New Trust Region SQP Methods for Continuous and Integer Optimization

Von der Universität Bayreuth
zur Erlangung des Grades eines
Doktors der Naturwissenschaften (Dr. rer. nat.)
genehmigte Abhandlung

von

Oliver Exler

aus Schwabach

1. Gutachter:	Prof. Dr. Klaus Schittkowski
2. Gutachter:	Prof. Dr. Hans Josef Pesch
3. Gutachter:	Prof. Dr. Michael Ulbrich, TU München

Tag der Einreichung:	20. November 2012
Tag des Kolloquiums:	18. November 2013

Contents

Zusammenfassung

In dieser Arbeit werden neue Verfahren zur Lösung restringierter nichtlinearer Optimierungsprobleme vorgestellt. Die vorgeschlagenen Algorithmen lassen sich den sequentiellen quadratischen Optimierungsverfahren – *sequential quadratic programming* (SQP) *methods* – zuordnen. Es werden zwei Arten von Problemstellungen betrachtet. Die Probleme der einen Klasse werden als nichtlineare Optimierungsprobleme bezeichnet – *nonlinear programs* (NLP). Sie zeichnen sich dadurch aus, dass der Wertebereich aller Optimierungsvariablen reell ist. Die zweite Problemklasse umfasst die gemischtganzzahligen nichtlinearen Probleme – *mixed-integer nonlinear programs* (MINLP). MINLPs sind eine Erweiterung der NLPs, da zusätzlich zu den reellen Variablen noch Variablen auftreten, deren Wertebereich auf die ganzen Zahlen beschränkt ist. Die betrachteten Probleme beider Klassen weisen sowohl Gleichungs- als auch Ungleichungsnebenbedingungen auf.

Motiviert ist die Arbeit durch die Weiterentwicklung eines neuartigen Algorithmus zur Lösung von MINLPs, der erstmals von Exler und Schittkowski [37] diskutiert wurde. Es handelt sich um eine Erweiterung der SQP Verfahren für die gemischtganzzahlige Optimierung. Der Ansatz ersetzt die kontinuierlichen Teilprobleme durch gemischt-ganzzahlige quadratische Probleme. Ziel ist es von den guten Konvergenzeigenschaften der SQP Verfahren bezüglich der kontinuierlichen Variablen zu profitieren. Es werden zwei neue Varianten des ursprünglichen Algorithmus vorgestellt.

Es ist bekannt, dass die Konvergenz eines SQP Verfahrens ohne zusätzliche Maßnahmen nicht für jeden beliebigen Startwert garantiert werden kann. Um die globale Konvergenz sicherzustellen, werden Techniken der Trust-Region-Verfahren – *trust region methods* – angewandt. In der ursprünglichen Fassung von Exler und Schittkowski verwendet der Algorithmus zur Lösung von MINLPs die L_∞-Penalty Funktion. Ohne spezielle Strategien kann bei Verwendung dieser Funktion die schnelle lokale Konvergenz von SQP Verfahren gestört werden. Das Auftreten des sogenannten Maratos-Effekts führt zu einer unnötigen Verkleinerung der Schrittlänge im Verfahren. Beim ersten hier vorgestellten Algorithmus für MINLPs werden zusätzliche Korrekturschritte zweiter Ordnung berechnet – *second order correction* (SOC) *steps*. Die Berechnung von SOC Schritten ist einer von mehreren möglichen Ansätzen, um die schnelle lokale Konvergenz zu bewahren. Diese Schritte erfordern jedoch weitere Funktionsauswertungen. Bei MINLPs aus der Anwendung im Ingenieurwesen geschieht die Bestimmung von Funktionswerten jedoch oftmals durch aufwendige Simulationscodes, so dass eine einzelne Funktionsauswertung bereits Minuten oder sogar Stunden dauern kann. Das Ziel muss es folglich sein, die Anzahl der erforderlichen Funktionsauswertungen möglichst gering zu halten.

Aus diesem Grund steht die Untersuchung von Methoden im Vordergrund, die lokal schnell konvergieren und dabei auf die Berechnung von SOC Schritten verzichten können. Da für NLPs fundierte theoretische Grundlagen vorliegen, die für MINLPs teilweise nicht existieren, liegt der Schwerpunkt dieser Arbeit auf der Entwicklung und theoretischen Analyse eines neuen Algorithmus zur Lösung restringierter nichtlinearer Optimierungsprobleme der Problemklasse NLP. Der neue Algorithmus verwendet

als Meritfunktion eine erweiterte Lagrange-Funktion. Die schnelle lokale Konvergenz bleibt auch ohne zusätzliche SOC Schritte erhalten. Die Verwendung einer differenzierbaren Meritfunktion, wie der erweiterten Lagrange-Funktion, wurde für SQP Verfahren in Kombination mit Trust-Region-Verfahren für gleichheitsrestringierte Probleme bereits untersucht. Verfahren, welche auch Ungleichungen betrachten, überführen die Ungleichungen oftmals durch Schlupfvariablen in Gleichungen. Der Ansatz dieser Arbeit behandelt Ungleichungen ohne eine solche Umformung.

Der neue *Trust Region SQP* Algorithmus für NLPs wird hinsichtlich seiner theoretischen Konvergenzeigenschaften analysiert. Hierbei wird sowohl auf die globale, als auch auf die lokale Konvergenz eingegangen. Es wird gezeigt, dass die von dem Algorithmus erzeugte Folge von Iterationspunkten unter geeigneten Voraussetzungen für jeden beliebigen Startpunkt mindestens einen Häufungspunkt besitzt, welcher die Karush-Kuhn-Tucker Bedingungen des Ausgangsproblems erfüllt. Ist die generierte Folge von Iterationspunkten nahe genug am Optimum, so werden unter geeigneten Voraussetzungen volle SQP Schritte akzeptiert und eine schnelle lokale Konvergenz tritt ein.

Die Erkenntnisse aus der Entwicklung des kontinuierlichen Algorithmus fließen direkt in die Weiterentwicklung des Algorithmus für MINLPs ein. Eine modifizierte Varianten des gemischt-ganzzahligen Algorithmus von Exler und Schittkowski [37] wird präsentiert. Hier findet keine Berechnung von Korrekturschritten mehr statt, so dass die zusätzlichen Funktionsauswertungen vermieden werden.

Alle entwickelten Algorithmen liegen als vollständig dokumentierte `FORTRAN` Implementierungen vor. Die Effizienz der Verfahren und ihrer Implementierungen wird anhand einer Vielzahl von Testproblemen demonstriert. Die theoretisch erzielbaren Konvergenzeigenschaften können für den vorgestellten Algorithmus für kontinuierliche Probleme auch numerisch verifiziert werden. Auch die Übertragung der Erkenntnisse aus der kontinuierlichen Optimierung auf den gemischt-ganzzahligen Fall erweist sich als effizient.

Abstract

In this thesis new algorithms are presented that address nonlinear optimization problems. The algorithms belong to the class of sequential quadratic programming (SQP) methods. Two problem formulations that arise frequently in real-world applications are considered. Both have in common that functions are nonlinear and the formulations contain equality and inequality constraints. For the one class of problems the domain of all variables is $\mathbb{R}$. These problems are called nonlinear programming (NLP) problems. Many applications also require that some of the featured variables are restricted to the domain $\mathbb{Z}$. Problems with additional integer variables are called mixed-integer nonlinear programs (MINLP) and are also considered here.

This work is motivated by the advancement of an algorithm for solving MINLPs that was first discussed by Exler and Schittkowski [37]. The algorithm adapts concepts of SQP methods to mixed-integer nonlinear optimization. The new approach replaces the continuous quadratic problems by mixed-integer quadratic problems. The aim is to profit from the fast local convergence properties of SQP methods at least with respect to the continuous variables when integer variables remain fixed. Two new versions of the underlying algorithm of Exler and Schittkowski are presented.

It is well-known that SQP methods might not converge for any arbitrary starting point. To obtain global convergence, techniques of trust region methods are employed by the new algorithms. The first version of an algorithm for MINLPs presented in this thesis employs the L_∞-penalty function as merit function. Applying this penalty function might lead to a slow convergence. The so-called Maratos effect requires the reduction of the step length so that fast convergence is lost. Hence, safeguards have to be added. The presented algorithm calculates additional second order correction (SOC) steps. Calculating SOC steps is a frequently used approach to obtain fast local convergence. There also exist other techniques. The SOC steps require additional function evaluations. Frequently, function values of mixed-integer problems arising in the field of engineering are evaluated by running time-consuming simulation tools, where a single function evaluation can take minutes or even hours. Thus, the goal of the development of an efficient method has to be that the number of needed function evaluations is as small as possible.

For that reason the investigation of methods that obtain fast local convergence without calculating SOC steps is the key aspect of this thesis. As a fundamental theory is available for NLPs, whereas MINLPs lack in most of these concepts, the main part of this thesis presents and analyzes a new trust region SQP algorithm addressing NLPs. The algorithm proposed here avoids the calculation of SOC steps by using an augmented Lagrangian function as merit function. In trust region methods a differentiable merit function, such as an augmented Lagrangian function, was employed in the past for equality constrained problems. Methods that also treat inequality constraints, transform these constraints into equality constraints. The new algorithm does not reformulate the underlying problem.

The proposed algorithm for NLPs is described in detail. The global and local con-

vergence properties of the new algorithm are investigated. Under suitable assumptions it is shown that for any arbitrary starting point the sequence of generated iterates contains at least one accumulation point that is a Karush-Kuhn-Tucker point of the underlying NLP. Under certain conditions fast local convergence is proved, as full SQP steps will be accepted close to a solution. Thus, no additional SOC steps are required.

Due to the insight that is gained by the development of the algorithm for NLPs, an additional version of the algorithm for MINLPs can be stated. The second algorithm also enhances the algorithm of Exler and Schittkowski [37], but does not calculate SOC steps anymore and the extra function evaluations are avoided.

All presented algorithms are implemented in `FORTRAN` and completely documented. The code is evaluated on a set of almost 500 test problems. Numerical results show the good performance of the new algorithms. The numerical tests of the algorithm for NLPs indicate that the theoretical convergence results hold in practice. Moreover, the efficiency of the second algorithm for MINLPs that does not calculate SOC steps has improved compared to the first version with SOC steps.

1 Introduction

This thesis introduces new methods to solve nonlinear optimization problems. Many areas of science and engineering employ mathematical optimization methods. Frequently, the problems arising in these fields are obtained by combining a performance criterion, e.g., a cost function that is minimized, and a mathematical model that approximates a real-world system. The underlying model consists of variables which represent different states of the considered system. Relations between these variables are expressed by functions that can be either linear or nonlinear. The performance criterion is optimized subject to these constraints. Typical constraints are, for example, physical laws like mass balances or heat equations. Depending on the domain of the variables, the resulting optimization problems are classified differently.

The methods proposed in this thesis address two kinds of problem formulations. The first class of problems under consideration contains variables whose domain is $\mathbb{R}$, whereas the second formulation features additionally discrete variables that are restricted to the domain $\mathbb{Z}$. The combinatorial structure of the second class makes these problems extremely difficult to solve. For both classes of optimization problems new algorithms are proposed in this work.

The development of the new algorithms is motivated by the advancement of an algorithm discussed by Exler and Schittkowski [37], which enhances a first version introduced and implemented by Exler [33]. The algorithm addresses the aforementioned second class of optimization problems, the so-called *mixed-integer nonlinear programming* (MINLP) problems. They are defined as

$$\begin{aligned} \underset{x \in \mathbb{R}^{n_c},\, y \in \mathbb{Z}^{n_i}}{\text{minimize}} \quad & f(x,y) \\ \text{subject to} \quad & g_j(x,y) = 0\ , \quad j = 1, \ldots, m_e\ , \\ & g_j(x,y) \geq 0\ , \quad j = m_e + 1, \ldots, m\ , \end{aligned} \tag{1.1}$$

where x denotes the continuous variables and y denotes integer variables, respectively. The constant n_c is the number of continuous variables and n_i denotes the number of integer variables. The objective function $f(x,y)$ and the constraint functions $g_j(x,y)$, $j = 1, \ldots, m$, are all smooth, real-valued functions that are assumed to be twice continuously differentiable at least on $\mathbb{R}^{n_c}$. The nonnegative constant m_e denotes the number of equality constraints and m_e is less or equal to m.

Algorithms for mixed-integer problems (1.1) are of great interest, as a lot of problems arising in practice contain discrete variables, see for example Exler et al. [34], and Antelo et al. [1] for process design problems from chemical engineering, or Sendín, Exler, and Banga [108], where a problem from systems biology is considered. Several methods addressing problem (1.1) have been proposed, see, for instance, Floudas [44], and Grossmann and Kravanja [56] for reviews.

The algorithm of Exler and Schittkowski [37] differs from known methods, as it adapts the concepts of *sequential quadratic programming* (SQP) to mixed-integer nonlinear optimization. SQP methods approximate the solution to a problem by generating a sequence of iterates where the step from one iterate to the next is obtained as solution of a quadratic subproblem. Instead of solving continuous quadratic subproblems the proposed adaptation solves mixed-integer quadratic problems.

The underlying SQP methods are well-established tools to solve problems that belong to the class of *nonlinear programming* (NLP) problems. Here the domain of all variables of the optimization problems is continuous. The nonlinear programming problem is formulated as

$$\begin{aligned} \underset{x \in \mathbb{R}^n}{\text{minimize}} \quad & f(x) \\ \text{subject to} \quad & g_j(x) = 0 \ , \quad j = 1, \ldots, m_e \ , \\ & g_j(x) \geq 0 \ , \quad j = m_e + 1, \ldots, m \ . \end{aligned} \tag{1.2}$$

Again x denotes the n continuous variables. The objective function $f(x)$ and the m constraints $g_j(x)$, $j = 1, \ldots, m$, are all smooth, real-valued functions, and assumed to be at least twice continuously differentiable on the whole $\mathbb{R}^n$.

Problems of form (1.2), with small and medium size, can be efficiently solved by SQP methods. This approach was proposed for the first time in the 1960's by Wilson [126]. In the 1970's, SQP methods became famous due to Han [58, 59] and Powell [86]. Since that time a lot of research has been done on the theoretical background of SQP methods. Several theorems concerning the local and global convergence properties are established. For reviews see the papers, for example, by Schittkowski and Yuan [107], Boggs and Tolle [7], and Gould and Toint [52]. Detailed descriptions of SQP methods are presented, for instance, in Fletcher [40], Gill, Murray, and Wright [49], Stoer [112], Spellucci [110], and Sun and Yuan [113].

The intention of adapting SQP methods to mixed-integer optimization is to obtain fast local convergence with respect to the continuous variables x, at least in situations when integer variables y remain fixed. The desired fast local convergence can be expected for SQP methods in case the starting point is close to the solution. But there is no guarantee that the method will converge for any arbitrary starting point. For this reason stabilization techniques have to be added to the basic SQP method. To obtain convergence, a trial step d, which is the minimizer of a quadratic subproblem, has to fulfill some necessary conditions to be applied. Therefore, a merit function is introduced that measures the progress toward the solution that is achieved by taking step d. If the trial step d does not reduce the merit function sufficiently, then the step is rejected and not applied and the length of the next trial step is restricted to a reduced size. The common stabilization techniques differ in the way the progress is measured and the step size is restricted and adapted.

In Exler and Schittkowski [37] a trust region stabilization is suggested to restrict the length of the generated trial steps d in their mixed-integer algorithm. The step

size is controlled by adding a trust region constraint to the subproblem, in general formulated as

$$\|d\| \leq \Delta \ , \tag{1.3}$$

where $\|.\|$ stands for an arbitrary norm and $\Delta > 0$ denotes the trust region radius. Restriction (1.3) is a key ingredient of trust region methods, see, e.g., Conn, Gould, and Toint [21] for an extensive textbook. The use of trust region techniques for the mixed-integer algorithm in Exler and Schittkowski [37] is motivated by the need that all trial steps d have to fulfill the integer requirement with respect to the discrete variables y. As the trial steps d are obtained by solving mixed-integer subproblems with additional trust region constraint (1.3), the maximum length of the steps d can easily be controlled, and d is integer with respect to y for sure. This cannot be guaranteed when, for example, line search techniques are used to generate the trial steps, as performing a search along a determined direction might lead to fractional values with respect to integer variables.

The mixed-integer algorithm in Exler and Schittkowski [37] is based on a trust region algorithm introduced by Yuan [130], which uses the L_∞-penalty function as merit function. The L_∞-penalty function belongs to a class of penalty functions which features an undesirable behavior. They may destroy the fast local convergence of SQP methods. This effect was discovered first by Maratos [71]. In some cases an unnecessary reduction of the step size occurs even arbitrary close to the solution of a problem. The convergence slows down significantly. An illustrative example is given in Chapter 3. To overcome this effect several approaches have been proposed. Fletcher [39] suggested the calculation of additional steps that are called *second order correction* (SOC) steps. He showed that applying additional SOC steps circumvent the effect described by Maratos. The algorithms proposed by Yuan [130], Mayne and Polak [73], and Fukushima [45], also calculate second order corrections steps to avoid negative side effects. Details on the SOC technique can be found in Yuan [129].

A new mixed-integer algorithm is introduced in Chapter 5 that enhances the algorithm of Exler and Schittkowski [37]. The stated algorithm also employs the L_∞-penalty function and calculates second order correction steps for the continuous variables x as suggested in the underlying algorithm of Yuan [130]. Because of these SOC steps, fast local convergence can be obtained with respect to x in case the integer variables y remain unchanged. On the other hand, the strategy requires extra effort for calculating the SOC steps and, moreover, additional function evaluations are needed. To be applicable to real-world problems, where function evaluations are frequently obtained by running time-consuming simulation tools, it is required that the developed method needs as few function evaluations as possible. Here time-consuming means that a single function evaluation might take several days. Thus, the calculation of SOC steps and the corresponding additional function evaluations should be avoided if possible.

The aim of this thesis is the development of strategies to get around the calculation of second order correction steps. Thus, methods are investigated that guarantee fast

local convergence without requiring additional SOC steps, when the SQP method is stabilized by the trust region framework. As the mixed-integer algorithm employs SOC steps only for continuous variables x, the analysis concentrates on the continuous nonlinear problem (1.2). Fundamental concepts, as optimality criteria, exist for problems of form (1.2), and they can be used to derive theoretical convergence properties of an algorithm addressing these problems. As mixed-integer optimization lacks in these concepts, a convergence analysis would be more complex in this case. Thus, the main part of this thesis outlines the development and investigation of a new trust region SQP algorithm for optimizing the continuous problem (1.2). The algorithm uses new techniques that avoid the calculation of SOC steps. The obtained insight is then applied to improve the mixed-integer algorithm addressing problem (1.1). In Chapter 5 a second algorithm for mixed-integer problems is presented that adapts the techniques of the new continuous algorithm. The second mixed-integer algorithm does not calculate SOC steps anymore.

The new continuous trust region SQP algorithm addressing problems of form (1.2) employs a differentiable merit function instead of the L_∞-penalty function. By applying a differentiable merit function the calculation of second order correction steps can be avoided. Differentiable merit functions, namely augmented Lagrangian functions, have already been used in combination with trust region methods but restricted to equality constrained problems, see, for instance, Celis, Dennis, and Tapia [17], Dennis, El-Alem, and Maciel [24], El-Alem [30, 31], and Powell and Yuan [93]. Schittkowski [100], and Powell and Yuan [92] use a differentiable augmented Lagrangian function as merit function in line search methods. Fast local convergence is achieved without additional safeguards.

In this thesis the augmented Lagrangian and the techniques proposed in Schittkowski [100] are adapted to trust region methods. Thus, the new trust region algorithm is applicable to problems that feature equality and inequality constrains. Moreover, the inequality constraints are treated without any modification of the original problem formulation. Frequently, the inequality constraints are transformed into equality constraints by introducing slack variables, see, for example, Byrd, Gilbert, and Nocedal [15], and Niu and Yuan [76]. In El-Alem and El-Sobky [32], and Omojokun [80] an active set strategy is proposed to handle inequality constraints, but there is no local convergence analysis available.

Adding the trust region constraint (1.3) to the subproblems of an SQP method can lead to infeasible problems. Thus, a strategy for handling inconsistency of the subproblems is proposed. Whenever, a subproblem cannot be solved the algorithm enters a feasibility restoration phase. This idea goes back to the filter methods proposed in Fletcher and Leyffer [42], and Fletcher, Leyffer, and Toint [43]. During the restoration phase the new trial step is obtained by solving reformulated subproblems where the constraints are scaled. Scaling is also done by Vardi [123], Byrd, Schnabel, and Shultz [16], and Omojokun [80], but here the scaling is applied to each subproblem and not only in a separate restoration phase.

The convergence properties of the proposed continuous algorithm are analyzed in

detail. It is shown that under suitable assumptions the new algorithm converges globally, i.e., for any starting point x_0 the sequence generated by the algorithm has at least one accumulation point that satisfies the Karush-Kuhn-Tucker conditions. Moreover, the local convergence properties of the algorithm are investigated. It is shown that fast local convergence is obtained without additional safeguards.

The following chapter introduces the notation used in this thesis. In addition, basic concepts in nonlinear programming as optimality conditions and convergence properties are stated for the continuous problem formulation (1.2). As the mixed-integer problem formulation lacks in some of these concepts, the differences between the two problem formulations are discussed in Section 2.4.

In Chapter 3 the motivation of sequential quadratic programming methods is given by highlighting the relation to Newton's method. Since global convergence of the basic SQP methods cannot be guaranteed for arbitrary starting points, safeguards have to be added. The most frequently used globalization strategies are presented. Different merit functions, that are frequently applied to measure progress toward a stationary point, are introduced. Moreover, the concept of a filter is explained. A filter differs from merit function as no penalty parameter is required. The basic ideas of line search methods are described. As the new algorithms are trust region methods, the basic concepts, namely models, predicted reduction, and actual reduction, are presented in Section 3.4. The remainder of the chapter gives an overview of existent approaches that differ mainly in the way of handling inconsistent subproblems. As above mentioned, the mixed-integer algorithms presented later in Chapter 5 are based on the algorithm of Yuan [130]. Therefore, the algorithm is stated in Section 3.4.3.

Chapter 4 presents the trust region SQP algorithm for continuous nonlinear optimization problems (1.2) with equality and inequality constraints. The key ingredients of the algorithm are motivated and described in detail. The subproblems are formulated that are solved and the procedure for handling inconsistent subproblems is explained. Applying an augmented Lagrangian function to measure progress, requires an appropriate adjustment of the involved penalty parameter. Update rules are presented and motivated. The convergence analysis is found in Section 4.2. The global convergence of the stated algorithm is analyzed in Section 4.2.1. Under adequate assumptions it is shown that the sequence of iterates generated by the algorithm has at least one accumulation point that satisfies the Karush-Kuhn-Tucker optimality conditions of the continuous problem (1.2). Section 4.2.2 addresses the local convergence analysis. Under suitable assumptions it is shown that the algorithm accepts full SQP steps and the trust region constraint is not active close to a solution.

Methods for mixed-integer nonlinear problems (1.1) are discussed in Chapter 5. A review of commonly used techniques is stated in the beginning of the chapter. In Section 5.2 the two new algorithms for mixed-integer optimization problems are presented. The advanced version of the algorithm introduced in Exler and Schittkowski [37] is discussed in Section 5.2.1. The modified second algorithm is described in Section 5.2.2. This algorithm does not calculate second order corrections steps by applying locally the augmented Lagrangian merit function as suggested for the new continuous algorithm

presented in Chapter 4.

The algorithms presented and discussed in this thesis are implemented in the code `MISQP`. Numerical results for the implementations are presented in Chapter 6. The code is tested on two collections of test problems. The first collection of test problems is published in Hock and Schittkowski [62] and Schittkowski [98] and consists of continuous nonlinear problems (1.2). The second set of problems contains mixed-integer problems (1.1), see Schittkowski [106]. `MISQP` integrates the algorithms that apply second order correction steps and those that do not calculate these steps. Thus, the efficiency of the different approaches can easily be compared. The program documentation of `MISQP` and a detailed description of the calling parameters and the reverse communication is given in Appendix A.

In Chapter 7 a final discussion is presented. Some comments are stated whether avoiding second order correction steps leads to an improved efficiency of the new mixed-integer algorithm. In addition, an outlook regarding the future work is given.

2 Concepts in Nonlinear Programming

This chapter summarizes basic theoretical concepts of nonlinear programming. In the beginning the used notation is stated, where the underlying problem formulation does not include integer variables. However, the introduced notation and definitions can easily be adapted to the mixed-integer problem formulation. As the problems contain constraints, the violation of these restrictions has to be analyzed. Thus, a measurement for restriction violation is introduced and the feasible region is defined. Moreover, the difference between active and inactive inequality constraints is explained.

In Section 2.2 optimality conditions are presented and additional requirements, such as constraint qualifications, are defined. Most of the described concepts are not applicable to mixed-integer optimization and an equivalent mixed-integer formulation does not exist. Section 2.3 introduces the superlinear and quadratic convergence rates, which can be used to measure the efficiency of an algorithm.

The final section highlights the difficulties arising in mixed-integer optimization. An example illustrates the differences between the mixed-integer problem formulation and the relaxed continuous counterpart.

2.1 Notation

The general *nonlinear programming* (NLP) problem is formulated as

$$\begin{aligned} \underset{x \in \mathbb{R}^n}{\text{minimize}} \quad & f(x) \\ \text{subject to} \quad & g_j(x) = 0 \ , \quad j = 1, \ldots, m_e \ , \\ & g_j(x) \geq 0 \ , \quad j = m_e + 1, \ldots, m \ , \end{aligned} \tag{2.1}$$

where the objective function $f(x)$ and the constraint functions $g_j(x)$, $j = 1, \ldots, m$, are all smooth, real-valued functions. All problem functions are assumed to be at least twice continuously differentiable for all $x \in \mathbb{R}^n$. The constants m_e and m are nonnegative integers with $0 \leq m_e \leq m$. The first m_e constraints are called *equality constraints*, whereas the remaining constraints are named *inequality constraints*. The vector $x \in \mathbb{R}^n$ contains the variables, also called *primal variables*, where the positive integer constant n is the number of variables.

Depending on the specific formulation and the characteristics of the problem functions, the general problem (2.1) is named differently. For example, a problem is called an *unconstrained problem* if m is zero and no constraints exist. In case no inequality constraints are defined, that is $m_e = m$, the problem is called an *equality constrained* problem. The *quadratic programming* (QP) problem, a special case of problem (2.1), plays a key role in the methods introduced in this work. All constraints of a QP

problem are linear functions and the objective function is quadratic.

The constraint functions $g_j(x)$, $j = 1, \ldots, m$, are also written as *constraint vector* $g(x) \in \mathbb{R}^m$, where

$$g(x) := (g_1(x), \ldots, g_m(x))^T \ . \tag{2.2}$$

Sometimes the analysis is restricted to either equality constraints or inequality constraints. To simplify the notation in these situations, the part of $g(x)$ that corresponds to the equality constraints is defined as

$$g_{\mathcal{E}}(x) := (g_1(x), \ldots, g_{m_e}(x))^T \in \mathbb{R}^{m_e} \ , \tag{2.3}$$

and, respectively, the part of $g(x)$ corresponding to inequality constraints is defined as

$$g_{\mathcal{I}}(x) := (g_{m_e+1}(x), \ldots, g_m(x))^T \in \mathbb{R}^{m-m_e} \ . \tag{2.4}$$

Here, the subscripts $\mathcal{E}$ and $\mathcal{I}$ highlight the specific constraints under consideration. The characters $\mathcal{E}$ and $\mathcal{I}$ also represent the sets of equality constraints and inequality constraints, respectively, where

$$\mathcal{E} := \{1, \ldots, m_e\} \ , \tag{2.5}$$

denotes the index set of equality constraints, and the index set of inequality constraints is denoted by

$$\mathcal{I} := \{m_e + 1, \ldots, m\} \ . \tag{2.6}$$

The proposed SQP methods require derivatives of the problem functions. The gradient of the objective function $f(x) : \mathbb{R}^n \to \mathbb{R}$ at x is defined as

$$\nabla f(x) := \left(\frac{\partial f}{\partial x_1}(x), \ldots, \frac{\partial f}{\partial x_n}(x)\right)^T \in \mathbb{R}^n \ . \tag{2.7}$$

If necessary a subscript is added to the notation to highlight which part of the gradient is considered, e.g., $\nabla_x f(x)$ denotes the partial derivatives with respect to x.

In addition, second order information are needed by the proposed algorithms. The Hessian of the objective function $f(x)$ with respect to x is defined as

$$\nabla^2 f(x) := \begin{pmatrix} \dfrac{\partial^2 f}{\partial x_1 \partial x_1}(x) & \ldots & \dfrac{\partial^2 f}{\partial x_1 \partial x_n}(x) \\ \vdots & \ddots & \vdots \\ \dfrac{\partial^2 f}{\partial x_n \partial x_1}(x) & \ldots & \dfrac{\partial^2 f}{\partial x_n \partial x_n}(x) \end{pmatrix} \in \mathbb{R}^{n \times n} \ . \tag{2.8}$$

The gradients $\nabla g_j(x) \in \mathbb{R}^n$ and the Hessians $\nabla^2 g_j(x) \in \mathbb{R}^{n \times n}$, $j = 1, \ldots, m$, of the constraints are defined accordingly. The Jacobian of the constraints is denoted by $\nabla g(x)^T \in \mathbb{R}^{m \times n}$, where

$$\nabla g(x) := \Big(\nabla g_1(x), \ldots, \nabla g_m(x)\Big) \in \mathbb{R}^{n \times m} \ , \tag{2.9}$$

and the columns of matrix $\nabla g(x)$ in (2.9) are the gradients of the constraints.

As problem (2.1) contains constraints $g_j(x)$, $j = 1, \ldots, m$, which have to be satisfied, the analysis of feasibility and restriction violation is inevitable. An $x \in \mathbb{R}^n$ is said to be a *feasible* point if all restrictions are fulfilled. To simplify the measurement of constraint violation, the vector $g(x)^- \in \mathbb{R}^m$ is introduced and defined as

$$\begin{aligned} g_j(x)^- &:= g_j(x) \;, & j &= 1, \ldots, m_e \;, \\ g_j(x)^- &:= \min(g_j(x), 0) \;, & j &= m_e + 1, \ldots, m \;. \end{aligned} \tag{2.10}$$

Thus, the equation $g(x)^- = 0$ holds at a feasible point x. Here 0 denotes a vector of zeros of appropriate size. This simplified notation for a vector 0 is also used in the remainder of this thesis.

Using definition (2.10), the *feasible region* $\mathcal{F}$ of problem (2.1), to say the set of all points $x \in \mathbb{R}^n$ that satisfy the constraints, can be defined as

$$\mathcal{F} := \{x \in \mathbb{R}^n \mid \; \|g(x)^-\|_1 = 0\} \;, \tag{2.11}$$

where $\|.\|_1$ denotes the L_1-norm, i.e., $\|g(x)^-\|_1 := \sum_{j=1}^{m} |g_j(x)^-|$. In the following chapters two more norms are used frequently. The L_2-norm is denoted by $\|.\|_2$ and the L_∞-norm is denoted by $\|.\|_\infty$.

The feasible region $\mathcal{F}$ as defined by (2.11) is enlarged to points where the constraint violation is less than a given threshold $\beta \in \mathbb{R}_0^+$, i.e., $\|g(x)^-\|_1 \leq \beta$. Here $\mathbb{R}_0^+$ denotes all nonnegative values in $\mathbb{R}$. The *extended feasible region* is defined as

$$\mathcal{F}(\beta) := \{x \in \mathbb{R}^n \mid \; \|g(x)^-\|_1 \leq \beta\} \;, \tag{2.12}$$

where $\beta \geq 0$. Obviously, equation $\mathcal{F}(\beta) = \mathcal{F}$ holds for $\beta = 0$.

Regarding inequality constraints, two cases are distinguished. Let $x \in \mathcal{F}$ and for a j, with $m_e < j \leq m$, the equation $g_j(x) = 0$ holds, then this constraint $g_j(x)$ is called an *active constraint* at x. Correspondingly, a constraint $g_j(x)$, with $m_e < j \leq m$, is called *inactive* at x if $g_j(x) > 0$ holds. The index set of active inequality constraints with respect to $x \in \mathcal{F}$ is defined as

$$\mathcal{A}(x) := \{j \in \mathcal{I} \mid \; g_j(x) = 0\} \;. \tag{2.13}$$

The set of inactive inequality constraints is therefore defined as

$$\mathcal{B}(x) := \mathcal{I} \setminus \mathcal{A}(x) \;. \tag{2.14}$$

The concept of active constraints is extended to nearly active constraints subject to $\gamma \in \mathbb{R}_0^+$, and the set of nearly active constraints is defined as

$$\mathcal{A}(x, \gamma) := \{j \in \mathcal{I} \mid \; g_j(x) \leq \gamma\} \;. \tag{2.15}$$

This set is also defined for infeasible points, as $\mathcal{A}(x, \gamma)$ also contains all $j \in \mathcal{I}$ with

$g_j(x) < 0$. The equation $\mathcal{A}(x) = \mathcal{A}(x, 0)$, with $\gamma = 0$, holds for all $x \in \mathcal{F}$. The complement to $\mathcal{A}(x, \gamma)$ is defined as

$$\mathcal{B}(x, \gamma) := \mathcal{I} \setminus \mathcal{A}(x, \gamma) \ . \tag{2.16}$$

Generally, a subscript k is written to highlight dependency on a specific iteration k. However, to improve readability a superscript (k) is used in case an entry of a vector is considered, e.g., $x_j^{(k)}$ denotes the j-th entry of iterate x_k. In the remainder of this thesis the notation $\mathcal{A}_k$ and $\mathcal{B}_k$ for the sets $\mathcal{A}(x_k, 0)$ and $\mathcal{B}(x_k, 0)$ at iterate x_k is used.

The introduced notation can be adapted to problems with additional integer variables in a straightforward way. All concepts also exist in mixed-integer optimization.

2.2 Optimality Conditions

Regarding a minimizer of problem (2.1), a distinction is made between a local and a global solution. A feasible point $x^\star \in \mathcal{F}$ is called a *global minimum* of problem (2.1) if

$$f(x^\star) \leq f(x) \ , \quad \text{for all } x \in \mathcal{F} \ . \tag{2.17}$$

In case inequality (2.17) only holds in a neighborhood of $x^\star$, then this is called a *local minimum* or *local minimizer*. An $x^\star \in \mathcal{F}$ is a *local minimum* of problem (2.1) if there exists a neighborhood $\mathcal{N}_\epsilon(x^\star)$ of $x^\star$ such that

$$f(x^\star) \leq f(x) \ , \ \text{ for all } x \in \mathcal{F} \cap \mathcal{N}_\epsilon(x^\star) \ , \tag{2.18}$$

where the ϵ-neighborhood $\mathcal{N}_\epsilon(x^\star)$ is defined as

$$\mathcal{N}_\epsilon(x^\star) := \{x \in \mathbb{R}^n \mid \ \|x - x^\star\| < \epsilon\} \ , \tag{2.19}$$

with $\epsilon > 0$. Here, $\|.\|$ denotes an arbitrary norm.

Assuming continuous functions in (2.1) allows the formulation of additional optimality conditions. A key role plays the *Lagrangian function* of (2.1), which is defined as

$$L(x, u) := f(x) - g(x)^T u \ , \tag{2.20}$$

where $x \in \mathbb{R}^n$ and $u \in \mathbb{R}^m$. The vector $u = (u_1, \ldots, u_m)^T$ contains the *Lagrangian multipliers* u_j, $j = 1, \ldots, m$, of problem (2.1). The vector u is also called the Lagrangian multipliers, Lagrange multipliers, or the *dual variables*.

In case some regularity assumptions are satisfied, necessary optimality conditions can be stated that use the Lagrangian function (2.20). A common but may be very restrictive regularity assumption is the so-called *linear independence constraint qualification* (LICQ). The LICQ holds at a feasible point x if the gradients of the equality constraints and the active constraints at x are linearly independent, i.e., $\nabla g_j(x)$, $j \in \mathcal{E} \cup \mathcal{A}(x)$, are linearly independent.

Definition 2.1 (LICQ) *If active constraint gradients $\nabla g_j(x)$, $j \in \mathcal{E} \cup \mathcal{A}(x)$, are linearly independent at $x \in \mathcal{F}$, then the linear independence constraint qualification (LICQ) holds at x.*

Also commonly used is the *Mangasarian-Fromowitz constraint qualification* (MFCQ). The MFCQ is weaker than the LICQ, i.e., the LICQ implies the MFCQ.

Definition 2.2 (MFCQ) *Let $x \in \mathcal{F}$. If the gradients $\nabla g_j(x)$, $j \in \mathcal{E}$, are linearly independent and there exists a $d \in \mathbb{R}^n$ such that*

$$\begin{aligned} \nabla g_j(x)^T d &= 0 \ , \quad j \in \mathcal{E} \ , \\ \nabla g_j(x)^T d &> 0 \ , \quad j \in \mathcal{A}(x) \ , \end{aligned} \tag{2.21}$$

then the Mangasarian-Fromowitz constraint qualification (MFCQ) holds at x.

In Chapter 4, the convergence analysis of the new continuous algorithm applies a slightly modified version of the Mangasarian-Fromowitz constraint qualification. The MFCQ is extended to the infeasible region and the extended feasible region $\mathcal{F}(\beta)$, as defined in (2.12), is considered.

Definition 2.3 (extended MFCQ) *Let $x \in \mathcal{F}(\beta)$ with $\beta \geq 0$. If the gradients $\nabla g_j(x)$, $j \in \mathcal{E}$, are linearly independent and there exists a $d \in \mathbb{R}^n$ such that*

$$\begin{aligned} \nabla g_j(x)^T d &= 0 \ , \quad j \in \mathcal{E} \ , \\ \nabla g_j(x)^T d &> 0 \ , \quad j \in \mathcal{A}(x, 0) \ , \end{aligned} \tag{2.22}$$

then the extended Mangasarian-Fromowitz constraint qualification (extended MFCQ) holds at x.

First order necessary conditions are stated now that employ the aforementioned constraint qualifications. Note that the statement of the following theorem remains valid if other regularity assumptions hold instead of the LICQ or MFCQ.

Theorem 2.4 (First order necessary conditions) *Let $x^\star \in \mathbb{R}^n$ be a local minimizer of problem (2.1) and the LICQ or MFCQ holds at $x^\star$, then there exist Lagrange multipliers $u^\star \in \mathbb{R}^m$ such that*

$$\nabla f(x^\star) - \sum_{j=1}^{m} u_j^\star \nabla g_j(x^\star) = 0 \ , \tag{2.23}$$

$$g_j(x^\star) = 0 \ , \quad j \in \mathcal{E} \ , \tag{2.24}$$

$$g_j(x^\star) \geq 0 \ , \quad j \in \mathcal{I} \ , \tag{2.25}$$

$$u_j^\star \geq 0 \ , \quad j \in \mathcal{I} \ , \tag{2.26}$$

$$u_j^\star g_j(x^\star) = 0 \ , \quad j \in \mathcal{I} \ , \tag{2.27}$$

hold. □

Conditions (2.23)-(2.27) are called the *Karush-Kuhn-Tucker* (KKT) optimality conditions. If the KKT conditions hold at a point $x^\star$, then $x^\star$ is called a *KKT point* or as well a *stationary point.*

As all constraints of a quadratic programming problem are linear, the KKT conditions (2.23)-(2.27) hold at the minimizer of the QP without the additional requirement of the LICQ or MFCQ, see, e.g., Geiger and Kanzow [47].

If the objective function $f(x)$ and the constraints $g_j(x)$, $j = 1, \ldots, m$, are twice continuously differentiable, then second order necessary conditions can be stated. Let $x^\star \in \mathbb{R}^n$ be a local minimizer of problem (2.1) and a constraint qualification holds, then the Hessian of the Lagrangian function (2.20) at $x^\star$ is positive semidefinite for all vectors in the null space of the Jacobian of the active constraints .

Theorem 2.5 (Second order necessary conditions) *Let $x^\star \in \mathbb{R}^n$ be a local minimizer of problem (2.1), $f(x)$ and $g_j(x)$, $j = 1, \ldots, m$, be twice continuously differentiable and the LICQ holds at $x^\star$. Then there exist Lagrange multipliers $u^\star \in \mathbb{R}^m$ such that $(x^\star, u^\star)$ satisfies the KKT conditions (2.23)-(2.27), and*

$$d^T \nabla^2_{xx} L(x^\star, u^\star)\, d \geq 0 \tag{2.28}$$

holds for all $d \in \mathbb{R}^n$ with

$$\nabla g_j(x^\star)^T d = 0\ , \quad j \in \mathcal{E}\ , \tag{2.29}$$

$$\nabla g_j(x^\star)^T d = 0\ , \quad j \in \mathcal{A}(x^\star)\ \textit{and}\ u_j^\star > 0\ , \tag{2.30}$$

$$\nabla g_j(x^\star)^T d \geq 0\ , \quad j \in \mathcal{A}(x^\star)\ \textit{and}\ u_j^\star = 0\ . \tag{2.31}$$

□

Sufficient conditions for $x^\star$ to be an isolated minimizer of problem (2.1) are formulated as follows.

Theorem 2.6 (Second order sufficient conditions) *Let $f(x)$ and $g_j(x)$, $j = 1, \ldots, m$, be twice continuously differentiable and the LICQ holds at $x^\star \in \mathbb{R}^n$. Let $x^\star$ and $u^\star \in \mathbb{R}^m$ be given such that $(x^\star, u^\star)$ satisfies the KKT conditions (2.23)-(2.27), and for all $d \in \mathbb{R}^n$ with $d \neq 0$ and*

$$\nabla g_j(x^\star)^T d = 0\ , \quad j \in \mathcal{E}\ , \tag{2.32}$$

$$\nabla g_j(x^\star)^T d = 0\ , \quad j \in \mathcal{A}(x^\star)\ \textit{and}\ u_j^\star > 0\ , \tag{2.33}$$

$$\nabla g_j(x^\star)^T d \geq 0\ , \quad j \in \mathcal{A}(x^\star)\ \textit{and}\ u_j^\star = 0\ , \tag{2.34}$$

holds

$$d^T \nabla^2_{xx} L(x^\star, u^\star)\, d > 0\ . \tag{2.35}$$

Then $x^\star$ is an isolated local minimizer of NLP (2.1). □

For further information on optimality conditions it is referred, for example, to Spellucci [110], Fletcher [40], Sun and Yuan [113], and Conn, Gould, and Toint [21].

2.3 Convergence Properties

Convergence properties play a major role within the theoretical analysis of an algorithm. A distinction is made between a global convergence analysis and a local one. The global convergence analysis investigates the behavior of an algorithm started at any arbitrary point. An algorithm should converge from any starting point to a stationary point, i.e., a point that satisfies the aforementioned KKT optimality conditions of the optimized problem.

The local convergence analysis concentrates on the behavior of the generated sequence of iterates when the iterates are already close to a stationary point. The efficiency of an algorithm is evaluated by measuring the convergence rate, i.e., how fast the iteration sequence converges toward a stationary point $x^\star$. Two important convergence rates are stated subject to an arbitrary norm $\|.\|$.

A sequence $\{x_k\}$ is said to converge *superlinearly* toward $x^\star$ if

$$\lim_{k\to\infty} \frac{\|x_{k+1} - x^\star\|}{\|x_k - x^\star\|} = 0 \tag{2.36}$$

holds. This is also called the *Q-superlinear* rate of convergence. Later, it is established that under suitable assumptions the new continuous algorithm can converge locally with superlinear rate.

A faster convergence is obtained in case the following condition holds. A sequence $\{x_k\}$ is said to converge *quadratically* toward $x^\star$ if a positive constant $\nu > 0$ exists such that

$$\lim_{k\to\infty} \frac{\|x_{k+1} - x^\star\|}{\|x_k - x^\star\|^2} = \nu \tag{2.37}$$

holds. For example, it can be shown that Newton's method converges quadratically under suitable assumptions. This rate of convergence is also named *Q-quadratic* convergence.

Further information on convergence rates, e.g., the R-superlinear and R-quadratic convergence rates, can be found in Ortega and Rheinboldt [81], and Conn, Gould, and Toint [21].

2.4 Difficulties in Mixed-Integer Optimization

The difficulties in mixed-integer nonlinear optimization arise due to the combinatorial nature of the problems. The combination of the continuous domain and the discrete domain makes the problems very complex. The number of discrete variables influences the complexity significantly, as the number of possible combinations increases exponentially.

A major difficulty concerns the evaluation of the quality or optimality, respectively, of a given point with a specific discrete configuration. In continuous optimization the common way is to check the optimality conditions outlined before in Section 2.2. But

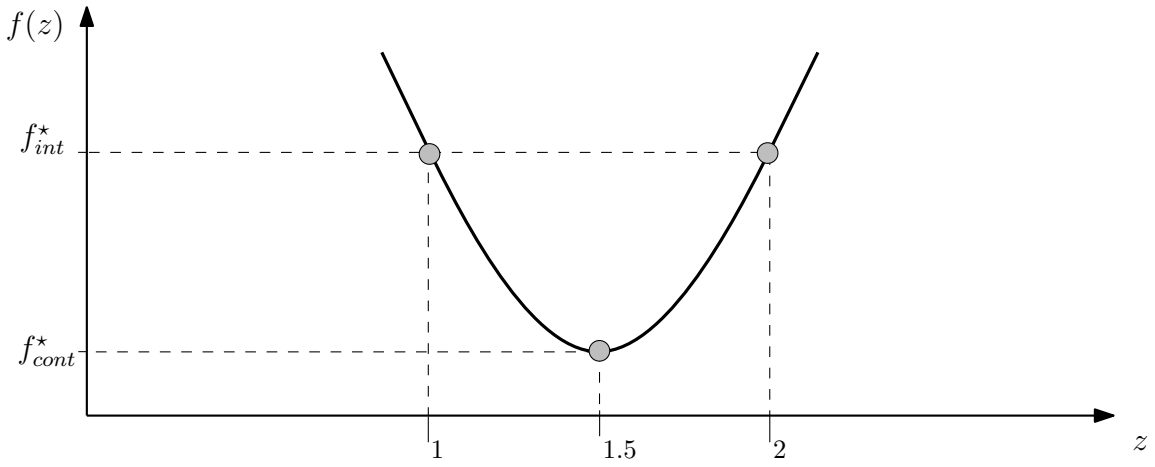

Figure 2.1: Mixed-Integer vs. Continuous Optimization

these conditions do not hold for mixed-integer optimization. This is demonstrated by an example shown in Figure 2.1. The considered function f is convex. The continuous minimizer, i.e., the domain of z is $\mathbb{R}$, is obtained at $z^\star_{cont} = 1.5$ with an objective function value $f^\star_{cont}$. According to the theory of continuous nonlinear optimization, this minimizer is unique, see, e.g., Conn et al.[21]. Adding the integer requirement changes the situation. Although the function is convex there exist two integer values $z^\star_{int_1} = 1$ and $z^\star_{int_2} = 2$ with the optimal objective function value $f^\star_{int}$. Thus, the minimizer is not necessarily unique for convex mixed-integer nonlinear optimization problems.

Applying the first order necessary conditions for the continuous problem, cf. Theorem 2.4, the derivative at the continuous minimizer $z^\star_{cont} = 1.5$ satisfies the KKT condition (2.23), that is the equation

$$\frac{\partial}{\partial z} f(z^\star_{cont}) = 0 \tag{2.38}$$

holds. However, this does not hold for the derivatives at the mixed-integer minimizers $z^\star_{int_1}$ and $z^\star_{int_2}$. Obviously, at these points the KKT condition (2.23) is not fulfilled as

$$\frac{\partial}{\partial z} f(z^\star_{int_1}) \neq 0 \tag{2.39}$$

and

$$\frac{\partial}{\partial z} f(z^\star_{int_2}) \neq 0 \ . \tag{2.40}$$

Further discussions on arising challenges in mixed-integer optimization can be found, for example, in Leyffer [67] and Floudas [44].

3 Sequential Quadratic Programming Methods

In this chapter a review of existing sequential quadratic programming (SQP) methods and globalization strategies in the context of nonlinear programs of form (1.2) is given. In the beginning SQP methods are motivated and the idea is described. As the basic procedure does not converge for any arbitrary starting point, globalization strategies have to be applied. In order to obtain convergence of an SQP method, a step from one iterate to another has to fulfill some necessary conditions to be taken.

Section 3.2 introduces techniques that are applied to measure the obtained progress toward a solution. Merit functions are introduced and the concept of a filter is explained. If a calculated step of an SQP method does not reduce these measurements sufficiently, then the step is rejected and a new trial step has to be determined.

Techniques which generate trial steps that fulfill the necessary conditions are presented in the following sections. In Section 3.3 the line search approach is discussed. Section 3.4 presents the basic ideas of trust region methods. Moreover, existent trust region algorithms addressing the nonlinear problem (1.2) are stated in the remainder of this chapter.

3.1 Foundations of Sequential Quadratic Programming

Problems of form (1.2) with small and medium size can be efficiently solved by SQP methods if all problem functions $f(x)$ and $g_j(x)$, $j = 1, \ldots, m$, are at least twice continuously differentiable on the whole $\mathbb{R}^n$. This approach was proposed for the first time in the 1960's by Wilson [126]. In the 1970's SQP methods spread due to the outstanding publications by Han [58, 59] and Powell [86] and not least because of the numerical performance of corresponding codes. In those days, the implemented algorithms outperformed other approaches significantly. For an extensive comparison see, for example, Schittkowski [98].

The basic idea of SQP methods can be summarized as follows. A solution of the considered problem is approximated iteratively by solving quadratic programming problems in each iteration k. The problem in iteration $k = 0, 1, \ldots$, is of the following form

$$\begin{aligned} \underset{d \in \mathbb{R}^n}{\text{minimize}} \quad & \nabla f(x_k)^T d + \frac{1}{2} d^T B_k d \\ \text{subject to} \quad & g_j(x_k) + \nabla g_j(x_k)^T d = 0 \,, \quad j = 1, \ldots, m_e \,, \\ & g_j(x_k) + \nabla g_j(x_k)^T d \geq 0 \,, \quad j = m_e + 1, \ldots, m \,, \end{aligned} \tag{3.1}$$

where x_k is the current approximation to the solution. The constraints of the underlying problem are linearized and the Lagrangian function $L(x_k, v_k)$, see (2.20), is

approximated quadratically, where v_k is the current approximation to the optimal Langrange multipliers. The symmetric matrix $B_k \in \mathbb{R}^{n\times n}$ is an approximation to the Hessian of the Lagrangian function. Let d_k be the optimal solution of subproblem (3.1) and u_k the corresponding multipliers, then the next iterate is set to $x_{k+1} := x_k + d_k$ and $v_{k+1} := u_k$.

The basic procedure, as just described, is motivated by an observation that can be made in case the considered problem (1.2) only contains equality constraints, i.e., $m = m_e$. In the following the similarity of SQP methods to Newton's method is demonstrated. Thus, the equality constrained problem

$$\begin{array}{ll} \underset{x\in\mathbb{R}^n}{\text{minimize}} & f(x) \\ \text{subject to} & g_j(x) = 0\ , \quad j = 1,\ldots,m\ , \end{array} \tag{3.2}$$

is considered now.

Let $x^\star$ be a minimizer of problem (3.2) and $u^\star$ be the corresponding multipliers according to the Karush-Kuhn-Tucker optimality conditions, see (2.23) and (2.24). Then $(x^\star, u^\star)$ is a solution to

$$\begin{pmatrix} \nabla f(x) - \nabla g(x)u \\ g(x) \end{pmatrix} = 0\ . \tag{3.3}$$

As equation (3.3) denotes a system of $n+m$ nonlinear equations, Newton's method can be applied to approximate a solution. To simplify the notation, the left-hand side of equation (3.3) is redefined as

$$\Gamma(x,u) := \begin{pmatrix} \nabla f(x) - \nabla g(x)u \\ g(x) \end{pmatrix}\ . \tag{3.4}$$

Applying Newton's method to the rewritten formulation of system (3.3), where definition (3.4) is used, yields the following iteration step. Let (x_k, v_k) be the current approximation to the solution $(x^\star, u^\star)$, then Newton's method determines the next iterate by $x_{k+1} := x_k + d_k$ and $v_{k+1} := v_k + w_k$, where d_k and w_k solve the system

$$\nabla\Gamma(x_k, v_k)\begin{pmatrix} d_k \\ w_k \end{pmatrix} + \Gamma(x_k, v_k) = 0\ . \tag{3.5}$$

Now $\Gamma(x_k, v_k)$ is substituted again by the right-hand side of definition (3.4), then (3.5) can be stated as

$$\begin{pmatrix} B_k & -\nabla g(x_k) \\ \nabla g(x_k)^T & 0 \end{pmatrix}\begin{pmatrix} d_k \\ w_k \end{pmatrix} + \begin{pmatrix} \nabla f(x_k) - \nabla g(x_k)v_k \\ g(x_k) \end{pmatrix} = 0\ , \tag{3.6}$$

where $B_k := \nabla^2_{xx} L(x_k, v_k)$. By setting $u_k := v_k + w_k$, (3.6) can be rewritten as

$$B_k d_k - \nabla g(x_k)u_k + \nabla f(x_k) = 0 \tag{3.7}$$

and

$$g(x_k) + \nabla g(x_k)^T d_k = 0 \ . \tag{3.8}$$

Equations (3.7) and (3.8) are also the optimality conditions of the equality constrained quadratic problem (3.1) with $m_e = m$.

It can be concluded that the basic sequential quadratic programming method outlined before is identical to Newton's method when applied to the Karush-Kuhn-Tucker optimality conditions of problem (3.2). This statement holds under the condition that the exact Hessian of the Lagrangian function is used in the subproblems of the SQP method. Consequently, the SQP method shares the advantages and disadvantages of Newton's method. The fast local quadratic convergence of Newton's method is retained in case the starting point is sufficiently close to the solution. However, Newton's method will not converge for arbitrary starting points.

The subsequent sections show how SQP methods can be forced to converge for any arbitrary starting point $x_0 \in \mathbb{R}^n$, $v_0 \in \mathbb{R}^m$. Convergence can only be guaranteed if the steps taken from one iterate to the next satisfy some necessary conditions. For this reason the quality of the steps obtained by an SQP method has to be evaluated. A trial step is accepted if a sufficient improved is obtained. Otherwise, the step is rejected and a new trial step has to be computed. There are several approaches to stabilize SQP methods. They differ in the acceptance test and the way of generating new trial points. In the following, some techniques are discussed, e.g., penalty functions, filter, line search methods, and trust region methods.

A lot of research has been done on the theoretical background of SQP methods. Several theorems concerning the local and global convergence properties are established. For reviews see, for example, Schittkowski and Yuan [107], Boggs and Tolle [7] and Gould and Toint [52]. Detailed descriptions of SQP methods are also presented in Fletcher [40], Gill, Murray, and Wright [49], Stoer [112], Spellucci [110], and Sun and Yuan [113]. A more practical view on SQP methods is given by Papalambros and Wilde [82] in the context of optimal design, or by Edgar and Himmelblau [29] coming from chemical engineering.

3.2 Measuring Progress

In unconstrained optimization the progress toward a solution of the underlying problem is measured by evaluating the objective function. A step d_k is accepted only if $f(x_k + d_k)$ is sufficiently less than $f(x_k)$. This is straightforward to see. In case additional constraints are introduced the progress has to be measured differently. Two goals of the optimization have to be achieved that might even conflict. On the one hand, the objective function should be reduced. On the other hand, feasibility of the obtained solution has to be guaranteed. The question how to combine these conflicting goals leads to the following sections. Two strategies are discussed, measuring progress subject to a merit function and subject to a filter.

3.2.1 Merit Functions

Merit functions combine the objective function and the feasibility measurement in one function. Thus, the constrained problem (1.2) is transformed into an unconstrained problem. In the early days of nonlinear programming unconstrained optimization techniques were employed to minimize these merit functions, as other methods for constrained problems were not available. Some commonly used merit functions are presented in this section. For more details on merit functions and the results presented below it is referred to, e.g., Geiger and Kanzow [47], and Conn, Gould, and Toint [21].

Merit functions of the following form are called *penalty functions*, they are defined as

$$P(x) := f(x) + \sigma \|g(x)^-\| \ , \tag{3.9}$$

where $\sigma > 0$ is a penalty parameter and $\|.\|$ denotes an arbitrary norm. The applied norm $\|.\|$ depends on the specific penalty function under consideration. A penalty function $P(x)$ of form (3.9) is said to be exact at a local minimum $x^\star$ of the underlying problem (1.2) if there exists a finite parameter $\overline{\sigma} > 0$ such that $x^\star$ is also a local minimizer of $P(x)$ for all $\sigma \geq \overline{\sigma}$.

A penalty function where this bound $\overline{\sigma}$ does not exist is the least squares penalty function

$$P_{ls}(x) := f(x) + \sigma_{ls} \|g(x)^-\|_2^2 \ , \tag{3.10}$$

where $\sigma_{ls} > 0$ is a positive penalty parameter. $P_{ls}(x)$ is differentiable but unfortunately not an exact penalty function. If $x^\star$ is a solution to problem (1.2), then $x^\star$ is a minimizer of $P_{ls}(x)$ only if σ_{ls} tends to infinity. A method that uses $P_{ls}(x)$ as a merit function will probably suffer numerical difficulties due to the unbounded penalty parameter σ_{ls}.

In the context of SQP methods the L_1-penalty function was the first penalty function studied, see Han [59]. The L_1-penalty function is formulated as follows

$$P_1(x) := f(x) + \sigma_1 \|g(x)^-\|_1 \ , \tag{3.11}$$

with a positive penalty parameter $\sigma_1 > 0$. $P_1(x)$ is an exact penalty function, i.e., if $x^\star$ is a minimizer of problem (1.2), $u^\star$ the corresponding multiplier, and the penalty parameter satisfies $\sigma_1 \geq \|u^\star\|_\infty$, then $x^\star$ is also a minimum of $P_1(x)$.

Another exact penalty function is the L_∞-penalty function that is defined as

$$P_\infty(x) := f(x) + \sigma_\infty \|g(x)^-\|_\infty \ , \tag{3.12}$$

with a positive penalty parameter $\sigma_\infty > 0$. If $x^\star$ is a minimizer of (1.2) with a corresponding multiplier vector $u^\star$ and the penalty parameter σ_∞ is greater or equal to $\|u^\star\|_1$, then $x^\star$ is also a minimizer of $P_\infty(x)$. For example, the algorithm of Yuan [130], which is the underlying algorithm for the mixed-integer algorithms presented in Chapter 5, uses the L_∞-penalty function,

The penalty functions (3.11) and (3.12) are exact at a minimizer $x^\star$ of problem (1.2), but they are not differentiable at $x^\star$. This property might lead to undesirable behav-

ior of the underlying optimization algorithm. In this context the so-called Maratos effect [71] has to be mentioned. By employing a merit functions which is not differentiable, SQP methods can lose the nice fast local convergence properties and slow down. The following example by Powell [91] illustrates the Maratos effect.

Example 3.1 Consider the problem

$$\begin{aligned} \underset{x\in\mathbb{R}^2}{\text{minimize}} \quad & 2\,(x_1^2 + x_2^2 - 1) \;-\; x_1 \\ \text{subject to} \quad & x_1^2 + x_2^2 - 1 = 0 \;. \end{aligned}$$

The objective function be denoted by $f(x)$ and $g(x)$ denotes the constraint. The solution is $x^\star = (1,0)^T$ and the corresponding Lagrangian multiplier is $u^\star = 3/2$. Let x_k be the current approximation to the primal solution $x^\star$. Then the trial step d_k in iteration k is obtained by solving the quadratic subproblem

$$\begin{aligned} \underset{d\in\mathbb{R}^n}{\text{minimize}} \quad & \nabla f(x_k)^T d + \frac{1}{2} d^T d \\ \text{subject to} \quad & g(x_k) + \nabla g(x_k)^T d = 0 \;, \end{aligned} \tag{3.13}$$

where the matrix B_k of the SQP subproblem is set to the Hessian of the Lagrangian at $(x_k, u^\star)$, that is the identity matrix I. Thus, the trial step will be identical to the one obtained by Newton's method.

The solution to problem (3.13) be denoted by (d_k, u_k). It solves the corresponding KKT system, cf. (2.23) and (2.24),

$$\begin{aligned} d_k + \nabla f(x_k) - \nabla g(x_k) u_k &= 0 \;, \\ g(x_k) + \nabla g(x_k)^T d_k &= 0 \;. \end{aligned} \tag{3.14}$$

As $f(x)$ and $g(x)$ are quadratic functions, Taylor expansion yields

$$\begin{aligned} f(x_k + d_k) &= f(x_k) + \nabla f(x_k)^T d_k + 2 d_k^T d_k \;, \\ g(x_k + d_k) &= g(x_k) + \nabla g(x_k)^T d_k + d_k^T d_k \;, \end{aligned} \tag{3.15}$$

with $\nabla^2 f(x_k) = 4I$ and $\nabla^2 g(x_k) = 2I$ inserted. Applying (3.14) to (3.15),

$$\begin{aligned} f(x_k + d_k) &= f(x_k) - u_k g(x_k) + d_k^T d_k \;, \\ g(x_k + d_k) &= d_k^T d_k \end{aligned}$$

is obtained. For any point x_k with $g(x_k) = 0$, except $x^\star$ and $-x^\star$, a solution $d_k \neq 0$ is obtained. This results in

$$f(x_k + d_k) > f(x_k)$$

and

$$g(x_k + d_k) > g(x_k) = 0 \;.$$

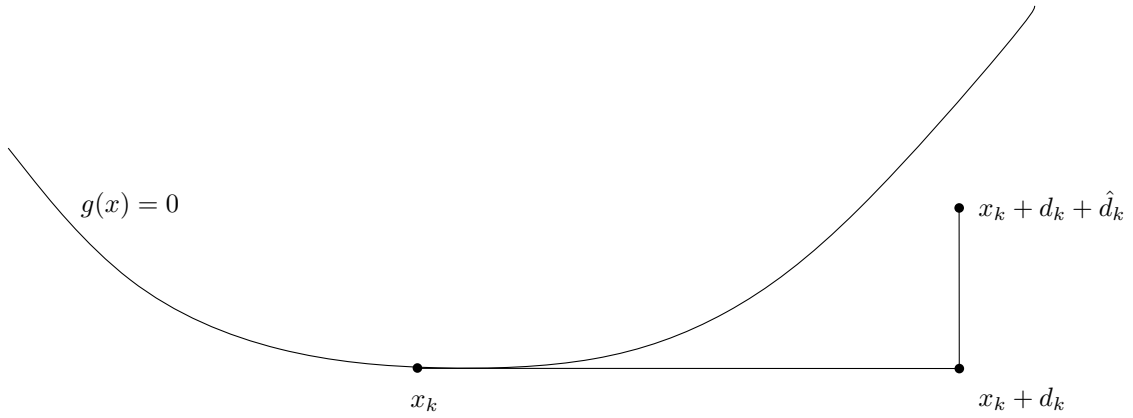

Figure 3.1: A Second Order Correction Step

Penalty functions of form (3.9), which only combine $f(x)$ and constraint violation $\|g(x)^-\|$, will reject the step d_k as

$$P(x_k + d_k) > P(x_k) \ .$$

This holds for any feasible point that is arbitrary close to the solution $x^\star$. Without any safeguards fast local convergence will not take place for an SQP method. □

The mentioned problem occurs as the constraints are only linearized in the quadratic subproblem and second order information is contained merely in the Hessian approximation of the Lagrangian function. This lack of second order information with respect to the constraints might lead to a rejection of the calculated step. There are different strategies to overcome this drawback, such as applying second order correction steps, non-monotone strategies for penalty functions or a differentiable merit function.

Adding second order correction steps was proposed by several authors, see, e.g., Fletcher [39] and Yuan [129] for details and convergence analysis. Fletcher [39] has shown that the SOC steps circumvent the Maratos effect. Mayne and Polak [73], Yuan [130], and Fukushima [45] also apply SOC steps. The use of SOC steps was motivated by feasible direction methods. Figure 3.1 illustrates how the second order correction step $\hat{d}_k$ reduces the infeasibility in the constraints that results from taking step d_k, where d_k denotes the solution of the quadratic problem (3.1).

The watch-dog technique by Chamberlain et al. [18] can also avoid the Maratos effect. Here some steps are allowed to increase the used merit function. A similar idea is used by non-monotone techniques, see, for example, Gould and Toint [53]. The basic idea of non-monotone strategies goes back to Grippo, Lampariello, and Lucidi [55], and was extended to constrained optimization and trust region methods in a series of subsequent papers, see, e.g., Toint [119, 120], and Ulbrich and Ulbrich [122]. Here the requirement that $P(x_k + d_k)$ has to be sufficiently less than $P(x_k)$ is relaxed, and a non-monotone sequence of $P(x_k)$ is accepted.

Instead of penalty functions of form (3.9), augmented Lagrangian merit functions, defined as

$$\Phi_{\sigma_a}(x, v) := f(x) - v^T g(x) + \frac{1}{2} \sum_{j=1}^{m} \sigma_a \Big(g_j(x)^- \Big)^2 \,, \tag{3.16}$$

where σ_a is a positive penalty parameter, are an appropriate alternative. The dual variables are included in function $\Phi_{\sigma_a}(x, v)$. There exist different strategies for the choice of the multiplier approximation, for details see, for instance, Boggs and Tolle [7], and Gill, Murray, Saunders, and Wright [48]. The augmented Lagrangian is differentiable at a minimizer of the underlying optimization problem and the Maratos effect can be avoided.

Using smooth merit functions, such as an augmented Lagrangian function, to achieve fast local convergence without additional safeguards was proposed by Schittkowski [100] for a line search method, and Powell and Yuan [92, 93] for a trust region method. Ulbrich [121] extended the filter approach, see below for a description, by techniques of differentiable merit functions to retain fast local convergence without further safeguards.

Rockafellar [95] suggested a slightly different augmented Lagrangian function that is stated as

$$\Phi_{\sigma}(x, v) := f(x) - \sum_{j \in \mathcal{S}} \left(v_j g_j(x) - \frac{1}{2} \sigma_j g_j(x)^2 \right) - \frac{1}{2} \sum_{j \in \overline{\mathcal{S}}} \frac{v_j^2}{\sigma_j} \,, \tag{3.17}$$

with $\mathcal{S} := \mathcal{E} \cup \{ j \in \mathcal{I} \mid \; g_j(x) \leq v_j / \sigma_j \}$ and $\overline{\mathcal{S}} := \{1, \ldots, m\} \setminus \mathcal{S}$. All entries of the penalty vector $\sigma \in \mathbb{R}^m$ are nonnegative. The augmented Lagrangian (3.17) is also employed by Schittkowski [100] and the new continuous algorithm introduced later.

3.2.2 Filter

The concept of a filter was introduced by Fletcher and Leyffer [42]. A filter does not require a penalty parameter. This is an advantage compared to penalty function such as functions (3.11) and (3.12). As mentioned before, these penalty functions are exact only if the penalty parameter satisfies certain conditions. In general, the required penalty parameter value is not known a priori. Therefore, methods that employ penalty functions have to update the penalty parameter, but this is a crucial part in penalty functions and the performance of the algorithms often depends on the updating procedure. The idea of a filter is to treat the constrained optimization problem (1.2) as two separate minimization problems stated as

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \quad f(x) \tag{3.18}$$

and

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \quad h\Big(g(x)^- \Big) \quad , \tag{3.19}$$

where $h(.)$ is a positive function that measures infeasibility of constraints and $h(0) = 0$. This kind of problem formulation can be seen as a multi-criteria or multi-objective

optimization problem. Fletcher, Leyffer, and Toint [43] suggested to use the L_1-norm for measuring restriction violation, i.e.,

$$h\Big(g(x)^-\Big) := \|g(x)^-\|_1 \ . \tag{3.20}$$

The measured constraint violation and the objective function are combined in the following tuple

$$(h_k, f_k) := \Big(h\Big(g(x_k)^-\Big),\ f(x_k)\Big) \ , \tag{3.21}$$

which is defined for each iterate x_k. In iteration k the filter F_k consists of pairs of form (3.21) that correspond to certain previous iterates, i.e.,

$$F_k \subset \{(h_j, f_j) \mid \ j = 0, 1, 2, \ldots, k-1\} \ . \tag{3.22}$$

Global convergence is obtained by evaluating the quality of a trial step d_k subject to filter F_k. In the first versions of filter methods, see Fletcher and Leyffer [42], a step d_k is rejected if the corresponding pair (h_k, f_k) is *dominated* by a pair $(h_j, f_j) \in F_k$, that is

$$h_k \geq h_j \tag{3.23}$$

and

$$f_k \geq f_j \ , \tag{3.24}$$

for a $(h_j, f_j) \in F_k$.

SQP methods that use a filter instead of a merit function can also suffer the Maratos effect, at least if no additional safeguards are added. For further details on the concept of a filter the original papers of Fletcher and Leyffer [42], and Fletcher, Leyffer, and Toint [43] are recommended. A review is also given in the textbook by Conn, Gould, and Toint [21].

3.3 Line Search Methods

Line search methods are frequently employed to ensure global convergence of SQP methods. They are used to determine steps that lead to a sufficient improvement with respect to a specific performance criterion, e.g., a merit function or a filter. The basic idea is summarized in the following.

In each iteration k, line search methods calculate an adequate search direction s_k at first. Now s_k denotes the solution of the quadratic program (3.1). Then a trial step lying on this line is determined. The obtained step $\alpha_k s_k$, where the scalar parameter α_k adjusts the step length, guarantees sufficient progress, and a new iterate is obtained by setting

$$x_{k+1} := x_k + \alpha_k s_k \ . \tag{3.25}$$

The calculated search direction s_k has to be a descent direction with respect to the

used merit function Φ_{σ_k}, with a positive penalty parameter σ_k, that is at least

$$\nabla\Phi_{\sigma_k}(x_k)^T s_k < 0 \ . \tag{3.26}$$

Necessary conditions on the decrease achieved by $\nabla\Phi_{\sigma_k}(x_k)^T s_k$ were investigated, for example, by Wolfe [128]. Schittkowski [100] performs a line search subject to the augmented Lagrangian (3.17). He proposes an update rule for the penalty parameter σ_k so that a stronger condition than (3.26) is satisfied.

The step length parameter α_k has to be chosen adequately so that a certain sufficient descent condition for the merit function Φ_{σ_k} holds. By requiring sufficient descent the convergence toward a stationary point of the underlying problem can be achieved. According to Armijo [2] the parameter α_k can be chosen so that

$$\Phi_{\sigma_k}(x_k + \alpha_k s_k) < \Phi_{\sigma_k}(x_k) + \rho\, \alpha_k \nabla\Phi_{\sigma_k}(x_k)^T s_k \tag{3.27}$$

is satisfied, where $0 < \rho < 1/2$ is constant.

A crucial point in line search algorithms is the determination of the step length α_k. On the one hand, the parameter α_k has to be calculated sufficiently accurate to obtain convergence. On the other hand, the number of function evaluations needed during the search for α_k should be as small as possible. There are different procedures to determine α_k, for more details see, e.g., Powell [86] and Schittkowski [100]. Line search methods, as Armijo line search, are also discussed, for example, in Dennis and Schnabel [26], Nocedal and Wright [77], and Geiger and Kanzow [47].

3.4 Trust Region Methods

Another globalization technique for nonlinear optimization methods is the trust region approach. Trust region methods differ from line search methods as they determine the search direction and step size simultaneously. This is achieved by directly restricting the step size in the subproblem.

The concepts of trust region methods go back to the field of nonlinear least-squares optimization and parameter estimation. It was the first time that the step size was controlled inside the subproblem itself, see Levenberg [66]. The steps were damped by adding multiples of the identity matrix to the Hessian matrix. A similar technique was further investigated by Morrison [75] and Marquardt [72].

In the beginning trust region methods were applied to unconstrained optimization, see, e.g., Powell [85] for minimizing nonlinear objective functions. Solving nonlinear equations by trust region techniques was also done by Powell [83, 84]. The proposed methods for unconstrained optimization seemed to be preferable compared to other approaches at that time. In the following years more research was done on trust region methods, see, for example, Winfield [127], Fletcher [40], Toint [118], Powell [90], and Sorensen [109]. The terminology of trust region methods goes back to Dennis [23] and Moré [74]. A historical review of trust region methods can be found in the textbook

by Conn, Gould, and Toint [21].

Later trust region methods were adapted to constrained optimization. The first algorithms only considered equality constrained problems, i.e., $m_e = m$, see for example the algorithms developed by Dennis, El-Alem, and Maciel [24], Celis, Dennis, and Tapia [17], El-Alem [30, 31], Powell and Yuan [93], Byrd, Schnabel, and Shultz [16], Vardi [123], and Omojokun [80]. Ulbrich and Ulbrich [122] proposed a trust region algorithm for the equality constrained problem that does not use a merit function as done by the other algorithms.

The idea of trust region methods is to minimize successively models of the employed merit function Φ_σ in a specified region around the current iterate. This procedure was highlighted by Griffith and Stewart [54]. The model in iteration k is denoted by $\Psi_k(d)$. It is required that the model $\Psi_k(d)$ approximates the merit function $\Phi_{\sigma_k}(x)$ sufficiently well at least in the trusted region around the current iterate x_k. Moreover, the model has to satisfy the conditions

$$\Psi_k(0) = \Phi_{\sigma_k}(x_k) \tag{3.28}$$

and

$$\nabla\Psi_k(0) = \nabla\Phi_{\sigma_k}(x_k) \ . \tag{3.29}$$

The following terminology was introduced by Goldfeldt, Quandt, and Trotter [51]. The predicted reduction obtained in the model is defined as

$$Pred_k := \Psi_k(0) - \Psi_k(d_k) \ , \tag{3.30}$$

where d_k denotes the calculated trial step. The predicted reduction $Pred_k$ has to be positive for all iterations. $Pred_k$ is then compared to the actual reduction in the merit function Φ_{σ_k}. Respectively, the actual reduction in the merit function is defined as

$$Ared_k := \Phi_{\sigma_k}(x_k) - \Phi_{\sigma_k}(x_k + d_k) \ . \tag{3.31}$$

The ratio r_k of the actual change in the merit function and the predicted reduction according to the model $\Psi_k(d)$, defined as

$$r_k := \frac{Ared_k}{Pred_k} \ , \tag{3.32}$$

is used to decide whether a trial step d_k is accepted or not. The model Ψ_k has to be chosen properly as it has to be guaranteed that the difference between actual reduction and predicted reduction vanishes faster than $Pred_k$ when the trust region radius Δ_k tends to zero.

The size of the trial step d_k in iteration k is restricted by adding a constraint to the subproblem, namely

$$\|d\| \leq \Delta_k \ , \tag{3.33}$$

where $\|.\|$ is an appropriate norm and $\Delta_k > 0$ is the current trust region radius. Thus, trust region methods guarantee $\|x_{k+1} - x_k\| \leq \Delta_k$. By adjusting the trust region radius

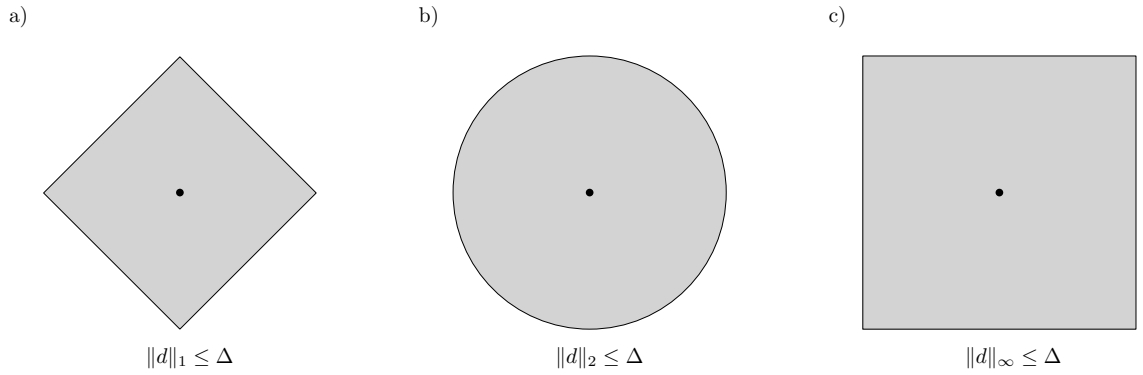

Figure 3.2: Illustration of Trust Regions for Different Norms

Δ_k the acceptance of trial steps can be influenced as r_k tends to one when Δ_k tends to zero.

Depending on the norm employed for determining the step size, the shape of the trust region differs. This is illustrated in Figure 3.2. Using the L_∞-norm in the trust region constraint (3.33) has an important advantage. In this case the trust region constraint (3.33) can be reformulated as simple linear bound constraints on the step, i.e., $2n$ linear inequality constraints are added to the original subproblem formulation. Consequently, any quadratic programming solver can handle the reformulated trust region constraint without additional modifications.

A general framework for trust region methods can be stated as follows.

Algorithm 3.2 (A General Trust Region Method) Let $0 < \tau_1 < \tau_2 < 1 < \tau_3$, $0 < \rho < 1$, and $x_0 \in \mathbb{R}^n$ be given. For $k = 0, 1, 2, \ldots$ repeat until convergence

STEP 1 Compute a trial step d_k.

STEP 2 Compute the ratio r_k according to (3.32).

STEP 3 Let

$$x_{k+1} := \begin{cases} x_k + d_k & \text{, if } r_k > \rho \text{ ,} \\ x_k & \text{, otherwise ,} \end{cases}$$

and

$$\Delta_{k+1} \in \begin{cases} [\Delta_k,\ \tau_3 \Delta_k] & \text{, if } \ r_k > \rho \text{ ,} \\ [\tau_1 \|d_k\|,\ \tau_2 \|d_k\|] & \text{, otherwise } \ . \end{cases}$$

STEP 4 Define a new model $\Psi_{k+1}(d)$.

In STEP 3 the acceptance of the trial step d_k is tested with respect to parameter ρ. A step d_k is taken if the ratio r_k is greater than the threshold ρ. Depending on the result of the test, the step size restriction for the next iteration is adapted.

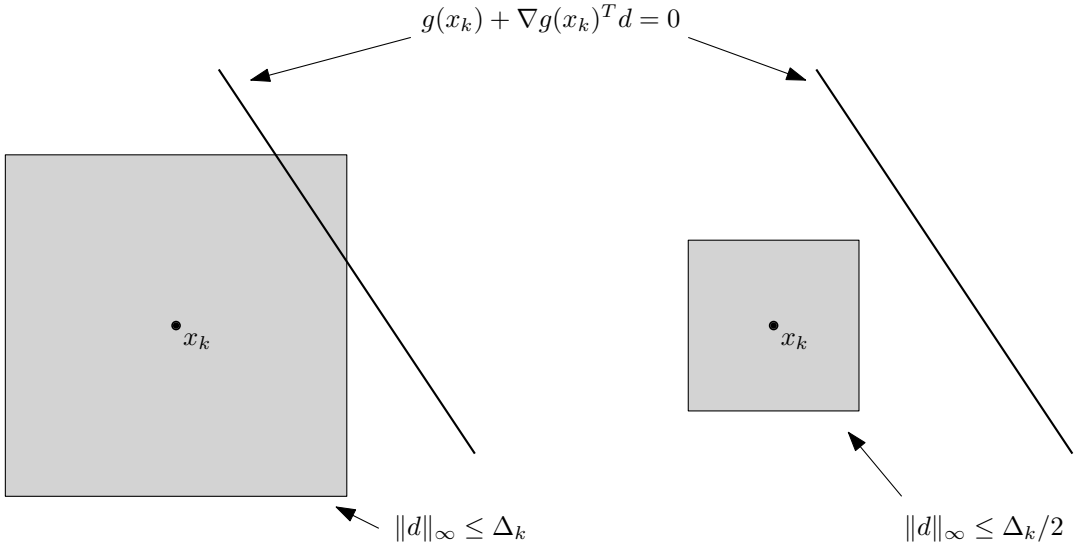

Figure 3.3: Inconsistency Caused by Trust Region Constraint

For detailed discussions on trust region methods, the textbooks by Conn, Gould, and Toint [21], and Sun and Yuan [113], and the review paper by Yuan [131] are recommended.

In the remainder of this chapter existing trust region SQP methods are presented. They extend the basic framework shown in Algorithm 3.2 as difficulties can arise in practice. For instance, adding the trust region constraint (3.33) to the quadratic program (3.1) of an SQP method can lead to infeasible subproblems. The trust region constraint (3.33) may prohibit steps that are necessary to satisfy the linearized constraints corresponding to the underlying optimization problem. In Figure 3.3 two different values for Δ_k are used to illustrate the difficulty. The example on the left-hand side is feasible, whereas the one on the right-hand side has no feasible solution.

There are several approaches to overcome inconsistent subproblems as described above. They differ in the way of constructing trust region subproblems and determining the trial steps. Most of the methods were proposed in the context of equality constrained problems (1.2). Thus, in some cases only equality constraints are considered. Note that all inequality constraints in (1.2) can be transformed into equality constraints by introducing slack variables $s_{m_e+1}, \ldots, s_m$, with $s_j \geq 0$, $j = m_e + 1, \ldots, m$, i.e.,

$$g_j(x) - s_j = 0 \ , \quad j = m_e + 1, \ldots, m \ . \tag{3.34}$$

The disadvantage of transforming the problem is the increasing number of variables. In addition, the nonnegative constraints $s_j \geq 0$, $j = m_e + 1, \ldots, m$, have to be satisfied. Different techniques can be applied, see, e.g., Dennis, Heinkenschloss, and Vincent [25], Coleman and Li [20], and Byrd, Gilbert, and Nocedal [15]. They propose the use of barrier functions or adapted trust regions, i.e., reshaping the trust region in order to assure $s_j \geq 0$, $j = m_e + 1, \ldots, m$.

3.4.1 Vardi-like Approach

The following method was originally suggested for equality constrained problems, i.e., problems where $m = m_e$ holds. Thus, the quadratic problem of the SQP method at iterate x_k, without the trust region constraint, is

$$\begin{aligned} \underset{d\in\mathbb{R}^n}{\text{minimize}} \quad & \nabla f(x_k)^T d + \frac{1}{2} d^T B_k d \\ \text{subject to} \quad & g_j(x_k) + \nabla g_j(x_k)^T d = 0 \ , \quad j = 1, \ldots, m_e \ . \end{aligned} \tag{3.35}$$

In order to avoid inconsistent subproblems, that may arise, e.g., by adding the trust region constraint $\|d\|_2 \leq \Delta_k$, the approach uses the idea of scaling the constraints by a parameter $\theta_k \in (0, 1]$ so that

$$\theta_k g_{\mathcal{E}}(x_k) + \nabla g_{\mathcal{E}}(x_k)^T d = 0 \tag{3.36}$$

holds for a d with $\|d\|_2 \leq \Delta_k$. This approach was studied by many authors, e.g., see Vardi [123], Byrd, Schnabel, and Shultz [16], and Omojokun [80]. The trial step d_k is the solution of the quadratic problem

$$\begin{aligned} \underset{d\in\mathbb{R}^n}{\text{minimize}} \quad & \nabla f(x_k)^T d + \frac{1}{2} d^T B_k d \\ \text{subject to} \quad & \theta_k g_j(x_k) + \nabla g_j(x_k)^T d = 0 \ , \quad j = 1, \ldots, m_e \ , \\ & \|d\|_2 \leq \Delta_k \ , \end{aligned} \tag{3.37}$$

for some parameter $\theta_k \in (0, 1]$. Using the L_2-norm for the trust region constraint requires adapted quadratic programming solvers as the problem is not in standard form anymore.

As θ_k is not known a priori, a strategy has to be applied to determine d_k. Therefore, the trial step d_k is decomposed into two steps, i.e., $d_k := d_k^t + d_k^n$, where d_k^n is called the normal step and d_k^t is the tangential step. d_k^n is a range space step and reduces the linearized constraint $\|g_{\mathcal{E}}(x_k) + \nabla g_{\mathcal{E}}(x_k)^T d\|_2$, whereas d_k^t reduces the approximated Lagrangian function in the null space of $\nabla g_{\mathcal{E}}(x_k)^T$. For example, d_k^n can be the solution of the following problem

$$\begin{aligned} \underset{d\in\mathbb{R}^n}{\text{minimize}} \quad & \|g_{\mathcal{E}}(x_k) + \nabla g_{\mathcal{E}}(x_k)^T d\|_2 \\ \text{subject to} \quad & \|d\|_2 \leq \tau \Delta_k \ , \end{aligned} \tag{3.38}$$

where $\tau \in (0, 1)$ is a constant independent of k. The parameter $\tau < 1$ leaves freedom for defining d_k^t.

Once, d_k^n is computed, e.g., by solving (3.38), d_k^t can be obtained by solving

$$\begin{aligned} \underset{d\in\mathbb{R}^n}{\text{minimize}} \quad & \nabla f(x_k)^T d + \frac{1}{2} (d_k^n + d)^T B_k (d_k^n + d) \\ \text{subject to} \quad & \nabla g_{\mathcal{E}}(x_k)^T d = 0 \ , \\ & \|d_k^n + d\|_2 \leq \Delta_k \ . \end{aligned} \tag{3.39}$$

3.4.2 Celis-Dennis-Tapia-like Approach

The equality constrained problem is considered again. Celis, Dennis, and Tapia [17] suggested the substitution of the linearized constraints in the quadratic problem (3.35) by a single constraint

$$\|g_{\mathcal{E}}(x_k) + \nabla g_{\mathcal{E}}(x_k)^T d\|_2 \leq \bar{\theta}_k \ , \tag{3.40}$$

where

$$\bar{\theta}_k \geq \min_{\|d\|_2 \leq \Delta_k} \|g_{\mathcal{E}}(x_k) + \nabla g_{\mathcal{E}}(x_k)^T d\|_2 \ . \tag{3.41}$$

The resulting subproblem, also investigated by El-Alem [30, 31], is feasible and formulated as

$$\begin{array}{ll} \underset{d \in \mathbb{R}^n}{\text{minimize}} & \nabla f(x_k)^T d + \frac{1}{2} d^T B_k d \\ \text{subject to} & \|g(x_k) + \nabla g(x_k)^T d\|_2 \leq \bar{\theta}_k \ , \\ & \|d\|_2 \leq \Delta_k \ , \end{array} \tag{3.42}$$

where $\bar{\theta}_k$ satisfies (3.41). The proposed methods differ in the choice of $\bar{\theta}_k$. The value of $\bar{\theta}_k$ has to be set sufficiently small to achieve convergence to a feasible point.

Celis, Dennis, and Tapia [17] chose $\bar{\theta}_k$ to be

$$\bar{\theta}_k = \|g_{\mathcal{E}}(x_k) + \nabla g_{\mathcal{E}}(x_k)^T d_k^{CP}\|_2 \ , \tag{3.43}$$

where $d_k^{CP} = -\alpha_k \nabla g_{\mathcal{E}}(x_k) g_{\mathcal{E}}(x_k)$ is the Cauchy step, i.e., the minimizer of $\|g_{\mathcal{E}}(x_k) + \nabla g_{\mathcal{E}}(x_k)^T d\|_2$, with $\|d\|_2 \leq \Delta_k$, along its negative gradient. $\alpha_k \geq 0$ is the step size parameter.

Powell and Yuan [93] proposed the following choice of $\bar{\theta}_k$

$$\min_{\|d\|_2 \leq \tau_1 \Delta_k} \|g(x_k) + \nabla g(x_k)^T d\|_2 \leq \bar{\theta}_k \leq \min_{\|d\|_2 \leq \tau_2 \Delta_k} \|g(x_k) + \nabla g(x_k)^T d\|_2 \ , \tag{3.44}$$

where $0 < \tau_2 < \tau_1 < 1$ are two constants.

3.4.3 Yuan-like Approach

This approach results in trust region subproblems that approximate directly a specific penalty function. Yuan [130] proposed an algorithm that can also handle inequality constraints. In Chapter 5 mixed-integer algorithms are introduced that are based on this algorithm by Yuan.

Based on the L_∞-penalty function, cf. (3.12), Yuan [130] suggested the following subproblem

$$\begin{array}{ll} \underset{d \in \mathbb{R}^n}{\text{minimize}} & \Psi_{\sigma_k}(d) := \nabla f(x_k)^T d + \frac{1}{2} d^T B_k d + \sigma_k \left\| \left(g(x_k) + \nabla g(x_k)^T d \right)^- \right\|_\infty \\ \text{subject to} & \|d\|_\infty \leq \Delta_k \ , \end{array} \tag{3.45}$$

which is always feasible. Here $\Psi_{\sigma_k}(d)$ also denotes the model that is used for calcu-

lating the predicted reduction $Pred_k$, cf. (3.30). Note that the problem can easily be transformed into a quadratic problem by adding a new variable. The L_∞-penalty function can be replaced by any other penalty functions, including an augmented Lagrangian function studied by Niu and Yuan [76]. Note that Niu and Yuan also consider inequality constraints, but the constraints are reformulated as equality constraints. Fletcher [38] used the L_1-penalty function in his Sl$_1$QP method. Subproblem (3.45) is adapted accordingly.

If a penalty function of form (3.9) is used, e.g., the L_∞-penalty function, the Maratos effect can occur. Fletcher [39] and Yuan [129, 130] suggested to try a second order correction step $\hat{d}_k$ whenever the solution of subproblem (3.45), denoted by d_k, is not acceptable. The step $\hat{d}_k$ is obtained by solving the following second order correction subproblem

$$\begin{aligned} \underset{d\in\mathbb{R}^n}{\text{minimize}} \quad & \nabla f(x_k)^T(d_k+d) + \frac{1}{2}(d_k+d)^T B_k(d_k+d) \\ & \qquad + \sigma_k \left\| \left(g(x_k+d_k) + \nabla g(x_k)^T d \right)^- \right\|_\infty \\ \text{subject to} \quad & \|d_k+d\|_\infty \le \Delta_k \ . \end{aligned} \tag{3.46}$$

The point $x_k+d_k+\hat{d}_k$ is then tested for acceptance. Subproblems (3.45) and (3.46) differ in $g(x_k)$ and $g(x_k+d_k)$, respectively. As $g(x_k+d_k) = g(x_k) + \nabla g(x_k)^T d_k + O(\|d_k\|_2^2)$, it follows for the second order correction step $\|\hat{d}_k\| = O(\|d_k\|_2^2)$. Consequently, by applying step $\hat{d}_k$ the convergence rate obtained by d_k retains, since d_k is identical to the SQP step when x_k is close to the solution. By adding the second order correction step $\hat{d}_k$ the Maratos effect can be avoided and the new point $x_k + d_k + \hat{d}_k$ will always be accepted when x_k is close to the solution. This result was proved by Yuan [129].

The following algorithm is a slight modification of Algorithm 5.8 in Yuan [130] . All constants are given explicitly to be as close as possible to the original presentation. The convergence results retain. An implementation of Algorithm 3.3 is part of the `FORTRAN` code `MISQP`, see Exler et al. [35]. In Chapter 6 numerical results for the implementation are presented and compared to other algorithms.

Algorithm 3.3 Let $\epsilon_{\text{tol}} > 0$ be a given constant.

STEP 0 Choose initial values for $x_0 \in \mathbb{R}^n$, $\Delta_0 > 0$, $\sigma_0 > 0$, $\zeta_0 > 0$, and $B_0 \in \mathbb{R}^{n\times n}$ symmetric. Evaluate $f(x_0)$, $g(x_0)$, $\nabla f(x_0)$, and $\nabla g(x_0)$. Set $k := 0$.

STEP 1 Solve subproblem (3.45) giving d_k.
if $\|g(x_k)^-\|_\infty \le \epsilon_{\text{tol}}$ **and** $\Psi_{\sigma_k}(0) - \Psi_{\sigma_k}(d_k) \le \epsilon_{\text{tol}}$ **then STOP** .

STEP 2 **if** $\|g(x_k)^-\|_\infty - \|(g(x_k) + \nabla g(x_k)^T d_k)^-\|_\infty < \epsilon_{\text{tol}}$ **and**
$\|(g(x_k) + \nabla g(x_k)^T d_k)^-\|_\infty > \epsilon_{\text{tol}}$ **then**
Set $\sigma_{k+1} := 10\sigma_k$ and $\zeta_{k+1} := \zeta_k/10$.

else Set $\sigma_{k+1} := \sigma_k$ and $\zeta_{k+1} := \zeta_k$.
if $\Psi_{\sigma_k}(0) - \Psi_{\sigma_k}(d_k) < \zeta_k \, \sigma_k \min(\Delta_k, \|g(x_k)^-\|_\infty)$ **then**
Replace $\sigma_{k+1} := 2\sigma_{k+1}$ and $\zeta_{k+1} := \zeta_{k+1}/4$.

STEP 3 Evaluate $f(x_k + d_k)$ and $g(x_k + d_k)$, and compute the ratio of the actual change and the predicted reduction

$$r_k := \frac{P_{\sigma_{k+1}}(x_k) - P_{\sigma_{k+1}}(x_k + d_k)}{\Psi_{\sigma_k}(0) - \Psi_{\sigma_k}(d_k)} \; . \tag{3.47}$$

STEP 4 **if** $r_k \leq 0.75$ **then** Solve SOC problem (3.46) to obtain a solution $\hat{d}_k$ and evaluate $f(x_k + d_k + \hat{d}_k)$ and $g(x_k + d_k + \hat{d}_k)$.
if $P_{\sigma_{k+1}}(x_k + d_k + \hat{d}_k) < P_{\sigma_{k+1}}(x_k + d_k)$ **then** Update r_k by

$$r_k := \frac{P_{\sigma_{k+1}}(x_k) - P_{\sigma_{k+1}}(x_k + d_k + \hat{d}_k)}{\Psi_{\sigma_k}(0) - \Psi_{\sigma_k}(d_k)} \; , \tag{3.48}$$

and replace $d_k := d_k + \hat{d}_k$.

STEP 5 Update the trust region radius by

$$\Delta_{k+1} := \begin{cases} \|d_k\|_\infty/2 & , \text{if} \quad r_k < 0.25 \; , \\ \Delta_k & , \text{if} \quad 0.25 \leq r_k \leq 0.75 \; , \\ \max(2\|d_k\|_\infty \, , \, \Delta_k) & , \text{if} \quad 0.75 < r_k \; . \end{cases} \tag{3.49}$$

STEP 6 **if** $r_k \leq 0$ **then** Set $x_{k+1} := x_k$, $B_{k+1} := B_k$, $k := k + 1$ and **goto** STEP 1.
else Set $x_{k+1} := x_k + d_k$.

STEP 7 Evaluate $\nabla f(x_{k+1})$ and $\nabla g(x_{k+1})$.
Generate a new matrix B_{k+1}. Set $k := k + 1$ and **goto** STEP 1.

Here ξ_k is an internal scaling parameter. In STEP 4 of Algorithm 3.3 second order correction steps are calculated to obtain fast local convergence. A convergence theorem can be stated, but additional conditions are required. These requirements are summarized in the following, where $x^\star$ is a minimizer of the problem under consideration.

Assumption 3.4
1. $f(x)$ and $g_j(x)$, $j = 1, \ldots, m$, are twice continuously differentiable,
2. $\lim\limits_{k \to \infty} x_k = x^\star$,
3. $\sigma_k = \sigma^\star$ for all large k,
4. $\{B_k\}$ is bounded,
5. $\nabla g_j(x^\star)$, $j \in \mathcal{E} \cup \mathcal{A}(x^\star)$, are linearly independent,
6. $\sigma^\star > \|u^\star\|_1$, where $u^\star$ is the unique Lagrange multiplier at the solution $x^\star$,

7. the following inequality
$$d^T \nabla^2_{xx} L(x^\star, u^\star)\, d > 0 \tag{3.50}$$
holds for all nonzero d that satisfy
$$\nabla g_j(x^\star)^T d = 0\ , \quad j \in \mathcal{E}\ , \tag{3.51}$$
$$\nabla g_j(x^\star)^T d \geq 0\ , \quad j \in \mathcal{A}(x^\star)\ , \tag{3.52}$$
with
$$\nabla^2_{xx} L(x^\star, u^\star) = \nabla^2 f(x^\star) - \sum_{j=1}^{m} u_j^\star \nabla^2 g_j(x^\star)\ . \tag{3.53}$$

8.
$$\lim_{k\to\infty} \frac{\left\| \overline{P}\Big(\nabla^2_{xx} L(x^\star, u^\star) - B_k\Big) d_k \right\|_2}{\|d_k\|_2} = 0\ , \tag{3.54}$$
where $\overline{P}$ is a projection from $\mathbb{R}^n$ to the null space of $\nabla g_{\mathcal{E}\cup\mathcal{A}(x^\star)}(x^\star)^T$.

The following result in Yuan [130] can be applied to Algorithm 3.3 although the algorithm is slightly modified compared to Algorithm 5.8 in the referenced work.

Theorem 3.5 *If the conditions of Assumption 3.4 are satisfied, then Algorithm 3.3 with* $\epsilon_{\text{tol}} = 0$ *either terminates at a KKT point or generates a sequence* $\{x_k\}$ *that converges Q-superlinearly, i.e.,*
$$\lim_{k\to\infty} \frac{\|x_{k+1} - x^\star\|_\infty}{\|x_k - x^\star\|_\infty} = 0\ . \tag{3.55}$$
Moreover, the trust region bound Δ_k *is bounded away from zero, and the trust region constraint* $\|d\|_\infty \leq \Delta_k$ *is inactive for all large* k. □

Theorem 3.5 can be proved similar to the analysis in Yuan [129].

3.4.4 Fletcher-Leyffer-Toint Filter Method

Filter were introduced by Fletcher and Leyffer [42] for inequality constrained problems. Fletcher, Leyffer, and Toint [43] extended the approach to problems with equality and inequality constraints. The filter notation introduced earlier in Section 3.2.2 is used again, i.e., the measurement of violation is defined by (3.20) and filter entries are defined by (3.21).

Let x_k be the current iterate and F_k the corresponding filter of iteration k. A trial step is denoted by $\bar{d}$. Note that $\bar{d}$ is used instead of d_k since iterations are counted differently in filter methods. Step $\bar{d}$ should reduce the constraint violation measured by h or the objective function value f. To ensure sufficient decrease of at least one of the two criteria, a new point $x_k + \bar{d}$ is *acceptable to the filter* F_k if for all $(h_j, f_j) \in F_k$ either
$$h\Big(g(x_k + \bar{d})^-\Big) \leq \lambda_1 h_j \tag{3.56}$$

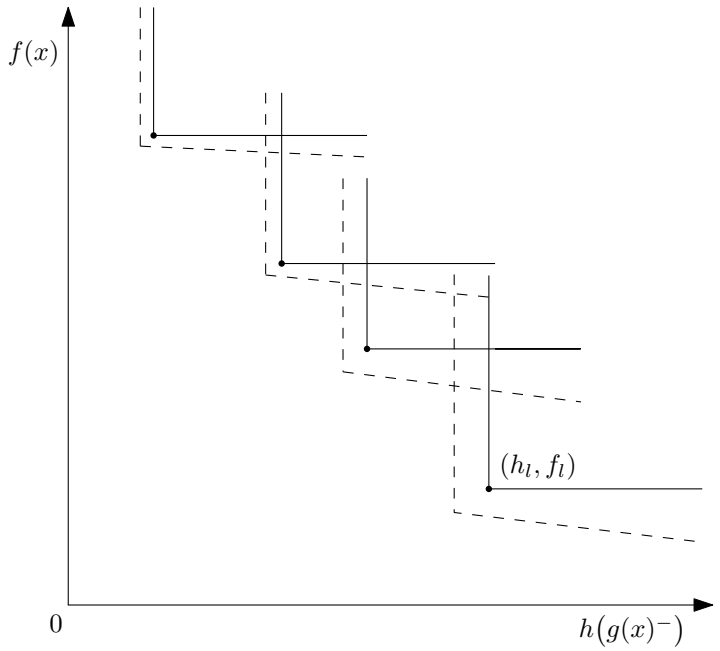

Figure 3.4: A Filter with Envelope

or

$$f(x_k + \bar{d}) + \lambda_2\, h\Big(g(x_k + \bar{d})^-\Big) \leq f_j \tag{3.57}$$

holds with constants $0 < \lambda_2 < \lambda_1 < 1$, see Fletcher, Leyffer, and Toint [43].

To prove convergence, the concept of *dominated* points, cf. (3.23) and (3.24), had to be extended to *acceptable* points, (3.56) and (3.57). Figure 3.4 shows a typical filter. The continuous line illustrates the area of points dominated by the filter as defined before. The envelope displayed by dashed line corresponds to the definition of acceptable points, i.e., (3.56) and (3.57). Thus, all points that are located above the dashed line in the upper right part are not acceptable to the filter and are rejected.

A filter-SQP algorithm tries to solve the following quadratic problem, denoted by QP(x_k,Δ),

$$\begin{aligned} \underset{d\in\mathbb{R}^n}{\text{minimize}} \quad & q_k(d) := \nabla f(x_k)^T d + \frac{1}{2} d^T B_k d \\ \text{subject to} \quad & g_j(x_k) + \nabla g_j(x_k)^T d = 0\ , \quad j = 1, \ldots, m_e\ , \\ & g_j(x_k) + \nabla g_j(x_k)^T d \geq 0\ , \quad j = m_e + 1, \ldots, m\ , \\ & \|d\|_\infty \leq \Delta\ . \end{aligned} \tag{3.58}$$

If this problem is inconsistent, then a feasibility restoration phase is entered. Otherwise, $\bar{d}$ is obtained. The aim of the restoration phase is the reduction of the constraint violation and the determination of a new point x_k acceptable to the filter F_k such that

the quadratic problem (3.58) is consistent for a $\Delta \geq \Delta_{\min} > 0$.

Applying (3.56) and (3.57), the filter guarantees convergence to feasible points. But to enforce convergence to a KKT point, in addition a sufficient reduction in the objective function has to be achieved if constraint violation is small enough. Defining

$$\Delta q_k(\bar{d}) := q_k(0) - q_k(\bar{d}) = -\nabla f(x_k)^T \bar{d} - \frac{1}{2}\bar{d}^T B_k \bar{d} \tag{3.59}$$

and

$$\Delta f_k(\bar{d}) := f(x_k) - f(x_k + \bar{d}) \ , \tag{3.60}$$

then the sufficient reduction condition is satisfied if $\Delta q_k(\bar{d}) > 0$ and

$$\Delta f_k(\bar{d}) \geq \rho \, \Delta q_k(\bar{d}) \tag{3.61}$$

holds for a constant $0 < \rho < 1$.

The filter-SQP algorithm by Fletcher, Leyffer, and Toint [43] can be stated as follows.

Algorithm 3.6 (A Filter-SQP Algorithm) Let $0 < \lambda_2 < \lambda_1 < 1$, $\rho \in (0, 1)$, and $\Delta_{\min} > 0$. Given $x_0 \in \mathbb{R}^n$, let $k := 1$. Let the initial filter be $F_1 = \{(U, -\infty)\}$ with $h(g(x_0)^-) \leq \lambda_1 U$.

STEP 1 Enter the restoration phase to find a point x_k acceptable to the filter F_k according to criteria (3.56) and (3.57) such that QP(x_k,$\widetilde{\Delta}$) is compatible for some $\widetilde{\Delta} \geq \Delta_{\min}$ and set $\Delta := \widetilde{\Delta}$.

STEP 2 **if** QP(x_k,Δ) is compatible **then** Calculate solution $\bar{d}$.
else Include (h_k, f_k) in the filter, set $k := k + 1$ and **goto** STEP 1.

STEP 3 **if** $\bar{d} = 0$ **then STOP** .

STEP 4 **if** $x_k + \bar{d}$ is not acceptable to $F_k \cup (h_k, f_k)$ with respect to (3.56) and (3.57) **then** Set $\Delta := \Delta/2$ and **goto** STEP 2.

STEP 5 **if** $\Delta q_k > 0$ and $\Delta f_k < \rho \, \Delta q_k$ **then** Set $\Delta := \Delta/2$ and **goto** STEP 2.

STEP 6 **if** $\Delta q_k \leq 0$ **then** Include (h_k, f_k) in the filter.

STEP 7 Set $x_{k+1} := x_k + \bar{d}$, initialize $\Delta \geq \Delta_{\min}$. Set $k := k + 1$ and **goto** STEP 2.

In STEP 1 a feasibility restoration phase is executed. The aim is to obtain a new iterate where the corresponding quadratic subproblem is consistent subject to a specific trust region radius Δ. The other trust region algorithms described before integrate the feasibility restoration in the main procedure. This is different for Algorithm 3.6 where this phase is not specified in detail. This is also the case in the paper by Fletcher, Leyffer, and Toint [43].

In case a trial step is accepted, then the trust region radius Δ is set to at least $\Delta_{\min}$ in Step 7. This modified update procedure, that differs from the basic Algorithm 3.2, is also used by Kanzow and Zupke [64], and Jiang et al. [63]. Requiring $\Delta \geq \Delta_{\min}$ is important to prove global convergence of Algorithm 3.6.

Note that the effect described by Maratos [71] might also occur for the filter method described above if the filter only measures the objective function and the constraint violation.

3.4.5 Ulbrich Filter Method

Ulbrich [121] modified the filter algorithm of Fletcher, Leyffer and Toint [43], cf. Algorithm 3.6, to obtain fast local convergence. The basic idea is to make use of the Lagrangian function instead of the objective function in the filter and include the dual variables. The second modification is a different measurement of constraint violation. Thus, the approach uses properties of augmented Lagrangian functions, see (3.16).

The first modification affects the filter. The filter entries $(h(g(x)^-), f(x))$ are substituted by $(\Lambda(x,u), L(x,u))$, where

$$\Lambda(x,u) := \|g_{\mathcal{E}}(x)\|_2^2 + \|g_{\mathcal{I}}(x)^-\|_2^2 + (u_{\mathcal{I}}^T g_{\mathcal{I}}(x)^-)^2 \ , \tag{3.62}$$

and $L(x,u)$ is the Lagrangian function (2.20) where the multipliers u are obtained by some multiplier function, see Ulbrich [121] for details. The definition $(\Lambda_k, L_k) := (\Lambda(x_k,u_k), L(x_k,u_k))$ is used. Consequently, a point (x,u) is *acceptable* to the filter if for all $(\Lambda_j, L_j) \in F_k$ either

$$\Lambda(x,u) \leq \lambda_1 \Lambda_j \tag{3.63}$$

or

$$L(x,u) + \lambda_2\, \Lambda(x,u) \leq L_j \ , \tag{3.64}$$

with constants $0 < \lambda_2 < \lambda_1 < 1$.

The second modification concerns Step 2 of Algorithm 3.6. QP(x_k,Δ) is *compatible* if QP(x_k,Δ) is feasible and

$$\Lambda_k^{1/2} \leq \kappa_\Delta \Delta^{1+\xi} \ , \tag{3.65}$$

with constants $\kappa_\Delta > 0$ and $\xi \in (0,1)$.

The last modification is related to Step 5 and Step 6 of Algorithm 3.6. The condition in Step 5 is replaced by

$$\Delta \widehat{q}_k(\bar{d}) := \Delta q_k(\bar{d}) + u_k^T g(x_k) > \kappa_\Lambda\, \Lambda_k^{\psi/2} \tag{3.66}$$

and

$$\Delta L_k(\bar{d}) := L(x_k,u_k) - L(x_k+d_k, u_k(\bar{d})) < \rho\, \Delta \widehat{q}_k(\bar{d}) \ , \tag{3.67}$$

with constants $\kappa_\Lambda > 0$, $\psi \in (1/2, 1]$, and $0 < \rho < 1$. $\Delta q_k(\bar{d})$ is defined by (3.59) and $u_k(\bar{d})$ denotes the new multiplier estimates obtained by a multiplier function. The condition in Step 6 is replaced by $\Delta \widehat{q}_k(\bar{d}) \leq \kappa_\Lambda\, \Lambda_k^{\psi/2}$.

Under suitable assumptions the modified algorithm accepts locally full SQP steps, see Ulbrich [121]. Thus, fast local convergence is retained.

4 A Trust Region SQP Algorithm for Constrained Nonlinear Programs

In this chapter a new sequential quadratic programming algorithm addressing nonlinear optimization problems of form (1.2) is presented. The inequality constraints of the problem are not transformed into equality constraints and no additional slack variables have to be optimized. The algorithm is stabilized by trust region techniques to obtain global convergence. An augmented Lagrangian function is employed as merit function to avoid the need of calculating second order correction steps to retain fast local convergence.

In the following section the algorithm is described in detail and the key ingredients are specified. The subproblems that are solved in each iteration in order to determine a trial step are stated. A feasibility restoration phase is introduced that is entered if inconsistency in the subproblems occurs. Moreover, the model that is used to evaluate the quality of a calculated trial step is presented and an update scheme for the penalty parameter is given.

In Section 4.2 the convergence properties of the algorithm are analyzed. The section is divided into two parts. In the beginning the global convergence of the algorithm is investigated. It is established that at least a subsequence exists that converges to a KKT point of the optimized problem. The second part of the section presents the local convergence analysis of the algorithm. It is shown that full SQP steps are accepted close to a solution of the underlying problem.

A discussion of the proposed algorithm follows in Section 4.3. The algorithm is compared to existing methods and the differences are highlighted.

4.1 Algorithm

The basic idea of the proposed algorithm is to employ a differentiable merit function to retain fast local convergence without additional safeguards. As merit function an augmented Lagrangian proposed by Rockafellar [94] is applied. A slightly modified version with multiple penalty parameters is used by Schittkowski [100] in an SQP method that is stabilized by line search techniques. In contrast to penalty functions of form (3.9), the augmented Lagrangian also involves the dual variables of the problem. Thus, the number of considered variables is $n+m$ and the domain of the optimization problem is $\mathbb{R}^{n+m}$.

Let (x_k, v_k) be the current iterate, then the augmented Lagrangian merit function at iteration k is formulated as

$$\Phi_{\sigma_k}(x_k, v_k) := f(x_k) - \sum_{j \in \mathcal{S}_k} \left(v_j^{(k)} g_j(x_k) - \frac{1}{2}\sigma_k g_j(x_k)^2 \right) - \frac{1}{2} \sum_{j \in \overline{\mathcal{S}}_k} \frac{v_j^{(k)^2}}{\sigma_k} , \tag{4.1}$$

where

$$\mathcal{S}_k := \mathcal{E} \cup \left\{ j \in \mathcal{I} \mid \ g_j(x_k) \leq v_j^{(k)}/\sigma_k \right\} \tag{4.2}$$

and

$$\overline{\mathcal{S}}_k := \{1, \ldots, m\} \setminus \mathcal{S}_k \ , \tag{4.3}$$

see also, e.g., Geiger and Kanzow [47], and Rockafellar [94]. Obviously, for equality constrained problems, i.e., in case $m_e = m$ holds, the augmented Lagrangian function $\Phi_{\sigma_k}(x_k, v_k)$ (4.1) reduces to

$$\Phi_{\sigma_k}(x_k, v_k) := f(x_k) - \sum_{j \in \mathcal{E}} \left(v_j^{(k)} g_j(x_k) - \frac{1}{2}\sigma_k g_j(x_k)^2 \right) = L(x_k, v_k) + \frac{1}{2}\sigma_k \|g(x_k)\|_2^2 \ . \tag{4.4}$$

Here the weighted value of the constraint violation is added to the Lagrangian function (2.20) of the equality constrained optimization problem.

The subsequent section specifies the subproblems that are solved in each iteration in order to determine a trial step (d_k, w_k), where d_k denotes the step in the primal variables and w_k denotes the step in the dual variables. Moreover, a feasibility restoration phase is introduced that is entered if inconsistency in the subproblems occurs.

4.1.1 Calculation of Trial Steps

In each iteration k a trial step (d_k, w_k), which consists of a step d_k in the primal variables and a step w_k in the dual variables, has to be calculated. The exact determination of w_k is explained later. Let (x_k, v_k) be the current iterate. The basic idea is to solve the standard SQP subproblem (3.1) extended by the trust region constraint $\|d\|_\infty \leq \Delta_k$. The trust region constraint can easily be transformed into simple bound constraints, i.e., $\|d\|_\infty \leq \Delta_k$ can be replaced by $2n$ linear inequality constraints, that is

$$d_i + \Delta_k \geq 0 \ , \quad i = 1, \ldots, n \ , \tag{4.5}$$

and

$$\Delta_k - d_i \geq 0 \ , \quad i = 1, \ldots, n \ . \tag{4.6}$$

The resulting subproblem is a quadratic problem that can be solved by any available solver. No special strategy for handling the trust region constraint has to be added. Reformulating the trust region constraint leads to the following subproblem formulation

$$\begin{aligned} \underset{d \in \mathbb{R}^n}{\text{minimize}} \quad & \nabla f(x_k)^T d + \frac{1}{2} d^T B_k d \\ \text{subject to} \quad & g_j(x_k) + \nabla g_j(x_k)^T d = 0 \ , \qquad j \in \mathcal{E} \ , \\ & g_j(x_k) + \nabla g_j(x_k)^T d \geq 0 \ , \qquad j \in \mathcal{I} \ , \\ & -\Delta_k \leq d_i \leq \Delta_k \ , \qquad\qquad i = 1, \ldots, n \ , \end{aligned} \tag{4.7}$$

where $B_k \in \mathbb{R}^{n \times n}$ is a symmetric matrix that approximates the Hessian of the Lagrangian function (2.20) of the underlying problem.

In the algorithm B_k is required to be positive definite. Positive definiteness is sufficient for the global convergence theory. Additional requirements are only necessary for the local convergence theory. Then matrix B_k has to be a good approximation of the Hessian of the Lagrangian function in some sense.

If the feasible region of problem (4.7) is not empty, then the solution of (4.7) is denoted by (d_k, u_k, μ_k), where $u_k \in \mathbb{R}^m$ is the Lagrangian multiplier vector corresponding to the linear constraints $g_j(x_k) + \nabla g_j(x_k)^T d_k$, $j = 1, \ldots, m$. To simplify the notation in the remainder of this work, $\mu_k \in \mathbb{R}$ is introduced and defined by

$$\mu_k := \sum_{i=1}^{n} \left(\underline{\mu}_i^{(k)} + \overline{\mu}_i^{(k)} \right) \ , \tag{4.8}$$

where $\underline{\mu}_k := (\underline{\mu}_1^{(k)}, \ldots, \underline{\mu}_n^{(k)})^T$ denotes the multipliers corresponding to the lower bounds (4.5) on step d_k, and $\overline{\mu}_k := (\overline{\mu}_1^{(k)}, \ldots, \overline{\mu}_n^{(k)})^T$ denotes the multipliers corresponding to the upper bounds (4.6), respectively. A more detailed derivation is stated in Section 4.2.

As illustrated by an example in Section 3.4, subproblem (4.7) can be infeasible and no solution exists. In order to overcome this situation, a *feasibility restoration phase* is introduced. Such a feasibility restoration phase is also used by the filter algorithm, see Algorithm 3.6 by Fletcher, Leyffer, and Toint [43]. This approach differs from other methods by the fact that the standard procedure is to solve the undisturbed subproblem (4.7). Approaches as the Vardi-like ones, the Celis-Dennis-Tapia ones or the one by Yuan, see earlier comments in Section 3.4, apply relaxation techniques in each iteration to guarantee consistency of the subproblems during the whole optimization process. Consequently, one has to take care of additional safeguards to achieve convergence. The aim of the strategy employed by the new algorithm presented here is to avoid the need of an additional penalty parameter in the subproblems. Therefore, subproblem (4.7) is solved whenever possible. Only in case the problem is inconsistent a switch to a different subproblem is performed. This strategy is also employed by an algorithm addressing equality constrained problems proposed by El-Alem [30]. El-Alem's algorithm also tries to solve the equality constrained formulation of subproblem (4.7) first. If the problem is infeasible, then a relaxed problem is solved to obtain a trial step.

During the restoration phase the trial steps are determined in two steps. First, the minimum constraint violation that can be achieved within the trust region bound is determined. The *feasibility restoration problem* that is solved in this situation is defined as

$$\begin{aligned}
\underset{d \in \mathbb{R}^n,\, \delta \in \mathbb{R}}{\text{minimize}} \quad & \sum_{j \in \mathcal{E} \cup \mathcal{A}_k} g_j(x_k)^2 \delta^2 \\
\text{subject to} \quad & g_j(x_k)(1-\delta) + \nabla g_j(x_k)^T d = 0 \ , && j \in \mathcal{E} \ , \\
& g_j(x_k)(1-\delta) + \nabla g_j(x_k)^T d \geq 0 \ , && j \in \mathcal{A}_k \ , \\
& g_j(x_k) + \nabla g_j(x_k)^T d \geq 0 \ , && j \in \mathcal{B}_k \ , \\
& -\Delta_k \leq d_i \leq \Delta_k \ , && i = 1, \ldots, n \ , \\
& 0 \leq \delta \leq 1 \ ,
\end{aligned} \tag{4.9}$$

where the sets $\mathcal{A}_k$ and $\mathcal{B}_k$ stand for $\mathcal{A}(x_k, 0)$ and $\mathcal{B}(x_k, 0)$ as defined by (2.15) and (2.16), respectively. This problem always has a solution, since $(d, \delta) = (0, 1)$ is feasible. The problem determines a relaxation parameter δ_k. Note that inactive inequality constraints, i.e., constraints in set $\mathcal{B}_k$, are not relaxed and the linearized constraints remain satisfied. The solution of problem (4.9) be denoted by $\bar{d}_k$ and δ_k.

After the required relaxation parameter δ_k has been calculated, a second subproblem is set up, where the violated constraints are relaxed. It is stated as

$$\begin{aligned} \underset{d\in\mathbb{R}^n}{\text{minimize}} \quad & \nabla f(x_k)^T d + \frac{1}{2} d^T B_k d \\ \text{subject to} \quad & g_j(x_k)(1-\delta_k) + \nabla g_j(x_k)^T d = 0 \ , \quad && j \in \mathcal{E} \ , \\ & g_j(x_k)(1-\delta_k) + \nabla g_j(x_k)^T d \geq 0 \ , \quad && j \in \mathcal{A}_k \ , \\ & g_j(x_k) + \nabla g_j(x_k)^T d \geq 0 \ , && j \in \mathcal{B}_k \ , \\ & -\Delta_k \leq d_i \leq \Delta_k \ , && i = 1, \ldots, n \ . \end{aligned} \tag{4.10}$$

In subproblem (4.10) the parameter δ_k remains fixed. Subproblem (4.10) is consistent, as $\bar{d}_k$ is a feasible point for (4.10). The solution of (4.10) is also denoted by (d_k, u_k, μ_k), where u_k is the multiplier vector with respect to the m linear approximations of the constraints and μ_k is obtained according to (4.8) with the corresponding multipliers $\underline{\mu}_k$ and $\overline{\mu}_k$. Since problem (4.7) is inconsistent in this case, there exists at least one linearized constraint that had to be relaxed.

The algorithm employs an additional variable $z_k \in \mathbb{R}^m$ that measures the violation of the linearized constraints at the solution of (4.7) or (4.10), respectively. In case the standard subproblem (4.7) is feasible, then the m entries of z_k are set to zero since all linearized constraints are satisfied. If subproblem (4.10) is solved, then the vector z_k is determined according to

$$z_j^{(k)} := \begin{cases} 0 & , \ j \in \mathcal{B}_k \ , \\ \max\left(0, \dfrac{g_j(x_k) + \nabla g_j(x_k)^T d_k}{g_j(x_k)}\right) & , \ j \in \mathcal{E} \cup \mathcal{A}_k \text{ and } g_j(x_k) \neq 0 \ , \\ 0 & , \ j \in \mathcal{E} \cup \mathcal{A}_k \text{ and } g_j(x_k) = 0 \ . \end{cases} \tag{4.11}$$

Note that $0 \leq z_j^{(k)} \leq 1$ holds for all $j = 1, \ldots, m$. Obviously, this is true if $j \in \mathcal{B}_k$ or $j \in \mathcal{E} \cup \mathcal{A}_k$ and $g_j(x_k) = 0$. Otherwise, it follows from $0 \leq \delta_k \leq 1$.

Now the trial step (d_k, w_k) can be specified. The primal step d_k is the solution of the corresponding subproblem, i.e., d_k is either the minimizer of problem (4.7), if the subproblem is consistent, or the solution to the relaxed problem (4.10). The step w_k in the dual variables is set to

$$w_j^{(k)} := \left(u_j^{(k)} - v_j^{(k)}\right)\left(1 - z_j^{(k)}\right) \ , \quad j = 1, \ldots, m \ , \tag{4.12}$$

where z_k is either zero, in case problem (4.7) is consistent, or defined by (4.11). Thus, the size of the dual step w_k is also controlled in some sense during the feasibility

restoration phase. Definition (4.12) is motivated by the convergence analysis.

The following section introduces the model that is applied to evaluate the quality of the calculated trial steps.

4.1.2 Model Formulation

As described previously in Section 3.4, trust region methods use a model to approximate the original problem. In each iteration k the predicted reduction in the model is compared to the actual change in the augmented Lagrangian Φ_{σ_k} (4.1). Depending on the calculated ratio of the actual change and the predicted reduction a trial step is either accepted or rejected.

The model employed by the proposed algorithm corresponds to the augmented Lagrangian function formulated for the quadratic subproblem (4.7) at iterate (x_k, v_k), where the trust region constraint is ignored and $f(x_k)$ is added to the objective function. Let (x_k, v_k) be the k-th iterate and B_k be the matrix of the corresponding subproblem, then the model is defined as

$$\begin{aligned}\Psi_{\sigma_k}(d,w) := \; & f(x_k) + \nabla f(x_k)^T d + \frac{1}{2} d^T B_k d \\ & - \sum_{j \in \mathcal{M}_k^a} \left(\left(v_j^{(k)} + w_j \right) \left(g_j(x_k) + \nabla g_j(x_k)^T d \right) - \frac{1}{2} \sigma_k \left(g_j(x_k) + \nabla g_j(x_k)^T d \right)^2 \right) \\ & - \sum_{j \in \overline{\mathcal{M}}_k^a} \frac{1}{2} \frac{\left(v_j^{(k)} + w_j \right)^2}{\sigma_k}\end{aligned} \tag{4.13}$$

for all (d, w), where the sets $\mathcal{M}_k^a$ and $\overline{\mathcal{M}}_k^a$ are defined in the following way

$$\mathcal{M}_k^a := \mathcal{E} \cup \left\{ j \in \mathcal{I} \mid \; g_j(x_k) + \nabla g_j(x_k)^T d \leq \left(v_j^{(k)} + w_j \right) / \sigma_k \right\} \tag{4.14}$$

and

$$\overline{\mathcal{M}}_k^a := \{1, \ldots, m\} \setminus \mathcal{M}_k^a \; . \tag{4.15}$$

In each iteration k the model is evaluated twice. First, the model value is determined for the trial steps (d_k, w_k), that is

$$\begin{aligned}\Psi_{\sigma_k}(d_k,w_k) = \; & f(x_k) + \nabla f(x_k)^T d_k + \frac{1}{2} d_k^T B_k d_k \\ & - \sum_{j \in \mathcal{M}_k} \left(\left(v_j^{(k)} + w_j^{(k)} \right) \left(g_j(x_k) + \nabla g_j(x_k)^T d_k \right) - \frac{1}{2} \sigma_k \left(g_j(x_k) + \nabla g_j(x_k)^T d_k \right)^2 \right) \\ & - \sum_{j \in \overline{\mathcal{M}}_k} \frac{1}{2} \frac{\left(v_j^{(k)} + w_j^{(k)} \right)^2}{\sigma_k} \; ,\end{aligned} \tag{4.16}$$

where the sets $\mathcal{M}_k$ and $\overline{\mathcal{M}}_k$ for step (d_k, w_k) are defined as

$$\mathcal{M}_k := \mathcal{E} \cup \left\{ j \in \mathcal{I} \mid \ g_j(x_k) + \nabla g_j(x_k)^T d_k \leq \left(v_j^{(k)} + w_j^{(k)} \right) / \sigma_k \right\} \tag{4.17}$$

and

$$\overline{\mathcal{M}}_k := \{1, \ldots, m\} \setminus \mathcal{M}_k \ . \tag{4.18}$$

Moreover, the model Ψ_{σ_k} is evaluated at $(d, w) = (0, 0)$, and the corresponding value of the model is

$$\Psi_{\sigma_k}(0,0) := \ f(x_k) - \sum_{j \in \mathcal{M}_k^0} \left(v_j^{(k)} g_j(x_k) - \frac{1}{2} \sigma_k g_j(x_k)^2 \right) - \sum_{j \in \overline{\mathcal{M}}_k^0} \frac{1}{2} \frac{{v_j^{(k)}}^2}{\sigma_k} \ , \tag{4.19}$$

where the index sets $\mathcal{M}_k^0$ and $\overline{\mathcal{M}}_k^0$ are defined according to

$$\mathcal{M}_k^0 := \mathcal{E} \cup \left\{ j \in \mathcal{I} \mid \ g_j(x_k) \leq v_j^{(k)} / \sigma_k \right\} \tag{4.20}$$

and

$$\overline{\mathcal{M}}_k^0 := \{1, \ldots, m\} \setminus \mathcal{M}_k^0 \ . \tag{4.21}$$

By the definitions of the index sets $\mathcal{S}_k$, $\overline{\mathcal{S}}_k$, $\mathcal{M}_k^0$, and $\overline{\mathcal{M}}_k^0$, see (4.2), (4.3), (4.20), and (4.21), it follows that $\mathcal{M}_k^0 = \mathcal{S}_k$ and $\overline{\mathcal{M}}_k^0 = \overline{\mathcal{S}}_k$. Thus, $\Psi_{\sigma_k}(0,0)$ is identical to $\Phi_{\sigma_k}(x_k, v_k)$, and the equation

$$\Psi_{\sigma_k}(0,0) = \Phi_{\sigma_k}(x_k, v_k) \tag{4.22}$$

holds for all iterations k.

Since the penalty parameter σ_k has to satisfy certain conditions to obtain a sufficient decrease in the model, an update scheme is introduced in the next section.

4.1.3 Penalty Parameter Update

The penalty parameter update is a crucial part as the performance of an algorithm might be influenced if the penalty parameter is set to a too large value. The following update scheme is motivated by an update formula suggested by Schittkowski [100] for an SQP method that is stabilized by line search techniques. Schittkowski showed that the penalty update guarantees a sufficient decrease in an augmented Lagrangian function similar to (4.1) when the determined search direction is applied. For the algorithm presented here an almost identical update rule leads to a sufficient reduction in the model Ψ_{σ_k} when evaluating the trial step (d_k, w_k).

Let (x_k, v_k) be the k-th iterate and (d_k, u_k, μ_k) be determined either by subproblem (4.7) or by problem (4.10) in case the feasibility restoration phase is executed. μ_k is obtained according to (4.8). If subproblem (4.7) is feasible z_k is set to zero. Otherwise, z_k is calculated by (4.11). Let w_k be determined according to (4.12), and the matrix B_k, that approximates the Hessian of the Lagrangian function in the subproblems, be positive definite. To assure that the predicted reduction $Pred_k$ in the model Ψ_{σ_k},

defined as

$$Pred_k := \Psi_{\sigma_k}(0,0) - \Psi_{\sigma_k}(d_k, w_k) \ , \tag{4.23}$$

is sufficiently large, the penalty parameter σ_k is updated according to

$$\sigma_k := \max\left(\sigma_{k-1} \ , \ \max_{1\le j\le m}\left(\frac{2m\left(u_j^{(k)} - v_j^{(k)}\right)^2}{d_k^T B_k d_k + 2\mu_k\Delta_k}\left(1 - z_j^{(k)^2}\right)\right)\right) \ . \tag{4.24}$$

Here σ_{k-1} is the penalty parameter of the previous iteration $k-1$ and Δ_k is the trust region bound on the trial step d_k. Since B_k is required to be positive definite, $d_k^T B_k d_k$ is greater than zero as long as $\|d_k\|_\infty > 0$, and thus all terms on the right-hand side are positive.

Compared to the update rule presented in Schittkowski [100], the one stated in (4.24) differs slightly. Instead of m penalty parameters as suggested by Schittkowski, only a single penalty parameter σ_k is employed here. The trust region radius Δ_k and the corresponding multiplier μ_k in (4.24) prevent the penalty parameter from increasing too fast. Moreover, z_k is introduced to control the penalty parameter with respect to the constraint violation.

4.1.4 Algorithm Formulation

The algorithm extends the basic trust region Algorithm 3.2 described in Section 3.4. Before the detailed algorithm is formulated, some comments on the choice and the purpose of the constants that appear in the algorithm are stated. The following restrictions have to be satisfied by the constants τ_1, τ_2, ρ_0, ρ_1, $\Delta_{\min}$, $\Delta_{\max} \in \mathbb{R}$

$$0 < \tau_1 < 1 < \tau_2 < \infty \ \ , \ \ 0 < \rho_0 \le \rho_1 < 1 \ \ , \ \ 0 < \Delta_{\min} < \Delta_{\max} < \infty \ . \tag{4.25}$$

The parameters τ_1 and τ_2 are used to update the trust region radius Δ_{k+1} of the next iteration. The radius is either reduced or increased dependent on the ratio of the actual change in the augmented Lagrangian and the predicted reduction in the model obtained by the trial step (d_k, w_k). A step is rejected if the ratio is less than ρ_0. Otherwise the trial step is accepted. In case the ratio is greater than ρ_1, then the trial step is categorized as a very successful step and the trust region radius is enlarged. The parameter $\Delta_{\min}$ denotes the lower bound on the trust region Δ_{k+1} in case a trial step has been accepted. The size of Δ_{k+1} is bounded by $\Delta_{\max}$.

The sequential quadratic programming algorithm is formulated as follows.

Algorithm 4.1 Let the constants τ_1, τ_2, ρ_0, ρ_1, $\Delta_{\min}$, and $\Delta_{\max}$ be chosen properly according to (4.25).

STEP 0 Choose initial values for $x_0 \in \mathbb{R}^n$, $v_0 \in R^m$, $\Delta_0 \in \mathbb{R}$, $\sigma_{-1} \in \mathbb{R}$, and a symmetric positive definite matrix $B_0 \in \mathbb{R}^{n\times n}$, where $\Delta_{\min} \le \Delta_0 \le \Delta_{\max}$ and $\sigma_{-1} \ge 1$. Additionally, $v_j^{(0)} \ge 0$ is required for all $j \in \mathcal{I}$.

Evaluate $f(x_0)$, $\nabla f(x_0)$, $g_j(x_0)$, and $\nabla g_j(x_0)$, $j = 1, \ldots, m$.
Set $k := 0$.

STEP 1 Try to solve the standard subproblem (4.7).
if a solution to subproblem (4.7) exists **then**
Obtain d_k, u_k, and μ_k.
Set $z_j^{(k)} := 0$ for all $j = 1, \ldots, m$.
else
Determine δ_k by solving the feasibility subproblem (4.9).
Solve subproblem (4.10) with fixed δ_k and obtain d_k, u_k, and μ_k.
Set z_k according to (4.11).
end if
Determine w_k according to (4.12).

STEP 2 **if** $\|d_k\|_\infty = 0$ **then STOP** .

STEP 3 Determine the penalty parameter σ_k according to (4.24).

STEP 4 Evaluate the function values $f(x_k + d_k)$ and $g_j(x_k + d_k)$, $j = 1, \ldots, m$.
Calculate the ratio of actual change and predicted reduction

$$r_k := \frac{\Phi_{\sigma_k}(x_k, v_k) - \Phi_{\sigma_k}(x_k + d_k, v_k + w_k)}{\Psi_{\sigma_k}(0, 0) - \Psi_{\sigma_k}(d_k, w_k)} \ . \tag{4.26}$$

STEP 5 Update the trust region radius according to

$$\Delta_{k+1} := \begin{cases} \tau_1 \|d_k\|_\infty & , \text{if} \quad \rho_0 > r_k \ , \\ \max\left(\Delta_{\min},\ \Delta_k\right) & , \text{if} \quad \rho_0 \leq r_k \leq \rho_1 \ , \\ \max\left(\Delta_{\min},\ \min\left(\tau_2 \Delta_k, \Delta_{\max}\right)\right) & , \text{if} \quad \rho_1 < r_k \ . \end{cases} \tag{4.27}$$

STEP 6 **if** $r_k < \rho_0$ **then** Set $x_{k+1} := x_k$, $v_{k+1} := v_k$, and $B_{k+1} := B_k$.
else
Set $x_{k+1} := x_k + d_k$ and $v_{k+1} := v_k + w_k$.
Evaluate the gradients $\nabla f(x_{k+1})$ and $\nabla g_j(x_{k+1})$, $j = 1, \ldots, m$.
Generate a new positive definite matrix B_{k+1}.
end if
Set $k := k + 1$ and **goto** STEP 1.

In Step 1 it is tried to solve the standard quadratic problem (4.7) that is not relaxed. If no solution to the problem exists, then a feasibility restoration phase is entered similar to filter methods, see Fletcher, Leyffer, and Toint [43]. In this situation the two problems (4.9) and (4.10) are solved to obtain a new trial step. The aim is to guide the iteration sequence to the feasible region and to obtain an iterate such that the standard subproblem (4.7) is feasible again. Thus, in the worst case three problems have to be solved. If the next trial step is rejected again, then the number of subproblems reduces to two as the standard problem (4.7) is still infeasible.

In Step 2 the algorithm terminates if the length of the obtained step d_k is zero. If d_k solves subproblem (4.7) then a KKT point of the underlying optimization problem has been obtained.

The penalty parameter σ_k is updated in Step 3. The penalty update guarantees that sufficient decrease in the model Ψ_{σ_k} is obtained, i.e., the predicted reduction $Pred_k$ (4.23) is sufficiently large. This is shown in the global convergence analysis in Section 4.2.1. The ratio of actual change and predicted reduction is calculated in Step 4. The obtained value is used to decide whether the step (d_k, w_k) is accepted or not. Iterations with $r_k \geq \rho_0$ are successful iterations, otherwise, the trial steps are rejected. This is a standard procedure in trust region algorithms as described in Section 3.4.

The trust region radius Δ_{k+1} is updated in Step 5. The actual values for τ_1 and τ_2 can be chosen so that condition (4.25) holds. If the ratio r_k of a trial step is greater than ρ_1, then the trust region radius Δ_{k+1} is enlarged. In case a trial step is accepted, then the trust region radius for the next iteration is always set to at least $\Delta_{\min}$. This is motivated by the convergence analysis in the subsequent section. Applying a lower bound $\Delta_{\min}$ on the trust region radius after a successful iteration is also done by Fletcher et al. [43], Kanzow and Zupke [64], and Jiang et al. [63]. The trust region radius is reduced for the next iteration if the trial step is rejected. Consequently, the only difference between subproblem (4.7) in iteration k and subproblem (4.7) in iteration $k+1$ is the reduced trust region radius Δ_{k+1}. The same is true for the subproblems in the feasibility restoration phase. Thus, warm start techniques of a quadratic solver can be applied to improve the performance of an implementation.

In Step 6 the next iterate (x_{k+1}, v_{k+1}) is determined. If the trial step is accepted, i.e., $r_k > \rho_0$, then d_k and w_k are applied and the next iterate is set to $x_{k+1} := x_k + d_k$ and $v_{k+1} := v_k + w_k$. Moreover, the matrix B_{k+1} is updated. Otherwise, the values remain unchanged for the subsequent iteration.

4.2 Convergence Analysis

This section is devoted to the theoretical analysis of Algorithm 4.1. The convergence properties of the algorithm are studied in two steps. The first part investigates the global behavior of the proposed algorithm. Under suitable assumptions it is shown that the algorithm generates a sequence of iterates that has at least one accumulation point that is a KKT point of the optimized problem. The convergence is independent

of the starting point.

The second part of this section concentrates on the local convergence analysis. It is shown that close to a stationary point the algorithm takes full SQP steps and the trust region bound remains inactive. Consequently, fast local convergence is obtained.

4.2.1 Global Convergence

In the beginning the assumptions are stated under which the global convergence of Algorithm 4.1 is proved. Here (x_k, v_k) denotes an iterate and d_k, u_k are obtained either by subproblem (4.7) or by subproblem (4.10) in the feasibility restoration phase.

Assumption 4.2

1. There exists a nonempty, convex, and compact set $\mathcal{X} \subset \mathbb{R}^n$ such that for all k the iterate x_k and $x_k + d_k$ lie in $\mathcal{X}$.
2. The problem functions $f(x)$ and $g_j(x)$, $j = 1, \ldots, m$, are twice continuously differentiable on an open set containing $\mathcal{X}$.
3. For all iterations k the matrix B_k is positive definite and there exists a constant[1] $\kappa_{\mathrm{lbB}} > 0$ independent of k such that
$$\kappa_{\mathrm{lbB}} \|d\|_2^2 \leq d^T B_k d \tag{4.28}$$
for all $d \in \mathbb{R}^n$.
4. The Lagrangian multipliers u_k obtained by the subproblems are bounded for all k and there exists a constant $\kappa \geq 1$ independent of k such that $\kappa \geq \kappa_{\mathrm{lbB}}$ and
$$\|u_k\|_\infty \leq \kappa$$
for all k. The initial guess for the multipliers v_0 is also bounded by κ, that is $\|v_0\|_\infty \leq \kappa$. Moreover, the matrices B_k are bounded and
$$\|B_k\|_2 \leq \kappa$$
holds for all k.

Assumption 4.2(1.) and Assumption 4.2(2.) are required to hold throughout the global convergence analysis without being mentioned explicitly. It is a consequence of these assumptions that the function values, the gradients, and the Hessian matrices of $f(x)$ and $g_j(x)$, $j = 1, \ldots, m$, are bounded on $\mathcal{X}$. Thus, a constant[2] $\kappa_{\mathrm{ubF}} > 0$ exists such that for all $x \in \mathcal{X}$
$$\begin{aligned} |f(x)| &\leq \kappa_{\mathrm{ubF}} \ , \\ |g_j(x)| &\leq \kappa_{\mathrm{ubF}} \ , \quad j = 1, \ldots, m \ , \end{aligned} \tag{4.29}$$
holds, and another constant[3] $\kappa_{\mathrm{ubG}} > 0$ can be determined such that
$$\begin{aligned} \|\nabla f(x)\|_2 &\leq \kappa_{\mathrm{ubG}} \ , \\ \|\nabla g_j(x)\|_2 &\leq \kappa_{\mathrm{ubG}} \ , \quad j = 1, \ldots, m \ , \end{aligned} \tag{4.30}$$

[1] lbB = lower bound B
[2] ubF = upper bound Function
[3] ubG = upper bound Gradient

is satisfied for all $x \in \mathcal{X}$. For the Hessian matrices a constant[4] $\kappa_{\text{ubH}} > 0$ exists such that

$$\begin{aligned} \|\nabla^2 f(x)\|_2 &\leq \kappa_{\text{ubH}} \ , \\ \|\nabla^2 g_j(x)\|_2 &\leq \kappa_{\text{ubH}} \ , \quad j = 1, \ldots, m \ , \end{aligned} \tag{4.31}$$

holds for all $x \in \mathcal{X}$.

The bound κ in Assumption 4.2(4.) is also valid for the multipliers v_k for all k, as the multipliers are updated either by

$$v_j^{(k+1)} := v_j^{(k)}$$

or

$$v_j^{(k+1)} := v_j^{(k)} + w_j^{(k)} = v_j^{(k)} + \left(u_j^{(k)} - v_j^{(k)}\right)\left(1 - z_j^{(k)}\right) \ ,$$

for $j = 1, \ldots, m$, according to Step 6 of Algorithm 4.1, where definition (4.12) of w_k is applied. Since $\|v_0\|_\infty \leq \kappa$ is assumed and $\|z_k\|_\infty \leq 1$, according to the update rule in Step 1 and definition (4.11), the boundedness follows by induction. Thus, the bound on $\|u_k\|_\infty$ implies that

$$\|v_k\|_\infty \leq \kappa \tag{4.32}$$

holds for all k.

To simplify the notation in the remainder of the convergence analysis, it is assumed without loss of generality that the constant $\kappa \geq 1$ also satisfies

$$\kappa \geq \max\left(\kappa_{\text{ubF}}, \kappa_{\text{ubG}}, \kappa_{\text{ubH}}\right) \ , \tag{4.33}$$

where the constants on the right-hand satisfy (4.29)-(4.31). Consequently, the bound κ is valid for the norms of the function values, the gradients, the Hessian matrices, the matrices B_k, and the multipliers.

In order to prove global convergence of Algorithm 4.1 to a KKT point, the extended Mangasarian-Fromowitz constraint qualification (extended MFCQ) is assumed to hold.

Assumption 4.3 1. There exists a $\beta > 0$, $\beta \in \mathbb{R}$, such that the set $\mathcal{F}(\beta)$, as defined in (2.12), is compact, and for all $x \in \mathcal{F}(\beta)$ the vectors $\nabla g_j(x)$, $j \in \mathcal{E}$, are linearly independent and there exists a $d \in \mathbb{R}^n$ such that

$$\begin{aligned} \nabla g_j(x)^T d &= 0 \ , \quad j \in \mathcal{E} \ , \\ \nabla g_j(x)^T d &> 0 \ , \quad j \in \mathcal{A}(x, 0) \ . \end{aligned}$$

2. The set $\mathcal{X}$ of Assumption 4.2 is a subset of $\mathcal{F}(\beta)$, i.e., $\mathcal{X} \subset \mathcal{F}(\beta)$.

Here $\mathcal{F}(\beta)$ denotes the extended feasible region as defined by (2.12) and $\mathcal{A}(x, 0)$ as defined in (2.15) with $\gamma = 0$. Assumption 4.3 ensures that for all k the subproblems set up in x_k will generate a search step d_k that reduces the violation of the linearized constraints.

[4] ubH = upper bound Hessian

For the convergence analysis it is assumed that the subproblems are solved exactly. The Karush-Kuhn-Tucker (KKT) optimality conditions at the minimizer d_k of the subproblems play a key role in the proofs. These conditions correspond to conditions (2.23)-(2.27) in Section 2.2 when applied to the quadratic subproblems. First, the case when subproblem (4.7) is consistent and a solution exists is considered. The solution of problem (4.7) in the k-th iteration is denoted by d_k, u_k, $\underline{\mu}_k$, and $\overline{\mu}_k$, where $\underline{\mu}_k \in \mathbb{R}^n$ and $\overline{\mu}_k \in \mathbb{R}^n$ are the multipliers corresponding to the reformulated trust region constraint. Thus, the KKT optimality conditions of subproblem (4.7) are

$$
\begin{aligned}
&\text{(a)} && B_k d_k + \nabla f(x_k) - \sum_{j=1}^{m} \nabla g_j(x_k) u_j^{(k)} - \underline{\mu}_k + \overline{\mu}_k = 0 \ , && \\
&\text{(b)} && g_j(x_k) + \nabla g_j(x_k)^T d_k = 0 \ , && j \in \mathcal{E} \ , \\
&\text{(c)} && g_j(x_k) + \nabla g_j(x_k)^T d_k \geq 0 \ , && j \in \mathcal{I} \ , \\
&\text{(d)} && d_i^{(k)} - \Delta_k \geq 0 \ , && i = 1, \ldots, n \ , \\
&\text{(e)} && \Delta_k - d_i^{(k)} \geq 0 \ , && i = 1, \ldots, n \ , \\
&\text{(f)} && u_j^{(k)} \left(g_j(x_k) + \nabla g_j(x_k)^T d_k \right) = 0 \ , && j \in \mathcal{I} \ , \\
&\text{(g)} && u_j^{(k)} \geq 0 \ , && j \in \mathcal{I} \ , \\
&\text{(h)} && \underline{\mu}_i^{(k)} \left(d_i^{(k)} - \Delta_k \right) = 0 \ , && i = 1, \ldots, n \ , \\
&\text{(i)} && \underline{\mu}_i^{(k)} \geq 0 \ , && i = 1, \ldots, n \ , \\
&\text{(j)} && \overline{\mu}_i^{(k)} \left(\Delta_k - d_i^{(k)} \right) = 0 \ , && i = 1, \ldots, n \ , \\
&\text{(k)} && \overline{\mu}_i^{(k)} \geq 0 \ , && i = 1, \ldots, n \ .
\end{aligned}
\tag{4.34}
$$

To simplify the notation in the remainder of this work

$$
\mu_k := \sum_{i=1}^{n} \left(\underline{\mu}_i^{(k)} + \overline{\mu}_i^{(k)} \right) \tag{4.35}
$$

is defined, where $\underline{\mu}_k$ and $\overline{\mu}_k$ are the multipliers according to conditions (4.34)(h)-(k). Moreover, these conditions and (4.35) are used to define a vector $\eta_k \in \mathbb{R}^n$ with

$$
\eta_i^{(k)} := \begin{cases} \overline{\mu}_i^{(k)} / \mu_k & , \text{ if } d_i^{(k)} = \Delta_k \text{ and } \mu_k > 0 \ , \\ -\underline{\mu}_i^{(k)} / \mu_k & , \text{ if } d_i^{(k)} = -\Delta_k \text{ and } \mu_k > 0 \ , \\ 0 & , \text{ otherwise} \ , \end{cases} \tag{4.36}
$$

for $i = 1, \ldots, n$.

In case subproblem (4.7) is inconsistent and the feasibility restoration phase is entered, the trial step (d_k, w_k) is obtained by solving subproblem (4.10). δ_k denotes the solution of the feasibility restoration subproblem (4.9). Then the KKT system of

problem (4.10) is

$$
\begin{aligned}
&\text{(a)} && B_k d_k + \nabla f(x_k) - \sum_{j=1}^{m} \nabla g_j(x_k) u_j^{(k)} + \mu_k \eta_k = 0 \ , && \\
&\text{(b)} && g_j(x_k)(1-\delta_k) + \nabla g_j(x_k)^T d_k = 0 \ , && j \in \mathcal{E} \ , \\
&\text{(c)} && g_j(x_k)(1-\delta_k) + \nabla g_j(x_k)^T d_k \geq 0 \ , && j \in \mathcal{A}_k \ , \\
&\text{(d)} && g_j(x_k) + \nabla g_j(x_k)^T d_k \geq 0 \ , && j \in \mathcal{B}_k \ , \\
&\text{(e)} && d_i^{(k)} - \Delta_k \geq 0 \ , && i = 1, \ldots, n \ , \\
&\text{(f)} && \Delta_k - d_i^{(k)} \geq 0 \ , && i = 1, \ldots, n \ , \\
&\text{(g)} && u_j^{(k)} \left(g_j(x_k)(1-\delta_k) + \nabla g_j(x_k)^T d_k \right) = 0 \ , && j \in \mathcal{A}_k \ , \\
&\text{(h)} && u_j^{(k)} \left(g_j(x_k) + \nabla g_j(x_k)^T d_k \right) = 0 \ , && j \in \mathcal{B}_k \ , \\
&\text{(i)} && u_j^{(k)} \geq 0 \ , && j \in \mathcal{I} \ , \\
&\text{(j)} && \underline{\mu}_i^{(k)} \left(d_i^{(k)} - \Delta_k \right) = 0 \ , && i = 1, \ldots, n \ , \\
&\text{(k)} && \underline{\mu}_i^{(k)} \geq 0 \ , && i = 1, \ldots, n \ , \\
&\text{(l)} && \overline{\mu}_i^{(k)} \left(\Delta_k - d_i^{(k)} \right) = 0 \ , && i = 1, \ldots, n \ , \\
&\text{(m)} && \overline{\mu}_i^{(k)} \geq 0 \ , && i = 1, \ldots, n \ .
\end{aligned}
\tag{4.37}
$$

Here μ_k and η_k in (4.37)(a) are determined according to (4.35) and (4.36), where the corresponding values, which satisfy (4.37)(j)-(m), are applied.

The acceptance of a trial step (d_k, w_k) depends on the ratio of the actual change and the predicted reduction calculated in Step 4 of Algorithm 4.1. The actual change in the augmented Lagrangian merit function Φ_{σ_k} is denoted by

$$
Ared_k := \Phi_{\sigma_k}(x_k, v_k) - \Phi_{\sigma_k}(x_k + d_k, v_k + w_k) \ , \tag{4.38}
$$

where the value of the augmented Lagrangian at the trial point $(x_k + d_k, v_k + w_k)$ is defined as

$$
\begin{aligned}
&\Phi_{\sigma_k}(x_k + d_k, v_k + w_k) := f(x_k + d_k) \\
&- \sum_{j \in \mathcal{L}_k} \left(\left(v_j^{(k)} + w_j^{(k)} \right) g_j(x_k + d_k) - \frac{1}{2} \sigma_k g_j(x_k + d_k)^2 \right) - \sum_{j \in \overline{\mathcal{L}}_k} \frac{1}{2} \frac{\left(v_j^{(k)} + w_j^{(k)} \right)^2}{\sigma_k} \ .
\end{aligned}
\tag{4.39}
$$

The index sets $\mathcal{L}_k$ and $\overline{\mathcal{L}}_k$ are defined as follows

$$
\mathcal{L}_k := \mathcal{E} \cup \left\{ j \in \mathcal{I} \mid \ g_j(x_k + d_k) \leq \left(v_j^{(k)} + w_j^{(k)} \right) / \sigma_k \right\} \tag{4.40}
$$

and

$$\overline{\mathcal{L}}_k := \{1, \ldots, m\} \setminus \mathcal{L}_k \ . \tag{4.41}$$

The global convergence of Algorithm 4.1 is proved in several stages. First, it is shown that the step (d_k, w_k) leads to a sufficient decrease in the model, where the step is calculated either by subproblem (4.7) or by subproblem (4.10). Thereafter, this lower bound on the predicted reduction is further investigated. A bound is specified with respect to the value $\|\nabla f(x_k) - \nabla g(x_k)u_k\|_2 + \mu_k$. Moreover, a second estimate is established that depends on the value of the constraint violation. In a next step, the difference between the actual change and the predicted reduction is estimated and a lower bound on the trust region radius is established such that the trust region radius is bounded away from zero if the iterate is not a stationary point of the optimized problem. At the end it is shown that at least one accumulation point of the sequence of iterates generated by Algorithm 4.1 is a KKT point of problem (1.2).

In the beginning some technical results are stated which are applied in the remainder of the global analysis. In the following it is shown that the update rules of Algorithm 4.1 guarantee that the multiplier approximations v_k with respect to the inequality constraints remain greater or equal to zero for all k.

Lemma 4.4 *Let $\{v_k\}$ be the sequence of multipliers generated by Algorithm 4.1, then for all k*

$$v_j^{(k)} \geq 0 \tag{4.42}$$

and

$$v_j^{(k)} + w_j^{(k)} \geq 0 \tag{4.43}$$

hold for all $j \in \mathcal{I}$.

Proof: According to the KKT conditions of the corresponding subproblem, see either condition (4.34)(g) or (4.37)(i), for all k the multipliers satisfies $u_j^{(k)} \geq 0$ for $j \in \mathcal{I}$. In Step 6 of Algorithm 4.1 the multipliers are updated either by

$$v_j^{(k+1)} := v_j^{(k)}$$

or

$$v_j^{(k+1)} := v_j^{(k)} + w_j^{(k)} \ , \tag{4.44}$$

for $j \in \mathcal{E} \cup \mathcal{I}$. In Step 1 the entries of z_k are either set to 0 or determined according to (4.11). In both cases $0 \leq z_j^{(k)} \leq 1$, $j = 1, \ldots, m$, holds. Applying definition (4.12) of $w_j^{(k)}$, $0 \leq z_j^{(k)} \leq 1$, and $u_j^{(k)} \geq 0$, $j \in \mathcal{I}$, yields

$$\begin{aligned} w_j^{(k)} &= \left(u_j^{(k)} - v_j^{(k)}\right)\left(1 - z_j^{(k)}\right) = \underbrace{u_j^{(k)}}_{\geq 0}\underbrace{\left(1 - z_j^{(k)}\right)}_{\geq 0} - v_j^{(k)}\left(1 - z_j^{(k)}\right) \\ &\geq -v_j^{(k)}\underbrace{\left(1 - z_j^{(k)}\right)}_{\leq 1} \geq -\left|v_j^{(k)}\right| \ , \end{aligned} \tag{4.45}$$

for all $j \in \mathcal{I}$. Thus, (4.44) and (4.45) guarantee $v_j^{(k+1)} \geq 0$ as long as $v_j^{(k)} \geq 0$, for $j \in \mathcal{I}$. As $v_j^0 \geq 0$ is required for all $j \in \mathcal{I}$, the lemma follows by induction. □

Since the definitions of the augmented Lagrangian Φ_{σ_k} (4.1) and the model Ψ_{σ_k} (4.13) contain different index sets, the properties of u_k and z_k, that are implied by these sets, are investigated. The following lemma focuses on the set $\overline{\mathcal{M}}_k$.

Lemma 4.5 *Let (x_k, v_k) be an iterate of Algorithm 4.1 and $\overline{\mathcal{M}}_k$ be defined by (4.18), then*

$$u_j^{(k)} = 0 \text{ and } z_j^{(k)} = 0 \tag{4.46}$$

hold for all $j \in \overline{\mathcal{M}}_k$.

Proof: Let $j \in \overline{\mathcal{M}}_k$. By definition of $\mathcal{M}_k$ and $\overline{\mathcal{M}}_k$, cf. (4.17) and (4.18), it follows that $j \in \mathcal{I}$. According to the KKT conditions of the corresponding subproblem at iteration k, i.e., either condition (4.34)(g) or (4.37)(i), the multiplier satisfies $u_j^{(k)} \geq 0$ for all $j \in \mathcal{I}$. From Lemma 4.4 it follows

$$v_j^{(k)} + w_j^{(k)} \geq 0 \ , \tag{4.47}$$

for $j \in \mathcal{I}$. Applying (4.47) and $j \in \overline{\mathcal{M}}_k$, we obtain

$$g_j(x_k) + \nabla g_j(x_k)^T d_k > \frac{v_j^{(k)} + w_j^{(k)}}{\sigma_k} \geq 0 \ . \tag{4.48}$$

If subproblem (4.7) is solved then the KKT condition (4.34)(f) implies $u_j^{(k)} = 0$ and $z_j^{(k)}$ is set to zero in Step 1 of Algorithm 4.1. Otherwise, the feasibility restoration phase is entered.

In case $j \in \mathcal{B}_k$, we obtain with the KKT condition (4.37)(h) and (4.48) that $u_j^{(k)} = 0$ and $z_j^{(k)} = 0$ according to the definition (4.11) of z_k.

If $j \in \mathcal{A}_k$, then with $g_j(x_k) \leq 0$, $0 \leq \delta_k \leq 1$, and (4.48) we get

$$g_j(x_k)(1 - \delta_k) + \nabla g_j(x_k)^T d_k = \underbrace{g_j(x_k) + \nabla g_j(x_k)^T d_k}_{>0} - \underbrace{\delta_k g_j(x_k)}_{\leq 0} > 0 \ .$$

The KKT condition (4.37)(g) implies $u_j^{(k)} = 0$. By definition (4.11) of z_k, it follows $z_j^{(k)} = 0$. This proves the lemma. □

Now some properties with respect to the sets $\overline{\mathcal{S}}_k$ and $\mathcal{M}_k$ are considered.

Lemma 4.6 *Let (x_k, v_k) be an iterate of Algorithm 4.1, then for all $j \in \overline{\mathcal{S}}_k \cap \mathcal{M}_k$*

$$z_j^{(k)} = 0 \ , \tag{4.49}$$

$$g_j(x_k) + \nabla g_j(x_k)^T d_k = 0 \ , \tag{4.50}$$

$$u_j^{(k)} g_j(x_k) > u_j^{(k)} \frac{v_j^{(k)}}{\sigma_k} \geq 0 \tag{4.51}$$

holds, where $\overline{\mathcal{S}}_k$ be defined by (4.3) and $\mathcal{M}_k$ be defined according to (4.18).

Proof: Let $j \in \overline{\mathcal{S}}_k \cap \mathcal{M}_k$. Since $j \in \overline{\mathcal{S}}_k$, it follows by definition (4.3) that $j \in \mathcal{I}$ and

$$g_j(x_k) > \frac{v_j^{(k)}}{\sigma_k} \geq 0 \ , \tag{4.52}$$

where we also applied $v_j^{(k)} \geq 0$ according to Lemma 4.4 and $\sigma_k \geq 1$. Thus, we obtain $j \in \mathcal{B}_k$, with $\mathcal{B}_k = \mathcal{B}(x_k, 0)$ as defined in (2.16). In case the feasibility restoration subproblem (4.10) is solved, then $z_j^{(k)} = 0$ holds according to the definition of z_k, cf. (4.11). If the quadratic subproblem (4.7) is consistent, then $z_j^{(k)}$ is set to zero in Step 1 of Algorithm 4.1 for all $j \in \mathcal{E} \cup \mathcal{I}$. Consequently, $z_j^{(k)} = 0$ holds for all $j \in \overline{\mathcal{S}}_k \cap \mathcal{M}_k$.

Together with the definition (4.12) of $w_j^{(k)}$ this yields

$$v_j^{(k)} + w_j^{(k)} = v_j^{(k)} + \left(u_j^{(k)} - v_j^{(k)}\right)\left(1 - z_j^{(k)}\right) = v_j^{(k)} + \left(u_j^{(k)} - v_j^{(k)}\right) = u_j^{(k)} \ . \tag{4.53}$$

Due to the construction of the subproblems and since $j \in \mathcal{B}_k$, the linearized constraint satisfies

$$g_j(x_k) + \nabla g_j(x_k)^T d_k \geq 0 \ . \tag{4.54}$$

Moreover, $j \in \mathcal{M}_k$ and (4.53) imply

$$g_j(x_k) + \nabla g_j(x_k)^T d_k \leq \frac{v_j^{(k)} + w_j^{(k)}}{\sigma_k} = \frac{u_j^{(k)}}{\sigma_k} \ . \tag{4.55}$$

According to the optimality conditions of the corresponding subproblem, i.e., (4.34)(f) or (4.37)(h), respectively,

$$u_j^{(k)}\left(g_j(x_k) + \nabla g_j(x_k)^T d_k\right) = 0 \tag{4.56}$$

holds.

If we assume that $g_j(x_k) + \nabla g_j(x_k)^T d_k > 0$, then (4.56) requires $u_j^{(k)} = 0$. But this contradicts (4.55) as $\sigma_k \geq 1$. Thus, it follows from (4.54) that

$$g_j(x_k) + \nabla g_j(x_k)^T d_k = 0 \ .$$

When we apply the fact that the multipliers for the inequality constraints satisfy $u_j^{(k)} \geq 0$, $j \in \mathcal{I}$, see KKT conditions (4.34)(g) and (4.37)(i), to (4.52), we obtain the desired result

$$u_j^{(k)} g_j(x_k) > u_j^{(k)} \frac{v_j^{(k)}}{\sigma_k} \geq 0 \ .$$

□

We consider a third set. This one contains the constraints that are in $\mathcal{S}_k$ and $\mathcal{M}_k$.

Lemma 4.7 *Let (x_k, v_k) be an iterate of Algorithm 4.1, then for all $j \in \mathcal{S}_k \cap \mathcal{M}_k$*

$$g_j(x_k) + \nabla g_j(x_k)^T d_k = g_j(x_k) z_j^{(k)} \tag{4.57}$$

holds, where $\mathcal{S}_k$ be defined by (4.2) and $\mathcal{M}_k$ be defined according to (4.18).

Proof: Let $j \in \mathcal{S}_k \cap \mathcal{M}_k$. A distinction is made between equality and inequality constraints. First, the case is investigated when $j \in \mathcal{E}$. If subproblem (4.7) is consistent, then

$$g_j(x_k) + \nabla g_j(x_k)^T d_k = 0 \tag{4.58}$$

and $z_j^{(k)}$ is set to zero in STEP 1 of Algorithm 4.1. With $z_j^{(k)} = 0$, $j \in \mathcal{E}$, $g_j(x_k) z_j^{(k)} = 0$, and (4.58), equation (4.57) follows.

Otherwise, the feasibility restoration phase is entered and subproblem (4.10) is solved so that

$$g_j(x_k)(1 - \delta_k) + \nabla g_j(x_k)^T d_k = 0 \tag{4.59}$$

holds, with $0 \leq \delta_k \leq 1$ as required by the definition of subproblem (4.9). If $g_j(x_k) = 0$, then (4.58) is also satisfied and $z_j^{(k)} = 0$ according to definition (4.11). Equation (4.57) follows as before. Finally, in case $g_j(x_k) \neq 0$, then $z_j^{(k)} = \delta_k$ is obtained by definition (4.11) and (4.59). Thus, equation (4.57) holds for all $j \in \mathcal{E}$.

Now an inequality constraint is considered, i.e., $j \in \mathcal{I}$. As $j \in \mathcal{S}_k \cap \mathcal{M}_k$, it follows by definition (4.12) of $w_j^{(k)}$ and $j \in \mathcal{M}_k$ that

$$g_j(x_k) + \nabla g_j(x_k)^T d_k \leq \frac{v_j^{(k)} + w_j^{(k)}}{\sigma_k} = \frac{v_j^{(k)} + \left(u_j^{(k)} - v_j^{(k)}\right)\left(1 - z_j^{(k)}\right)}{\sigma_k} \ . \tag{4.60}$$

Moreover, $j \in \mathcal{S}_k$ yields

$$g_j(x_k) \leq \frac{v_j^{(k)}}{\sigma_k} \ ,$$

and $g_j(x_k)$ can be less than zero. The rest of the proof depends on the value of $g_j(x_k)$.

We consider the case when $g_j(x_k) > 0$. If subproblem (4.10) is solved in the feasibility restoration phase, then $j \in \mathcal{B}_k$ and $z_j^{(k)} = 0$ follows by definition (4.11). In case the standard subproblem (4.7) is consistent and a solution exists, then $z_j^{(k)} = 0$ is set in STEP 1 of Algorithm 4.1. In both cases, the step in the dual variable is

$$w_j^{(k)} = u_j^{(k)} - v_j^{(k)} \ . \tag{4.61}$$

As $j \in \mathcal{M}_k$, applying (4.61) to (4.60) yields

$$g_j(x_k) + \nabla g_j(x_k)^T d_k \leq \frac{u_j^{(k)}}{\sigma_k} \ . \tag{4.62}$$

Together with the complementary condition in the KKT system of the subproblems, see (4.34)(f) or (4.37)(h), respectively, (4.62) implies

$$g_j(x_k) + \nabla g_j(x_k)^T d_k = 0 \ .$$

Thus, equation (4.57) holds with $z_j^{(k)} = 0$.

If $g_j(x_k) = 0$, then $j \in \mathcal{A}_k$ implies with definition (4.11) that $z_j^{(k)} = 0$ when the feasibility restoration phase is executed. Obviously, this also holds if the standard subproblem (4.7) is solved. Due to (4.61), (4.62), and the complementary condition of the KKT system, cf. (4.34)(f) and (4.37)(g), equation (4.57) holds as before.

Let us now consider the case when $g_j(x_k) < 0$. If subproblem (4.7) is consistent, then we get (4.61) as $z_j^{(k)} = 0$. Thus, (4.62) holds and with the complementary condition (4.34)(f) it follows that $g_j(x_k) + \nabla g_j(x_k)^T d_k = 0$. Consequently, (4.57) holds.

Otherwise, the feasibility restoration phase is executed and subproblem (4.10) is solved. According to the KKT conditions (4.37)(c),

$$g_j(x_k)(1 - \delta_k) + \nabla g_j(x_k)^T d_k \geq 0$$

holds. As $g_j(x_k) < 0$, it follows with $0 \leq \delta_k \leq 1$ that

$$g_j(x_k) + \nabla g_j(x_k)^T d_k \leq g_j(x_k)(1 - \delta_k) + \nabla g_j(x_k)^T d_k \tag{4.63}$$

is satisfied.

First, the case is considered when $g_j(x_k)(1 - \delta_k) + \nabla g_j(x_k)^T d_k = 0$. Here the inequality (4.63) yields

$$g_j(x_k) + \nabla g_j(x_k)^T d_k \leq 0 \ ,$$

and together with $g_j(x_k) < 0$ we obtain

$$z_j^{(k)} = \frac{g_j(x_k) + \nabla g_j(x_k)^T d_k}{g_j(x_k)} \geq 0 \ , \tag{4.64}$$

where $z_j^{(k)}$ is determined according to (4.11). Thus, equation (4.57) holds.

Finally, the situation is considered when $g_j(x_k)(1 - \delta_k) + \nabla g_j(x_k)^T d_k > 0$. In this case the KKT condition (4.37)(g), that is

$$u_j^{(k)} \left(g_j(x_k)(1 - \delta_k) + \nabla g_j(x_k)^T d_k \right) = 0 \ ,$$

implies $u_j^{(k)} = 0$. We assume that $g_j(x_k) + \nabla g_j(x_k)^T d_k > 0$. Then with $g_j(x_k) < 0$, it follows

$$\frac{g_j(x_k) + \nabla g_j(x_k)^T d_k}{g_j(x_k)} < 0 \ . \tag{4.65}$$

This results in $z_j^{(k)} = 0$ according to definition (4.11). Thus, with $u_j^{(k)} = 0$ and $z_j^{(k)} = 0$, the step in the dual variable is $w_j^{(k)} = -v_j^{(k)}$, see definition (4.12). Since $j \in \mathcal{M}_k$, this leads to

$$g_j(x_k) + \nabla g_j(x_k)^T d_k \leq \frac{v_j^{(k)} + w_j^{(k)}}{\sigma_k} = 0 \ , \tag{4.66}$$

what contradicts the assumption that $g_j(x_k) + \nabla g_j(x_k)^T d_k > 0$. It follows $g_j(x_k) + \nabla g_j(x_k)^T d_k \leq 0$ and according to (4.11) the obtained $z_j^{(k)}$ is equal to (4.64). Consequently, statement (4.57) follows. □

The last set under consideration contains the constraints which are in the sets $\overline{\mathcal{L}}_k$ (4.40) and $\mathcal{M}_k$ (4.18). Thus, the actual value of the augmented Lagrangian Φ_{σ_k} at a new trial point $(x_k + d_k, v_k + w_k)$ and the corresponding sets are considered .

Lemma 4.8 *Let (x_k, v_k) be an iterate of Algorithm 4.1, then for all $j \in \overline{\mathcal{L}}_k \cap \mathcal{M}_k$*

$$g_j(x_k) + \nabla g_j(x_k)^T d_k \leq 0 \tag{4.67}$$

holds, where $\overline{\mathcal{L}}_k$ be defined by (4.41) and $\mathcal{M}_k$ be defined according to (4.18).

Proof: Let $j \in \overline{\mathcal{L}}_k \cap \mathcal{M}_k$. As $j \in \overline{\mathcal{L}}_k$, it follows by definition (4.41) that $j \in \mathcal{I}$. We now assume that

$$g_j(x_k) + \nabla g(x_k)^T d_k > 0 \tag{4.68}$$

holds. We consider the two cases where the trial step is either obtained by the standard subproblem (4.7) or by the feasibility restoration phase, i.e., subproblem (4.10).

Let (d_k, u_k) be the solution to the standard subproblem (4.7). As $j \in \mathcal{I}$, it follows with the KKT optimality condition (4.34)(f) and (4.68) that

$$u_j^{(k)} = 0 \ , \tag{4.69}$$

and, consequently,

$$g_j(x_k) + \nabla g(x_k)^T d_k > \frac{v_j^{(k)} + w_j^{(k)}}{\sigma_k} = \frac{v_j^{(k)} + \left(u_j^{(k)} - v_j^{(k)}\right)}{\sigma_k} = \frac{u_j^{(k)}}{\sigma_k} = 0 \ , \tag{4.70}$$

with the definition of $w_j^{(k)}$ (4.12) and $z_j^{(k)} = 0$ according to Step 1. But this contradicts $j \in \mathcal{M}_k$, see definition (4.17). Thus, the inequality (4.67) holds.

Now consider the case when (d_k, u_k) is obtained by solving subproblem (4.10). If $g_j(x_k) > 0$, then $j \in \mathcal{B}_k = \mathcal{B}(x_k, 0)$ follows, see definition (2.16). Due to the KKT condition (4.37)(h) and assumption (4.68), (4.69) holds again for the multiplier $u_j^{(k)}$. Moreover, with $z_j^{(k)} = 0$, according to definition (4.11), and the definition of $w_j^{(k)}$ (4.12), (4.70) also holds. We obtain the same contradiction as before and thus the statement (4.67) follows.

In case $g_j(x_k) \leq 0$, then $j \in \mathcal{A}_k = \mathcal{A}_k(x_k, 0)$, according to definition (2.15), and we get

$$\begin{aligned} g_j(x_k)(1 - \delta_k) + \nabla g_j(x_k)^T d_k &= g_j(x_k) + \nabla g_j(x_k)^T d_k - \underbrace{g_j(x_k)\delta_k}_{\leq 0} \\ &\geq g_j(x_k) + \nabla g_j(x_k)^T d_k \overset{(4.68)}{>} 0 \ , \end{aligned} \tag{4.71}$$

where we applied $g_j(x_k) \leq 0$ and $0 \leq \delta_k \leq 1$. The estimate (4.71) and the KKT condition (4.37)(g) yield (4.69) for the multiplier $u_j^{(k)}$. By definition (4.11), $z_j^{(k)} = 0$ also holds, as $j \in \mathcal{A}_k$ and (4.68) is assumed. Thus, (4.70) follows. This is a contradiction since $j \in \mathcal{M}_k$. As (4.67) holds again, the lemma is proved. □

This was the last technical result needed. The next Theorem establishes a lower bound on the predicted reduction $Pred_k$ (4.23). It is shown that each determined trial step (d_k, w_k) leads to a sufficient decrease in the model. The penalty parameter update (4.24) plays a key role in the proof and is thereby motivated.

Theorem 4.9 *Let Assumption 4.2 hold and (x_k, v_k) be an iterate of Algorithm 4.1. Then the predicted reduction $Pred_k$ (4.23) satisfies*

$$Pred_k \geq \frac{1}{6}\left(d_k^T B_k d_k + 2\mu_k \Delta_k\right) + \frac{1}{8}\sigma_k \sum_{j\in\mathcal{S}_k} g_j(x_k)^2 \left(1 - {z_j^{(k)}}^2\right) \ , \tag{4.72}$$

where $\mathcal{S}_k$ be defined by (4.2).

Proof: The optimality conditions (4.34) and (4.37) of the subproblems are used. To simplify the notation they are reformulated by applying μ_k (4.35) and η_k (4.36). As the KKT conditions (4.37) are equal to (4.34) when $\delta_k = 0$, we do not have to distinguish between the cases when subproblem (4.7) or subproblem (4.10) is solved.

Moreover, the definitions of $\Phi_{\sigma_k}(x_k, v_k)$, $\Psi_{\sigma_k}(d_k, w_k)$, and the corresponding index sets, see (4.1)-(4.3) and (4.16)-(4.18), are applied.

As the model and the augmented Lagrangian at the current iterate (x_k, v_k) are identical, that is $\Psi_{\sigma_k}(0,0) = \Phi_{\sigma_k}(x_k, v_k)$ according to (4.19)-(4.22), we obtain

$$\begin{aligned}
Pred_k &= \Psi_{\sigma_k}(0,0) - \Psi_{\sigma_k}(d_k, w_k) \\
&= \Phi_{\sigma_k}(x_k, v_k) - \Psi_{\sigma_k}(d_k, w_k) \\
&= f(x_k) - \sum_{j\in\mathcal{S}_k}\left(v_j^{(k)} g_j(x_k) - \frac{1}{2}\sigma_k g_j(x_k)^2\right) - \sum_{j\in\overline{\mathcal{S}}_k} \frac{1}{2}\frac{{v_j^{(k)}}^2}{\sigma_k} \\
&\quad - f(x_k) - \nabla f(x_k)^T d_k - \frac{1}{2} d_k^T B_k d_k \\
&\quad + \sum_{j\in\mathcal{M}_k}\left(\left(v_j^{(k)} + w_j^{(k)}\right)\left(g_j(x_k) + \nabla g_j(x_k)^T d_k\right) - \frac{1}{2}\sigma_k\left(g_j(x_k) + \nabla g_j(x_k)^T d_k\right)^2\right) \\
&\quad + \sum_{j\in\overline{\mathcal{M}}_k} \frac{1}{2}\frac{\left(v_j^{(k)} + w_j^{(k)}\right)^2}{\sigma_k} \\
&= \frac{1}{2} d_k^T B_k d_k + \mu_k \eta_k^T d_k - \sum_{j=1}^{m}\left(u_j^{(k)} \nabla g_j(x_k)^T d_k\right) - \sum_{j\in\mathcal{S}_k}\left(v_j^{(k)} g_j(x_k) - \frac{1}{2}\sigma_k g_j(x_k)^2\right) \\
&\quad - \sum_{j\in\overline{\mathcal{S}}_k} \frac{1}{2}\frac{{v_j^{(k)}}^2}{\sigma_k} \\
&\quad + \sum_{j\in\mathcal{M}_k}\left(\left(v_j^{(k)} + w_j^{(k)}\right)\left(g_j(x_k) + \nabla g_j(x_k)^T d_k\right) - \frac{1}{2}\sigma_k\left(g_j(x_k) + \nabla g_j(x_k)^T d_k\right)^2\right) \\
&\quad + \sum_{j\in\overline{\mathcal{M}}_k} \frac{1}{2}\frac{\left(v_j^{(k)} + w_j^{(k)}\right)^2}{\sigma_k} \ .
\end{aligned} \tag{4.73}$$

In the last step $-\nabla f(x_k)^T d_k$ is substituted by the KKT conditions of the subproblem, cf. (4.34)(a), with μ_k and η_k, or (4.37)(a), respectively, which are multiplied by d_k.

Since by definition $\mathcal{S}_k \cup \overline{\mathcal{S}}_k = \{1, \ldots, m\}$ and $\mathcal{M}_k \cup \overline{\mathcal{M}}_k = \{1, \ldots, m\}$, it follows that $(\mathcal{S}_k \cap \mathcal{M}_k) \cup (\overline{\mathcal{S}}_k \cap \mathcal{M}_k) = \mathcal{M}_k$ and $(\mathcal{S}_k \cap \overline{\mathcal{M}}_k) \cup (\overline{\mathcal{S}}_k \cap \overline{\mathcal{M}}_k) = \overline{\mathcal{M}}_k$. Thus, a disjunct decomposition of the set of constraint indexes is obtained by

$$\{1, \ldots, m\} = (\mathcal{S}_k \cap \mathcal{M}_k) \cup (\overline{\mathcal{S}}_k \cap \mathcal{M}_k) \cup (\mathcal{S}_k \cap \overline{\mathcal{M}}_k) \cup (\overline{\mathcal{S}}_k \cap \overline{\mathcal{M}}_k) \ . \tag{4.74}$$

We proceed from (4.73) and apply (4.74). Recombining the index sets of the sums yields

$$\begin{aligned}
Pred_k = \frac{1}{2} d_k^T B_k d_k + \mu_k \eta_k^T d_k \\
- \sum_{j \in \mathcal{S}_k \cap \mathcal{M}_k} \Bigg(& u_j^{(k)} \nabla g_j(x_k)^T d_k + v_j^{(k)} g_j(x_k) - \frac{1}{2} \sigma_k g_j(x_k)^2 \\
& - \left(v_j^{(k)} + w_j^{(k)} \right) \left(g_j(x_k) + \nabla g_j(x_k)^T d_k \right) \\
& + \frac{1}{2} \sigma_k \left(g_j(x_k) + \nabla g_j(x_k)^T d_k \right)^2 \Bigg) \\
- \sum_{j \in \mathcal{S}_k \cap \overline{\mathcal{M}}_k} \Bigg(& u_j^{(k)} \nabla g_j(x_k)^T d_k + v_j^{(k)} g_j(x_k) - \frac{1}{2} \sigma_k g_j(x_k)^2 - \frac{1}{2} \frac{\left(v_j^{(k)} + w_j^{(k)} \right)^2}{\sigma_k} \Bigg) \\
- \sum_{j \in \overline{\mathcal{S}}_k \cap \mathcal{M}_k} \Bigg(& u_j^{(k)} \nabla g_j(x_k)^T d_k + \frac{1}{2} \frac{{v_j^{(k)}}^2}{\sigma_k} - \left(v_j^{(k)} + w_j^{(k)} \right) \left(g_j(x_k) + \nabla g_j(x_k)^T d_k \right) \\
& + \frac{1}{2} \sigma_k \left(g_j(x_k) + \nabla g_j(x_k)^T d_k \right)^2 \Bigg) \\
- \sum_{j \in \overline{\mathcal{S}}_k \cap \overline{\mathcal{M}}_k} \Bigg(& u_j^{(k)} \nabla g_j(x_k)^T d_k + \frac{1}{2} \frac{{v_j^{(k)}}^2}{\sigma_k} - \frac{1}{2} \frac{\left(v_j^{(k)} + w_j^{(k)} \right)^2}{\sigma_k} \Bigg) \ .
\end{aligned} \tag{4.75}$$

In the following the four sums in (4.75) are analyzed separately. In a first step, the constraints with $j \in \mathcal{S}_k \cap \mathcal{M}_k$ are considered. Reordering, eliminating parts, and applying the definition $w_j^{(k)} := \left(u_j^{(k)} - v_j^{(k)} \right) \left(1 - z_j^{(k)} \right)$ leads to

$$\begin{aligned}
& \sum_{j \in \mathcal{S}_k \cap \mathcal{M}_k} \Bigg(u_j^{(k)} \nabla g_j(x_k)^T d_k + v_j^{(k)} g_j(x_k) - \frac{1}{2} \sigma_k g_j(x_k)^2 \\
& \qquad - \left(v_j^{(k)} + w_j^{(k)} \right) \left(g_j(x_k) + \nabla g_j(x_k)^T d_k \right) + \frac{1}{2} \sigma_k \left(g_j(x_k) + \nabla g_j(x_k)^T d_k \right)^2 \Bigg) \\
= & \sum_{j \in \mathcal{S}_k \cap \mathcal{M}_k} \Bigg(\left(u_j^{(k)} - v_j^{(k)} \right) \nabla g_j(x_k)^T d_k - \left(u_j^{(k)} - v_j^{(k)} \right) \left(1 - z_j^{(k)} \right) \left(g_j(x_k) + \nabla g_j(x_k)^T d_k \right) \\
& \qquad - \frac{1}{2} \sigma_k g_j(x_k)^2 + \frac{1}{2} \sigma_k \left(g_j(x_k) + \nabla g_j(x_k)^T d_k \right)^2 \Bigg) \ .
\end{aligned} \tag{4.76}$$

Now the second sum in (4.75) with $j \in \mathcal{S}_k \cap \overline{\mathcal{M}}_k$ is considered. As $j \in \overline{\mathcal{M}}_k$, according to Lemma 4.5 it follows $u_j^{(k)} = 0$ and $z_j^{(k)} = 0$. Thus, the step in the dual variable for $j \in \mathcal{S}_k \cap \overline{\mathcal{M}}_k$ is

$$w_j^{(k)} = -v_j^{(k)} \ , \tag{4.77}$$

and with $u_j^{(k)} = 0$ the equation

$$u_j^{(k)} \nabla g_j(x_k)^T d_k = -u_j^{(k)} g_j(x_k) \tag{4.78}$$

holds. Applying (4.77) and (4.78) to the sum yields

$$\begin{aligned} &\sum_{j \in \mathcal{S}_k \cap \overline{\mathcal{M}}_k} \left(\underbrace{u_j^{(k)} \nabla g_j(x_k)^T d_k}_{=-u_j^{(k)} g_j(x_k)} + v_j^{(k)} g_j(x_k) - \frac{1}{2} \sigma_k g_j(x_k)^2 - \frac{1}{2} \underbrace{\frac{\left(v_j^{(k)} + w_j^{(k)}\right)^2}{\sigma_k}}_{=0} \right) \\ = &- \sum_{j \in \mathcal{S}_k \cap \overline{\mathcal{M}}_k} \left(\left(u_j^{(k)} - v_j^{(k)}\right) g_j(x_k) + \frac{1}{2} \sigma_k g_j(x_k)^2 \right) \ . \end{aligned} \tag{4.79}$$

The third sum in (4.75) consists of all $j \in \overline{\mathcal{S}}_k \cap \mathcal{M}_k$, and according to Lemma 4.6

$$g_j(x_k) = -\nabla g_j(x_k)^T d_k \tag{4.80}$$

and

$$u_j^{(k)} g_j(x_k) > u_j^{(k)} \frac{v_j^{(k)}}{\sigma_k} \geq 0 \tag{4.81}$$

hold. Applying (4.80) and (4.81) to the sum leads to the estimate

$$\begin{aligned} &- \sum_{j \in \overline{\mathcal{S}}_k \cap \mathcal{M}_k} \left(u_j^{(k)} \nabla g_j(x_k)^T d_k + \frac{1}{2} \frac{{v_j^{(k)}}^2}{\sigma_k} - \left(v_j^{(k)} + w_j^{(k)}\right) \underbrace{\left(g_j(x_k) + \nabla g_j(x_k)^T d_k\right)}_{=0} \right. \\ &\qquad \left. + \frac{1}{2} \sigma_k \underbrace{\left(g_j(x_k) + \nabla g_j(x_k)^T d_k\right)^2}_{=0} \right) \\ = &\sum_{j \in \overline{\mathcal{S}}_k \cap \mathcal{M}_k} \left(u_j^{(k)} g_j(x_k) - \frac{1}{2} \frac{{v_j^{(k)}}^2}{\sigma_k} \right) \\ \overset{(4.81)}{>} &\sum_{j \in \overline{\mathcal{S}}_k \cap \mathcal{M}_k} \left(u_j^{(k)} \frac{v_j^{(k)}}{\sigma_k} - \frac{1}{2} \frac{{v_j^{(k)}}^2}{\sigma_k} \right) \ . \end{aligned} \tag{4.82}$$

Since $j \in \overline{\mathcal{M}}_k$, Lemma 4.5 can be applied to transform the last sum in (4.75). For $j \in \overline{\mathcal{M}}_k$, $u_j^{(k)} = 0$ and $z_j^{(k)} = 0$ hold. Thus, it follows

$$v_j^{(k)} + w_j^{(k)} = v_j^{(k)} + \left(u_j^{(k)} - v_j^{(k)}\right)\left(1 - z_j^{(k)}\right) = u_j^{(k)} = 0 \ . \tag{4.83}$$

With (4.83) we obtain

$$\sum_{j\in\overline{\mathcal{S}}_k\cap\overline{\mathcal{M}}_k}\left(\underbrace{u_j^{(k)}\nabla g_j(x_k)^T d_k}_{=0}+\frac{1}{2}\frac{{v_j^{(k)}}^2}{\sigma_k}-\frac{1}{2}\underbrace{\frac{\left(v_j^{(k)}+w_j^{(k)}\right)^2}{\sigma_k}}_{=0}\right)=\sum_{j\in\overline{\mathcal{S}}_k\cap\overline{\mathcal{M}}_k}\frac{1}{2}\frac{{v_j^{(k)}}^2}{\sigma_k}$$

$$\overset{u_j^{(k)}=0}{=}\sum_{j\in\overline{\mathcal{S}}_k\cap\overline{\mathcal{M}}_k}\frac{1}{2}\frac{\left(u_j^{(k)}-v_j^{(k)}\right)^2}{\sigma_k}\ . \tag{4.84}$$

Now the sums in (4.75) are substituted by (4.76), (4.79), (4.82), and (4.84). In (4.82) a lower bound on the third sum is established, thus, the predicted reduction $Pred_k$ is also estimated from below. In a second step, $g_j(x_k)+\nabla g_j(x_k)^T d_k = g_j(x_k)z_j^{(k)}$, for all $j\in\mathcal{S}_k\cap\mathcal{M}_k$, is applied according to Lemma 4.7. It follows

$$\begin{aligned}
Pred_k > \ & \frac{1}{2}d_k^T B_k d_k+\mu_k\eta_k^T d_k\\
& -\sum_{j\in\mathcal{S}_k\cap\mathcal{M}_k}\Bigg(\left(u_j^{(k)}-v_j^{(k)}\right)\nabla g_j(x_k)^T d_k\\
& \qquad -\left(u_j^{(k)}-v_j^{(k)}\right)\left(1-z_j^{(k)}\right)\left(g_j(x_k)+\nabla g_j(x_k)^T d_k\right)\\
& \qquad -\frac{1}{2}\sigma_k g_j(x_k)^2+\frac{1}{2}\sigma_k\left(g_j(x_k)+\nabla g_j(x_k)^T d_k\right)^2\Bigg)\\
& +\sum_{j\in\mathcal{S}_k\cap\overline{\mathcal{M}}_k}\left(\left(u_j^{(k)}-v_j^{(k)}\right)g_j(x_k)+\frac{1}{2}\sigma_k g_j(x_k)^2\right)\\
& +\sum_{j\in\overline{\mathcal{S}}_k\cap\mathcal{M}_k}\left(u_j^{(k)}\frac{v_j^{(k)}}{\sigma_k}-\frac{1}{2}\frac{{v_j^{(k)}}^2}{\sigma_k}\right)\\
& -\sum_{j\in\overline{\mathcal{S}}_k\cap\overline{\mathcal{M}}_k}\frac{1}{2}\frac{\left(u_j^{(k)}-v_j^{(k)}\right)^2}{\sigma_k}\\
= \ & \frac{1}{2}d_k^T B_k d_k+\mu_k\eta_k^T d_k\\
& +\sum_{j\in\mathcal{S}_k\cap\mathcal{M}_k}\Bigg(\left(u_j^{(k)}-v_j^{(k)}\right)\left(1-z_j^{(k)}\right)g_j(x_k)+\left(u_j^{(k)}-v_j^{(k)}\right)\left(1-z_j^{(k)}\right)g_j(x_k)z_j^{(k)}\\
& \qquad +\frac{1}{2}\sigma_k g_j(x_k)^2-\frac{1}{2}\sigma_k\left(g_j(x_k)z_j^{(k)}\right)^2\Bigg)\\
& +\sum_{j\in\mathcal{S}_k\cap\overline{\mathcal{M}}_k}\left(\left(u_j^{(k)}-v_j^{(k)}\right)g_j(x_k)+\frac{1}{2}\sigma_k g_j(x_k)^2\right)
\end{aligned}$$

$$
\begin{aligned}
&+ \sum_{j \in \overline{\mathcal{S}}_k \cap \mathcal{M}_k} \left(u_j^{(k)} \frac{v_j^{(k)}}{\sigma_k} - \frac{1}{2} \frac{{v_j^{(k)}}^2}{\sigma_k} \right) - \sum_{j \in \overline{\mathcal{S}}_k \cap \overline{\mathcal{M}}_k} \frac{1}{2} \frac{\left(u_j^{(k)} - v_j^{(k)} \right)^2}{\sigma_k} \qquad \text{(cf. Lemma 4.7)} \\
= \; & \frac{1}{2} d_k^T B_k d_k + \mu_k \eta_k^T d_k \\
&+ \sum_{j \in \mathcal{S}_k \cap \mathcal{M}_k} \Big(\left(u_j^{(k)} - v_j^{(k)} \right) \left(1 - z_j^{(k)} \right) g_j(x_k) + \left(u_j^{(k)} - v_j^{(k)} \right) \left(1 - z_j^{(k)} \right) g_j(x_k) z_j^{(k)} \\
&\qquad + \frac{1}{2} \sigma_k g_j(x_k)^2 \left(1 - {z_j^{(k)}}^2 \right) \Big) \\
&+ \sum_{j \in \mathcal{S}_k \cap \overline{\mathcal{M}}_k} \left(\left(u_j^{(k)} - v_j^{(k)} \right) g_j(x_k) + \frac{1}{2} \sigma_k g_j(x_k)^2 \right) \\
&+ \sum_{j \in \overline{\mathcal{S}}_k \cap \mathcal{M}_k} \left(\frac{1}{2} \frac{{u_j^{(k)}}^2}{\sigma_k} - \frac{1}{2} \frac{{u_j^{(k)}}^2}{\sigma_k} + u_j^{(k)} \frac{v_j^{(k)}}{\sigma_k} - \frac{1}{2} \frac{{v_j^{(k)}}^2}{\sigma_k} \right) - \sum_{j \in \overline{\mathcal{S}}_k \cap \overline{\mathcal{M}}_k} \frac{1}{2} \frac{\left(u_j^{(k)} - v_j^{(k)} \right)^2}{\sigma_k} \\
= \; & \frac{1}{2} d_k^T B_k d_k + \mu_k \eta_k^T d_k \\
&+ \sum_{j \in \mathcal{S}_k \cap \mathcal{M}_k} \left(\left(u_j^{(k)} - v_j^{(k)} \right) \left(1 - z_j^{(k)} \right) \left(1 + z_j^{(k)} \right) g_j(x_k) + \frac{1}{2} \sigma_k g_j(x_k)^2 \left(1 - {z_j^{(k)}}^2 \right) \right) \\
&+ \sum_{j \in \mathcal{S}_k \cap \overline{\mathcal{M}}_k} \left(\left(u_j^{(k)} - v_j^{(k)} \right) g_j(x_k) + \frac{1}{2} \sigma_k g_j(x_k)^2 \right) \\
&+ \sum_{j \in \overline{\mathcal{S}}_k \cap \mathcal{M}_k} \frac{1}{2} \frac{{u_j^{(k)}}^2}{\sigma_k} - \sum_{j \in \overline{\mathcal{S}}_k} \frac{1}{2} \frac{\left(u_j^{(k)} - v_j^{(k)} \right)^2}{\sigma_k} \; . \qquad (4.85)
\end{aligned}
$$

In the last step we applied $\mathcal{M}_k \cup \overline{\mathcal{M}}_k = \{1, \ldots, m\}$. Since $j \in \overline{\mathcal{S}}_k \cap \mathcal{M}_k$, it follows $j \in \mathcal{I}$. Due to the KKT conditions of the subproblems, i.e., (4.34)(g) or (4.37)(i), respectively, we know that $u_j^{(k)} \geq 0$ holds for all $j \in \mathcal{I}$. With $\sigma_k \geq 1$, we get

$$
\sum_{j \in \overline{\mathcal{S}}_k \cap \mathcal{M}_k} \frac{1}{2} \frac{{u_j^{(k)}}^2}{\sigma_k} \geq 0 \; . \qquad (4.86)
$$

Applying (4.86) to (4.85) yields

$$
\begin{aligned}
Pred_k > \; & \frac{1}{2} d_k^T B_k d_k + \mu_k \eta_k^T d_k \\
&+ \sum_{j \in \mathcal{S}_k \cap \mathcal{M}_k} \left(\left(u_j^{(k)} - v_j^{(k)} \right) \left(1 - {z_j^{(k)}}^2 \right) g_j(x_k) + \frac{1}{2} \sigma_k g_j(x_k)^2 \left(1 - {z_j^{(k)}}^2 \right) \right) \\
&+ \sum_{j \in \mathcal{S}_k \cap \overline{\mathcal{M}}_k} \left(\left(u_j^{(k)} - v_j^{(k)} \right) g_j(x_k) + \frac{1}{2} \sigma_k g_j(x_k)^2 \right) - \sum_{j \in \overline{\mathcal{S}}_k} \frac{1}{2} \frac{\left(u_j^{(k)} - v_j^{(k)} \right)^2}{\sigma_k}
\end{aligned}
$$

$$= \frac{1}{2} d_k^T B_k d_k + \mu_k \eta_k^T d_k + \sum_{j \in \mathcal{S}_k} \left(1 - z_j^{(k)^2}\right) \left(\left(u_j^{(k)} - v_j^{(k)}\right) g_j(x_k) + \frac{1}{2}\sigma_k g_j(x_k)^2\right)$$
$$- \sum_{j \in \overline{\mathcal{S}}_k} \frac{1}{2} \frac{\left(u_j^{(k)} - v_j^{(k)}\right)^2}{\sigma_k} . \qquad \text{(cf. Lemma 4.5)}$$

In the last step $z_j^{(k)} = 0$ is applied for all $j \in \overline{\mathcal{M}}_k$ according to Lemma 4.5, and again $\mathcal{M}_k \cup \overline{\mathcal{M}}_k = \{1, \ldots, m\}$.

Now the penalty parameter σ_k is replaced by estimates obtained by the penalty update (4.24), that is for each $j \in \mathcal{E} \cup \mathcal{I}$ the specific value on the right-hand side of

$$\sigma_k \geq \max_{1 \leq j \leq m} \left(\frac{2m \left(u_j^{(k)} - v_j^{(k)}\right)^2}{d_k^T B_k d_k + 2\mu_k \Delta_k} \left(1 - z_j^{(k)^2}\right) \right) \tag{4.87}$$

is inserted, where $d_k^T B_k d_k > 0$ for $\|d_k\|_\infty > 0$ due to Assumption 4.2(3.).

Moreover, we make use of the fact that $z_j^{(k)} = 0$, for all $j \in \overline{\mathcal{S}}_k$, since $g_j(x_k) > 0$ implies $j \in \mathcal{B}_k$. This follows directly from the way z_k is obtained, cf. (4.11), the definition of $\overline{\mathcal{S}}_k$ (4.3), and the definition of $\mathcal{B}_k$, see (2.16) with $\gamma = 0$.

Note that $\eta_k^T d_k = \Delta_k$ if $\|d_k\|_\infty = \Delta_k$. Otherwise, $\mu_k = 0$ holds. This is implied by the definitions of μ_k (4.35) and η_k (4.36). Thus, we can substitute $\mu_k \eta_k^T d_k$ by $\mu_k \Delta_k$.

All indexes $j \in \overline{\mathcal{S}}_k$ with $u_j^{(k)} - v_j^{(k)} = 0$ do not influence the further investigation since they vanish. Without loss of generality we assume that $u_j^{(k)} - v_j^{(k)} \neq 0$ for all $j \in \overline{\mathcal{S}}_k$. We continue and obtain

$$Pred_k > \frac{1}{2} d_k^T B_k d_k + \mu_k \Delta_k + \sum_{j \in \mathcal{S}_k} \left(1 - z_j^{(k)^2}\right) \left(\left(u_j^{(k)} - v_j^{(k)}\right) g_j(x_k) + \frac{1}{2}\sigma_k g_j(x_k)^2\right)$$
$$- \sum_{j \in \overline{\mathcal{S}}_k} \frac{1}{2} \frac{\left(u_j^{(k)} - v_j^{(k)}\right)^2}{\sigma_k}$$

$$\overset{(4.87)}{\geq} \frac{1}{2} d_k^T B_k d_k + \mu_k \Delta_k$$
$$+ \sum_{j \in \mathcal{S}_k} \left(1 - z_j^{(k)^2}\right) \left(\left(u_j^{(k)} - v_j^{(k)}\right) g_j(x_k) + \frac{3}{8} \frac{2m \left(u_j^{(k)} - v_j^{(k)}\right)^2 \left(1 - z_j^{(k)^2}\right)}{d_k^T B_k d_k + 2\mu_k \Delta_k} g_j(x_k)^2 \right)$$
$$+ \frac{1}{8}\sigma_k \sum_{j \in \mathcal{S}_k} g_j(x_k)^2 \left(1 - z_j^{(k)^2}\right)$$
$$- \sum_{j \in \overline{\mathcal{S}}_k} \frac{1}{2} \frac{\left(u_j^{(k)} - v_j^{(k)}\right)^2 \left(d_k^T B_k d_k + 2\mu_k \Delta_k\right)}{2m \left(u_j^{(k)} - v_j^{(k)}\right)^2}$$

$$
\begin{aligned}
&= \frac{1}{2} d_k^T B_k d_k + \mu_k \Delta_k + \frac{1}{8} \sigma_k \sum_{j \in \mathcal{S}_k} g_j(x_k)^2 \left(1 - z_j^{(k)^2}\right) \\
&\quad - \sum_{j \in \overline{\mathcal{S}}_k} \frac{1}{4} \frac{d_k^T B_k d_k + 2\mu_k \Delta_k}{m} \\
&\quad + \sum_{j \in \mathcal{S}_k} \left(\frac{1}{3} \frac{d_k^T B_k d_k + 2\mu_k \Delta_k}{m} + \left(u_j^{(k)} - v_j^{(k)}\right) g_j(x_k) \left(1 - z_j^{(k)^2}\right) \right. \\
&\qquad \left. + \frac{3}{4} \frac{m \left(u_j^{(k)} - v_j^{(k)}\right)^2 g_j(x_k)^2 \left(1 - z_j^{(k)^2}\right)^2}{d_k^T B_k d_k + 2\mu_k \Delta_k} \right) \\
&\quad - \sum_{j \in \mathcal{S}_k} \frac{1}{3} \frac{d_k^T B_k d_k + 2\mu_k \Delta_k}{m} \\
&= \frac{1}{2} d_k^T B_k d_k + \mu_k \Delta_k + \frac{1}{8} \sigma_k \sum_{j \in \mathcal{S}_k} g_j(x_k)^2 \left(1 - z_j^{(k)^2}\right) \\
&\quad - \sum_{j \in \overline{\mathcal{S}}_k} \frac{1}{4} \frac{d_k^T B_k d_k + 2\mu_k \Delta_k}{m} - \sum_{j \in \mathcal{S}_k} \frac{1}{3} \frac{d_k^T B_k d_k + 2\mu_k \Delta_k}{m} \\
&\quad + \sum_{j \in \mathcal{S}_k} \underbrace{\left(\frac{\sqrt{d_k^T B_k d_k + 2\mu_k \Delta_k}}{\sqrt{3m}} + \frac{\sqrt{3m} \left(u_j^{(k)} - v_j^{(k)}\right) g_j(x_k) \left(1 - z_j^{(k)^2}\right)}{2\sqrt{d_k^T B_k d_k + 2\mu_k \Delta_k}} \right)^2}_{\geq 0} \\
&\geq \frac{1}{2} d_k^T B_k d_k + \mu_k \Delta_k + \frac{1}{8} \sigma_k \sum_{j \in \mathcal{S}_k} g_j(x_k)^2 \left(1 - z_j^{(k)^2}\right) - \frac{1}{3} m \frac{d_k^T B_k d_k + 2\mu_k \Delta_k}{m} \\
&= \frac{1}{6} \left(d_k^T B_k d_k + 2\mu_k \Delta_k\right) + \frac{1}{8} \sigma_k \sum_{j \in \mathcal{S}_k} g_j(x_k)^2 \left(1 - z_j^{(k)^2}\right) \ .
\end{aligned}
$$

The last inequality is obtained by eliminating the sum of squared expressions, applying

$$
\sum_{j \in \overline{\mathcal{S}}_k} \frac{1}{3} \frac{d_k^T B_k d_k + 2\mu_k \Delta_k}{m} \geq \sum_{j \in \overline{\mathcal{S}}_k} \frac{1}{4} \frac{d_k^T B_k d_k + 2\mu_k \Delta_k}{m} \geq 0 \ ,
$$

where $d_k^T B_k d_k + 2\mu_k \Delta_k > 0$, and making use of $\mathcal{S}_k \cup \overline{\mathcal{S}}_k = \{1, \ldots, m\}$. This proves the theorem. □

The next theorem establishes the dependency of the predicted reduction on the trust region radius Δ_k. Later it is used to contradict the assumption that Δ_k tends to zero in case the sequence generated by Algorithm 4.1 is not approaching any KKT point.

Theorem 4.10 *Let Assumption 4.2 hold with a constant $\kappa \geq 1$. Let (x_k, v_k) be an iterate of Algorithm 4.1 and (d_k, u_k, μ_k) be the solution to either subproblem (4.7) or subproblem (4.10) in case the feasibility restoration phase is entered. Then there exists a constant $c_1 > 0$ independent of k such that*

$$Pred_k \geq c_1 \left(\|\nabla f(x_k) - \nabla g(x_k) u_k\|_2 \min\left(\Delta_k, \frac{\|\nabla f(x_k) - \nabla g(x_k) u_k\|_2}{\kappa} \right) + \mu_k \Delta_k \right) \tag{4.88}$$

holds. Moreover, the step d_k satisfies

$$\|d_k\|_\infty \geq \min\left(\Delta_k, \frac{\|\nabla f(x_k) - \nabla g(x_k) u_k\|_2}{\sqrt{n}\kappa} \right) .$$

Proof: Theorem 4.9 establishes the lower bound (4.72) on the predicted reduction $Pred_k$ (4.23). Applying Assumption 4.2(3.), i.e., $d_k^T B_k d_k \geq \kappa_{\text{lbB}} \|d_k\|_2$, to (4.72) yields

$$\begin{aligned} Pred_k &\geq \frac{1}{6}\left(d_k^T B_k d_k + 2\mu_k \Delta_k \right) + \frac{1}{8}\sigma_k \sum_{j \in \mathcal{S}_k} g_j(x_k)^2 \left(1 - z_j^{(k)^2} \right) \\ &\geq \frac{1}{6}\left(d_k^T B_k d_k + 2\mu_k \Delta_k \right) \\ &\geq \frac{1}{6}\left(\kappa_{\text{lbB}} \|d_k\|_2^2 + 2\mu_k \Delta_k \right) . \end{aligned} \tag{4.89}$$

The fact that $0 \leq z_j^{(k)} \leq 1$, for all $j \in \mathcal{E} \cup \mathcal{I}$, and the resulting non-negativity of the term containing the penalty parameter σ_k has been applied to obtain (4.89).

Moreover, as $\|d_k\|_2 \geq \|d_k\|_\infty$, we also obtain

$$\begin{aligned} Pred_k &\geq \frac{1}{6}\left(\kappa_{\text{lbB}} \|d_k\|_2^2 + 2\mu_k \Delta_k \right) \\ &\geq \frac{1}{6}\left(\kappa_{\text{lbB}} \|d_k\|_2 \|d_k\|_\infty + 2\mu_k \Delta_k \right) . \end{aligned} \tag{4.90}$$

According to the optimality conditions of the corresponding subproblem, i.e., (4.34)(a) or (4.37)(a), respectively, d_k, u_k, μ_k, and η_k satisfy

$$B_k d_k + \nabla f(x_k) - \nabla g(x_k) u_k + \mu_k \eta_k = 0 , \tag{4.91}$$

where $\mu_k \geq 0$ is determined according to (4.35) and η_k is defined by (4.36). By definition (4.36), it follows $\|\eta_k\|_1 \leq 1$. With (4.91), the upper bound κ on $\|B_k\|_2$, cf. Assumption 4.2(4.), and $1 \geq \|\eta_k\|_1 \geq \|\eta_k\|_2$, we obtain the following estimate

$$\begin{aligned} \kappa \|d_k\|_2 &\geq \|B_k\|_2 \|d_k\|_2 \\ &\geq \|B_k d_k\|_2 \end{aligned}$$

$$
\begin{aligned}
&= \|\nabla f(x_k) - \nabla g(x_k)u_k + \mu_k \eta_k\|_2 \\
&\geq \|\nabla f(x_k) - \nabla g(x_k)u_k\|_2 - \mu_k \|\eta_k\|_2 \\
&\geq \|\nabla f(x_k) - \nabla g(x_k)u_k\|_2 - \mu_k \|\eta_k\|_1 \\
&\geq \|\nabla f(x_k) - \nabla g(x_k)u_k\|_2 - \mu_k \ .
\end{aligned}
$$

Dividing by the constant κ yields

$$
\|d_k\|_2 \geq \frac{\|\nabla f(x_k) - \nabla g(x_k)u_k\|_2}{\kappa} - \frac{\mu_k}{\kappa} \ . \tag{4.92}
$$

We apply (4.92) to (4.90). This leads to

$$
\begin{aligned}
Pred_k \quad &\geq \quad \frac{1}{6}\kappa_{\mathrm{lbB}} \left(\frac{\|\nabla f(x_k) - \nabla g(x_k)u_k\|_2}{\kappa} - \frac{\mu_k}{\kappa} \right) \|d_k\|_\infty + \frac{2}{6}\mu_k \Delta_k \\
&\overset{\|d_k\|_\infty \leq \Delta_k}{\geq} \quad \frac{1}{6}\frac{\kappa_{\mathrm{lbB}}}{\kappa} \|\nabla f(x_k) - \nabla g(x_k)u_k\|_2 \|d_k\|_\infty - \frac{1}{6}\frac{\kappa_{\mathrm{lbB}}}{\kappa}\mu_k \Delta_k + \frac{2}{6}\mu_k \Delta_k \\
&\overset{\frac{\kappa_{\mathrm{lbB}}}{\kappa} \leq 1}{\geq} \quad \frac{1}{6}\frac{\kappa_{\mathrm{lbB}}}{\kappa} \|\nabla f(x_k) - \nabla g(x_k)u_k\|_2 \|d_k\|_\infty + \frac{1}{6}\frac{\kappa_{\mathrm{lbB}}}{\kappa}\mu_k \Delta_k \ .
\end{aligned} \tag{4.93}
$$

Consider the case when $\|d_k\|_\infty = \Delta_k$, then (4.93) can be rewritten as

$$
Pred_k \quad \geq \quad \frac{1}{6}\frac{\kappa_{\mathrm{lbB}}}{\kappa} \|\nabla f(x_k) - \nabla g(x_k)u_k\|_2 \Delta_k + \frac{1}{6}\frac{\kappa_{\mathrm{lbB}}}{\kappa}\mu_k \Delta_k \ . \tag{4.94}
$$

In case of $\|d_k\|_\infty < \Delta_k$, the optimality conditions (4.34) and (4.37), respectively, yield $\mu_k = 0$. Thus, we conclude from (4.89) and (4.92) that

$$
Pred_k \quad \geq \quad \frac{1}{6}\frac{\kappa_{\mathrm{lbB}}}{\kappa} \frac{\|\nabla f(x_k) - \nabla g(x_k)u_k\|_2^2}{\kappa} \tag{4.95}
$$

holds. Thus, (4.88) results from (4.94) and (4.95), where we set $c_1 := \kappa_{\mathrm{lbB}}/(6\kappa)$.

Applying $\sqrt{n}\|d_k\|_\infty \geq \|d_k\|_2$ to (4.92), the size of the trial step d_k can be estimated by

$$
\|d_k\|_\infty \geq \frac{\|d_k\|_2}{\sqrt{n}} \geq \frac{\|\nabla f(x_k) - \nabla g(x_k)u_k\|_2}{\sqrt{n}\kappa} - \frac{\mu_k}{\sqrt{n}\kappa} \ .
$$

The rest of the theorem follows as $\mu_k = 0$ if $\|d_k\|_\infty < \Delta_k$, and otherwise $\|d_k\|_\infty = \Delta_k$. □

The following theorems are taken from Spellucci [111], where they are formulated as Theorem 3.4 and Theorem 3.5, respectively. The statements of the theorems are used in the global analysis to establish convergence toward feasible points. Both theorems require that the extended Mangasarian-Fromowitz constraint qualification (extended MFCQ), see Definition 2.3, holds. Proofs for the theorems can be found in the textbook by Spellucci [110].

Theorem 4.11 *If Assumption 4.3(1.) holds, then there exists a pair $\epsilon > 0$ and $\bar{\gamma} > 0$ such that for all γ with $0 \leq \gamma \leq \bar{\gamma}$, for all $\widetilde{x} \in \mathcal{F}(\beta)$, and for any bounded function $b : \mathcal{N}_\epsilon(\widetilde{x}) \to \mathbb{R}^{m_e}$, there exists a bounded function $d : \mathcal{N}_\epsilon(\widetilde{x}) \to \mathbb{R}^n$ such that*

$$\begin{aligned} \nabla g_j(x)^T d(x) &= b_j(x) \ , \quad && j \in \mathcal{E} \ , \\ \nabla g_j(x)^T d(x) &\geq 1 \ , && j \in \mathcal{A}(\widetilde{x}, \gamma) \ , \end{aligned}$$

holds for all $x \in \mathcal{N}_\epsilon(\widetilde{x})$, where $\mathcal{N}_\epsilon(\widetilde{x})$ is defined by (2.19). □

The second theorem applies Theorem 4.11 and will be used to estimate an upper bound on the relaxation parameter δ_k in the feasibility restoration subproblem (4.9).

Theorem 4.12 *If Assumption 4.3(1.) holds, then there exist $\theta^\star \in (0, 1]$, $\nu^\star > 0$, and $\Delta^\star > 0$ such that for all $x \in \mathcal{F}(\beta)$ and all $0 < \theta \leq \theta^\star$ there exists a vector $d \neq 0$, $d \in R^n$, satisfying*

$$\begin{aligned} \theta g_j(x) + \nabla g_j(x)^T d &= 0 \ , \quad && j \in \mathcal{E} \ , \\ \theta g_j(x) + \nabla g_j(x)^T d &\geq \nu^\star \ , && j \in \mathcal{A}(x, 0) \ , \\ g_j(x) + \nabla g_j(x)^T d &\geq \nu^\star \ , && j \in \mathcal{B}(x, 0) \ , \\ \|d\|_\infty &\leq \Delta^\star \ , \end{aligned}$$

where $\mathcal{A}(x, 0)$ and $\mathcal{B}(x, 0)$ are defined by (2.15) and (2.16), respectively. □

A slightly modified theorem can be derived from Theorem 4.12. It says that in each iteration k a bounded step s_k exists which reduces the constraint violation in the linear approximation to a fixed fraction of the current value. Thus, there always exists a step that results in a sufficient decrease with respect to the constraint violation measurement.

Theorem 4.13 *Let (x_k, v_k) be an iterate of Algorithm 4.1. If Assumption 4.2 and Assumption 4.3 hold, then there exists a vector $s_k \neq 0$, $s_k \in \mathbb{R}^n$, such that*

$$\begin{aligned} \theta^\star g_j(x_k) + \nabla g_j(x_k)^T s_k &= 0 \ , \quad && j \in \mathcal{E} \ , \\ \theta^\star g_j(x_k) + \nabla g_j(x_k)^T s_k &\geq 0 \ , && j \in \mathcal{A}_k \ , \\ g_j(x_k) + \nabla g_j(x_k)^T s_k &\geq 0 \ , && j \in \mathcal{B}_k \ , \\ \|s_k\|_\infty &\leq \Delta^\star \end{aligned}$$

holds with constants $\theta^\star \in (0, 1]$ and $\Delta^\star > 0$ which are independent of k.

Proof: Assumption 4.2 and Assumption 4.3 imply that $x_k \in \mathcal{F}(\beta)$. Thus, according to Theorem 4.12, there exist constants $\theta^\star \in (0, 1]$, $\nu^\star > 0$, and $\Delta^\star > 0$ independent of k and a vector $d \neq 0$ such that

$$\begin{aligned} \theta^\star g_j(x_k) + \nabla g_j(x_k)^T d = 0 \ , \qquad & j \in \mathcal{E} \ , \\ \theta^\star g_j(x_k) + \nabla g_j(x_K)^T d \geq \nu^\star \ , \qquad & j \in \mathcal{A}(x_k, 0) \ , \\ g_j(x_k) + \nabla g_j(x_k)^T d \geq \nu^\star \ , \qquad & j \in \mathcal{B}(x_k, 0) \ , \\ \|d\|_\infty \leq \Delta^\star & \end{aligned}$$

holds. By defining $s_k := d$, and since $\mathcal{A}_k := \mathcal{A}(x_k, 0)$, $\mathcal{B}_k := \mathcal{B}(x_k, 0)$, and $\nu^\star > 0$, the theorem is proved. □

Theorem 4.13 can be applied to establish a lower bound on the predicted reduction with respect to the constraint violation. It is shown that $Pred_k$ (4.23) depends on the trust region radius Δ_k.

Theorem 4.14 *Let (x_k, v_k) be an iterate of Algorithm 4.1. If Assumption 4.2 and Assumption 4.3 hold, then there exists a constant $c_2 \in (0, 1]$ independent of k such that*

$$Pred_k \geq \sigma_k \, c_2 \, \|g(x_k)^-\|_1^2 \min\left(1, \Delta_k\right) \tag{4.96}$$

holds. Moreover, there exists a constant $c_3 > 0$ independent of k such that the trial step d_k satisfies

$$\|d_k\|_\infty \geq c_3 \|g(x_k)^-\|_1 \min(1, \Delta_k) \ .$$

Proof: Theorem 4.9 establishes the lower bound (4.72) on the predicted reduction $Pred_k$ (4.23). We apply to (4.72) the Assumption 4.2(3.), that is $d_k^T B_k d_k \geq \kappa_{\mathrm{lbB}} \|d_k\|_2 \geq 0$, and $\mu_k \geq 0$, see definition (4.35). This yields

$$\begin{aligned} Pred_k \ &\geq \frac{1}{6}\left(d_k^T B_k d_k + 2\mu_k \Delta_k\right) + \frac{1}{8}\sigma_k \sum_{j \in \mathcal{S}_k} g_j(x_k)^2 \left(1 - z_j^{(k)^2}\right) \\ &\geq \frac{1}{8}\sigma_k \sum_{j \in \mathcal{S}_k} g_j(x_k)^2 \left(1 - z_j^{(k)^2}\right) \ . \end{aligned} \tag{4.97}$$

If the standard subproblem (4.7) is consistent and a solution exists, then $z_j^{(k)} = 0$ for $j = 1, \ldots, m$, according to Step 1 of Algorithm 4.1. Moreover, we make use of $\mathcal{E} \cup \mathcal{A}_k \subset \mathcal{S}_k$, what follows directly from the definition of $\mathcal{S}_k$ (4.2) and $v_j^{(k)} \geq 0$, for all $j \in \mathcal{I}$, according to Lemma 4.4. Thus, from (4.97) and $\sqrt{m}\|g(x_k)^-\|_2 \geq \|g(x_k)^-\|_1$ it follows

$$\begin{aligned} Pred_k \ &\geq \frac{1}{8}\sigma_k \sum_{j \in \mathcal{S}_k} g_j(x_k)^2 \left(1 - z_j^{(k)^2}\right) \ \geq \ \frac{1}{8}\sigma_k \sum_{j \in \mathcal{E} \cup \mathcal{A}_k} g_j(x_k)^2 \left(1 - z_j^{(k)^2}\right) \\ &= \frac{1}{8}\sigma_k \sum_{j \in \mathcal{E} \cup \mathcal{A}_k} g_j(x_k)^2 \ = \ \frac{1}{8}\sigma_k \|g(x_k)^-\|_2^2 \ \geq \ \frac{1}{8m}\sigma_k \|g(x_k)^-\|_1^2 \\ &\geq \sigma_k \, c_2 \, \|g(x_k)^-\|_1^2 \min\left(1, \Delta_k\right) \ , \end{aligned} \tag{4.98}$$

for any $0 < c_2 \leq 1/(8m)$.

Now we consider the case when the feasibility restoration phase is executed. Since Assumption 4.2 and Assumption 4.3 hold, according to Theorem 4.13 there exist constants $\theta^\star \in (0,1]$, $\Delta^\star > 0$, and a vector $s_k \neq 0$ such that

$$\begin{aligned} \theta^\star g_j(x_k) + \nabla g_j(x_k)^T s_k &= 0 , \quad j \in \mathcal{E} , \\ \theta^\star g_j(x_k) + \nabla g_j(x_k)^T s_k &\geq 0 , \quad j \in \mathcal{A}_k , \\ g_j(x_k) + \nabla g_j(x_k)^T s_k &\geq 0 , \quad j \in \mathcal{B}_k , \\ \|s_k\|_\infty &\leq \Delta^\star \end{aligned}$$

holds. In the following a distinction is made between the case when $\|s_k\|_\infty > \Delta_k$ and the situation when $\|s_k\|_\infty \leq \Delta_k$.

In case $\|s_k\|_\infty > \Delta_k$, then the vector

$$\widetilde{s_k} := (\Delta_k / \|s_k\|_\infty) s_k$$

is defined such that $\|\widetilde{s_k}\|_\infty = \Delta_k$ and

$$\begin{aligned} \theta^\star (\Delta_k / \|s_k\|_\infty) g_j(x_k) + \nabla g_j(x_k)^T \widetilde{s_k} &= 0 , \quad j \in \mathcal{E} , \\ \theta^\star (\Delta_k / \|s_k\|_\infty) g_j(x_k) + \nabla g_j(x_k)^T \widetilde{s_k} &\geq 0 , \quad j \in \mathcal{A}_k , \\ g_j(x_k) + \nabla g_j(x_k)^T \widetilde{s_k} &\geq 0 , \quad j \in \mathcal{B}_k , \\ \|\widetilde{s_k}\|_\infty &\leq \Delta^\star \end{aligned}$$

hold. Since $\Delta^\star \geq \|s_k\|_\infty$, we obtain the following estimate for the factor that relaxes the constraints

$$\theta^\star \frac{\Delta_k}{\|s_k\|_\infty} \geq \theta^\star \frac{\Delta_k}{\Delta^\star} \geq \theta^\star \frac{\Delta_k}{\bar{\Delta}} , \tag{4.99}$$

where $\bar{\Delta} := \max(1, \Delta^\star)$. Note that $(\widetilde{s_k}, 1 - \theta^\star(\Delta_k / \|s_k\|_\infty))$ is feasible for the feasibility restoration subproblem (4.9) that determines δ_k. As δ_k is a minimizer of the problem, an upper bound on δ_k is obtained, that is

$$1 - \frac{\theta^\star}{\bar{\Delta}} \min(1, \Delta_k) \geq 1 - \theta^\star \frac{\Delta_k}{\bar{\Delta}} \geq 1 - \theta^\star \frac{\Delta_k}{\|s_k\|_\infty} \geq \delta_k , \tag{4.100}$$

where the estimate on the left-hand side follows from (4.99).

If $\|s_k\|_\infty \leq \Delta_k$, then $(s_k, 1 - \theta^\star)$ is feasible for the feasibility restoration problem (4.9). Thus, δ_k can be estimated by

$$1 - \frac{\theta^\star}{\bar{\Delta}} \min(1, \Delta_k) \geq 1 - \frac{\theta^\star}{\bar{\Delta}} \geq 1 - \theta^\star \geq \delta_k , \tag{4.101}$$

since δ_k is the minimizer of problem (4.9) and $\bar{\Delta} \geq 1$ by definition.

From (4.100) and (4.101), it follows that in both cases the inequality

$$1 - \delta_k \geq \frac{\theta^\star}{\bar{\Delta}} \min(1, \Delta_k) \tag{4.102}$$

is valid. As the feasibility restoration phase is executed, z_k is determined according to (4.11). Thus, by definition

$$z_j^{(k)} \leq \delta_k \tag{4.103}$$

follows for all $j = 1, \ldots, m$. Moreover, we make use of $0 \leq \delta_k \leq 1$ and obtain

$$1 - \delta_k^2 \;\geq\; 1 - \delta_k \;. \tag{4.104}$$

Applying (4.102)-(4.104) to (4.97) yields

$$\begin{aligned}
Pred_k &\geq \frac{1}{8}\sigma_k \sum_{j\in\mathcal{S}_k} g_j(x_k)^2\left(1 - z_j^{(k)^2}\right) \\
&\overset{(4.103)}{\geq} \frac{1}{8}\sigma_k \sum_{j\in\mathcal{S}_k} g_j(x_k)^2(1-\delta_k^2) \\
&\overset{(4.104)}{\geq} \frac{1}{8}\sigma_k \sum_{j\in\mathcal{S}_k} g_j(x_k)^2(1-\delta_k) \\
&\overset{(4.102)}{\geq} \frac{1}{8}\sigma_k \frac{\theta^\star}{\bar{\Delta}} \min(1,\Delta_k) \sum_{j\in\mathcal{S}_k} g_j(x_k)^2 \\
&\geq \frac{1}{8}\sigma_k \frac{\theta^\star}{\bar{\Delta}} \min(1,\Delta_k) \sum_{j\in\mathcal{E}\cup\mathcal{A}_k} g_j(x_k)^2 \\
&= \frac{1}{8}\sigma_k \frac{\theta^\star}{\bar{\Delta}} \|g(x_k)^-\|_2^2 \min(1,\Delta_k) \\
&\geq \frac{1}{8m}\sigma_k \frac{\theta^\star}{\bar{\Delta}} \|g(x_k)^-\|_1^2 \min(1,\Delta_k) \;.
\end{aligned} \tag{4.105}$$

The second to last inequality holds as $\mathcal{E}\cup\mathcal{A}_k \subset \mathcal{S}_k$. The last inequality is obtained by applying $\sqrt{m}\|g(x_k)^-\|_2 \geq \|g(x_k)^-\|_1$. Thus, the estimates (4.98) and (4.105) show the first part of the theorem, where we define $c_2 := \theta^\star/(8m\bar{\Delta})$. As $\theta^\star \in (0,1]$ and $\bar{\Delta} \geq 1$, the restriction $c_2 \in (0, 1/(8m)]$ in (4.98) is satisfied.

The size of the trial step d_k can be estimated as follows. According to the results obtained before, the violated constraints are relaxed at most by the factor $(\theta^\star/\bar{\Delta})\min(1,\Delta_k)$. Thus, the inequality

$$\frac{\theta^\star}{\bar{\Delta}} |g_j(x_k)^-| \min(1,\Delta_k) \leq |\nabla g_j(x_k)^T d_k|$$

has to be satisfied for all $j \in \mathcal{E}\cup\mathcal{A}_k$, as the relaxed subproblems are consistent. We consider the most violated constraint l, with $l \in \mathcal{E}\cup\mathcal{A}_k$, so that

$$\|g(x_k)^-\|_\infty = |g_l(x_k)|$$

holds. Then for the constraint l the estimate

$$\frac{\theta^\star}{\bar{\Delta}}\|g(x_k)^-\|_\infty \min(1,\Delta_k) \le |\nabla g_l(x_k)^T d_k| \le \|\nabla g_l(x_k)\|_2 \|d_k\|_2 \le \kappa \|d_k\|_2$$

is satisfied, where $\|\nabla g_l(x_k)\|_2 \le \kappa$ is applied according to Assumption 4.2 and (4.33). Making use of $\sqrt{n}\|d_k\|_\infty \ge \|d_k\|_2$ and $\|g(x_k)^-\|_\infty \ge \|g(x_k)^-\|_1/m$ yields

$$\|d_k\|_\infty \ge \frac{\|d_k\|_2}{\sqrt{n}} \ge \frac{\theta^\star}{\sqrt{n}\kappa\bar{\Delta}}\|g(x_k)^-\|_\infty \min(1,\Delta_k) \ge \frac{\theta^\star}{\sqrt{n}m\kappa\bar{\Delta}}\|g(x_k)^-\|_1 \min(1,\Delta_k) \ .$$

The proof is completed by setting $c_3 := \theta^\star/(\sqrt{n}m\kappa\bar{\Delta})$. □

The next theorem establishes an upper bound on the error of the model $\Psi_k(d_k, w_k)$ when compared to the actual value of the augmented Lagrangian $\Phi_k(x_k+d_k, v_k+w_k)$. Thus, the difference of the actual change $Ared_k$ (4.38) and the predicted reduction $Pred_k$ (4.23) is considered.

Theorem 4.15 *Let (x_k, v_k) be an iterate of Algorithm 4.1. If Assumption 4.2 and Assumption 4.3 hold, then there exists a constant $c_4 \ge 1$ independent of k such that*

$$|Ared_k - Pred_k| \le \ c_4\|d_k\|_\infty^2 + \sigma_k c_4 \|d_k\|_\infty^4 + \sigma_k c_4 \|d_k\|_\infty^2 \sum_{j\in\mathcal{S}_k} \left| g_j(x_k) z_j^{(k)} \right| \tag{4.106}$$

holds, where $\mathcal{S}_k$ is defined by (4.2).

Proof: As aforementioned, Assumption 4.2 implies upper bounds on the norms that are used in this proof. Without loss of generality it is assumed that the constant $\kappa \ge 1$ in Assumption 4.2 satisfies (4.33). Thus, the bound κ is valid for the function values, the norm of the gradient, the norm of the Hessian matrices, $\|B_k\|_2$, and the multipliers.

First, we apply Taylor's theorem, see, e.g., Conn et al. [21], with $\xi_j \in [0,1]$ for $j = 0, 1, \ldots, m$, to the augmented Lagrangian at the trial point $(x_k + d_k, v_k + w_k)$. This results in

$$\begin{aligned}
\Phi_{\sigma_k}(x_k+d_k, v_k+w_k) &= f(x_k+d_k) \\
&\quad - \sum_{j\in\mathcal{L}_k} \left(\left(v_j^{(k)} + w_j^{(k)}\right) g_j(x_k+d_k) - \frac{1}{2}\sigma_k g_j(x_k+d_k)^2 \right) \\
&\quad - \sum_{j\in\overline{\mathcal{L}}_k} \frac{1}{2} \frac{\left(v_j^{(k)} + w_j^{(k)}\right)^2}{\sigma_k} \\
= \quad & f(x_k) + \nabla f(x_k)^T d_k + \frac{1}{2} d_k^T \nabla^2 f(x_k + \xi_0 d_k) d_k \\
&\quad - \sum_{j\in\mathcal{L}_k} \left(\left(v_j^{(k)} + w_j^{(k)}\right) \left(g_j(x_k) + \nabla g_j(x_k)^T d_k + \frac{1}{2} d_k^T \nabla^2 g_j(x_k+\xi_j d_k) d_k \right) \right. \\
&\qquad \left. - \frac{1}{2}\sigma_k \left(g_j(x_k) + \nabla g_j(x_k)^T d_k + \frac{1}{2} d_k^T \nabla^2 g_j(x_k+\xi_j d_k) d_k \right)^2 \right)
\end{aligned}$$

$$- \sum_{j \in \overline{\mathcal{L}}_k} \frac{1}{2} \frac{\left(v_j^{(k)} + w_j^{(k)}\right)^2}{\sigma_k} \ , \tag{4.107}$$

where the index sets $\mathcal{L}_k$ and $\overline{\mathcal{L}}_k$ are defined by (4.40) and (4.41), respectively.

An upper on the absolute value of the difference of the predicted reduction $Pred_k$ (4.23) and the actual reduction $Ared_k$ (4.38) is established. We make use of the identity $\Phi_{\sigma_k}(x_k, v_k) = \Psi_{\sigma_k}(0,0)$, see (4.19)-(4.22). Moreover, we apply (4.107) and the model value $\Psi_{\sigma_k}(d_k, w_k)$ at the trial point (d_k, w_k), where $\Psi_{\sigma_k}(d_k, w_k)$ and the corresponding index sets $\mathcal{M}_k$ and $\overline{\mathcal{M}}_k$ are defined according to (4.16)-(4.18). As a first step, we obtain by substitution

$$\begin{aligned}
&|Ared_k - Pred_k| = |\Phi_{\sigma_k}(x_k, v_k) - \Phi_{\sigma_k}(x_k + d_k, v_k + w_k) - \Psi_{\sigma_k}(0,0) + \Psi_{\sigma_k}(d_k, w_k)| \\
= \ & |\Psi_{\sigma_k}(d_k, w_k) - \Phi_{\sigma_k}(x_k + d_k, v_k + w_k)| \\
= \ & \Bigg| f(x_k) + \nabla f(x_k)^T d_k + \frac{1}{2} d_k^T B_k d_k \\
& - \sum_{j \in \mathcal{M}_k} \left(\left(v_j^{(k)} + w_j^{(k)}\right) \left(g_j(x_k) + \nabla g_j(x_k)^T d_k\right) - \frac{1}{2}\sigma_k \left(g_j(x_k) + \nabla g_j(x_k)^T d_k\right)^2 \right) \\
& - \sum_{j \in \overline{\mathcal{M}}_k} \frac{1}{2} \frac{\left(v_j^{(k)} + w_j^{(k)}\right)^2}{\sigma_k} \\
& - f(x_k) - \nabla f(x_k)^T d_k - \frac{1}{2} d_k^T \nabla^2 f(x_k + \xi_0 d_k) d_k \\
& + \sum_{j \in \mathcal{L}_k} \Bigg(\left(v_j^{(k)} + w_j^{(k)}\right) \left(g_j(x_k) + \nabla g_j(x_k)^T d_k + \frac{1}{2} d_k^T \nabla^2 g_j(x_k + \xi_j d_k) d_k\right) \\
& \qquad - \frac{1}{2}\sigma_k \left(g_j(x_k) + \nabla g_j(x_k)^T d_k + \frac{1}{2} d_k^T \nabla^2 g_j(x_k + \xi_j d_k) d_k\right)^2 \Bigg) \\
& + \sum_{j \in \overline{\mathcal{L}}_k} \frac{1}{2} \frac{\left(v_j^{(k)} + w_j^{(k)}\right)^2}{\sigma_k} \Bigg| \ .
\end{aligned} \tag{4.108}$$

Since by definition $\mathcal{L}_k \cup \overline{\mathcal{L}}_k = \{1, \ldots, m\}$ and $\mathcal{M}_k \cup \overline{\mathcal{M}}_k = \{1, \ldots, m\}$, it follows that $(\mathcal{L}_k \cap \mathcal{M}_k) \cup (\overline{\mathcal{L}}_k \cap \mathcal{M}_k) = \mathcal{M}_k$ and $(\mathcal{L}_k \cap \overline{\mathcal{M}}_k) \cup (\overline{\mathcal{L}}_k \cap \overline{\mathcal{M}}_k) = \overline{\mathcal{M}}_k$. Thus, a disjunct decomposition of the set of constraint indexes is obtained by

$$\{1, \ldots, m\} = (\mathcal{L}_k \cap \mathcal{M}_k) \cup (\overline{\mathcal{L}}_k \cap \mathcal{M}_k) \cup (\mathcal{L}_k \cap \overline{\mathcal{M}}_k) \cup (\overline{\mathcal{L}}_k \cap \overline{\mathcal{M}}_k) \ . \tag{4.109}$$

As a second step, the sums in (4.108) are recombined by applying (4.109). Moreover, a result implied by Lemma 4.5 is used. Since $z_j^{(k)} = 0$ and $u_j^{(k)} = 0$ for all $j \in \overline{\mathcal{M}}_k$, according to Lemma 4.5, the definition of w_k (4.12) yields

$$v_j^{(k)} + w_j^{(k)} = v_j^{(k)} + \left(u_j^{(k)} - v_j^{(k)}\right)\left(1 - z_j^{(k)}\right) = u_j^{(k)} = 0 \ , \tag{4.110}$$

for all $j \in \overline{\mathcal{M}}_k$. Thus, for all $j \in \overline{\mathcal{L}}_k \cap \overline{\mathcal{M}}_k$, the corresponding terms in $\Phi_{\sigma_k}(x_k + d_k, v_k + w_k)$ and $\Psi_{\sigma_k}(d_k, w_k)$ are equal to

$$\frac{1}{2}\frac{\left(v_j^{(k)} + w_j^{(k)}\right)^2}{\sigma_k} = \frac{1}{2}\frac{{u_j^{(k)}}^2}{\sigma_k} = 0 \ . \tag{4.111}$$

Consequently, the sum vanishes in this case.

We proceed from (4.108) and obtain

$$\begin{aligned}
|Ared_k - Pred_k| = \ & \Bigg| \frac{1}{2} d_k^T B_k d_k - \frac{1}{2} d_k^T \nabla^2 f(x_k + \xi_0 d_k) d_k \\
& + \sum_{j \in \mathcal{L}_k \cap \mathcal{M}_k} \Bigg(\left(v_j^{(k)} + w_j^{(k)}\right) \left(g_j(x_k) + \nabla g_j(x_k)^T d_k + \frac{1}{2} d_k^T \nabla^2 g_j(x_k + \xi_j d_k) d_k \right) \\
& \qquad - \frac{1}{2}\sigma_k \left(g_j(x_k) + \nabla g_j(x_k)^T d_k + \frac{1}{2} d_k^T \nabla^2 g_j(x_k + \xi_j d_k) d_k \right)^2 \\
& \qquad - \left(v_j^{(k)} + w_j^{(k)}\right) \left(g_j(x_k) + \nabla g_j(x_k)^T d_k \right) \\
& \qquad + \frac{1}{2}\sigma_k \left(g_j(x_k) + \nabla g_j(x_k)^T d_k \right)^2 \Bigg) \\
& + \sum_{j \in \mathcal{L}_k \cap \overline{\mathcal{M}}_k} \Bigg(\underbrace{\left(v_j^{(k)} + w_j^{(k)}\right)}_{=0} \left(g_j(x_k) + \nabla g_j(x_k)^T d_k + \frac{1}{2} d_k^T \nabla^2 g_j(x_k + \xi_j d_k) d_k \right) \\
& \qquad - \frac{1}{2}\sigma_k \left(g_j(x_k) + \nabla g_j(x_k)^T d_k + \frac{1}{2} d_k^T \nabla^2 g_j(x_k + \xi_j d_k) d_k \right)^2 \\
& \qquad - \frac{1}{2} \underbrace{\frac{\left(v_j^{(k)} + w_j^{(k)}\right)^2}{\sigma_k}}_{=0} \Bigg) \\
& + \sum_{j \in \overline{\mathcal{L}}_k \cap \mathcal{M}_k} \Bigg(\frac{1}{2}\frac{\left(v_j^{(k)} + w_j^{(k)}\right)^2}{\sigma_k} - \left(v_j^{(k)} + w_j^{(k)}\right) \left(g_j(x_k) + \nabla g_j(x_k)^T d_k \right) \\
& \qquad + \frac{1}{2}\sigma_k \left(g_j(x_k) + \nabla g_j(x_k)^T d_k \right)^2 \Bigg) \Bigg| \\
\overset{(4.110)}{=} & \left| \frac{1}{2} d_k^T B_k d_k - \frac{1}{2} d_k^T \nabla^2 f(x_k + \xi_0 d_k) d_k \right.
\end{aligned}$$

$$
\begin{aligned}
&+ \sum_{j \in \mathcal{L}_k \cap \mathcal{M}_k} \Bigg(\left(v_j^{(k)} + w_j^{(k)}\right) \left(g_j(x_k) + \nabla g_j(x_k)^T d_k + \frac{1}{2} d_k^T \nabla^2 g_j(x_k + \xi_j d_k) d_k\right) \\
&\qquad - \frac{1}{2}\sigma_k \left(g_j(x_k) + \nabla g_j(x_k)^T d_k + \frac{1}{2} d_k^T \nabla^2 g_j(x_k + \xi_j d_k) d_k\right)^2 \\
&\qquad - \left(v_j^{(k)} + w_j^{(k)}\right) \left(g_j(x_k) + \nabla g_j(x_k)^T d_k\right) \\
&\qquad + \frac{1}{2}\sigma_k \left(g_j(x_k) + \nabla g_j(x_k)^T d_k\right)^2 \Bigg) \\
&- \sum_{j \in \mathcal{L}_k \cap \overline{\mathcal{M}}_k} \frac{1}{2}\sigma_k \left(g_j(x_k) + \nabla g_j(x_k)^T d_k + \frac{1}{2} d_k^T \nabla^2 g_j(x_k + \xi_j d_k) d_k\right)^2 \\
&+ \sum_{j \in \overline{\mathcal{L}}_k \cap \mathcal{M}_k} \Bigg(\frac{1}{2} \frac{\left(v_j^{(k)} + w_j^{(k)}\right)^2}{\sigma_k} - \left(v_j^{(k)} + w_j^{(k)}\right) \left(g_j(x_k) + \nabla g_j(x_k)^T d_k\right) \\
&\qquad + \frac{1}{2}\sigma_k \left(g_j(x_k) + \nabla g_j(x_k)^T d_k\right)^2 \Bigg) \Bigg| \\
= \; &\Bigg| \frac{1}{2} d_k^T B_k d_k - \frac{1}{2} d_k^T \nabla^2 f(x_k + \xi_0 d_k) d_k \\
&+ \sum_{j \in \mathcal{L}_k \cap \mathcal{M}_k} \Bigg(\left(v_j^{(k)} + w_j^{(k)}\right) \frac{1}{2} d_k^T \nabla^2 g_j(x_k + \xi_j d_k) d_k \\
&\qquad - \frac{1}{2}\sigma_k \Bigg(\left(g_j(x_k) + \nabla g_j(x_k)^T d_k\right)^2 \\
&\qquad\qquad + \left(g_j(x_k) + \nabla g_j(x_k)^T d_k\right) d_k^T \nabla^2 g_j(x_k + \xi_j d_k) d_k \\
&\qquad\qquad + \frac{1}{4} \left(d_k^T \nabla^2 g_j(x_k + \xi_j d_k) d_k\right)^2 \Bigg) \\
&\qquad + \frac{1}{2}\sigma_k \left(g_j(x_k) + \nabla g_j(x_k)^T d_k\right)^2 \Bigg) \\
&- \sum_{j \in \mathcal{L}_k \cap \overline{\mathcal{M}}_k} \frac{1}{2}\sigma_k \left(g_j(x_k) + \nabla g_j(x_k)^T d_k + \frac{1}{2} d_k^T \nabla^2 g_j(x_k + \xi_j d_k) d_k\right)^2 \\
&+ \sum_{j \in \overline{\mathcal{L}}_k \cap \mathcal{M}_k} \Bigg(\frac{1}{2} \frac{\left(v_j^{(k)} + w_j^{(k)}\right)^2}{\sigma_k} - \left(v_j^{(k)} + w_j^{(k)}\right) \left(g_j(x_k) + \nabla g_j(x_k)^T d_k\right) \\
&\qquad + \frac{1}{2}\sigma_k \left(g_j(x_k) + \nabla g_j(x_k)^T d_k\right)^2 \Bigg) \Bigg|
\end{aligned}
$$

$$
\begin{aligned}
= \quad & \Bigg| \frac{1}{2} d_k^T B_k d_k - \frac{1}{2} d_k^T \nabla^2 f(x_k + \xi_0 d_k) d_k \\
& + \sum_{j \in \mathcal{L}_k \cap \mathcal{M}_k} \Bigg(\frac{1}{2} \left(v_j^{(k)} + w_j^{(k)} \right) d_k^T \nabla^2 g_j(x_k + \xi_j d_k) d_k \\
& \qquad - \frac{1}{2} \sigma_k \left(g_j(x_k) + \nabla g_j(x_k)^T d_k \right) d_k^T \nabla^2 g_j(x_k + \xi_j d_k) d_k \\
& \qquad - \frac{1}{8} \sigma_k \left(d_k^T \nabla^2 g_j(x_k + \xi_j d_k) d_k \right)^2 \Bigg) \\
& - \sum_{j \in \mathcal{L}_k \cap \overline{\mathcal{M}}_k} \frac{1}{2} \sigma_k \left(g_j(x_k) + \nabla g_j(x_k)^T d_k + \frac{1}{2} d_k^T \nabla^2 g_j(x_k + \xi_j d_k) d_k \right)^2 \\
& + \sum_{j \in \overline{\mathcal{L}}_k \cap \mathcal{M}_k} \Bigg(\frac{1}{2} \frac{\left(v_j^{(k)} + w_j^{(k)} \right)^2}{\sigma_k} - \left(v_j^{(k)} + w_j^{(k)} \right) \left(g_j(x_k) + \nabla g_j(x_k)^T d_k \right) \\
& \qquad + \frac{1}{2} \sigma_k \left(g_j(x_k) + \nabla g_j(x_k)^T d_k \right)^2 \Bigg) \Bigg| \\
\leq \quad & \underbrace{\left| \frac{1}{2} d_k^T B_k d_k - \frac{1}{2} d_k^T \nabla^2 f(x_k + \xi_0 d_k) d_k \right|}_{\text{(i)}} \\
& + \underbrace{\Bigg| \sum_{j \in \mathcal{L}_k \cap \mathcal{M}_k} \Bigg(\frac{1}{2} \left(v_j^{(k)} + w_j^{(k)} \right) d_k^T \nabla^2 g_j(x_k + \xi_j d_k) d_k \\
& \qquad - \frac{1}{2} \sigma_k \left(g_j(x_k) + \nabla g_j(x_k)^T d_k \right) d_k^T \nabla^2 g_j(x_k + \xi_j d_k) d_k \\
& \qquad - \frac{1}{8} \sigma_k \left(d_k^T \nabla^2 g_j(x_k + \xi_j d_k) d_k \right)^2 \Bigg) \Bigg|}_{\text{(ii)}} \\
& + \underbrace{\left| \sum_{j \in \mathcal{L}_k \cap \overline{\mathcal{M}}_k} \frac{1}{2} \sigma_k \left(g_j(x_k) + \nabla g_j(x_k)^T d_k + \frac{1}{2} d_k^T \nabla^2 g_j(x_k + \xi_j d_k) d_k \right)^2 \right|}_{\text{(iii)}} \\
& + \underbrace{\Bigg| \sum_{j \in \overline{\mathcal{L}}_k \cap \mathcal{M}_k} \Bigg(\frac{1}{2} \frac{\left(v_j^{(k)} + w_j^{(k)} \right)^2}{\sigma_k} - \left(v_j^{(k)} + w_j^{(k)} \right) \left(g_j(x_k) + \nabla g_j(x_k)^T d_k \right) \\
& \qquad + \frac{1}{2} \sigma_k \left(g_j(x_k) + \nabla g_j(x_k)^T d_k \right)^2 \Bigg) \Bigg|}_{\text{(iv)}} . \qquad (4.112)
\end{aligned}
$$

In the following, upper bounds on the terms (i)-(iv) in (4.112) are established. First, an upper bound on (4.112)(i) is derived, that is

$$\begin{aligned}
&\left|\frac{1}{2}d_k^T B_k d_k - \frac{1}{2}d_k^T \nabla^2 f(x_k+\xi_0 d_k)d_k\right| \leq \left|\frac{1}{2}d_k^T B_k d_k\right| + \left|\frac{1}{2}d_k^T \nabla^2 f(x_k+\xi_0 d_k)d_k\right| \\
&\leq \frac{1}{2}\|B_k\|_2\|d_k\|_2^2 + \frac{1}{2}\|\nabla^2 f(x_k+\xi_0 d_k)\|_2\|d_k\|_2^2 \\
&\leq \frac{1}{2}\kappa\|d_k\|_2^2 + \frac{1}{2}\kappa\|d_k\|_2^2 \;=\; \kappa\|d_k\|_2^2 \\
&\leq n\kappa\|d_k\|_\infty^2 \,, \qquad (4.113)
\end{aligned}$$

where the bound κ on the norms $\|B_k\|_2 \leq \kappa$ and $\|\nabla^2 f(x_k+\xi_0 d_k)\|_2 \leq \kappa$, cf. (4.33), is applied. Note that $\mathcal{X}$ being a convex set is required here, see Assumption 4.2(1.). The last estimate follows from $\|d_k\|_2 \leq \sqrt{n}\|d_k\|_\infty$.

In the remainder of this proof the following estimates are applied several times. The upper bound κ on the norm of the Hessian matrices of the constraint functions is used. Applying additionally the Cauchy-Schwarz inequality and $\|d_k\|_2 \leq \sqrt{n}\|d_k\|_\infty$, leads to

$$\left|d_k^T \nabla^2 g_j(x_k+\xi_j d_k)d_k\right| \leq \left\|\nabla^2 g_j(x_k+\xi_j d_k)\right\|_2 \|d_k\|_2^2 \overset{(4.33)}{\leq} \kappa\|d_k\|_2^2 \leq n\kappa\|d_k\|_\infty^2 \,, \quad (4.114)$$

for $j = 1, \ldots, m$. Moreover, for all $j \in \mathcal{E} \cup \mathcal{I}$

$$\left|v_j^{(k)} + w_j^{(k)}\right| \overset{(4.12)}{=} \left|v_j^{(k)} + \left(u_j^{(k)} - v_j^{(k)}\right)\left(1 - z_j^{(k)}\right)\right| \leq \kappa \qquad (4.115)$$

holds, as $0 \leq z_j^{(k)} \leq 1$, according to (4.11) and the update rule in Step 1 of Algorithm 4.1, and since $\|u_k\|_\infty \leq \kappa$ and $\|v_0\|_\infty \leq \kappa$ by Assumption 4.2 .

In a second step, the term (4.112)(ii) is considered. Applying (4.114) and (4.115) to the sum over all $j \in \mathcal{L}_k \cap \mathcal{M}_k$ yields

$$\begin{aligned}
&\left|\sum_{j\in\mathcal{L}_k\cap\mathcal{M}_k}\Big(\frac{1}{2}\left(v_j^{(k)}+w_j^{(k)}\right)d_k^T\nabla^2 g_j(x_k+\xi_j d_k)d_k\right. \\
&\qquad -\frac{1}{2}\sigma_k\left(g_j(x_k)+\nabla g_j(x_k)^T d_k\right)d_k^T\nabla^2 g_j(x_k+\xi_j d_k)d_k \\
&\qquad \left.-\frac{1}{8}\sigma_k\left(d_k^T\nabla^2 g_j(x_k+\xi_j d_k)d_k\right)^2\Big)\right| \\
\leq\;& \sum_{j\in\mathcal{L}_k\cap\mathcal{M}_k}\left|\frac{1}{2}\left(v_j^{(k)}+w_j^{(k)}\right)\right|\left|d_k^T\nabla^2 g_j(x_k+\xi_j d_k)d_k\right| \\
&+\sum_{j\in\mathcal{L}_k\cap\mathcal{M}_k}\left|\frac{1}{8}\sigma_k\left(d_k^T\nabla^2 g_j(x_k+\xi_j d_k)d_k\right)^2\right| \\
&+\sum_{j\in\mathcal{L}_k\cap\mathcal{M}_k}\left|\frac{1}{2}\sigma_k\left(g_j(x_k)+\nabla g_j(x_k)^T d_k\right)\right|\left|d_k^T\nabla^2 g_j(x_k+\xi_j d_k)d_k\right|
\end{aligned}$$

$$\begin{aligned}
&\overset{(4.114)}{\leq} \sum_{j \in \mathcal{L}_k \cap \mathcal{M}_k} \frac{1}{2} \left| v_j^{(k)} + w_j^{(k)} \right| n\kappa \|d_k\|_\infty^2 + \sum_{j \in \mathcal{L}_k \cap \mathcal{M}_k} \frac{1}{8} \sigma_k n^2 \kappa^2 \|d_k\|_\infty^4 \\
&\qquad + \sum_{j \in \mathcal{L}_k \cap \mathcal{M}_k} \frac{1}{2} \sigma_k \left| g_j(x_k) + \nabla g_j(x_k)^T d_k \right| n\kappa \|d_k\|_\infty^2 \\
&\overset{(4.115)}{\leq} \sum_{j \in \mathcal{L}_k \cap \mathcal{M}_k} \frac{1}{2} n\kappa^2 \|d_k\|_\infty^2 + \sum_{j \in \mathcal{L}_k \cap \mathcal{M}_k} \frac{1}{8} \sigma_k n^2 \kappa^2 \|d_k\|_\infty^4 \\
&\qquad + \sum_{j \in \mathcal{L}_k \cap \mathcal{M}_k} \frac{1}{2} \sigma_k \left| g_j(x_k) + \nabla g_j(x_k)^T d_k \right| n\kappa \|d_k\|_\infty^2 \, . \qquad (4.116)
\end{aligned}$$

The following upper bound on the last sum in (4.116) is established now, that is

$$\begin{aligned}
&\sum_{j \in \mathcal{L}_k \cap \mathcal{M}_k} \frac{1}{2} \sigma_k \left| g_j(x_k) + \nabla g_j(x_k)^T d_k \right| n\kappa \|d_k\|_\infty^2 \\
&\leq \sum_{j \in \mathcal{S}_k} \frac{1}{2} \sigma_k \left| g_j(x_k) z_j^{(k)} \right| n\kappa \|d_k\|_\infty^2 \, . \qquad (4.117)
\end{aligned}$$

First, the summands with $j \in \mathcal{L}_k \cap \mathcal{M}_k$ and $j \in \overline{\mathcal{S}}_k$ are considered. As $j \in \overline{\mathcal{S}}_k \cap \mathcal{M}_k$ holds, it follows according to Lemma 4.6 that

$$g_j(x_k) + \nabla g_j(x_k)^T d_k = 0$$

and $z_j^{(k)} = 0$. Thus, all summands with $j \in \mathcal{L}_k \cap \mathcal{M}_k$ and $j \in \overline{\mathcal{S}}_k$ vanish and do not influence the estimate.

Now we consider any $j \in \mathcal{L}_k \cap \mathcal{M}_k$ with $j \in \mathcal{S}_k$. Since $j \in \mathcal{S}_k \cap \mathcal{M}_k$, Lemma 4.7 can be applied and from

$$g_j(x_k) + \nabla g_j(x_k)^T d_k = g_j(x_k) z_j^{(k)}$$

it follows

$$\left| g_j(x_k) + \nabla g_j(x_k)^T d_k \right| = \left| g_j(x_k) z_j^{(k)} \right| \, .$$

Thus, the estimate (4.117) holds and we can apply (4.117) to (4.116), that yields

$$\begin{aligned}
&\left| \sum_{j \in \mathcal{L}_k \cap \mathcal{M}_k} \left(\frac{1}{2} \left(v_j^{(k)} + w_j^{(k)} \right) d_k^T \nabla^2 g_j(x_k + \xi_j d_k) d_k \right. \right. \\
&\qquad - \frac{1}{2} \sigma_k \left(g_j(x_k) + \nabla g_j(x_k)^T d_k \right) d_k^T \nabla^2 g_j(x_k + \xi_j d_k) d_k \\
&\qquad \left. \left. - \frac{1}{8} \sigma_k \left(d_k^T \nabla^2 g_j(x_k + \xi_j d_k) d_k \right)^2 \right) \right| \\
&\leq \sum_{j \in \mathcal{L}_k \cap \mathcal{M}_k} \frac{1}{2} n\kappa^2 \|d_k\|_\infty^2 + \sum_{j \in \mathcal{L}_k \cap \mathcal{M}_k} \frac{1}{8} \sigma_k n^2 \kappa^2 \|d_k\|_\infty^4
\end{aligned}$$

$$
\begin{aligned}
&+ \sum_{j \in \mathcal{L}_k \cap \mathcal{M}_k} \frac{1}{2}\sigma_k \left| g_j(x_k) + \nabla g_j(x_k)^T d_k \right| n\kappa \|d_k\|_\infty^2 \\
&\overset{(4.117)}{\leq} \sum_{j \in \mathcal{L}_k \cap \mathcal{M}_k} \left(\frac{1}{2} n\kappa^2 \|d_k\|_\infty^2 + \frac{1}{8}\sigma_k n^2 \kappa^2 \|d_k\|_\infty^4 \right) \\
&\quad + \sum_{j \in \mathcal{S}_k} \frac{1}{2}\sigma_k \left| g_j(x_k) z_j^{(k)} \right| n\kappa \|d_k\|_\infty^2 \, . \qquad (4.118)
\end{aligned}
$$

In a third step, an estimate for the term (4.112)(iii) is established. We consider the sum over all $j \in \mathcal{L}_k \cap \overline{\mathcal{M}}_k$. It follows from $j \in \overline{\mathcal{M}}_k$ and the definition of the index set $\overline{\mathcal{M}}_k$ (4.18) that

$$
g_j(x_k) + \nabla g_j(x_k)^T d_k \overset{(4.18)}{>} \frac{v_j^{(k)} + w_j^{(k)}}{\sigma_k} \geq 0 \, . \qquad (4.119)
$$

Moreover, according to Lemma 4.5 it follows that $u_j^{(k)} = 0$ and $z_j^{(k)} = 0$ hold. Thus, the step in the dual variable yields

$$
v_j^{(k)} + w_j^{(k)} = v_j^{(k)} + \left(u_j^{(k)} - v_j^{(k)} \right) \left(1 - z_j^{(k)} \right) = u_j^{(k)} = 0 \, . \qquad (4.120)
$$

Since $j \in \mathcal{L}_k$, we obtain by the definition of the set $\mathcal{L}_k$ (4.40) and (4.120) that

$$
g_j(x_k + d_k) = g_j(x_k) + \nabla g_j(x_k)^T d_k + \frac{1}{2} d_k^T \nabla^2 g_j(x_k + \xi_j d_k) d_k \leq \frac{v_j^{(k)} + w_j^{(k)}}{\sigma_k} = 0 \, . \qquad (4.121)
$$

From (4.119) and (4.121), it follows

$$
0 < g_j(x_k) + \nabla g_j(x_k)^T d_k \leq -\frac{1}{2} d_k^T \nabla^2 g_j(x_k + \xi_j d_k) d_k
$$

and

$$
\left| g_j(x_k) + \nabla g_j(x_k)^T d_k \right| \leq \left| \frac{1}{2} d_k^T \nabla^2 g_j(x_k + \xi_j d_k) d_k \right| \, . \qquad (4.122)
$$

Applying (4.122) and (4.114) to the sum over the index set $\mathcal{L}_k \cap \overline{\mathcal{M}}_k$, leads to the upper bound

$$
\begin{aligned}
&\left| \sum_{j \in \mathcal{L}_k \cap \overline{\mathcal{M}}_k} \frac{1}{2}\sigma_k \left(g_j(x_k) + \nabla g_j(x_k)^T d_k + \frac{1}{2} d_k^T \nabla^2 g_j(x_k + \xi_j d_k) d_k \right)^2 \right| \\
&\leq \sum_{j \in \mathcal{L}_k \cap \overline{\mathcal{M}}_k} \frac{1}{2}\sigma_k \left(\left| g_j(x_k) + \nabla g_j(x_k)^T d_k \right| + \left| \frac{1}{2} d_k^T \nabla^2 g_j(x_k + \xi_j d_k) d_k \right| \right)^2 \\
&\overset{(4.122)}{\leq} \sum_{j \in \mathcal{L}_k \cap \overline{\mathcal{M}}_k} \frac{1}{2}\sigma_k \left| d_k^T \nabla^2 g_j(x_k + \xi_j d_k) d_k \right|^2 \\
&\overset{(4.114)}{\leq} \sum_{j \in \mathcal{L}_k \cap \overline{\mathcal{M}}_k} \frac{1}{2}\sigma_k n^2 \kappa^2 \|d_k\|_\infty^4 \, . \qquad (4.123)
\end{aligned}
$$

Finally, the term (4.112)(iv) is under consideration. According to Lemma 4.8, for all $j \in \overline{\mathcal{L}}_k \cap \mathcal{M}_k$

$$g_j(x_k) + \nabla g_j(x_k)^T d_k \leq 0 \tag{4.124}$$

holds, and with $j \in \overline{\mathcal{L}}_k$, we deduce from the definition of $\overline{\mathcal{L}}_k$ (4.41) that

$$g_j(x_k+d_k) = g_j(x_k)+\nabla g_j(x_k)^T d_k+\frac{1}{2}d_k^T\nabla^2 g_j(x_k+\xi_j d_k)d_k > \frac{v_j^{(k)}+w_j^{(k)}}{\sigma_k} \geq 0\,. \tag{4.125}$$

Thus, (4.124) and (4.125) yield

$$\left|\frac{1}{2}d_k^T\nabla^2 g_j(x_k+\xi_j d_k)d_k\right| > |g_j(x_k)+\nabla g_j(x_k)^T d_k| \tag{4.126}$$

and

$$\frac{1}{2}d_k^T\nabla^2 g_j(x_k+\xi_j d_k)d_k > \frac{v_j^{(k)}+w_j^{(k)}}{\sigma_k} \geq 0\,. \tag{4.127}$$

In order to obtain the last estimate, we apply (4.114), (4.126), and (4.127) to the sum (4.112)(iv), this results in

$$\begin{aligned}
&\sum_{j\in\overline{\mathcal{L}}_k\cap\mathcal{M}_k}\left|\frac{1}{2}\frac{\left(v_j^{(k)}+w_j^{(k)}\right)^2}{\sigma_k}-\left(v_j^{(k)}+w_j^{(k)}\right)\left(g_j(x_k)+\nabla g_j(x_k)^T d_k\right)\right.\\
&\qquad\left.+\frac{1}{2}\sigma_k\left(g_j(x_k)+\nabla g_j(x_k)^T d_k\right)^2\right|\\
&\overset{(4.126)}{\leq}\sum_{j\in\overline{\mathcal{L}}_k\cap\mathcal{M}_k}\left(\frac{1}{2}\left|v_j^{(k)}+w_j^{(k)}\right|\left|\frac{v_j^{(k)}+w_j^{(k)}}{\sigma_k}\right|+\left|v_j^{(k)}+w_j^{(k)}\right|\left|\frac{1}{2}d_k^T\nabla^2 g_j(x_k+\xi_j d_k)d_k\right|\right.\\
&\qquad\left.+\frac{1}{2}\sigma_k\left|\frac{1}{2}d_k^T\nabla^2 g_j(x_k+\xi_j d_k)d_k\right|^2\right)\\
&\overset{(4.127)}{\leq}\sum_{j\in\overline{\mathcal{L}}_k\cap\mathcal{M}_k}\left(\frac{1}{2}\left|v_j^{(k)}+w_j^{(k)}\right|\left|\frac{1}{2}d_k^T\nabla^2 g_j(x_k+\xi_j d_k)d_k\right|\right.\\
&\qquad+\left|v_j^{(k)}+w_j^{(k)}\right|\left|\frac{1}{2}d_k^T\nabla^2 g_j(x_k+\xi_j d_k)d_k\right|\\
&\qquad\left.+\frac{1}{2}\sigma_k\left|\frac{1}{2}d_k^T\nabla^2 g_j(x_k+\xi_j d_k)d_k\right|^2\right)\\
&\leq\sum_{j\in\overline{\mathcal{L}}_k\cap\mathcal{M}_k}\left(\underbrace{\left|v_j^{(k)}+w_j^{(k)}\right|}_{\leq\kappa}\left|d_k^T\nabla^2 g_j(x_k+\xi_j d_k)d_k\right|+\frac{1}{8}\sigma_k\left|d_k^T\nabla^2 g_j(x_k+\xi_j d_k)d_k\right|^2\right)\\
&\overset{(4.114)}{\leq}\sum_{j\in\overline{\mathcal{L}}_k\cap\mathcal{M}_k}\left(n\kappa^2\|d_k\|_\infty^2+\frac{1}{8}\sigma_k n^2\kappa^2\|d_k\|_\infty^4\right)\,.
\end{aligned}\tag{4.128}$$

In the last step the upper bound κ is also applied to the multipliers, see (4.115).

We now apply (4.113), (4.118), (4.123), and (4.128) to (4.112) and obtain the final estimate

$$
\begin{aligned}
& |Ared_k - Pred_k| \\
\leq\ & n\kappa\|d_k\|_\infty^2 \\
& + \sum_{j\in\mathcal{L}_k\cap\mathcal{M}_k}\left(\frac{1}{2}n\kappa^2\|d_k\|_\infty^2 + \frac{1}{8}\sigma_k n^2\kappa^2\|d_k\|_\infty^4\right) + \sum_{j\in\mathcal{S}_k}\frac{1}{2}\sigma_k\left|g_j(x_k)z_j^{(k)}\right| n\kappa\|d_k\|_\infty^2 \\
& + \sum_{j\in\mathcal{L}_k\cap\overline{\mathcal{M}}_k}\frac{1}{2}\sigma_k n^2\kappa^2\|d_k\|_\infty^4 \\
& + \sum_{j\in\overline{\mathcal{L}}_k\cap\mathcal{M}_k}\left(n\kappa^2\|d_k\|_\infty^2 + \frac{1}{8}\sigma_k n^2\kappa^2\|d_k\|_\infty^4\right) \\
\leq\ & n\kappa\|d_k\|_\infty^2 \\
& + \sum_{j\in\mathcal{M}_k}\left(n\kappa^2\|d_k\|_\infty^2 + \frac{1}{8}\sigma_k n^2\kappa^2\|d_k\|_\infty^4\right) + \sum_{j\in\mathcal{S}_k}\frac{1}{2}\sigma_k\left|g_j(x_k)z_j^{(k)}\right| n\kappa\|d_k\|_\infty^2 \\
& + \sum_{j\in\mathcal{L}_k\cap\overline{\mathcal{M}}_k}\frac{1}{2}\sigma_k n^2\kappa^2\|d_k\|_\infty^4 \\
\leq\ & n\kappa\|d_k\|_\infty^2 + m\left(n\kappa^2\|d_k\|_\infty^2 + \frac{1}{2}\sigma_k n^2\kappa^2\|d_k\|_\infty^4\right) + \sum_{j\in\mathcal{S}_k}\frac{1}{2}\sigma_k\left|g_j(x_k)z_j^{(k)}\right| n\kappa\|d_k\|_\infty^2 \\
\leq\ & (m+1)\bar{\kappa}\|d_k\|_\infty^2 + \frac{1}{2}\sigma_k m\bar{\kappa}\|d_k\|_\infty^4 + \sum_{j\in\mathcal{S}_k}\frac{1}{2}\sigma_k\left|g_j(x_k)z_j^{(k)}\right|\bar{\kappa}\|d_k\|_\infty^2 \\
\leq\ & (m+1)\bar{\kappa}\|d_k\|_\infty^2 + \sigma_k m\bar{\kappa}\|d_k\|_\infty^4 + \sum_{j\in\mathcal{S}_k}\sigma_k\left|g_j(x_k)z_j^{(k)}\right|\bar{\kappa}\|d_k\|_\infty^2 \\
\leq\ & (m+1)\bar{\kappa}\|d_k\|_\infty^2 + \sigma_k(m+1)\bar{\kappa}\|d_k\|_\infty^4 + \sum_{j\in\mathcal{S}_k}\sigma_k(m+1)\bar{\kappa}\left|g_j(x_k)z_j^{(k)}\right|\|d_k\|_\infty^2 \\
\leq\ & c_4\|d_k\|_\infty^2 + \sigma_k c_4\|d_k\|_\infty^4 + \sum_{j\in\mathcal{S}_k}\sigma_k c_4\left|g_j(x_k)z_j^{(k)}\right|\|d_k\|_\infty^2\ ,
\end{aligned}
$$

where $\bar{\kappa} := \max(n\kappa, n\kappa^2, n^2\kappa^2)$ and $c_4 := (m+1)\bar{\kappa}$. Since κ is assumed to be greater or equal to one according to Assumption 4.2, the same holds for c_4. Moreover, c_4 is greater or equal to κ. This proves the theorem and c_4 is the required constant. □

Note that the sum $\sum_{j\in\mathcal{S}_k}\sigma_k c_4\left|g_j(x_k)z_j^{(k)}\right|\|d_k\|_\infty^2$ is zero if the standard subproblem (4.7) is solved successfully. In this case $z_j^{(k)}$ is set to zero in Step 1 of Algorithm 4.1 for all $j \in \mathcal{E}\cup\mathcal{I}$.

The following two theorems show that in case an iterate of Algorithm 4.1 is not a feasible stationary point, the algorithm accepts a new trial step after a finite number

of iterations and the trust region radius is bounded away from zero. Moreover, a bound on the penalty parameter is established.

The theorems are shown by contradiction. It is assumed that all trial steps are rejected. This implies that the trust region radius tends to zero, i.e., $\lim_{k\to\infty} \Delta_k = 0$, and $x_{k+1} = x_k$ for all $k > \bar{k}$ according to the update rules of Algorithm 4.1. Here $\bar{k}$ denotes the last iteration where the trial step has been accepted, that is $x_{\bar{k}+1} = x_{\bar{k}} + d_{\bar{k}}$ and $v_{\bar{k}+1} = v_{\bar{k}} + w_{\bar{k}}$. The assumption will be disproved by showing that

$$\frac{Ared_k}{Pred_k} \geq \rho_0$$

has to be satisfied for an iteration $k > \bar{k}$ and the corresponding trial step (d_k, w_k) is accepted. It is demonstrated that in case the trust region radius Δ_k is sufficiently small, then

$$\left| \frac{Ared_k - Pred_k}{Pred_k} \right| \leq 1 - \rho_0 \tag{4.129}$$

holds. If $Ared_k - Pred_k \geq 0$ the point is accepted as this yields a ratio greater or equal to one and hence greater than ρ_0, with $0 < \rho_0 < 1$. Thus, only the case when $Ared_k - Pred_k < 0$ has to be considered, and inequality (4.129) can be transformed in the following way

$$\begin{aligned}
& \left| \frac{Ared_k - Pred_k}{Pred_k} \right| \leq 1 - \rho_0 \\
\Leftrightarrow \quad & |Ared_k - Pred_k| \leq (1 - \rho_0) Pred_k \\
\Leftrightarrow \quad & Pred_k - Ared_k \leq (1 - \rho_0) Pred_k \\
\Leftrightarrow \quad & Ared_k \geq \rho_0 Pred_k \\
\Leftrightarrow \quad & \frac{Ared_k}{Pred_k} \geq \rho_0 \ .
\end{aligned}$$

Consequently, the trial step (d_k, w_k) is accepted what leads to the desired contradiction. Note that if inequality (4.129) holds for a value less than $1 - \rho_0$ on the right-hand side, then the a trial step is also accepted.

Theorem 4.16 *Let Assumption 4.2 and Assumption 4.3 hold. Let iteration $\bar{k}$ be either $\bar{k} = -1$ or a successful iteration of Algorithm 4.1 where the trial step $(d_{\bar{k}}, w_{\bar{k}})$ has been accepted. If*

$$\|\nabla f(x_k) - \nabla g(x_k) u_k\|_2 + \mu_k \geq \omega \tag{4.130}$$

holds with an $\omega > 0$ for all subsequent iterates (x_k, v_k), $\bar{k} + 1 \leq k$, then an iteration $\bar{l}$ exists that is successful, and (4.130) holds for all k with $\bar{k} < k \leq \bar{l}$. Moreover, there exist constants c_5, c_6, c_7, c_8, $c_9 > 0$ independent of k, $\bar{k}$, and $\bar{l}$ such that

$$\Delta_{\bar{l}} \geq \min \left(\frac{c_5}{\sqrt{\sigma_{\bar{k}}}},\, c_6,\, c_7\, \omega \right) \ ,$$

and the penalty parameter $\sigma_{\bar{l}}$ of the successful iteration $\bar{l}$ is bounded by

$$\sigma_{\bar{l}} \leq \max\left(\sigma_{\bar{k}},\, c_8,\, \frac{c_9}{\omega^2}\right) .$$

Proof: The theorem is shown by contradiction. We assume that all trial steps are rejected. This implies $\lim_{k\to\infty} \Delta_k = 0$, and $x_{k+1} = x_k$ for all $k > \bar{k}$ according to the update rules of Algorithm 4.1. As $\bar{k}$ is the last iteration where the trial step has been accepted, that is $x_{\bar{k}+1} = x_{\bar{k}} + d_{\bar{k}}$ and $v_{\bar{k}+1} = v_{\bar{k}} + w_{\bar{k}}$, the trust region radius of iteration $\bar{k}+1$ satisfies $\Delta_{\bar{k}+1} \geq \Delta_{\min}$. The assumption will be disproved by showing that (4.129) holds in case the trust region radius Δ_k satisfies sufficient conditions.

According to Theorem 4.15 there exists a constant $c_4 \geq 1$ independent of k and the difference between the predicted reduction and the actual change can be estimated by

$$|Ared_k - Pred_k| \leq \quad c_4\|d_k\|_\infty^2 + \sigma_k c_4 \|d_k\|_\infty^4 + \sum_{j\in\mathcal{S}_k} \sigma_k c_4 \left|g_j(x_k) z_j^{(k)}\right| \|d_k\|_\infty^2 . \quad (4.131)$$

We assume without loss of generality that $c_4 \geq \kappa \geq 1$, where $\kappa \geq 1$ is the constant according to Assumption 4.2(4.) that also satisfies (4.33). A constant $\epsilon := \omega/2$ is introduced now to simplify the notation and

$$\|\nabla f(x_k) - \nabla g(x_k) u_k\|_2 + \mu_k \geq 2\epsilon$$

holds. We analyze the different cases that can occur for an iteration $k > \bar{k}$.

According to Theorem 4.10, there exists a constant $c_1 > 0$ such that the predicted reduction satisfies

$$Pred_k \geq c_1 \left(\|\nabla f(x_k) - \nabla g(x_k) u_k\|_2 \min\left(\Delta_k, \frac{\|\nabla f(x_k) - \nabla g(x_k) u_k\|_2}{\kappa}\right) + \mu_k \Delta_k \right) ,$$

and the step d_k is bounded from below by

$$\|d_k\|_\infty \geq \min\left(\Delta_k, \frac{\|\nabla f(x_k) - \nabla g(x_k) u_k\|_2}{\sqrt{n}\kappa}\right) . \quad (4.132)$$

There are two cases to distinguish. If $\|\nabla f(x_k) - \nabla g(x_k) u_k\|_2 \geq \epsilon$, then with $\mu_k \geq 0$, see definition (4.35), it follows that

$$\begin{aligned} Pred_k &\geq c_1 \left(\|\nabla f(x_k) - \nabla g(x_k) u_k\|_2 \min\left(\Delta_k, \frac{\|\nabla f(x_k) - \nabla g(x_k) u_k\|_2}{\kappa}\right) + \mu_k \Delta_k \right) \\ &\geq c_1 \left(\|\nabla f(x_k) - \nabla g(x_k) u_k\|_2 \min\left(\Delta_k, \frac{\|\nabla f(x_k) - \nabla g(x_k) u_k\|_2}{\kappa}\right) \right) \\ &= c_1 \|\nabla f(x_k) - \nabla g(x_k) u_k\|_2 \Delta_k \;\geq c_1 \epsilon \Delta_k , \end{aligned} \quad (4.133)$$

in case the trust region radius Δ_k is sufficiently small, that is

$$\Delta_k \leq \epsilon/\kappa \leq \|\nabla f(x_k) - \nabla g(x_k) u_k\|_2/\kappa$$

holds. Furthermore, we can assume that in iteration k

$$\Delta_k \leq \frac{\epsilon}{\sqrt{n}\kappa} \leq \frac{\|\nabla f(x_k) - \nabla g(x_k)u_k\|_2}{\sqrt{n}\kappa} \leq \frac{\|\nabla f(x_k) - \nabla g(x_k)u_k\|_2}{\kappa} ,$$

what implies that $\|d_k\|_\infty = \Delta_k$ according to (4.132).

We now consider the second case. If $\mu_k \geq \epsilon$, then $\|d_k\|_\infty = \Delta_k$ and

$$\begin{aligned} Pred_k &\geq c_1 \left(\|\nabla f(x_k) - \nabla g(x_k)u_k\|_2 \min\left(\Delta_k, \frac{\|\nabla f(x_k) - \nabla g(x_k)u_k\|_2}{\kappa}\right) + \mu_k \Delta_k \right) \\ &\geq c_1 \mu_k \Delta_k \geq c_1 \epsilon \Delta_k \end{aligned} \tag{4.134}$$

is satisfied independent of the specific value of Δ_k. Consequently, this also holds for $\Delta_k \leq \epsilon/(\sqrt{n}\kappa)$.

In both cases, cf. (4.133) and (4.134), we obtain for all iterations k with a trust region radius $\Delta_k \leq \epsilon/(\sqrt{n}\kappa)$ that $\|d_k\|_\infty = \Delta_k$ and

$$Pred_k \geq c_1 \epsilon \Delta_k . \tag{4.135}$$

In the following we consider a specific iteration k where

$$\Delta_k \leq \min\left(\frac{\epsilon}{\sqrt{n}\kappa}, \tau_1 \Delta_{\min}\right) \tag{4.136}$$

holds. This implies that $k > \bar{k} + 1$, i.e., it is not the first iteration following the successful iteration $\bar{k}$. As for the primal step in iteration $\bar{k} + 1$ either

$$\|d_{\bar{k}+1}\|_\infty = \Delta_{\bar{k}+1} \geq \Delta_{\min} \quad \text{or} \quad \|d_{\bar{k}+1}\|_\infty \geq \epsilon/(\sqrt{n}\kappa) \tag{4.137}$$

holds according to (4.132), it also follows that $\|d_{\bar{k}+1}\|_\infty \geq \Delta_k$.

Now a distinction is made between feasible iterates and iterates that violate the constraints. Note that in case the standard subproblem (4.7) is consistent, then $z_j^{(k)} = 0$ for all $j \in \mathcal{E} \cup \mathcal{I}$ according to Step 1 of Algorithm 4.1. Therefore, the last term on the right-hand side of (4.131) is equal to zero.

Case 1: $\|g(x_k)^-\|_1 = 0$

As x_k is a feasible iterate, the standard subproblem (4.7) is consistent and $z_j^{(k)} = 0$ for all $j \in \mathcal{E} \cup \mathcal{I}$. Thus, the difference between the actual change and the predicted reduction (4.131) can be estimated now by

$$|Ared_k - Pred_k| \leq c_4 \|d_k\|_\infty^2 + \sigma_k c_4 \|d_k\|_\infty^4 . \tag{4.138}$$

First, we consider the case when the penalty parameter remains unchanged, i.e., $\sigma_k = \sigma_{\bar{k}}$ for all $k > \bar{k}$, where $\sigma_{\bar{k}}$ is the penalty parameter of iteration $\bar{k}$.

Case 1.1: $\sigma_k = \sigma_{\bar{k}}$

Let k be an iteration such that (4.136) is satisfied and in addition

$$\Delta_k \leq \sqrt{\frac{128\, m\, c_4^4\, \widetilde{\Delta}^2}{\kappa_{\mathrm{lbB}}\, c_2^2\, \tau_1^4\, \sigma_{\bar{k}}\, (1-\rho_0)^2}}\ , \tag{4.139}$$

where $0 < \tau_1, \rho_0 < 1$ are the constants from Algorithm 4.1, $\kappa_{\mathrm{lbB}} \geq 0$ is the constant according to Assumption 4.2, and $c_2 \in (0, 1]$ is the constant from Theorem 4.14. A constant $\widetilde{\Delta} \geq 1$ is introduced here to be able to reuse the following results later. Right now it is sufficient that $\widetilde{\Delta}$ is greater or equal to one. The actual choice of $\widetilde{\Delta}$ is motivated later.

Applying $\|d_k\|_\infty = \Delta_k$, $\sigma_k = \sigma_{\bar{k}}$, and (4.139) to (4.138) yields

$$\begin{aligned} |Ared_k - Pred_k| &\leq c_4\, \Delta_k^2 + \sigma_{\bar{k}}\, c_4\, \frac{128\, m\, c_4^4\, \widetilde{\Delta}^2}{\kappa_{\mathrm{lbB}}\, c_2^2\, \tau_1^4\, \sigma_{\bar{k}}\, (1-\rho_0)^2}\, \Delta_k^2 \\ &= c_4\, \Delta_k^2 + \frac{128\, m\, c_4^5\, \widetilde{\Delta}^2}{\kappa_{\mathrm{lbB}}\, c_2^2\, \tau_1^4\, (1-\rho_0)^2}\, \Delta_k^2\ , \end{aligned}$$

and it follows with the estimate (4.135) for the predicted reduction that

$$\begin{aligned} \left|\frac{Ared_k - Pred_k}{Pred_k}\right| &\leq \frac{\kappa_{\mathrm{lbB}}\, c_2^2\, \tau_1^4\, (1-\rho_0)^2\, c_4\, \Delta_k^2 + 128\, m\, c_4^5\, \widetilde{\Delta}^2\, \Delta_k^2}{\kappa_{\mathrm{lbB}}\, c_2^2\, \tau_1^4\, (1-\rho_0)^2\, c_1\, \epsilon\, \Delta_k} \\ &= \frac{\kappa_{\mathrm{lbB}}\, c_2^2\, \tau_1^4\, (1-\rho_0)^2\, c_4 + 128\, m\, c_4^5\, \widetilde{\Delta}^2}{\kappa_{\mathrm{lbB}}\, c_2^2\, \tau_1^4\, (1-\rho_0)^2\, c_1\, \epsilon}\, \Delta_k\ . \end{aligned}$$

Let the trust region radius Δ_k also satisfy

$$\Delta_k \leq \frac{1}{2}\, \frac{\kappa_{\mathrm{lbB}}\, c_2^2\, \tau_1^4\, (1-\rho_0)^2\, c_1\, \epsilon}{\kappa_{\mathrm{lbB}}\, c_2^2\, \tau_1^4\, (1-\rho_0)^2\, c_4 + 128\, m\, c_4^5\, \widetilde{\Delta}^2}\, (1-\rho_0)\ ,$$

then

$$\left|\frac{Ared_k - Pred_k}{Pred_k}\right| \leq \frac{1}{2}(1-\rho_0)\ , \tag{4.140}$$

and the trial step is accepted. We define

$$\begin{aligned} \overline{\Delta}_1 := \min\Bigg(&\frac{\epsilon}{\sqrt{n}\kappa},\ \tau_1 \Delta_{\min},\ \sqrt{\frac{128\, m\, c_4^4\, \widetilde{\Delta}^2}{\kappa_{\mathrm{lbB}}\, c_2^2\, \tau_1^4\, \sigma_{\bar{k}}\, (1-\rho_0)^2}}, \\ &\frac{1}{2}\frac{\kappa_{\mathrm{lbB}}\, c_2^2\, \tau_1^4\, c_1\, \epsilon\, (1-\rho_0)^3}{\kappa_{\mathrm{lbB}}\, c_2^2\, \tau_1^4\, (1-\rho_0)^2\, c_4 + 128\, m\, c_4^5\, \widetilde{\Delta}^2}\Bigg), \end{aligned} \tag{4.141}$$

and a trial step is accepted for sure as soon as the trust region radius is equal to the

minimum of the right-hand side of (4.141) or the first time it falls below the minimum. This is a contradiction to our assumption. Thus, a subsequent successful iteration $\bar{l}$ exists and the trust region radius of iteration $\bar{l}$ can not be less than $\tau_1\overline{\Delta}_1$, i.e.,

$$\Delta_{\bar{l}} \geq \tau_1\overline{\Delta}_1 \ . \tag{4.142}$$

This follows from $k > \bar{k}+1$ and the update rule for the trust region radius, i.e., $\Delta_k = \tau_1\|d_{k-1}\|_\infty$ according to Step 5 of Algorithm 4.1, and the fact that the previously rejected primal trial step d_{k-1} has to satisfy $\|d_{k-1}\|_\infty > \overline{\Delta}_1$. This is assured as (4.137) and $\Delta_{\bar{k}+1} \geq \Delta_{\min}$ hold. Thus, the trust region radius will fall below $\overline{\Delta}_1$ after a finite number of iterations. It has to be highlighted that the lower bound on the trust region radius is independent of $\Delta_{\bar{k}}$.

We replace ϵ by applying $\epsilon = \omega/2$ to (4.141) and define constants

$$\nu_1 := \min\left(\tau_1\frac{1}{2\sqrt{n}\kappa},\ \tau_1\frac{1}{4}\frac{\kappa_{\mathrm{lbB}}\, c_2^2\, \tau_1^4\, c_1\,(1-\rho_0)^3}{\kappa_{\mathrm{lbB}}\, c_2^2\, \tau_1^4\,(1-\rho_0)^2\, c_4 + 128\, m\, c_4^5\, \widetilde{\Delta}^2}\right)$$

and

$$\nu_2 := \tau_1\sqrt{\frac{128\, m\, c_4^4\, \widetilde{\Delta}^2}{\kappa_{\mathrm{lbB}}\, c_2^2\, \tau_1^4\,(1-\rho_0)^2}} \tag{4.143}$$

such that

$$\tau_1\overline{\Delta}_1 = \min\left(\nu_1\,\omega,\ \frac{\nu_2}{\sqrt{\sigma_{\bar{k}}}},\ \tau_1^2\Delta_{\min}\right) \ . \tag{4.144}$$

In a next step we consider the case when the penalty parameter has been increased since the last successful iteration $\bar{k}$.

Case 1.2: $\sigma_k > \sigma_{\bar{k}}$

Let k be an iteration such that (4.136) holds. If the penalty parameter has been increased in a previous iteration and $\sigma_{k-1} > \sigma_{\bar{k}}$ but the trial step has been rejected, then the update rule for the trust region radius Δ_k and the convexity of the subproblems imply the inequality

$$\|d_k\|_\infty \leq \|d_{k-1}\|_\infty \leq \ldots \leq \|d_{\bar{k}+1}\|_\infty \tag{4.145}$$

holds. The following estimate is also valid in case the penalty parameter is increased in iteration k. It can also be used to estimate $\sigma_{\bar{l}}$ later. An upper bound on σ_k can be estimated by

$$\sigma_k = \max_{j=1,\ldots,m}\left(\frac{2m\left(u_j^{(l)}-v_j^{(l)}\right)^2}{d_l^T B_k d_l + \mu_l\Delta_l}\left(1-z_j^{(l)2}\right)\right) \leq \frac{8m\kappa^2}{\kappa_{\mathrm{lbB}}\,\|d_l\|_2^2} \leq \frac{8m\kappa^2}{\kappa_{\mathrm{lbB}}\,\|d_l\|_\infty^2} \leq \frac{8m\kappa^2}{\kappa_{\mathrm{lbB}}\,\|d_k\|_\infty^2} \ , \tag{4.146}$$

where l, with $\bar{k} < l \leq k$, denotes the last iteration when the penalty parameter has been increased. Here we applied (4.145) and the upper bound κ on $\|u_l\|_\infty$ and $\|v_l\|_\infty$ with $|u_j^{(l)} - v_j^{(l)}| \leq 2\kappa$, for all $j = 1,\ldots,m$. Moreover, we used $\mu_l \geq 0$, $0 \leq z_j^{(l)} \leq 1$ for $j = 1,\ldots,m$, the lower bound on $d_l^T B_k d_l \geq \kappa_{\mathrm{lbB}}\|d_l\|_\infty^2$ according to Assumption 4.2,

and $\|d_l\|_2 \geq \|d_l\|_\infty$. Note that $B_l = B_k$ as the matrix remains unchanged.

Applying $\|d_k\|_\infty = \Delta_k$ and (4.146) to (4.138) yields

$$\begin{aligned} |Ared_k - Pred_k| &\leq c_4\|d_k\|_\infty^2 + \sigma_k c_4\|d_k\|_\infty^4 \leq c_4\|d_k\|_\infty^2 + \frac{8m\kappa^2}{\kappa_{\text{lbB}}\|d_k\|_\infty^2} c_4\|d_k\|_\infty^4 \\ &= c_4\|d_k\|_\infty^2 + \frac{8m\kappa^2 c_4}{\kappa_{\text{lbB}}}\|d_k\|_\infty^2 = \frac{\kappa_{\text{lbB}} c_4 + 8m\kappa^2 c_4}{\kappa_{\text{lbB}}}\|d_k\|_\infty^2 \\ &= \frac{\kappa_{\text{lbB}} c_4 + 8m\kappa^2 c_4}{\kappa_{\text{lbB}}}\Delta_k^2 \ . \end{aligned}$$

Since $Pred_k \geq c_1\epsilon\Delta_k$ according to (4.135), it follows

$$\left|\frac{Ared_k - Pred_k}{Pred_k}\right| \leq \frac{\kappa_{\text{lbB}} c_4 + 8m\kappa^2 c_4}{\kappa_{\text{lbB}}\, c_1\, \epsilon\, \Delta_k}\Delta_k^2 = \frac{\kappa_{\text{lbB}} c_4 + 8m\kappa^2 c_4}{\kappa_{\text{lbB}}\, c_1\, \epsilon}\Delta_k \leq \frac{1}{2}(1-\rho_0) \quad (4.147)$$

if we require that the trust region radius Δ_k satisfies additionally

$$\Delta_k \leq \frac{1}{2}\frac{\kappa_{\text{lbB}}\, c_1\, \epsilon}{\kappa_{\text{lbB}} c_4 + 8m\kappa^2 c_4}(1-\rho_0) \ .$$

The trial step is accepted in this case what is a contradiction to our assumption. We define

$$\overline{\Delta}_2 := \min\left(\frac{\epsilon}{\sqrt{n}\kappa},\, \tau_1\Delta_{\min},\, \frac{1}{2}\frac{\kappa_{\text{lbB}}\, c_1\, \epsilon\, (1-\rho_0)}{\kappa_{\text{lbB}} c_4 + 8m\kappa^2 c_4}\right) \ . \quad (4.148)$$

Thus, the trust region radius of the next successful iteration $\bar{l}$ can not be less than

$$\Delta_{\bar{l}} \geq \tau_1\overline{\Delta}_2 \quad (4.149)$$

if $\overline{\Delta}_2 \leq \overline{\Delta}_1$. This follows the same way as described for (4.142). We define a constant

$$\nu_3 := \min\left(\tau_1\frac{1}{2\sqrt{n}\kappa},\, \tau_1\frac{1}{4}\frac{\kappa_{\text{lbB}}\, c_1\, (1-\rho_0)}{\kappa_{\text{lbB}} c_4 + 8m\kappa^2 c_4}\right)$$

and therefore

$$\tau_1\overline{\Delta}_2 = \min\left(\nu_3\omega, \tau_1^2\Delta_{\min}\right) \ , \quad (4.150)$$

where we also replaced ϵ by $\epsilon = \omega/2$.

We have to consider the case when $\overline{\Delta}_1 < \overline{\Delta}_2$, $\sigma_{k-1} = \sigma_{\bar{k}}$, and $\Delta_k \leq \overline{\Delta}_2$. In this situation the penalty parameter σ_k might directly depend on the value of $\sigma_{\bar{k}}$. In the previous iteration still $\sigma_{k-1} = \sigma_{\bar{k}}$ holds and since the step was rejected $\Delta_{k-1} > \overline{\Delta}_1$. Otherwise, the step would have been accepted. Let the penalty parameter be increased in iteration k, then as $\Delta_k \leq \overline{\Delta}_2$ the step is accepted. The new value of the penalty parameter σ_k has to be estimated.

From (4.132) and the definition (4.141) of $\overline{\Delta}_1$, it follows that $\|d_{k-1}\|_\infty > \overline{\Delta}_1$. We get $\|d_k\|_\infty = \Delta_k \geq \tau_1\overline{\Delta}_1$ and therefore

$$\sigma_k \leq \frac{8m\kappa^2}{\kappa_{\text{lbB}}\,\tau_1^2\,\overline{\Delta}_1^2}\ , \tag{4.151}$$

according to the estimate (4.146). We are interested in the third term in (4.141) that contains $\sigma_{\bar{k}}$, as all other terms only consist of constants that are not changed by the algorithm and ϵ or ω, respectively. It follows from the estimate (4.151) that if $\overline{\Delta}_1 = \sqrt{\dfrac{128\,m\,c_4^4\,\widetilde{\Delta}^2}{\kappa_{\text{lbB}}\,c_2^2\,\tau_1^4\,\sigma_{\bar{k}}\,(1-\rho_0)^2}}$, then

$$\begin{aligned}\sigma_k &\leq \frac{8m\kappa^2}{\kappa_{\text{lbB}}\tau_1^2\,\overline{\Delta}_1^2} = \frac{8m\kappa^2}{\kappa_{\text{lbB}}\,\tau_1^2\sqrt{\dfrac{128\,m\,c_4^4\,\widetilde{\Delta}^2}{\kappa_{\text{lbB}}\,c_2^2\,\tau_1^4\,\sigma_{\bar{k}}(1-\rho_0)^2}}^{\,2}} = \frac{8\,m\,\kappa^2\,\kappa_{\text{lbB}}\,c_2^2\,\tau_1^4\,(1-\rho_0)^2}{128\,\kappa_{\text{lbB}}\,\tau_1^2\,m\,c_4^4\,\widetilde{\Delta}^2}\sigma_{\bar{k}} \\ &\leq \frac{c_2^2\,\tau_1^2\,(1-\rho_0)^2}{16\,c_4^2\,\widetilde{\Delta}^2}\sigma_{\bar{k}} \leq \sigma_{\bar{k}}\ .\end{aligned} \tag{4.152}$$

The last inequality is obtained as $0 < \tau_1,\ \rho_0 < 1$, $0 < c_2 \leq 1$, and $c_4 \geq \kappa \geq 1$. Thus, the penalty parameter is not increased in this situation and $\sigma_k = \sigma_{\bar{k}}$.

It follows from (4.142), (4.144), (4.149), and (4.150) that, in case $\|g(x_k)^-\|_1 = 0$, the lower bound on the trust region radius $\Delta_{\bar{l}}$ of the next successful iteration $\bar{l}$ is

$$\begin{aligned}\Delta_{\bar{l}} \geq\ & \min\left(\tau_1\overline{\Delta}_1,\ \tau_1\overline{\Delta}_2\right) = \min\left(\nu_1\omega,\ \frac{\nu_2}{\sqrt{\sigma_{\bar{k}}}},\ \nu_3\omega,\ \tau_1^2\Delta_{\min}\right) \\ =\ & \min\left(\frac{\nu_2}{\sqrt{\sigma_{\bar{k}}}},\ \nu_4\omega,\ \tau_1^2\Delta_{\min}\right)\ ,\end{aligned} \tag{4.153}$$

where

$$\nu_4 := \min(\nu_1,\ \nu_3)\ .$$

The penalty parameter $\sigma_{\bar{l}}$ is either $\sigma_{\bar{k}}$ according to the assumption of **Case 1.1** and (4.152), or there exists an upper bound on $\sigma_{\bar{l}}$, that is

$$\sigma_{\bar{l}} \leq \frac{8m\kappa^2}{\kappa_{\text{lbB}}}\max\left(\frac{1}{\nu_4^2\,\omega^2},\ \frac{1}{\tau_1^4\Delta_{\min}^2}\right)\ , \tag{4.154}$$

where we applied $\|d_{\bar{l}}\|_\infty \geq \min(\nu_4\omega,\ \tau_1^2\Delta_{\min})$, cf. (4.153), to the estimate (4.146). Note that $\|d_{\bar{l}}\|_\infty \geq \tau_1\|d_k\|_\infty = \tau_1\Delta_k$ holds, where k is the estimated iteration that leads to the contradiction, as a trial step might be accepted by the algorithm much earlier.

The estimate (4.154) can be reformulated by applying the constants

$$\nu_5 := \frac{8m\kappa^2}{\kappa_{\text{lbB}}\nu_4^2} \quad \text{and} \quad \nu_6 := \frac{8m\kappa^2}{\kappa_{\text{lbB}}\tau_1^4\Delta_{\min}^2}\ . \tag{4.155}$$

We obtain the upper bound on the penalty parameter at iteration $\bar{l}$

$$\sigma_{\bar{l}} \leq \max\left(\sigma_{\bar{k}},\ \frac{\nu_5}{\omega^2},\ \nu_6\right)\ . \tag{4.156}$$

In the following the case is investigated when x_k is an infeasible iterate.

Case 2: $\|g(x_k)^-\|_1 > 0$

As a first step, we consider the case when the iterate is close to the feasible region. We will show that if the constraint violation is sufficiently small, a step s_k exists such that

$$\|s_k\|_\infty \leq \min\left(\tau_1\overline{\Delta}_1, \tau_1\overline{\Delta}_2\right) = \min\left(\frac{\nu_2}{\sqrt{\sigma_{\bar{k}}}},\, \nu_4\omega,\, \tau_1^2\Delta_{\min}\right) ,$$

and thus the standard subproblem (4.7) remains consistent until the trial step is accepted, i.e., the feasibility restoration phase is not entered. We define

$$\overline{\Delta}_3 := \min\left(\tau_1\overline{\Delta}_1, \tau_1\overline{\Delta}_2\right) = \min\left(\frac{\nu_2}{\sqrt{\sigma_{\bar{k}}}},\, \nu_4\omega,\, \tau_1^2\Delta_{\min}\right) . \tag{4.157}$$

The following part, that shows the existence of a step s_k, is similar to the proof of Theorem 3.6 in Spellucci [111]. According to Theorem 4.11 there exists a constant $\bar{\gamma} > 0$ such that for all $0 < \gamma \leq \bar{\gamma}$ and all $x \in \mathcal{F}(\beta)$, there exists a bounded function $d(x)$ with

$$\begin{aligned} &\nabla g_j(x)^T d(x) = -g_j(x) / \left(\|g(x)^-\|_1 + \frac{\gamma}{4\kappa\widetilde{\Delta}}\right) \quad , \;\; j \in \mathcal{E} \; , \\ &\nabla g_j(x)^T d(x) \geq 1 \qquad\qquad\qquad\qquad\qquad\quad , \;\; j \in \mathcal{A}(x,\gamma) \; , \end{aligned} \tag{4.158}$$

since $g_j(x)/(\|g(x)^-\|_1 + \gamma/(4\kappa\widetilde{\Delta}))$, $j \in \mathcal{E}$, is bounded on $\mathcal{F}(\beta)$. Thus, there exists a $\widetilde{\Delta} \geq 1$ such that $\|d(x)\|_2 \leq \widetilde{\Delta}$ for all $x \in \mathcal{F}(\beta)$, since $\mathcal{F}(\beta)$ is compact according to Assumption 4.3. The previously mentioned $\widetilde{\Delta}$ be now the one introduced here.

As $\overline{\Delta}_3 > 0$, we can define

$$\gamma := \min(\kappa\overline{\Delta}_3, \bar{\gamma}) \tag{4.159}$$

so that (4.158) holds, where κ is defined in Assumption 4.2 and also satisfies (4.33).

It follows for all $x \in \mathcal{F}(\beta)$ that

$$g_j(x) \geq \gamma \; , \;\; j \in \mathcal{B}(x,\gamma) \; .$$

Applying $\|\nabla g_j(x)\|_2 \leq \kappa$, $j = 1, \ldots, m$, and $\|d(x)\|_2 \leq \widetilde{\Delta}$, we obtain for $x \in \mathcal{F}(\beta)$ and all $j \in \mathcal{B}(x,\gamma)$ that

$$\begin{aligned} &g_j(x) + \nabla g_j(x)^T d(x) \left(\|g(x)^-\|_1 + \frac{\gamma}{4\kappa\widetilde{\Delta}}\right) \geq \gamma - \kappa\widetilde{\Delta}\|g(x)^-\|_1 - \kappa\widetilde{\Delta}\frac{\gamma}{4\kappa\widetilde{\Delta}} \\ &\geq \gamma - \kappa\widetilde{\Delta}\frac{\gamma}{2\kappa\widetilde{\Delta}} - \kappa\widetilde{\Delta}\frac{\gamma}{4\kappa\widetilde{\Delta}} = \gamma - \frac{\gamma}{2} - \frac{\gamma}{4} = \frac{\gamma}{4} \; , \end{aligned}$$

if $\|g(x)^-\|_1 \leq \gamma/(2\kappa\widetilde{\Delta})$. On the other hand, we get

$$g_j(x) + \nabla g_j(x)^T d(x) \left(\|g(x)^-\|_1 + \frac{\gamma}{4\kappa\widetilde{\Delta}}\right) = 0 \; , \;\; j \in \mathcal{E} \; ,$$

and

$$g_j(x) + \nabla g_j(x)^T d(x) \left(\|g(x)^-\|_1 + \frac{\gamma}{4\kappa\widetilde{\Delta}} \right) \geq -\|g(x)^-\|_1 + \|g(x)^-\|_1 + \frac{\gamma}{4\kappa\widetilde{\Delta}}$$
$$= \frac{\gamma}{4\kappa\widetilde{\Delta}} > 0 \ ,$$

for $j \in \mathcal{A}(x, \gamma)$, as $\nabla g_j(x)^T d(x) \geq 1$ according to (4.158).

Consequently, $s := d(x)(\|g(x)^-\|_1 + \gamma/(4\kappa\widetilde{\Delta}))$ is a step that satisfies the linear constraints of the standard subproblem (4.7) in case $\|g(x)^-\|_1 \leq \gamma/(2\kappa\widetilde{\Delta})$. We return to a specific iterate of the algorithm. If $\|g(x_k)^-\|_1 \leq \gamma/(2\kappa\widetilde{\Delta})$ holds, then we define $s_k := d(x_k)(\|g(x_k)^-\|_1 + \gamma/(4\kappa\widetilde{\Delta}))$ and obtain with $\|d(x_k)\|_\infty \leq \|d(x_k)\|_2 \leq \widetilde{\Delta}$ the estimate on the step size

$$\|s_k\|_\infty = \|d(x_k)\|_\infty \left(\|g(x_k)^-\|_1 + \frac{\gamma}{4\kappa\widetilde{\Delta}} \right) \leq \|d(x_k)\|_2 \left(\|g(x_k)^-\|_1 + \frac{\gamma}{4\kappa\widetilde{\Delta}} \right)$$
$$\leq \widetilde{\Delta} \left(\|g(x_k)^-\|_1 + \frac{\gamma}{4\kappa\widetilde{\Delta}} \right) \ . \tag{4.160}$$

Applying $\|g(x_k)^-\|_1 \leq \gamma/(2\kappa\widetilde{\Delta})$ and the definition (4.159) of γ to (4.160) yields

$$\|s_k\|_\infty \leq \widetilde{\Delta}\frac{\gamma}{2\kappa\widetilde{\Delta}} + \widetilde{\Delta}\frac{\gamma}{4\kappa\widetilde{\Delta}} = \frac{3}{4}\frac{\gamma}{\kappa} = \frac{3}{4}\frac{\min(\kappa\overline{\Delta}_3, \bar{\gamma})}{\kappa} \leq \overline{\Delta}_3 \ , \tag{4.161}$$

and the standard subproblem (4.7) is consistent for all $\Delta_k \geq \overline{\Delta}_3$.

Case 2.1: $\|g(x_k)^-\|_1 \leq \dfrac{\gamma}{2\kappa\widetilde{\Delta}}$

As $\|g(x_k)^-\|_1 \leq \gamma/(2\kappa\widetilde{\Delta})$, a step s_k exists such that $\|s_k\|_\infty \leq \overline{\Delta}_3$ and the subproblem (4.7) is consistent for all $\Delta_k \geq \overline{\Delta}_3$. Thus, a trial step is accepted before entering the feasibility restoration phase, and the estimates (4.153) and (4.156) for the previous case, with $\|g(x_k)^-\|_1 = 0$, can be extended to the nearly feasible region and all x_k with $\|g(x_k)^-\|_1 \leq \gamma/(2\kappa\widetilde{\Delta})$, where γ is defined according to (4.159).

Case 2.2: $\|g(x_k)^-\|_1 > \dfrac{\gamma}{2\kappa\widetilde{\Delta}}$

Let k be an iteration such that (4.136) holds. According to Theorem 4.14, the predicted reduction satisfies

$$Pred_k \geq \sigma_k \, c_2 \, \|g(x_k)^-\|_1^2 \min(1, \Delta_k) \ , \tag{4.162}$$

with a constant $c_2 \in (0, 1]$. For the last term on the right-hand side of (4.131) we get the inequality

$$\sum_{j \in \mathcal{S}_k} \left| g_j(x_k) z_j^{(k)} \right| = \sum_{j \in \mathcal{E} \cup \mathcal{A}_k} \left| g_j(x_k) z_j^{(k)} \right| \leq \|g(x_k)^-\|_1 \ , \tag{4.163}$$

where we applied $\mathcal{E} \cup \mathcal{A}_k \subset \mathcal{S}_k$ and $0 \leq z_j^{(k)} \leq 1$, $j = \mathcal{E} \cup \mathcal{A}_k$. Moreover, we made use of $z_j^{(k)} = 0$ for all $j \in \mathcal{S}_k \cap \mathcal{B}_k$ according to STEP 1 of Algorithm 4.1 and the definition (4.11) of z_k.

Applying (4.163) to the absolute value of the difference of predicted and actual reduction yields

$$\begin{aligned} |Ared_k - Pred_k| &\leq c_4\|d_k\|_\infty^2 + \sigma_k c_4 \|d_k\|_\infty^4 + \sum_{j \in \mathcal{S}_k} \sigma_k c_4 \left| g_j(x_k) z_j^{(k)} \right| \|d_k\|_\infty^2 \\ &\leq \underbrace{c_4\|d_k\|_\infty^2 + \sigma_k c_4 \|d_k\|_\infty^4}_{(i)} + \underbrace{\sigma_k c_4 \|g(x_k)^-\|_1 \|d_k\|_\infty^2}_{(ii)} \ . \end{aligned} \tag{4.164}$$

The term (4.164)(i) can be estimated with $Pred_k \geq c_1 \epsilon \Delta_k$, cf. (4.135), as has been done for **Case 1**. It is straightforward to see that the obtained results for **Case 1**, see for example (4.140) and (4.147), can be reused here. Thus, we get

$$\frac{c_4\|d_k\|_\infty^2 + \sigma_k c_4 \|d_k\|_\infty^4}{Pred_k} \leq \frac{c_4\|d_k\|_\infty^2 + \sigma_k c_4 \|d_k\|_\infty^4}{c_1 \epsilon \Delta_k} \leq \frac{1}{2}(1 - \rho_0) \tag{4.165}$$

if $\Delta_k \leq \min(\overline{\Delta}_1, \overline{\Delta}_2)$, what can be assumed.

For the term (4.164)(ii) we derive the following estimate. Let k be an iteration that also satisfies $\Delta_k \leq 1$. Applying (4.162) and $\|g(x_k)^-\|_1 > \gamma/(2\kappa\widetilde{\Delta})$ to (4.164)(ii) yields

$$\begin{aligned} \frac{\sigma_k c_4 \|g(x_k)^-\|_1 \|d_k\|_\infty^2}{Pred_k} &\leq \frac{\sigma_k c_4 \|g(x_k)^-\|_1 \|d_k\|_\infty^2}{\sigma_k c_2 \|g(x_k)^-\|_1^2 \min(1, \Delta_k)} = \frac{\sigma_k c_4 \|d_k\|_\infty^2}{\sigma_k c_2 \|g(x_k)^-\|_1 \min(1, \Delta_k)} \\ &\leq \frac{2\, c_4\, \kappa\, \widetilde{\Delta}\, \Delta_k^2}{c_2\, \gamma\, \Delta_k} = \frac{2\, c_4\, \kappa\, \widetilde{\Delta}}{c_2 \gamma} \Delta_k \ . \end{aligned} \tag{4.166}$$

Let in addition $\Delta_k \leq \dfrac{c_2\, \gamma}{4\, c_4\, \kappa\, \widetilde{\Delta}}(1 - \rho_0)$ hold, then it follows from (4.166) that

$$\frac{\sigma_k c_4 \|g(x_k)^-\|_1 \|d_k\|_\infty^2}{Pred_k} \leq \frac{2\, c_4\, \kappa\, \widetilde{\Delta}}{c_2 \gamma} \Delta_k \leq \frac{1}{2}(1 - \rho_0) \ . \tag{4.167}$$

It results from (4.165) and (4.167) that the inequality

$$\begin{aligned} \left| \frac{Ared_k - Pred_k}{Pred_k} \right| &\leq \frac{c_4\|d_k\|_\infty^2 + \sigma_k c_4 \|d_k\|_\infty^4}{Pred_k} + \frac{\sigma_k c_4 \|g(x_k)^-\|_1 \|d_k\|_\infty^2}{Pred_k} \\ &\leq \frac{1}{2}(1 - \rho_0) + \frac{1}{2}(1 - \rho_0) \\ &= 1 - \rho_0 \end{aligned}$$

holds, as we require that the trust region radius Δ_k already satisfies

$$\Delta_k \leq \min\left(1, \overline{\Delta}_1, \overline{\Delta}_2, \frac{c_2 \gamma}{4\, c_4\, \kappa\, \widetilde{\Delta}}(1 - \rho_0)\right) \ . \tag{4.168}$$

The trial step is accepted and a contradiction to our assumption is obtained. Thus, the trust region radius of the successful iteration $\bar{l}$ is bounded from below by

$$\begin{aligned}\Delta_{\bar{l}} \geq\ & \min\left(\tau_1,\ \tau_1\overline{\Delta}_1,\ \tau_1\overline{\Delta}_2,\ \tau_1\frac{c_2\gamma}{4\,c_4\,\kappa\,\widetilde{\Delta}}(1-\rho_0)\right)\\ =\ & \min\left(\tau_1,\ \tau_1\frac{c_2\gamma}{4\,c_4\,\kappa\,\widetilde{\Delta}}(1-\rho_0),\ \frac{\nu_2}{\sqrt{\sigma_{\bar{k}}}},\ \nu_4\omega,\ \tau_1^2\Delta_{\min}\right)\ ,\end{aligned} \tag{4.169}$$

where we replaced $\tau_1\overline{\Delta}_1$ and $\tau_1\overline{\Delta}_2$ by applying the reformulation in (4.153).

We have to consider the case when the second term on the right-hand side of (4.169) is the minimum, as γ can take several values, see definition (4.159). Let $\tau_1 c_2\gamma(1-\rho_0)/(4\,c_4\,\kappa\,\widetilde{\Delta})$ be the minimum of (4.169), then the lower bound on $\Delta_{\bar{l}}$ is

$$\Delta_{\bar{l}}\ \geq\ \tau_1\frac{c_2\gamma}{4\,c_4\,\kappa\,\widetilde{\Delta}}(1-\rho_0)\ \overset{(4.159)}{=}\ \tau_1\frac{c_2(1-\rho_0)}{4\,c_4\,\kappa\,\widetilde{\Delta}}\min(\kappa\overline{\Delta}_3,\ \bar{\gamma})\ .$$

If $\gamma=\bar{\gamma}$, then we obtain the lower bound

$$\Delta_{\bar{l}}\geq\tau_1\frac{c_2\bar{\gamma}}{4\,c_4\,\kappa\,\widetilde{\Delta}}(1-\rho_0)\ .$$

In case $\gamma=\kappa\overline{\Delta}_3$, it follows by definition (4.157) of $\overline{\Delta}_3$ and the appropriate reformulation that the lower bound in this case is

$$\Delta_{\bar{l}}\geq\tau_1\frac{c_2\kappa\overline{\Delta}_3}{4\,c_4\,\kappa\,\widetilde{\Delta}}(1-\rho_0)=\tau_1\frac{c_2(1-\rho_0)}{4\,c_4\,\widetilde{\Delta}}\min\left(\frac{\nu_2}{\sqrt{\sigma_{\bar{k}}}},\ \nu_4\omega,\ \tau_1^2\Delta_{\min}\right)\ .$$

Defining the constants

$$\nu_7:=\min\left(\tau_1,\ \tau_1\frac{c_2\bar{\gamma}(1-\rho_0)}{4\,c_4\,\kappa\,\widetilde{\Delta}},\ \tau_1^3\frac{c_2(1-\rho_0)\Delta_{\min}}{4\,c_4\,\widetilde{\Delta}}\right)\ ,$$

$$\nu_8:=\tau_1\frac{c_2\,(1-\rho_0)\,\nu_2}{4\,c_4\,\widetilde{\Delta}}\ , \tag{4.170}$$

and

$$\nu_9:=\tau_1\frac{c_2(1-\rho_0)}{4\,c_4\,\widetilde{\Delta}}\nu_4\ ,$$

and applying them to (4.169) yields

$$\Delta_{\bar{l}}\geq\ \min\left(\frac{\nu_2}{\sqrt{\sigma_{\bar{k}}}},\ \nu_4\omega,\ \tau_1^2\Delta_{\min},\ \nu_7,\ \frac{\nu_8}{\sqrt{\sigma_{\bar{k}}}},\ \nu_9\omega\right)\ . \tag{4.171}$$

We estimate an upper bound on the penalty parameter $\sigma_{\bar{l}}$. For the first three terms on the right-hand side of (4.171), i.e., $\nu_2/\sigma_{\bar{k}}$, $\nu_4\omega$, and $\tau_1^2\Delta_{\min}$, the upper bound (4.156) has already been shown. We consider the remaining terms. The different cases for $\|d_{\bar{l}}\|_\infty\geq\min(\nu_7,\ \nu_8/\sqrt{\sigma_{\bar{k}}},\nu_9\omega)$ are stated separately.

Applying $\|d_{\bar{l}}\|_\infty \geq \nu_7$ to the estimate (4.146) yields

$$\sigma_{\bar{l}} \leq \frac{8m\kappa^2}{\kappa_{\mathrm{lbB}}\|d_{\bar{l}}\|_\infty^2} \leq \frac{8m\kappa^2}{\kappa_{\mathrm{lbB}}\nu_7^2} \ . \tag{4.172}$$

For $\|d_{\bar{l}}\|_\infty \geq \nu_9\omega$ it follows

$$\sigma_{\bar{l}} \leq \frac{8m\kappa^2}{\kappa_{\mathrm{lbB}}\|d_{\bar{l}}\|_\infty^2} \leq \frac{8m\kappa^2}{\kappa_{\mathrm{lbB}}\nu_9^2\omega^2} \ . \tag{4.173}$$

For the case $\|d_{\bar{l}}\|_\infty \geq \nu_8/\sqrt{\sigma_{\bar{k}}}$, we obtain by definition (4.170) of ν_8 that

$$\nu_8^2 = \left(\tau_1 \frac{c_2\,(1-\rho_0)\,\nu_2}{4\,c_4\,\widetilde{\Delta}}\right)^2 = \tau_1^2 \frac{c_2^2\,(1-\rho_0)^2}{16\,c_4^2\,\widetilde{\Delta}^2}\nu_2^2 \ . \tag{4.174}$$

It follows by definition (4.143) of ν_2, that

$$\nu_2^2 = \left(\tau_1\sqrt{\frac{128\,m\,c_4^4\,\widetilde{\Delta}^2}{\kappa_{\mathrm{lbB}}\,c_2^2\,\tau_1^4\,(1-\rho_0)^2}}\right)^2 = \tau_1^2 \frac{128\,m\,c_4^4\,\widetilde{\Delta}^2}{\kappa_{\mathrm{lbB}}\,c_2^2\,\tau_1^4\,\sigma_{\bar{k}}\,(1-\rho_0)^2} \ . \tag{4.175}$$

Applying (4.175) to (4.174) yields

$$\|d_{\bar{l}}\|_\infty^2 \geq \tau_1^4 \frac{c_2^2\,(1-\rho_0)^2}{16\,c_4^2\,\widetilde{\Delta}^2} \frac{128\,m\,c_4^4\,\widetilde{\Delta}^2}{\kappa_{\mathrm{lbB}}\,c_2^2\,\tau_1^4\,\sigma_{\bar{k}}\,(1-\rho_0)^2} = \frac{8m\,c_4^2}{\kappa_{\mathrm{lbB}}\,\sigma_{\bar{k}}} \ . \tag{4.176}$$

For the penalty parameter $\sigma_{\bar{l}}$ we get an estimate for the case the penalty parameter is increased by applying (4.176) to (4.146). We obtain

$$\sigma_{\bar{l}} \leq \frac{8m\kappa^2}{\kappa_{\mathrm{lbB}}\|d_{\bar{l}}\|_\infty^2} \leq \frac{8m\,\kappa^2\,\kappa_{\mathrm{lbB}}\,\sigma_{\bar{k}}}{8m\,\kappa_{\mathrm{lbB}}\,c_4^2} \leq \sigma_{\bar{k}} \ , \tag{4.177}$$

as $c_4 \geq \kappa$. Thus, the penalty parameter is still $\sigma_{\bar{k}}$.

It follows from (4.156), (4.172), (4.173), and (4.177), that the penalty parameter $\sigma_{\bar{l}}$ is bounded by

$$\sigma_{\bar{l}} \leq \max\left(\sigma_{\bar{k}},\ \frac{\nu_5}{\omega^2},\ \nu_6, \nu_{10},\ \frac{\nu_{11}}{\omega^2}\right) \ , \tag{4.178}$$

where

$$\nu_{10} := \frac{8m\kappa^2}{\kappa_{\mathrm{lbB}}\nu_7^2} \quad \text{and} \quad \nu_{11} := \frac{8m\kappa^2}{\kappa_{\mathrm{lbB}}\nu_9^2} \ .$$

At the end, we summarize the obtained results and define the constants stated in the theorem. From (4.142), (4.144), (4.149), (4.150), (4.153), and (4.171) it follows

$$\Delta_{\bar{l}} \geq \min\left(\frac{\nu_2}{\sqrt{\sigma_{\bar{k}}}},\ \nu_4\,\omega,\ \tau_1^2\Delta_{\min},\ \nu_7,\ \frac{\nu_8}{\sqrt{\sigma_{\bar{k}}}},\ \nu_9\,\omega\right) \ . \tag{4.179}$$

The first part of the theorem is shown by defining the constants $c_5 := \min(\nu_2, \nu_8)$, $c_6 := \min(\tau_1^2 \Delta_{\min}, \nu_7)$, and $c_7 := \min(\nu_4, \nu_9)$. Note that all constant are greater than zero.

The penalty parameter $\sigma_{\bar{l}}$ is bounded by

$$\sigma_{\bar{l}} \leq \max\left(\sigma_{\bar{k}}, \frac{\nu_5}{\omega^2}, \nu_6, \nu_{10}, \frac{\nu_{11}}{\omega^2}\right) , \tag{4.180}$$

what follows from (4.156) and (4.178). The theorem is shown by defining the missing constants $c_8 := \min(\nu_6, \nu_{10})$ and $c_9 := \min(\nu_5, \nu_{11})$. □

The following theorem considers infeasible iterates and gives a lower bound on the trust region radius of the next successful iteration. Moreover, it is shown that the corresponding penalty parameter is bounded.

Theorem 4.17 *Let Assumption 4.2 and Assumption 4.3 hold. Let iteration $\bar{k}$ be either $\bar{k} = -1$ or a successful iteration of Algorithm 4.1 where the trial step $(d_{\bar{k}}, w_{\bar{k}})$ has been accepted. If $x_{\bar{k}+1}$ is an infeasible point such that*

$$\|g(x_{\bar{k}+1})^-\|_1 \geq \omega \tag{4.181}$$

holds for an $\omega > 0$, then a trial step is accepted after a finite number of iterations and a subsequent successful iteration $\bar{l}$ exists. Moreover, there exist constants c_{10}, c_{11}, c_{12}, c_{13}, c_{14}, $c_{15} > 0$ independent of k, $\bar{k}$, and $\bar{l}$ such that

$$\Delta_{\bar{l}} \geq \min\left(c_{10}\,\omega, c_{11}\,\omega^2, c_{12}\,\omega^3\right) ,$$

and the penalty parameter $\sigma_{\bar{l}}$ of the next successful iteration $\bar{l}$ is bounded by

$$\sigma_{\bar{l}} \leq \max\left(\sigma_{\bar{k}}, \frac{c_{13}}{\omega^4}, \frac{c_{14}}{\omega^6}, \frac{c_{15}}{\omega^8}\right) .$$

Proof: The theorem is shown by contradiction. We assume that all trial steps are rejected. Thus, the trust region radius tends to zero, i.e., $\lim_{k\to\infty} \Delta_k = 0$, and $x_{k+1} = x_k$ for all $k > \bar{k}$ according to the update rules of Algorithm 4.1. As $\bar{k}$ is the last iteration where the trial step has been accepted, that is $x_{\bar{k}+1} = x_{\bar{k}} + d_{\bar{k}}$ and $v_{\bar{k}+1} = v_{\bar{k}} + w_{\bar{k}}$, the trust region radius of iteration $\bar{k}+1$ satisfies $\Delta_{\bar{k}+1} \geq \Delta_{\min}$.

Let k be an iteration such that

$$\Delta_k \leq \min\left(1, \tau_1\Delta_{\min}, \frac{1}{3}\frac{c_2\,\omega^2}{c_4}(1-\rho_0), \frac{1}{3}\frac{c_2\,\omega^2}{c_4\,\Delta_{\max}^2}(1-\rho_0), \frac{1}{3}\frac{c_2\,\omega}{c_4}(1-\rho_0)\right) . \tag{4.182}$$

The condition $\Delta_k \leq \tau_1\Delta_{\min}$ implies that $k > \bar{k}+1$, i.e., the iteration k is not the one following directly the last successful iteration $\bar{k}$.

As $\|g(x_k)^-\|_1 \geq \omega$, $\Delta_k \leq 1$, and according to Theorem 4.14, the predicted reduction of the considered iteration k satisfies

$$Pred_k \geq \sigma_k\,c_2\,\|g(x_k)^-\|_1^2 \min(1, \Delta_k) = \sigma_k\,c_2\,\|g(x_k)^-\|_1^2\,\Delta_k \geq \sigma_k\,c_2\,\omega^2\,\Delta_k , \tag{4.183}$$

with a constant $c_2 \in (0, 1]$.

According to Theorem 4.15 there exists a constant $c_4 \geq 1$ independent of k and the difference between the predicted reduction and the actual change can be estimated by

$$|Ared_k - Pred_k| \leq \quad c_4\|d_k\|_\infty^2 + \sigma_k c_4 \|d_k\|_\infty^4 + \sum_{j \in \mathcal{S}_k} \sigma_k c_4 \left| g_j(x_k) z_j^{(k)} \right| \|d_k\|_\infty^2 \ . \quad (4.184)$$

For the last term on the right-hand side of (4.184) we get

$$\sum_{j \in \mathcal{S}_k} \left| g_j(x_k) z_j^{(k)} \right| = \sum_{j \in \mathcal{E} \cup \mathcal{A}_k} \left| g_j(x_k) z_j^{(k)} \right| \leq \|g(x_k)^-\|_1 \ , \quad (4.185)$$

where we applied $\mathcal{E} \cup \mathcal{A}_k \subset \mathcal{S}_k$ and $0 \leq z_j^{(k)} \leq 1$, $j = \mathcal{E} \cup \mathcal{A}_k$. Moreover, we made use of $z_j^{(k)} = 0$ for all $j \in \mathcal{S}_k \cap \mathcal{B}_k$ according to Step 1 of Algorithm 4.1 and the definition (4.11) of z_k. Applying (4.185) to (4.184) results in

$$\begin{aligned} |Ared_k - Pred_k| &\leq c_4\|d_k\|_\infty^2 + \sigma_k c_4 \|d_k\|_\infty^4 + \sum_{j \in \mathcal{S}_k} \sigma_k c_4 \left| g_j(x_k) z_j^{(k)} \right| \|d_k\|_\infty^2 \\ &\leq \underbrace{c_4\|d_k\|_\infty^2}_{(i)} + \underbrace{\sigma_k c_4 \|d_k\|_\infty^4}_{(ii)} + \underbrace{\sigma_k c_4 \|g(x_k)^-\|_1 \|d_k\|_\infty^2}_{(iii)} \ . \end{aligned} \quad (4.186)$$

We consider the terms (4.186)(i)-(iii) separately. Dividing by the predicted reduction (4.183) yields the following estimates. For the first term (4.186)(i) we obtain

$$\frac{c_4\|d_k\|_\infty^2}{Pred_k} \leq \frac{c_4\,\Delta_k^2}{\sigma_k\,c_2\,\omega^2\,\Delta_k} \overset{\sigma_k \geq 1}{\leq} \frac{\sigma_k\,c_4\,\Delta_k^2}{\sigma_k\,c_2\,\omega^2\,\Delta_k} = \frac{c_4\,\Delta_k^2}{c_2\,\omega^2\,\Delta_k} = \frac{c_4\,\Delta_k}{c_2\,\omega^2} \leq \frac{1}{3}(1-\rho_0) \ , \quad (4.187)$$

as $\Delta_k \leq c_2\,\omega^2\,(1-\rho_0)/(3\,c_4)$ according to (4.182).

We make use of $\|d_k\|_\infty \leq \Delta_{\max}$ and thus the second term (4.186)(ii) is bounded by

$$\begin{aligned} \frac{\sigma_k\,c_4\,\|d_k\|_\infty^4}{Pred_k} &\leq \frac{\sigma_k\,c_4\,\Delta_{\max}^2\,\|d_k\|_\infty^2}{Pred_k} \leq \frac{\sigma_k\,c_4\,\Delta_{\max}^2\,\Delta_k^2}{\sigma_k\,c_2\,\omega^2\,\Delta_k} = \frac{c_4\,\Delta_{\max}^2\,\Delta_k^2}{c_2\,\omega^2\,\Delta_k} = \frac{c_4\,\Delta_{\max}^2\,\Delta_k}{c_2\,\omega^2} \\ &\leq \frac{1}{3}(1-\rho_0) \ , \end{aligned} \quad (4.188)$$

since $\Delta_k \leq c_2\,\omega^2\,(1-\rho_0)/(3\,c_4\,\Delta_{\max}^2)$, as assumed by (4.182).

The last term (4.186)(iii) can be estimated by

$$\begin{aligned} \frac{\sigma_k\,c_4\,\|g(x_k)^-\|_1\,\|d_k\|_\infty^2}{Pred_k} &\leq \frac{\sigma_k\,c_4\,\|g(x_k)^-\|_1\,\|d_k\|_\infty^2}{\sigma_k\,c_2\,\|g(x_k)^-\|_1^2\,\Delta_k} = \frac{c_4\,\|d_k\|_\infty^2}{c_2\,\|g(x_k)^-\|_1\,\Delta_k} \\ &\leq \frac{c_4\,\Delta_k^2}{c_2\,\omega\,\Delta_k} = \frac{c_4}{c_2\,\omega}\Delta_k \\ &\leq \frac{1}{3}(1-\rho_0) \ , \end{aligned} \quad (4.189)$$

as Δ_k satisfies $\Delta_k \leq c_2\,\omega\,(1-\rho_0)/(3\,c_4)$, what follows from (4.182).

From (4.187), (4.188), and (4.189) it follows that

$$\begin{aligned}\left|\frac{Ared_k - Pred_k}{Pred_k}\right| &\leq \frac{c_4\|d_k\|_\infty^2}{Pred_k} + \frac{\sigma_k c_4\|d_k\|_\infty^4}{Pred_k} + \frac{\sigma_k c_4\|g(x_k)^-\|_1\|d_k\|_\infty^2}{Pred_k} \\ &\leq \frac{1}{3}(1-\rho_0) + \frac{1}{3}(1-\rho_0) + \frac{1}{3}(1-\rho_0) \\ &= 1-\rho_0 \ . \end{aligned} \tag{4.190}$$

The trial step is accepted. This is a contradiction to our assumption. The lower bound on the trust region radius $\Delta_{\bar{l}}$ of the successful iteration $\bar{l}$ can be estimated as follows. Let $\widetilde{l}$ denote an iteration that has not been a successful one. It follows from (4.182) and (4.190) that the trust region radius $\Delta_{\widetilde{l}}$ has to satisfy

$$\Delta_{\widetilde{l}} \geq \min\left(1,\, \tau_1\Delta_{\min},\, \frac{1}{3}\frac{c_2\,\omega^2}{c_4}(1-\rho_0),\, \frac{1}{3}\frac{c_2\,\omega^2}{c_4\,\Delta_{\max}^2}(1-\rho_0),\, \frac{1}{3}\frac{c_2\,\omega}{c_4}(1-\rho_0)\right) \ , \tag{4.191}$$

as otherwise the trial step $(d_{\widetilde{l}}, w_{\widetilde{l}})$ would have been accepted. According to Theorem 4.14 there exists a constant $c_3 > 0$ such that

$$\|d_{\widetilde{l}}\|_\infty \geq c_3\|g(x_{\widetilde{l}})^-\|_1 \min(1, \Delta_{\widetilde{l}}) \geq c_3\,\omega \min(1, \Delta_{\widetilde{l}}) \ , \tag{4.192}$$

where $x_{\widetilde{l}} = x_{\bar{k}+1}$. Thus, the trust region radius of the subsequent iteration is

$$\Delta_{\widetilde{l}+1} = \tau_1\Delta_{\widetilde{l}} \geq \tau_1 c_3\,\omega \min(1, \Delta_{\widetilde{l}}) \ . \tag{4.193}$$

We can conclude from (4.191) and (4.193) that the trust region radius $\Delta_{\bar{l}}$ of the successful iteration $\bar{l}$ is bounded from below by

$$\begin{aligned}\Delta_{\bar{l}} &\geq \tau_1\, c_3\, w \min\left(1,\, \tau_1\Delta_{\min},\, \frac{1}{3}\frac{c_2\,\omega^2}{c_4}(1-\rho_0),\, \frac{1}{3}\frac{c_2\,\omega^2}{c_4\,\Delta_{\max}^2}(1-\rho_0),\, \frac{1}{3}\frac{c_2\,\omega}{c_4}(1-\rho_0)\right) \\ &= \min\left(\nu_1\omega,\, \nu_2\,\omega^3,\, \nu_3\,\omega^2\right) \ , \end{aligned} \tag{4.194}$$

where we applied the constants

$$\nu_1 := \tau_1\, c_3\, \min(1, \tau_1\Delta_{\min}) \ , \quad \nu_2 := \tau_1\, c_3\, \min\left(\frac{c_2\,(1-\rho_0)}{3\,c_4},\, \frac{c_2\,(1-\rho_0)}{3\,c_4\,\Delta_{\max}^2}\right) \ ,$$

and

$$\nu_3 := \frac{\tau_1\, c_3\, c_2\,(1-\rho_0)}{3\,c_4} \ .$$

As $\Delta_{\bar{k}+1} \geq \Delta_{\min}$, it is assured that the algorithm starts after a successful iteration with a trust region radius greater than the lower bound (4.194).

The step is taken independent of the penalty parameter σ_k. Thus, the penalty parameter σ_k is either still equal to $\sigma_{\bar{k}}$ or has been increased. We consider the case when the penalty parameter has been increased.

Theorem 4.14 and the constant $c_3 > 0$ are applied again, and we obtain

$$\|d_{\bar{l}}\|_\infty \geq c_3 \|g(x_{\bar{l}})^-\|_1 \min(1, \Delta_{\bar{l}}) \ . \tag{4.195}$$

With (4.194) and $\|g(x_{\bar{l}})^-\|_1 \geq \omega$, we get

$$\begin{aligned} \|d_{\bar{l}}\|_\infty &\geq c_3 \|g(x_{\bar{l}})^-\|_1 \min(1, \Delta_{\bar{l}}) \\ &\geq c_3 \, \omega \min(1, \Delta_{\bar{l}}) \\ &\geq c_3 \, \omega \min\left(\nu_1 \omega, \, \nu_2 \, \omega^3, \, \nu_3 \, \omega^2\right) \ . \end{aligned} \tag{4.196}$$

We denote the last iteration where the penalty parameter has been increased by l, with $\bar{k} < l \leq \bar{l}$. The inequality

$$\|d_{\bar{l}}\|_\infty \leq \|d_l\|_\infty \leq \|d_{\bar{k}+1}\|_\infty$$

holds and we obtain

$$\sigma_{\bar{l}} = \max_{j=1,\ldots,m} \left(\frac{2m \left(u_j^{(l)} - v_j^{(l)}\right)^2}{d_l^T B_k d_l + \mu_l \Delta_l} \left(1 - z_j^{(l)^2}\right) \right) \leq \frac{8m\kappa^2}{\kappa_{\text{lbB}} \, \|d_l\|_2^2} \leq \frac{8m\kappa^2}{\kappa_{\text{lbB}} \, \|d_l\|_\infty^2} \leq \frac{8m\kappa^2}{\kappa_{\text{lbB}} \, \|d_{\bar{l}}\|_\infty^2} \ . \tag{4.197}$$

Here we applied the upper bound κ on $\|u_l\|_\infty$ and $\|v_l\|_\infty$ with $|u_j^{(l)} - v_j^{(l)}| \leq 2\kappa$, for all $j = 1, \ldots, m$. Moreover, we used $\mu_l \geq 0$, $0 \leq z_j^{(l)} \leq 1$ for $j = 1, \ldots, m$, the lower bound on $d_l^T B_k d_l \geq \kappa_{\text{lbB}} \|d_l\|_\infty^2$ according to Assumption 4.2, and $\|d_l\|_2 \geq \|d_l\|_\infty$. Note that $B_l = B_k$ as the matrix remains unchanged.

In the following we apply the different cases of (4.196) to (4.197). For $\|d_{\bar{l}}\|_\infty \geq c_3 \nu_1 \omega^2$ the penalty parameter is bounded by

$$\sigma_{\bar{l}} \leq \frac{8m\kappa^2}{\kappa_{\text{lbB}} \|d_{\bar{l}}\|_\infty^2} \leq \frac{8 \, m \, \kappa^2}{\kappa_{\text{lbB}} \, c_3^2 \, \nu_1^2 \, \omega^4} \ . \tag{4.198}$$

In case $\|d_{\bar{l}}\|_\infty \geq \, c_3 \, \nu_2 \, \omega^4$ is the minimum of (4.196), then

$$\sigma_{\bar{l}} \leq \frac{8m\kappa^2}{\kappa_{\text{lbB}} \|d_{\bar{l}}\|_\infty^2} \leq \frac{8 \, m \, \kappa^2}{\kappa_{\text{lbB}} \, c_3^2 \, \nu_2^2 \, \omega^8} \tag{4.199}$$

holds. Finally, for $\|d_k\|_\infty \geq c_3 \, \nu_3 \, \omega^3$ we get the bound

$$\sigma_{\bar{l}} \leq \frac{8m\kappa^2}{\kappa_{\text{lbB}} \|d_{\bar{l}}\|_\infty^2} \leq \frac{8 \, m \, \kappa^2}{\kappa_{\text{lbB}} \, c_3^2 \, \nu_3^2 \, \omega^6} \ . \tag{4.200}$$

The case that $\sigma_{\bar{l}} = \sigma_{\bar{k}}$ holds and the inequalities (4.198)-(4.200) yield

$$\sigma_{\bar{l}} \leq \max\left(\sigma_{\bar{k}}, \, \frac{8 \, m \, \kappa^2}{\kappa_{\text{lbB}} \, c_3^2 \, \nu_1^2 \, \omega^4}, \, \frac{8 \, m \, \kappa^2}{\kappa_{\text{lbB}} \, c_3^2 \, \nu_2^2 \, \omega^8}, \, \frac{8 \, m \, \kappa^2}{\kappa_{\text{lbB}} \, c_3^2 \, \nu_3^2 \, \omega^6}\right) \ . \tag{4.201}$$

The theorem follows from (4.194) and (4.201). The proof is completed by setting the constants $c_{10} := \nu_1$, $c_{11} := \nu_3$, and $c_{12} := \nu_2$ for the lower bound on the trust region radius $\Delta_{\bar{l}}$. Moreover, we set $c_{13} := 8 \, m \, \kappa^2 / (\kappa_{\text{lbB}} \, c_3^2 \, \nu_1^2)$, $c_{14} := 8 \, m \, \kappa^2 / (\kappa_{\text{lbB}} \, c_3^2 \, \nu_3^2)$, and $c_{15} := 8 \, m \, \kappa^2 / (\kappa_{\text{lbB}} \, c_3^2 \, \nu_2^2)$. □

The following theorem will be used later to establish the convergence of a subsequence to a stationary point of the nonlinear problem.

Theorem 4.18 *Let* $\{(x_k, v_k)\}$ *and* $\{(d_k, u_k, \mu_k)\}$ *be sequences generated by Algorithm 4.1. If Assumption 4.2 and Assumption 4.3 hold, then*

$$\liminf_{k\to\infty} \left(\|\nabla f(x_k) - \nabla g(x_k) u_k\|_2 + \mu_k + \|g(x_k)^-\|_1 \right) = 0 \ . \tag{4.202}$$

Proof: We prove the statement by contradiction. We assume that the statement does not hold, then there exists a constant $\omega > 0$ such that

$$\|\nabla f(x_k) - \nabla g(x_k) u_k\|_2 + \mu_k + \|g(x_k)^-\|_1 > 2\omega \ , \tag{4.203}$$

for all k. Thus, we can apply Theorem 4.17 to all iterates (x_k, v_k), where $\|g(x_k)^-\|_1 > \omega$ holds. Moreover, Theorem 4.16 can be applied to the remaining iterates, as $\|\nabla f(x_k) - \nabla g(x_k) u_k\|_2 + \mu_k \geq \omega$ has to be satisfied.

From Theorem 4.16 and Theorem 4.17, it follows by induction that the penalty parameter σ_k is bounded by

$$\sigma_k \leq \max \left(\sigma_{-1}, \, c_8, \, \frac{c_9}{\omega^2}, \, \frac{c_{13}}{\omega^4}, \, \frac{c_{14}}{\omega^6}, \, \frac{c_{15}}{\omega^8} \right) \ , \tag{4.204}$$

for all k, where c_8, c_9, c_{13}, c_{14}, and c_{15} are the corresponding constants stated in Theorem 4.16 and Theorem 4.17, respectively. Thus, the sequence $\{\sigma_k\}$ remains bounded and as $\sigma_k \geq \sigma_{k-1}$, for all k, there exists a $\sigma > 0$ such that

$$\lim_{k\to\infty} \sigma_k = \sigma \ .$$

Moreover, Theorem 4.16, Theorem 4.17, and the boundedness of the penalty parameter, see (4.204), imply that there exists a constant $\nu > 0$ such that for the trust region radius $\Delta_k \geq \nu$ holds for all k.

According to Theorem 4.10, for all k the step d_k is bounded from below by

$$\|d_k\|_\infty \geq \min \left(\Delta_k, \frac{\|\nabla f(x_k) - \nabla g(x_k) u_k\|_2}{\sqrt{n}\kappa} \right) \ . \tag{4.205}$$

If $\mu_k > 0$, then $\|d_k\|_\infty = \Delta_k$ holds. Additionally, Theorem 4.14 states that there exists a constant $c_3 > 0$ such that

$$\|d_k\|_\infty \geq c_3 \|g(x_k)^-\|_1 \min(1, \Delta_k) \ . \tag{4.206}$$

A distinction of cases with respect to (4.203) as in the proofs of Theorem 4.16 and Theorem 4.17, let us conclude from (4.203), (4.205), (4.206), $\|d_k\|_\infty = \Delta_k$ in case $\mu_k > 0$, and $\Delta_k \geq \nu$, that $\|d_k\|_\infty$ is bounded away from zero, i.e., there exist an $\epsilon > 0$ independent of k such that

$$\|d_k\|_\infty > \epsilon > 0 \ , \tag{4.207}$$

for all k.

Let $\{(x_{k_i}, v_{k_i})\}_K$ denote the infinite subsequence of successful iterations, i.e., the iterations where $Ared_k/Pred_k \geq \rho_0$. The existence is implied by Theorem 4.16 and Theorem 4.17. We now consider only iterations k_i with $k_i \in K$. According to Theorem 4.9 the predicted reduction can be estimated by

$$\begin{aligned} Pred_{k_i} \geq\ & \frac{1}{6}\left(d_{k_i}^T B_{k_i} d_k + 2\mu_{k_i}\Delta_{k_i}\right) + \underbrace{\frac{1}{8}\sigma_{k_i}\sum_{j\in\mathcal{S}_{k_i}} g_j(x_{k_i})^2\left(1 - {z_j}^{(k_i)^2}\right)}_{\geq 0} \\ \geq\ & \frac{1}{6} d_{k_i}^T B_{k_i} d_{k_i} \ \geq\ \frac{1}{6}\kappa_{\mathrm{lbB}}\|d_{k_i}\|_2^2 \ \geq\ \frac{1}{6}\kappa_{\mathrm{lbB}}\|d_{k_i}\|_\infty^2 \ \geq\ \frac{1}{6}\kappa_{\mathrm{lbB}}\epsilon^2 \ , \end{aligned}$$

where we applied $d_{k_i}^T B_{k_i} d_{k_i} + 2\mu_{k_i}\Delta_{k_i} \geq d_{k_i}^T B_{k_i} d_{k_i} \geq \kappa_{\mathrm{lbB}}\|d_{k_i}\|_2^2 \geq \kappa_{\mathrm{lbB}}\|d_{k_i}\|_\infty^2 \geq \kappa_{\mathrm{lbB}}\epsilon^2$, what follows from Assumption 4.2, $\|d_{k_i}\|_2^2 \geq \|d_{k_i}\|_\infty^2$, $\mu_{k_i} \geq 0$, and (4.207). We obtain

$$\Phi_{\sigma_{k_i}}(x_{k_{i+1}}, v_{k_{i+1}}) \leq \Phi_{\sigma_{k_i}}(x_{k_i}, v_{k_i}) - \frac{1}{6}\rho_0\kappa_{\mathrm{lbB}}\epsilon^2 \ , \tag{4.208}$$

where $k_i \in K$, and $k_{i+1} \in K$ is the index of the next successful iteration. Since $\Phi_{\sigma_{k_{i+1}}}(x_{k_{i+1}}, v_{k_{i+1}}) \geq \Phi_{\sigma_{k_i}}(x_{k_{i+1}}, v_{k_{i+1}})$ might happen, we estimate the difference

$$\begin{aligned} & \Phi_{\sigma_{k_{i+1}}}(x_{k_{i+1}}, v_{k_{i+1}}) - \Phi_{\sigma_{k_i}}(x_{k_{i+1}}, v_{k_{i+1}}) \\ =& -\sum_{j\in\mathcal{S}_{k_{i+1}}}\left(v_j^{(k_{i+1})} g_j(x_{k_{i+1}}) - \frac{1}{2}\sigma_{k_{i+1}} g_j(x_{k_{i+1}})^2\right) - \frac{1}{2}\sum_{j\in\overline{\mathcal{S}}_{k_{i+1}}} \frac{{v_j^{(k_{i+1})}}^2}{\sigma_{k_{i+1}}} \\ & + \sum_{j\in\mathcal{S}^-_{k_{i+1}}}\left(v_j^{(k_{i+1})} g_j(x_{k_{i+1}}) - \frac{1}{2}\sigma_{k_i} g_j(x_{k_{i+1}})^2\right) + \frac{1}{2}\sum_{j\in\overline{\mathcal{S}}^-_{k_{i+1}}} \frac{{v_j^{(k_{i+1})}}^2}{\sigma_{k_i}} \end{aligned} \tag{4.209}$$

with

$$\begin{aligned} \mathcal{S}_{k_{i+1}} &:= \mathcal{E} \cup \left\{ j \in \mathcal{I} \mid \ g_j(x_{k_{i+1}}) \leq v_j^{(k_{i+1})}/\sigma_{k_{i+1}} \right\} \quad \text{and} \quad \overline{\mathcal{S}}_{k_{i+1}} := \{1,\ldots,m\} \setminus \mathcal{S}_{k_{i+1}} \ , \\ \mathcal{S}^-_{k_{i+1}} &:= \mathcal{E} \cup \left\{ j \in \mathcal{I} \mid \ g_j(x_{k_{i+1}}) \leq v_j^{(k_{i+1})}/\sigma_{k_i} \right\} \quad \text{and} \quad \overline{\mathcal{S}}^-_{k_{i+1}} := \{1,\ldots,m\} \setminus \mathcal{S}^-_{k_{i+1}} \ . \end{aligned}$$

A difference in (4.209) occurs in case $\sigma_{k_{i+1}} > \sigma_{k_i}$. In the following the different parts of (4.209) are investigated. For $j \in \mathcal{S}_{k_{i+1}} \cup \mathcal{S}^-_{k_{i+1}}$ it follows

$$\begin{aligned} & - v_j^{(k_{i+1})} g_j(x_{k_{i+1}}) + \frac{1}{2}\sigma_{k_{i+1}} g_j(x_{k_{i+1}})^2 + v_j^{(k_{i+1})} g_j(x_{k_{i+1}}) - \frac{1}{2}\sigma_{k_i} g_j(x_{k_{i+1}})^2 \\ =& \frac{1}{2}\left(\sigma_{k_{i+1}} - \sigma_{k_i}\right) g_j(x_{k_{i+1}})^2 \ . \end{aligned} \tag{4.210}$$

We obtain for $j \in \overline{\mathcal{S}}_{k_{i+1}} \cup \overline{\mathcal{S}}^-_{k_{i+1}}$

$$\frac{1}{2}\frac{{v_j^{(k_{i+1})}}^2}{\sigma_{k_i}} - \frac{1}{2}\frac{{v_j^{(k_{i+1})}}^2}{\sigma_{k_{i+1}}} = \frac{1}{2}\frac{\left(\sigma_{k_{i+1}} - \sigma_{k_i}\right)}{\sigma_{k_i}\,\sigma_{k_{i+1}}} {v_j^{(k_{i+1})}}^2 \ . \tag{4.211}$$

There is only one case remaining as other combinations can not occur by the definition of the sets. Let $j \in \mathcal{I}$ be the index of an inequality constraint such that $j \in \mathcal{S}_{k_{i+1}}^{-}$ and $j \in \overline{\mathcal{S}}_{k_{i+1}}$. Then $g_j(x_{k_{i+1}}) > 0$,

$$0 < g_j(x_{k_{i+1}}) \leq v_j^{(k_{i+1})}/\sigma_{k_i} \ , \tag{4.212}$$

and

$$g_j(x_{k_{i+1}}) > v_j^{(k_{i+1})}/\sigma_{k_{i+1}} \geq 0 \tag{4.213}$$

hold according to the definitions of the corresponding sets. We consider the difference of the relevant terms in (4.209), that is

$$v_j^{(k_{i+1})} g_j(x_{k_{i+1}}) - \frac{1}{2}\sigma_{k_i} g_j(x_{k_{i+1}})^2 - \frac{1}{2}\frac{{v_j^{(k_{i+1})}}^2}{\sigma_{k_{i+1}}} \ . \tag{4.214}$$

First, a lower bound for (4.214) is established. We get

$$\begin{aligned}
& v_j^{(k_{i+1})} g_j(x_{k_{i+1}}) - \frac{1}{2}\sigma_{k_i} g_j(x_{k_{i+1}})^2 - \frac{1}{2}\frac{{v_j^{(k_{i+1})}}^2}{\sigma_{k_{i+1}}} \\
\overset{(4.213)}{\geq} \ & v_j^{(k_{i+1})} g_j(x_{k_{i+1}}) - \frac{1}{2}\sigma_{k_i} g_j(x_{k_{i+1}})^2 - \frac{1}{2} v_j^{(k_{i+1})} g_j(x_{k_{i+1}}) \\
= \ & \frac{1}{2}\left(v_j^{(k_{i+1})} g_j(x_{k_{i+1}}) - \sigma_{k_i} g_j(x_{k_{i+1}})^2\right) \\
\overset{(4.212)}{\geq} \ & \frac{1}{2}\left(\sigma_{k_i} g_j(x_{k_{i+1}})^2 - \sigma_{k_i} g_j(x_{k_{i+1}})^2\right) \\
= \ & 0 \ .
\end{aligned} \tag{4.215}$$

Making use of (4.212) and (4.213) yields

$$\frac{1}{2}\sigma_{k_i} g_j(x_{k_{i+1}})^2 \overset{(4.213)}{\geq} \frac{1}{2}\frac{\sigma_{k_i}}{\sigma_{k_{i+1}}} v_j^{(k_{i+1})} g_j(x_{k_{i+1}}) \geq 0 \tag{4.216}$$

and

$$\frac{1}{2}\frac{{v_j^{(k_{i+1})}}^2}{\sigma_{k_{i+1}}} \overset{(4.212)}{\geq} \frac{1}{2}\frac{\sigma_{k_i}}{\sigma_{k_{i+1}}} v_j^{(k_{i+1})} g_j(x_{k_{i+1}}) \geq 0 \ . \tag{4.217}$$

Applying (4.216) and (4.217) to (4.214) gives the upper bound

$$\begin{aligned}
& v_j^{(k_{i+1})} g_j(x_{k_{i+1}}) - \frac{1}{2}\sigma_{k_i} g_j(x_{k_{i+1}})^2 - \frac{1}{2}\frac{{v_j^{(k_{i+1})}}^2}{\sigma_{k_{i+1}}} \\
\overset{(4.216),(4.217)}{\leq} \ & v_j^{(k_{i+1})} g_j(x_{k_{i+1}}) - \frac{1}{2}\frac{\sigma_{k_i}}{\sigma_{k_{i+1}}} v_j^{(k_{i+1})} g_j(x_{k_{i+1}}) - \frac{1}{2}\frac{\sigma_{k_i}}{\sigma_{k_{i+1}}} v_j^{(k_{i+1})} g_j(x_{k_{i+1}}) \\
= \ & v_j^{(k_{i+1})} g_j(x_{k_{i+1}}) - \frac{\sigma_{k_i}}{\sigma_{k_{i+1}}} v_j^{(k_{i+1})} g_j(x_{k_{i+1}}) \ .
\end{aligned} \tag{4.218}$$

Since for the whole sequence $\lim_{k\to\infty} \sigma_k = \sigma$, it follows that $\lim_{k_i\to\infty, k_i\in K}(\sigma_{k_{i+1}} - \sigma_{k_i}) = 0$. Consequently, $\lim_{k_i\to\infty, k_i\in K} \sigma_{k_i}/\sigma_{k_{i+1}} = 1$. The boundedness of $g_j(x_{k_{i+1}})$ and $v_{k_{i+1}}$ imply that (4.210), (4.211), and (4.218) tend to 0. Thus, it follows from the mentioned boundedness, (4.210), (4.211), (4.215), and (4.218) that for sufficiently large $k_i \in K$

$$\Phi_{\sigma_{k_{i+1}}}(x_{k_{i+1}}, v_{k_{i+1}}) - \Phi_{\sigma_{k_i}}(x_{k_{i+1}}, v_{k_{i+1}}) \leq \frac{1}{12}\rho_0\kappa_{\mathrm{lbB}}\epsilon^2 \tag{4.219}$$

holds. Applying (4.208) and (4.219) leads to

$$\Phi_{\sigma_{k_{i+1}}}(x_{k_{i+1}}, v_{k_{i+1}}) \leq \Phi_{\sigma_{k_i}}(x_{k_{i+1}}, v_{k_{i+1}}) + \frac{1}{12}\rho_0\kappa_{\mathrm{lbB}}\epsilon^2 \leq \Phi_{\sigma_{k_i}}(x_{k_i}, v_{k_i}) - \frac{1}{12}\rho_0\kappa_{\mathrm{lbB}}\epsilon^2 \ ,$$

for all sufficiently large $k_i \in K$ and to a contradiction, since $\{\Phi_{\sigma_k}(x_k, v_k)\}$ is bounded from below for the whole sequence, as $\mathcal{X}$ is a compact set according to Assumption 4.2. Thus, assumption (4.203) does not hold and the theorem is shown. □

The next Theorem is the main result of the global convergence analysis. Theorem 4.18 is applied to show that at least one accumulation point of the generated sequence fulfills the Karush-Kuhn-Tucker conditions of the nonlinear problem (1.2).

Theorem 4.19 *Let $\{(x_k, v_k, d_k, u_k, \mu_k, B_k)\}$ be determined by Algorithm 4.1. If Assumption 4.2 and Assumption 4.3 hold, then either Algorithm 4.1 stops at a Karush-Kuhn-Tucker point or there exists an accumulation point $(x^\star, u^\star)$ of the sequence $\{(x_k, u_k)\}$ satisfying the KKT conditions for problem (1.2).*

Proof: Let us consider the case when (d_k, u_k) is the minimizer of subproblem (4.7) and the step size is $\|d_k\|_\infty = 0$. Then Algorithm 4.1 terminates in Step 2. The KKT conditions of subproblem (4.7) are satisfied, i.e.,

$$\begin{aligned} \nabla f(x_k) - \sum_{j=1}^{m} \nabla g_j(x_k) u_j^{(k)} &= 0 \ , \\ g_j(x_k) &= 0 \ , \quad j \in \mathcal{E} \ , \\ g_j(x_k) &\geq 0 \ , \quad j \in \mathcal{I} \ , \\ u_j^{(k)} &\geq 0 \ , \quad j \in \mathcal{I} \ , \\ u_j^{(k)} g_j(x_k) &= 0 \ , \quad j \in \mathcal{I} \ , \end{aligned} \tag{4.220}$$

holds. The iteration k has to be an iteration that follows a successful one, what follows from the convexity of the subproblems. Thus, the trust region constraint can be neglected as the bound satisfies $\Delta_k \geq \Delta_{\min}$. Conditions (4.220) are also the KKT conditions of problem (1.2). Thus, (x_k, u_k) is a KKT point of problem (1.2).

In the following the case is considered when Algorithm 4.1 does not terminate and generates an infinite sequence $\{(x_k, u_k)\}$. Since all iterates x_k lie in $\mathcal{X}$ according to Assumption 4.2 and $\mathcal{X}$ is a compact set, the sequence $\{x_k\}$ is bounded. Moreover, the boundedness of $\{u_k\}$ and Theorem 4.18 guarantee the existence of $x^\star \in \mathbb{R}^n$, $u^\star \in \mathbb{R}^m$,

and an infinite subset $K \subset \mathbb{N}$ with

$$\lim_{k\to\infty, k\in K} x_k = x^\star \ ,$$

$$\lim_{k\to\infty, k\in K} u_k = u^\star$$

such that

$$\lim_{k\to\infty,\, k\in K} \left(\|\nabla f(x_k) - \nabla g(x_k) u_k\|_2 + \mu_k + \|g(x_k)^-\|_1 \right) = 0 \tag{4.221}$$

holds. From (4.221) and $\mu_k \geq 0$, it follows $\|g(x^\star)^-\|_1 = 0$, that is

$$g_j(x^\star) = 0 \ , \quad j \in \mathcal{E} \ ,$$

$$g_j(x^\star) \geq 0 \ , \quad j \in \mathcal{I} \ ,$$

and

$$\|\nabla f(x^\star) - \nabla g(x^\star) u^\star\|_2 = 0 \ .$$

We have to show that

$$u_j^\star g_j(x^\star) = 0 \ , \quad j = 1, \ldots, m \ , \tag{4.222}$$

and $u_j^\star \geq 0$, $j \in \mathcal{I}$, is satisfied. Obviously, (4.222) holds for all $j \in \mathcal{E}$.

From the KKT conditions of the subproblems, i.e., (4.34)(a) or subproblem (4.37)(a), respectively, we derive that

$$\mu_k = \|B_k d_k + \nabla f(x_k) - \nabla g(x_k)^T u_k\|_1 \ ,$$

what follows from the definition (4.35) of μ_k. Together with

$$\lim_{k\to\infty, k\in K} \mu_k = 0 \ ,$$

$$\lim_{k\to\infty, k\in K} \|\nabla f(x_k) - \nabla g(x_k)^T u_k\|_2 = 0 \ ,$$

what follows from (4.221), and $d_k^T B_k d_k \geq \kappa_{\mathrm{lbB}} \|d_k\|_2$, according to Assumption 4.2, we obtain

$$\lim_{k\to\infty, k\in K} \|d_k\|_2 = 0 \ . \tag{4.223}$$

For any $j \in \mathcal{I}$ with $g_j(x^\star) > 0$, it follows for sufficiently large $k \in K$ that $j \in \mathcal{B}_k$ and additionally from (4.223) that

$$g_j(x_k) + \nabla g_j(x_k)^T d_k > 0$$

holds. The KKT conditions (4.34)(f) or (4.37)(h), in case the feasibility restoration phase is entered, imply that $u_j^{(k)} = 0$ and therefore also $u_j^\star = 0$.

Applying the KKT conditions (4.34)(f) and (4.34)(g) or (4.37)(g)-(i), respectively, and

$$\lim_{k\to\infty, k\in K} \|g(x_k)^-\|_1 = 0 \ ,$$

to the active inequality constraints at $x^\star$, we conclude that (4.222) holds and $u_j^\star \geq 0$ for all $j \in \mathcal{I}$. This shows that the KKT conditions (4.220) are also satisfied at the accumulation point $(x^\star, u^\star)$ of the considered subsequence. □

The global convergence analysis for Algorithm 4.1 is completed. The following section presents the local convergence analysis.

4.2.2 Local Convergence

The following local convergence analysis adapts results obtained by Schittkowski [99]. The original analysis was carried out for an algorithm that applies line search techniques. The local convergence analysis uses basic ideas of the work done by Schittkowski [99]. The similar parts are highlighted.

The following analysis assumes that the set of active inequality constraints has already been determined, i.e., the sets $\mathcal{S}_k$, $\overline{\mathcal{S}}_k$, $\mathcal{M}_k$, $\overline{\mathcal{M}}_k$, $\mathcal{L}_k$, and $\overline{\mathcal{L}}_k$ do not change anymore. The following equations

$$\mathcal{S}_k = \mathcal{M}_k = \mathcal{L}_k$$

and

$$\overline{\mathcal{S}}_k = \overline{\mathcal{M}}_k = \overline{\mathcal{L}}_k$$

hold for sufficiently large k. Thus, the local convergence analysis considers equality constrained problems. This can be seen as a restart of the algorithm as soon as this situation occurs. The augmented Lagrangian reduces to the form (4.4). The problem is then formulated as

$$\begin{array}{ll} \underset{x\in\mathbb{R}^n}{\text{minimize}} & f(x) \\ \text{subject to} & g_j(x) = 0\ , \quad j = 1,\ldots,m\ . \end{array} \tag{4.224}$$

This assumption does not seem to be too restrictive and can be presumed without loss of generality. Moreover, it is assumed that some additional properties hold. Let $(x^\star, u^\star)$ denote the KKT point of problem (4.224).

Assumption 4.20

1. There exists a nonempty, convex, and compact set $\mathcal{X} \subset \mathbb{R}^n$ such that for all k the iterate x_k and $x_k + d_k$ lie in $\mathcal{X}$.
2. $f(x)$ and $g_j(x)$, $j = 1,\ldots,m$, are twice continuously differentiable on an open set containing $\mathcal{X}$.
3. $\nabla g(x) = (\nabla g_1(x),\ldots,\nabla g_m(x))$ has full rank on an open set containing $\mathcal{X}$.
4. The second derivatives of all problem functions are Lipschitz-continuous on an open set containing $\mathcal{X}$.
5. The optimal solution $x^\star$ lies in $\mathcal{X}$.
6. $\lim\limits_{k\to\infty} x_k = x^\star$.
7. $\lim\limits_{k\to\infty} v_k = u^\star$.
8. There exists a $\kappa \geq 1$ such that

$$\frac{\|u_k - v_k\|_\infty^2}{\|d_k\|_\infty^2} \leq \kappa \tag{4.225}$$

holds for sufficiently large k.

9. $\{B_k\}$ is bounded.
10. There exists a $\kappa_{\text{lbB}} > 0$ such that for all k

$$\kappa_{\text{lbB}} \|d_k\|_2^2 \leq d_k^T B_k d_k \ . \tag{4.226}$$

Assumption 4.20(1.)-(5.) imply properties that are stated in the following. Without loss of generality, it is assumed that the constant $\kappa \geq 1$ is large enough such that

$$\begin{aligned}
\|\nabla f(x) - \nabla f(y)\|_2 &\leq \kappa \|x - y\|_2 \ , \\
\|\nabla^2 f(x) - \nabla^2 f(y)\|_2 &\leq \kappa \|x - y\|_2 \ , \\
\|\nabla^2 g_j(x) - \nabla^2 g_j(y)\|_2 &\leq \frac{\kappa}{m} \|x - y\|_2 \ , \quad \text{for } j = 1, \ldots, m \ , \\
\|\nabla g(x)\|_2 &\leq \kappa \ , \\
\|\nabla g(x) - \nabla g(y)\|_2 &\leq \kappa \|x - y\|_2 \ , \\
\|\nabla^2 g_j(x)\|_2 &\leq \frac{\kappa}{m} \ , \quad \text{for } j = 1, \ldots, m \ ,
\end{aligned} \tag{4.227}$$

holds for all $x, y \in \mathcal{X}$.

Assumption 4.20(8) is also used by other authors, see for example El-Alem [30] and Gill, Murray, Saunders, and Wright [48].

The local superlinear convergence has been studied by several authors, e.g., Han [58], Boggs, Tolle, and Wang [8], and Powell [88]. It was proved that superlinear convergence requires the use of the unit step length in line search methods, and the acceptance of all trial steps with inactive trust region constraint in trust region methods, i.e., $x_{k+1} = x_k + d_k$ for all sufficiently large k and $\|d_k\|_\infty < \Delta_k$. Thus, the investigations of this section are restricted to the question whether the step calculated by Algorithm 4.1 fulfills $\|d_k\|_\infty < \Delta_k$ and

$$\frac{Ared_k}{Pred_k} \geq \rho_0 \tag{4.228}$$

holds in the neighborhood of a solution for all k sufficiently large.

Let (x_k, v_k) be an iterate of Algorithm 4.1. As problem (4.224) is considered, the augmented Lagrangian reduces to

$$\Phi_{\sigma_k}(x_k, v_k) := f(x_k) - g(x_k)^T v_k + \frac{1}{2} \sigma_k \|g(x_k)\|_2^2 \tag{4.229}$$

and the gradient of the augmented Lagrangian is

$$\nabla \Phi_{\sigma_k}(x_k, v_k) := \begin{pmatrix} \nabla f(x_k) - \nabla g(x_k) v_k + \sigma_k \nabla g(x_k) g(x_k) \\ -g(x_k) \end{pmatrix} \ . \tag{4.230}$$

Assumption 4.20 implies that the augmented Lagrangian $\Phi_\sigma(x, v)$ is now twice continuously differentiable. The model $\Psi_{\sigma_k}(d_k, w_k)$ can easily be derived from (4.16).

We consider the solution of the quadratic problem that corresponds to the equality constrained problem (4.224). Let (x_k, v_k) be an iterate of Algorithm 4.1, then the

quadratic subproblem is

$$\begin{array}{ll} \underset{d\in\mathbb{R}^n}{\text{minimize}} & \nabla f(x_k)^T d + \frac{1}{2} d^T B_k d \\ \text{subject to} & g_j(x_k) + \nabla g_j(x_k)^T d = 0 \ , \quad j = 1, \ldots, m \ , \\ & \|d\|_\infty \leq \Delta_k \ . \end{array} \tag{4.231}$$

Let (d_k, u_k) be a solution to problem (4.231) such that $\|d_k\|_\infty < \Delta_k$, i.e., the trust region bound is not active. Then the corresponding KKT optimality conditions of the subproblem (4.231) can be stated as

$$\begin{array}{ll} \text{a)} & B_k d_k + \nabla f(x_k) - \nabla g(x_k) u_k = 0 \\ \text{b)} & g_j(x_k) + \nabla g_j(x_k)^T d_k = 0 \ , \quad j = 1, \ldots, m \ . \end{array} \tag{4.232}$$

The following lemma states that the predicted reduction $Pred_k$ is related to the gradient of the augmented Lagrangian function (4.230) when multiplied with the trial step. This holds if the trust region bound is inactive at the solution of the quadratic subproblem (4.231).

Lemma 4.21 *Let Assumption 4.20 hold. Let (x_k, v_k) be an iterate of Algorithm 4.1 such that subproblem (4.7) is solved and the solution denoted by (d_k, u_k) satisfies*

$$\|d_k\|_\infty < \Delta_k \ , \tag{4.233}$$

then

$$Pred_k = -\frac{1}{2} \nabla \Phi_{\sigma_k}(x_k, v_k)^T \begin{pmatrix} d_k \\ u_k - v_k \end{pmatrix} \ . \tag{4.234}$$

Proof: We consider the gradient of the augmented Lagrangian (4.230). We get

$$\begin{aligned} -\nabla \Phi_{\sigma_k}(x_k, v_k)^T \begin{pmatrix} d_k \\ u_k - v_k \end{pmatrix} &= -\begin{pmatrix} \nabla f(x_k) - \nabla g(x_k) v_k + \sigma_k \nabla g(x_k) g(x_k) \\ -g(x) \end{pmatrix}^T \begin{pmatrix} d_k \\ u_k - v_k \end{pmatrix} \\ &= -d_k^T \nabla f(x_k) + d_k^T \nabla g(x_k) v_k - \sigma_k d_k^T \nabla g(x_k) g(x_k) + (u_k - v_k)^T g(x_k) \ . \end{aligned} \tag{4.235}$$

Since subproblem (4.7) is solved the step in the dual variables is set to $w_k = u_k - v_k$. As $\|d_k\|_\infty < \Delta_k$, the KKT system (4.232) can be applied. With $\Psi_{\sigma_k}(0,0) = \Phi_{\sigma_k}(x_k, v_k)$, we obtain for the predicted reduction

$$\begin{aligned} Pred_k = \quad & \Psi_{\sigma_k}(0,0) - \Psi_{\sigma_k}(d_k, w_k) = \Phi_{\sigma_k}(x_k, v_k) - \Psi_{\sigma_k}(d_k, w_k) \\ = \quad & f(x_k) - \sum_{j=1}^m \left(v_j^{(k)} g_j(x_k) - \frac{1}{2} \sigma_k g_j(x_k)^2 \right) - f(x_k) - \nabla f(x_k)^T d_k - \frac{1}{2} d_k^T B_k d_k \\ & + \sum_{j=1}^m \left(\left(v_j^{(k)} + w_j^{(k)} \right) \underbrace{\left(g_j(x_k) + \nabla g_j(x_k)^T d_k \right)}_{=0} - \frac{1}{2} \sigma_k \underbrace{\left(g_j(x_k) + \nabla g_j(x_k)^T d_k \right)^2}_{=0} \right) \end{aligned}$$

$$
\begin{aligned}
&\overset{(4.232)(b)}{=} -\nabla f(x_k)^T d_k - \frac{1}{2} d_k^T B_k d_k - \sum_{j=1}^{m} \left(v_j^{(k)} g_j(x_k) - \frac{1}{2}\sigma_k g_j(x_k)^2 \right) \\
&\overset{(4.232)(a)}{=} -\nabla f(x_k)^T d_k + \frac{1}{2}\left(\nabla f(x_k)^T d_k - \sum_{j=1}^{m} u_j^{(k)} \nabla g_j(x_k)^T d_k \right) \\
&\qquad - \sum_{j=1}^{m} \left(v_j^{(k)} g_j(x_k) - \frac{1}{2}\sigma_k g_j(x_k)^2 \right) \\
&= -\frac{1}{2}\nabla f(x_k)^T d_k - \frac{1}{2}\sum_{j=1}^{m} u_j^{(k)} \nabla g_j(x_k)^T d_k - \frac{1}{2}\sum_{j=1}^{m} v_j^{(k)} g_j(x_k) - \frac{1}{2}\sum_{j=1}^{m} v_j^{(k)} g_j(x_k) \\
&\qquad + \frac{1}{2}\sigma_k \sum_{j=1}^{m} g_j(x_k)^2 \\
&\overset{(4.232)(b)}{=} -\frac{1}{2}\nabla f(x_k)^T d_k + \frac{1}{2}\sum_{j=1}^{m} v_j^{(k)} \nabla g_j(x_k)^T d_k + \frac{1}{2}\sum_{j=1}^{m} u_j^{(k)} g_j(x_k) - \frac{1}{2}\sum_{j=1}^{m} v_j^{(k)} g_j(x_k) \\
&\qquad + \frac{1}{2}\sigma_k \sum_{j=1}^{m} g_j(x_k) \nabla g_j(x_k)^T d_k \\
&\overset{(4.235)}{=} -\frac{1}{2}\nabla \Phi_{\sigma_k}(x_k, v_k)^T \begin{pmatrix} d_k \\ u_k - v_k \end{pmatrix} ,
\end{aligned}
$$

what proves the lemma. □

Now it is shown that all steps are accepted in Algorithm 4.1 and $\|d_k\|_\infty < \Delta_k$ when approaching a solution regardless of the choice of the penalty parameter σ_k.

Theorem 4.22 *Let Assumption 4.20 hold. Let $\{(x_k, v_k)\}$ be an iteration sequence of Algorithm 4.1 and $(x^\star, u^\star)$ be a Karush-Kuhn-Tucker point of problem (4.224). B_k is sufficiently close to $\nabla^2_{xx} L(x^\star, u^\star)$ in the following sense*

$$
d_k^T (\nabla^2_{xx} L(x^\star, u^\star) - B_k) d_k \leq \nu \|d_k\|_2^2 , \tag{4.236}
$$

for sufficiently large k, where

$$
\nu \leq \min\left(\frac{\kappa_{\mathrm{lbB}}(1-\rho_0)^2}{120\kappa^2}, \frac{\kappa_{\mathrm{lbB}}}{2m\kappa}, \frac{\Delta_{\min}}{2} \right) . \tag{4.237}
$$

If there exists an iteration $\bar{k}$ such that $\sigma_k = \sigma_{\bar{k}}$ for all $k \geq \bar{k}$, then ν also satisfies

$$
\nu \leq \frac{1}{\sigma_{\bar{k}}} . \tag{4.238}
$$

Then the following holds for all k sufficiently large

$$
\frac{Ared_k}{Pred_k} \geq \rho_0 \tag{4.239}
$$

and

$$
\|d_k\|_\infty < \Delta_k . \tag{4.240}
$$

Proof: Assumption 4.20 implies that for all k sufficiently large

$$\begin{aligned} \|x_k - x^\star\|_2 &\leq \nu \ , \\ \|v_k - u^\star\|_2 &\leq \nu \ , \\ \|u_k - u^\star\|_2 &\leq \nu \ , \\ \|d_k\|_2 &\leq \nu \end{aligned} \tag{4.241}$$

holds and subproblem (4.231) is consistent for sufficiently large Δ_k. The existence of a solution to subproblem (4.231) follows for sufficiently large k from the full rank of $\nabla g(x_k)$ and from the convergence to a KKT point. In this case the solution d_k and u_k of the quadratic subproblems (4.231) are uniquely determined if the trust region constraint is not active. This follows from the positive definiteness of the matrices B_k.

Without loss of generality, we can assume that the constant κ from Assumption 4.20 also satisfies the requirements

$$\begin{aligned} \|u_k\|_2 &\leq \kappa \ , \quad \text{for all } k \ , \\ \|u^\star\|_2 &\leq \kappa \ . \end{aligned}$$

Now we consider an iterate (x_k, v_k) that follows a successful iteration, i.e., $x_k = x_{k-1} + d_k$. In this situation the trust region radius is at least $\Delta_{\min}$, i.e., $\Delta_k \geq \Delta_{\min}$. We assume that k is sufficiently large, that is (4.241) and Assumption 4.20(8) hold. According to the definition of ν (4.237) and the bound on $\|d_k\|_2$ (4.241), the trust region constraint is inactive at the solution to subproblem (4.231), that is $\|d_k\|_\infty \leq \|d_k\|_2 \leq \Delta_{\min}/2 < \Delta_k$, and thus μ_k (4.35) is equal to zero. Moreover, we obtain $z_j^{(k)} = 0$ for all $j = 1, \ldots, m$.

As a first step, we define a matrix

$$C_k := \begin{pmatrix} B_k + \sigma_k \nabla g(x_k) \nabla g(x_k)^T & -\nabla g(x_k) \\ -\nabla g(x_k)^T & 0 \end{pmatrix} \ . \tag{4.242}$$

By applying the optimality conditions (4.232) of subproblem (4.231), we obtain for $p_k := \begin{pmatrix} d_k \\ u_k - v_k \end{pmatrix}$

$$\begin{aligned} C_k p_k = & \begin{pmatrix} B_k d_k + \sigma_k \nabla g(x_k) \nabla g(x_k)^T d_k - \nabla g(x_k)(u_k - v_k) \\ -\nabla g(x_k)^T d_k \end{pmatrix} \\ \overset{(4.232)}{=} & \begin{pmatrix} -\nabla f(x_k) + \nabla g(x_k) u_k - \sigma_k \nabla g(x_k) g(x_k) - \nabla g(x_k)(u_k - v_k) \\ g(x_k) \end{pmatrix} \\ = & \ -\nabla \Phi_{\sigma_k}(x_k, v_k) \ . \end{aligned} \tag{4.243}$$

We first estimate some bounds that are applied in the proof later. To simplify the notation, the iteration index k is dropped now. The following estimates can also be

found in Schittkowski [99]. For a $\xi \in (0,1]$ we can estimate the following bound.

$$
\begin{aligned}
& |d^T(\nabla^2_{xx} L(x+\xi d, v+\xi(u-v)) - B)d| \\
\leq\ & |d^T(\nabla^2_{xx} L(x+\xi d, v+\xi(u-v)) - \nabla^2_{xx} L(x^\star,u^\star))d| \\
& + |d^T(\nabla^2_{xx} L(x^\star,u^\star) - B)d| \\
\overset{(4.236)}{\leq}\ & \|d\|_2^2 \Big(\|\nabla^2 f(x+\xi d) - \nabla^2 f(x^\star)\|_2 \\
& \qquad + \Big\| \sum_{j=1}^m \Big((v_j + \xi(u_j - v_j))\nabla^2 g_j(x+\xi d) - u_j^\star \nabla^2 g_j(x^\star) \Big) \Big\|_2 + \nu \Big) \\
=\ & \|d\|_2^2 \Big(\|\nabla^2 f(x+\xi d) - \nabla^2 f(x^\star)\|_2 \\
& \qquad + \Big\| \sum_{j=1}^m \Big((v_j + \xi(u_j - v_j))\nabla^2 g_j(x+\xi d) - u_j^\star \nabla^2 g_j(x+\xi d) \\
& \qquad\quad + u_j^\star \nabla^2 g_j(x+\xi d) - u_j^\star \nabla^2 g_j(x^\star) \Big) \Big\|_2 + \nu \Big) \\
=\ & \|d\|_2^2 \Big(\|\nabla^2 f(x+\xi d) - \nabla^2 f(x^\star)\|_2 \\
& \qquad + \Big\| \sum_{j=1}^m \Big((v_j + \xi(u_j - v_j) - u_j^\star)\nabla^2 g_j(x+\xi d) \\
& \qquad\quad + u_j^\star \left(\nabla^2 g_j(x+\xi d) - \nabla^2 g_j(x^\star) \right) \Big) \Big\|_2 + \nu \Big) \\
\leq\ & \|d\|_2^2 \Big(\|\nabla^2 f(x+\xi d) - \nabla^2 f(x^\star)\|_2 + \sum_{j=1}^m \Big\| (v_j + \xi(u_j - v_j) - u_j^\star)\nabla^2 g_j(x+\xi d) \Big\|_2 \\
& \qquad + \sum_{j=1}^m \Big\| u_j^\star \left(\nabla^2 g_j(x+\xi d) - \nabla^2 g_j(x^\star) \right) \Big\|_2 + \nu \Big) \\
\overset{(4.227)}{\leq}\ & \kappa \|d\|_2^2 (\|x - x^\star\|_2 + \|d\|_2) + \kappa \|d\|_2^2 \|v + \xi(u-v) - u^\star\|_2 \\
& + \kappa \|d\|_2^2 \|u^\star\|_2 (\|x^\star - x\|_2 + \|d\|_2) + \nu \|d\|_2^2 \\
\leq\ & (2\kappa^2 + 5\kappa + 1)\nu \|d\|_2^2 \ . \qquad (4.244)
\end{aligned}
$$

The last inequality is obtained by applying (4.241). Then there also exist $\xi_j' \in (0,\xi)$, $j = 1,\ldots,m$, such that

$$
\begin{aligned}
g_j(x+\xi d) &= g_j(x) + \xi \nabla g_j(x)^T d + \frac{1}{2}\xi^2 d^T \nabla^2 g_j(x+\xi_j' d) d \\
&= (1-\xi) g_j(x) + \frac{1}{2}\xi^2 d^T \nabla^2 g_j(x+\xi_j' d) d \ ,
\end{aligned}
$$

and we get by applying (4.227)

$$
\|g(x+\xi d)\|_2 \leq \|g(x)\|_2 + \kappa \|d\|_2^2 \ . \qquad (4.245)
$$

From

$$\begin{aligned}\|\nabla g(x+\xi d)^T d\|_2 - \|\nabla g(x)^T d\|_2 &\leq \|\nabla g(x+\xi d)^T d - \nabla g(x)^T d\|_2 \\ &\leq \|\nabla g(x+\xi d) - \nabla g(x)\|_2 \|d\|_2 \\ &\leq \kappa \|x+\xi d - x\|_2 \|d\|_2 \leq \kappa \|d\|_2^2 \ ,\end{aligned}$$

where we applied (4.227) in the next to last step, it follows

$$\begin{aligned}\|\nabla g(x+\xi d)^T d\|_2 &\leq \|\nabla g(x)^T d\|_2 + \kappa \|d\|_2^2 \\ &= \|g(x)\|_2 + \kappa \|d\|_2^2 \ . \end{aligned} \tag{4.246}$$

From Lemma 4.21 we know that $Pred = -\frac{1}{2}\nabla \Phi_\sigma(x,v)^T p$. According to Theorem 4.9, we have the lower bound on the predicted reduction

$$\begin{aligned}-\frac{1}{2}\nabla \Phi_\sigma(x,v)^T p = Pred &\geq \frac{1}{6}\left(d^T B d + 2\mu\Delta\right) + \frac{1}{8}\sigma \sum_{j=1}^{m} g_j(x)^2 \left(1 - {z_j}^2\right) \\ &\geq \frac{1}{6} d^T B d + \frac{1}{8}\sigma \|g(x)\|_2^2 \ , \end{aligned} \tag{4.247}$$

where $\mu = 0$ and $z_j = 0$, $j = 1, \ldots, m$, is applied.

Thus, we get

$$-\nabla \Phi_\sigma(x,v)^T p \geq \frac{1}{3} d^T B d + \frac{1}{4}\sigma \|g(x)\|_2^2 \geq \frac{1}{3}\kappa_{\mathrm{lbB}} \|d\|_2^2 + \frac{1}{4}\sigma \|g(x)\|_2^2 \ , \tag{4.248}$$

where we applied $d^T B d \geq \kappa_{\mathrm{lbB}} \|d\|_2^2$ according to Assumption 4.20(10.).

By the use of the following definitions

$$\nabla^2 \Phi_\sigma(y) := \begin{pmatrix} \nabla^2_{xx} L(x,v) + \sigma \nabla g(x) \nabla g(x)^T + \sigma(\nabla^2 g(x), g(x)) & -\nabla g(x) \\ -\nabla g(x)^T & 0 \end{pmatrix}$$

for all $y := \begin{pmatrix} x \\ v \end{pmatrix} \in \mathbb{R}^{n+m}$,

$$(\nabla^2 g(x), g(x)) := \sum_{j=1}^{m} g_j(x) \nabla^2 g_j(x) \ ,$$

the definition (4.242) of C, and (4.243), we obtain

$$\begin{aligned} & p^T (\nabla^2 \Phi_\sigma(y+\xi p) - C) p \\ = \quad & d^T (\nabla^2_{xx} L(x+\xi d, v + \xi(u-v)) - B) d + \sigma \|\nabla g(x+\xi d)^T d\|_2^2 \\ & + \sigma d^T (\nabla^2 g(x+\xi d), g(x+\xi d)) d - \sigma \|\nabla g(x)^T d\|_2^2 \\ & - 2 d^T (\nabla g(x+\xi d) - \nabla g(x))(u-v) \end{aligned}$$

$$
\begin{aligned}
\leq\quad & |d^T(\nabla^2_{xx}L(x+\xi d, v+\xi(u-v)) - B)d| + \sigma\|\nabla g(x+\xi d)^T d\|_2^2 \\
& + \sigma d^T(\nabla^2 g(x+\xi d), g(x+\xi d))d - \sigma\|\nabla g(x)^T d\|_2^2 \\
& - 2d^T(\nabla g(x+\xi d) - \nabla g(x))(u-v) \\
\overset{(4.244)}{\leq}\quad & (2\kappa^2+5\kappa+1)\nu\|d\|_2^2 + \sigma\|\nabla g(x+\xi d)^T d\|_2^2 \\
& + \sigma d^T(\nabla^2 g(x+\xi d), g(x+\xi d))d - \sigma\|\nabla g(x)^T d\|_2^2 \\
& - 2d^T(\nabla g(x+\xi d) - \nabla g(x))(u-v) \\
\overset{(4.246)}{\leq}\quad & (2\kappa^2+5\kappa+1)\nu\|d\|_2^2 + \sigma\|g(x)\|_2^2 + 2\sigma\kappa\|g(x)\|_2\|d\|_2^2 + \sigma\kappa^2\|d\|_2^4 \\
& + \sigma d^T(\nabla^2 g(x+\xi d), g(x+\xi d))d - \sigma\|\nabla g(x)^T d\|_2^2 \\
& - 2d^T(\nabla g(x+\xi d) - \nabla g(x))(u-v) \\
\overset{(4.227),(4.245)}{\leq}\quad & (2\kappa^2+5\kappa+1)\nu\|d\|_2^2 + \sigma\|g(x)\|_2^2 + 2\sigma\kappa\|g(x)\|_2\|d\|_2^2 + \sigma\kappa^2\|d\|_2^4 \\
& + \sigma\kappa\|d\|_2^2(\|g(x)\|_2 + \kappa\|d\|_2^2) - \sigma\|\nabla g(x)^T d\|_2^2 \\
& - 2d^T(\nabla g(x+\xi d) - \nabla g(x))(u-v) \\
\overset{(4.232)(b)}{\leq}\quad & (2\kappa^2+5\kappa+1)\nu\|d\|_2^2 + \sigma\|g(x)\|_2^2 + 2\sigma\kappa\|g(x)\|_2\|d\|_2^2 + \sigma\kappa^2\|d\|_2^4 \\
& + \sigma\kappa\|d\|_2^2(\|g(x)\|_2 + \kappa\|d\|_2^2) - \sigma\|g(x)\|_2^2 \\
& - 2d^T(\nabla g(x+\xi d) - \nabla g(x))(u-v) \\
\overset{(4.227)}{\leq}\quad & (2\kappa^2+5\kappa+1)\nu\|d\|_2^2 + \sigma\|g(x)\|_2^2 + 2\sigma\kappa\|g(x)\|_2\|d\|_2^2 + \sigma\kappa^2\|d\|_2^4 \\
& + \sigma\kappa\|d\|_2^2(\|g(x)\|_2 + \kappa\|d\|_2^2) - \sigma\|g(x)\|_2^2 \\
& + 2\kappa\|d\|_2^2\|u-v\|_2 \\
=\quad & (2\kappa^2+5\kappa+1)\nu\|d\|_2^2 + 2\sigma\kappa\|g(x)\|_2\|d\|_2^2 + \sigma\kappa^2\|d\|_2^4 \\
& + \sigma\kappa\|d\|_2^2(\|g(x)\|_2 + \kappa\|d\|_2^2) + 2\kappa\|d\|_2^2\|u-u^\star+u^\star-v\|_2 \\
\overset{(4.241)}{\leq}\quad & (2\kappa^2+9\kappa+1)\nu\|d\|_2^2 + \sigma(3\kappa\|g(x)\|_2 + 2\kappa^2\|d\|_2^2)\|d\|_2^2 \\
\leq\quad & 12\kappa^2\nu\|d\|_2^2 + \sigma(3\kappa\|g(x)\|_2 + 2\kappa^2\|d\|_2^2)\|d\|_2^2 \ . \qquad (4.249)
\end{aligned}
$$

The last inequality follows as $\kappa \geq 1$. Now we adapt the estimates of Schittkowski [99].

We consider the condition for accepting a trial step, that is

$$\frac{Ared}{Pred} \geq \rho_0 \ .$$

This can be rewritten as

$$Ared - \rho_0 Pred = \Phi_\sigma(y) - \Phi_\sigma(y+p) - \rho_0 Pred \geq 0 \ .$$

Lemma 4.21 provides $Pred = -\frac{1}{2}\nabla\Phi_\sigma(y)^T p$. We define a constant $\bar{\rho} := \frac{1}{2}\rho_0$. If we apply the Taylor-approximation of Φ_σ with a $\xi \in (0,1]$, then we obtain

$$
\begin{aligned}
& \Phi_\sigma(y) - \Phi_\sigma(y+p) - 2\bar{\rho}Pred \\
= \quad & \Phi_\sigma(y) - \Phi_\sigma(y+p) + \bar{\rho}\nabla\Phi_\sigma(y)^T p \\
= \quad & -\nabla\Phi_\sigma(y)^T p - \frac{1}{2}p^T\nabla^2\Phi_\sigma(y+\xi p)p + \bar{\rho}\nabla\Phi_\sigma(y)^T p \\
= \quad & (\bar{\rho}-1)\nabla\Phi_\sigma(y)^T p - \frac{1}{2}p^T\nabla^2\Phi_\sigma(y+\xi p)p \\
\overset{(4.243)}{=} \quad & -\left(\frac{1}{2}-\bar{\rho}\right)\nabla\Phi_\sigma(y)^T p - \frac{1}{2}p^T(\nabla^2\Phi_\sigma(y+\xi p) - C)p \\
\overset{(4.248),(4.249)}{\geq} \quad & \left(\frac{1}{2}-\bar{\rho}\right)\left(\frac{1}{3}\kappa_{\mathrm{lbB}}\|d\|_2^2 + \frac{1}{4}\sigma\|g(x)\|_2^2\right) - 6\kappa^2\nu\|d\|_2^2 \\
& -\frac{1}{2}\sigma\left(3\kappa\|g(x)\|_2 + 2\kappa^2\|d\|_2^2\right)\|d\|_2^2 \\
= \quad & \left(\frac{1}{3}\left(\frac{1}{2}-\bar{\rho}\right)\kappa_{\mathrm{lbB}} - 6\kappa^2\nu\right)\|d\|_2^2 + \frac{1}{4}\left(\frac{1}{2}-\bar{\rho}\right)\sigma\|g(x)\|_2^2 \\
& -\frac{3}{2}\sigma\kappa\|g(x)\|_2\|d\|_2^2 - \sigma\kappa^2\|d\|_2^4 \\
= \quad & \left(\frac{1}{3}\left(\frac{1}{2}-\bar{\rho}\right)\kappa_{\mathrm{lbB}} - 6\kappa^2\nu\right)\|d\|_2^2 + \frac{1}{4}\left(\frac{1}{2}-\bar{\rho}\right)\sigma\|g(x)\|_2^2 \\
& -\frac{3}{2}\sigma\kappa\|g(x)\|_2\|d\|_2^2 + \sigma\frac{9\kappa^2}{4(\frac{1}{2}-\bar{\rho})}\|d\|_2^4 - \sigma\frac{9\kappa^2}{4(\frac{1}{2}-\bar{\rho})}\|d\|_2^4 - \sigma\kappa^2\|d\|_2^4 \\
= \quad & \left(\frac{1}{3}\left(\frac{1}{2}-\bar{\rho}\right)\kappa_{\mathrm{lbB}} - 6\kappa^2\nu\right)\|d\|_2^2 \\
& +\sigma\left(\frac{1}{2}\sqrt{\frac{1}{2}-\bar{\rho}}\,\|g(x)\|_2 - \frac{3\kappa}{2\sqrt{\frac{1}{2}-\bar{\rho}}}\|d\|_2^2\right)^2 \\
& -\sigma\frac{9\kappa^2}{4(\frac{1}{2}-\bar{\rho})}\|d\|_2^4 - \sigma\kappa^2\|d\|_2^4 \\
\geq \quad & \left(\frac{1}{3}\left(\frac{1}{2}-\bar{\rho}\right)\kappa_{\mathrm{lbB}} - 6\kappa^2\nu\right)\|d\|_2^2 - \sigma\left(\frac{9}{4(\frac{1}{2}-\bar{\rho})}+1\right)\kappa^2\|d\|_2^4\,. \qquad (4.250)
\end{aligned}
$$

We need a bound for $\sigma\|d\|_2^2$ before we can continue. We consider two situations. We assume that in the current iteration the penalty parameter is increased and Assumption 4.20(8) holds. Then we obtain with (4.241) and the penalty update formula (4.24) the estimate

$$
\sigma\|d\|_2^2 \leq \sigma\nu^2 \leq \frac{2m\|u-v\|_\infty^2}{d^T B d}\nu^2 \leq \frac{2m\|u-v\|_\infty^2}{\kappa_{\mathrm{lbB}}\|d\|_2^2}\nu^2 \leq \frac{2m\|u-v\|_\infty^2}{\kappa_{\mathrm{lbB}}\|d\|_\infty^2}\nu^2 \leq \frac{2m\kappa}{\kappa_{\mathrm{lbB}}}\nu^2 \leq \nu\,, \qquad (4.251)
$$

where we also applied Assumption 4.20(10) and $\nu \leq \kappa_{\text{lbB}}/(2m\kappa)$ according to (4.237). The iteration index is reintroduced. The bound in (4.251) remains valid if the penalty parameter has been increased in a previous iteration l, with $l < k$, where Assumption 4.20(8) is satisfied as this also results in $\sigma_l \leq 2m\kappa/\kappa_{\text{lbB}}$, what follows the same way as in (4.251) where only the last inequality on the right-hand side is ignored. Thus, the case when the penalty parameter is not increased and $\sigma_k \leq 2m\kappa/\kappa_{\text{lbB}}$ is already included in the estimate (4.251). If $\sigma_k > 2m\kappa/\kappa_{\text{lbB}}$, then the penalty parameter will not be increased anymore for all k sufficiently large. This implies that there exists an iteration $\bar{k}$ such that $\sigma_k = \sigma_{\bar{k}}$ for all $k \geq \bar{k}$. This case is considered now.

If there exists an iteration $\bar{k}$ such that the penalty parameter $\sigma_k = \sigma_{\bar{k}}$ for all $k \geq \bar{k}$, then we can assume without loss of generality that the currently considered iteration k satisfies $k \geq \bar{k}$. We get

$$\sigma_{\bar{k}} \|d_k\|_2^2 \leq \sigma_{\bar{k}} \nu^2 \leq \nu \ , \tag{4.252}$$

where we applied $\nu \leq 1/\sigma_{\bar{k}}$ according to (4.238). The iteration index k is dropped again. We proceed from (4.250) where the estimates of both situations, i.e., inequalities (4.251) and (4.252), result in

$$\begin{aligned}
& \Phi_\sigma(y) - \Phi_\sigma(y+p) - 2\bar{\rho} Pred \\
\geq\ & \left(\frac{1}{3}\left(\frac{1}{2} - \bar{\rho}\right)\kappa_{\text{lbB}} - 6\kappa^2\nu\right)\|d\|_2^2 - \sigma\left(\frac{9}{4(\frac{1}{2}-\bar{\rho})} + 1\right)\kappa^2\|d\|_2^4 \\
\overset{(4.251),(4.252)}{\geq}\ & \left(\frac{1}{3}\left(\frac{1}{2} - \bar{\rho}\right)\kappa_{\text{lbB}} - 6\kappa^2\nu - \left(\frac{9}{4(\frac{1}{2}-\bar{\rho})} + 1\right)\kappa^2\nu\right)\|d\|_2^2 \\
=\ & \left(\frac{1}{3}\left(\frac{1}{2} - \bar{\rho}\right)\kappa_{\text{lbB}} - \left(6 + \left(\frac{9}{4(\frac{1}{2}-\bar{\rho})} + 1\right)\right)\kappa^2\nu\right)\|d\|_2^2 \\
=\ & \left(\frac{1}{3}\left(\frac{1}{2} - \bar{\rho}\right)\kappa_{\text{lbB}} - \left(7 + \frac{9}{4(\frac{1}{2}-\bar{\rho})}\right)\kappa^2\nu\right)\|d\|_2^2 \\
\overset{\rho_o = 2\bar{\rho}}{=}\ & \left(\frac{1}{6}(1-\rho_0)\kappa_{\text{lbB}} - \left(7 + \frac{9}{2(1-\rho_0)}\right)\kappa^2\nu\right)\|d\|_2^2 \\
\overset{(4.237)}{\geq}\ & \left(\frac{1}{6}(1-\rho_0)\kappa_{\text{lbB}} - \left(7 + \frac{9}{2(1-\rho_0)}\right)\kappa^2\frac{\kappa_{\text{lbB}}(1-\rho_0)^2}{120\kappa^2}\right)\|d\|_2^2 \\
=\ & \left(\frac{1}{6}(1-\rho_0)\kappa_{\text{lbB}} - \frac{7}{120}\kappa_{\text{lbB}}(1-\rho_0)^2 - \frac{9}{240(1-\rho_0)}\kappa_{\text{lbB}}(1-\rho_0)^2\right)\|d\|_2^2 \\
\geq\ & \left(\frac{1}{6}(1-\rho_0)\kappa_{\text{lbB}} - \frac{7}{120}(1-\rho_0)\kappa_{\text{lbB}} - \frac{9}{240}(1-\rho_0)\kappa_{\text{lbB}}\right)\|d\|_2^2 \\
=\ & \left(\frac{1}{6} - \frac{7}{120} - \frac{9}{240}\right)(1-\rho_0)\kappa_{\text{lbB}}\|d\|_2^2 \ = \ \frac{40-14-9}{240}(1-\rho_0)\kappa_{\text{lbB}}\|d\|_2^2 \\
\geq\ & 0 \ .
\end{aligned} \tag{4.253}$$

We also applied $(1-\rho_0)^2 \leq (1-\rho_0)$, as $0 < \rho_0 < 1$, and $\nu \leq \kappa_{\text{lbB}}(1-\rho_0)^2/(120\kappa^2)$

according to (4.237).

We reintroduce the iteration index k. It follows from (4.253) that the current iteration k, that follows a successful iteration, is a successful one and the trial step is accepted, i.e.,

$$\frac{Ared_k}{Pred_k} \geq \rho_0 \ .$$

Moreover, $\|d_k\|_\infty < \Delta_k$ holds. As all required conditions still hold for the following iteration $k+1$, the trial step (d_{k+1}, w_{k+1}) will also be accepted. Again $\|d_{k+1}\| < \Delta_{k+1}$, as $\Delta_{k+1} \geq \Delta_{\min}$. It follows by induction that for all k sufficiently large the iteration is successful and the trust region bound remains inactive. This proves the theorem. □

The local convergence is completed. Under adequate assumptions, it has been shown that Algorithm 4.1 accepts all trial steps and the trust region bound is inactive as soon as the sequence of iterates is close to the stationary point $(x^\star, u^\star)$.

4.3 Discussion

The so-called Maratos effect [71] can slow down the local convergence of an SQP method. The problem occurs as the constraints are only linearized in the quadratic subproblem and second order information is contained merely in the matrix that approximates the Hessian of the Lagrangian function. This lack of second order information can lead to a rejection of the calculated trial step if the progress is measured by a merit function that is not differentiable at the solution of the optimized problem. Different strategies were developed to overcome this drawback. The most frequently used techniques apply second order correction steps, non-monotone strategies for penalty functions or a differentiable merit function.

Chamberlain et al. [18] proposed the so-called watch-dog technique to avoid the Maratos effect. For some steps the applied merit function may increase. Non-monotone techniques are very similar to the watch-dog approach, see, for example, Gould and Toint [53]. The basic idea of non-monotone strategies goes back to Grippo, Lampariello, and Lucidi [55]. The technique was extended to constrained optimization and trust region methods in a series of subsequent papers, see, e.g., Toint [119, 120], and Ulbrich and Ulbrich [122]. Here the requirement that $P(x_k + d_k)$ has to be sufficiently less than $P(x_k)$ is relaxed, and a non-monotone sequence of $P(x_k)$ is accepted.

Calculating second order correction steps was proposed by several authors, see, e.g., Fletcher [39] and Yuan [129] for details and convergence analysis. Fletcher [39] has shown that the SOC steps circumvent the Maratos effect. Mayne and Polak [73], Yuan [130], and Fukushima [45] also apply second order correction steps. Methods, that require the calculation of second order correction steps to retain fast local convergence, have a significant disadvantage. The number of function evaluations increases so that they may not be applicable to real-world problems, where the function evaluations are time-consuming.

The aim of Algorithm 4.1 is to avoid the calculation of second order correction steps.

Under suitable assumptions, it has been shown that full steps are accepted close to the solution and the trust region constraints is inactive. Thus, fast local convergence can be expected without additional safeguards. Algorithm 4.1 differs from other trust region algorithm as it applies a differentiable augmented Lagrangian merit function. In line search algorithms the differentiable merit function is commonly used, see, for example, Powell and Yuan [92] for equality constrained problems, or Gill, Murray, Saunders, and Wright [48] for inequality constrained problems, or Schittkowski [99, 100] for problems with both kinds of constraints. In case the considered problems contain inequality constraints, then many trust region algorithms use a merit function that is not differentiable, see, e.g., Yuan [129, 130].

Algorithm 4.1 addresses problems with equality and inequality constraints. The inequality constraints are not transformed into equality constraints. Powell and Yuan [93], and El-Alem [30, 31] also employ an augmented Lagrangian in their trust region algorithms, but the underlying problems contain only equality constraints. In Niu and Yuan [76] the augmented Lagrangian is applied in a trust region algorithm to equality and inequality constrained problems, but the inequalities are transformed into equality constraints by adding slack variables.

Inequalities are also considered by El-Alem and El-Sobky [32]. They transform the problem (1.2) into a equality constrained problem

$$\begin{array}{ll} \underset{x\in\mathbb{R}^n}{\text{minimize}} & f(x) + u_{\mathcal{I}}^T g_{\mathcal{I}}(x) + \dfrac{1}{2}\sigma\|W(x)g_{\mathcal{I}}(x)\|_2^2 \\ \text{subject to} & g_{\mathcal{E}}(x) = 0 \ , \end{array} \tag{4.254}$$

where $u_{\mathcal{I}} \in \mathbb{R}^{m-m_e}$ is the Lagrange multiplier vector corresponding to $g_{\mathcal{I}}(x)$ and $W(x) \in \mathbb{R}^{(m-m_e)\times(m-m_e)}$ is a diagonal matrix that indicates active inequality constraints with diagonal entries

$$W_{(j-m_e)(j-m_e)}(x) := \begin{cases} 1 & \text{, if } \ g_j(x) \leq 0 \ , \\ 0 & \text{, if } \ g_j(x) > 0 \ , \end{cases}$$

for $j = m_e+1, \ldots, m$. The augmented Lagrangian function is then applied to problem (4.254).

In the following some comments on specific parts of Algorithm 4.1 are stated. In Step 1 of Algorithm 4.1 it is tried to solve the standard quadratic problem (4.7). Adding the trust region constraint to the quadratic programming subproblem may lead to infeasible subproblems as there may be no intersection of the trust region constraint and the hyperplane of the linearized constraints. Even if they intersect, there is no guarantee that this will remain true if the trust-region radius is decreased. If no solution to problem (4.7) exists, then a feasibility restoration phase is entered. In this situation the two problems (4.9) and (4.10) are solved to obtain a new trial step. The first subproblem reduces the constraint violation, whereas the second one leads to progress in the objective function. Decomposing the trial steps into two steps and determining the steps separately is also used by other approaches, see, for example, the

described techniques in Section 3.4.1 that are applied by Vardi [123], Byrd, Schnabel, and Shultz [16], and Omojokun [80].

In the worst case, Algorithm 4.1 requires the solution of three problems in a single iteration. If the step is rejected, then the number of subproblems can be reduced to two as the standard problem is still infeasible. It is possible to avoid the feasibility restoration subproblems by directly relaxing the standard quadratic problem (4.7). In Schittkowski [100] the problem is relaxed by introducing a scaling parameter for the constraints that is added to the objective function of the quadratic problem and then penalized by an additional penalty parameter. This procedure depends on the scaling of the underlying problem. The feasibility restoration phase described before avoids the need of an additional penalty parameter in the subproblems. Another approach for adding the relaxation parameter to the objective of the quadratic problem is proposed by Yuan [130], see also Algorithm 3.3.

Other relaxation strategies, as the ones described in Section 3.4.1 and Section 3.4.2, also require the determination of an adequate parameter θ_k or $\bar{\theta}_k$, respectively. Thus, actually they also may require an additional subproblem to be solved. The feasibility phase proposed for Algorithm4.1 follows the approach of Powell and Yuan [93], see also Section 3.4.2. Note that the technique proposed by Powell and Yuan also requires the determination of $\bar{\theta}_k$ that has to satisfy

$$\min_{\|d\|_2 \leq \tau_1 \Delta_k} \|g(x_k) + \nabla g(x_k)^T d\|_2 \leq \bar{\theta}_k \leq \min_{\|d\|_2 \leq \tau_2 \Delta_k} \|g(x_k) + \nabla g(x_k)^T d\|_2 \ , \qquad (4.255)$$

where $0 < \tau_2 < \tau_1 < 1$ are two constants. Thus, the approach of solving two problems in the feasibility restoration phase of Algorithm 4.1 is frequently applied in trust region methods.

A procedure that is similar to the feasibility restoration phase proposed in Algorithm 4.1 is applied by the filter method of Fletcher, Leyffer, and Toint [43], see also Section 3.4.4. The encouraging results of their implementation of a filter algorithm motivated the choice of the feasibility restoration phase of Algorithm 4.1. The strategy of first trying to solve the standard quadratic subproblem (4.7) and switching to a modified subproblem if necessary is also used in the trust region algorithm by El-Alem [30].

Setting the trust region radius Δ_k to at least $\Delta_{\min}$ after a successful iteration is also applied by Kanzow and Zupke [64], Jiang et al. [63], and Fletcher, Leyffer, and Toint [43]. The lower bound $\Delta_{\min}$ on the trust region radius after a successful iteration plays an important role in the proofs of Theorem 4.16 and Theorem 4.17. By requiring $\Delta_{\bar{k}+1} \geq \Delta_{\min}$, where $\bar{k}$ denotes the last successful iteration, it is guaranteed that the trust region radius approaches the established lower bounds from above and therefore it cannot fall below the bounds. The proof of Theorem 4.22 is simplified by introducing $\Delta_{\min}$, as close to the solution the behavior of Algorithm 4.1 is similar to the line search SQP method proposed by Schittkowski [99, 100]. Consequently, the results obtained by Schittkowski can be adapted for Algorithm 4.1.

In the global convergence proof it is assumed that the multipliers are bounded,

cf. Assumption 4.2(4.). Note that if it would be assumed that the solution of each subproblem satisfies the MFCQ then the desired bounded multipliers are obtained as shown by Gauvin [46]. But the additional trust region constraint might lead to situations where the MFCQ does not hold for the subproblem solution. That is why only bounded multipliers are assumed which is a weaker condition than requiring that the MFCQ holds at every subproblem solution.

In the local convergence analysis it is assumed that

$$\frac{\|u_k - v_k\|_\infty^2}{\|d_k\|_\infty^2} \leq \kappa \tag{4.256}$$

holds for sufficiently large k, with a $\kappa \geq 1$. This condition is also required by other authors, see, e.g., El-Alem [30] and Gill, Murray, Saunders, and Wright [48]. In the proof of Theorem 4.22 outlined before, (4.256) is applied to show the boundedness of the penalty parameter σ_k. Numerical results indicate that (4.256) holds in practice.

Moreover, the local convergence analysis assumes that the matrix B_k is a good approximation to $\nabla_{xx}^2 L(x^\star, u^\star)$ in some sense, that is

$$d_k^T(\nabla_{xx}^2 L(x^\star, u^\star) - B_k)d_k \leq \nu \|d_k\|_2^2 \ , \tag{4.257}$$

where ν satisfies some conditions stated in Theorem 4.22. By applying (4.257), it can be shown that full SQP steps are taken close to the solution and $\|d_k\|_\infty < \Delta_k$ holds for all k sufficiently large. The acceptance of full SQP steps and the inactive trust region constraint is also proved by Ulbrich [121] for a filter method that employs some kind of augmented Lagrangian in the filter.

In El-Alem [30] the quadratic convergence of a trust region algorithm for equality constrained problems is shown, under the additional condition that the matrix B_k is set to $\nabla_{xx}^2 L(x_k, v_k)$ for all iterates (x_k, v_k). The following requirement for the matrix B_k is commonly used for equality constrained problems, that is

$$\lim_{k\to\infty} \max_{\nabla g(x_k)^T d = 0,\ \|d\|_2 \leq 1} |d^T(\nabla_{xx}^2 L(x^\star, u^\star) - B_k)d_k| / \|d_k\|_2 = 0 \ . \tag{4.258}$$

Boggs, Tolle, and Wang [8], and Powell [88] proved that, if d_k solves QP (4.231) with $\|d_k\|_\infty < \Delta_k$ and if $x_{k+1} = x_k + d_k$ for all sufficiently large k, then the rate of convergence of the sequence x_k is superlinear if and only if condition (4.258) holds. Condition (4.258) is applied by Powell and Yuan [93] to establish the superlinear convergence of their algorithm for equality constrained problems.

A reformulation of (4.258) for the case where also inequality constraints are considered is the following

$$\lim_{k\to\infty} \frac{\left\| \overline{P}\left(\nabla_{xx}^2 L(x^\star, u^\star) - B_k\right) d_k \right\|_2}{\|d_k\|_2} = 0 \ , \tag{4.259}$$

where $\overline{P}$ is a projection from $\mathbb{R}^n$ to the null space of $\nabla g_{\mathcal{E}\cup\mathcal{A}(x^\star)}(x^\star)^T$.

In Yuan [130] the condition (4.259) is assumed to show the superlinear convergence of a trust region algorithm that addresses equality and inequality constraints, see Algorithm 3.3. Since the applied penalty function $P(x) = f(x) + \sigma\|g(x)^-\|_\infty$ is not differentiable, the computation of second order correction steps is required. The second order correction subproblem at iterate x_k suggested by Yuan is

$$\begin{array}{ll} \underset{d\in\mathbb{R}^n}{\text{minimize}} & \nabla f(x_k)^T(d_k+d)+\frac{1}{2}(d_k+d)^T B_k(d_k+d)+\sigma_k \left\| \left(g(x_k+d_k)+\nabla g(x_k)^T d\right)^- \right\|_\infty \\ \text{subject to} & \|d_k+d\|_\infty \leq \Delta_k \ , \end{array}$$

where d_k is the solution to

$$\begin{array}{ll} \underset{d\in\mathbb{R}^n}{\text{minimize}} & \nabla f(x_k)^T d + \frac{1}{2} d^T B_k d + \sigma_k \left\| \left(g(x_k) + \nabla g(x_k)^T d\right)^- \right\|_\infty \\ \text{subject to} & \|d\|_\infty \leq \Delta_k \ . \end{array}$$

The aim of the development of Algorithm 4.1 is to avoid the calculation of these second order correction steps, as they require additional function evaluations. This aim is achieved, as shown by the local convergence analysis.

5 Mixed-Integer Optimization

The preceding part of this thesis focuses on algorithms that are applicable to nonlinear optimization problems that only feature continuous variables. This chapter presents methods that address the more complex mixed-integer nonlinear programming problem

$$\begin{aligned} \underset{x\in\mathbb{R}^{n_c},\, y\in\mathbb{Z}^{n_i}}{\text{minimize}} \quad & f(x,y) \\ \text{subject to} \quad & g_j(x,y) = 0\ , \quad j = 1,\dots,m_e\ , \\ & g_j(x,y) \geq 0\ , \quad j = m_e+1,\dots,m\ , \\ & y \in Y\ , \end{aligned} \tag{5.1}$$

where y denotes the additional integer variables. Again, continuous variables are expressed by x. The constant n_c denotes the number of continuous variables and n_i identifies the number of integer variables. It is assumed that the functions $f(x,y)$ and $g_j(x,y)$, $j = 1,\dots,m$, are at least twice continuously differentiable with respect to x for all $x \in \mathbb{R}^{n_c}$. For the general formulation of problem (5.1) it is not assumed that the problem functions are also differentiable with respect to the integer variables y. Whenever differentiability with respect to y is required in the subsequent considerations, it is stated explicitly.

The set Y is defined by finite upper and lower bounds for the integer variables, that is

$$Y := \{y \in \mathbb{Z}^{n_i} \mid\ y_l \leq y \leq y_u\}\ . \tag{5.2}$$

The finiteness of set Y is a necessary condition that is frequently applied to show finite convergence of an optimization method that addresses the mixed-integer nonlinear problem (5.1).

In the subsequent section some well-established methods are reviewed. In addition, a selection of available software is presented. In section 5.2 two new optimization algorithms for mixed-integer problems are introduced. The algorithms are advancements of an algorithm developed by Exler and Schittkowski [37]. The concepts of sequential quadratic programming methods are adapted to mixed-integer nonlinear optimization. The key idea is the substitution of the continuous quadratic programming subproblem by a mixed-integer quadratic problem. The proposed algorithms differ in situation when the integer variables remain fixed to the current configuration. The first algorithm calculates second order correction steps to obtain fast local convergence with respect to the continuous variables, whereas the second algorithm avoids this additional effort. Convergence of the proposed algorithms is not investigated. Exler et al. [36] discuss a possible extension of the algorithms formulated in this thesis. The extension allows the statement of convergence properties. Parts of this chapter can be found in Exler et al. [36].

5.1 Overview of Existing Methods

Down to the present day, numerous methods to solve mixed-integer problems (5.1) have been proposed, see for example Floudas [44] or Grossmann and Kravanja [56] for review papers. Some of the approaches are heuristics, whereas other can be classified as deterministic. Methods as tabu search algorithms apply heuristic strategies to investigate the integer search space. Starting from an integer configuration the neighboring grid points are checked and the best neighbor is chosen to be the next iterate. Cycling is avoided since revisiting of already investigated integer points is prohibited for some iterations, see for example Exler et al. [34]. Pattern search algorithms, see for instance Audet and Dennis [3], are similar to tabu search algorithms. In general search algorithms differ in the way the integer space is explored.

Linear outer approximation is another technique used frequently. The approach was introduced by Duran and Grossmann [28] and was extended by Fletcher and Leyffer [41] later. Solving the mixed-integer nonlinear program is decomposed into solving continuous nonlinear problems and mixed-integer linear problems. The mixed-integer nonlinear problem is approximated by a mixed-integer linear program. For this reason, the problems are required to be convex and problem functions are assumed to be differentiable with respect to the continuous and the integer variables, since valid under- and overestimators are needed. The quality of the linear approximation improves iteratively by solving continuous nonlinear problems with fixed integer configurations. Convergence toward the global optimum of the problem can be shown for differentiable convex problems. A related approach was proposed by Westerlund and Pörn [125], and is called ECP. The ECP technique is able to solve pseudo-convex problems by generating cutting planes and solving linear programming problems.

The so far mentioned methods generate a sequence of iterates, where the integer constraint with respect to the discrete variables holds for all generated points. Other techniques require the continuous relaxation of integer variables. Continuous relaxations assume that integer variables can be treated as continuous ones, i.e., function values can be evaluated for all $y \in Y_{\mathbb{R}}$, where

$$Y_{\mathbb{R}} := \{y \in \mathbb{R}^{n_i} \mid \ y_l \leq y \leq y_u\} \ . \tag{5.3}$$

The transformed problem is a nonlinear program of form (1.2) which can be solved by any available nonlinear optimization code. The following methods require that $f(x, y)$ and $g_j(x, y)$, $j = 1, \ldots, m$, are also continuously differentiable with respect to y for all $y \in Y_{\mathbb{R}}$.

Branch-and-bound methods make use of this relaxation and set up a binary search tree that represents the integer search space. The leaves of the obtained tree comply with all possible integer combinations. The nodes of the search tree correspond to relaxed nonlinear problems. These problems are obtained by restricting the variable range of the relaxed integer variables, see, for example, Gupta and Ravindran [57], or Borchers and Mitchell [11]. Under certain conditions parts of the tree can be elimi-

nated and the number of subproblems decreases. The basic idea of branch-and-bound methods requires that the nonlinear problems are solved to optimality. By supposing that problem functions are continuously differentiable with respect to x and y, efficient methods can be applied to solve the nonlinear subproblems.

The early branching strategy proposed by Leyffer [68] tries to reduce the effort needed to solve each nonlinear subproblem to optimality. Here a SQP method is applied and the branch-and-bound search tree is built up within the solution process of the SQP algorithm. The intention is to branch as soon as possible, i.e., after solving a single quadratic subproblem. Although this strategy results in larger search trees, in most cases the overall performance improves compared to methods applying standard branch-and-bound techniques.

Other approaches transform the mixed-integer problem into a single continuous problem by reformulating the integer conditions as continuous nonlinear constraints. The resulting constraints make the problem highly non-convex, thus, the transformed problem has to be solved by global optimization techniques. This strategy is proposed by several authors, see, e.g., Li and Chou [69].

Several software packages that address mixed-integer problems are available. A review can be found in Bussieck and Vigerske [14]. Bonami et al. [10] present some results of a comparative study for a collection of solvers. Some selected solvers are presented here.

- `BARON` solves non-convex mixed-integer optimization problems to global optimality, see Sahinidis and Tawarmalani [97]. The theory was established by Tawarmalani and Sahinidis [114–116]. `BARON` combines constraint propagation, interval analysis, and duality with enhanced branch-and-bound concepts.
- `LaGO` is developed by Nowak et al. [79]. `LaGO` solves non-convex mixed-integer problems by generating inner and outer approximations. These approximations are obtained by convex and polyhedral relaxations, see Nowak [78].
- `DICOPT`, see Viswanathan and Grossmann [124], is an implementation of a linear outer approximation method. By applying relaxation strategies also nonlinear equality constraints can be handled. Some heuristics are implemented to improve the performance on non-convex problems.
- `BONMIN` is a hybrid code by Bonami et al. [9]. It combines branch-and-bound, outer approximation, and branch-and-cut techniques.
- `COUENNE` can be used to solve non-convex mixed-integer nonlinear programs, see Belotti et al. [5, 6]. `COUENNE` is based on convex over- and under-envelopes and a branch-and-bound algorithm.

The software called `MISQP`, see Exler et al. [35], implements the algorithms introduced in the following sections. A comparative study by Exler et al. [36] shows that `MISQP` can successfully be applied to non-convex mixed-integer nonlinear programming problems.

5.2 New Algorithms for Mixed-Integer Nonlinear Optimization

In the subsequent parts of this section two algorithms are presented that adapt the concepts of sequential quadratic programming methods to mixed-integer nonlinear programming. The first algorithm applies second order correction steps with respect to continuous variables. The intention of this additional steps is to retain the fast local convergence of the underlying SQP method in case the integer variables remain fixed to a particular configuration. The second algorithm applies concepts of Algorithm 4.1. The purpose of the modification is the avoidance of second order correction steps close to a solution of the nonlinear problem when integer variables are fixed. This is achieved by switching to the augmented Lagrangian merit function in this situation. At the beginning, some general statements follow.

The algorithms are developed under the paradigm that integer variables are non-relaxable, the problem is non-convex and the analytical structure of the problem is not known. Moreover, it is assumed that function values are determined by time-consuming simulation codes, thus, internal calculation time is dominated by the time spent in the simulation runs. Hence, the time needed for solving the optimization problem depends directly on the total number of function evaluation required by the algorithm. The efficiency of the developed algorithms can be measured by counting the total number of required function evaluations.

These assumptions are motivated by real-life mixed-integer problems arising in mechanical, electrical, aerospace, chemical, automotive, petroleum and related engineering. An example from systems biology is considered in Sendín et al. [108]. Frequently the integer variables involved are not relaxable. The functions are often highly nonlinear and non-convex. In some cases function values are obtained by running complex simulation software. An industrial case study is considered in Bünner et al. [12], where the number of fingers and layers of an electronic filter are modeled as integer variables, which cannot be relaxed due to the underlying simulation tools. Other examples for non-relaxable integer variables are the number of rips and rills of a horn antenna for satellite communication, as described by Hartwanger et al. [60], or the number of trays in the design of a distillation column, see Thomas and Kröner [117]. Exler et al. [34] consider a wastewater treatment plant, where fractional values of the position of the feed layer in a settler are not accepted by the underlying Simulink implementation of the model. In Antelo et al. [1] the decisions whether PI controls are applied in a specific design or not is modeled by non-relaxable binary variables. Changes in the state of a binary variable lead to executions of completely different simulation codes.

When considering non-relaxable integer variables, the focus lies on applications, especially in engineering sciences, which are modeled by non-relaxable integer variables with some *physical* meaning. All integer variables of the aforementioned examples have physical meanings, except the binary variables in the last example. The function values implicitly depend on each other, i.e., a change in the integer configuration by one, leads only to a slight variation in the objective and constraint function values. This observation is the main motivation for applying quadratic approximations within the

subsequent algorithms.

As the proposed algorithms are based on sequential quadratic programming techniques, first partial derivatives are required. If analytical derivatives are not available, the partial derivatives have to be approximated numerically. The question how to determine first partial derivatives with respect to non-relaxable integer variables arises. The typical approximation strategy used for continuous variables is not applicable, as $f(x,y)$ and $g_1(x,y),\ldots,g_m(x,y)$, cannot be evaluated at small perturbations of an integer variable value. To overcome this problem it is suggested to approximate the first partial derivatives at neighboring grid points. For the objective function and given x_k and y_k, this might be done, e.g., by the following two-sided difference formula

$$\begin{aligned}\frac{\partial f(x_k,y_k)}{\partial y_i} \approx \frac{1}{2}\Bigg(& f\left(x_k,\left(y_1^{(k)},\ldots,y_i^{(k)}+1,\ldots,y_{n_i}^{(k)}\right)^T\right) \\ & - f\left(x_k,\left(y_1^{(k)},\ldots,y_i^{(k)}-1,\ldots,y_{n_i}^{(k)}\right)^T\right)\Bigg)\,,\end{aligned} \tag{5.4}$$

where $i=1,\ldots,n_i$, and $y_k=(y_1^{(k)},\ldots,y_{n_i}^{(k)})^T$. The function evaluations at neighboring grid points give information about the structure of the underlying problem. The gained information is used by the proposed algorithms. Therefore, the new algorithms also have some characteristics known from search methods.

The assumption that integer variables cannot be relaxed exclude some of the methods presented in section 5.1 from being applied. This concerns all methods that require continuous relaxations, as for example branch-and-bound algorithms. Linear outer approximation algorithms can be used instead. On the other hand, they are less reliable in case analytical derivatives with respect to integer variables are not available and the problem functions are non-convex.

5.2.1 A Mixed-Integer Sequential Quadratic Programming Algorithm

The mixed-integer sequential quadratic programming method is a further development of the first version discussed and implemented by Exler and Schittkowski [37]. The algorithm adapts the SQP-based trust region method of Yuan [130], cf. Algorithm 3.3, to solve mixed-integer nonlinear optimization problems. The idea of the new mixed-integer algorithm is the substitution of the continuous quadratic subproblem by a mixed-integer quadratic subproblem.

Since the length of trial steps has to be controlled to obtain progress toward the solution, a trust region stabilization is suggested. The use of trust region techniques is motivated by the fact that the generated trial points always fulfill the integer requirement. Applying standard line search techniques contradicts the paradigm of non-relaxable integer variables, as performing a search along the determined direction might lead to fractional values for integer variables. Instead of a line search, a lattice search might be applied, but the investigation of the concept of a lattice search is future work and not considered here.

The changes applied to the underlying Algorithm 3.3 affect several parts. The most

important modification applies to the definition of the subproblems as mentioned before. The continuous subproblems known from SQP methods are substituted by mixed-integer subproblems. The trial steps generated by these subproblems promise progress in the continuous and integer space simultaneously. Thus, the proposed method differs from other techniques that decompose the process of solving a mixed-integer problem. These methods obtain progress toward the solution of the mixed-integer problem in the continuous variables and the integer variables separately.

Moreover, the step size is adjusted subject to a modified strategy compared to Algorithm 3.3. Instead of using a single trust region radius parameter, the step size is controlled separately with respect to continuous and integer variables. For this purpose the proposed algorithm uses two trust region radii, one that restricts steps in the continuous space and a second one related to the integer space, respectively.

The L_∞-penalty function of the mixed-integer problem is similar to the one already introduced for continuous problems, cf. (3.12). Now the L_∞-penalty function is defined as

$$P_\sigma(x,y) := f(x,y) + \sigma \|g(x,y)^-\|_\infty \ , \tag{5.5}$$

and $\sigma > 0$ is an associated penalty parameter. The measurement of constraint violation $g(x,y)^-$ is the straightforward adaptation of the continuous version as defined in (2.10). The formulation of the algorithm guarantees that all trial points and iterates always stay within the bounds given by Y, see (5.2). Therefore, the corresponding bounds on y are not included in penalty function $P_\sigma(x,y)$.

Now the main difference of the new mixed-integer algorithm compared to the continuous Algorithm 3.3 is formulated. To approximate $P_{\sigma_k}(x_k,y_k)$ in the k-th iteration step, where (x_k,y_k) is a current iterate, the mixed-integer subproblem

$$\begin{aligned} \underset{d^c \in \mathbb{R}^{n_c},\ d^i \in \mathbb{Z}^{n_i}}{\text{minimize}} \quad & \nabla f(x_k,y_k)^T d + \frac{1}{2} d^T B_k d + \sigma_k \left\| \left(g(x_k,y_k) + \ g(x_k,y_k)^T d \right)^- \right\|_\infty \\ \text{subject to} \quad & \|d^c\|_\infty \le \Delta_k^c \ , \ \ \|d^i\|_\infty \le \Delta_k^i \ , \\ & y_k + d^i \in Y \ , \end{aligned} \tag{5.6}$$

where

$$d := \left(d_1^c, \ldots, d_{n_c}^c, d_1^i, \ldots, d_{n_i}^i \right)^T \ , \tag{5.7}$$

is solved. It is assumed that the matrix $B_k \in \mathbb{R}^{(n_c+n_i)\times(n_c+n_i)}$ is positive definite. The solution (d_k^c, d_k^i) of subproblem (5.6) always leads to trial points that satisfy the bounds given by (5.2) due to restriction $y_k + d_k^i \in Y$. $\Delta_k^c > 0$ and $\Delta_k^i \ge 0$ denote the trust region radii for the continuous and integer search space, respectively. By controlling the step sizes separately the fast local convergence with respect to the continuous variables retains. Moreover, the separate trust region radius for the integer variables offers the opportunity to fix the integer variables for some iterations by setting the radius Δ_k^i equal to zero.

In the remainder of this section the objective function of the mixed-integer subproblem (5.6) is denoted by

$$\psi_{\sigma_k}(d) := \nabla f(x_k, y_k)^T d + \frac{1}{2} d^T B_k d + \sigma_k \left\| \left(g(x_k, y_k) + \ g(x_k, y_k)^T d \right)^- \right\|_\infty \ , \tag{5.8}$$

where d is defined according to (5.7). The formulation ψ_{σ_k} is chosen to highlight the dependency of the penalty function on the current penalty parameter value σ_k.

The objective function of the mixed-integer subproblem (5.6) is non-smooth. The non-smooth part of objective function can be eliminated by introducing a nonnegative slack variable $s \in \mathbb{R}$ and rewriting the constraint violation measurement as a set of linear inequality constraints. The reformulated problem (5.6) is defined as

$$\begin{aligned}
\underset{d^c \in \mathbb{R}^{n_c},\, d^i \in \mathbb{Z}^{n_i},\, s \in \mathbb{R}}{\text{minimize}} \quad & \nabla f(x_k, y_k)^T d + \frac{1}{2} d^T B_k d + \sigma_k s \\
\text{subject to} \quad & s - g_j(x_k, y_k) - \nabla g_j(x_k, y_k)^T d \geq 0 \ , \quad j = 1, \ldots, m_e \ , \\
& s + g_j(x_k, y_k) + \nabla g_j(x_k, y_k)^T d \geq 0 \ , \quad j = 1, \ldots, m \ , \\
& \|d^c\|_\infty \leq \Delta_k^c \ , \ \|d^i\|_\infty \leq \Delta_k^i \ , \\
& y_k + d^i \in Y \ , \ \ s \geq 0 \ ,
\end{aligned} \tag{5.9}$$

with d according to (5.7). Each of the m_e equality constraints is rewritten as two inequality constraints. Note that $\|d^c\|_\infty \leq \Delta_k^c$ and $\|d^i\|_\infty \leq \Delta_k^i$ can be rewritten as box constraints, i.e., a set of $2(n_c + n_i)$ linear inequality constraints. As a result, the box constrained formulation of problem (5.9) is a convex mixed-integer quadratic problem that can be solved by any available mixed-integer quadratic programming solver without further modifications.

The second order correction step, as in Algorithm 3.3, is also applied, but only with respect to the continuous variables. Numerical tests indicate that a significant improvement of convergence speed can be obtained due to these additional steps. Integer variables are fixed to d_k^i obtained by (5.9), and the problem is formulated as

$$\begin{aligned}
\underset{d^c \in \mathbb{R}^{n_c}}{\text{minimize}} \quad & \nabla f(x_k, y_k)^T \left(\begin{pmatrix} d_k^c \\ d_k^i \end{pmatrix} + \begin{pmatrix} d^c \\ 0 \end{pmatrix} \right) + \frac{1}{2} \left(\begin{pmatrix} d_k^c \\ d_k^i \end{pmatrix} + \begin{pmatrix} d^c \\ 0 \end{pmatrix} \right)^T B_k \left(\begin{pmatrix} d_k^c \\ d_k^i \end{pmatrix} + \begin{pmatrix} d^c \\ 0 \end{pmatrix} \right) \\
& + \sigma_k \left\| \left(g\left(x_k + d_k^c, y_k + d_k^i \right) + \nabla g(x_k, y_k)^T \begin{pmatrix} d^c \\ 0 \end{pmatrix} \right)^- \right\|_\infty \\
\text{subject to} \quad & \|d_k^c + d^c\|_\infty \leq \Delta_k^c \ ,
\end{aligned} \tag{5.10}$$

where (d_k^c, d_k^i) is the solution of (5.9). The non-smooth problem (5.10) can also be rewritten as a smooth quadratic programming problem in standard form similar to (5.9). The optimal solution is denoted by $\hat{d}_k^c$.

As mentioned before a special strategy for approximating partial derivatives with respect to integer variables is applied. The basic idea of calculating two-sided ap-

proximations at neighboring grid points is stated in formula (5.4). Since the algorithm guarantees satisfaction of box constraints, the formula is adapted at the bounds of Y. For variables at a bound, formula (5.4) is replaced by a forward or backward difference formula, respectively.

There is a very attractive side-effect of approximating integer derivatives at neighboring grid points. The best feasible neighbor visited in the approximation procedure is stored, and the algorithm returns to this point whenever it seems to be profitable. This strategy can be interpreted as a direct search in the neighborhood of the current iterate (x_k, y_k) and is known from other search algorithms, see, e.g., a tabu search algorithm by Exler et al. [34]. The calculation of partial derivatives with respect to integer variables is stated in Procedure 5.1. Besides approximating the needed gradients, Procedure 5.1 also returns the best feasible neighbor of (x_k, y_k), if one exists, which is denoted by (x_k^{bn}, y_k^{bn}) and the corresponding objective function value is denoted by f_k^{bn}, respectively. The iteration index k is omitted to improve readability. To simplify the notation $\nabla_y f(x, y)$ and $\nabla_y g(x, y)$ also denote the approximations to the partial derivatives and not only the exact gradients.

Procedure 5.1 Given $x \in \mathbb{R}^{n_c}$, $y \in Y$, $f(x, y)$ and $g(x, y)$. Let $\epsilon > 0$ be a small tolerance and $f^{bn} := \infty$, $(x^{bn}, y^{bn}) := (x, y)$.

Output: Approximations to $\nabla_y f(x, y)$, $\nabla_y g(x, y)$, f^{bn}, and (x^{bn}, y^{bn}).

begin

for $i = 1$ **to** n_i **do**

$z^{+1} := \left(x, (y_1, \ldots, y_i + 1, \ldots, y_{n_i})^T\right)$ and $z^{-1} := \left(x, (y_1, \ldots, y_i - 1, \ldots, y_{n_i})^T\right)$.

if $y_i^l < y_i < y_i^u$ **then**

Evaluate $f(z^{+1})$, $g(z^{+1})$ and $f(z^{-1})$, $g(z^{-1})$.

if $\|g(z^{+1})^-\|_\infty \leq \epsilon$ **and** $f(z^{+1}) < f^{bn}$ **then** $f^{bn} := f(z^{+1})$ and $(x^{bn}, y^{bn}) := z^{+1}$.

if $\|g(z^{-1})^-\|_\infty \leq \epsilon$ **and** $f(z^{-1}) < f^{bn}$ **then** $f^{bn} := f(z^{-1})$ and $(x^{bn}, y^{bn}) := z^{-1}$.

Set $\dfrac{\partial f(x, y)}{\partial y_i} := \dfrac{1}{2}\left(f(z^{+1}) - f(z^{-1})\right)$.

for $j = 1$ **to** m **do** Set $\dfrac{\partial g_j(x, y)}{\partial y_i} := \dfrac{1}{2}\left(g_j(z^{+1}) - g_j(z^{-1})\right)$.

else if $y_i = y_i^l$ **then**

Evaluate $f(z^{+1})$ and $g(z^{+1})$.

if $\|g(z^{+1})^-\|_\infty \leq \epsilon$ **and** $f(z^{+1}) < f^{bn}$ **then** $f^{bn} := f(z^{+1})$ and $(x^{bn}, y^{bn}) := z^{+1}$.

Set $\dfrac{\partial f(x, y)}{\partial y_i} := f(z^{+1}) - f(x, y)$.

for $j = 1$ **to** m **do** Set $\dfrac{\partial g_j(x, y)}{\partial y_i} := g_j(z^{+1}) - g_j(x, y)$.

else if $y_i = y_i^u$ **then**

Evaluate $f(z^{-1})$ and $g(z^{-1})$.

if $\|g(z^{-1})^-\|_\infty \leq \epsilon$ **and** $f(z^{-1}) < f^{bn}$ **then** $f^{bn} := f(z^{-1})$ and $(x^{bn}, y^{bn}) := z^{-1}$.

Set $\dfrac{\partial f(x,y)}{\partial y_i} := f(x,y) - f(z^{-1})$.

for $j = 1$ **to** m **do** Set $\dfrac{\partial g_j(x,y)}{\partial y_i} := g_j(x,y) - g_j(z^{-1})$.

end if

end do

end

Note that Procedure 5.1 returns $f^{bn} = \infty$ if no feasible neighbor exists, that simplifies the notation of the next algorithm. As Procedure 5.1 is invoked, the mixed-integer sequential quadratic programming algorithm with trust region stabilization can be seen as a hybrid algorithm that combines a modified SQP method with elements known from search methods. The algorithm is formulated in the following.

Algorithm 5.2 Let $\epsilon_{\text{tol}} > 0$ and $\bar{\sigma} > 0$ be given constants.

STEP 0 Choose initial values for $x_0 \in \mathbb{R}^{n_c}$, $y_0 \in Y$, $\Delta_0^c > 0$, $\Delta_0^i \geq 1$, $\sigma_0 > 0$, $\zeta_0 > 0$, and a positive definite matrix $B_0 \in \mathbb{R}^{(n_c+n_i)\times(n_c+n_i)}$.

Set $f^\star := \infty$ and $(x^\star, y^\star) := (x_0, y_0)$ to the current best known solution.

Evaluate function values $f(x_0, y_0)$ and $g(x_0, y_0)$.

Evaluate $\nabla_x f(x_0, y_0)$ and $\nabla_x g(x_0, y_0)$ with respect to continuous variables.

Use Procedure 5.1 to approximate $\nabla_y f(x_0, y_0)$ and $\nabla_y g(x_0, y_0)$ with respect to integer variables and obtain (x_0^{bn}, y_0^{bn}) and f_0^{bn}.

if $f_0^{bn} < f^\star$ **then** Set $f^\star := f_0^{bn}$ and $(x^\star, y^\star) := (x_0^{bn}, y_0^{bn})$.

Set $k := 0$.

STEP 1 Solve the mixed-integer quadratic problem (5.9) giving $d_k := \begin{pmatrix} d_k^c \\ d_k^i \end{pmatrix}$.

if $(\|g(x_k, y_k)^-\|_\infty \leq \epsilon_{\text{tol}}$ **or** $\sigma_k > \bar{\sigma})$ **and** $\psi_{\sigma_k}(0) - \psi_{\sigma_k}(d_k) \leq \epsilon_{\text{tol}}$ **then**
goto STEP 8.

STEP 2 **if** $\|g(x_k, y_k)^-\|_\infty - \|(g(x_k, y_k) + \nabla g(x_k, y_k)^T d_k)^-\|_\infty < \epsilon_{\text{tol}}$ **and**
$\|(g(x_k, y_k) + \nabla g(x_k, y_k)^T d_k)^-\|_\infty > \epsilon_{\text{tol}}$ **then**
Set $\sigma_{k+1} := 10\sigma_k$ and $\zeta_{k+1} := \zeta_k/10$.
else Set $\sigma_{k+1} := \sigma_k$ and $\zeta_{k+1} := \zeta_k$.
if $\psi_{\sigma_k}(0) - \psi_{\sigma_k}(d_k) < \zeta_k\, \sigma_k \min(\Delta_k^c, \|g(x_k, y_k)^-\|_\infty)$ **then**
Replace $\sigma_{k+1} := 2\sigma_{k+1}$ and $\zeta_{k+1} := \zeta_{k+1}/4$.

STEP 3 Evaluate $f(x_k + d_k^c, y_k + d_k^i)$ and $g(x_k + d_k^c, y_k + d_k^i)$, and calculate

$$r_k := \frac{P_{\sigma_{k+1}}(x_k, y_k) - P_{\sigma_{k+1}}(x_k + d_k^c, y_k + d_k^i)}{\psi_{\sigma_k}(0) - \psi_{\sigma_k}(d_k)} \ . \tag{5.11}$$

STEP 4 **if** $r_k \leq 0.75$ **then** Solve SOC problem (5.10) to obtain $\hat{d}_k^c$ and evaluate $f(x_k + d_k^c + \hat{d}_k^c, y_k + d_k^i)$ and $g(x_k + d_k^c + \hat{d}_k^c, y_k + d_k^i)$.

if $P_{\sigma_{k+1}}(x_k + d_k^c + \hat{d}_k^c, y_k + d_k^i) \ < \ P_{\sigma_{k+1}}(x_k + d_k^c, y_k + d_k^i)$ **then** Update r_k by

$$r_k := \frac{P_{\sigma_{k+1}}(x_k, y_k) - P_{\sigma_{k+1}}(x_k + d_k^c + \hat{d}_k^c, y_k + d_k^i)}{\psi_{\sigma_k}(0) - \psi_{\sigma_k}(d_k)} \ , \tag{5.12}$$

and replace $d_k := \begin{pmatrix} d_k^c + \hat{d}_k^c \\ d_k^i \end{pmatrix}$ and $d_k^c := d_k^c + \hat{d}_k^c$.

STEP 5 Update the trust region radii by

$$\Delta_{k+1}^c := \begin{cases} \min(\|d_k\|_\infty/2 \ , \ \Delta_k^c) & , \text{if} \quad 0.25 \ > \ r_k \ , \\ \Delta_k^c & , \text{if} \quad 0.25 \ \leq \ r_k \ \leq \ 0.75 \ , \\ \max(2\|d_k\|_\infty \ , \ \Delta_k^c) & , \text{if} \quad 0.75 \ < \ r_k \ , \end{cases} \tag{5.13}$$

and

$$\Delta_{k+1}^i := \begin{cases} \lfloor \|d_k^i\|_\infty/2 \rfloor & , \text{if} \quad 0.25 \ > \ r_k \ , \\ \Delta_k^i & , \text{if} \quad 0.25 \ \leq \ r_k \ \leq \ 0.75 \ , \\ \max(2\|d_k^i\|_\infty \ , \ \Delta_k^i, 1) & , \text{if} \quad 0.75 \ < \ r_k \ . \end{cases} \tag{5.14}$$

STEP 6 **if** $r_k \leq 0$ **then** Set $(x_{k+1}, y_{k+1}) := (x_k, y_k)$, $B_{k+1} := B_k$, $k := k + 1$ and **goto** STEP 1.

else Set $(x_{k+1}, y_{k+1}) := (x_k + d_k^c, y_k + d_k^i)$.

STEP 7 Evaluate partial derivatives $\nabla_x f(x_{k+1}, y_{k+1})$ and $\nabla_x g(x_{k+1}, y_{k+1})$ with respect to continuous variables.

Approximate $\nabla_y f(x_{k+1}, y_{k+1})$ and $\nabla_y g(x_{k+1}, y_{k+1})$ with respect to integer variables using Procedure 5.1 and obtain $(x_{k+1}^{bn}, y_{k+1}^{bn})$ and f_{k+1}^{bn}.

if $f_{k+1}^{bn} < f^\star$ **then** Set $f^\star := f_{k+1}^{bn}$ and $(x^\star, y^\star) := (x_{k+1}^{bn}, y_{k+1}^{bn})$.

Generate a positive definite matrix B_{k+1}. Set $k := k + 1$ and **goto** STEP 1.

STEP 8 **if** $\|g(x_k, y_k)^-\|_\infty \leq \epsilon_{\text{tol}}$ **and** $f^\star \geq f(x_k, y_k)$ **then** Set $f^\star := f(x_k, y_k)$, $(x^\star, y^\star) := (x_k, y_k)$ and **STOP** .

if $\|g(x_k, y_k)^-\|_\infty > \epsilon_{\text{tol}}$ **and** $f^\star = \infty$ **then** Report that the problem might be infeasible and **STOP** .

otherwise Set $(x_{k+1}, y_{k+1}) := (x^\star, y^\star)$. Evaluate function values $f(x_{k+1}, y_{k+1})$ and $g(x_{k+1}, y_{k+1})$, and **goto** Step 7.

In mixed-integer nonlinear programming, local optimality conditions comparable to the KKT conditions in continuous optimization are not known. The algorithm stops as soon as a sufficient reduction of the merit function (5.5) is no longer possible.

Note that in Step 2 the penalty parameter σ_k might grow arbitrarily large, in particular if the underlying mixed-integer program is infeasible. If σ_k is greater than a threshold $\bar{\sigma}$ and the predicted reduction of the merit function is small, the algorithm is supposed to terminate at an infeasible stationary point, see Yuan [130]. The constant values in the update rules in Step 2 are set according to Algorithm 3.3. Numerical tests indicate that these values seem to be the most effective. The parameter $\bar{\sigma}$ should be set to a sufficiently large value, e.g., 10^{20}. The parameter ζ_k is an automatically adapted scaling factor for the constraint violation measurement.

In Step 5 the trust region update for the continuous trust-region radius Δ_k^c uses the norm of the complete step d_k including the integer part, see (5.13), to guarantee that $\Delta_k^c > 0$. Expression $\lfloor \|d_k^i\|_\infty / 2 \rfloor$ in (5.14) denotes the largest integer value less than $\|d_k^i\|_\infty / 2$. Thus, the trust region radius Δ_k^i is integer for all k.

In Step 8 a restart is performed whenever the approximation of partial derivatives with respect to integer variables, i.e., the execution of Procedure 5.1, found a better point than the current iterate. This point is set to be the initial point for an additional execution of the main loop.

Algorithm 5.2 is stated in a basic form to illustrate the main ideas of the mixed-integer SQP method. The remainder of this section addresses aspects of an implementation. Modifications are discussed that can improve the performance and robustness of a specific implementation.

If exact gradients for integer variables are available, Procedure 5.1 in Step 0 and Step 7 can be omitted. The additional function evaluations for the internal approximations are avoided, but then Algorithm 5.2 also loses the characteristic of a search algorithm. Restarts are not performed in Step 8 and $f^\star$ remains unchanged. The corresponding changes of Algorithm 5.2 are straightforward.

A second modification affects the test whether a generated trial step is accepted or not. A non-monotone decrease of the penalty function values $P_{\sigma_k}(x_k, y_k)$ is suggested. The idea of accepting new iterates which increase the penalty function, is investigated in the context of trust region algorithms by several authors, see, e.g., Toint [120], Chen et al. [19], and Deng et al. [22]. In the continuous case, convergence can be proved. In the mixed-integer case, however, it might happen that after increasing the penalty function and changing the integer variables, the algorithm might not be able to decrease the penalty function value below the value at the non-monotone step.

The non-monotone strategy can be described as follows. An integer constant $M > 0$ is chosen and an actual penalty function value is always compared with the highest

one obtained during the previous M successful iterations. An iteration k is called a successful iteration if d_k is used to update an iterate, i.e., if $(x_{k+1}, y_{k+1}) = (x_k, y_k) + d_k$. The set of iterates that corresponds to the last M successful iterations be denoted by $\overline{K}_k$. Note that whenever $(x_{k+1}, y_{k+1}) = (x_k, y_k) + d_k$ the iterate (x_{k+1}, y_{k+1}) substitutes the element with the lowest iteration index in set $\overline{K}_{k+1}$. The alternative formulation of Step 3 is

Step 3 Evaluate $f(x_k + d_k^c, y_k + d_k^i)$ and $g(x_k + d_k^c, y_k + d_k^i)$, and calculate

$$r_k := \frac{P_{\sigma_{k+1}}(x_{l_k}, y_{l_k}) - P_{\sigma_{k+1}}(x_k + d_k^c, y_k + d_k^i)}{\psi_{\sigma_k}(0) - \psi_{\sigma_k}(d_k)} \quad , \tag{5.15}$$

where $P_{\sigma_{k+1}}(x_{l_k}, y_{l_k}) := \max\limits_{(x,y) \in \overline{K}_k} P_{\sigma_{k+1}}(x, y)$.

It is recommended to apply the second order correction steps in Step 5, even in case a non-monotone reduction condition on the penalty function is introduced. Intensive numerical tests indicate that this strategy improves efficiency in some situations.

A procedure for updating the matrix B_k might be a quasi-Newton update formula, e.g., the BFGS formula. However, a modification of B_k is recommended if a jump from Step 1 to Step 8 occurs. All entries in B_k are scaled by the same value such that

$$\|B_k\|_\infty \leq \frac{1}{n_c + n_i} \|\nabla f(x_k, y_k)\|_\infty \tag{5.16}$$

holds. The scaling strategy is also motivated by the fact that large values in B_k result in void integer steps. Numerical tests show that this heuristic scaling strategy (5.16) improves the robustness of the algorithm significantly.

Finally, some comments on the choice of the norm for determining the step length follow. The L_∞-norm is applied with respect to continuous variables and integer variables. A modification is suggested that depends on the domain of the integer variable. It is recommended to handle binary variables, i.e., variables with domain $\{0, 1\}$, differently. The length of a step in the binary variables should be measured with respect to the L_1-norm. The purpose of applying the L_1-norm is to obtain more freedom in restricting the search step in the binary space.

5.2.2 A Modification to Avoid Second Order Correction Steps

The following algorithm is a modification of Algorithm 5.2. The aim of the presented adjustment is to avoid the second order correction steps calculated in Step 4 of Algorithm 5.2. Calculating second order correction steps implies more function evaluations and an increase of internal calculation times. Under the assumption of time consuming function evaluations the additional function evaluations are not desirable. The modifications of Algorithm 5.2 affect the merit function. Under some circumstances the achieved progress of a trial step d_k at iteration k is not evaluated subject to the L_∞-penalty function (5.5). Instead the augmented Lagrangian merit function $\Phi_{\sigma_k^\Phi}$, see

(5.17) below, is applied. Now also the multiplier approximations v_k are involved. The mixed-integer augmented Lagrangian at the k-th iterate (x_k, y_k, v_k) is defined as

$$\Phi_{\sigma_k^\Phi}(x_k, y_k, v_k) := f(x_k, y_k) - \sum_{j \in \mathcal{S}_k} \left(v_j^{(k)} g_j(x_k, y_k) - \frac{1}{2} \sigma_k^\Phi g_j(x_k, y_k)^2 \right) - \frac{1}{2} \sum_{j \in \overline{\mathcal{S}}_k} \frac{v_j^{(k)^2}}{\sigma_k^\Phi} \ , \tag{5.17}$$

with

$$\mathcal{S}_k := \mathcal{E} \cup \left\{ j \in \mathcal{I} \mid \ g_j(x_k, y_k) \leq v_j^{(k)} / \sigma_k^\Phi \right\} \tag{5.18}$$

and

$$\overline{\mathcal{S}}_k := \{1, \ldots, m\} \setminus \mathcal{S}_k \ . \tag{5.19}$$

An additional penalty parameter σ_k^Φ is introduced and differs from the penalty parameter σ_k used by the L_∞-penalty function (5.5).

As convergence properties of the method are hard to derive in case mixed-integer steps are taken, only situations are taken into account where the calculated trial steps leave the integer variables unchanged, i.e., $\|d_k^i\|_\infty = 0$ holds for the solution of subproblem (5.9). This corresponds to a continuous step where convergence properties of Algorithm 4.1 can be applied. A switch to the augmented Lagrangian merit function (5.17) is performed if

$$\|d_k^i\|_\infty = 0 \tag{5.20}$$

and

$$\left\| \left(g(x_k, y_k) + \nabla g(x_k, y_k)^T d_k \right)^- \right\|_\infty = 0 \ , \tag{5.21}$$

where $d_k := (d_k^c, d_k^i)$ denotes the solution of problem (5.9).

Since the augmented Lagrangian (5.17) involves multipliers v_k, so some comments on the choice of these multipliers follow. Let (d_k^c, d_k^i) be the optimal solution of problem (5.9), in addition, (5.20) and (5.21) hold. Then d_k^c solves the following quadratic problem

$$\begin{aligned} \underset{d \in \mathbb{R}^{n_c}}{\text{minimize}} \quad & \nabla_x f(x_k, y_k)^T d + \frac{1}{2} d^T B_k^c d \\ \text{subject to} \quad & g_j(x_k, y_k) + \nabla_x g_j(x_k, y_k)^T d = 0 \ , \quad j = 1, \ldots, m_e \ , \\ & g_j(x_k, y_k) - \nabla_x g_j(x_k, y_k)^T d \geq 0 \ , \quad j = m_e + 1, \ldots, m \ , \\ & \|d\|_\infty \leq \Delta_k^c \ . \end{aligned} \tag{5.22}$$

Here $\nabla_x f(x_k, y_k)$ denotes the gradient of the objective function with respect to the continuous variables. For the m constraints $\nabla_x g_j(x_k, y_k)$, $j = 1, \ldots, m$, also denotes the corresponding part of the gradient. The matrix B_k^c is the upper left $n_c \times n_c$ matrix of B_k, i.e., the Hessian approximation with respect to the continuous variables.

Let the trust region constraint $\|d\|_\infty \leq \Delta_k^c$ in problem (5.22) be replaced by box constraints, i.e., linear inequalities are added instead. Then a triple (d_k^c, u_k, μ_k) exists such that the KKT optimality conditions of the reformulated problem (5.22) hold, where the multipliers μ_k correspond to the additional box constraints. The KKT conditions of problem (5.22) can easily be derived from the ones stated in (2.23)-(2.27),

therefore, they are not declared here. The multipliers u_k, which correspond to the original m linear constraints, are then applied to evaluate the augmented Lagrangian merit function (5.17). Moreover, these multipliers u_k are used to update the multiplier approximation in the enhanced version of Algorithm 5.2 formulated below.

To be able to evaluate the quality of a trial step subject to the augmented Lagrangian (5.17), some information about the model and the penalty update has to be stated. Let (x_k, y_k, v_k) be the current iterate and (d_k, u_k, μ_k) be determined by subproblem (5.22), where μ_k denotes the multipliers with respect to the reformulated trust region constraint $\|d\|_\infty \leq \Delta_k^c$, as aforementioned. The change in the multipliers be denoted by

$$w_k := u_k - v_k \ . \tag{5.23}$$

Although the notation could be simplified, as applying w_k always leads to $v_k + w_k = u_k$ in the situations considered, w_k is introduced to be as close as possible to the notation used in Algorithm 4.1.

Since $\|d_k^i\|_\infty = 0$ holds in case the augmented Lagrangian is evaluated, the integer step d_k^i can be neglected and a trial step (d_k^c, w_k) is considered. The model $\Psi_{\sigma_k^\Phi}$ can easily be deduced from the continuous model applied in Algorithm 4.1, cf. (4.13). Thus, the model at iterate (x_k, y_k, v_k) with step (d_k^c, w_k) is defined as

$$\begin{aligned}
\Psi_{\sigma_k^\Phi}(d_k^c, w_k) := \ & f(x_k, y_k) + \nabla_x f(x_k, y_k)^T d_k^c + \frac{1}{2} {d_k^c}^T B_k^c d_k^c \\
& - \sum_{j \in \mathcal{M}_k} \Big(\Big(v_j^{(k)} + w_j^{(k)} \Big) \Big(g_j(x_k, y_k) + \nabla_x g_j(x_k, y_k)^T d_k^c \Big) \\
& \qquad - \frac{1}{2} \sigma_k^\Phi \Big(g_j(x_k, y_k) + \nabla_x g_j(x_k, y_k)^T d_k^c \Big)^2 \Big) \\
& - \sum_{j \in \overline{\mathcal{M}}_k} \frac{1}{2} \frac{\Big(v_j^{(k)} + w_j^{(k)} \Big)^2}{\sigma_k^\Phi} \ ,
\end{aligned} \tag{5.24}$$

where the sets $\mathcal{M}_k$ and $\overline{\mathcal{M}}_k$ are defined the following way

$$\mathcal{M}_k := \mathcal{E} \cup \left\{ j \in \mathcal{I} \mid \ g_j(x_k, y_k) + \nabla_x g_j(x_k, y_k)^T d_k^c \leq \frac{v_j^{(k)} + w_j^{(k)}}{\sigma_k^\Phi} \right\} \tag{5.25}$$

and

$$\overline{\mathcal{M}}_k := \{1, \ldots, m\} \setminus \mathcal{M}_k \ . \tag{5.26}$$

Applying the step $(0, 0)$ to the model $\Psi_{\sigma_k^\Phi}$ results in the function value given by

$$\Psi_{\sigma_k^\Phi}(0, 0) := \ f(x_k, y_k) - \sum_{j \in \mathcal{M}_k^0} \Big(v_j^{(k)} g_j(x_k, y_k) - \frac{1}{2} \sigma_k^\Phi g_j(x_k, y_k)^2 \Big) \ - \sum_{j \in \overline{\mathcal{M}}_k^0} \frac{1}{2} \frac{{v_j^{(k)}}^2}{\sigma_k^\Phi} \ , \tag{5.27}$$

where the sets $\mathcal{M}_k^0$ and $\overline{\mathcal{M}}_k^0$ are defined by

$$\mathcal{M}_k^0 := \mathcal{E} \cup \left\{ j \in \mathcal{I} \mid \; g_j(x_k, y_k) \leq v_j^{(k)} / \sigma_k^\Phi \right\} \tag{5.28}$$

and

$$\overline{\mathcal{M}}_k^0 := \{1, \ldots, m\} \setminus \mathcal{M}_k^0 \; . \tag{5.29}$$

Obviously, the index sets obtained for the model $\Psi_{\sigma_k^\Phi}(0,0)$ and the augmented Lagrangian $\Phi_{\sigma_k^\Phi}(x_k, y_k, v_k)$ are identical, i.e., $\mathcal{M}_k^0 = \mathcal{S}_k$ and $\overline{\mathcal{M}}_k^0 = \overline{\mathcal{S}}_k$. Thus, the value of model $\Psi_{\sigma_k^\Phi}$ for the step $(0,0)$ equals the augmented Lagrangian function at iterate (x_k, y_k, v_k), i.e.,

$$\Psi_{\sigma_k^\Phi}(0,0) = \Phi_{\sigma_k^\Phi}(x_k, y_k, v_k) \tag{5.30}$$

holds.

Hence, the predicted reduction in the model is defined by

$$Pred_k := \Psi_{\sigma_k^\Phi}(0,0) - \Psi_{\sigma_k^\Phi}(d_k^c, w_k) \; . \tag{5.31}$$

To assure that the predicted reduction $Pred_k$ is sufficiently large, the penalty parameter σ_k^Φ has to be updated. Let σ_{k-1}^Φ be the penalty parameter of the last iteration. Then the penalty parameter σ_k^Φ is determined by

$$\sigma_k^\Phi := \max \left(\sigma_{k-1}^\Phi \; , \; \max_{1 \leq j \leq m} \left(\frac{2m \left(u_j^{(k)} - v_j^{(k)} \right)^2}{d_k^{c^T} B_k^c d_k^c + 2\mu_k \Delta_k^c} \right) \right) \; . \tag{5.32}$$

As B_k^c is required to be positive definite, $d_k^{c^T} B_k^c d_k^c$ is greater than zero as long as $\|d_k^c\|_\infty > 0$. The sequence of penalty parameter values is monotone increasing. The update formula (5.32) is stated in a reduced version compared to the formula (4.24) applied in Algorithm 4.1. The modification is straightforward to see. As only situations are considered where (5.21) holds, the variables $z_j^{(k)}$, $j = 1, \ldots, m$, present on the right-hand side of (4.24), are set to zero according to (4.11). Thus, they can be omitted here.

Now a mixed-integer algorithm that avoids second order correction steps can be stated. The algorithm modifies Algorithm 5.2 slightly. To simplify the notation in some steps it is referred to the corresponding steps and calculations of Algorithm 5.2.

Algorithm 5.3 Let $0 < \Delta_{\min} < \Delta_{\max} < \infty$, $\epsilon_{\text{tol}} > 0$, $0 < \rho < 1$, and $\bar{\sigma} > 0$ be given constants.

STEP 0 Choose initial values for $x_0 \in \mathbb{R}^{n_c}$, $y_0 \in Y$, $v_0 \in \mathbb{R}^m$, $\Delta_{\max} > \Delta_0^c > \Delta_{\min}$, $\Delta_0^i \geq 1$, $\sigma_0 > 0$, $\sigma_{-1}^\Phi \geq 1$, $\zeta_0 > 0$, and a positive definite matrix $B_0 \in \mathbb{R}^{(n_c+n_i)\times(n_c+n_i)}$.

Set $f^\star := \infty$ and $(x^\star, y^\star) := (x_0, y_0)$ to the current best known solution.

Evaluate function values $f(x_0, y_0)$ and $g(x_0, y_0)$.

Evaluate $\nabla_x f(x_0, y_0)$ and $\nabla_x g(x_0, y_0)$ with respect to continuous variables.

Use Procedure 5.1 to approximate $\nabla_y f(x_0, y_0)$ and $\nabla_y g(x_0, y_0)$ with respect to integer variables and obtain (x_0^{bn}, y_0^{bn}) and f_0^{bn}.

if $f_0^{bn} < f^\star$ **then** Set $f^\star := f_0^{bn}$ and $(x^\star, y^\star) := (x_0^{bn}, y_0^{bn})$.

Set $k := 0$.

STEP 1 Solve the mixed-integer quadratic problem (5.9) giving $d_k := \begin{pmatrix} d_k^c \\ d_k^i \end{pmatrix}$.

if $(\|g(x_k, y_k)^-\|_\infty \leq \epsilon_{\text{tol}}$ **or** $\sigma_k > \bar{\sigma})$ **and** $\psi_{\sigma_k}(0) - \psi_{\sigma_k}(d_k) \leq \epsilon_{\text{tol}}$ **then goto** STEP 7.

STEP 2 Determine σ_{k+1} and ζ_{k+1} as described in STEP 2 of Algorithm 5.2.

STEP 3 Evaluate $f(x_k + d_k^c, y_k + d_k^i)$ and $g(x_k + d_k^c, y_k + d_k^i)$.

if $\|(g(x_k, y_k) + \nabla g(x_k, y_k)^T d_k)^-\|_\infty = 0$ **and** $\|d_k^i\|_\infty = 0$ **then**

Obtain multipliers u_k that correspond to problem (5.22) and set $w_k := u_k - v_k$.

Determine penalty parameter σ_k^Φ according to (5.32).

Calculate ratio

$$r_k := \frac{\Phi_{\sigma_k^\Phi}(x_k, y_k, v_k) - \Phi_{\sigma_k^\Phi}(x_k + d_k^c, y_k, u_k)}{\Psi_{\sigma_k^\Phi}(0, 0) - \Psi_{\sigma_k^\Phi}(d_k^c, w_k)} \quad . \tag{5.33}$$

else Calculate ratio r_k according to (5.11).

Set $w_j^{(k)} := 0$, $j = 1, \ldots, m$.

Set penalty parameter $\sigma_k^\Phi := \sigma_{k-1}^\Phi$.

STEP 4 Update the trust region radii according to (5.13) and (5.14) in STEP 5 of Algorithm 5.2.

STEP 5 **if** $r_k < \rho$ **then** Set $(x_{k+1}, y_{k+1}) := (x_k, y_k)$, $v_{k+1} := v_k$, $B_{k+1} := B_k$, and $k := k + 1$ and **goto** STEP 1

else Set $(x_{k+1}, y_{k+1}) := (x_k + d_k^c, y_k + d_k^i)$ and $v_{k+1} := v_k + w_k$.

if $\Delta_{k+1}^c < \Delta_{\min}$ **then** Replace $\Delta_{k+1}^c := \Delta_{\min}$.

if $\Delta_{k+1}^c > \Delta_{\max}$ **then** Replace $\Delta_{k+1}^c := \Delta_{\max}$.

STEP 6 Evaluate partial derivatives $\nabla_x f(x_{k+1}, y_{k+1})$ and $\nabla_x g(x_{k+1}, y_{k+1})$ with respect to continuous variables.

Approximate $\nabla_y f(x_{k+1}, y_{k+1})$ and $\nabla_y g(x_{k+1}, y_{k+1})$ with respect to integer variables using Procedure 5.1 and obtain $(x_{k+1}^{bn}, y_{k+1}^{bn})$ and f_{k+1}^{bn}.

if $f_{k+1}^{bn} < f^\star$ **then** Set $f^\star := f_{k+1}^{bn}$ and $(x^\star, y^\star) := (x_{k+1}^{bn}, y_{k+1}^{bn})$.

Generate a positive definite matrix B_{k+1}. Set $k := k + 1$ and **goto** STEP 1.

STEP 7 **if** $\|g(x_k, y_k)^-\|_\infty \leq \epsilon_{\text{tol}}$ **and** $f^\star \geq f(x_k, y_k)$ **then** Set $f^\star := f(x_k, y_k)$, $(x^\star, y^\star) := (x_k, y_k)$ and **STOP** .

if $\|g(x_k, y_k)^-\|_\infty > \epsilon_{\text{tol}}$ **and** $f^\star = \infty$ **then** Report that the problem might be infeasible and **STOP** .

otherwise Set $(x_{k+1}, y_{k+1}) := (x^\star, y^\star)$ and $v_{k+1} := v_k$.

if $\Delta^c_{k+1} < \Delta_{\min}$ **then** Replace $\Delta^c_{k+1} := \Delta_{\min}$.

if $\Delta^c_{k+1} > \Delta_{\max}$ **then** Replace $\Delta^c_{k+1} := \Delta_{\max}$.

Evaluate function values $f(x_{k+1}, y_{k+1})$, $g(x_{k+1}, y_{k+1})$ and **goto** STEP 6.

The modified Algorithm 5.3 terminates the main loop as soon as no sufficient reduction is obtained subject to the model ψ_{σ_k} of the L_∞-penalty function (5.5), cf. (5.8). STEP 1 is identical in both mixed-integer algorithms. By going to STEP 7, the obtained information of the direct search strategy is checked, and a restart is performed when progress can be achieved.

Algorithm 5.3 employs the two penalty parameters σ_k and σ^Φ_k. Parameter σ_k is used by the L_∞-penalty function (5.5) and controls the progress with respect to the constraint violation in subproblem (5.9). The update rules are identical to the ones applied in Algorithm 5.2. Parameter σ^Φ_k relates to the augmented Lagrangian function $\Phi_{\sigma^\Phi_k}$, cf. (5.17), and the corresponding model $\Psi_{\sigma^\Phi_k}$, defined in (5.24). The penalty parameter σ^Φ_k is updated according to the rule of Algorithm 4.1, but only if the trial step is evaluated subject to augmented Lagrangian function. Otherwise, σ^Φ_k remains unchanged. The update rule is stated in adapted form in (5.32), where only the continuous trial step d^c_k is taken into account.

In STEP 3 a switch to the augmented Lagrangian merit function is performed in case conditions (5.20) and (5.21) are satisfied. Second order correction steps are not calculated anymore. In this case $\|d^i_k\|_\infty = 0$ holds, and the continuous part d^c_k of the solution to subproblem (5.9) is identical to the one obtained by the quadratic problem (5.22). The ratio r_k is calculated with respect to the augmented Lagrangian $\Phi_{\sigma^\Phi_k}$ and model $\Psi_{\sigma^\Phi_k}$ according to formula (5.33). If conditions (5.20) and (5.21) do not hold, then the ratio r_k is calculated with respect to the L_∞-penalty function, i.e., r_k is determined according to (5.11). In both cases a trial step is accepted if $r_k \geq \rho$ with a constant $0 < \rho < 1$. In Algorithm 5.2 only trial steps with $r_k \leq 0$ are rejected. The modification is required by the theory of the continuous Algorithm 4.1.

The trust region update with respect to the continuous variables is slightly modified compared to Algorithm 5.2. In case a step d_k is accepted, then the trust region radius Δ_{k+1} is set to at least $\Delta_{\min}$. Moreover, the upper bound $\Delta_{\max}$ on the continuous trial step d^c_k is introduced. This is similar to the procedure in Algorithm 4.1. Thus, the convergence properties obtained for the continuous Algorithm 4.1 can now be applied to Algorithm 5.3, at least in case y_k remains unchanged for a sequence of iterations.

The multiplier approximation is updated in STEP 5. In case the trial step is accepted subject to the augmented Lagrangian merit function (5.17), then the multipliers v_{k+1} are set to the multipliers u_k obtained by problem (5.22). Otherwise, the multipliers v_{k+1} remain unchanged.

The heuristics described for Algorithm 5.2 are also applicable and recommended for Algorithm 5.3. In particular the scaling of the Hessian approximation B_k, cf. (5.16), seems to be efficient as indicated by numerical tests.

5.3 Summary

Non-convex nonlinear mixed-integer optimization problems are extremely difficult to solve. Most of the concepts known from continuous nonlinear optimization are not available. In contrast to other methods, the relaxation of integer variables is not required by the new mixed-integer algorithms introduced here, since the algorithms do not evaluate function values for fractional values of an integer variable. Thus, the algorithms can also be applied to problems arising in different fields of engineering, where a relaxation with respect to integer variables is not possible due to the underlying simulation codes.

The presented adaptation of sequential quadratic programming techniques to mixed-integer optimization is a new approach that was discussed for the first time in Exler and Schittkowski [37], and Exler [33]. A convergence proof for Algorithm 5.2 and Algorithm 5.3 is not provided. A possible stabilization can be achieved by adding linear outer approximations techniques as done in Exler, Lehmann, and Schittkowski [36]. Convergence can be guaranteed for convex problems by adding the so-called linear mixed-integer master problem defined by linear outer approximation methods.

Both algorithms, i.e., Algorithm 5.2 and Algorithm 5.3, are implemented in the code `MISQP`, for more information see Exler et al. [35] or the documentation in Appendix A. In Chapter 6 numerical results obtained by the code `MISQP` are presented. The performance of the implementation of Algorithm 5.2, which applies second order correction steps, is compared to the implementation of Algorithm 5.3, where evaluating second order correction steps is avoided.

6 Numerical Results

The trust region SQP algorithms introduced in the previous chapters are implemented as components of the FORTRAN subroutine called MISQP, see Exler et al. [35]. In the following sections the numerical performance of the code MISQP is evaluated on two collections of test problems. The first set contains 306 nonlinear optimization problems without additional integer variables. The performance on mixed-integer optimization problems is evaluated on a second set consisting of 175 problems.

In the following section the test environment is described. The tested codes are listed and details on the implementations are stated. Section 6.2 presents the criteria that are used to evaluate and compare the performance of the different implementations. Thereafter, the test results are discussed.

6.1 Test Environment and Implementation Details

Numerical tests are performed for the new algorithms introduced in the previous chapters. The different algorithms are accessible by setting the corresponding option of MISQP. The performance of MISQP is compared to the well-established FORTRAN code NLPQLP, see Schittkowski [104]. This section summarizes implementation details concerning both codes.

If not mentioned explicitly, all test runs are performed with default values for options as defined in the corresponding documentation, see Exler et al. [35] or Appendix A, and Schittkowski [104]. The FORTRAN codes are compiled by the Intel Visual Fortran Compiler, Version 9.1, under Windows XP64 and executed on an Intel Core(TM)2 6600 64 bit processor with 2.40 GHz and 4 GB RAM.

6.1.1 The FORTRAN Package MISQP

Three different implementations are tested on continuous nonlinear optimization problems of form (1.2). MISQP offers the choice between versions that apply second order correction steps and versions that avoid these additional computations. Thus, the performance of both strategies can be compared easily. The different versions of MISQP are denoted by the highlighted names.

MISQP/lag	Implementation of Algorithm 4.1. Second order correction steps are avoided by applying an augmented Lagrangian merit function. MISQP is called with IOPT(7)=0.
MISQP/soc	Implementation of an SQP trust region method proposed by Yuan [130], cf. Algorithm 3.3. Additional second order correction steps are calculated. MISQP is called with IOPT(7)=1.

`MISQP/com`	Implementation of a modified version of Algorithm 3.3. Computation of SOC steps is avoided by locally switching to the augmented Lagrangian merit function. Combines versions `MISQP/lag` and `MISQP/soc`. The applied strategy is similar to the one used by the mixed-integer Algorithm 5.3. `MISQP` is called with `IOPT(7)=2`.

The code `MISQP` offers two versions that address the mixed-integer nonlinear problem (1.1). The corresponding algorithms are presented in Chapter 5. Both versions are evaluated and compared. The identifiers are extended by '`/minlp`' to highlight the mixed-integer characteristic of the problems under consideration.

`MISQP/soc/minlp`	Mixed-integer SQP-based trust region method, SOC steps are computed for continuous variables. Implementation of Algorithm 5.2. `MISQP` is called with `IOPT(7)=1`.
`MISQP/com/minlp`	Mixed-integer SQP-based trust region method, fast local convergence with respect to continuous variables is obtained without calculating SOC steps. Implementation of Algorithm 5.3. `MISQP` is called with `IOPT(7)=2`.

The remainder of this section presents some details concerning the implementation of `MISQP`. Algorithm 4.1 requires that some parameters are set to specific values when coding the algorithm. In `MISQP` these parameters are set to the following values

$$\tau_1 = 0.5\ ,\ \ \tau_2 = 2\ ,\ \ \rho_0 = 0.1\ ,\ \ \rho_1 = 0.75\ ,\ \ \Delta_{\min} = 10^{-5}\ ,\ \ \Delta_{\max} = 10^{10}\ . \tag{6.1}$$

The values were obtained by extensive numerical tests and seem to be the most efficient choice.

The Hessian matrix plays a crucial role in SQP methods. To avoid the computation of second derivatives, the matrix B_k used in the subproblems is set to a positive definite approximation of the Hessian of the Lagrangian function. To guarantee a superlinear convergence rate, B_k is updated by the BFGS formula

$$B_{k+1} := B_k + \frac{p_k p_k^T}{d_k^T p_k} - \frac{B_k d_k d_k^T B_k}{d_k^T B_k d_k}\ , \tag{6.2}$$

with

$$p_k := \nabla_x L(x_{k+1}, u_k) - \nabla_x L(x_k, u_k)\ , \tag{6.3}$$

$$d_k := x_{k+1} - x_k\ . \tag{6.4}$$

The initial matrix B_0 is set to the unit matrix and the formula is stabilized by requiring $p_k^T d_k \geq 0.2 d_k^T B_k d_k$. The BFGS update is also applied when mixed-integer problems are optimized. In this case, the formulas (6.2)-(6.4) are adapted accordingly.

As soon as the conditions outlined in the corresponding algorithms are satisfied, MISQP terminates. In addition, the following stopping criterion is implemented for the continuous versions of MISQP, see also Schittkowski [103]. MISQP terminates if

$$\|g(x_k)^-\|_\infty < \epsilon_{\text{tol}} \tag{6.5}$$

and

$$|\nabla f(x_k)^T d_k| + \mu_k \Delta_k + \sum_{j=1}^{m} \left| u_j^{(k)} g(x_k) \right| < \epsilon_{\text{tol}} \tag{6.6}$$

hold, where x_k is the current iterate and (d_k, u_k) denotes the solution of the corresponding subproblem at iteration k. μ_k denotes the approximation of the multipliers corresponding to the trust region constraint, see definition (4.35). The term $\mu_k \Delta_k$ is added to prevent early termination in case the trust region constraint is active. The desired accuracy is denoted by $\epsilon_{\text{tol}} > 0$. The termination criterion defined by (6.5) and (6.6) is adapted for the mixed-integer case appropriately.

In Chapter 5 some heuristics, e.g., the modification of the Hessian approximation, are discussed. They can be applied to Algorithm 5.2 and Algorithm 5.3. All proposed heuristics are implemented in MISQP. They can be switched on and off by changing the corresponding entries in the option arrays of MISQP.

The usage of MISQP and the parameters are described in Appendix A. Note that ϵ_{tol} corresponds to the parameter ACC of MISQP. An extended documentation can be found in Exler et al. [35]. The mixed-integer quadratic programming problems are solved by the code MIQL of Lehmann et al. [65], an implementation of a branch-and-cut method. The underlying continuous quadratic programs are solved by an extended version of the code QL. The code QL traces back to Powell [87], see also Schittkowski [105], and is an implementation of the primal-dual method of Goldfarb and Idnani [50].

6.1.2 A Reference Code – NLPQLP

In case the performance is evaluated on problems without additional integer variables, then MISQP is compared to the FORTRAN code NLPQLP, see Schittkowski [104]. The FORTRAN subroutine NLPQLP solves smooth nonlinear programming problems and is an extension of the code NLPQL, see Schittkowski [101]. The new algorithms introduced here and NLPQLP differ in the globalization strategy, since NLPQLP applies line search techniques instead of a trust region stabilization. NLPQLP is tuned to run under distributed systems and to apply non-monotone line search in error situations. The input parameter L of NLPQLP is introduced to specify the number of parallel machines. The tests are executed with L set to 1, i.e., NLPQLP behaves almost identical to NLPQL. The quadratic subproblems are solved by the subroutine QL, see Schittkowski [105]. NLPQLP also applies the BFGS update formula (6.2)-(6.4) for generating second order information. Moreover, the termination criterion stated before, see (6.5) and (6.6), is also implemented in NLPQLP. NLPQLP is executed with parameters set to the values shown in Table 6.1. If not stated differently, NLPQLP is tested in standard mode.

Option	Value	Description
`L`	1	number of parallel systems,
`STPMIN`	10^{-10}	minimum steplength in case of `L>1` (not used),
`MAXFUN`	30	upper bound for the number of function calls during the line search,
`MAXNM`	30	stack size for storing merit function values at previous iterations for non-monotone line search,
`RHOB`	10^4	parameter for performing a restart in case of `IFAIL=2` by setting the BFGS-update matrix to `RHOB` $\cdot$ `I`, where `I` denotes the identity matrix,
`MODE`	0	standard execution.
`MODE`	> 0	Safeguard strategy that applies non-monotone line search.

Table 6.1: Parameter settings for `NLPQLP`

6.2 Performance Evaluation

The following strategies for evaluating the performance of a code are frequently used and are also applied by Exler et al. [36] for a comparative study. First, criteria have to be defined to decide whether the outcome of a test run is considered as a successful return or not. The applied criteria are stated for the continuous case and can easily be adapted to mixed-integer optimization problems. Let $\epsilon_{succ} > 0$ be a tolerance for defining the relative accuracy, x_k be the returned value of a test run, and $x^\star$ the supposed exact solution known from the test problem collection. Then the output is called a *successful solution*, if the relative error in the objective function is less than ϵ_{succ} and if the maximum constraint violation is less than ϵ_{succ}^2, i.e., if

$$f(x_k) - f(x^\star) < \epsilon_{succ} |f(x^\star)| \ , \quad \text{if } f(x^\star) \neq 0 \ , \tag{6.7}$$

or

$$f(x_k) < \epsilon_{succ} \ , \quad \text{if } f(x^\star) = 0 \ , \tag{6.8}$$

and

$$\|g(x_k)^-\|_\infty < \epsilon_{succ}^2 \ . \tag{6.9}$$

Note that the tolerance for the allowed constraint violation, cf. (6.9), might lead to returned solutions with an objective function value better than the best known one in $x^\star$. These runs are also classified as successful solutions.

Another situation is taken into account. It might occur that the internal termination conditions of a code are satisfied subject to a reasonably small tolerance, but the objective function value of the returned solution is worse than the best known one. For non-convex problems this situation is not unusual. If such a solution is returned

by a test run, it is also accepted and is called an *acceptable solution*, respectively. For an acceptable solution

$$f(x_k) - f(x^\star) \geq \epsilon_{succ}|f(x^\star)| \ , \quad \text{if } f(x^\star) \neq 0 \ , \tag{6.10}$$

or

$$f(x_k) \geq \epsilon_{succ} \ , \quad \text{if } f(x^\star) = 0 \ , \tag{6.11}$$

and

$$\|g(x_k)^-\|_\infty < \epsilon_{succ}^2 \tag{6.12}$$

hold. The numerical tests are evaluated with $\epsilon_{succ} = 0.01$, i.e., a relative final accuracy of one percent is required for a run to be considered as *successful*.

Different methodologies are applied to evaluate the performance of the tested implementations. Arithmetic mean values are compared on different performance criteria, e.g., the number of function evaluations or the number of gradient evaluations. But it might be misleading to simply compare arithmetic mean value, as they might be dominated by few problems with extremely high results in the considered performance criterion. Especially, if some other codes are unable to solve these problems the average numbers might be inaccurate. On the other hand, restricting the comparison of arithmetic mean values to the set of test problems that are solved successfully by all tested codes would penalize the more reliable and efficient codes.

To overcome these difficulties, two additional techniques are applied. The first one is known under the name *priority theory*, see Saaty [96], and has been used for example by Schittkowski [98] and Hock and Schittkowski [62] for comparing optimization codes. Appendix B contains an outline of the theoretical background of the procedure. The basic idea can be summarized as follows. The output of this method is a unique priority value, by which the relative efficiency of one code over another one is measured. This is achieved by comparing the codes pairwise with respect to a specified performance criterion over sets of test examples, which are successfully solved by both codes. Then a reciprocal $N \times N$ matrix is determined, where N is the number of codes under consideration. The largest eigenvalue of this matrix is positive and its normalized eigenvector is computed. The priority values are deduced from the eigenvector after scaling the eigenvector such that the smallest coefficient becomes one. The interpretation of these priority values is illustrated by an example. Say a relative priority value 3.0 for the number of function evaluations leads to the conclusion that the corresponding code needs 3.0 times more function calls than the best one with priority value 1.0, and twice as many as a code with priority value 1.5.

In addition, the performance is evaluated according to an approach developed by Dolan and Moré [27]. The so-called *performance profiles* are frequently used in comparative numerical studies. The creation of performance profiles is explained in the following. Let the number of function evaluations be the performance criterion under consideration. It is assumed that N codes C_i, $i = 1, \ldots, N$, are compared on a set of M test problems TP_j, $j = 1, \ldots, M$. First, the minimum number of function evalua-

Parameter	Value	Description
`ACC`	10^{-7}	termination accuracy, i.e., ϵ_{tol},
`ACCQP`	10^{-12}	termination accuracy of QP solver,
`MAXIT`	$3,000$	maximum number of iterations.

Table 6.2: Parameter settings for 306 Continuous Tests for `MISQP` and `NLPQLP`

tions needed by the codes to solve the problem successfully is determined for each test problem TP_j, $j = 1, \ldots, M$. The minimum number is denoted by $n_j^\star$ and is defined as follows

$$n_j^\star := \min_{1 \le i \le N} (n_{ij}) \ , \tag{6.13}$$

where either n_{ij} corresponds to the actual number of function evaluations code C_i needed to solve problem TP_j *successfully*, or n_{ij} is set to a large constant value, otherwise.

Now $n_j^\star$ is the reference value for all codes on the test problem TP_j, and the number of function evaluations need by each code C_i, $i = 1, \ldots, N$, is compared to $n_j^\star$ by calculating the ratio

$$r_{ij} := \frac{n_{ij}}{n_j^\star} \ . \tag{6.14}$$

Thus, a value 1 for r_{ij} indicates that code C_i is the best solver on problem TP_j. A value 4 implies that the corresponding code C_i needed four times more function evaluations than the best solver on problem TP_j required.

For each code C_i, $i = 1, \ldots, N$, let $S_i(r)$ denote the set of test problems where the ratio r_{ij}, as defined by (6.14), is lower or equal to a given upper bound r, that is

$$S_i(r) := \{ j \mid \ r_{ij} \le r \ , \ \ 1 \le j \le M \} \ . \tag{6.15}$$

Applying (6.15), then the function $\phi_i(r) : [1, \infty) \to [0, 1]$ is defined as

$$\phi_i(r) := \frac{|S_i(r)|}{M} \ , \tag{6.16}$$

for each code C_i, $i = 1, \ldots, N$, where $|S_i(r)|$ is the cardinal number of set $S_i(r)$. The function $\phi_i(r)$ represents the percentage of test problems that are solved by code C_i with at most r times more function evaluations than the particular best solver on a problem. Performance profiles display the functions $\phi_i(r)$ for all codes under consideration, where r is given along the abscissa starting with 1.

Performance profiles can be interpreted in the following way. A high value for $\phi_i(1)$ indicates that the solver is efficient compared to the other solvers, as the function value represents the percentage of problems where the code needed the fewest value, with respect to the performance criterion under consideration, compared to all codes.

code	n_{succ}	n_{acc}	n_{err}	n_{func}	n_{grad}	n_{allf}	*time*
`MISQP/lag`	280	25	1	25	19	266	0.75
`MISQP/soc`	279	25	2	70	35	433	1.64
`MISQP/com`	283	22	1	25	19	267	0.94
`NLPQLP`	284	22	0	42	21	312	0.58

Table 6.3: Performance Results for a Set of 306 Continuous Test Problems

On the other hand, the robustness and reliability of a solver corresponds to a curve that approaches 1 when r increases.

6.3 Continuous Optimization Problems

The 306 academic and real-life test problems used to evaluate the performance on continuous optimization problems are published in Hock and Schittkowski [61], and in Schittkowski [102]. The test examples are provided with optimal objective function values $f(x^\star)$, either known from analytical investigations or from the best numerical data found so far.

Partial derivatives are approximated by forward differences,

$$\frac{\partial}{\partial x_i} f(x) \approx \frac{1}{\theta_i}\Big(f(x+\theta_i e_i) - f(x)\Big) \;, \tag{6.17}$$

where $\theta_i := \theta_m^{1/2} \max(10^{-5}, |x_i|)$ and e_i is the i-th unit vector, $i = 1, \ldots, n$. The tolerance θ_m represents the machine accuracy. In a similar way, derivatives of constraints are approximated. Note that each time a gradient is approximated, the corresponding function is evaluated n times.

`MISQP` and `NLPQLP` are executed within the same test environment and called with the same parameter values shown in Table 6.2. The obtained results are summarized in Table 6.3, where the columns are defined as

- n_{succ} - number of successful test runs according to above definition (6.7)-(6.9),
- n_{acc} - number of acceptable solutions according to (6.10)-(6.12),
- n_{err} - number of test runs terminated by an error message,
- n_{func} - mean value of number of function evaluations,
- n_{grad} - mean value of number of gradient evaluations,
- n_{allf} - mean value of number of all function evaluations (function evaluations for gradient approximation included),
- *time* - time for whole test set (in seconds).

The values for n_{func} or n_{grad} are obtained by counting each evaluation of a whole

code	n_{succ}	p_{func}	p_{grad}	p_{allf}	p_{time}
MISQP/lag	280	1.0	1.0	1.0	1.2
MISQP/soc	279	2.0	1.3	1.2	2.1
MISQP/com	283	1.0	1.0	1.0	1.4
NLPQLP	284	1.6	1.1	1.2	1.0

Table 6.4: Priority Values for a Set of 306 Continuous Test Problems

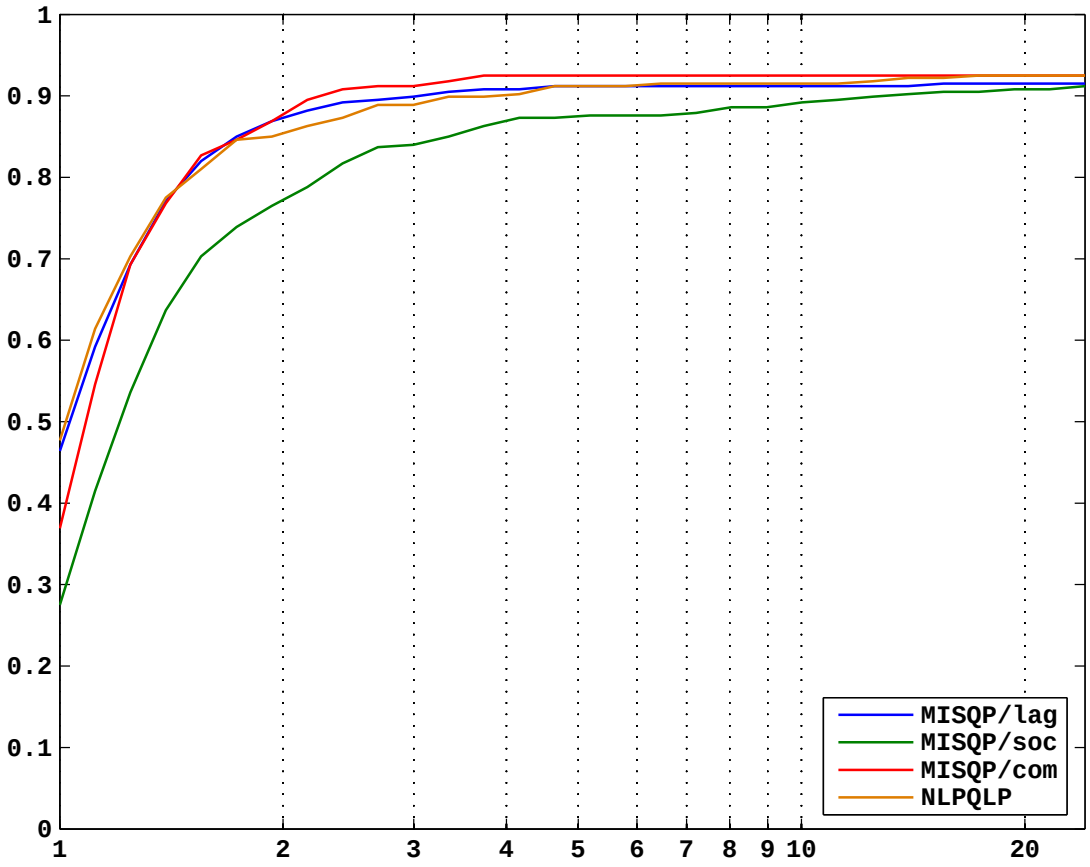

Figure 6.1: Performance Profiles for a Set of 306 Continuous Problems

set of function or gradient values, respectively, for a given iterate x_k. The additional function evaluations needed for gradient approximations are not counted for n_{func}, however, they are included in n_{allf}.

Table 6.3 shows that all codes can solve almost all problems to at least an acceptable solution according to (6.10)-(6.12). All three versions of `MISQP` terminate with an error message for one test problem. They are not able to terminate at a feasible iterate. `MISQP/soc` cannot solve a second problem. Again the code converges toward an infeasible point. The number of local solutions, i.e., n_{acc}, is almost identical for all codes.

`NLPQLP` is obviously the fastest solver with respect to the time needed for the whole test set. As `NLPQLP` is the only code using line search instead of trust region techniques to stabilize the convergence, this can be expected. Each search step in trust region methods corresponds to the solution of a quadratic subproblem. It may require several runs of the quadratic programming solver to obtain a new acceptable iterate, whereas in line search methods only one quadratic program is solved and the next iterate lies along the calculated search direction. Applying warm start techniques in the quadratic programming solver can improve the performance of trust region algorithms. `MISQP/lag` is the only version of `MISQP` that makes use of the warm start capability of the applied version of the solver `QL`. Comparing `MISQP/lag` and `MISQP/soc`, the difference in the measured time occurs probably due to the warm starts applied by `MISQP/lag`, since both codes behave identical in the other performance criteria. However, the gap to the line search code `NLPQLP` still remains.

Comparing the average number of function evaluations – with or without function evaluations for gradient approximations included – Table 6.3 leads to the conclusion that `MISQP/lag` and `MISQP/com` behave almost identically and outperform `MISQP/soc` significantly. The mean values shown for `MISQP/soc` are dominated by three problems, where `MISQP/soc` converges very slowly. Nevertheless, the results indicate that the algorithms, that avoid calculating second order correction steps, converge locally with fast rate. `NLPQLP` also needs less function and gradient evaluations than `MISQP/soc`, but a few more than `MISQP/lag` and `MISQP/com`.

For the calculation of priority values only runs that are considered to be successful according to (6.7)-(6.9) are taken into account. The calculated priority values are given in Table 6.4, where the columns denote

- n_{succ} - number of runs that obtain successful solutions,
- p_{func} - relative priority of function evaluations,
- p_{grad} - relative priority of gradient evaluations,
- p_{allf} - relative priority of all function calls including function calls used for gradient approximations,
- p_{time} - relative priority of execution time.

The results shown in Table 6.4 attest that `NLPQLP` is the fastest solver with respect to the time needed for all test problems. `MISQP/soc` needs more than twice the time of

Option	Value	Description
ACC	10^{-6}	termination accuracy,
ACCQP	10^{-10}	termination accuracy of mixed-integer QP solver,
MAXIT	500	maximum number of iterations,
MAXNDE	$5,000$	maximum number of branch-and-bound nodes in the subproblem solver.

Table 6.5: Parameter settings for 175 Mixed-Integer Test Problems for MISQP

NLPQLP. MISQP/lag and MISQP/com perform identically on the other performance criteria. Both codes outperform NLPQLP slightly. Comparing MISQP/lag and MISQP/soc, you see that the value of p_{grad} for MISQP/soc is 1.3 times higher than the one of MISQP/lag, but the value of p_{func} is two times the value of MISQP/lag. This can be explained by the additional second order correction steps calculated by MISQP/soc. Another indication of the need to avoid the SOC steps.

The performance profiles given in Figure 6.1 are based on the total number of function evaluations obtained by summing up the number of function calls required by the algorithm and the number of function calls needed for approximating gradients, i.e., n_{allf} is considered. Again only instances that are solved successfully according to (6.7)-(6.9) are taken into account. All other runs are assumed to fail. In case a run returns an acceptable solution or terminates with an error, then the number of required function evaluations is set to a predefined large constant when determining (6.13) and (6.14). Take a look at the green curve for MISQP/soc. In 28 percent MISQP/soc is the solver with the lowest number of function evaluations. 75 percent of the problems are solved by MISQP/soc within at most twice as much function evaluations than the best solver. The profile also shows that there are a few problems where MISQP/soc needs more than 20 times the number of function evaluations than the best solver. NLPQLP is the best solver in almost 50 percent of the problems. MISQP/lag follows closely. Once again MISQP/soc has the worst performance of all solvers.

The tests show that problems can be efficiently solved without the calculation of second order corrections steps. This was the intention of the development of Algorithm 4.1. The corresponding implementation, referred to as MISQP/lag, outperforms version MISQP/soc significantly. MISQP/soc is the implementation of Algorithm 3.3 with additional second order correction steps.

6.4 Mixed-Integer Optimization Problems

The performance on mixed-integer optimization problems is evaluated on a collection of 175 academic test examples published in Schittkowski [106]. The objective function value $f(x^\star)$ provided for each test problem has been found in the literature or has been

obtained by extensive testing over several years. There are at most 180 continuous, 100 integer, and 138 binary variables. The total number of variables is at most 200. Moreover, there are up to 189 constraints, including nonlinear equality constraints. The number of equality constraints is at most 87. The problems feature nonlinear functions where some of them are also non-convex. The feasible region of some instances is non-convex, too. Most of the test problems are taken from the GAMS MINLPLib, see Bussieck, Drud, and Meeraus [13].

`MISQP/soc/minlp` and `MISQP/com/minlp` are called with the parameter settings given in Table 6.5. Derivatives subject to continuous variables are approximated by forward differences, see (6.17), whereas by default integer derivatives are evaluated internally by `MISQP/soc/minlp` and `MISQP/com/minlp`, i.e., by two-sided differences at neighboring grid points. For binary variables or for variables at a bound, a forward or backward difference formula is applied, respectively, see Procedure 5.1.

`MISQP` is designed to handle integer variables that are non-relaxable and where partial derivatives have to be approximated internally. In addition, `MISQP` offers the opportunity to provide externally calculated partial derivatives with respect to the integer variables. In this case derivatives supplied by user are employed instead of the internal approximations. As all test problems are relaxable, the partial derivatives with respect to integer variables can be approximated the same way as for continuous variables. The problem functions are evaluated at fractional values with respect to integer variables only to approximate derivatives externally. All points visited by `MISQP` still fulfill the integer requirement. The performance of `MISQP` executed with external approximations is also evaluated. These runs are marked by adding "`/*`" to the names `MISQP/soc/minlp` and `MISQP/com/minlp`. The partial derivatives with respect to relaxed binary variables are computed by forward difference formula (6.17), as for the continuous variables. A two-sided difference formula is applied for the relaxed integer variables to make the effort comparable to the internal approximation procedure of `MISQP`. The two-sided differences formula is

$$\frac{\partial}{\partial x_i} f(x) \approx \frac{1}{2\theta_i}\Big(f(x+\theta_i e_i) - f(x-\theta_i e_i)\Big)\ , \tag{6.18}$$

where θ_i and e_i are defined as for the forward difference (6.17).

The criterion for evaluating the efficiency of the implementations is the total number of all function evaluations needed, i.e., the number n_{allf} is considered. The function calls needed for approximating gradients are included. In real-world applications when function evaluations require the execution of time-consuming simulation tools, the total number of function evaluations decides whether an optimization method is applicable or not. Table 6.6 shows the obtained results. The columns denote

- n_{succ} - number of successful test runs according to above definition, i.e., (6.7)-(6.9) adapted to mixed-integer problems,
- n_{acc} - number of acceptable solutions according to above definition, i.e., (6.10)-(6.12) adapted to mixed-integer problems,

code	n_{succ}	n_{acc}	n_{err}	p_{allf}	n_{allf}	p_{time}	$\varnothing time$
`MISQP/soc/minlp`	146	25	4	1.2	1629	1.3	24.17
`MISQP/com/minlp`	150	21	4	1.1	1506	1.3	23.74
`MISQP/soc/minlp/*`	136	34	5	1.4	1745	1.2	17.46
`MISQP/com/minlp/*`	139	31	5	1.0	1254	1.0	24.55

Table 6.6: Performance Results for a Set of 175 Mixed-Integer Problems

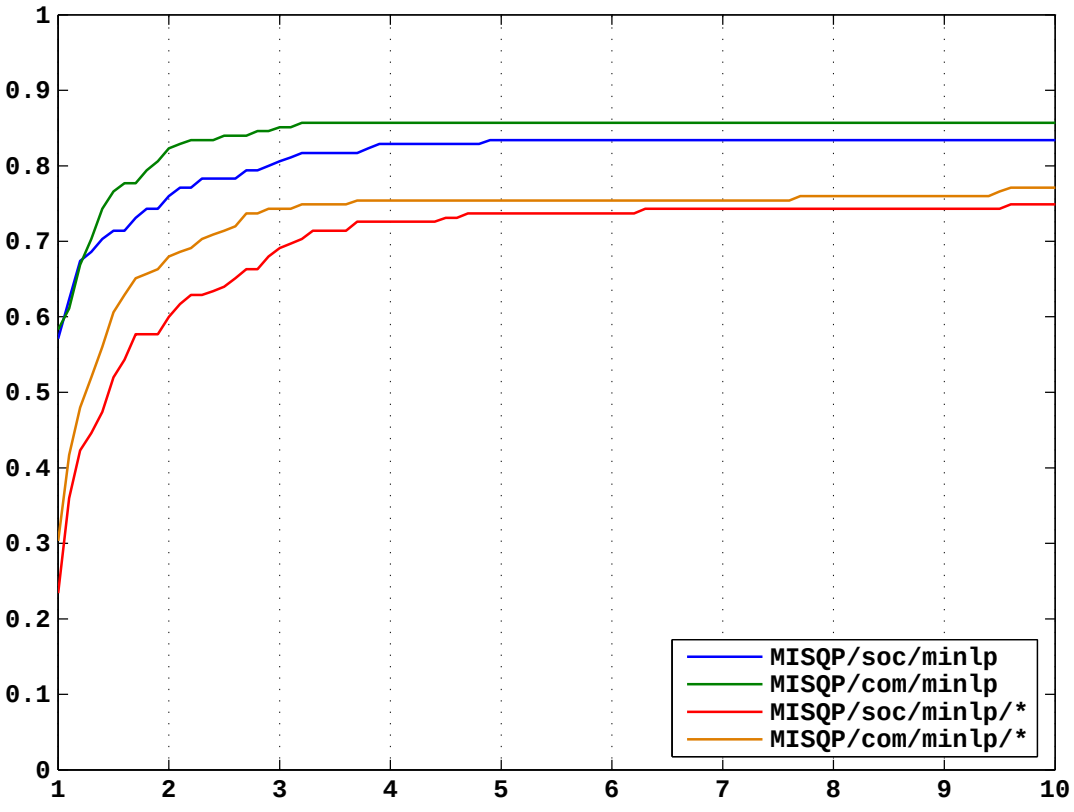

Figure 6.2: Performance Profiles for a Set of 175 Mixed-Integer Problems

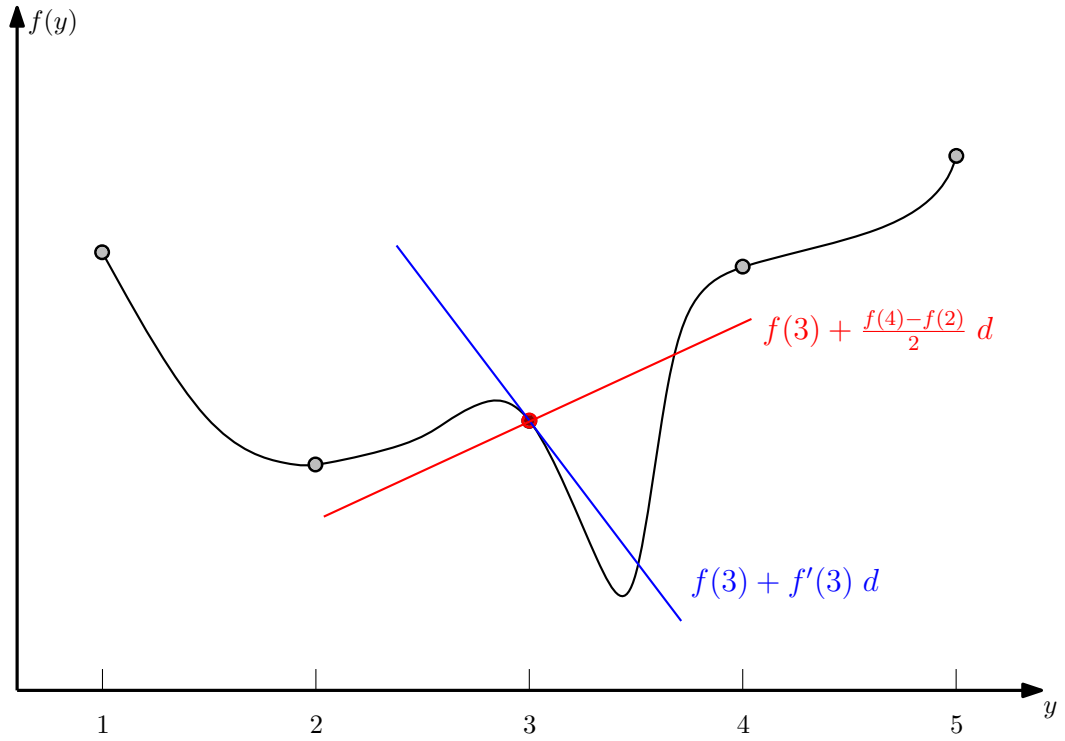

Figure 6.3: Example of Approximation of Integer Derivatives

n_{err}	-	number of test runs terminated by an error message,
p_{allf}	-	relative priority of all function calls including function calls used for gradient approximations, evaluated over all successful test runs,
n_{allf}	-	average number of all function calls including function calls used for gradient approximations, evaluated over all successful test runs,
p_{time}	-	relative priority of execution times, evaluated over all successful test runs,
$\varnothing time$	-	average execution times in seconds, evaluated over all successful test runs.

`MISQP/soc/minlp` and `MISQP/com/minlp` solve more problems successfully in case partial derivatives with respect to integer variables are approximated internally, instead of using external partial derivatives. On the other hand, it has to be mentioned that `MISQP/com/minlp/*` has the best priority value with respect to the total number of function evaluations. `MISQP/com/minlp` is following closely with a value 1.1. But taking into account that `MISQP/com/minlp` is able to solve 11 problems more, the small difference in the priority value can be neglected.

A possible explanation for the superiority of the versions, where partial derivatives with respect to integer variables are approximated internally, is given by Figure 6.3. The displayed function f is non-convex and a phenomenon is illustrated that might

arise for non-convex mixed-integer problems. The red line corresponds to the linear approximation of function f at point $y = 3$ obtained by approximating the derivative according to Procedure 5.1. Whereas, the blue line applies the analytical derivatives of f. You can see that the information obtained by the blue line is completely misleading for the mixed-integer problem, although it is a suitable approximation for the continuous problem.

The additional neighborhood search, which is performed in case partial derivatives are approximated by Procedure 5.1, can also be an explanation for the better results obtained by `MISQP/com/minlp` and `MISQP/soc/minlp`. The information gained by Procedure 5.1 can guide the search to more promising areas. Thus, combining the mixed-integer SQP steps and direct search strategies seems to be more reliable.

When comparing `MISQP/com/minlp` and `MISQP/soc/minlp`, you see that `MISQP/com/minlp` solves more problems to optimality than `MISQP/soc/minlp`, independent of the kind of partial derivative approximation. The average number of function evaluations required by `MISQP/com/minlp` reduces about more than 7% compared to `MISQP/soc/minlp`. Four problems are not solved to feasibility. Both codes generate an error message. When external partial derivatives are used, one more problem terminates at an infeasible iterate. In a couple of other situations, the codes are unable to solve the problems to global optimality and report that a feasible solution is obtained. Taking the total number of function evaluations as performance criterion, it can be concluded that `MISQP/com/minlp` is preferable. The aim of the development of Algorithm 5.3, that corresponds to `MISQP/com/minlp`, was the reduction of the required number of function evaluations. This goal is achieved obviously. This conclusion is confirmed by Figure 6.2, where the performance profiles for the mixed-integer problems are shown. Overall it turns out that `MISQP/com/minlp` is the most efficient and reliable code.

6.4.1 Results for Relaxed Problem Formulation

As already mentioned before, all test problems contained in the collection of mixed-integer nonlinear problems are relaxable. In the remainder of this chapter the performance results obtained on the mixed-integer problems are compared to the results achieved on the relaxed formulation of the mixed-integer problems, i.e., the domain of the integer variables is relaxed to $\mathbb{R}$. Again, the performance of `MISQP` is compared to the one of `NLPQLP`. The tests are executed with the same termination tolerances as the mixed-integer codes, see Table 6.5.

The specific parameter values set for `NLPQLP` are listed in Table 6.1. `NLPQLP` is tested with two different parameter settings. `NLPQLP` is executed in standard form by setting `MODE=0`. Additionally, `NLPQLP` is called in safeguard mode with `MODE=17`. The codes are denoted by `NLPQLP (mode=0)` and `NLPQLP (mode=17)`, respectively. The value 17 for `MODE` stands for initial and repeated scaling every 17th step, i.e., the BFGS matrix is reset to a multiple of the identity matrix.

For the relaxed problems the partial derivatives with respect to the integer and bi-

code	n_{succ}	n_{acc}	n_{err}	p_{allf}	n_{allf}	p_{time}	$\varnothing time$
`MISQP/lag`	150	17	8	1.0	1262	1.0	0.07
`MISQP/soc`	153	16	6	1.4	2300	2.5	0.55
`MISQP/com`	156	13	6	1.0	1790	1.2	0.35
`NLPQLP (mode=0)`	140	15	20	1.2	1296	1.0	0.05
`NLPQLP (mode=17)`	146	12	17	1.0	1042	2.9	0.14

Table 6.7: Results for a Set of 175 Relaxed Mixed-Integer Test Problems

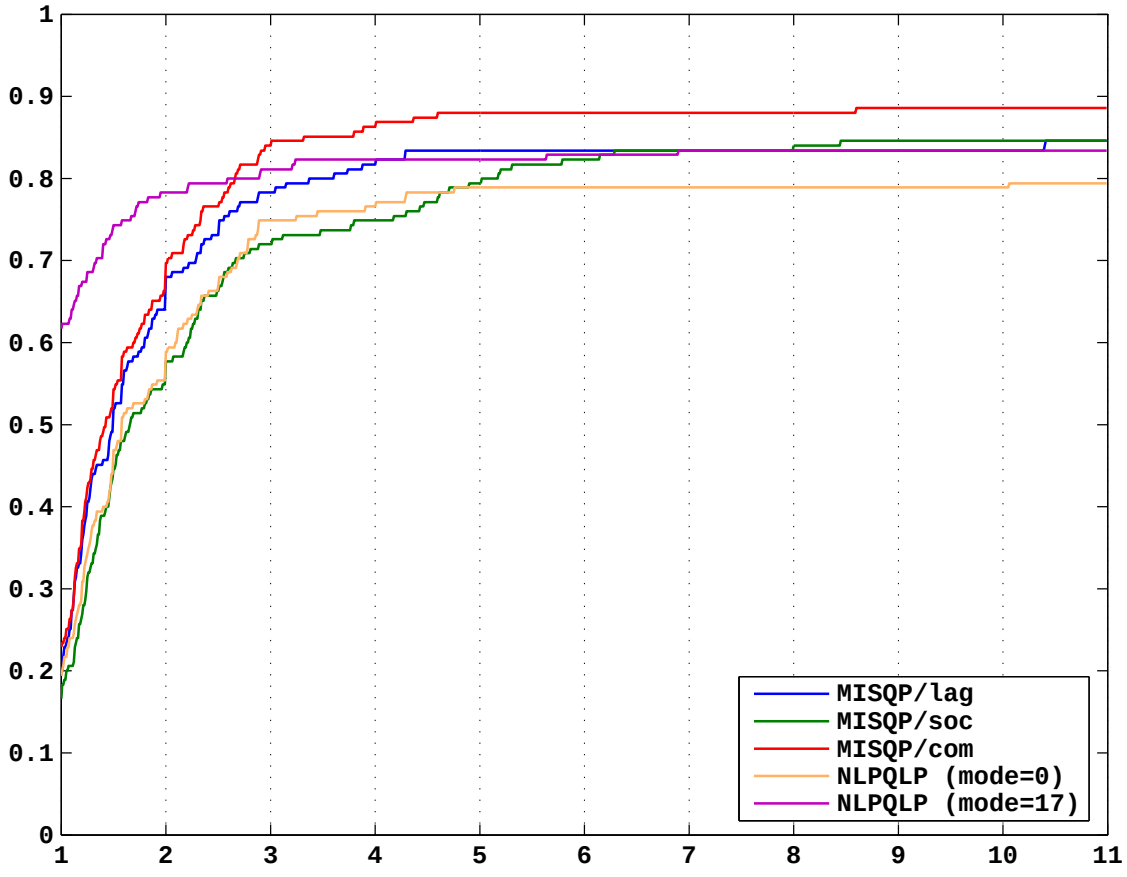

Figure 6.4: Performance Profiles for a Set of 175 Relaxed Mixed-Integer Problems

nary variables are approximated externally, as described above, i.e., partial derivatives with respect to relaxed binary variables are computed by forward difference formula (6.17), as for the continuous variables. For relaxed integer variables the two-sided formula (6.18) is applied.

Table 6.7 summarizes the numerical results obtained for the relaxed reformulations of the mixed-integer optimization problems. The column titles are the same as for Table 6.6. Note that the returned objective function value of a run on the relaxed problems has to be lower or equal to the best known objective function value of the corresponding mixed-integer problem, in order to be considered a successful return.

More runs are terminated with an error message than in the mixed-integer case. This indicates that the problems are hard to solve. The codes solve at most 156 problems successfully, i.e., they terminate at a feasible iterate where the KKT optimality conditions are satisfied subject to a tolerance 10^{-6}, and where the objective function value is less than or equal to the known one for the mixed-integer problem formulation. In all other situations, the codes either fail to solve the problem, or stop at a local solution which is worse than the known one for the mixed-integer formulation. Thus, it is hard to expect that `MISQP` will ever be able to get significantly higher scores for n_{succ} for the more complex mixed-integer version of the test set.

The number of problems successfully solved by `NLPQLP` is lower than the ones obtained by the different versions of `MISQP`. On this set of test problems, `NLPQLP` is less reliable and robust than `MISQP`. It is the other way around on the previously tested continuous set, see Section 6.3. The robustness of `NLPQLP` can be improved by changing option `MODE` to 17. The price to pay is an increase of the average time spent in solving a problem.

The average number of function evaluations needed by `MISQP/soc` is twice the one required by `MISQP/lag` and both versions of `NLPQLP`. As the value is also higher for `MISQP/com`, where the augmented Lagrangian merit function is chosen only locally, this difference might be explained by the use of the L_∞-penalty function as merit function and the resulting subproblems. Some problems require significantly more function evaluations in case the L_∞-penalty function is employed.

The performance profiles for the relaxed problems are shown in Figure 6.4. According to these profiles, `NLPQLP (mode=17)` is the most efficient code in more than 60% of the problems. On the other hand, `MISQP/com` is more robust. Again `MISQP/com` and `MISQP/lag` perform better than `MISQP/soc`. On the relaxed problems the performance profiles indicate a similar behavior of `MISQP/soc` and `NLPQLP (mode=0)`.

Finally, the results obtained on the relaxed test set, on the one hand, and the mixed-integer version of the test set, on the other hand, are compared. The data are shown in Table 6.8. The average number of function evaluations required by `MISQP/soc` on the relaxed set is much higher than the one needed by `MISQP/com/minlp` and `MISQP/soc/minlp` for the mixed-integer problems. Regarding priority value p_{allf}, the value calculated for `MISQP/soc` is almost identical to the ones obtained by the mixed-integer solvers. This result is very interesting. Assume `MISQP/soc` is employed in a basic branch-and-bound framework. Then the branch-and-bound method would re-

code	n_{succ}	n_{acc}	n_{err}	p_{allf}	n_{allf}	p_{time}	$\varnothing time$
`MISQP/lag`	150	17	8	1.0	1262	1.0	0.07
`MISQP/soc`	153	16	6	1.6	2300	3.5	0.55
`MISQP/com`	156	13	6	1.2	1790	2.3	0.35
`MISQP/soc/minlp`	146	25	4	1.6	1629	224.5	24.17
`MISQP/com/minlp`	150	21	4	1.5	1506	228.5	23.74

Table 6.8: Comparison Mixed-Integer Problems vs. Relaxed Problems

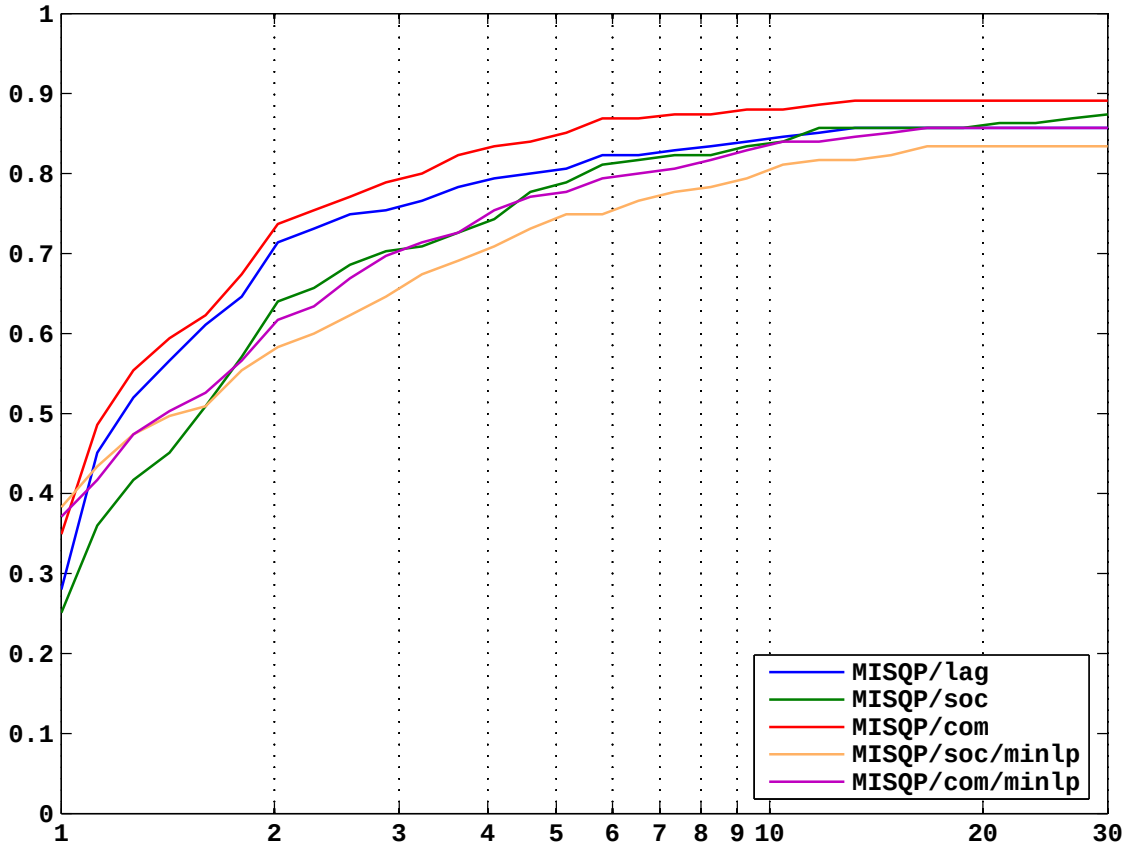

Figure 6.5: Performance Profiles – Mixed-Integer Problems vs. Relaxed Problems

quire more function evaluations just to obtain a solution to the root problem of the branch-and-bound tree than MISQP/com/minlp needs to solve the mixed-integer problem to optimality.

Expectedly, the mixed-integer codes require much more effort than the continuous counterparts with regard to the average time spent to solve the problems. Solving mixed-integer quadratic problems is more complex as the applied branch-and-bound method in the subproblem solver MIQL requires the solution of numerous continuous quadratic problems.

Figure 6.5 shows the corresponding performance profiles with respect to the total number of function evaluations. In almost 40% of the test problems MISQP/soc/minlp and MISQP/com/minlp are the best solvers subject to the criterion under consideration. The corresponding values for the continuous versions of MISQP are lower. Surprisingly, MISQP requires almost the same effort to solve continuous and mixed-integer problems.

6.5 Summary

The FORTRAN subroutine MISQP is tested on continuous and mixed-integer test problems. The numerical results show the efficiency of the proposed Algorithm 4.1 for continuous problems. The implementation is almost as efficient and reliable as the well-established software NLPQLP by Schittkowski [104]. Moreover, the coded Algorithm 4.1 outperforms an implementation of Algorithm 3.3, which applies second order correction steps as proposed by Yuan [130]. The results also indicate that fast local convergence is obtained in practice for Algorithm 4.1.

The aim of this thesis is to find techniques that lead to a reduction of the average number of function evaluations required by an implementation of Algorithm 5.2. The outcome is Algorithm 5.3 that avoids the calculation of second order correction steps. The test results show that the total number of function evaluations has been reduced about 7% when comparing the implementation of Algorithm 5.3 to the one of Algorithm 5.2. Furthermore, the calculation of second order correction steps can be avoided without getting less reliable. Since the numerical solution of the mixed-integer quadratic subproblems is time-consuming, the codes should be applied to problems where evaluating function values takes much more time than the internal calculations of the algorithms.

7 Conclusion and Outlook

New algorithms are presented that address nonlinear optimization problems. The considered problems are restricted and contain equality and inequality constraints. Moreover, mixed-integer nonlinear optimization problems are considered. The proposed algorithms approximate a solution to the considered problem iteratively by solving a sequence of quadratic subproblems. The idea of the mixed-integer algorithms is to extend the continuous SQP approach to mixed-integer optimization. The straightforward way is the substitution of the continuous quadratic subproblems by mixed-integer quadratic subproblems. Hence, all methods belong to the class of sequential quadratic programming methods. In order to stabilize the methods, the trust region approach is used. The size of the primal step computed as a trial step in each iteration is restricted by an additional constraint.

The main part of this thesis discuss the development of an algorithm that is applicable to continuous nonlinear problems, see Algorithm 4.1. The algorithm employs an augmented Lagrangian function as a merit function. Powell and Yuan [93] also use an augmented Lagrangian but their investigation is restricted to equality constrained problems. Niu and Yuan [76] apply the augmented Lagrangian to problems that also contain inequality constraints. In contrast to their algorithm the new trust region SQP algorithm does not transform the inequality constraints into equality constraints by introducing slack variables.

The occurrence of infeasible subproblems, that result from inconsistency of the linearized constraints of the problem or inconsistency induced by adding the trust region constraint, is handled by entering a feasibility restoration phase. Adding a feasibility restoration phase instead of permanently relaxing the quadratic subproblems is also applied by filter SQP methods, see Fletcher and Leyffer [42], and Fletcher, Leyffer, and Toint [43]. The advantage of entering a special restoration phase is that no additional penalty parameter in the subproblems is needed. The drawback of the suggested phase is that more subproblems have to be solved. Future work will be the investigation of a modified restoration phase.

Other trust region methods have to apply safety strategies in order to avoid the so-called Maratos effect – a situation where fast local convergence is prevented. In Yuan [130] a trust region algorithm is discussed that is applicable to equality and inequality constrained problems. The algorithm employs the L_∞-penalty function that can suffer the Maratos effect. Thus, additional second order correction steps are calculated as proposed by Fletcher [39] and Yuan [129]. This is a commonly used modification to avoid the Maratos effect. The aim of this work was the construction of a trust region algorithm that avoids these additional safeguards. As the local convergence analysis indicates the use of an augmented Lagrangian function as merit function leads to fast local convergence without the additional calculation of SOC steps.

Moreover, the theoretical analysis of the proposed algorithm shows that at least

one accumulation point of the generated sequence satisfies the Karush-Kuhn-Tucker optimality conditions. The assumptions made for the proof are used frequently by other authors. The extended MFCQ is applied to ensure the convergence to feasible points.

The new trust region SQP algorithm addressing continuous nonlinear problems, cf. Algorithm 4.1, is developed to find strategies that can be applied to improve the performance of the presented mixed-integer Algorithm 5.2 that is based on an algorithm by Yuan [130]. A modified algorithm is stated that locally employs the augmented Lagrangian merit function instead of the L_∞-penalty function. The corresponding Algorithm 5.3 does not calculate second order correction steps anymore.

It is well known that mixed-integer problems are extremely difficult to solve. Additionally, in highly complex technical simulation codes especially for engineering applications, it is often not possible to evaluate an objective or constraint function value for fractional values of an integer variable. In case integer variables are not relaxable some of the commonly used methods are not applicable. For example, branch-and-bound methods require a relaxation, see Borchers and Mitchell [11]. Outer approximation methods do not need relaxation but the convergence can be shown for convex problems only, see Fletcher and Leyffer [41]. Both mixed-integer algorithm introduced in this thesis are applicable in this situation.

The discussed algorithms are implemented as a `FORTRAN` subroutine called `MISQP`. Numerical results are presented which show the efficiency of the proposed algorithm for continuous problems. The code is almost as efficient as the well-established code `NLPQLP` by Schittkowski [104], where a line search strategy is used. It has also been shown that the implementation of the new algorithm outperforms an implementation of an algorithm proposed by Yuan [130], which applies second order correction steps. The numerical results also indicate that the implementation of the proposed continuous algorithm converges close to a solution with a fast rate what proves practically that the theoretical results hold.

The numerical results presented in Chapter 6 also lead to the conclusion that the goal of reducing the average number of function evaluations required by the new mixed-integer Algorithm 5.3 has been achieved. The total number of function evaluations has been reduced about 7% compared to the implementation of Algorithm 5.2. The tests also show that the numerical solution of the mixed-integer quadratic subproblems is costly. Thus, the approach is recommended in situations where the time spent in evaluating function values dominates the internal calculation time of the algorithm.

Although the implemented mixed-integer algorithms are extremely efficient and stop at a feasible solution after very few iterations, in most cases at the optimal solution, a convergence proof does not exist, even not for convex problems. Future work will be the extension of the mixed-integer algorithms to be able to prove convergence, at least for convex problems. A possible extension is already stated in Exler, Lehmann, and Schittkowski [36], where concepts of outer approximation methods are added to Algorithm 5.2.

A Program Documentation MISQP

MISQP is an implementation of the trust region SQP algorithms outlined before. MISQP is coded in FORTRAN. The arising mixed-integer quadratic programming subproblems are solved by the FORTRAN code MIQL of Lehmann et al. [65]. MIQL applies a branch-and-cut strategy to solve the mixed-integer quadratic problems. The underlying continuous quadratic programs are solved by QL, see Schittkowski [105], a FORTRAN code tracing back to Powell [89]. QL is an implementation of the primal-dual method of Goldfarb and Idnani [50]. An extended documentation of the one presented here can be found in Exler et al. [35].

MISQP requires user-provided model functions and gradients. The subroutine MISQP is called according to the following rules:

1. Choose starting values for the variables to be optimized, and store them in X, first the continuous, then the binary followed by the integer variables. Initialize ROPT, IOPT, and LOPT.
2. Compute objective and all constraint function values at X and store them in F and G, respectively.
3. Compute gradients of objective function and all constraints, and store them in DF and DG, respectively. The j-th row of DG contains the gradient of the j-th constraint, $j = 1,\ldots, m$. Only partial derivatives subject to the continuous variables and the integer variables specified in IDERIV need to be provided.
4. Set IFAIL=0 and execute MISQP.
5. If MISQP terminates with IFAIL=0, the internal stopping criteria are satisfied.
6. In case of IFAIL>0, an error occurred.
7. If MISQP returns with IFAIL=-1, compute objective function values and constraint values for all variables found in X, store them in F and G, and call MISQP again.
8. If MISQP terminates with IFAIL=-2, compute gradient values subject to variables stored in X, and store them in DF and DG. Only partial derivatives subject to the continuous variables and the integer variables specified in IDERIV need to be provided. Then call MISQP again.

Note that by default derivatives subject to integer variables are approximated by MISQP internally using one-sided or two-sided differences at neighboring grid points.

The usage of MISQP and the meaning of the parameters are described below. Default values, as far as applicable, are set in brackets. The option arrays contain parameters passed to the subprogram MIQL. They are also presented and highlighted by MIQL.

Usage

The subroutine MISQP can be called by any FORTRAN program by using the following command.

```
      CALL MISQP(  M       , ME      , MMAX    , N     , NBIN   ,
     /             NINT    , X       , F       , G     , DF     ,
     /             DG      , XL      , XU      , ACC   , MAXIT  ,
     /             MAXCUT  , MAXNDE  , IPRINT  , IOUT  , IFAIL  ,
     /             IDERIV  , ROPT    , IOPT    , LOPT  , RW     ,
     /             LRW     , IW      , LIW     , LW    , LLW    )
```

Parameter Definition

M	Input parameter defining the total number of constraints.
ME	Input parameter defining the number of equality constraints.
MMAX	Row dimension of array DG containing Jacobian of constraints. MMAX must be at least one and greater or equal to M.
N	Input parameter defining the total number of optimization variables, continuous, binary and integer ones.
NBIN	Input parameter defining the number of binary optimization variables, must be less than or equal to N.
NINT	Input parameter for the number of non-binary integer variables, must be less than or equal to N.
X(N)	Input and output vector containing starting point at first call. On return, X is replaced by the current iterate. The first N-NBIN-NINT positions are assigned to continuous variables, the subsequent NBIN coefficients to binary variables, and the remaining NINT positions to non-boolean integer variables.
F	Input parameter containing the objective function value at the current iterate X.
G(MMAX)	Input vector containing the values of the constraints at the current iterate X, first ME equality constraints, then M-ME inequality constraints.
DF(N)	Input vector containing the values of the gradient of the objective function at the current iterate X. It is sufficient to determine the gradients subject to the continuous variables and variables specified in IDERIV. See also description of IDERIV below.

DG(MMAX,N)	Input matrix containing the values of the Jacobian of the constraints at the current iterate X, first for the ME equality constraints, then for M-ME inequality constraints. In the driving program, the row dimension of DG must be equal to MMAX. It is sufficient to determine the gradients subject to the continuous variables and variables specified in IDERIV. See also description of IDERIV below.
XL(N), XU(N)	On input, the one-dimensional arrays XL and XU must contain the upper and lower bounds of the variables, first for the continuous, then for the binary and subsequently for the integer variables.
ACC	Input parameter defining the tolerance for detecting integer values and for termination. If ACC is less than machine precision, then ACC is internally set to machine precision multiplied by 1,000 (1.0E-6).
MAXIT	Maximum number of iterations (100).
MAXCUT	To enable cut generation of the QP solver, row dimension MMAX is enlarged internally by MAXCUT (500).
MAXNDE	Maximum number of branch-and-bound nodes for solving MIQP (10,000).
IPRINT	Specification of the desired output level: 0 No output of the program. 1 Only a final convergence analysis is given. 2 One line of intermediate results is printed in each iteration. 3 More detailed information is printed for each iteration. 4 In addition, some messages of the QP solver are displayed.
IOUT	Integer indicating the desired output unit number, i.e., all write-statements start with WRITE(IOUT,....
IFAIL	The parameter shows the reason for terminating a solution process. Initially IFAIL must be set to zero. On return IFAIL could contain the following values: -2 Compute new gradient values for continuous variables and variables specified in IDERIV in DF and DG. See description of DF, DG and IDERIV. -1 Compute new function values in F and G, see above. 0 Optimality conditions satisfied. 1 Termination after MAXIT iterations. 2 Trust region radius lower than ACCQP. 3 Penalty parameter SIGMA tends to infinity. 4 Termination at infeasible iterate. 5 Termination with zero trust region for integer variables. 6 Length of a working array is too short. 7 False dimensions, e.g., M>MMAX.

	8 Inconsistent box constraints for an integer variable. 11 The continuous quadratic solver QL could not solve a quadratic program after a maximal number of 40·(N+M) iterations. 12 The termination accuracy is insufficient for the continuous quadratic solver QL to satisfy the convergence criterion. 13 The continuous quadratic solver QL terminated due to an internal inconsistency, division by zero. 14 Numerical instabilities prevent successful termination of continuous quadratic solver QL. >90 QP solver terminated with an error message IFQL, IFAIL= IFQL+100.
IDERIV (NBIN+NINT)	Logical input array specifying integer variables whose derivatives are provided by the user and stored in DG and DF. Meaning of the value of IDERIV(I): .TRUE. Column NCONT+I of DG and position NCONT+I of DF are set by the user. .FALSE. Column NCONT+I of DG and position NCONT+I of DF are set by MISQP by evaluating functions at neighboring grid points. Here NCONT=N-NINT-NBIN.
ROPT(60)	Double precision option array, to be initialized with -1.0 for default parameter setting:
▷ MISQP	
ROPT(1)	Termination tolerance of the QP solver for several tests, e.g., whether optimality conditions are satisfied or whether a number is considered as zero or not (ACCQP, 1.0E-12).
ROPT(2)	Factor for increasing a penalty parameter, must be greater than one (RPEN, 10). Ignored in case MFV=0.
ROPT(3)	Factor for increasing the internal descent parameter ZETA, see ROPT(5). ROPT(3) must be less than one (ZETDEC, 0.1). Ignored in case MFV=0.
ROPT(4)	Initial penalty parameter σ_0 (SIGMA, 1,000).
ROPT(5)	Initial scaling parameter ζ_0 (ZETA, 0.05). Ignored in case MFV=0.
ROPT(6)	Initial continuous trust region radius Δ_0^c or Δ_0, respectively, greater than zero (TRUSTC, 50).
ROPT(7)	Initial integer trust region radius Δ_0^i, not less than one (TRUSTI, 50).

`IOPT(60)` Integer option array, to be initialized with `-1` for default parameter setting:

▷ `MISQP`

`IOPT(1)` Function values are scaled internally:

- `1` Subject to their absolute values, if greater than one.
- `2` As above, but new function values evaluated at lower bounds of integer variables, starting point not changed.

`IOPT(2)` Maximum number of successive iterations which are considered for the non-monotone trust region algorithm, must be less than `100` (`NONMON`, `10`). Ignored in case `MFV=0`.

`IOPT(3)` Print level of the subproblem solver `MIQL` (`IPRQP`, `2`).

`IOPT(4)` Output for number of gradient evaluations.

`IOPT(5)` Output for number of function evaluations.

`IOPT(6)` Maximum number of successive restarts without improving solution. Setting might lead to better results, but increases the number of function evaluations (`MRS`, `2`).

`IOPT(7)` Merit function (`MFV`):

- `0` Augmented Lagrangian (only if `N=NCONT`).
- `1` L_∞-penalty function.
- `2` Combination of version of `0` and `1`.

Default value is `0` if `N=NCONT`, and `2` otherwise.

▷ `MIQL`

`IOPT(41)` Branching rule (`IBR`, `1`):

- `1` Maximal fractional branching.
- `2` Minimal fractional branching.

`IOPT(42)` Node selection strategy (`INS`, `3`):

- `1` Best of all (large search trees).
- `2` Best of two (warm starts, less memory for search tree).
- `3` Depth first (good warm starts, less memory, many QPs solved).

`IOPT(44)` Maximal number of successive warm starts, to avoid numerical instabilities (`WSTART`, `100`).

`IOPT(45)` Calculate improved bounds if *best-of-all* selection strategy is used (`IMPB`, `0`).

`IOPT(46)` Select direction for *depth-first* according to value of Lagrange function (`DFDIR`, `0`).

IOPT(47)	Cholesky decomposition mode (CHOLM, 1): 0 Calculate Cholesky decomposition once and reuse it. 1 Calculate new decomposition if warm start is not activated.
IOPT(48)	Control the cutting process (CUT, 0): 0 No cuts. 1 Disjunctive cuts only. 2 Complemented mixed-integer rounding (CMIR) cuts only. 3 Both disjunctive and CMIR cuts.
IOPT(49)	Maximal number of cycles for disjunctive cuts (MAXDC, 1).
IOPT(50)	Maximal number of cycles for CMIR cuts (MAXCM, 1).
IOPT(51)	Primal heuristic mode (PHM, 0): 0 No primal heuristics. 1 Nearest integer. 2 Feasibility pump.
LOPT(60)	Logical option array, to be initialized with .TRUE. for default parameter setting:
▷ MISQP	
LOPT(2)	Internal scaling of continuous variables (SCALE, .TRUE.).
LOPT(3)	Activates modification of Hessian approximation to get more accurate search directions. Calculation time is increased in case of a large number of integer variables (BMOD, .TRUE.).
▷ MIQL	
LOPT(44)	Transformation of MIQP to positive orthant (LPOS).
RW(LRW)	Real working array of length LRW.
LRW	Input parameter defining the length of RW, must be at least 7·N·N/2 + MMAX0·N + 102·N + 37·MMAX0 + 3·MAXNDE + 3·M·M/2 + 4·M·N + 500, where MMAX0=M+ME+MAX(NINT+NBIN,MAXCUT)+20.
IW(LIW)	Integer working array of length LIW.
LIW	Input parameter defining the length of IW, must be at least 14·N + 5·MMAX0 + 6·MAXNDE + 150.
LW(LLW)	Logical working array of length LLW.
LLW	Input parameter defining the length of LW, must be at least 4·N + MMAX0 + 150.

B Priority Theory

The priority theory traces back to Saaty [96]. The idea was applied by Lootsma [70] in connection with comparing optimization software. Schittkowski [98] and Hock and Schittkowski [62] also used the methodology to compare the performance of optimization codes. The basic idea of the priority theory is summarized in the following.

It is assumed that N codes C_i, $i = 1,\ldots, N$, are compared on a set of M test problems TP_j, $j = 1, \ldots, M$. Let S_i denote the set of test problems that are solved successfully by code C_i, $i = 1, \ldots, N$, i.e.,

$$S_i := \{j \mid TP_j \text{ can be solved successfully by } C_i, 1 \le j \le M\} \ . \tag{B.1}$$

The N codes are compared pairwise with respect to a chosen performance criterion, e.g., calculation time or number of function evaluations. Let t_{ij} be the relevant result obtained by code C_i on test problem TP_j, $t_{ij} > 0$. Then a reciprocal matrix

$$R := (r_{ik})_{i,k=1,\ldots,N} \tag{B.2}$$

is defined, where the elements of R are calculated by

$$r_{ik} := \frac{\sum\limits_{j \in S_i \cap S_k} t_{ij}}{\sum\limits_{j \in S_i \cap S_k} t_{kj}} \ , \tag{B.3}$$

for $i, k = 1, \ldots, N$. Note that only test problems TP_j are considered which are solved successfully by both codes C_i and C_k, i.e., $j \in S_i \cap S_k$. For all elements of R the condition

$$r_{ik} = r_{ki}^{-1} > 0 \tag{B.4}$$

holds. Matrix R is considered to be an approximation to the matrix

$$P := \left(\frac{w_i}{w_k}\right)_{i,k=1,\ldots,N} \ , \tag{B.5}$$

where the entries $w_1, \ldots, w_N$ are the true mean values of the stochastic variables that are considered, i.e., the expectation value of the performance criterion for code C_i. For simplicity, it is assumed that

$$\sum_{i=1}^{N} w_i = 1$$

and $w := (w_1, \ldots, w_N)^T$. Then P is a rank-one matrix with

$$Pw = Nw \ , \tag{B.6}$$

code	TP_1	TP_2	TP_3	TP_4	TP_5	*mean*
C_1	–	5.2	1.3	4.0	7.0	4.4
C_2	0.3	–	1.5	–	8.2	3.3
C_3	3.0	11.2	–	–	12.2	8.8

Table B.1: Example of Calculation Times for 5 Test Problems and 3 Codes

i.e., N is the only positive eigenvalue of P and w is the uniquely determined normalized eigenvector with positive elements.

According to a theorem of Perron-Frobenius, see for example Bellman [4], the largest eigenvalue of R is real and positive, and there exists a uniquely determined eigenvector with positive elements.

Since it is assumed that matrix R approximates P, the performance evaluation consists of the following steps. First, the matrix R is calculated, see (B.3) and (B.2). Then the maximum eigenvalue of R with positive eigenvector $\overline{w}$ is computed, where $\overline{w}$ is considered to be a suitable approximation to w, cf. (B.6). In the end, the entries of the eigenvector are scaled so that the smallest coefficient becomes one. Thus, a performance value of one corresponds to the best code with respect to the criterion under consideration.

An example is given to explain the performance evaluation, see also Exler et al. [36]. Some results obtained by a fictitious test run are shown in Table B.1, where '–' indicates that the code cannot solve the corresponding test problem successfully. The mean values of calculation times shown in Table B.1 are taken over all successful test runs.

Applying the priority theory as described above leads to the matrix

$$R = \begin{pmatrix} 1.0 & 0.86 & 0.52 \\ 1.17 & 1.0 & 0.56 \\ 1.92 & 1.79 & 1.0 \end{pmatrix} ,$$

and the eigenvector corresponding to the largest eigenvalue of R is

$$w = (1.0, 1.1, 2.0)^T \ .$$

The resulting w can be interpreted as follows. C_1 is slightly faster than C_2, whereas C_3 is about two times slower than C_1 and, approximately, also twice slower than C_2. Taking the mean values given in Table B.1 leads to a different conclusion. The results obtained by priority theory can be seen as more meaningful, since simple mean values over successful test runs might penalize the more robust codes. A robust code that solves difficult and time consuming problems might be handicapped in case mean values are taken as performance criterion.

Bibliography

[1] Antelo, L.T., Exler, O., Banga, J.R., and Alonso, A.A. (2008): *Optimal tuning of thermodynamic-based decentralized PI control loops: Application to the Tennessee Eastman Process*. AIChE Journal, **54**(11):2904–2924.

[2] Armijo, L. (1966): *Minimization of functions having Lipschitz continuous first partial derivatives*. Pacific Journal of Mathematics, **16**:1–3.

[3] Audet, C. and Dennis, J.E. (2001): *Pattern search algorithm for mixed variable programming*. SIAM Journal on Optimization, **11**:573–594.

[4] Bellman, R.E. (1997): *Introduction to Matrix Analysis*, *Classics in Applied Mathematics*, vol. 19. SIAM, Philadelphia, 2nd edn.

[5] Belotti, P. (2009): *Couenne: a user's manual*. Technical Report, Department of Mathematical Sciences, Clemson University, Clemson.

[6] Belotti, P., Lee, J., Liberti, L., Margot, F., and Wächter, A. (2009): *Branching and bounds tightening techniques for non-convex MINLP*. Optimization Methods and Software, **24**(4-5):597–634.

[7] Boggs, P.T. and Tolle, J.W. (1995): *Sequential Quadratic Programming*. Acta Numerica, **4**:1–51.

[8] Boggs, P.T., Tolle, J.W., and Wang, P. (1982): *On the local convergence of quasi-Newton methods for constrained optimization*. SIAM Journal on Control and Optimization, **20**:161–171.

[9] Bonami, P., Biegler, L.T., Conn, A.R., Cornuéjols, G., Grossmann, I.E., Laird, C.D., Lee, J., Lodi, A., Margot, F., Sawaya, N., and Wächter, A. (2005): *An algorithmic framework for convex mixed integer nonlinear programs*. Technical Report RC23771, IBM Research Division, Pittsburgh.

[10] Bonami, P., Kilinc, M., and Linderoth, J.T. (2009): *Algorithms and software for convex mixed-integer nonlinear programs*. Technical Report 1664, Computer Science Department, University of Wisconsin, Madison.

[11] Borchers, B. and Mitchell, J.E. (1994): *An improved branch and bound algorithm for mixed integer nonlinear programming*. Computers and Operations Research, **21**(4):359–367.

[12] Bünner, M.J., Schittkowski, K., and van de Braak, G. (2004): *Optimal design of electronic components by mixed-integer nonlinear programming*. Optimization and Engineering, **5**:271–294.

[13] Bussieck, M., Drud, A.S., and Meeraus, A. (2007): *MINLPLib - A collection of test models for mixed-integer nonlinear programming*. Technical Report, GAMS Development Corp., Washington D.C.

[14] Bussieck, M. and Vigerske, S. (2010): *MINLP solver software.* In: J.J. Cochran, L.A. Cox, P. Keskinocak, J.P. Kharoufeh, and J.C. Smith (eds.), *Wiley Encyclopedia of Operations Research and Management Science.* John Wiley & Sons, Hoboken.

[15] Byrd, R.H., Gilbert, J.C., and Nocedal, J. (2000): *A trust region method based on interior point techniques for nonlinear programming.* Mathematical Programming, **89**:149–185.

[16] Byrd, R.H., Schnabel, R.B., and Shultz, G.A. (1987): *A trust region algorithm for nonlinearly constrained optimization.* SIAM Journal on Numerical Analysis, **24**:1152–1170.

[17] Celis, M., Dennis, J.E., and Tapia, R.A. (1985): *A trust region algorithm for nonlinear equality constrained optimization.* In: P.T. Boggs, R.H. Byrd, and R.B. Schnabel (eds.), *Numerical optimization 1984*, 71–82. SIAM, Philadelphia.

[18] Chamberlain, R., Lemarechal, C., Pedersen, H., and Powell, M.J.D. (1982): *The watchdog technique for forcing convergence in algorithms for constrained optimization.* Mathematical Programming Study, **16**:1–17.

[19] Chen, Z. and Zhang, X. (2004): *A nonmonotone trust-region algorithm with nonmonotone penalty parameters for constrained optimization.* Journal of Computational and Applied Mathematics, **172**:7–39.

[20] Coleman, T.F. and Li, Y. (2000): *A trust region and affine scaling interior point method for nonconvex minimization with linear inequality constraints.* Mathematicl Programming, **88**:1–31.

[21] Conn, A.R., Gould, N.I.M., and Toint, P.L. (2000): *Trust-Region Methods.* MPS-SIAM series on optimization. SIAM, Philadelphia.

[22] Deng, N.Y., Xiao, Y., and Zhou, F.J. (1993): *Nonmonotonic trust region algorithm.* Journal of Optimization Theory and Applications, **76**(2):259–285.

[23] Dennis, J.E. (1978): *A brief introduction to quasi-Newton methods.* In: G.H. Golub and J. Oliger (eds.), *Numerical Analysis*, 19–52. American Mathematical Society, Providence.

[24] Dennis, J.E., El-Alem, M., and Maciel, M.C. (1997): *A global convergence theory for general trust-region-based algorithms for equality constrained optimization.* SIAM Journal on Optimization, **7**(1):177–207.

[25] Dennis, J.E., Heinkenschloss, M., and Vicente, L. (1998): *Trust-region interior-point SQP algorithms for a class of nonlinear programming problems.* SIAM Journal on Control and Optimization, **36**:1750–1794.

[26] Dennis, J.E. and Schnabel, R.B. (1996): *Numerical methods for unconstrained optimization and nonlinear equations*, *Classics in Applied Mathematics*, vol. 16. SIAM, Philadelphia.

[27] Dolan, E.D. and Moré, J.J. (2002): *Benchmarking optimization software with performance profiles.* Mathematical Programming, **91**(2):201–213.

[28] Duran, M.A. and Grossmann, I.E. (1986): *An outer-approximation algorithm for a class of mixed-integer nonlinear programs.* Mathematical Programming, **36**:307–339.

[29] Edgar, T.F., Himmelblau, D.M., and Lasdon, L.S. (2001): *Optimization of Chemical Processes.* Chemical engineering series. McGraw-Hill, Boston, 2nd edn.

[30] El-Alem, M. (1988): *A global convergence theory for a class of trust region algorithms for constrained optimization.* Ph.D. thesis, Rice University, Houston.

[31] El-Alem, M. (1991): *A global convergence theory for the Celis-Dennis-Tapia Trust Region Algorithm for Constrained Optimization.* SIAM Journal on Numerical Analysis, **28**(1):266–290.

[32] El-Alem, M. and El-Sobky, B. (1997): *A new trust-region algorithm for general nonlinear programming.* Technical Report TR97-25, Department of Mathematics, Faculty of Science, Alexandria University, Alexandria.

[33] Exler, O. (2004): *Gemischt-ganzzahlige nichtlineare Programmierung mit SQP-Trust-Region-Verfahren.* Diplomarbeit, Mathematisches Institut, Universität Bayreuth.

[34] Exler, O., Antelo, L.T., Egea, J.A., Alonso, A.A., and Banga, J.R. (2008): *A Tabu search-based algorithm for mixed-integer nonlinear problems and its application to integrated process and control system design.* Computers and Chemical Engineering, **32**(8):1877–1891.

[35] Exler, O., Lehmann, T., and Schittkowski, K. (2011): *MISQP: A Fortran subroutine of a trust region SQP algorithm for mixed-integer nonlinear programming - user's guide.* Technical Report, Department of Computer Science, University of Bayreuth, Bayreuth.

[36] Exler, O., Lehmann, T., and Schittkowski, K. (2012): *A comparative study of SQP-type algorithms for nonlinear and nonconvex mixed-integer optimization.* Mathematical Programming Computation, **4**(4):383–412.

[37] Exler, O. and Schittkowski, K. (2007): *A trust region SQP algorithm for mixed-integer nonlinear programming.* Optimization Letters, **1**(3):269–280.

[38] Fletcher, R. (1982): *A model algorithm for composite non-differentiable optimization problems*. Mathematical Programming, **17**:67–76.

[39] Fletcher, R. (1982): *Second order corrections for non-differentiable optimization*. In: G.A. Watson (ed.), *Numerical Analysis*, *Lecture notes in mathematics*, vol. 912, 85–115. Springer, Berlin.

[40] Fletcher, R. (2000): *Practical Methods of Optimization*. John Wiley & Sons, Chichester, 2nd edn.

[41] Fletcher, R. and Leyffer, S. (1994): *Solving mixed integer nonlinear programs by outer approximation*. Mathematical Programming, **66**(1-3):327–349.

[42] Fletcher, R. and Leyffer, S. (2002): *Nonlinear programming without a penalty function*. Mathematical Programming, **91**:239–269.

[43] Fletcher, R., Leyffer, S., and Toint, P.L. (2002): *On the global convergence of a filter-SQP algorithm*. SIAM Journal on Optimization, **13**(1):44–59.

[44] Floudas, C.A. (1995): *Nonlinear and Mixed-Integer Optimization: Fundamentals and Applications*. Topics in chemical engineering. Oxford University Press, New York.

[45] Fukushima, M. (1986): *A successive quadratic programming algorithm with global and superlinear convergence properties*. Mathematical Programming, **35**:253–264.

[46] Gauvin, J. (1977): *A necessary and sufficient regularity condition to have bounded multipliers in nonconvex programming*. Mathematical Programming, **12**:136–138.

[47] Geiger, C. and Kanzow, C. (2002): *Theorie und Numerik restringierter Optimierungsaufgaben*. Springer, Berlin.

[48] Gill, P.E., Murray, W., Saunders, M.A., and Wright, M.H. (1986): *Some theoretical properties of an augmented Lagrangian merit function*. Technical Report SOL 86-6, Department of Operations Research, Stanford University, Stanford.

[49] Gill, P.E., Murray, W., and Wright, M.H. (1981): *Practical Optimization*. Academic Press, London.

[50] Goldfarb, D. and Idnani, A. (1983): *A numerically stable dual method for solving strictly convex quadratic programs*. Mathematical Programming, **27**(1):1–33.

[51] Goldfeldt, S.M., Quandt, R.E., and Trotter, H.F. (1966): *Maximization by quadratic hill-climbing*. Econometrica, **34**:541–551.

[52] Gould, N.I.M. and Toint, P. (2000): *SQP methods for large-scale nonlinear programming*. In: M.J.D. Powell and S. Scholtes (eds.), *System Modelling and Optimization*, IFIP / International Federation for Information Processing, 149–178. Kluwer Academic Publishers, Boston.

[53] Gould, N.I.M. and Toint, P.L. (2006): *Global convergence of a non-monotone trust region filter algorithm for nonlinear programming*. In: W. Hager, S. Huang, P. Pardalos, and O. Prokopyev (eds.), *Multiscale Optimization Methods and Applications*, 125–150. Springer, New York.

[54] Griffith, R.E. and Stewart, R.A. (1961): *A nonlinear programming technique for the optimization of continuous processing systems*. Management Science, **7**:379–392.

[55] Grippo, L., Lampariello, F., and Lucidi, S. (1986): *A nonmonotone line search technique for Newtons's method*. SIAM Journal on Numerical Analysis, **23**:707–716.

[56] Grossmann, I.E. and Kravanja, Z. (1997): *Mixed-integer nonlinear programming: A survey of algorithms and applications*. In: L.T. Biegler, T.F. Coleman, A.R. Conn, and F.N. Santosa (eds.), *Large-scale Optimization with Applications*, *The IMA volumes in mathematics and its applications*, vol. 93, 73–100. Springer, New York.

[57] Gupta, O.K. and Ravindran, V. (1985): *Branch and bound experiments in convex nonlinear integer programming*. Management Science, **31**:1533–1546.

[58] Han, S.P. (1976): *Superlinearly convergent variable metric algorithms for general nonlinear programming problems*. Mathematical Programming, **11**:263–282.

[59] Han, S.P. (1977): *A globally convergent method for nonlinear programming*. Journal of Optimization Theory and Applications, **22**(3):297–309.

[60] Hartwanger, C., Schittkowski, K., and Wolf, H. (2000): *Computer aided design of horn radiators for satellite communication by least squares optimization*. Engineering Optimization, **33**(2):221–244.

[61] Hock, W. and Schittkowski, K. (1981): *Test examples for nonlinear programming codes*, *Lecture notes in economics and mathematical systems*, vol. 187. Springer, Berlin.

[62] Hock, W. and Schittkowski, K. (1983): *A comparative performance evaluation of 27 nonlinear programming codes*. Computing, **30**(4):335–358.

[63] Jiang, H., Fukushima, M., Qi, L., and Sun, D. (1998): *A trust region method for solving generalized complementarity problems*. SIAM Journal on Optimization, **8**(1):140–157.

[64] Kanzow, C. and Zupke, M. (1999): *Inexact trust-region methods for nonlinear complementarity problems.* In: M. Fukushima and L. Qi (eds.), *Reformulation - Nonsmooth, Piecewise Smooth, Semismooth and Smoothing Methods*, 211–233. Kluwer Academic Publishers, Dordrecht.

[65] Lehmann, T., Schittkowski, K., and Spickenreuther, T. (2009): *MIQL: A Fortran subroutine for convex mixed-integer quadratic programming by branch-and-bound - user's guide.* Technical Report, Department of Computer Science, University of Bayreuth, Bayreuth.

[66] Levenberg, K. (1944): *A method for the solution of certain problems in least squares.* Quarterly Journal on Applied Mathematics, **2**:164–168.

[67] Leyffer, S. (1993): *Deterministic Methods for Mixed Integer Nonlinear Programming.* Ph.D. thesis, University of Dundee, Dundee, Scotland, UK.

[68] Leyffer, S. (2001): *Integrating SQP and branch-and-bound for mixed integer nonlinear programming.* Computational Optimization and Application, **18**:295–309.

[69] Li, H.L. and Chou, C.T. (1994): *A global approach for nonlinear mixed discrete programming in design optimization.* Engineering Optimization, **22**:109–122.

[70] Lootsma, F.A. (1982): *Performance evaluation of nonlinear optimization methods via multi-criteria analysis and via linear model analysis.* In: M.J.D. Powell (ed.), *Nonlinear Optimization*, vol. 82, 419–453. Academic Press, London.

[71] Maratos, N. (1978): *Exact Penalty Function Algorithms for Finite Dimensional and Control Optimization Problems.* Ph.D. thesis, Imperial College, London.

[72] Marquardt, D. (1963): *An algorithm for least-squares estimation of nonlinear parameters.* SIAM Journal on Applied Mathematics, **11**:431–441.

[73] Mayne, D. and Polak, E. (1982): *A superlinearly convergent algorithm for constrained optimization problems.* Mathematical Programming Study, **16**:45–61.

[74] Moré, J.J. (1983): *Recent developments in algorithms and software for trust region methods.* In: A. Bachem, M. Grötschel, and B. Korte (eds.), *Mathematical Programming: The State of the Art*, 258–287. Springer, Berlin.

[75] Morrison, D.D. (1960): *Methods for nonlinear least squares problems and convergence proofs.* In: J. Lorell and F. Yagi (eds.), *Proceedings of the Seminar on Tracking Programs and Orbit Determination*, 1–9. Jet Propulsion Laboratory, Pasadena.

[76] Niu, L. and Yuan, Y. (2010): *A new trust-region algorithm for nonlinear constrained optimization.* Journal of Computational Mathematics, **28**(1):72–86.

[77] Nocedal, J. and Wright, S.J. (1999): *Numerical optimization*. Springer Series in Operations Research. Springer, New York.

[78] Nowak, I. (2005): *Relaxation and Decomposition Methods for Mixed Integer Nonlinear Programming*, *International Series of Numerical Mathematics*, vol. 152. Birkhäuser, Basel.

[79] Nowak, I., Alperin, H., and Vigerske, S. (2003): *LaGO – An object oriented library for solving MINLPs*. In: C. Bliek, C. Jermann, and A. Neumaier (eds.), *Global Optimization and Constraint Satisfaction*, *Lecture Notes in Computer Science*, vol. 2861, 32–42. Springer, Berlin.

[80] Omojokun, E.O. (1989): *Trust Region Algorithms for Optimization with Nonlinear Equality and Inequality Constraints*. Ph.D. thesis, University of Colorado, Boulder.

[81] Ortega, J.M. and Rheinboldt, W.C. (1970): *Iterative Solution of Nonlinear Equations in Several Variables*. Academic Press, New York.

[82] Papalambros, P.Y. and Wilde, D.J. (2000): *Principles of Optimal Design: Modeling and Computation*. Cambridge Univ. Press, Cambridge, 2nd edn.

[83] Powell, M.J.D. (1970): *A Fortran subroutine for solving systems of nonlinear algebraic equations*. In: P. Rabinowitz (ed.), *Numerical Methods for Nonlinear Algebraic Equations*, 115–161. Gordon and Breach, London.

[84] Powell, M.J.D. (1970): *A hybrid method for nonlinear equations*. In: P. Rabinowitz (ed.), *Numerical Methods for Nonlinear Algebraic Equations*, 87–114. Gordon and Breach, London.

[85] Powell, M.J.D. (1970): *A new algorithm for unconstrained optimization*. In: J.B. Rosen, O.L. Mangasarian, and K. Ritter (eds.), *Nonlinear Programming*, 31–65. Academic Press, London.

[86] Powell, M.J.D. (1978): *A fast algorithm for nonlinearly constrained optimization calculations*. In: G.A. Watson (ed.), *Lecture Notes in Mathematics*, 144–157. Springer, Berlin.

[87] Powell, M.J.D. (1983): *On the quadratic programming algorithm of Goldfarb and Idnani*. Technical Report DAMTP/1983/NA19, University of Cambridge, Cambridge.

[88] Powell, M.J.D. (1983): *Variable metric methods for constrained optimization*. In: Bachem A., M. Grötschel, and B. Korte (eds.), *Mathematical Programming: The State of the Art*, 288–311. Springer, Berlin.

[89] Powell, M.J.D. (1983): *ZQPCVX, A Fortran subroutine for convex quadratic programming.* Technical Report DAMTP/1983/NA17, University of Cambridge, Cambridge.

[90] Powell, M.J.D. (1984): *On the global convergence of trust region algorithms for unconstrained minimization.* Mathematical Programming, **29**(3):297–303.

[91] Powell, M.J.D. (1986): *Convergence properties of algorithms for nonlinear optimization.* SIAM Review, **28**(4):487–500.

[92] Powell, M.J.D. and Yuan, Y. (1986): *A recursive quadratic programming algorithm that uses differentiable exact penalty functions.* Mathematical Programming, **35**:265–278.

[93] Powell, M.J.D. and Yuan, Y. (1991): *A trust region algorithm for equality constrained optimization.* Mathematical Programming, **49**:189–211.

[94] Rockafellar, R.T. (1973): *The multiplier method of Hestenes and Powell applied to convex programming.* Journal of Optimization Theory and Applications, **12**:555–562.

[95] Rockafellar, R.T. (1974): *Augmented Lagrangian multiplier functions and duality in nonconvex programming.* SIAM Journal of Control, **12**:268–285.

[96] Saaty, T.L. (1977): *A scaling method for priorities in hierarchical structures.* Journal of Mathematical Psychology, **15**:234–281.

[97] Sahinidis, N. and Tawarmalani, M. (2010). *Baron 9.0.4: Global optimization of mixed-integer nonlinear programs, user's manual.* Available at `http://www.gams.com/dd/docs/solvers/baron.pdf`.

[98] Schittkowski, K. (1980): *Nonlinear Programming Codes: Information, Tests, Performance*, *Lecture notes in economics and mathematical systems*, vol. 183. Springer, Berlin.

[99] Schittkowski, K. (1981): *The nonlinear programming method of Wilson, Han, and Powell with an augmented lagrangian type line search function - Part 1: Convergence analysis.* Numerische Mathematik, **38**:83–114.

[100] Schittkowski, K. (1983): *On the convergence of a sequential quadratic programming method with an augmented Lagrangian search direction.* Optimization, **14**:197–216.

[101] Schittkowski, K. (1986): *NLPQL: A Fortran subroutine solving constrained nonlinear programming problems.* Annals of Operations Research, **5**(2):485–500.

[102] Schittkowski, K. (1987): *More test examples for nonlinear programming codes*, *Lecture notes in economics and mathematical systems*, vol. 282. Springer, Berlin.

[103] Schittkowski, K. (1999): *Mathematische Grundlagen von Optimierungsverfahren*. Technical Report, Department of Mathematics, University of Bayreuth, Bayreuth.

[104] Schittkowski, K. (2010): *NLPQLP: A Fortran implementation of a sequential quadratic programming algorithm with distributed and non-monotone line search - user's guide, version 3.1*. Technical Report, Department of Computer Science, University of Bayreuth, Bayreuth.

[105] Schittkowski, K. (2010): *QL: A Fortran code for convex quadratic programming - user's guide, version 3.0*. Technical Report, Department of Computer Science, University of Bayreuth, Bayreuth.

[106] Schittkowski, K. (2011): *A collection of 175 test problems for nonlinear mixed-integer programming in Fortran - user's guide*. Technical Report, Department of Computer Science, University of Bayreuth, Bayreuth.

[107] Schittkowski, K. and Yuan, Y. (2010): *Sequential quadratic programming methods*. In: J.J. Cochran, L.A. Cox, P. Keskinocak, J.P. Kharoufeh, and J.C. Smith (eds.), *Wiley Encyclopedia of Operations Research and Management Science*. John Wiley & Sons, Hoboken.

[108] Sendín, J.O.H., Exler, O., and Banga, J.R. (2010): *Multi-objective mixed integer strategy for the optimisation of biological networks*. IET Systems Biology, **4**(3):236–248.

[109] Sorensen, D.C. (1982): *Newton's method with a model trust region modification*. SIAM Journal on Numerical Analysis, **19**(2):409–426.

[110] Spellucci, P. (1993): *Numerische Verfahren der nichtlinearen Optimierung*. Internationale Schriftenreihe zur numerischen Mathematik Lehrbuch. Birkhäuser, Basel.

[111] Spellucci, P. (1998): *A new technique for inconsistent QP problems in the SQP method*. Mathematical Methods of Operations Research, **47**:355–400.

[112] Stoer, J. (1985): *Foundations of recursive quadratic programming methods for solving nonlinear programs*. In: K. Schittkowski (ed.), *Computational Mathematical Programming*, *NATO ASI Series, Series F, Computer and Systems Sciences*, vol. 15, 165–208. Springer, Berlin.

[113] Sun, W. and Yuan, Y. (2006): *Optimization Theory and Methods: Nonlinear Programming*. Springer, Boston.

[114] Tawarmalani, M. and Sahinidis, N. (2002): *Convexification and global optimization in continous and mixed-integer nonlinear programming: Theory, algorithms, software, and applications*, *Nonconvex optimization and its applications*, vol. 65. Kluwer Academic Publishers, Dordrecht.

[115] Tawarmalani, M. and Sahinidis, N. (2004): *Global optimization of mixed-integer nonlinear programs: A theoretical and computational study.* Mathematical Programming, Series A, **99**:563–591.

[116] Tawarmalani, M. and Sahinidis, N. (2005): *A polyhedral branch-and-cut approach to global optimization.* Mathematical Programming, **103**:225–249.

[117] Thomas, I. and Kröner, A. (2006): *Mixed-integer optimization of distillation column tray positions in industrial practice.* In: W. Marquardt and C. Pantelides (eds.), *16th European Symposium on Computer Aided Process Engineering and 9th International Symposium on Process Systems Engineering*, *Computer-aided Chemical Engineering*, vol. 21, 1015–1020. Elsevier, Amsterdam.

[118] Toint, P.L. (1981): *Towards an efficient sparsity exploiting Newton method for minimization.* In: I.S. Duff (ed.), *Sparse Matrices and Their Uses*, 57–88. Academic Press, London.

[119] Toint, P.L. (1996): *An assessment of nonmontone line search techniques for unconstrained optimization.* SIAM Journal on Scientific Computing, **17**:725–739.

[120] Toint, P.L. (1997): *Non-monotone trust-region algorithms for nonlinear optimization subject to convex constraints.* Mathematical Programming, **77**:69–94.

[121] Ulbrich, S. (2004): *On the superlinear local convergence of a filter-SQP method.* Mathematical Programming, Series B, **100**:217–245.

[122] Ulbrich, S. and Ulbrich, M. (2003): *Non-monotone trust region methods for nonlinear equality constrained optimization without a penalty function.* Mathematical Programming, Series B, **95**:103–135.

[123] Vardi, A. (1985): *A trust region algorithm for equality constrained minimization: Convergence properties and implementation.* SIAM Journal on Numerical Analysis, **22**:575–591.

[124] Viswanathan, J. and Grossmann, I.E. (1990): *A combined penalty function and outer approximation method for MINLP optimization.* Computers and Chemical Engineering, **14**:769–782.

[125] Westerlund, T. and Pörn, R. (2002): *Solving pseudo-convex mixed integer optimization problems by cutting plane techniques.* Optimization and Engineering, **3**:253–280.

[126] Wilson, R.B. (1963): *A simplicial algorithm for concave programming.* Ph.D. thesis, Harvard University, Cambridge.

[127] Winfield, D. (1973): *Function minimization by interpolation in a data table.* Journal of the Institute of Mathematics and Its Applications, **12**:339–347.

[128] Wolfe, P. (1969): *Convergence conditions for ascent methods.* SIAM Review, **11**:226–235.

[129] Yuan, Y. (1985): *On the superlinear convergence of a trust region algorithm for nonsmooth optimization.* Mathematical Programming, **31**:269–285.

[130] Yuan, Y. (1995): *On the convergence of a new trust region algorithm.* Numerische Mathematik, **70**:515–539.

[131] Yuan, Y. (2000): *A review of trust region algorithms for optimization.* In: J.M. Ball and J.C.R. Hunt (eds.), *ICM99: Proceedings of the Fourth International Congress on Industrial and Applied Mathematics*, 271–282. Oxford University Press, Oxford.

Danksagung

Mein herzlicher Dank gilt Herrn Prof. Dr. Klaus Schittkowski für die gute langjährige Zusammenarbeit und Unterstützung. Viele hilfreiche Hinweise haben zur Entstehung dieser Arbeit beigetragen. Außerdem bedanke ich mich bei Herrn Prof. Dr. Hans Josef Pesch und Herrn Prof. Dr. Michael Ulbrich dafür, dass sie sich als Gutachter zur Verfügung gestellt haben.

Mein Dank gilt auch meinen ehemaligen Kollegen bei der Angewandten Informatik VII der Universität Bayreuth, meiner Familie und meinen Freunden, mit deren Hilfe so manche Hürde aus dem Weg geräumt wurde. Insbesondere bedanke ich mich bei Kathrin für ihre Unterstützung und ihre unendliche Geduld.

Oliver Exler
Nürnberg, im Dezember 2013