + \nabla_{\mathbf{p}}^2 \left[ \left(\frac{d\mathbf{y}}{d\mathbf{p}}\right)^T(\mathbf{p}_0) \cdot \mathcal{L}_0 \right] \frac{d\mathbf{y}}{d\mathbf{p}}(\mathbf{p}_0) + \nabla_{\mathbf{p}}^2 \mathcal{L}_0
\]

\[
= 2 \left(\frac{d\mathbf{y}}{d\mathbf{p}}\right)^T(\mathbf{p}_0) \cdot \nabla_{\mathbf{p}} \mathcal{L}_0 + \frac{d\mathbf{y}}{d\mathbf{p}}(\mathbf{p}_0) \cdot \nabla_{\mathbf{p}}^2 \left[ \mathcal{L}_0 \frac{d\mathbf{y}}{d\mathbf{p}}(\mathbf{p}_0) \right] + \nabla_{\mathbf{p}}^2 \mathcal{L}_0
\]
with the assumption that

$$
\left( \frac{dy}{dp} \right)^T (p_0) \nabla_{yp} \mathcal{L}_0 = \left( \left( \frac{dy}{dp} \right)(p_0) \nabla_{yp} \mathcal{L}_0 \right)^T = \nabla_{yp} \mathcal{L}_0 \left( \frac{dy}{dp} \right)(p_0).
$$

The numerical advantage of (2.67) in contrast with (2.66) is its independence from the Jacobian of the constraints and the Jacobian of the Lagrange multipliers.

### 2.4.2 Linear Perturbations

A special case of parameter disturbance is the linear perturbation in the constraints. If the objective function is independent of the parameter vector $p$ and the constraints involve linear perturbation we obtain an optimization problem of the form

$$
\min_{y \in \mathbb{R}^{Ny}} \quad f(y) \\
\text{subject to} \quad g_i(y) - p[i] \leq 0, \quad i = 1, \ldots, N_g \quad (2.73) \\
h_j(y) - p[N_g + j] = 0, \quad j = 1, \ldots, N_h
$$

where $p \in \mathbb{R}^{N_g + N_h}$ is the linear perturbation vector. Then, we obtain immediately the sensitivities of the optimal solution, the constraints, and the objective function with the following corollary.

**Corollary 2.6** (Sensitivity Differentials for Linear Perturbation in the Constraints)

Let the problem formulation (2.73) with linear perturbation in the constraints be given where $p_0 \in \mathbb{R}^{N_g + N_h}$ is the reference parameter vector. Assume that the implicit function Theorem 2.5 for the problem formulation (2.73) holds. Then, the sensitivity differentials with respect to $p$ are obtained by:

1. **optimal solution**:

$$
\begin{bmatrix}
\frac{dy}{dp_{[i]}(p_0)} \\
\frac{d\lambda}{dp_{[i]}(p_0)} \\
\frac{d\mu}{dp_{[i]}(p_0)} \\
\frac{d\mu}{dp_{[i]}(p_0)}
\end{bmatrix}
= \begin{bmatrix}
\nabla^2_{yp} \mathcal{L}_0 & \nabla_y g_0^T & \nabla_y h_0^T \\
\Delta \cdot \nabla_y g_0 & \Gamma & 0 \\
\nabla_y h_0 & 0 & 0
\end{bmatrix}_{[..,Ny+i]}^{-1}
$$
2. **constraints:**

\[
\frac{dg}{dp}(y^*_0, p_0) = \nabla_y g_0 \frac{dy}{dp}(p_0) - I_N;
\]

3. **objective function:**

\[
\frac{df}{dp}(y^*_0, p_0) = \nabla_p L_0 = \begin{pmatrix} \lambda_0 \\ \mu_0 \end{pmatrix}.
\]

In particular \(\frac{df}{dp}(y^*_0, p_0) = 0\) for \(i \notin I(y^*_0, p_0)\) holds; and

4. **second-order sensitivity of the object function:**

\[
\frac{d^2f}{dp^2}(y^*_0, p_0) = \begin{pmatrix} \frac{d\lambda}{dp}(y^*_0, p_0) \\ \frac{d\lambda}{dp}(y^*_0, p_0) \\ \frac{d\mu}{dp}(y^*_0, p_0) \end{pmatrix}.
\]

In particular \(\frac{d^2f}{dp^2}(y^*_0, p_0) = 0\) for \(i \notin I(y^*_0, p_0)\) holds.

\[\triangle\]

**Proof** The proof follows directly by insertion. \[\square\]

**Remark 2.10** The same procedure can be applied to the objective function of the form \(f(y) - r^Ty\). The interested reader may refer to Büskens [10] for more details.

### 2.4.3 Approximation of the Perturbed Solution

The calculated sensitivity differentials can now be used to approximate the perturbed solutions of the optimization problem, if the perturbation of the nominal parameters \(\Delta p := p - p_0\) is small enough, such that the set of active constraints does not change. How to calculate the confidence region for the active set will be the topic of Sect. 2.4.4.

The approximated solution \((\tilde{y}(p), \tilde{\lambda}(p), \tilde{\mu}(p))\) can be calculated very fast by the use of the sensitivity differentials and therefore this method is suitable for an online optimization strategy, which means that the approximated solutions can be calculated in real-time even on the electronic control unit of the vehicle, if the optimal solution \(y^*_0\) and the sensitivity differentials \((dy/dp)(p_0)\) are calculated offline before.

Instead of restarting the optimization procedure again, as it is done by the brute force method, the perturbed solution triple \((y(p), \lambda(p), \mu(p))\) is approximated by a first-order Taylor series expansion according to...
2.4 Sensitivity Analysis

\[ y(p) \approx \tilde{y}(p) := y^*_0 + \frac{dy}{dp}(p_0) \Delta p \]  
\[ \text{(2.74)} \]

where \( \tilde{\lambda}(\cdot) \) and \( \tilde{\mu}(\cdot) \) can be obtained analogously to (2.74). The computation of (2.74) is very fast because the first-order Taylor approximation uses only matrix-vector multiplication and vector addition.

For the objective function a Taylor approximation of second-order is even possible with

\[ f(y(p), p) \approx f(\tilde{y}(p), p) := f_0 + \frac{df}{dp}(y^*_0, p_0) \Delta p + \frac{1}{2} \Delta p^T \frac{d^2 f}{dp^2}(y^*_0, p_0) \Delta p \]  
\[ \text{(2.75)} \]

where \( (df/dp)(y^*_0, p_0) \) and \( (d^2 f/dp^2)(y^*_0, p_0) \) are calculated by (2.65) and (2.67), respectively. The approximated solution of the disturbed problem can deviate from the optimal solution of the disturbed problem. The following theorem shows that the error of the disturbed solution \( \tilde{y}(p) \), the functions \( f(\tilde{y}(p), p) \), \( g(\tilde{y}(p), p) \), \( h(\tilde{y}(p), p) \), the multipliers \( \tilde{\lambda}(p) \), \( \tilde{\mu}(p) \), and the Lagrangian \( L(\tilde{y}(p), \tilde{\lambda}(p), \tilde{\mu}(p), p) \) is of second-order with respect to the disturbance \( \Delta p \). Again, only the active constraints of the inequality set are important. Therefore, we define the vector of active inequality constraints as

\[ g_a := (g_i)_{i \in I(y^*_0, p_0)}. \]  
\[ \text{(2.76)} \]

**Theorem 2.8 (Error Estimation)** Let the assumptions from the sensitivity Theorem 2.6 hold. Let the functions \( f(\cdot), g(\cdot), \) and \( h(\cdot) \) be three times continuously differentiable with respect to \( y(\cdot) \) and \( p \). Then, there exists a neighborhood of \( p_0 \) with \( \mathcal{P} \) such that the following error estimations hold for all \( p = p_0 + \Delta p \in \mathcal{P} \):

\[
\| y(p) - \tilde{y}(p) \| = O(\| \Delta p \|^2)
\]
\[
\| f(y(p), p) - f(\tilde{y}(p), p) \| = O(\| \Delta p \|^2)
\]
\[
\| g_a(\tilde{y}(p), p) \| = O(\| \Delta p \|^2)
\]
\[
\| h(\tilde{y}(p), p) \| = O(\| \Delta p \|^2)
\]
\[
\| \lambda(p) - \tilde{\lambda}(p) \| = O(\| \Delta p \|^2)
\]
\[
\| \mu(p) - \tilde{\mu}(p) \| = O(\| \Delta p \|^2)
\]
\[
\| \nabla_y L(\tilde{y}(p), \tilde{\lambda}(p), \tilde{\mu}(p), p) \| = O(\| \Delta p \|^2).
\]

**Proof** The proof can be found in Büskens [11].
2.4.4 Approximation of the Confidence Region

The real-time strategy presented before can only be applied to perturbations that keep the set of active indices unchanged. This restriction is necessary since a change in the active set of constraints might result in a violation of the regularity condition and thus render the sensitivity differentials incorrect or even invalid. This means that the Eqs. (2.74) and (2.75) are only bounded by the assumption that the active set remains the same. An estimate of when the active set will change can be made by examining the Lagrange multipliers of the active inequality constraints (2.76) and linear approximations of the inactive constraints. In order to obtain linear approximations of the inactive constraints we need linear approximations of the Lagrange multiplier $\lambda(p)$ and inequality constraint $g(p)$ with

$$
\lambda(p) \approx \tilde{\lambda}(p) := \hat{\lambda} + \frac{d\lambda}{dp}(p_0) \Delta p
$$

$$
g(p) \approx \tilde{g}(p) := \hat{g} + \frac{dg}{dp}(y_0^*, p_0) \Delta p.
$$

An inactive constraint ($g_i \neq 0, \lambda_i = 0$) becomes active if $g_i$ vanishes under the influence of disturbance. Using the Taylor series expansion from (2.77), this means

$$
0 = g_i(p) \approx \hat{g}_i + \frac{dg_i}{dp}(y_0^*, p_0) \Delta p, \quad i \notin I(y_0^*, p_0).
$$

Accordingly, the disturbance $\Delta p_{[j]}$ which causes $g_i$ to enter the set of active indices can be approximated by using the linear prediction

$$
\Delta p_{[j]} \approx -\frac{\hat{g}_i}{\frac{dg_i}{dp}(y_0^*, p_0)} , \quad i \notin I(y_0^*, p_0), \quad j \in \{1, \ldots, N_p\}.
$$

We assume that $(dg_i/dp_{[j]})(y_0^*, p_0) \neq 0$ holds. For the case $(dg_i/dp_{[j]})(y_0^*, p_0) = 0$ the inequality constraint $g_i$ cannot become active and the region of confidence is not restricted by $\Delta p_{[j]}$.

Analogous to the previous case, an active constraint ($g_i = 0, \lambda_i \neq 0$) becomes inactive if $\lambda_i$ vanishes under the influence of disturbance. Using the Taylor series expansion from (2.77), this means

$$
0 = \lambda_i(p) \approx \hat{\lambda}_i + \frac{d\lambda_i}{dp}(p_0) \Delta p, \quad i \in I(y_0^*, p_0).
$$

Thus, the disturbance $\Delta p_{[j]}$ which causes $g_i$ to leave the set of active indices can be approximated by using the linear prediction
\[ \Delta \mathbf{p}_{[j]} \approx -\frac{\hat{\lambda}_i}{\frac{\partial}{\partial \mathbf{p}_{[j]}(\mathbf{p}_0)}}, \quad i \in \mathcal{I}(\mathbf{y}_0^*, \mathbf{p}_0), \quad j \in \{1, \ldots, N_p\}. \] (2.79)

We assume again that \((\frac{\partial}{\partial \mathbf{p}_{[j]}(\mathbf{p}_0)}) \neq 0\) holds.

This enables us to predict a change in the set of active indices to occur at the smallest value of \(\Delta \mathbf{p}_{[j]}\) obtained from applying (2.78) and (2.79) to all inequality constraints.

### 2.5 Multi-Objective Optimization

Many real-world optimization problems involve multiple objectives which often conflict with each other. This becomes apparent if an improvement of one objective may lead to a deterioration of another. Thus, a single solution, which can optimize all objectives simultaneously, does not exist. Logically, we need to introduce another concept for optimization of such problems which finds the best trade-off solutions.

Let us define a multi-objective optimization problem.

**Definition 2.25** (*Constrained Multi-Objective Optimization*) A constrained multi-objective optimization problem can be formulated as

\[
\min \quad \mathbf{f}(\mathbf{y}) = \begin{bmatrix} f_1(\mathbf{y}) \\ \vdots \\ f_N(\mathbf{y}) \end{bmatrix}
\] (2.80)

where \(\mathbf{y}\) is taken from the constrained decision space \(\Omega\) which is defined by

\[
\Omega := \{\mathbf{y} \in \mathbb{R}^N | \mathbf{g}(\mathbf{y}) \leq \mathbf{0} \land \mathbf{h}(\mathbf{y}) = \mathbf{0}\} \tag{2.81}
\]

and the constraints are defined as

\[
\mathbf{g}(\mathbf{y}) := \begin{bmatrix} g_1(\mathbf{y}) \\ \vdots \\ g_{N_g}(\mathbf{y}) \end{bmatrix} \quad \text{and} \quad \mathbf{h}(\mathbf{y}) := \begin{bmatrix} h_1(\mathbf{y}) \\ \vdots \\ h_{N_h}(\mathbf{y}) \end{bmatrix}.
\]

The objective functions and constraints \(f : \mathbb{R}^{N_f} \to \mathbb{R}^{N_f}, \mathbf{g} : \mathbb{R}^{N_g} \to \mathbb{R}^{N_g},\) and \(\mathbf{h} : \mathbb{R}^{N_h} \to \mathbb{R}^{N_h}\) are real-valued.

\[ \triangle \]

**Remark 2.11** To enlarge the potential algorithmic class that deals with problem (2.80) and (2.81) we do not enforce the requirement that the functions \(f(\cdot), \mathbf{g}(\cdot),\) and \(\mathbf{h}(\cdot)\) have to be all continuously differentiable.
The minimize operator in Definition 2.25 means, that we want to minimize all objectives simultaneously. If there exists a single solution $y^*$, which minimizes each of the objectives $f_1(y^*), \ldots, f_N(y^*)$, we call it a trivial solution. But, in general, a minimum of one objective will not be optimal for other objectives. Therefore, we need to define another optimality concept: the concept of Pareto optimality. This concept was first proposed by V. Pareto and is formally defined in terms of nondominated decision vectors as follows (cf. Miettinen [56] and Zhou et al. [74]).

**Definition 2.26 (Dominated Decision Vector)** A decision vector $y_1 \in \mathcal{O}$ is said to dominate another decision vector $y_2 \in \mathcal{O}$, $(y_1 < y_2)$ if $f_i(y_1) \leq f_i(y_2)$ for all $i = 1, \ldots, N$ and $f_j(y_1) < f_j(y_2)$ for at least one index $j$.

**Definition 2.27 (Pareto Optimality)** A decision vector $y^* \in \mathcal{O}$ is called Pareto optimal or nondominated if there does not exist another decision vector $y \in \mathcal{O}$ which dominates $y^*$.

The set of all Pareto optimal solutions is called a Pareto optimal set, which can be nonconvex and disconnected. So, the solution of problem definition 2.25 leads in general to a Pareto optimal set. The image of the Pareto optimal set is called its Pareto front, which is illustrated in Fig. 2.2.

### 2.5.1 Elitist Multi-Objective Evolutionary Algorithm

The elitist multi-objective evolutionary algorithm (MOEA) is categorized as a stochastic search technique which maintains and manipulates a population of solutions and implements a survival of the fittest strategy in its search for better solutions.
These solutions are clustered into nondominated solution fronts, which are used together with the best solutions found so far (elitism concept) to generate an offspring generation by selection, crossover, and mutation. There are several elitist MOEAs (Zhou et al. [74] gives a good overview), but we will concentrate only on the elitist nondominated search genetic algorithm (NSGA), more precisely the second generation NSGA-II, by Deb et al. [19].

The principle working scheme of NSGA-II is described in Algorithm 2.2.

**Algorithm 2.2** Elitist Multi-objective Evolutionary Algorithm NSGA-II (Deb et al. [19])

1: \( k \leftarrow 0 \)
2: Choose a population size \( N_{pop} \)
3: Generate a random initial parent population \( P_k \) of size \( N_{pop} \)
4: \( Q_k \leftarrow \text{MAKENEWPOPULATION}(P_k) \)
5: if prescribed number of generations \( N_{gen} \) is reached then
6: \( \text{stop} \)
7: end if
8: Generate a combined population \( R_k \leftarrow P_k \cup Q_k \)
9: Calculate \( F \leftarrow \text{FASTNONDOMINATEDSORT}(R_k) \)
10: Set \( P_{k+1} \leftarrow \emptyset \) and \( l \leftarrow 1 \)
11: while \( |P_{k+1}| + |F_l| \leq N_{pop} \) do
12: \( P_{k+1} \leftarrow P_{k+1} \cup F_l \)
13: \( l \leftarrow l + 1 \)
14: end while
15: \( I_{dist} \leftarrow \text{CROWDINGDISTANCEASSIGNMENT}(F_l) \)
16: \( F_l = \text{SORT}(F_l, I_{dist}) \)
17: \( P_{k+1} \leftarrow P_{k+1} \cup F_l \) \( 1,...,N_{pop}-|P_{k+1}| \)
18: \( k \leftarrow k + 1 \)
19: Return to step 4.

The first population \( P_0 \) is randomly generated. Then, for each iteration \( k = 0, \ldots, N_{gen} \) the population \( P_k \) is used to generate an offspring population \( Q_k \) with the same number of members \( N_{pop} \) as the parent population \( P_k \) (step 4 of Algorithm 2.2). Therefore, a tournament selection algorithm is applied to select some of the parents, which are then used to generate the children for the offspring population by crossover or mutation. A simple tournament selection algorithm which selects the fitter of two randomly selected members from the population \( P_k \) as parents is called a **binary tournament selection**. An algorithm \( \text{MAKENEWPOPULATION} \) which implements a binary tournament selection for the selection of \( N_{pool} \) parents (step 3 of Algorithm 2.3) and which performs a **simulated binary crossover** (SBX) of these parents with a chance of 90% and a polynomial mutation with a chance of 10% as it is implemented by Seshadri [69] can be stated as follows:
Algorithm 2.3 \textsc{MakeNewPopulation}(P)

1: \( Q \leftarrow \emptyset \)
2: \( N_q \leftarrow 0 \)
3: Perform a binary tournament selection for population \( P \) to determine a set of parents \( P_{pool} \) with a given size \( N_{pool} \).
4: \textbf{for} \( i \leftarrow 1 \) \textbf{to} \( N_{pop} \) \textbf{do}
5: \hspace{1em} Get a random number \( r \in [0, 1] \)
6: \hspace{1em} \textbf{if} \( r < 0.9 \) \textbf{then}
7: \hspace{2em} Calculate \( c_1 \) and \( c_2 \) by SBX from randomly selected parents \( p_1, p_2 \in P_{pool} \)
8: \hspace{2em} \( Q[N_q+1] \leftarrow c_1 \)
9: \hspace{2em} \( Q[N_q+2] \leftarrow c_2 \)
10: \hspace{2em} \( N_q \leftarrow N_q + 2 \)
11: \hspace{1em} \textbf{else}
12: \hspace{2em} Calculate \( c_3 \) by polynomial mutation from a randomly selected parent \( p \in P_{pool} \)
13: \hspace{2em} \( Q[N_q+1] \leftarrow c_3 \)
14: \hspace{2em} \( N_q \leftarrow N_q + 1 \)
15: \hspace{1em} \textbf{end if}
16: \textbf{end for}

For each decision variable the SBX (step 7 of Algorithm 2.3) selects a random number \( w \in [0, 1] \) and calculates

\[
\beta_j(w) = \begin{cases} 
(2w)^{\frac{1}{\eta_c+1}}, & \text{if } w \leq 0.5 \\
1 \cdot \left[2(1 - w)\right]^{\frac{1}{\eta_c+1}}, & \text{otherwise}
\end{cases}
\]

for all \( j = 1, \ldots, N_y \). The \( j \)-th decision variable of the two children \( c_1 \) and \( c_2 \) are then calculated from the two parents \( p_1 \) and \( p_2 \) by the calculation rule

\[
c_{j,1} = \frac{1}{2} \cdot [(1 - \beta_j) p_{j,1} + (1 + \beta_j) p_{j,2}] \\
c_{j,2} = \frac{1}{2} \cdot [(1 + \beta_j) p_{j,1} + (1 - \beta_j) p_{j,2}].
\]

If a child’s decision variable violates a constraint, it is set to the extremum.

For each decision variable, the polynomial mutation (step 12 of Algorithm 2.3) selects a random number \( w \in [0, 1] \) and calculates

\[
\delta_j(w) = \begin{cases} 
(2w)^{\frac{1}{\eta_m+1}} - 1, & \text{if } w < 0.5 \\
1 - [2(1 - w)]^{\frac{1}{\eta_m+1}}, & \text{otherwise}
\end{cases}
\]

for all \( j = 1, \ldots, N_y \). The \( j \)-th decision variable of the child \( c_3 \) is then calculated from the parent \( p \) by

\[
c_{j,3} = p_j + (p_j^{max} - p_j^{min}) \delta_j,
\]
where $p_j^{\text{min}}$ and $p_j^{\text{max}}$ are the lower and upper bounds of the $j$-th decision variable from the parent $p$, respectively.

The parent population $P_k$ and the offspring population $Q_k$ are then combined into a common population $R_k$ of double size. From $R_k$ the best entries are selected as members for the next parent population $P_{k+1}$. The selection from the combined population $R_k$ is a task of the elitism concept, which guarantees that the best members of all generations are preserved.

To select the best members from the combined population $R_k$, the set is first sorted by the fast nondominated sorting algorithm. This algorithm assigns each member to a front $F_r$, where $r$ is the domination rank of the member, i.e., the number of dominating members. The algorithm can be stated as follows:

\begin{algorithm}
\begin{algorithmic}[1]
\Function{FastNonDominatedSort}{P} \Comment{cf. Deb et al. [19]}
\State \ForAll{$p \in P$} \Do
\State $S_p \leftarrow \emptyset$
\State $N_p \leftarrow 0$
\EndFor
\State \ForAll{$q \in P$} \Do
\If{$p < q$} \Then
\State $S_p \leftarrow S_p \cup \{q\}$
\ElsIf{$q < p$} \Then
\State $N_p \leftarrow N_p + 1$
\EndIf
\EndFor
\State \If{$N_p = 0$} \Then
\State $prank \leftarrow 1$
\State $F_1 \leftarrow F_1 \cup \{p\}$
\EndIf
\EndFor
\State $r \leftarrow 1$
\While{$F_r \neq \emptyset$} \Do
\State $Q \leftarrow \emptyset$
\ForAll{$p \in F_r$} \Do
\ForAll{$q \in S_p$} \Do
\State $N_q \leftarrow N_q - 1$
\If{$N_q = 0$} \Then
\State $q_{\text{rank}} \leftarrow r + 1$
\State $Q \leftarrow Q \cup \{q\}$
\EndIf
\EndFor
\EndFor
\State $r \leftarrow r + 1$
\State $F_r \leftarrow Q$
\EndWhile
\EndFunction
\end{algorithmic}
\end{algorithm}

The fronts $F_r$ beginning with $r = 1$ are added to the next population $P_{k+1}$ until an index $s$ is reached for which the addition of the front $F_s$ would exceed the population size. For this front $F_s$ the crowding distances are calculated according to Algorithm 2.5.
Algorithm 2.5 CROWDDISTANCEASSIGNMENT(F) (cf. Deb et al. [19])

1: $j \leftarrow |F|
2: \text{for } i \leftarrow 1, \text{ to } j \text{ do}
3: \quad I_{\text{dist}}^{i} \leftarrow 0
4: \text{end for}
5: \text{for all objectives } m \in [1, \ldots, N_f] \text{ do}
6: \quad F \leftarrow \text{sort}(F, m)
7: \quad I_{\text{dist}}^{1} \leftarrow \infty
8: \quad I_{\text{dist}}^{j} \leftarrow \infty
9: \quad \text{for } i \leftarrow 2 \text{ to } j - 1 \text{ do}
10: \quad I_{\text{dist}}^{i} \leftarrow I_{\text{dist}}^{i} + \frac{F_{[m]}^{i+1} - F_{[m]}^{i-1}}{f_{[m]}^{\max} - f_{[m]}^{\min}}
11: \text{end for}
12: \text{end for}

In Algorithm 2.5, $F_{[m]}^{i}$ refers to the $m$-th objective function value of the $i$-th individual in the set $F$ and $f_{[m]}^{\min}$ and $f_{[m]}^{\max}$ are the minimum and maximum values of the $m$-th objective function.

The members of $F_s$ with greatest crowding distances are added to the population $P_{k+1}$ until the desired population size is reached. The selection of the greatest crowding distances leads to a greater diversity of the members.

2.5.2 Remarks for MOGAs

Although the underlying mechanisms of evolutionary algorithms (EA) are simple, these algorithms have some advantages compared with smooth nonlinear programming techniques such as:

- high robustness; and
- natural ability to cope with discontinuous and non-differentiable problems.

On the other hand is the none deterministic convergence of these algorithms which makes them slow and only applicable in offline optimizations, whereas SQP can even have a superlinear convergence property. However, SQP can perform badly on problems with non-convex and discontinuous behavior because of inaccurate gradient information used to determine the search direction.

Another popular, but more naive, approach is to transcribe the multi-objective problem into a single-objective problem by focusing on one particular Pareto optimal solution at a time (Johannesson et al. [44]). Such methods have the difficulty that this procedure has to be applied many times, hopefully finding different solutions which approximate the exact Pareto optimal front as good as possible.
2.6 Bibliographical Notes

The history of nonlinear programming is vast and diverse. A good historical overview is provided by Giorgi and Kjeldsen [38]. The first activity, involving finding a local minimizer of a nonlinear function, can be traced back to the years of the Second World War and the years immediately following the war. The term “nonlinear programming” was first mentioned in the fifties in the paper of Kuhn and Tucker [49]. It was later discovered that Karush and John [58] published similar results earlier. The emergence of Kuhn and Tucker [49], however, can be seen as a starting point of nonlinear programming as an autonomous field of research which has strong relationships to other disciplines like mathematical analysis, numerical and nonsmooth analysis, linear algebra, operations research, etc.

Some further subjects of nonlinear programming can be found in the following citations. Arrow et al. [3] weakened the KKT conditions and provided some analysis of the various constraint qualifications. An overview of constraint qualifications for the KKT conditions can be found in Eustaquio et al. [24]. Further examinations and special applications of Kantorovich’s theorem can be found in Polyak [63], Ferreira and Svaiter [25], and Potra [64].

A wide variety of algorithms exist for solving NLPs, none of which can be considered preferable for all problems, but there is a consensus in the literature that SQP is one of the most effective methods for solving constrained NLPs.

The first SQP method, which used the exact Hessian of the Lagrangian, was introduced in 1963 by Wilson [70].

There is a wealth of good surveys and overview papers of SQP methods. For instance, Eldersveld [23], Boggs and Tolle [8], Gould and Toint [41], Schittkowski and Yuan [68], and Gill and Wong [35], to name but a few. A comprehensive introduction to nonlinear programming and SQP can be found in the very good textbook of Nocedal and Wright [59]. The SQP approach can be employed in both line-search and trust-region frameworks. The trust-region methods are completely disregarded in this book but can be found in many textbooks of nonlinear programming. Interested readers are recommended to consult Conn et al. [15] for a comprehensive introduction to trust-region methods and Fletcher and Leyffer [30] and Fletcher et al. [32] for details regarding the filter SQP methods omitted from this book.

Alternative approaches for SQP which use an augmented Lagrangian

$$\Psi(y, \lambda, \mu, \varrho, \nu) = f(y) + \lambda^T g(y) + \mu^T h(y) + \frac{1}{2} \varrho^T g^2(y) + \nu^T h^2(y)$$

as a merit function are described in Boggs and Tolle [8] and Gill et al. [37]. These methods have the advantage that the merit function is differentiable and thus allows for more accurate line-search methods. Moreover, by construction they avoid the Maratos effect. Slightly modified augmented Lagrangian were also proposed by Rockafellar [66], Di Pillo and Grippo [22], Schittkowski [67], Byrd et al. [13], Anitescu [1], Gill and Robinson [34], and Bertsekas [6] as merit functions for constrained nonlinear programming.
More details about line-search methods can be found in Dennis Jr and Schnabel [21], Lemaréchal [50], and Hager and Zhang [42].

More information about the convergence of Quasi-Newton updates can be found in Boggs et al. [9].

In commercial solvers, the quadratic subproblems arising in SQP methods are often solved by AS methods, which were completely disregarded in this book because this solver class for quadratic subproblems is especially successful for small- and medium-scale problems but not for large-scale problems. A very good and stable AS algorithm is described in Goldfarb and Idnani [39].

Sensitivity analysis has been widely used in linear programming. Fiacco [26] made some substantial contributions to the theory of sensitivity analysis for nonlinear programming problems and laid the cornerstone for further research activities. For instance, the transfer of sensitivities to real-time applications, which was extensively investigated by Büskens [10–12].

Since the 1970s several evolutionary methodologies have been proposed, mainly genetic algorithms, evolutionary programming, and evolution strategies. All work on a population of solutions and are therefore blueprints for dealing with conflicting objectives (multi-objectives). An excellent introduction to multi-objective evolutionary algorithms is presented by Zitzler et al. [75] and a great survey paper is given by Zhou et al. [74].

References

16. Coope I (1985) The Maratos effect in sequential quadratic programming algorithms using the l1 exact penalty function. Research report (University of Waterloo. Faculty of Mathematics), University of Waterloo, Computer Science Department
52. Maratos N (1978) Exact penalty function algorithms for finite dimensional and control optimization problems. PhD thesis, Imperial College London (University of London)
68. Schittkowski K, Yuan YX (2011) Sequential quadratic programming methods. Wiley encyclopedia of operations research and management science
Hybrid Systems, Optimal Control and Hybrid Vehicles
Theory, Methods and Applications
Böhme, Th.J.; Frank, B.
2017, XXXIII, 530 p. 143 illus., 113 illus. in color.,
Hardcover
ISBN: 978-3-319-51315-7