Universität Bremen
Zentrum für Technomathematik

Effizienzsteigerung numerischer Verfahren der nichtlinearen Optimierung

Dissertation
zur Erlangung eines Doktorgrades der Naturwissenschaften
der Fakultät für Mathematik an der Universität Bremen

vorgelegt von
Sören Geffken

Datum des Promotionskolloquiums: 17. August 2017

1. Gutachter: Prof. Dr. Christof Büskens, Universität Bremen
2. Gutachter: Prof. Dr. Jörg Fliege, University of Southampton

29. Mai 2017

Vorwort

Die vorliegende Arbeit wurde nur durch die Hilfe verschiedener Personen ermöglicht, so dass ich diesen zunächst danken möchte.
Mein besonderer Dank gilt meiner Freundin Antonia Koch, die in mir den Wunsch weckte nach meiner Ausbildung das Studium der Technomathematik aufzunehmen und die mich durch ihre Unterstützung diese Entscheidung nie bereuen oder anzweifeln lies. Großer Dank gebührt genauso meinen Eltern für ihre bedingungslose Förderung meiner Ideen und Entscheidungen.

Weiterhin danke ich Prof. Dr. Christof Büskens, der für meine Anliegen, Fragen und Ideen immer ein offenes Ohr hatte und durch kritische Fragen und Anregungen eine wesentliche Hilfe für diese Arbeit war. Dennis Wassel möchte ich für die lehrreichen Gespräche und Anleitungen zur Programmierung besonders danken. Ebenso danke ich Matthias Knauer, der immer mit Rat und Tat hilfreich zur Verfügung stand.
Für die fruchtbaren Gespräche über nichtlineare Optimierung, deren Implementierung und über die Techniken und Grenzen der Parallelisierung danke ich außerdem meinen Kollegen Renke Kuhlmann, Arne Berger und Marcel Jacobse. Diesen danke ich zusätzlich für ihre Bereitschaft als Korrekturleser zur Verfügung zu stehen.
Zuletzt möchte ich natürlich auch allen übrigen Mitgliedern der Arbeitsgruppe danken, die als fleißige Anwender unserer Software stetig als Tester zur Verfügung standen und durch Fragen und Anforderungen neue Ideen gefördert haben.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Symbolverzeichnis

$A \in \mathbb{R}^{N_h \times N_x}$	Matrix der Gleichungsnebenbedingungen des quadratischen Optimierungsproblems
$\mathcal{A}(x)$	Indexmenge der aktiven Nebenbedingungen in x
$\alpha \in (0, 1] \subset \mathbb{R}$	Schrittweite innerhalb der Liniensuche
$\beta \in \mathbb{R}^{N_h}$	Rechte Seite der Gleichungsnebenbedingungen des quadratischen Optimierungsproblems
$C \in \mathbb{R}^{N_g \times N_x}$	Matrix der Ungleichungsnebenbedingungen des quadratischen Optimierungsproblems
$c \in \mathbb{R}^{N_x}$	Lineare Komponente der Zielfunktion des quadratischen Optimierungsproblems
$d \in \mathbb{R}^{N_x}$	Suchrichtung für die Optimierungsvariablen
$\delta \in \mathbb{R}^{N_\delta}$	Vektor der Relaxierungsvariablen
e	Spaltenvektor mit $e \coloneqq (1, 1, ..., 1)\top$
$e_{\{k\}}$	k-ter Einheitsvektor
ϵ_{con}	Zulässigkeitstoleranz des Algorithmus
ϵ_{opt}	Optimalitätstoleranz des Algorithmus
$\eta \in \mathbb{R}^{N_\delta}$	Bestrafungsparameter für die Relaxierungsvariablen
$f : \mathbb{R}^{N_x} \to \mathbb{R}$	Zielfunktion des nichtlinearen Optimierungsproblems
$g : \mathbb{R}^{N_x} \to \mathbb{R}^{N_g}$	Ungleichungsnebenbedingungen des nichtlinearen Optimierungsproblems
$g_{\mathcal{A}}$	Einschränkung der Ungleichungsnebenbedingungen auf die aktiven Komponenten gemäß der aktiven Menge $\mathcal{A}$
$\Gamma \in \mathbb{R}^{N_g \times N_g}$	Diagonalmatrix mit Vektor $g(x)$ auf der Diagonalen
$\gamma \in \mathbb{R}^{N_g}$	Rechte Seite der Ungleichungsnebenbedingungen des quadratischen Optimierungsproblems
$h : \mathbb{R}^{N_x} \to \mathbb{R}^{N_h}$	Gleichungsnebenbedingungen des nichtlinearen Optimierungsproblems
$I_m \in \mathbb{R}^{m \times m}$	Einheitsmatrix der Dimension m
$\text{im}(M)$	Bild der Matrix M
$\text{inertia}(M)$	Signatur der Matrix M
$\text{int}(\Sigma)$	Inneres der Menge Σ

$\ker(M)$	Kern der Matrix M
$K_{ip} \in \mathbb{R}^{N_x+N_h+N_g \times N_x+N_h+N_g}$	KKT-Matrix des quadratischen Optimierungsproblems (Innere-Punkte-Verfahren)
$\Lambda \in \mathbb{R}^{N_g \times N_g}$	Diagonalmatrix mit Vektor λ auf der Diagonalen
$\lambda \in \mathbb{R}^{N_g}$	Vektor der Lagrange-Multiplikatoren der Ungleichungsnebenbedingungen des nichtlinearen Optimierungsproblems
$\lambda^* \in \mathbb{R}^{N_x}$	Optimale Lagrange-Multiplikatoren der Ungleichungsnebenbedingungen des nichtlinearen Optimierungsproblems
$\mathcal{L}(x, y, z)$	Lagrange-Funktion des quadratischen Optimierungsproblems
$L(x, \lambda, \mu)$	Lagrange-Funktion des nichtlinearen Optimierungsproblems
$\mu \in \mathbb{R}^{N_h}$	Vektor der Lagrange-Multiplikatoren der Gleichungsnebenbedingungen des nichtlinearen Optimierungsproblems
$\mu^* \in \mathbb{R}^{N_h}$	Optimale Lagrange-Multiplikatoren der Gleichungsnebenbedingungen des nichtlinearen Optimierungsproblems
$N_\delta \in \mathbb{N}$	Anzahl der Relaxierungsvariablen
$N_g \in \mathbb{N}$	Anzahl der Ungleichungsnebenbedingungen
$N_h \in \mathbb{N}$	Anzahl der Gleichungsnebenbedingungen
$\|.\|$	Eine Norm für $\mathbb{R}^{N_x}$, wenn nicht anders erwähnt euklidischer Abstand
$N_p \in \mathbb{N}$	Dimension des Störungsvektors p
$N_x \in \mathbb{N}$	Anzahl der Optimierungsvariablen
Ω	Diagonalmatrix mit Vektor der Bestrafungswerte der Relaxierung η auf der Diagonalen
$p \in \mathbb{R}^{N_p}$	Vektor der allgemeinen parametrischen Störung
$q \in \mathbb{R}^{N_g+N_h}$	Vektor der additiven Störung der Nebenbedingungen
$Q \in \mathbb{R}^{N_x \times N_x}$	Matrix in der Zielfunktion des quadratischen Optimierungsproblems
$r \in \mathbb{R}^{N_x}$	Vektor der natürlichen Störung der Zielfunktion
$\mathcal{S} \subset \mathbb{R}^{N_x}$	Abgeschlossene und konvexe Menge
$S \in \mathbb{R}^{N_g \times N_g}$	Diagonalmatrix mit Vektor s auf der Diagonalen
$s \in \mathbb{R}^{N_g}$	Vektor der Schlupfvariablen für die Ungleichungsnebenbedingungen des quadratischen Optimierungsproblems
$\Sigma \subset \mathbb{R}^{N_x}$	Zulässige Menge des nichtlinearen Optimierungsproblems
$T(\Sigma, x)$	Tangentialkegel an Σ in x

$x \in \mathbb{R}^{N_x}$	Vektor der Optimierungsvariablen
$x^* \in \mathbb{R}^{N_x}$	Optimale Lösung eines Optimierungsproblems
$y \in \mathbb{R}^{N_h}$	Vektor der Lagrange-Multiplikatoren der Gleichungsnebenbedingungen des quadratischen Optimierungsproblems
$y^* \in \mathbb{R}^{N_h}$	Optimale Lagrange-Multiplikatoren der Gleichungsnebenbedingungen des quadratischen Optimierungsproblems
$Z \in \mathbb{R}^{N_g \times N_g}$	Diagonalmatrix mit Vektor z auf der Diagonalen
$z \in \mathbb{R}^{N_g}$	Vektor der Lagrange-Multiplikatoren der Ungleichungsnebenbedingungen des quadratischen Optimierungsproblems
$z^* \in \mathbb{R}^{N_x}$	Optimale Lagrange-Multiplikatoren der Ungleichungsnebenbedingungen des quadratischen Optimierungsproblems

Kapitel 1

Einführung

Inhaltsangabe

Die mathematische Modellierung wird von Forschern verschiedenster Disziplinen verwendet, um ein tieferes Verständnis der zu analysierenden Prozesse und Zusammenhänge zu gewinnen. Mit Hilfe der Modelle können anschließend Simulationen durchgeführt werden. Das Ziel der Simulation ist dabei, neben Erkenntnissen über den Ausgangsprozess, auch die Möglichkeit, die Auswirkungen verschiedener Konfigurationen und Parameter auf das simulierte System zu ergründen und daraufhin durch gezielte Einflussnahme ein gewünschtes Verhalten des Prozesses zu erreichen. Diverse Forschungsbereiche verfolgen diese Vorgehensweise. Als motivierende Beispiele seien die Luft- und Raumfahrt, der Automobilsektor, der Energiesektor, aber auch medizinische Anwendungen oder betriebswirtschaftliche Problemstellungen genannt.
Bereits während der Modellierungsphase müssen gegebenenfalls verwendete Modellparameter identifiziert werden, so dass das Modell in der Lage ist vorhandene Messdaten zu reproduzieren. Dieses Identifikationsproblem entspricht einem nichtlinearen Optimierungsproblem. Während der Simulationsphase sollen Steuerungsgrößen so konfiguriert werden, dass sich das System im Sinne des Entwicklers optimal verhält. Auch bei der Lösung dieses Problems ist die nichtlineare Optimierung von entscheidender Bedeutung. Insgesamt zeigt sich, dass die nichtlineare Optimierung in verschiedenen Phasen des Prozesses ein entscheidender Faktor ist.
Auf Grund der stetig steigenden Kapazitäten moderner Computer wachsen auch die behandelbaren Problemgrößen. Diese Entwicklung führt dazu, dass weitere Anwendungsmöglichkeiten der nichtlinearen Optimierung entstehen, da mittlerweile Prozesse optimiert werden können, deren Simulation vor einigen Jahren noch nicht vorstellbar war.

Die wachsende Anzahl von Verwendungsmöglichkeiten erhöht auch die Anforderungen an die verwendeten Algorithmen zur Optimierung. Gleichzeitig sind Mathematiker mit der grundsätzlichen Theorie der nichtlinearen Optimierung zufrieden, da Konvergenzbeweise für die gängigen Verfahren existieren und die bestehenden Softwarepakete prinzipiell in der Lage sind die auftretenden Probleme zu lösen. Gleichwohl weisen die verwendeten Algorithmen nach wie vor Verbesserungspotenzial auf. Im Verlauf dieser Arbeit werden diverse Anpassungen innerhalb der sequentiellen quadratischen Programmierung (SQP) diskutiert und deren Verbesserungspotenzial offenbart. Neben den Anpassungen der mathematischen Algorithmen liegt ein weiterer Schwerpunkt auf der Berücksichtigung aktueller Entwicklungen der verwendeten Hardware. Durch die zunehmende Verfügbarkeit und Verbreitung von Mehrkernprozessoren bietet die Berücksichtigung von Parallelisierungsmöglichkeiten großes Potenzial zur Steigerung der Effizienz von Optimierungsverfahren.

1.1 Zielsetzung

Das Ziel der vorliegenden Arbeit ist mit Hilfe verschiedener Techniken eine Effizienzsteigerung nichtlinearer Optimierungsverfahren zu erwirken. Die Effizienz eines numerischen Verfahrens zur Lösung nichtlinearer Optimierungsprobleme ist durch verschiedene Aspekte geprägt. Naheliegende Kriterien sind die Lösungsgeschwindigkeit, die Lösungsqualität und die Robustheit des Verfahrens. Aber auch leichte Bedienbarkeit und Konfigurationsmöglichkeiten erhöhen die Effizienz eines Verfahrens für den Anwender.
Als Grundlage für die Untersuchungen dient eine ausführliche Analyse der im Löser WORHP implementierten SQP-Methode mit einem Innere-Punkte-Verfahren zur Lösung der Unterprobleme. Auf Basis dieser Analyse werden mit Hilfe der parametrischen Sensitvitätsanalyse Techniken entwickelt, um die Behandlung der nichtlinearen Nebenbedingungen, trotz der Linearisierung innerhalb der Unterprobleme, zu verbessern.
Zur Behandlung inkonsistenter linearisierter Nebenbedingungen existiert in WORHP eine Relaxierungsstrategie. Im weiteren Verlauf schlägt diese Arbeit eine adaptive Strategie zur Effizienzsteigerung der Relaxierung vor. Weiterhin werden Strategien vorgestellt, um die notwendige Relaxierung der Nebenbedingungen sensitivitätsbasiert zu verbessern. Zusätzlich findet eine Übertragung dieser Ideen auf die Verbesserung der Regularisierung der Hesse-Matrix statt.
Neben den Erweiterungen basierend auf der parametrischen Sensitivitätsanalyse ist als weiteres Ziel zu untersuchen, ob durch eine Variation der Art der verwendeten Hesse-Matrix während der Optimierung eine Verbesserung des Algorithmus möglich ist. Mit Hilfe einer Wechselstrategie soll die notwendige Regularisierung zu Beginn der Optimierung reduziert und so eine Effizienzsteigerung erzielt werden.
Ein weiterer Schwerpunkt der Arbeit liegt auf der Evaluierung möglicher Parallelisierungsstrategien für das Optimierungswerkzeug WORHP. Bei der Einbindung

dieser Strategien liegt ein besonderes Augenmerk auf der Bedienbarkeit durch den Anwender, um eine effiziente Nutzung des Lösers zu ermöglichen.

1.2 Aufbau

Die Arbeit gliedert sich in die folgenden Kapitel: Das vorliegende Kapitel 1 beinhaltet die Einleitung, die Zielsetzung, den Aufbau und eine kurze Übersicht über den wissenschaftlichen Beitrag der Arbeit. Kapitel 2 stellt die wesentlichen mathematischen Grundlagen der nichtlinearen Optimierung vor. Das anschließende Kapitel 3 thematisiert zunächst Algorithmen zur Lösung von quadratischen Optimierungsproblemen. Daraufhin werden Algorithmen zur Lösung nichtlinearer Optimierungsprobleme diskutiert. Ein besonderer Fokus liegt dabei auf dem Löser WORHP.
Kapitel 4 diskutiert die theoretischen Hintergründe der parametrischen Sensitivitätsanalyse, um die Grundlage für die Anwendung dieser Sensitivitätsanalyse innerhalb des SQP-Verfahrens in Kapitel 5 bereitzustellen.
Folgend behandelt Kapitel 6 Parallelisierungsmöglichkeiten der nichtlinearen Optimierung anhand des Lösers WORHP.
Die numerischen Auswertungen der verschiedenen entwickelten Techniken werden anschließend in Kapitel 7 vorgestellt. Abschließend findet in Kapitel 8 eine Schlussbetrachtung mit Ausblick statt.

1.3 Wissenschaftlicher Beitrag

Der wissenschaftliche Beitrag dieser Arbeit gliedert sich in drei zentrale Bereiche. Diese sind in diesem Abschnitt kurz zusammenfassend dargestellt.

Allgemeine Techniken innerhalb des SQP-Verfahrens

Innerhalb der Darstellung der theoretischen Grundlagen des Lösers WORHP in Abschnitt 3.3.3.1 liegt ein besonderer Fokus auf den verschiedenen Möglichkeiten die Hesse-Matrix zu verwenden. Nach Kenntnis des Autors wird für den Löser WORHP in Folgerung 3.19 erstmals bewiesen, dass die Eigenwertverteilung der Systemmatrix innerhalb des Innere-Punkte-Verfahrens auf Unterproblemebene Rückschlüsse auf die positive Definitheit der Hesse-Matrix auf dem Kern der aktiven Nebenbedingungen ermöglicht.
Im Anschluss daran thematisiert Abschnitt 3.3.3.2 neuartige Techniken die verwendete Hesse-Matrix während des Optimierungsverlaufes zu variieren, um so einerseits von inhärent positiv definiten Näherungsmatrizen und andererseits von den lokal quadratischen Konvergenzeigenschaften der analytischen Hesse-Matrix profitieren zu können.

Das SQP-Verfahren linearisiert die Nebenbedingungen in jeder Iteration für die Verwendung innerhalb der Unterprobleme. Abschnitt 3.3.3.3 stellt zur Verbesserung der Relaxierungsstrategie innerhalb von WORHP einen Algorithmus zur adaptiven Konfiguration der Relaxierung vor. Der Algorithmus erhöht gezielt die Flexibilität der Relaxierung für einzelne Nebenbedingungen, indem bei numerischen Problemen zusätzliche Relaxierungsvariablen eingeführt werden können.

Parametrische Sensitivitätsanalyse innerhalb des SQP-Verfahrens

Während der Erläuterungen zur theoretischen Grundlage der parametrischen Sensitivitätsanalyse werden speziell strukturierte Störungen in Abschnitt 4.2 vorgestellt. Diese ergänzen die von Büskens [11, 12] eingeführten Störungsarten, um Störungen jeglicher Größen innerhalb von quadratischen Optimierungsproblemen numerisch effizient zu analysieren.
Kapitel 5 beschäftigt sich mit der Analyse und Erweiterung der erstmalig von Nikolayzik [81] vorgeschlagenen Verwendungsmöglichkeiten der parametrischen Sensitivitätsanalyse innerhalb des SQP-Verfahrens. Ein besonderer Schwerpunkt liegt in diesem Zusammenhang auf der Zulässigkeitskorrektur in Abschnitt 5.2, erste Ergebnisse dieser Technik wurden bereits in dem Artikel von Geffken und Büskens [42] veröffentlicht. Für diese Korrektur werden diverse Spezialisierungen des Algorithmus und Start- und Abbruchkriterien eingeführt. Erweiternde Techniken, basierend auf der parametrischen Sensitivität im Zusammenhang mit der Regularisierung der Hesse-Matrix, stellt Abschnitt 5.3 vor. Abschnitt 5.4 handelt von der sensitivitätsbasierten Erweiterung der Relaxierung der Nebenbedingungen innerhalb des SQP-Verfahrens.

Parallelisierung des SQP-Verfahrens

Ein weiterer Beitrag dieser Arbeit besteht in der Einführung einer neuen Mehrkernschnittstelle für den Löser WORHP in Abschnitt 6.3. Diese Schnittstelle ermöglicht es dem Benutzer seine Problemstellung komfortabel unter Ausnutzung von Objektorientierung einzugeben. Diese Objekte erlauben eine Verwendung des Benutzerproblems in parallelen Threads. Der klassische Master-Slave-Ansatz zur Parallelisierung wird daraufhin auf diese Situation übertragen und erzielt auf diese Weise eine Steigerung der Lösungsgeschwindigkeit und Robustheit von WORHP. Ein Parameterindividualisierungsmodus stellt eine weitere Anwendungsmöglichkeit der parallelen Schnittstelle dar. Dieser erleichtert dem Benutzer die Konfiguration des Lösers für sein konkretes Problem erheblich. Einen ersten Einblick in das Thema gibt der Konferenzbeitrag von Geffken und Büskens [43].

Kapitel 2

Grundlagen der nichtlinearen restringierten Optimierung

Inhaltsangabe

Dieses Kapitel stellt in kompakter Form die theoretischen Grundlagen der nichtlinearen Optimierung dar. Die vorgestellten Definitionen, Sätze und Folgerungen finden sich in diversen Büchern zum Beispiel von Geiger und Kanzow [44, 45], Alt [1], Spellucci [97], Nocedal und Wright [82], Fletcher [35] oder Gill, Murray und Wright [50].
Die folgenden Inhalte sind größtenteils auf die für den weiteren Verlauf der Arbeit relevanten Aspekte reduziert. Weitere Details und fehlende Beweise finden sich in den oben genannten Büchern. Die verwendeten Bezeichnungen orientieren sich an der Arbeit von Büskens [11] und sind zusätzlich im Symbolverzeichnis aufgelistet.

2.1 Das nichtlineare Optimierungsproblem

Die allgemeine Aufgabenstellung der nichtlinearen Optimierung ist die Minimierung einer gegebenen Funktion unter einzuhaltenden Nebenbedingungen. Mathematisch lässt sich dies wie folgt beschreiben.

Definition 2.1. *Es seien Dimensionen* $0 < N_x \in \mathbb{N}$, $0 \leq N_g \in \mathbb{N}$ *und* $0 \leq N_h \in \mathbb{N}$ *gegeben. Es seien weiterhin zweimal stetig differenzierbare Funktionen* $f : \mathbb{R}^{N_x} \to \mathbb{R}$, $g : \mathbb{R}^{N_x} \to \mathbb{R}^{N_g}$ *und* $h : \mathbb{R}^{N_x} \to \mathbb{R}^{N_h}$ *gegeben. Zusätzlich sei eine abgeschlossene und konvexe Menge* $\mathcal{S} \subset \mathbb{R}^{N_x}$ *mit nichtleerem Inneren* $int(\mathcal{S}) \neq \emptyset$

definiert. Dann heißt

$$\begin{aligned} \min_{x\in\mathbb{R}^{N_x}} \quad & f(x) \\ \textit{unter} \quad & g(x) \leq 0 \\ & h(x) = 0 \\ & x \in \mathcal{S} \end{aligned} \tag{2.1}$$

Standardproblem der nichtlinearen Optimierung. *In diesem Kontext heißt* $f(x)$ Zielfunktion*, die Funktion* $g(x)$ *beschreibt die* Ungleichungsnebenbedingungen *und* $h(x)$ *die* Gleichungsnebenbedingungen*.*

Bemerkung 2.2. *Die Menge* $\mathcal{S}$ *wird in praktischen Anwendungen in der Regel durch sogenannte Boxschranken mit Schranken* $l, u \in \mathbb{R}$ *der Form*

$$l \leq x_i \leq u, \quad i \in 1, ..., N_x \tag{2.2}$$

beschrieben. Im weiteren Verlauf sei zunächst angenommen, dass $\mathcal{S} = \mathbb{R}^{N_x}$ *gilt. Abweichungen werden gesondert erwähnt.*

Proposition 2.3. *Mit Hilfe der allgemeinen Formulierung* (2.1) *lassen sich durch Ersetzen der Zielfunktion durch* $\hat{f}(x) := -f(x)$ *auch Maximierungsaufgaben lösen.*

Für die späteren Betrachtungen, insbesondere zur numerischen Lösung von nichtlinearen Optimierungsproblemen ist die Aufgabenstellung der quadratischen Optimierung von zentraler Bedeutung.

Definition 2.4. *Seien* $x, c \in \mathbb{R}^{N_x}$*,* $\beta \in \mathbb{R}^{N_h}$ *und* $\gamma \in \mathbb{R}^{N_g}$*. Weiterhin seien die Matrizen* $Q \in \mathbb{R}^{N_x \times N_x}$*,* $A \in \mathbb{R}^{N_h \times N_x}$ *und* $C \in \mathbb{R}^{N_g \times N_x}$ *gegeben. Dann ist*

$$\begin{aligned} \min_{x\in\mathbb{R}^{N_x}} \quad & \frac{1}{2} x^\top Q x + c^\top x \\ \textit{unter} \quad & Ax = \beta \\ & Cx \leq \gamma \end{aligned} \tag{2.3}$$

die Standardformulierung des quadratischen Optimierungsproblems*. Es liegen eine quadratische Zielfunktion und lineare Nebenbedingungen vor.*

Im Folgenden werden einige für den Verlauf dieser Arbeit gültige Begrifflichkeiten anhand der allgemeinen nichtlinearen Problemstellung definiert. Diese gelten analog für Spezialisierungen der Problemformulierung.

Definition 2.5. *Die Einhaltung der Nebenbedingungen wird als* Zulässigkeit *eines Punktes bezeichnet. Die* zulässige Menge $\Sigma \subset \mathbb{R}^{N_x}$ *wird als*

$$\Sigma := \{x \in \mathcal{S} \mid g(x) \leq 0 \quad \textit{und} \quad h(x) = 0\} \tag{2.4}$$

definiert.

Bemerkung 2.6. *Auf Grund der Abgeschlossenheit der Menge $\mathcal{S}$ und der Stetigkeit der Abbildungen g und h ist auch die zulässige Menge Σ abgeschlossen.*

In numerischen Betrachtungen können Gleichungsnebenbedingungen in der Regel einfacher behandelt werden, deshalb hat sich bei der Entwicklung von Algorithmen die folgende Mengendefinition als nützlich erwiesen.

Definition 2.7. *Eine Ungleichung $g_i(x) \leq 0$ wird als* aktiv *bezeichnet wenn $g_i(x) = 0$ gilt. Der Punkt x liegt somit auf dem durch die Funktion $g_i(x)$ definierten Rand des zulässigen Bereichs Σ. Die Indexmenge*

$$\mathcal{A}(x) := \{i \mid g_i(x) = 0,\ i = 1, ..., N_g\} \tag{2.5}$$

wird als aktive Menge *bezeichnet.*

Ziel der nichtlinearen Optimierung ist das Bestimmen von Minimalstellen. Es können verschiedene Arten von Minimalstellen auftreten. Je nach Verhalten der betrachteten Funktionen gilt es lokale und globale Eigenschaften zu unterscheiden. Die folgende Definition erfasst die verschiedenen Situationen.

Definition 2.8. *Ein zulässiger Punkt $x^* \in \Sigma$ ist ein* globales Minimum*, wenn die Ungleichung*

$$f(x^*) \leq f(x), \quad \forall x \in \Sigma \tag{2.6}$$

gilt. Für ein strenges globales Minimum *kann die Aussage auf*

$$f(x^*) < f(x), \quad x \neq x^*,\ \forall x \in \Sigma \tag{2.7}$$

erweitert werden.
Es sei ein Radius $\varepsilon > 0$ gegeben. Damit wird eine Umgebung
$U_\varepsilon(x^*) := \{x \in \mathbb{R}^{N_x} \mid \|x - x^*\| \leq \varepsilon\}$ definiert. Ein lokales Minimum *von* (2.1) *erfüllt die Gleichung*

$$f(x^*) \leq f(x), \quad \forall x \in U_\varepsilon(x^*) \cap \Sigma. \tag{2.8}$$

Analog zum strengen globalen Minimum liefert die Beziehung

$$f(x^*) < f(x), \quad x \neq x^*,\ \forall x \in U_\varepsilon(x^*) \cap \Sigma. \tag{2.9}$$

ein strenges lokales Minimum*.*

Abbildung 2.1 zeigt lokales und globales Minimum einer Beispielfunktion.
Zentral für den Inhalt dieser Ausarbeitung ist die Beschreibung von Algorithmen zum Auffinden lokaler Minima für unterschiedliche Problemklassen. Eine wichtige Eigenschaft von iterativen Algorithmen ist die Konvergenzordnung der erzeugten Folge von Iterierten.

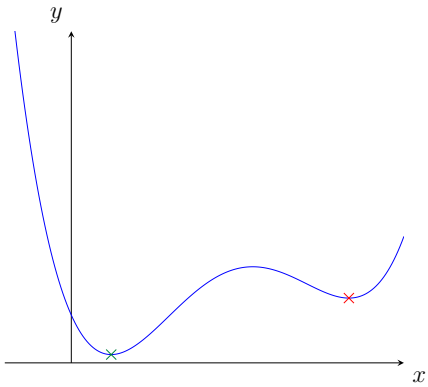


Abbildung 2.1: Unterscheidung lokales und globales Minimum. Lokales Minimum der Funktion in rot und globales Minimum in Grün.

Definition 2.9 (Konvergenzordnung)**.** *Es sei* $\{x^{[k]}\} \subset \mathbb{R}^{N_x}$ *eine Folge mit*

$$\lim_{k\to\infty} x^{[k]} = x^*. \tag{2.10}$$

Es sei eine Konstante $c \in (0,1)$ *gegeben. Ist die Bedingung*

$$\frac{\|x^{[k+1]} - x^*\|}{\|x^{[k]} - x^*\|} \leq c, \quad \textit{für alle} \quad k = 0, 1, 2, ... \tag{2.11}$$

erfüllt, so heißt die Folge $\{x^{[k]}\}$ linear konvergent*.*

Sei ferner $c_k \subset \mathbb{R}$ *eine Nullfolge, das heißt*

$$\lim_{k\to\infty} c_k = 0.$$

Ist die Bedingung

$$\frac{\|x^{[k+1]} - x^*\|}{\|x^{[k]} - x^*\|} \leq c_k, \quad \textit{für alle} \quad k = 0, 1, 2, ... \tag{2.12}$$

erfüllt, so heißt die Folge $\{x^{[k]}\}$ superlinear konvergent*.*

Sei $c \in (0,1)$ *und* $p > 1$*. Ist die verallgemeinerte Bedingung*

$$\frac{\|x^{[k+1]} - x^*\|}{\|x^{[k]} - x^*\|^p} \leq c, \quad \textit{für alle} \quad k = 0, 1, 2, ... \tag{2.13}$$

erfüllt, so heißt die Folge konvergent von der Ordnung p*. Der Spezialfall* $p = 2$ *wird als* quadratische Konvergenz *bezeichnet.*

2.2 Regularitäts- und Optimalitätsbedingungen

In der zu Beginn dieses Kapitels genannten Literatur finden sich verschiedene Bedingungen zur Charakterisierung der Minimalstellen von Optimierungsproblemen. Zunächst wird die unbeschränkte Optimierung als Motivation für die Situation mit Nebenbedingungen dargestellt.

Satz 2.10. *Es sei die Problemstellung 2.1 mit $N_g = N_h = 0$ und $\mathcal{S} = \mathbb{R}^{N_x}$, das heißt $\Sigma = \mathbb{R}^{N_x}$, gegeben. Ist die notwendige Optimalitätsbedingung (erster Ordnung)*

$$\nabla_x f(x^*) = 0 \tag{2.14}$$

und die hinreichende Optimalitätsbedingung (zweiter Ordnung)

$$\nabla^2_{xx} f(x^*) \ \textit{positiv definit} \tag{2.15}$$

erfüllt. Dann ist x^* lokale Minimalstelle von $f(x)$.

Beweis. Fletcher [35] beweist diese Aussage unter Verwendung einer geeigneten Taylor-Reihe mit Entwicklungspunkt im lokalen Minimum. □

Durch die Einführung von Nebenbedingungen verändert sich die Situation und eine alleinige Betrachtung der Zielfunktion kann nicht mehr ausreichend sein. Zur Verdeutlichung der Auswirkungen von Nebenbedingungen hilft die folgende anschauliche Betrachtung die weiteren Hintergründe zu verstehen.

Definition 2.11 (Tangentialkegel)**.** *Sei $x \in \Sigma$. Seien weiterhin Folgen $\{\alpha_k\}_{k\in\mathbb{N}}$ und $\{x_k\}_{k\in\mathbb{N}} \subset \Sigma$ mit*

$$\lim_{k\to\infty} \alpha_k = 0 \tag{2.16}$$

$$\lim_{k\to\infty} x_k = x \tag{2.17}$$

gegeben. Die Menge

$$T(\Sigma, x) = \left\{ d \in \mathbb{R}^{N_x} \mid \exists \{\alpha_k\}, \{x_k\}, \ \textit{mit} \lim_{k\to\infty} \frac{x_k - x}{\alpha_k} = d \right\} \tag{2.18}$$

beschreibt den Tangentialkegel *an Σ in x.*

In einem Minimum x^* beschreibt der Tangentialkegel $T(\Sigma, x^*)$ alle Richtungen die vom Minimum in den zulässigen Bereich zeigen. Abbildung 2.2 skizziert beispielhaft eine derartige Situation.

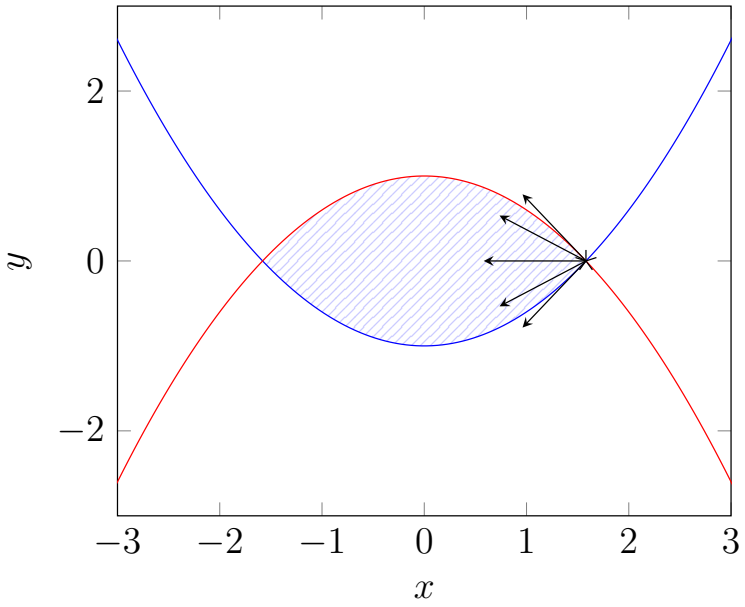


Abbildung 2.2: Beispielhafte Darstellung eines Tangentialkegels. Zulässiger Bereich in blau schraffiert. Zulässige Richtungen ausgehend vom Schnittpunkt der beiden Kurven als Vektoren dargestellt.

Es lässt sich folgende Aussage über das lokale Verhalten der Zielfunktion innerhalb des Tangentialkegels treffen.

Satz 2.12. *Sei x^* lokales Minimum der Optimierungsaufgabe 2.1. Die Zielfunktion $f : \mathbb{R}^{N_x} \to \mathbb{R}$ sei differenzierbar in x^*. Dann gilt*

$$\nabla_x f(x^*)^\top d \geq 0, \quad \forall d \in T(\Sigma, x^*). \tag{2.19}$$

Beweis. Im Buch von Fletcher [35] findet sich ein Beweis des Satzes. □

Bemerkung 2.13. *Im unrestringierten Fall $\Sigma = \mathbb{R}^{N_x}$ oder mit $x^* \in int(\Sigma)$ ist $T(\Sigma, x^*) = \mathbb{R}^{N_x}$ und die Aussage von Satz 2.12 reduziert sich auf die bereits beschriebene Bedingung* (2.14).

Bemerkung 2.14. *Mit Hilfe einer Schranke $\bar{\alpha} \in \mathbb{R}$ wird deutlich, dass der Tangentialkegel* zulässige *Richtungen beschreibt:*

$$Z(\Sigma, x) = \{d \in \mathbb{R}^{N_x} \mid \exists \bar{\alpha} > 0 \text{ so dass } \forall \alpha \in (0, \bar{\alpha}] : x + \alpha d \in \Sigma\} \tag{2.20}$$

Mit Hilfe dieser lässt sich eine erste notwendige Optimalitätsbedingung für ein Minimum x^* formulieren:

$$Z(\Sigma, x) \cap \{d \in \mathbb{R}^{N_x} \mid \nabla_x f(x^*)^\top d < 0\} = \emptyset \tag{2.21}$$

Anschaulich verlangt die Bedingung, dass die Zielfunktion in keiner zulässigen Richtung weiter verkleinert werden kann.

Geometrisch ist der Tangentialkegel zunächst eine anschauliche Bedingung. In der praktischen Anwendung ist der Tangentialkegel allerdings schwierig zu verwerten, da es im Allgemeinen schwer möglich ist diesen effizient zu bestimmen. Die folgenden Betrachtungen helfen bei der Suche nach numerisch verwertbaren Kriterien.
Zunächst erfordert die Formulierung von Optimalitätsbedingungen für restringierte Optimierungsprobleme eine Kombination der Zielfunktion mit den Nebenbedingungen.

Definition 2.15 (Lagrange-Funktion)**.** *Seien* Multiplikatoren $l_0 \in \mathbb{R}, \lambda \in \mathbb{R}^{N_g}$ *und* $\mu \in \mathbb{R}^{N_h}$ *gegeben. Die Funktion*

$$L(x, l_0, \lambda, \mu) = l_0 f(x) + \lambda^\top g(x) + \mu^\top h(x) \tag{2.22}$$

wird als Lagrange-Funktion *bezeichnet. Die Multiplikatoren* λ *und* μ *werden auch als* Lagrange-Multiplikatoren *oder* duale *Variablen bezeichnet. In diesem Zusammenhang wird* x *auch* primale *Variable genannt.*

Unter Verwendung der Lagrange-Funktion lässt sich eine notwendige Bedingung erster Ordnung für eine lokale Minimalstelle formulieren. Die Bezeichnung erster Ordnung bezieht sich auf den Umstand, dass nur Ableitungsinformationen erster Ordnung benötigt werden.

Satz 2.16. *(Fritz-John Bedingungen) Sei* x^* *ein lokales Minimum der Problemformulierung 2.1. Weiterhin sei die Menge* $\mathcal{S} \subset \mathbb{R}^{N_x}$ *abgeschlossen und konvex mit* $\text{int}(\mathcal{S}) \neq \emptyset$*. Die Funktionen* f, g *und* h *seien stetig differenzierbar. Dann existieren Multiplikatoren* $l_0 \in \mathbb{R}, \lambda \in \mathbb{R}^{N_g}$ *und* $\mu \in \mathbb{R}^{N_h}$ *mit* $(l_0, \lambda, \mu) \neq 0$*, so dass die folgenden Bedingungen gelten:*
Vorzeichenbedingung:

$$l_0 \geq 0, \quad \lambda_i \geq 0, \quad i = 1, ..., N_g \tag{2.23}$$

Optimalitätsbedingung:

$$\nabla_x L(x^*, l_0, \lambda, \mu) = \Big(l_0 \nabla_x f(x^*) + \lambda^\top \nabla_x g(x^*) + \mu^\top \nabla_x h(x^*) \Big) = 0 \tag{2.24}$$

Komplementaritätsbedingung:

$$\lambda_i g_i(x^*) = 0, \quad i = 1, ..., N_g \tag{2.25}$$

Zulässigkeit:

$$g(x^*) \leq 0 \tag{2.26}$$
$$h(x^*) = 0 \tag{2.27}$$

Beweis. Im Buch von Geiger und Kanzow [45] ist ein Beweis des Satzes abgedruckt. □

Definition 2.17. *Erfüllt ein Tupel* $(x, l_0, \lambda, \mu) \neq 0$ *die Fritz-John Bedingungen, so wird dieses als* Fritz-John Punkt *des Optimierungsproblems bezeichnet.*

Der Multiplikator l_0 ist die Besonderheit der Fritz-John Bedingungen. Insbesondere darf der Fall $l_0 = 0$ auftreten. In diesem Fall entarten die Bedingungen und die Zielfunktion tritt nicht mehr explizit auf. Das folgende Beispiel aus dem Buch von Alt [1] verdeutlicht die Situation.

Beispiel 2.18. *Es sei die Problemstellung*

$$\begin{aligned} \min_{x \in \mathbb{R}^2} \quad & f(x) = -x_1 \\ \textit{unter} \quad & g_1(x) = x_2 + x_1^3 \leq 0 \\ & g_2(x) = -x_2 \leq 0 \end{aligned} \tag{2.28}$$

gegeben. In Abbildung 2.3 ist der zulässige Bereich des Problems eingezeichnet. Offensichtlich ist $x^* = (0,0)^\top$ *die globale Minimalstelle und somit Lösung der Aufgabenstellung. Die Gradienten der beteiligten Funktionen sind*

$$\nabla_x f(x^*) = \begin{pmatrix} -1 \\ 0 \end{pmatrix}, \nabla_x g_1(x^*) = \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \nabla_x g_2(x^*) = \begin{pmatrix} 0 \\ -1 \end{pmatrix} \tag{2.29}$$

und demnach kann Gleichung (2.24) *nur mit* $l_0 = 0$ *erfüllt werden, da ansonsten kein passender Multiplikator* λ *gefunden werden kann.*

Durch die Einführung geeigneter Regularitätsbedingungen kann sichergestellt werden, dass nur Fälle mit $l_0 \neq 0$ auftreten. Insbesondere kann in diesen Fällen ohne Beschränkung der Allgemeinheit $l_0 \equiv 1$ gesetzt werden, da alle Multiplikatoren nur linear auftreten.
Fletcher beschreibt in seinem Buch [35], dass durch eine Linearisierung des zulässigen Bereichs derartige Kriterien entwickelt werden können. Die folgenden Bedingungen charakterisieren über die Gradienten der Nebenbedingungen Punkte innerhalb des zulässigen Bereichs.

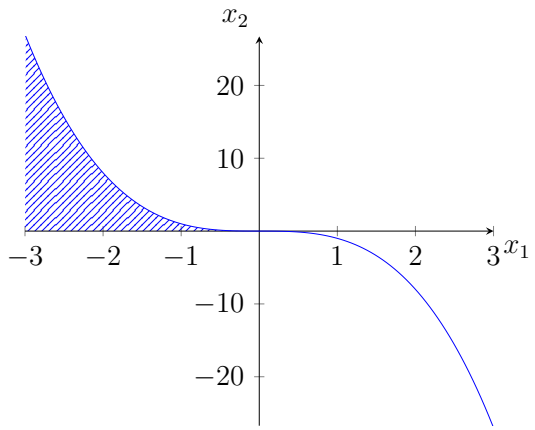


Abbildung 2.3: Zulässiger Bereich des Problems (2.28)

Definition 2.19 (Regularität, Normalität)**.** *Es sei $x^* \in \Sigma$ ein zulässiger Punkt.*

i. Regularitätsbedingung von Mangasarian und Fromowitz [75] *: Der Punkt x^* heißt* regulär*, wenn gilt*

(a) Die Gradienten $\nabla_x h_1(x^*), ..., \nabla_x h_{N_h}(x^*)$ sind linear unabhängig.

(b) Es existiert ein $d \in \mathbb{R}^{N_x} \setminus \{0\}$ mit

$$\nabla_x g_i(x^*)^\top d < 0, \qquad i \in \mathcal{A}(x^*) \tag{2.30}$$

$$\nabla_x h_j(x^*)^\top d = 0, \qquad j = 1, ..., N_h \tag{2.31}$$

ii. Regularitätsbedingung der linearen Unabhängigkeit*: Der Punkt x^* heißt* normal*, wenn die Gradienten $\nabla_x g_i(x^*), \nabla_x h_j(x^*)$ mit $i \in \mathcal{A}(x^*)$ und $j = 1, ..., N_h$ linear unabhängig sind.*

Unter Verwendung dieser Regularitätsbedingungen kann die gewünschte Situation erreicht werden.

Satz 2.20. *Es seien die Voraussetzungen des Satzes 2.16 erfüllt. Weiterhin erfülle die Minimalstelle x^* die Regularitätsbedingung von Mangasarian und Fromowitz, dann gelten die Aussagen von Satz 2.16 mit $l_0 = 1$.*

Beweis. Mangasarian und Fromowitz beweisen diesen Satz in ihrem Artikel [75]. □

Die Normalität eines Punktes ist eine stärkere Eigenschaft als die Regularität. Insbesondere impliziert Normalität die Regularität des Punktes. Die Aussagekraft des Satzes 2.16 wird unter Normalität sogar noch erweitert und liefert ein numerisch überprüfbares Kriterium.

Satz 2.21 (Karush-Kuhn-Tucker (KKT)-Bedingungen)**.** *Es seien die Voraussetzungen des Satzes 2.16 erfüllt. Weiterhin erfülle die Minimalstelle x^* die Regularitätsbedingung der linearen Unabhängigkeit, dann gelten die Aussagen von Satz 2.16 mit $l_0 = 1$. Weiterhin ergibt sich die Optimalitätsbedingung*

$$\begin{aligned} &\nabla_x L(x^*, \lambda^*, \mu^*) = \\ &\nabla_x f(x^*) + (\lambda^*)^\top g(x^*) + (\mu^*)^\top h(x^*) = 0, \end{aligned} \tag{2.32}$$

mit eindeutigen Lagrange-Multiplikatoren λ^*, μ^*.

Beweis. Ein Beweis ist im Buch von Nocedal und Wright [82] zu finden. □

Die Komplementaritätsbedingung (2.25) aus Satz 2.16 erlaubt in der allgemeinen Form den entarteten Fall, dass

$$\lambda_i g_i(x^*) = 0, \quad \lambda_i = g_i(x^*) = 0 \text{ für ein } i \in 1, ..., N_g \tag{2.33}$$

gilt. Die Nebenbedingung g_i ist demnach in x^* aktiv. In Kapitel 4 wird gezeigt, dass die Multiplikatoren λ die Sensitivität der Zielfunktion bezüglich einer Störung der entsprechenden Nebenbedingung darstellen. Dieser entartete Fall entspricht damit einer aktiven Nebenbedingung ohne Auswirkung auf die Zielfunktion in einer kleinen Umgebung um x^*. Spellucci [97] formuliert zur Vermeidung dieser Situation die strikte Komplementaritätsbedingung.

Definition 2.22 (Strikte Komplementaritätsbedingung)**.** *Das Tupel* (x^*, λ^*) *erfüllt die* strikte Komplementaritätsbedingung *wenn die Ungleichung*

$$\lambda^* - g(x^*) > 0 \tag{2.34}$$

gültig ist.

Weiterhin sind die Optimalitätsbedingungen aus Satz 2.21 in dieser Form zwar notwendig, allerdings noch nicht hinreichend. Ein Maximum oder Sattelpunkt erfüllt diese Bedingungen ebenfalls. Damit die Kriterien nur noch von Minimalstellen erfüllt werden, müssen, wie im Satz 2.10 für unrestringierte Probleme, zusätzlich hinreichende Bedingungen formuliert werden. Diese beinhalten Krümmungsinformationen und helfen Minimalstellen von Maximalstellen zu unterscheiden.
Im unbeschränkten Fall wurde verlangt, dass die Hesse-Matrix der Zielfunktion positiv definit in der Minimalstelle ist. Wie bei der Formulierung der notwendigen Bedingungen wird im Fall der restringierten Optimierung die Ableitung der Lagrange-Funktion verwendet. Durch die Nebenbedingungen reicht es hier allerdings die Erfüllung der Bedingung entlang zulässiger Richtungen zu fordern, da lediglich diese von Bedeutung sind. Die mathematische Formulierung des Kriteriums erfolgt im folgenden Satz.

Satz 2.23 (Hinreichende Bedingungen zweiter Ordnung)**.** *Es seien die Voraussetzungen von Satz 2.21 und die strikte Komplementaritätsbedingung für* (x^*, λ^*) *erfüllt. Der kritische Kegel zulässiger Richtungen ist durch*

$$T_C(x^*) = \ker(\nabla_x g_i(x^*), \nabla_x h_j(x^*),\ i \in \mathcal{A}(x^*),\ j = 1, ..., N_h) \tag{2.35}$$

in dieser Situation gegeben. Ist die Bedingung

$$d^\top \nabla^2_{xx} L(x^*, \lambda^*, \mu^*) d > 0 \ \ \forall d \in T_C(x^*) \tag{2.36}$$

erfüllt, so ist (x^*, λ^*, μ^*) *strenge lokale Minimalstelle der Problemstellung 2.1.*

Beweis. Im Buch von Fletcher [35] findet sich ein Beweis des Satzes. □

Bemerkung 2.24. *Der kritische Kegel* $T_C(x^*)$ *kann hier im Bezug zum geometrisch motivierten Tangentialkegel gesehen werden. Von einem zulässigen Punkt ausgehend liefern lineare Approximationen der Nebenbedingungen zulässige Richtungen.*

Bemerkung 2.25. *Numerisch ist es möglich den Kern* $\ker(\nabla_x g_i(x^*), \nabla_x h_j(x^*))$ *zum Beispiel über eine QR-Zerlegung zu bestimmen. In der praktischen Anwendung ist dies aber in der Regel nicht explizit notwendig.*

Kapitel 3

Numerische Lösung von Optimierungsproblemen

Inhaltsangabe

Die Ausführungen in diesem Kapitel dienen zur Motivation und Beschreibung der sequentiellen quadratischen Programmierung zur Lösung von nichtlinearen Optimierungsproblemen der Form 2.1. Diese Methode ist zentral für die weiteren Ergebnisse dieser Arbeit. Die SQP-Methode basiert auf der iterativen Lösung von quadratischen Unterproblemen zur Bestimmung einer Suchrichtung, um weitere Iterierte zu bestimmen. Somit ist die Lösung von quadratischen Unterproblemen essenziell für den Erfolg des gesamten Verfahrens. Deshalb wird zu Beginn des Kapitels deren Lösung thematisiert und umfassend diskutiert.
Es existieren viele ausführliche Quellen in denen die vorgestellten Algorithmen inklusive weiterer Details gefunden werden können. Für das vorliegende Kapitel wurden

die Ergebnisse aus den Büchern von Nocedal und Wright [82], Alt [1], Geiger und Kanzow [45], Luenberger [74], Spellucci [97], sowie die Arbeiten von Gould, Orban und Toint [58] und Gertz und Wright [47] geeignet zusammengetragen.

3.1 Lösung quadratischer Optimierungsprobleme

Dieser Abschnitt beschreibt verschiedene Algorithmen zur Lösung von quadratischen Problemen der Form 2.3. Die folgenden Betrachtungen zeigen, dass die Behandlung der Ungleichungsnebenbedingungen die größte Herausforderung bei der Formulierung eines Algorithmus zur Lösung derartiger Probleme darstellt. Je nach verwendetem Algorithmus ist die Lösung einer Folge von gleichungsbeschränkten quadratischen Optimierungsproblemen erforderlich.
Die Lagrange-Funktion für die quadratische Optimierungsaufgabe (2.3) sei durch

$$\mathcal{L}(x, y, z) := \frac{1}{2} x^\top Q x + c^\top x + y^\top (Ax - \beta) + z^\top (Cx - \gamma) \tag{3.1}$$

definiert. Zur Unterscheidung von den Multiplikatoren der nichtlinearen Optimierungsaufgabe (2.1) seien $y \in \mathbb{R}^{N_h}$ und $z \in \mathbb{R}^{N_g}$ als Multiplikatoren des quadratischen Optimierungsproblems eingeführt.

3.1.1 Gleichungsbeschränkte quadratische Optimierungsprobleme

Als Ausgangspunkt für die weiteren Betrachtungen soll zunächst die Lösung von lediglich durch Gleichungsnebenbedingungen beschränkten quadratischen Optimierungsproblemen vorgestellt werden. Die vereinfachte Problemformulierung ist in diesem Fall wie folgt gegeben.

$$\begin{aligned} \min_{x \in \mathbb{R}^{N_x}} \quad & q(x) := \frac{1}{2} x^\top Q x + c^\top x \\ \text{unter} \quad & Ax = \beta \end{aligned} \tag{3.2}$$

Nach Gleichung (3.1) ist

$$\mathcal{L}(x, y) := \frac{1}{2} x^\top Q x + c^\top x + y^\top (Ax - \beta) \tag{3.3}$$

die Lagrange-Funktion zur Problemformulierung (3.2). Eine Auswertung der Gleichung (2.32) der notwendigen Optimalitätsbedingungen liefert das lineare Gleichungssystem

$$\begin{pmatrix} Q & A^\top \\ A & 0 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} -c \\ \beta \end{pmatrix}. \tag{3.4}$$

Die Betrachtungen zeigen, dass die notwendigen Bedingungen eines gleichungsbeschränkten quadratischen Optimierungsproblems auf die Lösung eines linearen Gleichungssystems führen. Der folgende Satz, basierend auf zwei Aussagen aus dem Buch von Nocedal und Wright [82, cf. Lemma 16.1 und Theorem 16.2], beschreibt die theoretischen Voraussetzungen für die Lösbarkeit des Systems in Gleichung (3.4), sowie weitere Eigenschaften der Lösung der Problemstellung (3.2).

Satz 3.1. *Die Matrix $A \in \mathbb{R}^{N_h \times N_x}$ habe vollen Zeilenrang, das heißt die Nebenbedingungen seien konsistent. Weiterhin seien die Spalten der Matrix $T \in \mathbb{R}^{N_x \times (\mathbb{N}_x - \mathbb{N}_h)}$ eine Basis für den Kern von A, so dass T vollen Spaltenrang hat und $AT = 0$ erfüllt. Ferner sei die reduzierte Matrix $T^\top QT$ positiv definit.*
Dann ist die Matrix

$$\begin{pmatrix} Q & A^\top \\ A & 0 \end{pmatrix} \tag{3.5}$$

regulär und es existiert ein eindeutiges Paar (x^*, y^*), so dass Gleichung (3.4) *erfüllt ist.*
Weiterhin ist x^* globale Lösung der Problemstellung (3.2).

Beweis. Zunächst wird die Regularität der Matrix in Gleichung (3.5) gezeigt. Seien Vektoren w, v so gewählt, dass

$$\begin{pmatrix} Q & A^\top \\ A & 0 \end{pmatrix} \begin{pmatrix} w \\ v \end{pmatrix} = 0 \tag{3.6}$$

gilt. Es ist zu zeigen, dass aus dieser Annahme $w = 0$ und $v = 0$ folgt, um die Regularität zu beweisen. Aus der zweiten Zeile von Gleichung (3.6) folgt direkt

$$Aw = 0 \tag{3.7}$$

und somit ist $w \in \ker(A)$. Weiterhin lässt sich die Aussage

$$0 = \begin{pmatrix} w \\ v \end{pmatrix}^\top \begin{pmatrix} Q & A^\top \\ A & 0 \end{pmatrix} \begin{pmatrix} w \\ v \end{pmatrix} = w^\top Q w \tag{3.8}$$

aus Gleichung (3.6) ableiten. Da $w \in \ker(A)$ ist, existiert ein $u \in \mathbb{R}^{N_x \times (\mathbb{N}_x - \mathbb{N}_h)}$, so dass $w = Tu$. Somit lässt sich

$$0 = w^\top Q w = u^\top T^\top Q T u \tag{3.9}$$

folgern. Auf Grund der positiven Definitheit der reduzierten Matrix $T^\top QT$ ergibt sich $u = 0$ und daraus folgt direkt $w = 0$. Aus Gleichung (3.6) ergibt sich damit $A^\top v = 0$ und auf Grund des vollen Zeilenranges von A folgt $v = 0$, womit die Regularität der Matrix bewiesen ist. Aus der Regularität folgt direkt die eindeutige Existenz von (x^*, y^*) als Lösung von Gleichung (3.4).

Es bleibt zu zeigen, dass x^* globale Lösung der Problemstellung (3.2) ist. Sei x ein weiterer zulässiger Punkt. Dann gilt $Ax^* = Ax = \beta$ und für ein $p := x^* - x$ folgt direkt $Ap = 0$. Für die Zielfunktion in der Aufgabenstellung (3.2) ergibt sich

$$\begin{aligned} q(x) &= \frac{1}{2}(x^* - p)^\top Q(x^* - p) + c^\top(x^* - p) \\ &= \frac{1}{2}p^\top Qp - p^\top Qx^* - c^\top p + q(x^*). \end{aligned} \tag{3.10}$$

Auf Grund der notwendigen Bedingung aus Gleichung (3.4) gilt $Qx^* = -c - A^\top y^*$, so dass mit $Ap = 0$

$$p^\top Qx^* = p^\top(-c - A^\top y^*) = -p^\top c \tag{3.11}$$

gilt. Durch Einsetzen dieser Beziehung in Gleichung (3.10) folgt

$$q(x) = \frac{1}{2}p^\top Qp + q(x^*) \tag{3.12}$$

für die Zielfunktion im Punkt x. Da $p \in \ker(A)$ ist $p^\top Qp > 0$ und es muss $q(x) > q(x^*)$ gelten. Somit ist x^* globales Minimum der Aufgabenstellung (3.2). □

Der Beweis stammt ebenfalls aus dem Buch von Nocedal und Wright [82]. Auf Grund der zentralen Bedeutung von quadratischen Problemen wurde zum besseren Verständnis der weiteren Betrachtungen der Beweis detailliert dargestellt und insbesondere die Argumentationen bezüglich der Regularität der KKT-Matrix werden im weiteren Verlauf der Arbeit noch häufiger in Beweisen thematisiert.

Bemerkung 3.2. *Unter den gegeben Voraussetzungen sind die notwendigen Bedingungen auch hinreichend, da die Voraussetzungen des Satzes 2.23 durch die positive Definitheit der reduzierten Matrix gegeben sind. Insbesondere sind auch die Regularitätsbedingungen der linearen Unabhängigkeit durch die Zeilenregularität der Matrix A gegeben.*

Bemerkung 3.3. *Die Argumentationskette am Ende des Beweises zeigt weiterhin, dass die positive Definitheit der reduzierten Matrix notwendig ist. Ist $T^\top QT$ nur positiv semidefinit folgt auf Grund von Gleichung* (3.12) *der Verlust der Eindeutigkeit und es kann mehrere Minima mit demselben Zielfunktionswert geben.*

3.1.2 Ungleichungsbeschränkte quadratische Optimierungsprobleme

In praktischen Anwendungen und insbesondere im weiteren Verlauf dieser Arbeit müssen quadratische Optimierungsprobleme mit Ungleichungsbeschränkungen der Form (2.3) gelöst werden. Es gibt zwei große Gruppen zur Lösung von Problemen

dieser Art. Einerseits existieren bereits seit den 1970er Jahren [32, 53, 48] Aktive-Mengen-Methoden. Diese identifizieren die relevanten Nebenbedingungen (Zeilen der Matrix C) und lösen das resultierende gleichungsbeschränkte Problem. Andererseits werden seit Ende der 1990er Jahren [54, 104, 47] Innere-Punkte-Verfahren entwickelt. Bei diesen wird versucht, durch die Einführung eines Barriere-Terms, in der Regel mit Hilfe von Schlupfvariablen, die Ungleichungsnebenbedingungen in Gleichungsnebenbedingungen zu transformieren.
Damit sich der Leser einen hinreichenden Überblick über die Vor- und Nachteile der beiden Verfahren verschaffen kann, wird die generelle Vorgehensweise dieser Methoden in diesem Kapitel erklärt. Ein besonderer Fokus wird hierbei auf das Innere-Punkte-Verfahren gelegt, da in den späteren numerischen Betrachtungen dieser Arbeit eine Implementierung dieser Methode zur Lösung der auftretenden quadratischen Teilprobleme verwendet wird.

3.1.2.1 Aktive-Mengen-Methode

Die folgende Beschreibung der Aktive-Mengen-Methode basiert auf den Büchern von Alt [1], Nocedal und Wright [82] sowie Spellucci [97].
Für die Betrachtungen in diesem Abschnitt wird die positive Definitheit der Matrix Q auf dem Kern der aktiven Nebenbedingungen entsprechend des Satzes 2.23 angenommen. Seien mit C_j, $j \in \{1, ..., N_g\}$ die Zeilen der Nebenbedingungsmatrix C von (2.3) bezeichnet. Entsprechend der Definition 2.7 wird eine Nebenbedingung von (2.3) als aktiv bezeichnet, wenn

$$C_j x - \gamma_j = 0 \tag{3.13}$$

gilt. Wenn die Menge der aktiven Nebenbedingungen $\mathcal{A}(x^*)$ in der Minimalstelle x^* bekannt wäre, so könnte x^* durch Lösung des gleichungsbeschränkten Optimierungsproblems

$$\begin{aligned} \min_{x \in \mathbb{R}^{N_x}} \quad & q(x) := \frac{1}{2} x^\top Q x + c^\top x \\ \text{unter} \quad & Ax = \beta \\ & C_i x = \gamma_i, \ \ \forall i \in \mathcal{A}(x^*) \end{aligned} \tag{3.14}$$

bestimmt werden. Durch Kenntnis der aktiven Menge kann die Lösung eines ungleichungsbeschränkten Problems demnach auf die Lösung eines gleichungsbeschränkten Problems reduziert werden.
Das Ziel der Aktiven-Mengen-Methode ist das Auffinden der Minimalstelle x^*, wobei die Lösung einer Folge von gleichungsbeschränkten Problemen der Form (3.14) bei der Bestimmung der aktiven Menge $\mathcal{A}(x^*)$ hilft. Da während der Laufzeit die endgültige aktive Menge noch nicht bekannt ist, führen unter anderem Nocedal und Wright [82] die *Arbeitsmenge*

$$\mathcal{W}^{[k]} := \mathcal{A}(x^{[k]}) \tag{3.15}$$

ein. In der Arbeitsmenge sind alle Indizes der vorläufigen aktiven Menge enthalten. Diese wird während der Iterationen immer weiter angepasst, um letztendlich die endgültige aktive Menge zu ermitteln.
Zunächst muss ein zulässiger Startpunkt $x^{[0]}$ und die Arbeitsmenge $\mathcal{W}^{[0]}$ vorliegen. Zur Bestimmung eines zulässigen Startpunktes schlägt Alt [1] die Verwendung der Phase 1 des Simplex-Verfahrens vor. Eine Beschreibung des Simplex-Verfahrens findet sich zum Beispiel im Buch von Luenberger [74].
Ausgehend vom Startpunkt wird ein iterativer Algorithmus formuliert. Zur Vereinfachung der zu lösenden Unterprobleme definieren wir

$$p := x - x^{[k]} \tag{3.16}$$

$$\theta^{[k]} := Qx^{[k]} + c \tag{3.17}$$

sowie $r^{[k]} := \frac{1}{2}(x^{[k]})^\top Q(x^{[k]}) + c^\top x^{[k]}$. Somit ist $x = p + x^{[k]}$ und die Zielfunktion von (3.14) lässt sich durch

$$q(x) = q(x^{[k]} + p) = \frac{1}{2}p^\top Qp + (\theta^{[k]})^\top p + r^{[k]} \tag{3.18}$$

darstellen. Der Rest $r^{[k]}$ ist für eine Optimierung bezüglich p lediglich eine additive Konstante, wirkt sich somit nicht aus und kann vernachlässigt werden. Die zu bestimmende Suchrichtung $p^{[k]}$ darf sich ausschließlich im Kern der zur Zeit aktiven Nebenbedingungen bewegen, so dass die Zulässigkeit erhalten bleibt. Die aktuelle Arbeitsmenge $\mathcal{W}^{[k]}$ liefert die zu verwendenden Gleichungsnebenbedingungen. Insgesamt kann demnach durch Lösung des gleichungsbeschränkten Unterproblems

$$\begin{aligned} \min_{p\in\mathbb{R}^{N_x}} \quad & \frac{1}{2}p^\top Qp + (\theta^{[k]})^\top p \\ \text{unter} \quad & Ap = 0 \\ & C_i p = 0, \quad \forall i \in \mathcal{W}^{[k]} \end{aligned} \tag{3.19}$$

eine Suchrichtung $p^{[k]}$ bestimmt werden. Da der Punkt $x^{[k]}$ bereits zulässig war, erfüllt auf Grund der Linearität der Nebenbedingungen $x^{[k+1]} = x^{[k]} + \alpha^{[k]}p^{[k]}$ ebenfalls $C_i x^{[k+1]} - \gamma_i \leq 0, \forall i \in \mathcal{W}^{[k]}$ für jedes $\alpha^{[k]} \in \mathbb{R}$.
Für die Lösung von (3.19) können folgende Fälle auftreten:
Fall 1:
Die optimale Lösung ist $p^{[k]} = 0$ und die Multiplikatoren der aktiven Nebenbedingungen erfüllen $z_i^{[k]} \geq 0, \forall i \in \mathcal{W}^{[k]}$. Werden die Einträge des Multiplikatorenvektors $z_i^{[k]} = 0$ für $i \notin \mathcal{W}^{[k]}$ gesetzt, sind die Optimalitätsbedingungen entsprechend Satz 2.23 erfüllt mit $x^* = x^{[k]}$ und $z^* = z^{[k]}$ und $\mathcal{A}(x^*) = \mathcal{W}^{[k]}$.
Fall 2:
Die optimale Lösung ist $p^{[k]} = 0$ und es existiert mindestens ein Index i mit

$$z_i^{[k]} < 0. \tag{3.20}$$

Der negative Multiplikator zeigt an, dass durch Inaktivierung der entsprechenden Nebenbedingung eine Verringerung der Zielfunktion möglich ist. Dies kann zum Beispiel mit Hilfe der Sensitivitätsanalyse aus Kapitel 4 begründet werden. Erfüllen mehrere Indizes Gleichung (3.20) bietet der kleinste Multiplikator das größte Potential zur Verringerung der Zielfunktion. Dementsprechend sei

$$l := \arg \min_{i \in \mathcal{W}^{[k]}} z_i^{[k]}, \tag{3.21}$$

so dass die Arbeitsmenge für die nächste Iteration entsprechend der Vorschrift

$$\mathcal{W}^{[k+1]} = \mathcal{W}^{[k]} \setminus \{l\} \tag{3.22}$$

gesetzt werden kann. Dieser Schritt wird in der Literatur, zum Beispiel bei Alt [1], als *Inaktivierungsschritt* bezeichnet.

Fall 3:
Die optimale Lösung ist $p^{[k]} \neq 0$. In diesem Fall wird eine Schrittweite $\alpha^{[k]} \in (0, 1]$ bestimmt, um die nächste Iterierte entsprechend der Vorschrift

$$x^{[k+1]} = x^{[k]} + \alpha^{[k]} p^{[k]} \tag{3.23}$$

zu generieren. Entlang dieser Suchrichtung ist die Einhaltung der zur Zeit inaktiven Nebenbedingungen nicht garantiert, deshalb muss eine Liniensuche durchgeführt werden. Kann die Schrittweite $\alpha^{[k]} = 1$ akzeptiert werden, so wird $\mathcal{W}^{[k+1]} = \mathcal{W}^{[k]}$ gesetzt und der Algorithmus durch Lösung des Problems (3.19) in $x^{[k+1]}$ fortgesetzt. Muss eine Schrittweite $\alpha^{[k]} < 1$ gewählt werden, so wird die aktiv werdenden Nebenbedingung bestimmt. Eine inaktive Nebenbedingung kann nur aktiv werden, wenn

$$C_m p^{[k]} > 0 \tag{3.24}$$

gilt. Mit Hilfe der Definition

$$\alpha^{[k]} := \min \left(1, \min_{m \notin \mathcal{W}^{[k]}, C_m p^{[k]} > 0} \frac{\gamma_m - C_m x^{[k]}}{C_m p^{[k]}} \right) \tag{3.25}$$

kann die Schrittweite $\alpha^{[k]}$ bestimmt werden. Sei j der Index einer aktiv werdenden Nebenbedingung, dann ergibt sich die neue Arbeitsmenge durch

$$\mathcal{W}^{[k+1]} = \mathcal{W}^{[k]} \cup \{j\} \tag{3.26}$$

und das entstehende Problem (3.19) muss in $x^{[k+1]}$ gelöst werden. Algorithmus 1 beschreibt die gesamte Vorgehensweise erneut in kompakter Form.
Die Aktive-Mengen-Methode weist einige numerische Probleme auf, die insbesondere bei der hochdimensionalen Optimierung die Anwendbarkeit des Algorithmus einschränken. Als kritischster Punkt ist die Bestimmung der aktiven Menge zu sehen. Bereits bei der Analyse der Simplex-Methode für lineare Probleme wurde gezeigt,

Algorithmus 1 Aktive-Mengen-Strategie für konvexe QP

1: Finde zulässigen Startpunkt $x^{[0]}$
2: Wähle $\mathcal{W}^{[0]} \subset \mathcal{A}(x^{[0]})$
3: **for** $k = 0, 1, 2, ...$ **do**
4: Bestimme $p^{[k]}$ als Lösung der Aufgabe (3.19)
5: **if** $p^{[k]} = 0$ **then**
6: Berechne Multplikatoren $z_i^{[k]}, \ \forall i \in \mathcal{W}^{[k]}$ aus (3.4) unter Hinzunahme der aktiven Ungleichungsnebenbedingungen
7: **if** $z_i^{[k]} \geq 0, \ \forall i \in \mathcal{W}^{[k]}$ **then**
8: Abbruch, Lösung $x^* = x^{[k]}$, mit $z_i^{[k]} = 0, \forall i \notin \mathcal{W}^{[k]}, z^* = z^{[k]}$
9: **else**
10: $l \leftarrow \arg\min_{i \in \mathcal{W}^{[k]}} z_i^{[k]}$
11: $x^{[k+1]} \leftarrow x^{[k]}; \mathcal{W}^{[k+1]} \leftarrow \mathcal{W}^{[k]} \setminus \{l\}$
12: **end if**
13: **else**
14: Berechne $\alpha^{[k]}$ nach (3.25)
15: $x^{[k+1]} \leftarrow x^{[k]} + \alpha^{[k]} p^{[k]}$
16: **if** Wechselnde Nebenbedingung j **then**
17: $\mathcal{W}^{[k+1]} \leftarrow \mathcal{W}^{[k]} \cup \{l\}$
18: **else**
19: $\mathcal{W}^{[k+1]} \leftarrow \mathcal{W}^{[k]}$
20: **end if**
21: **end if**
22: **end for**

dass der Aufwand zur Bestimmung der korrekten aktiven Menge im schlimmsten Fall exponentiell zur Anzahl der Nebenbedingungen N_g sein kann [69], auch wenn gezeigt wurde, dass diese in der Praxis in der Regel nicht auftritt [8]. Ist N_g hinreichend groß, so ist ein derartiger Aufwand unpraktikabel und macht den hier beschriebenen Algorithmus sehr rechenintensiv für diese Situation. Die Aktive-Mengen-Methode kann zur Lösung hochdimensionaler Probleme verwendet werden, wenn eine gute Schätzung der aktiven Menge vorliegt. Zur weiteren Verbesserung der Effizienz sollten in diesem Fall spezialisierte Verfahren zur Lösung der linearen Algebra verwendet werden, welche die bereits vorhandenen Zerlegungen durch Updatestrategien an die sich ändernde aktive Menge anpassen.
Ein dimensionsunabhängiges Problem ist der Inaktivierungsschritt. Nocedal und Wright [82] kritisieren die Skalierungsabhängigkeit der Lagrange-Multiplikatoren. Durch einen Blick auf die notwendige Bedingung (2.32) wird leicht deutlich, dass die Skalierung einer Nebenbedingung mit einem Faktor $\nu \in \mathbb{R}, \nu \neq 0$ durch eine Skalierung des Multiplikators mit $\frac{1}{\nu}$ kompensiert werden kann. Somit ist der Algorithmus in dieser Form abhängig von der Skalierung, auch wenn die grundsätzliche Problemstellung nicht geändert wurde.

Auf Grund dieser Defizite wurden Innere-Punkte-Verfahren für quadratische Optimierungsprobleme entwickelt. Der folgende Abschnitt thematisiert diese.

3.1.2.2 Innere-Punkte-Verfahren

Die Aktive-Mengen-Methode kann als Erweiterung der Simplex-Methode für lineare Probleme auf quadratische Probleme betrachtet werden. Diese wird zum Beispiel im Buch von Luenberger [74] erläutert. Bei der linearen Optimierung befinden sich die lokalen Minima in den Ecken des zulässigen Bereichs (dieser ist auf Grund der Form der Nebenbedingungen ein Simplex) und diese können iterativ abgearbeitet werden. Bei der Aktiven-Mengen-Methode für quadratische Optimierungsprobleme erweitert sich dieses Verhalten auf die Ränder des zulässigen Bereichs. Diese werden mit Hilfe der im Algorithmus 1 beschriebenen Schritte abgearbeitet.
Im Kontrast zu diesen Verfahren bewegen sich die Iterierten bei einem Innere-Punkte-Verfahren im Inneren des zulässigen Bereichs und nähern sich von dort dem lokalen Minimum, welches ohne Beschränkung der Allgemeinheit am Rand des zulässigen Bereichs zu finden ist (Liegt das lokale Minimum nicht am Rand des zulässigen Bereichs, so hatten die Nebenbedingungen auf die Lösung keine Auswirkung und können ignoriert werden.). Auch Innere-Punkte-Verfahren wurden für lineare Optimierungsprobleme entwickelt. Beispielsweise in den Büchern von Luenberger [74] oder Nocedal und Wright [82] werden diese beschrieben. Für die weiteren Betrachtungen im Rahmen dieser Arbeit ist der Algorithmus von Mehrotra [79] von besonderer Bedeutung. Insbesondere lässt sich dieser auf die Lösung von quadratischen Optimierungsproblemen übertragen, weshalb im Folgenden eine detaillierte Beschreibung folgt. Die weiteren Betrachtungen basieren auf den Ausführungen im Buch von Nocedal und Wright [82], Gould, Orban und Toint [58], aber insbesondere auf dem Artikel zur Implementierung eines objekt-orientierten Lösers für quadratische Optimierungsprobleme von Gertz und Wright [47].
Ähnlich wie bei der Aktiven-Mengen-Methode wird bei diesem Innere-Punkte-Verfahren eine Folge von gleichungsbeschränkten quadratischen Optimierungsproblemen gelöst. Die hier auftretenden Hilfsprobleme weisen allerdings zusätzlich Ungleichungsbeschränkungen an die Optimierungsvariablen auf. Statt der Bestimmung einer Arbeitsmenge werden sogenannte Schlupfvariablen eingeführt, um Ungleichungsnebenbedingungen in Gleichungsnebenbedingungen zu transformieren. Sei $s \in \mathbb{R}^{N_g}$ ein Vektor mit Schlupfvariablen. Die Problemformulierung 2.3 wird mit Hilfe der Schlupfvariablen in das Innere-Punkte-QP

$$\begin{aligned} \min_{x \in \mathbb{R}^N_x} \quad & \frac{1}{2} x^\top Q x + c^\top x \\ \text{unter} \quad & Ax = \beta \\ & Cx + s = \gamma \\ & s \geq 0 \end{aligned} \tag{3.27}$$

überführt. Hierbei sei kurz auf die Ungleichungsbeschränkungen an s hingewiesen, da diese den Unterschied zu den gleichungsbeschränkten Hilfsproblemen bei Anwendung der Aktiven-Mengen-Methode darstellen. Werden auch hier die Lagrange-Multiplikatoren y für die Gleichungsnebenbedingungen und z für die Ungleichungsnebenbedingungen verwendet, sind die notwendigen Optimalitätsbedingungen erster Ordnung nach Satz 2.21 für die Problemstellung 3.27 durch

$$Qx + A^\top y + C^\top z = -c \tag{3.28}$$
$$Ax = \beta \tag{3.29}$$
$$Cx + s = \gamma \tag{3.30}$$
$$z \geq 0 \tag{3.31}$$
$$s \geq 0 \tag{3.32}$$
$$z^\top s = 0 \tag{3.33}$$

gegeben. Durch die Einführung der Schlupfvariablen lässt sich die Komplementaritätsbedingung durch Gleichung (3.33) beschreiben. Damit während des zu beschreibenden Algorithmus die Einhaltung der Komplementaritätsbedingung überwacht werden kann, wird das Komplementaritätsmaß

$$\bar{\mu} = \frac{z^\top s}{N_g}. \tag{3.34}$$

eingeführt. Durch die Division mit der Anzahl der Ungleichungsnebenbedingungen N_g entsteht ein dimensionsunabhängiges Maß. Zur Beschreibung des Algorithmus werden weiterhin die Residuen

$$r_Q = Qx + A^\top y + C^\top z + c \tag{3.35}$$
$$r_A = Ax - \beta \tag{3.36}$$
$$r_C = Cx + s - \gamma \tag{3.37}$$

definiert. Diese messen im folgenden Iterationsablauf die Verletzung der notwendigen Bedingungen (3.28) - (3.30).
Der Predictor-Corrector Algorithmus von Mehrotra ist ein primal-duales Innere-Punkte-Verfahren. Es werden neben den Bedingungen für das primale Problem auch die Kriterien für das duale Problem innerhalb des Algorithmus berücksichtigt. An dieser Stelle sind keine tieferen Einblicke in die theoretischen Hintergründe zu dualen Problemen nötig. Für den interessierten Leser sei trotzdem auf die schöne Darstellung der Theorie im Buch von Spellucci [97] hingewiesen.
Die Komplementaritätsbedingung (3.33) stellt sicher, dass sowohl primale, als auch duale Zulässigkeit am Ende des Verfahrens erfüllt sind und die Verwendung des Komplementaritätsmaßes (3.34) bindet diese direkt in die Iterationen ein.
Die Idee von Innere-Punkte-Verfahren ist, dass sich die Iterierten vom Inneren des zulässigen Bereichs über einen sogenannten zentralen Pfad dem Minimum nähern.

Zur Bestimmung der Suchrichtung entlang dieses zentralen Pfades werden die gestörten KKT-Bedingungen mit Hilfe der Definitionen

$$Z := \begin{pmatrix} z_1 & & \\ & \ddots & \\ & & z_{N_g} \end{pmatrix}, \qquad S := \begin{pmatrix} s_1 & & \\ & \ddots & \\ & & s_{N_g} \end{pmatrix} \tag{3.38}$$

und $e := (1, 1, ..., 1)^\top$ durch

$$F(x, y, z, s) = \begin{pmatrix} Qx + A^\top y + C^\top z + c \\ Ax - \beta \\ Cx + s - \gamma \\ ZSe - \sigma\bar{\mu}e \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \tag{3.39}$$

bestimmt. Der Zentrierungsparameter $\sigma \in [0, 1] \subset \mathbb{R}$ bestimmt die Lage der Lösung in Bezug auf die Ränder des zulässigen Bereichs. Werden verschiedene Lösungen der Gleichung für fallende $\sigma\bar{\mu} \to 0$ verwendet, entsteht eine Trajektorie zur optimalen Lösung [82].
Die Nullstelle von Gleichung (3.39) mit $\sigma = 0$ wird in diesem Verfahren durch Lösung des linearen Gleichungssystems

$$\begin{pmatrix} Q & A^\top & C^\top & 0 \\ A & 0 & 0 & 0 \\ C & 0 & 0 & I \\ 0 & 0 & S & Z \end{pmatrix} \begin{pmatrix} \Delta x^{aff} \\ \Delta y^{aff} \\ \Delta z^{aff} \\ \Delta s^{aff} \end{pmatrix} = \begin{pmatrix} -r_Q \\ -r_A \\ -r_C \\ -ZSe \end{pmatrix} \tag{3.40}$$

mit Hilfe des Newton-Verfahrens umgesetzt. Gleichung (3.39) beschreibt die Bedingungen, wie sie im lokalen Minimum gelten müssen. Allerdings kann während der Iterationen nicht sicher gestellt werden, dass ein solcher Schritt zum Optimum einen hinreichend Fortschritt innerhalb des zulässigen Bereichs ermöglicht. Dieser erste Schritt mit $\sigma = 0$ wird als Predictor- oder affiner Skalierungsschritt bezeichnet. Mit Hilfe der bestimmten Suchrichtungen wird jetzt anhand der Bedingung

$$\alpha^{aff} = \underset{\alpha \in (0,1]}{\arg\max} \left((z, s) + \alpha(\Delta z^{aff}, \Delta s^{aff}) \geq 0 \right) \tag{3.41}$$

eine maximale Schrittweite in dieser Richtung bestimmt, so dass die Iterierten positiv bleiben. Die Notation $(z, s) \geq 0$ bedeutet, dass alle Einträge des aus z und s zusammengesetzten Vektors größer gleich 0 sein sollen.
Bedingung (3.41) kombiniert primale und duale Zulässigkeit und kann den Fortschritt in Richtung der optimalen Lösung stark bremsen, deshalb wird zusätzlich ein Corrector-Schritt durchgeführt. Mit Hilfe der bestimmten Werte wird das genäherte Komplementaritätsmaß durch

$$\bar{\mu}^{aff} = \frac{(z + \alpha^{aff}\Delta z^{aff})^\top (s + \alpha^{aff}\Delta s^{aff})}{N_g} \tag{3.42}$$

bestimmt und mit Hilfe von Mehrotras Heuristik der Zentrierungsparameter durch

$$\sigma = \left(\frac{\bar{\mu}^{aff}}{\bar{\mu}}\right)^3 \tag{3.43}$$

festgesetzt. Es seien

$$\Delta Z^{aff} := \begin{pmatrix} \Delta z_1^{aff} & & \\ & \ddots & \\ & & \Delta z_{N_g}^{aff} \end{pmatrix}, \qquad \Delta S^{aff} := \begin{pmatrix} \Delta s_1^{aff} & & \\ & \ddots & \\ & & \Delta s_{N_g}^{aff} \end{pmatrix} \tag{3.44}$$

definiert. Durch Lösung des linearen Gleichungssystems

$$\begin{pmatrix} Q & A^\top & C^\top & 0 \\ A & 0 & 0 & 0 \\ C & 0 & 0 & I \\ 0 & 0 & S & Z \end{pmatrix} \begin{pmatrix} \Delta x \\ \Delta y \\ \Delta z \\ \Delta s \end{pmatrix} = \begin{pmatrix} -r_Q \\ -r_A \\ -r_C \\ -ZSe - \Delta Z^{aff}\Delta S^{aff}e + \sigma\bar{\mu}e \end{pmatrix} \tag{3.45}$$

unter Wiederverwendung der Zerlegung aus Gleichung (3.40) wird der Corrector-Schritt bestimmt. Zur Bestimmung der Schrittweite existieren verschiedene Ansichten. Nocedal und Wright [82] diskutieren Möglichkeiten primal und dual unterschiedliche Schrittweiten zu verwenden. Diese Vorgehensweise zeigt nach ihrer Meinung in numerischen Anwendungen teilweise schnelleres Konvergenzverhalten. Gertz und Wright [47] empfehlen zur Sicherstellung der Konvergenz die Verwendung einer Schrittweite für alle Variablen. Bei der Bestimmung des Schritts muss sichergestellt werden, dass

$$s + \alpha\Delta s \geq 0 \tag{3.46}$$

$$z + \alpha\Delta z \geq 0 \tag{3.47}$$

gilt. Dies wird umgesetzt, indem die primale und duale Schrittweite durch

$$\alpha_\tau^{pri} = \arg\max_{\alpha\in(0,1]} s + \alpha\Delta s \geq (1-\tau)s \tag{3.48}$$

$$\alpha_\tau^{dual} = \arg\max_{\alpha\in(0,1]} z + \alpha\Delta z \geq (1-\tau)z \tag{3.49}$$

ermittelt werden und die zu verwendende Schrittweite α als

$$\alpha = \min(\alpha_\tau^{pri}, \alpha_\tau^{dual}) \tag{3.50}$$

gewählt wird. Insbesondere stellen diese Vorschriften sicher, dass während der Iterationen $s^{[k]} > 0$ und $z^{[k]} > 0$ erfüllt ist. Der Parameter $\tau \in (0, 1]$ wird im Iterationsverlauf angepasst und nähert sich während der Iterationen 1.
Algorithmus 2 beschreibt in kompakter Form die einzelnen Schritte des Innere-Punkte-Verfahrens. Die Matrix in den Gleichungen (3.40) und (3.45) ist durch die

Algorithmus 2 Predictor-Corrector für konvexe QP

1: Berechne $(x^{[0]}, y^{[0]}, z^{[0]}, s^{[0]})$ mit $(z^{[0]}, s^{[0]}) > 0$
2: **for** $k = 0, 1, 2, ...$ **do**
3: Berechne $(\Delta x^{aff}, \Delta y^{aff}, \Delta z^{aff}, \Delta s^{aff})$ nach (3.40)
4: Berechne $\bar{\mu} = (z^{[k]})^\top (s^{[k]})/N_g$
5: Berechne α^{aff} nach (3.41)
6: Berechne $\bar{\mu}^{aff}$ nach (3.42)
7: Setze σ gemäß (3.43)
8: Berechne $(\Delta x, \Delta y, \Delta z, \Delta s)$ aus (3.45)
9: Wähle $\tau \in (0, 1)$ und setze α nach (3.50)
10: $(x^{[k+1]}, y^{[k+1]}, z^{[k+1]}, s^{[k+1]}) \leftarrow (x^{[k]}, y^{[k]}, z^{[k]}, s^{[k]}) + \alpha(\Delta x, \Delta y, \Delta z, \Delta s)$
11: **end for**

Einführung der Schlupfvariablen nicht mehr symmetrisch. Die folgenden Betrachtungen gelten für beide Gleichungssysteme, zur Vereinfachung sei deshalb die rechte Seite der vierten Zeile mit $r_{z,s}$ abgekürzt. Die vierte Zeile der Matrizen ergibt die Beziehung

$$\begin{aligned} S\Delta z + Z\Delta s &= -r_{z,s} \\ \Leftrightarrow Z\Delta s &= -r_{z,s} - S\Delta z. \end{aligned} \tag{3.51}$$

Es ergibt sich somit

$$\begin{aligned} C\Delta x + \Delta s &= -r_c \\ ZC\Delta x + Z\Delta s &= -Zr_c \\ \Leftrightarrow ZC\Delta x - S\Delta z &= -Zr_c + r_{z,s} \end{aligned} \tag{3.52}$$

für die dritte Zeile des Systems. Mit Hilfe dieser Identität folgt für die Matrix aus (3.40) und (3.45)

$$\begin{pmatrix} Q & A^\top & C^\top \\ A & 0 & 0 \\ ZC & 0 & S \end{pmatrix} \begin{pmatrix} \Delta x \\ \Delta y \\ \Delta z \end{pmatrix} = \begin{pmatrix} -r_Q \\ -r_A \\ -Zr_C + r_{z,s} \end{pmatrix}, \tag{3.53}$$

wobei $r_{z,s}$ passend gewählt werden muss. Die Änderung der Schlupfvariablen lässt sich während des Algorithmus auf Grund der Positivität der Matrix Z nach Berechnung von Δz aus Gleichung (3.51) über

$$\Delta s = Z^{-1}(-r_{z,s} - S\Delta z) \tag{3.54}$$

bestimmen. Mit derselben Begründung kann während der Iterationen das symmetrische System

$$\begin{pmatrix} Q & A^\top & C^\top \\ A & 0 & 0 \\ C & 0 & Z^{-1}S \end{pmatrix} \begin{pmatrix} \Delta x \\ \Delta y \\ \Delta z \end{pmatrix} = \begin{pmatrix} -r_Q \\ -r_A \\ -r_C + Z^{-1}r_{z,s} \end{pmatrix}, \tag{3.55}$$

verwendet werden.
Boxschranken an die Optimierungsvariablen der Form

$$x_i \leq U_i, \qquad i \in 1, ..., N_x$$

können besonders effizient in das Verfahren integriert werden, ohne die Dimension der zu lösenden Gleichungssysteme zu erhöhen. Seien N_{box} derartige Nebenbedingungen gefordert. Mit $\bar{z}$ seien die dazugehörigen Multiplikatoren und mit $\bar{s}$ passende Schlupfvariablen bezeichnet. Das Residuum für die rechten Seiten von (3.40) und (3.45) sei $r_{\bar{z}}$ für die Zulässigkeitsbedingung der Nebenbedingung und $r_{\bar{s}}$ für die entsprechende Komplementaritätsbedingung. Die Ableitungsmatrix der Boxschranken sei C_x. Diese enthält für eine Boxschranke an x_i eine 1 in der i-ten Spalte und ist ansonsten 0. Es ergibt sich das erweiterte System ausgehend von Gleichung (3.55)

$$\begin{pmatrix} Q & A^\top & C^\top & C_x^\top & 0 \\ A & 0 & 0 & 0 & 0 \\ C & 0 & -Z^{-1}S & 0 & 0 \\ C_x & 0 & 0 & 0 & I_{N_{\text{box}}} \\ 0 & 0 & 0 & \bar{S} & \bar{Z} \end{pmatrix} \begin{pmatrix} \Delta x \\ \Delta y \\ \Delta z \\ \Delta \bar{z} \\ \Delta \bar{s} \end{pmatrix} = \begin{pmatrix} -r_Q \\ -r_A \\ -r_C + Z^{-1} r_{z,s} \\ -r_{\bar{z}} \\ -r_{\bar{s}} \end{pmatrix}, \tag{3.56}$$

hier sind wie bereits zuvor $\bar{S}, \bar{Z}$ Diagonalmatrizen mit den Einträgen der Vektoren $\bar{s}, \bar{z}$ auf der Diagonalen. Zur Vereinfachung des Systems können die neu entstandenen Zeilen direkt aufgelöst und in die bestehende Struktur integriert werden. Die vierte Zeile liefert

$$\Delta \bar{s} = -(r_{\bar{z}} + C_x \Delta x)$$

und eingesetzt in die fünfte Zeile folgt

$$\begin{aligned} \bar{S}\Delta\bar{z} - \bar{Z}(r_{\bar{z}} + C_x\Delta x) &= -r_{\bar{s}} \\ \Leftrightarrow \Delta\bar{z} &= \bar{S}^{-1}(-r_{\bar{s}} + \bar{Z}r_{\bar{z}}) + \bar{S}^{-1}\bar{Z}C_x\Delta x. \end{aligned} \tag{3.57}$$

Um $\Delta\bar{z}$ und $\Delta\bar{s}$ komplett aus dem Gleichungssystem zu entfernen muss die erste Gleichung noch entsprechend angepasst werden. Einsetzen von (3.57) ergibt

$$Q\Delta x + A^\top \Delta y + C^\top \Delta z + C_x^\top \Delta\bar{z} = -r_Q \tag{3.58}$$

$$\begin{aligned} &Q\Delta x + A^\top \Delta y + C^\top \Delta z \quad + \\ \Leftrightarrow\ &C_x^\top(\bar{S}^{-1}(-r_{\bar{s}} + \bar{Z}r_{\bar{z}}) + \bar{S}^{-1}\bar{Z}C_x\Delta x) = -r_Q \end{aligned} \tag{3.59}$$

$$\Leftrightarrow (Q + C_x^\top \bar{S}^{-1}\bar{Z}C_x)\Delta x + A^\top\Delta y + C^\top\Delta z = -r_Q - C_x^\top(\bar{S}^{-1}(-r_{\bar{s}} + \bar{Z}r_{\bar{z}})), \tag{3.60}$$

so dass mit den Definitionen

$$\begin{aligned} \bar{r}_Q &:= r_Q + C_x^\top(\bar{S}^{-1}(-r_{\bar{s}} + \bar{Z}r_{\bar{z}})) \\ \bar{Q} &:= Q + C_x^\top \bar{S}^{-1}\bar{Z}C_x \end{aligned} \tag{3.61}$$

das symmetrische System mit integrierten Boxschranken

$$\begin{pmatrix} \bar{Q} & A^\top & C^\top \\ A & 0 & 0 \\ C & 0 & -Z^{-1}S \end{pmatrix} \begin{pmatrix} \Delta x \\ \Delta y \\ \Delta z \end{pmatrix} = \begin{pmatrix} -\bar{r}_Q \\ -r_A \\ -r_C + Z^{-1}r_{z,s} \end{pmatrix} \tag{3.62}$$

folgt. Aufgrund der Division durch $\bar{s}$ innerhalb von $\bar{Q}$ sollte $\bar{s}_i > \epsilon_s$ mit $\epsilon_s > 0$ gesetzt werden, um numerische Probleme zu vermeiden. In der Praxis hat sich $\epsilon_s := 10^{-3}$ in diesem Zusammenhang bewährt.
Das Innere-Punkte-Verfahren vermeidet mit Hilfe der beschriebenen Vorgehensweise die teilweise aufwendige Bestimmung der aktiven Menge. Die auftretenden Unterprobleme sind deshalb im Gegensatz zur Aktiven-Mengen-Strategie größer, da in jeder Iteration alle Nebenbedingungen in der zu zerlegenden Matrix enthalten sind. Dafür entfällt die Notwendigkeit effektive Update-Techniken zur Anpassung der Zerlegung im Laufe des Verfahrens zu verwenden.
Wird eine Folge von ähnlichen quadratischen Optimierungsproblemen gelöst, bietet die Aktive-Mengen-Strategie den großen Vorteil, dass mit Hilfe der Lösung der vorherigen Probleme bereits gute Startschätzungen, insbesondere der Arbeitsmenge, vorliegen und deshalb in der Regel nur wenige Iterationen zur Lösung des nächsten Problems nötig sind. Im Gegensatz dazu wird das Innere-Punkte-Verfahren bei Änderungen der Nebenbedingungen in den ersten Iterationen zunächst Schritte in den zulässigen Bereich durchführen und kann dabei bereits erzielte Fortschritte zunichte machen. Mit der starken Verbreitung von Innere-Punkte-Verfahren für allgemeine nichtlineare Probleme hat auch das Interesse an sogenannten Warmstart-Techniken in den letzten Jahren zugenommen. Einen Einblick in das Thema gewährt der Artikel von Gondzio und Grothey [55] und die dort zitierte Literatur.
Auf Grund der Effizienz des Algorithmus wurde das hier beschrieben Innere-Punkte-Verfahren inklusive einer vereinfachten Warmstarttechnik[1] basierend auf den erwähnten Ideen von Gondzio und Grothey in dem Softwarepaket QPSOL [46] für den Löser WORHP implementiert und wird in den späteren numerischen Untersuchungen als Löser für die auftretenden quadratischen Unterprobleme verwendet.

3.2 Lösung nichtlinearer Optimierungsprobleme

Optimierungsprobleme mit nichtlinearer Zielfunktion und nichtlinearen Nebenbedingungen der Form (2.1) beschreiben die allgemeinste Problemformulierung unter den gegebenen Glattheitsbedingungen. Die Lösung derartiger Optimierungsprobleme wird bereits seit vielen Jahren erforscht. Die sequentielle quadratische Programmierung basiert auf den Ideen von Wilson [110] von 1963. Diese wurde in verschiedenen bekannten und weit verbreiteten Optimierungspaketen wie beispielsweise

[1] Die vereinfachte Warmstarttechnik besteht darin, dass während eines Optimierungslaufes die primalen und dualen Variablen sowie die Schlupfvariablen geeignet abgespeichert werden.

SNOPT [49], FilterSQP [87] oder WORHP [13] erweitert und umgesetzt.
In den letzten Jahren wurde die Forschung im Bereich der nichtlinearen Optimierung in erster Linie auf Innere-Punkte-Verfahren zur direkten Lösung von Problemen der Art (2.1) fokussiert. Unter anderem in den Optimierungspaketen IPOPT [107], Knitro [14], IPFilter [101] oder LOQO [104] wurden diese implementiert.
Diese Arbeit stellt verschiedene Ansätze und Algorithmen zur Effizienzsteigerung nichtlinearer Optimierung am Beispiel der Optimierungssoftware WORHP vor. Wie bereits erwähnt, verwendet WORHP die SQP-Methode, weshalb diese im weiteren Verlauf des Abschnitts detailliert dargestellt wird.
Zunächst wird das lokale SQP-Verfahren inklusive seiner Konvergenzeigenschaften diskutiert. Anschließend wird ausführlich auf notwendige Globalisierungsmaßnahmen eingegangen, so dass letztendlich die Formulierung des Algorithmus des globalen SQP-Verfahrens erfolgt.

3.2.1 Das lokale SQP-Verfahren

Die bisherigen Betrachtungen liefern alle Grundlagen, um die Lösung von nichtlinearen Optimierungsproblemen mit nichtlinearen Nebenbedingungen der Form (2.1) mit Hilfe der sequentiellen quadratischen Programmierung zu beschreiben. Im Folgenden wird die Vorgehensweise der SQP-Methode motiviert. Das zentrale Resultat des Abschnitts liefert eine Konvergenzaussage für das Verfahren.
Es werden zunächst die KKT-Bedingungen für ein gleichheitsbeschränktes nichtlineares Optimierungsproblem betrachtet und der Zusammenhang zur quadratischen Näherung des Ausgangsproblems aufgezeigt. Dieser Argumentation folgend werden die Betrachtungen auf ungleichungsbeschränkte Probleme übertragen. Danach folgt eine Diskussion der lokalen Konvergenzeigenschaften. Die folgenden Ausführungen orientieren sich eng an der Argumentationskette aus dem Buch von Geiger und Kanzow [45]. Alternative Beschreibungen des Algorithmus finden sich beispielsweise in den Büchern von Nocedal und Wright [82], Gill, Murray und Wright [50] oder Alt [1].
Es sei zunächst das gleichheitsbeschränkte nichtlineare Optimierungsproblem

$$\begin{aligned} \min_{x \in \mathbb{R}^{N_x}} \quad & f(x) \\ \text{unter} \quad & h(x) = 0 \end{aligned} \tag{3.63}$$

gegeben. Die Lagrange-Funktion für das Problem (3.63)

$$L(x, \lambda) = f(x) + \mu^\top h(x) \tag{3.64}$$

liefert die notwendigen Bedingungen

$$\Phi(x, \mu) \coloneqq \begin{pmatrix} \nabla_x L(x, \mu) \\ h(x) \end{pmatrix} = 0. \tag{3.65}$$

Durch Anwendung des Newton-Verfahrens auf die nichtlineare Gleichung (3.65) kann ein KKT-Punkt (x^*, μ^*) gefunden werden. Die Iterierten werden unter Verwendung der Jacobi-Matrix $\Phi'(x, \mu)$ berechnet. Mit Hilfe der Vorschriften

$$\begin{aligned} x^{[k+1]} &:= x^{[k]} + d^{[k]} \\ \mu^{[k+1]} &:= \mu^{[k]} + \Delta\mu^{[k]} \end{aligned} \tag{3.66}$$

können durch Verwendung der Lösung $(d^{[k]}, \Delta\mu^{[k]})$ des linearen Gleichungssystems

$$\Phi'(x^{[k]}, \mu^{[k]}) \begin{pmatrix} d^{[k]} \\ \Delta\mu^{[k]} \end{pmatrix} = -\Phi(x^{[k]}, \mu^{[k]}) \tag{3.67}$$

neue Iterierte bestimmt werden. Das Gleichungssystem (3.67) ergibt ausmultipliziert die Gleichungen

$$\begin{aligned} \nabla^2_{xx} L(x^{[k]}, \mu^{[k]}) d^{[k]} + \nabla_x h(x^{[k]})^\top \Delta\mu^{[k]} &= -\nabla_x L(x^{[k]}, \mu^{[k]}) \\ \nabla_x h(x^{[k]})^\top d^{[k]} &= -h(x^{[k]}) \end{aligned} \tag{3.68}$$

und vereinfacht sich mit Gleichung (3.66) zu

$$\begin{aligned} \nabla^2_{xx} L(x^{[k]}, \mu^{[k]}) d^{[k]} + \nabla_x h(x^{[k]})^\top \mu^{[k+1]} &= -\nabla_x f(x^{[k]}) \\ \nabla_x h(x^{[k]})^\top d^{[k]} &= -h(x^{[k]}). \end{aligned} \tag{3.69}$$

Für die primalen Variablen wird lediglich eine Suchrichtung $d^{[k]}$ bestimmt, wohingegen für die dualen Variablen direkt der Wert der nächsten Iterierten $\mu^{[k+1]}$ berechnet wird. Bezugnehmend auf die Betrachtungen zu gleichungsbeschränkten Optimierungsproblemen im Abschnitt 3.1.1 zeigt sich, dass die Bedingungen (3.69) die notwendigen Bedingungen für das gleichungsbeschränkte quadratische Optimierungsproblem

$$\begin{aligned} \min_{d \in \mathbb{R}^{N_x}} \quad & \frac{1}{2} d^\top \nabla^2_{xx} L(x^{[k]}, \mu^{[k]}) d + \nabla_x f(x^{[k]})^\top d \\ \text{unter} \quad & h(x^{[k]}) + \nabla_x h(x^{[k]})^\top d = 0 \end{aligned} \tag{3.70}$$

darstellen. Somit entspricht die Anwendung des Newton-Verfahrens auf die notwendigen Bedingungen des Problems (3.63) der iterativen Lösung von gleichungsbeschränkten quadratischen Optimierungsproblemen. Durch Übertragung dieser Vorgehensweise auf allgemeine ungleichungsbeschränkte nichtlineare Optimierungsprobleme der Form (2.1) ergibt sich die iterative Lösung von quadratischen Unterproblemen der Form

$$\begin{aligned} \min_{d \in \mathbb{R}^{N_x}} \quad & \tfrac{1}{2} d^\top \nabla^2_{xx} L(x^{[k]}, \lambda^{[k]}, \mu^{[k]}) d + \nabla_x f(x^{[k]})^\top d \\ \text{unter} \quad & g(x^{[k]}) + \nabla_x g(x^{[k]})^\top d \leq 0 \\ & h(x^{[k]}) + \nabla_x h(x^{[k]})^\top d = 0 \end{aligned} \tag{3.71}$$

zur Bestimmung neuer Iterierter. Für die dualen Variablen λ zu den Ungleichungsnebenbedingungen ergeben sich die neuen Iterierten $\lambda^{[k+1]}$ analog zu den dualen

Algorithmus 3 Lokales SQP-Verfahren

1: Wähle Startschätzung $(x^{[0]}, \lambda^{[0]}, \mu^{[0]})$
2: **for** $k = 0, 1, 2, ...$ **do**
3: Erfüllt $(x^{[k]}, \lambda^{[k]}, \mu^{[k]})$ die KKT-Bedingungen für (2.1), Abbruch
4: Löse das Unterproblem (3.71) zur Bestimmung von $(x^{[k+1]}, \lambda^{[k+1]}, \mu^{[k+1]})$
5: **end for**

Variablen der Gleichungsnebenbedingungen μ direkt aus dem Unterproblem. Bei dieser Vorgehensweise wird sequentiell eine Folge von quadratischen Unterproblemen gelöst. Deshalb wird das Verfahren auch als SQP-Verfahren (sequentielle quadratische Programmierung) bezeichnet. Algorithmus 3 fasst die einzelnen Schritte kompakt zusammen.

Bemerkung 3.4. *Es kann nicht garantiert werden, dass die auftretenden Teilprobleme innerhalb des SQP-Algorithmus die Voraussetzungen von Satz 3.1 immer erfüllen. Geiger und Kanzow [45] verlangen deshalb, falls mehrere KKT-Punkte des Unterproblems existieren, dass der KKT-Punkt $(x^{[k+1]}, \lambda^{[k+1]}, \mu^{[k+1]})$ mit minimalem Abstand zur letzten Iterierten $(x^{[k]}, \lambda^{[k]}, \mu^{[k]})$ gewählt wird.*

Unter diesen Voraussetzungen beweisen Sie den folgenden Konvergenzsatz für das lokale SQP-Verfahren (Vergleiche Satz 5.31 in [45]).

Satz 3.5 (Konvergenz des lokalen SQP-Verfahrens)**.** *Sei $(x^*, \lambda^*, \mu^*) \in \mathbb{R}^{N_x \times N_g \times N_h}$ ein KKT-Punkt des Problems* (2.1)*, der die strikte Komplementaritätsbedingung nach Definition 2.22 erfüllt. Weiterhin seien die Regularitätsbedingungen der linearen Unabhängigkeit nach Definition 2.19 und die hinreichende Optimalitätsbedingung zweiter Ordnung nach Satz 2.23 erfüllt.*
Dann existiert ein $\epsilon > 0$ und eine Umgebung $U_\epsilon(x^*, \lambda^*, \mu^*)$ mit Radius ϵ um den KKT-Punkt, so dass für jeden Startvektor $(x^{[0]}, \lambda^{[0]}, \mu^{[0]}) \in U_\epsilon(x^*, \lambda^*, \mu^*)$ und jede mit Hilfe des Algorithmus 3 erzeugte Folge $\{(x^{[k]}, \lambda^{[k]}, \mu^{[k]})\}$ unter Verwendung der Abstandsbedingung aus Bemerkung 3.4 die folgenden Aussagen gelten:

i. Das lokale SQP-Verfahren 3 ist wohldefiniert und die Folge $\{(x^{[k]}, \lambda^{[k]}, \mu^{[k]})\}$ konvergiert gegen den KKT-Punkt (x^*, λ^*, μ^*).

ii. Das Verfahren konvergiert superlinear.

iii. Sind die zweiten Ableitungen $\nabla^2_{xx} f, \nabla^2_{xx} g$ und $\nabla^2_{xx} h$ lokal Lipschitz-stetig, so ist die Konvergenzrate quadratisch.

Beweis. Die grundsätzliche Beweisidee ist es, die durch das SQP-Verfahren erzeugte Folge in Beziehung zur Nullstellensuche mit Hilfe des Newton-Verfahrens für die Funktion

$$\Phi(x, \lambda, \mu) := \begin{pmatrix} \nabla_x L(x, \lambda, \mu) \\ h(x) \\ \min\{-g(x), \lambda\} \end{pmatrix} \tag{3.72}$$

zu setzen. Eine Nullstelle von $\Phi(x, \lambda, \mu)$ erfüllt gerade die KKT-Bedingungen für das Problem (2.1) und ist somit ein KKT-Punkt.
Die lokal gültige stetige Differenzierbarkeit von $\Phi(x, \lambda, \mu)$ kann mit Hilfe der strikten Komplementarität gezeigt werden. Die weiteren Beweisschritte sind eher technischer Natur und der Beweis wird über die Eigenschaften des Newton-Verfahrens, welche sich auf die vom SQP-Verfahren erzeugte Folge übertragen, geführt. Die Details finden sich im Buch von Geiger und Kanzow [45]. □

Das beschriebene lokale SQP-Verfahren setzt für den theoretischen Konvergenzbeweis die Lage des Startvektors $(x^{[0]}, \lambda^{[0]}, \mu^{[0]})$ innerhalb einer nicht näher beschriebenen Umgebung mit Radius ϵ um das lokale Minimum voraus. Eine solche Wahl kann in der Praxis nicht garantiert werden. Der folgende Abschnitt thematisiert deshalb die Globalisierung des SQP-Verfahrens.

3.2.2 Globalisierungsmaßnahmen

Der Beweis des Konvergenzsatzes für das lokale SQP-Verfahren (Satz 3.5) basiert auf der Rückführung des Algorithmus auf das Newton-Verfahren. Mit diesem Hintergrund erbt das SQP-Verfahren die lediglich lokalen Konvergenzeigenschaften des Newton-Verfahrens. Analog zum globalen Newton-Verfahren muss der Algorithmus zur Sicherstellung der globalen Konvergenz angepasst werden. Es ist an dieser Stelle wichtig anzumerken, dass globale Konvergenz lediglich bedeutet, dass von einem beliebigen Startpunkt ein lokales Minimum gefunden werden kann (welches allerdings nicht zwangsläufig das globale Minimum der Aufgabenstellung ist).
Die Erweiterung des Konvergenzradius des Verfahrens ähnelt der Globalisierung des Newton-Verfahrens. Zur Bestimmung der neuen Iterierten $x^{[k+1]}$ darf nicht automatisch der volle Schritt $d^{[k]}$ durchgeführt werden. Stattdessen muss dieser je nach Situation passend verkürzt werden.
Zur Bewertung der Qualität des Schritts können beispielsweise sogenannte *Bewertungsfunktionen* verwendet werden. In der unbeschränkten Optimierung kann als Bewertungsfunktion direkt die Zielfunktion eingesetzt werden. Im Gegensatz dazu müssen bei der Optimierung unter Nebenbedingungen die in der Regel gegensätzlichen Ziele der Minimierung der Zielfunktion und der Einhaltung der Nebenbedingungen Berücksichtigung finden. Eine geeignete Kombination der beiden Ziele vereint diese in einer Bewertungsfunktion. Mit einem Gewichtungsparameter $0 < \eta \in \mathbb{R}$ ist eine Bewertungsfunktion $P(x, \lambda, \mu; \eta)$ eine skalare Abbildung der Form $P : \mathbb{R}^{N_x \times N_g \times N_h} \to \mathbb{R}$. Diese kombiniert die Zielfunktion und die Nebenbedingungen geeignet zu einer skalaren Größe, welche einen Schritt bewertet.
In der Literatur finden sich zwei grundsätzliche Methoden zur Globalisierung. Einerseits werden Trust-Region-Methoden vorgeschlagen. Bei diesen Verfahren wird unter Verwendung eines Vertrauensradius $0 < \tau_{\mathrm{TR}} \in \mathbb{R}$ durch Einführung einer weiteren Nebenbedingung

$$\|d^{[k]}\|_\infty \leq \tau_{\mathrm{TR}}^{[k]}$$

in den quadratischen Unterproblemen die Gültigkeit der quadratischen Näherung an das Originalproblem sichergestellt. Diese schränkt durch geeignete Wahl des Vertrauensradius die Größe der resultierenden Suchrichtung ein. Nachdem die Suchrichtung $d^{[k]}$ bestimmt wurde, wird anhand einer geeigneten Modellfunktion, typischerweise basierend auf einer Taylor-Approximation zweiter Ordnung der beteiligten Problemfunktionen, eine Voraussage für die Verringerung der Bewertungsfunktion berechnet. Dieser Näherungswert wird mit dem Wert der Bewertungsfunktion basierend auf den echten Problemfunktionen verglichen. Der Schritt wird anschließend bei guter Übereinstimmung der Werte akzeptiert. Bei unzureichender Übereinstimmung der Werte muss der Vertrauensradius verringert werden und erneut ein quadratisches Unterproblem gelöst werden.
Auf Grund dieser Vorgehensweise wird $\tau_{\mathrm{TR}}^{[k]}$ als Vertrauensradius für die Gültigkeit der quadratischen Näherung bezeichnet. In dem Buch von Conn, Gould und Toint [22] finden sich ausführliche Beschreibungen und diverse Algorithmen zum Thema Trust-Region-Methoden.
Neben diesen Methoden wurden andererseits sogenannte Liniensuch-Verfahren als alternativer Ansatz entwickelt. Wegen der zentralen Bedeutung für die späteren Untersuchungen werden zwei unterschiedliche Verfahren zur Durchführung der Liniensuche vorgestellt. Einerseits existieren auch bei der Liniensuche Verfahren, welche eine Bewertungsfunktion verwenden. Diese wird ausgenutzt, um in Iteration k entlang der Suchrichtung $d^{[k]}$ eine Schrittweite $\alpha^{[k]}$ zu bestimmen, so dass ein hinreichender Abstieg für die neue Iterierte

$$x^{[k+1]} = x^{[k]} + \alpha^{[k]} d^{[k]} \tag{3.73}$$

garantiert werden kann. In vielen Büchern über nichtlineare beschränkte Optimierung finden sich Inhalte über Bewertungsfunktionen, so zum Beispiel in den Büchern von Geiger und Kanzow [45], Nocedal und Wright [82] oder Alt [1].
Neben den verschiedenen Bewertungsfunktionen wurden seit Ende der 1990er Jahre sogenannte Filter-Verfahren entwickelt. Bei diesen werden die beiden Ziele, Minimierung der Zielfunktion und Einhaltung der Nebenbedingungen, entkoppelt und getrennt bewertet. Zur Bestimmung der Schrittweite $\alpha^{[k]}$ ist eine hinreichende Verringerung eines der beiden Kriterien ausreichend. Die Idee wurde erstmals in dem Artikel von Fletcher und Leyffer [36] vorgestellt. Eine Untersuchung der lokalen Konvergenzeigenschaften, und insbesondere der Vermeidung des Maratos-Effektes[2] [76], wurde von S. Ulbrich [102] und von Wächter und Biegler [105] durchgeführt. Die globalen Konvergenzeigenschaften ihres Algorithmus untersuchen Wächter und Biegler in ihrem Artikel [106]. Eine Übertragung des Filterverfahrens auf das SQP-Verfahren WORHP wird in der Arbeit von Kemper [68] beschrieben.

[2]Als Maratos-Effekt wird beispielsweise von Nocedal und Wright [82] das Problem bezeichnet, dass Suchrichtungen, die einen guten Fortschritt in Richtung eines lokalen Minimums erreichen würden, von der verwendeten Bewertungsfunktion oder dem Filter abgelehnt werden und so das Gesamtverfahren stark bremsen.

3.2.2.1 Bewertungsfunktion

Während der Liniensuche muss die Qualität neuer Iterierter bezüglich Einhaltung der Nebenbedingungen und Minimierung der Zielfunktion bewertet werden. Durch Kombination beider Kriterien wird durch eine Bewertungsfunktion ein geeignetes skalares Maß für den Fortschritt der Iterierten $x^{[k]}$ während der Optimierung bestimmt.
Allgemein lassen sich Bewertungsfunktionen mit einem Gewichtungsparameter $0 < \eta \in \mathbb{R}$ in der Form

$$P(x, \lambda, \mu; \eta) \coloneqq f(x) + \eta r(x, \lambda, \mu) \tag{3.74}$$

konstruieren, wobei $r(x, \lambda, \mu) : \mathbb{R}^{N_x \times N_h \times N_g} \to \mathbb{R}$ eine stetige Funktion ist, die genau dann $r(x, \lambda, \mu) = 0$ erfüllt, wenn x zulässig bezüglich der Ausgangsnebenbedingungen ist. Weiterhin muss $r(x, \lambda, \mu)$ nicht explizit von den Multiplikatoren abhängen (Die Variablen wurden an dieser Stelle nur für eine später auftretende Bewertungsfunktion aufgenommen).
Es stellt sich die Frage, für welche Werte von η ein Minimum der Bewertungsfunktion einem Minimum der beschränkten Optimierungsaufgabe entspricht. Die folgende Definition hilft dies zu präzisieren.

Definition 3.6. *Eine Bewertungsfunktion $P(x; \eta)$ der Form* (3.74) *heißt* exakt *in einem lokalen Minimum x^* des nichtlinearen Optimierungsproblems* (2.1)*, falls es einen endlichen Parameter $\eta_0 > 0$ gibt, so dass x^* ein lokales Minimum von $P(x; \eta)$ für alle $\eta \geq \eta_0$ ist.*

Mit Hilfe von exakten Bewertungsfunktionen kann ein beschränktes Optimierungsproblem auf ein unbeschränktes Problem zurückgeführt werden. Allerdings muss der Parameter η iterativ angepasst werden, so dass eine Folge von unbeschränkten Problemen gelöst werden muss. Der folgende Satz stellt eine Gruppe von exakten Bewertungsfunktionen vor.

Satz 3.7. *Sei x^* ein isoliertes Minimum der nichtlinearen Optimierungsaufgabe* (2.1)*, welches die Regularitätsbedingung von Mangasarian-Fromowitz aus Definition 2.19 erfüllt. Dann sind die Bewertungsfunktionen*

$$l_p(x; \eta) \coloneqq f(x) + \eta \left(\sum_{i=1}^{N_h} |h_i(x)|^p + \sum_{i=1}^{N_g} (\max\{0, g_i(x)\})^p \right)^{1/p}, \tag{3.75}$$

sowie

$$l_\infty(x; \eta) \coloneqq f(x) + \eta \max\{0, |h_1(x)|, ..., |h_{N_h}(x)|, g_1(x), ..., g_{N_g}(x)\} \tag{3.76}$$

für $1 \leq p \leq \infty$ exakt.

Beweis. Geiger und Kanzow [45] beweisen den Satz und verweisen auf Han und Mangasarian [62], die den Beweis als Erste führten. □

Auf Grund der zentralen Rolle der Lagrange-Funktion innerhalb des Optimierungsprozesses untersuchten unter anderem Powell [85] und Schittkowski [91] die Verwendung der sogenannten *erweiterten Lagrange-Funktion* $L_a(x, \lambda, \mu; \eta)$

$$L_a(x, \lambda, \mu; \eta) := f(x) + \sum_{i=1}^{N_h} \mu_i h_i(x) + \frac{\eta}{2} \sum_{i=1}^{N_h} h_i^2(x) + \frac{1}{2} \sum_{i=1}^{N_g} \frac{1}{\eta} \left(\left(\max\{0, \lambda_i + \eta g_i(x)\} \right)^2 - \lambda_i^2 \right)$$

als Bewertungsfunktion. Schittkowski zielte insbesondere darauf ab sowohl den Maratos-Effekt [76], als auch zyklisches Verhalten bei der Verwendung einer Bestrafungsfunktion und ungeschickter Wahl der Gewichte in Abhängigkeit der aktuellen Lagrange-Multiplikatoren, welches von Chamberlain [16] beschrieben wurde, zu vermeiden.
Zur Bestimmung der Schrittweite wird eine Bewertungsfunktion wie in Gleichung (3.74) verwendet, um die Funktion

$$\varphi(\alpha^{[k]}) := P(x^{[k]} + \alpha d^{[k]}, \lambda^{[k]}, \mu^{[k]}; \eta^{[k]}) \tag{3.77}$$

zu definieren. Die Suchrichtung $d^{[k]}$ wurde im Unterproblem bestimmt. Unter gewissen Voraussetzungen können Geiger und Kanzow [45] am Beispiel einer l_1-Bewertungsfunktion zeigen, dass es sich um eine Abstiegsrichtung für diese handelt. Deshalb sollte ein $\alpha \in (0, 1]$ existieren, so dass

$$\varphi(\alpha^{[k]}) < \varphi(0) \tag{3.78}$$

erfüllt ist. Hierbei entspricht $\varphi(0)$ dem Wert der Bewertungsfunktion in der alten Iterierten und $\varphi(\alpha^{[k]})$ dem Wert in der neu zu bestimmenden Iterierten. Das Ziel der Liniensuche ist $\alpha^{[k]}$ so zu bestimmen, dass $\varphi(\alpha^{[k]})$ minimal wird. Der numerische Aufwand für eine exakte Liniensuche, das heißt die Bestimmung des exakten lokalen Minimums von $\varphi(\alpha^{[k]})$, ist zu hoch, da dies bedeuten würde in jeder Iteration des Optimierungsverfahrens ein weiteres Optimierungsproblem zu lösen. Stattdessen wird mit Hilfe der Armijo-Regel [3] eine passende Schrittweite bestimmt, indem das Problem nur inexakt gelöst wird.
Es seien Parameter $\alpha_{\max}^{[k]} \in (0, 1], \beta_\alpha \in (0, 1)$ und $\sigma \in (0, 1)$ gegeben, dann ist

$$\alpha_j^{[k]} := \{\alpha_{\max}^{[k]} \beta_\alpha^j \mid j = 0, 1, 2, ...\} \tag{3.79}$$

eine Folge von Testschrittweiten. Es wird iterativ die Armijo-Bedingung

$$\varphi(\alpha_j^{[k]}) \leq \varphi(0) + \sigma \alpha_j^{[k]} \varphi'(0) \tag{3.80}$$

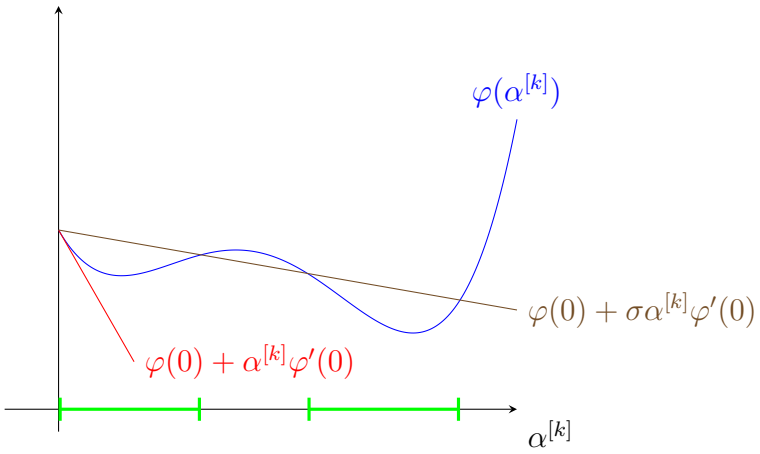


Abbildung 3.1: Akzeptable Schrittweiten bei Verwendung der Armijo-Bedingung (3.80)

ausgewertet. Da es sich bei der Suchrichtung um eine Abstiegsrichtung für die Bewertungsfunktion handelt, ist $\varphi'(0) < 0$, so dass aus der Armijo-Bedingung (3.80) das Akzeptanzkriterium für die Schrittweite (3.78) folgt und die gefundene Schrittweite somit akzeptiert werden kann. Abbildung 3.1 veranschaulicht die Bedingung anhand eines beispielhaften Verlaufes für $\varphi(\alpha^{[k]})$. Um die Anzahl der Testschrittweiten einzuschränken, kann zusätzlich noch eine untere Schranke für die Schrittweite $\alpha_{\min} \in (0, 1]$ definiert werden.
Neben der Armijo-Bedingung existieren noch verschiedene andere Bedingungen zur Verifizierung der Schrittweiten innerhalb der Liniensuche. Die Armijo-Regel mit Aufweitung (Vergleiche Kosmol [70]) erlaubt auch Schrittweiten $\alpha^{[k]} > 1$, damit der Algorithmus stärker von besonders guten Suchrichtungen profitieren kann. Eine Erweiterung der Armijo-Regel stellt die sogenannte Wolfe-Powell-Regel [111, 112] dar. Diese verwendet zusätzlich Ableitungsbedingungen und wird infolgedessen strenger. Durch

$$\varphi(\alpha^{[k]}) \leq \varphi(0) + \sigma\alpha^{[k]}\varphi'(0) \tag{3.81}$$
$$\varphi'(\alpha^{[k]}) \geq \rho\varphi'(0) \tag{3.82}$$

sind die Bedingungen der Wolfe-Powell-Regel gegeben. Ungleichung (3.82) soll sicherstellen, dass die nächste Iterierte nicht in Bereichen liegt, in denen die Bewertungsfunktion wieder stark ansteigend ist. Auf Grund der zusätzlichen Ableitungsinformationen wird die Durchführung der Liniensuche bei Verwendung der Wolfe-Powell-Regel aufwendiger, so dass in den numerischen Untersuchungen innerhalb dieser Arbeit lediglich die einfache Armijo-Regel verwendet wird.

3.2.2.2 Filterverfahren

Als Alternative zur Verwendung einer Bewertungsfunktion während der Liniensuche wurden Filterverfahren entwickelt. Im Gegensatz zu den Betrachtungen zur Bewertungsfunktion im Abschnitt 3.2.2.1 werden die häufig gegensätzlichen Ziele der Minimierung der Zielfunktion und die Einhaltung der Nebenbedingung bei der Verwendung eines Filters getrennt betrachtet. Dadurch entfallen die Abhängigkeit der Liniensuche von den Skalierungen der Zielfunktion und der Nebenbedingungen und die iterative Anpassung des Gewichtungsparameters η innerhalb der Bewertungsfunktion. Die folgenden Betrachtungen orientieren sich an der Arbeit von Kemper [68].
Die grundsätzliche Idee ist, dass neue Iterierte akzeptiert werden, wenn diese entweder die Zielfunktion oder die Zulässigkeit verbessern. Durch die Hilfsfunktion

$$H(x) \coloneqq \max\{0, |h_1(x)|, ..., |h_{N_h}(x)|, g_1(x), ..., g_{N_g}(x)\} \tag{3.83}$$

wird ein skalares Maß für die Verletzung der Nebenbedingungen definiert. Formal lassen sich die beiden möglicherweise widersprüchlichen Ziele der Optimierung durch

$$\min_{x \in \mathbb{R}^{N_x}} f(x) \tag{3.84}$$

und

$$\min_{x \in \mathbb{R}^{N_x}} H(x) \tag{3.85}$$

beschreiben. Zur Bewertung einer Iterierten $x^{[k]}$ wird mit den Definitionen $H^{[k]} \coloneqq H(x^{[k]})$ und $f^{[k]} \coloneqq f(x^{[k]})$ das Paar $(H^{[k]}, f^{[k]})$ verwendet. Zur Beschreibung des Filterverfahrens folgen einige grundlegende Definitionen.

Definition 3.8 (Dominanz)**.** *Gelten die Bedingungen*

$$H^{[j]} \leq H^{[k]} \quad \textit{und} \quad f^{[j]} \leq f^{[k]}.$$

so wird der Punkt $(H^{[k]}, f^{[k]})$ vom Punkt $(H^{[j]}, f^{[j]})$ dominiert.

Mit Hilfe des Dominanzkriteriums lässt sich die Qualität eines Filtereintrages bewerten. Dieses Kriterium ist nötig, weil durch die Entkopplung von Nebenbedingungsverletzung und Zielfunktion einer Iterierten nicht einfach eine Zahl wie bei Verwendung einer Bewertungsfunktion zugeordnet werden kann. Der eigentliche Filter ist eine Menge von Paaren der Form $(H(x), f(x))$.

Definition 3.9 (Filter)**.** *Ein Filter $\mathcal{F}$ ist eine Menge von $(H(x), f(x))$-Paaren, in der kein Eintrag einen anderen dominiert. Eine neue Iterierte $x^{[k+1]}$ wird vom Filter akzeptiert, wenn das Paar $(H^{[k+1]}, f^{[k+1]})$ von keinem Eintrag des Filters dominiert wird. Das Paar $(H^{[k+1]}, f^{[k+1]})$ ist zur Aufnahme in den Filter geeignet, wenn es vom Filter akzeptiert wird.*

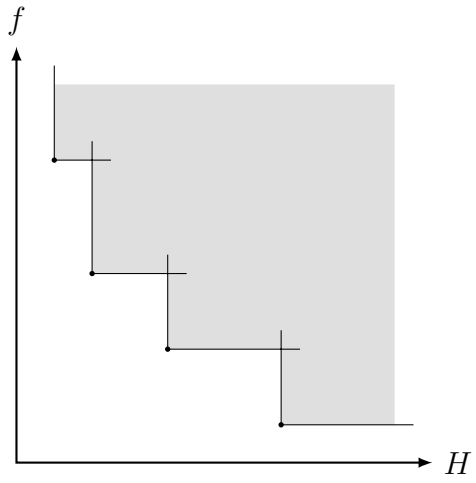


Abbildung 3.2: Schematische Darstellung eines Filters aus Definition 3.9 in der (H, f)-Ebene (Abbildung aus Kemper [68]). Der graue Bereich wird von den Filtereinträgen dominiert.

Bemerkung 3.10. *Um die Konsistenz des Filters zu gewährleisten muss bei Aufnahme eines neuen Paares geprüft werden, ob alte Einträge durch das neue Paar dominiert werden. Diese dominierten Einträge müssen gegebenenfalls aus dem Filter entfernt werden.*

In Abbildung 3.2 ist ein beispielhafter Filter zur Veranschaulichung des Prinzips dargestellt.
Durch die Entkopplung von Zielfunktion und Nebenbedingungen im Filter entstehen bei der Verwendung eines einfachen Filters entsprechend der Definition 3.9 Probleme. Der Filter würde eine Folge von Iterierten akzeptieren, die lediglich die Zielfunktion verbessern und einen unzulässigen Häufungspunkt haben. Analog ist es möglich, dass eine Folge akzeptiert wird, die in jedem Schritt lediglich die Zulässigkeit verbessert, so dass das Verfahren gegen einen beliebigen zulässigen Punkt konvergieren könnte.
Um die Konvergenz gegen unzulässige Punkte zu unterbinden führen Fletcher, Leyffer und Toint [37] das folgende strengere Akzeptanzkriterium ein.

Definition 3.11 (Akzeptanzkriterium)**.** *Es seien Parameter $\gamma_f, \gamma_H \in (0, 1)$ gewählt. Eine Iterierte $x^{[k+1]}$ mit zugehörigen Werten $(H^{[k+1]}, f^{[k+1]})$ wird vom Filter akzeptiert, wenn für jeden Filtereintrag $(H^{[j]}, f^{[j]})$ die Bedingung*

$$H^{[k+1]} \leq (1 - \gamma_H) H^{[j]} \tag{3.86}$$

oder

$$f^{[k+1]} + \gamma_f H^{[k+1]} \leq f^{[j]} \tag{3.87}$$

erfüllt ist.

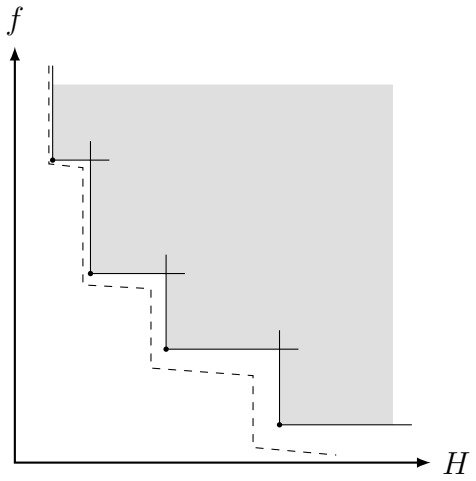


Abbildung 3.3: Schematische Darstellung eines Filters inklusive Filterhülle durch das Akzeptanzkriterium 3.11 (Abbildung aus Kemper [68]).

Das Akzeptanzkriterium verschärft die Dominanzbedingungen bezüglich der Zulässigkeit. Bedingung (3.86) verlangt eine hinreichende Verbesserung der Nebenbedingungsverletzung und die Bedingung (3.87) forderte eine stärkere Verringerung der Zielfunktion, solange die Nebenbedingungsverletzung groß ist. Zu Beginn dieses Abschnitts wurde der Filter mit der Entkopplung von Zielfunktion und Nebenbedingungen motiviert, allerdings entsteht durch die Bedingung (3.87) wieder eine Verbindung der beiden Größen. Im Gegensatz zur Bewertungsfunktion muss der Parameter γ_f aber nicht während der Optimierung angepasst werden, sondern wird lediglich vorab festgelegt. Durch das Akzeptanzkriterium wird eine schräge Hülle um den dominierten Bereich gelegt, so dass neue Iterierte einen größeren Fortschritt erzeugen müssen. Weiterhin kann mit Hilfe der schrägen Hülle bewiesen werden, dass bei der Entfernung von dominierten Filtereinträgen keine Informationen verloren gehen. Kemper zeigt diese sogenannte Inklusionseigenschaft des Filters mit dem erweiterten Akzeptanzkriteriums (Vergleiche Satz 3.6 in [68]). Weiterhin zeigt sie Konvergenzeigenschaften des Filters unter diesen Voraussetzungen (Vergleiche Satz 3.8 und Korollar 3.9 in [68]). In Abbildung 3.3 ist ein Filter mit beispielhafter schräger Hülle abgedruckt. Bereiche mit kleinerer Zielfunktion, aber noch bestehender Verletzung der Nebenbedingungen, gelten für neue Iterierte als verboten. Im Gegensatz dazu ist die Hülle nahe des zulässigen Bereichs (Nahe der y-Achse) wenig bis gar nicht schräg. Kemper schreibt in ihrer Arbeit [68], dass in der Praxis $\gamma_f \approx 0$, sowie $\gamma_H \approx 0$ gewählt werden, um die Kopplung der beiden Größen gering zu halten. Die andere problematische Situation bei der Verwendung eines Filters stellt die Konvergenz gegen einen beliebigen zulässigen Punkt dar, der insbesondere kein KKT-Punkt und somit kein lokales Minimum des Ausgangsproblems ist. Die folgenden Betrachtungen werden insbesondere bei den Analysen im Abschnitt 5.2 wieder aufgegriffen, weil ein ähnliches Verhalten dort beobachtet werden kann.
Um eine hinreichende Verkleinerung der Zielfunktion zu garantieren, werden Ideen

aus der Verwendung der Bewertungsfunktionen angewendet. Das Ziel ist die Konstruktion einer geeigneten Armijo-Bedingung, um die Konvergenz gegen beliebige zulässige Punkte zu verhindern. Mit den Parametern $\delta > 0, s_H > 1$ und $s_f \geq 1$ führen Wächter und Biegler [106] die sogenannte *Switching-Condition*

$$\alpha^{[k]} \nabla_x f(x^{[k]})^\top d^{[k]} < 0 \tag{3.88}$$

und

$$\alpha^{[k]} (-\nabla_x f(x^{[k]})^\top d^{[k]})^{s_f} > \delta (H^{[k]})^{s_H} \tag{3.89}$$

ein. Die Bedingung (3.88) stellt sicher, dass es sich in linearer Näherung bei $d^{[k]}$ um eine Abstiegsrichtung der Zielfunktion handelt. Die zweite Bedingung (3.89) fordert, dass die vorhergesagte Verringerung im Verhältnis zur aktuellen Verletzung der Nebenbedingungen groß genug ist. Es wird eine Konstante $H_{\min} > 0$ festgelegt. Die Konstante $H_{\min}$ definiert einen Schlauch um den zulässigen Bereich. Die Switching Condition wird nur geprüft, wenn die aktuelle Iterierte $x^{[k]}$ bereits wegen

$$H(x^{[k]}) < H_{\min}$$

innerhalb des Schlauches um den zulässigen Bereich liegt. Sind diese Voraussetzungen erfüllt, wird mit der Konstanten $\sigma_f \in (0, \frac{1}{2})$ die Armijo-Bedingung

$$f^{[k+1]} \leq f^{[k]} + \sigma_f \alpha^{[k]} \nabla_x f(x^{[k]})^\top d^{[k]} \tag{3.90}$$

geprüft. Die Iterierte $x^{[k+1]} = x^{[k]} + \alpha^{[k]} d^{[k]}$ wird akzeptiert, wenn diese Armijo-Bedingung erfüllt ist. Ansonsten muss die Schrittweite verringert werden.
Neben den hier diskutierten allgemeinen Eigenschaften von Filter-Verfahren für den SQP-Algorithmus sind noch zusätzliche Techniken wie die Zulässigkeitsrestaurierung nötig. Die Ideen dazu stammen aus der Entwicklung von Innere-Punkte-Verfahren für nichtlineare Optimierungsprobleme. Diese werden im weiteren Verlauf dieser Arbeit nicht weiter thematisiert, deshalb sei an diese Stelle erneut auf die Arbeit von Kemper [68] verwiesen (Siehe dort Abschnitt 3.4). Weitere Details finden sich auch in der dort zitierten Literatur.
Eine interessante Abwandlung des Filters wurde von S. Ulbrich [102] vorgeschlagen. Anstelle der Zielfunktion wird innerhalb des Filters die Lagrange-Funktion verwendet. Durch die Verwendung der Lagrange-Funktion soll insbesondere in der Umgebung eines lokalen Minimums die Konvergenz auch trotz lokal problematischer Auswirkungen der Nebenbedingungen, wie zum Beispiel des Maratos-Effektes [76], erhalten bleiben, ohne dass die Verwendung von Korrekturen zweiter Ordnung notwendig werden. Im Abschnitt 5.2 wird dieser Ansatz in abgewandelter Form mit der Idee einer Watchdog-Funktion [17] kombiniert, um den Fortschritt der dort diskutierten Methode zu bewerten.

3.2.3 Das globale SQP-Verfahren

Mit Hilfe der beschriebenen Liniensuch-Verfahren kann das lokale SQP-Verfahren 3 zu einem global konvergenten Verfahren erweitert werden. Algorithmus 4 beschreibt

Algorithmus 4 Globales SQP-Verfahren

1: Wähle Startschätzung $(x^{[0]}, \lambda^{[0]}, \mu^{[0]})$
2: **for** $k = 0, 1, 2, ...$ **do**
3: Erfüllt $(x^{[k]}, \lambda^{[k]}, \mu^{[k]})$ die KKT-Bedingungen für (2.1), Abbruch
4: Löse das Unterproblem (3.71) zur Bestimmung von $(d^{[k]}, \lambda^{[k+1]}, \mu^{[k+1]})$
5: Bestimme Schrittweite $\alpha^{[k]}$ durch Liniensuche für $x^{[k+1]} = x^{[k]} + \alpha^{[k]} d^{[k]}$
6: **end for**

die Vorgehensweise des globalen SQP-Verfahrens. Ein Konvergenzbeweis für das globalisierte SQP-Verfahren bei Verwendung einer Bewertungsfunktion findet sich beispielsweise bei Han [61] und bei Verwendung eines Filters bei Fletcher, Leyffer und Toint [37] und analog dazu für Innere-Punkte-Verfahren bei Wächter und Biegler [106]. Durch die Globalisierungsphase wird eine Annäherung der Iterierten an ein lokales Minimum sichergestellt und im Einzugsbereich des lokalen Minimums bleibt die Konvergenzgeschwindigkeit des Verfahrens superlinear, beziehungsweise quadratisch, entsprechend der Aussage des Satzes 3.5. Hierbei ist zu beachten, dass gegebenenfalls Maßnahmen getroffen werden müssen, um Probleme durch die Linearisierung der Nebenbedingungen in der Umgebung des lokalen Minimums, wie beispielsweise den Maratos-Effekt zu vermeiden.
Auf Grund dieser Gegebenheiten wird die Umgebung des lokalen Minimums auch *Bereich quadratischer Konvergenz* oder *Einzugsbereich des lokalen Minimums* genannt. Bei der Betrachtung von Iterationsverläufen wird das Eintreten der Iterierten in diesen Bereich häufig durch eine deutliche Beschleunigung des Konvergenzverhaltens begleitet. Für Details zu diesen Beobachtungen sei an dieser Stelle auf die numerischen Ergebnisse in Kapitel 7 verwiesen.
Die bisherigen Überlegungen gehen stillschweigend davon aus, dass die entstehenden quadratischen Unterprobleme lösbar sind. Im Einzugsbereich eines lokalen Minimums kann davon ausgegangen werden, dass diese Annahme in der Regel korrekt ist. Während der Globalisierungsphase können allerdings zwei verschiedene Aspekte Probleme verursachen.
Einerseits kann die Hesse-Matrix der Lagrange-Funktion nicht die gewünschte Krümmung aufweisen. Diese muss positiv definit auf dem Kern der aktiven Nebenbedingungen sein. Insbesondere bei der hochdimensionalen Optimierung wird diese Bedingung während der Globalisierungsphase häufig nicht erfüllt sein. Es bestehen verschiedene Möglichkeiten dieser Problematik entgegen zu wirken und die Hesse-Matrix zu regularisieren. Im Abschnitt 3.3.3.1 wird die Regularisierung der Hesse-Matrix innerhalb des Lösers WORHP thematisiert und die zentrale Bedeutung dieses Themas für die Effizienz des Verfahrens verdeutlicht. Im Zuge dessen werden Techniken diskutiert, um eine Effizienzsteigerung zu erwirken.
Neben der beschriebenen Problematik können andererseits nicht erfüllbare Nebenbedingungen auf Grund der Linearisierung der Nebenbedingungen entstehen. Im Abschnitt 3.3.3.3 wird die Relaxierung der Nebenbedingungen innerhalb von WORHP

vorgestellt und auch in diesem Zusammenhang werden Ideen diskutiert, um eine Effizienzsteigerung zu erwirken.
Abschließend sei noch angemerkt, dass in der Literatur verschiedene Bedeutungen des SQP-Verfahrens existieren. So weisen Gould, Orban und Toint [58] darauf hin, dass insbesondere im Zusammenhang mit Optimierungsproblemen unter der Einhaltung partieller Differentialgleichungen lediglich gleichungsbeschränkte Optimierungsprobleme als SQP-Verfahren bezeichnet werden.
Nocedal und Wright [82] erwähnen auch Verfahren, bei denen nur Teile der Nebenbedingungen linearisiert und im quadratischen Unterproblem als Gleichungsbeschränkungen verwendet werden, hierbei handelt es sich um eine Abwandlung der aktiven-Mengen-Strategien direkt auf der Ebene des NLP und nicht nur zur Lösung der quadratischen Unterprobleme.
Im Weiteren gibt es viele verschiedene spezialisierte Ausführungen des SQP-Verfahrens, insgesamt ist, nach Meinung des Autors, das hier vorgestellte SQP-Verfahren in seiner allgemeinen Form als das klassische SQP-Verfahren anzusehen. Insbesondere kann eine gute Implementierung dieses SQP-Verfahrens inklusive der beschriebenen Globalisierungsstrategien auf verschiedene Situationen und diverse Problemklassen direkt angewendet werden.

3.3 Der NLP Löser WORHP

Das Ziel der vorliegenden Arbeit ist die Analyse weiterer Ideen im Rahmen der zuvor erläuterten Theorie zur algorithmischen Lösung von nichtlinearen Optimierungsproblemen, um eine Effizienzsteigerung des Lösungsverfahrens zu erwirken. Als Grundlage für die weiteren Betrachtungen und insbesondere die verschiedenen neuen Algorithmen dient die Software WORHP [13]. Das Akronym WORHP steht für "We Optimize Really Huge Problems". WORHP ist ein Löser für nichtlineare Optimierungsprobleme der allgemeinen Form 2.1 und wurde im Rahmen verschiedener Projekte der europäischen Raumfahrtorganisation ESA, des deutschen Zentrums für Luft- und Raumfahrt DLR und des Forschungsprogrammes Clean Sky unter der Leitung der Universität Bremen in Zusammenarbeit mit der Universität der Bundeswehr München entwickelt.
Ziel bei der Entwicklung von WORHP ist die Lösung von hochdimensionalen Optimierungsproblemen. Während der Entwicklung von WORHP wurden diverse Abschlussarbeiten über verschiedene Teilbereiche der Theorie, Numerik und Implementierung geschrieben. Die Dissertation von Kalmbach [67] beschäftigt sich mit verschiedenen Möglichkeiten zur effizienten Bestimmung numerischer Ableitungen. In Abschnitt 3.3.2 sind die grundlegenden Ideen dieser Arbeit kurz zusammengefasst. Die Arbeit von Nikolayzik [81] befasst sich mit neuen Ideen zur Einbindung der parametrischen Sensitivitätsanalyse in das SQP-Verfahren und diese werden im Kapitel 5 aufgegriffen, analysiert und erweitert. Eine technische Dokumentation verschiedener Implementierungstechniken und Entwicklungskriterien für WORHP sind

in der Arbeit von Wassel [109] enthalten. Die Diplomarbeit von Kemper [68] beschreibt die Einbindung des Filterverfahrens von Fletcher und Leyffer [36] innerhalb von WORHP. Eine Beschreibung des Sensitivitätsmoduls WORHP Zen ist in der Bachelorarbeit von Schäfer [90] zu finden. Zuletzt sei an dieser Stelle auch noch auf die Masterarbeit des Autors [41] verwiesen. In dieser wurde bereits eine spezielle Technik zur Effizienzsteigerung mittels Algorithmen gemischter Präzision diskutiert. Im nächsten Abschnitt werden einige generelle Hintergrundinformationen und Details der verwendeten allgemeinen Algorithmen innerhalb von WORHP konkretisiert. In den darauf folgenden Abschnitten werden ausgewählte spezielle Details der Algorithmen des Lösers dargestellt. Insbesondere werden jene Teile thematisiert, die im weiteren Verlauf der Arbeit bezüglich Verbesserungspotential und Erweiterungsmöglichkeiten analysiert werden.

3.3.1 Technischer Hintergrund

Der Löser WORHP bietet die Möglichkeit ein globalisiertes SQP-Verfahren entsprechend Algorithmus 4 zur Lösung allgemeiner nichtlinearer Optimierungsprobleme zu verwenden[3]. Zur Liniensuche stehen verschiedene Bewertungsfunktionen sowie ein Filterverfahren zur Verfügung. Zur Lösung der quadratischen Unterprobleme innerhalb des SQP-Verfahrens wird der QP Löser QPSOL [46] der Universität der Bundeswehr München verwendet. Die Implementierung von QPSOL basiert zu großen Teilen auf dem Artikel von Gertz und Wright [47] über die Implementierung einer objektorientierten Software zur Lösung quadratischer Optimierungsprobleme mit einigen Erweiterungen insbesondere im Bezug auf die Warmstart-Fähigkeit des Lösers basierend auf den Ideen von Gondzio und Grothey [55]. Die quadratischen Probleme werden innerhalb von QPSOL mit Hilfe eines Innere-Punkte-Verfahrens, wie in Algorithmus 2 beschrieben, gelöst.
Als letzter Baustein für die Implementierung des SQP-Verfahrens in WORHP wird noch ein Löser für die innerhalb des Innere-Punkte-Verfahrens für quadratische Optimierungsprobleme auftretenden linearen Gleichungssysteme entsprechend der Gleichungen (3.40) und (3.45) benötigt. Die Gleichungssysteme werden innerhalb der Implementierung in die symmetrische Form aus Gleichung (3.55) gebracht, so dass spezialisierte Löser für schwachbesetzte symmetrische indefinite lineare Gleichungssysteme anwendbar sind. Als Standardlöser für lineare Gleichungssystem verwendet WORHP das Paket MA97 aus der Harwell Subroutine Library [66]. Alternativ existieren weitere Schnittstellen, wie zum Beispiel zu SuperLU [72]. Eine komplette Liste ist im Handbuch von WORHP enthalten [98].
Weitere Details zur allgemeinen Implementierung, zur Auswahl der verwendeten Programmiersprachen und zu technischen Hintergründen finden sich in der Arbeit von Wassel [109].

[3] Ein alternativer Innere-Punkte-Algorithmus befindet sich zur Zeit in der Entwicklung.

3.3.2 Berechnung von Ableitungen

Bei der Formulierung der quadratischen Unterprobleme und zur Prüfung der Abbruchbedingungen benötigt ein Löser für nichtlineare Optimierungsprobleme mindestens Ableitungen erster Ordnung der Zielfunktion und der Nebenbedingungen. Idealerweise liegt auch die Hesse-Matrix der Lagrange-Funktion vor. In der praktischen Umsetzung eines ableitungsbasierten Optimierungsverfahrens ist es deshalb von zentraler Bedeutung dem Benutzer Möglichkeiten zu bieten die Ableitungen zu berechnen, falls diese nicht in analytischer Form vorliegen, oder deren Bestimmung schlicht zu aufwendig ist.
WORHP wurde entwickelt, um insbesondere hochdimensionale Optimierungsprobleme lösen zu können. Die praktische Umsetzung erfordert deshalb eine besondere Behandlung der auftretenden Matrizen. Durch die Ausnutzung der Schwachbesetztheit der Matrizen ist eine effiziente Umsetzung de beschriebenen Algorithmen möglich. Dieser Abschnitt geht zunächst kurz auf die Eigenschaften von schwachbesetzten Matrizen ein. Anschließend werden die typischen Ansätze zur Kompensation des Fehlens analytischer Ableitungen dargestellt. Als erstes wird die Bestimmung numerischer Ableitungen mit Hilfe finiter Differenzen für Ableitungen erster und zweiter Ordnung beschrieben. Danach thematisiert der Abschnitt kurz die Ansätze zur Realisierung von Quasi-Newton Verfahren mit Hilfe von Broyden-Fletcher-Goldfarb-Shanno-Matrizen (BFGS-Matrizen) als Ersatz für die analytische Hesse-Matrix der Lagrange-Funktion.
Die folgenden Beschreibungen basieren auf den Ausführungen aus dem Buch von Gill, Murray und Wright [50]. Neben finiten Differenzen behandeln sie dort auch BFGS-Verfahren. Die Arbeit von Kalmbach [67] beschäftigt sich ausführlich mit den innerhalb des Lösers WORHP implementierten Verfahren zur effizienten Ableitungsbestimmung. Wassel [109] erläutert ebenfalls die Implementierung innerhalb von WORHP in kompakter Form.

3.3.2.1 Schwachbesetzte Matrizen

Die folgende Definition konkretisiert die Eigenschaft der Schwachbesetztheit von Matrizen.

Definition 3.12 (Schwachbesetztheit)**.** *Eine Matrix $A \in \mathbb{R}^{m\times n}$ mit Einträgen $a_{ij}, i = 1,...,m, j = 1,...,n$ heißt schwachbesetzt, wenn $a_{ij} = 0$ für viele Paare (i,j) gilt.*
Die Jacobi-Matrix einer differenzierbaren Funktion $f = (f_1,...,f_m) : \mathbb{R}^n \to \mathbb{R}^m$ ist schwachbesetzt, wenn

$$\frac{\partial f_i}{\partial x_j} \equiv 0$$

für viele Paare (i,j) gilt. Diese Einträge werden strukturelle Nullen genannt. Hierbei ist zu beachten, dass eine strukturelle Null unabhängig vom betrachteten Aus-

wertungspunkt bei einer Ableitungsmatrix ist, während ein von Null verschiedener Eintrag für bestimmte Punkte trotzdem eine numerische Null enthalten kann.
Die Anzahl der von Null verschiedenen Einträge sei mit n_{nz} bezeichnet. Als Maß für die Schwachbesetztheit wird der Quotient

$$\frac{n_{nz}}{nm} \in (0, 1].$$

betrachtet. Bei einem Wert von 1 *wird von einer dichtbesetzten Matrix gesprochen, wohingegen Werte nahe* 0 *als schwachbesetzt bezeichnet werden.*

Insbesondere hochdimensionale Optimierungsprobleme weisen in der Regel eine starke Schwachbesetztheit der Ableitungsmatrizen auf. In den Arbeiten von Büskens [11] und Betts [6] wird beispielsweise die Schwachbesetztheit bei der Diskretisierung von Optimalsteuerungsproblemen diskutiert.

3.3.2.2 Finite Differenzen

Es sei $e_{\{i\}}$ der i-te Einheitsvektor und $h > 0, h \in \mathbb{R}$ gegeben. Die analytische Ableitung einer Funktion $f(x) : \mathbb{R}^{N_x} \to \mathbb{R}$ nach der Variablen x_i kann durch den Grenzwert

$$\frac{\partial f}{\partial x_i}(x) = \lim_{h \to 0} \frac{f(x + h \cdot e_{\{i\}}) - f(x)}{h} \tag{3.91}$$

bestimmt werden. Bei der Anwendung finiter Differenzen wird dieser Grenzwert näherungsweise für festes h bestimmt. Standardmäßig verwendet WORHP beispielsweise $h = 10^{-5}$. Somit ist eine Näherung für die partielle Ableitung durch

$$\frac{\partial f}{\partial x_i}(x) \approx \frac{f(x + h \cdot e_{\{i\}}) - f(x)}{h} \tag{3.92}$$

gegeben. Dieser Differenzenquotient wird als *Vorwärtsdifferenz* bezeichnet. Mit Hilfe des Satzes von Taylor ergibt sich direkt die Fehlerabschätzung

$$\frac{\partial f}{\partial x_i}(x) - \frac{f(x + h \cdot e_{\{i\}}) - f(x)}{h} = \mathcal{O}(h) \tag{3.93}$$

für die Näherung. Analog kann auch die *Rückwärtsdifferenz* durch

$$\frac{\partial f}{\partial x_i}(x) \approx \frac{f(x) - f(x - h \cdot e_{\{i\}})}{h} \tag{3.94}$$

definiert werden und es ergibt sich

$$\frac{\partial f}{\partial x_i}(x) - \frac{f(x) - f(x - h \cdot e_{\{i\}})}{h} = \mathcal{O}(h) \tag{3.95}$$

als Fehlerabschätzung. Durch Kombination der beiden Differenzenquotienten kann der *zentrale Differenzenquotient*

$$\frac{\partial f}{\partial x_i}(x) \approx \frac{f(x + h \cdot e_{\{i\}}) - f(x - h \cdot e_{\{i\}})}{2h} \tag{3.96}$$

begründet werden. Der zentrale Differenzenquotient stellt eine bessere Approximation an die exakte Ableitung dar und genügt der Fehlerabschätzung

$$\frac{\partial f}{\partial x_i}(x) - \frac{f(x + h \cdot e_{\{i\}}) - f(x - h \cdot e_{\{i\}})}{2h} = \mathcal{O}(h^2). \tag{3.97}$$

Werden die Differenzenquotienten innerhalb eines Optimierungsverfahrens berechnet, ist zu beachten, dass die Auswertung von $f(x)$ bereits in der vorherigen Iteration während der Liniensuche erfolgt ist. Somit ist für die Berechnung der Vorwärts- oder Rückwärtsdifferenzen lediglich eine neue Funktionsauswertung nötig. Im Gegensatz dazu müssen bei der Bestimmung eines zentralen Differenzenquotienten zwei neue Auswertungen erfolgen.
Bei der Berechnung der Jacobi-Matrix der Nebenbedingungen ergibt sich eine abweichende Situation. Gesucht sind die Einträge

$$\frac{\partial g_i}{\partial x_j}(x) \quad \text{für } i = 1, ..., N_g,\ j = 1, ..., N_x,$$
$$\frac{\partial h_k}{\partial x_j}(x) \quad \text{für } k = 1, ..., N_h,\ j = 1, ..., N_x.$$

der Jacobi-Matrizen von g und h. Die folgenden Erklärungen werden beispielhaft für den Vektor der Ungleichungsnebenbedingungen g durchgeführt, können aber völlig analog auch auf die Gleichungsnebenbedingungen h übertragen werden. In der praktischen Optimierung werden die Nebenbedingungen durch eine Implementierung des Benutzers zur Verfügung gestellt. In der Regel ist es deshalb nur möglich alle N_g Nebenbedingungen an einer beliebigen Stelle x gleichzeitig auszuwerten. Kalmbach [67] befasst sich in seiner Arbeit mit der Aufgabe durch eine möglichst geringe Anzahl von Auswertungen des Nebenbedingungsvektors die Jacobi-Matrix mittels finiter Differenzen zu bestimmen. Die grundsätzliche Idee lässt sich durch einige anschauliche Beispiele verdeutlichen.
Es seien die Nebenbedingungsvektoren $g^{[1]}, g^{[2]}$ und $g^{[3]}$ mit den dazugehörigen Strukturen der resultierenden Jacobi-Matrizen durch

$$g^{[1]} = \begin{pmatrix} x_1 + x_2 \\ x_2 + x_3 \\ x_3 + x_4 \end{pmatrix}, \qquad \nabla g^{[1]} = \begin{pmatrix} \times & \times & & \\ & \times & \times & \\ & & \times & \times \end{pmatrix}$$

$$g^{[2]} = \begin{pmatrix} x_1 + x_2 \\ x_3^2 \\ x_4^2 \end{pmatrix}, \qquad \nabla g^{[2]} = \begin{pmatrix} \times & \times & & \\ & & \times & \\ & & & \times \end{pmatrix}$$

$$g^{[3]} = \begin{pmatrix} x_1 + x_2 \\ x_1 + x_2 + x_3 \\ x_3^2 \end{pmatrix}, \qquad \nabla g^{[3]} = \begin{pmatrix} \times & \times & & \\ \times & \times & \times & \\ & & \times & \end{pmatrix}$$

definiert. Die Einträge in der Jacobi-Matrix können entsprechend der farblichen Markierungen in Gruppen unterteilt werden. Für $g^{[1]}$ bedeutet dies, dass die Ableitungen bezüglich der Variablen x_1 und x_3 gleichzeitig berechnet werden können. Bei der Verwendung einer Vorwärtsdifferenz können mit Hilfe des Quotienten

$$\frac{1}{h}\left(g^{[1]}(x + h \cdot (e_{\{1\}} + e_{\{3\}})) - g^{[1]}(x)\right) \tag{3.98}$$

die Einträge $\nabla g^{[1]}_{(1,1)}, \nabla g^{[1]}_{(2,3)}$ und $\nabla g^{[1]}_{(3,3)}$ gleichzeitig ermittelt werden. Analog wird danach der Quotient bezüglich x_2 und x_4 ausgewertet und die übrigen Einträge können bestimmt werden. Neben den Funktionswerten in x sind somit lediglich zwei zusätzliche Auswertungen des Vektors nötig um sechs Einträge der Matrix zu berechnen.
Für die Funktion $g^{[2]}$ ergeben sich die Gruppen $\{x_1, x_3, x_4\}$ und $\{x_2\}$, so dass auch dort zwei zusätzliche Auswertungen nötig sind. Das Beispiel $g^{[3]}$ zeigt, dass eine Nebenbedingung wie $g^{[3]}_2$ für dieses Verfahren störend ist. Dort ist auf Grund der zweiten Zeile keine Gruppierung der Variablen möglich.
Die Bestimmung der Gruppen kann mit Hilfe von Algorithmen aus der Graphentheorie erfolgen. Eine Einführung in diese Theorie findet sich im Buch von Diestel [25]. Formal lässt sich ein ungerichteter Graph wie folgt definieren.

Definition 3.13 (Ungerichteter Graph)**.** *Ein* Graph *besteht aus einer endlichen und nichtleeren Menge von Knoten V und dazugehörigen Kanten E. Eine Kante verbindet zwei Knoten und wird durch diese charakterisiert:*

$$E \subset \{(u, v) : \ u \neq v, \ u, v \in V\}. \tag{3.99}$$

Die Schwachbesetztheitsstruktur entsprechend der Definitionen im Abschnitt 3.3.2.1 induziert einen Graphen derart, dass jedes $x_i, \ i \in [1, ..., N_x]$ durch einen Knoten repräsentiert wird und es genau dann eine Kante zwischen den Knoten x_i und x_j mit $i \neq j, \ i, j \in [1, ..., N_x]$ gibt, wenn es eine Komponente des Vektors $g(x)$ gibt, die sowohl von x_i, als auch von x_j abhängt. Zwei Variablen x_i und x_j mit $i \neq j$ dürfen gleichzeitig ausgewertet werden, wenn keine Komponente des Nebenbedingungsvektors sowohl von x_i, als auch von x_j abhängt, das heißt wenn sie keine gemeinsame Kante haben.
Eine zur Gruppenfindung äquivalente Problemstellung ergibt sich bei dem sogenannten Färbungsproblem aus der Graphentheorie. Zunächst wird die Färbung eines Graphen definiert.

Definition 3.14 (Knotenfärbung)**.** *Es sei die Menge aller Farben durch die Menge $C \subset \mathbb{N}$ gegeben. Für einen Graphen (V, E) heißt die Funktion $\theta : V \to C$ eine* Knotenfärbung*. Eine Knotenfärbung ist zulässig, wenn keinen Knoten mit gemeinsamer Kante dieselbe Farbe zugewiesen wird. Es muss demnach für alle $v \in V$ erfüllt sein, dass aus $w \in \{u \in V \mid (v, u) \in E\}$ folgt, dass $\theta(v) \neq \theta(w)$ gilt.*
Weiterhin gilt mit einer zulässigen Knotenfärbung θ mit $k = |C|$, dass der Graph k-knotenfärbbar ist.

Mit Hilfe dieser Definitionen lässt sich das Auffinden der Gruppen für die effiziente Berechnung der Einträge der Jacobi-Matrix durch das Färbungsproblem

$$\begin{aligned} \min_{\theta} \quad & |C| \\ \text{unter} \quad & \theta \text{ zulässig} \end{aligned} \tag{3.100}$$

beschreiben. Garey, Johnson und Stockmeyer [40] zeigen, dass das Entscheidungsproblem, ob ein Graph 3-knotenfärbbar ist, NP-vollständig ist. Demnach existiert auch für das Färbungsproblem kein Algorithmus, der dieses in polynomieller Zeit lösen kann. Innerhalb von WORHP wurden suboptimale Algorithmen zur Lösung des Problems implementiert.
Damit während der Optimierung das quadratische Unterproblem aufgestellt werden kann, wird zusätzlich die Hesse-Matrix der Lagrange-Funktion benötigt. Entsprechend der Definition der Lagrange-Funktion (2.22) mit $l_0 = 1$ ergeben sich die Ableitungen

$$\nabla_x L(x, \lambda, \mu) = \nabla_x f(x) + \lambda^\top \nabla_x g(x) + \mu^\top \nabla_x h(x), \tag{3.101}$$

$$\nabla^2_{xx} L(x, \lambda, \mu) = \nabla^2_{xx} f(x) + \sum_{i=1}^{N_g} \lambda_i \nabla^2_{xx} g_i(x) + \sum_{j=1}^{N_h} \mu_j \nabla^2_{xx} h_j(x). \tag{3.102}$$

Zur Berechnung der zweiten Ableitungen können analog zu den bisherigen Betrachtungen finite Differenzen verwendet werden. Die folgende Beschreibung wird am Beispiel der Zielfunktion vorgenommen. Die Ansätze lassen sich analog auf die einzelnen Komponenten der beiden Nebenbedingungsvektoren übertragen.
Die Vorwärtsdifferenz zur Berechnung der zweiten Ableitung kann erneut durch die Taylor-Formel bestimmt werden und ist für einen Eintrag der Hesse-Matrix durch

$$\left(\nabla^2_{xx} f(x)\right)_{(i,j)} \approx \frac{f(x + h(e_{\{i\}} + e_{\{j\}})) - f(x + he_{\{i\}}) - f(x + he_{\{j\}}) + f(x)}{h^2} \tag{3.103}$$

gegeben. Analog zur Fehlerabschätzung für die Vorwärtsdifferenz zur Berechnung der Ableitung erster Ordnung ergibt sich

$$\left(\nabla^2_{xx} f(x)\right)_{(i,j)} - \frac{f(x + h(e_{\{i\}} + e_{\{j\}})) - f(x + he_{\{i\}}) - f(x + he_{\{j\}}) + f(x)}{h^2} = \mathcal{O}(h) \tag{3.104}$$

zur Quantifizierung des Fehlers. Liegen der Gradient der Zielfunktion und die Jacobi-Matrix der Nebenbedingungen analytisch vor, können diese verwendet werden, um mit Hilfe der Formeln zur Berechnung der Ableitungen erster Ordnungen die Einträge der Hesse-Matrix zu ermitteln.
Kalmbach [67] erweitert in seiner Arbeit die Ideen der Gruppen zur effizienten Berechnung der Matrixeinträge auch auf die Hesse-Matrix und reduziert mit seinen

Techniken den Aufwand zur Berechnung der Hesse-Matrix zum Teil erheblich.
Weitere Details über finite Differenzen können im Buch von Gill, Murray and Wright [50] gefunden werden. Sie diskutieren verschiedene Fehler, die bei der Berechnung finiter Differenzen auftreten können, und beschreiben Techniken, um h möglichst geschickt zu wählen. Innerhalb von WORHP findet aber keine adaptive Wahl von h statt, da in diesem Fall die bereits vorhandenen Auswertungen nicht erneut verwendet werden können.

3.3.2.3 Quasi-Newton-Verfahren

Der Beweis des Satzes 3.5 über die Konvergenzordnung des lokalen SQP-Verfahrens zeigt, dass die Verwendung von Ableitungsinformationen zweiter Ordnung essentiell für die Konvergenzgeschwindigkeit des Optimierungsprozesses ist. Allerdings kann in der hochdimensionalen Optimierung selbst die Verwendung der Gruppenstrategie zur effizienten Berechnung der Einträge der Hesse-Matrix sehr aufwendig sein. Deshalb wurde alternativ das eindimensionale Sekantenverfahren für den mehrdimensionalen Fall erweitert. Broyden [10], Fletcher [33], Goldfarb [52] und Shanno [95] haben unabhängig voneinander diesen Ansatz verfolgt und auf diese Weise das sogenannte Broyden-Fletcher-Goldfarb-Shanno (BFGS)-Verfahren entwickelt. Die Idee des BFGS-Verfahrens ist es durch geeignete Differenzen bezüglich der Suchrichtung in jeder Iteration Krümmungsinformationen zu bestimmen und diese kumulativ in einer Matrix zu sammeln.
Die folgende kurze Darstellung orientiert sich an den Arbeiten von Kalmbach [67], dem Buch von Gill, Murray und Wright [50] und dem Buch von Nocedal und Wright [82].
Zunächst wird die allgemeine Berechnung der BFGS-Matrizen beschrieben und im weiteren Verlauf werden dann kurz die von Kalmbach beschriebenen und innerhalb von WORHP implementierten Varianten aufgeführt.
Innerhalb der Optimierung sollen die Krümmungsinformationen der Lagrange-Funktion bezüglich der Optimierungsvariablen x in einer Approximationsmatrix gesammelt werden. Wurde in der Iteration k die Matrix $B^{[k]}$ verwendet um die Suchrichtung zu bestimmen, so sollen die dort gesammelten Informationen in einer Krümmungsmatrix $U^{[k]}$ integriert werden. Die Matrix $B^{[k+1]}$ für die nächste Iteration ergibt sich anschließend durch

$$B^{[k+1]} = B^{[k]} + U^{[k]}. \tag{3.105}$$

Als Startmatrix wird in der Regel die Einheitsmatrix verwendet. Demnach wird durch

$$B^{[0]} = I^{[0]} \tag{3.106}$$

in der ersten Iteration ein Schritt des Gradientenverfahrens durchgeführt. Die benötigten Informationen zur Berechnung der Matrix U_k werden durch die

Änderungen in der Suchrichtung und im Gradienten der Lagrange-Funktion ermittelt. Hierfür sei $x^{[k+1]} - x^{[k]} = d^{[k]}$ gegeben. Erneut hilft uns der Satz von Taylor und liefert die Beziehung

$$\nabla L(x^{[k+1]}) = \nabla L(x^{[k]}) + \nabla^2_{xx} L(x^{[k]}) d^{[k]} + \mathcal{O}(d^{[k]})^2. \tag{3.107}$$

Gill, Murray und Wright [50] verwenden diese Formel, um die Näherung für die lokalen Krümmungsinformationen entlang der Suchrichtung $d^{[k]}$ zu erhalten. Es sei die Änderung im Gradienten der Lagrangefunktion abgekürzt durch

$$v^{[k]} := \nabla L(x^{[k+1]}, \lambda^{[k+1]}, \mu^{[k+1]}) - \nabla L(x^{[k]}, \lambda^{[k+1]}, \mu^{[k+1]}) \tag{3.108}$$

definiert. Hierbei ist zu beachten, dass beide Auswertungen des Gradienten mit den Multiplikatoren der Iteration $k+1$ erfolgen. In der numerischen Praxis liegen die Ableitungen der Zielfunktion und Nebenbedingungen nach Durchführung eines Schrittes wegen der Überprüfung der Abbruchkriterien in der Regel sowieso schon vor, so dass hier keine neuen Auswertungen der potentiell teuren benutzerdefinierten Funktionen nötig werden.
Damit die Krümmungsinformationen in die Approximationsmatrix integriert werden sollte nach Möglichkeit entsprechend Gleichung (3.107) die sogenannte *Quasi-Newton-Bedingung*

$$B^{[k+1]} d^{[k]} = v^{[k]} \tag{3.109}$$

gelten. Nach Konstruktion des SQP-Verfahrens kann diese aber nicht direkt gelten, da die Berechnung des Schrittes bereits Gleichung (3.68) fordert. Mit Hilfe der Änderung des Gradienten $v^{[k]}$ und der Suchrichtung $d^{[k]}$ wird eine Rang-2 Update-Matrix mit Hilfe der Formel

$$U^{[k]} := \frac{(v^{[k]})(v^{[k]})^\top}{(v^{[k]})^\top d^{[k]}} - \frac{B^{[k]}(d^{[k]})(d^{[k]})^\top (B^{[k]})^\top}{(d^{[k]})^\top B^{[k]} d^{[k]}} \tag{3.110}$$

konstruiert. Unter Einsatz der Formel (3.110) in Gleichung (3.105) zeigt Kalmbach [67], dass die resultierende Matrix $B^{[k+1]}$ genau dann positiv definit ist, wenn die Bedingung

$$(v^{[k]})^\top d^{[k]} > 0 \tag{3.111}$$

erfüllt ist.
Mit dem beschriebenen Verfahren kann demnach eine Ersatzmatrix anstelle der analytischen oder mittels finiten Differenzen berechneten Hesse-Matrix konstruiert werden. Die Verwendung der dyadischen Produkte in Gleichung (3.110) liefert dichtbesetzte Updatematrizen U_k. Deshalb kann diese Vorgehensweise in der hochdimensionalen Optimierung nicht direkt verfolgt werden. Kalmbach [67] schlägt in seiner Arbeit verschiedene Techniken vor, um die vorhandene Schwachbesetztheitsstruktur möglichst gut zu erhalten. Der Rest dieses Abschnitts ist drei dieser Varianten gewidmet. Innerhalb der folgenden Betrachtungen ist es immer wieder nötig möglichst

gute Abdeckungen oder Permutationen zu finden. Für eine detailliertere Beschreibung der dazu verwendeten Algorithmen sei an dieser Stelle ebenfalls lediglich auf die Arbeit von Kalmbach [67] verwiesen.
Es sei die Struktur der zu approximierenden Hesse-Matrix durch

$$A := \begin{pmatrix} A_{11} & A_{12} & & & & \\ A_{21} & A_{22} & A_{23} & & & \\ & A_{32} & A_{33} & A_{34} & & \\ & & A_{43} & A_{44} & A_{45} & \\ & & & A_{54} & A_{55} & A_{56} \\ & & & & A_{65} & A_{66} \end{pmatrix} \tag{3.112}$$

vorgegeben. Durch Einsatz des normalen BFGS-Verfahrens würden alle strukturellen Nullen in der Approximationsmatrix verschwinden. Das Ziel eines alternativen Ansatzes sollte demnach die Erhaltung möglichst vieler struktureller Nullen sein. Ein einfaches, leicht umzusetzendes, Verfahren ist das sogenannte Block-BFGS-Verfahren (ohne Überschneidungen). Es sei eine beliebige Matrix mit Blockstruktur

$$M = \begin{pmatrix} M_1 & \\ & M_2 \end{pmatrix} \tag{3.113}$$

gegeben. Es ist leicht ersichtlich, dass die zusammengesetzte Matrix M genau dann positiv definit ist, wenn sowohl M_1, als auch M_2 positiv definit sind. Zwei mögliche Ansätze zur Approximation der Matrix A aus Gleichung (3.112) bieten sich durch

$$B = \begin{pmatrix} B_{11} & B_{12} & B_{13} & & & \\ B_{21} & B_{22} & B_{23} & & & \\ B_{31} & B_{32} & B_{33} & - & & \\ & & - & B_{44} & B_{45} & B_{46} \\ & & & B_{54} & B_{55} & B_{56} \\ & & & B_{64} & B_{65} & B_{66} \end{pmatrix}, \hat{B} = \begin{pmatrix} \hat{B}_{11} & \hat{B}_{12} & & & & \\ \hat{B}_{21} & \hat{B}_{22} & - & & & \\ & - & \hat{B}_{33} & \hat{B}_{34} & & \\ & & \hat{B}_{43} & \hat{B}_{44} & - & \\ & & & - & \hat{B}_{55} & \hat{B}_{56} \\ & & & & \hat{B}_{65} & \hat{B}_{66} \end{pmatrix}$$

an. Wird die Matrix B verwendet, erzeugen die Einträge B_{13}, B_{31}, B_{46} und B_{64} zusätzliche Nichtnulleinträge. Die Minuszeichen kennzeichnen die Einträge A_{34} und A_{43}, diese werden bei dieser Vorgehensweise von der gewählten Blockstruktur nicht erfasst. Alternativ könnte die aus drei Blöcken bestehende Matrix $\hat{B}$ verwendet werden. Diese erzeugt keine unnötigen Nichtnulleinträge, ist dafür aber nicht in der Lage die Einträge A_{23}, A_{32}, A_{45} und A_{54} abzudecken. Durch Steuerung der Blockgrößen und der Anzahl der Blöcke kann beim Block-BFGS-Verfahren (ohne Überschneidungen) Einfluss auf die Anzahl der nicht abgedeckten Einträge und der zusätzlich erzeugten Nichtnulleinträge genommen werden. Die Abdeckungsqualität hat in der Regel einen Einfluss auf die Konvergenzgeschwindigkeit. Gleichzeitig bedeuten zusätzliche Nichtnulleinträge zusätzlichen Rechenaufwand, so dass bei der Parametrisierung des Verfahrens ein Kompromiss gefunden werden muss.

Dieses Beispiel verdeutlich gleichzeitig die Nachteile des Verfahrens. Selbst die vorliegende relativ einfache tridiagonale Struktur kann nicht zufriedenstellend approximiert werden. Als Erweiterung des Verfahrens können Blöcke mit Überschneidungen erlaubt werden. Bei Verwendung von sechs 2×2-Blöcken kann durch die Matrix

$$C = \begin{pmatrix} C_{11} & C_{12} & & & & \\ C_{21} & C_{22} & C_{23} & & & \\ & C_{32} & C_{33} & C_{34} & & \\ & & C_{43} & C_{44} & C_{45} & \\ & & & C_{54} & C_{55} & C_{56} \\ & & & & C_{65} & C_{66} \end{pmatrix}$$

die Struktur der Matrix A direkt abgebildet werden. Die Einträge C_{22}, C_{33}, C_{44} und C_{55} liegen bei dieser Variante jeweils innerhalb von zwei verschiedenen Blöcken und müssen bei der Einarbeitung der Krümmungsinformationen in jeder Iteration gesondert betrachtet werden. Werden größere Blöcke verwendet ist es auch denkbar, dass einzelne Einträge in mehr als zwei Blöcken gleichzeitig berücksichtigt werden müssen. Kalmbach [67] behandelt diese Fälle in seiner Arbeit und gibt Formeln an, um die positive Definitheit der resultierenden Matrizen zu gewährleisten. Weiterhin kann gezeigt werden, dass die superlineare Konvergenzgeschwindigkeit des normalen BFGS-Verfahrens auch im Fall von Block-BFGS-Verfahren eingehalten wird [109]. Je nach Struktur der zu approximierenden Hesse-Matrix kann die Approximationsgüte beider Block-BFGS-Verfahren trotzdem unzureichend sein. Zur Näherung der Matrix

$$D = \begin{pmatrix} D_{11} & & & D_{14} \\ & D_{22} & & \\ & & D_{33} & \\ D_{41} & & & D_{44} \end{pmatrix} \tag{3.114}$$

bietet sich lediglich ein dichter 4×4-Block an. Durch Permutation der Variablen kann die Situation aber deutlich verbessert werden. Durch Tausch von x_2 und x_4 entsteht die Matrix

$$\begin{pmatrix} D_{11} & \tilde{D}_{12} & & \\ \tilde{D}_{21} & \tilde{D}_{22} & & \\ & & D_{33} & \\ & & & \tilde{D}_{44} \end{pmatrix}. \tag{3.115}$$

Diese kann durch nicht überschneidende Blöcke der Größen $\{2, 1, 1\}$ passend abgebildet werden. Als weiteres Beispiel führt Kalmbach [67] die Matrix

$$Q = \begin{pmatrix} Q_{11} & Q_{12} & & \\ Q_{21} & Q_{22} & Q_{23} & Q_{24} \\ & Q_{32} & Q_{33} & \\ & Q_{42} & & Q_{44} \end{pmatrix} \tag{3.116}$$

ein. Die verschiedenen Block-BFGS-Verfahren sind auch mit Permutation der Variablen nicht in der Lage die Struktur der Matrix Q exakt abzubilden. Dieser Umstand motiviert die Einführung des Sparse-BFGS-Verfahrens. Bei diesem Verfahren wird für die Teilstrukturen

$$\begin{pmatrix} Q_{11} & Q_{12} \\ Q_{21} & Q_{22} \end{pmatrix}, \begin{pmatrix} Q_{22} & Q_{23} \\ Q_{32} & Q_{33} \end{pmatrix}, \begin{pmatrix} Q_{22} & Q_{24} \\ Q_{42} & Q_{44} \end{pmatrix} \tag{3.117}$$

separat ein BFGS-Update durchgeführt. Danach werden diese wieder zusammengesetzt und in der Approximationsmatrix geeignet verrechnet. In diesem Fall muss lediglich der Eintrag Q_{22} gesondert behandelt werden, um die positive Definitheit der resultierenden Matrix zu gewährleisten.
Ist die Struktur der Hesse-Matrix nicht bekannt, kann über ein Block-BFGS-Verfahren mit fester Blockgröße zumindest eine Variante des Verfahrens angeboten werden, die in der Regel einem simplen Gradientenverfahren überlegen ist. Liegt die Struktur vor, kann zwischen finiten Differenzen mit Gruppenstrategie oder einer möglichst guten Approximation durch die verschiedenen BFGS-Verfahren gewählt werden. Insgesamt bieten diese Verfahren eine Vielzahl von Möglichkeiten auch im hochdimensionalen Fall als Ersatz für finite Differenzen zu dienen.

3.3.3 Globalisierungsphase

Die beschriebenen Hauptbestandteile eines globalen SQP-Lösers sind ein Verfahren zur Lösung der quadratischen Unterprobleme, inklusive der entstehenden linearen Gleichungssysteme, sowie eine geeignete Strategie zur Schrittweitenbestimmung. In der allgemeinen Beschreibung des Algorithmus in Abschnitt 3.2.3 wurde nur am Rande erwähnt, dass die Lösbarkeit der Unterprobleme insbesondere in der Globalisierungsphase sichergestellt werden muss. Dieser Punkt ist für die Effizienz einer praktischen Umsetzung allerdings von zentraler Bedeutung.
Satz 3.1 definiert Kriterien für die eindeutige Lösbarkeit gleichungsbeschränkter quadratischer Optimierungsprobleme. Der Ansatz der Aktive-Mengen-Methode aus Abschnitt 3.1.2.1 motiviert darüber hinaus, dass sich die Lösung eines ungleichungsbeschränkten quadratischen Optimierungsproblems bei geeigneter Wahl der aktiven Menge auf die Lösung eines gleichungsbeschränkten Problems reduzieren lässt. Demnach geben die Voraussetzungen von Satz 3.1 unter Berücksichtigung der aktiven Menge eine Vorstellung der möglichen Effekte die in der Lage sind die Lösbarkeit der Unterprobleme zu gefährden. Wie bereits in Abschnitt 3.2.3 erläutert wurde, ist es einerseits möglich, dass die positive Definitheit der Hesse-Matrix der Lagrange-Funktion nicht in dem erforderlichen Maße (auf dem Kern der aktiven Nebenbedingungen) vorliegt, andererseits können auf Grund der Problemstellung oder der Linearisierung inkonsistente Nebenbedingungen entstehen. Im Folgenden werden die innerhalb von WORHP implementierten Techniken zur Sicherstellung der Lösbarkeit der Unterprobleme vorgestellt. Gleichzeitig werden neue Techniken zur Verbesserung

des Verfahrens diskutiert. Zunächst beschäftigt sich der folgende Abschnitt mit der Regularisierung der Hesse-Matrix und thematisiert anschließend die Relaxierung der Nebenbedingungen. Abschließend werden Möglichkeiten diskutiert während der Globalisierungsphase des Algorithmus durch Variation von Konfigurationen des Lösers eine Effizienzsteigerung zu erwirken.

3.3.3.1 Regularisierung der Hesse-Matrix

Zunächst stellt sich die Frage, wie eine Überprüfung der Voraussetzung der positiven Definitheit auf dem Kern der aktiven Nebenbedingungen bei der Lösung hochdimensionaler Optimierungsprobleme durchgeführt werden kann.
Um eine Überprüfung vorzunehmen, muss zunächst der Kern der aktiven Nebenbedingungen, beziehungsweise eine Basis für diesen Untervektorraum, bekannt sein. Numerisch kann eine Basis T für den Kern der aktiven Nebenbedingungen beispielsweise mit Hilfe einer QR-Faktorisierung der Matrix

$$\tilde{A} = \begin{pmatrix} \nabla_x g_{\mathcal{A}}(x^{[k]}) \\ \nabla_x h(x^{[k]}) \end{pmatrix}, \tag{3.118}$$

bestehend aus den Jacobi-Matrizen der Nebenbedingungen, erfolgen. Mit $g_{\mathcal{A}}(x^{[k]})$ werden nur die aktiven Ungleichungsnebenbedingungen in $x^{[k]}$ beschrieben, so dass sich die Zeilen der Matrix $\tilde{A}$ aus den Gradienten der aktiven Ungleichungsnebenbedingungen und der Gleichungsnebenbedingungen zusammensetzen. Die Matrix

$$T^\top \nabla^2_{xx} L(x^{[k]}, \lambda^{[k]}, \mu^{[k]}) T \tag{3.119}$$

wird als *reduzierte Hesse-Matrix* bezeichnet. Die positive Definitheit der reduzierten Hesse-Matrix ist äquivalent zur positiven Definitheit der Ausgangsmatrix auf dem Kern der aktiven Nebenbedingungen. Eine mögliche Strategie zur Sicherstellung der gewünschten Eigenschaft stellt die Addition einer positiv definiten Matrix dar.
Levenberg [71] und Marquart [77] schlugen zunächst bei der Lösung von Kleinste-Quadrate-Optimierungsproblemen die Verwendung der Matrix κI_{N_x} mit $\kappa \in \mathbb{R}$, $\kappa > 0$ als Hesse-Matrix vor.
Die Regularisierungsmatrix κI_{N_x} kann auch in der hier vorliegenden Situation zur Herstellung der positiven Definitheit verwendet werden und wird auf Grund der Ideen von Levenberg und Marquart als Levenberg-Marquart-Regularisierung bezeichnet.
Die im Rahmen dieser Arbeit vorgestellten Algorithmen und insbesondere der Löser WORHP zielen auf die Lösung hochdimensionaler Optimierungsprobleme ab, deshalb ist es nicht ohne deutlichen Rechenaufwand möglich in jedem Iterationsschritt des SQP-Verfahrens den Kern der zusammengesetzten Matrix $\tilde{A}$ zu bestimmen.
Ein alternativer Ansatz regularisiert die gesamte Hesse-Matrix $\nabla^2_{xx} L(x^{[k]}, \lambda^{[k]}, \mu^{[k]})$, um so die Voraussetzungen von Satz 3.1 zu erfüllen, auch wenn bei diesem Ansatz möglicherweise eine zu starke Regularisierung durchgeführt wird.

Der folgende Satz aus dem Buch von Plato [84, cf. Theorem 12.9] liefert eine numerisch leicht auszuwertende Abschätzung für die Eigenwerte einer Matrix.

Satz 3.15 (Gerschgorin-Kreise). *Es sei eine Matrix $M \in \mathbb{R}^{N_x \times N_x}$ mit den Einträgen $m_{ij}, i, j \in \{1, ..., N_x\}$ gegeben. Für das Spektrum der Matrix $\sigma(M)$ gilt*

$$\sigma(M) \subset \bigcup_{i=1}^{N_x} G_i,$$

wobei die Gerschgorin-Kreise G_i durch

$$G_i := \{z \in \mathbb{C} : |z - m_{ii}| \leq \sum_{\substack{i=1 \\ i \neq j}}^{N_x} |m_{ij}|\} \quad i = 1, ..., N_x \tag{3.120}$$

definiert sind.
Zusätzlich gilt, dass wenn die Vereinigung von q Gerschgorin-Kreisen

$$K^{(1)} := G_{i_1} \cup ... \cup G_{i_q} \quad (i_l \neq i_m \ \textit{für} \ l \neq m),$$

disjunkt von der Vereinigung $K^{(2)}$ der restlichen $N_x - q$ Gerschgorin-Kreise ist, so enthält $K^{(1)}$ genau q Eigenwerte von M und $K^{(2)}$ genau $N_x - q$ Eigenwerte von M (jeweils entsprechend der algebraischen Vielfachheit gezählt).

Beweis. Im Buch von Plato [84] wird dieser Satz bewiesen. □

Es seien $G_i^{[k]}, i = 1, ..., N_x$ im Folgenden die Gerschgorin-Kreise der Matrix $\nabla^2_{xx} L(x^{[k]}, \lambda^{[k]}, \mu^{[k]})$. Mit Hilfe der Kreise kann leicht eine Mindestgröße für die Regularisierung zur Sicherstellung der positiven Definitheit der Hesse-Matrix angegeben werden. Die Definition

$$\sigma_{min}^{[k]} := \min_{\sigma \in \bigcup_{i=1}^{N_x} G_i^{[k]}} \sigma \tag{3.121}$$

bietet eine Abschätzung für den kleinsten Eigenwert. Bei der Regularisierung innerhalb von WORHP ist das Ziel, dass der kleinste Eigenwert mindestens den Wert 1 hat[4]. Somit muss eine Regularisierung vorgenommen werden, falls $\sigma_{min}^{[k]} \leq 1$ erfüllt ist. Die Matrix

$$\nabla^2_{xx} L(x^{[k]}, \lambda^{[k]}, \mu^{[k]}) + \left(\max(0, 1 - \sigma_{min}^{[k]}) \right) I_{N_x} \tag{3.122}$$

erfüllt die gewünschte Eigenschaft, da der kleinste Eigenwert zur 1 verschoben wird. Wie bereits erwähnt, ist theoretisch nur die positive Definitheit der Hesse-Matrix

[4]Diese Grenze kann über einen internen Parameter gesteuert werden (Vergleiche [98] "Hessian Regularisation").

auf dem Kern der aktiven Nebenbedingungen gefordert, deshalb führt die Verwendung der Abschätzung (3.121) zu einer übermäßig starken Regularisierung. Betts [5] schlägt deshalb eine abgeschwächte Regularisierung mit

$$\nabla^2_{xx} L(x^{[k]}, \lambda^{[k]}, \mu^{[k]}) + \tau \left(\max(0, 1 - \sigma^{[k]}_{min}) \right) I_{N_x} \tag{3.123}$$

vor. Der Dämpfungsparameter $\tau \in [0, 1]$ wird eingeführt um diesem Umstand Sorge zu tragen. Weiterhin sagt Satz 3.15 lediglich aus, dass die Eigenwerte innerhalb der Kreise, aber nicht zwangsläufig auf dem Rand liegen. Betts bezeichnet τ als Levenberg-Parameter motiviert durch die Ähnlichkeit dieser Regularisierung zum Levenberg-Marquart Verfahren. Mit Hilfe einer iterativen Anpassung des Levenberg-Parameters wird versucht die Auswirkungen der Regularisierung gering, aber gleichzeitig ausreichend zur Lösung der quadratischen Unterprobleme, zu halten.
Im Gegensatz zur Verwendung der Gerschgorin-Kreise ist die exakte numerische Bestimmung von Eigenwerten zu aufwendig, um dies während der Optimierung durchzuführen. Deshalb ist es insbesondere schwierig die Qualität der verwendeten Regularisierung im Bezug auf die reduzierte Hesse-Matrix zu überprüfen. Trotzdem existieren Möglichkeiten Informationen über den Erfolg der Regularisierung zu gewinnen. Die folgende Definition hilft dies zu präzisieren.

Definition 3.16. *Die* Signatur *einer Matrix M ist das folgende Zahlentripel.*

$$\text{inertia}(M) := (|\lambda_+|, |\lambda_-|, |\lambda_0|) \quad \textit{mit} \quad \begin{cases} \lambda_+ & := \{\lambda > 0 \mid \lambda \in \sigma(M)\} \\ \lambda_- & := \{\lambda < 0 \mid \lambda \in \sigma(M)\} \\ \lambda_0 & := \{\lambda = 0 \mid \lambda \in \sigma(M)\} \end{cases} \tag{3.124}$$

Die Signatur beschreibt die Verteilung der Eigenwerte.

Der folgende Satz von Gould [56] liefert eine wichtige Aussage unter Verwendung der Signatur der KKT-Matrix.

Satz 3.17. *Es seien Matrizen $H \in \mathbb{R}^{N_x \times N_x}$ und $A \in \mathbb{R}^{N_h \times N_x}$ gegeben. Die Matrix A habe vollen Zeilenrang. Es sei die KKT-Matrix*

$$K := \begin{pmatrix} H & A^\top \\ A & 0 \end{pmatrix} \tag{3.125}$$

gegeben. Es gilt $p^\top H p > 0$ für alle $\|p\|_2 > 0$ mit $A^\top p = 0$ genau dann, wenn $\text{inertia}(K) = (N_x, N_h, 0)$ *erfüllt ist.*

Beweis. Der Beweis findet sich zum Beispiel in den Arbeiten von Gould [56] oder von Higham und Cheng [63]. □

Der Satz 3.17 kann bei Verwendung einer Aktiven-Mengen-Strategie zur Lösung der quadratischen Unterprobleme direkt angewendet werden. Im Gegensatz dazu wird in WORHP allerdings ein Innere-Punkte-Verfahren, wie in Algorithmus 2 verwendet. Somit wird die Matrix K aus Gleichung (3.125) nicht direkt verwendet. Deshalb muss eine allgemeinere Aussage gefunden werden, um Rückschlüsse auf die verwendete Regularisierung treffen zu können. In der Arbeit von Forsgren [38, Proposition 2] findet sich die folgende Aussage.

Satz 3.18. *Es sei $\tilde{K}$ durch*

$$\tilde{K} := \begin{pmatrix} H & J^\top \\ J & -D \end{pmatrix} \tag{3.126}$$

mit symmetrischem $H \in \mathbb{R}^{n\times n}$, $J \in \mathbb{R}^{m\times n}$, symmetrischem und positiv semidefinitem $D \in \mathbb{R}^{m\times m}$ gegeben. Weiterhin sei $r \in \mathbb{N}$ durch

$$r := rang(\begin{pmatrix} J & -D \end{pmatrix})$$

definiert. Ferner sei U_0 eine Matrix deren Spalten eine Basis für den Kern von D bilden und es sei N eine Matrix deren Spalten eine Basis für den Kern von $U_0^\top J$ bilden. Dann gilt, dass

$$\text{inertia}(\tilde{K}) = \text{inertia}(N^\top(H + J^\top D^- J)N) + (m_0 - m + r, r, m - r) \tag{3.127}$$

und

$$rang(U_0^\top J) = m_0 - m + r.$$

Durch D^- wird die Pseudoinverse von D bezeichnet und weiterhin ist

$$m_0 = dim(\ker(D)).$$

Beweis. Der Satz wird in der Arbeit von Forsgren [38] bewiesen. □

Im Folgenden wird der Satz auf die Matrix K_{ip} mit

$$K_{ip} := \begin{pmatrix} \bar{Q} & A^\top & C^\top \\ A & 0 & 0 \\ C & 0 & -Z^{-1}S \end{pmatrix} \tag{3.128}$$

aus dem Innere-Punkte-Verfahren für quadratische Optimierungsprobleme aus Abschnitt 3.1.2.2 angewendet. Die Matrix K_{ip} beinhaltet bereits die getrennt integrierten Boxschranken an die Optimierungsvariablen.
Das Ziel ist es einen Zusammenhang zwischen der Signatur von K_{ip} und der positiven Definitheit der verwendeten Hesse-Matrix $\bar{Q}$ auf dem Kern der aktiven Nebenbedingungen während der Optimierung analog zum Satz 3.17 zu finden.

Folgerung 3.19. *Es sei die Matrix K_{ip} mit $\bar{Q} \in \mathbb{R}^{N_x \times N_x}, A \in \mathbb{R}^{N_h \times N_x}, C \in \mathbb{R}^{N_g \times N_x}$ gegeben. Die Zeilen von A seien linear unabhängig. Weiterhin seien $Z, S \in \mathbb{R}^{N_g \times N_g}$ Diagonalmatrizen wie in Abschnitt 3.1.2.2 mit $z_i > 0$ und $s_i > 0$ für $i = 1, ..., N_g$. Ferner sei $x^\top \bar{Q} x > 0$ für alle $x \in \ker A$. Dann gilt*

$$\mathit{inertia}(K_{ip}) = (N_x, N_h + N_g, 0) \tag{3.129}$$

für die Signatur der Matrix K_{ip}.

Beweis. Zunächst wird gezeigt, dass die vorliegende Matrix inklusive aller Komponenten die Voraussetzungen von Satz 3.18 erfüllt. Danach kann über das Resultat die Signatur der Matrix K_{ip} bestimmt werden.
Die Notationen des Satzes liefern $H = \bar{Q}$ und somit $n = N_x$. Weiterhin ist

$$J = \begin{pmatrix} A \\ C \end{pmatrix},$$

und deshalb $m = N_h + N_g$. Die letzte Komponente der Matrix $\tilde{K}$ ist durch

$$D = \begin{pmatrix} 0 & 0 \\ 0 & Z^{-1}S \end{pmatrix} \tag{3.130}$$

gegeben und durch $z_i > 0$ und $s_i > 0$ ist diese positiv semidefinit. Somit kann auch r bestimmt werden. Mit Hilfe der Definition ergibt sich

$$\begin{aligned} r &= \operatorname{rang}(\begin{pmatrix} J & -D \end{pmatrix}) && (3.131)\\ &= \operatorname{rang}\left(\begin{pmatrix} A & 0 & 0 \\ C & 0 & -Z^{-1}S \end{pmatrix}\right) && (3.132)\\ &= N_h + N_g, && (3.133) \end{aligned}$$

da die Zeilen von A linear unabhängig sind und offensichtlich auch $-Z^{-1}S$ nur linear unabhängige Zeilen enthält (unabhängig von C). Aus Gleichung (3.130) ergibt sich außerdem, dass

$$m_0 \coloneqq \dim(\ker(D)) = N_h \tag{3.134}$$

ist. Damit folgt weiterhin $U_0 \in \mathbb{R}^{N_h + N_g \times N_h}$ und

$$U_0 \coloneqq \begin{pmatrix} I_{N_h} \\ 0_{N_g \times N_h} \end{pmatrix} \tag{3.135}$$

erfüllt die geforderten Voraussetzungen. Mit Hilfe von Gleichung (3.135) kann $U_0^\top J = A$ ermittelt werden. Es wurde vorausgesetzt, dass die Zeilen von A linear unabhängig sind. Somit ist $\dim(\operatorname{im}(A)) = N_h$ und aus dem Dimensionssatz für lineare Abbildungen folgt $\dim(\ker(A)) = N_x - N_h$. Zuletzt wird noch die Pseudoinverse D^- benötigt. Ein naheliegender Kandidat ist die Matrix

$$D^- \coloneqq \begin{pmatrix} 0 & 0 \\ 0 & S^{-1}Z \end{pmatrix}. \tag{3.136}$$

Im Buch von Deuflhardt [24] findet sich in Satz 3.16, dass die Pseudoinverse eindeutig ist, wenn die folgenden sogenannten Penrose-Axiome erfüllt sind:

$$\begin{aligned} DD^-D &= D \\ D^-DD^- &= D^- \\ (DD^-)^\top &= DD^- \\ (D^-D)^\top &= D^-D \end{aligned}$$

Die Gültigkeit dieser Gleichungen lässt sich leicht für die Matrix D^- nachrechnen. Somit kann der Satz 3.18 angewendet werden und es folgt:

$$\begin{aligned} \text{inertia}(K_{ip}) &= \text{inertia}\left(N^\top\left(\bar{Q} + \left(A^\top C^\top\right)\begin{pmatrix}0 & 0\\ 0 & S^{-1}Z\end{pmatrix}\begin{pmatrix}A\\ C\end{pmatrix}\right)N\right) \\ &\quad + (N_h - (N_h+N_g) + (N_h+N_g), N_h+N_g, N_h+N_g-(N_h+N_g)) \\ &= \text{inertia}(N^\top(\bar{Q}+C^\top S^{-1}ZC)N) + (N_h, N_h+N_g, 0) \qquad (3.137) \end{aligned}$$

Zuletzt muss deshalb die Signatur der Matrix $N^\top(\bar{Q}+C^\top S^{-1}ZC)N$ bestimmt werden. Es sei $x \in \mathbb{R}^{N_x-N_h}, x \neq 0$ beliebig gewählt. Weiterhin seien $\xi := Nx$ und $\nu := CNx$ definiert. Auf Grund der linearen Unabhängigkeit der Spalten von N folgt aus $x \neq 0$ direkt $\xi \neq 0$. Dann ergibt sich

$$\begin{aligned} x^\top N^\top(\bar{Q}+C^\top S^{-1}ZC)Nx &= x^\top N^\top \bar{Q}Nx + x^\top N^\top C^\top S^{-1}ZCNx \\ &= \xi^\top \bar{Q}\xi + \nu^\top S^{-1}Z\nu > 0, \end{aligned}$$

weil $\xi \in \text{im}(N) = \ker(A)$ und $\bar{Q}$ positiv definit auf $\ker(A)$ gilt und S sowie Z Diagonalmatrizen mit lediglich positiven Einträgen sind, woraus auch $\nu^\top S^{-1}Z\nu \geq 0$ folgt. Weiterhin ist $N^\top(\bar{Q}+C^\top S^{-1}ZC)N \in \mathbb{R}^{N_x-N_h \times N_x-N_h}$ und somit ist

$$\text{inertia}(N^\top(\bar{Q}+C^\top S^{-1}ZC)N) = (N_x - N_h, 0, 0).$$

Dadurch ergibt sich zusammen mit Gleichung (3.137) die Behauptung. □

Bemerkung 3.20. *In Folgerung 3.19 wurde vorausgesetzt, dass die Schlupfvariablen $s_i > 0$ für $i = 1, ..., N_g$ erfüllen. Existieren Schlupfvariablen $s_j = 0$ mit $j \in [1, ..., N_g]$, so bleibt die Aussage der Folgerung trotzdem gültig. Die entsprechenden Zeilen der Matrix C können für die Beweisführung in die Matrix A verschoben werden. In diesem Fall entspricht die Matrix A dann der in Gleichung* (3.118) *eingeführten zusammengesetzten Matrix $\tilde{A}$. Es handelt sich hierbei lediglich um eine korrekte Einbindung der aktiven Menge und es behalten alle Überlegungen in dieser Situation ihre Gültigkeit.*

In der numerischen Praxis können aus dem Beweis der Folgerung wichtige Erkenntnisse gewonnen werden. Wenn bei der Überprüfung der Signatur der Systemmatrix des Innere-Punkte-Verfahrens nicht die passenden Eigenwertverteilungen abgelesen

wurden, so muss eine der Voraussetzungen verletzt sein. Bei genauerer Betrachtung bleiben nur zwei mögliche Situationen übrig.
Einerseits besteht die Möglichkeit, dass die verwendete Hesse-Matrix $\bar{Q}$ die benötigten spektralen Eigenschaften nicht aufweist. Hierbei ist zu beachten, dass die Integration der Boxschranken auf diese Aussage keine Auswirkung hat. Gleichung (3.61) zeigt, dass bei der Integration lediglich eine positiv semidefinite Matrix addiert wird. Somit gilt, dass wenn $\bar{Q}$ nicht positiv definit auf dem Kern der zusammengesetzten Matrix $\tilde{A}$ ist, dann gilt dies erst recht nicht für Q.
Andererseits kann die Situation auftreten, dass die Matrix $\tilde{A}$ nicht vollen Zeilenrang hat. Motiviert über das Ziel die Regularisierung der Hesse-Matrix geeignet zu überprüfen liefert demnach Folgerung 3.19 zusätzlich Hinweise auf die Inkonsistenz der linearisierten Nebenbedingungen. Die Behandlung dieser Situation wird im Abschnitt 3.3.3.3 weiter thematisiert.
Zusammenfassend bleibt festzuhalten, dass in beiden Fällen die Lösbarkeit des quadratischen Unterproblems nicht garantiert werden kann.
Wird bei der Lösung des QP eine Faktorisierung der Systemmatrix (beispielsweise zur Lösung von (3.55)) vorgenommen, so kann, bei Verwendung eines geeigneten Verfahrens, die Signatur abgelesen werden [5]. Auf diese Weise kann während der Lösung des quadratischen Unterproblems überprüft werden, ob die vorgenommene Regularisierung erfolgreich war oder diese gegebenenfalls angepasst werden muss.
Die Regularisierung der Hesse-Matrix ist ein nötiges Mittel, um die Lösbarkeit der quadratischen Unterprobleme sicherzustellen. Allerdings führt eine zu starke Regularisierung auf Suchrichtungen ähnlich zum Gradientenverfahren und somit auf lediglich lineare Konvergenzgeschwindigkeit, im Gegensatz zur quadratischen Konvergenz in der Nähe eines lokalen Minimums unter Verwendung der exakten Hesse-Matrix, wenn die entsprechenden Voraussetzungen von Satz 3.5 erfüllt sind. Durch die abgeschwächte Regularisierung nach Gleichung (3.123) und durch die Überprüfung der Signatur kann ein Kompromiss zwischen der Abschätzung aller Eigenwerte der KKT-Matrix mit Hilfe der Gerschgorin-Kreise und einer exakten Überprüfung der Eigenwerte der reduzierten Hesse-Matrix mit Hilfe einer QR-Faktorisierung der Matrix $\tilde{A}$ gefunden werden.
Innerhalb des SQP-Verfahrens müssen die spektralen Eigenschaften der verwendeten Hesse-Matrix bei Verwendung der analytischen Hesse-Matrix oder bei der Anwendung finiter Differenzen sichergestellt werden. BFGS-Matrizen werden entsprechend der Darstellungen in Abschnitt 3.3.2.3 so aufgebaut, dass sie bereits positiv definit sind und deshalb keine Regularisierung benötigen.
Die Umsetzung von WORHP enthält zu diesem Zweck eine Implementierung der Strategie nach Betts [5]. Gleichung (3.123) beschreibt die Korrektur. Mit dieser Technik wird die Hesse-Matrix angepasst bevor das quadratische Unterproblem mit Hilfe von QPSOL gelöst wird. Ist QPSOL nicht in der Lage eine Lösung zu finden wird davon ausgegangen, dass die vorgenommene Regularisierung nicht ausreichend war. Der Dämpfungsparameter τ wird iterativ weiter erhöht bis eine Lösung gefunden wurde. Alternativ muss der Algorithmus gestoppt werden, wenn die Regularisierung

eine maximale Schranke überschreitet. In diesem Fall würde eine Rettungsstrategie, wie in den Arbeiten von Wassel [109] oder Nikolayzik [81] beschrieben, angewendet werden. Die Realisierung des Regularisierungsalgorithmus innerhalb von WORHP ist in Algorithmus 5 vereinfacht dargestellt.

Algorithmus 5 Regularisierung der Hesse-Matrix innerhalb von WORHP

1: Bestimme $\sigma_{\min}$ mit Hilfe der Gerschgorin Kreise
2: Wähle $\tau_{[0]} = 9.2 \cdot 10^{-4}$, $\rho_\tau = 2.0$, $\tau_{\max} = 1 \cdot 10^7$
3: Setze $j = 0$
4: **repeat**
5: **if** $j > 0$ **then**
6: $\tau_{[j]} \leftarrow \tau_{[j-1]}\rho_\tau$
7: **end if**
8: $Q \leftarrow \nabla^2_{xx} L(x^{[k]}, \lambda^{[k]}, \mu^{[k]}) + \tau_{[j]} \left(\max(0, 1 - \sigma^{[k]}_{min}) \right) I_{N_x}$
9: Löse das quadratische Unterproblem (3.152)
10: $j \leftarrow j + 1$
11: **until** QP erfolgreich gelöst oder $\tau_{[j]} > \tau_{\max}$

Bemerkung 3.21. *In der tatsächlichen Implementierung wird zusätzlich die Signatur der Systemmatrix des Innere-Punkte-Algorithmus betrachtet, um den Aussagen von Folgerung 3.19 Sorge zu tragen. Weiterhin werden je nach gewählter Regularisierungsstrategie zusätzlich Vorhersagen analog zu den Ideen eines Trust-Region-Verfahrens verwendet, um die Erhöhung des Parameters $\tau_{[j]}$ gezielt zu kontrollieren. Die Beschreibung des Algorithmus soll lediglich die Ideen verdeutlichen, um die im weiteren Verlauf der Arbeit diskutierten Strategien zur Effizienzsteigerung dieser Komponente des SQP-Verfahrens zu motivieren.*

Bemerkung 3.22. *Algorithmus 5 offenbart die große Schwäche dieses Regularisierungsansatzes. Um eine möglichst gute Suchrichtung bei der Lösung des quadratischen Unterproblems zu erhalten, sollte der Faktor $\tau_{[j]}$ so klein wie möglich sein. Werden allerdings $\tau_{[0]}$ und ρ_τ zu klein gewählt, so wird möglicherweise mehrmals vergeblich versucht ein quadratisches Unterproblem zu lösen. Gerade dieser Schritt ist aber, wie bereits mehrfach betont wurde, der numerisch aufwendigste Teil bei der Lösung eines nichtlinearen Optimierungsproblems.*
Zu starke Zurückhaltung bei der Regularisierung der Hesse-Matrix wird demnach durch erhöhten numerischen Aufwand bestraft. Eine zu starke Regularisierung führt hingegen auf potenziell schlechte Suchrichtungen, welche wiederum die Anzahl der Hauptiterationen unnötig erhöhen und somit die Lösung weiterer quadratischer Unterprobleme erforderlich machen.

Die Betrachtungen verdeutlichen die zentrale Bedeutung der effizienten Regularisierung der Hesse-Matrix innerhalb des SQP-Algorithmus. Im folgenden Abschnitt wird eine neue Strategie zur Überwindung der gerade beschriebenen Probleme diskutiert.

3.3.3.2 Konfigurationsvariationen

Die beschriebene Problematik, dass auf Grund mangelhafter Regularisierung zusätzliche quadratische Unterprobleme gelöst werden müssen, rechtfertigt es, diese Situation weiter zu hinterfragen. In anderen Lösern wie zum Beispiel SNOPT von Gill, Saunders und Murray werden lediglich BFGS-Variationen verwendet [49], um diese Schwierigkeiten zu umgehen. In einer späteren Veröffentlichung betonen Gill, Saunders und Wong [51], dass für bestimmte Problemstellungen die BFGS-Matrizen der analytischen Hesse-Matrix schlicht überlegen sind, weil keine schlecht gestellten quadratischen Unterprobleme entstehen und somit die Problematik der Regularisierung der Hesse-Matrix nicht gelöst werden muss. Nach der Theorie konvergiert das SQP-Verfahren lokal zwar unter gewissen Voraussetzungen bei Verwendung der analytischen Hesse-Matrix quadratisch, aber auf Grund der beschriebenen Probleme bei der Lösung der Unterprobleme erscheinen die Beobachtungen von Gill, Saunders und Wong plausibel. In ihrer Veröffentlichung entwickeln sie daraufhin verschiedene Techniken zur Regularisierung, die teilweise ähnlich zu den beschriebenen Ideen von Betts sind. Diese werden um Techniken zur Regularisierung während und nach der Lösung der Unterprobleme erweitert.
An dieser Stelle wird eine alternative Vorgehensweise vorgestellt. Es seien Radien $0 < \tilde{r} \in \mathbb{R}$ und $0 < \hat{r} \in \mathbb{R}$ gegeben. Die Theorie garantiert die quadratische Konvergenz lediglich in einer gewissen Umgebung $U_1 = U_{\tilde{r}}(x^*) \subset \mathbb{R}^{N_x}$ um ein lokales Minimum. Weiterhin gilt auf Grund der bereits in Kapitel 2 angenommenen Stetigkeit der zweiten Ableitung, dass es eine weitere Umgebung $U_2 = U_{\hat{r}}(x^*) \subset \mathbb{R}^{N_x}$ gibt, so dass die analytische Hesse-Matrix positiv definit auf dem Kern der Jacobi-Matrix der aktiven Nebenbedingungen innerhalb von U_2 ist. Insbesondere für Iterierte $x^{[k]} \in \mathbb{R}^{N_x}$ während der Globalisierungsphase des Optimierungsalgorithmus ist zu erwarten, dass diese in keiner der Umgebungen U_1 oder U_2 liegen.
Es erscheint somit naheliegend, in diesem Fall in den ersten Iterationen nicht die analytische Hesse-Matrix zu verwenden, da weder die Definitheitseigenschaft gegeben ist, noch der Vorteil bezüglich der Konvergenzrate zu erwarten ist. Stattdessen kann eine geeignete BFGS-Matrix verwendet werden, um von deren inhärent gegebener positiver Definitheit zu profitieren und auf diese Weise die Lösbarkeit der quadratischen Unterprobleme sicherzustellen. Je nach gewählter BFGS-Technik kann sich die resultierende Struktur zusätzlich auf die Effizienz des Lösers für lineare Gleichungssysteme auswirken. Beispielsweise liefern Block-BFGS-Verfahren, wie in Abschnitt 3.3.2.3 beschrieben, Bandmatrizen innerhalb des oberen linken Blocks in der zusammengesetzten Matrix aus den Gleichungen (3.55) beziehungsweise (3.62) innerhalb des Innere-Punkte-Verfahrens für die quadratischen Unterprobleme. Diese können von geeigneten Verfahren besonders effizient gelöst werden. Nähern sich die Iterierten einem lokalen Minimum, so wird, wenn vorhanden, für die restliche Laufzeit des Algorithmus die analytische Hesse-Matrix verwendet. Liegt diese nicht vor, könnte es lohnenswert sein, statt weiterhin die BFGS-Matrizen zu verwenden, finite Differenzen zur Approximation der analytischen Hesse-Matrix einzuschalten und so

eine bessere Konvergenzrate zu erzielen.

3.3.3.2.1 Technische Aspekte Bei der praktischen Umsetzung muss beachtet werden, dass die Verwaltung der Hesse-Matrix einen wesentlichen Teil des Speicherbedarfs des gesamten Algorithmus ausmacht. Effiziente Implementierungen speichern nur das obere oder untere Dreieck der Hesse-Matrix, da diese unter der vorausgesetzten Stetigkeit der zweiten Ableitung nach dem Satz von Schwarz symmetrisch ist. Trotzdem kann je nach Schwachbesetztheit der Speicher auch in heutigen Rechnern die Anzahl der verwendbaren Nichtnulleinträge schlimmstenfalls limitieren. Nichtsdestotrotz werden in der Umsetzung der hier beschriebenen Ideen innerhalb des Lösers WORHP die analytische Hesse-Matrix und eventuell zu verwendende BFGS-Matrizen getrennt gespeichert, um einen fließenden Wechsel zu ermöglichen und bei der Einbindung des Algorithmus größtmögliche Flexibilität zu garantieren. Diese Vorgehensweise bietet verschiedene Vorteile. Durch die Aufteilung der Matrizen verringert sich der Verwaltungsaufwand innerhalb des Programmes. Die Betrachtungen im Abschnitt 3.3.2.3 haben gezeigt, dass BFGS-Matrizen, im Gegensatz zur Bestimmung der Hesse-Matrix mittels finiter Differenzen, auch abweichende Strukturen aufweisen. Durch eine klare Trennung der Daten beider Matrixarten innerhalb des Programmes wird dieses auch in der Datenstruktur deutlich und mögliche Fehlerquellen in der Umsetzung werden vermieden.
Weiterhin ist es in diesem Fall möglich, zu jedem Zeitpunkt frei zu entscheiden, welche Matrix zum Einsatz kommt, ohne dass bei jedem Wechsel Strukturen und Werte neu bestimmt werden müssen und gegebenenfalls sogar der reservierte Speicher angepasst werden muss. Diese Aspekte fördern die Wart- und Erweiterbarkeit der Software, da bei künftigen Entwicklungen der Zugriff auf die unterschiedlichen Matrixvarianten klar strukturiert ist. Insgesamt rechtfertigt die gewonnene Flexibilität den zusätzlichen Speicherbedarf.
Sollte die analytische Hesse-Matrix bei einem sehr großen Problem bereits viel Speicher benötigen, ist es in der Regel sowieso ratsam direkt auf ein sparsameres, schwachbesetztes BFGS-Verfahren zu wechseln.

3.3.3.2.2 Praktische Umsetzung Der Löser WORHP basiert auf dem Prinzip der Reverse-Communication [109]. Der Programmablauf ist in sogenannte *Stufen* (engl. stages) unterteilt. Diese Struktur erlaubt es eine neue Stufe für den Wechsel von der Globalisierungsphase in die Annäherungsphase an das lokale Minimum zu implementieren. In der praktischen Umsetzung bedeutet dies, dass in jeder Iteration, bevor das quadratische Unterproblem erstellt wird, geprüft werden muss, ob sich die aktuelle Iterierte noch im Globalisierungsbereich oder bereits im Einzugsbereich des lokalen Minimums befindet. Wird ein Eintritt in die Zone quadratischer Konvergenz vermutet, werden die entsprechenden Einstellungen des Lösers angepasst und die analytische Hesse-Matrix aktiviert.
Für viele Anwender kann auch der Wechsel von einer BFGS-Matrix auf eine mittels

finiter Differenzen ermittelter Hesse-Matrix interessant sein. Auch wenn die Berechnung der Hesse-Matrix durch finite Differenzen schlecht konditioniert ist, weist diese potenziell dieselben Konvergenzeigenschaften wie die analytische Hesse-Matrix auf und ist im Bezug darauf einer BFGS-Näherung überlegen.
Die Implementierung der BFGS-Methoden innerhalb von WORHP erlaubt es in jedem Schritt lediglich die Einheitsmatrix zu verwenden und auf diese Weise ein Gradientenverfahren nachzubilden. Diese Variante kann für bestimmte Problemfälle besonders interessant sein, da die entstehenden zu lösenden linearen Gleichungssysteme numerisch besonders effizient zu lösen sind. Deshalb wird diese Variante als spezielle Version der hier vorgestellten Technik angesehen und separat bereitgestellt.
Die Detektion des Überganges von der Globalisierungsphase in die lokale Annäherungsphase bietet Diskussionspotenzial. Die Radien $\tilde{r}$ und $\hat{r}$ der relevanten Umgebungen sind nicht bekannt und auch der Abstand zum lokalen Minimum $\|x^{[k]} - x^*\|$ ist während der Optimierung unbekannt.
Die einfachste Variante ist die Verwendung einer festen Anzahl Iterationsschritte innerhalb der Globalisierungsphase mit anschließendem Wechsel zur lokalen Phase ohne weitere Bedingungen zu prüfen. Dieser Ansatz ist mangelhaft, da die Anzahl der nötigen Schritte zum Erreichen der lokalen Annäherungsphase offensichtlich stark von der Qualität der gewählten Startschätzung abhängt.
In der Masterarbeit des Autors [41] wurde ein alternativer Ansatz zur Anpassung des Lösers innerhalb der Globalisierungsphase diskutiert. Dort wurde die Gleitpunktzahlgenauigkeit des linearen Lösers angepasst, um auf diesem Weg die Effizienz des Verfahrens zu steigern. In den dortigen Ausführungen wurde als naheliegende Strategie vorgeschlagen die Werte der Abbruchbedingungen in der aktuellen Iterierten $x^{[k]}$ als Maß für die Entfernung zum lokalen Minimum zu verwenden.

Definition 3.23. *Es seien die Optimalitätstoleranz* $0 < \epsilon_{opt} \in \mathbb{R}$ *und die Zulässigkeitstoleranz* $0 < \epsilon_{con} \in \mathbb{R}$ *gegeben. Die Abschätzungen*

$$\left\|\nabla_x L(x^{[k]}, \lambda^{[k]}, \mu^{[k]})\right\|_\infty < \sqrt{\epsilon_{opt}} \tag{3.138}$$

$$\left\|h(x^{[k]}), \max\{0, g(x^{[k]})\}\right\|_\infty < \sqrt{\epsilon_{con}} \tag{3.139}$$

stellen zusammen die toleranzbasierte Phasenwechselbedingung *dar.*

Bemerkung 3.24. *Eine gesonderte Betrachtung der Komplementaritätsbedingung findet nicht statt. Zur Beurteilung der Annäherung an ein lokales Minimum hat sich in den numerischen Untersuchungen die Betrachtung der Optimalität und Zulässigkeit als ausreichend herausgestellt.*

Bemerkung 3.25. *Die toleranzbasierte Phasenwechselbedingung ist sehr konservativ. Je nach Anwendung und gewählten Toleranzen kann bereits das Einhalten dieser Bedingungen eine Iterierte als lokales Minimum qualifizieren. Tatsächlich werden die numerischen Untersuchungen zeigen, dass bereits ein früherer Wechsel lukrativ für die Effizienz des Verfahrens sein kann.*

Bemerkung 3.26. *Werden innerhalb von WORHP die von Nikolayzik [81] beschriebenen skalierten Abbruchbedingungen verwendet, ersetzen diese den Gradienten der Lagrange-Funktion in Gleichung* (3.138)*. Die Herleitung der skalierten Abbruchbedingungen zeigt, dass diese auf einer lokalen Taylor-Approximation basieren. Sei* $0 < \bar{r} \in \mathbb{R}$ *ein geeigneter Radius, dann gilt diese Approximation lediglich in einer häufig sehr kleinen Umgebung* $U_3 = U_{\bar{r}}(x^*)$*. Es kann für nichtlineare Probleme nicht garantiert werden, dass* $\tilde{r} < \bar{r}$ *gegeben ist, so dass diese Bedingung infolgedessen zur Detektion des Phasenwechsels nur bedingt geeignet ist.*

Das wichtigste Ziel bei der Verwendung unterschiedlicher Hesse-Matrizen während der Optimierung ist die Steigerung der Effizienz des Verfahrens und dies insbesondere indem die Lösung der quadratischen Unterprobleme stabiler und idealerweise auch schneller abläuft. Typischerweise lässt sich während der Iterationen von WORHP bei Verwendung von QPSOL mit der eingebauten Warmstartfunktionalität zur Lösung der quadratischen Unterprobleme beobachten, dass die Anzahl der benötigten Nebeniterationen zur Lösung der Unterprobleme im Laufe der Hauptiterationen abnimmt. Dieses Verhalten tritt auch auf wenn BFGS-Matrizen verwendet werden, so dass es sich nicht allein auf Grund von Krümmungseigenschaften der analytischen Hesse-Matrix bei Annäherung an ein lokales Minimum erklären lässt. Deshalb kann hieraus eine alternative WORHP-spezifische Phasenwechselbedingung abgeleitet werden. Es sei eine Schranke $0 < L$, beispielsweise etwa $L = 8$ gegeben. Benötigt QPSOL in zwei aufeinanderfolgenden Hauptiterationen weniger als L Iterationen, wird davon ausgegangen, dass sich das Verhalten des Lösers stabilisiert. In dieser Situation ist ein Wechsel auf die analytische Hesse-Matrix sinnvoll.

3.3.3.2.3 Weitere Möglichkeiten Durch die Einführung der zusätzlichen Stufe innerhalb der Architektur des Lösers bieten sich neben der vorgeschlagenen Technik zur Variation der verwendeten Hesse-Matrix und dem bereits erwähnten Wechsel der Gleitpunktzahlgenauigkeit des linearen Lösers weitere Ansätze zur Beeinflussung der Optimierung an. Im Folgenden werden einige Ideen vorgestellt, die für weitere Untersuchungen von Interesse sein könnten, von der eigentlichen Absicht im hier präsentierten Rahmen aber zu stark abweichen und deshalb nur kurz erwähnt, aber nicht weiter untersucht werden.
Die aktuelle Implementierung des Lösers WORHP verwendet zur Lösung der linearen Gleichungssysteme immer direkte Löser. Auf Grund der Wechselmöglichkeit während der Optimierung bietet es sich an näher zu untersuchen, ob es für die Effizienz des Lösungsalgorithmus förderlich sein könnte, während der Globalisierungsphase schnellere und dafür potenziell etwas ungenauere indirekte Verfahren zu verwenden. Die direkten Verfahren werden dann erst in der Nähe des lokalen Minimums eingesetzt, um die Genauigkeit der Lösung zu verbessern und so die Probleme bei der Verwendung von indirekten Verfahren zu vermeiden.
Bei der Umsetzung der Algorithmen gemischter Präzision in der Masterarbeit des Autors [41] wurde nicht nur die Gleitpunktzahlgenauigkeit des linearen Lösers in

der Globalisierungsphase angepasst, sondern es wurden auch die Toleranzen für die Abbruchbedingungen von QPSOL angepasst. Diese Vorgehensweise war im dortigen Zusammenhang nötig, um die Lösbarkeit auch bei verringerter Gleitpunktzahlgenauigkeit zu garantieren. Diese Idee lässt sich aber auch im allgemeinen SQP-Kontext betrachten. Die grundsätzliche Idee ist, dass Unterprobleme in der Globalisierungsphase lediglich näherungsweise gelöst werden. In dieser Phase sollte es ausreichend sein grobe Suchrichtungen zu bestimmen, um einen Fortschritt der Iterierten garantieren zu können. Auf diese Weise besteht die Möglichkeit den Rechenaufwand zur Lösung der Unterprobleme zu reduzieren und so die Effizienz des Verfahrens zu steigern. Mit Hilfe der Wechselmöglichkeit kann dies ohne großen Mehraufwand effektiv untersucht und bewertet werden.

3.3.3.3 Relaxierung der Nebenbedingungen

In jeder Iteration des SQP-Algorithmus wird ein quadratisches Unterproblem der Form (3.71) aufgestellt. Durch die Linearisierung der Nebenbedingungen können inkonsistente Nebenbedingungen auftreten, so dass diese Unterprobleme nicht lösbar sind.
Bei der Betrachtung dieser Problematik müssen innerhalb der nichtlinearen Optimierung zwei verschiedene Situationen unterschieden werden. Einerseits kann die beobachtete Inkonsistenz der linearen Nebenbedingungen auf Grund lokaler Eigenschaften der nichtlinearen Nebenbedingungen auftreten. In diesem Fall besteht für den Benutzer kein Handlungsbedarf und der Löser kann mit Hilfe einer geeigneten Relaxierung der linearen Nebenbedingungen, wie sie im weiteren Verlauf dieses Abschnitts beschrieben wird, den Optimierungsvorgang unter den nichtlinearen Nebenbedingungen fortsetzen.
Andererseits ist es möglich, dass auch die nichtlinearen Nebenbedingungen bereits die beobachtete Inkonsistenz aufweisen. In dieser Situation ist es für den Anwender des Lösers vorteilhaft, wenn bei einem Abbruch der Optimierung zusätzliche Informationen über die betroffenen Nebenbedingungen generiert werden können. Zur Behandlung dieser Aufgabenstellung finden sich in der Literatur, insbesondere aus dem Bereich der linearen Optimierung, verschiedene Algorithmen. Grundlegende Begriffe und eine gute Auswahl weiterer Quellen bietet der Artikel von Amaldi, Pfetsch und Trotter [2]. Ausführliche Beschreibungen finden sich weiterhin in dem Buch von Chinneck [19]. Spezielle Algorithmen werden beispielsweise in den Veröffentlichungen von Chinneck [18] oder Pfetsch [83] beschrieben. Eine Implementierung derartiger Techniken während der Lösung nichtlinearer Optimierungsprobleme innerhalb von IPOPT stellen Wan und Biegler [108] vor. Im weiteren Verlauf dieser Arbeit wird davon ausgegangen, dass die nichtlinearen Nebenbedingungen konsistent sind. Die Vorgehensweise zur Behandlung inkonsistenter linearisierter Nebenbedingungen wird im Folgenden vorgestellt.
Powell [86] schlägt eine Relaxierung der Nebenbedingungen vor, um trotz derartiger Inkonsistenzen die Lösbarkeit der quadratischen Unterprobleme sicherzustellen. Es

werden eine Relaxierungsvariable $\delta \in [0,1]$ und ein Bestrafungsparameter $\eta \in \mathbb{R}_+$ eingeführt. Zusätzlich seien noch die Aktivitätsindikatoren

$$\sigma_i := \begin{cases} 0, & \text{wenn } g_i(x^{[k]}) < 0, \text{ d.h. } g_i \text{ ist inaktiv} \\ 1, & \text{sonst} \end{cases} \tag{3.140}$$

definiert. Es sei angemerkt, dass die Aktivitätsindikatoren nicht der aktiven Menge aus Definition 2.7 entsprechen, da in Gleichung (3.140) auch unzulässige $g_i(x^{[k]}) > 0$ auftreten dürfen. Somit markieren Indikatoren in diesem Zusammenhang alle verletzten oder aktiven Nebenbedingungen.
Das einfach relaxierte quadratische Unterproblem in Iteration k

$$\begin{aligned} \min_{d \in \mathbb{R}^{N_x}, \delta \in [0,1]} \quad & \tfrac{1}{2} d^\top Q d \; + \\ & \nabla_x f(x^{[k]})^\top d + \tfrac{\eta}{2} \delta^\top \delta \\ \text{unter} \quad & g_i(x^{[k]})(1 - \sigma_i \delta) + \nabla_x g_i(x^{[k]})^\top d \leq 0, \quad i = 1, ..., N_g \\ & h_j(x^{[k]})(1 - \delta) + \nabla_x h_j(x^{[k]})^\top d = 0, \quad j = 1, ..., N_h \\ & \delta - 1 \leq 0, \\ & -\delta \leq 0, \end{aligned} \tag{3.141}$$

bietet dem Algorithmus einen zusätzlichen Freiheitsgrad, um durch Ausnutzung der Relaxierungsvariable δ bei festem η einfacher eine Lösung zu finden.
Das folgende Beispiel aus der Arbeit von Wassel [109] demonstriert das Verfahren und insbesondere wird dort betont, dass die Wahl einer zu einfachen Startschätzung, etwa $x^{[0]} = 0$, dazu neigt solche Probleme zu verursachen.

Beispiel 3.27. *Die nichtlineare Optimierungsaufgabe*

$$\begin{aligned} \min_{x \in \mathbb{R}} \quad & (x-1)^2 \\ \textit{unter} \quad & 1 - x^2 \leq 0 \end{aligned} \tag{3.142}$$

wird mit dem globalen SQP-Algorithmus (Vergleiche Algorithmus 4) mit Startpunkt $x^{[0]} = 0, \lambda^{[0]} = 0$ *gelöst. Das quadratische Unterproblem* (3.71) *in der ersten Iteration*

$$\begin{aligned} \min_{d \in \mathbb{R}} \quad & d^2 - 2d \\ \textit{unter} \quad & 1 \leq 0 \end{aligned} \tag{3.143}$$

weist auf Grund der Linearisierung eine Inkonsistenz in der Nebenbedingung auf und ist deshalb nicht lösbar. Durch Relaxierung der Nebenbedingung ergibt sich mit einem festen Bestrafungsparameter $\eta > 0$ *entsprechend Gleichung* (3.141) *das folgende quadratische Unterproblem:*

$$\begin{aligned} \min_{(d,\delta) \in \mathbb{R} \times \mathbb{R}} \quad & d^2 - 2d + \tfrac{\eta}{2} \delta^2 \\ \textit{unter} \quad & \hat{g}_1(\delta) = 1 - \delta \leq 0 \\ & \hat{g}_2(\delta) = -\delta \leq 0 \\ & \hat{g}_3(\delta) = \delta - 1 \leq 0 \end{aligned} \tag{3.144}$$

Die Suchrichtung d taucht in diesem Fall nicht in den Nebenbedingungen auf, so dass aus der notwendigen Bedingung erster Ordnung direkt $d = 1$ folgt. Die Nebenbedingungen $\hat{g}_1$ und $\hat{g}_3$ liefern zusammen $\delta = 1$. In diesem Fall handelt es sich um ein Minimum, dass die Fritz-John Bedingungen nach Satz 2.16 erfüllt. Die allgemeinen KKT-Bedingungen sind nicht erfüllt, da die Regularitätsbedingung der linearen Unabhängigkeit verletzt ist. Trotzdem kann mit der Lösung $(d, \delta) = (1, 1)$, die hier sogar unabhängig vom Bestrafungsparameter η ist, der SQP-Algorithmus fortgesetzt werden und mit der Schrittweite $\alpha^{[0]} = 1$ wird das globale Minimum von (3.142) *$x^* = 1$ erreicht.*

Der Vektor $(d, \delta) = (0, 1)$ ist immer eine Lösung für (3.141). Tritt diese Lösung auch für größere Bestrafungsparameter η auf, weist dies auf inkonsistente Nebenbedingungen hin, für die keine Suchrichtung $\|d\|_2 > 0$ gefunden werden kann. In diesem Fall ist eine Umformulierung der Nebenbedingungen oder eine alternative Startschätzung erforderlich. Ergibt sich hingegen die Lösung $(d, \delta) = (d^*, 0)$, so ist d^* auch die optimale Lösung für das unrelaxierte Problem (3.71).
Das folgende Beispiel verdeutlicht die Auswirkungen des Bestrafungsparameters η auf die Lösung des relaxierten quadratischen Unterproblems (3.141).

Beispiel 3.28. *Es sei die quadratische Minimierungsaufgabe*

$$\begin{aligned} \min_{d\in\mathbb{R}^2,\delta\in[0,1]} \quad & -3d_1^2 + 2d_1d_2 + d_2^2 + \tfrac{\eta}{2}\delta^2 \\ \text{unter} \quad & (1-\delta)5 + 17d_1 - 3d_2 \quad \leq 0, \\ & (1-\delta)7 - 12d_1 + 2d_2 \quad \leq 0, \\ & \delta - 1 \quad \leq 0, \\ & -\delta \quad \leq 0, \end{aligned} \tag{3.145}$$

gegeben. Die Nominallösung für das unrelaxierte Probleme mit $\delta \equiv 0$ ist $d = \begin{pmatrix}15{,}5 & 89{,}5\end{pmatrix}$. Wird die Aufgabenstellung für verschiedene $\eta \in [0{,}1; 10^7]$ gelöst, kann anschaulich der Übergang der Lösung von $d^{[0]} \approx \begin{pmatrix}0 & 0\end{pmatrix}$ und $\delta^{[0]} \approx 1$ für sehr kleines $\eta^{[0]} \approx 0$ zu $d^{[1]} \approx \begin{pmatrix}15{,}5 & 89{,}5\end{pmatrix}$ mit $\delta^{[1]} \approx 0$ für großes $\eta^{[1]}$ beobachtet werden. Abbildung 3.4 zeigt den Verlauf von d und δ für steigende Bestrafungsparameter η. Wird die Lösung dieses quadratischen Problems als Suchrichtung im SQP-Algorithmus verwendet ist weiterhin von Interesse, dass die hier gefundenen Lösungen im Rahmen numerischer Ungenauigkeiten linear abhängig sind und lediglich betragsmäßig für steigende η zunehmen. Eine zu schwach bestrafte Relaxierung führt demnach auf zu kurze Schritte in die, im Sinne des nichtlinearen Problems, korrekte Richtung und bremst somit den übergeordneten Algorithmus.

Bemerkung 3.29. *Die Beobachtungen in Beispiel 3.28 treffen in der numerischen Praxis bei hochdimensionalen Problemen nur eingeschränkt zu. Insbesondere der stetige Übergang von $(d^{[0]}, \delta^{[0]}) = (0, 1)$ zu $(d^{[1]}, \delta^{[1]}) = (d^*, 0)$ wird durch diverse Effekte gestört, so dass $\delta^{[1]} \approx 0$ in den meisten Fällen mit vertretbarem numerischen Aufwand nicht zu erreichen ist. Die Relaxierung stellt aber einen wichtigen Baustein*

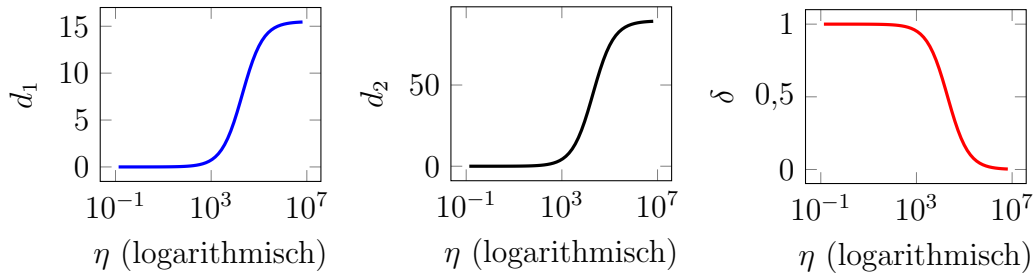


Abbildung 3.4: Variation des Bestrafungsparameters η für Beispiel 3.28. Die Abbildungen zeigen in blau den Verlauf von d_1, in schwarz d_2 und in rot d_3.

eines SQP-Algorithmus dar, um die Lösbarkeit der quadratischen Unterprobleme deutlich zu verbessern.

Es bleibt noch zu klären, wie die konkrete Einbindung des Verfahrens in den allgemeinen SQP-Algorithmus aussieht.
Die Relaxierung der Nebenbedingungen innerhalb des SQP-Verfahrens ermöglicht es in jeder Hauptiteration durch geeignete Wahl des Bestrafungsparameters η einen Kompromiss zwischen guter Lösbarkeit des Unterproblems und qualitativ hochwertiger Suchrichtung zu finden. Ähnlich wie bei der Wahl eines geeigneten Regularisierungsparameters stehen aber auch bei der Wahl des Parameters η vor der Lösung des Unterproblems keine Informationen zur Verfügung. Motiviert durch die Beobachtungen des Beispiels 3.28 wird das Unterproblem zunächst für ein kleines η zum Beispiel $\eta = 1$ gelöst. Anschließend wird die resultierende Relaxierungsvariable δ als Indikator für die Qualität der Suchrichtung verwendet. Ist $\delta \approx 1$, kann davon ausgegangen werden, dass die Suchrichtung klein ist oder unzureichend die linearisierten Nebenbedingungen erfüllt. Es wird eine Schranke $\delta_{\max}$ mit beispielsweise $\delta_{\max} = 0{,}92$ definiert. Nach Lösung des Unterproblems wird geprüft, ob $\delta < \delta_{\max}$ gegeben ist. Andernfalls wird η erhöht, zum Beispiel um einen konstanten Faktor $\rho \in \mathbb{R}$ und das Unterproblem wird mit diesem Parameter erneut gelöst, wobei die alte Lösung im Sinne der Warmstartstrategien von QPSOL als Startschätzung verwendet werden kann. In der Praxis hat sich etwa $\rho = 4{,}8$ bewährt. In Algorithmus 6 ist dieses Vorgehen schematisch dargestellt. Die Effizienz der Vorgehensweise des Algorithmus ist stark abhängig von dem Verstärkungsfaktor ρ. Dieser müsste eigentlich für jedes Problem adaptiv gewählt werden. Gleichzeitig ist es aber schwierig diesen ohne einige Testläufe des zu lösenden Problems geschickt zu wählen. Als Abhilfe für diese Problematik wird in Abschnitt 5.4 eine sensitivitätsbasierte Anpassung des Bestrafungswertes η vorgeschlagen.
Die Einführung einer einzelnen Relaxierungsvariablen kann in einigen Fällen nicht ausreichend sein und die Bestimmung einer guten Suchrichtung stören. Das folgende Beispiel verdeutlicht die Situation.

Algorithmus 6 Relaxierung der Nebenbedingungen innerhalb von WORHP

1: Wähle $\delta_{\max} = 0{,}92$, $\rho = 4{,}8$, $\eta_{\max} = 1 \cdot 10^7$
2: Setze $j = 0$, $\eta^{[0]} = 1{,}0$ und die Startschätzung $\delta^{[0]} = 0$,
3: **repeat**
4: **if** $j > 0$ **then**
5: $\eta^{[j+1]} \leftarrow \rho\eta^{[j]}$
6: **end if**
7: Löse das Unterproblem (3.141) mit Lösung $(d^{[j]}, \delta^{[j]})$
8: **until** (QP erfolgreich gelöst und $\delta^{[j]} < \delta_{\max}$) oder $\eta^{[j]} > \eta_{\max}$

Beispiel 3.30. *Es seien die Nebenbedingungen*

$$g_1(x) = 1 + (x_2 - 2)^2 + (x_3 - 3)^3 \leq 0 \tag{3.146}$$
$$g_2(x) = 3x_1^2 - 5x_3 - 2 \leq 0 \tag{3.147}$$
$$g_3(x) = x_1 + x_2 + x_3 \leq 0 \tag{3.148}$$

gegeben. Wird als Startwert $x = (1, 2, 3)$ gewählt, ist

$$\begin{pmatrix} 1 \\ -14 \\ 6 \end{pmatrix} + \begin{pmatrix} 0 & 0 & 0 \\ 6 & 0 & -5 \\ 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} d_1 \\ d_2 \\ d_3 \end{pmatrix} \leq \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

die Linearisierung der Nebenbedingungen. Durch Relaxierung werden die Nebenbedingungen zu

$$\begin{pmatrix} 1 \\ -14 \\ 6 \\ -1 \\ 0 \end{pmatrix} + \begin{pmatrix} 0 & 0 & 0 & -1 \\ 6 & 0 & -5 & 14 \\ 1 & 1 & 1 & -6 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & -1 \end{pmatrix} \begin{pmatrix} d_1 \\ d_2 \\ d_3 \\ \delta \end{pmatrix} \leq \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

und die erste Zeile erzwingt $\delta = 1$ und somit ist $d = (0, 0, 0)$ ein zulässiger Punkt.

Bemerkung 3.31. *Ob tatsächlich $d = (0, 0, 0)$ oder eventuell ein anderes d mit $\|d\|_2 > 0$ gefunden wird hängt von der Zielfunktion und dem verwendeten Algorithmus zur Lösung des quadratischen Optimierungsproblems ab.*

Die Beobachtungen in Beispiel 3.30 motivieren die Einführung mehrerer Relaxierungsvariablen. WORHP bietet die Möglichkeit jede Nebenbedingung einzeln für sich zu relaxieren. Es sei die Dimension des Relaxierungsvektors durch

$$N_\delta := N_g + N_h \tag{3.149}$$

gegeben. Die Summe der Aktivitätsindikatoren in Gleichung (3.149) entspricht der Anzahl der verletzten Ungleichungsnebenbedingungen. Weiterhin kann für jeden

Eintrag des Vektors ein eigener Bestrafungswert $\eta_i, \quad i = 1, ..., N_\delta$ verwendet werden. Die Matrix der Bestrafungsparameter sei als

$$\Omega := \begin{pmatrix} \eta_1 & & \\ & \ddots & \\ & & \eta_{N_\delta} \end{pmatrix} \tag{3.150}$$

bezeichnet.

Beispiel 3.32. *Die Nebenbedingungen in Beispiel 3.30 mit mehrfacher Relaxierung sind in linearisierter Form*

$$\begin{pmatrix} 1 \\ -14 \\ 6 \\ -1 \\ 0 \\ -1 \\ 0 \\ -1 \\ 0 \end{pmatrix} + \begin{pmatrix} 0 & 0 & 0 & -1 & 0 & 0 \\ 6 & 0 & -5 & 0 & 14 & 0 \\ 1 & 1 & 1 & 0 & 0 & -6 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & -1 \end{pmatrix} \begin{pmatrix} d_1 \\ d_2 \\ d_3 \\ \delta_1 \\ \delta_2 \\ \delta_3 \end{pmatrix} \leq \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}. \tag{3.151}$$

Die erste Zeile erzwingt erneut $\delta_1 = 1$. Die verbleibenden Gleichungen sind nicht eindeutig lösbar, aber eine zulässige Lösung ist beispielsweise $d = (0, \frac{16}{5}, \frac{14}{5})$ mit $\delta = (1, 0, 0)$. Demnach kann trotz der inkonsistenten linearisierten Nebenbedingung in der ersten Zeile ein zulässiger Punkt mit $\|d\|_2 > 0$ gefunden werden.

Bemerkung 3.33. *Das Beispiel 3.32 verdeutlicht, dass durch die Einführung einer Relaxierungsvariablen für jede Nebenbedingung die Lösbarkeit der linearisierten Nebenbedingungen bedeutend verbessert werden kann. Zusätzlich zeigt Gleichung* (3.151)*, dass durch die neuen Einträge in der erweiterten Jacobi-Matrix die lineare Unabhängigkeit der linearisierten Nebenbedingungen garantiert werden kann. Hierbei ist zu beachten, dass für jede Relaxierungsvariable zwei Nebenbedingungen auf Grund der Boxschranken neu eingeführt werden müssen, von diesen kann aber jeweils nur eine aktiv sein.*
Durch die Einführung der zusätzlichen Variablen erhöht sich die Dimension des quadratischen Unterproblems um N_δ. Außerdem müssen auch noch N_δ zusätzliche Einträge in der Hesse-Matrix der Lagrange-Funktion hinzugefügt werden. Diese enthalten die zweite Ableitung der Zielfunktion nach den Relaxierungsvariablen. Diese Ableitungswerte sind gerade die Bestrafungsparameter $\eta_i, \quad i = 1, ..., N_\delta$. Insgesamt wird das zu lösende KKT-System um N_δ Spalten und Zeilen vergrößert. Im schlimmsten Fall, dass alle Ungleichungsnebenbedingungen entsprechend der Fallunterscheidung (3.140) *relaxiert werden, wird das System um $N_g + N_h$ Zeilen und Spalten vergrößert. Dies kann den numerischen Aufwand zur Bestimmung einer Suchrichtung für das nichtlineare Problem erheblich erhöhen.*

Algorithmus 7 Relaxierung der Nebenbedingungen im Fall $N_\delta = N_g + N_h$ innerhalb von WORHP

Wähle $\delta_{\max} = 0{,}92$, $\rho = 4{,}8$, $\eta_{\max} = 1 \cdot 10^7$
Setze $j = 0$, $\eta^{[0]} = e$ und die Startschätzung $\delta^{[0]} = 0_{N_\delta \times 1}$,
repeat
  if $j > 0$ **then**
    for $k = 1, ..., N_\delta$ **do**
      if $\delta_k^{[j]} > \delta_{\max}$ **then**
        $\eta_k^{[j+1]} \leftarrow \rho \eta_k^{[j]}$
      end if
    end for
  end if
  Löse das Unterproblem (3.141) mit $N_\delta = N_g + N_h$ mit Lösung $(d^{[j]}, \delta^{[j]})$
until QP erfolgreich gelöst und $\delta_k^{[j]} < \delta_{\max}, \quad k = 1, ..., N_\delta$ oder $\exists k, \eta_k^{[j]} > \eta_{\max}$

Die Vorgehensweise zur Anpassung der Bestrafungsparameter während der Optimierung erfolgt ähnlich wie im Fall $N_\delta = 1$. Das quadratische Unterproblem wird gelöst und die resultierenden Relaxationsvariablen $\delta_i, \quad i = 1, ..., N_\delta$ werden analysiert. Für jedes $\delta_i > \delta_{\max}$ muss der Bestrafungsparameter η_i erhöht werden. Anschließend wird das angepasste quadratische Unterproblem erneut gelöst. Die Vorgehensweise ist im Algorithmus 7 dargestellt.

Bemerkung 3.34. *In der praktischen Optimierung hochdimensionaler Aufgabenstellungen ist die Verwendung von einer Relaxierungsvariablen für jede Nebenbedingung häufig durch einen starken Anstieg des Rechenaufwandes geprägt. Die Strategie kann eher als Möglichkeit angesehen werden bei problematischen Optimierungsverläufen zur Einschätzung der vorliegenden Situation zu dienen. Ein deutlich verbesserter Optimierungsverlauf bei der Verwendung von vielen Relaxierungsvariablen deutet in der Regel auf schwierige Nebenbedingungen hin, so dass der Benutzer des Optimierers möglicherweise durch Umformulierung der Nebenbedingungen die Situation verbessern kann.*

Die Relaxierung stellt eine interessante Möglichkeit dar, um schwierige Nebenbedingungen numerisch behandeln zu können. Im Folgenden wird ein neuartiger Ansatz zur adaptiven Wahl der Anzahl der Relaxierungsvariablen vorgeschlagen. Das Ziel dieses Algorithmus soll sein die Flexibilität vieler Relaxierungsvariablen mit dem geringen Aufwand bei Verwendung lediglich einer Relaxierungsvariablen zu verknüpfen.
Konkret bedeutet dies, dass die Anzahl N_δ während des Optimierungsverlaufes automatisiert angepasst werden soll. Es sei ein Zuordnungsvektor
$\omega \in \mathbb{N}^{N_g+N_h}, \omega_i \in \{1, ..., N_\delta\}$ definiert. Dieser Vektor ordnet jeder Nebenbedingung eine Relaxierungsvariable zu. Die zuvor erläuterte Variante mit einer Relaxierungs-

Algorithmus 8 Adaptive Anpassung der Anzahl der Relaxierungsvariablen in WORHP

Wähle $\delta_{\max} = 0{,}92$, $\delta_{\min} = 0{,}01$, $\rho = 4{,}8$, $\eta_{\max} = 1 \cdot 10^7$
Setze $j = 0$, $N_\delta = 1$, $\eta^{[0]} = e$ und die Startschätzung $\delta^{[0]} = 0_{N_\delta \times 1}$,
repeat
  if $j > 0$ **then**
    for $k = 1, ..., N_\delta$ **do**
      if $\delta_k^{[j]} < \delta_{\min}$ **then**
        $\omega_k \leftarrow 1$
      end if
      if $\delta_k^{[j]} > \delta_{\max}$ **then**
        $\eta_k^{[j+1]} \leftarrow \rho \eta_k^{[j]}$
        $N_\delta \leftarrow N_\delta + 1$
        $l \leftarrow \arg\min_{i=1,...,N_g+N_h} \zeta_i, \quad \zeta = (\lambda^{[k]}, \mu^{[k]})$
        $\omega_l \leftarrow N_\delta$
      end if
    end for
  end if
  Löse das Unterproblem (3.141) mit N_δ variable mit Lösung $(d^{[j]}, \delta^{[j]})$
  $j \leftarrow j + 1$
until QP erfolgreich gelöst und $\delta_k^{[j]} < \delta_{\max}, \;\; k = 1, ..., N_\delta, \; \exists k, \eta_k^{[j]} > \eta_{\max}$

variablen für alle Nebenbedingungen entspricht $\omega = (1, 1, ..., 1)$, das heißt jeder Nebenbedingung ist die erste Relaxierungsvariable zugeordnet. Die aufwendigere Strategie mit einer eigenen Relaxierungsvariablen für jede Nebenbedingung wird durch $\omega = (1, 2, 3, ..., N_g + N_h)$ repräsentiert.
Bei der adaptiven Relaxierung gibt es eine Relaxierungsvariable δ_1, welche dem Großteil der Nebenbedingungen zugeordnet ist. Gleichzeitig gibt es die Variablen $\delta_i, 1 < i \leq N_g + N_h$, welche einzelnen Nebenbedingungen zugeordnet werden können. Für die Adaption des Zuordnungsvektors sei eine untere Schranke für die Relaxierungsvariablen durch $\delta_{\min} = 0{,}01$ definiert.
Zum Start der Optimierung wird der Vektor $\omega = e$ initialisiert, so dass die Konfiguration zunächst der einfachen Relaxierung von WORHP entspricht.
Anschließend kann der Vektor ω auf zwei Arten verändert werden. Einerseits kann einer Nebenbedingung eine eigene Relaxierungsvariable zugeordnet werden, andererseits kann nachdem einer Nebenbedingung eine eigene Variable zugeordnet wurde, dieser wieder die allgemeine Relaxierungsvariable zugeordnet werden. Es wird zunächst versucht mit der aktuellen Konfiguration das quadratische Unterproblem zu lösen. Schlägt dies fehl, wird im Falle der adaptiven Relaxierung nicht nur der Bestrafungsparameter der Relaxierung wie in den Algorithmen 6 und 7 erhöht. Stattdessen wird zusätzlich geprüft, welche Relaxierungsvariable größer als die obere Schranke $\delta_{\max}$ war. Handelt es sich um die erste Variable, welche dem Großteil

der Nebenbedingungen zugeordnet ist, so werden zusätzlich in Anlehnung an die Aktive-Mengen-Strategie aus Algorithmus 1 die den Nebenbedingungen zugeordneten Lagrange-Multiplikatoren λ und μ verwendet, um zu entscheiden welche Nebenbedingung eine eigene Relaxierung bekommt. Um zu vermeiden, dass schnell viele Relaxierungsvariablen verwendet werden, wird schrittweise immer nur eine zusätzliche Nebenbedingung mit einer eigenen Variable versehen. Gleichzeitig wird außerdem geprüft ob der Wert einer Relaxierungsvariablen kleiner als die untere Schranke $\delta_{\min}$ ist. Es wird für $i = 1, ..., N_g + N_h$ die Bedingung $\delta_i \leq \delta_{\min}$ getestet. Diese Bedingung wird als *Reduktionskriterium* verwendet. Ist das Reduktionskriterium erfüllt, wird die entsprechende Nebenbedingung wieder mit der allgemeinen Relaxierungsvariablen verbunden. Nach der erfolgreichen Lösung des quadratischen Unterproblemes kann ebenfalls mit Hilfe des Reduktionskriteriums überprüft werden, ob der Zuordnungsvektor angepasst werden muss. Die Vorgehensweise ist in Algorithmus 8 kompakt zusammengefasst.

3.3.4 Quadratisches Optimierungsproblem in WORHP

Werden alle beschriebenen Maßnahmen zur Relaxierung und Regularisierung der quadratischen Unterprobleme innerhalb des SQP-Algorithmus in die Problemformulierung integriert, so ergibt sich ausgehend von der Standardformulierung 2.3 das erweiterte quadratische Unterproblem

$$\begin{aligned}
\min_{d\in\mathbb{R}^{N_x},\delta\in[0,1]^{N_\delta}} \quad & \tfrac{1}{2}d^\top Q d \; + \\
& \nabla_x f(x^{[k]})^\top d + \tfrac{1}{2}\delta^\top \Omega \delta \\
\\
\text{unter} \quad & g_i(x^{[k]})(1-\sigma_i\delta_i) + \nabla_x g_i(x^{[k]})^\top d && \leq 0, \quad i = 1, ..., N_g \\
& h_j(x^{[k]})(1-\delta_{N_g+j}) + \nabla_x h_j(x^{[k]})^\top d && = 0, \quad j = 1, ..., N_h \\
& d_l + x_l - c_l && \leq 0, \quad l = 1, ..., N_{\text{box}} \\
& \delta_l - 1 && \leq 0, \quad l = 1, ..., N_\delta \\
& -\delta_l && \leq 0, \quad l = 1, ..., N_\delta
\end{aligned} \tag{3.152}$$

zur Bestimmung der Suchrichtung innerhalb von WORHP. Es wurde zur Vereinfachung der Indizierung ohne Beschränkung der Allgemeinheit angenommen, dass Boxschranken nur an die ersten N_{box} Variablen vorliegen. Die Matrix Q kann auf verschiedene Arten zusammengestellt werden. Idealerweise liegen analytische Ableitungen vor, in diesem Fall ergibt sich mit einem passenden Regularisierungsterm

$$Q := \left(\nabla^2_{xx} L(x^{[k]}, \lambda^{[k]}, \mu^{[k]}) + \tau \left(\max(0, 1 - \sigma^{[k]}_{min}) \right) I_{N_x} \right). \tag{3.153}$$

Alternativ kann Q über ein BFGS-Verfahren oder über finite Differenzen, wie in Abschnitt 3.3.2 beschrieben, erstellt werden. Mit Hilfe der folgenden Definitionen lässt sich diese erweiterte Form wieder in die Standardform entsprechend (2.3) bringen.

Wird der kombinierte Vektor $\tilde{x} := (d, \delta)^\top$ verwendet, so ergibt sich als quadratische Komponente der Zielfunktion

$$\tilde{Q} := \begin{pmatrix} Q & 0 \\ 0 & \Omega \end{pmatrix}.$$

Die lineare Komponente der Zielfunktion

$$c := \begin{pmatrix} \nabla_x f(x^{[k]}) \\ 0_{N_\delta} \end{pmatrix}$$

muss mit Nullen entsprechend der Dimension des Relaxierungsvektors ergänzt werden. Weiterhin wurden zusätzliche Nebenbedingungen

$$\begin{aligned} \delta_l - 1 &\leq 0, \qquad l = 1, ..., N_\delta \\ -\delta_l &\leq 0, \qquad l = 1, ..., N_\delta \end{aligned}$$

ergänzt, um der Einschränkung $\delta \in [0, 1]^{N_\delta}$ zu genügen. Diese wurden in der Formulierung (3.152) als zusätzliche Boxschranken formuliert und können somit direkt im Algorithmus, der Vorgehensweise aus Abschnitt 3.1.2.2 folgend, behandelt werden. Um die Relaxierung der Nebenbedingungen in die Matrix der Ungleichungsnebenbedingungen zu integrieren, sei

$$C := \begin{pmatrix} \nabla_x g(x^{[k]}) & -\sigma g(x^{[k]}) \end{pmatrix}$$

definiert. Der Vektor σ wird entsprechend Gleichung (3.140) verwendet, um sicher zu stellen, dass nur in $x^{[k]}$ verletzte Nebenbedingungen relaxiert werden. Die Schranke für die Ungleichungsnebenbedingungen ist durch

$$\gamma := -g(x^{[k]})$$

gegeben. Analog wird die Matrix A

$$A := \begin{pmatrix} \nabla_x h(x^{[k]}) & h(x^{[k]}) \end{pmatrix} \tag{3.154}$$

zur Integration der Gleichungsnebenbedingungen mit Relaxierung gefüllt. Die Schranke β wird durch

$$\beta := -h(x^{[k]})$$

definiert. Mit Hilfe dieser Setzungen können die quadratischen Unterprobleme innerhalb von WORHP direkt an den integrierten QP-Löser QPSOL gegeben werden, welcher diese in der Standardform 2.3, ergänzt um Boxschranken an die Variablen, erwartet.

Kapitel 4

Parametrische Sensitivitätsanalyse

Inhaltsangabe

Die Optimierung einer konkreten Aufgabenstellung erfordert zunächst die mathematische Modellierung der vorliegenden Situation. Während dieses Vorganges ist es häufig nötig Modellparameter einzuführen, um zunächst unbekannte Aspekte des zugrundeliegenden Prozesses einzubinden. In diesem Kapitel werden die Auswirkungen von Änderungen allgemeiner Modellparameter auf die Lösung des nichtlinearen Optimierungsproblems diskutiert und theoretische Resultate dieser Fragestellung vorgestellt. Die beschriebenen Veränderungen der Modellparameter werden als parametrische Störungen bezeichnet.

4.1 Grundlagen der Sensitivitätsanalyse

Die folgenden Resultate helfen Störungen zu simulieren und anschließend die Auswirkungen auf die Lösung des Systems zu analysieren und quantifizieren.
In der Arbeit von Büskens [11] findet sich eine ausführliche Darstellung der historischen Entwicklung der Sensitivitätsanalyse. Die in diesem Kapitel vorgestellten Theorien wurden entscheidend von Fiacco geprägt und werden ausführlich in seinen Veröffentlichungen [28, 29, 30] und in Fiacco und Ghaemi [31] diskutiert.

Die folgende Definition erweitert das allgemeine nichtlineare Optimierungsproblem (2.1) um parametrische Störungen.

Definition 4.1. *Seien $x \in \mathbb{R}^{N_x}, p \in \mathbb{R}^{N_p}$ und seien Funktionen $f : \mathbb{R}^{N_x} \times \mathbb{R}^{N_p} \to \mathbb{R}$, $g : \mathbb{R}^{N_x} \times \mathbb{R}^{N_p} \to \mathbb{R}^{N_g}$ und $h : \mathbb{R}^{N_x} \times \mathbb{R}^{N_p} \to \mathbb{R}^{N_h}$ gegeben. Dann heißt*

$$\begin{aligned} \min_{x \in \mathbb{R}^{N_x}} \quad & f(x,p) \\ \textit{unter} \quad & g(x,p) \leq 0 \\ & h(x,p) = 0 \end{aligned} \tag{NLP(p)}$$

parametergestörtes Standardproblem der nichtlinearen Optimierung*. Mit einer Nominalstörung $p_0 \in \mathbb{R}^{N_p}$ wird das Problem NLP(p_0)* ungestörtes Problem *oder* Nominalproblem *genannt.*

Das Ziel der parametrischen Sensitivitätsanalyse ist es einen funktionalen Zusammenhang zwischen der optimalen Lösung (x^*, λ^*, μ^*) und dem Störungsparameter p zu finden. Bei näherer Betrachtung der KKT-Bedingungen aus Satz 2.21 wird klar, dass derartige Zusammenhänge implizit für diese Bedingungen für Aufgabenstellung NLP(p) existieren. Hierzu sei kurz an den Satz über implizte Funktionen erinnert.

Satz 4.2 (Satz über implizite Funktionen)**.** *Es sei die einmal stetig differenzierbare Abbildung $F : \mathbb{R}^{N_x} \times \mathbb{R}^{N_p} \to \mathbb{R}^{N_x}, (x,p) \mapsto F(x,p)$, gegeben. Ferner existiere $(x^*, p_0) \in \mathbb{R}^{N_x} \times \mathbb{R}^{N_p}$, so dass $F(x^*, p_0) = 0$ gelte und die Jacobi-Matrix $\nabla_x F(x^*, p_0) \in \mathbb{R}^{N_x} \times \mathbb{R}^{N_x}$ invertierbar sei.*
Dann existieren offene Umgebungen $U \subset \mathbb{R}^{N_x}$ von x^* und $P \subset \mathbb{R}^{N_x}$ von p_0, sowie eine stetig differenzierbare Funktion $x : P \to U$ mit $x(p_0) = x^*$ und $F(x(p), p) = 0$ für alle $p \in P$. Ist $(x,p) \in U \times P$ ein Punkt mit $F(x,p) = 0$, so folgt:

$$x = x(p).$$

Weiterhin gilt für die Jacobi-Matrix

$$\nabla_p x(p_0) = (\nabla_x F(x^*, p_0))^{-1} \nabla_p F(x^*, p_0).$$

Beweis. Der Beweis findet sich beispielsweise im Buch von Forster [39]. □

Durch Anwendung des Satzes über implizite Funktionen kann das zentrale Resultat dieses Abschnitts, der Sensitivitätssatz, formuliert und bewiesen werden. Dieser wurde von Fiacco [28] und Robinson [88] bewiesen. Der folgende Beweis orientiert sich an der Vorgehensweise von Büskens [11], ergänzt den dortigen Beweis aber um die explizite Behandlung der Ungleichungsnebenbedingungen mit Aspekten des Beweises aus dem Buch von Spellucci [97]. Zunächst wird ein kleiner Hilfssatz formuliert und anschließend folgt der Sensitivitätssatz.

Hilfssatz 4.3. *Es seien $f : \mathbb{R}^{N_x} \to \mathbb{R}, g : \mathbb{R}^{N_x} \to \mathbb{R}^{N_g}, h : \mathbb{R}^{N_x} \to \mathbb{R}^{N_h}$ zweimal stetig differenzierbare Funktionen in einer Umgebung des lokalen Minimums x^* der nichtlinearen Optimierungsaufgabe*

$$\begin{aligned} \min_{x \in \mathbb{R}^{N_x}} \quad & f(x) \\ \textit{unter} \quad & g(x) \leq 0 \\ & h(x) = 0. \end{aligned} \tag{4.1}$$

Das Minimum x^* erfülle die notwendigen Optimalitätsbedingungen aus Satz 2.21 und die hinreichenden Bedingungen aus Satz 2.23 mit den Lagrange-Multiplikatoren $\lambda^* \in \mathbb{R}^{N_g}$ und $\mu^* \in \mathbb{R}^{N_h}$. Weiterhin sei $L(x, \lambda, \mu)$ die Lagrange-Funktion zur Problemstellung (4.1). Zuletzt seien die Matrizen $\Lambda^* \in \mathbb{R}^{N_g \times N_g}$ und $\Gamma^* \in \mathbb{R}^{N_g \times N_g}$ als

$$\Lambda^* := \begin{pmatrix} \lambda_1^* & & \\ & \ddots & \\ & & \lambda_{N_g}^* \end{pmatrix} \quad \textit{und} \quad \Gamma^* := \begin{pmatrix} g_1(x^*) & & \\ & \ddots & \\ & & g_{N_g}(x^*) \end{pmatrix} \tag{4.2}$$

definiert. Dann ist die Matrix $A \in \mathbb{R}^{(N_x+N_g+N_h)\times(N_x+N_g+N_h)}$

$$\mathcal{M} := \begin{pmatrix} \nabla_{xx}^2 L(x^*, \lambda^*, \mu^*) & \nabla_x g(x^*) & \nabla_x h(x^*) \\ \Lambda^* \nabla_x g(x^*)^\top & \Gamma^* & 0 \\ \nabla_x h(x^*)^\top & 0 & 0 \end{pmatrix} \tag{4.3}$$

invertierbar.

Beweis. Sei $k \geq 0$ die Anzahl der inaktiven Nebenbedingungen. Mittels Umordnung sei ohne Beschränkung der Allgemeinheit $\mathcal{A}(x^*) = \{k+1, ..., N_g\}$. Auf Grund der strikten Komplementarität sind

$$\Lambda^* = \begin{pmatrix} 0 & 0 \\ 0 & \Lambda_2^* \end{pmatrix}, \quad \Lambda_2^* = \begin{pmatrix} \lambda_{k+1}^* & & \\ & \ddots & \\ & & \lambda_{N_g}^* \end{pmatrix} \text{ invertierbar und} \tag{4.4}$$

$$\Gamma^* = \begin{pmatrix} \Gamma_1^* & 0 \\ 0 & 0 \end{pmatrix}, \quad \Gamma_1^* = \begin{pmatrix} g_1(x^*) & & \\ & \ddots & \\ & & g_k(x^*) \end{pmatrix} \text{ invertierbar.} \tag{4.5}$$

Zur Vereinfachung der folgenden Betrachtungen seien

$$\mathcal{B} := \nabla_{xx}^2 L(x^*, \lambda^*, \mu^*) \tag{4.6}$$

$$\mathcal{C} := (\nabla_x g_1, ..., \nabla_x g_k)(x^*) \tag{4.7}$$

$$\mathcal{D} := \left(\nabla_x g_{k+1}, ..., \nabla_x g_{N_g}, \nabla_x h\right)(x^*) \tag{4.8}$$

definiert. $\mathcal{C}$ enthält die Gradienten der inaktiven Ungleichungsnebenbedingungen, wohingegen $\mathcal{D}$ die übrigen Gradienten enthält. Durch Division der zu den aktiven

Nebenbedingungen korrespondierenden Zeilen mit den dazugehörigen Multiplikatoren ergibt sich

$$\begin{pmatrix} I_{N_x+k} & 0 & 0 \\ 0 & (\Lambda_2^*)^{-1} & 0 \\ 0 & 0 & I_{N_h} \end{pmatrix} \mathcal{M} = \begin{pmatrix} \mathcal{B} & \mathcal{C} & \mathcal{D} \\ 0 & \Gamma_1^* & 0 \\ \mathcal{D}^\top & 0 & 0 \end{pmatrix}. \tag{4.9}$$

Sei $v \in \mathbb{R}^{N_x+N_g+N_h}$ passend partitioniert mit $v_1 \in \mathbb{R}^{N_x}$, $v_2 \in \mathbb{R}^{N_g-k}$ und $v_3 \in \mathbb{R}^{k+Nh}$. Um zu zeigen, dass $\mathcal{M}$ invertierbar ist, genügt es zu zeigen, dass aus $\mathcal{M}v = 0$ folgt, dass $v = 0$ ist. Mit Gleichung (4.9) folgt aus $\mathcal{M}v = 0$, dass die Beziehungen

$$\mathcal{B}v_1 + \mathcal{C}v_2 + \mathcal{D}v_3 = 0 \tag{4.10}$$
$$\Gamma_1^* v_2 = 0 \tag{4.11}$$
$$\mathcal{D}^\top v_1 = 0 \tag{4.12}$$

gelten. Aus Gleichung (4.11) folgt auf Grund der strikten Komplementarität direkt $v_2 = 0$. Die Multiplikation von Gleichung (4.10) mit $v_1^\top$ von links liefert somit

$$0 = v_1^\top \mathcal{B}v_1 + v_1^\top \mathcal{D}v_3 = v_1^\top \mathcal{B}v_1 + (v_1\mathcal{D}^\top)^\top v_3 = v_1^\top \mathcal{B}v_1, \tag{4.13}$$

woraus auf Grund der Normalität von x^* folgt, dass $v_1 = 0$ ist. Jetzt zeigt Gleichung (4.10), dass

$$\mathcal{D}v_3 = 0 \tag{4.14}$$

gilt. Hieraus folgt auf Grund der Spaltenregularität von $\mathcal{D}$ auch $v_3 = 0$. Insgesamt ergibt sich $v = 0$ und auf Grund der getroffenen Annahme folgt damit die Invertierbarkeit der Matrix $\mathcal{M}$. □

Satz 4.4 (Sensitivitätssatz)**.** *Für die Problemstellung* (4.1) *seien folgende Voraussetzungen erfüllt:*

1. *Die Funktionen f, g und h seien zweimal stetig bezüglich x differenzierbar. Weiterhin seien die Nebenbedingungen g und h, sowie die Ableitungen $\nabla_x f, \nabla_x g$ und $\nabla_x h$ in einer Umgebung um die optimale Lösung (x^*, p_0) stetig bezüglich p differenzierbar.*

2. *Es sei die Nominalstörung $p_0 \in \mathbb{R}^{N_p}$ gegeben. Die Nominallösung $x^* \in \mathbb{R}^{N_x}$ erfülle die notwendigen Optimalitätsbedingungen aus Satz 2.21. Weiterhin soll für x^* strikte Komplementarität nach Definition 2.22 gelten. Die Lagrange-Multiplikatoren λ^* und μ^* seien so bestimmt, dass das Tripel (x^*, λ^*, μ^*) die hinreichenden Bedingungen zweiter Ordnung von NLP(p_0) nach Satz 2.23 erfüllt.*

Dann gelten die folgenden Aussagen:

i. Es gibt Radien $\delta > 0$, $\epsilon > 0$ und Umgebungen $U_\epsilon(p_0) \subset \mathbb{R}^{N_p}$, $U_\delta(\zeta^*) := U_\delta(x^*, \lambda^*, \mu^*) \subset \mathbb{R}^{N_x+N_g+N_h}$, so dass für jedes $p \in U_\epsilon(p_0)$ NLP(p) eine eindeutige normale lokale Minimalstelle

$$\zeta(p) := (x(p), \lambda(p), \mu(p)) \in U_\delta(x^*, \lambda^*, \mu^*), \tag{4.15}$$

hat.

ii. Die Funktionen $x(p), \lambda(p)$ und $\mu(p)$ sind stetig differenzierbar bezüglich $p \in U_\epsilon(p_0)$ und es sind

$$\begin{aligned} x(p_0) &= x^*, \\ \lambda(p_0) &= \lambda^*, \\ \mu(p_0) &= \mu^*. \end{aligned}$$

Weiterhin erfüllen $x(p), \lambda(p), \mu(p)$ die notwendigen Optimalitätsbedingungen nach Satz 2.21, strikte Komplementarität und die hinreichenden Bedingungen zweiter Ordnung für NLP(p). Außerdem ändert sich die aktive Menge nicht. Es gilt

$$\mathcal{A}(x^*, p_0) = \mathcal{A}(x(p), p). \tag{4.16}$$

Beweis. Sei zunächst die Hilfsfunktion

$$F(x, \lambda, \mu, p) = \begin{pmatrix} \nabla_x L(x, \lambda, \mu, p) \\ \Lambda g(x, p) \\ h(h, p) \end{pmatrix} = 0 \tag{4.17}$$

gewählt. Nach Voraussetzung ist mit Definition von $\zeta^* := (x^*, \lambda^*, \mu^*)$ die Aussage $F(\zeta^*, p_0) = 0$ gültig. Zur Anwendung des Satzes über implizite Funktionen 4.2 ist noch zu zeigen, dass die Jacobi-Matrix

$$\nabla_z F(\zeta^*, p_0) = \begin{pmatrix} \nabla^2_{xx} L(x^*, \lambda^*, \mu^*) & \nabla_x g(x^*) & \nabla_x h(x^*) \\ \Lambda^* \nabla_x g(x^*)^\top & \Gamma^* & 0 \\ \nabla_x h(x^*)^\top & 0 & 0 \end{pmatrix} \tag{4.18}$$

invertierbar ist. Unter den gegebenen Voraussetzungen ist der Hilfssatz 4.3 anwendbar. Somit ist die Regularität der Matrix $\nabla_\zeta F(\zeta^*, p_0)$ gezeigt. Der Satz über implizite Funktionen ist anwendbar und damit existiert für $p \in U_\epsilon(p_0)$ genau ein $\zeta(p) \in U_\delta(\zeta^*)$, so dass $F(\zeta(p), p) = 0$ gilt. Demnach erfüllt $\zeta(p)$ für $p \in U_\epsilon(p_0)$ die KKT-Bedingungen von 4.1. Weiterhin folgt aus dem Satz die Differenzierbarkeit von $\zeta(p)$ bezüglich p und der Zusammenhang

$$\frac{\mathrm{d}}{\mathrm{d}p}\zeta(p) = \begin{pmatrix} \frac{\mathrm{d}x}{\mathrm{d}p} \\ \frac{\mathrm{d}\lambda}{\mathrm{d}p} \\ \frac{\mathrm{d}\mu}{\mathrm{d}p} \end{pmatrix} = -\nabla_\zeta F(\zeta^*, p_0)^{-1} \nabla_p F(\zeta^*, p_0) \tag{4.19}$$

gilt.
Es bleibt zu zeigen, dass $x(p)$ eine normale lokale Minimalstelle von 4.1 ist. Es sei ohne Beschränkung der Allgemeinheit eine Sortierung der Nebenbedingungen wie im Hilfssatz 4.3 gegeben. Es sind $\lambda^*_{k+1}, ..., \lambda^*_{N_g} > 0$ und $g^*_1, ..., g^*_k > 0$. Auf Grund der Stetigkeit bezüglich p von g und λ ergibt sich, dass für hinreichend kleine $p \in U_\epsilon(p_0)$ auch $\lambda_{k+1}(p), ..., \lambda_{N_g}(p) > 0$ und $g_1(x(p), p), ..., g_k(x(p), p) > 0$ gilt. Zusammen mit der Gültigkeit von $F(\zeta(p), p) = 0$ folgt somit

$$g_{k+1}(x(p), p) = ... = g_{N_g}(x(p), p) = 0 \tag{4.20}$$

und

$$\lambda_1(p) = ... = \lambda_k(p) = 0, \tag{4.21}$$

woraus sich die strikte Komplementarität ergibt. Somit ist auch gezeigt, dass

$$\mathcal{A}(x^*, p_0) = \mathcal{A}(x(p), p). \tag{4.22}$$

gilt. Die Folgerungen des Bestandes der Regularitätsbedingung der linearen Unabhängigkeit und der hinreichenden Bedingungen zweiter Ordnung folgen unter der Annahme eines hinreichend kleinen p aus der stetigen Abhängigkeit bezüglich p der beteiligten Ableitungen. □

Mit Hilfe der Ergebnisse des Sensitivitätssatzes ist es möglich nach der Lösung eines gut gestellten Optimierungsproblems weitere Informationen über die Aufgabenstellung zu gewinnen. Der Satz zeigt die Abhängigkeit der optimalen Lösung bezüglich beliebiger Parameter in der Aufgabenstellung. Insbesondere kann durch geeignete Wahl von Parametern eine Störungsanalyse durchgeführt werden. Ein wichtiges Werkzeug stellen hierbei die parametrischen Sensitivitätsableitungen dar.
Gleichung 4.19 liefert den entscheidenden Zusammenhang zur Berechnung dieser Ableitungen:

$$\begin{pmatrix} \frac{\mathrm{d}x}{\mathrm{d}p}(p_0) \\ \frac{\mathrm{d}\lambda}{\mathrm{d}p}(p_0) \\ \frac{\mathrm{d}\mu}{\mathrm{d}p}(p_0) \end{pmatrix} = - \begin{pmatrix} \nabla^2_{xx} L(\zeta^*, p_0) & \nabla_x g(x^*, p_0)^\top & \nabla_x h(x^*, p_0)^\top \\ \Lambda^* \nabla_x g(x^*, p_0) & \Gamma^* & 0 \\ \nabla_x h(x^*, p_0) & 0 & 0 \end{pmatrix}^{-1} \begin{pmatrix} \nabla^2_{xp} L(\zeta^*, p_0) \\ \Lambda^* \nabla_p g(x^*, p_0) \\ \nabla_p h(x^*, p_0) \end{pmatrix} \tag{4.23}$$

Diese Ableitungen ermöglichen nach der Lösung des Nominalsproblems NLP(p) näherungsweise Lösungen für abweichende Parameter $p \in \mathbb{R}^{N_p}$ zu berechnen. Der Satz von Taylor liefert die linearen Approximationen

$$x(p) \approx \tilde{x}(p) = x(p_0) + \frac{\mathrm{d}x}{\mathrm{d}p}(p_0)(p - p_0) \tag{4.24}$$

$$\mu(p) \approx \tilde{\mu}(p) = \mu(p_0) + \frac{\mathrm{d}\mu}{\mathrm{d}p}(p_0)(p - p_0) \tag{4.25}$$

$$\lambda(p) \approx \tilde{\lambda}(p) = \lambda(p_0) + \frac{\mathrm{d}\lambda}{\mathrm{d}p}(p_0)(p - p_0) \tag{4.26}$$

für die primalen und dualen Variablen. Neben einer Analyse der Variablen ist auch von Interesse die Auswirkung von Störungen auf die Zielfunktion und Nebenbedingungen zu quantifizieren. Hierzu sei kurz an die Definition von Zugehörigkeitsfunktionen aus der Arbeit von Büskens [11] erinnert.

Definition 4.5. *Sei $\tilde{k} : \mathbb{R}^{N_x} \times \mathbb{R}^{N_g} \times \mathbb{R}^{N_h} \times \mathbb{R}^{N_p} \to \mathbb{R}$ eine einmal stetig differenzierbare Abbildung. Es sei eine Nominalstörung $p_0 \in \mathbb{R}^{N_p}$ gegeben. Weiterhin seien die Voraussetzungen von Satz 4.4 in einer Umgebung $P \subset \mathbb{R}^{N_p}$ um p_0 für die stetig differenzierbaren Funktionen $x : P \to \mathbb{R}^{N_x}$, $\lambda : P \to \mathbb{R}^{N_g}$ und $\mu : P \to \mathbb{R}^{N_h}$ erfüllt. Dann wird $k(p) := \tilde{k}(x(p), \lambda(p), \mu(p), p)$ eine* Zugehörigkeitsfunktion *der Problemstellung* (NLP(p)) *genannt.*

Bemerkung 4.6. *Es seien die Funktionen f, g und h einmal stetig differenzierbar bezüglich p. Dann handelt es sich bei diesen um Zugehörigkeitsfunktionen. Weiterhin ist in diesem Fall auch die Lagrange-Funktion eine Zugehörigkeitsfunktion.*

Durch Anwendung des Sensitivitätssatzes kann die Auswirkung von Störungen auf die Zugehörigkeitsfunktionen berechnet werden. In den folgenden Betrachtungen werden die Argumente weggelassen, wenn sie sich aus dem Kontext ergeben.

Satz 4.7. *Es sei eine Zugehörigkeitsfunktion $k : P \to \mathbb{R}$ gegeben. Die Voraussetzungen des Sensitivitätssatzes 4.4 seien erfüllt. Dann kann die Sensitivitätsableitung der Zugehörigkeitsfunktion über die Gleichung*

$$\frac{\mathrm{d}k}{\mathrm{d}p}(p_0) = -\left(\nabla_x k(p_0), \nabla_\lambda k(p_0), \nabla_\mu k(p_0)\right) \begin{pmatrix} \nabla_{xx}^2 L & \nabla_x g^\top & \nabla_x h^\top \\ \Lambda^* \nabla_x g & \Gamma^* & 0 \\ \nabla_x h & 0 & 0 \end{pmatrix}^{-1} \begin{pmatrix} \nabla_{xp}^2 L \\ \Lambda^* \nabla_p g \\ \nabla_p h \end{pmatrix} + \nabla_p k(p_0) \tag{4.27}$$

bestimmt werden.

Beweis. Die Anwendung der Kettenregel, in Kombination mit dem Sensitivitätssatz 4.4, ergibt die Behauptung. □

Im Folgenden werden kurz einige Ergebnisse für die Sensitivitäten der Zielfunktion, Nebenbedingungen und der Lagrange-Funktion aus der Arbeit von Büskens [11] vorgestellt. Diese werden im weiteren Verlauf erneut aufgegriffen und insbesondere im nächsten Abschnitt für spezielle Störungen analysiert.

Folgerung 4.8 (Sensitivität der Nebenbedingungen)**.** *Es seien die Voraussetzungen von Satz 4.4 erfüllt. Dann gilt für alle $p \in P$ die Gleichung*

$$\frac{\mathrm{d}h_j}{\mathrm{d}p}(p) = 0, \qquad j = 1, ..., N_h, \tag{4.28}$$

weil sich die aktive Menge im Gültigkeitsbereich des Sensitivitätssatzes nicht ändern darf. Für die Ungleichungsnebenbedingungen ergibt sich durch Anwendung von Gleichung (4.27)

$$\begin{aligned}\frac{\mathrm{d}g_i}{\mathrm{d}p}(p_0) &= -\left(\nabla_x g_i, 0, 0\right)\begin{pmatrix}\nabla_{xx}^2 L & \nabla_x g^\top & \nabla_x h^\top \\ \Lambda^*\nabla_x g & \Gamma^* & 0 \\ \nabla_x h & 0 & 0\end{pmatrix}^{-1}\begin{pmatrix}\nabla_{xp}^2 L \\ \Lambda^*\nabla_p g \\ \nabla_p h\end{pmatrix} + \nabla_p g_i \\ &= \nabla_x g_i \frac{\mathrm{d}x}{\mathrm{d}p} + \nabla_p g_i, \qquad i = 1, ..., N_g \end{aligned} \tag{4.29}$$

und

$$\frac{\mathrm{d}g_i}{\mathrm{d}p}(p) = 0, \qquad \forall i \in \mathcal{A}(x^*, p_0) \tag{4.30}$$

mit der Einschränkung auf die aktive Menge, welche unverändert bleibt.

Bemerkung 4.9. *Auf Grund von Gleichung* (4.28) *werden bei Betrachtung der speziell strukturierten Störungen im folgenden Abschnitt die Gleichungsnebenbedingungen nur im Zusammenhang mit den Sensitivitäten der Lagrange-Funktion erwähnt.*

Folgerung 4.10 (Sensitivität der Lagrange-Funktion)**.** *Es seien erneut die Voraussetzungen von Satz 4.4 erfüllt. Die Lagrange-Funktion*

$$L(x, \lambda, \mu, p) = f(x, p) + \lambda^\top g(x, p) + \mu^\top h(x, p)$$

ist ebenfalls eine Zugehörigkeitsfunktion. Für ihre Sensitivität gilt:

$$\frac{\mathrm{d}L}{\mathrm{d}p}(p_0) = \nabla_p L \tag{4.31}$$

Beweis. Es gelten $\nabla_x L(x^*, \lambda^*, \mu^*, p_0) = 0$, $\nabla_\lambda L(x^*, \lambda^*, \mu^*, p_0) = g(x^*, p_0)$ und $\nabla_\mu L(x^*, \lambda^*, \mu^*, p_0) = h(x^*, p_0) = 0$. Um die Gleichung (4.31) zu erhalten, müssen diese Beziehungen in Gleichung (4.27) eingesetzt werden. Es ergibt sich

$$\begin{aligned}\frac{\mathrm{d}L}{\mathrm{d}p}(p_0) &= -\left(\nabla_x L, \nabla_\lambda L, \nabla_\mu L\right)\begin{pmatrix}\nabla_{xx}^2 L & \nabla_x g^\top & \nabla_x h^\top \\ \Lambda^*\nabla_x g & \Gamma^* & 0 \\ \nabla_x h & 0 & 0\end{pmatrix}^{-1}\begin{pmatrix}\nabla_{xp}^2 L \\ \Lambda^*\nabla_p g \\ \nabla_p h\end{pmatrix} + \nabla_p L \\ &= \nabla_\lambda L \frac{\mathrm{d}\lambda}{\mathrm{d}p} + \nabla_p L \\ &= g\frac{\mathrm{d}\lambda}{\mathrm{d}p} + \nabla_p L \\ &= \nabla_p L, \end{aligned}$$

wobei die letzte Gleichheit gilt, weil entweder $g_i = 0$ für $i \in \mathcal{A}(x^*, p_0)$, oder $\lambda_i = 0$ für $i \notin \mathcal{A}(x^*, p_0)$ und somit die Sensitivität auf Grund von Satz 4.4 $\frac{\mathrm{d}\lambda_i}{\mathrm{d}p} = 0$ erfüllt, weil die aktive Menge unverändert bleibt. □

Folgerung 4.11 (Sensitivität erster Ordnung der Zielfunktion)**.** *Es seien die Voraussetzungen des Sensitivitätssatzes 4.4 erfüllt. Dann können die Formeln*

$$\begin{aligned}\frac{\mathrm{d}f}{\mathrm{d}p}(p_0) &= -\left(\nabla_x f, 0, 0\right)\begin{pmatrix}\nabla_{xx}^2 L & \nabla_x g^\top & \nabla_x h^\top \\ \Lambda^*\nabla_x g & \Gamma^* & 0 \\ \nabla_x h & 0 & 0\end{pmatrix}^{-1}\begin{pmatrix}\nabla_{xp}^2 L \\ \Lambda^*\nabla_p g \\ \nabla_p h\end{pmatrix} + \nabla_p f \\ &= \nabla_x f\frac{\mathrm{d}x}{\mathrm{d}p} + \nabla_p f, \qquad (4.32)\\ \frac{\mathrm{d}f}{\mathrm{d}p}(p_0) &= \frac{\mathrm{d}L}{\mathrm{d}p}(p_0) = \nabla_p L = \nabla_p f + (\lambda^*)^\top\nabla_p g + (\mu^*)^\top\nabla_p h \qquad (4.33)\end{aligned}$$

aus Gleichung (4.27) *hergeleitet werden.*

Beweis. Gleichung (4.32) ergibt sich direkt aus Gleichung (4.27). Um die Beziehung in Gleichung (4.33) zu erhalten muss der Beweis aus der Arbeit von Büskens [11] auch hier mit den Aspekten aus dem Buch von Spellucci [97] kombiniert werden, um die aktive Menge korrekt zu berücksichtigen:

$$\begin{aligned}\frac{\mathrm{d}f}{\mathrm{d}p}(p_0) &= -\left(\nabla_x f, 0, 0\right)\begin{pmatrix}\nabla_{xx}^2 L & \nabla_x g^\top & \nabla_x h^\top \\ \Lambda^*\nabla_x g & \Gamma^* & 0 \\ \nabla_x h & 0 & 0\end{pmatrix}^{-1}\begin{pmatrix}\nabla_{xp}^2 L \\ \Lambda^*\nabla_p g \\ \nabla_p h\end{pmatrix} + \nabla_p f \\ &= \left(-(\lambda^*)^\top\nabla_x g - (\mu^*)^\top\nabla_x h, 0, 0\right)\begin{pmatrix}\frac{\mathrm{d}x}{\mathrm{d}p} \\ \frac{\mathrm{d}\lambda}{\mathrm{d}p} \\ \frac{\mathrm{d}\mu}{\mathrm{d}p}\end{pmatrix} + \nabla_p f \\ &= (\lambda^*)^\top\nabla_p g + (\mu^*)^\top\nabla_p h + \nabla_p f = \nabla_p L\end{aligned}$$

Um die letzte Gleichung zu erhalten, muss die Komplementaritätsbedingung ausgenutzt werden. Diese impliziert, dass $\lambda_i^* = 0$, für $i \notin \mathcal{A}(x^*, p_0)$. Werden dann noch die Gleichungen (4.29) und (4.30) eingesetzt, ergibt sich die Behauptung. □

Die Folgerung 4.11 offenbart einen Zusammenhang zwischen dem lokalen Verhalten der Zielfunktion bezüglich p und der Lagrange-Funktion. Büskens [11] merkt an, dass die Formeln (4.31) und (4.33) unabhängig von der invertierten KKT-Matrix sind. Deshalb können diese Sensitivitäten effizient berechnet werden, wenn die Gradienten $\nabla_p f$ und $\nabla_p g$ einfach zu bestimmen sind.

Folgerung 4.12 (Sensitivität zweiter Ordnung der Zielfunktion)**.** *Es seien die Voraussetzungen des Satzes 4.4 erfüllt. Die Formeln*

$$\frac{\mathrm{d}^2 f}{\mathrm{d}p^2}(p_0) = \left(\frac{\mathrm{d}x}{\mathrm{d}p}(p_0)\right)^\top\nabla_{xp}^2 L + \left(\frac{\mathrm{d}\lambda}{\mathrm{d}p}(p_0)\right)^\top\nabla_p g + \left(\frac{\mathrm{d}\mu}{\mathrm{d}p}(p_0)\right)^\top\nabla_p h + \nabla_{pp}^2 L \qquad (4.34)$$

$$\frac{\mathrm{d}^2 f}{\mathrm{d}p^2}(p_0) = \left(\frac{\mathrm{d}x}{\mathrm{d}p}(p_0)\right)^\top\nabla_{xx}^2 L\frac{\mathrm{d}x}{\mathrm{d}p}(p_0) + 2\left(\nabla_{px}^2 L\frac{\mathrm{d}x}{\mathrm{d}p}(p_0)\right)^\top + \nabla_{pp}^2 L \qquad (4.35)$$

beschreiben die Sensitivitäten zweiter Ordnung der Zielfunktion.

Beweis. Büskens [11] beweist die Aussage ohne Ungleichungsnebenbedingungen. Unter Berücksichtigung der Details des Beweises von Folgerung 4.8 ergeben sich die Formeln (4.34) und (4.35) analog. □

4.2 Speziell strukturierte Störungen

In einer numerischen Umsetzung wird die inverse Matrix in der Berechnungsvorschrift (4.23) nicht explizit berechnet, sondern das Gleichungssystem durch eine geeignete Zerlegung gelöst. In Kapitel 3 wurde gezeigt, dass zur Lösung des Nominalproblems bereits eine passende Zerlegung bestimmt wurde und wiederverwendet werden kann. Streng genommen ist zu beachten, dass die Zerlegung der Matrix nicht in der optimalen Lösung des quadratischen Unterproblems ermittelt wurde, sondern in den vorletzten Iterierten. Der hieraus resultierende Fehler ist für die im Folgenden vorgestellten numerischen Verfahren aber vernachlässigbar. Zur effizienten Berechnung der Sensitivitätsableitungen ist demnach nur noch eine numerisch einfache Berechnung der Ableitungen nach p zur Bestimmung der rechten Seite in Gleichung (4.23) notwendig. Büskens zeigt in seinen Arbeiten [11, 12], dass für lineare Störungen der Nebenbedingungen sowie natürliche Störungen der Zielfunktion die Berechnung nicht notwendig ist, da diese Ableitungen bereits a priori bekannt sind. In der Arbeit von Nikolayzik [81] wird weiterhin eine quadratische Störung der Zielfunktion genutzt, die auch im weiteren Verlauf dieser Arbeit erneut aufgegriffen wird. Im Folgenden wird gezeigt, dass sich diese Ideen erweitern lassen, so dass es möglich ist bei der Postoptimalitätsanalyse von quadratischen Optimierungsproblemen der Form 2.3 Störungen bezüglich sämtlicher auftretender Größen zu behandeln. Trotz der Absicht alle relevanten Größen innerhalb quadratischer Optimierungsprobleme zu analysieren, werden die folgenden Diskussionen für das allgemeinere nichtlineare Optimierungsproblem mit parametrischer Störung nach Gleichung NLP(p) durchgeführt.
Es sei $[p_0] := (x^*, \lambda^*, \mu^*, p_0)$ zur Vereinfachung der folgenden Gleichungen definiert und die Nominalstörung p_0 enthalte alle Störungsarten passend zum Kontext.

4.2.1 Natürliche Störung der Zielfunktion

Die Störung der Zielfunktion $r \in \mathbb{R}^{N_x}$ führt auf die Problemstellung

$$\begin{array}{lll} \min\limits_{x \in \mathbb{R}^{N_x}} & f(x) + r^\top x & \\ \text{unter} & g_i(x) \leq 0, & i = 1, ..., N_g, \\ & h_j(x) = 0, & j = 1, ..., N_h. \end{array} \tag{4.36}$$

Die Gradienten der Zielfunktion und Lagrange-Funktion der Aufgabenstellung sind

$$\nabla_x (f(x) + r^\top x) = \nabla_x f(x) + r,$$
$$\nabla_x (L(x) + r^\top x) = \nabla_x f(x) + \lambda^\top \nabla_x g(x) + \mu^\top \nabla_x h(x) + r.$$

Es wird deutlich, dass die Störung r in der Zielfunktion als Störung des Optimalitätsmaßes 2.32 der KKT-Bedingungen interpretiert werden kann. Als weitere Verallgemeinerung könnten sogar Störungen der Zielfunktion in der Form $r^\top Mx$ mit einer beliebigen Matrix $M \in \mathbb{R}^{N_x \times N_x}$ betrachtet werden. In diesem Zusammenhang könnte die Matrix M beispielsweise eine Gewichtung repräsentieren.
Es sei die Einheitsmatrix $I_k \in \mathbb{R}^{k \times k}$ definiert. Die rechte Seite für Gleichung (4.23) zur Berechnung der Sensitivitätsableitungen der primalen und dualen Variablen vereinfacht sich in dieser Situation zu

$$\begin{pmatrix} \nabla^2_{xp} L[p_0] \\ \Lambda \nabla_p g[p_0] \\ \nabla_p h[p_0] \end{pmatrix} = \begin{pmatrix} I_{N_x} \\ 0 \\ 0 \end{pmatrix}. \tag{4.37}$$

Die potentiell aufwendig zu berechnende Ableitung $\nabla^2_{xp} L[p_0]$ ist somit direkt bekannt. Auch die Sensitivitäten der zuvor diskutierten Zugehörigkeitsfunktionen vereinfachen sich für diese speziell strukturierte Störung. Gleichung (4.29) ergibt

$$\frac{\mathrm{d}g_i}{\mathrm{d}p}(p_0) = \nabla_x g_i \frac{\mathrm{d}x}{\mathrm{d}p}, \quad i = 1, ..., N_g, \tag{4.38}$$

welche nach Berechnung der Sensitivitätsableitungen der Variablen direkt bekannt sind. Weiterhin kann auch die Sensitivitätsableitung der Lagrange- und Zielfunktion über die Gleichungen (4.31) und (4.33) betrachtet werden. Es ergibt sich das auf Grund der speziellen Struktur zu erwartende Ergebnis

$$\frac{dL}{dp}(p_0) = \frac{df}{dp}(p_0) = \nabla_p L = x.$$

Demnach stellen die primalen Variablen die Sensitivitätsableitung der Lagrange- und Zielfunktion dar.

4.2.2 Quadratische Störung der Zielfunktion

Auf Grund des Satzes von Schwarz und den bereits für den Sensitivitätssatz 4.4 getroffenen Annahmen ist gesichert, dass die Hesse-Matrix der Lagrange-Funktion symmetrisch ist. Deshalb sei die Störung $\Psi \in \mathbb{R}^{N_x \times N_x}$ eine symmetrische Matrix. Weiterhin sei Ψ so gewählt, dass es dieselbe Schwachbesetztheitsstruktur wie $\nabla^2_{xx} L$ habe. Die Matrix Ψ soll zunächst nur zur Analyse der Auswirkung von Störungen auf die Hesse-Matrix verwendet werden, deshalb stellt diese Einschränkung der Struktur keinen Nachteil dar. Die allgemeine Problemformulierung des nichtlinearen Optimierungsproblems mit quadratischer Störung der Zielfunktion ist

$$\begin{aligned} \min_{x \in \mathbb{R}^{N_x}} \quad & f(x) + \tfrac{1}{2} x^\top \Psi x \\ \text{unter} \quad & g_i(x) \leq 0, \quad i = 1, ..., N_g, \\ & h_j(x) = 0, \quad j = 1, ..., N_h. \end{aligned} \tag{4.39}$$

Die matrixwertige Störung $\Psi \in \mathbb{R}^{N_x \times N_x}$ kann durch einen Vektor $p \in \mathbb{R}^{(N_x{}^2)}$, mit p als zeilenweise vektorisierter Matrix Ψ, dargestellt werden. In diesem Fall ist $\nabla^2_{xp} L[p_0] \in \mathbb{R}^{N_x \times N_x \times N_x}$ ein Rang-3 Tensor. Weiterhin ist leicht ersichtlich, dass $\nabla_p g[p_0] = 0$ und $\nabla_p h[p_0] = 0$ gelten.
Zur Berechnung der rechten Seite von Gleichung (4.23) müssen die Ableitungen bezüglich der Matrixeinträge von Ψ bestimmt werden. Die Ableitung bezüglich des Eintrages Ψ_{ij} ist

$$\frac{\partial \left(\nabla_x L\right)_k}{\partial \Psi_{ij}} = \begin{cases} x_j & \text{wenn} \qquad k = i \\ 0 & \text{sonst} \end{cases}$$

$$\Rightarrow \frac{\partial \nabla_x L[p_0]}{\partial \Psi_{ij}} = x_j \cdot e_{\{i\}} \qquad \text{für } i, j = 1, ..., N_x. \tag{4.40}$$

Obwohl $\nabla^2_{xp} L[p_0]$ ein Rang-3 Tensor ist, können diese Ableitungen ohne zusätzlichen Rechenaufwand direkt bestimmt werden und aus Gleichung (4.40) ergibt sich

$$n_{nz}(\nabla^2_{xp} L[p_0]) = n_{nz}(\nabla^2_{xx} L) \tag{4.41}$$

für die Anzahl der Nichtnulleinträge. Die Sensitivitätsableitungen der Nebenbedingungen ergeben sich für derartig strukturierte Störungen analog zu den Betrachtungen über natürliche Störungen der Zielfunktion aus Abschnitt 4.2.1. Gleichung (4.29) vereinfacht sich zu

$$\frac{\mathrm{d}g_i}{\mathrm{d}p}(p_0) = \nabla_x g_i \frac{\mathrm{d}x}{\mathrm{d}p}, \quad i = 1, ..., N_g. \tag{4.42}$$

Die Sensitivitätsableitung von Lagrange- und Zielfunktion kann unter Verwendung der Gleichungen (4.31) und (4.33) durch

$$\frac{\mathrm{d}L}{\mathrm{d}p}(p_0) = \frac{\mathrm{d}f}{\mathrm{d}p}(p_0) = \nabla_p L = \frac{1}{2} x x^\top \tag{4.43}$$

dargestellt werden.
In der Regel werden in der numerischen Praxis Teilstrukturen von Ψ bei der Implementierung von Lösungsmethoden für nichtlineare Optimierungsprobleme verwendet. In Kapitel 3 wurden verschiedene Regularisierungsansätze für Hesse-Matrizen diskutiert. Es kann zum Beispiel mit der Störung

$$\Psi_d = \begin{pmatrix} p_1 & & \\ & \ddots & \\ & & p_{N_x} \end{pmatrix} \tag{4.44}$$

mit der Dimension des Parameters $p \in \mathbb{R}^{N_x}$ eine Diagonalmatrix zur Regularisierung dargestellt werden. In diesem Fall ist $\nabla^2_{xp} L[p_0] \in \mathbb{R}^{N_x \times N_x}$ und die rechte Seite von Gleichung (4.23) ist

$$\nabla^2_{xp} L[p_0] = \begin{pmatrix} x_1 & & \\ & \ddots & \\ & & x_{N_x} \end{pmatrix}. \tag{4.45}$$

Auch dies ist ohne zusätzlichen Rechenaufwand möglich. Diese Störungsart kann benutzt werden, um Regularisierungen vom Levenberg-Marquardt-Typ mit unterschiedlichen Parametern für jede Zeile von $\nabla L_{xx}^2[p_0]$ zu analysieren. Unter Berücksichtigung der veränderten Sensitivitätsableitungen $\frac{\mathrm{d}x}{\mathrm{d}p}$ behält die Formel (4.42) ihre Gültigkeit. Die Berechnung der Sensitivitätsableitung der Lagrange- und Zielfunktion kann durch

$$\frac{\mathrm{d}L}{\mathrm{d}p}(p_0) = \frac{\mathrm{d}f}{\mathrm{d}p}(p_0) = \nabla_p L = \frac{1}{2}\begin{pmatrix} x_1^2 \\ \vdots \\ x_{N_x}^2 \end{pmatrix} \tag{4.46}$$

erfolgen.
Eine weitere Vereinfachung wird insbesondere in den Abschnitten 3.3 und 5.3 thematisiert. Die Regularisierung $\nabla L_{xx}^2[p_0] + pI_{N_x}$ benötigt lediglich einen skalaren Störungsparameter $p \in \mathbb{R}$ und die rechte Seite $\nabla_{xp}^2 L[p_0] \in \mathbb{R}^{N_x}$ ist durch

$$\nabla_{xp}^2 L[p_0] = x \tag{4.47}$$

bestimmt. Somit können die Sensitivitätsableitungen für die primalen und dualen Variablen für diese Störungsart sehr effizient berechnet werden. Auch in diesem Fall müssen für die Sensitivitäten der Nebenbedingungen wieder lediglich die geänderten Sensitivitäten der Variablen berücksichtigt werden. Für die Sensitivitäten der Lagrange- und Zielfunktion ergibt sich die Formel

$$\frac{dL}{dp}(p_0) = \frac{df}{dp}(p_0) = \nabla_p L = \frac{1}{2}x^\top x, \tag{4.48}$$

welche auf Grund des Zusammenhanges zur Regularisierung im späteren Verlauf dieser Arbeit erneut aufgegriffen wird.

4.2.3 Additive Störung der Nebenbedingungen

Es sei $q \in \mathbb{R}^{N_g+N_h}$ eine weiterer Spezialfall der allgemeinen Störung p. Die Störung q sei eine additive Störung der Nebenbedingungen. Die erweiterte Problemformulierung ist

$$\begin{array}{lll} \min\limits_{x\in\mathbb{R}^{N_x}} & f(x) & \\ \text{unter} & g_i(x) + q_i & \leq 0, \quad i = 1, ..., N_g, \\ & h_j(x) + q_{N_g+j} & = 0, \quad j = 1, ..., N_h. \end{array} \tag{4.49}$$

Die Störung q kann als Störung der Zulässigkeit interpretiert werden. Büskens [12] nutzt diese Zusammenhänge, um einen Algorithmus zur echtzeitfähigen Zulässigkeitskorrektur zu formulieren.

Die rechte Seite in Gleichung (4.23) kann für diese spezielle Störung wie folgt vereinfacht werden:

$$\begin{pmatrix} \nabla^2_{xp} L[p_0] \\ \Lambda \nabla_p g[p_0] \\ \nabla_p h[p_0] \end{pmatrix} = \begin{pmatrix} 0 \\ \Lambda_{N_g} \\ I_{N_h} \end{pmatrix} \tag{4.50}$$

Demnach kann diese ohne zusätzlichen Rechenaufwand aufgestellt werden. Es sei durch $e_{\{k\}}$ der k-te Einheitsvektor bezeichnet. Durch Auswertung von Gleichung (4.29) ergibt sich die Formel

$$\frac{\mathrm{d}g_i}{\mathrm{d}p}(p_0) = \nabla_x g_i \frac{\mathrm{d}x}{\mathrm{d}p} + e_{\{i\}},\ i = 1, ..., N_g \tag{4.51}$$

zur Bestimmung der Sensitivitäten der Nebenbedingungen. Die Sensitivitäten von Lagrange- und Zielfunktion können nach Gleichung (4.33) durch die Beziehung

$$\frac{dL}{dp} = \frac{df}{dp} = \left(\lambda_1, , ..., \lambda_{N_g}, \mu_1, ..., \mu_{N_h} \right)^\top \tag{4.52}$$

dargestellt werden. Dieses Ergebnis dient zur Interpretation der dualen Variablen als Schattenpreise, wie beispielsweise Büskens [11] oder Spellucci [97] erläutern.

4.2.4 Lineare Störung der Nebenbedingungen

In den vorherigen Abschnitten wurden bereits Störungen für die lineare und die quadratische Komponente der Zielfunktion, sowie additive Störungen der Nebenbedingungen diskutiert. Die letzte für quadratische Optimierungsprobleme relevante Störung wirkt sich auf die Matrizen in den Nebenbedingungen aus. Hierzu seien Störungsmatrixen $\mathcal{T} \in \mathbb{R}^{N_g \times N_x}$ und $\mathcal{W} \in \mathbb{R}^{N_h \times N_x}$ gegeben. Weiterhin sei die zusammengesetzte Matrix

$$\mathcal{R} \coloneqq \begin{pmatrix} \mathcal{T} \\ \mathcal{W} \end{pmatrix} \tag{4.53}$$

als weiterer Spezialfall des allgemeinen Störungsparameters p mit
$N_p = (N_g + N_h) \times N_x$ definiert. Die Problemstellung

$$\begin{array}{lll} \min\limits_{x \in \mathbb{R}^{N_x}} & f(x) & \\ \text{unter} & g(x) + \mathcal{T}x & \leq\ 0, \\ & h(x) + \mathcal{W}x & =\ 0. \end{array} \tag{4.54}$$

konkretisiert die Einbindung der Matrizen $\mathcal{T}$ und $\mathcal{W}$. Der Gradient der Lagrange-Funktion dieses Problems ist durch

$$\nabla_x f[p_0] + (\nabla_x g[p_0] + \mathcal{T})^\top \lambda + (\nabla_x h[p_0] + \mathcal{W})^\top \mu$$

gegeben. Analog zur Betrachtung der quadratischen Störung der Zielfunktion Ψ in Abschnitt 4.2.2 seien die Schwachbesetztheitsstrukturen von $\mathcal{T}$ und $\nabla_x g[p_0]$ sowie von $\mathcal{W}$ und $\nabla_x h[p_0]$ als identisch angenommen.
Bei der Bestimmung der Sensitivitätsableitungen der primalen und dualen Variablen ist die rechte Seite in Gleichung (4.23) $\nabla^2_{xp} L[p_0] \in \mathbb{R}^{N_x \times (N_g+N_h) \times N_x}$ erneut ein Rang-3 Tensor. Aber auch in diesem Fall ist dieser schwachbesetzt.
Mit $i = 1, ..., N_x$, $j = 1, ..., N_g + N_h$ und $k = 1, ..., N_x$ können über die Formel

$$(\nabla^2_{xp} L[p_0])_{ijk} = \frac{\partial L}{\partial x_i \partial \mathcal{R}_{jk}} = \begin{cases} \lambda_j, & \text{wenn } k = i, \qquad \forall j = 1, ..., N_g \\ \mu_j, & \text{wenn } k = i, \qquad \forall j = N_g + 1, ..., N_g + N_h \\ 0, & \text{sonst} \end{cases}$$

die Einträge einfach bestimmt werden.
Weiterhin müssen noch die Ableitungen der Nebenbedingungen bezüglich der Störung bestimmt werden. Auch diese sind für diese Störungsart ein schwachbesetzter Rang-3 Tensor. Dessen Einträge können mit den Gleichungen

$$(\nabla_p g[p_0])_{ijk} = \frac{\partial g_i}{\partial \mathcal{R}_{jk}} = \begin{cases} x_k, & \text{wenn } \quad i = j, \qquad \forall k = 1, ..., N_x, \\ 0, & \text{sonst} \end{cases}$$

$$(\nabla_p h[p_0])_{ijk} = \frac{\partial h_i}{\partial \mathcal{R}_{jk}} = \begin{cases} x_k, & \text{wenn } \quad N_g + i = j, \qquad \forall k = 1, ..., N_x \\ 0, & \text{sonst} \end{cases}$$

bestimmt werden.
Zur Bestimmung der Sensitivitätsableitungen der Nebenbedingungen seien
$\hat{g}(x) := g(x) + \mathcal{T}x$ und $\hat{h}(x) := h(x) + \mathcal{W}x$ zur Vereinfachung der folgenden Formeln definiert. Es gilt

$$\hat{g}_i(x) = g_i(x) + \sum_{k=1}^{N_x} \mathcal{T}_{ik} x_i,$$

$$\hat{h}_j(x) = h_j(x) + \sum_{k=1}^{N_x} \mathcal{W}_{jk} x_j,$$

woraus

$$\nabla_p \hat{g}_i = e_{\{i\}} \cdot x^\top \quad \text{für } i = 1, ..., N_g \tag{4.55}$$

ersichtlich wird. Analog gilt

$$\nabla_p \hat{h}_j = e_{\{j\}} \cdot x^\top \quad \text{für } j = 1, ..., N_h. \tag{4.56}$$

für die Gleichheitsnebenbedingungen. Durch Einsetzen der Gleichung (4.55) in Gleichung (4.29) können die gewünschten Sensitivitätsableitungen der Ungleichungsnebenbedingungen bestimmt werden.
Zur Berechnung der Sensitivitätsableitungen der Lagrange- und Zielfunkion müssen einige Tensorprodukte ausgewertet werden. Die Dimensionen der beteiligten Größen

sind $\lambda \in \mathbb{R}^{\mathbb{N}_g}$, $\mu \in \mathbb{R}^{\mathbb{N}_h}$, $\nabla_p g \in \mathbb{R}^{N_g \times (N_g+N_h) \times N_x}$ und $\nabla_p h \in \mathbb{R}^{N_h \times (N_g+N_h) \times N_x}$. Demnach sind die resultierenden Dimensionen durch $\lambda^\top \nabla_p g \in \mathbb{R}^{(N_g+N_h) \times N_x}$ und $\mu^\top \nabla_p h \in \mathbb{R}^{(N_g+N_h) \times N_x}$ gegeben. Weiterhin gilt

$$\lambda^\top \nabla_p g = \sum_{k=1}^{N_g} \lambda_k \nabla_p g_k = \begin{pmatrix} \lambda_1 x^\top \\ \vdots \\ \lambda_{N_g} x^\top \\ 0 \\ \vdots \\ 0 \end{pmatrix} \quad \text{and} \quad \mu^\top \nabla_p h = \sum_{k=1}^{N_g} \mu_k \nabla_p h_k = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ \mu_1 x^\top \\ \vdots \\ \mu_{N_h} x^\top \end{pmatrix}.$$

Zusammen mit Gleichung (4.33) und $\nabla_p f = 0$ führt dies zu

$$\frac{\mathrm{d}f}{\mathrm{d}p} = \frac{\mathrm{d}L}{\mathrm{d}p} = \left(\lambda_1 x^\top, ..., \lambda_{N_g} x^\top, \mu_1 x^\top, ..., \mu_{N_h} x^\top \right)^\top, \tag{4.57}$$

welches eine verallgemeinerte Variante von Gleichung (4.52) darstellt. Die Sensitivitätsableitungen der Nebenbedingungen bezogen auf lineare Störungen sind von besonderem Interesse im Zusammenhang mit SQP-Methoden. Die matrixwertige Komponente der Nebenbedingungen innerhalb der Unterprobleme des SQP-Verfahrens entspricht gerade der Jacobi-Matrix der Nebenbedingungen, wie zum Beispiel die Erläuterungen im Abschnitt 3.2.1 gezeigt haben. Somit können durch die hier beschriebene speziell strukturierte Störung die Auswirkungen von fehlerbehafteten Einträgen innerhalb der Jacobi-Matrizen der Nebenbedingungen analysiert werden. Weiterhin zeigen die Überlegungen, dass $\nabla_x L[p_0]$ lediglich von den Lagrange-Multiplikatoren abhängt, wohingegen die Ableitungen von $g[p_0]$ und $h[p_0]$ nur von den primalen Variablen beeinflusst wird.
Auch in diesem Zusammenhang könnte durch spezielle Wahl der Störungen der Rang-3 Tensor vermieden werden. Zwei mögliche Vereinfachungen sind naheliegend. Einerseits könnte $\mathcal{R}$ lediglich ein Zeilenvektor sein, dies könnte entweder der Störung einer einzelnen Nebenbedingung oder einer identischen Störung für alle Nebenbedingungen entsprechen. Andererseits könnte $\mathcal{R}$ ein einzelner Spaltenvektor sein, in diesem Fall würden alle Nebenbedingungen eine Störung in derselben Komponente von x erleiden.

4.3 Echtzeitoptimierung

Die linearen Approximationen $\tilde{x}(p), \tilde{\lambda}(p)$ und $\tilde{\mu}(p)$ der Funktionen $x(p), \mu(p)$ und $\lambda(p)$ aus den Gleichungen (4.24), (4.26) und (4.25) ermöglichen es Büskens in [12] echtzeitfähige Algorithmen zu konstruieren, um die Anpassung an abweichende Parameter nach der Lösung des Nominalproblems zu realisieren. Mit dem folgenden Satz quantifiziert er die entstehenden Fehler in den resultierenden Näherungswerten. Hierbei werden lediglich die aktiven Ungleichungsnebenbedingungen betrachtet, deshalb seien diese im Folgenden durch $g_{\mathcal{A}}(x(p), p)$ bezeichnet.

Algorithmus 9 Zulässigkeitsselbstkorrektur

1: Wähle $\epsilon > 0$ und berechne $\tilde{x}^{[1]}(p)$ durch Gleichung (4.24), setze $k := 1$
2: $q_{\mathcal{A}}^{[1]} \leftarrow \left(g_{\mathcal{A}}(\tilde{x}^{[1]}), h(\tilde{x}^{[1]}) \right)$
3: **while** $\|q_{\mathcal{A}}^{[k]}\|_\infty > \epsilon$ **do**
4: $\quad \tilde{x}^{[k+1]}(p) := \tilde{x}^{[k]}(p) - \frac{\mathrm{d}x}{\mathrm{d}q_{\mathcal{A}}}(p_0) q_{\mathcal{A}}^{[k]}$
5: $\quad q_{\mathcal{A}}^{[k]} \leftarrow \left(g_{\mathcal{A}}(\tilde{x}^{[k]}), h(\tilde{x}^{[k]}) \right)$
6: **end while**

Satz 4.13. *Es seien die Voraussetzungen des Sensitivitätssatzes (Satz 4.4) erfüllt. Weiterhin seien die Funktionen f, g und h bezüglich x und p dreimal stetig differenzierbar. Dann existiert eine Umgebung $U(p_0)$ um die Nominalstörung p_0, so dass für alle $p = p_0 + \Delta p \in U(p_0)$ die folgenden Fehlerabschätzungen gelten:*

$$\begin{aligned}
\|x(p) - \tilde{x}(p)\| &= \mathcal{O}(\|\Delta p\|^2) \\
\|\lambda(p) - \tilde{\lambda}(p)\| &= \mathcal{O}(\|\Delta p\|^2) \\
\|\mu(p) - \tilde{\mu}(p)\| &= \mathcal{O}(\|\Delta p\|^2) \\
\|f(x(p), p) - f(\tilde{x}(p), p)\| &= \mathcal{O}(\|\Delta p\|^2) \\
\|g_{\mathcal{A}}(x(p), p) - g_{\mathcal{A}}(\tilde{x}(p), p)\| &= \mathcal{O}(\|\Delta p\|^2) \\
\|h(x(p), p) - h(\tilde{x}(p), p)\| &= \mathcal{O}(\|\Delta p\|^2) \\
\|\nabla_x L(\tilde{x}(p), \tilde{\lambda}(p), \tilde{\mu}(p), p)\| &= \mathcal{O}(\|\Delta p\|^2)
\end{aligned} \tag{4.58}$$

Beweis. Der Beweis erfolgt über Verwendung geeigneter Taylor-Approximationen und ist in der Arbeit von Büskens [12] zu finden. □

Ausgehend von diesen Erkenntnissen kann zur Korrektur einer Störung $p \in U(p_0) \subset \mathbb{R}^{N_p}$ mit Hilfe von Gleichung (4.24) eine Näherung der Lösung $x^*(p)$ gefunden werden. Gleichung (4.58) zeigt, dass durch diese Korrektur ein Fehler in den Nebenbedingungen induziert wird. Dieser Fehler entspricht einer Störung q aus Abschnitt 4.2.3. Mit Hilfe einer weiteren Anwendung von Gleichung (4.24) für die passende Störung $q_{\mathcal{A}} = \left(g_{\mathcal{A}}(\tilde{x}^{[1]}), h(\tilde{x}^{[1]}) \right)$ kann der entstehende Fehler in den Nebenbedingungen verbessert werden. Diese Korrektur induziert erneut einen Fehler in den Nebenbedingungen, welcher wieder korrigiert werden kann. Diese Beobachtungen nutzt Büskens zur Definition des Algorithmus 9 zur Zulässigkeitsselbstkorrektur. Weiterhin diskutiert er die Frage nach den Konvergenzeigenschaften dieses Algorithmus. Seine Folgerungen liefern den folgenden Satz.

Satz 4.14. *Es seien die Voraussetzungen des Sensitivitätssatzes 4.4 erfüllt und die Funktionen f, g und h seien dreimal stetig differenzierbar bezüglich x und p. Dann*

existiert eine Umgebung $U(p_0)$, so dass für alle $p = p_0 + \Delta p \in U(p_0)$ die folgenden Aussagen gelten. Betrachtet man die orthogonale Zerlegung

$$\frac{1}{2}(\Delta p)^\top \frac{\mathrm{d}^2 x}{\mathrm{d}p^2}(p_0)\Delta p = v + w, \tag{4.59}$$

mit $v \in \ker(\nabla_x g_{\mathcal{A}}(p_0), \nabla_x h(p_0)) \subset \mathbb{R}^{N_x}$ und $w \in (\ker(\nabla_x g_{\mathcal{A}}(p_0), \nabla_x h(p_0)))^\perp \subset \mathbb{R}^{N_x}$, so gelten unter Verwendung der Iterationsvorschrift

$$\tilde{x}^{[k+1]}(p) := \tilde{x}^{[k]}(p) - \frac{\mathrm{d}x}{\mathrm{d}q}(p_0)q^{[k]} \tag{4.60}$$

mit $q^{[k]} = (g(\tilde{x}^{[k]}(p), p), h(\tilde{x}^{[k]}(p), p))$ die Fehlerabschätzungen

$$\begin{aligned} \|v\| &= \mathcal{O}(\|\Delta p\|^2), \\ \|x(p) - \tilde{x}^{[\infty]}(p)\| &= \|v\| + \mathcal{O}(\|\Delta p\|^3), \\ \|f(x(p), p) - f(\tilde{x}^{[\infty]}(p), p)\| &= \mathcal{O}(\|\Delta p\|^3), \\ \|g_{\mathcal{A}}(\tilde{x}^{[\infty]}(p), p)\| &= 0. \end{aligned} \tag{4.61}$$

Werden auch die Lagrange-Multiplikatoren iterativ durch

$$\begin{aligned} \tilde{\lambda}^{[k+1]}(p) &:= \tilde{\lambda}^{[k]}(p) - \frac{\mathrm{d}\lambda_{\mathcal{A}}}{\mathrm{d}q_{\mathcal{A}}}(p_0) g_{\mathcal{A}}(\tilde{x}^{[k]}(p), p) \\ \tilde{\mu}^{[k+1]}(p) &:= \tilde{\mu}^{[k]}(p) - \frac{\mathrm{d}\mu}{\mathrm{d}q_{\mathcal{A}}}(p_0) h(\tilde{x}^{[k]}(p), p) \end{aligned} \tag{4.62}$$

angepasst, so konvergieren diese gegen Fixpunkte $\tilde{\lambda}^{[\infty]}(p)$ und $\tilde{\mu}^{[\infty]}(p)$. Es gelten die folgenden Fehlerabschätzungen:

$$\begin{aligned} \|\lambda(p) - \tilde{\lambda}^{[\infty]}(p)\| &= \mathcal{O}(\|\Delta p\|^2), \\ \|\mu(p) - \tilde{\mu}^{[\infty]}(p)\| &= \mathcal{O}(\|\Delta p\|^2), \\ \|\nabla_x L(\tilde{x}^{[\infty]}(p), \tilde{\lambda}^{[\infty]}(p), \tilde{\mu}^{[\infty]}(p), p)\| &= \mathcal{O}(\|\Delta p\|^2). \end{aligned} \tag{4.63}$$

Die durch die Iterationsvorschriften (4.60) *und* (4.62) *entstehenden Folgen konvergieren linear und die Fixpunkte $\tilde{x}^{[\infty]}(p), \tilde{\lambda}^{[\infty]}(p)$ und $\tilde{\mu}^{[\infty]}(p)$ sind abhängig vom Nominalwert p_0 und der Störung p. Es sind somit*

$$\begin{aligned} \tilde{x}^{[\infty]}(p) &= \tilde{x}^{[\infty]}(p; p_0), \\ \tilde{\lambda}^{[\infty]}(p) &= \tilde{\lambda}^{[\infty]}(p; p_0), \\ \tilde{\mu}^{[\infty]}(p) &= \tilde{\mu}^{[\infty]}(p; p_0). \end{aligned} \tag{4.64}$$

Beweis. Diesen Satz beweist Büskens ebenfalls in seiner Arbeit [12]. □

Die speziell strukturierten Störungen in Kombination mit den Echtzeitoptimierungsalgorithmen stellen alle Werkzeuge bereit, damit im folgenden Kapitel die Einbindung der parametrischen Sensitivitätsanalyse innerhalb des SQP-Verfahrens erfolgen kann.

Kapitel 5

Parametrische Sensitivitätsanalyse innerhalb des SQP-Verfahrens

Inhaltsangabe

Die parametrische Sensitivitätsanalyse bietet die Möglichkeit eine Postoptimalitätsanalyse eines Optimierungsproblems nach dessen Lösung durchzuführen. Das Ziel dieser Analyse ist die Auswirkungen von verschiedenen Störungen auf das Ergebnis der Optimierung zu quantifizieren. In diesem Kapitel werden verschiedene

Techniken diskutiert wie diese Methoden zur Verbesserung eines SQP-Verfahrens ausgenutzt werden können.
Eine erste Umsetzung derartiger Ansätze beschrieb Nikolayzik in seiner Dissertation [81]. Diese Ergebnisse werden hier aufgegriffen, analysiert und erweitert.
Während der Durchführung eines SQP-Algorithmus, wie in den Algorithmen 3 und 4 beschrieben, müssen die quadratischen Unterprobleme in jeder Iteration zur Bestimmung der Suchrichtung gelöst werden. Die verschiedenen Algorithmen innerhalb dieses Kapitels basieren auf einer Postoptimalitätsanalyse der Lösungen dieser Unterprobleme. Einleitend zeigt Abschnitt 5.1, dass die zur Bestimmung der Sensitivitätsableitungen benötigte Matrixzerlegung bei Verwendung eines Innere-Punkte-Verfahrens zur Lösung der quadratischen Unterprobleme bereits vorliegt. Somit lassen sich die innerhalb der vorgestellten Algorithmen benötigten Sensitivitäten nahezu ohne zusätzlichen numerischen Aufwand bestimmen.
In Abschnitt 5.2 wird daraufhin die Zulässigkeitskorrektur zur Verbesserung der Suchrichtung vorgestellt. Es handelt sich um eine Anwendung des im Abschnitt 4.3 beschriebenen echtzeitfähigen Korrekturverfahrens. Anfangs wird eine generelle Umsetzung des Verfahrens innerhalb des SQP-Verfahrens diskutiert. Anschließend werden Kriterien vorgestellt, die helfen zu beurteilen in welchen Situationen es sinnvoll ist die Korrektur einzusetzen und wie lange diese durchgeführt werden soll.
Als weitere analysierbare Größe kann die Regularisierung der Hesse-Matrix aus Abschnitt 3.3.3.1 erneut aufgegriffen werden. In diesem Zusammenhang werden in Abschnitt 5.3 Strategien vorgestellt, die unter Ausnutzung der parametrischen Sensitivitäten darauf abzielen die Einbindung der Regularisierung innerhalb des SQP-Verfahrens zu verbessern.
Zuletzt werden die parametrischen Sensitivitäten im Abschnitt 5.4 verwendet, um eine Analyse der Auswirkungen der Relaxierung der Nebenbedingungen aus Abschnitt 3.3.3.3 vorzunehmen. Mit Hilfe der daraus gewonnenen Informationen wird ein Algorithmus zur Anpassung der Relaxierungsparameter konstruiert.
In diesem Kapitel werden verschiedene Algorithmen beschrieben, die während des SQP-Algorithmus durchgeführt werden. Die Bezeichnung *Hauptiteration* bezieht sich in diesem Kontext auf die Iterationen des SQP-Algorithmus, während Iteration dem Kontext entsprechend freier verwendet wird und sich in der Regel auf die zu beschreibenden Algorithmen bezieht.

5.1 Sensitivitätsableitungen der quadratischen Unterprobleme

Die quadratischen Unterprobleme innerhalb von WORHP werden, wie in Abschnitt 3.3.4 beschrieben, mit Hilfe des Innere-Punkte-Verfahrens aus Abschnitt 3.1.2.2 gelöst. Im Folgenden werden kurz die Besonderheiten bei der Anwendung des Sensitivitätssatzes in dieser Situation erläutert, da die spezielle Struktur der zu zerlegenden

Matrix zur Bestimmung der Suchrichtung nach Einführung der Schlupfvariablen ein paar Anmerkungen erfordert. Eine alternative Beschreibung der Zusammenhänge findet sich in der Arbeit von Nikolayzik [81].
Die folgenden Betrachtungen berücksichtigen die Relaxierungsvariablen δ nicht. Diese werden innerhalb von WORHP als zusätzliche Optimierungsvariablen, und somit als weitere Komponenten des Vektors d, in das Problem integriert (Vergleiche Abschnitt 3.3.4) und können deshalb an dieser Stelle ohne Beschränkung der Allgemeinheit ignoriert werden.
Der Sensitivitätssatz 4.4 liefert Gleichung (4.23) zur Bestimmung der parametrischen Sensitivitätsableitungen für die primalen und dualen Variablen. Die Funktionen des quadratischen Unterproblems (3.152) ohne Relaxierung innerhalb von WORHP seien abkürzend durch

$$\begin{aligned}
\tilde{f}(d,p_0) &:= \frac{1}{2}d^\top \tilde{Q} d + \nabla_x f(x^{[k]},p_0),\\
\tilde{g}_i(d,p_0) &:= g_i(x^{[k]},p_0) + \nabla_x g_i(x^{[k]},p_0)^\top d \le 0, \qquad i = 1,...,N_g\\
\tilde{h}_j(d,p_0) &:= h_j(x^{[k]},p_0) + \nabla_x h_j(x^{[k]},p_0)^\top d = 0, \qquad j = 1,...,N_h\\
\tilde{L}(d,y,z,p_0) &:= \tilde{f}(d,p_0) + z^\top \tilde{g}(d,p_0) + y^\top \tilde{h}(d,p_0)
\end{aligned}$$

gegeben. Es sei (d^*, y^*, z^*) die optimale Lösung des quadratischen Unterproblems des SQP-Algorithmus, dann folgt aus Gleichung (4.23)

$$-\begin{pmatrix} Q(x^{[k]},\lambda^{[k]},\mu^{[k]},p_0) & \nabla_x h(x^{[k]},p_0)^\top & \nabla_x g(x^{[k]},p_0)^\top \\ \nabla_x h(x^{[k]},p_0) & 0 & 0 \\ Z^*\nabla_x g(x^{[k]},p_0) & 0 & \tilde{\Gamma}(d^*,p_0) \end{pmatrix} \begin{pmatrix} \frac{\mathrm{d}d}{\mathrm{d}p}(p_0) \\ \frac{\mathrm{d}y}{\mathrm{d}p}(p_0) \\ \frac{\mathrm{d}z}{\mathrm{d}p}(p_0) \end{pmatrix} = \begin{pmatrix} \nabla^2_{dp}\tilde{L}(d^*,y^*,z^*,p_0) \\ \nabla_p \tilde{h}(d^*,p_0) \\ Z^*\nabla_p \tilde{g}(d^*,p_0) \end{pmatrix} \tag{5.1}$$

zur Bestimmung der Sensitivitätsableitungen. Die Matrix $\tilde{\Gamma} \in \mathbb{R}^{N_g \times N_g}$ ist durch

$$\tilde{\Gamma}(d^*,p_0) = \begin{pmatrix} \tilde{g}_1(d^*,p_0) & & \\ & \ddots & \\ & & \tilde{g}_{N_g}(d^*,p_0) \end{pmatrix}$$

gegeben. Bei der Anwendung des Innere-Punkte-Verfahrens, wie in Algorithmus 2 beschrieben, wird eine Zerlegung der Matrix in Gleichung (3.55) bestimmt. Nach Abschluss des Innere-Punkte-Verfahrens erfüllen die Schlupfvariablen die Gleichungen

$$s_i = -\tilde{g}_i, \quad i = 1,...,N_g, \tag{5.2}$$

so dass die Äquivalenz der Matrizen aus Gleichung (3.53) und aus Gleichung (5.1) direkt ersichtlich ist. Es ist aus theoretischer Sicht zu beachten, dass $z_j^* = 0$ für $1 \le j \le N_g$ auftreten kann und somit die entsprechende Division zur Symmetrisierung des Systems in der optimalen Lösung nicht durchgeführt werden kann. In der numerischen Praxis kann dies aber vernachlässigt werden, da zur Berechnung der Sensitivitäten für die im weiteren Verlauf dieses Abschnitts vorgestellten Verfahren die Zerlegung aus der letzten Iteration des Unterproblemlösers verwendet wird.

Deshalb gilt weiterhin die strikte Positivität für die Multiplikatoren und Schlupfvariablen und es muss lediglich beachtet werden, dass die ermittelten Sensitivitäten einen zusätzlichen, aber vernachlässigbaren, numerischen Fehler aufweisen.
Auch eine effiziente Behandlung der Boxschranken an die Optimierungsvariablen innerhalb des quadratischen Problems lässt sich wie in Abschnitt 3.1.2.2 auf die Berechnung der parametrischen Sensitivitätsableitungen übertragen. Die folgenden Betrachtungen zeigen, dass ausgehend von Gleichung (5.1) die Berechnung der Sensitivitätsableitungen im Falle der effizienten Behandlung der Boxschranken weiterhin möglich ist. Es seien $B \in \mathbb{R}^{N_{\text{box}} \times N_x}$ und Schranken $c_{\text{box}} \in \mathbb{R}^{N_{\text{box}}}$ gegeben, dann lassen sich die Boxschranken an die Optimierungsvariablen durch

$$b(d) := Bd - c_{\text{box}}$$

mit $b(d) \leq 0$ passend zur Problemstellung (3.152) wählen. Die Matrix B schränkt den Vektor d auf die Komponenten mit Boxschranken ein. Es sei die Indexmenge der Dimensionen mit Boxschranken durch $\mathcal{K}$ gegeben. Dann ist die Matrix B durch

$$B_{ij} = \begin{cases} 1, & \text{wenn } \mathcal{K}_i = j \\ 0, & \text{sonst} \end{cases}$$

definiert. Somit ergibt sich weiterhin

$$\nabla_d b(d) = B$$

als Jacobi-Matrix dieser Nebenbedingungen. Im Beweis des Sensitivitätssatzes wird der Satz über implizite Funktionen auf die Hilfsfunktion in (4.17) angewendet. Die optimalen Lagrange-Multiplikatoren für die Boxschranken seien in der folgenden Betrachtung durch $\bar{\lambda} \in \mathbb{R}^{N_{\text{box}}}$ bezeichnet und $\bar{\Lambda}$ ist als passende Matrix

$$\bar{\Lambda} := \begin{pmatrix} \bar{\lambda}_1 & & \\ & \ddots & \\ & & \bar{\lambda}_{N_{\text{box}}} \end{pmatrix}$$

definiert. Weiterhin sei $\zeta^* := (d^*, y^*, \bar{\lambda}, z^*)$ zur Vereinfachung definiert. In Anlehnung an Gleichung (4.17) wird eine Hilfsfunktion passend für die Situation mit Boxschranken für das quadratische Unterproblem als

$$F(d, y, \bar{\lambda}, z, p) := \begin{pmatrix} \nabla_d \tilde{L}(d, y, \bar{\lambda}, z, p) \\ \tilde{h}(h, p) \\ \bar{\Lambda} b(x, p) \\ Z \tilde{g}(x, p) \end{pmatrix} = 0 \tag{5.3}$$

definiert. Die Matrix $\mathcal{B}^* \in \mathbb{R}^{N_{\text{box}} \times N_{\text{box}}}$ mit

$$\mathcal{B}^* := \begin{pmatrix} b_1(d^*) & & \\ & \ddots & \\ & & b_{N_{\text{box}}}(d^*) \end{pmatrix} \tag{5.4}$$

enthalte die Werte der Boxschranken in der optimalen Lösung d^*. Die Formel zur Berechnung der Sensitivitätsableitungen (4.23) muss insgesamt zu

$$\begin{pmatrix} \nabla^2_{xx}\tilde{L}(\zeta^*,p_0) & \nabla_x\tilde{h}(d^*,p_0)^\top & B^\top & \nabla_x\tilde{g}(d^*,p_0)^\top \\ \nabla_x\tilde{h}(d^*,p_0) & 0 & 0 & 0 \\ \bar{\Lambda}B & 0 & \mathcal{B}^* & 0 \\ Z\nabla_x\tilde{g}(d^*,p_0) & 0 & 0 & \Gamma^* \end{pmatrix} \begin{pmatrix} \frac{\mathrm{d}d}{\mathrm{d}p}(p_0) \\ \frac{\mathrm{d}y}{\mathrm{d}p}(p_0) \\ \frac{\mathrm{d}\bar{\lambda}}{\mathrm{d}p}(p_0) \\ \frac{\mathrm{d}z}{\mathrm{d}p}(p_0) \end{pmatrix} = - \begin{pmatrix} \nabla^2_{dp}\tilde{L}(\zeta^*,p_0) \\ \nabla_p\tilde{h}(d^*,p_0) \\ \bar{\Lambda}\nabla_p b(d^*,p_0) \\ Z\nabla_p\tilde{g}(d^*,p_0) \end{pmatrix} \tag{5.5}$$

erweitert werden. Es sei zunächst angenommen, dass $b_i(d^*) < 0, i = 1, ..., N_{\text{box}}$ gilt. Unter Ausnutzung der dritten Zeile folgt

$$\begin{aligned} \bar{\Lambda}B\frac{\mathrm{d}d}{\mathrm{d}p}(p_0) + \mathcal{B}^*\frac{\mathrm{d}\bar{\lambda}}{\mathrm{d}p}(p_0) &= -\bar{\Lambda}\nabla_p b(d^*,p_0) \\ \Leftrightarrow \frac{\mathrm{d}\bar{\lambda}}{\mathrm{d}p}(p_0) &= -\mathcal{B}^{*-1}\bar{\Lambda}\left(\nabla_p b(d^*,p_0) - B\frac{\mathrm{d}d}{\mathrm{d}p}(p_0)\right), \end{aligned} \tag{5.6}$$

um die Sensitivitäten der neuen Multiplikatoren $\frac{\mathrm{d}\bar{\lambda}}{\mathrm{d}p}(p_0)$ direkt zu bestimmen.
Es sei zur Vereinfachung

$$\bar{H} := \nabla^2_{xx}\tilde{L}(\zeta^*,p_0) + B^\top\mathcal{B}^{*-1}\bar{\Lambda}B \tag{5.7}$$

definiert. Durch Einsetzen der Beziehung (5.6) in das erweiterte System (5.5) und mit der Vereinfachung aus Gleichung (5.7) ergibt sich das reduzierte System

$$\begin{pmatrix} \bar{H} & \nabla_x\tilde{h}(d^*,p_0)^\top & \nabla_x\tilde{g}(d^*,p_0)^\top \\ \nabla_x\tilde{h}(d^*,p_0) & 0 & 0 \\ Z^*\nabla_x\tilde{g}(d^*,p_0) & 0 & \Gamma^*(d^*,p_0) \end{pmatrix} \begin{pmatrix} \frac{\mathrm{d}d}{\mathrm{d}p}(p_0) \\ \frac{\mathrm{d}y}{\mathrm{d}p}(p_0) \\ \frac{\mathrm{d}z}{\mathrm{d}p}(p_0) \end{pmatrix} = - \begin{pmatrix} \nabla^2_{xp}\tilde{L}(\zeta^*,p_0) - B^\top\mathcal{B}^{*-1}\bar{\Lambda}\nabla_p b(d^*,p_0) \\ \nabla_p\tilde{h}(d^*,p_0) \\ Z^*\nabla_p\tilde{g}(d^*,p_0) \end{pmatrix}. \tag{5.8}$$

Die numerische Beschränkung der Schlupfvariablen durch ϵ_s zur Vermeidung von Problemen bei der Division durch Null führt auf Grund derselben Argumente wie bei der Betrachtung der Symmetrisierung des Systems auch in diesem Zusammenhang zu einem Fehler, weil nicht die Zerlegung im Minimum des Unterproblems verwendet wird. Auf Grund dieses Umstandes konnte für die Begründung der numerischen Vorgehensweise die Annahme $b_i(d^*) < 0, i = 1, ..., N_{\text{box}}$ getroffen werden. Aber unter diesen Einschränkungen kann die während der Lösung des quadratischen Unterproblems erstellte Matrix-Zerlegung auch bei effizienter Behandlung der Boxschranken zur Berechnung der parametrischen Sensitivitätsableitungen verwendet werden.
Theoretisch muss beachtet werden, dass auf Grund des Sensitivitätssatzes 4.4 aus $b_i(d^*) = 0$ direkt $\frac{\mathrm{d}d_i}{\mathrm{d}p}(p_0) = 0$ folgt. Folglich können diese Boxschranken aus dem System zur Berechnung der Sensitivitätsableitungen und aus dem System in Gleichung (3.61) entfernt werden. Weiterhin erfüllen nach Abschluss der Optimierung die verbleibenden Schlupfvariablen $\bar{s} = -b(x^*,p_0)$ und es wird ersichtlich, dass die Matrizen $\bar{Q}$ aus Gleichung (3.61) und $\bar{H}$ aus Gleichung (5.8) identisch sind.

5.2 Zulässigkeitskorrektur

Die Lösung der quadratischen Unterprobleme innerhalb des SQP-Verfahrens soll mit Hilfe der parametrischen Sensitivitätsanalyse verbessert werden. Nikolayzik [81] beschreibt hierzu eine Anwendung der Echtzeitoptimierung durch Zulässigkeitskorrektur (vergleiche Algorithmus 9). Es werden iterativ die nichtlinearen Nebenbedingungen des Ausgangsproblems auf Unterproblemebene berücksichtigt. Eine erste Analyse verschiedener Abbruchbedingungen wurde in der Arbeit des Autors [42] vorgenommen. Die grundlegende Darstellung des Algorithmus basiert auf den Beschreibungen jener Arbeit.
Nach der allgemeinen Vorstellung des Algorithmus werden kurz auf Basis des zu Grunde liegenden Korrekturalgorithmus Eigenschaften der Konvergenz analysiert. Weiterhin werden die Auswirkungen der Verwendung verschiedener Hesse-Matrizen innerhalb des SQP-Verfahrens auf den Algorithmus hinterfragt. Daraufhin wird zur Beschleunigung der Methode eine Erweiterung mit Hilfe der Steffensen-Extrapolation vorgestellt. Vorbereitend für die praktische Umsetzung werden im Anschluss Start- und Abbruchbedingungen diskutiert. Abschließend wird der Zusammenhang zu Verfahren zulässiger Richtungen (feasible direction methods) und zu Korrekturen zweiter Ordnung (second order corrections) aufgezeigt.
Die Lösung von nichtlinearen Optimierungsproblemen mit der SQP-Methode profitiert in der numerischen Umsetzung besonders von der effizienten Lösbarkeit der quadratischen Unterprobleme. Wie bereits im Kapitel 3 beschrieben, werden bei der Aufstellung der Unterprobleme die Nebenbedingungen linearisiert. Allerdings stellt gerade diese Linearisierung eine erhebliche Einschränkung dar. Nichtlinearitäten in den Nebenbedingungen können deshalb die Iterationsanzahl des SQP-Algorithmus stark erhöhen, da die quadratischen Unterprobleme die Nebenbedingungen potenziell nur schlecht approximieren.
Die Zulässigkeitskorrektur der quadratischen Unterprobleme wurde entwickelt, um die Defizite der Linearisierung zu kompensieren. Das Ziel ist es die Einbindung der nichtlinearen Nebenbedingungen in das Unterproblem zu verbessern und so bestenfalls das gesamte SQP-Verfahren zu beschleunigen. Hierzu werden die Verletzungen der Nebenbedingungen des Ausgangsproblems (2.1) durch zusätzliche Auswertungen bestimmt und danach als Störung der Art q, wie in der Problemstellung (4.49) dargestellt, in das quadratische Unterproblem (3.152) integriert.

5.2.1 Beschreibung des Algorithmus

Die Zulässigkeitskorrektur kann immer nach der Lösung der quadratischen Unterprobleme in jeder Hauptiteration des SQP-Verfahrens angewendet werden. Die folgenden Formeln basieren auf den Ausführungen aus der Arbeit von Geffken und Büskens über die Zulässigkeitskorrektur [42]. Zur besseren Lesbarkeit wird der hochgestellte Index $[k]$ der SQP-Iterationen im weiteren Verlauf weggelassen. Es seien deshalb für

den Rest des Abschnitts die Iterierten des SQP-Verfahrens $x := x^{[k]}$, $\hat{x} := x^{[k+1]}$, $\hat{\mu} := \mu^{[k+1]}$, $\hat{\lambda} := \lambda^{[k+1]}$ und $\alpha := \alpha^{[k]}$ definiert. Zusätzlich sei der tiefgestellte Index $[j]$ als Zähler für die Iterationen der Zulässigkeitskorrektur definiert.
Mit der neuen Notation lässt sich Gleichung (3.73) mit $j = 0$ zu

$$\hat{x}_{[0]} = x + \alpha d_{[0]}, \tag{5.9}$$

umschreiben.
Die Sensitivitätsanalyse ermöglicht es die Suchrichtung d als Lösung des quadratischen Unterproblems zu analysieren und mit Hilfe der Formeln (4.24), (4.25) und (4.26) zu verbessern. Als Nominalstörung sei $q_0 = 0$ gewählt. Die Übertragung von Gleichung (4.24) auf diese Situation ergibt

$$d_{[j+1]} = d_{[j]} + \frac{\mathrm{d}d}{\mathrm{d}q}(x, q_0) q_{[j]} \tag{5.10}$$

für die Suchrichtung. Damit ausgehend von Gleichung (5.9) die nächste Iterierte des Korrekturverfahrens bestimmt werden kann, muss Gleichung (5.10) eingesetzt werden. Es ergibt sich die Formel

$$\begin{aligned} \hat{x}_{[1]} &= x + \alpha \left(d_{[0]} + \frac{\mathrm{d}d}{\mathrm{d}q}(x, q_0) q_{[0]} \right) \\ &= \hat{x}_{[0]} + \alpha \frac{\mathrm{d}d}{\mathrm{d}q}(x, q_0) q_{[0]} \end{aligned} \tag{5.11}$$

zur Verbesserung der nächsten Iterierten des SQP-Verfahrens. Eine iterative Anwendung der Taylor-Entwicklung erster Ordnung liefert die Formel

$$\hat{x}_{[j+1]} = \hat{x}_{[j]} + \alpha \left(\frac{\mathrm{d}d}{\mathrm{d}q}(x, q_0) q_{[j]} \right) \tag{5.12}$$

$$= \hat{x}_{[0]} + \alpha \left(\sum_{i=0}^{j} \frac{\mathrm{d}d}{\mathrm{d}q}(x, q_0) q_{[i]} \right). \tag{5.13}$$

Die entsprechenden Taylor-Approximationen für die dualen Variablen lassen sich mit den Formeln

$$\hat{\mu}_{[j+1]} = \hat{\mu}_{[j]} + \frac{\mathrm{d}y}{\mathrm{d}q}(x, q_0) q_{[j]}, \tag{5.14}$$

$$\hat{\lambda}_{[j+1]} = \hat{\lambda}_{[j]} + \frac{\mathrm{d}z}{\mathrm{d}q}(x, q_0) q_{[j]}. \tag{5.15}$$

darstellen.
Zur Realisierung des Algorithmus muss zunächst die Sensitivitätsableitung $\frac{\mathrm{d}d}{\mathrm{d}q}(x, q_0)$ berechnet werden. Die Nebenbedingungen müssen in jedem Schritt ausgewertet werden, damit ihre Verletzung als Störung $q_{[j]}$ verwendet werden kann.
Es seien die aktuell verletzten Nebenbedingungen durch

$$g_i^a(x) := \begin{cases} g_i(x), & \text{falls } \; g_i(x) > 0 \\ 0, & \text{sonst} \end{cases} \qquad\qquad i = 1, ..., N_g \tag{5.16}$$

definiert. Der Vektor $q_{[j]}$ setzt sich insgesamt als

$$q_{[j]} = \left(g^a(\hat{x}_{[j]}), h(\hat{x}_{[j]}) \right)^\top \tag{5.17}$$

aus den aktuellen Werten der Gleichungs- und Ungleichungsnebenbedingungen zusammen. Für den ersten Schritt müssen die Nebenbedingungen des Ausgangsproblems ausgewertet werden, um die initial zu korrigierende Störung $q_{[0]}$ zu bestimmen.

Bemerkung 5.1. *Die Konvergenzaussage des Satzes 4.14 aus der Arbeit von Büskens [11] für den Echtzeitkorrekturalgorithmus gilt lediglich für die Korrektur von Störungen der aktiven Nebenbedingungen. Demnach dürften in Gleichung* (5.17) *nur Verletzungen der nichtlinearen Nebenbedingungen berücksichtigt werden, wenn die entsprechende lineare Nebenbedingung in der optimalen Lösung des quadratischen Unterproblems aktiv war.*
Die späteren Betrachtungen innerhalb dieses Kapitels motivieren aber ebenfalls die Verwendung der Störung q, wie in Gleichung (5.17) *definiert, auch wenn dies im Widerspruch zu den Annahmen von Satz 4.14 steht. Zur Vereinfachung der Notationen in diesem Abschnitt werden deshalb die Betrachtungen anhand von q durchgeführt. In den numerischen Untersuchungen wird dieser Umstand erneut aufgegriffen und der Algorithmus für beide Varianten der Störung q analysiert.*

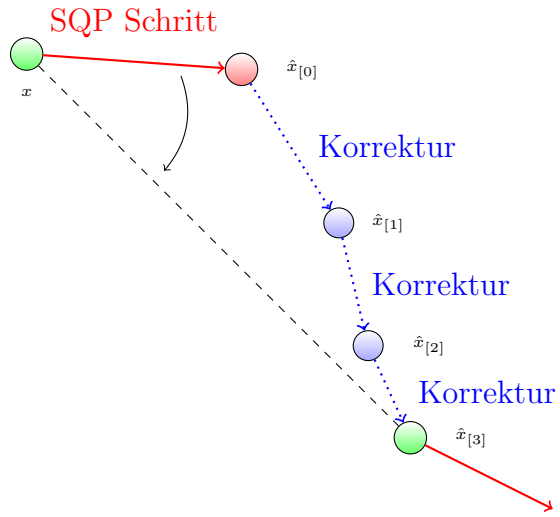


Abbildung 5.1: Darstellung der Zulässigkeitskorrektur über 3 Schritte.

Anschließend wird iterativ mit der Störung $q_{[j]}$ Gleichung (5.12) zur Berechnung des Updates der Iterierten angewendet und die Nebenbedingungen werden erneut ausgewertet. Auf diese Weise wird ein iterativer Algorithmus definiert, der bis zur Erfüllung eines geeigneten Abbruchkriteriums durchgeführt wird. Die Vorgehensweise wird in Abbildung 5.1 schematisch verdeutlicht. Die Qualität der neu berechneten

Iterierten wird überprüft und eine der folgenden Situationen kann eintreten: Der Schritt wird

i. akzeptiert und eine weitere Iteration durchgeführt.

ii. akzeptiert und der Vorgang endet.

iii. verworfen und der Vorgang endet.

Die gesamte Zulässigkeitskorrektur ist in Algorithmus 10 kompakt dargestellt. In

Algorithmus 10 Zulässigkeitskorrektur im SQP-Verfahren

1: $\hat{x}_{[0]} \leftarrow x + \alpha d_{[0]}$
2: Setze $q_0 = 0$ und berechne $\frac{\mathrm{d}d}{\mathrm{d}q}(x, q_0)$, $\frac{\mathrm{d}y}{\mathrm{d}q}(x, q_0)$ und $\frac{\mathrm{d}z}{\mathrm{d}q}(x, q_0)$
3: Setze $j = 0$
4: Wähle ϵ_∞, zum Beispiel $\epsilon_\infty = 10^{-6}$
5: **while** $\left\| \left(g^a(\hat{x}_{[j]}), h(\hat{x}_{[j]}) \right) \right\|_\infty^\top > \epsilon_\infty$ **do**
6: $\quad q_{[j]} \leftarrow \left(g^a(\hat{x}_{[j]}), h(\hat{x}_{[j]}) \right)^\top$
7: $\quad \hat{x}_{[j+1]} \leftarrow \hat{x}_{[j]} + \alpha \left(\frac{\mathrm{d}d}{\mathrm{d}q}(x, q_0) q_{[j]} \right)$
8: $\quad \left(\text{Optional: } \hat{\mu}_{[j+1]} \leftarrow \hat{\mu}_{[j]} + \frac{\mathrm{d}y}{\mathrm{d}q}(x, q_0) q_{[j]}, \quad \hat{\lambda}_{[j+1]} \leftarrow \hat{\lambda}_{[j]} + \frac{\mathrm{d}z}{\mathrm{d}q}(x, q_0) q_{[j]} \right)$
9: $\quad$ Berechne $g^a(\hat{x}_{[j]}), h(\hat{x}_{[j]})$
10: $\quad$ **if** Akzeptanzkriterium (Vergleiche Abschnitt 5.2.4) **then**
11: $\quad\quad$ STOP
12: $\quad$ **else if** Ablehnungskriterium (Vergleiche Abschnitt 5.2.4) **then**
13: $\quad\quad \hat{x}_{[j+1]} \leftarrow \hat{x}_{[j]}, \left(\text{optional: } \hat{\mu}_{[j+1]} \leftarrow \hat{\mu}_{[j]} \; \hat{\lambda}_{[j+1]} \leftarrow \hat{\lambda}_{[j]} \right)$ STOP
14: $\quad$ **end if**
15: $\quad j \leftarrow j + 1$
16: **end while**

Schritt 2 des Algorithmus wird die Berechnung der Ableitungen $\frac{\mathrm{d}d}{\mathrm{d}q}(x, q_0)$, $\frac{\mathrm{d}y}{\mathrm{d}q}(x, q_0)$ und $\frac{\mathrm{d}z}{\mathrm{d}q}(x, q_0)$ gefordert. Da die Zulässigkeitskorrektur zur Verbesserung hochdimensionaler nichtlinearer Optimierung verwendet werden soll, darf nicht außer Acht gelassen werden, dass diese Sensitivitätsableitungen in praktischen Anwendungen potentiell sehr große und dichtbesetzte Matrizen sein können. Deshalb wird gezeigt, dass es mit Hilfe geeigneter Überlegungen möglich ist die Korrekturterme für d, y und z direkt zu berechnen, ohne die Sensitivitätsableitungen explizit abspeichern zu

müssen. Es sei der Korrekturterm als

$$\kappa(q^{[j]}) := \begin{pmatrix} \frac{\mathrm{d}d}{\mathrm{d}q}(x, q_0) \\ \frac{\mathrm{d}y}{\mathrm{d}q}(x, q_0) \\ \frac{\mathrm{d}z}{\mathrm{d}q}(x, q_0) \end{pmatrix} q^{[j]}$$

definiert. Weiterhin sei zur Vereinfachung der folgenden Argumentationen die Sensitivitätsmatrix $\hat{\mathcal{S}} \in \mathbb{R}^{(N_x+N_h+N_g)\times(N_h+N_g)}$ wie folgt definiert:

$$\hat{\mathcal{S}}(x) := \left(\frac{\mathrm{d}d}{\mathrm{d}q}(x, q_0)^\top, \frac{\mathrm{d}y}{\mathrm{d}q}(x, q_0)^\top, \frac{\mathrm{d}z}{\mathrm{d}q}(x, q_0)^\top\right)^\top$$

Aus der obigen Definition des Korrekturterms folgt somit

$$\kappa(q^{[j]}) = \hat{\mathcal{S}}(x) q^{[j]}.$$

Unter Berücksichtigung der effizienten Behandlung der Boxschranken können mit Hilfe von Gleichung (5.8) die benötigten Sensitivitätsableitungen bestimmt werden. Die Systemmatrix sei nach Symmetrisierung vereinfachend durch

$$K_{QP}(d^*, p_0; x) := \begin{pmatrix} \bar{H} & \nabla_x \tilde{h}(d^*, p_0)^\top & \nabla_x \tilde{g}(d^*, p_0)^\top \\ \nabla_x \tilde{h}(d^*, p_0) & 0 & 0 \\ Z^* \nabla_x \tilde{g}(d^*, p_0) & 0 & \Gamma^*(d^*, p_0) \end{pmatrix} \tag{5.18}$$

definiert. Diese ist im Sinne des SQP-Algorithmus abhängig vom Ausgangspunkt x des quadratischen Unterproblems, ändert sich aber während der Anwendung der Zulässigkeitskorrektur nicht. Zur Berechnung der Sensitivitätsableitungen unter Verwendung der speziellen Störung q aus Abschnitt 4.2.3 gilt mit diesen Definitionen

$$-K_{QP}(d^*, p_0; x)\hat{\mathcal{S}}(x) = \begin{pmatrix} 0_{N_x\times(N_h+N_g)} \\ I_{(N_h+N_g)\times(N_h+N_g)} \end{pmatrix}. \tag{5.19}$$

Durch Multiplikation von Gleichung (5.19) mit $q^{[j]}$ ergibt sich die Berechnungsvorschrift

$$-K_{QP}(d^*, p_0; x)\kappa(q^{[j]}) = \begin{pmatrix} 0_{N_x} \\ q^{[j]} \end{pmatrix} \tag{5.20}$$

zur Bestimmung des Korrekturterms. Insbesondere wurde die Matrix $K_{QP}(d^*, p_0; x)$ bereits zur Lösung des quadratischen Unterproblems faktorisiert und somit muss zur Bestimmung eines Korrekturterms lediglich das Vorwärts- und Rückwärtseinsetzen durchgeführt werden. Weiterhin muss die Sensitivitätsmatrix $\hat{\mathcal{S}}(x)$ nicht explizit abgespeichert werden. Somit kann die Zulässigkeitskorrektur auf beliebige Problemgrößen angewendet werden, insofern der Löser in der Lage war das quadratische Unterproblem zu lösen. Es müssen keine zusätzlichen Anforderungen an die Schwachbesetztheit der Ableitungsmatrizen oder Ähnliches gestellt werden.

Eine Aussage über die Konvergenzgeschwindigkeit des Korrekturalgorithmus innerhalb des SQP-Verfahrens kann durch Anwendung des Satzes 4.14 getroffen werden, wobei kurz an die Wahl der Störungen $q_{[j]}$ und die Berücksichtigung der aktiven Menge der Lösung des quadratischen Unterproblems, in Bemerkung 5.1 erinnert sei. Weiterhin ist die Voraussetzung, dass die zu korrigierende Initialstörung $q_{[0]} \in U_{\epsilon_q}(q_0)$ innerhalb des Gültigkeitsbereich des Sensitivitätssatzes liegt, kritisch zu hinterfragen und kann insbesondere a priori nicht sichergestellt werden. Speziell der Radius $\epsilon_q > 0$ ist nicht bekannt.
Die numerische Anwendung des Algorithmus kann trotzdem durchgeführt werden. Allerdings kann die allgemeingültige Konvergenz nicht garantiert werden und in der Umsetzung muss deshalb der Fortschritt des Algorithmus kritisch begutachtet werden.
Ausgehend von der Berechnungsvorschrift aus Gleichung (5.12) ergibt sich aus der Argumentationskette

$$\begin{aligned}
\hat{x}_{[j+1]} &= \hat{x}_{[j]} + \alpha\left(\frac{\mathrm{d}d}{\mathrm{d}q}(x, q_0)q_{[j]}\right) \\
&\overbrace{=}^{(5.20)} \hat{x}_{[j]} - \alpha\left(K_{QP}^{-1}(d^*, p_0; x)\begin{pmatrix} 0_{N_x} \\ q^{[j]} \end{pmatrix}\right) \\
&\overbrace{=}^{(5.17)} \hat{x}_{[j]} - \alpha\left(K_{QP}^{-1}(d^*, p_0; x)\begin{pmatrix} 0_{N_x} \\ h(\hat{x}_{[j]}) \\ g^a(\hat{x}_{[j]}) \end{pmatrix}\right)
\end{aligned} \tag{5.21}$$

ein interessanter Zusammenhang. Die Vorschrift (5.21) ähnelt einem vereinfachten Newton-Verfahren, da während der Iterationen des Korrekturalgorithmus die Matrix $K_{QP}(d^*, p_0; x)$ nicht aktualisiert wird.
Für die folgenden Überlegungen sei angenommen, dass $N_g = 0$ und $N_h > 0$ ist. Ein Blick auf Gleichung (3.68) offenbart den Zusammenhang zu den Hauptiterationen des SQP-Verfahrens. Das vorgestellte Verfahren arbeitet im lediglich gleichungsbeschränkten Fall analog zum vereinfachten Newtonverfahren zum Auffinden einer Nullstelle der Funktion $\Phi(x, \mu)$ aus Gleichung (3.65), mit der Einschränkung, dass die Iterierten lediglich auf die Verbesserung der Nebenbedingungen abzielen.
Büskens [12] schlug vor neben der Zulässigkeit auch die Verletzung der Optimalitätsbedingung $\nabla_x L(x, \mu) = 0$ in der Korrektur zu berücksichtigen. Mit Hilfe der speziellen Störung r aus Abschnitt 4.2.1 könnte dies auch im Rahmen des vorgeschlagenen Algorithmus durchgeführt werden. Den Überlegungen zur effizienten Berechnung des Korrekturterms folgend, ergibt sich mit $r^{[j]} = \nabla_x L(\hat{x}_{[j]}, \mu)$ in diesem Fall analog zu Gleichung (5.21)

$$\hat{x}_{[j+1]} = \hat{x}_{[j]} - \alpha\left(K_{QP}^{-1}(d^*, p_0; x)\begin{pmatrix} r^{[j]} \\ q^{[j]} \end{pmatrix}\right) \tag{5.22}$$

$$= \hat{x}_{[j]} - \alpha\left(K_{QP}^{-1}(d^*, p_0; x)\begin{pmatrix} \nabla_x L(\hat{x}_{[j]}, \hat{\mu}_{[j]}) \\ h(\hat{x}_{[j]}) \end{pmatrix}\right). \tag{5.23}$$

Dies entspricht dem gedämpften vereinfachten Newton-Verfahren zur Nullstellensuche für die Funktion $\Phi(x, \mu)$.
In der Literatur finden sich ebenfalls einige Ansätze die auf ähnliche Korrekturen, wie das hier vorgestellte Verfahren führen. Fletcher [35] greift das von Coleman und Conn [20, 21] beschriebene Zwei-Schritt-Verfahren erneut auf. Die Grundidee des Verfahrens ist es ausgehend von einer Iterierten x einen ersten sogenannten horizontalen Schritt $\tilde{d}$ zu berechnen, dieser wird ähnlich zum Hauptschritt des SQP-Verfahrens bestimmt. Anschließend werden die Nebenbedingungen im resultierenden Punkt $\tilde{x} = x + \tilde{d}$ ausgewertet und es wird ein Gleichungssystem analog zu Gleichung (5.20) gelöst, um einen zweiten Schritt $\bar{d}$ zu bestimmen. Daraufhin wird im Punkt $\bar{x} = \tilde{x} + \bar{d} = x + \tilde{d} + \bar{d}$ wieder der horizontale Schritt berechnet. Es handelt sich in diesem Fall um eine Art Predictor-Corrector-Verfahren. Sargent [89] konstruiert in diesem Zusammenhang ein iteratives Projektionsverfahren, dass ohne die hier beschriebene Sensitivitätsmotivation auskommt, aber letztendlich auf ein ähnliches Verfahren führt.
Beispielsweise im Buch von Nocedal und Wright [82] oder in der Arbeit von Fletcher [34] werden Korrekturen zweiter Ordnung (*second order corrections*) beschrieben. Diese Technik wurde entwickelt, um Problemen, wie dem Maratos-Effekt [76], welche durch Linearisierung der Nebenbedingungen, insbesondere in der Nähe eines lokalen Minimums, auftreten, entgegen zu wirken. Es sei die Iterierte x gegeben und die Suchrichtung $\tilde{d}$ bestimmt. Wird während der Liniensuche die Schrittweite $\alpha = 1$ abgelehnt, so wird zunächst α nicht verkleinert. Stattdessen werden die Nebenbedingungen im Punkt $\tilde{x} = x + \tilde{d}$ ausgewertet und es wird auch hier ein zweiter Schritt $\bar{d}$ ähnlich zu Gleichung (5.20) berechnet. Wächter und Biegler [105] beschreiben verschiedene Möglichkeiten den Schritt $\bar{d}$ zu bestimmen.
Die auf der parametrischen Sensitivitätsanalyse gestützten Herleitungen in diesem Abschnitt stellen eine alternative Herleitung für das erwähnte Zwei-Schritt-Verfahren und für die *second order corrections* dar.
Die Motivation des vorgestellten Verfahrens basiert auf der Anwendung der Sensitivitätsanalyse auf das quadratische Unterproblem. Werden an Stelle der analytischen Hesse-Matrix $\nabla^2_{xx} L(x, \lambda, \mu)$, oder einer Näherung durch finite Differenzen, alternative Approximationen wie zum Beispiel BFGS-Varianten aus Abschnitt 3.3.2 verwendet, behalten die bisherigen Überlegungen somit ihre Gültigkeit.
Nach der allgemeinen Beschreibung des Algorithmus wird im Folgenden kurz eine Beschleunigungsstrategie vorgestellt. Anschließend werden die zur Zeit noch offenen Abbruchkriterien innerhalb des Verfahrens ausführlich diskutiert. Zusätzlich werden Bedingungen vorgestellt, die darauf abzielen den Algorithmus nur dann in das SQP-Verfahren einzubringen, wenn eine Effizienzsteigerung zu erwarten ist.

5.2.2 Steffensen Extrapolation

Zur Beschleunigung und Erhöhung der Stabilität der Zulässigkeitskorrektur schlägt Büskens [12] die Verwendung der Extrapolationsmethode von Steffensen vor. Diese

kann auch bei der Anwendung innerhalb des SQP-Verfahrens eingesetzt werden.
Es sei eine skalarwertige Folge $\xi^{[n]}$ gegeben. Die Methode von Steffensen kann durch die Gleichung

$$\xi^{[n+3]} := \xi^{[n]} - \left(\xi^{[n+1]} - \xi^{[n]}\right)^2 \left(\xi^{[n+2]} - 2\xi^{[n+1]} + \xi^{[n]}\right)^{-1} \tag{5.24}$$

beschrieben werden. Die Extrapolationsvorschrift (5.24) kann durch einfache Anwendung einer eindimensionalen Taylor-Reihen-Entwicklung unter der Annahme linearer Konvergenz hergeleitet werden. Allerdings existiert deshalb nur ein Beweis für skalarwertige Folgen. In numerischen Betrachtungen zeigt sich aber, dass eine komponentenweise Anwendung auf vektorwertige Folgen ebenfalls beschleunigende und stabilisierende Wirkung hat.
Bei der numerischen Umsetzung der Steffensen-Extrapolation muss die Größe des Nenners

$$\xi^{[n+2]} - 2\xi^{[n+1]} + \xi^{[n]} \tag{5.25}$$

beachtet werden. Insbesondere bei der Anwendung für hochdimensionale vektorwertige Folgen wie sie häufig im Rahmen der Optimierung auftreten, muss berücksichtigt werden, dass für einzelne Komponenten des Vektors der Nenner beliebig klein werden kann. Dieser Umstand ist aber nicht weiter problematisch. Gleichung (5.25) macht deutlich, dass betragsmäßig sehr kleine Werte lediglich zeigen, dass die Folge in dieser Komponente des Vektors bereits konvergiert ist, da

$$\xi^{[n+2]} \approx \xi^{[n+1]} \approx \xi^{[n]}$$

gilt. Somit kann in diesem Fall der Wert der Folge in dieser Komponente des Vektors einfach festgehalten werden.
Die Steffensen-Extrapolation setzt lineare Konvergenzgeschwindigkeit der zu extrapolierenden Folge voraus. Satz 4.14 zeigt, dass die Zulässigkeitskorrektur für hinreichend kleine Störungen mit linearer Geschwindigkeit konvergiert, deshalb kann für diese die Steffensen-Extrapolation genutzt werden.

5.2.3 Startkriterien

Die bisher diskutierten Hintergründe helfen nicht zu entscheiden in welchen Situationen es sinnvoll ist den vorgeschlagenen Algorithmus innerhalb des SQP-Verfahrens durchzuführen, da eine theoretisch sichergestellte Konvergenz lediglich in der Nähe des lokalen Minimums durch die Aussage von Satz 4.14 zu erwarten ist. Stattdessen muss im Einzelfall entschieden werden, wie mit der vorgeschlagenen Korrektur umzugehen ist.
Eine Möglichkeit a priori zu entscheiden, ob ein berechneter Korrekturschritt sinnvoll ist, kann unabhängig von der in der Liniensuche verwendeten Methode durch

Verwendung einer Bewertungsfunktion realisiert werden. Zu diesem Zweck wird die Bewertungsfunktion

$$\tilde{l}_2(x;\eta) := f(x) + \eta \left(\sum_{i=1}^{N_h} h_i(x)^2 + \sum_{i=1}^{N_g} (\max\{0, g_i(x)\})^2 \right) \tag{5.26}$$

in Anlehnung an Gleichung 3.75 verwendet. Die Zulässigkeitskorrektur zielt zunächst lediglich auf die Verbesserung der Nebenbedingungen ab und die Zielfunktion wird nicht direkt berücksichtigt. Mit Hilfe der Bewertungsfunktion wird eine Kopplung beider Kriterien umgesetzt. Die stückweise gültigen Ableitungen der Bewertungsfunktion sind durch

$$\frac{\partial \tilde{l}_2}{\partial x_j}(x;\eta) = \frac{\partial f}{\partial x_j} + \eta \left(\sum_{i=1}^{N_h} 2h_i(x) \frac{\partial h_i}{\partial x_j}(x) + \sum_{i=1}^{N_g} k_{i,j}(x) \right), \quad j = 1, ..., N_x \tag{5.27}$$

gegeben, wobei die Hilfsfunktionen $k_{i,j}(x)$ als

$$k_{i,j}(x) = \begin{cases} 0 & \text{falls } \; g_i(x) \leq 0 \\ 2g_i(x) \frac{\partial g_i}{\partial x_j}(x) & \text{sonst} \end{cases} \tag{5.28}$$

definiert sind. Es liege die in Algorithmus 10 berechnete Störung $q_{[m]}$ in Iteration m vor. Nach dem Satz von Taylor gibt das Skalarprodukt

$$\left(\nabla_x \tilde{l}_2(x;\eta) \right)^\top \left(\frac{\mathrm{d}d}{\mathrm{d}q}(x, q_0) q_{[m]} \right) =: w_{[m]} \tag{5.29}$$

in linearer Näherung Auskunft über das Verhalten der Funktion $\tilde{l}_2$ entlang der Richtung $\left(\frac{\mathrm{d}d}{\mathrm{d}q}(x, q_0) q_{[j]} \right)$. Somit kann vorausgesagt werden, welche Auswirkungen der vom Algorithmus vorgeschlagene Schritt auf die betrachtete Bewertungsfunktion haben wird. Entsprechend wird der Schritt nur durchgeführt, wenn $w_{[m]} < 0$ gegeben ist.

Es ist allerdings anzumerken, dass zur Berechnung des Skalarproduktes die Auswertung des Gradienten der Zielfunktion und der Jacobi-Matrix der Nebenbedingungen notwendig ist. Deshalb muss in der Praxis abgewogen werden, ob der entstehende Nutzen dem zusätzlichen Aufwand bei der Auswertung der Ableitungen überwiegt. Bei Ablehnung der Korrektur ist zu beachten, dass eine erneute Bestimmung der Ableitungswerte zur Aufstellung des nächsten quadratischen Unterproblems nicht nötig ist.

Verschiedene Umsetzungen des Kriteriums sind denkbar. Es könnte beispielsweise lediglich für den ersten Schritt mit der Störung $q_{[0]}$ geprüft werden, alternativ kann in jedem Schritt die voraussichtliche Auswirkung bei Korrektur mit der Störung $q_{[j]}$ bewertet werden.

Neben der Beurteilung ob ein bestimmter Schritt nach seiner Berechnung durchgeführt werden soll, stellt sich noch die Frage, wann dieser überhaupt berechnet

werden soll.
Offensichtlich muss die Zulässigkeitskorrektur nur aktiviert werden, wenn die aktuelle Iterierte noch nicht zulässig ist. Weiterhin wurde in den Abschnitten 3.3.3.1 und 3.3.3.3 die Regularisierung der Hesse-Matrix und die Relaxierung der Nebenbedingungen innerhalb von WORHP beschrieben. Beide Heuristiken werden verwendet, um die globale Konvergenz des Lösers zu verbessern. Es sei eine Toleranz $\delta_{\text{reg}} > 0$ vorgegeben. Die Zulässigkeitskorrektur wird nach der erfolgreichen Lösung eines quadratischen Unterproblems nur dann aktiviert wenn die Bedingung

$$\tau\left(\max(0, 1 - \sigma_{min}^{[k]}) \right) \leq \delta_{\text{reg}} \tag{5.30}$$

erfüllt ist. Eine Verletzung dieser Bedingung deutet an, dass die aktuelle Iterierte sehr weit vom Attraktionsbereich eines lokalen Minimums entfernt ist und die Hesse-Matrix auf Grund der Regularisierung stark verändert ist. In dieser Situation ist der zu erwartende Nutzen der Zulässigkeitskorrektur nur gering, so dass auf die Berechnung komplett verzichtet werden sollte.

5.2.4 Abbruchbedingungen

Zu diesem Zeitpunkt müssen die im Algorithmus 10 erwähnten Akzeptanz- und Ablehnungskriterium zur Bewertung der erfolgten Schritte noch spezifiziert werden. Die folgenden Überlegungen basieren darauf, dass die Gültigkeit der Sensitivitätsableitungen auf Grund der Voraussetzungen des Satzes über implizite Funktionen 4.2 nur innerhalb der dort definierten Umgebungen gegeben sind. Weiterhin sind auch die Approximationen (4.24), (4.25) und (4.26) nur lineare Näherungen und haben somit potenziell nur sehr lokale Gültigkeit. Deshalb muss bei der Anwendung der Zulässigkeitskorrektur der Fortschritt jeder Iteration kritisch bewertet werden.
In Schritt 9 des Algorithmus 10 werden die Nebenbedingungen des Ausgangsproblem ausgewertet und diese Werte zur Beurteilung der Qualität eines Schritts verwendet. Auf Grund der lokalen Gültigkeit der Sensitivitäten und auf Grund der Nichtlinearität der Nebenbedingungen ist es möglich, dass der Schritt die Werte dieser sogar verschlechtert. Dies muss durch ein geeignetes Ablehnungskriterium geprüft und die Iterierte gegebenenfalls zurückgesetzt werden. Im Gegensatz dazu wird das Akzeptanzkriterium verwendet, um nach einem erfolgreichen Schritt einschätzen zu können, ob weitere Schritte eine zusätzliche Verbesserung der Nebenbedingungen erzielen können oder ob es sinnvoller ist mit der nächsten Hauptiteration fortzufahren und die Zulässigkeitskorrektur zu beenden.
Die folgenden Abschnitten diskutieren verschiedene Akzeptanz- und Ablehnungskriterien.

5.2.4.1 Kontraktionsfaktor

Auf Grund der zu erwartenden linearen Konvergenzordnung des Verfahrens kann der Kontraktionsfaktor der Iterationsvorschrift über die Gleichung

$$c_{[j]} = \frac{\left\|x^{[j+1]} - x^{[j]}\right\|_\infty}{\left\|x^{[j]} - x^{[j-1]}\right\|_\infty} \tag{5.31}$$

näherungsweise bestimmt werden. Der Faktor $c_{[j]}$ dient zur Bewertung der vorliegenden Konvergenzgeschwindigkeit. Gilt $c_{[j]} \ll 1$ ist auch von den nächsten Iterationen ein guter Fortschritt zu erwarten. Ist $c_{[j]} \leq 1$, aber gleichzeitig $c_{[j]} \approx 1$, so ist nur ein geringer Fortschritt zu erwarten. Im ungünstigsten Fall kann auch $c_{[j]} > 1$ gelten, so dass die Konvergenz nicht garantiert werden kann.
Um zu entscheiden, ob weitere Iterationen des Korrekturverfahrens durchgeführt werden sollen, kann der Faktor $c_{[j]}$ in Bezug zur aktuell vorliegenden Nebenbedingungsverletzung gesetzt werden. Es seien Schranken $\eta^c \in \mathbb{R}$ und $\eta^q \in \mathbb{R}$ gegeben. Innerhalb des Korrekturverfahrens ist die aktuelle Verletzung der Nebenbedingungen durch die Störung $q_{[j]}$ gegeben. Ein Abbruch des Verfahrens ist sinnvoll, wenn vermutlich keine Konvergenz vorliegt, also $c_{[j]} > 1$ gilt. Seien η^c und η^q geeignet gewählt, dann zeigt die Gleichung

$$1 > c_{[j]} > \eta^c \tag{5.32}$$

an, dass nur langsamer Fortschritt zu erwarten ist. Ist gleichzeitig die aktuelle Verletzung der Nebenbedingungen

$$q_{[j]} > \eta^q \tag{5.33}$$

noch groß, so stellen die Gleichungen (5.32) und (5.33) Abbruchbedingungen dar. Gilt

$$c_{[j]} < \eta^c \tag{5.34}$$

ist zu erwarten, dass sich die Zulässigkeit noch deutlich verbessert. Ist alternativ Gleichung (5.32) erfüllt und gilt gleichzeitig

$$q_{[j]} < \eta^q \tag{5.35}$$

ist es möglich, dass weitere Iterationen trotz des langsamen Fortschritts einen zulässigen Punkt finden. Deshalb sollte in diesen Fällen das Verfahren fortgesetzt werden. Die beiden folgenden Abschnitte stellen Entscheidungskriterien vor, welche die Auswirkungen auf Größen des Ausgangsproblem zur Bewertung zu Grunde legen. Das hier vorgestellte Kriterium ist mit diesen kombinierbar.

5.2.4.2 Nebenbedingungen als Entscheidungskriterium

Abschnitt 3.2.2 beschreibt verschiedene Techniken zur Schrittweitenbestimmung innerhalb des SQP-Verfahrens. Im Prinzip können bei der Beurteilung der Schritte innerhalb der Zulässigkeitskorrektur ähnliche Strategien zum Einsatz kommen. Allerdings ist eines der Ziele des neuen Algorithmus so wenig Rechenaufwand wie möglich zu generieren. Trotzdem soll Fortschritt für die Iterierten erreicht werden. Deshalb zielt der erste Ansatz zur Beurteilung der durchgeführten Schritte darauf ab lediglich die Nebenbedingungen im neuen Punkt auszuwerten und anschließend zu bewerten.
Die Verletzung der Nebenbedingungen wird vor und nach einem Korrekturschritt ermittelt und die Differenz mit einer wachsenden Schranke verglichen. Ist der erzielte Fortschritt kleiner als die gewählte Schranke, ist das Akzeptanzkriterium erfüllt. Dies bedeutet, dass der letzte Schritt nur noch eine mäßige Verbesserung geliefert hat. Die Zulässigkeitskorrektur wird beendet und der Hauptalgorithmus fortgesetzt. Die Schranke wächst mit der Zeit, damit nach einigen Iterationen nur noch weitere Schritte durchgeführt werden, wenn der zu erwartende Fortschritt noch groß ist. Mit diesem Kriterium soll der Algorithmus in akzeptabler Zeit terminieren, aber gleichzeitig in der Lage sein guten Fortschritt zu erzielen. Das Entscheidungskriterium wird durch

$$0 < \left\| \left(h(\hat{x}_{[j]}), g^a(\hat{x}_{[j]}) \right) \right\|_\infty - \left\| \left(h(\hat{x}_{[j+1]}), g^a(\hat{x}_{[j+1]}) \right) \right\|_\infty \leq (j+1)^2 \epsilon_\infty \quad (5.36)$$

überprüft. Das Kriterium wird gleichzeitig verwendet um zu überprüfen, ob der Schritt auf Grund der verwendeten linearen Näherungen überhaupt in der Lage war die Nebenbedingungen zu verringern. Ist die Ungleichung (5.36) auf der linken Seite verletzt, so ist das Ablehnungskriterium erfüllt und der letzte Schritt wird verworfen. Alleinstehend kann diese Bedingung über die Ungleichung

$$\left\| \left(h(\hat{x}_{[j]}), g^a(\hat{x}_{[j]}) \right) \right\|_\infty - \left\| \left(h(\hat{x}_{[j+1]}), g^a(\hat{x}_{[j+1]}) \right) \right\|_\infty \leq 0 \quad (5.37)$$

beschrieben werden.
Abschließend sei noch angemerkt, dass die Nebenbedingungen in jedem Schritt zur Bestimmung der neuen Störung $q_{[j]}$ in jedem Fall ausgewertet werden, insofern werden durch dieses Entscheidungskriterium keine zusätzlichen Funktionsauswertungen generiert.

5.2.4.3 Lagrange-Funktion als Entscheidungskriterium

Zur Bestimmung der Schrittweite während des SQP-Algorithmus werden die beiden Zielkriterien der Optimalität und Zulässigkeit entsprechend der Konfiguration des Filters oder der Bewertungsfunktion gegeneinander abgewogen. In der numerischen Praxis hat es sich bewährt sowohl Optimalität, als auch Zulässigkeit während

des gesamten Optimierungsverlaufs in gleichem Maße zu behandeln. Durch die Anwendung der Zulässigkeitskorrektur kann dieses Gleichgewicht gestört werden. Auf Grund dieses Ungleichgewichtes kann der Fortschritt der Iterierten leiden, weil die Hauptiterationen darauf abzielen beide Größen gleichmäßig zu reduzieren. Es besteht die Gefahr, dass die Korrekturschritte und die Schritte des SQP-Verfahrens gegensätzlich arbeiten.
Chamberlain, Powell, Lemarechal und Pedersen [17] schlugen vor die Lagrange-Funktion (2.22) während der Optimierung als Bewertungsfunktion zu verwenden. Diese Idee lässt sich auf die Zulässigkeitskorrektur übertragen. Durch die Verwendung der Lagrange-Funktion als Entscheidungskriterium wird neben der Einhaltung der Nebenbedingungen auch die Entwicklung der Zielfunktion betrachtet. Durch die Berücksichtigung der Zielfunktion während der Zulässigkeitskorrektur soll verhindert werden, dass ein zu starkes Ungleichgewicht beider Zielkriterien entsteht. Das SQP-Verfahren sucht einen stationären Punkt der Lagrange-Funktion, somit erscheint diese zur Bewertung an dieser Stelle natürlich. Weiterhin entfällt die Wahl eines passenden Bestrafungsparameters für die Gewichtung der Nebenbedingungen im Gegensatz zu anderen Bewertungsfunktionen.
Für die Implementierung sei $\epsilon_{cv} = 10^{-5}$ als Toleranz gewählt. Das Akzeptanzkriterium wird durch die gleichzeitige Erfüllung der Bedingungen

$$L(\hat{x}_{[j+1]}, \hat{\lambda}_{[j+1]}, \hat{\mu}_{[j+1]}) - L(\hat{x}_{[j]}, \hat{\lambda}_{[j]}, \hat{\mu}_{[j]}) < 0 \tag{5.38}$$

und

$$\frac{\left\| \left(h(\hat{x}_{[j+1]}), g^a(\hat{x}_{[j+1]}) \right) \right\|_\infty - \left\| \left(h(\hat{x}_{[j]}), g^a(\hat{x}_{[j]}) \right) \right\|_\infty}{\left\| \left(h(\hat{x}_{[j]}), g^a(\hat{x}_{[j]}) \right) \right\|_\infty} < \epsilon_{cv} \tag{5.39}$$

beschrieben. Wenn die Gleichungen (5.38) und (5.39) beide gültig sind, so hat sich der Wert der Lagrange-Funktion verringert, während die relative Verbesserung der Nebenbedingungen nur klein ist. Deshalb wird die Iteration beendet und der Hauptalgorithmus fortgesetzt. Als Abbruchkriterium wird

$$L(\hat{x}_{[j+1]}, \hat{\lambda}_{[j+1]}, \hat{\mu}_{[j+1]}) - L(\hat{x}_{[j]}, \hat{\lambda}_{[j]}, \hat{\mu}_{[j]}) \geq 0 \tag{5.40}$$

überprüft. In diesem Fall hat sich der Wert der Lagrange-Funktion erhöht, dann wird die aktuelle Iterierte verworfen und der Hauptalgorithmus fortgesetzt.
Zur Auswertung der Lagrange-Funktion wird neben der in jedem Fall notwendigen Auswertung der Nebenbedingungen eine zusätzliche Auswertung der Zielfunktion in jedem Schritt der Zulässigkeitskorrektur benötigt.

5.3 Sensitivitätsanalyse der Regularisierung der Hesse-Matrix

In den Beschreibungen in Abschnitt 3.3.3.1 wurde die zentrale Rolle der Regularisierung der Hesse-Matrix für das SQP-Verfahren betont. Es wurde insbesondere diskutiert, dass die Effizienz des gesamten Verfahrens stark davon abhängt, ob schnell eine passende Größe der Regularisierung gefunden wird.
Die Regularisierung stellt eine quadratische Störung der Zielfunktion entsprechend der Definition in Abschnitt 4.2.2 dar. Einleitend wird kurz die Berechnung der Sensitivitäten in dieser speziellen Situation beschrieben. Anschließend werden Möglichkeiten dargestellt mit Hilfe der parametrischen Sensitivitätsanalyse Einfluss auf die Regularisierung zu nehmen. Weiterhin werden Strategien zur besseren Bestimmung der Größe der Regularisierung vorgestellt. Abschließend wird eine Methode zur Quantifizierung der Änderung der Suchrichtung auf Basis der Regularisierung diskutiert.

5.3.1 Berechnung der speziellen Sensitivitätsableitung

Es sei im Folgenden die Störung $\kappa \in \mathbb{R}$ des quadratischen Unterproblems (3.152) definiert, so dass

$$Q = \nabla^2_{xx} L(x^{[k]}, \lambda^{[k]}, \mu^{[k]}) + \kappa I_{N_x} \tag{5.41}$$

gilt.
Der Sensitivitätssatz 4.4 sagt aus, dass die optimale Lösung $d^{[k]}$ in einer Umgebung um die Nominalstörung $\kappa_0 \in \mathbb{R}$ stetig differenzierbar von κ abhängt. Diese Abhängigkeit sei durch die Funktion $d(\kappa)$ beschrieben und die Nominalstörung in Hauptiteration k durch $\kappa_0 := \tau^{[k]}\left(\max(0, 1-\sigma^{[k]}_{min})\right)$ definiert. Mit Hilfe der Taylor-Approximation erster Ordnung

$$d(\kappa) \approx \tilde{d}(\kappa) = d(\kappa_0) + \frac{\mathrm{d}d}{\mathrm{d}\kappa}(\kappa_0)\left(\kappa - \kappa_0\right) \tag{5.42}$$

kann die Funktion $d(\kappa)$ lokal angenähert werden.
Die Störung κ ist eine quadratische Störung der Zielfunktion im Sinne des Abschnitts 4.2.2. Es ist zu beachten, dass die Optimierungsvariablen im quadratischen Unterproblem von WORHP durch (d, δ) gegeben sind. Somit kann die Sensitivitätsableitung $\frac{\mathrm{d}d}{\mathrm{d}\kappa}(\kappa_0)$ Gleichung (4.47) folgend, durch Lösung des linearen Gleichungssystems (4.23) mit rechter Seite

$$\nabla^2_{d\kappa} L[p_0] = d \tag{5.43}$$
$$\nabla^2_{\delta\kappa} L[p_0] = 0 \tag{5.44}$$
$$\nabla_{\kappa} g[p_0] = 0 \tag{5.45}$$
$$\nabla_{\kappa} h[p_0] = 0 \tag{5.46}$$

bestimmt werden.

5.3.2 Anpassung des Regularisierungsparameters

Die Einbindung des Regularisierungsalgorithmus 5 in das SQP-Verfahren zeigt, dass die Bestimmung eines guten Wertes für die erste Modifikation der Hesse-Matrix entscheidend für die Effizienz des Algorithmus ist. Deshalb werden im Folgenden alternative Heuristiken vorgestellt, um mit Hilfe der parametrischen Sensitivitätsableitungen die Bestimmung eines initialen Regularisierungswertes für die nächste Hauptiteration durchzuführen und die Verwendung der Gerschgorin-Kreise zu ersetzen.
Die Sensitivitätsanalyse kann genutzt werden um die Auswirkungen der Regularisierung auf das quadratische Unterproblem zu analysieren. Nachdem das Unterproblem erfolgreich gelöst wurde, kann mit Hilfe der Bedingung

$$\left(d^{[k]}\right)^{\top} Q \left(d^{[k]}\right) > 0 \tag{5.47}$$

die Krümmung der Hesse-Matrix entlang der Suchrichtung als Maß für die positive Definitheit der Matrix Q beurteilt werden. Die folgenden Betrachtungen gehen davon aus, dass entlang der Suchrichtung $d^{[k]}$ positive Krümmung vorliegt, das heißt Gleichung (5.47) ist erfüllt. Es sei an dieser Stelle explizit angemerkt, dass diese nicht zwangsläufig erfüllt sein muss, da $d^{[k]}$ nicht im Kern der aktiven Nebenbedingungen liegt. Auf Grund der vorher bestimmten Regularisierung auf Basis der Gerschgorin-Kreise wird die Krümmungsbedingung trotzdem, abhängig vom verwendeten Dämpfungsparameter τ in Gleichung (3.123), in den meisten Fällen erfüllt sein.
Es kann weiterhin die genäherte Krümmungsbedingung

$$p(\kappa) := \tilde{d}(\kappa)^{\top} \left(\nabla^2_{xx} L(x^{[k]}, \lambda^{[k]}, \mu^{[k]}) + \kappa I_{N_x} \right) \tilde{d}(\kappa) > 0 \tag{5.48}$$

definiert werden. Gleichheit in dieser Bedingung liefert eine Approximation an den kritischen Regularisierungswert κ^* der zum Verlust der positiven Definitheit entlang der genäherten Suchrichtung führen würde. Eine direkte Anwendung des Newtonverfahrens zur Nullstellensuche auf Gleichung (5.48) ist im hochdimensionalen Fall nicht ratsam, da die Auswertung der Matrix- und Vektoroperationen je nach Grad der Schwachbesetztheit der auftretenden Größen numerisch aufwendig sein können. Die folgenden Betrachtungen zeigen einen Weg auf, der derartige Rechenoperationen

nur zur Vorbereitung vor dem Newtonverfahren benötigt. Die Definitionen

$$\begin{aligned}
\hat{\kappa} &:= \kappa - \kappa_0 \\
\hat{\alpha} &:= \frac{\mathrm{d}d}{\mathrm{d}\kappa}(\kappa_0) \\
\hat{\beta} &:= \left(\nabla^2_{xx} L(x^{[k]}, \lambda^{[k]}, \mu^{[k]}) + \kappa_0 I\right)\hat{\alpha} \\
c_0 &:= (d^{[k]})^\top \left(\nabla^2_{xx} L(x^{[k]}, \lambda^{[k]}, \mu^{[k]}) + \kappa_0 I\right)(d^{[k]}) \\
c_1 &:= 2(d^{[k]})^\top \hat{\beta} + \|d^{[k]}\|_2^2 \\
c_2 &:= \hat{\alpha}^\top \hat{\beta} + 2\hat{\alpha}^\top (d^{[k]}) \\
c_3 &:= \|\hat{\alpha}\|_2^2
\end{aligned}$$

verbessern die Lesbarkeit der folgenden Argumentationsfolge. Ausgehend von der Nullstellensuche für die genäherte Krümmungsbedingung (5.48) ergeben sich die folgenden Beziehungen:

$$\begin{aligned}
0 &= p(\kappa) \\
&= \tilde{d}(\kappa)^\top \left(\nabla^2_{xx} L(x^{[k]}, \lambda^{[k]}, \mu^{[k]}) + \kappa I_{N_x}\right)\tilde{d}(\kappa) \\
&= (d^{[k]} + \hat{\alpha}\hat{\kappa})^\top \left(\nabla^2_{xx} L(x^{[k]}, \lambda^{[k]}, \mu^{[k]}) + (\kappa_0 + \hat{\kappa})I\right)(d^{[k]} + \hat{\alpha}\hat{\kappa}) \\
&= (d^{[k]} + \hat{\alpha}\hat{\kappa})^\top \left(\nabla^2_{xx} L(x^{[k]}, \lambda^{[k]}, \mu^{[k]}) + \kappa_0 I\right)(d^{[k]} + \hat{\alpha}\hat{\kappa}) \\
&\quad + (d^{[k]} + \hat{\alpha}\hat{\kappa})^\top \left(\hat{\kappa} I\right)(d^{[k]} + \hat{\alpha}\hat{\kappa}) \\
&= (d^{[k]})^\top \left(\nabla^2_{xx} L(x^{[k]}, \lambda^{[k]}, \mu^{[k]}) + \kappa_0 I\right)(d^{[k]}) \\
&\quad + 2\hat{\kappa}(d^{[k]})^\top \left(\nabla^2_{xx} L(x^{[k]}, \lambda^{[k]}, \mu^{[k]}) + \kappa_0 I\right)\hat{\alpha} \\
&\quad + \hat{\kappa}^2(\hat{\alpha})^\top \left(\nabla^2_{xx} L(x^{[k]}, \lambda^{[k]}, \mu^{[k]}) + \kappa_0 I\right)(\hat{\alpha}) \\
&\quad + \hat{\kappa}(d^{[k]})^\top (d^{[k]}) + 2\hat{\kappa}^2 \hat{\alpha}^\top (d^{[k]}) + \hat{\kappa}^3 (\hat{\alpha})^\top (\hat{\alpha}) \\
&= c_0 + c_1\hat{\kappa} + c_2\hat{\kappa}^2 + c_3\hat{\kappa}^3 = \hat{p}(\hat{\kappa}) rz
\end{aligned}$$

Insgesamt reduziert sich die Nullstellensuche der genäherten Krümmungsbedingung somit auf die Suche einer Nullstelle des Polynoms

$$\hat{p}(\hat{\kappa}) = c_0 + c_1\hat{\kappa} + c_2\hat{\kappa}^2 + c_3\hat{\kappa}^3 \tag{5.49}$$

vom Grad drei. Eine Nullstelle des Näherungspolynoms $\hat{p}(\hat{\kappa})$ liefert den kritischen Wert $\hat{\kappa}^*$. Da nur positive Regularisierungswerte sinnvoll sind und eine Reduzierung

der Regularisierung erzielt werden soll, kann die Nullstellensuche auf das Intervall $\hat{\kappa} \in [-\kappa_0, 0]$ beschränkt werden. Wurde ein passendes $\hat{\kappa}^* = \kappa^* - \kappa_0$ gefunden, handelt es sich entsprechend der obigen Betrachtungen lediglich um eine Näherung an den kritischen Regularisierungswert. Bereits die Betrachtungen von Nikolayzik [81] hatten gezeigt, dass der Versuch die Regularisierung mit Hilfe von Sensitivitäten direkt auf null zu reduzieren in der numerischen Praxis zu optimistisch ist. Deshalb soll ein guter Wert für die initiale Regularisierung in der nächsten Hauptiteration im Intervall $[\kappa^*, \kappa_0]$ mit Hilfe des Parameters $\theta_\kappa \in [0, 1]$ über die Beziehung

$$\kappa_0^{[k+1]} = \kappa_0 + \theta_\kappa \hat{\kappa}^* = \kappa_0 + \theta_\kappa(\kappa^* - \kappa_0) \tag{5.50}$$

gefunden werden. Dieser ersetzt für den folgenden Iterationsschritt die Abschätzung über die Gerschgorin-Kreise.
Die Regularisierung muss danach bei Misserfolg des QP-Algorithmus wieder analog zur bisherigen Strategie angepasst werden. Die gesamte Vorgehensweise aus Sicht der folgenden Iteration ist in Algorithmus 11 dargestellt. Bei einer Implementie-

Algorithmus 11 Regularisierung der Hesse-Matrix: Sensitivitätsansatz

QP in Iteration k erfolgreich gelöst.
Suche Nullstelle $\hat{\kappa}^*$ für das Polynom (5.49)
if $\hat{\kappa}^* \in [-\kappa_0, 0]$ **then**
    $\kappa_0^{[k+1]} \leftarrow \kappa_0 + \theta_\kappa(\kappa^* - \kappa_0)$ (Vergleiche Gleichung (5.50))
else
    $\kappa_0^{[k+1]}$ über Gerschgorin-Kreise wie in Algorithmus 5 festlegen
end if
Wähle $\tau_0 = 1.0$, $\rho_\tau = 2.0$, $\tau_{\max} = 1 \cdot 10^7$
Setze $j = 0$
repeat
    if $j > 0$ **then**
        $\tau_j \leftarrow \tau_{j-1}\rho_\tau$
    end if
    $Q \leftarrow \nabla^2_{xx} L(x^{[k+1]}, \lambda^{[k+1]}, \mu^{[k+1]}) + \tau_j \kappa_0^{[k+1]} I$
    Löse das quadratische Unterproblem (3.152)
    $j \leftarrow j + 1$
until QP erfolgreich gelöst oder $\tau_{[j]} > \tau_{\max}$

rung des Ansatzes muss beachtet werden, dass die Bestimmung von $\kappa_0^{[k+1]}$ erfolgen muss, solange die Zerlegung der KKT-Matrix aus Gleichung (4.23) noch verfügbar ist, um unnötigen Rechenaufwand zu vermeiden.
Durch die Verbesserung des initialen Regularisierungsparameters mit Hilfe von Algorithmus 11 soll die Anzahl der benötigten Versuche zur Lösung des QP reduziert werden, da idealerweise der Regularisierungsparameter seltener angepasst werden muss.

Bei der Herleitung des Näherungspolynoms $\hat{p}(\hat{\kappa})$ wurden verschiedene Vereinfachungen wie zum Beispiel die Linearisierung der Funktion $d(\kappa)$ vorgenommen. Deshalb ist es möglich, dass keine Nullstelle im Suchintervall $[-\kappa_0, 0]$ vorliegt und das Newton-Verfahren fehlschlägt. In dieser Situation kann neben dem bisherigen Ansatz basierend auf den Gerschgorin-Kreisen eine weitere Strategie zur Bestimmung eines neuen Regularisierungswertes $\kappa_0^{[k+1]}$ umgesetzt werden.
Im Abschnitt 4.1 wurde erläutert, dass mit Hilfe der parametrischen Sensitivätsanalyse neben den Auswirkungen auf die primalen und dualen Variablen auch die Veränderung der beteiligten Zugehörigkeitsfunktionen quantifiziert werden kann. Dieser Ansatz wird im Folgenden aufgegriffen, um eine Strategie zu entwickeln auf Basis der Sensitivitäten der Zielfunktion des quadratischen Unterproblems eine Vorhersage für einen geeigneten Regularisierungsparameter $\kappa_0^{[k+1]}$ für die folgende Hauptiteration des SQP-Verfahrens zu bestimmen. Der Zielfunktionswert des quadratischen Unterproblems in Iteration k sei abkürzend durch

$$f^{\mathrm{QP}} := \frac{1}{2}(d^{[k]})^\top(\nabla^2_{xx}L(x^{[k]}, \lambda^{[k]}, \mu^{[k]}) + \kappa_0 I)(d^{[k]}) + \nabla_x f(x^{[k]})^\top d^{[k]} \tag{5.51}$$

definiert. Mit Hilfe der Sensitivitätsableitungen der Zielfunktion bezüglich κ kann die Änderung im Rahmen der Gültigkeit des Sensitivitätssatzes vorhergesagt werden. Unter Verwendung der Sensitivitäten erster und zweiter Ordnung der Zielfunktion aus den Folgerungen 4.11 und 4.12 ergibt sich

$$\hat{f}^{\mathrm{QP}} \approx f^{\mathrm{QP}} + \frac{\mathrm{d}f}{\mathrm{d}\kappa}(\kappa_0)\Big(\kappa - \kappa_0\Big) + \frac{1}{2}\frac{\mathrm{d}^2 f}{\mathrm{d}\kappa^2}(\kappa_0)\Big(\kappa - \kappa_0\Big)^2 \tag{5.52}$$

als Näherung für den Zielfunktionswert bei Verwendung der Regularisierung κ. Es sei ein Parameter $\nu \in (0, 1)$ gewählt, welcher festlegt wie stark sich der Zielfunktionswert ändern darf. Unter dem Einfluss der Störung darf sich der approximative Wert $\hat{f}^{\mathrm{QP}}$ auf das $(1 \pm \nu)$-fache des Ausgangszielfunktionswertes ändern, so dass die Gleichung

$$\hat{f}^{\mathrm{QP}} = f^{\mathrm{QP}} + \frac{\mathrm{d}f}{\mathrm{d}\kappa}(\kappa_0)\Big(\kappa - \kappa_0\Big) + \frac{1}{2}\frac{\mathrm{d}^2 f}{\mathrm{d}\kappa^2}(\kappa_0)\Big(\kappa - \kappa_0\Big)^2 = (1 \pm \nu) f^{\mathrm{QP}} \tag{5.53}$$

angesetzt wird. Im hier vorliegenden Zusammenhang ergibt sich aus den Gleichungen (4.48) und (4.34) für die Sensitivitätsableitungen der Zielfunktion

$$\tilde{b} := \frac{\mathrm{d}f}{\mathrm{d}\kappa}(\kappa_0) = \frac{1}{2}(d^{[k]})^\top(d^{[k]}) = \frac{1}{2}\|d^{[k]}\|_2^2 \tag{5.54}$$

$$\tilde{c} := \frac{\mathrm{d}^2 f}{\mathrm{d}\kappa^2}(\kappa_0) = \left(\frac{\mathrm{d}d}{\mathrm{d}\kappa}(\kappa_0)\right)^\top d^{[k]}, \tag{5.55}$$

so dass zusammen mit den Definitionen

$$\tilde{a} := f^{\mathrm{QP}} - (1 \pm \nu) f^{\mathrm{QP}} = \pm \nu f^{\mathrm{QP}} \tag{5.56}$$

$$\xi := \Big(\kappa - \kappa_0\Big) \tag{5.57}$$

eine Nullstelle der quadratischen Gleichung

$$\tilde{a} + \tilde{b}\xi + \frac{\tilde{c}}{2}\xi^2 = 0 \tag{5.58}$$

gefunden werden muss. Es sei angemerkt, dass $\tilde{c} \neq 0$ angenommen werden kann. Ist dies nicht der Fall, so wurde keine Suchrichtung gefunden und es muss zunächst keine Anpassung der Regularisierung für die nächste Iteration erfolgen.
Die Lösung der quadratischen Gleichung ist bekanntermaßen durch

$$\xi_{1,2} = -\frac{\tilde{b}}{\tilde{c}} \pm \sqrt{\left(\frac{\tilde{b}}{\tilde{c}}\right)^2 - \frac{2\tilde{a}}{\tilde{c}}} \tag{5.59}$$

gegeben. Ersetzen der Platzhalter $\tilde{a}, \tilde{b}$ und $\tilde{c}$ ergibt vier Kandidaten für einen möglicherweise passenden Regularisierungswert

$$\kappa_{1,2,3,4} = \kappa_0 - \frac{\|d^{[k]}\|_2^2}{\left(2\frac{\mathrm{d}d}{\mathrm{d}\kappa}(\kappa_0)\right)^\top d^{[k]}} \pm \sqrt{\left(\frac{\|d^{[k]}\|_2^2}{\left(2\frac{\mathrm{d}d}{\mathrm{d}\kappa}(\kappa_0)\right)^\top d^{[k]}}\right)^2 \pm \frac{2\nu f^{\mathrm{QP}}}{\left(\frac{\mathrm{d}d}{\mathrm{d}\kappa}(\kappa_0)\right)^\top d^{[k]}}}. \tag{5.60}$$

Das Ziel der hier vorgeschlagenen Strategie ist eine neue Regularisierung für die nächste Iteration $\kappa_0^{[k+1]}$ zu finden. Im Verlauf der SQP-Iterationen sollte in der Regel die benötigte Regularisierung auf Grund der Annäherung an ein lokales Minimum abnehmen, weil davon ausgegangen werden kann, dass die hinreichenden Bedingungen zweiter Ordnung bereits in einer Umgebung um das Minimum erfüllt sind. Deshalb sollte ein Kandidat für die Regularisierung der nächsten Iteration sinnvollerweise im Intervall $[0, \kappa_0]$ liegen. Tritt der Fall ein, dass mehrere Kandidaten innerhalb des Intervalles liegen, so wird der kleinste Wert als neue Regularisierung festgelegt.
Die Vorgehensweise wird in den Algorithmus 11 eingebettet und kommt zum Einsatz falls die Strategie basierend auf der Approximation der Krümmungsbedingung fehlgeschlagen ist. In Algorithmus 12 ist diese Einbindung zusammenfassend dargestellt. Der Algorithmus ersetzt die Schritte 2-7 im Ausgangsalgorithmus 11. Nach wie vor bleibt die Abschätzung über die Gerschgorin-Kreise als alternative im Algorithmus, so dass sichergestellt wird, dass in jedem Fall eine neuer Wert für die Regularisierung in der folgenden Iteration zur Verfügung steht. Die vorgeschlagenen Strategien dienen als Erweiterung der bisherigen Regularisierung und zielen darauf ab kleinere Regularisierungswerte zu ermitteln, so dass die Störung der originalen Unterproblemformulierung in der nächsten Hauptiteration geringer ausfällt.

5.3.3 Korrektur der Suchrichtung

In seiner Dissertation fasst Nikolayzik [81] $\kappa_0 := \tau^{[k]}\left(\max(0, 1 - \sigma_{min}^{[k]})\right)$ ebenfalls als Nominalstörung des quadratischen Unterproblems auf und wendet die parame-

Algorithmus 12 Regularisierung der Hesse-Matrix: Sensitivitätsansatz Zusatz für Schritte 2-7

1: Suche Nullstelle $\hat{\kappa}^*$ für das Polynom (5.49)
2: **if** $\hat{\kappa}^* \in [-\kappa_0, 0]$ **then**
3: $\quad \kappa_0^{[k+1]} \leftarrow \kappa_0 + \theta_\kappa(\kappa^* - \kappa_0)$ (Vergleiche Gleichung (5.50))
4: **else**
5: $\quad$ Bestimme Kandidaten $\kappa_{1,2,3,4}$ nach Gleichung (5.60)
6: $\quad \bar{\kappa} \leftarrow \min_{\kappa>0} \kappa \in \left\{ \kappa_1, \kappa_2, \kappa_3, \kappa_4 \right\}$
7: $\quad$ **if** $\bar{\kappa} \in [0, \kappa_0]$ **then**
8: $\quad\quad \kappa_0^{[k+1]} \leftarrow \bar{\kappa}$
9: $\quad$ **else**
10: $\quad\quad \kappa_0^{[k+1]}$ über Gerschgorin-Kreise wie in Algorithmus 5 festlegen
11: $\quad$ **end if**
12: **end if**

trische Sensitivitätsanalyse auf das gestörte Problem (3.152) an. Die Störung κ_0 geht passend zu Gleichung (3.153) in die Problemstellung ein.
Nikolayzik verwendet die Sensitivitätsableitungen, um eine Verbesserung der Suchrichtung mit Hilfe der Taylor-Approximation

$$d(\kappa) \approx d^{[k]} + \frac{\mathrm{d}d}{\mathrm{d}\kappa}(\kappa_0)(\kappa - \kappa_0) \tag{5.61}$$

durchzuführen. Analog wendet er Approximationen für die dualen Variablen an. Das Ziel ist eine korrigierte Suchrichtung für $\kappa \approx 0$ zu finden und so die Störung des Unterproblems durch Regularisierung herauszurechnen. Für die korrigierte Suchrichtung wird anschließend die normale Liniensuche angewendet. Schlägt diese fehl, wird die Suchrichtung zurückgesetzt und der Hauptalgorithmus mit der unkorrigierten Suchrichtung fortgesetzt. Die Details seiner Implementierung finden sich in Algorithmus 4.2.3 in [81]. Die numerischen Ergebnisse bei der Verbesserung der Suchrichtung sind eher negativ. Nikolayzik zeigt, dass je geringer die Eingriffe der Methode sind, desto besser ist das Gesamtergebnis von WORHP auf den untersuchten Testmengen.
Die Korrektur offenbart einige Probleme dieses Ansatzes. Der SQP-Algorithmus reagiert sehr empfindlich auf Eingriffe die Suchrichtung betreffend, insbesondere im Zusammenhang mit der äußerst sensiblen Regularisierung der Hesse-Matrix. Es wurde bereits mehrfach die lokale Gültigkeit des Sensitivitätssatzes thematisiert, so dass die Annahme, die Regularisierungsstörung auf $\kappa \approx 0$ reduzieren zu können, sehr optimistisch erscheint. Nikolayzik versucht die beschriebenen Probleme durch Einführung einer maximal zu korrigierenden Nominalstörung zu begrenzen. In dieser Arbeit wurden neue Ansätze zur Bestimmung des Regularisierungsparameters vorgestellt. Diese sind weniger optimistisch als das Ziel die Störung direkt auf 0 zu reduzieren, so dass die Näherung (5.61) als alternative Suchrichtung berücksichtigt

werden kann.
Schlägt die Liniensuche mit der korrigierten Suchrichtung $d(\kappa_0^{[k+1]})$ fehl, muss anschließend eine Liniensuche mit der originalen Suchrichtung d durchgeführt werden. Diese Vorgehensweise birgt aber das Risiko, dass im schlimmsten Fall doppelt so viele Auswertungen der Zielfunktion und Nebenbedingungen benötigt werden. Deshalb liegt es nahe die Konfiguration der Liniensuche für die Untersuchung zweier Suchrichtungen anzupassen. In Gleichung (3.79) wurde die Verringerung des Armijo-Parameters α durch Multiplikation mit dem Faktor β_α^j beschrieben. Für die hier beschriebene Situation mit zwei Suchrichtungen sollte der Parameter β_α durch β_α^2 ersetzt werden. Auf diese Weise bleibt die maximale Anzahl Auswertungen identisch zur normalen Liniensuche mit nur einer Suchrichtung. Neben der Verwendung der korrigierten Suchrichtung innerhalb der Liniensuche kann diese auch als Startschätzung für das nächste zu lösende quadratische Unterproblem verwendet werden.

5.4 Sensitivitätsanalyse der Relaxierung der Nebenbedingungen

Die Definition des quadratischen Unterproblems innerhalb von WORHP nach Gleichung (3.152) verdeutlicht, dass neben der Regularisierung der Hesse-Matrix, die Relaxierung der Nebenbedingungen eine starke Veränderung gegenüber der originalen Problemformulierung aus Abschnitt 3.2.1 darstellt. Weiterhin haben die Betrachtungen im Abschnitt 3.3.3.3 bereits gezeigt, dass die Einbindung der Relaxierung inklusive der Wahl der Bestrafungsparameter $\eta_i,\ i = 1, ..., N_\delta$ einen großen Einfluss auf die resultierenden Lösungen der Unterprobleme hat.
Eine Postoptimalitätsanalyse der Auswirkung dieser Parameter wird im Folgenden dazu verwendet die Auswirkungen der Relaxierung näher zu untersuchen und Strategien zu entwickeln um die Einbindung der Relaxierung in den SQP-Algorithmus zu verbessern. Die grundsätzliche Idee zur hier beschriebenen Vorgehensweise stammt erneut aus der Arbeit von Nikolayzik [81]. Die dort vorgestellten Verwendungsmöglichkeiten der Sensitivitätsableitungen des quadratischen Unterproblems bezüglich des Bestrafungsvektors η werden hier aufgegriffen, kritisch hinterfragt und erweitert.
Zunächst wird kurz erläutert wie die beteiligten Sensitivitätsableitungen bestimmt werden können. Anschließend werden Strategien zur besseren Konfiguration der Bestrafungsparameter basierend auf den Sensitivitätsableitungen thematisiert. Der verbleibende Rest des Abschnitts diskutiert kurz die Auswirkung auf die resultierende Suchrichtung mit Hilfe der Postoptimalitätsanalyse.

5.4.1 Berechnung der speziellen Sensitivitätsableitung

Anhand der Definition des quadratischen Unterproblems (3.152) ist gut ersichtlich, dass die Berechnung parametrischer Sensitivitäten bezüglich eines oder mehrerer Bestrafungsparameter analog zur Berechnung der Sensitivitäten bezüglich der Regularisierung in Abschnitt 5.3.1 erfolgen kann. Es handelt sich auch hier um eine quadratische Störung der Zielfunktion passend zu den Folgerungen des Abschnitts 4.2.2. Wie bereits bei der Regularisierung, ist zu beachten, dass die Optimierungsvariable innerhalb des quadratischen Unterproblems durch (d, δ) gegeben ist. Somit können die Sensitivitätsableitungen $\frac{\mathrm{d}d}{\mathrm{d}\eta}(\eta_0)$ und $\frac{\mathrm{d}\delta}{\mathrm{d}\eta}(\eta_0)$ durch Lösung des linearen Gleichungssystems (4.23) mit der rechten Seite

$$\nabla^2_{d\eta} L[p_0] = 0 \tag{5.62}$$
$$\nabla^2_{\delta\eta} L[p_0] = \delta \tag{5.63}$$
$$\nabla_p g[p_0] = 0 \tag{5.64}$$
$$\nabla_p h[p_0] = 0 \tag{5.65}$$

bestimmt werden.

5.4.2 Anpassung des Bestrafungsparameters

Es sei zunächst daran erinnert, dass die Relaxierungsvariablen $\delta_i \in [0, 1], i = 1, ..., N_\delta$ innerhalb des Einheitsintervalles liegen. Im Abschnitt 3.3.3.3 wurde deutlich, dass es erstrebenswert ist, idealerweise $\delta_i \approx 0$ sicherzustellen. Auf diese Weise können möglichst gute Suchrichtungen gefunden werden. Für die folgenden Betrachtungen sei bis auf Weiteres $N_\delta = 1$ angenommen. Der Fall mit einer Relaxierungsvariablen für jede Nebenbedingung wird später gesondert diskutiert.
Die Beschreibung von Algorithmus 6 hat bereits verdeutlicht, dass für eine gute Einbindung der Relaxierung der Nebenbedingungen in ein SQP-Verfahren eine gute Wahl des Bestrafungsparameters η essentiell ist. Wird schnell ein guter Wert für η gefunden, so müssen nicht unnötig quadratische Unterprobleme gelöst werden, deren Lösung anschließend höchstens noch für einen Warmstart verwendet werden kann. Es sei j der Iterationszähler für die wiederholte Lösung der quadratischen Unterprobleme analog zu Algorithmus 6. In Iteration j wird der Bestrafungsparameter $\eta^{[j]}$ verwendet. Das Kriterium $\delta < \delta_{\max}$ soll sicherstellen, dass tatsächlich eine Suchrichtung gefunden wurde, vergleiche hierzu Beispiel 3.28. Die dortigen Beobachtungen motivieren die Verwendung der Sensitivitätsableitung $\frac{\mathrm{d}\delta}{\mathrm{d}\eta}(\eta^{[j]})$, damit beurteilt werden kann, wie sich eine Änderung von $\eta^{[j]}$ auf die resultierende Relaxierungsvariable $\delta = \delta(\eta)$ auswirkt. Auf Grund des quadratischen Bestrafungstermes in der Zielfunktion des Unterproblems (3.152) sollte

$$\frac{\mathrm{d}\delta}{\mathrm{d}\eta}(\eta^{[j]}) < 0 \tag{5.66}$$

gelten, ansonsten ist davon auszugehen, dass auf Grund von numerischen Ungenauigkeiten die Ableitung $\frac{\mathrm{d}\delta}{\mathrm{d}\eta}(\eta^{[j]})$ unbrauchbar ist und eine sensitivitätsbasierte Anpassung des Bestrafungsparameters somit nicht möglich ist.
Nikolayzik [81] schlägt vor mit Hilfe der Sensitivitäten das optimistische Ziel $\delta(\eta) \approx 0$ zu verfolgen. Erneut dient die Taylor-Approximation

$$\delta(\eta) \approx \delta(\eta^{[j]}) + \frac{\mathrm{d}\delta}{\mathrm{d}\eta}(\eta^{[j]})(\eta - \eta^{[j]}) \tag{5.67}$$

zur Abbildung des funktionalen Zusammenhanges von $\delta(\eta)$. Demnach liefert

$$0 = \delta(\eta^{[j]}) + \frac{\mathrm{d}\delta}{\mathrm{d}\eta}(\eta^{[j]})(\bar{\eta} - \eta^{[j]}) \tag{5.68}$$

$$\bar{\eta} = \eta^{[j]} - \left(\frac{\mathrm{d}\delta}{\mathrm{d}\eta}(\eta^{[j]})\right)^{-1} \delta(\eta^{[j]}) \tag{5.69}$$

eine Näherung für den Bestrafungswert $\bar{\eta}$ für den $\delta(\bar{\eta}) \approx 0$ erfüllt sein könnte, wenn die lineare Approximation hinreichend genau ist. Die Sensitivitätsableitung $\frac{\mathrm{d}\delta}{\mathrm{d}\eta}(\eta^{[j]})$ ist unter der getroffenen Annahme $N_\delta = 1$ skalarwertig und auf Grund der Forderung von Ungleichung (5.66) ist die Inversion in Gleichung (5.69) wohldefiniert.
Die Annahme mit Hilfe der Sensitivitäten $\delta(\bar{\eta}) \approx 0$ durch Anwendung einer Taylor-Approximation erster Ordnung erreichen zu können, scheint aber zu optimistisch. Bereits der Verlauf von $\delta(\eta)$ in Abbildung 3.4 für das Beispiel 3.28 offenbarte einen nichtlinearen Verlauf der Funktion. Somit kann im Allgemeinen nicht erwartet werden mit Hilfe der linearen Näherung in Gleichung (5.68) direkt $\delta(\bar{\eta}) \approx 0$ erreichen zu können.
Ein Kritikpunkt an der bisherigen Implementierung in Algorithmus 6 ist, dass die Anpassung des Parameters η lediglich durch Multiplikation mit einem Faktor realisiert wird. Während der Schleife innerhalb des Algorithmus gehen in keiner Weise Eigenschaften des konkreten Problems ein. Diese Überlegungen motivieren den Ansatz die Relaxierungsvariable η auf einen gewünschten Bruchteil $\theta \in [0, 1)$ zu reduzieren. Ausgehend von (5.68) ergibt sich die Formel

$$\theta\delta(\eta^{[j]}) = \delta(\eta^{[j]}) + \frac{\mathrm{d}\delta}{\mathrm{d}\eta}(\eta^{[j]})(\bar{\eta} - \eta^{[j]}) \tag{5.70}$$

$$\bar{\eta} = \eta_j + \left(\frac{\mathrm{d}\delta}{\mathrm{d}\eta}(\eta^{[j]})\right)^{-1} (\theta - 1)\delta(\eta^{[j]}) \tag{5.71}$$

zur Umsetzung einer derartigen Anpassung. Optimistische Anwender erhalten für $\theta = 0$ weiterhin den Ansatz von Nikolayzik, wohingegen vorsichtigere Wahlen von θ ebenfalls möglich sind. Die Einbindung in das SQP-Verfahren erfolgt dann analog zu Algorithmus 6. Die Vorgehensweise ist in Algorithmus 13 dargestellt.

Algorithmus 13 Relaxierung der Nebenbedingungen mit Hilfe von Sensitivitäten innerhalb von WORHP

1: Wähle $\delta_{\max} = 0{,}92$; $\theta = 0{,}75$; $\eta_{\max} = 1 \cdot 10^7$
2: Setze $\eta = 1{,}0$; $j = 0$ und die Startschätzung $\delta^{[0]} = 0$,
3: **repeat**
4: **if** $j > 0$ **then**
5: $\eta^{[j+1]} \leftarrow \eta^{[j]} + \left(\frac{\mathrm{d}\delta}{\mathrm{d}\eta}\right)^{-1} (\theta - 1)\delta^{[j]}$
6: **end if**
7: Löse das Unterproblem (3.152) mit Lösung $(d^{[j]}, \delta^{[j]})$
8: **until** (QP erfolgreich gelöst und $\delta^{[j]} < \delta_{\max}$) oder $\eta^{[j]} > \eta_{\max}$

5.4.3 Korrektur der Suchrichtung

Analog zur Anpassung des Regularisierungsparameters kann auch hier mit Hilfe der Sensitivitätsableitung $\frac{\mathrm{d}d}{\mathrm{d}\eta}(\eta_j)$ die Korrektur der Suchrichtung

$$d(\bar{\eta}) \approx d(\eta^{[j]}) + \frac{\mathrm{d}d}{\mathrm{d}\eta}(\eta^{[j]})(\bar{\eta} - \eta^{[j]}) \tag{5.72}$$

erfolgen. Diese kann einerseits als Startschätzung für das nächste zu lösende quadratische Unterproblem verwendet werden, um die Anzahl der Nebeniterationen zu reduzieren. Andererseits kann diese wie bereits von Nikolayzik [81] vorgeschlagen auch direkt als Suchrichtung verwendet werden. Auf diese Weise findet eine Korrektur der Suchrichtung statt, da für größere Bestrafungswerte η die Qualität der Suchrichtung zunehmen sollte, so dass diese den Fortschritt der Hauptiterationen verbessern.
Auch in diesem Zusammenhang sollte wie in Abschnitt 5.3.3 beschrieben ein größerer Wert für den Parameter β_α verwendet werden.

5.4.4 Mehrere Relaxierungsvariablen

Im Abschnitt 3.3.3.3 wurde die Möglichkeit diskutiert innerhalb von WORHP jede Nebenbedingung mit einer eigenen Relaxierungsvariablen zu versehen. Insbesondere das Beispiel 3.30 hat gezeigt, dass diese Vorgehensweise je nach Situation notwendig und sinnvoll sein kann. Im Folgenden sei $N_\delta = N_g + N_h$ passend über Gleichung (3.149) definiert. Nikolayzik diskutiert in seiner Arbeit [81] auch für diesen Fall die Verwendung der Sensitivitätsableitungen zur Korrektur der Suchrichtung und zur Bestimmung eines neuen Bestrafungswertes für die Relaxierung. Im hochdimensionalen Fall ist zu beachten, dass die Sensitivitätsmatrix $\frac{\mathrm{d}d}{\mathrm{d}\eta} \in \mathbb{R}^{N_x \times N_\delta}$ sehr groß und in der Regel dicht besetzt ist. Nikolayzik trifft für diesen Fall Annahmen an die Bestrafungsparameter, so dass er seine Berechnungen vereinfachen kann. Im Folgenden wird eine alternative Vereinfachung vorgestellt.

Algorithmus 14 Relaxierung der Nebenbedingungen im Fall $N_\delta = N_g + N_h$ mit $\eta \in \mathbb{R}$ innerhalb von WORHP

Wähle $\delta_{\max} = 0{,}92$; $\rho = 4{,}8$; $\eta_{\max} = 1 \cdot 10^7$
Setze $j = 0$, $\eta^{[0]} = 1$ und die Startschätzung $\delta^{[0]} = 0_{N_\delta \times 1}$,
repeat
  if $j > 0$ **then**
    if $\delta_k^{[j]} > \delta_{\max}$ **then**
      $\eta^{[j+1]} \leftarrow \rho \eta^{[j]}$
    end if
  end if
  Löse das Unterproblem (3.152) mit Lösung $(d^{[j]}, \delta^{[j]})$
until (QP erfolreich gelöst und $\delta_k^{[j]} < \delta_{\max}, \quad k = 1, ..., N_\delta$) oder $\eta^{[j]} > \eta_{\max}$

Die allgemeine Formulierung des quadratischen Unterproblems für den Fall mehrerer Relaxierungsvariablen wurde in Gleichung (3.152) vorgestellt. Die Bestrafungsparameter für die Relaxierungsvariablen wurden mit Hilfe der Matrix Ω aus Gleichung (3.150) in die Problemstellung integriert. Die in diesem Abschnitt zu betrachtende parametrische Störung ist durch $p = \eta \in \mathbb{R}^{N_\delta}$ gegeben. Um eine Reduzierung der Dimension der Sensitivitätsmatrix zu bewirken, können alle Relaxierungsvariablen mit nur einem Bestrafungswert verrechnet werden. Durch diese Vereinfachung resultiert das quadratische Unterproblem

$$
\begin{aligned}
\min_{d \in \mathbb{R}^{N_x}, \delta \in [0,1]^{N_\delta}} \quad & \tfrac{1}{2} d^\top Q d + \nabla_x f(x^{[k]})^\top d + \tfrac{\eta}{2} \delta^\top \delta \\
\text{unter} \quad & g_i(x^{[k]})(1 - \sigma_i \delta_i) + \nabla_x g_i(x^{[k]})^\top d \leq 0, \quad i = 1, ..., N_g \\
& h_j(x^{[k]})(1 - \delta_{N_g + j}) + \nabla_x h_j(x^{[k]})^\top d = 0, \quad j = 1, ..., N_h \\
& d_l + x_l - c_l \leq 0, \quad l = 1, ..., N_{\text{box}} \\
& \delta_l - 1 \leq 0, \quad l = 1, ..., N_\delta \\
& -\delta_l \leq 0, \quad l = 1, ..., N_\delta
\end{aligned}
\tag{5.73}
$$

mit nur einem Bestrafungsparameter $\eta \in \mathbb{R}$. Die Anpassung des Bestrafungsparameters η wird immer dann vorgenommen, wenn für mindestens einen Index j die Bedingung $\delta_j > \delta_{\max}$ erfüllt ist. Die Anpassungen ergeben den vereinfachten Algorithmus 14. Die numerischen Untersuchungen werden zeigen, dass der Einfluss dieser Vereinfachung vernachlässigbar ist.
Die Anpassung des Bestrafungsparameters $\eta \in \mathbb{R}$ kann ähnlich wie im Fall $N_\delta = 1$ vorgenommen werden. Da es in diesem Fall jedoch nur einen Parameter gibt, wird

für jeden Index k mit Hilfe der Sensitivität $\frac{\mathrm{d}\delta_k}{\mathrm{d}\eta}$ die Gleichung

$$\theta\delta_k(\eta^{[j]}) = \delta_k(\eta^{[j]}) + \frac{\mathrm{d}\delta_k}{\mathrm{d}\eta}(\eta^{[j]})(\bar{\eta} - \eta^{[j]}) \tag{5.74}$$

$$\bar{\eta}_k = \eta_j + \left(\frac{\mathrm{d}\delta_k}{\mathrm{d}\eta}(\eta^{[j]})\right)^{-1}(\theta - 1)\delta_k(\eta^{[j]}) \tag{5.75}$$

$$\tag{5.76}$$

verwendet, um den für diesen Index passenden Bestrafungswert $\bar{\eta}_k$ zu ermitteln. Danach können diese Werte verwendet werden, um mit Hilfe der Beziehung

$$\bar{\eta} = \max_k \bar{\eta}_k \tag{5.77}$$

eine Schätzung für den Bestrafungswert $\bar{\eta}$ für die Relaxierung in der nächsten Hauptiteration zu erhalten. Diese Vorgehensweise ist anschaulich in Algorithmus 15 dargestellt.

Algorithmus 15 Relaxierung der Nebenbedingungen im Fall $N_\delta = N_g + N_h$ mit Hilfe von Sensitivitäten innerhalb von WORHP

1: Wähle $\delta_{\max} = 0{,}92$; $\theta = 0{,}75$; $\rho = 4{,}8$; $\eta_{\max} = 1 \cdot 10^7$
2: Setze $j = 0$, $\eta^{[0]} = e$ und die Startschätzung $\delta^{[0]} = 0_{N_\delta}$,
3: **repeat**
4: **if** $j > 0$ **then**
5: **if** $\exists k$ mit $\delta_k^{[j]} > \delta_{\max}$ **then**
6: **if** $\frac{\mathrm{d}\delta_k}{\mathrm{d}\eta} < 0,\ \forall k$ **then**
7: $\bar{\rho} \leftarrow \max_k \left\{ \left(\frac{\mathrm{d}\delta_k}{\mathrm{d}\eta}(\eta^{[j]})\right)^{-1}(\theta - 1)\delta_k^{[j]}(\eta^{[j]}) \right\}$
8: $\eta^{[j+1]} \leftarrow \eta^{[j]} + \bar{\rho}$
9: **else**
10: $\eta^{[j+1]} \leftarrow \rho\eta^{[j]}$
11: **end if**
12: **end if**
13: **end if**
14: Löse das Unterproblem (3.152) mit Lösung $(d^{[j]}, \delta^{[j]})$
15: **until** (QP erfolreich gelöst und $\delta^{[j]} < \delta_{\max}$) oder $\eta^{[j]} > \eta_{\max}$

Bemerkung 5.2. *In der numerischen Umsetzung muss bei der Implementierung von Schritt 7 aus Algorithmus 15 beachtet werden, dass Sensitivitäten* $\frac{\mathrm{d}\delta_k}{\mathrm{d}\eta}(\eta^{[j]}) \approx 0$ *gegebenenfalls nicht zur Berechnung des Maximums verwendet werden dürfen. Sehr kleine Sensitivitäten sind häufig durch numerisches Rauschen bei der Lösung des linearen Gleichungssystems zu begründen und können in diesem Zusammenhang ignoriert werden, da sie sonst einen zu großen Einfluss auf den Algorithmus haben würden.*

Kapitel 6

Parallelisierungsansätze für SQP-Verfahren

Inhaltsangabe

In den letzten Jahren hat sich die Entwicklung von Prozessoren stark verändert. Bis etwa 2005 konnte neben anderen Verbesserungen ein exponentielles Wachstum der Taktfrequenz beobachtet werden. Auf diese Weise wurde bestehende Software auf Grund der Weiterentwicklung der verwendeten Hardware ohne zusätzlichen Aufwand der Programmierer schneller. Dieser Umstand wird als "Free Performance Lunch" bezeichnet.
Etwa im Jahr 2004 endete diese Entwicklung. Führende Chiphersteller wie Intel und AMD stießen an physikalische Grenzen in der Herstellung noch schnellerer Prozessoren. In erster Linie stellen die Hitzeentwicklung und der Energieverbrauch Barrieren für die weitere Entwicklung dar. Deshalb fokussieren sich die Chiphersteller heute

auf die Integration mehrerer Prozessorkerne innerhalb eines Chips, im Gegensatz zur alleinigen Forschung an der Verbesserung einzelner Kerne. Sutter beschreibt diesen Umstand in seinem Artikel [99]. Insbesondere betont er, dass diese Entwicklung eine neue Herausforderung für Softwareentwickler darstellt. Wurden früher allein auf Grund der sich regelmäßig verbessernden Hardware Geschwindigkeitssteigerungen für Software erzielt, so ist heute ein Umdenken der Entwickler nötig. Durch die Einführung der Mehrkernprozessoren müssen die Kontrollflüsse innerhalb der Programme überarbeitet werden, so dass durch die Anwendung von Nebenläufigkeit mit Hilfe der zur Verfügung stehenden Mehrkernprozessoren Geschwindigkeitssteigerungen erzielt werden können.
Zur Zeit sind im Bereich der Arbeitsplatzrechner und Laptops zwei bis vier Kerne der Standard und bei Servern im mittleren Preissegment sogar zwölf bis sechzehn Kerne (Stand April 2017). Neben den physikalischen Kernen steht zusätzlich häufig Hyperthreading zur Verfügung. Diese Entwicklung rechtfertigt die Frage, inwiefern Anwender mathematischer Software von dieser Entwicklung profitieren können. Das Ziel der Betrachtungen in diesem Kapitel ist deshalb die Verwendung von Optimierungssoftware für Anwender auf ihren Arbeitsrechnern zu verbessern.
Im Gegensatz dazu wurde die Anpassung allgemeiner Algorithmen zur Verwendung auf Hoch- und Höchstleistungsrechner sowie Clustern bereits 1989 ausführlich im Buch von Bertsekas und Tsitsiklis [4] diskutiert. Die Anwendung dieser Vorgehensweisen bei der Lösung von nichtlinearen Optimierungsproblemen wurde beispielsweise von Lootsma und Ragsdell [73], Boden, Gehne und Grauer [7] und der dort zitierten Literatur untersucht und ist nicht Ziel der hier vorgestellten Techniken. Insbesondere unterscheiden sich die Implementierungen für derartige Architekturen zum Teil signifikant gegenüber der Entwicklung für Mehrkernprozessoren, da sich die Kommunikation der einzelnen Programmteile beispielsweise bei Cluster-Implementierungen deutlich aufwendiger gestaltet. Wohingegen die einzelnen Kerne eines Mehrkernprozessors physikalisch, als auch programmatisch direkt auf demselben Speicher arbeiten. Deshalb müssen diese Architekturen bei der Entwicklung von Algorithmen getrennt betrachtet werden. Trotzdem kann es durchaus sinnvoll sein einige der folgenden Strategien auch für die Parallelisierung auf derartigen Maschinen beziehungsweise Umgebungen zur Verbesserung einzusetzen.
Im folgenden Abschnitt wird einleitend eine Übersicht über mögliche Parallelisierungsstrategien gegeben. Im Abschnitt 6.2 wird anschließend auf die Architektur des Lösers WORHP eingegangen, damit diese bei der Beschreibung konkreter Parallelisierungsansätze im restlichen Teil des Kapitels berücksichtigt werden kann.

6.1 Parallelisierungsstrategien

Die Betrachtungen im Kapitel 3 haben gezeigt, dass ableitungsbasierte Optimierungsverfahren durch das klassische Newton-Verfahren motiviert sind. Dementsprechend handelt es sich um iterative Algorithmen, die zunächst sequentieller Natur

sind. Demnach ist nicht zu erwarten, dass es möglich ist mit Hilfe von Parallelisierung eine Verbesserung aller Teile des Algorithmus erwirken zu können.
Nichtsdestotrotz existieren verschiedene Ansätze Teile der Algorithmen durch Ausnutzung von nebenläufigen Programmstrukturen möglichst effizient umzusetzen und so zumindest partiell durch Parallelisierung eine Effizienzsteigerung zu erwirken. Der Artikel von Schnabel [94] gibt eine gute Übersicht über die verschiedenen Möglichkeiten zur Parallelisierung innerhalb nichtlinearer Optimierungsverfahren. Eine Übersicht über eine Vielzahl von weiteren Artikeln, welche sich mit Parallelisierung in diesem Gebiet befassen, geben Migdalas, Toraldo und Kumar [80].
Im Laufe dieser Arbeit wurden diverse Möglichkeiten vorgestellt das Optimierungsverfahren durch partielle Anpassungen der Vorgehensweise zu individualisieren. Die numerischen Ergebnisse im nächsten Kapitel werden zeigen, dass je nach gewählter Strategie in der Regel Teilmengen der betrachteten Probleme mit speziellen Algorithmen besser gelöst werden können, wohingegen der Lösungsvorgang anderer Beispiele empfindlich auf Änderungen am Algorithmus reagiert. Diese Beobachtungen motivieren eine Parallelisierung des Algorithmus von Außen.
Es wird die Vielzahl an Konfigurationsmöglichkeiten eines Lösers, wie beispielsweise WORHP, ausgenutzt um die Aufrufe des Lösers selbst zu parallelisieren. Im Abschnitt 6.3 werden die technische Umsetzung, eine Strategie zur Geschwindigkeitssteigerung und eine Vielzahl alternativer Anwendungsszenarien für die neu entwickelte WORHP Mehrkernschnittstelle (WORHP Multi-Core Interface, kurz WMCI) diskutiert.
Neben der Berechnungen innerhalb des Optimierungsverfahrens kann auch die Auswertung der gegebenen Zielfunktion, Nebenbedingungen oder Ableitungen der Problemstellung einen entscheidenden Anteil des Gesamtaufwandes haben. In Abbildung 6.1 ist beispielsweise die Verteilung des Rechenaufwandes bei der Lösung eines Parameteridentifikationsproblem mit Hilfe von WORHP bei Verwendung finiter Differenzen abgebildet. In der Abbildung ist ersichtlich, dass in diesem konkreten Fall die Berechnung der Hesse-Matrix etwa 92% der gesamten Rechenzeit des Optimierungsprozesses verbraucht. Dies verdeutlicht, dass auf Grund komplexer Implementierungen der Modellfunktionen ein hoher Rechenaufwand entstehen kann. Insbesondere die Verwendung finiter Differenzen zur Kompensation fehlender analytischer Ableitungen kann zu einem erheblichen Zeitaufwand während der Optimierung führen.
Es stehen verschiedene Möglichkeiten zur Verfügung in diesem Fall durch den Einsatz von Nebenläufigkeit die Effizienz der Optimierung zu verbessern. Eine direkte Parallelisierung innerhalb des Modells stellt eine gute Möglichkeit dar, liegt aber im Allgemeinen nicht im Verantwortungsbereich der Entwicklers eines NLP Lösers, deshalb kann dieser Fall nicht als allgemeine Möglichkeit diskutiert werden.
Weitere Strategien lassen sich umsetzen, wenn es zumindest möglich ist Funktionsauswertungen des Modells in parallel anzufordern. Auf diese Weise können während des Optimierungslaufes die Liniensuche und die finiten Differenzen verbessert werden. Schittkowski [93] beschreibt, auf welche Weise die Möglichkeit zur Auswertung

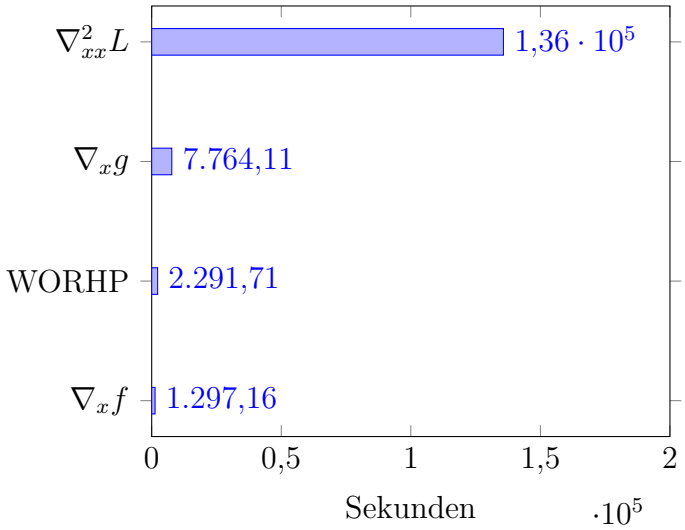


Abbildung 6.1: Aufwandsverteilung bei Lösung eines beispielhaften Parameteridentifikationsproblems mit $N_x = 11.408$ Optimierungsvariablen und $N_g = 43.214$ Ungleichungsnebenbedingungen bei Verwendung finiter Differenzen

von Funktionen in parallel ausgenutzt werden kann, um die Liniensuche oder Ableitungsberechnung mittels finiter Differenzen zu verbessern. Als weitere Quelle sei an dieser Stelle auf Schnabel [94] verwiesen, neben einem guten Überblick über Parallelisierung im Zusammenhang mit nichtlinearer Optimierung im Allgemeinen werden dort ebenfalls Anwendungsmöglichkeiten für parallele Funktionsauswertungen dargestellt. Im Abschnitt 6.4 werden diese Möglichkeiten anhand des Lösers WORHP analysiert und eine mögliche Umsetzung diskutiert.
Weiterhin besteht die Möglichkeit gezielt Teile des Algorithmus zu parallelisieren. Grundsätzlich ist eine mögliche Parallelisierungsstrategie die Verwendung von parallelen Implementierungen grundlegender Operationen wie Matrix-Vektor- oder Matrix-Matrix-Multiplikationen. Die Auswirkungen derartiger Techniken werden im Folgenden aber nicht weiter analysiert, da WORHP diese Operationen nahezu ausschließlich innerhalb des linearen Algebrapaketes MA97 benötigt und dieser eigene Parallelisierungsoptionen bietet. Im Abschnitt 6.5 wird deshalb kurz die Parallelisierung der linearen Algebra analysiert.
Zuletzt muss noch eine weitere Parallelisierungsmöglichkeit erwähnt werden. Viele Algorithmen basieren auf dem „Teile-und-Herrsche“-Prinzip und es existieren Ansätze diese Vorgehensweise zur Lösung von nichtlinearen Optimierungsproblemen anzuwenden. In der Regel müssen einschränkende Forderungen an die Aufgabenstellung in Form von separablen Nebenbedingungen getroffen werden. Beispielsweise Xu und Chen [114] beschreiben eine parallele Implementierung basierend auf der Partitionierung der Nebenbedingungen. Der Fokus dieser Arbeit liegt auf der allgemeinen Effizienzsteigerung nichtlinearer Optimierung, ohne einschränkende Forderungen an die Problemstellung treffen zu müssen. Deshalb werden Parallelisierungstechniken basierend auf der Partitionierung des Ausgangsproblems nicht weiter thematisiert.

6.2 Softwarearchitektur von WORHP

In diesem Abschnitt wird die Architektur des Lösers WORHP dargestellt, um mögliche Probleme bei der Parallelisierung aufzuzeigen und die Vorgehensweise in den folgenden Abschnitten zu motivieren. Eine ausführliche Beschreibung der Softwarearchitektur und Implementierungshintergründe findet sich in der Arbeit von Wassel [109] und in gekürzter Form auch im Handbuch [98]. Die folgenden Darstellungen sind den Ausführungen dieser Quellen entnommen.
Für die weiteren Betrachtungen sind im Wesentlichen zwei Grundsätze des Softwareaufbaus von WORHP entscheidend. Einerseits ist die interne Datenstruktur wichtig, um zu erkennen ob es potenzielle Probleme bei der Parallelisierung geben kann. Die Daten werden innerhalb des Lösers in den vier Strukturen *OptVar, Workspace, Params* und *Control* gesammelt:

***OptVar*:** In dieser Struktur sind die primalen und dualen Optimierungsvariablen, Dimensionen und Beschränkungen enthalten.

***Workspace*:** Jegliche Arbeitsdaten des Lösers sind in dieser Struktur enthalten. Hier finden sich beispielsweise aktuelle Werte der Ableitungsmatrizen, Bestrafungsparameter und alle weiteren relevanten Größen während der Optimierung.

***Params*:** Diese Struktur beinhaltet die vom Benutzer konfigurierten Einstellungen des Lösers.

***Control*:** Die Variablen zur Steuerung der im folgenden beschriebenen Reverse-Communication werden hier gesammelt.

Diese vier Strukturen enthalten zusammen alle während der Optimierung benötigten Daten. Jede große Routine innerhalb der Software arbeitet auf diesen. Diese Technik wird als *Unified-Solver-Interface* (Einheitliche Schnittstelle des Lösers) bezeichnet. Der von Wassel [109] beschriebene Umstand, dass der QP Löser über einen internen Zustand verfügt und somit nicht parallel aufgerufen werden kann, ist durch die Beseitigung der innerhalb des Fortran-Codes auftretenden *SAVE*-Variablen nicht mehr zutreffend. Somit kann eine Parallelisierung durchgeführt werden, wenn für jeden Thread eine eigene Kopie dieser vier Strukturen bereitgestellt wird.
Neben des internen Aufbaus des Programmes ist die Schnittstelle zu den Problemfunktionen des Benutzers bei der Parallelisierung zu berücksichtigen. Hierbei ist zwischen Code des Benutzers und den WORHP-internen Routinen zu unterscheiden. Der Benutzer bindet die Optimierungssoftware als Bibliothek in seinem Programm ein. Die Optimierung wird durch Implementierung der sogenannten Reverse-Communication-Schleife realisiert. In dieser Schleife wird der Löser aufgerufen und nachdem eine sogenannte Phase (stage) abgearbeitet wurde, kommt der Kontrollfluss zurück in die vom Benutzer implementierte Schleife. Dort werden jetzt Anfragen des

Lösers nach Auswertungen der Zielfunktion, Nebenbedingungen oder Ableitungen verarbeitet und danach wird die nächste Phase des Lösers aufgerufen. Abbildung 6.2 stellt diesen Ablauf schematisch dar[1]. Auf der linken Seite ist der Programmcode des Benutzers zu sehen und auf der rechten Seite der Optimierungsalgorithmus. Diese Technik wird als *Reverse-Communication* bezeichnet, da die Hauptschleife der Optimierung im Programmcode des Benutzers implementiert wird und der Kontrollfluss in jedem Durchlauf der Optimierungsschleife an den Anwender zurückgegeben wird. Die Verwendung der Reverse-Communication verkompliziert die Parallelisierung der Auswertung von Modellfunktionen. Einerseits werden die Auswertungsanforderungen vom Löser formuliert, andererseits muss die parallele Bearbeitung der Anforderungen auf Seiten des Nutzers durchgeführt werden. Dieser Umstand wird durch Abbildung 6.2 verdeutlicht. Im Abschnitt 6.4 wird eine Implementierungsstrategie vorgeschlagen, die es WORHP erlaubt über die Reverse-Communication mehrere Auswertungen der Zielfunktion und Nebenbedingungen anzufordern. Auf diese Weise hat der Benutzer die Möglichkeit Nebenläufigkeit auszunutzen.

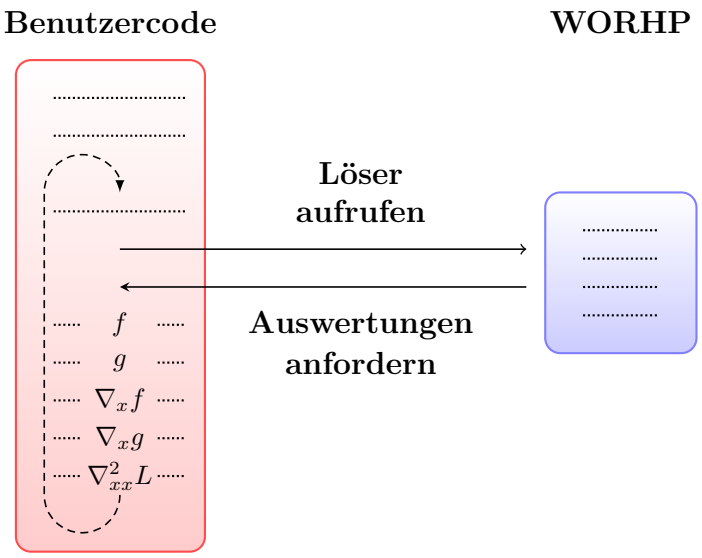


Abbildung 6.2: Aufteilung WORHP-Code und Benutzercode in der Reverse-Communication.

6.3 Mehrkernschnittstelle

Algorithmen wie das SQP-Verfahren zielen darauf ab lokale Minima zu finden. Soll die Suche ausgeweitet werden, um das globale Minimum einer Aufgabenstellung zu finden, oder um zumindest ein besseres lokales Minimum auffinden zu können,

[1] Abbildung 6.2 basiert auf Abbildung 2.1 aus der Arbeit von Wassel [109]. Dort wird das Schema noch mit der klassischen *Direct-Communication* verglichen.

werden lokale Verfahren klassischerweise mit Heuristiken gekoppelt. Beispielsweise kann dies mit Hilfe sogenannter genetischer Algorithmen umgesetzt werden. Diese erfordern in ihrer Umsetzung die Lösung vieler nichtlinearer Optimierungsprobleme mit abweichenden Startschätzungen und möglicherweise weiteren Anpassungen der Problemformulierung. Beispielsweise Cantú-Paz [15] oder Van Veldhuizen [103] beschäftigen sich mit der Frage, wie eine effiziente Parallelisierung in dieser Situation aussehen kann. In diesem Rahmen wird häufig die sogenannte Master-Slave-Parallelisierung angewendet. Der Hauptalgorithmus beauftragt die Lösung von Teilproblemen und startet Threads zur Erledigung dieser Aufgaben. Abbildung 6.3 verdeutlicht die Vorgehensweise. Diese Technik kann beispielsweise auch zur Lösung

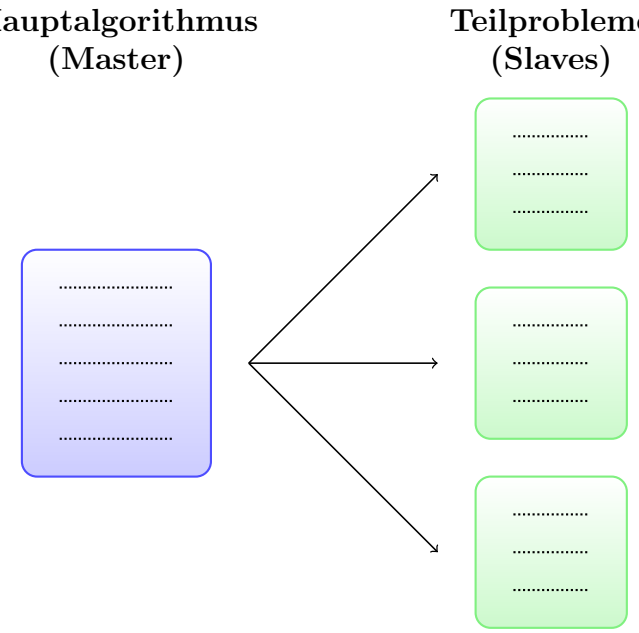


Abbildung 6.3: Master-Slave-Parallelisierung: Der Hauptalgorithmus (Master) startet Threads zur Lösung von Teilproblemen (Slaves)

mehrkriterieller Optimierungsprobleme mit Hilfe diverser Heuristiken genutzt werden. In dem Artikel von Talbi et al. [100] werden verschiedene Parallelisierungsansätze für diese Art von Optimierungsproblemen beschrieben.
Dieser Abschnitt ist im Folgenden der Frage gewidmet, ob es möglich ist mit Hilfe des Master-Slave-Ansatzes die Lösung eines allgemeinen nichtlinearen Optimierungsproblems zu verbessern, ohne dabei spezielle Ziele wie die globale Optimierung oder die Berücksichtigung zusätzlicher Zielfunktionen zu verfolgen.
Im bisherigen Verlauf dieser Arbeit wurden diverse Optionen zur Beeinflussung des Optimierungsverfahrens vorgestellt. Wolpert und Macready zeigen in ihrer Arbeit [113], dass in der kombinatorischen Optimierung die Variation eines Optimierungsalgorithmus auf einer Teilmenge der zu lösenden Probleme positive Auswirkungen haben kann, gleichzeitig aber zu einer Verschlechterung auf einer anderen Teilmenge führen wird. Die numerischen Untersuchungen innerhalb dieser Arbeit werden die

Vermutung aufwerfen, dass sich die Aussage von Wolpert und Macready auch auf die Lösung allgemeiner nichtlinearer Optimierungsprobleme übertragen lässt.
Die Verfügbarkeit vieler Konfigurationsmöglichkeiten innerhalb von WORHP, in Kombination mit der beschriebenen Vermutung, motivieren die Parallelisierung verschiedener Konfigurationen des Optimierungsalgorithmus zur Lösung des gleichen Problemes. Zur Umsetzung dieser Idee wurde eine neue Benutzerschnittstelle entwickelt, um die nötigen Rahmenbedingungen zur technischen Durchführung zu schaffen. Die folgenden Beschreibungen basieren zum Teil auf dem Konferenzbeitrag von Geffken und Büskens [43]. In diesem wurde die Schnittstelle erstmalig vorgestellt.
Die Implementierung der sogenannten Mehrkernschnittstelle wurde in C++ in Ergänzung zu den bereits vorhandenen Schnittstellen des Lösers WORHP vorgenommen. Die Wahl der Sprache wurde auf Grund verschiedener Kriterien getroffen. Das wichtigste Argument für diese Wahl war die Verfügbarkeit der notwendigen Sprachfeatures in allen aktuellen Compilern, um die Implementierung der Nebenläufigkeitsfunktionalitäten ohne externe Bibliotheken realisieren zu können. So kann sichergestellt werden, dass keine zusätzlichen Abhängigkeiten für die Software entstehen. Ein weiterer wichtiger Punkt ist die Möglichkeit die Implementierung der gewünschten Funktionalitäten betriebssystemunabhängig vornehmen zu können und so die Portabilität der Software zu gewährleisten. Die Parallelisierung der Aufrufe des Optimierers stellt weiterhin besondere Anforderungen an die Implementierung der Problemstellung und des Lösers.

i. Die Auswertung der Funktionen des Benutzers muss für verschiedene Instanzen in parallel möglich sein.

ii. Es sollen mehrfache Instanzen des Lösers erstellt werden können.

iii. Es dürfen nur threadsichere statische beziehungsweise globale Funktionen verwendet werden.

Die Parallelisierung der Modellfunktionen meint in diesem Zusammenhang nicht, die Reverse-Communication-Schleife zu überarbeiten, weil durch die Erstellung mehrfacher Instanzen des Lösers auch eine Vervielfältigung der enthaltenen Reverse-Communication-Schleifen umgesetzt wird. Eine Parallelisierung der Auswertungen innerhalb dieser Schleifen ist hier nicht zielführend, da die vorhandenen Kapazitäten des Prozessors bereits für die zusätzlichen Instanzen des Lösers verwendet werden.
Die neue Schnittstelle ist objektorientiert aufgebaut. Die Problemformulierung übergibt der Benutzer durch die Implementierung eines sogenannten `NLPProblem`-Objektes an den Lösers. Dieses muss kopierbar sein und die erforderlichen Funktionsauswertungen müssen bei gleichzeitiger Auswertung verschiedener Instanzen threadsicher integriert sein. Die threadsichere Implementierung der Modellfunktionen innerhalb des `NLPProblem` erfordert ein gewisses Verständnis des Nutzers. Wie bereits erwähnt wurde, liegt die Bereitstellung der Modellfunktionen in dessen Aufgabenbereich, so dass die Entwickler der parallelen Löserschnittstelle auf diese Situation keinen direkten Einfluss haben.

Die zweite Forderung kann erfüllt werden, indem alle benötigten Objekte innerhalb der Implementierung kopierbar sind. Durch die Einführung der neuen Objektstruktur sollte dies mit Hilfe der C++ Kopierfunktionalität sichergestellt werden. Die Erfüllung dieser Anforderung muss vom Anwender bei der Implementierung seiner Modellfunktionen beachtet werden.
Die dritte Forderung stellt für den Löser insofern eine Einschränkung dar, dass innerhalb von WORHP die Ausgabe des Iterationsverlaufs über statische Ausgabefunktionen erfolgt und diese sind bekanntermaßen nicht threadsicher. Deshalb bietet die Mehrkernschnittstelle eine eigene Technik zur Ausgabe der Iterationsverläufe. Durch die erste Forderung müssen auch in den Modellfunktionen die Aufrufe von statischen und globalen Funktionen durch den Nutzer kritisch betrachtet werden.
Der folgende Abschnitt beschreibt die Objektstruktur der neu eingeführten Schnittstelle. Zusätzlich wird noch der Zusammenhang zur bestehenden Architektur des Lösers dargestellt.

6.3.1 Struktur der Schnittstelle

Die Umsetzung einer Parallelisierungsstrategie erfordert in der Regel eine Unterscheidung zwischen der Organisationsebene der verschiedenen Threads und den Programmteilen, in denen die eigentliche Berechnung stattfindet. Dies entspricht der Vorgehensweise der Master-Slave-Parallelisierung (vergleiche Abbildung 6.3). Bei der Entwicklung der Mehrkernschnittstelle führte diese Situation zu zwei unterschiedlichen Implementierungsebenen.
Der Benutzer implementiert seine konkreten Problemfunktionen in der inneren Ebene. Die abstrakte `NLPProblem`-Klasse stellt eine Schnittstelle bereit, so dass der Benutzer durch Vererbung seine Implementierung vornehmen kann. In der abgeleiteten Klasse können die Funktionen, und wenn verfügbar die Ableitungen, in Übereinstimmung mit der im Codeausschnitt 6.1 teilweise abgedruckten Basisklasse, überschrieben werden.
Analog zur Verwendung der verschiedenen Standard-Schnittstellen zu WORHP müssen mindestens die Zielfunktion, und falls benötigt die Nebenbedingungen, angegeben werden. Die Ableitungen werden entweder implementiert, oder als leere Funktionen bereitgestellt, um den Anforderungen der Schnittstelle zu genügen. Wenn die Ableitungen nicht verfügbar sind, muss über die Konfiguration des Lösers sichergestellt werden, dass WORHP die Berechnung mit Hilfe der im Abschnitt 3.3.2 beschriebenen Methoden selbst durchführt.
Die Verwaltung der Threads wird in der äußeren Ebene durch die `NLPSolver`-Klasse realisiert. Weiterhin sind die in den folgenden Abschnitten beschriebenen unterschiedlichen Arbeitsmodi innerhalb dieser Schicht umgesetzt. Während des Optimierungsablaufes werden jegliche Probleme auf Grund der Parallelisierung umgangen, indem für jeden zur Lösung zur Verfügung stehenden Thread eine Instanz des konkreten vom Benutzer gegebenen Problems erstellt wird. Intern werden in einer Liste

die zu lösenden Probleme verwaltet. Der Benutzer konfiguriert über die Schnittstelle die Anzahl Threads, die dem Löser zur Verfügung gestellt werden. Diese Anzahl sollte maximal der Anzahl an Kernen innerhalb des Systems entsprechen, da sich die verschiedenen Threads ansonsten gegenseitig bremsen. Es ist zu beachten, dass idealerweise für den Verwaltungsthread ein zusätzlicher Kern zur Verfügung stehen sollte.

```
class NLPProblem {
public :
/* Prototype for evaluation of
* objective function */
virtual void evalF(double *f)      = 0;

/* Prototype for evaluation of
* constraints function */
virtual void evalG(double *G)      = 0;

/* Prototype for evaluation of
* objective gradient */
virtual void evalDF(double *dfVal) = 0;

/* Prototype for evaluation of
* jacobian of the constraints */
virtual void evalDG(double *dgVal) = 0;

/* Prototype for evaluation of
* hessian of the lagrangian */
virtual void evalHM(double *hmVal) = 0;
...
};
```

Listing 6.1: `NLPProblem`-Klasse als Schnittstelle für Problemfunktionen des Benutzers

Im konkreten Anwendungsfall ist zu prüfen, ob die eventuell durch Hyperthreading vorhandenen logischen Kerne berücksichtigt werden sollen[2]. Die Verwendung von allen logischen Kernen ist beispielsweise dann sinnvoll, wenn Engpässe in der Berechnung auf Grund der Kommunikation des Prozessors mit dem Speicher entstehen. Im Gegensatz dazu ist von deren Verwendung abzuraten, wenn der Großteil der Rechenzeit für die Rechenoperationen aufgebracht werden muss. Allgemeine Erklärungen zum Hyperthreading sind beispielsweise in dem Artikel von Marr et al. [78] zu finden.
Der Verwaltungsthread, in der Implementierung repräsentiert durch die `NLPSolver`-Klasse, weist jedem verfügbaren Slot ein zu lösendes Problem zu und startet dessen Lösung in einem Thread. Während die einzelnen Instanzen des Optimierers die zugewiesenen Probleme lösen, überwacht der Verwalter den Fortschritt, verarbeitet

[2]Die Verteilung der Threads erfolgt letztendlich über das Betriebssystem, aber der Benutzer kann über die Threadanzahl die Last trotzdem steuern.

die Ausgaben der Instanzen und organisiert den weiteren Programmablauf entsprechend des gewählten Arbeitsmodus. Diese Organisationsstruktur ist in Abbildung 6.4, welche dem Konferenzbeitrag von Geffken und Büskens [43] entnommen wurde, abgedruckt.
Die Kommunikation zwischen den Arbeiterthreads, den Slaves, und dem Verwal-

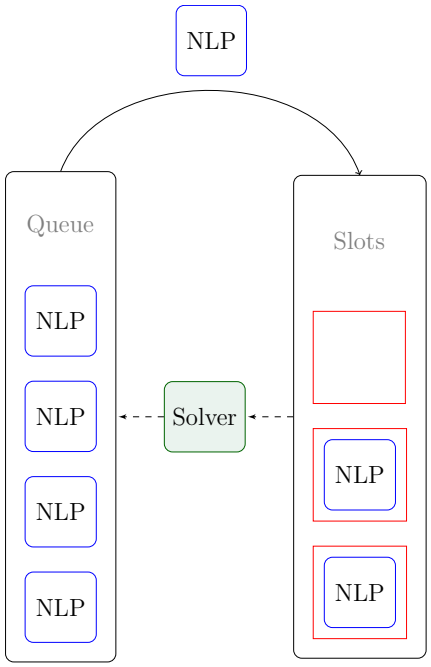


Abbildung 6.4: Verwaltung der Threads durch die `NLPSolver`-Klasse

tungsthread, dem Master, läuft über ein Meldesystem ab. Der Verwaltungsthread prüft regelmäßig, ob in einer Liste neue Benachrichtungen der einzelnen Arbeiter eingegangen sind. Mit Hilfe dieser Benachrichtigungen werden Informationen über den aktuellen Iterationsfortschritt in Form von Zielfunktionswert, Nebenbedingungsverletzung und weiterer interner Größen übermittelt. Bei Erhalt einer neuen Nachricht verarbeitet der Verwaltungsthread diese und erstellt eine passende Ausgabe für den Benutzer. Auf diese Weise ist der Verwalter jederzeit über den Stand der unterschiedlichen Optimierungen informiert. Gleichzeitig wurde die Problematik umgangen, dass die Ausgabe der Iterationsverläufe nicht threadsicher erfolgen kann. Terminiert ein Arbeiter den Optimierungsprozess und es stehen noch Probleme zur Lösung bereit, werden diese den erneut zur Verfügung stehenden Slots zugewiesen.

6.3.2 Automatische Codegenerierung

Die initiale Konfiguration der Parameter erfolgt für WORHP über eine Parameterdatei. Innerhalb des Programmcodes kann zur Laufzeit durch Zugriff auf die `Params`-Struktur weiterhin Einfluss auf die Optimierung genommen werden. Einige Arbeitsmodi der Mehrkernschnittstelle zielen darauf ab mehrere Instanzen des Lösers mit

unterschiedlichen Parametersetzungen zu starten. Für diese Fälle ist es aber nicht praktikabel, dass der Benutzer für jede Instanz eine eigene Parameterdatei anlegen und konfigurieren muss.
Das WMCI kann über ein Skript derart konfiguriert werden, dass intern Kombinationen der gewählten Parameter gebildet werden. Der Benutzer stellt eine Basisparameterdatei für WORHP wie gewohnt im XML-Format zur Verfügung und konfiguriert in einer weiteren Datei die zu kombinierenden Parametervariationen. Mit Hilfe eines Skriptes wird anschließend automatisch Programmcode generiert, um die gewünschten Parametereinstellungen für die verschiedenen Threads zu realisieren. Im Anhang A.2 wird anhand einer kurzen Konfigurationsdatei die Funktionalität erklärt.
Die Konfigurationsmöglichkeiten der Skriptdatei für die Codeerstellung der automatischen Parametervariationen zeigen, dass je nach Einstellung zum Teil eingehende Kenntnisse der internen Struktur des Lösers erforderlich sind. Damit auch unerfahrenere Nutzer des Lösers von der vorgestellten Technik profitieren können, sind häufig verwendete Konfigurationen in der Standardversion der Skriptkonfigurationsdatei enthalten und werden mit der Schnittstelle geliefert. Auf diese Weise muss der Anwender nur aus dem bereits vorhandene Angebot auswählen oder kann alternativ alle Voreinstellungen verwenden. Die Standardkonfiguration ist ebenfalls im Anhang dieser Arbeit im Abschnitt A.1 inklusive einer kurzen Dokumentation abgedruckt.

6.3.3 Verschiedene Arbeitsmodi

Die parallele Ausführung mehrerer Instanzen des Lösers kann mit unterschiedlichen Zielsetzungen erfolgen. In diesem Abschnitt werden verschiedene Strategien vorgestellt, die es einem Anwender erlauben, den Einsatz eines Optimierers zu verbessern. Die vielseitigen Möglichkeiten zur Individualisierung eines Programmes wie WORHP motivieren die automatisierte Anpassung der Parameter des Lösers als klassische Anwendungsvariante der Mehrkernschnittstelle. Die Standard-Konfiguration der Parameter wurde entwickelt, um eine möglichst große Anzahl von Problemen aus verschiedenen Testsammlungen erfolgreich zu lösen. Dieser Ansatz erscheint naheliegend, da die Qualität der Software für eine Vielzahl von Problemen verbessert wird. Für einen Benutzer, der lediglich an der Lösung seines konkreten Problems oder einer Gruppe von verwandten Problemen interessiert ist, kann diese Vorgehensweise aber zu allgemein sein. Gleichzeitig kennt der Anwender aber in der Regel die Interna des Lösers nicht ausreichend, um eine Anpassung der Konfiguration für seine konkrete Situation vorzunehmen. Somit finden sich beispielsweise Abschlussarbeiten, wie die von Söderström [96]. In dieser Arbeit werden Ansätze entwickelt, um die Konfiguration eines Optimierers, dort IPOPT, für die konkrete Problemstellung der Strahlentherapie anzupassen.
Das WMCI erlaubt es verschiedene Instanzen desselben Problems mit unterschiedlichen Konfigurationen des Lösers in parallel zu starten. Nachdem eine Instanz eine optimale Lösung gefunden hat, werden die übrigen Lösungsversuche gestoppt und die

	Optimal	Akzeptabel
Optimalität	10^{-6}	10^{-3}
Zulässigkeit	10^{-6}	10^{-3}

Tabelle 6.1: Standardeinstellung: Optimale und akzeptable Abbruchbedingungen innerhalb von WORHP

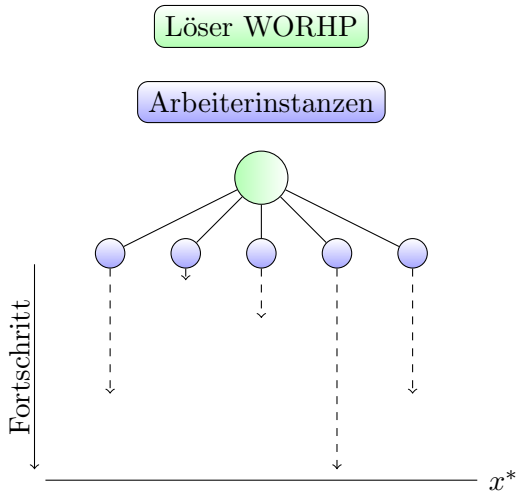


Abbildung 6.5: First-Across-The-Line, Es werden mehrere Instanzen des Löser gestartet, um eine optimale Lösung x^* aufzufinden.

gefundene Lösung wird dem Anwender zur Verfügung gestellt. WORHP unterscheidet zwischen Fehlschlägen, akzeptablen und optimalen Lösungen. Standardmäßig sind die Toleranzen bezüglich Optimalität und Zulässigkeit wie in Tabelle 6.1 gesetzt. Nur beim Auffinden einer optimalen Lösung werden alle Instanzen abgebrochen, wird lediglich eine akzeptable Lösung gefunden, oder schlägt eine Instanz fehlt, setzen die übrigen Threads ihre Arbeit fort. Dieser Ansatz wird als *First-Across-The-Line* bezeichnet. Abbildung 6.5 zeigt schematisch die Vorgehensweise. In der Abbildung erreicht eine Arbeiterinstanz ein lokales Minimum und somit können die übrigen Instanzen gestoppt werden.
In erster Linie zielt der beschriebene Ansatz darauf ab, die Lösungsgeschwindigkeit zu verbessern. Zusätzlich wird aber durch die verschiedenen Konfigurationen auch die Robustheit des Verfahrens erhöht. Unter Vernachlässigung des zusätzlichen Verwaltungsaufwands für die Organisation der Threads ist diese Vorgehensweise niemals langsamer als die Verwendung einer einzelnen Instanz des Lösers. Gleichzeitig besteht je nach Beschaffenheit des Problems und Wahl der Konfigurationen die Möglichkeit eine deutliche Geschwindigkeitssteigerung zu erzielen. Die Argumentation bezüglich der Robustheit ist ähnlich. Arbeitet eine Instanz auf Basis der Standardkonfiguration, so dient diese als Absicherung für andere, möglicherweise

experimentellere, Parametersetzungen. Diese Basisinstanz stellt sicher, dass Probleme die ohne Parallelisierung gelöst wurden auch weiterhin erfolgreich bearbeitet werden.
Mit Hilfe der zuvor beschriebenen automatisch generierten Parameterkonfigurationen muss der Benutzer sich zunächst nicht näher mit den Bedeutungen der Einstellungen beschäftigen. Trotzdem ist er mit Hilfe der Mehrkernschnittstelle in der Lage, von deren Möglichkeiten zu profitieren.
Im Kapitel 3 wurde erläutert, dass SQP-Verfahren auf dem Newton-Verfahren basieren und dessen lokale Konvergenzeigenschaften erben. Insbesondere Optimierungsprobleme mit vielen lokalen Minimalstellen werden deshalb häufig nicht zur Zufriedenheit des Anwenders gelöst. Neben der Wahl der Startschätzung $x^{[0]}$ für die Optimierung hängt das Ergebnis in einem solchen Fall häufig auch stark von der Wahl der Parameter ab. Beispielsweise kann die Wahl der ersten Iterationsschritte große Auswirkungen auf den weiteren Iterationsverlauf haben. Deshalb kann der Löser bereits bei unterschiedlicher Konfiguration der Hesse-Matrix stark abweichende Ergebnisse liefern.
Dieser Umstand kann mit Hilfe der Mehrkernschnittstelle teilweise verbessert werden. Mit dem *Best-Of-All*-Modus kann der Benutzer eine Vielzahl unterschiedlicher Probleminstanzen mit variierenden Startschätzungen oder Parameterkonfigurationen vom Optimierer lösen lassen und am Ende das beste erreichte Ergebnis verwenden. In diesem Fall werden alle Arbeiterinstanzen bis zur jeweiligen Terminierung fortgeführt. Somit stellt es auch kein Problem dar, wenn mehr Instanzen gestartet werden als parallele Threads zur Verfügung stehen.
Das modulare Design der Mehrkernschnittstelle erlaubt es dem Benutzer weitere Anwendungsmöglichkeiten passend zu seinen Anforderungen zu ergänzen. So wird zur Zeit beispielsweise eine Schnittstelle zur Lösung von mehrkriteriellen Optimierungsproblemen implementiert. Analog dazu ist es auch denkbar Methoden zur Lösung von Gemischt-Ganzzahligen Optimierungsproblemen zu entwickeln und die dabei resultierenden nichtlinearen Optimierungsprobleme in parallel, mit Hilfe des WMCI zu lösen.

6.3.4 Parameterindividualisierungsmodus

Die Verwendung eines NLP-Lösers ist für den Benutzer häufig selbst ein iterativer Prozess. Nachdem eine erste Optimierung erfolgt ist, bewertet der Anwender das Ergebnis. Abhängig vom erzielten Ergebnis werden anschließend kleine Anpassungen der Modellierung vorgenommen oder die Modellfunktionen, wie Zielfunktion und Nebenbedingungen, verfeinert. Mit diesen Änderungen wird die Optimierung erneut durchgeführt und das Ergebnis wieder bewertet. Dieser Vorgang kann beliebig oft wiederholt werden.
Insgesamt wird im beschriebenen Fall der Optimierer mehrfach zur Lösung sehr ähnlicher Probleme verwendet. Diese Überlegung motiviert die Implementierung eines *Parameterindividualisierungsmodus* für den Löser, bezogen auf die konkrete

Aufgabenstellung.
Für die Mehrkernschnittstelle wurde ein spezieller Modus entwickelt, der dem Benutzer das händische Anpassen der Löserparameter abnimmt. Mit Hilfe der bereits beschriebenen automatischen Codeerzeugung können ohne großen Aufwand zahlreiche verschiedene Parametersetzungen definiert werden. Die vorgefertigte Sammlung möglicher Konfigurationen zur Variation ist im Anhang in Abschnitt A.1 zu finden. Auch in diesem Fall werden unter Ausnutzung der parallelen Architektur alle erzeugten Instanzen des Problems individuell gelöst. Anschließend kann anhand der Ergebnisse ein Parametersatz gefunden werden, der es dem Optimierer erlaubt das Problem so schnell wie möglich zu lösen. Alternativ bietet die Ausgabe des Modus auch die beste gefunden Lösung und die dabei verwendete Parametersetzung an, so dass der Benutzer diese für seine oben beschriebene Entwicklungsschleife verwenden kann. Auf Grund der Ähnlichkeit der zu lösenden Probleme innerhalb dieser Schleife, wird die so identifizierte Parametersetzung die Leistungsfähigkeit des Lösers verbessern.
Wassel beschreibt in seiner Arbeit [109] einen alternativen Ansatz zur Anpassung der Parameter des Lösers. Er entwickelt ein Skript, dass es ermöglicht eine ganze Sammlung von Problemen in parallel laufenden Prozessen zu lösen und anhand der Ergebnisse die gewählte Parametersetzung zu bewerten. Mit Hilfe der dort beschriebenen Mechanik wurden die aktuell gültigen Standardparameter für WORHP identifiziert.
Der Parameterindividualisierungsmodus zielt im Gegensatz dazu auf die Anpassung des Optimierers für ein konkretes Problem des Nutzers ab und kann bei Verwendung der Mehrkernschnittstelle problemlos aktiviert werden. Auf diese Weise kann eine derartige Funktionalität für alle Anwender (auch ohne Vorkenntnisse) zur Verfügung gestellt werden.

6.4 Parallele Reverse-Communication

Die Verwendung mehrerer Zielfunktions- oder Nebenbedingungsauswertungen in einem Durchlauf der Reverse-Communication-Schleife (vergleiche Abbildung 6.2) erfordert einerseits eine Anpassung im Programmcode des Benutzers und andererseits innerhalb von WORHP. Im Folgenden werden die nötigen Anpassungen der Schleife und mögliche Strategien zur Effizienzsteigerung des Optimierungsalgorithmus durch diese Erweiterung diskutiert.
Die Idee mehrere Auswertungen in parallel anzufordern wird unter anderem von Schnabel [94] diskutiert und wurde beispielsweise in NLPQLP von Schittkowski [93] umgesetzt. Im Rahmen dieses Kapitels werden kurz die nötigen Änderungen innerhalb von WORHP beschrieben und mögliche Verwendungsmöglichkeiten vorgestellt.

6.4.1 Implementierung

Innerhalb der Reverse-Communication-Schleife müssen die Auswertungsanforderungen um eine Funktionalität ergänzt werden, für verschiedene Vektoren x die Zielfunktion oder die Nebenbedingungen auswerten zu können. Weiterhin müssen diese Anforderungen dem Benutzercode kommuniziert werden.
Die Beauftragung benötigter Funktionsauswertungen erfolgt über boolesche Variablen in der `Control`-Struktur. Damit der Benutzer in seinem Code die erforderlichen Auswertungen für verschiedene x abarbeiten kann, mussten zunächst zwei zusätzliche Zähler ergänzt werden. Mit Hilfe von `evalFNumberRequests` kann die Anzahl der angeforderten Zielfunktionsauswertungen angegeben werden und analog gibt `evalGNumberRequests` dies für die Nebenbedingungen an.

```
// Sequential version
if (GetUserAction(&cnt, evalF))
{
    opt->F = wsp->ScaleObj *
        (X[0] * X[0] + 2.0 * X[1] * X[1] - X[2]);
    DoneUserAction(&cnt, evalF);
}
// Parallel version
if (GetUserAction(&c, evalF)) {
    // Determine number of requested objective evaluatons
    // TODO for User: Parallelise the following loop
    for (int k = 0; k < c.evalFNumberRequests; ++k) {
        o.multiF[k] = w.ScaleObj * (
            o.X[0 + k*o.n]*o.X[0 + k*o.n] -
            o.X[1 + k*o.n]*o.X[1 + k*o.n] );
    }
    /* ------------------------------------------- */
    /* This block is necessary in order to         */
    /*  not break API to single threaded user codes  */
    o.F = o.multiF[0]; // Just set the first objective to old o.F
    /* ------------------------------------------- */
    DoneUserAction(&c, evalF);
}
```

Listing 6.2: Nötige Änderungen für parallele Auswertungen der Zielfunktion innerhalb der Reverse-Communication

Ein Entwicklungskriterium ist, dass ein Schleifendurchlauf bei Verwendung mehrfacher Auswertungsanforderungen, abzüglich des Mehraufwandes für die Verwaltung der Threads, idealerweise nicht länger dauert als bei Verwendung nur eines Threads. Deshalb werden vom Optimierungscode nicht mehr Auswertungen angefordert als Threads zur Verfügung stehen. Auf diese Weise vereinheitlicht und vereinfacht sich auch die initiale Reservierung des Speichers. Der Benutzer legt vor dem ersten Aufruf von WORHP die Anzahl der zu verwendenden Threads innerhalb der `OptVar`-Struktur über die Variable `numberThreads` fest. Für die Auswertungen

muss in diesem Zusammenhang Speicherplatz für die zu verwendenden x-Stellen und die Werte von Zielfunktion und Nebenbedingungen reserviert werden. Anhand des Wertes `numberThreads` stehen WORHP alle Informationen zur Verfügung um ausreichend Speicher zu reservieren. Die Berechnung der angeforderten Werte wickelt der Benutzer anschließend selbst ab und kann dabei in parallel die Anzahl Threads verwenden, die ihm zur Verfügung steht. Demnach erfordert eine Umstellung von einem Thread auf mehrere Threads, dass der Benutzer innerhalb der Reverse-Communication-Schleife `numberThreads` definiert und seine Zielfunktions- und Nebenbedingungsauswertungen anpasst. Es sei mahnend angemerkt, dass die Anpassung der Auswertungen je nach implementierten Funktionen des Nutzers mit erheblichem Implementierungsaufwand verbunden sein kann, um die parallelen Aufrufe fehlerfrei durchführen zu können.
Im Codeausschnitt 6.2 ist die Implementierung der Zielfunktion $x_1^2 + 2x_2^2 - x_3$ in der sequentiellen und parallelen Version aus Sicht des Benutzers abgedruckt. Die Berücksichtigung der verschiedenen x-Werte verkompliziert den Programmcode bereits für dieses akademische Beispiel beträchtlich. Durch eine geschickte Implementierung kann dieser Umstand offensichtlich vereinfacht werden. Allerdings birgt die sequentielle Speicherung der verschiedenen x-Werte trotzdem zusätzliches Fehlerpotenzial.

6.4.2 Liniensuche

Im Abschnitt 3.2.2 wurde ausführlich die Liniensuche innerhalb von WORHP mittels Filter oder Bestrafungsfunktion diskutiert. Ohne Parallelisierung werden Zielfunktion und Nebenbedingungen in Iteration k ausgewertet und es erfolgt eine Bewertung des durchgeführten Schritts, dieser wird anschließend akzeptiert oder der Schrittweitenparameter $\alpha^{[k]}$ reduziert.
Steht die Möglichkeit zur Verfügung mehrere Zielfunktionswerte gleichzeitig anzufordern kann die Liniensuche verbessert werden. Es können direkt mehrere Schrittweiten mit Parametern $\alpha_j^{[k]}$ entsprechend Gleichung (3.79) getestet werden. Auf Grund von Nichtlinearitäten ist es denkbar, dass ein $\alpha_l^{[k]}$ im Sinne des Filters oder der Bestrafungsfunktion besser ist als ein anderes $\alpha_m^{[k]}$ mit $l > m$. Soll der Löser sich unabhängig von der verwendeten Anzahl Threads identisch verhalten, muss die größte Schrittweite, die der Liniensuchalgorithmus akzeptiert, verwendet werden. Es könnte aber auch positive Effekte haben, die im Sinne der gewählten Kriterien beste Schrittweite zu verwenden.
Die Ausführungen zur Verwendung der Sensitivitätsableitungen bezüglich Regularisierung in Abschnitt 5.3 und Relaxierung in Abschnitt 5.4 eröffnen eine alternative Nutzungsmöglichkeit der Parallelität innerhalb der Liniensuche. Nikolayzik korrigiert in seiner Arbeit [81] die Suchrichtung mit Hilfe der Sensitivitäten bezüglich dieser Größen und führt anschließend eine Liniensuche mit den so berechneten Suchrichtungen durch. In seinen numerischen Untersuchungen hat sich diese Vor-

gehensweise als experimentell und häufig zu optimistisch herausgestellt. Im Gegensatz zur Umsetzung mit nur einer Funktionsauswertung je Reverse-Communication-Schleifendurchlauf bietet die parallele Anforderung verschiedener Funktionswerte die Möglichkeit den dadurch entstehenden zusätzlichen Aufwand zu kompensieren und diese Suchrichtungen zu berücksichtigen.
Die bezüglich der angewendeten Regularisierung der Hesse-Matrix bereinigte Suchrichtung kann durch die Taylor-Approximation

$$d_{\text{reg}} = d(\kappa) \approx d(\kappa_0) + \frac{\mathrm{d}d}{\mathrm{d}\kappa}(\kappa_0)(\kappa - \kappa_0) \tag{6.1}$$

bestimmt werden. Die Berechnung des Wertes κ erfolgt mit Hilfe der in Abschnitt 5.3.2 beschriebenen Techniken. Analog kann die bezüglich der Relaxierung bereinigte Suchrichtung über die Beziehung

$$d_{\text{relax}} = d(\bar{\eta}) \approx d(\eta^{[j]}) + \frac{\mathrm{d}d}{\mathrm{d}\eta}(\eta^{[j]})(\bar{\eta} - \eta^{[j]}) \tag{6.2}$$

ermittelt werden. In diesem Zusammenhang wird der Wert $\bar{\eta}$ wie im Abschnitt 5.4.2 über Gleichung (5.71) festgesetzt. Insgesamt stehen somit neben der direkt aus dem Unterproblem ermittelten Suchrichtung d noch die beiden korrigierten Suchrichtungen d_{reg} und d_{relax} zur Verfügung, insofern die zur Berechnung benötigten Sensitivitätsmodi aktiviert wurden.
Eine gründliche Suche, das heißt es werden alle Schrittweiten für alle Suchrichtungen getestet, entlang der drei zur Verfügung stehenden Richtungen d, d_{reg} und d_{relax} erfordert die dreifache Anzahl Zielfunktions- und Nebenbedingungsauswerten. Demnach kann der zeitliche Aufwand mit mindestens drei Threads gegenüber einer nicht parallelen Implementierung kompensiert werden.

6.4.3 Weitere Ansätze

Die vorgestellten Techniken stellen nur einen Bruchteil der Verwendungsmöglichkeiten paralleler Funktionsauswertungen während der Optimierung dar. Der Fokus dieser Arbeit liegt auf der Parallelisierung mit Hilfe der Mehrkernschnittstelle. Deshalb werden an dieser Stelle nur kurz einige weitere Strategien, welche parallele Funktionsauswertungen ausnutzen, vorgestellt, um einen Überblick über die Möglichkeiten zu schaffen.
Naheliegend ist eine Parallelisierung der Ableitungsberechnung. Werden finite Differenzen zur Berechnung der Ableitungsmatrizen verwendet, liegt es nahe die Möglichkeit der parallelen Funktionsauswertungen auszunutzen. Werden diese Ansätze mit der im Abschnitt 3.3.2 vorgestellten Gruppenstrategie kombiniert, kann die Berechnung weiter verbessert werden. Bei der Verwendung der Reverse-Communication-Strategie könnte dieser Ansatz beispielsweise über eine Warteschlangenfunktionalität realisiert werden. Der Löser übergibt dem Benutzer

eine Liste von benötigten Funktionsauswertungen und dieser arbeitet die Anforderungen unter Ausnutzung der Nebenläufigkeit so schnell wie möglich ab. Dies steht insofern im Widerspruch zu den oben genannten Ansätzen, dass bei den finiten Differenzen höchstwahrscheinlich mehr Auswertungen angefordert werden müssen, als Threads zur Verfügung stehen. Die oben getroffenen Annahmen bezüglich der Speicherverwaltung sind somit in diesem Fall zu restriktiv und eine alternative Vorgehensweise ist erforderlich.
Schnabel [94] schlägt vor während der Liniensuche auf Verdacht bereits Funktionsauswertungen zur Verwendung innerhalb der finiten Differenzen zu berechnen. Es sei $N_T \leq N_x$ die Anzahl zur Verfügung stehender Threads und $h > 0$ die Schrittweite zur Berechnung der finiten Differenzen. Bei der Anforderung der Auswertung von $f(x^{[k]} + \alpha^{[k]} d^{[k]})$ werden auch die gestörten Werte $f(x^{[k]} + \alpha^{[k]} d^{[k]} + he_i)$ mit $i = 1, ..., N_T - 1$ berechnet. Diese stehen anschließend bei Akzeptanz des Schritts direkt zur Verfügung und die benötigte Zeit zur Berechnung der Ableitungen kann reduziert werden. Insbesondere lässt sich diese Technik mit der parallelisierten Berechnung der Ableitungsmatrizen kombinieren. Diese Vorgehensweise steht aber in Konkurrenz zur vorgestellten Strategie unterschiedliche Schrittweiten oder Suchrichtungen gleichzeitig zu testen.
Bei Verwendung einer Filterstrategie innerhalb der Liniensuche könnte es auch interessant sein Strategien zu entwickeln die Filterfront (Vergleiche Abbildung 3.2) in parallel aufzubauen. Hierzu könnten beispielsweise erneut die alternativen Suchrichtungen d_{reg} und d_{relax} verwendet werden.

6.5 Parallele Lineare Algebra

Die Lösung nichtlinearer Optimierungsprobleme reduziert sich im Endeffekt auf die Lösung von linearen Gleichungssystemen während der Iterationen des Unterproblems, wie in Abschnitt 3.1.2.2 beschrieben. Eine Laufzeitanalyse des Optimierungsalgorithmus, zum Beispiel in der Masterarbeit des Autors [41], offenbart, dass der Großteil der Rechenkapazitäten zur Lösung dieser Gleichungssysteme aufgebracht werden muss. Es liegt demnach nahe, die Parallelisierung hier anzusetzen. Standardmäßig wird innerhalb von Worhp der lineare Löser MA97 verwendet. Dieser bietet die Möglichkeit mittels Parallelisierung die Rechenlast auf mehrere Prozessorkerne aufzuteilen.
In den numerischen Untersuchungen wird diese Parallelisierungsstrategie getestet und kurz analysiert. Da es sich bei dem linearen Löser innerhalb von Worhp aber um Software von Dritten handelt wird in dieser Arbeit nicht näher auf die verschiedenen Strategien zur Parallelisierung in diesem Zusammenhang eingegangen. In dem Artikel von Hogg und Scott [65] werden Anhand der Löser MA77, MA86, MA87 und MA97 Parallelisierungsmöglichkeiten vorgestellt und Laufzeitanalysen durchgeführt. Weitere Informationen finden sich auch in der dort zitierten Literatur.

Kapitel 7

Numerische Ergebnisse

Inhaltsangabe

Im Verlauf dieser Ausarbeitung wurde eine Reihe von Strategien zur Effizienzsteigerung nichtlinearer Optimierung vorgeschlagen. Nachdem bisher die mathematisch theoretischen Hintergründe und die algorithmischen Details im Mittelpunkt standen, thematisiert dieses Kapitel praktische Untersuchungen der Algorithmen sowohl an einzelnen Testbeispielen, als auch anhand einer großen Sammlung von Problemen. Dieser Untersuchungen helfen die Stärken und Schwächen der verschiedenen Anpassungen des Algorithmus herauszuarbeiten.
Abschnitt 7.1 führt zunächst die Testumgebung für die numerischen Untersuchungen ein und stellt anschließend kurz die Sammlung von Testproblemen vor. Daraufhin werden im Abschnitt 7.2 Bewertungskriterien für die Änderungen des Algorithmus diskutiert. Eine Bewertung der im Laufe der Arbeit beschriebenen Strategien anhand numerischer Auswertungen findet im Abschnitt 7.3 statt.

7.1 Testumgebung des Lösers

Die Validierung und Verifizierung neu- oder weiter entwickelter Algorithmen zur Lösung hochdimensionaler Optimierungsprobleme erfordert es beobachtete Ergebnisse sowohl quantitativ, als auch qualitativ bewerten zu können. Die folgenden

Untersuchungen konzentrieren sich auf die verschiedenen Erweiterungen des Algorithmus. Dabei werden teilweise neu eingeführte Parameter anhand kurzer Studien untersucht, um die Wahl der Standardwerte für diese zu erklären.
Wassel [109] beschreibt in Abschnitt 4.3 seiner Arbeit die Infrastruktur zur Durchführung von Tests des Lösers WORHP. Er entwirft ein Skript, welches es den Entwicklern ermöglicht, parallel verschiedene Prozessinstanzen von WORHP zu starten. Dieses speichert die Ergebnisse in Tabellenform, welche für anschließende Auswertungen verwendbar sind.
Zunächst wurde bei der Entwicklung von WORHP die CUTEr-Testsammlung (A **c**onstrained and **u**nconstrained **t**esting **e**nvironment, **r**evisited) von Gould, Orban und Toint [57] und die COPS-Testsammlung (a large-scale **C**onstrained **O**ptimization **P**roblem **S**et) von Dolan, Moré und Munson [27] verwendet. Gould, Orban und Toint haben ihre Sammlung um neue, insbesondere größere, Probleme erweitert und die Implementierung überarbeitet, so dass parallele Modellauswertungen möglich sind. Eine Schnittstelle zur neuen Testumgebung CUTEst (a **C**onstrained and **U**nconstrained **T**esting **E**nvironment with **s**afe **t**hreads for Mathematical Optimization) [59] wurde im Rahmen dieser Arbeit entwickelt und zur Bewertung der neuen Implementierungen verwendet. Die Anbindung der alten CUTEr-Testsammlung erfolgt weiterhin über die AMPL-Schnittstelle von WORHP. Es wurden zwei SIF-Schnittstellen (standard input format - Standardeingabeformat) für WORHP entwickelt. Der erste Schritt war die Umsetzung einer nativen FORTRAN-Schnittstelle. Die Implementierung der Mehrkernschnittstelle erforderte zusätzlich eine C++-Schnittstelle zur CUTEst-Testsammlung, um eine nahtlose Integration in die C++-Objektstruktur zu realisieren.
In der CUTEst-Testmenge sind Probleme verschiedenster Dimensionen und Eigenschaften enthalten. Die Sammlung enthält sowohl Aufgabenstellungen aus konkreten praktischen Anwendungen, als auch rein akademische Probleme, welche entworfen wurden um Probleme für klassische Lösungsverfahren zu verursachen. Es sind beispielsweise große Teile der Sammlung von Hock und Schittkowksi [64] und Schittkowski [92] enthalten.
Die kleinsten Probleme sind *bqp1var* mit nur 1 Optimierungsvariable mit Boxschranke und das Problem *hs1* mit 2 Optimierungsvariablen mit einer Boxschranke. Wohingegen auch sehr große Probleme wie zum Beispiel *bdry2* mit 251001 Optimierungsvariablen und 250498 Nebenbedingungen vorhanden sind. Insgesamt enthält die Testsammlung in der aktuell vorliegenden Version 1257 Beispiele. In den folgenden numerischen Untersuchungen wird eine zeitliche Begrenzung von 30 Minuten zur Lösung eines Problems angesetzt. Innerhalb dieser Zeit ist WORHP mit keiner Konfiguration in der Lage die Beispiele *five20b, five20c, lubrif, lubrifc* und *twod* zu lösen, deshalb werden diese aus der Testsammlung entfernt. Weiterhin verursacht die Auswertung der Nebenbedingungen des Beispiels *hs67* in einigen Fällen eine Endlosschleife und das Beispiel wird deshalb ebenfalls entfernt. Nach diesen Reduktionen verbleiben 1251 Probleme in der Testmenge. Bei 891 Beispielen handelt es sich um beschränkte Probleme mit Nebenbedingungen.

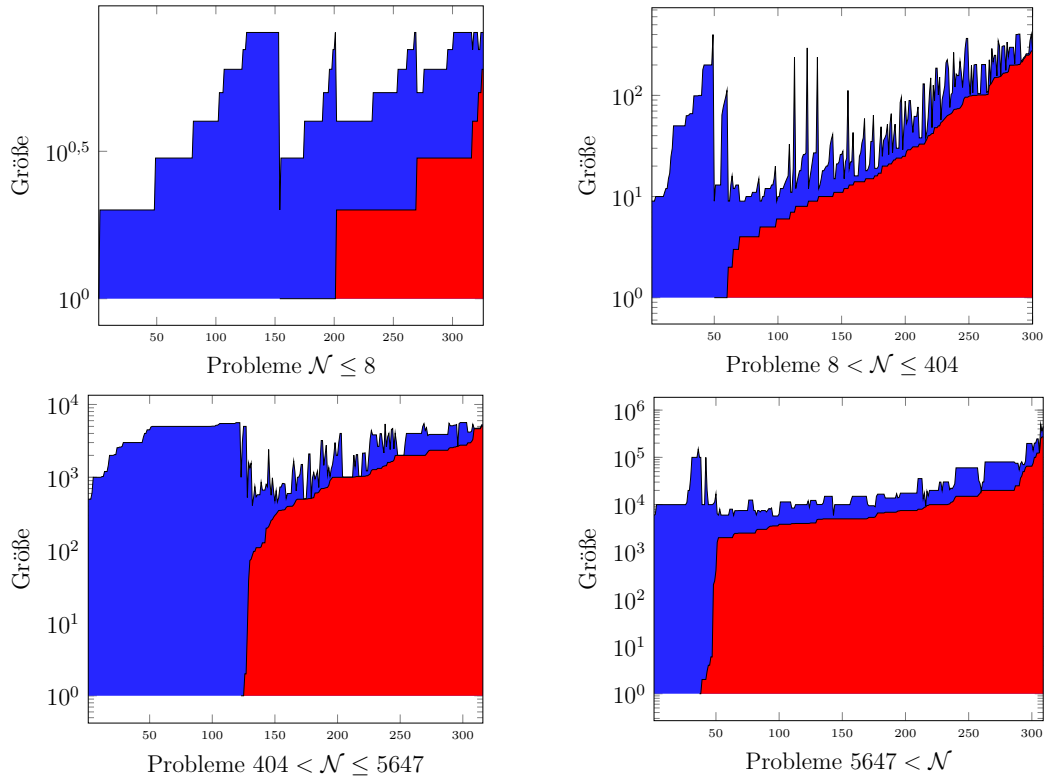


Abbildung 7.1: Größen der KKT-Matrizen der CUTEst-Beispiele aufgeteilt in die vier Hauptquantile. Die blauen Anteile der Darstellungen repräsentieren die Anzahl der Optimierungsvariablen N_x, wohingegen rot die Anzahl der Nebenbedingungen $N_g + N_h$ darstellt.

In der numerischen Umsetzung eines Optimierungsalgorithmus ist die Lösung der linearen Gleichungssysteme zur Bestimmung der Suchrichtung in vielen Fällen der rechenintensivste Teil, dies belegen zum Beispiel Benchmarks mit Profiling-Informationen in der Arbeit des Autors [41]. Deshalb wird entsprechend der Betrachtungen in Abschnitt 3.1.2.2 die Dimension $\mathcal{N} := N_x + N_h + N_g$ des linearen Gleichungssystems verwendet um die Testsammlung in die vier Standardquantile aufzuteilen. In diesem Fall liegen die Grenzwerte bei $\mathcal{N}_{0.25} = 8$, $\mathcal{N}_{0.5} = 404$ und $\mathcal{N}_{0.75} = 5647$. In Abbildung 7.1 sind die Größen der enthaltenen Beispiele der vier Quantile grafisch aufgetragen.
Neben der Größe des zu lösenden linearen Gleichungssystems kann die Schwierigkeit eines Optimierungsproblems durch den Grad der Nichtlinearität in Zielfunktion und Nebenbedingungen bewertet werden. Analog zur Abbildung 4.3 aus der Arbeit von Wassel [109] für die CUTEr-Testsammlung zeigt Abbildung 7.2 die Aufteilung der Zielfunktions- und Nebenbedingungsarten innerhalb der CUTEst-Testsammlung.

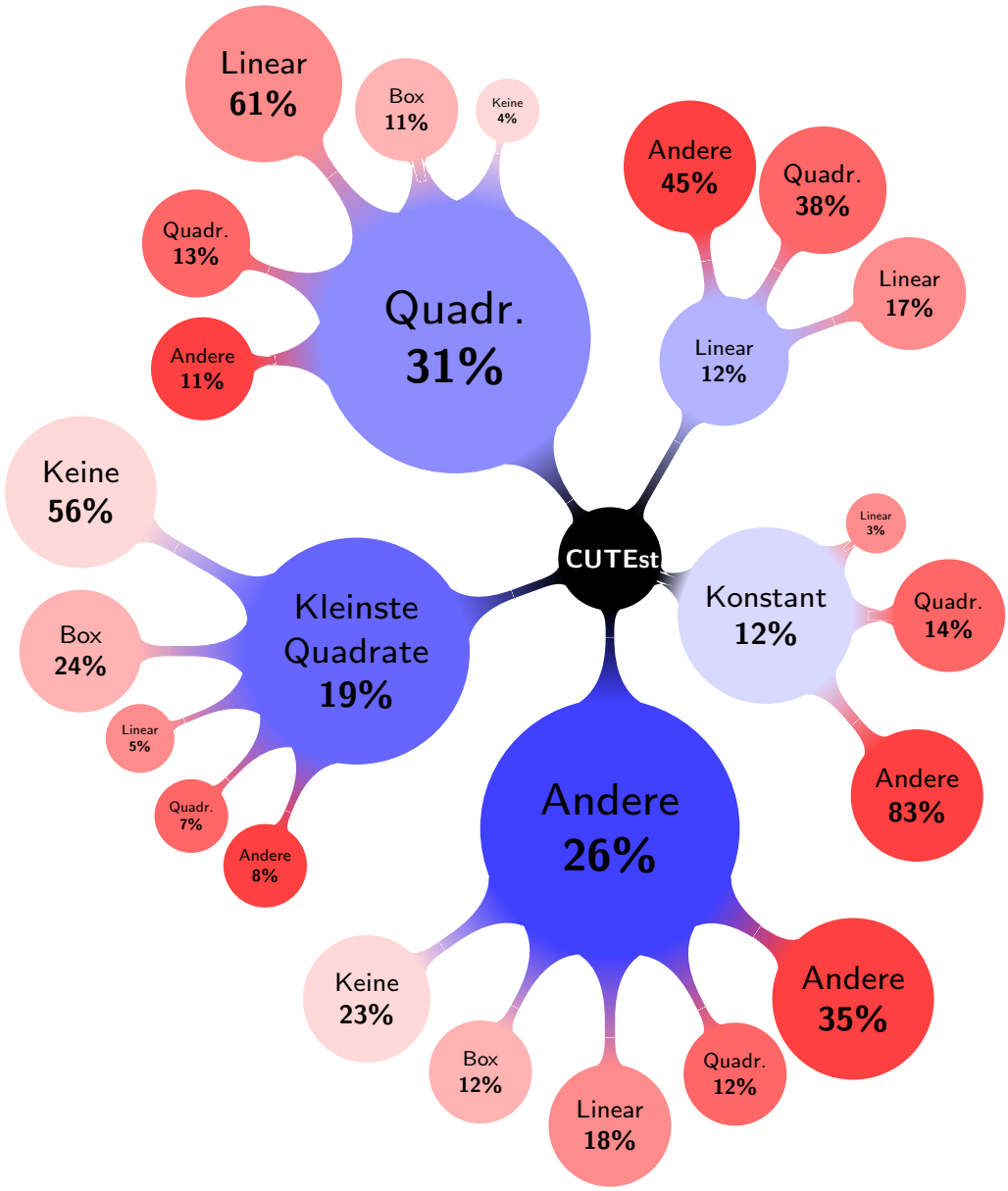


Abbildung 7.2: Klassifikationen der CUTEst-Probleme. Die inneren bläulichen Knoten bilden die Art der Zielfunktion ab und die rötlichen äußeren Knoten die Anteile der auftretenden Nebenbedingungen bei Problemen der jeweiligen Zielfunktionsart.

7.2 Messbarkeit der Effizienzsteigerung

Die Effizienz eines Optimierungsverfahrens lässt sich in erster Linie durch die benötigte Zeit zum Auffinden eines (lokalen) Minimums bestimmen. Die benötigte Zeit für einen Durchlauf des Algorithmus ist allerdings auch von externen Faktoren wie der allgemeinen Rechenlast des verwendeten Computers abhängig. Deshalb bietet es sich an, bei Veränderungen des Algorithmus neben der Zeit alternative Kriterien zur Bewertung der Effizienz zu verwenden.

Anhand der allgemeinen Beschreibung des SQP-Verfahrens ergeben sich im Prinzip zwei weitere konkrete Möglichkeiten die Effizienz zu messen. Zur Lösung von Beispielen mit numerisch aufwändigen Nebenbedingungen und Zielfunktionen ist das Ziel die Anzahl der Auswertungen dieser Modellfunktionen zu reduzieren. Insbesondere sind bei derartigen Beispielen häufig auch die Auswertungen der benötigten Ableitungen aufwändig oder erfordern eventuell sogar die Verwendung finiter-Differenzen, so dass die Anzahl der Modellfunktionsauswertung noch deutlich stärker ansteigt. Die Anzahl der Ableitungsauswertungen steht in direktem Zusammenhang mit der Anzahl der Hauptiterationen. Als gutes Maß für die Effizienz des SQP-Algorithmus kann dementsprechend für derartige Modelle die Anzahl der benötigten Hauptiterationen verwendet werden. Zur Untersuchung einer Verbesserung der Liniensuche dient hingegen die Anzahl der Funktionsauswertungen als passendes Maß.
Eine alternative Klasse von Problemen ist über die Größe der zu lösenden KKT-Matrix entsprechend Gleichung (3.62) definiert. Der Aufwand der Lösung der resultierenden linearen Gleichungssysteme ist abhängig von der Anzahl der Nichtnulleinträge der KKT-Matrix und wie bereits erwähnt, benötigt WORHP in diesem Fall die meiste Rechenzeit zur Lösung des linearen Gleichungssystems. Wie die Betrachtungen des Innere-Punkte-Verfahrens für quadratische Optimierungsprobleme in Abschnitt 3.1.2.2 gezeigt haben, muss in jeder Iteration des QP-Algorithmus eine neue Faktorisierung der KKT-Matrix berechnet werden. Somit liefert die Anzahl der benötigten Nebeniterationen für solche Modelle ein geeignetes Maß für die Effizienz des SQP-Algorithmus.

7.3 Auswertung

Nach der Vorstellung der Testumgebung und der zu verwendenden Kriterien zur Beurteilung der neu entwickelten Techniken, thematisiert dieser Abschnitt die verschiedenen Auswertungen. Abschnitt 7.3.1 beschäftigt sich zunächst mit jenen Algorithmen, die im Zusammenhang mit der Behandlung der Nebenbedingungen stehen. Anschließend folgen in Abschnitt 7.3.2 die verschiedenen Techniken, welche Einfluss auf die verwendete Hesse-Matrix nehmen. Zuletzt werden in Abschnitt 7.3.3 die verschiedenen Parallelisierungstechniken angewendet und deren Ergebnisse ausgewertet.
Für die numerischen Untersuchungen wurde ein *Intel(R) Xeon(R) CPU E5-4617 0 @ 2.90GHz* mit 24 logischen Kernen und *512 GB RAM* verwendet. Auf dem verwendeten System lief während der Tests *Ubuntu 16.04*.
Die Konfiguration von WORHP erfolgt über eine Vielzahl von Parametern. Die zu testenden Algorithmen können durch verschiedene dieser Parameter aktiviert und individualisiert werden. Im Anhang B sind in diversen Tabellen die Standardeinstellungen abgedruckt. Die Beschreibung der numerischen Versuche in diesem Abschnitt enthält jeweils eine kurze Erklärung der angepassten Parameter.
Lediglich die Parallelisierung über die Mehrkernschnittstelle erfordert eine alternati-

ve Anbindung des Lösers und ist deshalb nicht durch Parametersetzungen steuerbar. Abschnitt 7.3.3 beschreibt die dort vorgenommenen Einstellungen gesondert.
Zur Visualisierung der Ergebnisse dienen einerseits anschauliche Balkenprofile zum direkten Vergleich zweier Methoden. Diese Art der Darstellung stammt aus der Arbeit von Curtis [23]. Andererseits werden in erster Linie Performance-Profile verwendet. Diese bieten eine gute Möglichkeit verschiedene Strategien miteinander zu vergleichen. Die Idee der Profile beschreiben Dolan und Moré [26]. Diese vergleichen bezüglich einer Metrik, wie zum Beispiel Berechnungszeit, Iterationen oder Funktionsauswertungen, die erzielten Ergebnisse der verschiedenen Läufe mit dem besten Ergebnis für jedes konkrete Beispiel. Es sei der Vergleichswert

$$t_{p,k} := \text{Vergleichswert für Beispiel } p \text{ für Konfiguration } k \tag{7.1}$$

gegeben. Weiterhin seien alle verfügbaren Konfigurationen durch die Menge K beschrieben und es sei das Performance-Verhältnis

$$r_{p,k} := \frac{t_{p,k}}{\min\{t_{p,k} : k \in K\}} \tag{7.2}$$

definiert. Zuletzt sei noch die Sammlung von Problemen durch P beschrieben. Dann berechnet sich über die Beziehung

$$\rho_k(\tau_p) := \frac{1}{|P|} \left(\sum_{r_{p,k} \leq \tau_p} r_{p,k} \right)$$

die Wahrscheinlichkeitsverteilung für das Performance-Verhältnis. In den folgenden Abbildungen wird zur besseren Übersicht als x-Achse τ_p logarithmisch aufgetragen. Diese Profile minimieren die Auswirkungen von Ausreißern oder vereinzelten fehlgeschlagenen Optimierungen auf den Gesamtvergleich. Zum besseren Verständnis werden jeweils im Zusammenhang die wichtigsten Beobachtungen aus den Abbildungen konkret erläutert.
Auf Grund der Normierung aller Konfigurationen bezüglich des jeweils besten Optimierungslaufes für jedes Beispiel können teilweise Schwierigkeiten beim Vergleich von mehr als zwei Einstellungen in einem Bild auftreten. Gould und Scott beschreiben in ihrem Artikel [60], dass es unter Umständen in diesen Situationen nicht möglich ist, Aussagen über die qualitative Reihenfolge der schlechteren Konfigurationen zu treffen. In den folgenden Auswertungen trat dieser Effekt aber nicht störend auf. Zusätzlich zu den Performance-Profilen werden außerdem verschiedene Tabellen zur Bewertung der Ergebnisse angeführt, um die Aussagen der Profile zu untermauern.

7.3.1 Strategien für die Nebenbedingungen

Im Laufe dieser Ausarbeitung wurden diverse Strategien entwickelt, um die Behandlung der Nebenbedingungen innerhalb eines SQP-Verfahrens zu verbessern. Der eine

Teil dieser Techniken beschäftigt sich in den Abschnitten 3.3.3.3 und 5.4 mit der Verbesserung der Relaxierung der Nebenbedingungen. Weiterhin diskutiert Abschnitt 5.2 die Möglichkeit der Einbindung einer Echtzeitkorrektur der Nebenbedingungen während des Verlaufes des SQP-Verfahrens. Dieser Abschnitt bewertet die Algorithmen anhand von numerischen Versuchen.
Für die folgenden Untersuchungen werden als Testprobleme alle Beispiele mit Nebenbedingungen aus der CUTEst-Testsammlung verwendet. Somit stehen insgesamt 891 Beispiele zur Auswertung zur Verfügung.

7.3.1.1 Relaxierung der Nebenbedingungen

Die Relaxierung weicht die Nebenbedingungen der nichtlinearen Problemformulierung auf. Dies ermöglicht beispielsweise die Beseitigung von Inkonsistenzen auf Grund der Linearisierung der Nebenbedingungen auf Unterproblemebene. Es sei kurz an die Motivation durch die Beispiele 3.27, 3.28 und 3.30 erinnert. Innerhalb der CUTEst-Testsammlung sind auch Beispiele mit linearen Nebenbedingungen enthalten, so dass im Prinzip eine Relaxierung nicht notwendig ist. Aktuell bietet WORHP keine Möglichkeit diese Information zu verarbeiten und auszunutzen, deshalb wurde an dieser Stelle bewusst darauf verzichtet diese Probleme aus den Vergleichstests zu entfernen.

7.3.1.1.1 Adaptive Relaxierung Die relevanten Parameter bezüglich der Relaxierung der Nebenbedingungen sind in Tabelle B.9 abgedruckt. In der Grundeinstellung ist die adaptive Relaxierung deaktiviert und es gibt genau eine Relaxierungsvariable $N_\delta \in \mathbb{R}$ für alle Nebenbedingungen. Die adaptive Relaxierung zeichnet sich durch ein adaptives $N_\delta \in [1, N_g+N_h]$, wie in Algorithmus 8 beschrieben, aus. Als Vergleichskonfiguration wurde zusätzlich jede Nebenbedingung mit einem eigenen Relaxierungsparameter versehen. Demnach gilt $N_\delta = N_g + N_h$. Abschnitt 5.4.4 schlägt zusätzlich vor, nur noch einen Bestrafungsparameter η zu verwenden. Ein Vergleich dieser Konfiguration mit den anderen Einstellungen findet ebenfalls statt.
In Tabelle 7.1 sind die Ergebnisse der verschiedenen Durchläufe im Vergleich abgedruckt. In der Beschreibung der Relaxierung wurde verdeutlicht, dass der Löser durch Verwendung vieler Relaxierungsvariablen zusätzliche Flexibilität bei der Behandlung der Nebenbedingungen erhält. Die Daten zeigen, dass die Standardvariante mit $N_\delta = 1$ am wenigsten Beispiele bis zur optimalen Toleranz lösen kann. Bereits in der Theorie hatte sich aber gezeigt, dass die Relaxierung die Anzahl der Optimierungsvariablen innerhalb der Unterprobleme um N_δ erhöht, so dass der Rechenaufwand zur Lösung der Gleichungssysteme zunimmt. Diese Situation motivierte die Einführung der adaptiven Relaxierung mit variablem $N_\delta \in [1, N_g + N_h]$. Die Ergebnisse in Tabelle 7.1 belegen den Erfolg der adaptiven Variante. Die Konfiguration löst 765 Probleme im Gegensatz zu 758 Beispielen optimal. Gleichzeitig

verursachen 18 Beispiele einen Timeout-Fehler, wohingegen die Varianten mit einer Relaxierungsvariablen für jede Nebenbedingung sogar bei 21 Beispielen eine Zeitüberschreitung verursachen. Im Vergleich dazu tritt dies in lediglich 11 Fällen für die Standardvariante auf. Die Auswirkungen auf das resultierende Minimum der verschiedenen Optimierungen fallen nicht stark aus. Bei 84% der gelösten Beispiele finden alle Varianten dasselbe lokale Minimum.
Der gestiegene Rechenaufwand für die Varianten mit $N_\delta > 1$ äußert sich in der erhöhten Rechenzeit. Die Daten in der Tabelle zeigen, dass die Gesamtrechenzeit[1] von 16 Stunden und 34 Minuten für die Standardeinstellung auf 20 Stunden und 12 Minuten für die adaptive Relaxierung steigt. Die beiden Varianten mit $N_\delta = N_g + N_h$ benötigen sogar 22 Stunden und 16 beziehungsweise 18 Minuten. Dieser gestiegene Zeitaufwand ist der Preis, den der Benutzer bereit sein muss zu zahlen, um von der beschriebenen gesteigerten Robustheit zu profitieren. Die adaptive Relaxierung stellt einen gelungenen Kompromiss zwischen zusätzlichem Zeitaufwand und gewonnener Robustheit dar.
Die Ergebnisse zeigen weiterhin, dass es nicht sinnvoll ist im Fall $N_\delta = N_g + N_h$ jeder Relaxierungsvariablen einen eigenen Bestrafungswert zuzuordnen. Die Auswertung belegt somit, dass die Vereinfachung $\eta \in \mathbb{R}$ für diesen Fall nicht einschränkend ist und im folgenden wird nur noch der Fall $\eta \in \mathbb{R}$ behandelt. Abbildung 7.3 ver-

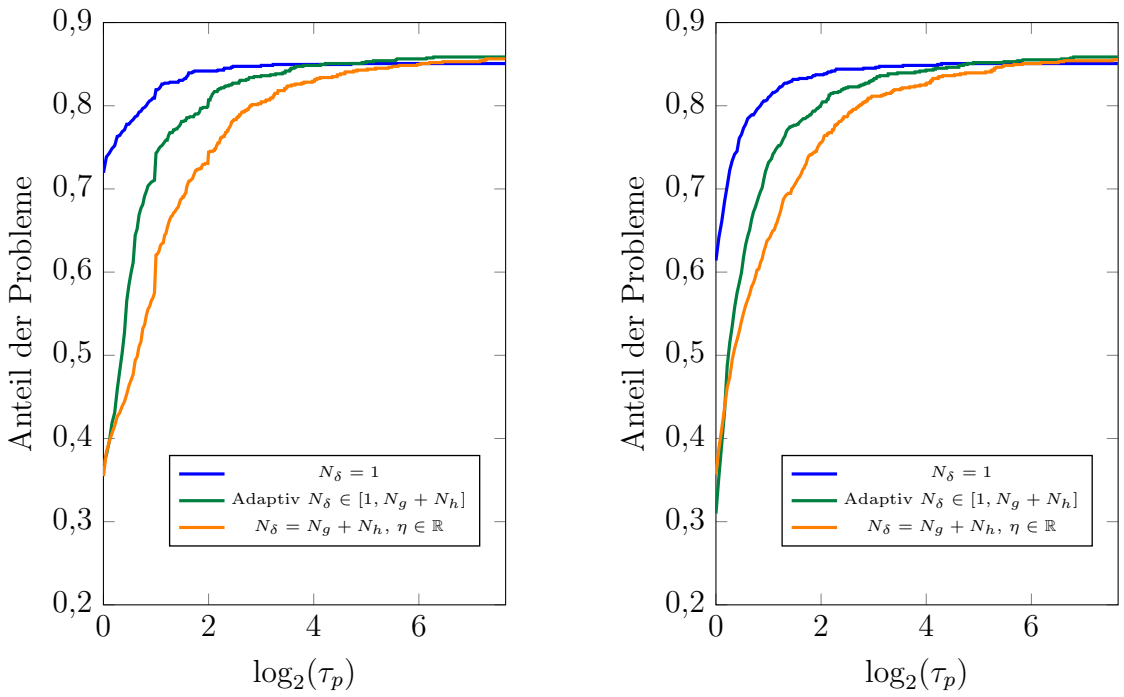
	$N_\delta = 1$		Adaptives $N_\delta \in [1, N_g + N_h]$		$N_\delta = N_g + N_h$ $\eta \in \mathbb{R}^{N_\delta}$		$N_\delta = N_g + N_h$ $\eta \in \mathbb{R}$	
Optimal	758	85,07%	**765**	**85,86%**	763	85,63%	763	85,63%
Akzeptabel	**22**	**2,47%**	19	2,13%	19	2,13%	19	2,13%
Misserfolg	111	12,46%	**107**	**12,01%**	109	12,23%	109	12,23%
(Timeout)	(10)		(18)		(21)		(21)	
Dauer	**16h 34m 26s**		20h 12m 52s		22h 16m 38s		22h 18m 48s	

Tabelle 7.1: Ergebnisse von WORHP für verschiedene Konfigurationen der Anzahl der Relaxierungsvariablen und der dazugehörigen Bestrafungsparameter

gleicht die drei verbleibenden Varianten in zwei Performance-Profilen bezüglich der Rechenzeit und der benötigten Nebeniterationen auf Unterproblemebene miteinander. Bei der Betrachtung der verschiedenen Relaxierungsvarianten ist es sinnvoll die Nebeniterationen zu betrachten, weil die Relaxierung darauf abzielt problematische linearisierte Nebenbedingungen auf dieser Ebene aufzuweichen und so das Lösungsverhalten der quadratischen Hilfsprobleme zu verbessern.
Die Grafik bestätigt die bisherigen Beobachtungen. Die Standardvariante mit $N_\delta = 1$ ist bei 69% der Probleme die schnellste Konfiguration, wohingegen die adaptive Relaxierung nur für 38,72% der Beispiele besser ist. Die Variante mit $N_\delta = N_g + N_h$ ist in 37,15% ($\eta \in \mathbb{R}$) der Beispiele am Schnellsten. Hierbei ist zu beachten, dass auf

[1] Die Rechenzeiten innerhalb der Analyse der verschiedenen Konfigurationen wurden mit Hilfe des Linux-Befehls *time* über die *user time* ermittelt.

Grund einer Rundung der Rechenzeit verschiedene Konfigurationen die gleiche Zeit benötigen können, so dass die Summe deshalb nicht 100% ergeben muss. Die adaptive Relaxierung benötigt in der Regel etwas mehr Zeit als die Standardvariante, ist dafür aber in der Lage mehr Probleme zu lösen. Dies gelingt deutlich schneller, als wenn jede Nebenbedingung mit einer eigenen Relaxierungsvariablen versehen wird, wie die beiden übrigen Kurven zeigen. Wenn genügend Zeit zur Verfügung steht, ist auch die Konfiguration mit einer Relaxierungsvariablen für jede Nebenbedingung in der Lage mehr Beispiele als die Standardeinstellung zu lösen.
Das zweite Profil, basierend auf den benötigten Nebeniterationen, stützt die bis-



(a) CPU-Zeit

(b) Nebeniterationen

Abbildung 7.3: Performance-Profile für verschiedene Anzahlen an Relaxierungsvariablen im Vergleich

herigen Aussagen. Zusätzlich ist noch zu erkennen, dass unter diesem Kriterium die Standardvariante nur noch in 61,39% der Fälle am besten ist. Verglichen damit ist die adaptive Relaxierung für 30,98% der Beispiele die beste Variante. Im Vergleich Berechnungszeit zu Nebeniterationen verlieren sowohl Standardvariante, als auch adaptive Relaxierung an Effizienz, während der Fall $N_\delta = N_g + N_h$ nahezu unverändert in 35,69% der Problemfälle die beste Wahl darstellt. Dieses Ergebnis ist zu erwarten, da die Standardeinstellung und die adaptive Relaxierung darauf abzielen die Rechenzeit im Vergleich geringer zu halten als die Relaxierung aller Nebenbedingungen, welche dafür alle Nebenbedingungen gleichzeitig entkoppelt voneinander aufweicht. Diese Aufweichung kann das Unterproblem zum Teil erheblich vereinfachen, erfordert aber auf Grund der erhöhten Systemgröße inner-

halb des Innere-Punkte-Verfahrens mehr Rechenzeit.
Ein typischer Optimierungsverlauf für WORHP benötigt in den ersten Iterationen

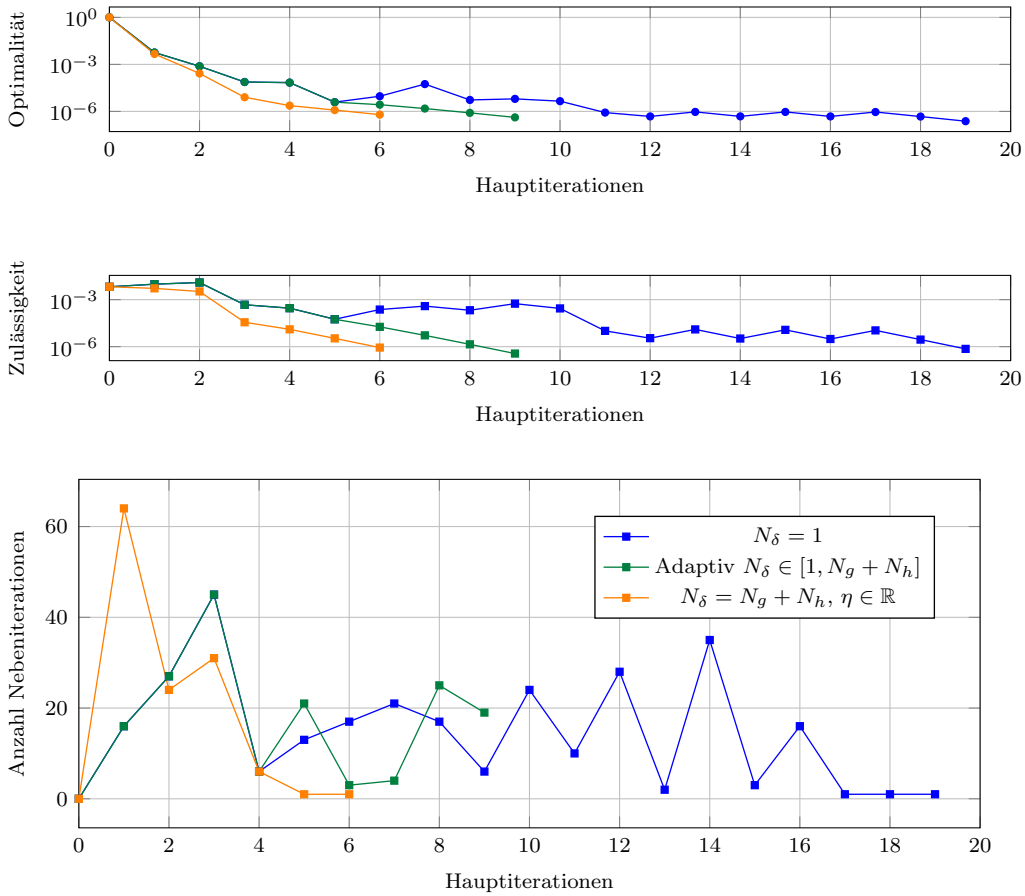


Abbildung 7.4: Optimalität, Zulässigkeit und Anzahl Nebeniterationen während des Optimierungsverlaufes für das Beispiel *twirimd1*

viele Nebeniterationen zur Lösung der Unterprobleme und bei Annäherung an ein lokales Minimum stabilisiert sich das Verhalten des Unterproblemlösers. Im weiteren Verlauf sind dann häufig nur sehr wenige Iterationen innerhalb des Innere-Punkte-Verfahrens zur Lösung der quadratischen Unterprobleme nötig. Die Relaxierung der Nebenbedingungen wurde eingeführt, um die Behandlung der linearen Nebenbedingungen im Unterproblem zu verbessern. Das gewünschte Verhalten der adaptiven Relaxierung im Vergleich zur einfachen Relaxierung und der Relaxierung aller Nebenbedingungen für sich lässt sich gut bei Betrachtung des Beispiels *twirimd1* aus der CUTEst-Sammlung beobachten. Das Beispiel hat 1247 Optimierungsvariablen und 712 Nebenbedingungen. Es handelt sich um ein relaxiertes nichtlineares ganzzahliges Optimierungsproblem basierend auf der Simulation des Ladevorgangs eines Nuklearreaktors. Die Zielfunktion ist lediglich linear, aber die Nebenbedingungen sind nichtlinear und die Ableitungen nicht regulär. In Abbildung 7.4 sind die Optimalität, die Zulässigkeit der Iterierten und die jeweils benötigte Anzahl an Nebenite-

rationen abgedruckt. Zur Lösung des Beispiels benötigt die Standardvariante 19 Iterationen, die adaptive Relaxierung 9 Iterationen und die Variante mit $N_\delta = N_g + N_h$ sogar nur 6 Iterationen. Von besonderem Interesse im hier betrachteten Zusammenhang ist die Anzahl der benötigten Nebeniterationen in Abbildung 7.4. Die Variante mit vielen Relaxierungsvariablen, dargestellt in orange, benötigt zwar für die erste Iteration 64 Nebeniterationen, ist aber in der Lage mit Hilfe der zusätzlichen Freiheitsgrade einen besseren Optimierungsverlauf zu initiieren. Innerhalb der Iterationen 3 bis 6 tritt dann bereits die typische Stabilisierung in den Nebeniterationszahlen auf. Beim Vergleich der grünen und blauen Kurve ist gut erkennbar, dass in der 5. Iteration erstmalig die adaptive Relaxierung aktiviert wird und die Anzahl der Relaxierungsvariablen verändert wird. Mit Hilfe dieser Veränderung ist dann auch die adaptive Relaxierung in der Lage in den Iterationen 6 bis 9 das Minimum zu erreichen, allerdings stellt sich hier die Stabilisierung der Nebeniterationszahlen nicht im gewünschten Maße ein. Im Vergleich dazu zeigt sich bei der Standardvariante bis zur Iteration 17, dass die entstehenden Unterprobleme den Algorithmus stärker fordern und erst in den letzten 3 Iterationen (17 bis 19) stellt sich das stabile Verhalten in der Nähe des lokalen Minimums ein.
Das Beispiel verdeutlicht die Idee, dass die adaptive Relaxierung einen Kompromiss zwischen der Standardvariante und der Relaxierung aller Nebenbedingungen darstellt. In diesem konkreten Fall benötigt die Standardvariante etwa 18 Sekunden, die adaptive Relaxierung 9,7 Sekunden und der Optimierungslauf mit $N_\delta = N_g + N_h$ 9 Sekunden. Abhängig von der Anzahl der Nebenbedingungen kann bei ähnlichem Verhalten aber die adaptive Relaxierung in Bezug auf die Rechenzeit die bessere Wahl darstellen.

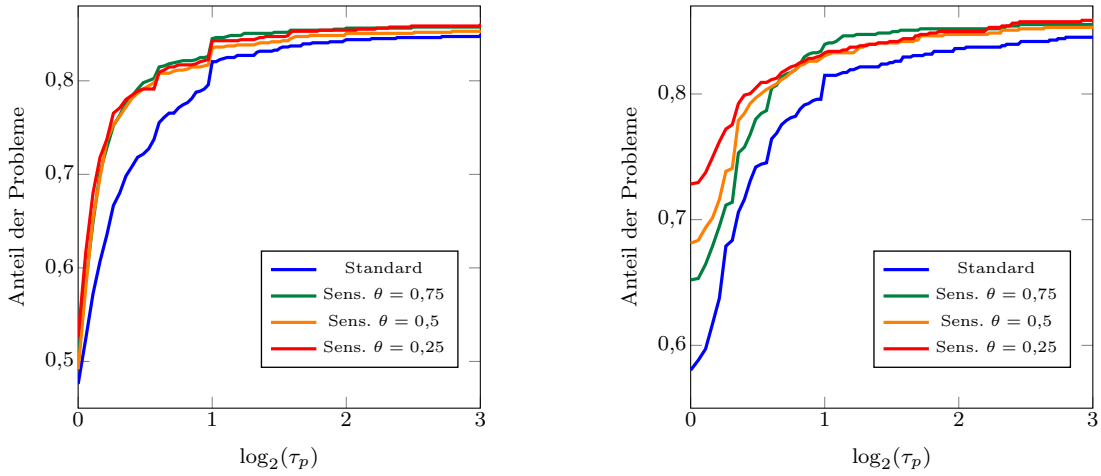
7.3.1.1.2 Sensitivitätsbasierte Anpassung des Relaxierungsparameter

Neben der Einführung einer adaptiven Anzahl Relaxierungsvariablen wurden im Abschnitt 5.4 Ideen vorgestellt mit Hilfe der parametrischen Sensitivitätsanalyse die Relaxierung innerhalb des SQP-Verfahrens zu verbessern. Im Folgenden wird die sensitivitätsbasierte Anpassung des Bestrafungsparameters η der Relaxierung analysiert.
In der Standardvariante des innerhalb von WORHP implementierten Relaxierungsalgorithmus (vergleiche Algorithmus 6) wird der Bestrafungswert durch Multiplikation mit einem festen Wert angepasst. Im Gegensatz dazu verwendet die sensitivitätsbasierte Variante Gleichung (5.71), um Sensitivitätsinformationen des Unterproblems zu verwenden und so die Anpassung des Bestrafungswertes adaptiv zu gestalten. In den folgenden Auswertungen wird die neue Methode mit verschiedenen Werten für den Parameter θ getestet, um ein Gefühl für die Auswirkung des Parameters zu vermitteln. Es sei kurz daran erinnert, dass kleinere Werte von θ einer stärkeren Erhöhung des Bestrafungsparameters entsprechen. Die Parameter zur Konfiguration der sensitivitätsbasierten Anpassung des Relaxierungsparameters sind im Anhang in Tabelle B.8 zusammen mit ihren Standardsetzungen einzusehen.

	Standard		Sens. $\theta = 0{,}75$		Sens. $\theta = 0{,}5$		Sens. $\theta = 0{,}25$	
Optimal	758	85,07%	765	85,86%	763	85,63%	**767**	**86,08%**
Akzeptabel	22	2,47%	**25**	**2,81%**	23	2,58%	23	2,58%
Misserfolg	111	12,46%	**101**	**11,34%**	105	11,78%	**101**	**11,34%**
Dauer	16h 34m 26s		15h 45m 53s		**15h 27m 27s**		15h 42m 17s	

Tabelle 7.2: Ergebnisse von WORHP mit sensitivitätsbasierter Relaxierung im Vergleich zur Standardvariante

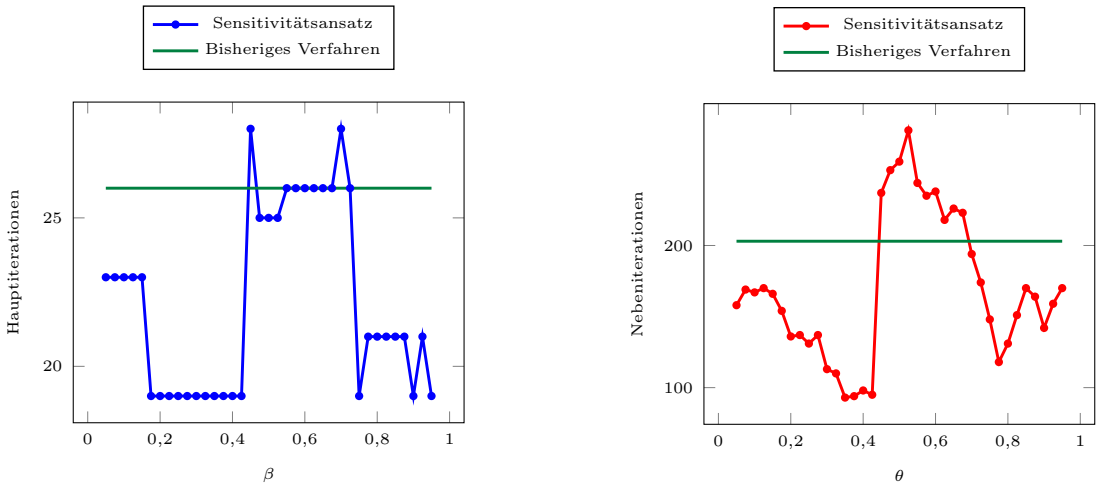
In Tabelle 7.2 sind die Gesamtergebnisse auf den beschränkten Beispielen aus der CUTEst-Sammlung abgedruckt. Die Standardvariante benötigt 16 Stunden und 34 Minuten zur Lösung dieser Probleme. Im Vergleich dazu sind die sensitivitätsbasierten Varianten schneller. Mit der Einstellung $\theta = 0{,}5$ kann WORHP die Probleme in 15 Stunden und 27 Minuten lösen. Die anderen beiden Einstellungen benötigen auch nur 15 Minuten ($\theta = 0{,}25$) beziehungsweise 18 Minuten ($\theta = 0{,}75$) länger. In Anbetracht der Gesamtzeit und störenden Faktoren in der Serverlast können die Zeiten der sensitivitätsbasierten Varianten als gleichwertig betrachtet werden und sind alle schneller als die Standardvariante. Als weiterer positiver Effekt ist zu beobachten, dass die Anzahl der optimal gelösten Probleme von 758 mit der klassischen Einstellung auf 763 ($\theta = 0{,}5$), 765 ($\theta = 0{,}75$) und bis zu 767 ($\theta = 0{,}25$) erhöht werden kann. Die hier verglichenen Konfigurationen liefern für ungefähr 92% der Beispiele dasselbe lokale Minimum.
Zum Vergleich der drei sensitivitätsbasierten Varianten untereinander sind in Abbildung 7.5 die vier Konfigurationen in zwei Performance-Profilen bezüglich Zeit und Nebeniterationen aufgetragen. Auch in diesem Zusammenhang ist eine Betrachtung der Nebeniterationen sinnvoller, als der Hauptiterationen, weil eine bessere Wahl des Bestrafungsparameters für weniger Schleifendurchläufe in Algorithmus 13 sorgen sollte. Die Iterationen jedes Aufrufes des Unterproblemlösers werden zusammengezählt, somit sorgen weniger Schleifendurchläufe für eine geringere Anzahl Nebeniterationen.
In beiden Profilen ist deutlich zu erkennen, dass die sensitivitätsbasierte Anpassung des Relaxierungsparameters der klassischen Variante überlegen ist. Der neue Algorithmus ist für alle drei Wahlen von θ sowohl schneller, als auch robuster. Der Vergleich der verschiedenen Wahlen von θ untereinander ist nicht so deutlich. Wie bereits oben erwähnt, fällt die Interpretation der CPU-Zeit für die drei nahe beieinander liegenden Kurven schwer, weil die beobachteten Abweichungen im Rahmen der Messgenauigkeit auf Grund allgemeiner Prozessorlast während der Tests liegen. Bezüglich der Nebeniterationen ist die Konfiguration ($\theta = 0{,}25$) den anderen aber überlegen. Eine Kombination der Beobachtungen des Profiles bezüglich der Nebeniterationen und der erzielten Ergebnisse in Tabelle 7.2 ergibt somit ($\theta = 0{,}25$) als beste Wahl für den Parameter.
Die Auswertung der Variation von θ auf der gesamten Testsammlung vermittelt



(a) CPU-Zeit (b) Nebeniterationen

Abbildung 7.5: Performance-Profile für verschiedene sensitivitätsbasierte Relaxierungskonfigurationen im Vergleich

einen guten Eindruck wie sich die Änderung des Parameters auf eine Vielzahl von Beispielen verteilt auswirkt. Zum besseren Verständnis der Methode soll zusätzlich anhand eines ausgewählten Beispiels die Wirkung des Parameters beurteilt werden. Das Beispiel *lobsterz* entsteht im Zuge einer Parameteridentifikation für die Modellierung von Spannungen entlang von Zellmembranen. Die hier verwendete Diskretisierung resultiert in 16263 Optimierungsvariablen mit 16243 Nebenbedingungen. Die Zielfunktion des Problems ist quadratisch und die Nebenbedingungen sind nichtlinear. In Abbildung 7.6 wurde $\theta \in [0{,}05; 0{,}95]$ variiert und die benötigten Haupt- und Nebeniterationszahlen der resultierenden Optimierungsläufe abgedruckt. Die grüne Linie repräsentiert die Anzahl Iterationen für das Standardverfahren ohne Ausnutzung der parametrischen Sensitivitätsanalyse. Für dieses konkrete Beispiel zeigt sich, dass der Bereich $\theta \in [0{,}4; 0{,}7]$ zu problematischem Verhalten führt. Die ersichtlichen Schwankungen sind typisch für Eingriffe innerhalb des SQP-Verfahrens. Auf Grund der lokalen Eigenschaften des Optimierungsverfahrens ist es möglich, dass Veränderungen am Algorithmus zu verschiedenen Wegen zum Optimum führen. Diese führen wiederum auf stark abweichende Iterationsverläufe und auf Grund der Nichtlinearität innerhalb der Probleme kann dies sehr unterschiedliche Auswirkungen auf die beobachtete Effizienzsteigerung haben. Das Beispiel verdeutlicht die Problematik eine gute Einstellung für θ zu bestimmen. Gleichzeitig ist zu betonen, dass die Betrachtung einzelner Beispiele auf Grund der beschriebenen Problematik schlicht ungeeignet ist, um die Qualität eines Verfahrens beurteilen zu können. Stattdessen sind die bereits aufgeführten Performance-Profile basierend auf einer hinreichend großen Testbasis zu Rate zu ziehen.
Zur Abrundung der Untersuchung wird im nächsten Schritt analysiert, wie sich die sensitivitätsbasierte Anpassung des Bestrafungsparameters der Relaxierung in Kombination mit der bereits betrachteten adaptiven Relaxierung verhält. In Abbildung



(a) Hauptiterationen

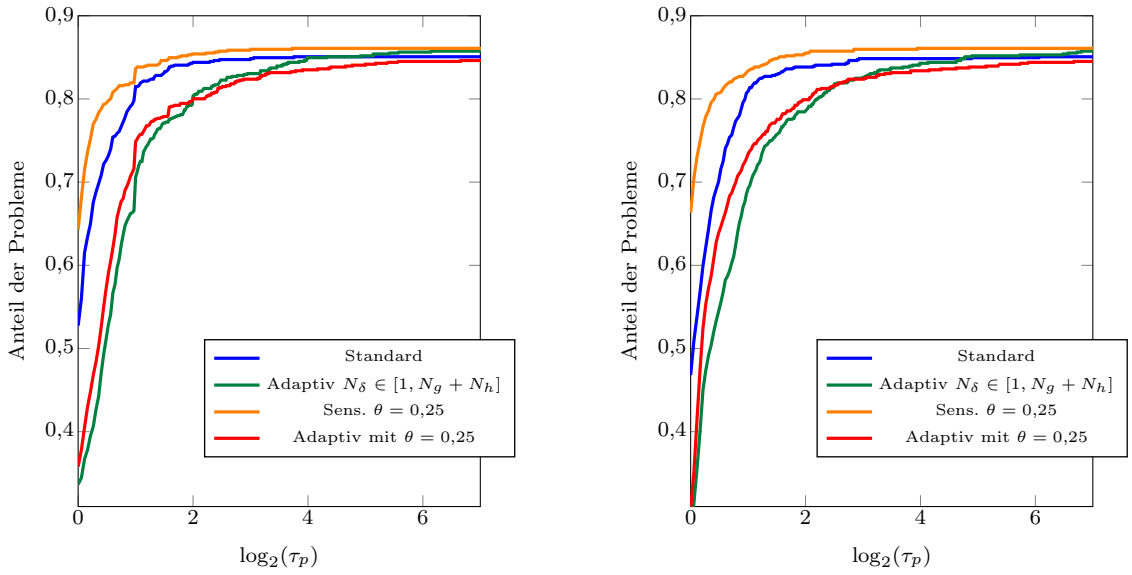
(b) Nebeniterationen

Abbildung 7.6: Variation des Parameters θ für das Problem (*lobsterz*). Die blaue Kurve zeigt die Anzahl der resultierenden Hauptiterationen an und die rote Kurve die entsprechende Anzahl der benötigten Nebeniterationen.

7.7 sind die Performance-Profile der Standardvariante (in blau), der Variante mit $\theta = 0{,}25$ (in orange), der adaptiven Relaxierung (in grün) und der adaptiven Relaxierung mit $\theta = 0{,}25$ (in rot) abgedruckt.
Die Performance-Profile bestätigen die bisherigen Eindrücke. Die sensitivitätsbasierte Anpassung ist in der Lage auch den Optimierungslauf mit adaptiver Relaxierung zu beschleunigen. Trotzdem bleiben aber die Standardvariante und die sensitivitätsbasierte Version ohne adaptive Relaxierung bezüglich Geschwindigkeit die bessere Wahl. Es wird weiterhin deutlich, dass die Kombination von adaptiver Relaxierung und sensitiviätsbasierter Relaxierung weniger robust ist als der Grundalgorithmus der adaptiven Relaxierung. Es werden zwar 22 statt 19 Beispiele akzeptabel gelöst, aber im Gegensatz dazu werden lediglich 756 im Vergleich zu 765 Beispielen optimal gelöst. Auch in diesem Fall sind die Auswirkungen auf die gefundenen lokalen Minima eher gering. Die verschiedenen Konfigurationen finden bei 88% der Beispiele dasselbe lokale Minimum. Insgesamt zeigen die Profile und diese Zahlen, dass die sensitivitätsbasierte Anpassung in Kombination mit der adaptiven Relaxierung eine Einbußung an Robustheit zur Folge hat, dafür aber eine leichte Geschwindigkeitssteigerung verbuchen kann.

7.3.1.2 Zulässigkeitskorrektur

Die Durchführung einer Echtzeitkorrektur zur Verbesserung der Einhaltung der nichtlinearen Nebenbedingungen während des SQP-Verfahrens wurde im Abschnitt 5.2 ausführlich diskutiert und es wurden diverse Kriterien und Spezialisierungen des Algorithmus vorgestellt. In diesem Abschnitt werden die verschiedenen Varianten



(a) CPU-Zeit

(b) Nebeniterationen

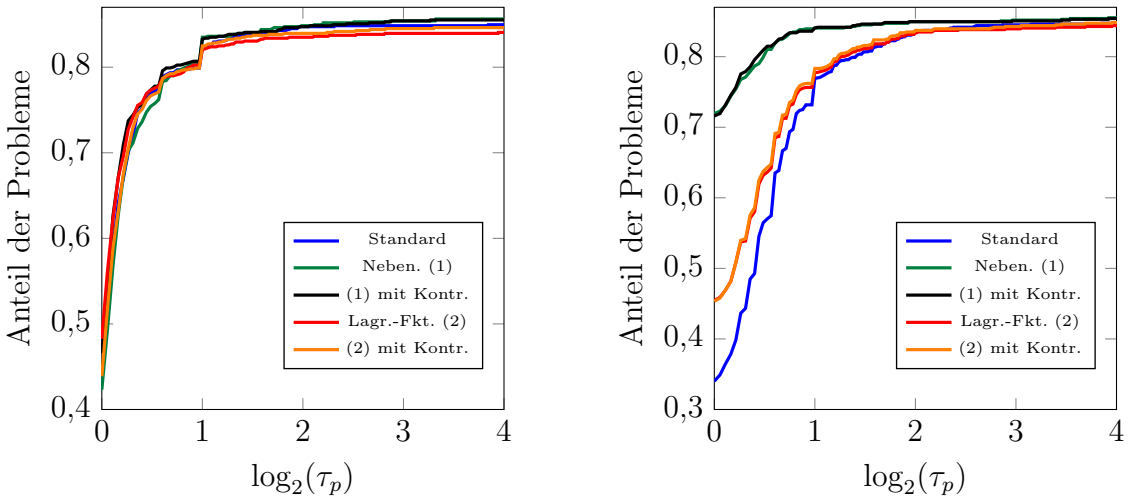
Abbildung 7.7: Performance-Profile für adaptive und sensitivitätsbasierte Relaxierung im Vergleich

ausgewertet und mit dem normalen SQP-Verfahren, sowie untereinander verglichen. Die relevanten Parameter zur Konfiguration der Zulässigkeitskorrektur sind in Tabelle B.7 im Anhang abgedruckt. Es sei kurz angemerkt, dass die folgenden Untersuchungen mit aktivierter Steffensen-Extrapolation durchgeführt wurden. Diese wird im weiteren Verlauf separat analysiert.
Während der Beschreibungen des Algorithmus wurde ein besonderes Augenmerk auf die Formulierung geeigneter Abbruchbedingungen gelegt. Naheliegend ist die Verwendung der Nebenbedingungsverletzung und deren Entwicklung während der Korrektur aus Abschnitt 5.2.4.2. In Anlehnung an Bewertungsfunktionen wurde weiterhin die Lagrange-Funktion als Kriterium in Abschnitt 5.2.4.3 eingeführt. Beide Varianten können durch eine Überwachung der Kontraktivität der Iterationsvorschrift, wie in Abschnitt 5.2.4.1 beschrieben, ergänzt werden. Eine Auswertung für die Beispiele mit Nebenbedingungen innerhalb der CUTEst-Sammlung ist in Tabelle 7.3 abgedruckt. Die besten Ergebnisse erzielen in diesem Vergleich die Varianten der Zulässigkeitskorrektur mit Verwendung der Nebenbedingungen als Abbruchkriterium. Die Probleme wurden am schnellsten mit der Korrektur und unter Berücksichtigung des Kontraktionskriterium in 15 Stunden und 30 Minuten, im Vergleich zu den bekannten 16 Stunden und 34 Minuten der Standardkonfiguration, gelöst. Ohne Berücksichtigung des Kontraktionsfaktors wird ein zusätzliches Beispiel optimal gelöst.
Wird die Lagrange-Funktion als Kriterium verwendet ist das Verfahren nicht in der Lage mit der Standardkonfiguration zu konkurrieren. Der Zeitaufwand ist mit 16

	Standard		Nebenbed.		Nebenbed. (mit Kontrakt.)		Lagrange-Fkt.		Lagrange-Fkt. (mit Kontrakt.)	
Optimal	758	85,07%	**764**	**85,75%**	763	85,63%	752	84,40%	756	84,85%
Akzeptabel	**22**	**2,47%**	17	1,91%	18	2,02%	21	2,36%	20	2,24%
Misserfolg	111	12,46%	**110**	**12,35%**	**110**	**12,35%**	118	13,24%	115	12,91%
Dauer	16h 34m 26s		16h 17m 50s		**15h 30m 40s**		16h 1m 2s		16h 41m 2s	

Tabelle 7.3: Ergebnisse von WORHP mit Zulässigkeitskorrektur und verschiedenen Abbruchbedingungen im Vergleich zur Standardvariante

Stunden und 1 Minute ohne Betrachtung des Kontraktionsfaktors zwar geringer als bei Verwendung der Standardkonfiguration, aber dafür werden lediglich 752 Beispiele optimal gelöst. Durch Berücksichtigung des Kontraktionsfaktors wird diese Variante zwar robuster und löst 756 Beispiele, ist dafür aber wieder langsamer als die Standardeinstellung.
In diesem Zusammenhang ist es interessant anzumerken, dass auch bei Verwendung dieser Strategien für 92% der Beispiele das gefundene lokale Minimum nicht abweicht. Zur anschaulichen Betrachtung der Ergebnisse sind in Abbildung 7.8 die



(a) CPU-Zeit (b) Hauptiterationen

Abbildung 7.8: Performance-Profile für unterschiedliche Abbruchbedingungen der Zulässigkeitskorrektur innerhalb des SQP-Verfahrens

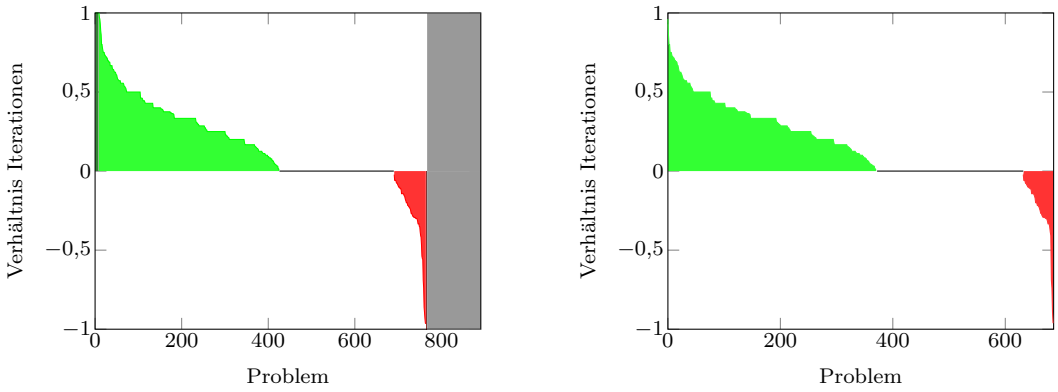
Performance-Profile bezüglich Zeit und Hauptiterationen für die verschiedenen Abbruchkriterien der Zulässigkeitskorrektur im Vergleich abgebildet. Die Daten in Tabelle 7.3 haben bereits gezeigt, dass die zeitlichen Unterschiede der Methoden gering ausfallen. Umso interessanter ist aber das Profil basierend auf den Hauptiterationszahlen. Die Abbildung 7.8b verdeutlicht, dass alle Konfigurationen der Zulässigkeitskorrektur in der Lage sind die Anzahl der benötigten Hauptiterationen im Vergleich zur Standardvariante teilweise deutlich zu reduzieren. Das Profil zeigt

weiterhin, dass die Betrachtung des Kontraktionsfaktors zu vernachlässigen ist und auf der Gesamtheit der Testbeispiele lediglich leicht positive Auswirkungen zeigt.
Besonders auffällig sind in Abbildung 7.8 die unterschiedlichen Verläufe bei Betrachtung von Zeit beziehungsweise Hauptiterationen im direkten Vergleich. Die Beschreibungen zur effizienten Berechnung des Korrekturterms in Abschnitt 5.2.1 haben gezeigt, dass sich der Ansatz der Zulässigkeitskorrektur darauf reduziert aus einer Zerlegung der KKT-Matrix mehrere Schritte zu berechnen und diese anhand von Nebenbedingungsauswertungen, und Zielfunktionsauswertungen bei Verwendung der Lagrange-Funktion, zu bewerten. Diese zusätzlichen Schritte sollen helfen die Anzahl der benötigten Hauptiterationen des Optimierungsverfahrens zu reduzieren. Durch eine Reduktion der Iterationsanzahl wird gleichzeitig die Menge der erforderlichen Ableitungsauswertungen verringert, da diese in jeder Iteration durchgeführt werden müssen, um das Unterproblem aufzustellen.
Bosse und Griewank [9] zeigen, dass die Auswertung der Ableitungen innerhalb der CUTEst-Beispiele auf Grund der speziellen Struktur der Probleme nur wenig zusätzlichen Rechenaufwand im Vergleich zur Auswertung der Zielfunktion und Nebenbedingungen benötigt. Dieser Umstand erklärt, warum sich die beobachtete Einsparung an Hauptiterationen in den Auswertungen nicht in der gesparten Zeit widerspiegelt.
Das Profil bezüglich der Hauptiterationen in Abbildung 7.8b belegt aber, dass die Korrektur die gewünschte Wirkung auf den Iterationsverlauf hat.
Einen besseren Eindruck über das Einsparpotential liefert Abbildung 7.9. In der Abbildung sind die Einsparungen an Iterationen für jedes Beispiel relativ zur besseren Lösung als Balken aufgetragen. Die Höhe der Balken $H_{p,k}$ berechnet sich für jedes Beispiel mit Hilfe des Performance-Verhältnisses aus Gleichung (7.2) durch

$$H_{p,k} := 1 - r_{p,k}$$

und wird als positiver Wert in grün aufgetragen, wenn der Korrekturalgorithmus weniger Iterationen benötigt, und negativ in rot, wenn die Standardvariante besser ist. Jeweils in dunkleren Tönen sind Balken eingezeichnet, die nur von einer der beiden Methoden gelöst wurden und in grau werden die Beispiele eingezeichnet, bei denen beide Varianten fehlschlagen.
In Abbildung 7.9a wurden alle Beispiele aus der Sammlung berücksichtigt. Bereits bei den Betrachtungen zur Variation des Parameters θ innerhalb der sensitivitätsbasierten Relaxierung wurde deutlich, dass auf Grund von Nichtlinearität innerhalb der Problemfunktionen stark unterschiedliche Optimierungsverläufe beobachtet werden können, auch wenn nur kleine Änderungen am Algorithmus vorgenommen werden. Damit derartige Effekte beim Vergleich von Zulässigkeitskorrektur und Standardvariante vermieden werden können, wurden in Abbildung 7.9b lediglich jene Optimierungsläufe berücksichtigt, deren resultierende Zielfunktion relativ um maximal 2% abweicht.
Im linken Graphen sind insgesamt 758 Probleme von wenigstens einer der beiden Varianten optimal gelöst worden. Die Reduktion bezüglich der relativen Abweichung

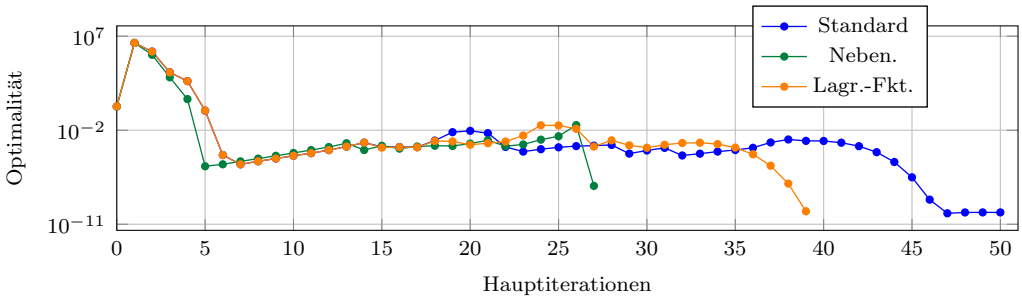
der Zielfunktion führt auf 686 Beispiele. Diese Zahlen zeigen, dass trotz der Reduktion der Hauptiterationen beide Algorithmen in über 90% der Fälle das gleiche lokale Minimum liefern. In diesen Fällen kann die Reduktion der benötigten Hauptiterationen demnach größtenteils mit den Auswirkungen der Korrektur begründet werden und wird nicht durch auftretende Effekte der Nichtlinearität verursacht.
Beide Abbildungen zeigen den Erfolg der Methode. In 371 Fällen erreichen bei-



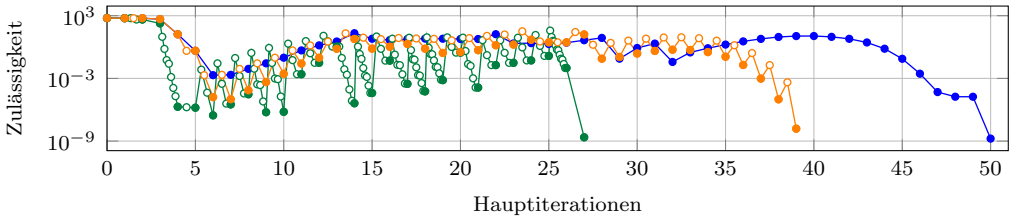
(a) Alle Beispiele (b) Nur gleichwertige Zielfunktion

Abbildung 7.9: Zulässigkeitskorrektur und Standardeinstellung im direkten Vergleich bezüglich Anzahl Iterationen. In grün alle Beispiele, bei denen die Zulässigkeitskorrektur ein besseres Ergebnis erzielt. In rot die Beispiele, welche von der Standardeinstellung besser gelöst werden. In dunkler Farbe, wenn nur der entsprechende Algorithmus in der Lage war das Beispiel zu lösen und in grau alle Beispiele die von beiden Algorithmen nicht gelöst werden.

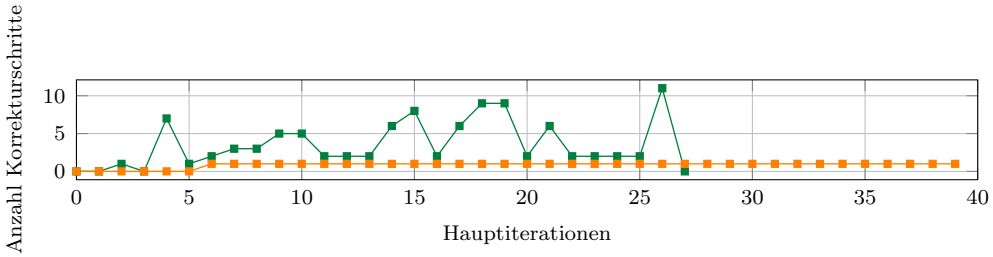
de Varianten dasselbe Minimum und das SQP-Verfahren mit Zulässigkeitskorrektur benötigt dafür weniger Hauptiterationen, wohingegen in lediglich 53 Fällen die Korrektur die Anzahl der Iterationen erhöht. Die übrigen Optimierungen verlaufen mit beiden Einstellungen identisch. Diese Zahlen belegen, dass das Ziel mit Hilfe der Korrektur die Anzahl der Hauptiterationen zu reduzieren erfüllt wurde.
Um die Auswirkungen der Zulässigkeitskorrektur anhand eines konkreten Beispiels zu verdeutlichen wird das Beispiel *tenbars4* aus der CUTEst-Sammlung mit verschiedenen Einstellungen gelöst. Das Problem ist mit 18 Optimierungsvariablen und 9 Nebenbedingungen zwar relativ klein, zeigt aber ein interessantes Verhalten bei Verwendung der verschiedenen Korrekturvarianten. Das Beispiel hat eine lineare Zielfunktion, nichtlineare Nebenbedingungen und die Ableitungen sind regulär. In Abbildung 7.10 sind Optimalität, Zulässigkeit und die Anzahl der Korrekturschritte aufgetragen. Im Graphen der Zulässigkeit in Abbildung 7.10b sind die Nebenbedingungsverletztungen am Ende einer Hauptiteration, das heißt nach Durchführung des Korrekturalgorithmus, durch gefüllte Punkte repräsentiert. Demnach ist der erste leer gezeichnete Punkt die Verletzung nach Durchführung der Liniensuche und die darauf folgenden Punkte, inklusive des gefüllten Punktes, sind durch Korrekturschritte bestimmt worden. Im direkten Vergleich der drei Kurven in den Abbildungen



(a) Optimierungsverlauf (Optimalität im Vergleich)



(b) Optimierungsverlauf (Zulässigkeit im Vergleich)



(c) Anzahl Korrekturschritte

Abbildung 7.10: Optimalität, Zulässigkeit und Anzahl Korrekturschritte während des Optimierungsverlaufes für das Beispiel *tenbars4*

7.10a und 7.10a zeigt sich zunächst, dass die Optimierung inklusive Korrektur deutlich weniger Hauptiterationen benötigen. Werden nur die Nebenbedingungen als Abbruchkriterium verwendet, benötigt die Optimierung 27 Hauptiterationen und 100 Korrekturschritte. Die Lagrange-Funktion als Kriterium liefert 39 Iterationen mit 34 Korrekturschritten und die normale Optimierung führt sogar 50 Iterationen durch. Bei Betrachtung der Iterationsverläufe offenbart sich ein Nachteil der Korrektur. Die Entwicklung der Nebenbedingungsverletzung bei Verwendung der beiden Korrekturvarianten zeigt, dass die Schritte des normalen SQP-Verfahrens die zuvor erreichte Verbesserung der Zulässigkeit in vielen Fällen in einem gewissen Maß wieder zurücksetzen. Dieses Verhalten lässt sich durch Betrachtung der Struktur der Korrektur erklären. Bei der Berechnung des Schritts wird lediglich die aktuelle Nebenbedingungsverletzung betrachtet. Büskens [12] gibt eine Fehlerabschätzung bezüglich der Optimalitätsverletzung dieser Korrektur an, diese bezieht sich aber im

hier betrachteten Zusammenhang nur auf die Optimalitätskriterien des Unterproblemes und lässt sich nicht direkt auf das nichtlineare Ausgangsproblem übertragen. Wird in der nächsten Hauptiteration die Optimalität wieder berücksichtigt werden im vorliegenden Beispiel die zuvor erzielten Verbesserungen der Zulässigkeit teilweise revidiert.
Das beobachtete Verhalten war einer der Gründe für die Einführung der Lagrange-Funktion oder des Kontraktionsfaktors als weitere Kriterien zur Bewertung der Qualität der Korrekturschritte. Das Beispiel zeigt aber, dass dieses Verhalten auch auftritt, wenn die Lagrange-Funktion als Kriterium verwendet wird und in jeder Iteration nur einen Korrekturschritt zulässt. Gleichzeitig zeigt sich aber auch, dass trotz dieser Problematik die Korrekturalgorithmen das Beispiel schneller lösen. Die offensivere Variante basierend auf den Nebenbedingungen ist weiterhin der konservativeren Verwendung der Lagrange-Funktion, bezogen auf die Anzahl Hauptiterationen, überlegen und spart dabei die zusätzliche Auswertung der Zielfunktion in jeder Iteration der Korrektur. Die Korrekturschritte der auf den Nebenbedingungen basierenden Variante erfüllen für dieses Beispiel zudem die Kontraktionsbedingung, so dass sich das Optimierungsverhalten auch bei zusätzlicher Berücksichtigung dieser Bedingung nicht verändert.
Als zweites Beispiel wird erneut *lobsterz* betrachtet. In Abbildung 7.11 sind auch hier die Optimalität, Zulässigkeit und die Anzahl der Korrekturschritte abgebildet. Die zuvor beobachtete unerwünschte Entwicklung der Nebenbedingungsverletzung tritt in diesem Fall bei Verwendung der Korrektur mit nebenbedingungsbasierter Abbruchbedingung nicht auf. Die einzelnen Korrekturschritte erzielen nur eine leichte Verbesserung der Zulässigkeit. Trotzdem zeigt die Reduktion von 26 Iterationen auf 22 Iterationen, dass auch in diesem Fall die Verwendung der Korrektur positive Auswirkungen auf den Optimierungsverlauf hat.
Als Erweiterung der Zulässigkeitskorrektur wurde in Abschnitt 5.2.2 die Extrapolationsmethode von Steffensen vorgestellt. Diese erlaubt es eine Folge von linear konvergenten Iterierten zu extrapolieren und so die Konvergenzgeschwindigkeit der Folge zu erhöhen. Die Extrapolationstechnik wurde als optionale Ergänzung innerhalb der Zulässigkeitskorrektur implementiert. Der Tabelle B.7 im Anhang ist entnehmbar, dass die Steffensen-Extrapolation standardmäßig für die Zulässigkeitskorrektur aktiviert ist und somit insbesondere in den bisherigen Analysen der Korrektur enthalten war. Im Folgenden sollen die Auswirkungen der Extrapolation auf die Korrektur kurz analysiert werden, um die Aktivierung in der Standardkonfiguration zu rechtfertigen.
Zur Beurteilung der Extrapolation wird die Korrektur mit den Abbruchbedingungen basierend auf den Nebenbedingungen und der Lagrange-Funktion getestet. Die Tests werden jeweils einmal mit und einmal ohne die Extrapolation durchgeführt. In Tabelle 7.4 sind die Ergebnisse dieser Testläufe im Vergleich zur Grundeinstellung abgedruckt. Werden die Nebenbedingungen als Abbruchkriterium verwendet, ist die Effizienzsteigerung auf Grund der Steffensen-Extrapolation deutlich zu erkennen. Der Optimierungslauf benötigt statt 17 Stunden und 14 Minuten ohne Extrapolati-

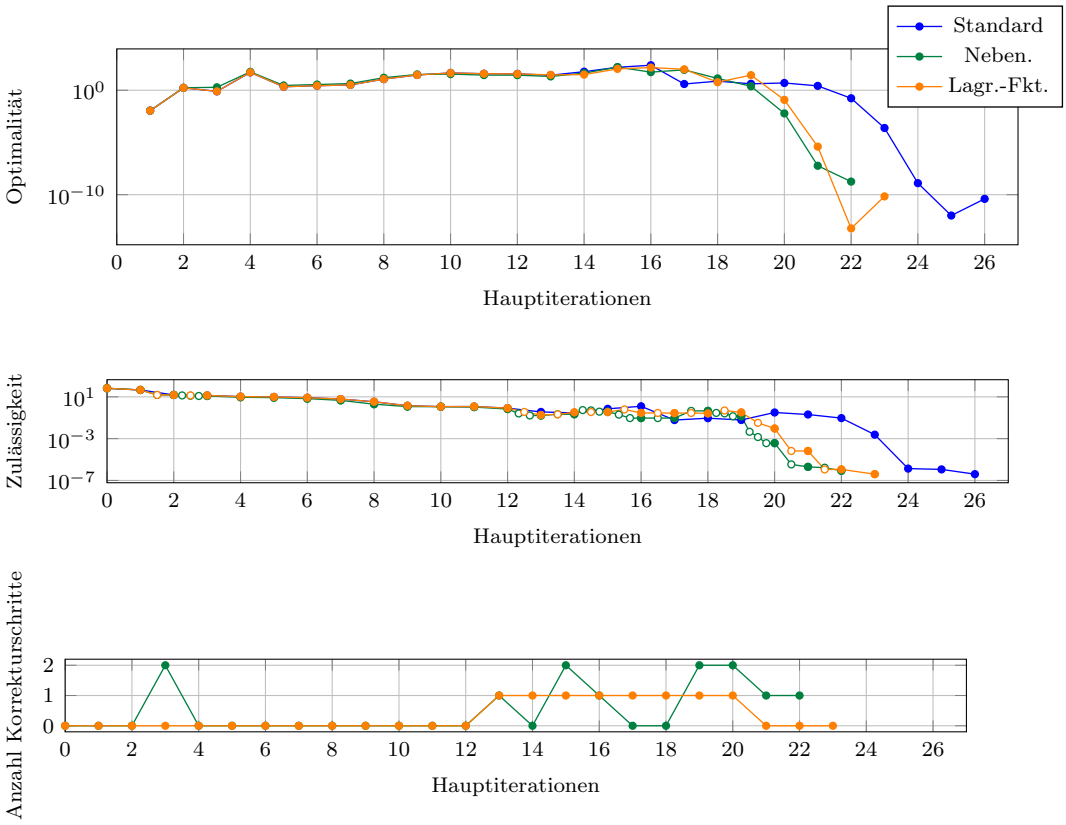
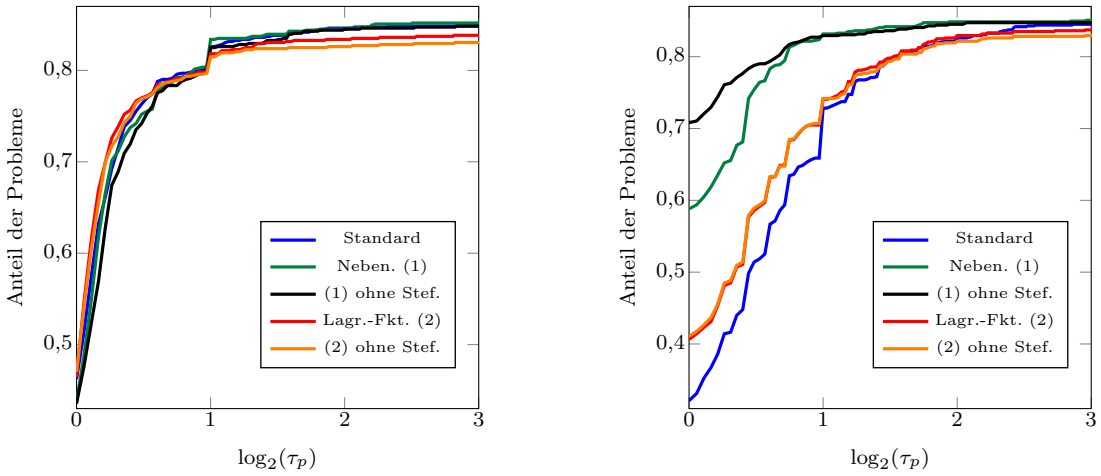


Abbildung 7.11: Optimalität, Zulässigkeit und Anzahl Korrekturschritte während des Optimierungsverlaufes für das Beispiel *lobsterz*

on nur noch 16 Stunden und 17 Minuten und ist somit etwa 57 Minuten schneller. Weiterhin verbessert sich auch die Qualität des Optimierungslaufes. Mit der Extrapolation können 764 Beispiele optimal und 17 akzeptabel gelöst werden. Im Vergleich dazu werden mit der einfachen Korrektur nur 760 Beispiele optimal und 20 akzeptabel gelöst.
Wird die Lagrange-Funktion für den Abbruch zu Grunde gelegt, ergibt sich ein anderes Bild. Zunächst fällt auf, dass die Testläufe etwas schneller sind. Mit Extrapolation werden 16 Stunden und 1 Minute benötigt und ohne Extrapolation 15 Stunden und 48 Minuten. In diesem Fall ist demnach der Algorithmus schneller, wenn keine Extrapolation verwendet wird. Allerdings löst der Optimierer mit Korrektur basierend auf der Lagrange-Funktion nur 752 Beispiele mit Extrapolation und 744 Beispiele ohne Extrapolation optimal. Dieses Ergebnis ist wesentlich schlechter als bei Verwendung der Nebenbedingungen oder der Grundeinstellung.
Zur anschaulichen Bewertung der Steffensen-Extrapolation sind weiterhin in Abbildung 7.12 die Performance-Profile bezüglich Rechenzeit und Hauptiterationen abgebildet. Es ergibt sich ein ähnliches Bild wie bei der Beurteilung der Abbruchbedingungen in Abbildung 7.8. Bezüglich der Rechenzeit liegen die Kurven für kleine Werte von τ_p nahe beieinander. Für ansteigendes τ_p wird die verringerte Robustheit

	Standard		Nebenbed.		Nebenbed. (ohne Stef.)		Lagrange-Fkt.		Lagrange-Fkt. (ohne Stef.)	
Optimal	758	85,07%	**764**	**85,75%**	760	85,20%	752	84,40%	744	83,50%
Akzeptabel	**22**	**2,47%**	17	1,91%	20	2,24%	21	2,36%	**22**	**2,47%**
Misserfolg	111	12,46%	**110**	**12,35%**	112	12,56%	118	13,24%	125	14,03%
Dauer	16h 34m 26s		16h 17m 50s		17h 14m 6s		16h 1m 2s		**15h 48m 58s**	

Tabelle 7.4: Ergebnisse der Steffensen-Extrapolation innerhalb der Zulässigkeitskorrektur



(a) CPU-Zeit (b) Hauptiterationen

Abbildung 7.12: Performance-Profile zur Bewertung der Steffensen-Extrapolation innerhalb der Zulässigkeitskorrektur.

bei Verwendung der Lagrange-Funktion als Abbruchkriterium deutlich. Der Abstand ist in Abbildung 7.12a deutlicher als bei der Beurteilung des Kontraktionsfaktors in Abbildung 7.8a. Die beiden entsprechenden Kurven in rot und orange liegen deutlich unterhalb der restlichen Graphen.
Werden die Hauptiterationen für das Profil zu Grunde gelegt, so kann eine weitere Besonderheit beobachtet werden. In Abbildung 7.12b zeigt sich, dass die Variante basierend auf den Nebenbedingungen ohne Extrapolation zur Lösung von etwa 71 % der Probleme die geringste Anzahl Hauptiterationen benötigt. Gleichzeitig liegt der y-Achsenabschnitt der dazugehörigen Kurve in Abbildung 7.12a aber unterhalb der restlichen Graphen. Demnach ist diese Konfiguration in der Lage eine große Menge Hauptiterationen zu sparen. Allerdings werden hierfür im Vergleich zur Variante mit Extrapolation deutlich mehr Iterationen im Korrekturalgorithmus benötigt, so dass die zusätzlichen Auswertungen der Nebenbedingungen dafür sorgen, dass insgesamt mehr Rechenzeit benötigt wird. Zusammenfassend besticht die Steffensen-Extrapolation für beide Abbruchbedingungen durch eine gesteigerte Robustheit des Verfahrens. Auf Grund des besseren Ergebnisses und der deutlichen Geschwindigkeitssteigerung bei Verwendung der Nebenbedingungen als Abbruchkriterium, hat

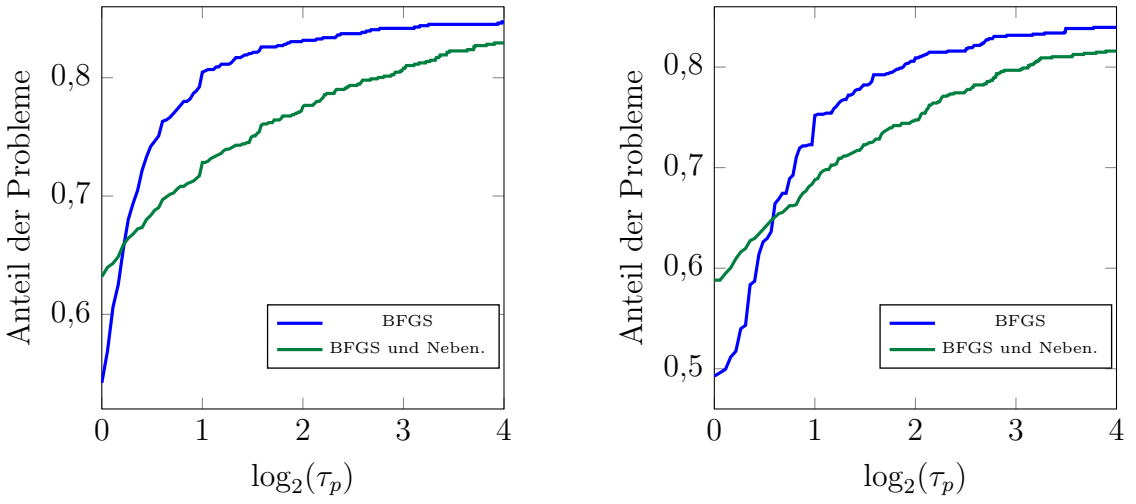
	Standard		Verletzte NLP Neben.		Aktive QP Neben.	
Optimal	758	85,07%	761	85,31%	**765**	**85,76%**
Akzeptabel	**22**	**2,47%**	21	2.35%	17	1,91%
Misserfolg	111	12,46%	110	12,33%	110	12,33%
(Timeout)	(10)		(8)		(8)	
Dauer	16h 34m 14s		**16h 15m 24s**		16h 17m 50s	

Tabelle 7.5: Ergebnisse von WORHP mit Zulässigkeitskorrektur für unterschiedliche Störungsarten im Vergleich zur Standardvariante

sich die Kombination dieses Abbruchkriteriums mit der Extrapolation als beste Konfiguration in diesem Vergleich herausgestellt.
In Bemerkung 5.1 wurden Anmerkungen über die Art der zu korrigierenden Störung gemacht. Basierend auf der Grundlage des Sensitivitätssatzes ist es nur sinnvoll diejenigen nichtlinearen Nebenbedingungen zur Berechnung der Störung auszuwerten, deren Linearisierungen im Minimum des Unterproblems aktiv waren. Gleichzeitig motiviert aber die Analogie zum vereinfachten Newton-Verfahren, dass alle verletzten nichtlinearen Nebenbedingungen in die Störung einfließen sollten.
Die Ergebnisse der beiden Varianten sind in Tabelle 7.5 einzusehen. Zeitlich unterscheiden sich die Resultate kaum. Die Qualität der Ergebnisse wird im Gegensatz dazu aber leicht von der Art der Störung beeinflusst. Bei Einschränkung der Auswertung der nichtlinearen Nebenbedingungen auf die aktiven Nebenbedingungen des Unterproblems können im direkten Vergleich zur allgemeineren Störung vier zusätzliche Beispiele optimal gelöst werden. Diese werden bei Berücksichtigung aller verletzten Nebenbedingungen nur akzeptabel gelöst.
Die Zahlen zeigen, dass es leichte Vorteile hat nur die aktiven Nebenbedingungen aus dem Unterproblem zu betrachten. Gleichzeitig wird aber auch deutlich, dass der Unterschied nur marginal ausfällt und das Ergebnis bekräftigt die Behauptung, dass es sich bei der Zulässigkeitskorrektur um eine Art vereinfachtes Newton-Verfahren handelt.
Ein weiterer interessanter Aspekt der Zulässigkeitskorrektur ist, dass nur die parametrischen Sensitivitäten der quadratischen Unterprobleme verwendet werden. Somit kann die Methode auch angewendet werden, wenn statt der analytischen Hesse-Matrix beispielsweise BFGS-Matrizen verwendet werden. In Tabelle B.5 des Anhanges sind die verschiedenen Konfigurationsmöglichkeiten der BFGS-Verfahren innerhalb von WORHP inklusive ihrer Standardkonfiguration abgedruckt. Für diese Auswertung wird ein strukturbasiertes Block-BFGS-Verfahren mit variabler Blockgröße verwendet (Vergleiche Abschnitt 3.3.2.3). In Tabelle 7.6 sind die erzielten Resultate der Standardeinstellung mit BFGS und der Zulässigkeitskorrektur mit BFGS abgedruckt. Zunächst fällt auf, dass die Gesamtrechenzeit im Vergleich zu den bisherigen Auswertungen deutlich geringer ausfällt. Weiterhin ist die Zulässigkeitskorrektur etwa 27 Minuten langsamer und löst 13 Probleme weniger optimal. Bei dieser Unter-

	Standard (BFGS)		Neben. (BFGS)	
Optimal	**757**	**84,96%**	746	83,63%
Akzeptabel	15	1,68%	**22**	**2.47%**
Misserfolg	**119**	**13,36%**	124	13,90%
Dauer	**14h 22m 56s**		14h 49m 31s	

Tabelle 7.6: Ergebnisse von WORHP mit Zulässigkeitskorrektur bei Verwendung von BFGS



(a) CPU-Zeit

(b) Hauptiterationen

Abbildung 7.13: Performance-Profile zur Bewertung der Zulässigkeitskorrektur bei Verwendung von BFGS als Ersatz für die Hesse-Matrix.

suchung zeigt sich ein etwas stärkerer Einfluss auf die gefundenen lokalen Minima. Nur für 81% der Beispiele wurde dasselbe lokale Minimum gefunden.
Trotz dieser Ergebnisse kann die Verwendung der Korrektur in diesem Zusammenhang sinnvoll sein. Zur besseren Veranschaulichung der Situation sind in Abbildung 7.13 auch die dazugehörigen Performance-Profile abgedruckt. In den Profilen ist ersichtlich, dass die Konfiguration mit BFGS und Korrektur 63,2% der Probleme mindestens so schnell wie dieselbe Konfiguration ohne Korrektur löst. Im Vergleich dazu ist die Einstellung ohne Korrektur nur in 54,2% die beste Wahl. In diesen Prozentzahlen sind Beispiele mit gleichem Zeitaufwand für beide Konfigurationen enthalten. Werden nur diese Zahlenwerte betrachtet, ist BFGS mit Korrektur im direkten Vergleich die bessere Wahl. Allerdings zeigen die Profile auch, dass für ansteigende τ_p die Variante ohne Korrektur mehr Beispiele lösen kann und somit robuster ist. Analoge Beobachtungen ergeben sich bei Betrachtung der benötigten Hauptiterationen. Zusammenfassend kann gesagt werden, dass die Korrektur auch

	Standard		Adaptives $N_\delta \in [1; N_g + N_h]$		Aktive QP Neben., $\theta = 0{,}25$		$\theta = 0{,}25$, adapt. N_δ		Aktive QP Neben., $\theta = 0{,}25$, adapt. N_δ	
Optimal	758	85,07%	765	85,86%	**770**	**86,42%**	756	84,85%	753	84,51%
Akzeptabel	22	2,47%	19	2,13%	10	2,13%	22	2,47%	**23**	**2,58%**
Misserfolg	111	12,46%	107	12,01%	**102**	**11,45%**	113	12,68%	115	12,91%
(Timeout)	(10)		(18)		(7)		(13)		(14)	
Dauer	16h 34m 14s		20h 12m 52s		**14h 24m 23s**		18h 28m 17s		18h 23m 42s	

Tabelle 7.7: Ergebnisse von WORHP mit den neu eingeführten Strategien Zulässigkeitskorrektur, sensitivitätsbasierter Relaxierung und adaptiver Relaxierung kombiniert im Vergleich

bei Verwendung von BFGS den Optimierungsprozess beschleunigen kann. Gleichzeitig verliert das Verfahren aber an Robustheit.
In einer abschließenden Auswertung wird die Korrektur mit der Abbruchbedingung basierend auf den Nebenbedingungen in Kombination mit der sensitivitätsbasierten Relaxierung und der adaptiven Relaxierung bewertet. In Tabelle 7.7 werden die Grundeinstellung, die Korrektur und die adaptive Relaxierung jeweils mit sensitivitätsbasierter Relaxierung und gleichzeitig aktivierte Korrektur, adaptive Relaxierung und sensitivitätsbasierte Relaxierung miteinander verglichen.
Bemerkenswert ist die Geschwindigkeitssteigerung von Korrektur und sensitivitätsbasierter Relaxierung in Kombination. Werden beide Optionen gleichzeitig aktiviert, kann die Optimierungsdauer auf 14 Stunden und 24 Minuten reduziert werden und ist somit 1 Stunde und 50 Minuten kürzer als bei der Standardvariante. Die adaptive Relaxierung verzeichnet eine Rechenzeit von 20 Stunden und 12 Minuten. Die sensitivitätsbasierte Relaxierung verringert diese Dauer auf 18 Stunden und 28 Minuten und eine zusätzliche Korrektur der Zulässigkeit benötigt 18 Stunden und 23 Minuten. Beide Versionen büßen aber Robustheit ein. Die einfache adaptive Relaxierung löst 765 Beispiele optimal und 19 akzeptabel, mit sensitivitätsbasierter Relaxierung verschieben sich die Ergebnisse zu 756 optimalen und 22 akzeptablen Lösungen, durch zusätzliche Korrektur der Nebenbedingungen sogar zu 753 optimalen und 23 akzeptablen.
Zusammenfassend stellt sich die Kombination von Zulässigkeitskorrektur und sensitivitätsbasierter Relaxierung als äußerst effizient heraus. Die Kombination ist die robusteste und gleichzeitig schnellste Kombination der vorgestellten Konfigurationen.

7.3.2 Strategien für die Hesse-Matrix

Neben den bereits ausgewerteten Techniken im Zusammenhang mit der Behandlung der Nebenbedingungen lag ein weiterer Schwerpunkt der Entwicklung von Algorithmen zur Effizienzsteigerung der Optimierung innerhalb dieser Arbeit auf der verwendeten Hesse-Matrix zur Aufstellung der Unterprobleme. Zwei verschiedene Strategien werden in diesem Abschnitt anhand geeigneter Tests bewertet.

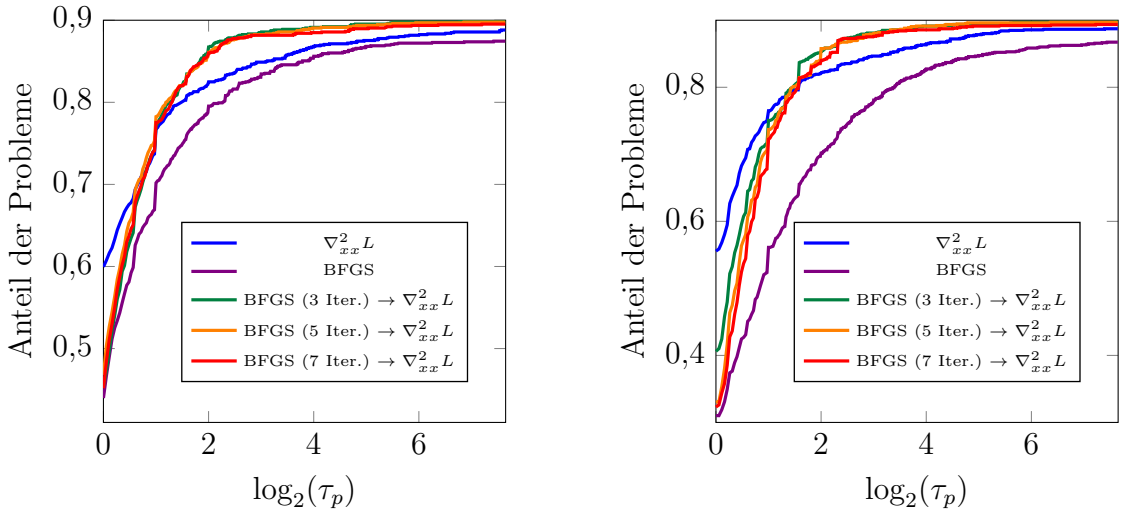
	$\nabla^2_{xx}L$		BFGS		BFGS (3 Iter.)		BFGS (5 Iter.)		BFGS (7 Iter.)	
Optimal	1117	89,30%	1094	87,45%	**1125**	**89,93%**	1123	89,77%	1120	89,53%
Akzeptabel	23	1,84%	**36**	**2,88%**	18	1,44%	15	1,20%	16	1,28%
Misserfolg	111	8,86%	121	9,67%	**108**	**8,63%**	113	9,03%	115	9,19%
(Timeout)	(10)		(10)		(11)		(15)		(15)	
Rechenzeit	17h 19m 25s		**15h 18m 43s**		17h 15m 12s		17h 12m 41s		16h 37m 38s	

Tabelle 7.8: Ergebnisse von WORHP für verschiedene Wechselzeitpunkte abhängig von den erfolgten Hauptiterationen

Zunächst wird die Variation des verwendeten Hesse-Matrix-Typs untersucht. Im Anschluss daran wird die sensitivitätsbasierte Regularisierung der Hesse-Matrix betrachtet.

7.3.2.1 Phasenwechsel

Nach Satz 3.5 über die Konvergenz des lokalen SQP-Verfahrens ermöglichen BFGS-Matrizen lediglich lokal superlineare Konvergenz, wohingegen bei Verwendung der analytischen Hesse-Matrix lokal quadratische Konvergenz erreicht werden kann. Diese Konvergenzaussagen gelten aber jeweils nur in der Nähe des lokalen Minimums und zusätzlich sind bei Verwendung der echten Hesse-Matrix während der Globalisierungsphase Maßnahmen wie die Regularisierung notwendig, um die Lösbarkeit der quadratischen Unterprobleme garantieren zu können.
Aus diesen Umständen ist die Idee entstanden, während der Globalisierungsphase zu Beginn der Optimierung BFGS-Matrizen zu verwenden, um die Problematik der Regularisierung der Hesse-Matrix zu umgehen und die benötigte positive Definitheit auf dem Kern der aktiven Nebenbedingungen für das quadratische Unterproblem auf diese Weise zu garantieren. Entsprechende Strategien wurden im Abschnitt 3.3.3.2 vorgeschlagen. Die relevanten Parameter zur Konfiguration dieser Strategie innerhalb von WORHP sind im Anhang in Tabelle B.6 abgedruckt.
Der schwierigste Aspekt dieser Technik ist die Identifikation eines geeigneten Zeitpunktes für den Wechsel von den BFGS-Matrizen zur echten Hesse-Matrix. Die einfachste Möglichkeit besteht darin nach einer festen Anzahl von Iterationen die verwendete Hesse-Matrix zu wechseln. In Tabelle 7.8 sind die Ergebnisse für einen Wechsel nach drei, fünf beziehungsweise sieben Iterationen im Vergleich zur Verwendung der analytischen Hesse-Matrix oder von BFGS-Matrizen im gesamten Zeitraum abgedruckt.
Die Ergebnisse in der Tabelle zeigen, dass durch die vorgeschlagenen Wechselstrategien eine Steigerung der Robustheit der Optimierung erreicht werden kann. Das beste Ergebnis liefert bezüglich der Robustheit der Wechsel nach drei Hauptiterationen. Mit dieser Strategie können 1125 Beispiele optimal gelöst werden, im Vergleich zu 1117 Beispielen bei Verwendung der echten Hesse-Matrix und 1094 Beispielen bei Verwendung von BFGS-Matrizen ohne Wechsel. Bei Betrachtung der Rechenzeit sticht BFGS-Variante ohne Wechsel durch eine Geschwindigkeitssteigerung von



(a) CPU-Zeit (b) Hauptiterationen

Abbildung 7.14: Performance-Profile zur Bewertung des Phasenwechsels von BFGS auf analytische Hesse-Matrix

2 Stunden und 1 Minute gegenüber der analytischen Hesse-Matrix deutlich hervor. Diese Geschwindigkeitssteigerung wird allerdings zulasten der Robustheit erzielt. Wird nur bis zur siebten Iteration BFGS verwendet und anschließend auf die analytische Hesse-Matrix gewechselt, so kann eine Geschwindigkeitssteigerung von 42 Minuten erzielt werden, wobei gleichzeitig drei zusätzliche Beispiele optimal gelöst werden. Die Variation der Hesse-Matrix hat im Vergleich zu den bisher betrachteten Konfigurationen eine deutlich stärkere Auswirkung auf die gefundenen lokalen Minima. In den vorgestellten numerischen Untersuchungen wurde in lediglich 70% der Fälle dasselbe lokale Minimum gefunden. Weitere Einblicke gewährt auch in diesem Fall die Verwendung von Performance-Profilen bezüglich Rechenzeit und Hauptiterationen. Diese sind in Abbildung 7.14 abgedruckt.
Dem Profil bezüglich der Zeit ist zu entnehmen, dass die Verwendung der analytischen Hesse-Matrix für etwa 60% der Beispiele die schnellste Variante ist. Gleichzeitig zeigt sich auch, dass die Wechselstrategie bereits für leicht größere Werte von τ_p in der Lage sind mehr Beispiele zu lösen. Ein ähnliches Bild ergibt sich bezüglich der Hauptiterationen. Auch dort ist die Robustheit der Wechselstrategien gut erkennbar. Insgesamt zeigt sich, dass die Wechselstrategien auf die gesamte Testmenge gute Möglichkeiten zur Effizienzsteigerung des Optimierungsverfahrens bieten.
Im Abschnitt 3.3.3.2.2 wurden neben den bereits getesteten iterationsbasierten Wechselbedingungen weitere Kriterien vorgestellt. Toleranzbasiert erfolgt der Wechsel erst nachdem Optimalitäts- und Zulässigkeitsmaß eine gewählte Schwelle unterschreiten. Als weitere Alternative wurde vorgeschlagen die Iterationszahlen innerhalb des Unterproblemlösers der letzten beiden Hauptiterationen zu betrachten. Wenn diese hinreichend gering waren, deutet dies auf gut gestellte Hilfsprobleme

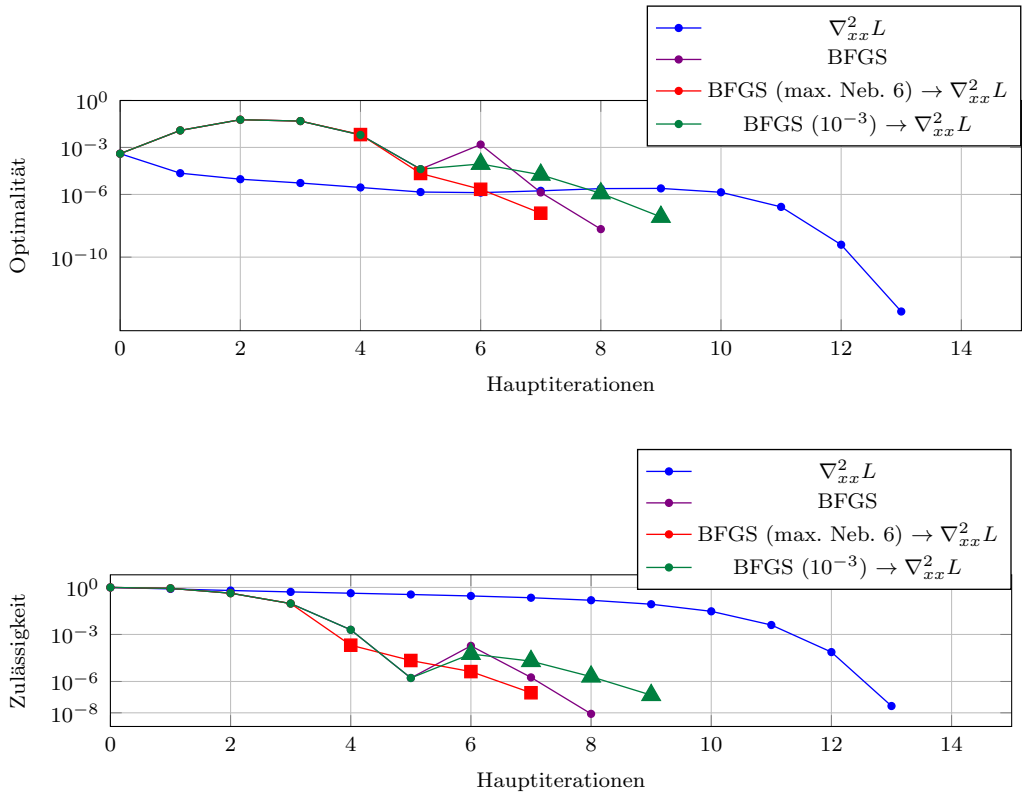


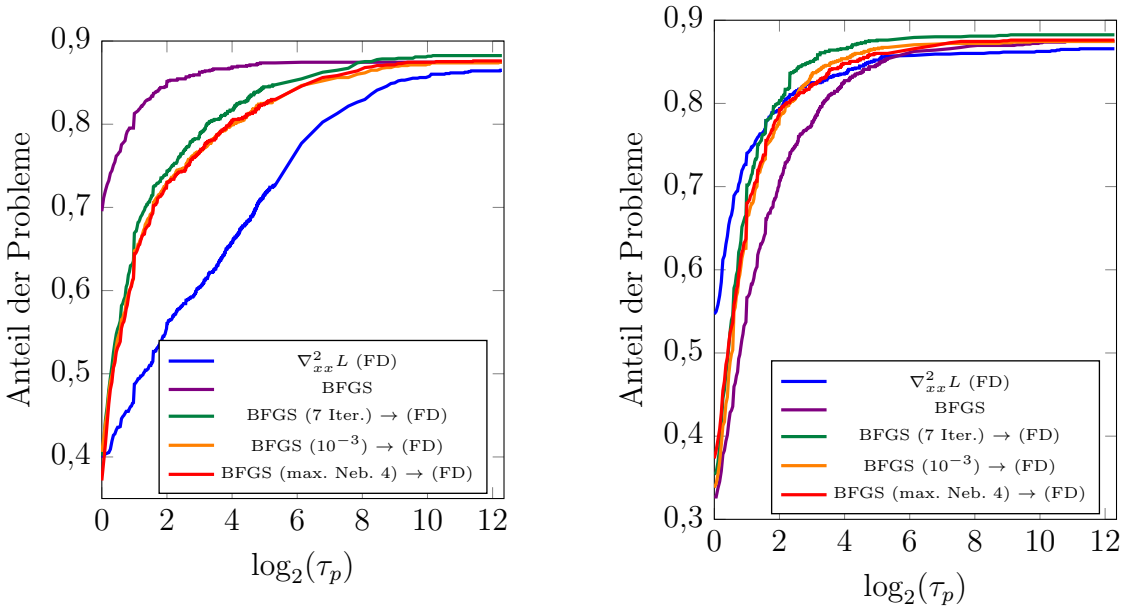
Abbildung 7.15: Optimalität und Zulässigkeit während des Optimierungsverlaufes für das Beispiel *dtoc5* mit verschiedenen Wechselkriterien. Symbole nach dem Wechsel in groß.

hin und eine Annäherung an ein lokales Minimum ist zu vermuten, so dass auf die echte Hesse-Matrix zurück gewechselt werden kann.
Die Wirkung der verschiedenen Kriterien im Vergleich mit der Variante ohne Wechsel wird anhand des Beispiels *dtoc5* gezeigt. Das Beispiel *dtoc5* ist ein quadratisch beschränktes quadratisches Optimierungsproblem und hat 10000 Optimierungsvariablen und 4999 Nebenbedingungen. Es handelt sich um ein diskretes zeitoptimales Steuerungsproblem. Als Wechselkriterien wurden eine maximale Nebeniterationszahl von sechs in den letzten zwei Hauptiterationen und eine Toleranz von 10^{-3} gewählt. In Abbildung 7.15 sind Optimalität und Zulässigkeit der Iterierten für diese Wechselbedingungen im Vergleich zum Verlauf bei Verwendung von BFGS und der analytischen Hesse-Matrix abgebildet.
Die Verläufe zeigen, dass durch die Verwendung von BFGS-Matrizen in den ersten Iterationen bei diesem Beispiel der Fortschritt in Richtung Zulässigkeit zulasten des Optimalitätskriteriums einen insgesamt schnelleren Optimierungslauf ermöglicht. Die toleranzbasierte Variante in grün wechselt auf die analytische Hesse-Matrix in Iteration sechs. Der Wechsel auf Grund geringer Nebeniterationszahlen erfolgte bereits in der zweiten Iteration, diese Kurve ist in rot abgebildet. Beide Varianten sind in der Lage die Optimierung im Vergleich zur ausschließlichen Nutzung der ana-

	$\nabla^2_{xx}L$ (FD)		BFGS		BFGS (7 Iter.)		BFGS (max. Neb. 4)		BFGS (10^{-3})	
Optimal	1083	86,57%	1094	87,45%	**1104**	**88,25%**	1096	87,61%	1094	87,45%
Akzeptabel	**39**	**3,12%**	36	2,88%	27	2,16%	34	2,72%	34	2,72%
Misserfolg	129	10,31%	121	9,67%	**120**	**9,59%**	121	9,67%	123	9,83%
(Timeout)	(37)		(10)		(18)		(18)		(12)	
Rechenzeit	62h 11m 19s		**15h 18m 43s**		35h 37m 33s		35h 47m 30s		33h 26m 52s	

Tabelle 7.9: Ergebnisse von WORHP für verschiedene Wechselstrategien im Vergleich zu finiten Differenzen

lytischen Hesse-Matrix zu verbessern. Werden während der gesamten Optimierung BFGS-Matrizen verwendet ergibt sich ein leicht besserer Verlauf als bei dem späten Wechsel der toleranzbasierten Wechselvariante.
Die bisherigen Auswertungen haben gezeigt, dass es sich für einige Beispiele als Ersatz für die analytische Hesse-Matrix lohnen kann während der Globalisierungsphase BFGS-Matrizen zu verwenden. Im weiteren Verlauf dieses Abschnitt soll analysiert werden, welche Auswirkungen die Wechselstrategie auf den Lösungsverlauf hat, wenn die analytische Hesse-Matrix nicht zur Verfügung steht. Der Benutzer hat somit die Wahl die echte Hesse-Matrix mit Hilfe von finiten Differenzen zu ermitteln, für den gesamten Iterationsverlauf BFGS-Matrizen zu verwenden oder während der Globalisierungsphase BFGS-Matrizen zu verwenden und in der Nähe des lokalen Minimums auf die Verwendung finiter Differenzen zu wechseln. Für die Auswertung wurden drei verschiedene Wechselstrategien angesetzt. Ein iterationsbasierter Wechsel nach sieben Hauptiterationen, ein toleranzbasierter Wechsel mit einer Schwelle von 10^{-3} und der Wechsel basierend auf den Nebeniterationen mit einer Grenze von maximal vier Iterationen innerhalb des Unterproblemlösers in zwei aufeinanderfolgenden Hauptiterationen. Die Ergebnisse dieser Varianten sind in Tabelle 7.9 im Vergleich zu finiten Differenzen und der Verwendung von BFGS-Matrizen ohne Wechsel abgedruckt.
Besonders auffällig ist der enorme Zeitaufwand bei Verwendung finiter Differenzen während der gesamten Optimierung. Im Vergleich zur Verwendung von BFGS-Matrizen ohne Wechsel werden statt 15 Stunden und 18 Minuten 62 Stunden und 11 Minuten zur Lösung der gesamten Testbeispiele benötigt. Der stark ansteigende Zeitbedarf äußerst sich auch in der Anzahl der Timeout-Fehler, so dass insgesamt nur 1083 Beispiele optimal mit Hilfe von finiten Differenzen gelöst werden können. In diesem Vergleich zeigt sich, dass die Wechselstrategien in der Lage sind die Robustheit der Optimierung zu verbessern. Wird nach sieben Iterationen auf die Verwendung finiter Differenzen gewechselt, können 1104 Beispiele optimal gelöst werden. Mit dieser Strategie können somit 10 Beispiele mehr als bei Verwendung von BFGS ohne Wechsel gelöst werden und sogar 21 Beispiele mehr als bei Verwendung finiter Differenzen. Die zusätzliche Robustheit erfordert allerdings auch bei den verschiedenen Wechselvarianten zwischen 33 und 35 Stunden Rechenzeit gegenüber den etwa 15 Stunden bei Verwendung von BFGS ohne Wechsel.
In Abbildung 7.16 sind die Performance-Profile für diese Strategien im Vergleich zur



(a) CPU-Zeit

(b) Hauptiterationen

Abbildung 7.16: Performance-Profile zur Bewertung des Phasenwechsels von BFGS auf finite Differenzen für die Hesse-Matrix

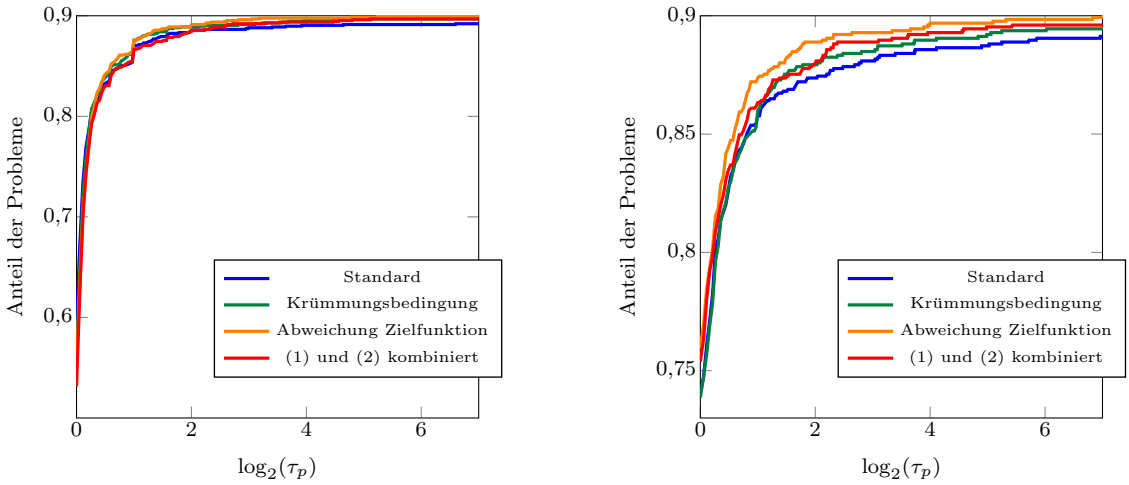
Verwendung finiter Differenzen und BFGS ohne Wechsel abgebildet.
Wie auf Grund der Daten aus Tabelle 7.9 zu erwarten hebt sich die Kurve der BFGS-Variante ohne Wechsel deutlich von den übrigen Strategien ab. Etwa 70% der Beispiele werden bei Verwendung von BFGS ohne Wechsel am schnellsten gelöst. Das andere Profil zeigt, dass die Verwendung finiter Differenzen zu deutlich kleineren Hauptiterationszahlen führt, für diese wird aber wie das zeitliche Profil zeigt, deutlich mehr Rechenzeit benötigt. Die getesteten Wechselstrategien liegen bezüglich Zeit und Iterationen zwischen den beiden Referenzvarianten. Gleichzeitig zeigen beide Profile, wie bereits anhand der Daten zu erwarten die erhöhte Robustheit der Wechselvarianten.
Zusammenfassend hat sich gezeigt, dass im Vergleich zur Verwendung der analytischen Hesse-Matrix die Wechselstrategien mit passender Konfiguration des Wechselzeitpunktes in der Lage sind sowohl die Lösungsgeschwindigkeit, als auch die Robustheit des Optimierungsverfahrens zu verbessern. Liegen keine analytischen Ableitungen vor, kann gegenüber der Verwendung finiter Differenzen eine deutliche Geschwindigkeitssteigerung und eine Verbesserung der Robustheit erzielt werden. Im Vergleich zu BFGS ohne Wechsel kann in diesem Fall die Robustheit verbessert werden. Diese Verbesserung benötigt aber mehr Rechenzeit.

	Standard		Krümmungs-bedingung		Abweichung Zielfunktion		Kombiniert	
Optimal	1117	89,29%	1122	89,69%	**1126**	**90,01%**	1122	89,69%
Akzeptabel	**23**	**1,84%**	22	1,76%	19	1,52%	18	1,44%
Misserfolg	111	8,87%	107	8,55%	**106**	**8,47%**	111	8,87%
(Timeout)	(10)		(10)		(10)		(12)	
	17h 19m 25s		**16h 54m 11s**		17h 7m 52s		18h 2m 47s	

Tabelle 7.10: Ergebnisse von WORHP bei Verwendung sensitivitätsbasierter Regularisierung im Vergleich

7.3.2.2 Sensitivitätsbasierte Anpassung der Regularisierung

Im Abschnitt 3.3.3.1 wurden ausführlich die innerhalb von WORHP implementierten Techniken zur Regularisierung der Hesse-Matrix vorgestellt. Besonders hervorgehoben wurde, dass die Regularisierung für die Lösbarkeit der Unterprobleme von essentieller Bedeutung ist. Mit Hilfe der parametrischen Sensitivitätsanalyse wurden in Abschnitt 5.3 Algorithmen zur Verbesserung der Regularisierung entwickelt. Diese Algorithmen verwenden die parametrischen Sensitivitätsableitungen der Unterprobleme, um nach einer erfolgreichen Iteration eine Schätzung für die Höhe der Regularisierung in der folgenden Iteration zu ermitteln. Zur Bestimmung der zu verwendenden Störungsgröße wurden zwei Varianten vorgestellt. Die erste Variante basierte auf einer Krümmungsbedingung an die Hesse-Matrix entlang der ermittelten Suchrichtung und die zweite Variante auf einer erlaubten prozentualen Abweichung der Zielfunktion. Die Konfiguration dieser Techniken innerhalb von WORHP erfolgt mit Hilfe der Parameter aus Tabelle B.8, welche im Anhang abgedruckt ist.
Für die folgenden numerischen Auswertungen wurde $\theta_\kappa = 0{,}75$ angesetzt und eine relative Abweichung der Zielfunktion von $\nu = 0{,}1$ erlaubt. Für den Vergleich wurden beide Varianten der sensitivitätsbasierten Regularisierung jeweils einzeln und kombiniert ausgewertet. In Tabelle 7.10 sind die Ergebnisse im Vergleich zur Standardregularisierung abgedruckt.
Die Daten zeigen, dass die Verwendung der sensitivitätsbasierten Regularisierung nutzbringend sein kann. Auch wenn die Geschwindigkeitssteigerungen bei Verwendung der beiden Varianten einzeln mit 12 bis 25 Minuten nur gering sind, können die Auswirkungen dieser Varianten positiv bewertet werden, da diese gleichzeitig eine größere Anzahl von Problemen optimal lösen. Die Kombination beider Varianten ist etwa 43 Minuten langsamer als die Standardvariante, löst aber ebenfalls mehr Beispiele optimal. Die verschiedenen Konfigurationen der Optimierung haben hierbei für 94% der Beispiele dasselbe lokale Minimum gefunden.
Die Performance-Profile in Abbildung 7.17 helfen die Auswirkungen weiter zu analysieren. Es sind auch hier die Profile bezüglich Rechenzeit und Hauptiterationen aufgetragen.

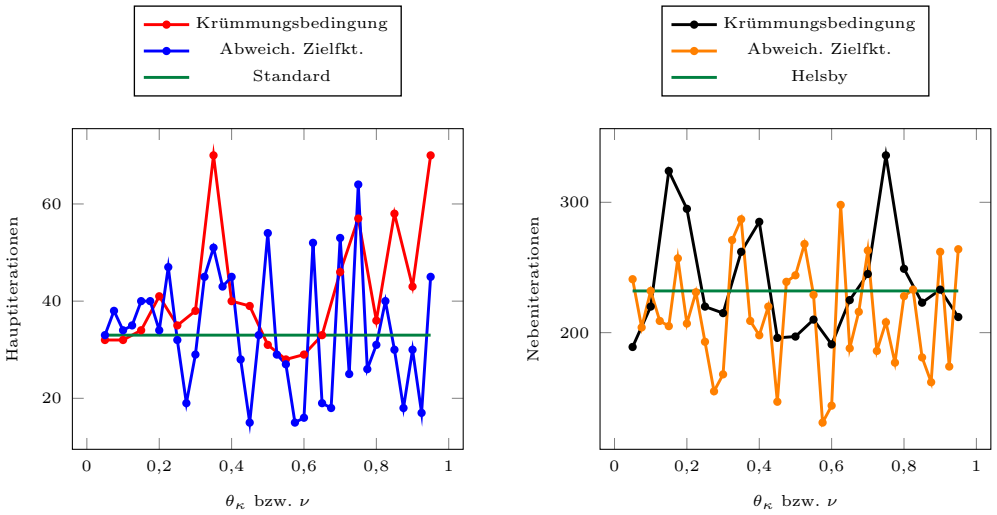


(a) CPU-Zeit

(b) Hauptiterationen

Abbildung 7.17: Performance-Profile für verschiedene Konfigurationen der sensitivitätsbasierten Regularisierung

Wie bereits die Daten aus Tabelle 7.10 gezeigt haben fällt die Geschwindigkeitssteigerung dieser Anpassung des Optimierungsalgorithmus nur gering aus. Die bereits in der Tabelle beobachtete gesteigerte Robustheit ist aber im Profil 7.17a ebenfalls zu erkennen. Etwas deutlicher fällt der Unterschied der vorgestellten Varianten im Profil bezüglich der Hauptiterationen in Abbildung 7.17b auf, wobei zu beachten ist, dass der gezeigte Ausschnitt der y-Achse verhältnismäßig klein ist.
Die gewählten Konfigurationen für die erlaubte Abweichung der Zielfunktion ν und für den Parameter θ_κ wurden durch wiederholte Tests auf der Vergleichsmenge ermittelt. Hierbei hat sich gezeigt, dass diese Parameter zum Teil große Auswirkungen auf den Verlauf der Optimierung haben. Zur Verdeutlichung dieser Situation wird eine Variation dieser Parameter für das Beispiel *helsby* durchgeführt. Das Beispiel besteht aus 1408 Optimierungsvariablen und 1399 Gleichungsnebenbedingungen und hat eine lineare Zielfunktion, sowie nichtlineare Nebenbedingungen. In Abbildung 7.18 sind die Haupt- und Nebeniterationen der resultierenden Optimierungsläufe abgedruckt.
Qualitativ unterscheiden sich die Ergebnisse für Haupt- und Nebeniterationen kaum. In beiden Abbildungen ist eine Vielzahl von Parametereinstellungen zu erkennen, die zu einer Erhöhung der Iterationszahlen des Optimierungslaufes führen. Gleichzeitig finden sich auch viele Einstellungen die verringerte Anzahlen von Iterationen liefern. Die Standardeinstellung für die erlaubte Abweichung der Zielfunktion ist $\nu = 0{,}1$. Es sei an dieser Stelle an die lokale Gültigkeit des Sensitivitätssatzes erinnert. Auf Grund der bezüglich der Störungsgröße begrenzten Aussagekraft ist es nicht sinnvoll, zu starke Störungen zu erlauben. Trotzdem fällt auf, dass es im Bereich um 0,55 ein Minimum in der Anzahl der Hauptiterationen gibt und die Kurve



(a) Hauptiterationen (b) Nebeniterationen

Abbildung 7.18: Variation der Parameter θ_κ und ν für das Problem (*helsby*). Die blaue und rote Kurve zeigen die Anzahl der resultierenden Hauptiterationen an und die orange und schwarze Kurve die entsprechende Anzahl der benötigten Nebeniterationen. Die grüne Linie repräsentiert die Ausgangsgröße für die Standardregularisierung.

lokal einer Parabel entspricht. Die Verwendung der Krümmungsbedingung liefert bei diesem Beispiel scheinbar chaotische Veränderungen der Iterationszahl. Positiv fällt dabei aber auf, dass es viele Einstellungen gibt, die trotzdem geringere Iterationszahlen als die Standardkonfiguration liefern.
Zusammenfassend ergibt sich, dass die gewählten Standardeinstellungen in der Lage sind, auf die gesamte Testmenge betrachtet, die Ergebnisse zu verbessern und eine leichte Effizienzsteigerung des Algorithmus zu erwirken. Ob die Verwendung der sensitivitätsbasierten Regularisierung sinnvoll ist, muss aber im Einzelfall kritisch hinterfragt werden.

7.3.3 Parallelisierungsansätze

Neben den bereits analysierten Anpassungen des Optimierungsalgorithmus wurden im Kapitel 6 verschiedene Möglichkeiten zur Parallelisierung der nichtlinearen Optimierung vorgestellt. Ein besonderer Schwerpunkt lag in der Entwicklung der Mehrkernschnittstelle für den Löser WORHP. Diese wird im folgenden Abschnitt mit verschiedenen Konfigurationen und für unterschiedliche Einsatzzwecke analysiert.
Abschließend werden Laufzeitvergleiche bei Parallelisierung der linearen Algebra innerhalb von WORHP durchgeführt.

7.3.3.1 Mehrkernschnittstelle

Basierend auf der Idee einer Master-Slave-Architektur wurde in Abschnitt 6.3 eine Mehrkernschnittstelle für WORHP vorgestellt. Zunächst wird der First-Across-The-Line-Modus zur Verbesserung der Robustheit und seiner Auswirkung auf die Lösungsgeschwindigkeit getestet. Anschließend wird der Parameterindividualisierungsmodus zur Anpassung der Konfiguration des Lösers für konkrete Testbeispiele angewendet.
Innerhalb der CUTEst-Testsammlung traten im Zusammenhang mit der Mehrkernschnittstelle teilweise Fehler in der parallelen Auswertung der Modellfunktionen auf, so dass die Beispiele *rosepetal* (sequentiell: Timeout) und *bleachng* (sequentiell: Optimale Lösung) zusätzlich entfernt wurden. Die Einbindung der CUTEst-Sammlung an die Mehrkernschnittstelle erfolgt über die eigens dafür entwickelte C++-Schnittstelle, im Gegensatz zur FORTRAN-Schnittstelle für die Anbindung an das Reverse-Communication-Interface von WORHP. Auf Grund der speziellen Anforderungen an die Sortierung der Hesse-Matrix innerhalb von WORHP müssen der Zielfunktionsanteil und der Nebenbedingungsanteil der Hesse-Matrix getrennt berechnet werden und innerhalb des Schnittstellencodes verrechnet werden. Diese Berechnungen führen auf Grund der unterschiedlichen Programmiersprachen zu minimalen Abweichungen, so dass die Beispiele *bdry2, bridgend, elec, gulfne, helsby, lukvli4, nvxqp1 - nvxqp7, roszman1, saro* in der Mehrkernschnittstelle mit nur einem Thread minimal abweichendes Verhalten zur sequentiellen Schnittstelle aufweisen. Um derartige Probleme in den folgenden Vergleichen zu eliminieren wurden diese Beispiele ebenfalls aus der betrachteten Testmenge entfernt. Insgesamt verbleiben somit 1234 Beispiele in der Sammlung für die weiteren Untersuchungen. Bei Verwendung der sequentiellen Variante werden davon 1112 Beispiele optimal gelöst, 16 akzeptabel und 106 Optimierungen schlagen fehl.

7.3.3.1.1 First-Across-The-Line Die Mehrkernschnittstelle wird verwendet, um in parallel verschiedene Konfigurationen von WORHP dasselbe Problem lösen zu lassen. Im First-Across-The-Line-Modus wird der Optimierungslauf gestoppt nachdem ein Thread eine optimale Lösung des zu bearbeitenden Problems gefunden hat. Schlagen einzelne Varianten des Lösers fehl, suchen die verbleibenden Threads weiterhin nach einer Lösung. Dieser Ansatz wurde für verschiedene Anzahlen von parallelen Konfigurationen mit bis zu acht Threads angewendet. Die hierfür verwendeten Einstellungen sind in Tabelle 7.11 abgedruckt. In Tabelle 7.12 ist die Anzahl der optimal und akzeptabel gelösten Probleme in Abhängigkeit der Anzahl an Konfigurationen aufgetragen. Beide Werte werden mit dem sequentiellen Lauf verglichen. Die abgedruckten Ergebnisse verdeutlichen, dass der First-Across-The-Line-Ansatz eine deutliche Steigerung der Robustheit des Optimierungsverfahrens erzielt. Bei der Verwendung von mindestens sechs parallelen Threads konnten insgesamt 31 zusätzliche Beispiele optimal gelöst werden. Es zeigt sich, dass bei mehr als sieben Threads kaum noch Verbesserungen auftreten. Die Anzahl der akzeptabel gelösten

Thread	Beschreibung
1	Standard
2	Nebenbed. (mit Kontrakt.)
3	Standard (BFGS)
4	Bewertungsfunktion
5	Sens. Relax. $\theta = 0{,}25$
6	(2) und (5)
7	Sens. Regularisierung (Abweichung Zielfunktion)
8	Wechselstrategie: BFGS (3 Iter.)

Tabelle 7.11: Konfigurationen der einzelnen Threads für die Auswertungen des First-Across-The-Line-Ansatzes

# Threads	# Optimal (FATL)	Änd.	# Akzeptabel (FATL)	Änd.
1	1112 (90,11%)	(-)	16 (1,30%)	(-)
2	1119 (90,68%)	+7	10 (0,81%)	-6
3	1131 (91,65%)	+19	10 (0,81%)	-6
4	1136 (92,06%)	+24	9 (0,73%)	-7
5	1142 (92,54%)	+30	8 (0,65%)	-8
6	1143 (92,63%)	+31	8 (0,65%)	-8
7	1143 (92,63%)	+31	8 (0,65%)	-8
8	1143 (92,63%)	+31	6 (0,49%)	-10

Tabelle 7.12: Ergebnisse für verschiedene Threadanzahlen. Änderungen bezogen auf den Vergleichslauf mit nur einem Thread.

Beispiele sinkt bei acht Threads auf Grund der zeitlichen Begrenzung von 30 Minuten pro Optimierungslauf sogar leicht.
Die Aussagekraft einer Laufzeitanalyse der verschiedenen Läufe fällt gering aus. Die Reihenfolge der verwendeten Konfigurationen ist in dieser Betrachtung entscheidend für die beobachtbaren Auswirkungen der Parallelisierung. Bei Verwendung der bisherigen Standardkonfiguration kann bei Verwendung von maximal acht Threads beispielsweise eine Beschleunigung um den Faktor 1,5 erzielt werden. Wird stattdessen als Referenzkonfiguration die Zulässigkeitskorrektur aktiviert, ergibt sich als Beschleunigungsfaktor lediglich 1,15. Insgesamt zeigen diese Beobachtungen, dass die vorgeschlagene Parallelisierung von Konfigurationen eine deutliche Steigerung der Robustheit des Verfahrens zur Folge hat, eine mit der Anzahl verfügbarer Threads skalierende Geschwindigkeitssteigerung ergibt sich aber nicht.
Die Struktur des First-Across-The-Line-Modus wirft die Frage auf, wie oft welche Einstellung die optimale Lösung geliefert hat. In Tabelle 7.13 wird diese Frage in Abhängigkeit von der Anzahl der verwendeten Threads beantwortet. Die Daten sind hierbei auf die Optimierungsläufe reduziert, welche dasselbe lokale Minimum geliefert haben. Auch diese Ergebnisse sind wie die Laufzeitanalyse von der Reihen-

Threads	Einst. 1	Einst. 2	Einst. 3	Einst. 4	Einst. 5	Einst. 6	Einst. 7	Einst. 8
1	892	(-)	(-)	(-)	(-)	(-)	(-)	(-)
2	575	317	(-)	(-)	(-)	(-)	(-)	(-)
3	456	266	170	(-)	(-)	(-)	(-)	(-)
4	294	267	221	110	(-)	(-)	(-)	(-)
5	212	226	227	89	138	(-)	(-)	(-)
6	226	187	218	78	77	106	(-)	(-)
7	147	166	220	128	70	118	43	(-)
8	147	115	184	96	115	115	41	79

Tabelle 7.13: Aufteilung der Konfigurationen bei Verwendung des First-Across-The-Line-Modus

folge abhängig. Die Tabelle zeigt, dass die gewählten Konfigurationen als eine gute Kombination betrachtet werden können, da keine Einstellung den anderen deutlich überlegen ist.

```
##        WMCI parameter identification mode          ##
### Least major iterations required by these settings ###

...

Thread 79:
--------
par->NLPmethod = 3;
par->RefineFeasibility = 3; par->SteffensenOnRefine = true;
par->UpdateMu = true; par->RefineFeasTermination = 1;
par->MoreRelax = false; par->AdaptiveConstrRelax = true;
par->PostQPSensRelaxPen = true; par->PostQPSensRelaxFrac = 75e-2;
par->PostQPSensRegVal = 1; par->PostQPSensRegValFrac = 25e-2;
par->ScaledKKT = true;
par->LowPassFilter = true;

Final values after iteration 7:
Final objective value ..............  1.4739523988E+05
Final constraint violation ........  5.0407236551E-08
Final KKT conditions ...............  9.1518296183E-10
Successful termination: Optimal Solution Found.

...

------------------------------------------------------------

### Best objective value achieved by these settings ###

Thread 1:
--------
par->NLPmethod = 3;
par->RefineFeasibility = 3; par->SteffensenOnRefine = true; par->UpdateMu = true;
par->RefineFeasTermination = 1;
par->MoreRelax = false;
par->PostQPSensRelaxPen = false;
par->PostQPSensRegVal = 0;
par->ScaledKKT = true;
par->LowPassFilter = true;

Final values after iteration 8:
Final objective value ..............  1.3505310868E+05
Final constraint violation ........  4.6736539842E-07
Final KKT conditions ...............  3.3572780562E-10
Successful termination: Optimal Solution Found.
------------------------------------------------------------
```

Listing 7.1: Ergebnisse des Parameterindividualisierungsmodus für das Beispiel *dtoc6*

Die erzielte Verteilung der gewählten Konfigurationen ist somit zufriedenstellend. Es lässt sich aber bereits eine Sättigung bei Verwendung von mehr als sechs Threads beobachten.

Insgesamt zeigen die erzielten Ergebnisse, dass die Parallelisierung verschiedener Konfigurationen innerhalb von WORHP eine gute Möglichkeit zur Effizienzsteigerung bietet. Wie zu erwarten war, kann diese Art der Parallelisierung aber nicht als beliebig skalierbare Technik für große Threadanzahlen verwendet werden.

7.3.3.1.2 Parameterindividualisierung Die Aufteilung der verschiedenen Konfigurationen bei Verwendung des First-Across-The-Line-Modus in Tabelle 7.13 motiviert die Anwendung des Parameterindividualisierungsmodus. Dieser Modus erlaubt es dem Benutzer für eine konkrete Aufgabenstellung die beste Konfiguration des Lösers aus einer vorgegebenen Menge von Einstellungen zu finden.
Die Vorgehensweise soll anhand des Beispiels *dtoc6* aus der CUTEst-Sammlung vorgestellt werden. Das Beispiel besteht aus nichtlinearer Zielfunktion, nichtlinearen Nebenbedingungen und hat 10001 Optimierungsvariable und 5000 Nebenbedingungen. Mit Hilfe der automatischen Parametervariation wurden 100 der 648 möglichen Konfigurationen aus Listing A.4 verwendet und die entsprechenden Optimierungsprobleme in parallel gelöst. Die Ergebnisse sind in Listing 7.1 aufgeführt. Mit verschiedenen Konfigurationen wurde die Anzahl der benötigten Hauptiterationen von WORHP auf 7 reduziert. Unter der Nummer des Threads sind die jeweils verwendeten Parametereinstellungen aufgelistet.
Nach 7 Iterationen konnte ein Zielfunktionswert von $1{,}474 \cdot 10^5$ erreicht werden. Im Gegensatz dazu konnte mit der Konfiguration von Thread 1 der beste Zielfunktionswert von $1{,}351 \cdot 10^5$ nach 8 Iterationen erreicht werden.

7.3.3.2 Lineares Gleichungssystem

In dieser Arbeit wurde bereits mehrfach erwähnt, dass der numerisch aufwendigste Teil eines SQP-Verfahrens die Lösung der linearen Gleichungssysteme ist. Innerhalb von WORHP wird zur Lösung dieser Systeme der Löser MA97 verwendet. Verschiedene Parametereinstellungen aktivieren die Parallelisierung innerhalb von MA97. Zur Vervollständigung der hier vorgestellten Betrachtungen zur Parallelisierung von WORHP wurde MA97 mit verschiedenen Threadanzahlen unter Verwendung der angebotenen OpenMP-Parallelisierung [66] zum Vergleich getestet. Für den Test wurde das von Wassel [109] entwickelte Skript verwendet, welches parallel verschiedene Prozesse von WORHP startet. Es wurden fünf Prozesse in parallel gestartet und für diese wurde die Anzahl der zur Verfügung stehenden Threads für OpenMP variiert. In Tabelle 7.14 sind die benötigten Rechenzeiten für das gesamte Testset abgedruckt.
Die Ergebnisse in der Tabelle zeigen, dass durch die Verwendung paralleler linearer Algebra, wie zu erwarten, eine Beschleunigung der Optimierungsläufe erzielt werden kann. Je nach Anzahl der zur Verfügung stehenden Kerne ist es somit sinnvoll die beiden vorgestellten Parallelisierungsansätze zu kombinieren. Im direkten Vergleich bewirkt eine Parallelisierung der linearen Algebra eine gute Beschleunigung, wohin-

gegen die Verwendung verschiedener Konfigurationen zusätzliche Robustheit für das Verfahren liefert.

OpenMP-Threads	Rechenzeit	Faktor
1	7h 50m 43s	1,0
2	4h 25m 33s	1,77
3	2h 44m 57s	2,85
4	2h 22m 19s	3,31

Tabelle 7.14: Geschwindigkeit bei der Verwendung von OpenMP-Parallelisierung von MA97 innerhalb von WORHP

Kapitel 8

Schlussbetrachtung

Inhaltsangabe

Zum Abschluss der Arbeit gibt dieses Kapitel eine kurze Zusammenfassung der entwickelten Techniken und der erzielten Ergebnisse. Weiterhin bietet der Ausblick weiterführende Anwendungsmöglichkeiten, basierend auf den vorgestellten Ideen und Algorithmen.

8.1 Zusammenfassung

Das Ziel der vorliegenden Arbeit war die Entwicklung verschiedener Techniken zur Effizienzsteigerung nichtlinearer Optimierungsverfahren. Zunächst wurde die Optimierungstheorie und insbesondere die numerische Lösung von nichtlinearen Optimierungsproblemen ausführlich beschrieben. Weiterhin thematisiert die Arbeit detailliert die innerhalb des Lösers WORHP implementierten Techniken. Auf Grundlage dieser Beschreibungen wurden verschiedene Möglichkeiten zur Effizienzsteigerung entwickelt.
Die Möglichkeit bei der Lösung eines nichtlinearen Optimierungsproblems verschiedene Varianten der Hesse-Matrix innerhalb der zu lösenden quadratischen Unterprobleme zu verwenden, führte auf die Idee, eine Wechselstrategie einzuführen. Auf Grund der Tatsache, dass BFGS-Matrizen keine Regularisierung benötigen und deshalb die heuristische Ermittlung eines passenden Regularisierungswertes entfallen kann, bietet es sich an, während der Globalisierungsphase anstelle der analytischen Hesse-Matrix, oder einer Näherung durch finite Differenzen, BFGS-Matrizen zu verwenden. Die Ergebnisse haben gezeigt, dass diese Strategie in der Lage ist, sowohl die Rechengeschwindigkeit, als auch die Robustheit der Optimierung zu verbessern. Neben der allgemeinen Theorie der nichtlinearen Optimierung wurden auch die

Grundlagen der parametrischen Sensitivitätsanalyse angeführt. Diese Theorie diente zur Erweiterung der existierenden Techniken zur Korrektur innerhalb des SQP-Verfahrens. Zunächst wurde die Zulässigkeitskorrektur zur Verbesserung der Behandlung von Nichtlinearitäten in den Nebenbedingungen während der Optimierung analysiert. Ein eingehendes Studium dieses Algorithmus lieferte passende Abbruchbedingungen. Die Einbindung einer Extrapolationsstrategie verbesserte die Korrektur zusätzlich. Die numerischen Auswertungen haben gezeigt, dass der Korrekturalgorithmus in der Lage ist, die Anzahl benötigter Hauptiterationen für große Teile der betrachteten Testsammlung erheblich zu reduzieren. Gleichzeitig erzielte die Anwendung des Algorithmus ebenfalls eine Steigerung der Robustheit.
Weiterhin wurde die Relaxierung der Nebenbedingungen innerhalb der quadratischen Unterprobleme analysiert. Dies führte zu zwei wesentlichen Neuerungen. Einerseits ermöglicht die adaptive Relaxierung, in Anlehnung an eine Aktive-Mengen-Strategie, adaptiv die Anzahl der Relaxierungsvariablen anzupassen und gezielt störende Nebenbedingungen einzeln zu relaxieren. Die Einführung zusätzlicher Relaxierungsvariablen führt zu einem gestiegenen Bedarf an Rechenzeit, gleichzeitig aber auch zu einer Steigerung der Anzahl der gelösten Beispiele.
Neben der adaptiven Relaxierung wurde andererseits ein sensitivitätsbasierter Algorithmus vorgestellt, der es erlaubt den Bestrafungsparameter für die Relaxierung durch Hinzunahme der parametrischen Sensitivitäten der Unterprobleme anzupassen. Diese Technik ermöglicht eine problembezogene Anpassung der Relaxierung, wohingegen die Standardimplementierung für jedes Beispiel die gleichen Bestrafungswerte verwendet. Die numerischen Auswertungen haben gezeigt, dass diese Technik eine deutliche Geschwindigkeitssteigerung bewirken kann und gleichzeitig zu einer Steigerung der Anzahl der gelösten Beispiele führt. Die weiteren Untersuchungen belegten außerdem, dass die gleichzeitige Verwendung dieser Strategie mit der oben erwähnten Zulässigkeitskorrektur zu noch besseren Ergebnissen führt.
Als letzte sensitivitätsbasierte Techniken wurden zwei neue Regularisierungsstrategien für die Hesse-Matrix vorgestellt. Die Krümmung der Hesse-Matrix entlang der Suchrichtung lieferte in diesem Zusammenhang eine Schätzung für einen passenden Regularisierungswert. Die parametrischen Sensitivitäten ermöglichen eine Quantifizierung der Auswirkungen der Regularisierung auf den Zielfunktionswert der Unterprobleme. Eine Beschränkung der erlaubten Veränderung durch die Regularisierung hilft ebenfalls einen Schätzwert für die nächste Iteration zu ermitteln. Die Auswertungen der Techniken haben gezeigt, dass die Einbindung dieser Regularisierungsstrategien eine Geschwindigkeits- und Robustheitssteigerung des Gesamtalgorithmus liefert.
Als letzter zentraler Punkt wurden in der vorliegenden Arbeit verschiedene Parallelisierungsmöglichkeiten vorgestellt. Ein besonderer Fokus lag hierbei auf der Einführung einer Mehrkernschnittstelle für WORHP. Diese ermöglicht die parallele Ausführung diverser Konfigurationen des Lösers. Es sind verschiedene Anwendungsszenarien dieser Schnittstelle denkbar. Der Start mehrerer Instanzen gleichzeitig liefert eine Steigerung der Robustheit sowie eine leichte Beschleunigung der Opti-

mierung. Nach erfolgreicher Terminierung einer Optimierungskonfiguration werden die übrigen Instanzen abgebrochen. Diese Technik erlaubt es neben einer robusten Standardkonfiguration auch experimentelle Strategien zu versuchen. Die numerischen Untersuchungen zeigen, dass die Parallelisierung von Konfigurationen den relativen Anteil der optimal gelösten Beispiele der Testsammlung von 90,11% auf 92,63% verbessern konnte.

8.2 Ausblick

Die sensitivitätsbasierte Anpassung der Relaxierung und Regularisierung führte zu einer Reduktion der Anzahl der Aufrufe des internen Unterproblemlösers. Eine weitere Effizienzsteigerung könnte in diesem Zusammenhang durch eine direkte Implementierung der beschriebenen Strategien innerhalb des Lösers für die quadratischen Unterprobleme erzielbar sein.
Die vorgestellte Mehrkernschnittstelle des Lösers eröffnet neben den innerhalb dieser Arbeit beschriebenen Strategien zur Effizienzsteigerung weitere Möglichkeiten, um von der parallelen Lösung verschiedener Optimierungsprobleme zu profitieren. Basierend auf der beschriebenen Architektur befindet sich bereits eine Schnittstelle zur Lösung von mehrkriteriellen Optimierungsproblemen in der Entwicklung. Weiterhin dienten in der Arbeit Heuristiken zur globalen Optimierung als motivierende Beispiele. In einer zukünftigen Erweiterung der vorgestellten Schnittstelle sollte auch dieses Anwendungsszenario umgesetzt werden.
Die Motivation zur Parallelisierung mehrerer Löseraufrufe mit unterschiedlichen Parametersetzungen war die Tatsache, dass je nach Aufgabenstellung andere Konfigurationen des Lösers effizienter in der Lage sind, das jeweilige Optimierungsproblem zu lösen. Diese Beobachtungen werfen die Frage auf, ob es möglich ist, einen adaptiven Optimierungsalgorithmus zu entwickeln, der automatisch die internen Konfigurationen anpasst und beispielsweise die vorgestellten sensitivitätsbasierten Strategien aktiviert, wenn eine Effizienzsteigerung zu erwarten ist. Bei einer derartigen Umsetzung stellt sich weiterhin die Frage nach geeigneten Kriterien zur Bewertung eines konkreten Optimierungsproblems. Umgebungen wie CUTEst stellen zur Zeit lediglich Informationen über Linearität von Nebenbedingungen zur Verfügung. Ein weiterer Anknüpfungspunkt ist die Frage, ob anhand des Löserverhaltens möglicherweise eine Clusteranalyse einer Sammlung von Problemen umsetzbar ist. Denkbar wäre es dabei Teilmengen der Problemsammlungen der jeweils besten Löserkonfiguration zuzuordnen. Anhand dieser Analyse müsste versucht werden, Merkmale zur Einordnung weiterer Problemstellungen zu identifizieren und so ein adaptives Verhalten des Lösers zu ermöglichen.
Die vorgestellte Einbindung paralleler Funktionsauswertungen innerhalb der Reverse-Communcation wurde auf Grund des resultierenden unübersichtlichen Programmcodes auf Seiten des Benutzers nicht weiter verfolgt. Durch eine geeignete Erweiterung der bestehenden Mehrkernschnittstelle könnte dieses Defizit vor dem

Anwender versteckt werden, so dass dieser von der möglichen Effizienzsteigerung ohne die aktuell bestehenden Nachteile profitieren könnte. Damit ein maximaler Nutzen für den Benutzer entsteht, müssten gleichzeitig die aktuell innerhalb des Lösers implementierten finiten Differenzen überarbeitet werden, um von der Verfügbarkeit paralleler Funktionsauswertungen profitieren zu können.

Anhang A

Konfigurationsskript Mehrkernschnittstelle

Die Variation der Parameter für die Mehrkernschnittstelle in Abschnitt 6.3 wird vom Anwender über eine Skriptsprache konfiguriert. In dem bereitzustellenden Skript werden einzelne Einstellungen gruppiert, so dass diese gegeneinander austauschbar sind. Für einen konkreten Optimierungslauf wird aus jeder angegebenen Gruppe eine Einstellung gewählt.

A.1 Standardkonfigurationsdatei

In dem Listing A.1 ist die vorgefertigte Konfigurationsdatei für die Mehrkernschnittstelle abgedruckt. Die Datei enthält die folgenden Gruppierungen:

i. Einstellungen für die Hesse-Matrix

ii. Verschiedene Methoden für die Liniensuche

iii. Konfiguration der Zulässigkeitskorrektur

iv. Variation der Anzahl der Relaxierungsvariablen

v. Aktivierung der sensitivitätsbasierten Relaxierung

vi. Strategien zur sensitivitätsbasierten Regularisierung

vii. Anpassung des linearen Lösers (Vergleiche [66])

viii. Skalierung der Abbruchbedingungen (Vergleiche [98])

ix. Einstellung des Tiefpassfilters (Vergleiche [98])

Werden alle Gruppierungen gleichzeitig für einen Parameterindividualisierungslauf verwendet führt dies auf 7776 verschiedene Konfigurationen. Deshalb ist es sinnvoll, dass der Benutzer sich auf einige der Gruppierungen beschränkt und diese für sein konkretes Problem verwendet, um den Rechenaufwand zu begrenzen.

```
1  ########################################
2  ## File is organised as follows:       #
3  ## (Setting) (optional: Requirement)   #
4  ########################################
6  # Parameter Settings for the Hessian
7  (par->UserHM = true) (par->UserHM == true)
8  (par->FidifHM = false && par->BFGSmethod = 2 &&
    par->UserHM = false) (par->UserHM == true)
9  (par->FidifHM = false && par->BFGSmethod = 102 &&
    par->UserHM = false) (par->UserHM == true)
10 (par->FidifHM = false && par->BFGSmethod = 1 &&
    par->UserHM = false && par->BFGSminblockSize = 1 &&
    par->BFGSmaxblockSize = 3) ()
11 (par->SwitchMode = 1 && par->SwitchModeTermination = 0 &&
    par->SwitchModeTermTol = 1e-1) (par->UserHM == true ||
    par->FidifHM == true)
12 (par->SwitchMode = 1 && par->SwitchModeTermination = 2 &&
    par->SwitchModeMaxIter = 3) (par->UserHM == true ||
    par->FidifHM == true)
13 # General Method Parameters
14 (par->NLPmethod = 3) (par->qp.ipLsMethod == 1)
15 (par->NLPmethod = 1) ()
16 # Feasibility Refinement
17 (par->RefineFeasibility = 3 && par->SteffensenOnRefine = true &&
    par->UpdateMu = true && par->RefineFeasTermination = 1)
   (opt->m > 0)
18 (par->RefineFeasibility = 3 && par->SteffensenOnRefine = true &&
    par->UpdateMu = false && par->RefineFeasTermination = 1)
   (opt->m > 0)
19 (par->RefineFeasibility = 0) (opt->m > 0)
20 # Number of Relaxation Variables
21 (par->MoreRelax = false) (opt->m > 0)
22 (par->MoreRelax = false && par->AdaptiveConstrRelax = true)
   (opt->m > 0)
23 (par->MoreRelax = true) (opt->m > 0)
24 # Sensitivity based relaxation
25 (par->PostQPSensRelaxPen = false) (opt->m > 0)
26 (par->PostQPSensRelaxPen = true &&
    par->PostQPSensRelaxFrac = 75e-2)
   (opt->m > 0)
27 (par->PostQPSensRelaxPen = true &&
    par->PostQPSensRelaxFrac = 25e-2)
   (opt->m > 0)
28 # Sensitivity based regularisation
29 (par->PostQPSensRegVal = 0) (par->UserHM == true ||
```

```
    par->FidifHM == true)
30 (par->PostQPSensRegVal = 1 && par->PostQPSensRegValFrac = 25e-2)
   (par->UserHM == true || par->FidifHM == true)
31 (par->PostQPSensRegVal = 2 && par->PostQPSensRegValObjDev = 1e-1)
   (par->UserHM == true || par->FidifHM == true)
32 # MA97 u
33 (par->MA97u = 0.1) (par->qp.ipLsMethod == 1)
34 (par->MA97u = 0.01) (par->qp.ipLsMethod == 1)
35 # ScaledKKT for Thoroughness
36 (par->ScaledKKT = true) ()
37 (par->ScaledKKT = false) ()
38 # LowPassFilter for Thoroughness
39 (par->LowPassFilter = true) ()
40 (par->LowPassFilter = false) ()
```

Listing A.1: Vorgefertigte Parametervariationen für die Mehrkernschnittstelle

Die verschiedenen Einstellungen sind so gewählt, dass für die meisten Situationen eine passende Konfiguration enthalten ist.

A.2 Erklärungen zum Konfigurationsskript

Im Listing A.2 ist ein kurzer Ausschnitt der Standard-Konfigurationsdatei für die Parametervariationen in der Skriptsprache abgedruckt.

```
1  ########################################
2  ## File is organised as follows:       #
3  ## (Setting) (optional: Requirement)   #
4  ########################################
5  # General Method Parameters
6  (par->NLPmethod = 3) (par->qp.ipLsMethod == 1)
7  (par->NLPmethod = 1) ()
8  # Parameter Settings for the Hessian
9  (par->UserHM = true) (par->UserHM == true)
10 (par->FidifHM = false && par->BFGSmethod = 2 &&
    par->UserHM = false) (par->UserHM == true)
11 # Feasibility Refinement
12 (par->RefineFeasibility = 2 &&
    par->SteffensenOnRefine = true) (opt->m > 0)
13 (par->RefineFeasibility = 0) (opt->m > 0)
```

Listing A.2: Mögliche Einstellungen in Skriptsprache für die automatische Codegenerierung zur Variation der gewünschten Parameter

Innerhalb der Konfigurationsdatei werden untereinander austauschbare Parametereinstellungen gruppiert. Eine neue Gruppe wird durch eine Überschrift, welche mit „#“ startet, eingeleitet. Jede Zeile einer Gruppe besteht aus zwei geklammerten Teilen. Innerhalb der ersten Klammer steht der Code (in C++), um die gewünschte Parametereinstellung vorzunehmen. Die zweite Klammer enthält eventuell zu prüfende

Voraussetzungen. Die erste Gruppe im oben abgedruckten Beispiel dient beispielsweise dazu die Liniensuche zu konfigurieren. Die Parametersetzung
„`NLPmethod = 3`“ entspricht dem in Abschnitt 3.2.2.2 beschriebenen Filter. Die Alternative ist die Verwendung einer Bewertungsfunktion. Soll der Filter verwendet werden, erfordert die innerhalb von WORHP implementierte Variante die Bestimmung der Signatur der KKT-Matrix. Diese steht aber nur zur Verfügung wenn als linearer Löser MA97 verwendet wird, deshalb muss die entsprechende Einstellung als Voraussetzung geprüft werden.
Der zweite Block veranschaulicht die Möglichkeit verschiedene Optionen mit Hilfe des „&&“-Operators zu kombinieren. Die Konfiguration in Zeile 10 des Ausschnitts soll ein Block-BFGS-Verfahren aktivieren. Dazu muss die vom Benutzer gegebene Hesse-Matrix über „`UserHM = false`“ und finite Differenzen über
„`FidifHM = false`“ deaktiviert werden. Gleichzeitig muss das passende BFGS-Verfahren über „`BFGSmethod`“ gewählt werden. Das gewählte BFGS-Verfahren erfordert in diesem Fall, dass eine Struktur für die Hesse-Matrix vorliegt. Deshalb wird als Voraussetzung verlangt, dass der Benutzer initial über „`UserHM = true`“ angegeben hat, dass er die Hesse-Matrix bereitstellt. Somit ist es möglich analytische Ableitungen zweiter Ordnung zu deaktivieren und gleichzeitig die zu verwendende alternative Strategie zu definieren.
Der letzte Block in den Zeilen $11 - 13$ aktiviert, beziehungsweise deaktiviert, die Zulässigkeitskorrektur inklusive der optionalen Steffensen-Extrapolation (Vergleiche Abschnitt 5.2). Offensichtlich ist diese Strategie nur sinnvoll, wenn das Problem beschränkt ist. Deshalb muss geprüft werden, ob die Anzahl der Nebenbedingungen „`o->m > 0`“ erfüllt[1].
Neben den bereits beschriebenen Funktionalitäten unterstützt die Skriptsprache zur Konfiguration der Parameter noch die Möglichkeit die Werte einzelner Parameter in einem gegebenen Intervall zu variieren. In dem Listing A.3 sind zwei Variationen implementiert.
Zeile 6 in dem Ausschnitt dient zur Konfiguration der Optionen der Zulässigkeitskorrektur aus Abschnitt 5.2. Die Angabe von `0 +1 3` bedeutet, dass die Parameterwerte zwischen 0 und 3 mit additiver Schrittweite 1 gesetzt werden. In diesem Fall bedeutet dies konkret, dass die Zulässigkeitskorrektur deaktiviert wird für den Wert 0. Der Wert 1 aktiviert die Korrektur erst nachdem die Iterierten das erste Mal zulässig gewesen sind, der Wert 2 entspricht der Korrektur für alle Nebenbedingungen während des gesamten Iterationsverlaufes und der Wert 3 korrigiert lediglich die aktiven Nebenbedingungen des jeweils gelösten Unterproblems. Mit Hilfe von Zeile 6 des Skriptes können demnach die verschiedenen Grundeinstellungen der Zulässigkeitskorrektur in einem Optimierungslauf in parallel ausgeführt werden.
Zeile 8 verdeutlicht im Gegensatz dazu die Möglichkeit, multiplikativ einen Parameter zu verändern. Durch die Eingabe von `5e-2 *10 5e-0` werden für den Parameter

[1] Den Notationen dieser Arbeit folgend, entspricht $m := N_g + N_h$.

`ArmijoSigma` die Werte $5e-2, 5e-1$ und $5e-0$ konfiguriert. Dieser Parameter entspricht dem σ aus Gleichung (3.80) zur Bestimmung der Schrittweite.

```
1  ########################################
2  ## File is organised as follows:       #
3  ## (Setting) (optional: Requirement)   #
4  ########################################
5  # Integer valued parameter
6  (par->RefineFeasibility 0 +1 3) (opt->m > 0)
7  # Double parameter
8  (par->ArmijoSigma 5e-2 *10 5e-0) (opt->m > 0)
```

Listing A.3: Variation einzelner Parameter mit Hilfe der Skriptsprache

A.3 Konfigurationsdateien der numerischen Tests

Bei der Vorstellung des Parameterindividualisierungsmodus in Abschnitt 7.3.3.1.2 wurde eine Auswahl der Möglichkeiten aus Listing A.1 verwendet. Die reduzierte Konfigurationsdatei ist in Listing A.4 abgedruckt.

```
1  ########################################
2  ## File is organised as follows:       #
3  ## (Setting) (optional: Requirement)   #
4  ########################################
5  # General Method Parameters
6  (par->NLPmethod = 3) (par->qp.ipLsMethod == 1)
7  (par->NLPmethod = 1) ()
8  # Feasibility Refinement
9  (par->RefineFeasibility = 3 && par->SteffensenOnRefine = true
   && par->UpdateMu = true && par->RefineFeasTermination = 1)
   (opt->m > 0)
10 (par->RefineFeasibility = 3 && par->SteffensenOnRefine = true
   && par->UpdateMu = false && par->RefineFeasTermination = 1)
   (opt->m > 0)
11 (par->RefineFeasibility = 0) (opt->m > 0)
12 # Number of Relaxation Variables
13 (par->MoreRelax = false) (opt->m > 0)
14 (par->MoreRelax = false && par->AdaptiveConstrRelax = true)
   (opt->m > 0)
15 (par->MoreRelax = true) (opt->m > 0)
16 # Sensitivity based relaxation
17 (par->PostQPSensRelaxPen = false) (opt->m > 0)
18 (par->PostQPSensRelaxPen = true &&
    par->PostQPSensRelaxFrac = 75e-2) (opt->m > 0)
19 (par->PostQPSensRelaxPen = true &&
    par->PostQPSensRelaxFrac = 25e-2) (opt->m > 0)
20 # Sensitivity based regularisation
21 (par->PostQPSensRegVal = 0) (par->UserHM == true ||
    par->FidifHM == true)
```

```
22 (par->PostQPSensRegVal = 1 && par->PostQPSensRegValFrac = 25e-2)
   (par->UserHM == true || par->FidifHM == true)
23 (par->PostQPSensRegVal = 2 && par->PostQPSensRegValObjDev = 1e-1)
   (par->UserHM == true || par->FidifHM == true)
24 # ScaledKKT for Thoroughness
25 (par->ScaledKKT = true) ()
26 (par->ScaledKKT = false) ()
27 # LowPassFilter for Thoroughness
28 (par->LowPassFilter = true) ()
29 (par->LowPassFilter = false) ()
```

Listing A.4: Konfiguration im Test des Parameterindividualisierungsmodus

Anhang B

Standardparametersetzungen

Die Auswertung der verschiedenen Techniken im Abschnitt 7.3 erfolgt anhand des Lösers WORHP. Die neu entwickelten Strategien werden jeweils über verschiedene Parameter aktiviert und konfiguriert. Für weitere Einblicke sei auf die Dokumentation des Lösers [98] verwiesen. Im Folgenden sind zur Reproduzierbarkeit der Ergebnisse die Standard-Einstellungen des Lösers abgedruckt.
In Tabelle B.1 sind die Toleranzen der Abbruchbedingungen der Optimierung abgedruckt. Spezielle weitere Abbruchbedingungen können über die Parameter aus Tabelle B.2 konfiguriert werden.
Zur Aktivierung und Deaktivierung von Ableitungen des Benutzers können die Parameter aus Tabelle B.3 verwendet werden.
Bei der Verwendung von finiten Differenzen als Ersatz für die Ableitungen des Benutzers bietet WORHP verschiedene Möglichkeiten die Verwendung zu vereinfachen. Die Standardeinstellungen der entsprechenden Parameter sind in Tabelle B.4 abgedruckt. Diese finden beispielsweise bei der Bewertung des Phasenwechsels in Abschnitt 7.3.2.1 ihre Anwendung.
Für dieselben Auswertungen sind weiterhin die Einstellungen der BFGS-Matrizen von entscheidender Bedeutung. Die Grundeinstellungen sind Tabelle B.5 entnehmbar.
Die dazugehörigen Einstellungen zur Konfiguration des Phasenwechsels sind in Tabelle B.6 zusammengefasst.
Die Zulässigkeitskorrektur wurde als sensitivitätsbasierte Strategie in Abschnitt

TolFeas	1.0000000000000000e-06
TolOpti	1.0000000000000000e-06
TolComp	1.0000000000000000e-03
AcceptTolFeas	1.0000000000000000e-03
AcceptTolOpti	1.0000000000000000e-03

Tabelle B.1: Parameter: Toleranzen

KeepAcceptableSol	True
LowPassFilter	True
TooBig	True
MaxCalls	2147483647
MaxIter	10000
Timeout	1.8000000000000000e+03
InftyUnbounded	1.0000000000000000e+20
LowPassAlphaF	9.5000000000000000e-01
LowPassAlphaG	9.5000000000000000e-01
TooBigCV	1.0000000000000000e+25
TooBigKKT	1.0000000000000000e+30
FJandND	True
sKKTOnlyAcceptable	False
ScaledKKT	True
BoundTolFac	1.0000000000000000e+03
CheckFJ	1.0000000000000000e+12
LowPassAlphaMerit	1.0000000000000000e-01

Tabelle B.2: Parameter: Abbruchbedingungen

UserDF	True
UserDG	True
UserHM	True
FGtogether	False
UserHMstructure	2

Tabelle B.3: Parameter: Ableitungen des Benutzers

5.2 eingeführt. Die verschiedenen Kriterien zur Individualisierung des Algorithmus können mit Hilfe der Parameter in Tabelle B.7 ausgewählt werden.
Basierend auf der Sensitivitätsanalyse der Unterprobleme wurden weiterhin Strategien bezüglich der Relaxierung und Regularisierung vorgestellt. Die verschiedenen Varianten können mit Hilfe der Parameter aus Tabelle B.8 konfiguriert werden.
Während der Erläuterungen der SQP-Methode innerhalb des Lösers WORHP wurde ein Schwerpunkt der Betrachtungen auf die Relaxierung der Nebenbedingungen gelegt. Zusätzlich wurde die adaptive Relaxierung als verfeinernde Technik vorgestellt. In Tabelle B.9 sind die Konfigurationsmöglichkeiten der Relaxierung abgedruckt.
Von zentraler Bedeutung innerhalb der numerischen nichtlinearen Optimierung liegt die Lösung der linearen Gleichungssysteme zur Lösung der Unterprobleme. Standardmäßig verwendet WORHP den Löser MA97 dafür. Einige Parameter des Lösers können direkt über die Parameterdatei von WORHP beeinflusst werden. Diese Pa-

FidifHM	False
FidifGroups	True
ScaledFD	True
CheckGroups	False
FirstDifCentral	True
SecondDifCentral	True
GroupMethod	1
MaxGPart	1
PairMethod	11
FidifEps	1.0000000000000000e-05

Tabelle B.4: Parameter: Finite Differenzen

BFGSmethod	2
BFGSmaxblockSize	3
BFGSminblockSize	1
BFGSrestart	44
CurvBCond	2.0000000000000000e-02
CurvBFac	3.0000000000000000e-01
CurvCond	2.0000000000000000e-02
CurvFac	3.0000000000000000e-01

Tabelle B.5: Parameter: BFGS

SwitchMode	0
SwitchModeMaxMinor	8
SwitchModeMaxIter	0
SwitchModeTermination	0
SwitchModeTolerances	0
SwitchModeIpComTol	4.0000000000000000e-05
SwitchModeIpResTol	4.0000000000000000e-06
SwitchModeLsTol	1.0000000000000000e-07
SwitchModeTermTol	1.0000000000000000e-03

Tabelle B.6: Parameter: Phasenwechsel

rameter sind in Tabelle B.10 aufgeführt.
Neben dem linearen Löser hat auch der interne Löser für die quadratischen Unterprobleme eigene Parameter. Diese können ebenfalls direkt konfiguriert werden. Eine Auflistung ist in Tabelle B.11 zu finden.

RefineOnlyOnAlpha	False
RefineFeasMerit	False
SteffensenOnRefine	True
UpdateMu	False
RefineFeasMaxIter	500
RefineFeasibility	0
RefineFeasTermination	0
RefineContrLimitc	8.0000000000000000e-01
RefineContrLimitq	1.0000000000000000e-03
RefineStartTol	1.0000000000000000e-06
RefineMaxRelax	9.5000000000000000e-01
RefineMaxHMReg	1.0000000000000000e+03

Tabelle B.7: Parameter: Zulässigkeitskorrektur

PostQPSensRegVal	0
PostQPSensRelaxPen	False
PostQPSensitivity	0
PostQPSensRelaxFrac	7.5000000000000000e-01
PostQPSensRegValFrac	7.5000000000000000e-01
PostQPSensRegValObjDev	1.0000000000000000e-01

Tabelle B.8: Parameter: Sensitivitätskorrektur Relaxierung und Regularisierung

RelaxCon	True
AdaptiveConstrRelax	False
MoreRelax	False
RelaxPenOnlyOne	False
RelaxMaxDelta	9.2000000000000000e-01
RelaxMaxPen	1.0000000000000000e+07
RelaxRho	4.8000000000000000e+00
RelaxStart	1.0000000000000000e+00

Tabelle B.9: Parameter: Relaxierung der Nebenbedingungen

In den theoretischen Betrachtungen der Algorithmen zur Lösung nichtlinearer Optimierungsprobleme wurde in Abschnitt 3.2.2 die Liniensuche zur Bestimmung der nächsten Iterierten ausführlich thematisiert. Die verschiedenen Varianten und Einstellungen können über die Parameter in Tabelle B.12 gesteuert werden.
Alle übrigen Parameter zur Anpassung des Lösers WORHP sind in Tabelle B.13 zusammenfasst.

MA97blas3	False
MA97mf	False
MA97ordering	5
MA97scaling	0
MA97print	-1
MA97nemin	8
MA97factorMin	20000000
MA97small	1.0000000000000000e-20
MA97u	1.0000000000000000e-10
MA97umax	1.0000000000000000e-04

Tabelle B.10: Parameter: MA97

qp.ipComTol	2.0000000000000000e-07
qp.ipResTol	4.0000000000000000e-08
qp.lsTol	1.0000000000000000e-09
qp.strict	True
qp.maxIter	500
qp.ipTryRelax	True
qp.ipRelaxDiv	2.0000000000000000e+00
qp.ipRelaxMax	1.0000000000000000e-12
qp.ipRelaxMin	9.1000000000000000e-10
qp.ipRelaxMult	1.0000000000000000e+01
qp.ipBarrier	8.9000000000000000e+00
qp.ipFracBound	8.8000000000000000e-01
qp.ipMinAlpha	1.0000000000000000e-11
qp.nsnGradStep	True
qp.nsnLsMethod	4
qp.nsnBeta	9.0000000000000000e-01
qp.nsnKKT	1.0000000000000000e-06
qp.nsnMinAlpha	1.0000000000000000e-11
qp.nsnSigma	1.0000000000000000e-02
qp.lsScale	True
qp.lsTrySimple	False
qp.lsItMaxIter	1000
qp.lsItMethod	0
qp.lsItPrecondMethod	0
qp.lsRefineMaxIter	10
qp.ipLsMethod	1
qp.scaleIntern	False
qp.method	1
qp.printLevel	0

Tabelle B.11: Parameter: QPSOL

NLPmethod	3
LinMult	False
ArmijoBeta	8.9500000000000000e-01
ArmijoBetaAres	8.9500000000000000e-01
ArmijoMaxAlpha	9.9999999999990000e-01
ArmijoMinAlpha	5.0000000000000000e-07
ArmijoMinAlphaRec	1.0000000000000000e-06
ArmijoSigma	5.0000000000000000e-03
AlphaMinConst	False
FilterBisecAlpha	False
FilterIntersecAlpha	True
IgnoreFilterCrit	True
ReinitFilter	True
MaxNorm	True
FilterRestFeas	True
RestUntilFeas	True
MaxLScounter	13
RegStrategy	3
FilterGammaCV	5.0000000000000000e-04
FilterGammaF	5.0000000000000000e-03
GammaAlpha	1.0000000000000000e-02
MinBettsTau	6.0000000000000000e-16
ReduceBettsTau	5.5000000000000000e-01
SwitchingDelta	1.0000000000000000e-02
SwitchingSF	1.1000000000000000e+00
SwitchingSCV	1.1000000000000000e+00
TakeQPSol	False
MeritFunction	4
PenUpdEpsKSequence	2
PenUpdEpsBar	9.0000000000000000e-01
PenUpdEpsKFac	2.0000000000000000e+00
PenUpdMaxDeltaK	1.1000000000000000e+01
PenUpdMaxFac	1.0000000000000000e+08
PenUpdRBar	2.0000000000000000e+00
MeritGradTol	2.2204460492503131e-16

Tabelle B.12: Liniensuche; Konfiguration des Filters beziehungsweise der Bestrafungsfunktion

ShowMonitor	False
GiveNewEntriesX	False
LogLevel	0
LogResult	0
NLPprint	2
CheckStructureDF	False
CheckStructureDG	False
CheckStructureHM	False
CheckValuesDF	False
CheckValuesDG	False
CheckValuesHM	False
CheckDerivIter	0
CheckDerivTol	1.0000000000000000e-03
AutoQPRecovery	True
BettsFactor	2.1000000000000000e+00
BettsPoint	1.0000000000000000e+00
IncBettsTau	2.0000000000000000e+00
IncBettsTauMore	3.0000000000000000e+00
StartBettsTau	9.2000000000000000e-04
Infty	1.0000000000000000e+20
IncreaseIWS	1.0000000000000000e+00
IncreaseRWS	1.0000000000000000e+00
ScaleConIter	True
ScaledObj	True
ScaledQP	True
ScaleFacObj	1.0000000000000000e+01
ScaleFacQP	1.0000000000000000e+01
InitialLMest	True
LMestQPipComTol	1.0000000000000000e-03
LMestQPipResTol	1.0000000000000000e-03
FeasibleDual	False
FeasibleInit	False
FeasibleOnly	False
FocusOnFeas	True
FeasibleInitTol	1.0000000000000000e-03
FocusOnFeasFactor	1.0100000000000000e+00
MaxForce	4
Ares	42, 41, 42, 43, 44, 41, 50

Tabelle B.13: Restliche Parameter

Literaturverzeichnis

[1] W. Alt. *Nichtlineare Optimierung*. Vieweg, 2002.

[2] E. Amaldi, M. E. Pfetsch, and L. E. Trotter, Jr. On the maximum feasible subsystem problem, iiss and iis-hypergraphs. *Mathematical Programming*, 95(3):533–554, Mar 2003.

[3] L. Armijo. Minimization of functions having lipschitz continuous first partial derivatives. *Pacific J. Math.*, 16(1):1–3, 1966.

[4] D. P. Bertsekas and J. N. Tsitsiklis. *Parallel and Distributed Computation: Numerical Methods*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1989.

[5] J. T. Betts. *Practical Methods for Optimal Control Using Nonlinear Programming*. Society for Industrial and Applied Mathematics, 2001.

[6] J. T. Betts and W. P. Huffman. Exploiting sparsity in the direct transcription method for optimal control. *Computational Optimization and Applications*, 14(2):179–201, 1999.

[7] H. Boden, R. Gehne, and M. Grauer. *Parallel Nonlinear Optimization on a Multiprocessor System with Distributed Memory*, pages 65–78. Springer Berlin Heidelberg, Berlin, Heidelberg, 1991.

[8] K.-H. Borgwardt. *The Simplex Method, A Probabilistic Analysis*. Springer-Verlag, Berlin, 1987.

[9] T. Bosse and A. Griewank. The relative cost of function and derivative evaluations in the CUTEr test set. In Shaun Forth, Paul Hovland, Eric Phipps, Jean Utke, and Andrea Walther, editors, *Recent Advances in Algorithmic Differentiation*, volume 87 of *Lecture Notes in Computational Science and Engineering*, pages 233–240. Springer, Berlin, 2012.

[10] C. G. Broyden. The convergence of a class of double-rank minimization algorithms. ii. the new algorithm. *J. Inst. Math. Appl.*, 6:222–231, 1970.

[11] C. Büskens. *Optimierungsmethoden und Sensitivitätsanalyse für optimale Steuerprozesse mit Steuer- und Zustands-Beschränkungen*. Dissertation, Universität Münster, 1998.

[12] C. Büskens. *Echtzeitoptimierung und Echtzeitoptimalsteuerung parametergestörter Probleme*. Habilitation, Universität Münster, 2002.

[13] C. Büskens and D. Wassel. The esa nlp solver worhp. In Giorgio Fasano and Janos D. Pinter, editors, *Modeling and Optimization in Space Engineering*, volume 73 of *Springer Optimization and Its Applications*, pages 85–110. Springer New York, 2013.

[14] R. H. Byrd, J. Nocedal, and R. A. Waltz. *Knitro: An Integrated Package for Nonlinear Optimization*, pages 35–59. Springer US, Boston, MA, 2006.

[15] E. Cantú-Paz. *Designing Efficient Master-Slave Parallel Genetic Algorithms*. PhD thesis, Instituto Tecnológoico Autónomo de México, 1999.

[16] R. M. Chamberlain. Some examples of cycling in variable metric methods for constrained minimization. *Mathematical Programming*, 16(1):378–383, 1979.

[17] R. M. Chamberlain, M. J. D. Powell, C. Lemarechal, and H. C. Pedersen. The watchdog technique for forcing convergence in algorithms for constrained optimization. *Mathematical Programming*, 16:1–17, 1982.

[18] J. W. Chinneck. An effective polynomial-time heuristic for the minimum-cardinality iis set-covering problem. *Annals of Mathematics and Artificial Intelligence*, 17(1):127–144, Mar 1996.

[19] J. W. Chinneck. *Feasibility and infeasibility in optimization: algorithms and computational methods, International Series in Operations Research and Management Sciences*, volume 118. Springer, 2008.

[20] T. F. Coleman and A. R. Conn. Nonlinear programming via an exact penalty function: Asymptotic analysis. *Mathematical Programming*, 24(1):123–136, 1982.

[21] T. F. Coleman and A. R. Conn. Nonlinear programming via an exact penalty function: Global analysis. *Mathematical Programming*, 24(1):137–161, 1982.

[22] A. Conn, N. Gould, and P. Toint. *Trust Region Methods*. Society for Industrial and Applied Mathematics, 2000.

[23] F. E. Curtis. A penalty-interior-point algorithm for nonlinear constrained optimization. *Mathematical Programming Computation*, 4(2):181–209, 2012.

[24] P. Deuflhard and A. Hohmann. *Numerische Mathematik: Eine algorithmisch orientierte Einführung*. De Gruyter, Berlin, 2008.

[25] R. Diestel. *Graphentheorie*. Springer-Lehrbuch. Springer, 2006.

[26] E. D. Dolan and J. J. Moré. Benchmarking optimization software with performance profiles. *Mathematical Programming*, 91(2):201–213, 2002.

[27] E. D. Dolan, J. J. Moré, and T. S. Munson. Benchmarking optimization software with cops 3.0. Technical Report ANL/MCS-TM-273, Argonne National Laboratory, February 2004.

[28] A. V. Fiacco. Sensitivity analysis for nonlinear programming using penalty methods. *Mathematical Programming*, 10(1):287–311, 1976.

[29] A. V. Fiacco. Nonlinear programming sensitivity analysis using strong second order assumptions. *Numerical Optimization of Dynamic Systems*, pages 349–362, 1980.

[30] A. V. Fiacco. *Introduction to Sensitivity and Stability Analysis in Nonlinear Programming*, volume 165. Academic Press, 1983.

[31] A. V. Fiacco and A. Ghaemi. Sensitivity analysis of a nonlinear structural design problem. *Computers and Operations Research*, 9(1):29–55, 1982.

[32] R. Fletcher. The calculation of feasible points for linearly constrained optimization problems. *UKAEA Research Group Rept., AERE R6354*, 1970.

[33] R. Fletcher. A new approach to variable metric algorithms. *Comput. J.*, 13(3):317–322, 1970.

[34] R. Fletcher. *Second order corrections for non-differentiable optimization*, pages 85–114. Springer Berlin Heidelberg, Berlin, Heidelberg, 1982.

[35] R. Fletcher. *Practical Methods of Optimization*. Wiley, 1987.

[36] R. Fletcher and S. Leyffer. Nonlinear programming without a penalty function. *Mathematical Programming*, 91(2):239–269, 2002.

[37] R. Fletcher, S. Leyffer, and P. L. Toint. On the global convergence of a filter–sqp algorithm. *SIAM Journal on Optimization*, 13(1):44–59, 2002.

[38] A. Forsgren. Inertia-controlling factorizations for optimization algorithms. *Applied Numerical Mathematics*, 43(1):91 – 107, 2002.

[39] O. Forster. *Analysis 1: Differential- und Integralrechnung einer Veränderlichen*. Analysis / Otto Forster. Vieweg, 2008.

[40] M. R. Garey, D. S. Johnson, and L. Stockmeyer. Some simplified np-complete graph problems. *Theoretical Computer Science*, 1(3):237 – 267, 1976.

[41] S. Geffken. *Effizienzsteigerung nichtlinearer Optimierung mit Hilfe von Algorithmen gemischter Präzision*. Masterarbeit, Universität Bremen, 2013.

[42] S. Geffken and C. Büskens. Feasibility refinement in sequential quadratic programming using parametric sensitivity analysis. *Optimization Methods and Software*, 2016.

[43] S. Geffken and C. Büskens. Worhp multi-core interface, parallelisation approaches for an nlp solver. In *Proceedings of the 6th International Conference on Astrodynamics Tools and Techniques, 14.03. - 17.03.2016, Darmstadt, Germany*, 2016.

[44] C. Geiger and C. Kanzow. *Numerische Verfahren zur Lösung unrestringierter Optimierungsaufgaben.* Springer, 1999.

[45] C. Geiger and C. Kanzow. *Theorie und Numerik restringierter Optimierungsaufgaben.* Springer, 2005.

[46] M. Gerdts. User's guide qp solver. Technical report, Universität der Bundeswehr München, February 2013.

[47] E. M. Gertz and S. J. Wright. Object-oriented software for quadratic programming. *ACM Transactions on Mathematical Software*, 29(1), 2003.

[48] P. E. Gill and W. Murray. Numerically stable methods for quadratic programming. *Mathematical Programming*, 14(1):349–372, 1978.

[49] P. E. Gill, W. Murray, and M. A. Saunders. Snopt: An sqp algorithm for large-scale constrained optimization. *SIAM J. on Optimization*, 12(4):979–1006, April 2002.

[50] P. E. Gill, W. Murray, and M. H. Wright. *Practical Optimization.* academic Press, 1981.

[51] P. E. Gill, M. A. Saunders, and E. Wong. *On the Performance of SQP Methods for Nonlinear Optimization*, pages 95–123. Springer International Publishing, Cham, 2015.

[52] D. Goldfarb. A family of variable-metric methods derived by variational means. *Math. Comp*, 24:23–26, 1970.

[53] D. Goldfarb. Extension of newton's method and simplex methods for solving quadratic programs. *Numerical Methods for Nonlinear Optimization*, pages 239–254, 1972.

[54] J. Gondzio. Interior point methods 25 years later. Technical report, University of Edinburgh, 2011.

[55] J. Gondzio and A. Grothey. A new unblocking technique to warmstart interior point methods based on sensitivity analysis. *SIAM Journal on Optimization*, 19(3):1184–1210, 2008.

[56] N. Gould. On practical conditions for the existence and uniqueness of solutions to the general equality quadratic programming problem. *Mathematical Programming*, 32(1):90–99, 1985.

[57] N. Gould, D. Orban, and P. Toint. Cuter and sifdec: A constrained and unconstrained testing environment, revisited. *ACM Transactions on Mathematical Software*, 29(4):–, January 2003.

[58] N. Gould, D. Orban, and P. Toint. Numerical methods for large-scale nonlinear optimization. *Acta Numerica*, 14:299–361, 2005.

[59] N. Gould, D. Orban, and P. Toint. Cutest: A constrained and unconstrained testing environment with safe threads for mathematical optimization. *Computational Optimization and Applications*, 60(3):545–557, April 2015.

[60] N. Gould and J. Scott. A note on performance profiles for benchmarking software. *ACM Trans. Math. Softw.*, 43(2):15:1–15:5, August 2016.

[61] S. P. Han. A globally convergent method for nonlinear programming. *Journal of Optimization Theory and Applications*, 22(3):297–309, 1977.

[62] S. P. Han and O. L. Mangasarian. Exact penalty functions in nonlinear programming. *Mathematical Programming*, 48:161–220, 1979.

[63] N. J. Higham and S. Hun Cheng. Modifying the inertia of matrices arising in optimization. *Linear Algebra and its Applications*, 275-276:261 – 279, 1998. Proceedings of the Sixth Conference of the International Linear Algebra Society.

[64] W. Hock and K. Schittkowski. *Test Examples for Nonlinear Programming Codes*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1981.

[65] J. Hogg and J. Scott. New parallel sparse direct solvers for multicore architectures. *Algorithms*, 6(4):702–725, 2013.

[66] HSL. A collection of fortran codes for large scale scientific computation. `http://www.hsl.rl.ac.uk/`, 2015 (accessed January 30, 2015).

[67] P. Kalmbach. *Effiziente Ableitungsbestimmung bei hochdimensionaler nichtlinearer Optimierung*. Dissertation, Universität Bremen, 2011.

[68] A. E. Kemper. *Filtermethoden zur Bewertung der Suchrichtung in NLP-Verfahren*. Diplomarbeit, Universität Bremen, 2010.

[69] V. L. Klee and G. J. Minty. How good is the simplex method? *Inequalities III*, 4:159–175, 1972.

[70] P. Kosmol. *Methoden zur numerischen Behandlung nichtlinearer Gleichungen und Optimierungsaufgaben*. Springer Fachmedien Wiesbaden, 1989.

[71] K. Levenberg. A method for the solution of certain non-linear problems in least squares. *Quarterly Journal of Applied Mathmatics*, II(2):164–168, 1944.

[72] X. S. Li, J. W. Demmel, J. R. Gilbert, L. Grigori, M. Shao, and I. Yamazaki. Superlu users' guide. Technical Report LBNL-44289, Lawrence Berkeley National Laboratory, September 1999. `http://crd.lbl.gov/~xiaoye/SuperLU/`. Last update: August 2011.

[73] F. A. Lootsma and K. M. Ragsdell. State-of-the-art in parallel nonlinear optimization. *Parallel Computing*, 6(2):133 – 155, 1988.

[74] D. G. Luenberger and Y. Ye. *Linear and Nonlinear Programming*. Springer, 2008.

[75] O. L. Mangasarian and S. Fromovitz. The fritz john necessary optimality conditions in the presence of equality and inequality constraints. *Journal of Mathematical Analysis and Applications*, 17(1):37 – 47, 1967.

[76] N. Maratos. *Exact Penalty Function Algorithms for Finite Dimensional and Control Optimization Problems*. PhD thesis, University of London, 1978.

[77] D. W. Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *Journal of the Society for Industrial and Applied Mathematics*, 11(2):431–441, 1963.

[78] D. T. Marr, F. Binns, D. L. Hill, G. Hinton, D. A. Koufaty, J. A. Miller, and M. Upton. Hyper-threading technology architecture and microarchitecture. *Intel Technology Journal*, 6(1):4–15, 2002.

[79] S. Mehrotra. On the implementation of a primal-dual interior point method. *Siam Journal Optimization*, 2:575–601, 1992.

[80] A. Migdalas, G. Toraldo, and V. Kumar. Nonlinear optimization and parallel computing. *Parallel Computing*, 29(4):375 – 391, 2003. Parallel computing in numerical optimization.

[81] T. Nikolayzik. *Korrekturverfahren zur numerischen Lösung nichtlinearer Optimierungsprobleme mittels Methoden der parametrischen Sensitivitätsanalyse*. Dissertation, Universität Bremen, 2011.

[82] J. Nocedal and S.J. Wright. *Numerical Optimization*. Springer, 2006.

[83] M. E. Pfetsch. Branch-and-cut for the maximum feasible subsystem problem. *SIAM Journal on Optimization*, 19(1):21–38, 2008.

[84] R. Plato. *Numerische Mathematik kompakt*. Vieweg+Teubner Verlag, 2010.

[85] M. J. D. Powell. Algorithms for nonlinear constraints that use lagrangian functions. *Mathematical Programming*, 14(1):224–248, 1978.

[86] M. J. D. Powell. *Numerical Analysis: Proceedings of the Biennial Conference Held at Dundee, June 28–July 1, 1977*, chapter A fast algorithm for nonlinearly constrained optimization calculations, pages 144–157. Springer Berlin Heidelberg, Berlin, Heidelberg, 1978.

[87] S. Leyffer R. Fletcher. User manual for filtersqp. numerical analysis report na/181. Technical report, University of Dundee, 1998.

[88] S. M. Robinson. Perturbed kuhn-tucker points and rates of convergence for a class of nonlinear-programming algorithms. *Mathematical Programming*, 7:1–16, 1974.

[89] R. W. H. Sargent. Reduced gradient and projection methods for nonlinear programming. *Numerical Methods for Constrained Optimization*, 1974.

[90] R. Schäfer. *Parametrische Sensitivitätsanalyse*. Bachelorarbeit, Universität Bremen, 2012.

[91] K. Schittkowski. The nonlinear programming method of wilson, han, and powell with an augmented lagrangian type line search function. *Numerische Mathematik*, 38(1):83–114, 1982.

[92] K. Schittkowski. *More Test Examples for Nonlinear Programming Codes*. Springer-Verlag Berlin Heidelberg, 1987.

[93] K. Schittkowski. Nlpqlp: A new fortran implementation of a sequential quadratic programming algorithm for parallel computing. Technical report, Universität Bayreuth, 2001.

[94] R. Schnabel. Parallel nonlinear optimization: Limitations, opportunities, and challenges. Computer Science Technical Reports CU-CS-715-94, University of Colorado, Boulder, 1994.

[95] D. F. Shanno. Conditioning of quasi-newton methods for function minimization. *Math. Comp*, 26:647–656, 1970.

[96] O. Söderström. *Testing and Tuning of Optimization Algorithms*. Examensarbete, Uppsala Universitet, 2015.

[97] P. Spellucci. *Numerische Verfahren der nichtlinearen Optimierung*. Birkhäuser Verlag, 1993.

[98] Steinbeis Forschungszentrum Optimierung, Steuerung und Regelung. Users' guide to worhp 1.9. Technical report, April 2017.

[99] H. Sutter. The free lunch is over: A fundamental turn toward concurrency in software. *Dr. Dobb's Journal*, 30(3):202–210, 2005.

[100] E. Talbi, S. Mostaghim, T. Okabe, H. Ishibuchi, G. Rudolph, and C. A. Coello Coello. *Parallel Approaches for Multiobjective Optimization*, pages 349–372. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.

[101] M. Ulbrich, S. Ulbrich, and L. N. Vicente. A globally convergent primal-dual interior-point filter method for nonlinear programming. *Mathematical Programming*, 100(2):379–410, 2004.

[102] S. Ulbrich. On the superlinear local convergence of a filter-sqp method. *Mathematical Programming*, 100(1):217–245, 2004.

[103] D. A. Van Veldhuizen, J. B. Zydallis, and G. B. Lamont. Considerations in engineering parallel multiobjective evolutionary algorithms. *Trans. Evol. Comp*, 7(2):144–173, April 2003.

[104] R. J. Vanderbei and D. F. Shanno. An interior-point algorithm for nonconvex nonlinear programming. *Computational Optimization and Applications*, 13(1):231–252, 1999.

[105] A. Wächter and L. T. Biegler. Line search filter methods for nonlinear programming: Local convergence. *SIAM Journal on Optimization*, 16(1):32–48, 2005.

[106] A. Wächter and L. T. Biegler. Line search filter methods for nonlinear programming: Motivation and global convergence. *SIAM Journal on Optimization*, 16(1):1–31, 2005.

[107] A. Wächter and L. T. Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106(1):25–57, 2006.

[108] W. Wan and L. T. Biegler. Structured regularization for barrier nlp solvers. *Computational Optimization and Applications*, 66(3):401–424, Apr 2017.

[109] D. Wassel. *Exploring Novel Designs of NLP Solvers*. Dissertation, Universität Bremen, 2013.

[110] R. Wilson. *A Simplicial Algorithm for Concave Programming*. PhD thesis, Harvard Business School, Boston, 1963.

[111] P. Wolfe. Convergence conditions for ascent methods. *SIAM Review*, 11(2):226–235, 1969.

[112] P. Wolfe. Convergence conditions for ascent methods. ii: Some corrections. *SIAM Review*, 13(2):185–188, 1971.

[113] D. H. Wolpert and W. G. Macready. No free lunch theorems for optimization. *IEEE Transactions on evolutionary computation*, 1, 1997.

[114] Y. Xu and Y. Chen. A framework for parallel nonlinear optimization by partitioning localized constraints. In *Proc. International Symposium on Parallel Architectures, Algorithms and Programming*, 2008.