Universität Bremen

Effiziente Ableitungsbestimmung bei hochdimensionaler nichtlinearer Optimierung

---

Dissertation zur Erlangung des Doktorgrades
Dr. rer. nat.
der Fakultät für Mathematik an der Universität Bremen
vorgelegt von Patrik Kalmbach

---

1. Gutachter: Prof. Dr. rer. nat. C. Büskens
2. Gutachter: Prof. Dr. rer. nat. M. Gerdts

23. Juni 2011

Patrik Kalmbach
Matrikel-Nummer 1694567

Ich versichere, dass ich diese Dissertation selbstständig verfasst und keine anderen als die angegebenen Hilfsmittel verwendet habe.

Bremen, den 23. Juni 2011

Danksagungen

Mein Dank gilt in erster Linie meinen Eltern, die das exakte Denken in mir von Kindesbeinen an und auf spielerische Weise förderten und mich gleichzeitig zur Freigeistigkeit erzogen. Ich bin überzeugt davon, dass es in erster Linie diese Erziehung war, die meinen Weg zum promovierten Mathematiker ermöglichte.
Ich danke außerdem meiner Freundin, Jana Jirásková, für all die Kraft, die sie mir in den Jahren gegeben hat und für all die Geduld und das Verständnis, was sie mir vor allem in der Schlussphase der Promotion entgegen gebracht hat.
Größte Dankbarkeit empfinde ich gegenüber meinem Doktorvater, Prof. Dr. Christof Büskens, der früher als ich wusste, dass ich in der Lage sein würde zu promovieren. Ihn bewundere ich insbesondere für seine Fähigkeiten, ohne lange Einarbeitung komplizierte Sachverhalte gerade weit genug zu umreißen, um grobe aber wertvollste Lösungsrichtungen zu skizzieren. Seiner Förderung verdanke ich nicht nur das vorliegende Ergebnis meiner Arbeit, sondern, wie ich glaube, auch die Fortentwicklung meiner Persönlichkeit in den vergangenen Jahren.
Ein ausdrücklicher Dank gilt auch meinem Kollegen, Jan Vogelsang, der stets ein offenes Ohr hatte, wenn ich glaubte, einen brauchbaren Einfall zu haben und der nicht aufhörte, kluge und wertvolle Fragen zu stellen bis entweder die Schwächen meines Einfalls offenkundig wurden oder er selbst davon überzeugt war.
Großer Dank gebührt auch meinen Kollegen, Dr. Matthias Knauer und Tim Nikolyzik, für das Korrekturlesen der Arbeit.
Ich bedanke mich des Weiteren bei meinen Kollegen Jan Tietjen, Dennis Wassel und Bodo Blume, die mir durch Anregungen oder technische Unterstützung geholfen haben, diese Arbeit in der vorliegenden Form zu verwirklichen.

Inhaltsverzeichnis

1 Einleitung **1**

2 Grundlagen der Nichtlinearen Optimierung **5**
2.1 Einleitung . . . 5
2.2 Problemstellung und Optimalitätskriterien . . . 6
2.3 Newton-Verfahren . . . 9
2.4 SQP-Verfahren . . . 11
2.5 WORHP . . . 17

3 Finite Differenzen **19**
3.1 Einleitung . . . 19
3.2 Graph Coloring . . . 22
3.2.1 NP-Vollständigkeit . . . 22
3.2.2 Definition und Eigenschaften des Graph Coloring Problems . . . 24
3.3 Graphentheoretischer Ansatz zur Bestimmung erster Ableitungen . . . 27
3.4 Heuristische Algorithmen zur Lösung des Graph Coloring Problems . . . 31
3.4.1 Der CPR-Algorithmus . . . 31
3.4.2 Der LFO-Algorithmus . . . 33
3.4.3 Der SLO-Algorithmus . . . 33
3.4.4 Die Austauschmethode . . . 35
3.5 Graphentheoretischer Ansatz zur Bestimmung zweiter Ableitungen . . . 37
3.5.1 Hessegruppen . . . 37
3.5.2 Paargruppen . . . 47
3.6 Fallbeispiel: Planarer Niedrigschubtransfer zu einem geosynchronen Orbit . . . 54
3.7 Komplexes Differenzieren . . . 61

4 Update-Techniken **63**
4.1 Einleitung . . . 63
4.2 Klassisches Rang-2-BFGS-Verfahren . . . 65
4.2.1 Methodik . . . 65
4.2.2 Konvergenzeigenschaften des BFGS-Verfahrens . . . 70
4.3 Existierende dünnbesetzte BFGS-Verfahren . . . 83
4.4 Partitioniertes BFGS-Verfahren . . . 88
4.4.1 Methodik . . . 88
4.4.2 Konvergenzeigenschaften des partitionierten BFGS-Verfahrens . . . 92
4.5 Block-BFGS . . . 97
4.5.1 Blockmatrizen ohne Überschneidungen . . . 97
4.5.2 Einfache Überschneidungen . . . 97

4.5.2.1 Methodik . . . 97
4.5.2.2 Konvergenzeigenschaften des Block-BFGS-Verfahrens mit einfachen Überschneidungen . . . 110
4.5.3 Mehrfache Überschneidungen . . . 118
4.5.3.1 Methodik . . . 118
4.5.3.2 Konvergenzeigenschaften des Block-BFGS-Verfahrens mit mehrfachen Überschneidungen . . . 125
4.5.4 Variablen-Permutation . . . 125
4.6 Verallgemeinertes Sparse BFGS . . . 128
4.6.1 Methodik . . . 128
4.6.2 Konvergenzeigenschaften des Verallgemeinerten Sparse BFGS . . . 134

5 Numerische Ergebnisse **135**
5.1 CUTEr . . . 135
5.2 Auswertung . . . 136
5.2.1 Erste Ableitungen . . . 136
5.2.2 Zweite Ableitungen . . . 139

6 Fazit **145**

Tabellenverzeichnis

3.1 Verschiedene Zeitkomplexitätsfunktionen. 23

5.1 Ergebnis mit analytischen Routinen. 136
5.2 Restringierte Testprobleme mit Jacobimatrix mittels Finiter Differenzen. . . 137
5.3 Restringierte Vergleichsprobleme mit Jacobimatrix mittels Finiter Differenzen. 138
5.4 Restringierte Vergleichsprobleme mit Jacobimatrix mittels Finiter Differenzen. 140
5.5 Restringierte Vergleichsprobleme mit analytischer Jacobimatrix, Teil 1. . . . 142
5.6 Restringierte Vergleichsprobleme mit analytischer Jacobimatrix, Teil 2. . . . 143
5.7 Alle BFGS-Testprobleme mit analytischer Jacobimatrix, Teil 1. 144
5.8 Alle BFGS-Testprobleme mit analytischer Jacobimatrix, Teil 2. 144

Abbildungsverzeichnis

2.1 Quadratische Approximationen im SQP-Verfahren 11
2.2 Grober Aufbau eines SQP-Verfahren 12
2.3 Schrittweiten, die der Armijoregel genügen. 14
2.4 Bereiche (schwarz), die der Wolfe-Powell-Regel und Bereiche (grün), die der Armijoregel genügen. 15
2.5 Bereiche (schwarz), die der strengen Wolfe-Powell-Regel genügen und Bereiche (grün), die der Wolfe-Powell-Regel genügen. 16

3.1 Bekanntes Graph Coloring Problem 24
3.2 Zweiteiliger Graph 25
3.3 Zu colorierender Graph 25
3.4 Colorierter Graph 25
3.5 Zweiteiliger Graph 31
3.6 Zweiteiliger Graph 2 35

1 Einleitung

"Wer das erste Knopfloch verfehlt, kommt mit dem Zuknöpfen nicht zu Rande."
Johann Wolfgang von Goethe, 1749-1832

Diese Einleitung ist zum Teil an [9] angelehnt.
Nichtlineare Optimierung hat sich in zahlreichen Bereichen zu einer Wettbewerbsvorteile erbringenden Schlüsseltechnologie entwickelt. Ohne Anspruch auf Vollständigkeit sind hierbei

- naturwissenschaftliche Anwendungen
- wirtschaftswissenschaftliche Anwendungen
- medizinische Anwendungen und
- ingenieurwissenschaftliche Anwendungen

zu nennen. Sie verwendet effiziente mathematische Verfahren und nutzt Fortschritte in der Computertechnologie aus, um etwa Fragestellungen nach der bestmöglichen Festlegung technischer Vorgänge zu beantworten. Der Erfolg eines Optimierungsprojekts hängt in der Regel von einer Vielzahl von Faktoren ab, die von der Modellierungsphase durch die Wahl des Modellierungsansatzes bis zur finalen Interpretation und Umsetzung der Ergebnisse reicht. Hierbei muss den mathematisch numerischen Lösungsverfahren als zentrales Bindeglied und als entscheidendes Werkzeug der nichtlinearen Optimierung in besonderer Weise Rechnung getragen werden.
Zwei wesentliche Klassen von Lösern werden durch die Innere-Punkte-Methoden und die SQP (Sequential Quadratic Programming) Verfahren gebildet. Geiger und Kanzow [34], [35] bieten einen guten und recht aktuellen Überblick über gebräuchliche Ansätze. Des Weiteren sei auf die Arbeiten von Schittkowski, [69], [70], [71], [72], [73], [74], Gill, Murray und Wright [43], Gill, Murray, Saunders und Wright [42] und Steinbach [79] hingewiesen. In Kapitel 2 führen wir die wichtigsten Grundlagen der Nichtlinearen Optimierung ein und beschreiben die SQP-Verfahren.
Die traditionellen Verfahren sind allesamt gekennzeichnet durch die Eigenschaft, dass sie für dicht besetzte und kleine bis mittlere Probleme entwickelt wurden. Der Ausdruck *dicht besetzt* weist in diesem Zusammenhang darauf hin, dass alle auftretenden Matrizen, etwa die Hesse-Matrix der Lagrange-Funktion oder die Jacobimatrix der Nebenbedingungen, die in allen gängigen Verfahren eine entscheidende Rolle spielen, nur wenige Nichtnulleinträge besitzen.
Unter kleinen bis mittleren Optimierungsproblemen versteht man Probleme in einer Dimension von bis zu 1000 (gelegentlich auch 5000) Optimierungsvariablen und Beschränkungen. Hierdurch lassen sich bereits eine Vielzahl von Problemen lösen, jedoch stoßen die benannten Verfahren in mehrerlei Hinsicht an ihre Grenzen. Die Stabilität der Verfahren, die Anzahl der Freiheitsgrade und damit die Genauigkeit der Lösung und insbesondere die erforderlichen Rechenzeiten liegen in Dimensionen, die aus Anwendersicht nicht mehr tolerierbar

sind. Tatsächlich möchte man heute jedoch mehr: 1.000.000 Optimierungsvariablen und Beschränkungen (dies sind dann große Probleme) sind keine Seltenheit mehr.
Gesucht sind dann Verfahren, die ohne Verlust an Stabilität und ohne überproportional steigenden Rechen- und Speicherbedarf solche Probleme lösen können. An dieser Stelle kommt nun ein Umstand zum Tragen, den es gewinnbringend zu nutzen gilt. In der Regel und insbesondere für Probleme der optimalen Steuerung, sind die auftretenden Matrizen (Hesse- oder Jacobimatrix) dünnbesetzt ("sparse"), das heißt die Matrizen besitzen eine große Anzahl von Nulleinträgen; je nach Problem und Dimension sind oftmals weit weniger als 1 Prozent der Einträge von Null verschieden. Die Verfahren, die diesen Umstand auf intelligente Weise berücksichtigen, kommen demnach einerseits mit weniger als 1 Prozent des Speicherbedarfs aus, welches ein Verfahren benötigen würde, das ein dicht besetztes Problem erwartet. Andererseits können zahlreiche Null-Rechenoperationen (Addition, Multiplikation) umgangen werden, da das Ergebnis aufgrund der gegebenen Struktur vorausgesagt werden kann.
Die Bestimmung dieser Matrizen sowie des Gradienten der Zielfunktion ist häufig einer der Flaschenhälse innerhalb der gängigen Verfahren. Das bedeutet auf der anderen Seite, dass die Effizienz der Verfahren erheblich gesteigert werden kann, wenn diese Ableitungsinformationen auf effiziente Weise bereitgestellt werden. Davon handelt die vorliegende Arbeit.
Eine Möglichkeit zur Ableitungsbestimmung besteht darin, dass diese Funktionen als Routine gegeben sind und in gleicher Weise wie die zugrunde liegenden Zielfunktion und Nebenbedingungsfunktionen ausgewertet werden können. Im Allgemeinen ist dies jedoch nicht der Fall, weswegen ein Löser von allgemeinen nichtlinearen Optimierungsproblemen nicht von dieser Möglichkeit ausgehen sollte.
Eine weitere Möglichkeit bietet die so genannte Automatische Differentiation, siehe Griewank beziehungsweise Griewank et al. [46], [47] und [49] und Walther beziehungsweise Walther et al. [32], [33], [81] und [84]. Diese Möglichkeit ist jedoch ebenfalls nicht immer durchführbar, denn sie setzt voraus, dass die Zielfunktion und Nebenbedingungsfunktionen als Routinen vorliegen. Es kann jedoch sein, dass diese Funktionen nur in Tabellenform oder Ähnlichem verfügbar sind und die Automatische Differentiation daher nicht verwendet werden kann.
Eine der gängigsten Methoden ist die der Finiten Differenzen, welche in Kapitel 3 behandelt werden. Diese Methode ist immer anwendbar (sofern die Funktionen differenzierbar sind) und daher tauglich für einen allgemeinen Löser für nichtlineare Optimierungsprobleme. Abgesehen davon, dass eine numerische Berechnung stets Rundungs- und Darstellungsfehlern unterworfen ist und somit niemals ganz exakt ist, sind Approximationen mittels Finiter Differenzen exakter als viele andere Verfahren. Der Nachteil der Finiten Differenzen ist jedoch, dass sie - zumindest in ihrer klassischen Form - sehr vieler Funktionsauswertungen bedürfen und daher verhältnismäßig langsam sind. In Kapitel 3 werden jedoch zahlreiche Methoden entwickelt, wie dieses Verfahren, durch intelligente Ausnutzung der Dünnbesetztheit sowie Zuhilfenahme der Graphentheorie, erheblich beschleunigt werden kann.
Trotz dieser Beschleunigungsstrategien zeigt es sich, dass es nicht immer sinnvoll ist, die Hessematrix (theoretisch) exakt zu berechnen. Dies gilt sowohl im Falle vorgegebener Routinen zur Auswertung der Einträge der Matrix als auch bei einer Berechnung mittels finiter Differenzen. Die Ergebnisse solcher Berechnungen sind zweifelsohne sehr genau und führen zu guten Konvergenzeigenschaften. Es existieren Alternativen, welche zwar in der Regel mehr Iterationen zur Lösung des Problems benötigen, insgesamt aber schneller sind, da die einzelnen Iterationen deutlich weniger Zeit bedürfen. Zu diesen zählen die so genannten Broyden, Fletcher Goldfarb Shanno (BFGS)-Verfahren.

Während die Verwendung der exakten Hessematrix im Regelfall zu lokal quadratischer Konvergenz führt, liefert eine geeignete Approximation der Hessematrix immernoch superlineare Konvergenz. Insbesondere bei kleinen, dichtbesetzten (das heißt nicht-dünnbesetzten) Problemen, haben sich die BFGS-Verfahren längst als Standard etabliert. Bei großen, dünnbesetzten Problemen bewähren sich die klassischen BFGS-Verfahren jedoch nur sehr bedingt. Die Approximation der Hessematrix ist nämlich selbst dann dichtbesetzt, wenn die tatsächliche Hessematrix sehr dünnbesetzt ist. Aufgrund der dichten Struktur, entsteht ein erhöhter Speicheraufwand bei der Speicherung der Hessematrix im Vergleich zur ursprünglichen Problemformulierung. Des Weiteren werden die weiterführenden Rechnungen (beispielsweise die linearen Gleichungssysteme), die innerhalb des Optimierungsprozesses häufig durchgeführt werden, somit unnötig und drastisch verkompliziert, wodurch die Rechenzeit erheblich erhöht wird und der ursprüngliche Vorteil von BFGS-Verfahren, nämlich die schnelle Berechnung der Approximation, konterkariert wird.
BFGS-ähnliche Verfahren, welche die Dünnbesetztheit berücksichtigen und erhalten (Sparse BFGS) sind daher Gegenstand aktueller Forschung, siehe beispielsweise Nocedal [64], Liu und Nocedal [56], Buckley und LeNir [6], June und Hassan [54], Perry [66], Shanno [77], [78], Buckley [6], [7], Nazareth [62], Gill und Leonard [40], Fletcher [28], [29], Herskovits [51], Toint [82], [83] und Griewank und Toint [48].
Bis auf Griewank und Toint liefert keiner dieser Ansätze ein Konvergenzresultat. Das Resultat von Griewank und Toint ist leider für allgemeine Ansätze untauglich, da hierbei von sehr vereinfachenden Bedingungen ausgegangen wird. In Kapitel 4 entwickeln wir, zum Teil auf Griewank und Toint aufbauend, mehrere allgemein anwendbare Sparse BFGS-Verfahren und beweisen deren superlineare Konvergenzeigenschaften.
In Kapitel 5 zeigen wir die Effizienz der erarbeiteten Verfahren an einer Vielzahl von Beispielproblemen und schließen mit einem Fazit in Kapitel 6.

2 Grundlagen der Nichtlinearen Optimierung

"In der Welt geschieht nichts, worin man nicht den Sinn eines bestimmten Maximums oder Minimums erkennen könnte."
Leonhard Euler, 1707-1783

Dieses Kapitel ist in Teilen an [10] angelehnt.

2.1 Einleitung

Die Lösung von Optimierungsproblemen ist eine der häufigsten Anwendung der Mathematik in Wirtschaftswissenschaften (Operations Research), Technik und Naturwissenschaften. Nocedal und Wright [65] schreiben dazu:
"**People optimize:** *Airline companies schedule crews and aircraft to minimize cost. Investors seek to create portfolios that avoid risks while achieving a high rate of return. Manufacturers aim for maximizing efficiency in the design and operation of their production processes.*
Nature optimizes: *Physical systems tend to a state of minimum energy. The molecules in an isolated chemical system react with each other until the total potential energy of their electrons is minimized, Rays of light follow paths that minimize their travel time."*
Vor der Optimierung steht in der Regel die Modellierung. Nachdem ein mathematisches Modell für die jeweilige Anwendung aufgestellt ist, werden mathematische Optimierungstechniken verwendet, um gewisse unbekannte Modellparameter oder -funktionen so zu bestimmen, dass eine *Zielfunktion* unter vorgegebenen *Nebenbedingungen* minimiert (oder maximiert) wird. In Nocedal und Wright [65] findet sich neben dem genannten Zitat eine ausführliche Theorie über numerische Optimierung. Des Weiteren sei auf Geiger und Kanzow verwiesen [34], [35].
In diesem Kapitel werden die wichtigsten theoretischen Resultate aus der nichtlinearen Optimierung behandelt.

2.2 Problemstellung und Optimalitätskriterien

Problem 2.2.1. *Das Standardproblem der nichtlinearen Optimierung (NLP) lautet*

$$\min_x f(x) \tag{2.1}$$
$$\text{unter } g_i(x) = 0, \quad i \in I_G \tag{2.2}$$
$$g_i(x) \leq 0, \quad i \in \{1, 2, ..., m\} \setminus I_G. \tag{2.3}$$

Hierbei bezeichnet $f : \mathbb{R}^n \to \mathbb{R}$ die Zielfunktion, $x \in \mathbb{R}^n$ den Vektor der Optimierungsvariablen und $g : \mathbb{R}^n \to \mathbb{R}^m$ die Nebenbedingungen. Die Indexmenge I_G kennzeichnet die Gleichungsnebenbedingungen.

Bemerkung 2.2.2. Nebenbedingungen der Form

$$g_i(x) = x_j - a_i, \quad i \in \{1, ..., m\},\ j \in \{1, ..., n\}$$

mit $a_i \in \mathbb{R}$ werden *Box-Beschränkungen* genannt.

Definition 2.2.3. $x \in \mathbb{R}^n$ heißt *zulässig*, falls

$$\begin{aligned} g_i(x) &= 0, \quad i \in I_G \\ g_i(x) &\leq 0, \quad i \in \{1, 2, ..., m\} \setminus I_G \end{aligned}$$

gilt. Die Menge

$$\begin{aligned} A(x) &:= \{i \in \{1, 2, ..., m\} : \quad g_i(x) = 0\} \text{ beziehungsweise} \\ \tilde{A}(x) &:= \{i \in \{1, 2, ..., m\} \setminus I_G : \quad g_i(x) = 0\} \end{aligned}$$

heißt *Menge der aktiven Indizes* und eine Nebenbedingung $g_i,\ i \in A(x)$ heißt *aktiv*.

Die Unterscheidung zwischen aktiven und inaktiven (das heißt nicht aktiven) Nebenbedingungen ist insofern wichtig, da (Stetigkeit vorausgesetzt) inaktive Nebenbedingungen zumindest in einer kleinen Umgebung von x auch erfüllt sind und daher lokal keine Beschränkungen darstellen. Es werden im Folgenden Optimimalitätskriterien eingeführt. Hierfür benötigen wir zunächst einige Begriffe.

Definition 2.2.4. Nach Mangasarian [58] nennt man einen bezüglich Problem 2.2.1 mit stetig differenzierbaren $g : \mathbb{R} \to \mathbb{R}^m$ zulässigen Punkt $x \in \mathbb{R}^n$ *regulär*, falls

$$\nabla g_i(x), \quad i \in I_G$$

linear unabhängig sind und es ein $v \in \mathbb{R}^n \setminus \{0\}$ gibt, so dass

$$\nabla g_i(x)^T v < \quad 0, \quad i \in \tilde{A}(x) \tag{2.4}$$
$$\nabla g_i(x)^T v = \quad 0, \quad i \in I_G. \tag{2.5}$$

Gilt die stärkere Bedingung, dass

$$\nabla g_i(x), \quad i \in A(x)$$

linear unabhängig sind, so heißt x *streng regulär* oder *normal*.

Bemerkung 2.2.5. Ist x streng regulär, gilt (2.4)-(2.5) trivialerweise.

Eine wichtige Rolle spielt die so genannte Lagrange-Funktion.

Definition 2.2.6. Es sei $\lambda \in \mathbb{R}^m$. Die *Lagrange-Funktion* $L : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$ zum Problem 2.2.1 ist definiert durch

$$L(x, \lambda) := f(x) + \lambda^T g(x).$$

Die Multiplikatoren $\lambda_i,\ i = 1, ..., m$ werden *Lagrange-Multiplikatoren* genannt.

Hiermit kann man das von Karush, Kuhn und Tucker unabhängig voneinander formulierte notwendige Optimalitätskriterium darstellen. Allgemein bezeichnet man es als das KKT-Kriterium.

Satz 2.2.7. *Es sei $\hat{x}$ ein lokales Minimum von Problem 2.2.1, f und g stetig differenzierbar in einer Umgebung von $\hat{x}$. Ist $\hat{x} \in \mathbb{R}^n$ regulär, so existiert ein $\lambda \in \mathbb{R}^m$ so dass*

$$\begin{aligned} \nabla_x L(\hat{x}, \lambda) &= \nabla f(\hat{x}) + \lambda^T \nabla g(\hat{x}) = 0 && (2.6)\\ \lambda_i &\geq 0, \quad i \in \tilde{A}(\hat{x}), && (2.7)\\ \lambda_i &= 0, \quad i \notin A(\hat{x}). && (2.8) \end{aligned}$$

Ist $\hat{x}$ sogar normal, ist λ eindeutig.

Beweis. siehe [27] □

Da Satz 2.2.7 nur eine notwendige Bedingung beschreibt, bezeichnet man einen Punkt, der (2.6)-(2.8) erfüllt als *kritischen Punkt* oder *KKT-Punkt*. Es sei angemerkt, dass Punkte, die die Zielfunktion unter Einhaltung der Nebenbedingungen maximieren (statt minimieren) ebenfalls KKT-Punkte sind. Viele Optimierungsverfahren konzentrieren sich lediglich auf die Bestimmung von KKT-Punkten. Unter der Annahme, dass es entweder keine lokalen Maxima gibt oder die Startschätzungen nahe genug an dem zu findenden lokalen Minimum liegen, ist dies auch ausreichend. Anderenfalls gibt es so genannte *Globalisierungsstrategien*, die dieses Problem umgehen. Hinreichende Nebenbedingungen sind nicht immer erfüllt und häufig schwer nachzuweisen. Dennoch sollen sie hier in aller Kürze dargelegt werden. Hierzu ist folgender Kegel von Bedeutung:

$$C := \left\{ v \in \mathbb{R}^n : \begin{cases} \nabla g_i(x)^T v \leq 0, & i \in \tilde{A}(x), \quad \text{falls } \lambda_i = 0, \\ \nabla g_i(x)^T v = 0, & i \in \tilde{A}(x), \quad \text{falls } \lambda_i > 0, \\ \nabla g_i(x)^T v = 0, & i \in I_G \end{cases} \right\}$$

Hiermit lassen sich notwendige und hinreichende Bedingungen zweiter Ordnung definieren. Der Begriff „zweiter Ordnung“ bedeutet hierbei, dass eine zweite Ableitung mit einfließt.

Satz 2.2.8. *Seien $x \in \mathbb{R}^n$ ein zulässiger normaler Punkt und alle Funktionen hinreichend*

oft differenzierbar. Dann gilt

i. ***Notwendige Bedingung zweiter Ordnung:***
Sei x eine lokale Minimalstelle des Problems 2.2.1, so existiert ein eindeutiger Vektor $\lambda \in \mathbb{R}^m$ mit

$$\begin{aligned} \lambda_i &\geq 0, \quad i \in \tilde{A}(x), && (2.9)\\ \lambda_i &= 0, \quad i \notin A(x) && (2.10)\\ \nabla_x L(x,\lambda) &= 0, && (2.11)\\ v^T \nabla_x^2 L(x,\lambda) v &\geq 0, \quad v \in C \setminus \{0\}. && (2.12) \end{aligned}$$

ii. ***Hinreichende Bedingung zweiter Ordnung:***
Existiert ein Vektor $\lambda \in \mathbb{R}^m$ mit

$$\begin{aligned} \lambda_i &\geq 0, \quad i \in \tilde{A}(x), && (2.13)\\ \lambda_i &= 0, \quad i \notin A(x) && (2.14)\\ \nabla_x L(x,\lambda) &= 0, && (2.15)\\ v^T \nabla_x^2 L(x,\lambda) v &> 0, \quad v \in C \setminus \{0\}. && (2.16) \end{aligned}$$

so ist x ein strenges lokales Minimum.

iii. ***Strenge hinreichende Bedingung zweiter Ordnung:***
Existiert zu einem Punkt $x \in \mathbb{R}^n$ ein Vektor $\lambda \in \mathbb{R}^m$, so dass (2.13)-(2.16) gilt mit

$$\lambda_i > 0, \quad i \in \tilde{A}(x),$$

so gilt neben der strengen Minimalitätseigenschaft von x zusätzlich, dass sich C vereinfacht zu

$$C = Kern(\nabla_x g_i(x), \quad i \in A(x)). \tag{2.17}$$

Beweis. siehe Fletcher [27]. □

Wie man leicht sieht, spielen die inaktiven Nebenbedingungen aufgrund der Multiplikation mit $\lambda_i = 0$ bei den Optimalitätskriterien keine Rolle. Der Vektor der aktiven Nebenbedingungen erfährt daher eine besondere Bedeutung.

2.3 Newton-Verfahren

Sei $G : \mathbb{R}^n \to \mathbb{R}^n$ eine beliebige nichtlineare Funktion. Um die Gleichung

$$G(x) = 0, \quad x \in \mathbb{R}^n \tag{2.18}$$

iterativ zu lösen, verwendet man in der Regel das so genannte Newton-Verfahren. Ausgehend von einer Startschätzung $x^{(0)}$ liefert die Iterationsvorschrift

$$\begin{aligned} G'(x^{(k)})\Delta x^{(k)} &= -G(x^{(k)}) && (2.19)\\ x^{(k+1)} &= x^{(k)} + \Delta x^{(k)},\ k = 0, 1, 2, ... && (2.20) \end{aligned}$$

eine gegen die Lösung von (2.18) konvergierende Folge.

Definition 2.3.1. Eine beliebige Folge $(x^{(k)})_{k\in\mathbb{N}} \subset \mathbb{R}^n$ und $x^* \in \mathbb{R}^n$ mit

$$\lim_{k\to\infty} x^{(k)} = x^*$$

heißt

i. *(mindestens) linear konvergent*, falls ein $r \in (0, 1)$ und ein $K \in \mathbb{N}$ existieren, so dass

$$\|x^{(k+1)} - x^*\| \leq r\|x^{(k)} - x^*\|, \quad k \geq K,$$

ii. *(mindestens) superlinear konvergent*, falls eine Nullfolge $(\varepsilon_k)_{k\in\mathbb{N}} \subset \mathbb{R}^+$ und ein $K \in \mathbb{N}$ existieren, so dass

$$\|x^{(k+1)} - x^*\| \leq \varepsilon_k\|x^{(k)} - x^*\|, \quad k \geq K,$$

iii. *(mindestens) quadratisch konvergent*, falls ein $R > 0$ und ein $K \in \mathbb{N}$ existieren, so dass

$$\|x^{(k+1)} - x^*\| \leq R\|x^{(k)} - x^*\|^2, \quad k \geq K.$$

Bemerkung 2.3.2. Die durch (2.19) und (2.20) erzeugte Folge konvergiert superlinear.

Gehen wir von einem lediglich gleichungsrestringierten Problem (2.2.1) aus, das heißt gilt

$$I_G = \{1, ..., m\},$$

so bilden die KKT-Bedingungen in Satz 2.2.7 zusammen mit der dort vorausgesetzten Zulässigkeit ein nichtlineares Gleichungssystem:

$$K(x, \lambda) := \begin{bmatrix} \nabla_x L(x, \lambda) \\ g(x) \end{bmatrix} = 0.$$

Das Newton-Verfahren ist dafür definiert durch

$$\begin{bmatrix} x^{(k+1)} \\ \lambda^{(k+1)} \end{bmatrix} = \begin{bmatrix} x^{(k)} \\ \lambda^{(k)} \end{bmatrix} - \nabla_{(x,\lambda)} K(x^{(k)}, \lambda^{(k)})^{-1} \begin{bmatrix} \nabla_x L(x^{(k)}, \lambda^{(k)}) \\ g(x^{(k)}) \end{bmatrix}. \tag{2.21}$$

Die Matrix

$$\nabla_{(x,\lambda)} K(x^{(k)}, \lambda^{(k)}) = \begin{bmatrix} \nabla^2_{xx} L(x^{(k)}, \lambda^{(k)}) & \nabla g(x^{(k)})^T \\ \nabla g(x^{(k)}) & 0 \end{bmatrix}$$

ist sehr aufwändig zu berechnen. Um ein effizientes numerisches Verfahren zur Lösung der Newtongleichung (2.21) und damit des Problems 2.2.1 zu entwickeln, muss diese Berechnung so effizient wie möglich erfolgen. Ein Ansatz, den das so genannte *vereinfachte Newton-Verfahren* verfolgt, ist beispielsweise die Verwendung von

$$\nabla_{(x,\lambda)} K(x^{(0)}, \lambda^{(0)})$$

in jeder Iteration. Die Matrix braucht daher nicht in jeder Iteration neu berechnet zu werden. Dies gewährleistet zwar eine deutlich kürzere Rechenzeit pro Iteration, jedoch leidet darunter die Konvergenzrate in erheblichem Maße. Diese Arbeit befasst sich mit der effizienten Berechnung beziehungsweise Approximation der Matrix $\nabla_{(x,\lambda)} K(x^{(k)}, \lambda^{(k)})$.

2.4 SQP-Verfahren

Zur Lösung nichtlinearer Optimierungsprobleme haben sich in den vergangenen Jahren die so genannten SQP (**S**equentielle **Q**uadratische **P**rogrammierung)-Verfahren durchgesetzt. Für eine detaillierte Theorie über SQP-Verfahren sei auf [1], [35], [41], [50], [68], [73] und [80] verwiesen. Diese Arbeit zeigt Wege auf, wie diese Verfahren bezüglich ihrer Genauigkeit und vor allem ihres Rechenaufwandes verbessert werden können. Die grundlegende Idee hinter den SQP-Verfahren besteht darin, das nichtlineare Optimierungsproblem lokal durch eine quadratische Näherung darzustellen. Bei der Verwendung von SQP-Verfahren ist die hinreichende Differenzierbarkeit der auftretenden Funktionen in einer hinreichend großen Umgebung um das zu findende lokale Minimum allerdings Voraussetzung.
In Abbildung 2.1 sieht man die lokale quadratische Approximation einer nichtlinearen Funktion an vier unterschiedlichen Iterationspunkten. Die quadratische Näherung entsteht durch

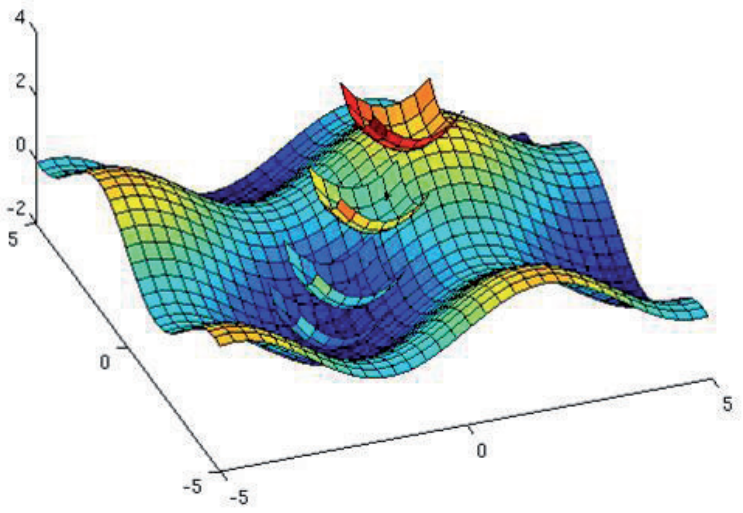

Abbildung 2.1: Quadratische Approximationen im SQP-Verfahren

die Taylor-Approximation zweiter Ordnung der Lagrangefunktion und durch die Linearisierung der Nebenbedingung. Hieraus resultieren in jeder Iteration quadratische Teilprobleme, so genannte *QP-Probleme*.
SQP-Verfahren gliedern sich demnach in Haupt- und Nebeniterationen. In den Nebeniterationen werden die QP-Teilprobleme gelöst, während in den Hauptiterationen eine Folge von Punkten $x^{(k)}$ erzeugt wird, die gegen die lokale Minimalstelle konvergiert.
Eine SQP-Hauptiteration besteht aus den folgenden wesentlichen Schritten (siehe Abbildung 2.2):

i. Auswertung der aktuellen Iterierten

ii. Überprüfung der Optimalitätsbedingungen

iii. Bereitstellung der Ableitungsinformationen beziehungsweise -approximationen

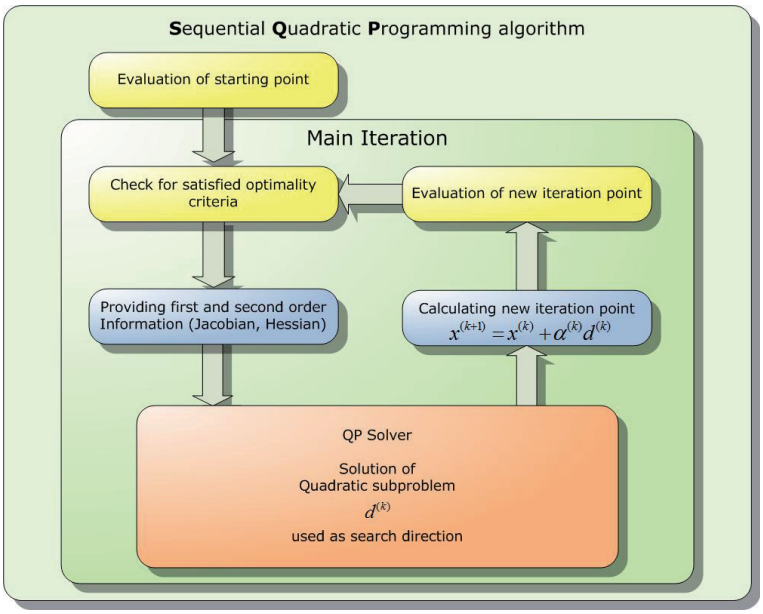

Abbildung 2.2: Grober Aufbau eines SQP-Verfahren

iv. Bestimmung der Suchrichtung durch Lösen des QP-Teilproblems,

v. Berechnung einer adäquaten Schrittweite und

vi. Berechnung der neuen Iterierten.

Die Auswertung der aktuellen Iterierten umfasst die Auswertung der Zielfunktion und Nebenbedingung, sowie deren Gradienten und Jacobimatrix. Mit diesen Auswertungen ist es möglich, die notwendigen Bedingungen erster Ordnung (2.6)-(2.8) zu überprüfen. Da folglich die ersten Ableitungen im Schritt iii bereits vorliegen, bezieht sich die Bereitstellung der Ableitungsinformationen eigentlich lediglich auf die Berechnung beziehungsweise Approximation der Hessematrix. Wie wir später sehen werden, empfiehlt es sich jedoch, die Berechnung der ersten und zweiten Ableitungen simultan durchzuführen.
Die Suchrichtung berechnet sich durch die Lösung des bereits erwähnten QP-Teilproblems.

Definition 2.4.1. Sei $H^{(k)}$ eine geeignete Approximation der Hesse-Matrix $\nabla^2_{xx} L(x^{(k)}, \lambda^{(k)})$ und $x^{(k)}$ in einer Umgebung der Optimallösung $\hat{x}$. Dann definiert

$$\min_{d \in \mathbb{R}^n} \quad \frac{1}{2} d^T H^{(k)} d + \nabla_x f(x^{(k)})^T d \tag{2.22}$$

$$\text{unter} \quad g_i(x^{(k)}) + \nabla g_i(x^{(k)}) d = 0, \quad i \in I_G \tag{2.23}$$

$$g_i(x^{(k)}) + \nabla g_i(x^{(k)}) d \leq 0, \quad i \in \{1, 2, ..., m\} \setminus I_G. \tag{2.24}$$

das quadratische Unterproblem zum NLP-Problem 2.2.1. Die Lösung $\hat{d} = d^{(k)}$ ist die Suchrichtung der k-ten Iteration.

Korollar 2.4.2. *Sei $\lambda^{(k)} \in \mathbb{R}^m$ der Vektor der zur optimalen Lösung $\hat{d}^{(k)}$ von (2.22)-(2.24) gehörenden Lagrange-Multiplikatoren. Dann gilt*

$$H^{(k)} \hat{d}^{(k)} + \nabla f(x^{(k)}) = -\lambda^{(k)T} \nabla g(x^{(k)}). \tag{2.25}$$

Beweis. Folgt direkt durch Auswertung der KKT-Bedingungen. □

Die *Strategie der aktiven Menge* sieht vor, zur Lösung des QP-Problems iterativ die Indexmenge $A(x^*) \subset \{1, ..., m\} \setminus I_G$ zu schätzen. Diese geschätzte Indexmenge bezeichnen wir mit J.
Um eine zulässige Lösung zu erhalten, werden zumindest zwischenzeitlich insbesondere diejenigen $i \in \{1, ..., m\} \setminus I_G$ auch zu J gehören, für die an der jeweiligen Iteration

$$g_i(x) > 0$$

gilt.
Diejenigen Indizes, die nicht zur Indexmenge J gehören, zählen (laut der Schätzung) nicht zur aktiven Menge und daher können diese Nebenbedingungen (zunächst) vernachlässigt werden. Bezeichne

$$\tilde{g} := (g_i,\ i \in J \cup I_G)$$

den Vektor der geschätzten aktiven Nebenbedingungen und $\tilde{\lambda}$ die zugehörigen Komponenten der Lagrange-Multiplikatoren. Mit dieser Schätzung erhalten wir ein gewöhnliches Newton-Verfahren: Das QP-Problem vereinfacht sich zu einem großen linearen Gleichungssystem:

$$\begin{bmatrix} H^{(k)} & \nabla\tilde{g}(x^{(k)})^T \\ \nabla\tilde{g}(x^{(k)}) & 0 \end{bmatrix} \begin{bmatrix} d^{(k)} \\ \tilde{\lambda}^{(k)} \end{bmatrix} + \begin{bmatrix} \nabla f(x^{(k)}) \\ \tilde{g}(x^{(k)}) \end{bmatrix} = 0.$$

Auch wenn die effiziente Lösung eines in unserem Falle derartig großen linearen Gleichungssystems alles andere als trivial ist, so ist ein solches Problem deutlich leichter zu lösen als ein allgemeines nichtlineares Optimierungsproblem. Problematisch wird die Lösung allerdings falls $H^{(k)}$ nicht positiv definit ist. Die Approximationen der Hessematrix werden daher stets so gestaltet, dass sie auf dem Kern der Jacobimatrix der aktiven Nebenbedingungen $\nabla\tilde{g}(x^{(k)})$ positiv definit sind, selbst dann, wenn dies für die tatsächliche Matrix $\nabla_{xx}L(x, \lambda)$ nicht gilt. Sind die strengen hinreichenden Optimalitätsbedingungen (2.13)-(2.16) erfüllt, so hat jedoch auch die tatsächliche Matrix $\nabla^2_{xx}L(x, \lambda)$ diese Eigenschaft.
Nachdem auf die beschriebene Weise eine Suchrichtung ermittelt worden ist, wird in jeder Hauptiteration eine Aktualisierung des Iterationspunktes $x^{(k)}$ der Form

$$x^{(k+1)} = x^{(k)} + \alpha^{(k)} d^{(k)} \tag{2.26}$$

durchgeführt. Hierbei bezeichnet $\alpha^{(k)}$ die Schrittweite in der k-ten Iteration. Die Bestimmung einer geeigneten Schrittweite ist hierbei die zentrale Aufgabe. Hierfür bedient man sich so genannter *Bewertungsfunktionen* (*Merit functions*). Eine Bewertungsfunktion besteht stets aus der Lagrangefunktion und einem additiven Ausdruck, welcher die Unzulässigkeit in den Nebenbedingungen „bestraft“. Man unterscheidet zwischen nicht differenzierbaren Bewertungsfunktionen, wie etwa die l_1-Penaltyfunktion

$$l_1(x, \rho) = L(x, \lambda) + \sum_{j \in I_G} \rho_j |g_j(x)| + \sum_{j \in \{1,...,m\} \setminus I_G} \rho_j \max\{g_j(x), 0\},$$

und differenzierbaren, wie etwa der erweiterten Lagrangefunktion

$$L_a(x, \lambda, \rho) = L(x, \lambda) + \frac{1}{2} \sum_{j \in I_G} \rho_j g_j(x)^2 + \frac{1}{2} \sum_{j \in \{1,...,m\} \setminus I_G} \rho_j \max\{g_j(x), 0\}^2.$$

Die so genannten *Penalty-Verfahren* sind darauf angelegt, lediglich die Bewertungsfunktionen mit geeignet gewählten ρ_j zu minimieren, wodurch aus dem beschränkten Problem ein unbeschränktes gebildet wird. Da Penalty-Verfahren den SQP-Verfahren jedoch weit unterlegen sind, werden sie im Rahmen dieser Arbeit nicht näher betrachtet. Die Bewertungsfunktion dient jedoch, wie oben bereits erwähnt, zur Schrittweitenbestimmung. Es erscheint sinnvoll, die Schrittweite $\alpha^{(k)}$ so zu bestimmen, dass

$$\alpha^{(k)} = \underset{\alpha\in(0,1]}{argmin}\ \varphi(\alpha) := \underset{\alpha\in(0,1]}{argmin}\ L_a(x^{(k)} + \alpha d^{(k)}, \lambda^{(k)}, \rho), \tag{2.27}$$

mit geeignet gewähltem ρ (analog für die l_1-Penalty-Funktion). Unter der Bedingung, dass die Suchrichtung d auch tatsächlich eine Abstiegsrichtung darstellt, das heißt dass

$$\varphi'(0) < 0$$

wird dadurch der Wert der jeweiligen Penaltyfunktion verringert. Das eindimensionale Minimierungsproblem (2.27) ist zwar exakt lösbar, allerdings ist eine schneller zu berechnende, suboptimale Schrittweite im Allgemeinen effizienter. Bekannte Regeln sind beispielsweise die Armijo-Regel mit und ohne Aufweitung und die (strenge) Wolfe-Powell-Regel.
Die Armijoregel lautet wie folgt: Zu vorgegebenen Zahlen $\beta \in (0,1)$ und $\sigma \in (0,1)$ bestimme

$$\alpha_i := \max\{\beta^j, \quad j = 0, 1, 2, ...\},$$

so dass

$$\varphi(\alpha_i) \leq \varphi(0) + \sigma \cdot \alpha_i \cdot \varphi'(0) \tag{2.28}$$

gilt.

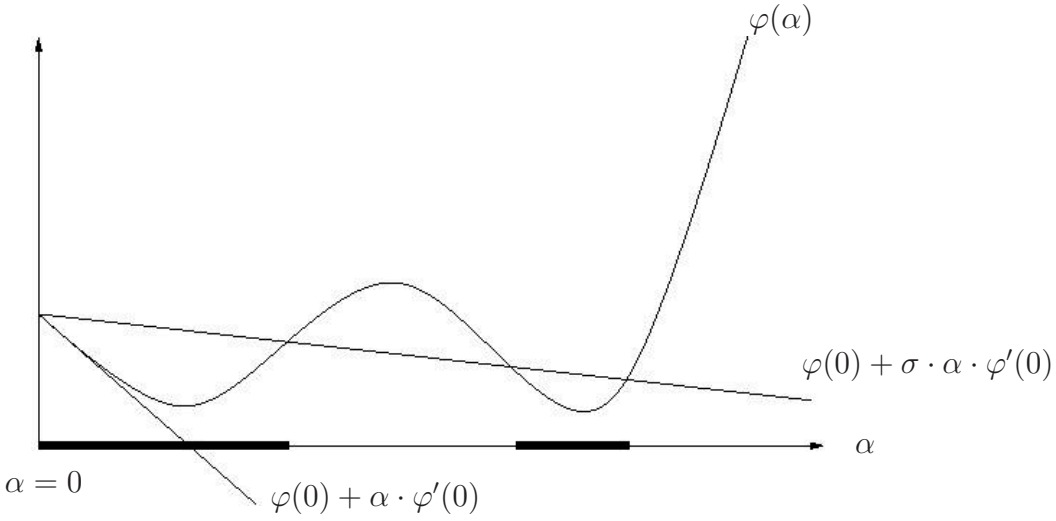

Abbildung 2.3: Schrittweiten, die der Armijoregel genügen.

Die fett eingezeichneten Intervalle in Abbildung 2.3 erfüllen die Armijo-Bedingung (2.28).
Bei der Armijoregel mit Aufweitung verfährt man wie folgt. Gilt die Armijo-Bedingung (2.28) bereits für

$$\alpha = 1,$$

so testet man, ob sie auch für

$$\alpha = \frac{\alpha}{\beta} > 1$$

gilt. Ist dies der Fall, wird so lange erneut durch β geteilt und erneut getestet bis die Armijo-Bedingung (2.28) nicht mehr gilt und das letzte α, welches die Bedingung erfüllte als Schrittweite verwendet.
Die Wolfe-Powell-Regel lautet: Zu vorgegebenen Zahlen $\sigma \in (0, 1/2)$ und $\varrho \in [\sigma, 1)$ bestimme $\alpha > 0$, so dass

$$\begin{aligned} \varphi(\alpha) &\leq \varphi(0) + \sigma \cdot \alpha \cdot \varphi'(0), \\ \varphi'(\alpha) &\geq \varrho \cdot \varphi'(0) \end{aligned}$$

gilt.
Die strenge Wolfe-Powell-Regel lautet: Zu vorgegebenen Zahlen $\sigma \in (0, 1/2)$ und $\varrho \in [\sigma, 1)$ bestimme $\alpha > 0$, so dass

$$\begin{aligned} \varphi(\alpha) &\leq \varphi(0) + \sigma \cdot \alpha \cdot \varphi'(0), \\ |\varphi'(\alpha)| &\leq -\varrho \cdot \varphi'(0) \end{aligned}$$

gilt. Offensichtlich gilt die Wolfe-Powell-Regel nur falls auch die Armijo-Regel gilt. Die Abbildungen 2.4 und 2.5 verdeutlichen die beiden Regeln.

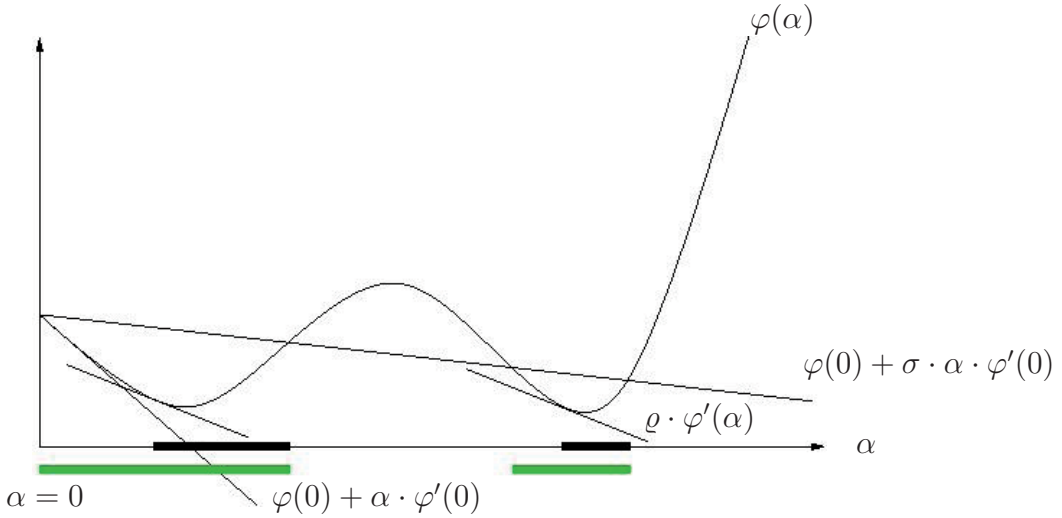

Abbildung 2.4: Bereiche (schwarz), die der Wolfe-Powell-Regel und Bereiche (grün), die der Armijoregel genügen.

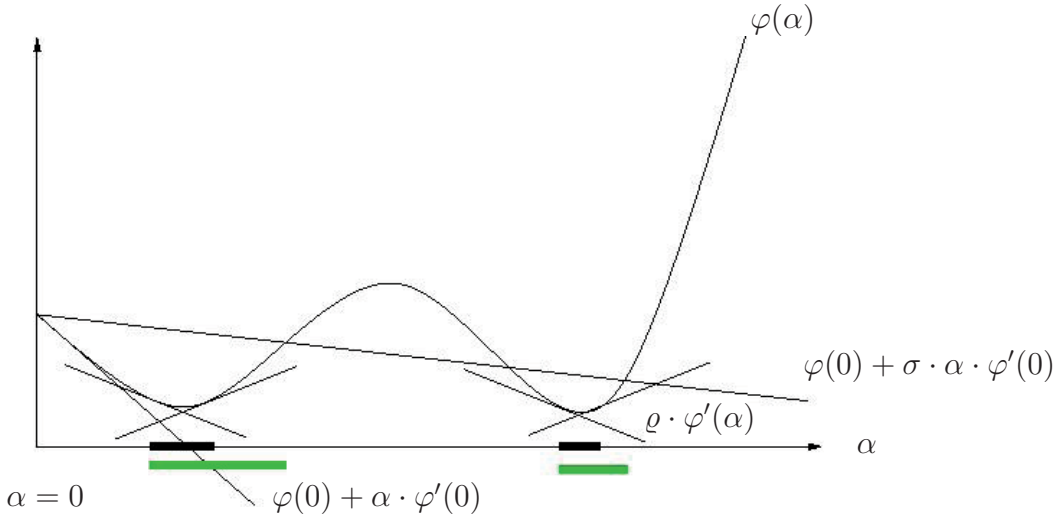

Abbildung 2.5: Bereiche (schwarz), die der strengen Wolfe-Powell-Regel genügen und Bereiche (grün), die der Wolfe-Powell-Regel genügen.

Da aufgrund der Taylorapproximation in einer hinreichend kleinen Umgebung

$$\begin{aligned} \varphi(1) &\approx \varphi(0) + \varphi'(0) \\ |\varphi'(\alpha)| &\approx \varphi'(0) \end{aligned}$$

gilt, leuchtet ein, dass nahe am Optimum die Schrittweite 1 stets akzeptiert wird.
Die neu bestimmte Iterierte wird anschließend erneut auf Optimalität überprüft und so weiter.

2.5 WORHP

Dieser Abschnitt ist an [13] angelehnt.
Die Software WORHP („**W**e **O**ptimize **R**eally **H**uge **P**roblems“) ist ein innovativer NLP-Solver, der in Zukunft bei der Missionsplanung der Europäischen Weltraumagentur Verwendung finden soll, und von der AG Optimierung und Optimale Steuerung am Zentrum für Technomathematik (ZeTeM) an der Universität Bremen seit 2010 unter www.worhp.de auch Interessenten aus dem akademischen oder kommerziellen Umfeld zur Verfügung gestellt wird. Die Software ist ein Ergebnis der Zusammenarbeit mit der Universität Würzburg, der School of Mathematics der University of Birmingham und der Universität der Bundeswehr München. Im Gegensatz zu den meisten NLP-Verfahren ist WORHP nicht als reine Testumgebung für mathematische Verfahren konzipiert, sondern als Werkzeug zur Lösung realer Optimierungsprobleme; neue mathematische Methoden werden bei WORHP parallel erprobt und bei Erfolg in den Solver integriert. Gefördert wird die Entwicklung vom Deutschen Zentrum für Luft- und Raumfahrt DLR, der Europäischen Weltraumorganisation ESA und von der Universität Bremen.
Auf dem Feld der NLP-Solver konkurriert WORHP mit etablierten Verfahren, von denen die meisten seit Jahrzehnten an Institutionen wie IBM, Boeing oder der University of Stanford entwickelt wurden. Unter Federführung des ZeTeM werden dank umfangreicher Arbeiten eines Teams junger Wissenschaftler bereits heute, nach gut vier Jahren Entwicklungszeit, fast alle konkurrierenden Verfahren bezüglich Performance und Robustheit überflügelt: Bezüglich der gängigen CUTEr-Testsets, siehe Abschnitt 5.1, ist WORHP Ende 2010 nicht nur 45-fach schneller als der schwächste Konkurrent und vergleichbar schnell wie der stärkste, sondern löst auch mehr Testprobleme als alle anderen getesteten Verfahren.
Im Gegensatz zu vielen seit den 70er-Jahren entwickelten Konkurrenz-Solvern ist WORHP außerdem durch die konsequente Anwendung moderner Programmiertechniken technisch auf dem Stand der Zeit und bietet Anwendern dadurch klare und einfache Schnittstellen.
Aktuell konzentriert sich die Entwicklung auf die Lösung höchstdimensionaler Probleme: In ersten Testreihen wurde bereits der lineare Zusammenhang von Speicherbedarf und Rechenzeit mit der Problemdimension empirisch bestätigt. Erste Testprobleme mit bis zu 400.000.000 Variablen und 800.000.000 Nebenbedingungen können von Worhp auf leistungsfähiger Hardware inzwischen in weniger als einer Stunde gelöst werden.
Für den Technologietransfer in industrielle Anwendungen bildet WORHP einen wichtigen Baustein im Portfolio der AG Optimierung und Optimale Steuerung, da alle praktischen Anwendungen im Kern auf einem Optimierungsverfahren basieren.
Alle im Rahmen dieser Arbeit entwickelten Verfahren wurden in WORHP implementiert. In Kapitel 5 finden sich Testrechnungen, die die Effekte der unterschiedlichen Verfahren auf die Performance von WORHP demonstrieren.

3 Finite Differenzen

3.1 Einleitung

"Wie zahlreich sind doch die Dinge, deren ich nicht bedarf."
Sokrates, 469-399 v. Chr.

Um innerhalb eines SQP-Verfahrens das QP-Problem (2.22)-(2.24) aufzustellen, bedarf es offensichtlich geeigneter Approximationen von $\nabla_x L(x,\lambda), \nabla^2_{xx} L(x,\lambda)$ und $\nabla g(x)$. Zunächst machen wir uns klar, dass

$$\nabla_x L(x,\lambda) = \nabla f(x) + \lambda^T \nabla g(x)$$

und

$$\nabla^2_{xx} L(x,\lambda) = \nabla^2 f(x) + \sum_{i=1}^{m} \lambda_i \nabla^2 g_i(x).$$

Ein gängiges Verfahren zur Bestimmung solcher Approximationen sind die Finiten Differenzen. Die Standardmethode ist der so genannte *Vorwärtsdifferenzenquotient*. Für eine beliebige Funktion, insbesondere für die Zielfunktion $f : \mathbb{R}^n \to \mathbb{R}$ lautet er

$$\frac{\partial f(x)}{\partial x_i} \approx \frac{f(x+\varepsilon e_i) - f(x)}{\varepsilon}, \tag{3.1}$$

wobei e_i den i-ten Einheitsvektor bezeichnet.
Die Motivation für den Vorwärtsdifferenzenquotienten ergibt sich aus der Taylorapproximation erster Ordnung, wonach

$$f(x+\varepsilon e_i) = f(x) + \varepsilon \nabla f(x)^T e_i + O(\varepsilon^2) = f(x) + \varepsilon \frac{\partial f(x)}{\partial x_i} + O(\varepsilon^2).$$

gilt. Eine höhere Ordnung kann durch den *zentralen Differenzenquotienten* erzielt werden. Dieser lautet

$$\frac{\partial f(x)}{\partial x_i} \approx \frac{f(x+\varepsilon e_i) - f(x-\varepsilon e_i)}{2\varepsilon}. \tag{3.2}$$

Auch diese Approximation rechtfertigt sich durch die Taylorapproximationen

$$f(x+\varepsilon e_i) = f(x) + \varepsilon \frac{\partial f(x)}{\partial x_i} + \varepsilon^2 \frac{\partial^2 f(x)}{\partial x_i^2} + O(\varepsilon^3) \text{ und} \tag{3.3}$$

$$f(x-\varepsilon e_i) = f(x) - \varepsilon \frac{\partial f(x)}{\partial x_i} + \varepsilon^2 \frac{\partial^2 f(x)}{\partial x_i^2} + O(\varepsilon^3). \tag{3.4}$$

(3.3) und (3.4) ergeben

$$f(x + \varepsilon e_i) - f(x - \varepsilon e_i) = 2\varepsilon \nabla f(x)^T e_i + O(\varepsilon^3),$$

weswegen offensichtlich der zentrale Differenzenquotient exakter ist als der Vorwärtsdifferenzenquotient. Zur Bestimmung des vollständigen Gradienten sind jedoch bei Verwendung des zentralen Differenzenquotienten mehr Funktionsauswertungen notwendig als beim Vorwärtsdifferenzenquotienten. Die Auswertung von $f(x)$ kann schließlich einmalig berechnet und für alle Komponenten des Gradienten verwendet werden, während beim zentralen Differenzenquotienten für jede Komponente zwei Funktionsauswertungen notwendig sind.
Unabhängig von der Wahl der Approximationsmethode (3.1) oder (3.2) sind im Falle der hochdimensionalen nichtlinearen Optimierung, bei der Millionen von Variablen behandelt werden, eine so hohe Anzahl an Funktionsauswertungen notwendig, dass die Methode der Finiten Differenzen, zumindest in dieser Form, numerisch zu aufwändig wird.
Abhilfe kann dadurch geschaffen werden, dass die *Dünnbesetztheit* der Probleme ausgenutzt wird. Hochdimensionale Optimierungsprobleme haben zumeist die Eigenschaft, dass die auftretenden Funktionen f und g_i nicht alle von allen Variablen $x_1, ..., x_n$ abhängen. Vielmehr hängt jede dieser Funktion in der Regel von lediglich einer kleinen Auswahl der Variablen ab. Das hat zur Folge, dass die meisten Einträge in Gradient, Jacobi- und Hessematrizen nur einen sehr kleinen Anteil von *Nichtnulleinträgen* haben. Unter einem *Nulleintrag* des Gradienten ∇f verstehen wir eine Komponente $(\nabla f(x))_k$ mit

$$(\nabla f(x))_k \equiv 0.$$

Entsprechend ist ein Nichtnulleintrag eine Komponente mit

$$(\nabla f(x))_k \not\equiv 0.$$

Analoges gilt für Jacobi- und Hessematrizen. Unter der Dünnbesetztheit versteht man den nur geringen Anteil an Nichtnulleinträgen innerhalb der auftretenden Matrizen. Wenn wir uns im Folgenden für die Dünnbesetztheitsstruktur eine Matrix interessieren, wird ein Nichtnulleintrag in der Matrix mit $\times$ gekennzeichnet während ein Nulleintrag einfach freigelassen wird (siehe beispielsweise (3.7)).
Da die Abhängigkeiten der auftretenden Funktionen bekannt sind, ist es sinnvoll, nur die Nichtnulleinträge zu berechnen (welche natürlich für bestimmte x durchaus dennoch Null ergeben können). Sofern nicht anders angemerkt, gehen wir im Folgenden stets davon aus, dass der Vorwärtsdifferenzenquotient verwendet werden soll. Prinzipiell lassen sich alle Ansätze auch auf den zentralen Differenzenquotienten übertragen. Demnach geht man für den Gradienten der Zielfunktion so vor, dass die Auswertung $f(x + \varepsilon e_i),\ i \in \{1, ..., n\}$ nur erfolgt, wenn f tatsächlich von x_i abhängt.
Der Vorwärtsdifferenzenquotient für die Bestimmung eines Eintrags der Hessematrix lautet

$$(\nabla^2 f(x))_{ij} = \frac{f(x + \varepsilon(e_i + e_j)) - f(x + \varepsilon e_i) - f(x + \varepsilon e_j) + f(x)}{\varepsilon^2} \tag{3.5}$$

Nach Berücksichtigung der Dünnbesetztheit ist es sinnvoll, dass die Auswertung

$$f(x + \varepsilon(e_i + e_j)), \quad i, j \in \{1, ..., n\}$$

nur dann erfolgt wenn f sowohl von x_i als auch von x_j abhängt. Die anderen in (3.5) auftretenden Funktionsauswertungen von f liegen bereits vor, sofern die erste Ableitung bereits berechnet wurde, wovon hier ausgegangen wird. Insofern ist die Berechnung der ersten und zweiten Ableitung der Zielfunktion f auf hinreichend effiziente Weise möglich.
Komplizierter ist die effiziente Bestimmung der Ableitungen der Nebenbedingungen g. Hierbei besteht in der Regel das Problem, dass der Vektor der Nebenbedingungen g nur komplett ausgewertet werden kann. Das bedeutet für die oben beschriebene Vorgehensweise, dass auf die Auswertung $g(x + \varepsilon e_i)$ höchstens dann verzichtet werden kann, falls keine der Nebenbedingungen von x_i abhängt. Dies wird äußerst selten der Fall sein, so dass diesbezüglich andere Wege der Einsparung gefunden werden müssen. Hiervon handelt dieses Kapitel.

3.2 Graph Coloring

3.2.1 NP-Vollständigkeit

Der Begriff der NP-Vollständigkeit wird in diesem Kapitel häufig verwendet. In [30] findet sich eine ausführliche Theorie über die NP-Vollständigkeit. Dieser Abschnitt soll einen Einblick bieten und orientiert sich an der angegebenen Quelle. In der Informatik trennt man strikt zwischen Problemen, die sich in polynomialer Zeit lösen lassen und anderen, so genannten *Intractable Problems*. Für ein gegebenes Problem gebe es eine Anzahl n von Inputgrößen, die die Komplexität eines Algorithmus' und damit die Rechendauer bestimmt. Damit lässt sich eine so genannte *Zeitkomplexitätsfunktion* t definieren, die von n abhängt. Es leuchtet ein, dass die tatsächliche Rechendauer eines Algorithmus' von weiteren Parametern des Problems abhängt, so dass die exakte Rechendauer im Allgemeinen nicht a priori ermittelbar ist. Von Interesse ist jedoch die Größenordnung, mit der die Komplexität zunimmt bei steigendem n.
Für eine Funktion $g : \mathbb{N} \to \mathbb{R}$ sagt man, eine Funktion $f : \mathbb{N} \to \mathbb{R}$ ist $O(g(n))$, wenn es eine konstante $c \in \mathbb{R}$ gibt, so dass gilt

$$|f(n)| \leq c \cdot |g(n)|, \quad n \in \mathbb{N}.$$

Definition 3.2.1. Ein *Algorithmus mit polynomialer Zeit* ist ein Algorithmus mit einer Zeitkomplexitätsfunktion t, die $O(p(n))$ ist für ein Polynom p. Alle anderen Algorithmen, die nicht so beschränkt werden können, werden als *Algorithmen mit exponentieller Zeit* bezeichnet.

Bemerkung 3.2.2. Es sei angemerkt, dass diese Definition nicht ganz exakt ist, da es gewisse nichtpolynomiale Zeitkomplexitätsfunktionen gibt, wie etwa $g(n) = n^{\log(n)}$, die wie polynomiale kategorisiert werden.

Der Grund für diese Unterscheidung wird durch die Tabelle 3.1 klar. Hier sind für mehrere Zeitkomplexitätsfunktionen die unterschiedlichen Wachstumsraten der Zeit in Abhängigkeit von den Größe n angegeben. Hieraus ist ersichtlich, dass auch die Fortentwicklung der Computerprozessoren angesichts des Zeitaufwandes bei Algorithmen mit exponentieller Zeit nur wenig bewirken kann.
Ein solches kombinatorisches Problems wird allgemein erst dann als „gut gelöst“ bezeichnet, wenn ein Algorithmus mit polynomialer Zeit gefunden wurde, der das Problem löst. Ein Problem wird als *intractable* bezeichnet, wenn kein Algorithmus mit polynomialem Aufwand gefunden werden kann, der es löst.
In der Praxis kann es unter Umständen passieren, dass sich Algorithmen mit exponentieller Zeit dennoch besser bewähren als andere, die vielleicht einen polynomialen Zeitaufwand haben. In der Tabelle 3.1 sieht man, dass für $n \leq 20$ ein Algorithmus mit der zugehörigen Zeitkomplexitätsfunktion

$$t(n) = 2^n$$

schneller ist als einer mit

$$t(n) = n^5.$$

Zeit-komplexitäts-funktion:	Anzahl n					
	10	20	30	40	50	60
n	10^{-5} s	$2 \cdot 10^{-5}$ s	$3 \cdot 10^{-5}$ s	$4 \cdot 10^{-5}$ s	$5 \cdot 10^{-5}$ s	$6 \cdot 10^{-5}$ s
n^2	10^{-4} s	$4 \cdot 10^{-5}$ s	$9 \cdot 10^{-4}$ s	$0,016$ s	$0,0025$ s	$0,0036$ s
n^3	$0,01$ s	$8 \cdot 10^{-5}$ s	$0,27$ s	$0,64$ s	$0,125$ s	$0,216$ s
n^5	$0,1$ s	$3,2$ s	$24,3$ s	$1,7$ Min	$5,2$ Min	$13,0$ Min
2^n	$0,01$ s	$1,0$ s	$17,9$ Min	$12,7$ Tage	$35,7$ Jahre	$3,7 \cdot 10^4$ Jahre
3^n	$0,059$ s	58 Min	$6,5$ Jahre	$3,9 \cdot 10^5$ Jahre	$2 \cdot 10^{10}$ Jahre	$1,3 \cdot 10^{15}$ Jahre

Tabelle 3.1: Verschiedene Zeitkomplexitätsfunktionen.

Des Weiteren kann die Abschätzung durch eine Zeitkomplexitätsfunktion nur für einige, womöglich exotische Sonderfälle eines Problems scharf sein, während der selbe Algorithmus für die meisten Anwendungen das Problem auf effiziente Weise löst. Die Zeitkomplexitätsfunktion gibt nur an, in welcher Größenordnung der Zeitaufwand eines Algorithmus' im schlimmsten Fall liegt.
Zu zeigen, dass ein Problem intractable ist, ist oft extrem schwierig. Als Alternative werden häufig Klassen von in einander überführbaren Problemen aufgestellt, für die bisher weder die Intractability bewiesen, noch ein Algorithmus mit polynomialem Aufwand gefunden werden konnte. Sollte es jemals gelingen, für ein Problem dieser Klasse einen Algorithmus mit polynomialem Aufwand zu finden, werden sich alle Probleme dieser Klasse als mit polynomialem Aufwand lösbare herausstellen. Eine dieser Klassen stellt die Klasse der *NP-vollständigen Probleme* dar, über die sich in [22] interessante Resultate finden.
Ein so genanntes *Entscheidungsproblem* ist ein Problem, dessen Lösung nur „ja“ oder „nein“ sein kann. Die Klasse NP ist die Klasse derjenigen Entscheidungsprobleme, die durch einen nichtdeterministischen Computer in polynomialer Zeit gelöst werden können. Die meisten Probleme, von denen man annimmt, dass sie intractable sind, gehören zu dieser Klasse. In [22] wird bewiesen, dass ein bestimmtes Problem in NP, das so genannte *Satisfiability Problem*, die Eigenschaft hat, dass jedes andere Problem in NP in dieses transformiert werden kann, aber nicht notwendigerweise umgekehrt. Wenn demnach das Satisfiability Problem mit polynomialem Aufwand gelöst werden könnte, so kann es auch jedes andere Problem in NP. Kann allerdings eines Tages nachgewiesen werden, dass irgendein Problem in NP intractable ist, so ist es auch das Satisfiability Problem. In diesem Sinne ist dieses Problem daher das „schwierigste“ in NP. Es stellt sich heraus, dass mehrere Probleme in NP die Eigenschaften dieses schwierigsten Problems aufweisen. Daher gibt es also mehrere „schwierigste“ Probleme. Diese Unterklasse von äquivalenten Problemen in NP bilden die Klasse

der *NP-vollständigen* Probleme. All diejenigen Probleme, in die NP-vollständige Probleme transformiert werden können, werden *NP-hard* genannt. Diese sind somit mindestens so „schwierig“ (hard) wie die NP-vollständigen. Diese Probleme können höchstens dann in polynomialer Zeit gelöst werden, wenn es für die NP-vollständigen möglich ist.

3.2.2 Definition und Eigenschaften des Graph Coloring Problems

Die hier vorgestellten Resultate sind an [21] angelehnt. Zunächst definieren wir die wichtigsten Begriffe dieses Abschnitts.

Abbildung 3.1: Bekanntes Graph Coloring Problem

Definition 3.2.3. Ein *Graph* ist ein geordnetes Paar $Graph = (V, E)$, wobei V eine endliche, nichtleere Menge von *Ecken* (vertices) ist. Die *Kanten* (edges) E sind ungeordnete Paare von unterschiedlichen Ecken. Daher

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

Die Ecken u und v sind *angrenzend*, wenn es eine Kante (u, v) gibt mit Endpunkten u und v. Die Anzahl der Kanten wird mit $|E|$ bezeichnet.
Ein p-Coloring, $p \in \mathbb{N}$, eines Graphen $Graph = (V, E)$ ist eine Funktion

$$\tilde{\Phi} : V \to \{1, 2, ..., p\},$$

so dass $\tilde{\Phi}(u) \neq \tilde{\Phi}(v)$, falls u und v angrenzend sind. Besitzt der Graph ein p-Coloring, so bezeichnen wir ihn als *p-colorabel* und das kleinste p, für das der Graph p-colorabel ist, wird als *chromatische Zahl* $\chi(Graph)$ des Graphen bezeichnet. Ein p-Coloring ist *optimal*, falls $p = \chi(Graph)$.
Ein Graph $Graph_0 = (V_0, E_0) \subset G$ heißt *Subgraph* des Graphen $Graph = (V, E)$, wenn $V_0 \subset V$ und $E_0 \subset E$. Zu einer nichtleeren Teilmenge V_0 von V ist der Graph $Graph_0 = (V_0, E_0)$ von V_0 *induziert*, falls

$$E_0 = \{(u, v) : (u, v) \in E, u, v, \in V_0\}.$$

Sind alle Ecken von V_0 an einander angrenzend, dann heißt $Graph_0$ *vollständiger Subgraph* oder *Clique*. Von einer Clique spricht man jedoch nur bei mindestens zwei Ecken.

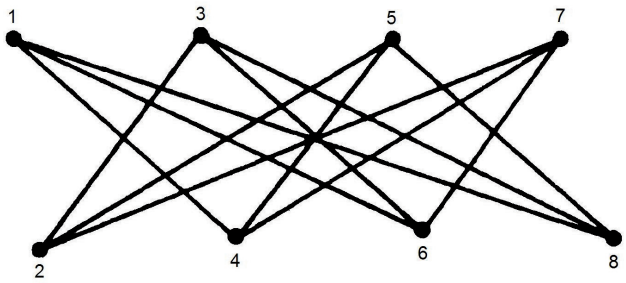

Abbildung 3.2: Zweiteiliger Graph

Bemerkung 3.2.4. Ein 2-colorabler Graph wird auch *zweiteilig* oder *bipartit* genannt (s. Abb. 3.2).
Der Begriff "coloring" entstammt aus der Vorstellung, dass die Werte von $\tilde{\Phi}$ Farben seien, mit denen die Ecken angemalt (coloriert) werden, siehe dazu die Abbildungen 3.3 und 3.4.

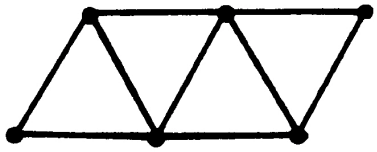

Abbildung 3.3: Zu colorierender Graph

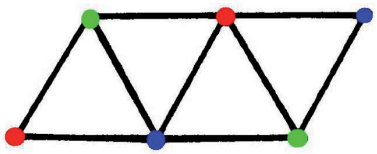

Abbildung 3.4: Colorierter Graph

Eine Ecke v, die an keine andere angrenzt, ist insofern unproblematisch, da sie mit einer beliebigen Farbe coloriert werden kann, das heißt $\tilde{\Phi}(v)$ kann beliebig festgesetzt werden. Daher werden im Folgenden nur Ecken betrachtet, die an andere angrenzen. Kann ein Graph in mehrere Subgraphen aufgeteilt werden, so dass zwischen diesen Teilen keine verbindende Kante besteht, kann man die beiden Graphen unabhängig von einander colorieren. Daher kann ohne Beschränkung der Allgemeinheit auch ein solcher Fall ausgeschlossen werden. Es werden daher nur verbundene Graphen betrachtet.
Ein p-Coloring $\tilde{\Phi}$ eines Graphen $Graph = (V, E)$ induziert eine Partitionierung der Ecken in die Teilmengen (im folgenden *Gruppen* genannt) $\tilde{\Phi}^{-1}(c),\ c = 1, 2, ..., p$. Es gibt eine untere Grenze für die chromatische Zahl:

Satz 3.2.5. *Sei* $Graph := (V, E)$ *ein Graph mit den Cliquen*

$$Graph_i = (V_i, E_i) \subset Graph,\ i = 1, ..., M.$$

Bezeichne

$$\rho_i := |V_i|$$

die Anzahl der Ecken in $Graph_i,\ i = 1, ..., M$. *Dann gilt*

$$\max_{i \in \{1,...,M\}} \rho_i \leq \chi(Graph). \tag{3.6}$$

Beweis. In einer Clique $Graph_i = (V_i, E_i),\ i \in \{1, ..., M\}$ von $Graph$ sind definitionsgemäß alle Ecken aneinander angrenzend. Damit gilt für alle Ecken $u, v \in V_i$:

$$\tilde{\Phi}(u) \neq \tilde{\Phi}(v).$$

Daher müssen mindestens ρ_i verschiedene Farben vorhanden sein. □

In [31] wird gezeigt, dass bereits das Entscheidungsproblem, ob ein Graph 3-colorabel ist oder nicht, ein NP-vollständiges Problem ist. Damit ist klar, dass wenn die NP-vollständigen Probleme nicht in polynomialer Zeit lösbar sind, wovon nach heutigem Stand der Wissenschaft ausgegangen werden muss, das allgemeine Graph Coloring Problem erst recht nicht in polynomialer Zeit gelöst werden kann. Das Entscheidungsproblem hingegen, ob ein Graph zweiteilig ist oder nicht, ist nicht NP-vollständig.
Im Allgemeinen wird man sich mit suboptimalen Algorithmen zufrieden geben müssen, die dafür jedoch in polynomialer Zeit Lösungen zu liefern vermögen.

3.3 Graphentheoretischer Ansatz zur Bestimmung erster Ableitungen

Auch die Überlegungen dieses Abschnittes knüpfen teilweise an [21] an. Wir nehmen an, wie bereits eingangs erwähnt, dass der Vektor g nur komplett ausgewertet werden kann. Wie in Abschnitt 3.1 gesehen, führt dies bei der Berechnung der Jacobimatrix ∇g zu einigen unnötigen Auswertungen. Die hier vorgestellte Strategie erlaubt es, diese überschüssigen Auswertungen dahingehend zu verwenden, dass gleichzeitig weitere Ableitungsinformationen berechnet werden können. Dies impliziert, dass deutlich weniger Auswertungen erfolgen müssen, um die gesamte Jacobimatrix ∇g zu berechnen.

Beispiel 3.3.1. Die zu berechnende Jacobimatrix der vektorwertigen Funktion g habe folgende Struktur:

$$\nabla g(x) = \begin{bmatrix} \times & \times & & & & \\ \times & \times & \times & & & \\ & \times & \times & \times & & \\ & & \times & \times & \times & \\ & & & \times & \times & \times \\ & & & & \times & \times \end{bmatrix} \tag{3.7}$$

Aus dieser Strukturinformation wird ersichtlich, dass bei der Auswertung von g an der Stelle

$$x + \varepsilon e_1,$$

die normalerweise für die Berechnung der ersten Spalte verwendet wird, lediglich die ersten zwei Einträge g_1 und g_2 von Bedeutung sind. Aufgrund der Struktur erkennen wir, dass

$$g_k(x + \varepsilon e_1) = g_k(x), \quad k = 3, 4, 5, 6.$$

Wie man jedoch unschwer erkennt, gibt es keine Gleichung g_k, die sowohl von x_1 als auch von x_4 abhängt. Demnach kann eine Funktionsauswertung gespart werden, wenn man g an der Stelle

$$x + \varepsilon(e_1 + e_4)$$

auswertet. In den Einträgen g_1 und g_2 ändert sich dadurch nichts im Vergleich zur Auswertung an der Stelle

$$x + \varepsilon e_1.$$

Die Werte in g_3, g_4 und g_5 erlauben es, darüber hinaus die vierte Spalte von ∇g zu bestimmen. Da alle anderen Variablen gemeinsame Gleichungen mit x_1 oder x_4 haben, sind die Möglichkeiten für weitere Einsparungen durch diese eine Auswertung an dieser Stelle erschöpft. In gleicher Weise lassen sich offensichtlich x_2 und x_5 zusammenfassen, ebenso x_3 und x_6.

Wir definieren etwas formaler:

Definition 3.3.2. Die Indexmenge

$$Rele_g(i) := \left\{ k \in \{1, .., m\} : \ \frac{\partial g_k(x)}{\partial x_i} \not\equiv 0 \} \right\}$$

bezeichnet die für den Index i *relevanten Einträge* von g.
Eine Abbildung

$$\Phi : \{1, 2, ..., n\} \to \{1, ..., p\}$$

beschreibe eine *Partitionierung* der Spalten(-indizes) einer $m \times n$-Matrix A. Eine Partitionierung heißt *konsistent*, wenn alle Spalten i, j mit

$$\Phi(i) = \Phi(j)$$

keine Nichtnulleinträge in der selben Zeilenposition haben.

Korollar 3.3.3. *Φ ist genau dann eine konsistente Partitionierung der Matrix ∇g, falls*

$$\forall i, j \text{ mit } \Phi(i) = \Phi(j) : \quad Rele_g(i) \cap Rele_g(j) = \emptyset$$

gilt. Dies wiederum ist äquivalent dazu, dass für alle i, j mit $\Phi(i) = \Phi(j)$

$$\forall k \in Rele_g(i) : \ g_k(x + \varepsilon e_j) = g_k(x), \qquad x \in \mathbb{R}^n \ \text{ und} \tag{3.8}$$
$$\forall k \in Rele_g(j) : \ g_k(x + \varepsilon e_i) = g_k(x), \qquad x \in \mathbb{R}^n. \tag{3.9}$$

gilt.

Im Beispiel 3.3.1 wurden drei Gruppen erstellt, die jeweils zwei Variablen enthalten. Für jede Gruppe benötigt man eine Auswertung des Vektors g.
Zur möglichst effizienten Berechnung einer dünn besetzten Jacobimatrix muss es daher das Ziel sein, eine Einteilung aller Variablen in derartige Gruppen zu finden, so dass die Anzahl der Gruppen minimal wird. Für dieses einfache Beispiel ist dies gelungen. Das ist daran ersichtlich, dass die Einträge g_2 bis g_5 jeweils von drei Variablen abhängen. Folglich bedarf es für diese Komponenten mindestens dreier Auswertungen. Somit lässt sich bereits an dieser Stelle festhalten, dass die maximale Anzahl der Variablen, von denen eine Nebenbedingung abhängt, als untere Grenze für die optimale Anzahl an Gruppen dient. Im folgenden wird ersichtlich, dass es keinesfalls immer möglich ist, diese untere Grenze zu erreichen. Nichtsdestotrotz dient diese Grenze als hinreichender Indikator für die Optimalität einer Gruppeneinteilung.
Das Problem kann somit wie folgt formuliert werden:

Problem 3.3.4. *Sei $A \in \mathbb{R}^{m \times n}$ die zu berechnende Matrix.*

$$\min \{p : \ \exists \text{ konsistente Partitionierung } \Phi : \{1, ..., n\} \to \{1, ..., p\} \text{ der Matrix } A\}$$

Das heißt: Erstelle eine konsistente Partitionierung der Spalten der Matrix A mit der kleinstmöglichen Anzahl von Gruppen.

Das Problem 3.3.4 ist nichts anderes als ein Graph Coloring Problem, wie im folgenden gezeigt wird. Dafür benötigen wir zunächst einen Graphen $Graph(A)$, der die Matrix A in geeigneter Weise beschreibt.

Definition 3.3.5. Sei A eine $m \times n$-Matrix. Der Graph

$$Graph(A) := (E, V)$$

mit den Ecken

$$V = \{v_1, v_2, ..., v_n\},$$

ist der *von der Matrix A induzierte Graph* falls gilt:

$$(v_i, v_j) \in E \Longleftrightarrow \exists k \in 1, ..., m : \ A_{ki} \neq 0 \wedge A_{kj} \neq 0$$

Für den Fall einer matrixwertigen Funktion $A(x)$ gilt für den von $A(x)$ induzierten Graphen $Graph(A(x)) = (E, V)$:

$$(v_i, v_j) \in E \Longleftrightarrow \exists k \in 1, ..., m : \ A_{ki} \not\equiv 0 \wedge A_{kj} \not\equiv 0$$

Hiermit erkennt man:

Bemerkung 3.3.6. Der Graph aus Abbildung 3.3 beschreibt die Strukturmatrix (3.7).

Damit ist folgendes leicht einzusehen:

Satz 3.3.7.

$$\tilde{\Phi} : V \to \{1, ..., p\}$$

ist genau dann ein Coloring von $Graph(A)$ *wenn*

$$\Phi : \{1, ..., n\} \to \{1, ..., p\}$$

mit

$$\Phi(i) = \tilde{\Phi}(v_i)$$

eine konsistente Partitionierung der Spalten von A induziert.

Aufgrund von Satz 3.3.7 kann bei der Erstellung der Partitionierung einer Matrix auf die zahlreichen Verfahren zur Lösung des Graph-Coloring Problems zurückgegriffen werden. Aufgrund der Identifikation der Partitionierung über die Colorierung können wir, da wir die Ecken des betrachteten Graphen mit 1 bis n indizieren, die Abbildung Φ auch als Coloring auffassen und daher im Folgenden auf die Bezeichnung $\tilde{\Phi}$ verzichten.
Nach Satz 3.2.5 ist die minimal erreichbare Gruppenzahl mindestens so groß wie die Anzahl der Elemente in der größten Clique. Die Zeilenelemente bilden offensichtlich eine Clique, weswegen die Aussage von oben, dass es mindestens so viele Gruppen gibt wie die maximale Anzahl an Elementen in den Zeilen ist, bestätigt wird. Es ist jedoch keinesfalls so, dass die maximale Anzahl der Elemente in den Zeilen gleichzusetzen ist mit der Anzahl der Elemente in der größten Clique eines Graphen. Betrachte beispielsweise die Matrix

$$\begin{bmatrix} \times & \times & \\ & \times & \times \\ \times & & \times \end{bmatrix},$$

deren maximale Anzahl an Zeilenelementen zwei beträgt. Die drei zu den Spalten gehörigen Kanten bilden jedoch eine Clique. Darüber hinaus ist, wie bereits erwähnt, auch die Anzahl der Elemente in der größten Clique nicht notwendigerweise gleich der chromatischen Zahl des zugehörigen Graphen.
In Abschnitt 3.2.2 wurde der triviale Fall einer Ecke ohne angrenzende Ecken vernachlässigt. Dies ist gleichbedeutend mit einer Variable, die in der betrachteten Matrix mit keiner anderen Variable in einer Zeile auftritt. Eine solche Variable kann offensichtlich in jede beliebige Gruppe einsortiert werden, weswegen ein solcher Fall auch bei dem Problem 3.3.4 vernachlässigt werden kann. Gibt es innerhalb einer Matrix mehrere Blöcke, so dass für jede Spalte gilt, dass sie nur in Zeilen eines Blocks Nichtnulleinträge hat, können die Blöcke als einzelne Matrizen einzeln partitioniert werden. Dies ist gleichbedeutend mit dem in Abschnitt 3.2.2 ebenfalls vernachlässigten Fall nicht zusammenhängender Subgraphen.
In Abschnitt 3.4 präsentieren wir einige aufwändige und weniger aufwändige heuristische Methoden zur Lösung des Graph Coloring Problems beziehungsweise des Partitionierungsproblems, welche zur effizienten Ableitungsbestimmung verwendet werden können. Da sich die Struktur der Ableitungen nicht ändert, muss die Gruppenbestimmung bei einem Optimierungsproblem nur einmalig erfolgen, während die Ableitungen selber sehr viel häufiger berechnet werden müssen. Insofern kann es sich unter Umständen lohnen, einen aufwändigen Partitionierungsalgorithmus zu verwenden, auch wenn man dadurch die Anzahl der Gruppen nur unwesentlich verkleinert.

3.4 Heuristische Algorithmen zur Lösung des Graph Coloring Problems

Dieser Abschnitt ist an die Arbeiten [4], [21], [23], [52], [53], [60], [61] und [63] angelehnt. Es werden diverse Algorithmen zur Lösung des Graph Coloring Problems vorgestellt.

3.4.1 Der CPR-Algorithmus

A.R. Curtis, M.J.D. Powell und J.K. Reid gelten als die Väter des Gruppenansatzes zur Berechnung dünn besetzter Jacobi-Matrizen. Sie führten in [23] einen Algorithmus ein, der auf relativ intuitive Weise die Colorierung vornimmt. Dieser wird in Anlehnung an die Namen der Urheber allgemein *CPR-Algorithmus* genannt. Hierbei werden die Ecken $v_1, v_2, ..., v_n$ nacheinander abgearbeitet und der „kleinstmöglichen“ Farbe (welche durch die Zahlen 1 bis p identifiziert werden) zugeordnet. Die Ecken werden nacheinander coloriert und sofern keine Farbe existiert, der die Ecke v_i zugeordnet werden kann, wird eine neue Farbe definiert. Für die Colorierung $\Phi : \{1, ..., n\} \to \{1, ...p\}$ gilt demnach

$$\Phi(i) = \min\{m \in \mathbb{N} : \ m \neq \Phi(j), \ v_j \text{ angrenzend an } v_i, \ j < i\}, \tag{3.10}$$

wobei p, wie erwähnt, ex ante unbekannt ist.
Das Ergebnis des CPR-Algorithmus' hängt in erheblichem Maße von der Nummerierung der Ecken ab.

Beispiel 3.4.1. Betrachte den Graphen in Abbildung 3.5. Dieser Graph ist zweiteilig. Unter

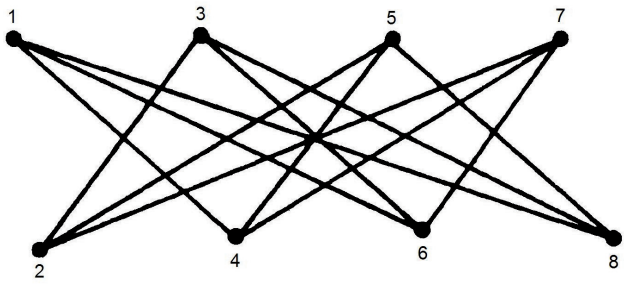

Abbildung 3.5: Zweiteiliger Graph

Verwendung der hier angegebenen Nummerierung, liefert der CPR-Algorithmus jedoch eine 4-Colorierung. Betrachtet man dagegen erst die „ungeraden“ Ecken und dann die geraden, erhält man eine 2-Colorierung.

Die Defizite dieses Algorithmus' werden noch klarer, wenn man den Graph in analoger Weise in die Breite fortführt. Hat der Graph dann $2 \cdot n$ Ecken, ermittelt der CPR-Algorithmus bei

ungünstiger Nummerierung eine n-Colorierung, obwohl auch dann zwei Farben ausreichen. Man kann den CPR-Algorithmus dahingehend abändern, dass man die Ecken nach gewissen Regeln permutiert und somit der optimalen Lösung zumindest näher kommt. Dies soll Gegenstand der folgenden beiden Algorithmen sein. Bei der Anordnung der Ecken spielt der Grad der Ecken eine zentrale Rolle.

Definition 3.4.2. Die Anzahl der Kanten, die die Ecke v als Start- oder Endpunkt haben, wird der *Grad* $d(v)$ der Ecke v genannt. Unterscheidet man zwischen dem Grad bezüglich des ganzen Graphen

$$Graph = (V, E)$$

und dem Grad bezüglich eines Subgraphen

$$Graph_0 = (V_0, E_0)$$

von $Graph$, bezeichnet man mit $d_{Graph_0}(v)$ den Grad einer Ecke bezüglich des Subgraphen $Graph_0$. Den minimalen Grad eines Graphen $Graph$ bezeichnen wir mit

$$\delta(Graph) = \min_{v \in V} d(v).$$

Damit kann man eine Aussage über die Farbe einer jeden Ecke treffen:

Proposition 3.4.3. *Es sei* $Graph := (V, E)$ *ein Graph und* $\Phi : \{1, ..., n\} \to \{1, ..., p\}$ *eine zulässige Colorierung. Für sämtliche Anordnungen von* V *gilt, dass*

$$\Phi(i) \leq 1 + d_{Graph_i}(v_i),$$

wobei $Graph_i$ *den von* $\{v_1, v_2, ..., v_i\}$ *induzierten Subgraphen von* $Graph$ *bezeichnet.*

Beweis. Folgt unter der Annahme, dass alle an v_i angrenzenden Ecken unterschiedliche Farben haben. □

Damit kann eine Abschätzung für p getroffen werden.

Proposition 3.4.4. *Der CPR-Algorithmus liefert im schlimmsten Fall eine Colorierung mit*

$$\max_{i \in \{1,...,n\}} \{\min[d(v_i) + 1, i]\} \tag{3.11}$$

Farben.

Beweis. Vgl. [21]. Aus Proposition 3.4.3 folgt bereits, dass

$$p \leq \max_{i \in \{1,...,n\}} d(v_i) + 1.$$

Man gehe nun vom schlimmsten Fall aus, das heißt die Ecke v_i sei an alle Ecken $v_1, ..., v_{i-1}$ angrenzend, ist damit Teil von mindestens $i - 1$ Kanten. Damit ist

$$d_{Graph_i}(v_i) = i - 1$$

und somit bekommt v_i die Farbe i zugewiesen. □

3.4.2 Der LFO-Algorithmus

Der *LFO*(Largest - First - Ordering)- *Algorithmus* von D.J.A.Welsh und M.B.Powell [85] ist durch die Minimierung der in Proposition 3.4.4 gegebenen worst-case-Abschätzung motiviert. Dies wird dadurch erreicht, dass die Ecken v_i so angeordnet werden, dass die Folge

$$(d(v_i))_{i=1,...,n} \tag{3.12}$$

monoton fallend ist. Auf die auf diese Weise permutierten Ecken wird der CPR-Algorithmus angewandt. Bedauerlicherweise wird der Graph 3.5 auch mit dem LFO-Algorithmus nicht notwendigerweise optimal coloriert. Da alle Ecken den Grad 3 haben, findet keine echte Permutation statt und die Algorithmen unterscheiden sich nicht. Der LFO-Algorithmus vermag es zwar, das Ergebnis im worst-case zu verkleinern, weist jedoch im Allgemeinen ähnliche Defizite auf wie der CPR-Algorithmus.

3.4.3 Der SLO-Algorithmus

Der *SLO* (Smallest-Last-Ordering) - *Algorithmus* von D.W.Matula, G.Marble, J.D.Isaacson [61] beinhaltet eine weitere Variante der Permutation, bevor der CPR-Algorithmus angewandt wird. Man nehme an, dass die Ecken $v_{k+1}, ..., v_n$ ausgewählt worden seien. Die Ecke v_k soll nun so gewählt werden, dass sie im Subgraphen $Graph_k$, der von

$$V - \{v_{k+1}, ..., v_n\} \tag{3.13}$$

induziert wird, minimalen Grad hat. Man beachte, dass dabei diejenigen Kanten, die $Graph_k$ aus (3.13) mit $\{v_{k+1}, ..., v_n\}$ verbinden, nicht berücksichtigt werden. Die Minimalität ist demnach nur im Vergleich zu den anderen Ecken im Subgraphen gefordert und nicht im Vergleich zu allen Ecken des Graphen *Graph*.
Nach Gleichung (3.10) gilt mit dieser Anordnung

$$\Phi(i) \leq 1 + d_{Graph_i}(v_i) = 1 + \delta(Graph_i).$$

Damit gilt für die Anzahl der Farben:

$$p = \max_{i \in \{1,...,n\}} \Phi(i) = 1 + \max_{i \in \{1,...,n\}} \delta(Graph_i).$$

Bevor man die SL-Anordnung durchgeführt hat, kennt man allerdings

$$\delta(Graph_i),\ i = 1, ..., n$$

nicht. Darum schätzt man etwas großzügiger ab, indem man sagt, dass der SLO-Algorithmus im schlimmsten Fall

$$1 + \max_{Graph_0 \subset Graph} \delta(Graph_0) \tag{3.14}$$

Farben benötigt.

Satz 3.4.5. *Die Anzahl (3.14) der Farben, die vom SLO-Algorithmus im schlimmsten Fall erstellt werden, ist kleiner als die Anzahl (3.11) der Farben, die der CPR-Algorithmus im schlimmsten Fall erzeugt.*

Beweis. Man nehme an, dass die Aussage falsch sei, so dass

$$\max_{i\in\{1,\dots,n\}}\{\min[d(v_i)+1,i]\} < 1 + \max_{Graph_0\subset Graph}\delta(Graph_0).$$

Die linke Seite sei mit max_{CPR} bezeichnet. Sei

$$\tilde{i} = argmax_{i\in\{1,\dots,n\}}\{\min[d(v_i)+1,i]\}.$$

Aufgrund der Maximierung über ein Minimum untersuchen wir zwei Fälle.
$\max_{CPR} = d(v_{\tilde{i}})+1 < \tilde{i}$:
Es gilt

$$\forall v \in V_0 :\ d(v) \geq \delta(Graph_0).$$

Ein Subgraph $Graph_0$ mit Grad $\delta(Graph_0)$ muss notwendigerweise mindestens $\delta(Graph_0)+1$ Ecken haben. Da

$$\delta(Graph_0) > 1 + d(v_{\tilde{i}}),$$

muss es folglich im Graphen $Graph$ mindestens $2 + d(v_{\tilde{i}})$ Ecken v geben mit

$$d(v) \geq \delta(Graph_0) > d(v_{\tilde{i}}).$$

Unabhängig davon, wie die Ecken angeordnet sind, gibt es daher notwendigerweise eine Ecke v_j auf der Position $j > d(v_{\tilde{i}})$ mit

$$d(v_j) > d(v_{\tilde{i}}).$$

Somit gilt jedoch

$$\min[d(v_j)+1, j] > d(v_{\tilde{i}}),$$

was einen Widerspruch ergibt.
$\max_{CPR} = \tilde{i} < d(v_{\tilde{i}})+1$:
Es kann insgesamt bis zu $\tilde{i}$ Ecken v geben mit

$$d(v) \geq \tilde{i} + 1.$$

Nach Annahme gibt es einen Subgraphen $Graph_0$ mit

$$\delta(Graph_0) \geq \tilde{i} + 1$$

Dieser Subgraph muss mindestens $\tilde{i} + 2$ Ecken v haben, für die

$$d(v) \geq d_{Graph_0}(v) \geq \tilde{i} + 1,$$

was ebenfalls einen Widerspruch ergibt. □

Der SLO-Algorithmus ist das erste hier vorgestellte Verfahren, das in der Lage ist, den Graphen in Abbildung 3.5 optimal zu colorieren, das heißt mit Hilfe von nur zwei Farben. Wie die LF-Anordnung ist auch die SLO nicht notwendigerweise eindeutig. Man stellt jedoch leicht fest, dass bei diesem Beispiel alle SL-Anordnungen die gleiche Anzahl an Farben erzeugt.
Dennoch gibt es zweiteilige Graphen, für die der SLO-Algorithmus unbefriedigende Ergebnisse liefert. Man betrachte beispielsweise den Graphen in Abbildung 3.6. Dieser „dreizeilige“

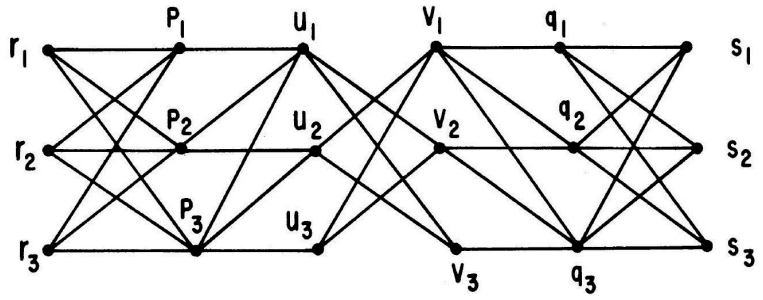

Abbildung 3.6: Zweiteiliger Graph 2

Graph kann in analoger Weise auf n Zeilen ausgeweitet werden. Er bleibt jedoch stets zweiteilig. Dies sieht man leicht ein, denn abwechselnde Colorierung der „Spalten“ ist zulässig und führt damit trivialerweise zur Zweiteiligkeit.
Leider ist die zeilenweise Anordnung keine SLO. Eine SLO ist beispielsweise:

$$(q_1, s_1, ..., q_n, s_n, p_1, r_1, ..., p_n, r_n, u_1, v_1, ..., u_n, v_n),$$

wobei n die Anzahl der Zeilen sei. Man prüft leicht nach, dass mit dieser Anordnung der CPR-Algorithmus

$$\begin{aligned} \Phi(p_i) = \Phi(q_i) &= 1, \\ \Phi(r_i) = \Phi(s_i) &= 2, \\ \Phi(u_i) = \Phi(v_i) &= i + 1 \end{aligned}$$

liefert. Damit benötigt der SLO-Algorithmus bei diesem Graph $n + 1$ Farben.
Der auf Brélaz [4] zurückgehende *IDO* (Incidence Degree Ordering) - *Algorithmus* coloriert zweiteilige Graphen auf optimale Weise. Für gewöhnlich ist er jedoch den übrigen Algorithmen nicht überlegen. Darüber hinaus ist er im Vergleich deutlich rechen- und speicheraufwändiger als die hier behandelten Verfahren.

3.4.4 Die Austauschmethode

Alle bisher betrachteten Methoden unterscheiden sich lediglich darin, dass sie unterschiedliche Permutationen der Ecken behandeln, auf die sie den CPR-Algorithmus anwenden. In [53] wird eine vielversprechende Strategie vorgeschlagen, die in bestimmten Fällen in der Lage ist, noch während der Colorierungsphase die überflüssige Einführung zusätzlicher Farben zu vermeiden.
Die Grundidee besteht darin, sobald man im Begriff ist, eine neue Farbe einzuführen, die Farben der bereits colorierten Ecken so zu vertauschen, dass die aktuelle Ecke doch mit einer der bisherigen Farben coloriert werden kann. All die bisher betrachteten Algorithmen können um die Austauschmethode erweitert werden, wodurch selbstverständlich der numerische Aufwand erhöht wird.

Definition 3.4.6. Eine Folge von unterschiedlichen Ecken

$$v_1, v_2..., v_{l+1},$$

so dass für alle $i \in 1, ..., l$ gilt, dass v_i an v_{i+1} angrenzt, heißt *Pfad der Länge l*.

Im Detail lautet das Verfahren wie folgt: Ist die aktuell betrachtete Ecke nicht mit den bereits eingeführten Farben colorierbar, betrachtet man Farbenpaare von bereits existierenden Farben. Zu jedem Farbenpaar betrachtet man zunächst lediglich diejenigen Ecken, die mit den Farben des Farbenpaares coloriert sind und die neu zu colorierende Ecke. Gibt es in diesem nun betrachteten Subgraphen keinen Pfad, der mit beiden Farben an die neu zu colorierende Ecke grenzt, können die Farben vertauscht werden.
In diesem Fall wählt man beliebig eine der beiden Farben des Farbenpaares aus. Der Austausch erfolgt lediglich an denjenigen Pfaden des Subgraphen, die mit der ausgewählten Farbe an die neue Ecke grenzen. Nach dem Tausch kann die neue Ecke mit dieser ausgewählten Farbe coloriert werden.
Lässt sich kein solches Farbenpaar finden, ist kein Austausch möglich und die Einführung der neuen Farbe kann durch die Austauschmethode nicht verhindert werden.

3.5 Graphentheoretischer Ansatz zur Bestimmung zweiter Ableitungen

Die verbleibende zu bestimmende Größe zur Aufstellung des QP-Problems (2.22)-(2.24) ist $\nabla^2_{xx}L(x,\lambda)$. Da, wie in Abschnitt 3.1 erwähnt, die Berechnung von $\nabla^2 f(x)$ unproblematisch ist, benötigen wir lediglich eine effiziente Bestimmung von

$$\nabla^2_{xx}G(x) := \nabla^2_{xx}(L-f)(x) = \sum_{j=1}^{m}\big(\lambda_j \nabla^2 g_j(x)\big).$$

In diesem Abschnitt gehen wir daher davon aus, dass die ersten Ableitungen entweder bereits vorliegen oder gleichzeitig, ebenfalls mittels finiter Differenzen, berechnet werden sollen. Die in Abschnitt 3.5.2 vorgestellte Methode zur effizienten Berechnung der zweiten Ableitung mittels finiter Differenzen ist jedoch nur sinnvoll, wenn auch die ersten Ableitungen mittels finiter Differenzen berechnet werden.

3.5.1 Hessegruppen

In diesem Abschnitt gehen wir davon aus, dass die Berechnung der ersten Ableitungen bereits vollzogen ist. Wie erwähnt, muss dies nicht zwangsläufig mittels finiter Differenzen erfolgt sein. Es gibt jedoch Unterschiede innerhalb der hier vorgestellten Methode, je nachdem ob für die ersten Ableitungen Finte Differenzen verwendet wurden oder nicht. Bei der hier vorgestellten Methode lässt man sich zunächst von dem Gedanken leiten, dass eine Hessematrix vom Prinzip her eine Jacobimatrix einer vektorwertigen Funktion ist. Diese Funktion ist entsprechend der Gradient $\nabla G(x)$ der Ausgangsfunktion.
Der traditionelle Finite-Differenzen-Ansatz zum Bestimmen der j-ten Spalte von $\nabla^2 G(x)$ ist, zusätzlich zum vorliegenden $\nabla G(x)$, den Vektor $\nabla G(x+\varepsilon e_j)$ zu bestimmen und den Vektor

$$\frac{\nabla G(x+\varepsilon e_j) - \nabla G(x)}{\varepsilon}$$

zu berechnen.
Obwohl die Auswertung von $\nabla G(x+\varepsilon e_j)$ zwar auf effiziente Weise erfolgen kann, entweder durch Verwendung der Methode aus Abschnitt 3.3 oder dadurch, dass die Funktion analytisch gegeben ist, ist die n-malige Auswertung des Gradienten jedoch insgesamt zu rechenintensiv. Recheneinsparungen können vor allem dadurch erzielt werden, dass man gemäß Definition 3.3.5 den von der Matrix $\nabla^2 G(x)$ induzierten Graphen $Graph(\nabla^2 G(x))$ betrachtet und, wie in Abschnitt 3.3 erläutert, coloriert. Sei Φ eine daraus entstehende konsistente Partitionierung von $\nabla^2 G(x)$. Wir erinnern daran, dass wenn für zwei Spaltenindizes j_1 und j_2

$$\Phi(j_1) = \Phi(j_2)$$

gilt, so gibt es keinen Zeilenindex i mit

$$(\nabla^2 G)_{ij_1} \not\equiv 0 \wedge (\nabla^2 G)_{ij_2} \not\equiv 0.$$

Gehören folglich j_1 und j_2 zur gleichen Hessegruppe, so kann die j_1-te und die j_2-te Spalte nach Auswertung von

$$\nabla G\left(x+\varepsilon(e_{j_1}+e_{j_2})\right)$$

und der Berechnung des Vektors

$$\frac{\nabla G\left(x+\varepsilon(e_{j_1}+e_{j_2})\right)-\nabla G(x)}{\varepsilon}. \tag{3.15}$$

gleichzeitig berechnet werden. Die Einträge des Vektors (3.15) müssen anschließend entsprechend in die j_1-te und j_2-te Spalte von $\nabla^2 G(x)$ einsortiert werden.
Aufgrund der Symmetrie der Hessematrix ist eine weitere Einsparung möglich. Wegen

$$\frac{(\nabla G(x+\varepsilon e_{j_1}))_{j_2}-(\nabla G(x))_{j_2}}{\varepsilon}\approx\nabla^2 G(x)_{j_1 j_2}=\nabla^2 G(x)_{j_2 j_1}\approx\frac{(\nabla G(x+\varepsilon e_{j_2}))_{j_1}-(\nabla G(x))_{j_1}}{\varepsilon} \tag{3.16}$$

ist eine der beiden Berechnungen überflüssig. Dies kann bei der Partitionierung der Hessematrix ausgenutzt werden.
Angenommen,

$$(\nabla^2 G(x))_{j_1 j_2} \text{ und } (\nabla^2 G(x))_{j_2 j_1}$$

seien Nulleinträge und

$$(\nabla^2 G(x))_{j_1 i}, (\nabla^2 G(x))_{i j_1}, (\nabla^2 G(x))_{j_2 i} \text{ und } (\nabla^2 G(x))_{i j_2}$$

seien Nichtnulleinträge. Aufgrund von Zeile i muss

$$\Phi(j_1)\neq\Phi(j_2)$$

gelten. Nehmen wir an, es sei

$$\Phi(i)<\Phi(j_1)$$

und

$$\Phi(i)<\Phi(j_2).$$

Dann approximieren wir

$$(\nabla^2 G(x))_{i j_1}=(\nabla^2 G(x))_{j_1 i} \approx \frac{\left(\nabla G\left(x+\varepsilon\sum\limits_{l\in\Phi^{-1}(\Phi(i))}e_l\right)\right)_{j_1}-(\nabla G(x))_{j_1}}{\varepsilon}, \tag{3.17}$$

$$(\nabla^2 G(x))_{i j_2}=\nabla^2 G(x))_{j_2 i} \approx \frac{\left(\nabla G\left(x+\varepsilon\sum\limits_{l\in\Phi^{-1}(\Phi(i))}e_l\right)\right)_{j_2}-(\nabla G(x))_{j_2}}{\varepsilon}. \tag{3.18}$$

In den Spalten j_1 und j_2 ist, nachdem die Berechnungen (3.17) und (3.18) durchgeführt wurden, die Zeile i nicht mehr von Interesse, da der zu berechnende Wert bereits vorliegt. Es gilt zwar

$$i\in Rele_{\nabla G}(j_1)$$

und

$$(\nabla g(x+\varepsilon(e_{j_1}+e_{j_2})))_i\neq(\nabla g(x+\varepsilon e_{j_1}))_i,$$

was den ursprünglichen Bedingungen (3.8)-(3.9) zur gemeinsamen Einordnung zweier Variablen widerspricht, für die Partitionierung der Hessematrix kann diese Bedingung jedoch abgeschwächt werden. Dafür definieren wir:

Definition 3.5.1. Die Indexmenge

$$ReleH_{\nabla G}(c,j) := \left\{ k \in \{1,..,n\} :\ \Phi_H(k) \not\leq c \wedge \frac{\partial^2 G(x)}{\partial x_k \partial x_j} \not\equiv 0 \} \right\} \tag{3.19}$$

bezeichnet die für den Index j *zur Zuordnung in die Hessematrix-Partition* c *relevanten Einträge* von ∇G. Die Abbildung

$$\Phi_H :\ \{1,2,...,n\} \to \{1,...,p\}$$

beschreibe eine Partitionierung der Spalten(-indizes) einer $n \times n$-Matrix A. Φ_H ist eine *konsistente Hesse-Partitionierung* wenn

$$\forall i,j \in \{1,...,n\} \text{ mit } \Phi_H(i) = \Phi_H(j) : \quad ReleH_{\nabla G}(\Phi_H(i),i) \cap ReleH_{\nabla G}(\Phi_H(i),j) = \emptyset$$

gilt. Die aus einer konsistenten Hesse-Partitionierung resultierenden Teilmengen

$$\Phi_H^{-1}(c) \subset \{1,...,n\},\ c = 1,...,p$$

bezeichnen wir als *Hessegruppen*.

Bemerkung 3.5.2. Die in (3.19) benutzte Schreibweise „$\not\leq$“ wurde nicht umsonst gewählt. Die Indexmenge $ReleH_{\nabla G}(c,j)$ ist insbesondere während des Partitionierungsprozesses von Bedeutung, so dass es auch Elemente $k \in \{1,...,n\}$ geben mag, für die noch keine Partitionierung vorgenommen wurde. Für diese gilt entsprechend

$$\forall c \in \mathbb{N} :\ \Phi(k) \not\leq c$$

woraus folgt, dass falls

$$\frac{\partial^2 G(x)}{\partial x_k \partial x_j} \not\equiv 0$$

erfüllt ist,

$$\forall c \in \mathbb{N} :\ i \in ReleH_{\nabla G}(c,j)$$

gilt.

Bemerkung 3.5.3. Das (3.8)-(3.9) entsprechende Kriterium, nach dem für zwei Spalten i und j

$$\Phi_H(i) = \Phi_H(j)$$

im Sinne einer konsistenten Hesse-Partitionierung zulässig ist, lautet

$$\begin{aligned} &\forall k \in ReleH_{\nabla G}(\Phi_H(i),i) :\ g_k(x+\varepsilon e_j) = g_k(x), \qquad x \in \mathbb{R}^n \text{ und} \\ &\forall k \in ReleH_{\nabla G}(\Phi_H(i),j) :\ g_k(x+\varepsilon e_i) = g_k(x), \qquad x \in \mathbb{R}^n. \end{aligned}$$

Dies Kriterium ist offensichtlich schwächer als die Bedingungen (3.8)-(3.9), da

$$\forall i \in \{1,...,n\}, \forall c = 1,...,p :\ ReleH_{\nabla G}(c,i) \subset Rele_{\nabla G}(i).$$

Es wird deutlich, dass es bei den Hessegruppen viel mehr auf die betrachtete Reihenfolge der zu partitionierenden Spalten ankommt als dies bei den gewöhnlichen Gruppen der Fall ist. Im Gegensatz zu den gewöhnlichen Gruppen, ist die Reihenfolge bei den Hessegruppen auch mitverantwortlich dafür, ob zwei Variablen für sich betrachtet in die gleiche Gruppe einsortiert werden können. Bei den gewöhnlichen Gruppen konnte eine ungünstige Reihenfolge lediglich dazu führen, dass zwei Spalten, die prinzipiell eine Gruppe teilen könnten und deren gemeinsame Einsortierung einer möglichst kleinen Gruppenanzahl zuträglich wäre, aufgrund von weiteren bereits einsortierten Spalten, in unterschiedliche Gruppen sortiert wurden.
Im Interesse einer geringen Anzahl an Hessegruppen ist es daher erstrebenswert, die Spalten so anzuordnen, dass möglichst viele Spalten, paarweise betrachtet, eine Hessegruppe teilen können. Es erscheint daher sinnvoll, die zu partitionierenden Spalten so anzuordnen, dass die Anzahl der Spaltenelemente abnimmt, was der Reihenfolge des LFO-Algorithmus' aus Abschnitt 3.4.2 gleich kommt.

Beispiel 3.5.4. Die Hessematrix $\nabla^2 G(x)$ habe folgende Struktur:

$$\nabla^2 G(x) = \begin{bmatrix} \times & & & \times \\ & \times & & \times \\ & & \times & \times \\ \times & \times & \times & \times \end{bmatrix}.$$

In der gegebenen Reihenfolge können aufgrund der letzten Zeile keine zwei Spalten in die gleiche Hessegruppe sortiert werden. Daher entstehen vier Hessegruppen. Wird jedoch die vierte Spalte zuerst einer Gruppe zugewiesen, so können die Spalten, $1, 2$ und 3 in eine gemeinsame Hessegruppe eingeordnet werden. Die konsistente Hesse-Partitionierung lautet daher

$$\begin{aligned} \Phi_H(4) &= 1 \\ \Phi_H(1) &= \Phi_H(2) = \Phi_H(3) = 2. \end{aligned}$$

Die Diagonaleinträge $(\nabla^2 G(x))_{ii},\ i = 1, 2, 3$ erfolgen über die Approximation

$$(\nabla^2 G(x))_{ii} \approx \frac{(\nabla G(x + \varepsilon(e_1 + e_2 + e_3)))_i - (\nabla G(x))_i}{\varepsilon}, \quad i = 1, 2, 3$$

während die letzte Spalte und Zeile unter Beachtung der Symmetrieeigenschaft durch

$$(\nabla^2 G(x))_{i4} = (\nabla^2 G(x))_{4i} \approx \frac{(\nabla G(x + \varepsilon e_4))_i - (\nabla G(x))_i}{\varepsilon}, \quad i = 1, 2, 3, 4$$

approximiert wird.

Der Algorithmus der Hessegruppenmethode zur Berechnung der Hessematrix im Falle gegebener, das heißt nicht mittels Finiter Differenzen berechneter erster Ableitungen, lautet folglich:

Algorithmus 3.5.5.

i. Permutiere die Variablen gemäß der LF-Anordnung bezüglich der Hessematrix.

ii. Erstelle gemäß Definition 3.5.1 eine konsistente Hesse-Partitionierung Φ_H. Bezeichne p die maximale Hessegruppe. Setze $c = 1$.

iii. Berechne

$$\nabla G\left(x + \varepsilon \sum_{j \in \Phi_H^{-1}(c)} e_j\right)$$

iv. Berechne für alle $j \in \Phi_H^{-1}(c)$

$$\frac{\partial^2 G(x)}{\partial x_j^2} \approx \frac{\left(\nabla G\left(x + \varepsilon \sum_{l \in \Phi_H^{-1}(c)} e_l\right)\right)_j - (\nabla G(x))_j}{\varepsilon}$$

und für alle Hesseeinträge (i, j) mit $i \neq j$ und $j \in \Phi_H^{-1}(c)$ und $i \in \Phi_H^{-1}(\tilde{c})$, $\tilde{c} > c$

$$\frac{\partial^2 G(x)}{\partial x_i \partial x_j} \approx \frac{\left(\nabla G\left(x + \varepsilon \sum_{l \in \Phi_H^{-1}(c)} e_l\right)\right)_i - (\nabla G(x))_i}{\varepsilon}.$$

v. falls $c = p$ STOP, sonst setze $c = c + 1$ und gehe zu iii.

Im Falle mittels Finiter Differenzen bestimmter erster Ableitungen wird die Idee der Gruppen offensichtlich zweimal angewendet, sowohl für die ersten Ableitungen $\nabla g(x)$ beziehungsweise $\nabla G(x)$ als auch, in Form von Hessegruppen, für die zweite Ableitung $\nabla^2 G(x)$.

Definition 3.5.6. Zwei Indizes $i, j \in \{1, ..., n\}$ mit $i \neq j$ definieren ein *Paar* (i, j) wenn es ein $k \in \{1, ..., m\}$ gibt, so dass

$$\frac{\partial^2 g_k(x)}{\partial x_i \partial x_j} \not\equiv 0. \tag{3.20}$$

In dem Fall nennen wir g_k *relevant für das Paar* (i, j). Aufgrund der zweimaligen, stetigen Differenzierbarkeit ist unsere Hessematrix stets symmetrisch, weswegen Paare ungeordnet sind, das heißt

$$(i, j) = (j, i).$$

Sei

$$hg := \Phi_H^{-1}(c_H)$$

eine Hessegruppe und

$$gr := \Phi^{-1}(c)$$

eine Gruppe. Für ein Paar (i, j) mit

$$i \in \Phi_H^{-1}(c_H)$$

und

$$j \in \Phi^{-1}(c)$$

approximieren wir

$$
\begin{aligned}
(\nabla^2 G(x))_{ij} &\approx \frac{\left(\nabla G\left(x+\varepsilon\sum_{l\in hg} e_l\right)\right)_i - (\nabla G(x))_i}{\varepsilon} \\
&\approx \left(\lambda^T \left[\frac{g\left(x+\varepsilon\left(\sum_{l\in hg} e_l + \sum_{l\in gr} e_l\right)\right) - g\left(x+\varepsilon\sum_{l\in hg} e_l\right)}{\varepsilon^2}\right.\right. \\
&\qquad \left.\left.\frac{-g\left(x+\varepsilon\sum_{l\in gr} e_l\right) + g(x)}{}\right]\right)_i
\end{aligned}
\tag{3.21}
$$

Bewusst wurde hierbei ein Paar (i,j) so definiert, dass $i \neq j$ gilt. Für die Diagonalelemente $i = j$ kann eine höhere Genauigkeit erzielt werden, ohne im Allgemeinen mehr Auswertungen verwenden zu müssen. Die Approximation eines Diagonaleintrages über den zentralen Differenzenquotienten lautet

$$
\frac{\partial^2 G(x)}{\partial x_i^2} \approx \frac{G(x+\varepsilon e_i) - 2G(x) + G(x-\varepsilon e_i)}{\varepsilon} \tag{3.22}
$$

und verlangt unter der Annahme, dass zur Berechnung der ersten Ableitungen die Auswertungen der Form $g(x)$ und $g_k(x+\varepsilon e_i)$ bereits vorliegen, ausschließlich nach Auswertungen der Form $g_k(x-\varepsilon e_i)$. Diese können wiederum mit den (gewöhnlichen) Gruppen auf effiziente Weise bereitgestellt werden.
Da die Berechnung der Diagonaleinträge hiermit abgeschlossen ist, braucht für die letzte Hessegruppe $\Phi_H(p)$ keine Berechnung mehr durchgeführt zu werden, da alle Nichtdiagonaleinträge wegen (3.16) bereits aufgrund der Berechnung anderer Spalten vorliegen und der Diagonaleintrag mittels (3.22) ermittelt wurde.
Es gibt eine weitere Einsparungsmöglichkeit. Neben dem Ignorieren derjenigen Spalten, die zur letzten Hessegruppe gehören, kann es auch Komponenten von g geben, die entweder von keinem Element einer betrachteten Hessegruppe abhängen oder, was noch wahrscheinlicher ist, von keinem Paar abhängen, dessen Eintrag mit der betrachteten Hessegruppe berechnet werden soll. Formal definieren wir:

Definition 3.5.7. Sei $\Phi_H : \{1,...,n\} \to \{1,...,p_H\}$ eine Hesse-Partitionierung von $\nabla^2 G(x)$. Für

$$c_H \in \{1,...,p_H\}$$

bezeichne

$$hg := \Phi_H^{-1}(c_H).$$

Dann ist

$$Relv(hg) := \left\{ l \in \{1,...,n\} : \ \exists \bar{c}_H > c_H \text{ mit } l \in \Phi_H^{-1}(\bar{c}_H) \ \wedge \ \exists \text{ Paar } (j,l),\ j \in hg \right\}$$

die Indexmenge der *relevanten Variablen* der Hessegruppe hg und

$$Rele(hg) := \Big\{ k \in \{1,..,m\} : \ \exists l \in Relv(hg) \text{ und } \exists \text{ Paar } (j,l),\ j \in hg \text{ mit } \frac{\partial^2 g_k(x)}{\partial x_j \partial x_l} \not\equiv 0 \Big\}. \tag{3.23}$$

die Indexmenge der für die Hessegruppe hg *relevanten Einträge* von g. Außerdem bezeichnen wir mit

$$Rele^+(hg) := \Big\{ k \in \{1,..,m\} : \ \exists l \in Relv(hg) \text{ und } \exists \text{ Paar } (j,l),\ j \in hg \text{ mit } \frac{\partial g_k(x)}{\partial x_j} \not\equiv 0 \ \wedge \ \frac{\partial g_k(x)}{\partial x_l} \not\equiv 0 \Big\}.$$

die Indexmenge der für die Hessegruppe hg *vermutlich relevanten Einträge* von g.
Eine Partitionierung, bei der nur die für eine Hessegruppe hg relevanten Variablen und Einträge berücksichtigt werden, bezeichnen wir als *Subpartitionierung*. Das Ergebnis einer solchen Subpartitionierung wird *Untergruppe* von hg genannt.

Bemerkung 3.5.8. Man beachte, dass

$$Rele(hg) \subset Rele^+(hg)$$

gilt. Die Umkehrung (und damit die Gleichheit) gilt nicht, denn für $k \in Rele^+(hg)$ ist die Komponente g_k zwar von den beiden Variablen x_j und x_l aus (3.23) abhängig, aber deswegen nicht automatisch für das Paar (j,l) gemäß Definition 3.5.6 relevant. Eine in x_j und x_l lineare Funktion erfüllt nämlich keinesfalls (3.20).
Zumeist liegen jedoch so detaillierte Informationen wie Linearitäten einzelner Variablen in den einzelnen Funktionen nicht vor, so dass statt der Indexmenge $Rele(hg)$ häufig auf die Indexmenge $Rele^+(hg)$ zurückgegriffen werden muss. Deren Verwendung führt jedoch schlimmstenfalls dazu, dass ein Nulleintrag durch Finite Differenzen unnötigerweise und mit kleinen Rundungsfehlern behaftet approximiert wird und nicht zu einer vollkommen fehlerhaften Approximation.

Um die Begriffe zu illustrieren, betrachten wir das

Beispiel 3.5.9. Die Hessematrix $\nabla^2 G(x)$ habe die folgende Struktur

$$\nabla^2 G(x) = \begin{bmatrix} \times & \times & & \times \\ \times & \times & \times & \\ & \times & \times & \\ \times & & & \times \end{bmatrix}$$

während die Jacobimatrix die Struktur

$$\nabla g(x) = \begin{bmatrix} \times & \times & \times & \\ \times & & & \times \\ \times & \times & \times & \times \end{bmatrix}$$

habe. g_3 sei gegeben durch

$$g_3(x) = x_1 + x_2 + x_3 + x_4. \tag{3.24}$$

Eine konsistente Hesse-Partitionierung lautet

$$\begin{aligned} \Phi_H(1) &= 1, \\ \Phi_H(2) &= \Phi_H(4) = 2, \\ \Phi_H(3) &= 3. \end{aligned}$$

Die entstehenden Hessegruppen bezeichnen wir mit

$$hg_c := \Phi_H^{-1}(c), \quad c = 1, 2, 3$$

Die Berechnung von

$$\left(\nabla G\left(x + \varepsilon \sum_{j \in hg_1}\right)\right)_3 = (\nabla G(x + \varepsilon e_1))_3 = \lambda^T \frac{\partial g(x + \varepsilon e_1)}{\partial x_3}$$

ist unnötig, da in der Matrix $\nabla^2 G(x)$ in der ersten Spalte (das heißt in allen Spalten der Hessegruppe hg_1) kein Element in der 3. Zeile steht. Ebenso unnötig, wenn auch aus einem anderen Grunde, ist die Berechnung von

$$\left(\nabla G\left(x + \varepsilon \sum_{j \in hg_3}\right)\right)_1.$$

Hier existiert zwar ein Eintrag in der ersten Zeile bei einer Spalte eines Hessegruppenelement (nämlich in der vierten Spalte). Da jedoch die Hessegruppe hg_1 vor hg_3 bearbeitet wird, liegt der approximierte Wert für

$$\frac{\partial^2 G(x)}{\partial x_1 \partial x_4} = \frac{\partial^2 G(x)}{\partial x_4 \partial x_1}$$

aufgrund der Symmetrie bereits vor.
Die in Abschnitt 3.3 eingeführte konsistente Partitionierung lautet hier trivialerweise:

$$\Phi(i) = i, \quad i = 1, 2, 3, 4.$$

Analog zu den Hessegruppen hg_c bezeichnen wir

$$gr_c := \Phi^{-1}(c), \ c = 1, 2, 3, 4.$$

Für jede Auswertung der ersten Ableitung bedarf es vierer Auswertungen von g. Betrachtet man jedoch lediglich die relevanten Variablen und Einträge, kann zur Berechnung von

$$\nabla G\left(x + \varepsilon \sum_{j \in hg_1}\right)$$

diese Anzahl reduziert werden.
Es genügt, die Untergruppen ("subgroups") zu den jeweiligen Hessegruppen aufzustellen. Dafür benötigen wir die relevanten Variablen der jeweiligen Hessegruppen

$$\begin{aligned} Relv(hg_1) &= \{2, 4\}, \\ Relv(hg_2) &= \{3\}, \\ Relv(hg_3) &= \emptyset, \end{aligned}$$

sowie die relevanten Einträge

$$\begin{aligned} Rele(hg_1) &= \{1,2\}, \\ Rele(hg_2) &= \{1\}, \\ Rele(hg_3) &= \emptyset. \end{aligned}$$

Im Falle, dass uns die detaillierte Gestalt von g_3 aus (3.24) nicht bekannt ist, müssen wir uns mit den vermutlich relevanten Einträgen begnügen, das heißt

$$\begin{aligned} Rele^+(hg_1) &= \{1,2,3\} \\ Rele^+(hg_2) &= \{1,3\} \\ Rele^+(hg_3) &= \emptyset. \end{aligned}$$

Bei Verwendung von $Rele(hg_1)$ erhalten wir die Subpartitionierung zu hg_1

$$\begin{aligned} \Phi_{sub,hg_1}(1) &= 1, \\ \Phi_{sub,hg_1}(2) &= \Phi_{sub,hg_1}(4) = 2. \end{aligned}$$

Unter Verwendung von $Rele^+(hg_1)$ müssen wir eine Auswertung mehr hinnehmen:

$$\begin{aligned} \Phi_{sub,hg_1}(1) &= 1, \\ \Phi_{sub,hg_1}(2) &= 2 \\ \Phi_{sub,hg_1}(4) &= 3. \end{aligned}$$

Wieder bezeichnen wir, analog zu hg_c und gr_c

$$sg_c(hg_{\tilde{c}}) = \Phi^{-1}_{sub,hg_{\tilde{c}}}(c), \quad c = 1, ..., p_{sub}(hg_{\tilde{c}}),$$

wobei $p_{sub}(hg_{\tilde{c}})$ die Anzahl der Untergruppen zu einer Hessegruppe $hg_{\tilde{c}}$ bezeichne.
Nach Aufstellung dieser Untergruppen liefert die Berechnung

$$\frac{g\left(x+\varepsilon\left(\sum\limits_{l\in hg_1} e_l + \sum\limits_{l\in sg_c(hg_1)} e_l\right)\right) - g\left(x+\varepsilon\sum\limits_{l\in hg_1} e_l\right)}{\varepsilon}, \quad c = 1,2 \text{ bzw. } c = 1,2,3$$

in den jeweiligen relevanten Spalten eine geeignete Approximation von

$$\frac{\partial g\left(x+\varepsilon\sum\limits_{l\in hg_1} e_l\right)}{\partial x_j}, \quad j \in \bigcup_{c=1}^{p_{sub}(hg_1)},$$

mit

$$p_{sub}(hg_1) = 2 \text{ bzw. } 3.$$

Somit erhalten wir wegen

$$\frac{\partial G\left(x+\varepsilon\sum\limits_{l\in hg_1} e_l\right)}{\partial x_j} = \lambda^T \frac{\partial g\left(x+\varepsilon\sum\limits_{l\in hg_1} e_l\right)}{\partial x_j}, \quad j \in \bigcup_{c=1}^{p_{sub}(hg_1)}$$

die relevanten Einträge von

$$\nabla G\left(x+\varepsilon\sum_{l\in hg_1}e_l\right).$$

Die Spalten, deren Indizes zur Hessegruppe hg_1 gehören (hier nur die erste Spalte), werden somit wie folgt approximiert:

$$\frac{\partial^2 G(x)}{\partial x_j \partial x_1} \approx \frac{g\left(x+\varepsilon\left(\sum_{l\in hg_1}e_l+\sum_{l\in sg_c(hg_1)}e_l\right)\right)-g\left(x+\varepsilon\sum_{l\in hg_1}e_l\right)-g\left(x+\varepsilon\sum_{l\in gr_d}e_l\right)+g(x)}{\varepsilon^2}, \tag{3.25}$$

wobei $j\in sg_c(hg_1)$ und $j\in gr_d$ gelte.

An diesem Beispiel ist ersichtlich, dass sich die Anzahl der Untergruppen von der Anzahl der Gruppen unterscheiden kann. Bei hochdimensionalen Problemen kann dieser Unterschied sehr deutlich sein.
Unter Umständen kann bei einer Hessegruppe hg bei der Berechnung von (3.25) sogar auf die Auswertung von

$$g\left(x+\varepsilon\sum_{l\in hg}e_l\right)$$

verzichtet werden. Das ist dann der Fall, wenn jeder für hg relevante Eintrag g_k nur von einem Element von hg abhängt, das heißt falls

$$\forall k\in Rele(hg)\ \exists! j_k\in hg:\ g_k\left(x+\varepsilon\sum_{l\in hg}e_l\right)=g_k(x+\varepsilon e_{j_k}). \tag{3.26}$$

Man betrachte in diesen Fällen für jeden relevanten Eintrag g_k diejenige (gewöhnliche) Gruppe

$$gr(k):=\Phi^{-1}(\Phi(j_k)),$$

die den in (3.26) erklärten Index j_k enthält. Da für diese, von dieser Gruppe $gr(k)$ induzierte, Auswertung

$$g_k\left(x+\varepsilon\sum_{l\in gr(k)}e_l\right)=g_k(x+\varepsilon e_{j_k})$$

gilt, enthält sie den relevanten Wert, weswegen eine weitere Auswertung offensichtlich überflüssig ist. Das ist im Beispiel 3.5.9 mit der Hessegruppe hg_1 der Fall. Hier entspricht

$$g\left(x+\varepsilon\sum_{l\in hg_1}e_l\right)=g(x+\varepsilon e_1)=g\left(x+\varepsilon\sum_{l\in gr_1}e_l\right).$$

Zusammenfassend kann aufgrund von (3.22) und (3.25) festgehalten werden, dass für die Bestimmung von $\nabla G(x)$ und $\nabla^2 G(x)$ bei einer Partitionierung mit p Gruppen, und einer Hessepartitionierung mit p_H Hessegruppen

$$2p+(p_H-1)+\sum_{c=1}^{p_H-1}p_{sub,hg_c}+1=2p+p_H+\sum_{c=1}^{p_H-1}p_{sub,hg_c},$$

Auswertungen nötig sind, wobei p_{sub,hg_c} die Anzahl der zu hg_c gehörenden Untergruppen bezeichne.

3.5.2 Paargruppen

Während die Strategie der Hessegruppen sowohl mit durch Finite Differenzen bestimmten ersten Ableitungen durchführbar ist, als auch mit auf anderem Wege gegebenen, muss für den in diesem Abschnitt behandelten Ansatz davon ausgegangen werden, dass die Berechnung von ∇g beziehungsweise ∇G über Finite Differenzen erfolgt und die dafür notwendigen Auswertungen bereits vorliegen. Wieder sei dafür die in Abschnitt 3.3 vorgestellte Gruppenmethode verwendet worden.
Für die zweiten Ableitungen werden die Diagonalelemente auch bei der hier vorgestellten Paargruppenstrategie durch (3.22) berechnet, da mit dem zentralen Differenzenquotienten eine höhere Genauigkeit erzielt werden kann und die Auswertungen der Form

$$g_k(x - \varepsilon e_i)$$

aufgrund der Gruppen auf sehr effiziente Weise erfolgt.
Nun gilt es, die für die Berechnung der Nichtnulleinträge

$$(\nabla^2 G(x))_{i,j},\ i \neq j$$

notwendigen Auswertungen zu reduzieren. Die Approximation von $(\nabla^2 G(x))_{i,j}$ über den Vorwärtsdifferenzenquotienten lautet

$$(\nabla^2 G(x))_{i,j} \approx \frac{G(x + \varepsilon(e_i + e_j)) - G(x + \varepsilon e_i) - G(x + \varepsilon e_j) + G(x)}{\varepsilon^2}, \quad i \neq j. \tag{3.27}$$

Aufgrund der Annahme, dass die ersten Ableitungen bereits per Finite Differenzen bestimmt wurden, liegen die für das Paar (i, j) relevanten Einträge

$$g_k(x + \varepsilon e_i)$$

bereits vor. Um nicht für jeden Nichtdiagonaleintrag von $\nabla^2 G(x)$ den ganzen Vektor

$$g(x + \varepsilon(e_i + e_j))$$

auszuwerten, führen wir eine so genannte *Paar-Partitionierung* durch und verwenden die daraus resultierenden *Paargruppen*. In gleicher Weise wie in Abschnitt 3.3 mehrere Variablen in eine Gruppe eingeordnet wurden, werden hier die Paare in Gruppen aufgeteilt.

Definition 3.5.10. Bezeichne

$$Pairs \subset \{1, ..., n\} \times \{1, ..., n\}$$

die Menge der Paare.
Sei $(i, j) \in Pairs$ ein Paar. Dann bezeichnet

$$Rele((i, j)) := \left\{ k \in \{1, .., m\} :\ \frac{\partial^2 g_k(x)}{\partial x_i \partial x_j} \not\equiv 0 \right\}.$$

die Indexmenge der für das Paar (i,j) *relevanten Einträge* von g und

$$Rele^+((i,j)) := \left\{ k \in \{1,..,m\} : \ \frac{\partial g_k(x)}{\partial x_i} \not\equiv 0 \ \wedge \ \frac{\partial g_k(x)}{\partial x_j} \not\equiv 0 \right\}.$$

die Indexmenge der für das Paar (i,j) *vermutlich relevanten Einträge* von g.
Die Funktion $\Phi_P : Pairs \to \{1,...,p_P\}$ beschreibt eine *Paar-Partitionierung*. Bezeichne

$$pg_c := \Phi_P^{-1}(c)$$

eine *Paargruppe* und

$$I_{pg_c} := \{l \in \{1,...,n\} : \ \exists i \in \{1,...,n\} \text{ mit } (i,j) \in pg_c\}$$

die Indexmenge der Paarelemente von pg_c. Φ_P ist eine *konsistente Paar-Partitionierung* falls

$$\forall c \in \{1,...,p_P\} \ \forall (i,j) \text{ mit } \Phi_P((i,j)) = c \ \forall k \in Rele((i,j)) :$$
$$g_k(x + \varepsilon(e_i + e_j)) = g_k\left(x + \varepsilon\left(\sum_{l \in I_{pg}} e_l\right)\right).$$

Bemerkung 3.5.11. Die Einführung der Indexmenge $Rele^+((i,j))$ ist aus dem gleichem Grund notwendig wie in Bemerkung 3.5.8 über die Einführung von $Rele^+(hg)$ dargelegt.

Beispiel 3.5.12. Die Struktur der Hessematrix von G sei

$$\nabla^2 G(x) = \begin{bmatrix} \times & \times & & \\ \times & \times & & \\ & & \times & \times \\ & & \times & \times \end{bmatrix}$$

und die der Jacobimatrix der Nebenbedingungen sei

$$\frac{dg(x)}{dx} = \begin{bmatrix} \times & \times & & \\ & & \times & \times \end{bmatrix}.$$

Wie man leicht sieht, können die Auswertungen von g für die beiden Paare $(1,2)$ und $(3,4)$ gleichzeitig durchgeführt werden. Die Auswertung von

$$g\left(x + \varepsilon \sum_{i=1}^{4} e_i\right)$$

ist in diesem Beispiel ausreichend und keine weitere Auswertung der Form

$$g(x + \varepsilon(e_i + e_j))$$

ist von Nöten.

Ähnlich wie bei der Gruppen- und Hessegruppenmethode ist auch bei den Paargruppen das Ziel, die Anzahl der Paargruppen so klein wie möglich zu halten. Für die Nichtdiagonaleinträge entspricht diese Zahl gleichzeitig der Anzahl der zusätzlich nötigen Funktionsauswertung zur Berechnung der zweiten Ableitung $\nabla^2 G(x)$, sofern, wie angenommen, die Auswertungen, die zu Berechnung der ersten Ableitungen notwendig sind, bereits vorliegen. Die Bestimmung der Paargruppen ist deutlich komplizierter und rechenaufwändiger als die Bestimmung der Gruppen. Dies liegt zum Einen daran, dass die Anzahl der Paare in der Regel deutlich höher ist als die Anzahl der Variablen. Des Weiteren ist das Kriterium, das entscheidet, ob zwei Paare in die gleiche Paargruppe eingeordnet werden dürfen, deutlich komplizierter als das Kriterium für die Gruppierung zweier Variablen.

Bemerkung 3.5.13.

i. Sei

$$(i,j) \in pg$$

ein Paar in einer Paargruppe pg. Sei s mit

$$i \neq s \neq j$$

ein weiterer Index aus $\{1,...,n\}$. Es gilt

$$\exists k \in Rele((i,j))\ \exists s \in \{1,...,n\} \setminus \{i,j\} : \ \frac{\partial g_k(x)}{\partial x_s} \not\equiv 0 \implies \forall t \in \{1,...,n\} : \ (s,t) \notin pg.$$

Mit anderen Worten: Bereits wenn nur eine Variable eines Paares Einfluss auf einen relevanten Eintrag eines anderen Paares hat, können die beiden Paare nicht in die gleiche Paargruppe einsortiert werden. Es genügt nicht, dass die Mengen der relevanten Einträge der beiden Paare disjunkt sind.

ii. Zwei Paare (i,j) und (j,s) mit einem gemeinsamen Element j dürfen nicht in die gleiche Paargruppe eingeordnet werden, wenn

$$\exists k \in Rele((j,s)) : \quad \frac{\partial g_k(x)}{\partial x_i} \not\equiv 0 \ \vee \ \exists k \in Rele((i,j)) : \quad \frac{\partial g_k(x)}{\partial x_s} \not\equiv 0. \tag{3.28}$$

Auch dies ist nicht gleichbedeutend damit, dass die Mengen der relevanten Einträge der beiden Paare disjunkt sind. Aus (3.28) folgt zwar, dass für das betreffende k

$$\begin{aligned} \frac{\partial g_k(x)}{\partial x_i} &\not\equiv 0, \\ \frac{\partial g_k(x)}{\partial x_j} &\not\equiv 0, \\ \frac{\partial g_k(x)}{\partial x_s} &\not\equiv 0 \end{aligned}$$

gilt. Daraus folgt aber nicht zwangsläufig, dass

$$k \in Rele((i,j)) \cap Rele((j,s)).$$

Zur Illustration betrachten wir zwei Beispiele.

Beispiel 3.5.14. Die Strukturen von $\nabla^2 G$ und ∇g seien wie folgt:

$$\begin{aligned} \nabla^2 G(x) &= \begin{bmatrix} \times & \times & & \\ \times & \times & & \\ & & \times & \times \\ & & \times & \times \end{bmatrix} \\ \nabla g(x) &= \begin{bmatrix} \times & \times & \times & \\ & & \times & \times \end{bmatrix} \end{aligned}$$

Hier gibt es die zwei Paare $(1,2)$ und $(3,4)$. Diese Paare können nicht in eine Paargruppe eingeordnet werden, da ein Element des einen Paares, nämlich x_3, Einfluss hat auf den Eintrag g_1, welcher für das andere Paar $(1,2)$ relevant ist. Dieser Sachverhalt gilt selbst dann, wenn die Abhängigkeit der Funktion g_1 von der Variable x_3 lediglich linear ist, denn in jedem Fall gilt

$$g_1(x + \varepsilon(e_1 + e_2 + e_3)) \neq g_1(x + \varepsilon(e_1 + e_2)).$$

Beispiel 3.5.15. Die Struktur von $\nabla^2 G$ und ∇g seien wie folgt:

$$\begin{aligned} \nabla^2 G(x) &= \begin{bmatrix} \times & \times & \\ \times & \times & \times \\ & \times & \times \end{bmatrix} \\ \nabla g(x) &= \begin{bmatrix} \times & \times & \\ & \times & \times \end{bmatrix} \end{aligned}$$

Hier gibt es die zwei Paare $(1,2)$ und $(2,3)$, welche eine Paargruppe bilden dürfen. x_1 hat keinen Einfluss auf g_2, welches für das Paar $(2,3)$ relevant ist. Gleiches gilt andersrum für x_3 und Element g_1.

Angesichts dessen wird klar, dass es wahrscheinlicher ist, dass zwei Paare in eine Paargruppe eingeordnet werden können, wenn sie ein gemeinsames Element haben.
Im Folgenden werden wir nachweisen, dass die Hessegruppenmethode nicht mit der Paargruppenmethode übereinstimmt. Dazu betrachten wir das

Beispiel 3.5.16. Es sei

$$\nabla^2 G(x) = \begin{bmatrix} \times & \times & & \\ \times & \times & & \\ & & \times & \times \\ & & \times & \times \end{bmatrix}.$$

Die zugehörige Jacobi-Matrix sehe wie folgt aus

$$\nabla g(x) = (\times \quad \times \quad \times \quad \times).$$

Die gewöhnlichen Gruppen, die ohnehin zur Bestimmung von $\nabla g(x)$ bestimmt werden, sind in diesem Fall

$$gr_i = \{i\}, \quad i = 1, 2, 3, 4.$$

Daher können wir davon ausgehen, dass die Auswertungen der Form

$$g(x + \varepsilon e_i), \quad i = 1, 2, 3, 4$$

bereits vorliegen.
Die Hessegruppen hingegen lauten

$$\begin{aligned} hg_1 &= \{1, 3\}, \\ hg_2 &= \{2, 4\}. \end{aligned}$$

Für die Indexmenge der relevanten Variablen von hg_1 gilt

$$Relv(hg_1) := \{2, 4\}.$$

Der einzige Eintrag von g ist natürlich relevant. Die relevanten Variablen können nicht in eine Untergruppe einsortiert werden und dadurch sind drei Auswertungen notwendig, nämlich

$$g(x + \varepsilon(e_1 + e_3)),\ g(x + \varepsilon(e_1 + e_3 + e_2)) \text{ und } g(x + \varepsilon(e_1 + e_3 + e_4)).$$

Mit der Hessegruppenstrategie können die Nichtdiagonaleinträge von $\nabla^2 G(x)$ wie folgt berechnet werden

$$\begin{aligned} \nabla^2 G(x)_{1,2} &= \lambda^T \frac{g(x + (e_1 + e_3 + e_2)) - g(x + \varepsilon(e_1 + e_3)) - g(x + \varepsilon e_2) + g(x)}{\varepsilon^2}, \\ \nabla^2 G(x)_{3,4} &= \lambda^T \frac{g(x + (e_1 + e_3 + e_4)) - g(x + \varepsilon(e_1 + e_3)) - g(x + \varepsilon e_4) + g(x)}{\varepsilon^2}. \end{aligned}$$

Für die Paargruppenstrategie stellen wir folgende Überlegung an:
Es gibt hier die Paare $(1, 2)$ und $(3, 4)$, welche nicht in eine gemeinsame Paargruppe eingeordnet werden dürfen. Daher wertet man mit der Paargruppenstrategie

$$g(x + \varepsilon(e_1 + e_2))$$

und

$$g(x + \varepsilon(e_3 + e_4))$$

aus und kann beide Nichtdiagonaleinträge von $\nabla^2 G(x)$ durch

$$\begin{aligned} (\nabla^2 G(x))_{1,2} &= \lambda^T \frac{g(x + \varepsilon(e_1 + e_2)) - g(x + \varepsilon e_1) - g(x + \varepsilon e_2) + g(x)}{\varepsilon^2}, \\ (\nabla^2 G(x))_{3,4} &= \lambda^T \frac{g(x + \varepsilon(e_3 + e_4)) - g(x + \varepsilon e_3) - g(x + \varepsilon e_4) + g(x)}{\varepsilon^2} \end{aligned}$$

bestimmen.
In diesem einfachen Beispiel, hat sich die Paargruppenstrategie als vorteilhaft erwiesen.

Den gegenteiligen Fall zeigt das folgende

Beispiel 3.5.17. Seien

$$\nabla^2 G(x) = \begin{bmatrix} \times & \times & & & & \\ \times & \times & & & & \\ & & \times & \times & & \\ & & \times & \times & & \\ & & & & \times & \times \\ & & & & \times & \times \end{bmatrix}$$

und

$$\nabla g(x) = \begin{bmatrix} \times & \times & \times & & \times & \\ & & \times & \times & \times & \\ & & & & \times & \times \end{bmatrix}.$$

Die gewöhnlichen Gruppen lassen sich wie folgt aufstellen:

$$\begin{aligned} gr_1 &= \{1, 4\} \\ gr_2 &= \{2, 6\} \\ gr_3 &= \{3\} \\ gr_4 &= \{5\} \end{aligned}$$

Die Hessegruppen sind (beispielsweise)

$$\begin{aligned} hg_1 &= \{1, 3, 5\} \\ hg_2 &= \{2, 4, 6\}. \end{aligned}$$

Die Indexmengen der relevanten Variablen beziehungsweise Einträgen lauten

$$Relv(hg_1) = \{2, 4, 6\}$$

beziehungsweise

$$Rele(hg_1) = \{1, 2, 3\}$$

Da die relevanten Variablen nur auf unterschiedliche Einträge von g einen Einfluss haben, gibt es nur eine Untergruppe. nämlich

$$sgr_1(hg_1) = \{2, 4, 6\}.$$

Die notwendigen Auswertungen sind daher

$$g\left(x + \varepsilon \sum_{i=1}^{6} e_i\right)$$

und

$$g(x + \varepsilon(e_1 + e_3 + e_5)).$$

Die Approximation der Nichtdiagonaleinträge lautet wie folgt:

$$
\begin{aligned}
(\nabla^2 G(x))_{1,2} &= \lambda_1 \frac{g_1\left(x+\varepsilon\sum_{i=1}^{6} e_i\right) - g_1(x+\varepsilon(e_1+e_3+e_5)) - g_1(x+\varepsilon(e_2+e_6)) + g_1(x)}{\varepsilon^2},\\
(\nabla^2 G(x))_{3,4} &= \lambda_2 \frac{g_2\left(x+\varepsilon\sum_{i=1}^{6} e_i\right) - g_2(x+\varepsilon(e_1+e_3+e_5)) - g_2(x+\varepsilon(e_1+e_4)) + g_2(x)}{\varepsilon^2},\\
(\nabla^2 G(x))_{5,6} &= \lambda_3 \frac{g_3\left(x+\varepsilon\sum_{i=1}^{6} e_i\right) - g_3(x+\varepsilon(e_1+e_3+e_5)) - g_3(x+\varepsilon(e_2+e_6)) + g_3(x)}{\varepsilon^2}.
\end{aligned}
$$

Für die Paargruppenstrategie gilt: Die drei Paare bedürfen alle unterschiedlicher Paargruppen. Daher bedarf es dreier Auswertungen, um die Approximationen

$$
\begin{aligned}
(\nabla^2 G(x))_{1,2} &= \lambda_1 \frac{g_1(x+\varepsilon(e_1+e_2)) - g_1(x+\varepsilon(e_1+e_4)) - g_1(x+\varepsilon(e_2+e_6)) + g_1(x)}{\varepsilon^2},\\
(\nabla^2 G(x))_{3,4} &= \lambda_2 \frac{g_2(x+\varepsilon(e_3+e_4)) - g_2(x+\varepsilon e_3) - g_2(x+\varepsilon(e_1+e_4)) + g_3(x)}{\varepsilon^2},\\
(\nabla^2 G(x))_{5,6} &= \lambda_3 \frac{g_3(x+\varepsilon(e_5+e_6)) - g_3(x+\varepsilon e_5) - g_3(x+\varepsilon(e_2+e_6)) + g_3(x)}{\varepsilon^2},
\end{aligned}
$$

durchzuführen.
In diesem Fall ist die Hessegruppenmethode überlegen.

In Abschnitt 5.2 wird untersucht, ob eine der beiden Strategien bei Betrachtung einer Vielzahl von Beispielen überlegen ist.

3.6 Fallbeispiel: Planarer Niedrigschubtransfer zu einem geosynchronen Orbit

Da die Vorteile der verschiedenen Gruppenmethoden erst bei großen, dünnbesetzten Problemen sichtbar werden, wird in diesem Abschnitt ein solches Problem vorgestellt und die verschiedenen Verfahren exemplarisch darauf angewandt. Große Optimierungsprobleme resultieren häufig aus Optimalsteuerungsproblemen, siehe dazu Büskens et al. [2], [8], [11], [12], [14], [15], [16], [17], [57] und [75] und Gerdts et al. [36], [37], [38]. Das Ziel bei dem hier betrachteten Problem ist es, die Steuerung der Schubrichtung

$$u(t), \quad 0 \leq t \leq t_f,$$

zu bestimmen, die die Endzeit minimiert. Formal lautet es wie folgt.

Problem 3.6.1.

$$\begin{aligned}
\min_{x,u,t_f} \tilde{F}(x,u,t_f) &:= t_f \\
\textit{unter } \dot{x}(t) &= \tilde{f}(x,u,t), \\
x_1(0) &= 6.0, \\
x_1(t_f) &= 6.6, \\
x_2(0) &= 0.0, \\
x_2(t_f) &= 0.0, \\
x_3(0) &= \sqrt{\frac{r_\mu}{x_1(0)}}, \\
x_4(0) &= 0.0
\end{aligned}$$

mit

$$\tilde{f}(x,u,t) := \begin{bmatrix} x_2(t) \\ \frac{x_3^2}{x_1} - \frac{r_\mu}{x_1^2} + 0.01\sin(u) \\ -\frac{x_2 x_3}{x_1} + 0.01\cos(u) \\ \frac{x_3}{x_1} \end{bmatrix}.$$

Die Variablen $x(t)$ beschreiben die Zustände des dynamischen Systems. $x_1(t)$ ist die radiale Position, $x_2(t)$ die radiale Geschwindigkeit, $x_3(t)$ die Umfangsgeschwindigkeit und $x_4(t)$ der Polarwinkel. Der Gravitationsparameter für die Erde wird durch $r_\mu = 62.5$ beschrieben. Dieses Problem stammt aus [55].
Um aus dem Optimalsteuerungsproblem 3.6.1 ein Optimierungsproblem zu machen, müssen wir die Zeit diskretisieren. N bezeichne die Anzahl der Intervalle. Die Zeitpunkte sind daher gegeben durch

$$0 = t_0 < t_1 < ... < t_N = t_f.$$

Die Zeitpunkte seien äquidistant, so dass

$$\forall i \in \{0, ..., N-1\}: \quad t_{i+1} - t_i = h,$$

gilt mit

$$h := \frac{t_f}{N}.$$

Das resultierende Optimierungsproblem hat $N \cdot 4 + (N-1) + 1$ Optimierungsvariablen:

$$\begin{aligned} x_{1,i} &:= x_1(t_i), \\ x_{2,i} &:= x_2(t_i), \\ x_{3,i} &:= x_3(t_i), \\ x_{4,i} &:= x_4(t_i), \quad i = 0, ..., N, \\ x_{5,i} &:= u(t_i), \quad i = 0, ..., N-1 \end{aligned}$$

und t_N.
$u(t_f)$ hat keinen Einfluss auf das dynamische System und wird daher nicht als Variable berücksichtigt. Aufgrund der zum Teil fest gesetzten Start- und Endwerte für die Zustände ließe sich die Anzahl um einige wenige Variablen reduzieren, wovon hier jedoch abgesehen wird.
Zur Diskretisierung der Bedingung

$$\dot{x}(t) = f(x, u, t) \tag{3.29}$$

gibt es eine Vielzahl von Ansätzen. Der einfachste ist das so genannte explizite Euler-Verfahren. Aufgrund der Taylornäherung

$$x(t_{i+1}) - x(t_i) \approx h\dot{x}(t_i)$$

ergibt sich die Approximation

$$x_{j,i+1} = x_{j,i} + hf_j(x_i, t_i), \quad j = 1, 2, 3, 4,$$

wobei

$$x_i := \begin{bmatrix} x_{1,i} \\ x_{2,i} \\ x_{3,i} \\ x_{4,i} \\ x_{5,i} \end{bmatrix}$$

den Vektor der Optimierungsvariablen zum Zeitpunkt i bezeichne.
Hieraus ergeben sich die Nebenbedingungen:

$$x_{j,i+1} - x_{j,i} - hf_j(x_i, t_i) = 0, \quad j = 1, 2, 3, 4, \quad i = 0, ..., N-1. \tag{3.30}$$

Das resultierende Optimierungsproblem lautet damit:

Problem 3.6.2.

$$
\begin{aligned}
\min_x f(x) &:= t_N \\
\textit{unter} \quad & (3.30), \\
x_1(0) &= 6.0, && (3.31) \\
x_1(t_f) &= 6.6, && (3.32) \\
x_2(0) &= 0.0, && (3.33) \\
x_2(t_f) &= 0.0, && (3.34) \\
x_3(0) &= \sqrt{\frac{r_\mu}{x_1(0)}}, && (3.35) \\
x_4(0) &= 0.0 && (3.36)
\end{aligned}
$$

Die Struktur der Jacobimatrix ist wie folgt:

$$
\nabla g(x) = \left[\begin{array}{cccccccccccccccccccccccc}
\times & \times & & & & \times & & & & & & & & & & & & & & & & & & \times \\
\times & \times & \times & & \times & & \times & & & & & & & & & & & & & & & & & \times \\
\times & \times & \times & & \times & & & \times & & & & & & & & & & & & & & & & \times \\
\times & & \times & \times & & & & & \times & & & & & & & & & & & & & & & \times \\
& & & & & \times & \times & & & & \times & & & & & & & & & & & & & \times \\
& & & & & \times & \times & \times & & \times & & \times & & & & & & & & & & & & \times \\
& & & & & \times & \times & \times & & \times & & & \times & & & & & & & & & & & \times \\
& & & & & \times & & \times & \times & & & & & \times & & & & & & & & & & \times \\
& & & & & & & & & & & & & & \cdot & & & & & & & & & \times \\
& & & & & & & & & & & & & & & \cdot & & & & & & & & \times \\
& & & & & & & & & & & & & & & & \cdot & & & & & & & \times \\
& & & & & & & & & & & & & & \times & \times & & & & \times & & & & \times \\
& & & & & & & & & & & & & & \times & \times & \times & & \times & & \times & & & \times \\
& & & & & & & & & & & & & & \times & \times & \times & & \times & & & \times & & \times \\
& & & & & & & & & & & & & & \times & & \times & \times & & & & & \times & \times
\end{array}\right].
$$

Die Anordnung ist hierbei chronologisch, d.h.

$$
x_{10}, x_{20}, x_{30}, x_{40}, u_0, x_{11}, x_{21}, x_{31}, x_{41}, u_1, \ldots, x_{1N}, x_{2N}, x_{3N}, x_{4N}, t_N
$$

Die Box-Beschränkungen (3.31)-(3.36) werden nicht als Nebenbedingungen aufgefasst. Im Übrigen entstünden andernfalls (zumindest nach Einsetzten von $x_1(0)$ in (3.35)) lediglich weitere Zeilen mit nur einem Element. Solche Zeilen sind für die hier betrachteten Verfahren ohnehin unerheblich.
Da h in jeder der Bedingungen (3.30) vorkommt, ist t_N in jeder Zeile von ∇g vorhanden.
Die chromatische Zahl des zu $\nabla g(x)$ gehörenden Graphen ist 7. Bei der gegebenen Anordnung wird eine optimale Colorierung bereits durch den CPR-Algorithmus erreicht. Wir betrachten immer fünf zu einem Zeitpunkt gehörende Spalten. Der CPR-Algorithmus coloriert die ersten fünf Spalten, so dass

$$
\begin{aligned}
\Phi(1) &= 1, \\
\Phi(2) &= 2, \\
\Phi(3) &= 3, \\
\Phi(4) &= 2, \\
\Phi(5) &= 4.
\end{aligned}
$$

Für den zweiten Zeitpunkt gilt:

$$\begin{aligned}\Phi(6) &= 3,\\ \Phi(7) &= 5,\\ \Phi(8) &= 6,\\ \Phi(9) &= 4,\\ \Phi(10) &= 1.\end{aligned}$$

Ab dem dritten Zeitpunkt entstehen abwechselnd immer die gleichen zwei 5-Tupel von Farben, nämlich

$$\begin{aligned}\Phi((t-1)\cdot 5+1) &= \begin{cases}1, & \text{falls } t \text{ ungerade}\\ 3, & \text{sonst}\end{cases}\\ \Phi((t-1)\cdot 5+2) &= \begin{cases}2, & \text{falls } t \text{ ungerade}\\ 5, & \text{sonst}\end{cases}\\ \Phi((t-1)\cdot 5+3) &= \begin{cases}4, & \text{falls } t \text{ ungerade}\\ 6, & \text{sonst}\end{cases}\\ \Phi((t-1)\cdot 5+4) &= \begin{cases}2, & \text{falls } t \text{ ungerade}\\ 5, & \text{sonst}\end{cases}\\ \Phi((t-1)\cdot 5+5) &= \begin{cases}3, & \text{falls } t \text{ ungerade}\\ 1, & \text{sonst}\end{cases}\end{aligned} \quad , \quad t \leq N,$$

wobei $3 \leq t \leq N+1$ mit der Einschränkung, dass die letzte Steuervariable nicht existiert. Da die Variable t_N in allen Zeilen auftritt, gilt zwangsläufig

$$\Phi(5 \cdot (N+1)) = 7.$$

Dass 7 die chromatische Zahl ist, sieht man beispielsweise daran, dass die sieben Spalten, die zu den Variablen $x_{1,0}, x_{2,0}, x_{3,0}, u_0, x_{2,1}, x_{3,1}$ und t_N gehören, eine Clique bilden.
Wählt man eine zur Partitionierung ungünstigere Reihenfolge, wie etwa

$$u_0, u_1, ..., u_{N-1}, x_{10}, x_{11}, ..., x_{1,N}, x_{20}, ..., x_{2N}, ..., x_{40}, ..., x_{4N}, t_N$$

so ermittelt der CPR-Algorithmus 8 Gruppen, während alle anderen in Abschnitt 3.4 vorgestellten Verfahren auch dann 7 Gruppen erstellen.
Wie man sieht, ist die Anzahl der Gruppen unabhängig von der Anzahl der gewählten Diskretisierungspunkte. Damit ist die Ersparnis durch die Gruppen nicht in Zahlen zu fassen. Die Variablenanzahl kann durch hinreichend viele Diskretisierungspunkte beliebig hoch gesetzt werden, während die Gruppenanzahl im Vergleich dazu verschwindend gering ist.

Die Hessematrix hat die folgende Struktur

$$\nabla^2 G(x) = \left[\begin{array}{cccccccccccccccccccccccc}
\times & \times & \times & & & & & & & & & & & & & & & & & & & & & \times \\
\times & \times & \times & & & & & & & & & & & & & & & & & & & & & \times \\
\times & \times & \times & & & & & & & & & & & & & & & & & & & & & \times \\
 & & & \times & & & & & & & & & & & & & & & & & & & & \\
 & & & & \times & & & & & & & & & & & & & & & & & & & \times \\
 & & & & & \times & \times & \times & & & & & & & & & & & & & & & & \times \\
 & & & & & \times & \times & \times & & & & & & & & & & & & & & & & \times \\
 & & & & & \times & \times & \times & & & & & & & & & & & & & & & & \times \\
 & & & & & & & & \times & & & & & & & & & & & & & & & \\
 & & & & & & & & & \times & & & & & & & & & & & & & & \times \\
 & & & & & & & & & & \cdot & & & & & & & & & & & & & \cdot \\
 & & & & & & & & & & & \cdot & & & & & & & & & & & & \cdot \\
 & & & & & & & & & & & & \cdot & & & & & & & & & & & \cdot \\
 & & & & & & & & & & & & & \times & \times & \times & & & & & & & & \times \\
 & & & & & & & & & & & & & \times & \times & \times & & & & & & & & \times \\
 & & & & & & & & & & & & & \times & \times & \times & & & & & & & & \times \\
 & & & & & & & & & & & & & & & & \times & & & & & & & \\
 & & & & & & & & & & & & & & & & & \times & & & & & & \times \\
 & & & & & & & & & & & & & & & & & & \times & \times & \times & & & \\
 & & & & & & & & & & & & & & & & & & \times & \times & \times & & & \\
 & & & & & & & & & & & & & & & & & & \times & \times & \times & & & \\
 & & & & & & & & & & & & & & & & & & & & & \times & & \\
 & & & & & & & & & & & & & & & & & & & & & & \times & \\
\times & \times & \times & & \times & \times & \times & \times & & \times & \cdot & \cdot & \cdot & \times & \times & \times & & \times & & & & & & \times
\end{array}\right].$$

Diese Struktur ergibt sich gemäß Definition 3.5.6 daraus, für welche $i, j \in \{1, ..., n\}$ es einen (tatsächlich) relevanten Eintrag g_k im Sinne von (3.20) gibt.

Wie in Bemerkung 3.5.8 erwähnt, gehen wir im Folgenden, nachdem die Struktur von $\nabla^2 G(x)$ bekannt ist, nicht davon aus, detailliertere Informationen über die Struktur zu haben, beispielsweise welche der Einträge g_k für die einzelnen Paare relevant sind. Aufgrund der Existenz des Paares (i, j) ist uns lediglich bekannt, dass es einen relevanten Eintrag für dieses Paar geben muss. Ansonsten verwenden wir jedoch die Indexmenge $Rele^+((i, j))$ der vermutlich relevanten Einträge. Das heißt, existiert laut der Struktur von $\nabla g(x)$ eine Abhängigkeit des Eintrags g_k von x_i und x_j, so wird nachfolgend davon ausgegangen, dass g_k für das Paar (i, j) relevant ist.

Die Hesse-Partitionierung lautet:

$$\begin{aligned}
\Phi_H((N+1) \cdot 5) &= 1, \\
\Phi_H((t-1) \cdot 5 + 1) &= 2, \\
\Phi_H((t-1) \cdot 5 + 2) &= 3, \\
\Phi_H((t-1) \cdot 5 + 3) &= 4, \\
\Phi_H((t-1) \cdot 5 + 4) &= 1, \\
\Phi_H((t-1) \cdot 5 + 5) &= 2.
\end{aligned}$$

Hierbei wurde, wie in Abschnitt 3.5.1 angeregt, die LFO-Reihenfolge gewählt. Daher wurde t_N der Hessegruppe 1 zugeordnet. Da die Variablen x_{4t}, $t = 0, ..., N$ mit keiner anderen Variable ein Paar bilden, können sie alle ebenfalls in die Hessegruppe 1 einsortiert werden. Die Anzahl 4 der hier ermittelten Hessegruppen ist optimal, was daran ersichtlich ist, dass die Variablen x_{10}, x_{20}, x_{30} und t_N eine Clique bilden.

Die relevanten Variablen der unterschiedlichen Hessegruppen

$$hg_c := \Phi_H^{-1}(c), \quad c = 1, 2, 3, 4$$

lauten:

$$\begin{aligned}
Relv(hg_1) &:= \{(t-1)\cdot 5+1,\ t=1,...,N+1,\\
&\quad (t-1)\cdot 5+2,\ t=1,...,N+1,\\
&\quad (t-1)\cdot 5+3,\ t=1,...,N+1,\\
&\quad (t-1)\cdot 5+5,\ t=1,...,N\}\\
Relv(hg_2) &:= \{(t-1)\cdot 5+2,\ t=1,...,N+1,\\
&\quad (t-1)\cdot 5+3,\ t=1,...,N+1\}\\
Relv(hg_3) &:= \{(t-1)\cdot 5+3,\ t=1,...,N+1\}\\
Relv(hg_4) &:= \emptyset
\end{aligned}$$

und die relevanten Einträge

$$\begin{aligned}
Rele(hg_1) &:= \{1,...,m\},\\
Rele(hg_2) &:= \{1,...,m\},\\
Rele(hg_3) &:= \{(t-1)\cdot 4+2,\ t=1,...,N,\\
&\quad (t-1)\cdot 4+3,\ t=1,...,N\}.
\end{aligned}$$

Dadurch ergibt sich, dass die Subpartitionierung zur Hessegruppe hg_1 der gewöhnlichen Partitionierung gleicht:

$$\Phi_{sub,hg_1}(i) = \Phi(i), \quad i=1,...,n.$$

Für die Subpartitionierung zur Hessegruppe hg_2 gilt:

$$\begin{aligned}
\Phi_{sub,hg_2}((t-1)\cdot 5+2) &= \begin{cases} 1, & \text{falls } t \text{ ungerade}\\ 2, & \text{sonst}\end{cases}\\
\Phi_{sub,hg_2}((t-1)\cdot 5+3) &= \begin{cases} 3, & \text{falls } t \text{ ungerade}\\ 4, & \text{sonst}\end{cases}
\end{aligned}$$

und für die Hessegruppe hg_3

$$\Phi_{sub,hg_3}((t-1)\cdot 5+2) = \begin{cases} 1, & \text{falls } t \text{ ungerade}\\ 2, & \text{sonst}\end{cases} .$$

Die Anzahl der Auswertungen für die Bestimmung der zweiten Ableitung, sofern die Auswertungen für die Bestimmung der ersten Ableitungen bereits vorliegen, beträgt daher

$$3+7+4+2=16.$$

Wieder stellen wir fest, dass die Anzahl der benötigten Auswertungen von der Anzahl der Diskretisierungspunkte und damit von der Anzahl der Variablen unabhängig ist.
Alternativ lässt sich auch die Paargruppenmethode anwenden. Die resultierende Paar-Partitionierung lautet:

$$\Phi_P((t-1)\cdot 5+1,(t-1)\cdot 5+2) = \begin{cases} 1, & \text{falls } t \text{ ungerade}\\ 2, & \text{sonst}\end{cases} ,$$

$$
\begin{aligned}
\Phi_P((t-1)\cdot 5+1,(t-1)\cdot 5+3) &= \begin{cases} 3, & \text{falls } t \text{ ungerade} \\ 4, & \text{sonst} \end{cases}, \\
\Phi_P((t-1)\cdot 5+1,(N+1)\cdot 5) &= \begin{cases} 5, & \text{falls } t \text{ ungerade} \\ 6, & \text{sonst} \end{cases}, \\
\Phi_P((t-1)\cdot 5+2,(t-1)\cdot 5+3) &= \begin{cases} 7, & \text{falls } t \text{ ungerade} \\ 8, & \text{sonst} \end{cases}, \\
\Phi_P((t-1)\cdot 5+2,(N+1)\cdot 5) &= \begin{cases} 9, & \text{falls } t \text{ ungerade} \\ 10, & \text{sonst} \end{cases}, \\
\Phi_P((t-1)\cdot 5+3,(N+1)\cdot 5) &= \begin{cases} 11, & \text{falls } t \text{ ungerade} \\ 12, & \text{sonst} \end{cases}, \\
\Phi_P((t-1)\cdot 5+5,(N+1)\cdot 5) &= \begin{cases} 13, & \text{falls } t \text{ ungerade} \\ 14, & \text{sonst} \end{cases}, \quad t \leq N,
\end{aligned}
$$

wobei

$$1 \leq t \leq N+1$$

mit der Einschränkung, dass die letzte Steuervariable nicht existiert.
Auch die Anzahl der Paargruppen ist offensichtlich von der Anzahl N der Diskretisierungspunkte unabhängig, so dass auch hier die Ersparnis beliebig hoch sein kann.

3.7 Komplexes Differenzieren

Martins, Kroo und Alonso präsentieren in [59] einen Finite-Differenzen-Ansatz, der es ermöglicht, die Genauigkeit erheblich zu verbessern.
Dafür benötigen wir die Cauchy-Riemannschen partiellen Differentialgleichungen:

Satz 3.7.1. *Sei $f : \mathbb{C} \to \mathbb{C}$ eine differenzierbare Funktion mit*

$$f =: u + iv$$

und beschreibe

$$z := x + iy \in \mathbb{C}$$

mit $x, y \in \mathbb{R}$ eine komplexe Zahl. Dann gilt

$$\frac{\partial u}{\partial x} = \frac{\partial v}{\partial y}, \tag{3.37}$$

$$\frac{\partial v}{\partial x} = -\frac{\partial u}{\partial y}. \tag{3.38}$$

Beweis. Vgl. [59]. Es gilt

$$f'(z_0) = \frac{\partial f}{\partial z}(z_0) = \lim_{\mathbb{C} \ni h \to 0} \frac{f(z_0 + h) - f(z_0)}{h},$$

$z_0 \in \mathbb{C}$. Nach der Kettenregel gilt

$$\frac{\partial f}{\partial x}(z_0) = \lim_{\mathbb{R} \ni h \to 0} \frac{f(z_0 + h) - f(z_0)}{h} = \underbrace{\frac{\partial z}{\partial x}}_{=1} \underbrace{\frac{\partial f}{\partial z}(z_0)}_{=f'(z_0)} = f'(z_0)$$

$$\frac{\partial f}{\partial y}(z_0) = \lim_{\mathbb{R} \ni h \to 0} \frac{f(z_0 + ih) - f(z_0)}{h} = \lim_{\mathbb{R} \ni h \to 0} i \frac{f(z_0 + ih) - f(z_0)}{ih} = \underbrace{\frac{\partial z}{\partial y}}_{=i} \underbrace{\frac{\partial f}{\partial z}(z_0)}_{=f'(z_0)} = if'(z_0).$$

Damit gilt

$$0 = f'(z_0) + i^2 f'(z_0) = \frac{\partial f}{\partial x}(z_0) + i \frac{\partial f}{\partial y}(z_0)$$

und somit

$$0 = 0 + i \cdot 0 = \frac{\partial u}{\partial x} + i \frac{\partial v}{\partial x} + i \frac{\partial u}{\partial y} + i^2 \frac{\partial v}{\partial y} = \left(\frac{\partial u}{\partial x} - \frac{\partial v}{\partial y} \right) + i \left(\frac{\partial v}{\partial x} + \frac{\partial u}{\partial y} \right)$$

Hieraus folgt unmittelbar

$$\frac{\partial u}{\partial x} = \frac{\partial v}{\partial y},$$

$$\frac{\partial v}{\partial x} = -\frac{\partial u}{\partial y}.$$

□

Diesen Satz machen wir uns zu Nutze bei der Differentiation einer reellen Funktion. In unserem Fall gilt folglich

$$\begin{aligned} u(x) &= f(x) \\ v(x) &= 0, \end{aligned}$$

sowie

$$y = 0$$

Nach Gleichung (3.37) gilt

$$\frac{\partial u}{\partial x} = \lim_{h \to 0} \frac{v(x + i(y + h)) - v(x + iy)}{h}$$

beziehungsweise

$$\frac{\partial u}{\partial x} = \lim_{h \to 0} \frac{Im(f(x + i(y + h)))}{h}$$

Für ein hinreichend kleines h kann die Ableitung daher wie folgt approximiert werden

$$\frac{\partial u}{\partial x} \approx \frac{Im(f(x + i(y + h)))}{h}.$$

Der große Vorteil dieses Verfahrens gegenüber dem gewöhnlichen Finite-Differenzen-Verfahren ist, dass es hierbei zu keinerlei Auslöschung kommt, da keine Subtraktion vorgenommen wird. Zur Betrachtung der Fehlerordnung entwickeln wir eine Taylorreihe mit einem imaginären Schritt. Dies setzt jedoch voraus, dass die Funktion analytisch ist.

$$f(x + ih) = f(x) + ihf'(x) - h^2 \frac{f''(x)}{2!} - ih^3 \frac{f'''(x)}{3!} + ...$$

Wir betrachten lediglich den Imaginärteil und teilen durch h und erhalten

$$f'(x) = \frac{Im(f(x + ih))}{h} + h^2 \frac{f'''(x)}{3!} + ...$$

Daher ist die Fehlerordnung der Approximation $O(h^2)$, die gleiche Ordnung folglich wie beim zentralen Differenzenquotienten. Hier bedarf es dafür jedoch lediglich einer Funktionsauswertung. Für eine vektorwertige Funktion g lässt sich dieses Verfahren mit den graphentheoretischen Verfahren aus diesem Kapitel kombinieren und höchste Effizienz mit hoher Genauigkeit kombinieren.

4 Update-Techniken

4.1 Einleitung

"Der Mangel an mathematischer Bildung gibt sich durch nichts so auffallend zu erkennen wie durch maßlose Schärfe im Zahlenrechnen."
Carl Friedrich Gauß, 1777-1855

Ist die Approximation der Hessematrix trotz der in Abschnitt 3.5 eingeführten Verfahren zu aufwändig, besteht die Möglichkeit, diese auf eine vollkommen andere Art zu approximieren. Bei den so genannten *Update-Verfahren* wird ausgehend von einer Startschätzung H_0 die Hessematrix durch Addition eines matrixwertigen Terms aktualisiert. Dieser matrixwertige Term entsteht in der Regel aus einer numerisch wenig aufwändigen Rechnung, so dass die Bestimmung der neuen Approximation auf sehr effiziente Weise erfolgen kann.
Der Nachteil, den diese Verfahren haben, ist, dass die Approximation sehr viel ungenauer ist als die Approximation mittels finiter Differenzen. Da hiermit einhergeht, dass die quadratische Approximation des ursprünglichen NLP-Problems (2.1)-(2.3) sehr viel ungenauer ist, leuchtet ein, dass Update-Verfahren für gewöhnlich mehr Iterationen benötigen als exaktere Verfahren.
Insbesondere bei kleinen, dicht besetzten Problemen haben sich Update-Verfahren jedoch zum Standard innerhalb von SQP-Verfahren entwickelt. Das meist gebrauchte Update-Verfahren ist das so genannte *Broyden-Fletcher-Goldfarb-Shanno (BFGS)-Verfahren*, welches von Broyden [5], Fletcher [26], Goldfarb [44] und Shanno [76] unabhängig voneinander entwickelt wurde. Dieses Verfahren weist lokal superlineare Konvergenz auf. Das Update besteht dabei aus der Summe zweier dyadischer Produkte von Vektoren, die einerseits aus einer Vektordifferenz und andererseits aus einer einfachen Matrix-Vektor-Multiplikation entstehen. Somit ist ersichtlich, wie schnell dieses Update erstellt werden kann und dass dies in den meisten Fällen Grund genug ist, auf die quadratische Konvergenzeigenschaft exakterer Verfahren zu Gunsten der superlinearen Konvergenz des BFGS-Verfahrens zu verzichten.
Wie in Abschnitt 3.1 bereits dargelegt, sind die in modernen Anwendungen auftretenden großen Optimierungsprobleme häufig dünn besetzt. Bedauerlicherweise führt das BFGS-Verfahren im Allgemeinen auch dann zu dicht besetzten Matrizen, wenn die tatsächliche Struktur der Hessematrix dünn besetzt ist. Hierdurch entstehen bei hochdimensionalen Problemen zwei zentrale Schwierigkeiten:
Aufgrund der dichten Struktur entsteht ein erhöhter Speicheraufwand bei der Speicherung der auftretenden Matrizen im Vergleich zur ursprünglichen Problemformulierung. Des Weiteren werden die weiterführenden Rechnungen (beispielsweise lineare Gleichungssysteme), die innerhalb des Optimierungsprozesses häufig durchgeführt werden, somit unnötig und drastisch verkompliziert, wodurch die Rechenzeit erheblich erhöht wird. Daher besteht ein Bedarf an Update-Verfahren, die die Dünnbesetztheit berücksichtigen und dennoch gute Konver-

genzeigenschaften aufweisen.
Nachdem wir in Abschnitt 4.2 das klassische BFGS-Verfahren einführen, geben wir in Abschnitt 4.3 einen kurzen Überblick über existierende Update-Verfahren, die dünnbesetzte Approximationsmatrizen erzeugen. Bis auf ein Verfahren liefern die dort genannten Arbeiten entweder keine Konvergenzaussage oder dies nur für eine sehr eingeschränkte Problemklasse. Die einzige Ausnahme bildet das partitionierte BFGS-Verfahren von Griewank und Toint. Zur Implementierung in einen allgemeinen Löser für hochdimensionale nichtlineare Optimierung ist dieses Verfahren jedoch nur sehr bedingt geeignet, da für die Anwendung des partitionierten BFGS-Verfahren Informationen bereit gestellt werden müssen, die die meisten Anwender nicht liefern können. Obwohl die im Rahmen dieser Arbeit entwickelten dünn besetzten Update-Verfahren nicht darauf aufbauend entwickelt wurden, hilft uns das partitionierte BFGS-Verfahren dabei, die superlineare Konvergenz unserer Verfahren nachzuweisen. Aufgrunddessen wird das partitionierte BFGS-Verfahren in Abschnitt 4.4 ausführlich behandelt.
In den darauffolgenden Abschnitten entwickeln wir verschiedene, auf einander aufbauende, dünnbesetzte Update-Verfahren. Für diese Verfahren weisen wir die superlineare Konvergenzeigenschaft nach.

4.2 Klassisches Rang-2-BFGS-Verfahren

Die Update-Verfahren lassen sich prinzipiell für die Approximation jeder beliebigen (hinreichend differenzierbaren) Funktion anwenden. In unserem Fall suchen wir die Approximation

$$H \approx \nabla^2 f(x)$$

aus (2.22). Da die Gestalt der Funktion für das Updateverfahren unerheblich ist, gehen wir in diesem Kapitel davon aus, dass die Funktion, deren Hessematrix zu approximieren ist, f sei. Andernfalls wäre f jeweils durch L zu ersetzen und die zu approximierende Matrix lautete dann

$$H \approx \nabla^2_{xx} L(x, \lambda).$$

Weiterhin sei n die Anzahl der Optimierungsvariablen und $x^{(k)} \in \mathbb{R}^n$ die Iterierte in der k-ten Iteration. Aufgrund der verwendeten Approximation spricht man statt von einem Newtonverfahren, von einem *Quasi-Newton-Verfahren.*

4.2.1 Methodik

Ausgehend von einer Startmatrix $H^{(0)}$ entwickeln wir in jeder Iteration eine neue Approximation $H^{(k)}$ der Hessematrix $\nabla^2 f(x^{(k)})$. Es hat sich gezeigt, dass Update-Verfahren gute Konvergenzeigenschaften aufweisen wenn sie die Bedingung

$$\lim_{k \to \infty} \frac{\|H^{(k)} d^{(k)} - y^{(k)}\|}{\|d^{(k)}\|} = 0 \tag{4.1}$$

erfüllen, wobei

$$d^{(k)} := x^{(k+1)} - x^{(k)} \text{ und} \tag{4.2}$$
$$y^{(k)} := \nabla f(x^{(k+1)}) - \nabla f(x^{(k)}). \tag{4.3}$$

Man beachte, dass im Falle der restringierten Optimierung, das heißt $m \neq 0$ die Lagrangemultiplikatoren festgehalten werden und

$$y^{(k)} := \nabla L(x^{(k+1)}, \lambda^{(k+1)}) - \nabla L(x^{(k)}, \lambda^{(k+1)})$$

gewählt wird. Um (4.1) im Grenzfall zu erfüllen, wird $H^{(k+1)}$ so bestimmt, dass

$$H^{(k+1)} d^{(k)} = y^{(k)} \tag{4.4}$$

gilt. (4.4) wird auch als *Quasi-Newton-Bedingung* bezeichnet. Man beachte, dass die Bedingung

$$H^{(k)} d^{(k)} = y^{(k)}$$

nicht erfüllbar ist, da wegen des verwendeten Quasi-Newton-Verfahrens bereits

$$H^{(k)} d^{(k)} = -\nabla f(x^{(k)}) \tag{4.5}$$

gilt. Ausgehend von einer gegebenen Matrix H wird im Folgenden nur noch von H und ihrem Update H^+ die Rede sein, sowie von d und y, statt von $H^{(k)}, H^{(k+1)}, d^{(k)}, y^{(k)}$.
Der Rang-2-Ansatz mit zwei Korrekturtermen

$$H^+ = H + \gamma uu^T + \delta vv^T, \quad u, v, \in \mathbb{R}^n$$

stellt sicher, dass die Hessematrix stets symmetrisch ist. (4.4) verlangt, dass

$$y = H^+ d = Hd + \gamma u(u^T d) + \delta v(v^T d)$$

gilt. Somit muss $y - Hd$ eine Linearkombination von u und v sein. Mit der Wahl

$$\begin{aligned} u &:= y, \\ v &:= -Hd \end{aligned}$$

folgt

$$\begin{aligned} \gamma &= \frac{1}{u^T d} = \frac{1}{y^T d}, \\ \delta &= \frac{1}{v^T d} = \frac{1}{d^T H d}. \end{aligned}$$

Die Rang-2-BFGS-Update-Formel lautet daher

$$H^+ = H + \frac{yy^T}{y^T d} - \frac{(Hd)(Hd)^T}{d^T H d}. \tag{4.6}$$

Wir zitieren den Spektralsatz:

Satz 4.2.1. *Sei $A \in \mathbb{R}^{n\times n}$ eine symmetrische Matrix. Dann existiert eine orthogonale Matrix $Q \in \mathbb{R}^{n\times n}$ und eine Diagonalmatrix*

$$\Lambda = \mathrm{diag}(\lambda_1, ..., \lambda_n) \in \mathbb{R}^{n\times n}$$

mit

$$A = Q^T \Lambda Q.$$

Hierbei sind $\lambda_i,\ i = 1, ..., n$ die Eigenwerte der Matrix A und die Zeilenvektoren von Q die zugehörigen Eigenvektoren.

Bevor wir die Bedingung für die positive Definitheit des Updates aufstellen, benötigen wir das folgende

Lemma 4.2.2. *Sei $A \in \mathbb{R}^{n\times n}$ eine symmetrische und positiv definite Matrix. Dann existiert eine ebenfalls symmetrische und positiv definite Matrix $A^{\frac{1}{2}} \in \mathbb{R}^{n\times n}$ mit*

$$A^{\frac{1}{2}} A^{\frac{1}{2}} = A.$$

$A^{\frac{1}{2}}$ bezeichnen wir als Quadratwurzel von A.

Beweis. Nach dem Spektralsatz 4.2.1 existiert zur symmetrischen Matrix A eine orthogonale Matrix $Q \in \mathbb{R}^{n\times n}$ und eine Diagonalmatrix

$$\Lambda = \operatorname{diag}(\lambda_1, ..., \lambda_n) \in \mathbb{R}^{n\times n},$$

mit den Eigenvektoren $\lambda_1, ..., \lambda_n$ der Matrix A, so dass

$$A = Q^T \Lambda Q.$$

Wegen der positiven Definitheit von A, gilt

$$\lambda_i > 0, \quad i = 1, ..., n.$$

Setze

$$\Lambda^{\frac{1}{2}} := \operatorname{diag}(\sqrt{\lambda_1}, ..., \sqrt{\lambda_n}) \in \mathbb{R}^{n\times n}.$$

Offensichtlich kann damit

$$A^{\frac{1}{2}} = Q^T \Lambda^{\frac{1}{2}}$$

gewählt werden, so dass

$$A^{\frac{1}{2}} \cdot A^{\frac{1}{2}} = Q^T \Lambda^{\frac{1}{2}} \Lambda^{\frac{1}{2}} Q = A$$

ist. □

Für den nachfolgenden Satz benötigen wir die Cauchy-Schwarzsche Ungleichung, die besagt, dass für einen beliebigen reellen (oder komplexen) Vektorraum V mit einem inneren Produkt $(\cdot;\cdot)$ die Beziehung

$$(v;w)^2 \leq (v;v) \cdot (w;w) \tag{4.7}$$

gilt, woraus unmittelbar

$$(v;w) \leq \|v\| \cdot \|w\|$$

folgt. Hiermit lässt sich zeigen:

Satz 4.2.3. *Sei $H \in \mathbb{R}^{n\times n}$ eine symmetrische, positiv definite Matrix und sei $d \neq 0$. Dann gilt:*

$$H^+ \text{ ist positiv definit } \iff y^T d > 0.$$

Beweis. Vgl. [34].
"$\Longrightarrow$":
Sei H^+ positiv definit. Dann gilt insbesondere

$$0 < d^T H^+ d = d^T H d + d^T \frac{yy^T}{y^T d} d - d^T \frac{(Hd)(Hd)^T}{d^T H d} d = d^T H d + d^T y - d^T H d = d^T y,$$

da $d \neq 0$ vorausgesetzt wurde.
"$\Longleftarrow$":
Sei $x \in \mathbb{R}^n \setminus \{0\}$ beliebig. Da H als symmetrisch und positiv definit vorausgesetzt wurde, existiert nach Lemma 4.2.2 eine ebenfalls symmetrische und positiv definite Quadratwurzel $H^{\frac{1}{2}}$.
Sei

$$z := H^{\frac{1}{2}} d$$

und

$$\omega := H^{\frac{1}{2}}x.$$

Dann gilt

$$\begin{aligned} x^T H^+ x &= x^T Hx + \frac{(x^T y)^2}{y^T d} - \frac{(x^T Hd)^2}{d^T Hd} \\ &= \omega^T \omega + \frac{(x^T y)^2}{y^T d} - \frac{(\omega^T z)^2}{z^T z} \\ &\overset{(4.7)}{\geq} \omega^T \omega + \frac{(x^T y)^2}{y^T d} - \frac{\|\omega\|^2 \|z\|^2}{\|z\|^2} \\ &= \frac{(x^T y)^2}{y^T d} \\ &\geq 0, \quad \text{falls } y^T d > 0. \end{aligned}$$

Unter der Bedingung

$$y^T d > 0 \tag{4.8}$$

gilt tatsächlich die echte Ungleichung und somit die positive Definitheit:

$$x^T H^+ x = 0$$

kann nämlich nur gelten, falls sowohl

$$\omega z = \|\omega\| \|z\| \tag{4.9}$$

als auch

$$x^T y = 0 \tag{4.10}$$

gilt.
(4.9) impliziert die lineare Abhängigkeit von ω und z, wodurch folglich ein $\alpha \in \mathbb{R}$ existiert, so dass

$$\omega = \alpha z$$

gilt. Nach Konstruktion ist dann auch

$$x = \alpha d$$

und somit wegen (4.10)

$$d^T y = 0,$$

was im Widerspruch zu (4.8) steht. □

Gelten in x^* die hinreichenden Optimalitätskriterien zweiter Ordnung (2.13)-(2.16) dann ist tatsächlich aufgrund des folgenden Satzes (4.8) in einer hinreichend kleinen Umgebung um x^* immer erfüllt. Hier und in der gesamten Arbeit gelten die folgenden Bezeichnungen:

$$\begin{aligned} U_\varepsilon(x^*) &:= \{x \in \mathbb{R}^n : \ \|x - x^*\|_2 < \varepsilon\}, \\ \overline{U}_\varepsilon(x^*) &:= \{x \in \mathbb{R}^n : \ \|x - x^*\|_2 \leq \varepsilon\}. \end{aligned}$$

und $\lambda_{\min}(A)$ beziehungsweise $\lambda_{\max}(A)$ bezeichnen den kleinsten beziehungsweise größten Eigenwert einer Matrix A.

Satz 4.2.4. *Sei f zweimal stetig differenzierbar und $\nabla^2 f$ lokal Lipschitz-stetig mit Lipschitz-Konstante L. Außerdem sei $x^* \in \mathbb{R}^n$ mit $\nabla f(x^*) = 0$ und $\nabla^2 f(x^*)$ positiv definit. Dann existiert für jedes $\delta \in (0,1)$ ein $\varepsilon > 0$ so dass für alle $x, x^+ \in U_\varepsilon(x^*)$ mit $x \neq x^+$*

$$\big(\nabla f(x^+) - \nabla f(x)\big)^T (x^+ - x) > \delta (x^+ - x)^T \nabla^2 f(x)(x^+ - x) > 0$$

gilt.

Beweis. Vgl. [34]. Wegen der vorausgesetzten Stetigkeit von $\nabla^2 f$ ist für ein hinreichend kleines $r > 0$ auch $\nabla^2 f(x),\ x \in U_r(x^*)$ positiv definit. Demnach ist

$$\rho := \inf\{\lambda_{\min}(\nabla^2 f(x)),\ x \in U_r(x^*)\} > 0.$$

Wähle

$$\varepsilon := \frac{\min\,(r; (1-\delta)\rho)}{L}.$$

Nach dem Mittelwertsatz existiert ein $\xi \in U_\varepsilon(x^*)$, so dass

$$\begin{aligned}
\big(\nabla f(x^+) - \nabla f(x)\big)^T (x^+ - x) &= (x^+ - x)^T \nabla^2 f(\xi)(x^+ - x) \\
&= (x^+ - x)^T \nabla^2 f(x)(x^+ - x) \\
&\quad -(x^+ - x)^T \big(\nabla^2 f(x) - \nabla^2 f(\xi)\big)(x^+ - x) \\
&\geq (x^+ - x)^T \nabla^2 f(x)(x^+ - x) \\
&\quad - \underbrace{\|\nabla^2 f(x) - \nabla^2 f(\xi)\|}_{\leq L\varepsilon} \|x^+ - x\|^2 \\
&\geq (x^+ - x)^T \nabla^2 f(x)(x^+ - x) - \underbrace{L\varepsilon \|x^+ - x\|^2}_{\leq (1-\delta)\rho} \\
&> \delta (x^+ - x)^T \nabla^2 f(x)(x^+ - x) \\
&> 0.
\end{aligned}$$

□

Falls (4.8) nicht gilt, folglich die aktuelle Iterierte nicht nahe genug bei x^* ist, verzichtet man beim Update (in der jeweiligen Iteration) auf (4.4) und setzt

$$u = q := \theta y + (1-\theta) Hd \tag{4.11}$$

mit

$$\theta = \begin{cases} 1, & \text{falls } d^T y \geq 0.2 d^T H d, \\ \dfrac{0.8 d^T H d}{d^T H d - y^T d}, & \text{sonst,} \end{cases} \tag{4.12}$$

wobei 0.2 ein heuristischer Wert ist. Diese Anpassung dient offensichtlich lediglich der Globalisierung, um hinreichend nahe an x^* zu gelangen, so dass (4.8) stets erfüllt ist. Powell [67] zeigt, dass auch hiermit superlineare Konvergenz auftritt. Wir werden uns bei unseren Konvergenzuntersuchungen in Abschnitt 4.2.2 jedoch auf den Fall beschränken, in dem y verwendet werden kann.
Mit q aus (4.11) lautet (4.6) entsprechend

$$H^+ = H + \frac{qq^T}{q^T d} - \frac{(Hd)(Hd)^T}{d^T H d}. \tag{4.13}$$

Zur Formulierung des Algorithmus' gehen wir jedoch davon aus, dass $x^{(0)} \in U_\varepsilon(x^*)$ mit $U_\varepsilon(x^*)$ aus Satz 4.2.4. Somit entwickeln wir den

Algorithmus 4.2.5.

i. Wähle $x^{(0)} \in \mathbb{R}^n$ und $H^{(0)} \in \mathbb{R}^{n\times n}$ symmetrisch und positiv definit. Wähle außerdem $\varepsilon \geq 0$ und setze $k := 0$.

ii. Ist $\|\nabla f(x^{(k)})\| \leq \varepsilon$, dann STOP.

iii. Bestimme $d^{(k)}$ aus dem Gleichungssystem

$$H^{(k)}d^{(k)} = -\nabla f(x^{(k)})$$

iv. Setze

$$\begin{aligned} x^{(k+1)} &:= x^{(k)} + d^{(k)} \text{ und} \\ y^{(k)} &:= \nabla f(x^{(k+1)}) - \nabla f(x^{(k)}). \end{aligned}$$

v. Berechne

$$H^{(k+1)} := H^{(k)} + \frac{y^{(k)}y^{(k)T}}{y^{(k)T}d^{(k)}} - \frac{(H^{(k)}d^{(k)})(H^{(k)}d^{(k)})^T}{d^{(k)T}H^{(k)}d^{(k)}}$$

vi. Setze k:=k+1 und gehe zu ii.

4.2.2 Konvergenzeigenschaften des BFGS-Verfahrens

Alle folgenden Konvergenzaussagen setzen voraus, dass die Startschätzungen $x^{(0)}$ und $H^{(0)}$ in hinreichender Weise x^* beziehungsweise $\nabla^2 f(x^*)$ approximieren und die hinreichenden Optimalitätskriterien zweiter Ordnung (2.13)-(2.16) gelten und entsprechend nach Satz 4.2.4 davon abgesehen werden kann, den Vektor q aus (4.13) zu verwenden. Darüber hinaus wird vorausgesetzt, dass $\nabla^2 f$ lokal Lipschitz-stetig ist in einer hinreichend kleinen Umgebung um x^* mit der Lipschitz-Konstanten L.
Die Resultate dieses Abschnitts stützen sich auf Griewank und Toint [48], sowie auf Dennis und Moré [24]. In Geiger und Kanzow [34] findet sich das gleiche Resultat, allerdings über das inverse BFGS-Verfahren hergeleitet.
Für die Konvergenzanalyse definieren wir uns zunächst ein alternatives Update zum BFGS-Update. Ausgehend von einer positiv definiten Matrix H sei die Matrix H' mit dem bekannten Vektor d definiert durch

$$H' := H + \frac{dd^T}{d^Td} - \frac{Hd(Hd)^T}{d^THd}, \tag{4.14}$$

wobei im Vergleich zum BFGS-Update d für y eingesetzt wurde. Für $d \neq 0$ ist das Update (4.14) nach Satz 4.2.3 ebenfalls positiv definit.
Im nachfolgenden Lemma tritt die Frobeniusnorm auf. Für eine beliebige Matrix $A \in \mathbb{R}^{n\times n}$ ist diese definiert durch

$$\|A\|_F := \sqrt{\sum_{i=1}^{n}\sum_{j=1}^{n}|a_{ij}|^2}.$$

Lemma 4.2.6. *Für das Update (4.14) gilt zwischen H' und dessen Vorgänger H die Beziehung*

$$\|H' - I\|_F^2 \leq \|H - I\|_F^2,$$

wobei I die Einheitsmatrix bezeichne.

Beweis. Vgl. [48]. Mit dem Kroneckersymbol

$$\delta_{ij} := \begin{cases} 1, & \text{falls } i = j \\ 0, & \text{sonst} \end{cases}$$

gilt:

$$\begin{aligned}
\|H' - I\|_F^2 - \|H - I\|_F^2 =& \sum_{i=1}^{n}\sum_{j=1}^{n}\left[(H_{ij} - \delta_{ij}) + \left(\frac{d_i d_j}{d^T d} - \frac{(Hd)_i (Hd)_j}{d^T H d}\right)\right]^2 \\
& - \sum_{i=1}^{n}\sum_{j=1}^{n}\Big(H_{ij} - \delta_{ij}\Big)^2 \\
=& \sum_{i=1}^{n}\sum_{j=1}^{n}\left[2\left(\frac{d_i d_j}{d^T d} - \frac{(Hd)_i (Hd)_j}{d^T H d}\right)(H_{ij} - \delta_{ij})\right. \\
& \left. + \left(\frac{d_i d_j}{d^T d} - \frac{(Hd)_i (Hd)_j}{d^T H d}\right)^2\right] \\
=& 2\sum_{i=1}^{n}\left(-\frac{d_i^2}{d^T d} + \frac{(Hd)_i^2}{d^T H d}\right) \\
& + \sum_{i=1}^{n}\sum_{j=1}^{n}\left(2\frac{d_i H_{ij} d_j}{d^T d} - 2\frac{(Hd)_i H_{ij} (Hd)_j}{d^T H d} + \frac{d_i^2 d_j^2}{(d^T d)^2}\right. \\
& \left. - 2\frac{d_i (Hd)_i d_j (Hd)_j}{d^T d d^T H d} + \frac{(Hd)_i^2 (Hd)_j^2}{(d^T H d)^2}\right) \\
=& -2 + 2\frac{d^T H H d}{d^T H d} + 2\frac{d^T H d}{d^T d} - 2\frac{d^T H H H d}{d^T H d} + 1 \\
& - 2\frac{d^T H d}{d^T d} + \left(\frac{d^T H H d}{d^T H d}\right)^2 \\
=& -1 + 2\frac{d^T H H d}{d^T H d} - \left(\frac{d^T H H d}{d^T H d}\right)^2 + 2\left(\frac{d^T H H d}{d^T H d}\right)^2 \\
& - 2\frac{d^T H H H d}{d^T H d} \\
=& -\left(1 - \frac{d^T H H d}{d^T H d}\right)^2 - 2\left[\frac{d^T H H H d}{d^T H d} - \left(\frac{d^T H H d}{d^T H d}\right)^2\right]. \qquad (4.15)
\end{aligned}$$

Da H positiv definit ist, existiert nach Lemma 4.2.2 eine symmetrische Quadratwurzel $H^{\frac{1}{2}}$.

Definiere

$$\begin{aligned} v &:= H^{\frac{1}{2}} H d \\ w &:= H^{\frac{1}{2}} d. \end{aligned}$$

Nach der Cauchy-Schwarzschen Ungleichung (4.7) gilt

$$(d^T H H d)^2 = (d^T H H^{\frac{1}{2}} H^{\frac{1}{2}} d)^2 = (v^T w)^2 \leq v^T v w^T w = (d^T H H H d)(d^T H d),$$

weswegen die eckige Klammer in (4.15) nicht negativ ist, woraus die Behauptung folgt. □

Lemma 4.2.7. *Für alle $u, v \in \mathbb{R}^n$ gilt*

$$\|uv^T\|_F = \|u\|\|v\|.$$

Beweis. Vgl. [34]. Für eine beliebige Matrix $A \in \mathbb{R}^{n \times n}$ gilt

$$\|A\|_F^2 = \sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij}^2 = \sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij} \cdot a_{ij} = \sum_{i=1}^{n} A_i^T A_i = \text{Spur}(A^T A). \tag{4.16}$$

Des Weiteren gilt für beliebige $u, v \in \mathbb{R}^n$:

$$(uv^T)_{ii} = u_i v_i \implies \text{Spur}(uv^T) = \sum_{i=1}^{n} u_i v_i = u^T v. \tag{4.17}$$

Aus (4.16), (4.17) und der offensichtlichen Linearität der Spurabbildung folgt

$$\|uv^T\|_F^2 = \text{Spur}(vu^T uv^T) = u^T u \cdot \text{Spur}(vv^T) = \|u\|^2 v^T v = \|u\|^2 \|v\|^2$$

□

Damit zeigen wir den

Satz 4.2.8. *Es existiert ein $\varepsilon > 0$ und eine Konstante $c > 0$, so dass für die Iterierten*

$$x^{(k)} + d^{(k)}, x^{(k)} \in U_\varepsilon(x^*)$$

die Ungleichung

$$\|H^{(k+1)} - \nabla^2 f(x^*)\| \leq \|H^{(k)} - \nabla^2 f(x^*)\| + c \max\{\|x^{(k)} + d^{(k)} - x^*\|, \|x^{(k)} - x^*\|\}$$

gilt.

Beweis. Sei zunächst $\varepsilon > 0$ beliebig. Nach dem Mittelwertsatz existiert für

$$x^{(k)}, x^{(k+1)} \in U_\varepsilon(x^*)$$

ein $\xi \in U_\varepsilon(x^*)$, so dass

$$y^{(k)} = \nabla f(x^{(k+1)}) - \nabla f(x^{(k)})$$

$$\begin{aligned} &= \nabla^2 f(\xi) d^{(k)} \\ &= \nabla^2 f(x^*) d^{(k)} + (\nabla^2 f(\xi) - \nabla^2 f(x^*)) d^{(k)}. \end{aligned} \tag{4.18}$$

Wir betrachten zunächst die Funktion

$$\tilde{f} : \mathbb{R}^n \to \mathbb{R}, \quad \tilde{f}(z) = f(Az) = f(x),$$

wobei

$$\begin{aligned} A &:= \nabla^2 f(x^*)^{-\frac{1}{2}} \text{ und} \\ z &:= A^{-1} x \end{aligned}$$

sei und verwenden

$$\tilde{H}^{(0)} := A H^{(0)} A$$

als Startmatrix. Seien z und $\hat{z} := A^{-1}\hat{x}$ beliebig. Dann gilt offenbar

$$\|x - \hat{x}\| = \|A(z - \hat{z})\| \leq \|A\| \|z - \hat{z}\|.$$

Definieren wir

$$\tilde{\varepsilon} := \frac{1}{\|A\|} \varepsilon,$$

so gilt

$$z \in U_{\tilde{\varepsilon}}(z^*) \Longrightarrow x \in U_{\varepsilon}(x^*).$$

Für die Ableitungen der Funktion $\tilde{f}$ gilt

$$\begin{aligned} \nabla \tilde{f}(z) &= A \nabla f(Az) = A \nabla f(x) \text{ und} \\ \nabla^2 \tilde{f}(z) &= A \nabla^2 f(Az) A = A \nabla^2 f(x) A \end{aligned}$$

und daher für beliebige $z, \hat{z} \in U_{\tilde{\varepsilon}}(z^*)$

$$\|\nabla^2 \tilde{f}(z) - \nabla^2 \tilde{f}(\hat{z})\| = \|A(\nabla^2 f(x) - \nabla^2 \tilde{f}(\hat{x}))A\| \leq \|A\|^2 L \|x - \hat{x}\| \leq \underbrace{\|A\|^3 L}_{=\tilde{L}} \|z - \hat{z}\|,$$

weswegen $\tilde{L}$ die Lipschitz-Konstante von $\nabla^2 \tilde{f}$ bezeichnet.
Setze nun

$$\tilde{\varepsilon} := \frac{1}{\tilde{L}}$$

und entsprechend

$$\varepsilon := \|A\| \tilde{\varepsilon}.$$

Das Minimum von $\tilde{f}$ liegt offensichtlich bei $z^* := A^{-1} x^*$. Des Weiteren gilt nach Lemma 4.2.2 angewandt auf $\nabla^2 f(x^*)$, dass

$$\nabla^2 \tilde{f}(z^*) = A \nabla^2 f(x^*) A = I.$$

Bezeichne

$$\tilde{d}^{(k)} := z^{(k+1)} - z^{(k)},$$

$$\begin{aligned}\tilde{y}^{(k)} &:= \nabla\tilde{f}(z^{(k+1)}) - \nabla\tilde{f}(z^{(k)}),\\ \tilde{\varepsilon}^{(k)} &:= \max\{\|z^{(k)} + \tilde{d}^{(k)} - z^*\|, \|z^{(k)} - z^*\|\}.\end{aligned}$$

Unter der Annahme, dass Suchrichtungen nur akzeptiert werden, wenn sie die Zielfunktion verbessern, gilt offensichtlich

$$\tilde{\varepsilon}^{(k)} \leq \tilde{\varepsilon}.$$

Wegen (4.18), angewandt auf $\tilde{f}$, existiert ein $\xi \in U_{\tilde{\varepsilon}}(z^*)$, so dass

$$\begin{aligned}\|\tilde{y}^{(k)} - \tilde{d}^{(k)}\| &\leq \|\nabla^2\tilde{f}(\xi) - \nabla^2\tilde{f}(z^*)\|\|\tilde{d}^{(k)}\|\\ &\leq \tilde{L}\tilde{\varepsilon}^{(k)}\|\tilde{d}^{(k)}\|\end{aligned}$$

und daher

$$\frac{\|\tilde{y}^{(k)} - \tilde{d}^{(k)}\|}{\|\tilde{d}^{(k)}\|} \leq \tilde{L}\tilde{\varepsilon}^{(k)}. \tag{4.19}$$

Es folgt, dass

$$\frac{\tilde{y}^{(k)T}\tilde{d}^{(k)}}{\|\tilde{d}^{(k)}\|^2} = \underbrace{\frac{\tilde{d}^{(k)T}\tilde{d}^{(k)}}{\|\tilde{d}^{(k)}\|^2}}_{=1} + \frac{(\tilde{y}^{(k)} - \tilde{d}^{(k)})^T\tilde{d}^{(k)}}{\|\tilde{d}^{(k)}\|^2} \geq 1 - \frac{\|\tilde{y}^{(k)} - \tilde{d}^{(k)}\|}{\|\tilde{d}^{(k)}\|} \geq 1 - \tilde{L}\tilde{\varepsilon}^{(k)}.$$

Über die geometrische Reihe argumentieren wir, dass für den Kehrwert eine Konstante c_1 existiert, so dass

$$\frac{\|\tilde{d}^{(k)}\|^2}{\tilde{y}^{(k)T}\tilde{d}^{(k)}} \leq \frac{1}{1 - \tilde{L}\tilde{\varepsilon}^{(k)}} = \sum_{i=0}^{\infty}(\tilde{L}\tilde{\varepsilon}^{(k)})^i \leq 1 + c_1\tilde{L}\tilde{\varepsilon}^{(k)}, \tag{4.20}$$

gilt. Hierbei wurde ausgenutzt, dass nach Konstruktion von $\tilde{\varepsilon}$

$$\tilde{L}\tilde{\varepsilon}^{(k)} < 1$$

gilt.
Bezeichne

$$\tilde{H}'^{(k+1)} := \tilde{H}^{(k)} + \frac{\tilde{d}^{(k)}\tilde{d}^{(k)T}}{\tilde{d}^{(k)T}\tilde{d}^{(k)}} - \frac{\tilde{H}^{(k)}\tilde{d}^{(k)}(\tilde{H}^{(k)}\tilde{d}^{(k)})^T}{\tilde{d}^{(k)T}\tilde{H}^{(k)}\tilde{d}^{(k)}}$$

das Update der Funktion $\tilde{f}$ gemäß (4.14). Ausgehend von der gleichen Matrix $\tilde{H}^{(k)}$ untersuchen wir $\|\tilde{H}^{(k+1)} - \tilde{H}'^{(k+1)}\|_F$. Die folgende Rechnung zeigt die Existenz einer Konstanten c_2, so dass

$$\begin{aligned}\|\tilde{H}^{(k+1)} - \tilde{H}'^{(k+1)}\|_F &= \left\|\frac{\tilde{y}^{(k)}\tilde{y}^{(k)T}}{\tilde{y}^{(k)T}\tilde{d}^{(k)}} - \frac{\tilde{d}^{(k)}\tilde{d}^{(k)T}}{\tilde{d}^{(k)T}\tilde{d}^{(k)}}\right\|_F\\ &= \left\|\frac{\left(\tilde{d}^{(k)} + \left(\tilde{y}^{(k)} - \tilde{d}^{(k)}\right)\right)\left(\tilde{d}^{(k)} + \left(\tilde{y}^{(k)} - \tilde{d}^{(k)}\right)\right)^T}{\tilde{y}^{(k)T}\tilde{d}^{(k)}} - \frac{\tilde{d}^{(k)}\tilde{d}^{(k)T}}{\tilde{d}^{(k)T}\tilde{d}^{(k)}}\right\|_F\\ &= \left\|\frac{\tilde{d}^{(k)}\tilde{d}^{(k)T} + \tilde{d}^{(k)}\left(\tilde{y}^{(k)} - \tilde{d}^{(k)}\right)^T}{\tilde{y}^{(k)T}\tilde{d}^{(k)}}\right.\end{aligned}$$

$$
\begin{aligned}
& + \frac{\left(\tilde{y}^{(k)} - \tilde{d}^{(k)}\right)\tilde{d}^{(k)T} + \left(\tilde{y}^{(k)} - \tilde{d}^{(k)}\right)\left(\tilde{y}^{(k)} - \tilde{d}^{(k)}\right)^T}{\tilde{y}^{(k)T}\tilde{d}^{(k)}} - \frac{\tilde{d}^{(k)}\tilde{d}^{(k)T}}{\tilde{d}^{(k)T}\tilde{d}^{(k)}}\Bigg\|_F \\
= \quad & \Bigg\| \underbrace{\frac{\tilde{d}^{(k)T}\tilde{d}^{(k)}}{\tilde{y}^{(k)T}\tilde{d}^{(k)}}}_{\overset{(4.20)}{\leq} 1+c_1\tilde{L}\tilde{\varepsilon}^{(k)}} \left(\frac{\tilde{d}^{(k)}\tilde{d}^{(k)T} + \tilde{d}^{(k)}\left(\tilde{y}^{(k)} - \tilde{d}^{(k)}\right)^T}{\tilde{d}^{(k)T}\tilde{d}^{(k)}} \right. \\
& \left. + \frac{\left(\tilde{y}^{(k)} - \tilde{d}^{(k)}\right)\tilde{d}^{(k)T} + \left(\tilde{y}^{(k)} - \tilde{d}^{(k)}\right)\left(\tilde{y}^{(k)} - \tilde{d}^{(k)}\right)^T}{\tilde{d}^{(k)T}\tilde{d}^{(k)}} \right) - \frac{\tilde{d}^{(k)}\tilde{d}^{(k)T}}{\tilde{d}^{(k)T}\tilde{d}^{(k)}}\Bigg\|_F \\
\leq \quad & \left\| \frac{\tilde{d}^{(k)}\tilde{d}^{(k)T}}{\tilde{d}^{(k)T}\tilde{d}^{(k)}} - \frac{\tilde{d}^{(k)}\tilde{d}^{(k)T}}{\tilde{d}^{(k)T}\tilde{d}^{(k)}} \right\|_F + c_1\tilde{L}\tilde{\varepsilon}^{(k)} \left\| \frac{\tilde{d}^{(k)}\tilde{d}^{(k)T}}{\tilde{d}^{(k)T}\tilde{d}^{(k)}} \right\|_F \\
& + (1 + c_1\tilde{L}\tilde{\varepsilon}^{(k)}) \left(2 \left\| \frac{\tilde{d}^{(k)}\left(\tilde{y}^{(k)} - \tilde{d}^{(k)}\right)^T}{\tilde{d}^{(k)T}\tilde{d}^{(k)}} \right\|_F \right. \\
& \left. + \left\| \frac{\left(\tilde{y}^{(k)} - \tilde{d}^{(k)}\right)\left(\tilde{y}^{(k)} - \tilde{d}^{(k)}\right)^T}{\tilde{d}^{(k)T}\tilde{d}^{(k)}} \right\|_F \right) \\
\overset{\text{Lemma 4.2.7}}{=} \quad & c_1\tilde{L}\tilde{\varepsilon}^{(k)} + (1 + c_1\tilde{L}\tilde{\varepsilon}^{(k)}) \cdot \\
& \left(2\frac{\|\tilde{d}^{(k)}\|\|\tilde{y}^{(k)} - \tilde{d}^{(k)}\|}{\|\tilde{d}^{(k)}\|^2} + \frac{\|\tilde{y}^{(k)} - \tilde{d}^{(k)}\|\|\tilde{y}^{(k)} - \tilde{d}^{(k)}\|}{\|\tilde{d}^{(k)}\|^2} \right) \\
\overset{(4.19)}{\leq} \quad & c_1\tilde{L}\tilde{\varepsilon}^{(k)} + (1 + c_1\tilde{L}\tilde{\varepsilon}^{(k)})(2\tilde{L}\tilde{\varepsilon}^{(k)} + \tilde{L}^2\tilde{\varepsilon}^{(k)2}) \\
\leq \quad & c_2\tilde{\varepsilon}^{(k)}
\end{aligned}
\tag{4.21}
$$

gilt. Zusammen mit Lemma 4.2.6 erhalten wir

$$
\begin{aligned}
\|\tilde{H}^{(k+1)} - \underbrace{\nabla^2\tilde{f}(x^*)}_{=I}\|_F &= \|\tilde{H}^{(k+1)} - I\|_F \\
&\leq \|\tilde{H}^{(k+1)} - \tilde{H}'^{(k+1)}\|_F + \underbrace{\|\tilde{H}'^{(k+1)} - I\|_F}_{\leq \|\tilde{H}^{(k)} - I\|_F} \\
&\leq \|\tilde{H}^{(k)} - I\|_F + c_2\tilde{\varepsilon}^{(k)}.
\end{aligned}
$$

Für den allgemeinen Fall mit der Funktion f bemerken wir zuerst dass

$$
\begin{aligned}
\tilde{d}^{(k)} &= A^{-1}(x^{(k+1)} - x^{(k)}) = A^{-1}d^{(k)}, \\
\tilde{y}^{(k)} &= A(\nabla f(x^{(k+1)}) - \nabla f(x^{(k)})) = Ay^{(k)}, \\
\tilde{\varepsilon}^{(k)} &= \max\{\|A^{-1}x^{(k)} + A^{-1}d^{(k)} - A^{-1}x^*\|, \|A^{-1}x^{(k)} - A^{-1}x^*\|\}.
\end{aligned}
$$

Daraus folgt, dass

$$
\begin{aligned}
AH^{(k+1)}A &= AH^{(k)}A + A\frac{y^{(k)}y^{(k)T}}{y^{(k)T}d^{(k)}}A - A\frac{H^{(k)}d^{(k)}(H^{(k)}d^{(k)})^T}{d^{(k)T}H^{(k)}d^{(k)}}A \\
&= AH^{(k)}A + \frac{Ay^{(k)}(Ay^{(k)})^T}{y^{(k)T}AA^{-1}d^{(k)}} - \frac{AH^{(k)}AA^{-1}d^{(k)}(AH^{(k)}AA^{-1}d^{(k)})^T}{d^{(k)T}A^{-1}AH^{(k)}AA^{-1}d^{(k)}}
\end{aligned}
$$

$$
\begin{aligned}
&= \tilde{H}^{(k)} + \frac{\tilde{y}^{(k)}\tilde{y}^{(k)T}}{\tilde{y}^{(k)T}\tilde{d}^{(k)}} - \frac{\tilde{H}^{(k)}\tilde{d}^{(k)}(\tilde{H}^{(k)}\tilde{d}^{(k)})^T}{\tilde{d}^{(k)T}\tilde{H}^{(k)}\tilde{d}^{(k)}} \\
&= \tilde{H}^{(k+1)}, \quad k = 0, 1, ...
\end{aligned} \tag{4.22}
$$

Da A^{-1} positiv definit ist, definiert die Abbildung

$$\|\cdot\|_{A^{-1}} := \|A^{-1}\cdot\|$$

eine Norm im $\mathbb{R}^n$. Wegen der Normäquivalenz im $\mathbb{R}^n$ existiert ein $\eta_1 > 0$, so dass

$$\tilde{\varepsilon}^{(k)} \leq \eta_1 \max\{\|x^{(k)} + d^{(k)} - x^*\|_{A^{-1}}, \|x^{(k)} - x^*\|_{A^{-1}}\} =: \eta_1\varepsilon^{(k)}$$

und somit ist

$$c_2\tilde{\varepsilon}^{(k)} \leq c_2\eta_1\varepsilon^{(k)} =: c\varepsilon^{(k)}. \tag{4.23}$$

Andererseits definiert

$$\|\cdot\|_{AA} := \|A\cdot A\|_F$$

eine Matrixnorm. Hiermit gilt

$$
\begin{aligned}
\|H^{(k+1)} - \nabla^2 f(x^*)\|_{AA} &= \|A(H^{(k+1)} - \nabla^2 f(x^*))A\|_F \\
&= \|AH^{(k+1)}A - I\|_F \\
&\overset{(4.22)}{=} \|\tilde{H}^{(k+1)} - I\|_F \\
&\leq \|\tilde{H}^{(k)} - I\|_F + c_2\tilde{\varepsilon}^{(k)} \\
&\underset{(4.23)}{\overset{(4.22)}{\leq}} \|AH^{(k)}A - I\|_F + c\varepsilon^{(k)} \\
&= \|A\left(H^{(k)} - \nabla^2 f(x^*)\right)A\|_F + c\varepsilon^{(k)} \\
&= \|H^{(k)} - \nabla^2 f(x^*)\|_{AA} + c\varepsilon^{(k)}.
\end{aligned}
$$

Die Aussage folgt aufgrund der Normäquivalenz. □

Das folgende technische Lemma wird zum Nachweis der (mindestens) linearen Konvergenz benötigt:

Lemma 4.2.9. *Seien $A, B \in \mathbb{R}^{n\times n}$ mit $\|I - BA\| < 1$. Dann sind A und B regulär und es gilt die Abschätzung*

$$\|B^{-1}\| \leq \frac{\|A\|}{1 - \|I - BA\|}.$$

Beweis. Vgl. [34]. Bezeichne

$$M := I - BA.$$

Nach Voraussetzung gilt

$$\|M\| < 1.$$

Für jedes $x \in \mathbb{R}^n \setminus 0$ gilt dann

$$\|(I - M)x\| = \|x - Mx\|$$

$$\begin{aligned} &\geq \|x\| - \|Mx\| \\ &\geq (1 - \|M\|)\|x\|. \end{aligned} \tag{4.24}$$

Wegen

$$1 - \|M\| > 0$$

folgt

$$(I - M)x \neq 0,$$

weswegen $(I - M)$ regulär ist. Für ein beliebiges $y \in \mathbb{R}^n$ setze man

$$x := (I - M)^{-1}y.$$

Dann folgt mit (4.24)

$$\|y\| \geq (1 - \|M\|)\|(I - M)^{-1}y\|.$$

Hieraus folgt

$$\|(I - M)^{-1}\| = \max_{y \neq 0} \frac{\|(I - M^{-1})y\|}{\|y\|} \leq \frac{1}{1 - \|M\|} = \frac{1}{1 - \|I - BA\|}.$$

Nach Definition gilt

$$I - M = BA$$

und somit

$$B^{-1} = A(I - M)^{-1}$$

und daher

$$\|B^{-1}\| \leq \|(I - M)^{-1}\|\|A\| \leq \frac{\|A\|}{1 - \|I - BA\|}.$$

□

Hiermit folgt das

Korollar 4.2.10. *Seien $A, B \in \mathbb{R}^{n \times n}$ mit $\|A - B\| < \beta$ und A invertierbar mit $\|A^{-1}\| \leq \alpha$ mit $\alpha\beta < 1$. Dann ist B regulär und es gilt die Abschätzung*

$$\|B^{-1}\| \leq \frac{\alpha}{1 - \alpha\beta}.$$

Beweis. Vgl. [34]. Es gilt

$$\|I - BA^{-1}\| = \|(A - B)A^{-1}\| \leq \|A - B\|\|A^{-1}\| \leq \beta\alpha < 1.$$

Anwendung von Lemma 4.2.9 liefert die Aussage. □

Aus Satz 4.2.8 ergibt sich der folgende

Satz 4.2.11. *Es existieren ein $\varepsilon > 0$ und ein $\delta > 0$, so dass der Algorithmus 4.2.5 für jeden Startvektor $x^{(0)} \in \mathbb{R}^n$ mit $\|x^{(0)} - x^*\| < \varepsilon$ und jede symmetrische positiv definite Startmatrix $H^{(0)} \in \mathbb{R}^{n \times n}$ mit*

$$\|H^{(0)} - \nabla^2 f(x^*)\| < \delta \tag{4.25}$$

wohldefiniert ist und eine Folge $(x^{(k)})_{k \in \mathbb{N}} \subset \mathbb{R}^n$ erzeugt, die (mindestens) linear gegen x^* konvergiert.

Beweis. Vgl. [48]. Sei $r \in (0,1)$ fest und setze

$$C := \max\left\{\|\nabla^2 f(x^*)\|, \|\nabla^2 f(x^*)^{-1}\|\right\}.$$

Wähle $\varepsilon, \delta \in \mathbb{R}$ so, dass mit c aus (4.23)

$$c \cdot \frac{\varepsilon}{1-r} \leq \delta \tag{4.26}$$

und

$$C(1+r)(L\varepsilon + 2\delta) \leq r \tag{4.27}$$

gilt. Per Induktion zeigen wir, dass

$$\|H^{(k)} - \nabla^2 f(x^*)\| < 2\delta. \tag{4.28}$$

und weisen außerdem die lineare Konvergenz nach.
$\underline{k = 0:}$
(4.28) gilt wegen der Voraussetzung (4.25). (4.27) impliziert, dass

$$2C(1+r)\delta \leq r < 1.$$

Hiermit gilt nach Korollar 4.2.10, dass $H^{(0)}$ regulär ist und dass

$$\|(H^{(0)})^{-1}\| \leq \frac{C}{1 - C \cdot 2\delta} = \frac{(1+r)C}{(1+r) - \underbrace{2C(1+r)\delta}_{\leq r}} \leq (1+r)C. \tag{4.29}$$

Außerdem gilt nach dem Mittelwertsatz

$$\exists \xi \in U_\varepsilon(x^*) : \nabla f(x^{(0)}) - \nabla f(x^*) = \nabla^2 f(\xi)(x^{(0)} - x^*). \tag{4.30}$$

Hiermit folgt

$$\begin{aligned}
\|x^{(1)} - x^*\| &= \|x^{(0)} - (H^{(0)})^{-1}\nabla f(x^{(0)}) - x^*\| \\
&= \|(H^{(0)})^{-1}\left(H^{(0)}(x^{(0)} - x^*) - \left(\nabla f(x^{(0)}) - \nabla f(x^*)\right)\right)\| \\
&\leq \|(H^{(0)})^{-1}\| \cdot \Big(\|H^{(0)}(x^{(0)} - x^*) - \nabla^2 f(x^*)(x^{(0)} - x^*)\| \\
&\quad + \overbrace{\|\underbrace{\nabla f(x^{(0)}) - \nabla f(x^*)}_{\overset{(4.30)}{=}\nabla^2 f(\xi)(x^{(0)} - x^*)} - \nabla^2 f(x^*)(x^{(0)} - x^*)\|}^{\leq L\varepsilon\|x^{(0)} - x^*\|}\Big) \\
&\leq \underbrace{(1+r)C(L\varepsilon + 2\delta)}_{\overset{(4.27)}{\leq} r}\|x^{(0)} - x^*\| \\
&\leq r\|x^{(0)} - x^*\|.
\end{aligned} \tag{4.31}$$

$\underline{k \curvearrowright k+1:}$
Es gelte

$$\|x^{(j+1)} - x^*\| \leq r\|x^{(j)} - x^*\|, \quad j = 1, ..., k \tag{4.32}$$

und

$$\|H^{(k)} - \nabla^2 f(x^*)\| < 2\delta, \quad j = 1, ..., k.$$

Aus Satz 4.2.8 folgt, dass mit hinreichend kleinem $\varepsilon > 0$

$$\begin{aligned} & \|H^{(j+1)} - \nabla^2 f(x^*)\| - \|H^{(j)} - \nabla^2 f(x^*)\| \\ \leq \quad & c \max\{\|x^{(j)} + d^{(j)} - x^*\|, \|x^{(j)} - x^*\|\} \\ \overset{(4.32)}{\leq} \quad & c\varepsilon r^j \end{aligned}$$

gilt.
Summiert man beide Seiten für $j = 0, ..., k$ auf, erhält man wegen der Eigenschaften der geometrischen Reihe

$$\|H^{(k+1)} - \nabla^2 f(x^*)\| \leq \|H^{(0)} - \nabla^2 f(x^*)\| + c \cdot \frac{\varepsilon}{1-r} \overset{(4.26)}{\leq} 2\delta,$$

was (4.28) beweist. In gleicher Weise wie in (4.29) kann gefolgert werden, dass $H^{(k+1)}$ regulär ist und dass

$$\|(H^{(k+1)})^{-1}\| \leq (1+r)C \tag{4.33}$$

gilt. Die Regularität von allen $H^{(k)},\ k \in \mathbb{N}_0$ impliziert die Wohldefiniertheit des Verfahrens. Außerdem erhalten wir durch (4.33) in analoger Weise wie in (4.31)

$$\begin{aligned} \|x^{(k+2)} - x^*\| &= \|x^{(k+1)} + (H^{(k+1)})^{-1}\nabla f(x^{(k+1)}) - x^*\| \\ &\leq \|(H^{(k+1)})^{-1}\| \cdot \Big(\|H^{(k+1)}(x^{(k+1)} - x^*) - \nabla^2 f(x^*)(x^{(k+1)} - x^*)\| \\ &\quad + \overbrace{\|\nabla f(x^{(k+1)}) - \nabla f(x^*) - \nabla^2 f(x^*)(x^{(k+1)} - x^*)\|}^{\leq L\varepsilon\|x^{(k+1)} - x^*\|}\Big) \\ &\leq (1+r)C(L\varepsilon + 2\delta)\|x^{(k+1)} - x^*\| \\ &\leq r\|x^{(k+1)} - x^*\|. \end{aligned}$$

□

Zum Nachweis der superlinearen Konvergenz benötigen wir das folgende

Lemma 4.2.12. *Es existieren ein $\varepsilon > 0$ und ein $\delta > 0$, so dass der Algorithmus 4.2.5 für jeden Startvektor $x^{(0)} \in \mathbb{R}^n$ mit*

$$\|x^{(0)} - x^*\| < \varepsilon$$

und jede symmetrische positiv definite Startmatrix $H^{(0)} \in \mathbb{R}^{n\times n}$ mit

$$\|H^{(0)} - \nabla^2 f(x^*)\| < \delta$$

eine Folge von Matrizen $(H^{(k)})_{k\in\mathbb{N}} \subset \mathbb{R}^{n\times n}$ erzeugt, für die gilt:

$$\lim_{k\to\infty} \frac{\|(H^{(k)} - \nabla^2 f(x^*))d^{(k)}\|}{\|d^{(k)}\|} = 0.$$

Beweis. Vgl. [48]. Wie im Beweis von Satz 4.2.8 gesehen, genügt es, den Fall

$$\nabla^2 f(x^*) = I$$

zu betrachten. Der allgemeine Fall kann wieder mit einer Transformation gefolgert werden. Wegen (4.28) mit Dreiecksungleichung beziehungsweise (4.33) sind die Matrizen $H^{(k)}$ und $(H^{(k)})^{-1}$ gleichmäßig beschränkt. Aus (4.21) und (4.23) folgt

$$\begin{aligned}\sum_{k=0}^{\infty}\left(\|H^{(k)} - I\|_F^2 - \|H'^{(k+1)} - I\|_F^2\right) &\leq \sum_{k=0}^{\infty}\left(\|H^{(k)} - I\|_F^2 - \|H^{(k+1)} - I\|_F^2 + c\varepsilon^{(k)}\right)\\ &\leq \|H^{(0)} - I\|_F^2 + c\sum_{k=0}^{\infty}\varepsilon^{(k)}\\ &< \infty.\end{aligned}$$

Daher muss

$$\lim_{k\to\infty}\left(\|H^{(k)} - I\|_F^2 - \|H'^{(k+1)} - I\|_F^2\right) = 0 \tag{4.34}$$

gelten. Wegen der Gleichheit (4.15) folgt aus (4.34), dass

$$\lim_{k\to\infty}\frac{d^{(k)T}H^{(k)}H^{(k)}d^{(k)}}{d^{(k)T}H^{(k)}d^{(k)}} = 1 \tag{4.35}$$

und

$$\lim_{k\to\infty}\frac{d^{(k)T}H^{(k)}H^{(k)}H^{(k)}d^{(k)}}{d^{(k)T}H^{(k)}d^{(k)}} = 1. \tag{4.36}$$

Somit gilt auch

$$\begin{aligned}&\lim_{k\to\infty}\frac{\|H^{(k)\frac{1}{2}}(H^{(k)} - I)d^{(k)}\|}{\|H^{(k)\frac{1}{2}}d^{(k)}\|}\\ =\ &\lim_{k\to\infty}\frac{\sqrt{\left(H^{(k)\frac{1}{2}}(H^{(k)} - I)d^{(k)}\right)^T H^{(k)\frac{1}{2}}(H^{(k)} - I)d^{(k)}}}{\sqrt{(H^{(k)\frac{1}{2}}d^{(k)})^T H^{(k)\frac{1}{2}}d^{(k)}}}\\ =\ &\lim_{k\to\infty}\sqrt{\frac{\left(d^{(k)T}H^{(k)}H^{(k)\frac{1}{2}} - d^{(k)T}H^{(k)\frac{1}{2}}\right)\left(H^{(k)\frac{1}{2}}H^{(k)}d^{(k)} - H^{(k)\frac{1}{2}}d^{(k)}\right)}{d^{(k)T}H^{(k)\frac{1}{2}}H^{(k)\frac{1}{2}}d^{(k)}}}\\ =\ &\lim_{k\to\infty}\sqrt{\frac{d^{(k)T}H^{(k)}H^{(k)}H^{(k)}d^{(k)} - 2d^{(k)T}H^{(k)}H^{(k)}d^{(k)} + d^{(k)T}H^{(k)}d^{(k)}}{d^{(k)T}H^{(k)}d^{(k)}}}\\ \overset{(4.35)}{\underset{(4.36)}{=}}\ &\sqrt{1-2+1}\\ =\ &0\end{aligned}$$

Wegen der Beschränktheit und der positiven Definitheit von $(H^{(k)})_{k\in\mathbb{N}}$ induziert $\|H^{(k)\frac{1}{2}}\cdot\|$ eine Norm im $\mathbb{R}^n$. Wegen der Normäquivalenz der Normen im $\mathbb{R}^n$ folgt die Behauptung. □

Hieraus folgt unmittelbar die superlineare Konvergenz:

Satz 4.2.13. *Es existieren ein $\varepsilon > 0$ und ein $\delta > 0$, so dass der Algorithmus 4.2.5 für jeden Startvektor $x^{(0)} \in \mathbb{R}^n$ mit $\|x^{(0)} - x^*\| < \varepsilon$ und jede symmetrische positiv definite Startmatrix $H^{(0)} \in \mathbb{R}^{n\times n}$ mit*

$$\|H^{(0)} - \nabla^2 f(x^*)\| < \delta \tag{4.37}$$

eine Folge $(x^{(k)})_{k\in\mathbb{N}} \subset \mathbb{R}^n$ erzeugt, die superlinear gegen x^* konvergiert.

Beweis. Vgl. [24]. Die (lineare) Konvergenz wurde bereits in Satz 4.2.11 gezeigt. Es bleibt lediglich die Superlinearität nachzuweisen, das heißt dass

$$\lim_{k\to\infty} \frac{\|x^{(k+1)} - x^*\|}{\|x^{(k)} - x^*\|} = 0.$$

Sei

$$\varepsilon^{(k)} := \|x^{(k)} - x^*\|.$$

Nach dem Mittelwertsatz existiert ein $\xi^{(k)} \in U_{\varepsilon^{(k)}}(x^*)$, so dass

$$\begin{aligned}
\big(H^{(k)} - \nabla^2 f(x^*)\big)d^{(k)} &= -\nabla f(x^{(k)}) - \nabla^2 f(x^*)d^{(k)} \\
&= \nabla f(x^{(k+1)}) - \nabla f(x^{(k)}) - \nabla^2 f(x^*)d^{(k)} - \nabla f(x^{(k+1)}) \\
&= \nabla^2 f(\xi^{(k)})d^{(k)} - \nabla^2 f(x^*)d^{(k)} - \nabla f(x^{(k+1)}).
\end{aligned} \tag{4.38}$$

Offensichtlich gilt auch

$$\lim_{k\to\infty} \xi^{(k)} = x^*. \tag{4.39}$$

Nach Lemma 4.2.12 gilt

$$\lim_{k\to\infty} \frac{\|(H^{(k)} - \nabla^2 f(x^*))d^{(k)}\|}{\|d^{(k)}\|} = 0.$$

woraus

$$\begin{aligned}
\lim_{k\to\infty} \frac{\|\nabla f(x^{(k+1)})\|}{\|d^{(k)}\|} &\leq \lim_{k\to\infty} \frac{\|\big(\nabla^2 f(\xi^{(k)}) - \nabla^2 f(x^*)\big)d^{(k)}\|}{\|d^{(k)}\|} \\
&\quad + \lim_{k\to\infty} \underbrace{\frac{\|\big(\nabla^2 f(\xi^{(k)}) - \nabla^2 f(x^*)\big)d^{(k)} - \nabla f(x^{(k+1)})\|}{\|d^{(k)}\|}}_{\overset{(4.38)}{=} \lim_{k\to\infty} \frac{(H^{(k)} - \nabla^2 f(x^*))d^{(k)}}{\|d^{(k)}\|} = 0} \\
&\leq \lim_{k\to\infty} \|\nabla^2 f(\xi^{(k)}) - \nabla^2 f(x^*)\| \\
&\leq L \lim_{k\to\infty} \|\xi^{(k)} - x^*\| \\
&\overset{(4.39)}{\leq} 0
\end{aligned} \tag{4.40}$$

folgt. Da $\nabla^2 f(x^*)$ nicht singulär ist, existiert ein $\hat{\xi} \in U_\varepsilon(x^*)$ und ein $\beta > 0$, so dass

$$\begin{aligned}
\|\nabla f(x^{(k+1)})\| &= \|\nabla f(x^{(k+1)}) - \nabla f(x^*)\| \\
&= \|\nabla^2 f(\hat{\xi})(x^{(k+1)} - x^*)\| \\
&\geq \beta \|x^{(k+1)} - x^*\|.
\end{aligned}$$

Daher gilt die Abschätzung

$$\frac{\|\nabla f(x^{(k+1)})\|}{\|x^{(k+1)} - x^{(k)}\|} \geq \frac{\beta\|x^{(k+1)} - x^*\|}{\|x^{(k+1)} - x^*\| + \|x^{(k)} - x^*\|} = \beta\frac{\rho^{(k)}}{1 + \rho^{(k)}},$$

mit

$$\rho^{(k)} := \frac{\|x^{(k+1)} - x^*\|}{\|x^{(k)} - x^*\|}.$$

Aus (4.40) folgt, dass

$$\lim_{k\to\infty} \frac{\rho^{(k)}}{1 + \rho^{(k)}} = 0$$

und somit auch

$$\lim_{k\to\infty} \rho^{(k)} = 0,$$

was den Beweis abschließt. □

4.3 Existierende dünnbesetzte BFGS-Verfahren

Dem Problem der dichten BFGS-Approximation tragen so genannte *Limited-Memory-BFGS-Verfahren* Rechnung. Ein solches wird beispielsweise von Nocedal [64] eingeführt. Hierzu betrachten wir zunächst das *Inverse BFGS-Verfahren.*

Satz 4.3.1. *Ausgehend von der Matrix* $B^{(0)} := (H^{(0)})^{-1}$ *erzeugt die Updatevorschrift*

$$\begin{aligned} B^{(k+1)} &:= B^{(k)} + \frac{(d^{(k)} - B^{(k)}y^{(k)})d^{(k)T} + d^{(k)}(d^{(k)} - B^{(k)}y^{(k)})^T}{y^{(k)T}d^{(k)}} \\ &\quad - \frac{(d^{(k)} - B^{(k)}y^{(k)})^T y^{(k)} d^{(k)} d^{(k)T}}{(y^{(k)T}d^{(k)})^2}, \quad k = 0, 1, ... \end{aligned} \tag{4.41}$$

die Inverse von H^{k+1} *aus (4.6).*

Beweis.

$$\begin{aligned} B^{(k+1)} \cdot H^{(k+1)} &= \left(B^{(k)} + \frac{(d^{(k)} - B^{(k)}y^{(k)})d^{(k)T} + d^{(k)}(d^{(k)} - B^{(k)}y^{(k)})^T}{y^{(k)T}d^{(k)}} \right. \\ &\quad \left. - \frac{(d^{(k)} - B^{(k)}y^{(k)})^T y^{(k)} d^{(k)} d^{(k)T}}{(y^{(k)T}d^{(k)})^2} \right) \\ &\quad \cdot \left(H^{(k)} + \frac{y^{(k)}y^{(k)T}}{y^{(k)T}d^{(k)}} - \frac{(H^{(k)}d^{(k)})(H^{(k)}d^{(k)})^T}{d^{(k)T}H^{(k)}d^{(k)}} \right) \\ &= \underbrace{B^{(k)}H^{(k)}}_{=I} + \frac{B^{(k)}y^{(k)}y^{(k)T}}{y^{(k)T}d^{(k)}} - \frac{d^{(k)}(H^{(k)}d^{(k)})^T}{d^{(k)T}H^{(k)}d^{(k)}} \\ &\quad + \frac{(d^{(k)} - B^{(k)}y^{(k)})d^{(k)T}H^{(k)} + d^{(k)}(d^{(k)} - B^{(k)}y^{(k)})^T H^{(k)}}{y^{(k)T}d^{(k)}} \\ &\quad + \frac{(d^{(k)} - B^{(k)}y^{(k)})d^{(k)T}y^{(k)}y^{(k)T} + d^{(k)}(d^{(k)} - B^{(k)}y^{(k)})^T y^{(k)}y^{(k)T}}{(y^{(k)T}d^{(k)})^2} \\ &\quad - \frac{(d^{(k)} - B^{(k)}y^{(k)})d^{(k)T}H^{(k)}d^{(k)}d^{(k)T}H^{(k)}}{y^{(k)T}d^{(k)}d^{(k)T}H^{(k)}d^{(k)}} \\ &\quad - \frac{d^{(k)}(d^{(k)} - B^{(k)}y^{(k)})^T H^{(k)}d^{(k)}d^{(k)T}H^{(k)}}{y^{(k)T}d^{(k)}d^{(k)T}H^{(k)}d^{(k)}} \\ &\quad - \frac{(d^{(k)} - B^{(k)}y^{(k)})^T y^{(k)}d^{(k)}d^{(k)T}H^{(k)}}{(y^{(k)T}d^{(k)})^2} - \frac{(d^{(k)} - B^{(k)}y^{(k)})^T y^{(k)}d^{(k)}y^{(k)T}}{(y^{(k)T}d^{(k)})^2} \\ &\quad + \frac{(d^{(k)} - B^{(k)}y^{(k)})^T y^{(k)}d^{(k)}d^{(k)T}H^{(k)}}{(y^{(k)T}d^{(k)})^2} \\ &= I + \frac{B^{(k)}y^{(k)}y^{(k)T}}{y^{(k)T}d^{(k)}} - \frac{d^{(k)}(H^{(k)}d^{(k)})^T}{d^{(k)T}H^{(k)}d^{(k)}} + \frac{d^{(k)}(H^{(k)}d^{(k)})^T}{y^{(k)T}d^{(k)}} \\ &\quad - \frac{B^{(k)}y^{(k)}(H^{(k)}d^{(k)})^T}{y^{(k)T}d^{(k)}} + \frac{d^{(k)}(H^{(k)}d^{(k)})^T}{y^{(k)T}d^{(k)}} - \frac{d^{(k)}y^{(k)T}}{y^{(k)T}d^{(k)}} + \frac{d^{(k)}y^{(k)T}}{y^{(k)T}d^{(k)}} \\ &\quad - \frac{B^{(k)}y^{(k)}y^{(k)T}}{y^{(k)T}d^{(k)}} + \frac{d^{(k)}y^{(k)T}}{y^{(k)T}d^{(k)}} - \frac{d^{(k)}y^{(k)T}B^{(k)}y^{(k)}y^{(k)T}}{(y^{(k)T}d^{(k)})^2} - \frac{d^{(k)}d^{(k)T}H^{(k)}}{y^{(k)T}d^{(k)}} \end{aligned}$$

$$
\begin{aligned}
&+\frac{B^{(k)}y^{(k)}d^{(k)T}H^{(k)}}{y^{(k)T}d^{(k)}} - \frac{d^{(k)}(H^{(k)}d^{(k)})^T}{y^{(k)T}d^{(k)}} + \frac{d^{(k)}(H^{(k)}d^{(k)})^T}{d^{(k)T}H^{(k)}d^{(k)}} - \frac{d^{(k)}(H^{(k)}d^{(k)})^T}{y^{(k)T}d^{(k)}} \\
&+\frac{y^{(k)T}B^{(k)}y^{(k)}d^{(k)}(H^{(k)}d^{(k)})^T}{(y^{(k)T}d^{(k)})^2} - \frac{d^{(k)}y^{(k)T}}{y^{(k)T}d^{(k)}} + \frac{y^{(k)T}B^{(k)}y^{(k)}d^{(k)}y^{(k)T}}{(y^{(k)T}d^{(k)})^2} \\
&+\frac{d^{(k)}(H^{(k)}d^{(k)})^T}{y^{(k)T}d^{(k)}} - \frac{y^{(k)T}B^{(k)}y^{(k)}d^{(k)}(H^{(k)}d^{(k)})^T}{(y^{(k)T}d^{(k)})^2} \\
= \; & I
\end{aligned}
$$

□

Bei Verwendung der Inversen verändert sich das Quasi-Newtonverfahren (4.5) zu

$$d^{(k)} = -B^{(k)}\nabla f(x^{(k)}). \tag{4.42}$$

Die Verwendung der inversen BFGS-Update-Formel eignet sich insofern besser, da in (4.42) kein lineares Gleichungssystem zu lösen ist. Die Bestimmung der Suchrichtung über (4.42) empfiehlt sich jedoch ausschließlich für das BFGS-Verfahren. Wird die Hessematrix beispielsweise über Finite Differenzen approximiert, wäre hierbei deren Inverse zu bestimmen, was deutlich aufwändiger wäre als die Bestimmung der Suchrichtung über die Lösung des Gleichungssystems (4.5). Für eine Lösungssoftware, die flexibel mit den unterschiedlichen Bestimmungsarten der zweiten Ableitung umgehen können soll, empfiehlt sich daher die bislang verwandte Darstellung.
Bezeichne

$$U(d,y,B) = \frac{(d-By)d^T + d(d-By)^T}{y^Td} - \frac{(d-By)^Tydd^T}{(y^Td)^2}$$

den zu addierenden Term beim Inversen BFGS-Update (4.41), so dass

$$B^{(k+1)} = B^{(k)} + U(d^{(k)}, y^{(k)}, B^{(k)})$$

gilt. Die Grundidee des Limited-Memory-BFGS-Verfahrens ist, die Matrix $B^{(k)}$ niemals vollständig zu berechnen und zu speichern. Für ein festes $p \in \mathbb{N}$ werden lediglich

$$d^{(i)}, y^{(i)},\ i = \max(k-p+1;1), ..., k$$

und die Startmatrix $B^{(0)}$ gespeichert. Als Startmatrix verwendet man zumeist eine Diagonalmatrix, beispielsweise die Einheitsmatrix. Die Matrizen $B^{(k)}$, die implizit verwendet werden, ergeben sich aus der Updatevorschrift

$$
\begin{aligned}
\hat{B}^{(\max(k-p+1;1))} &:= B^{(0)}, && (4.43)\\
\hat{B}^{(i+1)} &:= \hat{B}^{(i)} + U(d^{(i)}, y^{(i)}, \hat{B}^{(i)}),\ i = \max(k-p+1;1), ..., k, && (4.44)\\
B^{(k+1)} &:= \hat{B}^{(k+1)}. && (4.45)
\end{aligned}
$$

Um zu erklären, inwiefern diese Matrizen lediglich implizit verwendet werden, benötigen wir das folgende

Lemma 4.3.2. *Das Update (4.41) ergibt sich auch durch die Updatevorschrift*

$$B^{(k+1)} = v_k^T v_{k-1}^T ... v_0^T B^{(0)} v_0 ... v_{k-1} v_k +$$

$$v_k^T v_{k-1}^T ... v_1^T \rho_0 d^{(0)} d^{(0)T} v_1 ... v_{k-1} v_k + ... + v_k^T v_{k-1}^T \rho_{k-2} d^{(k-2)} d^{(k-2)T} v_{k-1} v_k$$
$$+ v_k^T \rho_{k-1} d^{(k-1)} d^{(k-1)T} v_k + \rho_k d^{(k)} d^{(k)T}.$$

mit

$$\begin{aligned} \rho_i &:= \frac{1}{y^{(i)T} d^{(i)}} \\ v_i &:= (I - \rho_i y^{(i)} d^{(i)T}). \end{aligned}$$

Beweis. Vgl. [64]. Es genügt zu zeigen, dass

$$B^{(k+1)} = v_k^T B^{(k)} v_k + \rho_k d^{(k)} d^{(k)T}.$$

In der Tat gilt

$$\begin{aligned} v_k^T B^{(k)} v_k + \rho_k d^{(k)} d^{(k)T} &= (I - \rho_k y^{(k)} d^{(k)T})^T B^{(k)} (I - \rho_k y^{(k)} d^{(k)T}) + \frac{d^{(k)} d^{(k)T}}{y^{(k)T} d^{(k)}} \\ &= B^{(k)} - \frac{\overbrace{(y^{(k)} d^{(k)T})^T B^{(k)}}^{= d^{(k)} y^{(k)T} B^{(k)} = d^{(k)} (B^{(k)} y^{(k)})^T}}{y^{(k)T} d^{(k)}} - \frac{B^{(k)} y^{(k)} d^{(k)T}}{y^{(k)T} d^{(k)}} \\ &\quad + \frac{(y^{(k)} d^{(k)T})^T B^{(k)} y^{(k)} d^{(k)T}}{(y^{(k)T} d^{(k)})^2} + \frac{d^{(k)} d^{(k)T}}{y^{(k)T} d^{(k)}} \\ &= B^{(k)} + \frac{d^{(k)} (d^{(k)} - B^{(k)} y^{(k)})^T}{y^{(k)T} d^{(k)}} + \frac{(d^{(k)} - B^{(k)} y^{(k)}) d^{(k)T}}{y^{(k)T} d^{(k)}} \\ &\quad - \underbrace{\frac{d^{(k)} d^{(k)T}}{y^{(k)T} d^{(k)}}}_{= \frac{d^{(k)T} y^{(k)} d^{(k)} d^{(k)T}}{(y^{(k)T} d^{(k)})^2}} + \frac{\overbrace{(y^{(k)} d^{(k)T})^T B^{(k)} y^{(k)} d^{(k)T}}^{= d^{(k)} y^{(k)T} B^{(k)} y^{(k)} d^{(k)} d^{(k)T} = y^{(k)T} B^{(k)} y^{(k)} d^{(k)} d^{(k)T}}}{(y^{(k)T} d^{(k)})^2} \\ &= B^{(k)} + \frac{(d^{(k)} - B^{(k)} y^{(k)}) d^{(k)T} + d^{(k)} (d^{(k)} - B^{(k)} y^{(k)})^T}{y^{(k)T} d^{(k)}} \\ &\quad - \frac{(d^{(k)} - B^{(k)} y^{(k)})^T y^{(k)} d^{(k)} d^{(k)T}}{(y^{(k)T} d^{(k)})^2} \\ &= B^{(k+1)}. \end{aligned}$$

□

Korollar 4.3.3. *Die durch (4.43)-(4.45) konstruierte Matrix lässt sich folglich durch*

$$\begin{aligned} B^{(k+1)} &= v_k^T v_{k-1}^T ... v_s^T B^{(0)} v_s ... v_{k-1} v_k + v_k^T v_{k-1}^T ... v_{s+1}^T \rho_s d^{(s)} d^{(s)T} v_{s+1} ... v_{k-1} v_k \\ &\quad + ... + v_k^T \rho_{k-1} d^{(k-1)} d^{(k-1)T} v_k + \rho_k d^{(k)} d^{(k)T} \end{aligned} \tag{4.46}$$

mit

$$s := \max(k - p + 1; 0)$$

darstellen.

Die Darstellung (4.46) hat den Vorteil, dass alle Matrizen $B^{(k+1)}$ bestimmt werden können ohne zuvor die Vorgänger $\hat{B}^{(i)},\ i = k-1, k-2, ...$ zu berechnen. Dies ermöglicht es, die Matrizen $B^{(k+1)}$ lediglich implizit zu benutzen, ohne sie vollständig aufzustellen. Aus der Gleichung (4.42), zusammen mit (4.46), ist ersichtlich, dass die Suchrichtung unmittelbar durch

$$\begin{aligned} d^{(k+1)} &= v_k^T v_{k-1}^T ... v_s^T B^{(0)} v_s ... v_{k-1} v_k \nabla f(x^{(k+1)}) \\ &\quad + v_k^T v_{k-1}^T ... v_{s+1}^T \rho_s d^{(s)} d^{(s)T} v_{s+1} ... v_{k-1} v_k \nabla f(x^{(k+1)}) \\ &\quad + ... + v_k^T \rho_{k-1} d^{(k-1)} d^{(k-1)T} v_k \nabla f(x^{(k+1)}) + \rho_k d^{(k)} d^{(k)T} \nabla f(x^{(k+1)}) \end{aligned}$$

berechnet werden kann. Wie bereits erwähnt, wird für die Matrix $B^{(0)}$ häufig die Einheitsmatrix oder zumindest eine Diagonalmatrix verwendet, so dass die Suchrichtung ausschließlich durch einige Vektormultiplikationen bestimmt wird. In [19] findet sich eine gute Übersicht über diverse kompakte Darstellungen unterschiedlicher Update-Formeln, ähnlich wie (4.46). Aufgrund der Konstruktion erhält man in den ersten p Iterationen die gleichen Suchrichtungen wie beim klassischen BFGS-Verfahren. Eine allgemeine Konvergenzaussage lässt sich daraus jedoch nicht ableiten. Der für die Konvergenz wesentliche Satz 4.2.8 lässt sich für den p-ten Schritt nicht nachweisen.
In [56] vergleichen Liu und Nocedal das beschriebene Verfahren mit der Methode von Buckley und LeNir [6] und dem partitionierten BFGS-Verfahren von Griewank und Toint [48], welches in Abschnitt 4.4 ausführlich beschrieben wird. Die Methode von Buckley und LeNir ist eine Mischung aus dem Quasi-Newton-Verfahren und dem Verfahren der Konjugierten Gradienten. Alle drei Verfahren vermeiden die Speicherung der dichten Hesse- bzw. BFGS-Matrix. Liu und Nocedal kommen zu dem Ergebnis, dass das partitionierte BFGS-Verfahren von Griewank und Toint deutlich überlegen ist und dass Nocedals Limited-Memory-BFGS das Verfahren von Buckley und LeNir dominiert. Des Weiteren zeigen Liu und Nocedal, dass das Limited-Memory-BFGS-Verfahren für gleichmäßig konvexe Zielfunktionen linear konvergiert.
June und Hassan präsentieren in [54] ein modifiziertes Limited-Memory-BFGS. Dieses baut auf einer Idee von Yuan [86] auf, der wiederum das klassische BFGS-Update modifiziert hat, um eine etwas schnellere lokale Konvergenz zu erhalten. Diese wird jedoch nicht theoretisch nachgewiesen, sondern lediglich anhand von fünf unterschiedlichen Testbeispielen angedeutet. Diese dort beobachtete Tendenz überträgt sich jedoch auch auf die Limited-Memory-Variante von June und Hassan, die in ihren Testrechnungen ebenfalls eine etwas schnellere Konvergenz im Vergleich zu Nocedals Limited-Memory-BFGS-Verfahren beobachten.
Weitere Ausführungen oder Erweiterungen des Limited-Memory-BFGS-Verfahren finden sich bei Perry [66], Shanno [78], [77], Buckley [6], [7] und Nazareth [62]. Eine allgemeine Konvergenzaussage liefern jedoch auch diese Arbeiten nicht.
Byrd und Nocedal führen in [18] das *Verfahren der reduzierten Hessematrix* für gleichungsbeschränkte Optimierungsprobleme ein. Da die notwendigen Bedingungen zweiter Ordnung (2.9)-(2.12) besagen, dass die Hessematrix der Lagrangefunktion auf dem Kern der Jacobimatrix der (aktiven) Nebenbedingungen zumindest positiv semidefinit ist, während die Hessematrix der Lagrangefunktion insgesamt auch negative Eigenwerte besitzen kann, wird darauf verzichtet, diese Matrix zu approximieren. Stattdessen multipliziert man die Hessematrix der Lagrangefunktion von links und rechts mit der Matrix, deren Spaltenvektoren eine Orthonormalbasis des Kerns der Nebenbedingungen, ausgewertet an der aktuellen Ite-

rierten, bilden. Dies macht die in Abschnitt 4.2 dargelegten Globalisierungsmaßnahmen von Powell überflüssig.
Einen ähnlichen Ansatz verfolgen auch Gill und Leonard in [39] für unbeschränkte Probleme. Auch sie reduzieren die Hessematrix in ähnlicher Weise auf einen Unterraum und verringern somit den Rechen- und Speicheraufwand. Als Fortsetzung dieser Arbeit entwickelten sie in [40] eine Limited-Memory-Variante. Ihre numerischen Ergebnisse zeigen, dass dieses Verfahren konventionellen Limited-Memory-BFGS-Verfahren überlegen ist.
Des Weiteren existieren einige Ansätze, die versuchen, die vorgegebene Hessematrixstruktur bei der Approximation durch ein Update-Verfahren zu erhalten. Toint [82] beschreibt solch ein Verfahren. Die neue Matrix ist dabei in jeder Iteration das Ergebnis eines eigenen quadratischen Minimierungsproblem mit Nebenbedingungen. Eine dieser Nebenbedingungen besagt, dass die Dünnbesetztheit zu erhalten ist. Wie gut sich dieses Verfahren zum Einsatz in der Optimierung eignet, wird in [83] untersucht. Die dort präsentierten numerischen Resultate beruhen jedoch nur auf einer sehr geringen Anzahl von Testproblemen. Des Weiteren sind alle Probleme unrestringiert und gemessen an heutigen Standards niedrigdimensional.
Auch Fletcher [28], [29] kommt zu der Schlussfolgerung, dass das Verfahren in der beschriebenen Form nicht für den Einsatz in der nichtlinearen Optimierung geeignet ist. Die von ihm in diesen Arbeiten selbst entwickelten dünn besetzten Update-Verfahren bezeichnet er jedoch ebenfalls als in dieser Form ungeeignet für den Einsatz in einer Optimierungssoftware.
Herskovits [51] führt die Idee von Toint dahingehend fort, dass er eine Diagonalstruktur vorgibt und die Forderung nach positiven Einträgen auf der Diagonalen als weitere Bedingung aufnimmt, wodurch positive Definitheit erreicht wird, was bei Toint nicht immer der Fall ist. Obwohl Herskovits keine wirklich hochdimensionalen Probleme testet, ist zu beobachten, dass sein Diagonalupdate umso besser abschneidet, je dünner besetzt das Problem ist. Auch in diesen Arbeiten wird kein Konvergenzbeweis geliefert.

4.4 Partitioniertes BFGS-Verfahren

In diesem Abschnitt beschreiben wir das bereits angesprochene Verfahren von Griewank und Toint [48], welches die Dünnbesetztheit der behandelten Optimierungsprobleme berücksichtigt.

4.4.1 Methodik

Das Partitionierte BFGS-Verfahren, das in diesem Abschnitt vorgestellt wird, ist einer der wenigen Ansätze, für die ein Konvergenzbeweis existiert. Leider wird hierbei von Voraussetzungen ausgegangen, auf die sich diese Arbeit nicht beschränken soll. Dennoch dient der Konvergenzbeweis dieses Verfahrens als wichtige Grundlage dazu, die Konvergenz der in den folgenden Abschnitten entwickelten Verfahren nachzuweisen.
Wie bereits erwähnt, haben die hochdimensionalen, nichtlinearen Optimierungsprobleme zumeist eine dünn besetzte Struktur, so dass in der zugehörigen Hessematrix der Funktion f lediglich ein kleiner Teil der Werte nicht identisch Null ist. Ein (konstanter) Nulleintrag in der (i,j)-ten Komponente der Hessematrix bedeutet, dass die Variablen x_i und $x_j,\ i \neq j$ in der Funktion f nur in unterschiedlichen additiven Termen auftreten können.
Die wichtigste der angesprochenen, restriktiven Annahmen dieses Ansatzes ist, dass die Zerlegung von f in die Summe von M zweimal stetig differenzierbaren Unterfunktionen $f_{(p)} : \mathbb{R}^{n_p} \to \mathbb{R}$ bekannt sei, so dass gilt

$$f(x) = \sum_{p=1}^{M} f_{(p)}(x_{I_p}). \tag{4.47}$$

Die Indexmenge $I_p \subset \{1, 2..., n\}$ bezeichne die Indizes der $n_p < n$ (idealerweise $n_p \ll n$) Variablen, von denen die Unterfunktionen $f_{(p)} : \mathbb{R}^{n_p} \to \mathbb{R}$ abhängen. Entsprechend bezeichnet

$$x_{I_p} := (x_i,\ i \in I_p) \tag{4.48}$$

den Vektor x, eingeschränkt auf die Indexmenge I_p.
Um bei dem hier vorgestellten Verfahren die Dünnbesetztheit der (gesamten) approximierten Hessematrix zu erhalten, ist es ratsam, die Anzahl der Variablen n_p, von denen die p-te Unterfunktion abhängt, so gering wie möglich zu wählen und womöglich dadurch eine höhere Anzahl M an zu summierenden Unterfunktionen zu akzeptieren. Es sei darauf hingewiesen, dass die Indexmengen I_p keinesfalls disjunkt sein müssen, ein Sachverhalt, auf den Griewank und Toint nicht hinweisen, der aber von zentraler Bedeutung ist. Es gilt lediglich

$$\left(\nexists p \in \{1, ..., M\} : i \in I_p \wedge j \in I_p\right) \Longrightarrow \frac{\partial^2 f(x)}{\partial x_i \partial x_j} \equiv 0. \tag{4.49}$$

Mit anderen Worten: Der konstante Nulleintrag an der Stelle (i,j) in der Hessematrix bleibt nur dann bei der Approximation der Hessematrix erhalten, wenn x_i und x_j nur in unterschiedlichen Summanden $f_{(p)}$ auftreten. Ist die Anzahl der Summanden jedoch sehr hoch, entsteht ein hoher Speicher- und Rechenaufwand, so dass es sinnvoll sein kann, gewisse Nulleinträge als Nichtnulleinträge zu behandeln und die Dünnbesetztheit demnach nicht vollständig zu erhalten. Darum gilt in (4.49) die Umkehrung nicht notwendigerweise.

Beispiel 4.4.1. Die Funktion $f : \mathbb{R}^4 \to \mathbb{R} : f(x) = x_1^2 + 2x_1 + x_1x_2 + x_2x_3 + x_4^2$ kann als Summe der Funktionen

$$\begin{aligned} f_{(1)} : &\quad \mathbb{R}^2 \to \mathbb{R} : \quad f_{(1)}(x_1, x_2) = x_1^2 + 2x_1 + x_1x_2, \\ f_{(2)} : &\quad \mathbb{R}^2 \to \mathbb{R} : \quad f_{(2)}(x_2, x_3) = x_2x_3, \\ f_{(3)} : &\quad \mathbb{R} \to \mathbb{R} : \quad f_{(3)}(x_4) = x_4^2, \end{aligned}$$

aufgefasst werden. Diese Aufteilung erhält bei der Approximation der Hessematrix die ursprüngliche Dünnbesetztheit. Eine Aufteilung der Form

$$\begin{aligned} f_{(1)} : &\quad \mathbb{R}^3 \to \mathbb{R} : \quad f_{(1)}(x_1, x_2, x_4) = x_1^2 + 2x_1 + x_1x_2 + x_4^2, \\ f_{(2)} : &\quad \mathbb{R}^2 \to \mathbb{R} : \quad f_{(2)}(x_2, x_3) = x_2x_3, \end{aligned}$$

würde zwar die Dünnbesetztheit an den Stellen $(1, 4)$ und $(2, 4)$ zerstören, wäre jedoch genauso zulässig und alle folgenden Konvergenzresultate gelten auch für eine solche Zerlegung.

Der Ansatz von Griewank und Toint sieht vor, für jede der Unterfunktionen ein BFGS-Update durchzuführen. Die Einträge $H_{i,j}$ in der Hessematrix, für die es mehrere Unterfunktion $f_{(p)}$ gibt, so dass

$$\frac{\partial^2 f_{(p)}(x_{I_p})}{\partial x_i \partial x_j} \not\equiv 0$$

ist, werden durch Summation gebildet. Die zur Teilfunktion $f_{(p)}$ gehörige BFGS-Matrix bezeichnen wir mit $H_{(p)}$. Um die Summation der Matrizen $H_{(p)}$ durchführen zu können, wird die $n_p \times n_p$-Matrix $H_{(p)}$ mit Nullen aufgefüllt. Dadurch entstehen die Matrizen

$$(\overline{H}_{(p)})_{i,j} := \begin{cases} (H_{(p)})_{i,j}, & \text{falls } i, j \in I_p \\ 0, & \text{sonst,} \end{cases}$$

so dass $(\overline{H}_{(p)}) \in \mathbb{R}^{n \times n}, p = 1, 2, ..., M$.
Die gesamte Hessematrix $\nabla^2 f(x)$ wird somit durch

$$H^+ = \sum_{p=1}^{M} \overline{H}^+_{(p)} \tag{4.50}$$

approximiert. Zu einem wie in (4.48) erklärten Vektor $x_{I_p} \in \mathbb{R}^{n_p}$ bezeichnet entsprechend $\overline{x_{I_p}} \in \mathbb{R}^{n_p}$ den mit Nullen aufgefüllten Vektor.
Aus den Sätzen 4.2.3 und 4.2.4 folgt, dass die Matrizen $H_{(p)}$ beziehungsweise $H^+_{(p)}$ positiv definit sind, falls $\nabla^2 f_{(p)}(x^*_{I_p}),\ p = 1, ..., M$ positiv definit, das heißt falls die Funktionen $f_{(p)},\ p = 1, ..., M$ streng konvex sind. Der folgende Satz begründet, warum in diesem Fall auch die Matrix (4.50) positiv definit ist.

Satz 4.4.2. *Sei $n \in \mathbb{N}$ und $f : \mathbb{R}^n \to \mathbb{R}$ eine Funktion mit der Darstellung (4.47). Entsprechend dieser Darstellung seien $n_1, n_2, ..., n_M \in \mathbb{N}$ und*

$$\bigcup_{p=1}^{M} I_p = \{1, ..., n\}.$$

Sind $H_{(p)} \in \mathbb{R}^{n_p \times n_p},\ p = 1, ..., M$ positiv definit, so ist auch

$$H := \sum_{p=1}^{M} \overline{H}_{(p)}$$

positiv definit.

Beweis. Wir betrachten zunächst den Fall $M = 2$. Ohne Beschränkung der Allgemeinheit seien die Variablen im $\mathbb{R}^n$ so angeordnet, dass für jeden Vektor $v \in \mathbb{R}^n$ die folgende Unterteilung vorgenommen werden kann:

$$v := \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

mit

$$f(v) = f_{(1)}\left(\begin{bmatrix} x \\ y \end{bmatrix}\right) + f_{(2)}\left(\begin{bmatrix} y \\ z \end{bmatrix}\right),$$

wobei $\begin{bmatrix} x \\ y \end{bmatrix} \in \mathbb{R}^{n_1}$ und $\begin{bmatrix} y \\ z \end{bmatrix} \in \mathbb{R}^{n_2}$ gilt.
Sei

$$v = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \in \mathbb{R}^n \setminus \{0\}$$

beliebig. Dann gilt

$$\begin{aligned} v^T H v &= v^T(\overline{H}_{(1)} + \overline{H}_{(2)})v \\ &= \begin{bmatrix} x & y & z \end{bmatrix} \overline{H}_{(1)} \begin{bmatrix} x \\ y \\ z \end{bmatrix} + \begin{bmatrix} x & y & z \end{bmatrix} \overline{H}_{(2)} \begin{bmatrix} x \\ y \\ z \end{bmatrix} \\ &= \underbrace{\begin{bmatrix} x & y \end{bmatrix} H_{(1)} \begin{bmatrix} x \\ y \end{bmatrix}}_{\geq 0,\ \text{da } H_{(1)} \text{ pos. def.}} + \underbrace{\begin{bmatrix} y & z \end{bmatrix} H_{(2)} \begin{bmatrix} y \\ z \end{bmatrix}}_{\geq 0,\ \text{da } H_{(2)} \text{ pos. def.}} \\ &> 0. \end{aligned}$$

Wegen $v \neq 0$, ist aufgrund der positiven Definitheit von $H_{(1)}$ und $H_{(2)}$ mindestens ein Summand größer als Null.
Die Fälle $M \geq 3$ folgen induktiv, indem $f_{(1)} + f_{(2)}$ (beziehungsweise $f_{(1)} + f_{(2)} + f_{(3)} + ...$) als Funktion $f_{(1)}$ interpretiert wird. □

Satz 4.4.3. *Sind $\nabla^2 f_{(p)}(x^*_{I_p}),\ p = 1, ..., M$ positiv definit. Dann existiert ein $\varepsilon > 0$, so dass für alle $x, x^+ \in U_\varepsilon(x^*)$ die Quasi-Newton-Bedingung*

$$Hd = y$$

gilt.

Beweis. Nach Satz 4.2.4 kann in einer hinreichend kleinen Umgebung $U_\varepsilon(x^*)$ um x^* stets y statt q verwendet werden. H^+ beziehungsweise $H^+_{(p)}$ bezeichne wieder das Update, ausgehend von der Matrix H beziehungsweise $H_{(p)}$. Für das blockweise Update verwenden wir den Vektor

$$y_{(p)} := \nabla f_{(p)}(x^+_{I_p}) - \nabla f_{(p)}(x_{I_p}) \in \mathbb{R}^{n_p},$$

welcher entsprechend ebenfalls nicht durch ein analoges $q_{(p)}$ ersetzt werden muss. Aus Abschnitt 4.2 wissen wir bereits, dass damit wegen der Konstruktion des BFGS-Updates

$$H^+_{(p)} d_{I_p} = y_{(p)}$$

gilt. Demnach gilt auch

$$H^+ d = \Big(\sum_{p=1}^{M} \overline{H^+_{(p)}}\Big) d = \sum_{p=1}^{M} \overline{(H^+_{(p)} d_{I_p})} = \sum_{p=1}^{M} \overline{y_{(p)}} = y.$$

□

Somit erzeugt (4.50) für streng konvexe Teilfunktionen $f_{(p)},\ p = 1, ..., M$ ein dünn besetztes positiv definites Update, welches die Quasi-Newton-Bedingung erfüllt.

Bemerkung 4.4.4. Die Annahme der positiven Definitheit der Unterfunktionen ist sehr viel restriktiver als die Annahme der hinreichenden Optimalitätskriterien, wie sie in Abschnitt 4.2 vorgenommen wurden. Dies stellt eine weitere der restriktiven Annahmen dieses Ansatzes dar.
Laut Griewank und Toint genügt es unter gewissen Voraussetzungen für die positive Definitheit des Updates, dass die Matrizen

$$\nabla^2 f_{(p)}(x^*_{I_p}), \quad p = 1, ..., M$$

lediglich positiv semidefinit sind. Dieses Detail ist jedoch für die weiteren Untersuchungen unerheblich.

Wir gehen nachfolgend wieder davon aus, dass $x^{(0)}$ in der Umgebung $U_\varepsilon(x^*)$ aus Satz 4.4.3 liegt, so dass wir den Vektor $q^{(k)}_{(p)}$, der wie in (4.13) zu berechnen wäre, für keinen der Blöcke benötigen.
Für eine gegebene Zerlegung (4.47) von f mit lokal konvexen Unterfunktion $f_{(p)}$ lautet der Algorithmus demnach wie folgt:

Algorithmus 4.4.5.

i. Wähle $x^{(0)} \in \mathbb{R}^n$ *und* $H_{(p)}^{(0)} \in \mathbb{R}^{n_p \times n_p},\ p = 1, ..., M$ *symmetrisch und positiv definit und setze*

$$H^{(0)} := \sum_{p=1}^{M} \overline{H}_{(p)}^{(0)}.$$

Wähle außerdem $\varepsilon \geq 0$ *und setze* $k := 0$.

ii. Ist $\|\nabla f(x^{(k)})\| \leq \varepsilon$, *STOP.*

iii. Bestimme $d^{(k)}$ *aus dem Gleichungssystem*

$$H^{(k)} d^{(k)} = -\nabla f(x^{(k)})$$

iv. Setze

$$\begin{aligned} x^{(k+1)} &:= x^{(k)} + d^{(k)} \text{ und} \\ y_{(p)}^{(k)} &:= \nabla f_{(p)}(x_{I_p}^{(k+1)}) - \nabla f_{(p)}(x_{I_p}^{(k)}) \end{aligned}$$

v. Berechne

$$H_{(p)}^{(k+1)} = H_{(p)}^{(k)} + \frac{y_{(p)}^{(k)} y_{(p)}^{(k)T}}{y_{(p)}^{(k)T} d_{I_p}^{(k)}} - \frac{(H_{(p)}^{(k)} d_{I_p}^{(k)})(H_{(p)}^{(k)} d_{I_p}^{(k)})^T}{d_{I_P}^{(k)T} H_{(p)}^{(k)} d_{I_p}^{(k)}}, \quad p = 1, ..., M \tag{4.51}$$

und die gesamte Hessematrix

$$H^{(k+1)} = \sum_{p=1}^{M} \overline{H}_{(p)}^{(k+1)}$$

vi. Setze k:=k+1 und gehe zu ii.

4.4.2 Konvergenzeigenschaften des partitionierten BFGS-Verfahrens

Wie in Abschnitt 4.2.2, gehen wir auch hier davon aus, dass aufgrund der hinreichend guten Startschätzungen gemäß Satz 4.4.3 beim Update stets $y_{(p)}^{(k)}$ verwendet werden kann. Des Weiteren habe f in diesem Abschnitt stets die Darstellung (4.47), alle $\nabla^2 f_{(p)},\ p = 1, ..., M$ seien lokal Lipschitz-stetig mit Lipschitz-Konstante L und darüber hinaus seien $\nabla^2 f_{(p)}(x_{I_p}^*),\ p = 1, ..., M$ positiv definit.
Analog zu Satz 4.2.8 gilt:

Satz 4.4.6. *Für die Matrizen* $H_{(p)}^{(k)},\ p = 1, ..., M$ *aus Algorithmus 4.4.5 gibt es für feste Matrixnormen* $\|\cdot\|_p,\ p = 1, ..., M$ *und für ein* $\varepsilon > 0$ *Konstanten* $c_p \in \mathbb{R},\ p = 1, ..., M$, *so dass für die Iterierten* $x_{I_p}^{(k)} + d_{I_p}^{(k)}, x_{I_p}^{(k)} \in U_\varepsilon(x_{I_p}^*)$

$$\|H_{(p)}^{(k+1)} - \nabla^2 f_{(p)}(x_{I_p}^*)\|_p \leq \|H_{(p)}^{(k)} - \nabla^2 f_{(p)}(x_{I_p}^*)\|_p + c_p \max\{\|x_{I_p}^{(k)} + d_{I_p}^{(k)} - x_{I_p}^*\|, \|x_{I_p}^{(k)} - x_{I_p}^*\|\}$$

gilt.

Beweis. Der Beweis erfolgt für alle $p = 1, ..., M$ exakt wie der von Satz 4.2.8, unter Verwendung des alternativen Updates (4.14). □

Bemerkung 4.4.7. Aufgrund von Satz 4.4.2 impliziert die Bedingung, dass $\nabla^2 f_{(p)}(x^*_{i_p}),\ p = 1, ..., M$ positiv definit sind, dass auch $\nabla^2 f(x^*)$ positiv definit ist.
Des Weiteren ist die Funktion f offensichtlich Lipschitz-stetig, wenn alle Funktionen $f_{(p)},\ p = 1, ..., M$ Lipschitz-stetig sind.

Aus Satz 4.4.6 ergibt sich bereits die (mindestens) lineare Konvergenz, deren Beweis sehr ähnlich erfolgt wie der von Satz 4.2.11.

Satz 4.4.8. *Es existieren ein $\varepsilon > 0$ und ein $\delta > 0$, so dass der Algorithmus 4.4.5 für jeden Startvektor $x^{(0)} \in \mathbb{R}^n$ mit $\|x^{(0)} - x^*\| < \varepsilon$ und jede symmetrische positiv definite Startmatrix*

$$H^{(0)} = \sum_{p=1}^{M} \overline{H}^{(0)}_{(p)}$$

mit

$$\|H^{(0)}_{(p)} - \nabla^2 f_{(p)}(x^*_{I_p})\|_p < \delta, \quad p = 1, ..., M \tag{4.52}$$

wohldefiniert ist und eine Folge $(x^{(k)})_{k\in\mathbb{N}} \subset \mathbb{R}^n$ erzeugt, die (mindestens) linear gegen x^* konvergiert.

Beweis. Vgl. [48]. Sei $r \in (0, 1)$ fest und setze

$$C = \max\left\{\|\nabla^2 f(x^*)\|, \|\nabla^2 f(x^*)^{-1}\|\right\}.$$

Wähle $\varepsilon, h \in \mathbb{R}$ so, dass mit c_p aus Satz 4.4.6

$$c_p \frac{\varepsilon}{1-r} \leq \frac{h}{M} =: \delta \tag{4.53}$$

und

$$C(1+r)\big(L\varepsilon + 2\eta h\big) \leq r \tag{4.54}$$

gilt, wobei η eine positive Konstante ist, für die für alle $p = 1, ..., M$

$$\|X\|_2 \leq \eta \|X\|_p, \quad X \in \mathbb{R}^{n_p \times n_p}$$

erfüllt ist. Per Induktion zeigen wir, dass

$$\|H^{(k)}_{(p)} - \nabla^2 f_{(p)}(x^*_{I_p})\|_p < 2\delta \tag{4.55}$$

und weisen außerdem die lineare Konvergenz nach.
$\underline{k = 0:}$
(4.55) gilt wegen der Voraussetzung (4.52). Genauso gilt wegen (4.52)

$$\|H^{(0)} - \nabla^2 f(x^*)\| \;=\; \left\|\sum_{p=1}^{M}\Big(H^{(0)}_{(p)} - \nabla^2 f_{(p)}(x^*_{I_p})\Big)\right\|$$

$$\begin{aligned} &\leq \eta \sum_{p=1}^{M} \left(\left\| (H_{(p)}^{(0)} - \nabla^2 f_{(p)}(x^*_{I_p})) \right\|_p \right) \\ &\leq \eta h \leq 2\eta h. \end{aligned}$$

(4.54) impliziert, dass

$$2C(1+r)\eta h \leq r < 1.$$

Hiermit gilt nach Korollar 4.2.10, dass $H^{(0)}$ regulär ist und dass

$$\|(H^{(0)})^{-1}\| \leq \frac{C}{1 - C \cdot 2\eta h} = \frac{(1+r)C}{(1+r) - \underbrace{2C(1+r)\eta h}_{\leq r}} \leq (1+r)C. \tag{4.56}$$

Außerdem gilt nach dem Mittelwertsatz

$$\exists \xi \in U_\varepsilon(x^*) : \nabla f(x^{(0)}) - \nabla f(x^*) = \nabla^2 f(\xi)(x^{(0)} - x^*). \tag{4.57}$$

Hiermit folgt

$$\begin{aligned} \|x^{(1)} - x^*\| &= \|x^{(0)} - (H^{(0)})^{-1}\nabla f(x^{(0)}) - x^*\| \\ &= \|(H^{(0)})^{-1}\big(H^{(0)}(x^{(0)} - x^*) - (\nabla f(x^{(0)}) - \nabla f(x^*))\big)\| \\ &\leq \|(H^{(0)})^{-1}\|\big(\|H^{(0)}(x^{(0)} - x^*) - \nabla^2 f(x^*)(x^{(0)} - x^*)\| \\ &\quad + \overbrace{\|\underbrace{\nabla f(x^{(0)}) - \nabla f(x^*)}_{\overset{(4.57)}{=}\nabla^2 f(\xi)(x^{(0)} - x^*)} - \nabla^2 f(x^*)(x^{(0)} - x^*)\|}^{\leq L\varepsilon\|x^{(0)} - x^*\|}\big) \\ &\leq \underbrace{(1+r)C\big(L\varepsilon + 2\eta h\big)}_{\overset{(4.54)}{\leq} r} \|x^{(0)} - x^*\| \\ &\leq r\|x^{(0)} - x^*\| \end{aligned} \tag{4.58}$$

$\underline{k \curvearrowright k+1:}$
Es gelte

$$\|x^{(j+1)} - x^*\| \leq r\|x^{(j)} - x^*\|, \quad j = 1, ..., k \tag{4.59}$$

und

$$\|H_{(p)}^{(k)} - \nabla^2 f_{(p)}(x^*_{I_p})\|_p < \frac{2h}{M}, \quad p = 1, ..., M.$$

Aus Satz 4.2.8 folgt, dass

$$\begin{aligned} &\|H_{(p)}^{(j+1)} - \nabla^2 f_{(p)}(x^*_{I_p})\|_p - \|H_{(p)}^{(j)} - \nabla^2 f_{(p)}(x^*_{I_p})\|_p \\ \leq\ & c_p \max\{\|x^{(j)} + d^{(j)} - x^*\|, \|x^{(j)} - x^*\|\} \\ \overset{(4.59)}{\leq}\ & c_p \varepsilon r^j \end{aligned}$$

gilt. Summiert man beide Seiten für $j = 0, ..., k-1$ auf, erhält man wegen der Eigenschaften der geometrischen Reihe

$$\|H_{(p)}^{(k+1)} - \nabla^2 f_{(p)}(x^*_{I_p})\|_p \leq \|H_{(p)}^{(0)} - \nabla^2 f_{(p)}(x^*_{I_p})\|_p + c_p \frac{\varepsilon}{1-r} \overset{(4.53)}{\leq} \frac{2h}{M} = 2\delta,$$

was (4.55) beweist. Außerdem folgt hieraus, dass

$$\|H^{(k+1)} - \nabla^2 f(x^*)\| \leq \eta M \frac{2h}{M} \leq 2\eta h, \tag{4.60}$$

womit in gleicher Weise wie in (4.56) gefolgert werden kann, dass $H^{(k+1)}$ regulär ist und dass

$$\|(H^{(k+1)})^{-1}\| \leq (1+r)C. \tag{4.61}$$

Wir erhalten durch (4.61) in analoger Weise wie in (4.58)

$$\begin{aligned}
\|x^{(k+2)} - x^*\| &= \|x^{(k+1)} + (H^{(k+1)})^{-1}\nabla f(x^{(k+1)}) - x^*\| \\
&\leq \|(H^{(k+1)})^{-1}\| \cdot \big(\|H^{(k+1)}(x^{(k+1)} - x^*) - \nabla^2 f(x^*)(x^{(k+1)} - x^*)\| \\
&\quad + \overbrace{\|\nabla f(x^{(k+1)}) - \nabla f(x^*) - \nabla^2 f(x^*)(x^{(k+1)} - x^*)\|}^{\leq L\varepsilon\|x^{(k+1)} - x^*\|}\big) \\
&\leq (1+r)C\big(L\varepsilon + 2\eta h\big)\|x^{(k+1)} - x^*\| \\
&\leq r\|x^{(k+1)} - x^*\|.
\end{aligned}$$

□

Zum Nachweis der superlinearen Konvergenz benötigen wir das zu Lemma 4.2.12 analoge

Lemma 4.4.9. *Es existieren ein $\varepsilon > 0$ und ein $\delta > 0$, so dass der Algorithmus 4.4.5 für jeden Startvektor $x^{(0)} \in \mathbb{R}^n$ mit*

$$\|x^{(0)} - x^*\| < \varepsilon$$

und jede symmetrische positiv definite Startmatrix

$$H^{(0)} = \sum_{p=1}^{M} \overline{H}^{(0)}_{(p)}$$

mit

$$\|H^{(0)}_{(p)} - \nabla^2 f_{(p)}(x^*_{I_p})\|_p < \delta, \quad p = 1, ..., M$$

eine Folge von Matrizen $(H^{(k)})_{k\in\mathbb{N}} \subset \mathbb{R}^{n\times n}$ erzeugt, für die gilt:

$$\lim_{k\to\infty} \frac{\|(H^{(k)} - \nabla^2 f(x^*))d^{(k)}\|}{\|d^{(k)}\|} = 0.$$

Beweis. Wegen

$$\begin{aligned}
\lim_{k\to\infty} \frac{\|(H^{(k)} - \nabla^2 f(x^*))d^{(k)}\|}{\|d^{(k)}\|} &= \lim_{k\to\infty} \frac{\left\|\sum_{p=1}^{M}(H^{(k)}_{(p)} - \nabla^2 f_{(p)}(x^*_{I_p}))d^{(k)}_p\right\|}{\|d^{(k)}\|} \\
&\leq \lim_{k\to\infty} \sum_{p=1}^{M} \frac{\|(H^{(k)}_{(p)} - \nabla^2 f_{(p)}(x^*_{I_p}))d^{(k)}_p\|}{\underbrace{\|d^{(k)}\|}_{\geq\|d^{(k)}_p\|}}
\end{aligned}$$

$$\leq \sum_{p=1}^{M} \lim_{k\to\infty} \frac{\|(H_{(p)}^{(k)} - \nabla^2 f_{(p)}(x_{I_p}^*))d_p^{(k)}\|}{\|d_p^{(k)}\|}$$

genügt es,

$$\lim_{k\to\infty} \frac{\|(H_{(p)}^{(k)} - \nabla^2 f_{(p)}(x_{I_p}^*))d_p^{(k)}\|}{\|d_p^{(k)}\|} = 0$$

zu zeigen. Daher kann der Index p hier weggelassen werden und ohne Beschränkung der Allgemeinheit von einer voll besetzten Hessematrix $\nabla^2 f(x^*)$ ausgegangen werden. Daher liefert das Lemma 4.2.12 bereits die Aussage. □

Hiermit folgt nun unmittelbar die superlineare Konvergenz:

Satz 4.4.10. *Es existieren ein $\varepsilon > 0$ und ein $\delta > 0$, so dass der Algorithmus 4.4.5 für jeden Startvektor $x^{(0)} \in \mathbb{R}^n$ mit $\|x^{(0)} - x^*\| < \varepsilon$ und jede symmetrische positiv definite Startmatrix*

$$H^{(0)} = \sum_{p=1}^{M} \overline{H}_{(p)}^{(0)}$$

mit

$$\|H_{(p)}^{(0)} - \nabla^2 f_{(p)}(x_{I_p}^*)\|_p < \delta, \quad p = 1, ..., M$$

eine Folge $(x^{(k)})_{k\in\mathbb{N}} \subset \mathbb{R}^n$ erzeugt, die superlinear gegen x^* konvergiert.

Beweis. Die (lineare) Konvergenz wurde bereits in Satz 4.4.8 gezeigt. Der Beweis der Superlinearität erfolgt exakt wie im Beweis von Satz 4.2.13, denn aufgrund der Verwendung von Lemma 4.4.9 spielt die Darstellung (4.47) an dieser Stelle keine Rolle und kann daher außer Acht gelassen werden. □

4.5 Block-BFGS

In dieser Arbeit wird davon ausgegangen, dass die Funktion f (beziehungsweise L) nicht in detaillierter Form bekannt ist. Es sei lediglich die dünnbesetzte Struktur der ersten und zweiten Ableitung bekannt. Demnach ist es im Allgemeinen nicht möglich, die einzelnen Funktionen $f_{(p)}$ aus (4.47) zu ermitteln oder einzeln auszuwerten. Aus der Struktur kann jedoch zumindest auf die Existenz einer solchen Zerlegung zurückgeschlossen werden, obwohl sie nicht in exakter Form bekannt ist.

4.5.1 Blockmatrizen ohne Überschneidungen

Eine Ausnahme bietet der Fall, in dem die Struktur der Hessematrix durch eine Menge sich nicht überschneidender Blöcke überdeckt werden kann. Die Hessematrix habe beispielsweise die folgende Struktur:

$$\nabla^2 f(x) = \begin{bmatrix} \times & \times & \times & & \\ \times & \times & & & \\ \times & & \times & & \\ & & & \times & \times \\ & & & \times & \times \end{bmatrix}.$$

Diese Matrix lässt sich durch einen 3×3- und einen 2×2-Block überdecken. Die Nulleinträge $(2,3)$ und $(3,2)$ werden hier als Nichtnulleinträge behandelt. Hieraus lassen sich zwar die zwei Teilfunktionen $f_{(1)}$ und $f_{(2)}$ nicht unmittelbar identifizieren, wesentlich ist jedoch laut Algorithmus 4.4.5 lediglich die Kenntnis von $\nabla f_{(1)}$ und $\nabla f_{(2)}$. Diese Größen sind in diesem Fall jedoch leicht zu extrahieren, da die beiden Funktionen von vollständig unterschiedlichen Variablen abhängen. Es gilt:

$$\nabla f_{(p)}(x_{I_p}) = (\nabla f(x))_{I_p}, \quad p = 1, 2.$$

In diesen Fällen gelingt es folglich, das Verfahren aus Abschnitt 4.4 durchzuführen, obwohl die Teilfunktionen an sich nicht bekannt sind. Es ist hierbei natürlich im Sinne der Effizienz des Verfahrens wünschenswert, möglichst wenig Nulleinträge als Nichtnulleinträge zu behandeln. Insofern ist es oft hilfreich, die Variablen auf geeignete Weise zu permutieren, so dass möglichst kleine Blöcke entstehen und folglich die Block-Überdeckung möglichst "scharf" ist.
Eine weitere, jedoch heuristische Methode besteht darin, im Sinne einer hohen Dünnbesetztheit einzelne Nichtnulleinträge als Nulleinträge zu behandeln. Wie wir in Abschnitt 5.2 sehen werden, führt dieses Verfahren in vielen Fällen zu überraschend guten Ergebnissen. Hierzu erscheint eine theoretische Konvergenzaussage jedoch nicht möglich, da der wegzulassende Eintrag unter Umständen so dominant für die Krümmung der Zielfunktion sein kann, dass das Verfahren sehr langsam oder überhaupt nicht konvergiert.

4.5.2 Einfache Überschneidungen

4.5.2.1 Methodik

In diesem Abschnitt wird eine Block-Struktur für die approximierte Hessematrix festgelegt, so dass alle oder zumindest fast alle tatsächlichen Einträge der Matrix innerhalb dieser

Struktur liegen. Die Blockgröße n_p des p-ten Blocks kann abgesehen davon beliebig gewählt werden. Die Blöcke dürfen sich überschneiden und paarweise unterschiedliche Größe haben. Eine Überschneidung von mehr als 2 Blöcken sei zunächst ausgeschlossen. Aufgrund der Symmetrie sind Blöcke und Überschneidungen jedoch stets quadratisch. Demnach hat H die Form

$$H = \begin{bmatrix} \begin{bmatrix} H_1 & \\ & \end{bmatrix} & & & & & & \\ & \begin{bmatrix} \begin{bmatrix} H_2 & \\ & \end{bmatrix} & \\ & \end{bmatrix} & & & & & \\ & & \ddots & & & & \\ & & & \ddots & & & \\ & & & & \ddots & & \\ & & & & & \begin{bmatrix} \begin{bmatrix} & \\ & H_M \end{bmatrix} & \\ & \end{bmatrix} \end{bmatrix} \in \mathbb{R}^{n\times n}. \tag{4.62}$$

Die Indizierung der Elemente der Blöcke $H_p \in \mathbb{R}^{n_p\times n_p},\ p = 1, ..., M$ erfolge über die Indizes bezüglich der Gesamtmatrix. Das heißt, dass der Eintrag $(1, 1)$, wie in (4.62) zu erkennen ist, ausschließlich für den Block H_1 erklärt ist. Einträge (i, j) innerhalb einer Überschneidung sind folglich für beide Blöcke dieser Überschneidung erklärt.
Die Indexmengen

$$\begin{aligned} I_0 &:= \emptyset =: I_{M+1} \\ I_p &:= \Big\{ i \in \{1, ..., n\} : \quad (H_p)_{i,i} \text{ ist erklärt} \Big\} \\ I_{p_1\cap p_2} &:= I_{p_1} \cap I_{p_2}, \\ I_{p_1\cap p_2\cap p_3} &:= I_{p_1} \cap I_{p_2} \cap I_{p_3}, \\ I_{p_1\setminus p_2} &:= I_{p_1} \setminus I_{p_2}, \\ I_{p_1\setminus p_2\setminus p_3} &:= (I_{p_1} \setminus I_{p_2}) \setminus I_{p_3}, \end{aligned} \tag{4.63}$$

mit $p_1, p_2, p_3 \in \{1, 2, ..., M\}$ werden im Folgenden benötigt. Es sei angemerkt, dass aus $i, j \in I_p$ stets folgt, dass auch $(H_p)_{i,j}$ erklärt ist. Aufgrund der hier vorgenommenen Einschränkung auf nur einfache Überschneidungen, gilt

$$I_{p\cap(p+1)\cap(p+2)} = \emptyset, \quad k = 0, ..., M - 1.$$

Wie in Abschnitt 4.4 erstellen wir nun Updates für Blockmatrizen

$$H_{(p)} \in \mathbb{R}^{n_p\times n_p}, \quad p = 1, ..., M,$$

die auf folgende Weise die Matrix H aufstellen.

$$H_{i,j} = (H_p)_{i,j} = \begin{cases} (H_{(p)})_{i,j}, & \text{falls } i, j \in I_{p\setminus(p-1)\setminus(p+1)} \\ (H_{(p)})_{i,j} + (H_{(p+1)})_{i,j}, & \text{falls } i, j \in I_{p\cap(p+1)} \end{cases}, \quad p = 1, ..., M.$$

Man beachte den Unterschied zwischen H_p und $H_{(p)}$. Zur Vereinfachung der Schreibweise bezeichne

$$(H_{(p_1)})_{p_1\setminus p_2} := \{(H_{(p_1)})_{i,j} : \quad i, j \in I_{p_1\setminus p_2}\}$$

$$(H_{p_1})_{p_1\setminus p_2} := \{(H_{p_1})_{i,j} : \quad i,j \in I_{p_1\setminus p_2}\}$$

und entsprechend analog für die anderen Indexmengen in (4.63). Für einen beliebigen Vektor $x \in \mathbb{R}^n$ definieren wir die Teilvektoren

$$\begin{aligned} x_p &:= (x_i)_{i\in I_p} = x_{I_p} \\ x_{p_1\cap p_2} &:= (x_i)_{i\in I_{p_1\cap p_2}} \\ x_{p_1\cap p_2\cap p_3} &:= (x_i)_{i\in I_{p_1\cap p_2\cap p_3}} \\ x_{p_1\setminus p_2} &:= (x_i)_{i\in I_{p_1\setminus p_2}} \\ x_{p_1\setminus p_2\setminus p_3} &:= (x_i)_{i\in I_{p_1\setminus p_2\setminus p_3}}, \end{aligned} \tag{4.64}$$

mit $p, p_1, p_2, p_3 \in \{1, ..., M\}$.
Wie in Abschnitt 4.2 erwähnt, erfüllt das klassische BFGS-Verfahren die Quasi-Newton-Gleichung

$$H^+ d = y$$

sofern wegen $y^T d > 0$ der Vektor y benutzt werden kann. Durch das hier vorgestellte Verfahren wird dies beim Block-BFGS ebenfalls sichergestellt.

Satz 4.5.1. *Sei $n \in \mathbb{N}$. $H \in \mathbb{R}^{n\times n}$ habe die Struktur (4.62). Seien $y, d \in \mathbb{R}^n$, $\eta \in \mathbb{R}^{M-1}$ und*

$$u_{(1)} = \begin{bmatrix} y_{1\setminus 2} \\ \eta_1 y_{1\cap 2} \end{bmatrix}, \tag{4.65}$$

$$u_{(p)} = \begin{bmatrix} (1-\eta_{p-1}) y_{(p-1)\cap p} \\ y_{p\setminus(p-1)\setminus(p+1)} \\ \eta_p y_{p\cap(p+1)} \end{bmatrix}, \quad p = 2, ..., M-1 \tag{4.66}$$

$$u_{(M)} = \begin{bmatrix} (1-\eta_{M-1}) y_{(M-1)\cap M} \\ y_{M\setminus(M-1)} \end{bmatrix}. \tag{4.67}$$

Analog zum dichten BFGS sei

$$v_{(p)} := -H_{(p)} d_p.$$

Dann erfüllt die aktualisierte Hessematrix

$$H^+ := \sum_{p=1}^{M} \overline{H^+_{(p)}} \tag{4.68}$$

$$H^+_{(p)} := H_{(p)} + \frac{u_{(p)} u_{(p)}^T}{u_{(p)}^T d_p} + \frac{v_{(p)} v_{(p)}^T}{v_{(p)}^T d_p} \tag{4.69}$$

die Quasi-Newton-Bedingung $H^+ d = y$.

Beweis. Die Bedingung ist offensichtlich erfüllt, falls

$$\sum_{p=1}^{M} \overline{\Big(\frac{u_{(p)} {u_{(p)}}^T}{{u_{(p)}}^T d_p}\Big)} d = y \text{ und}$$

$$\sum_{p=1}^{M} \overline{\Big(\frac{v_{(p)}v_{(p)}{}^T}{v_{(p)}{}^T d_p}\Big)d} = -Hd.$$

Beides ist leicht nachzuprüfen:

$$\begin{aligned}
\sum_{p=1}^{M} \overline{\Big(\frac{u_{(p)}u_{(p)}{}^T}{u_{(p)}{}^T d_p}\Big)d} &= \sum_{p=1}^{M} \overline{\Big(\Big(\frac{u_{(p)}u_{(p)}{}^T}{u_{(p)}{}^T d_p}\Big)d_p\Big)} \\
&= \sum_{p=1}^{M} \overline{u_{(p)}} = \overline{y_{1\setminus 2}} + \big(\eta_1 \overline{y_{1\cap 2}} + (1-\eta_1)\overline{y_{1\cap 2}}\big) + \overline{y_{2\setminus 1\setminus 3}} \\
&\quad +\ldots + \big(\eta_{M-1}\overline{y_{(M-1)\cap M}} + (1-\eta_{M-1})\overline{y_{(M-1)\cap M}}\big) + \overline{y_{M\setminus(M-1)}} \\
&= y
\end{aligned}$$

$$\begin{aligned}
\sum_{p=1}^{M} \overline{\Big(\frac{v_{(p)}v_{(p)}{}^T}{v_{(p)}{}^T d_p}\Big)d} &= \sum_{p=1}^{M} \overline{\Big(\frac{v_{(p)}v_{(p)}{}^T}{v_{(p)}{}^T d_p} d_p\Big)} \\
&= \sum_{p=1}^{M} \overline{v_{(p)}} = \sum_{p=1}^{M} \overline{\big(-H_{(p)}d_p\big)} = -\sum_{p=1}^{M} \big(\overline{H}_{(p)}d\big) = -\Big(\sum_{p=1}^{M} \overline{H}_{(p)}\Big)d \\
&= -Hd
\end{aligned}$$

□

Somit erfüllt dieses Update die Quasi-Newton-Bedingung. Nach Satz 4.4.2 ist die Matrix H^+ aus (4.68) positiv definit, wenn die Matrizen $H^+_{(p)}$ aus (4.69) positiv definit sind. Daher gilt es, das Update so zu gestalten, dass alle $H_{(p)},\ p = 1, ..., M$ positiv definit sind. Laut Satz 4.2.3 ist dies genau dann der Fall, wenn

$$u_{(p)}^T d_p > 0, \quad p = 1, ..., M \tag{4.70}$$

gilt. Da jedoch offenbar

$$\sum_{p=1}^{M} u_{(p)}^T d_p = y^T d$$

gilt, kann auch die Bedingung (4.70) nur erfüllt sein, falls auch hier

$$y^T d > 0$$

gilt. Beim klassischen BFGS-Verfahren in Abschnitt 4.2 haben wir andernfalls die positive Definitheit dadurch sicher gestellt, dass y durch q aus (4.13) ersetzt wurde. In gleicher Weise verfahren wir, falls notwendig, bei den Blockmatrizen und verwenden die Vektoren

$$u_{(1)} = \begin{bmatrix} q_{1\setminus 2} \\ \eta_1 q_{1\cap 2} \end{bmatrix}, \tag{4.71}$$

$$u_{(k)} = \begin{bmatrix} (1-\eta_{p-1})q_{(p-1)\cap p} \\ q_{k\setminus(p-1)\setminus(p+1)} \\ \eta_p q_{p\cap(p+1)} \end{bmatrix}, \quad p = 2, ..., M-1 \tag{4.72}$$

$$u_{(M)} \quad = \quad \begin{bmatrix} (1-\eta_{M-1})q_{(M-1)\cap M} \\ q_{M\setminus(M-1)} \end{bmatrix} \tag{4.73}$$

für das Update (4.68)-(4.69). Dies allein garantiert jedoch noch nicht, dass (4.70) erfüllt ist. Der folgende Satz benennt eine Bedingung, wann ein $\eta \in \mathbb{R}^{M-1}$ existiert, so dass $u_{(p)},\ p = 1, ..., M$ die gewünschte Eigenschaft haben.

Satz 4.5.2. *Seien $q, d \in \mathbb{R}^n$ mit $q^T d > 0$. Seien $I_p,\ p = 1, ..., M$, die gemäß (4.63) zur Matrix H aus (4.62) definierten Indexmengen. Bezeichne $\hat{p}_j \in \{1, ..., M-1\},\ j = 1, ..., \hat{M}$, diejenigen $p \in \{1, ..., M-1\}$, für die gilt*

$$q^T_{\hat{p}_j \cap (\hat{p}_j+1)} d_{\hat{p}_j \cap (\hat{p}_j+1)} = 0. \tag{4.74}$$

Diese seien aufsteigend sortiert und $\hat{p}_0$ und $\hat{p}_{M+1}$ so definiert, dass

$$0 =: \hat{p}_0 < \hat{p}_1 < \hat{p}_2 < ... < \hat{p}_{\hat{M}} < \hat{p}_{\hat{M}+1} := M$$

gilt.
Dann existiert ein $\eta \in \mathbb{R}^{M-1}$, so dass (4.70) gilt, falls

$$q^T_{\hat{\mathcal{J}}_j} d_{\hat{\mathcal{J}}_j} > 0, \quad j = 0, ..., \hat{M} \tag{4.75}$$

wobei

$$\hat{\mathcal{J}}_j := \bigcup_{p=1+\hat{p}_j}^{\hat{p}_{j+1}} I_p, \quad j = 0, 1, 2, ..., \hat{M}.$$

Beweis. Sei $j \in \{0, ..., \hat{M}\}$ beliebig und $v \in \mathbb{R}_+^{\hat{p}_{j+1}-\hat{p}_j}$ ein beliebiger Vektor positiver Zahlen, für den gilt

$$\sum_{i=1}^{\hat{p}_{j+1}-\hat{p}_j} v_i = q^T_{\hat{\mathcal{J}}_j} d_{\hat{\mathcal{J}}_j}.$$

Dann kann die Existenz von einem zugehörigen $\eta \in \mathbb{R}^{\hat{p}_{j+1}-\hat{p}_j-1}$ nachgewiesen werden, so dass

$${u_{(p)}}^T d_p = v_{p-\hat{p}_j} > 0, \quad p \in \{\hat{p}_j + 1, ..., \hat{p}_{j+1}\}. \tag{4.76}$$

(4.76) folgt nämlich aus der Lösung des folgenden Gleichungssystems:

$$q^T_{(\hat{p}_j+1)\setminus(\hat{p}_j+2)} d_{(\hat{p}_j+1)\setminus(\hat{p}_j+2)} + \eta_{\hat{p}_j+1} q^T_{(\hat{p}_j+1)\cap(\hat{p}_j+2)} d_{(\hat{p}_j+1)\cap(\hat{p}_j+2)} = v_1, \tag{4.77}$$

$$(1-\eta_{p-1}) q^T_{(p-1)\cap p} d_{(p-1)\cap p} + q^T_{p\setminus(p-1)\setminus(p+1)} d_{p\setminus(p-1)\setminus(p+1)} + \eta_p q^T_{p\cap(p+1)} d_{p\cap(p+1)} = v_{p-\hat{p}_j}, \tag{4.78}$$

$$(1-\eta_{\hat{p}_{j+1}-1}) q^T_{(\hat{p}_{j+1}-1)\cap(\hat{p}_{j+1})} d_{(\hat{p}_{j+1}-1)\cap\hat{p}_{j+1}} + q^T_{\hat{p}_{j+1}\setminus(\hat{p}_{j+1}-1)} d_{\hat{p}_{j+1}\setminus(\hat{p}_{j+1}-1)} = v_{\hat{p}_{j+1}-\hat{p}_j}, \tag{4.79}$$

mit $p = \hat{p}_j + 2, ..., \hat{p}_{j+1} - 1$.
(4.77) impliziert, dass

$$\eta_{\hat{p}_j+1} = \frac{v_1 - q^T_{(\hat{p}_j+1)\setminus(\hat{p}_j+2)} d_{(\hat{p}_j+1)\setminus(\hat{p}_j+2)}}{q^T_{(\hat{p}_j+1)\cap(\hat{p}_j+2)} d_{(\hat{p}_j+1)\cap(\hat{p}_j+2)}}.$$

Eingesetzt in (4.78) folgt

$$\eta_{\hat{p}_j+2} = \frac{v_1 + v_2 - q^T_{(\hat{p}_j+1)\cap(\hat{p}_j+2)} d_{(\hat{p}_j+1)\cap(\hat{p}_j+2)} - \overbrace{q^T_{(\hat{p}_j+1)\setminus(\hat{p}_j+2)} d_{(\hat{p}_j+1)\setminus\hat{p}_j\setminus(\hat{p}_j+2)}}^{=q^T_{(\hat{p}_j+1)\setminus\hat{p}_j\setminus(\hat{p}_j+2)}}}{q^T_{(\hat{p}_j+2)\cap(\hat{p}_j+3)} d_{(\hat{p}_j+2)\cap(\hat{p}_j+3)}} - \frac{q^T_{(\hat{p}_j+2)\setminus(\hat{p}_j+1)\setminus(\hat{p}_j+3)} d_{(\hat{p}_j+2)\setminus(\hat{p}_j+1)\setminus(\hat{p}_j+3)}}{q^T_{(\hat{p}_j+2)\cap(\hat{p}_j+3)} d_{(\hat{p}_j+2)\cap(\hat{p}_j+3)}}$$

und induktiv folgt für $p = \hat{p}_j + 2, ..., \hat{p}_{j+1} - 1$

$$\eta_p = \frac{\sum_{i=\hat{p}_j+1}^{p} v_{i-\hat{p}_j} - \sum_{i=\hat{p}_j+1}^{p-1} q^T_{i\cap(i+1)} d_{i\cap(i+1)} - \sum_{i=\hat{p}_j+1}^{p} q^T_{i\setminus(i-1)\setminus(i+1)} d_{i\setminus(i-1)\setminus(i+1)}}{q^T_{p\cap(p+1)} d_{p\cap(p+1)}}$$

Aufgrund der Konstruktion von v ist (4.79) folglich automatisch erfüllt.
Wendet man dieses Verfahren für alle $j = 0, ..., \hat{M}$ an, so bleiben nur die $\eta_{\hat{p}_j},\ j = \{1, ..., \hat{M}\}$ zunächst unbestimmt. Da jedoch

$$q^T_{\hat{p}_j\cap(\hat{p}_j+1)} d_{\hat{p}_j\cap(\hat{p}_j+1)} = 0$$

gilt, werden diese mit Null multipliziert und können daher beliebig gewählt werden. Die Existenz des Vektors η ist daher nachgewiesen. □

Um die Bedingung (4.75) aus Satz 4.5.2 zu erfüllen, müssen unter Umständen einige der Vektoren $u_{(p)}$ nochmals angepasst werden:

Satz 4.5.3. *Sei $q^T d > 0$ und $H_{(p)},\ p = 1, ..., M$ positiv definit. Die Bedingung (4.75) aus Satz 4.5.2 sei nicht erfüllt. Somit existiert eine Überschneidung in der Matrix zwischen den Blöcken $\hat{p}_j$ und $\hat{p}_j + 1$ so dass (4.74) gilt und (4.75) nicht gilt. Man ersetze q_p durch*

$$\tilde{q}_p := \theta q_p + (1-\theta) H_{(p)} d_p, \quad \hat{p}_j < p \leq \hat{p}_{j+1} \tag{4.80}$$

mit

$$\theta = \frac{\sigma \cdot \sum_{p=\hat{p}_j+1}^{\hat{p}_{j+1}} d_p^T H_{(p)} d_p}{\sum_{p=\hat{p}_j+1}^{\hat{p}_{j+1}} \left(d_p^T H_{(p)} d_p - q_p^T d_p \right)},$$

und $\sigma \in (0,1)$. Dann gilt

$$\tilde{q}^T_{\hat{\mathcal{J}}_j} d_{\hat{\mathcal{J}}_j} > 0.$$

Beweis. Es gilt:

$$\tilde{q}^T_{\hat{\mathcal{J}}_j} d_{\hat{\mathcal{J}}_j} = \theta \sum_{p=\hat{p}_j+1}^{\hat{p}_{j+1}} q_p^T d_p + (1-\theta) \sum_{p=\hat{p}_j+1}^{\hat{p}_{j+1}} d_p^T H_{(p)} d_p$$

$$
\begin{aligned}
&= \frac{\sigma \cdot \sum_{p=\hat{p}_j+1}^{\hat{p}_{j+1}} d_p^T H_{(p)} d_p \sum_{p=\hat{p}_j+1}^{\hat{p}_{j+1}} q_p^T d_p}{\sum_{p=\hat{p}_j+1}^{\hat{p}_{j+1}} \left(d_p^T H_{(p)} d_p - q_p^T d_p \right)} + \frac{(1-\sigma) \sum_{p=\hat{p}_j+1}^{\hat{p}_{j+1}} \left(d_p^T H_{(p)} d_p - q_p^T d_p \right) \sum_{p=\hat{p}_j+1}^{\hat{p}_{j+1}} d_p^T H_{(p)} d_p}{\sum_{p=\hat{p}_j+1}^{\hat{p}_{j+1}} \left(d_p^T H_{(p)} d_p - q_p^T d_p \right)} \\
&= \frac{(1-\sigma) \left(\sum_{p=\hat{p}_j+1}^{\hat{p}_{j+1}} d_p^T H_{(p)} d_p \right)^2 - (1-\sigma) \sum_{p=\hat{p}_j+1}^{\hat{p}_{j+1}} d_p^T H_{(p)} d_p \cdot \sum_{p=\hat{p}_j+1}^{\hat{p}_{j+1}} q_p^T d_p}{\sum_{p=\hat{p}_j+1}^{\hat{p}_{j+1}} \left(d_p^T H_{(p)} d_p - q_p^T d_p \right)} \\
&= (1-\sigma) \sum_{p=\hat{p}_j+1}^{\hat{p}_{j+1}} d_p^T H_{(p)} d_p \\
&> 0,
\end{aligned}
$$

da alle $H_{(p)},\ \hat{p}_j < p \leq \hat{p}_{j+1}$ positiv definit sind. □

Bemerkung 4.5.4.

a) Durch die Verwendung von $\tilde{q}_p,\ \hat{p}_j < p \leq \hat{p}_{j+1}$ verändern sich die Updates der Blöcke $H_{(p)},\ p \notin \hat{\mathcal{J}}_j$ offensichtlich nicht.

b) Möglicherweise bewährt es sich hier, σ anders zu wählen als im klassischen BFGS, in dem üblicherweise der heuristische Wert $\sigma = 0.8$ gewählt wird.

c) Durch die Anpassung von q nach Satz 4.5.3 kann es dazu kommen, dass (4.74) bereits nicht mehr gilt, wodurch (4.75) gar nicht mehr erfüllt sein müsste. Des Weiteren kann der Fall auftreten, dass durch die Anpassung eine weitere Überschneidung innerhalb $\hat{\mathcal{J}}_j$ die Gleichung (4.74) erfüllt. Dies würde es erforderlich machen, Satz 4.5.3 erneut anzuwenden. Es leuchtet jedoch ein, dass diese sukzessive Anpassung nach endlich vielen Schritten endet; nämlich nach maximal so vielen wie die Anzahl der Überschneidungen. Die Anpassung einer solchen mehrmaligen Anwendung von Satz 4.5.3 würde sich im Übrigen auf immer kleinere Teile von q beziehen.

Nach Satz 4.2.4 kann in einer hinreichend kleinen Umgebung von x^* stets y statt q verwendet werden. Gleiches gilt auch für die einzelnen Blöcke. In einer hinreichend kleinen Umgebung können stets die Vektoren $u_{(p)}$ aus (4.65)-(4.67) statt aus (4.71)-(4.73) verwendet werden. Um dies nachzuweisen, bedarf es einiger Vorbereitungen.
Wir wissen bereits aus Satz 4.4.2, dass wir positiv definite Blöcke auf die bekannte Weise (das heißt durch Summation in den Überschneidungen) zusammenschieben können, so dass die Gesamtmatrix der Form (4.62) positiv definit ist. An dieser Stelle stellen wir die umgekehrte Frage: Existieren zu einer positiv definiten Matrix der Form (4.62) stets positiv definite Blöcke, die zusammen die Gesamtmatrix aufbauen?
Zur Beantwortung dieser Frage benötigen wir zunächst das Sylvester-Kriterium. Es besagt

Lemma 4.5.5. *Sei $A \in \mathbb{R}^{n \times n}$. Unter einem Hauptminor versteht man die Determinante*

der linken oberen $k \times k$-Teilmatrix von A, wobei $k \leq n$ gilt. Die Matrix A ist genau dann positiv definit, wenn alle Hauptminoren positiv sind.

Beweis. Siehe Fischer [25] Seite 217f. □

Bemerkung 4.5.6. Durch Permutation der Matrix ändern sich offensichtlich die Eigenvektoren nicht, so dass das Kriterium in gleicher Weise für die rechten unteren Teilmatrizen gilt.

Lemma 4.5.7. *Sei $n \in \mathbb{N}$ und $A \in \mathbb{R}^{n \times n}$ positiv definit. Dann existiert A^{-1} und ist ebenfalls positiv definit.*

Beweis. Wegen der positiven Definitheit von A, folgt nach dem Sylvester-Kriterium, dass insbesondere

$$\det(A) > 0$$

gilt und A folglich vollen Rang besitzt und somit invertierbar ist. Daraus folgt die Existenz von A^{-1}. Seien $\lambda_i,\ i = 1, ..., n$, die Eigenwerte der Matrix A und $v_i \in \mathbb{R}^n,\ i = 1, ..., n$, die zugehörigen Eigenvektoren. Da A positiv definit ist, gilt

$$\lambda_i > 0, \quad i = 1, ..., n.$$

Wegen

$$Av_i = \lambda_i v_i \iff v_i = A^{-1}\lambda_i v_i = \lambda_i A^{-1} v_i \overset{\lambda_i > 0}{\iff} \lambda_i^{-1} v_i = A^{-1} v_i.$$

sind offensichtlich $\lambda_i^{-1},\ i = 1, ..., n$ die Eigenwerte von A^{-1}. Sei $\lambda_{\max}(A)$ der größte Eigenwert von A, so ist $\lambda_{\max}(A)^{-1}$ der kleinste Eigenwert von A^{-1}. Da wegen der positiven Eigenwerte $\lambda_i,\ i = 1, ..., n$ auch $\lambda_{\max}(A)^{-1}$ positiv ist, folgt die positive Definitheit von A^{-1}. □

Lemma 4.5.8. *Sei $n \in \mathbb{N}$ und $A \in \mathbb{R}^{n \times n}$ symmetrisch und positiv definit. Die Matrix $C \in \mathbb{R}^{n \times n}$ habe vollen Rang. Dann ist auch $C^T AC$ positiv definit.*

Beweis. Vgl. [34]. Sei $x \in \mathbb{R}^n$ beliebig. Setze

$$y := Cx.$$

Dann gilt wegen der positiven Definitheit von A

$$x^T C^T ACx = y^T Ay \geq 0.$$

und

$$x^T C^T ACx > 0 \Longleftrightarrow Cx \neq 0.$$

Wegen des vollen Rangs von C gilt aber

$$Cx = 0 \Longleftrightarrow x = 0.$$

Somit folgt

$$x^T C^T ACx = 0 \Longleftrightarrow x = 0,$$

was den Beweis abschließt. □

Lemma 4.5.9. *Seien $n, n_1, n_2 \in \mathbb{N}$ und*

$$\begin{bmatrix} A & B \\ B^T & C \end{bmatrix} \in \mathbb{R}^{n \times n}$$

eine symmetrische Matrix mit den Blöcken $A \in \mathbb{R}^{n_1 \times n_1}, B \in \mathbb{R}^{n_1 \times n_2}$ und $C \in \mathbb{R}^{n_2 \times n_2}$. Dann gilt

$$\begin{bmatrix} A & B \\ B^T & C \end{bmatrix} \text{ positiv definit} \iff A \text{ und } C - B^T A^{-1} B \text{ positiv definit} \tag{4.81}$$

Beweis. Vgl. [34].
"$\Longrightarrow$":
Dass A positiv definit sein muss, folgt unmittelbar aus dem Sylvester-Kriterium. Nach Lemma 4.5.7 existiert

$$\begin{bmatrix} A & B \\ B^T & C \end{bmatrix}^{-1} = \begin{bmatrix} (A - BC^{-1}B^T)^{-1} & A^{-1}B(B^T A^{-1} B - C)^{-1} \\ (B^T A^{-1} B - C)^{-1} B^T A^{-1} & (C - B^T A^{-1} B)^{-1} \end{bmatrix}. \tag{4.82}$$

und ist positiv definit. Nach (der Bemerkung 4.5.6 zu) dem Sylvester-Kriterium folgt, dass insbesondere

$$(C - B^T A^{-1} B)^{-1}$$

positiv definit ist.
"$\Longleftarrow$":
Seien A und $C - B^T A^{-1} B$ positiv definit. Setze $X := A^{-1}B$. Dann gilt

$$\begin{bmatrix} I & 0 \\ X^T & I \end{bmatrix} \begin{bmatrix} A & 0 \\ 0 & C - B^T A^{-1} B \end{bmatrix} \begin{bmatrix} I & X \\ 0 & I \end{bmatrix} = \begin{bmatrix} A & B \\ B^T & C \end{bmatrix}.$$

Die Matrix

$$\begin{bmatrix} A & 0 \\ 0 & C - B^T A^{-1} B \end{bmatrix}$$

ist nach Voraussetzung positiv definit und

$$\begin{bmatrix} I & 0 \\ X^T & I \end{bmatrix}$$

hat vollen Rang. Nach Lemma 4.5.8 ist daher

$$\begin{bmatrix} A & B \\ B^T & C \end{bmatrix}$$

positiv definit. □

Korollar 4.5.10. *Analog gilt entsprechend auch*

$$\begin{bmatrix} A & B \\ B^T & C \end{bmatrix} \text{ positiv definit} \iff C \text{ und } A - BC^{-1}B^T \text{ positiv definit.}$$

Satz 4.5.11. *Seien $n_1, n_2, n_3 \in \mathbb{N}$ und $A \in \mathbb{R}^{n_1 \times n_1}$, $B \in \mathbb{R}^{n_1 \times n_2}$, $C \in \mathbb{R}^{n_2 \times n_2}$, $D \in \mathbb{R}^{n_2 \times n_3}$, $E \in \mathbb{R}^{n_3 \times n_3}$. A, C und E seien symmetrisch. Die Matrix*

$$M := \begin{bmatrix} A & B & 0 \\ B^T & C & D \\ 0 & D^T & E \end{bmatrix}$$

sei positiv definit. Dann existieren symmetrische Matrizen $C_1, C_2 \in \mathbb{R}^{n_2 \times n_2}$ mit $C_1 + C_2 = C$ so dass $\begin{bmatrix} A & B \\ B^T & C_1 \end{bmatrix}$ und $\begin{bmatrix} C_2 & D \\ D^T & E \end{bmatrix}$ positiv definit sind.

Beweis. Nach dem Sylvester-Kriterium sind insbesondere die Matrizen

$$\begin{bmatrix} A & B \\ B^T & C \end{bmatrix}$$

und E positiv definit. Nach Lemma 4.5.9 gilt somit, dass auch $C - B^T A^{-1} B$ positiv definit ist. Andererseits gilt ebenfalls unter Anwendung von Lemma 4.5.9, dass wegen der positiven Definitheit von M die Matrix

$$E - \begin{bmatrix} 0 & D^T \end{bmatrix} \begin{bmatrix} A & B \\ B^T & C \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ D \end{bmatrix}$$

positiv definit ist. Aus (4.82) folgt, dass

$$\begin{bmatrix} 0 & D^T \end{bmatrix} \begin{bmatrix} A & B \\ B^T & C \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ D \end{bmatrix} = D^T (C - B^T A^{-1} B)^{-1} D.$$

Es gilt folglich, dass $E - D^T (C - B^T A^{-1} B)^{-1} D$ positiv definit ist. Dies, zusammen mit der bereits festgestellten Tatsache, dass $C - B^T A^{-1} B$ positiv definit ist, ist wiederum laut Lemma 4.5.9 äquivalent dazu, dass die Matrix

$$\begin{bmatrix} C - B^T A^{-1} B & D \\ D^T & E \end{bmatrix}$$

positiv definit ist. Nach Korollar 4.5.10 folgt hiermit, dass

$$X := C - B^T A^{-1} B - D E^{-1} D^T$$

positiv definit ist. Sei $\lambda_{\min}$ der kleinste Eigenwert von X und α mit $0 < \alpha < \lambda_{\min}$ beliebig. Setze $C_1 := B^T A^{-1} B + \alpha I$ und entsprechend $C_2 = C - C_1$. Dann ist die Matrix

$$\begin{bmatrix} A & B \\ B^T & C_1 \end{bmatrix}$$

wegen Lemma 4.5.9 positiv definit, denn A ist positiv definit und

$$C_1 - B^T A^{-1} B = B^T A^{-1} B + \alpha I - B^T A^{-1} B = \alpha I$$

ist ein positives Vielfaches der Einheitsmatrix. Da die Matrix

$$X - \alpha I = C_1 + C_2 - B^T A^{-1} B - D E^{-1} D^T - \alpha I = C_2 - D E^{-1} D^T$$

mit $\lambda_{\min} - \alpha > 0$ einen positiven kleinsten Eigenwert hat, ist sie positiv definit. Dass E positiv definit ist, haben wir bereits erwähnt. Nach Korollar 4.5.10 folgt somit, dass auch

$$\begin{bmatrix} C_2 & D \\ D^T & E \end{bmatrix}$$

positiv definit ist. □

Korollar 4.5.12. *Für eine positiv definite Matrix mit beliebig vielen sich überschneidenden Blöcken existieren ebenfalls positiv definite Blöcke, die die Matrix zusammensetzen.*

Beweis. Seien $n_1, n_2, n_3, n_4 \in \mathbb{N}$ und $A \in \mathbb{R}^{n_1 \times n_1}$, $B \in \mathbb{R}^{n_1 \times n_2}$, $C \in \mathbb{R}^{n_2 \times n_2}$, $D \in \mathbb{R}^{n_2 \times n_3}$, $E \in \mathbb{R}^{n_3 \times n_3}$, $F \in \mathbb{R}^{n_3 \times n_4}$, $G \in \mathbb{R}^{n_4 \times n_4}$. A, C, E und G seien symmetrisch. Sei die positiv definite Matrix

$$M := \begin{bmatrix} A & B & 0 & 0 \\ B^T & C & D & 0 \\ 0 & D^T & E & F \\ 0 & 0 & F^T & G \end{bmatrix}$$

gegeben. Nach Satz 4.5.11 können wir E_1 und E_2 finden mit

$$E_1 + E_2 = E$$

so dass

$$\begin{bmatrix} A & B & 0 \\ B^T & C & D \\ 0 & D^T & E_1 \end{bmatrix}$$

und

$$\begin{bmatrix} E_2 & F \\ F^T & G \end{bmatrix}$$

positiv definit sind. Genauso können nach Satz 4.5.11 für die Matrix

$$\begin{bmatrix} A & B & 0 \\ B^T & C & D \\ 0 & D^T & E_1 \end{bmatrix}$$

C_1 und C_2 gefunden werden mit

$$C_1 + C_2 = C$$

so dass

$$\begin{bmatrix} A & B \\ B^T & C_1 \end{bmatrix}$$

und

$$\begin{bmatrix} C_2 & D \\ D^T & E_1 \end{bmatrix}.$$

positiv definit sind. Sukzessive kann daher für jede Anzahl von Blöcken eine Aufteilung gefunden werden, so dass alle Blöcke positiv definit sind. □

Damit ist die Antwort auf die oben gestellte Frage gefunden: Es existieren stets positiv definite Blöcke, die die positiv definite Blockmatrix in dem hier behandelten Sinne zusammensetzen. Dieser Sachverhalt erlaubt es uns, den folgenden Satz zu formulieren.

Satz 4.5.13. *$\nabla^2 f$ habe die Form (4.62). Dann existiert ein $\varepsilon > 0$ so dass für $x^+, x \in U_\varepsilon(x^*)$ stets ein Vektor $\eta \in \mathbb{R}^{M-1}$ existiert, so dass für die Vektoren*

$$\begin{aligned} u_{(1)} &= \begin{bmatrix} y_{1\setminus 2} \\ \eta_1 y_{1\cap 2} \end{bmatrix}, \\ u_{(p)} &= \begin{bmatrix} (1-\eta_{p-1}) y_{(p-1)\cap p} \\ y_{p\setminus(p-1)\setminus(p+1)} \\ \eta_p y_{p\cap(p+1)} \end{bmatrix}, \quad p = 2, ..., M-1 \\ u_{(M)} &= \begin{bmatrix} (1-\eta_{M-1}) y_{(M-1)\cap M} \\ y_{M\setminus(M-1)} \end{bmatrix}, \end{aligned} \tag{4.83}$$

gilt

$$u_{(p)}^T d_p > 0, \quad p = 1, ..., M.$$

Beweis. Nach dem Mittelwertsatz existiert ein $\xi \in U_\varepsilon(x^*)$, so dass

$$\nabla^2 f(\xi) d = y.$$

Da wir f als zweimal stetig differenzierbar und $\nabla^2 f(x^*)$ als positiv definit annehmen, kann ε so gewählt werden, dass $\nabla^2 f(\xi)$ ebenfalls positiv definit ist. Somit existieren nach Satz 4.5.11 positiv definite Matrizen $\mathcal{H}_{(p)} \in \mathbb{R}^{n_p \times n_p},\ p = 1, ..., M$ mit

$$\sum_{p=1}^{M} \overline{\mathcal{H}_{(p)}} = \nabla^2 f(\xi).$$

Daraus folgt

$$y = \nabla^2 f(\xi) d = \overline{\sum_{p=1}^{M} \left(\underbrace{\mathcal{H}_{(p)} d_p}_{=:\tilde{u}_{(p)}} \right)}$$

Wegen der positiven Definitheit von $\mathcal{H}_{(p)}$ gilt offensichtlich

$$\tilde{u}_{(p)}^T d_p > 0.$$

Jedoch sind die Vektoren $\tilde{u}_{(p)},\ p = 1, ..., M$ nicht notwendigerweise durch einen Vektor $\eta \in \mathbb{R}^{M-1}$ gemäß (4.83) konstruierbar. Es existieren jedoch Vektoren $\nu_p \in \mathbb{R}^{n_{p\cap(p+1)}},\ p = 1, ..., M-1$, so dass

$$\begin{aligned} \tilde{u}_{(1)} &= \begin{bmatrix} y_{1\setminus 2} \\ \operatorname{diag}(\nu_p) y_{1\cap 2} \end{bmatrix}, \\ \tilde{u}_{(p)} &= \begin{bmatrix} (I - \operatorname{diag}(\nu_{p-1})) y_{(p-1)\cap p} \\ y_{p\setminus(p-1)\setminus(p+1)} \\ \operatorname{diag}(\nu_p) y_{p\cap(p+1)} \end{bmatrix}, \quad p = 2, ..., M-1 \end{aligned}$$

$$\tilde{u}_{(M)} = \begin{bmatrix} (I - \operatorname{diag}(\nu_{M-1}))y_{(M-1)\cap M} \\ y_{M\setminus(M-1)} \end{bmatrix}. \tag{4.84}$$

Wähle für $p = 1, ..., M-1$

$$\eta_p := \frac{(\operatorname{diag}(\nu_p)y_{p\cap(p+1)})^T d_{p\cap p+1}}{y_{p\cap(p+1)}^T d_{p\cap p+1}}$$

und definiere damit $u_{(p)},\ p = 1, ..., M$ gemäß (4.83). Es gilt

$$u_{(p)}^T d_p = \tilde{u}_{(p)}^T d_p > 0.$$

□

Somit erhalten wir ein dünn besetztes BFGS-artiges Update, welches stets positiv definit ist und in einer Umgebung um x^* die Quasi-Newton-Bedingung erfüllt.
Wir gehen wieder davon aus, dass $x^{(0)}$ eine hinreichend gute Startschätzung sei und folglich, nach Satz 4.5.13 die Verwendung von q beziehungsweise q_p und $\tilde{q}_p$ aus (4.80) unnötig ist. Somit formulieren wir den folgenden

Algorithmus 4.5.14.

i. Identifiziere eine Struktur der Form (4.62).

ii. Wähle $x^{(0)} \in \mathbb{R}^n$ und $H_{(p)}^{(0)} \in \mathbb{R}^{n_p \times n_p},\ p = 1, ..., M$ symmetrisch und positiv definit und setze

$$H^{(0)} := \sum_{p=1}^{M} H_{(p)}^{(0)}.$$

Wähle außerdem $\varepsilon \geq 0$ und setze $k := 0$.

iii. Ist $\|\nabla f(x^{(k)})\| \leq \varepsilon$. STOP.

iv. Bestimme $d^{(k)}$ aus dem Gleichungssystem

$$H^{(k)} d^{(k)} = -\nabla f(x^{(k)}).$$

v. Setze

$$\begin{aligned} x^{(k+1)} &:= x^{(k)} + d^{(k)} \text{ und} \\ y^{(k)} &:= \nabla f(x^{(k+1)}) - \nabla f(x^{(k)}). \end{aligned}$$

vi. Bestimme $\eta^{(k)} \in \mathbb{R}^{M-1}$, so dass für alle Blöcke

$$u_{(p)}^{(k)T} d_p^{(k)} > 0$$

gilt, mit $u_{(p)}^{(k)}$ aus (4.65)-(4.67).

vii. Berechne

$$H_{(p)}^{(k+1)} = H_{(p)}^{(k)} + \frac{y_p^{(k)} y_p^{(k)T}}{y_p^{(k)T} d_p^{(k)}} - \frac{(H_{(p)}^{(k)} d_p^{(k)})(H_{(p)}^{(k)} d_p^{(k)})^T}{d_p^{(k)T} H_{(p)}^{(k)} d_p^{(k)}}, \quad p = 1, ..., M$$

und berechne die gesamte Matrix durch

$$H^{(k+1)} = \sum_{p=1}^{M} \overline{H_{(p)}^{(k+1)}}.$$

viii. Setze k:=k+1 und gehe zu iii.

Bemerkung 4.5.15. Schritt *vi.* kann beispielsweise wie im Beweis von Satz 4.5.2 erfolgen.

4.5.2.2 Konvergenzeigenschaften des Block-BFGS-Verfahrens mit einfachen Überschneidungen

Für die Aussagen über die Konvergenzgeschwindigkeit vom Algorithmus 4.5.14 verwenden wir Resultate aus Abschnitt 4.4.
Es gelten die Voraussetzungen aus Abschnitt 4.2.2. Darüber hinaus habe $\nabla^2 f$ eine dünnbesetzte Struktur. Es existiere eine Blockstruktur der Form (4.62), so dass alle Nichtnulleinträge innerhalb der Struktur liegen. Hierbei sei es jedoch zulässig, dass zu Gunsten einer Blockstruktur ein Nulleintrag innerhalb der Struktur liegt und somit als Nichtnulleintrag behandelt wird. Wie in Abschnitt 4.4 erörtert, gibt es insbesondere für so eine Blockstruktur eine (nicht notwendigerweise eindeutige) Darstellung der Zielfunktion

$$f(x) = \sum_{p=1}^{M} f_{(p)}(x_p) \tag{4.85}$$

wie in (4.47), für die die Funktionen $f_{(p)}$ zweimal stetig differenzierbar sind.
Im folgenden werden wir uns die Aussage von Satz 4.4.10 zu Nutze machen und die superlineare Konvergenz sicherstellen, obwohl wir keine Kenntnis über eine geeignete Zerlegung (4.85) haben. Zur Vorbereitung der Konvergenzaussage zitieren wir den Satz von Bauer und Fike:

Satz 4.5.16. *Sei $\|\cdot\|$ eine submultiplikative Norm und $A \in \mathbb{R}^{n\times n}$ eine diagonalisierbare Matrix mit den Eigenwerten λ_i, $i = 1, ..., n$, so dass mit $S \in \mathbb{R}^{n\times n}$*

$$S^{-1}AS = \operatorname{diag}(\lambda_1, ..., \lambda_n)$$

gelte. Sei $B \in \mathbb{R}^{n\times n}$ eine weitere Matrix. Dann gilt für jeden Eigenwert $\lambda(A+B)$ der Matrix $A+B$:

$$\min_{i=1,...,n} |\lambda(A+B) - \lambda_i| \leq \kappa(S)\|B\|,$$

wobei $\kappa(S)$ die Kondition der Matrix S bezeichnet.

Beweis. Siehe [3]. □

Korollar 4.5.17. *Für eine symmetrische Matrix A gilt im Satz 4.5.16 sogar*

$$\min_{i=1,...,n} |\lambda(A+B) - \lambda_i| \leq \|B\|,$$

da $\kappa(S) = 1$.

Lemma 4.5.18. *Sei* $(A^{(k)})_{k\in\mathbb{N}} \subset \mathbb{R}^{n\times n}$ *eine Folge positiv definiter Matrizen. Dann existieren für jedes* $K \in \mathbb{N}$ *Konstanten* $b^{(j)} \in \mathbb{R}^+,\ j = 2, ..., K$ *so dass für alle* $k < K$

$$A^{(k)} + \sum_{j=k+1}^{K} \beta^{(j)}(A^{(j+1)} - A^{(j)})$$

mit

$$\beta^{(j)} \in [-b^{(j)}, b^{(j)}], \quad j = 2, ..., K$$

positiv definit sind.

Beweis. Sei $K \in \mathbb{N}$ beliebig und $A^{(k)},\ k \leq K$ nach Voraussetzung positiv definit. Setze mit $0 < \varepsilon^{(k)} < \lambda_{min}(A^{(k)})$

$$b^{(kj)} := \frac{\lambda_{\min}(A^{(k)}) - \frac{1}{2}\varepsilon^{(k)}}{2^{j-k}|\lambda_{\max}(A^{(j+1)} - A^{(j)})|}, \quad j = k+1, ..., K.$$

Es gilt nach Satz 4.5.16 von Bauer und Fike mit $\|\cdot\|_2$ und beliebigem $\beta^{(kj)} \in [-b^{(kj)}, b^{(kj)}]$:

$$\begin{aligned}
\lambda_{\min}(A^{(k)} - \beta^{(kj)}(A^{(j+1)} - A^{(j)})) &\geq \lambda_{\min}(A^{(k)}) - b^{(kj)}\|A^{(j+1)} - A^{(j)}\|_2 \\
&= \lambda_{\min}(A^{(k)}) - b^{(kj)}|\lambda_{\max}(A^{(j+1)} - A^{(j)})| \\
&= \lambda_{\min}(A^{(k)}) - \frac{\lambda_{\min}(A^{(k)}) - \frac{1}{2}\varepsilon^{(k)}}{2^{j-k}} \\
&\geq \frac{\lambda_{\min}(A^{(k)})}{2} + \frac{\varepsilon^{(k)}}{2^{j-k+1}} \\
&> 0.
\end{aligned}$$

Genauso folgt für alle $k \in \mathbb{N}$

$$\begin{aligned}
\lambda_{\min}\left(A^{(k)} - \sum_{j=k+1}^{K} \beta^{(kj)}(A^{(j+1)} - A^{(j)})\right) &\geq \lambda_{\min}(A^{(k)}) - \sum_{j=k+1}^{K} b^{(kj)}\|A^{(j+1)} - A^{(j)}\| \\
&= \lambda_{\min}(A^{(k)}) - \sum_{j=k+1}^{K} b^{(kj)}|\lambda_{\max}(A^{(j+1)} - A^{(j)})| \\
&= \lambda_{\min}(A^{(k)}) - \sum_{j=k+1}^{K} \frac{\lambda_{\min}(A^{(k)}) - \frac{1}{2}\varepsilon^{(k)}}{2^{j-k}}
\end{aligned}$$

$$
\begin{aligned}
&= \lambda_{\min}(A^{(k)}) \underbrace{\left(1 - \sum_{j=1}^{K-k} 2^{-j}\right)}_{>0} + \sum_{j=k+1}^{K} \frac{\varepsilon^{(k)}}{2^{j-k+1}} \\
&\geq \frac{\varepsilon^{(k)}}{2} \\
&> 0, \quad \beta^{(kj)} \in [-b^{(kj)}, b^{(kj)}].
\end{aligned}
$$

Wähle daher die Konstanten

$$b^{(j)} := \min_{k=1,\ldots,j-1} b^{(kj)}, \quad j = 2, ..., K.$$

□

Satz 4.5.19. *Sei $K \in \mathbb{N}$ und seien $x^{(1)}, ..., x^{(K)} \in \mathbb{R}^n$ Iterierte des Algorithmus' 4.5.14. Seien weiter $g^{(1)}, ..., g^{(K)} \in \mathbb{R}^n$ beliebige Vektoren mit*

$$(g^{(k+1)} - g^{(k)})^T (x^{(k+1)} - x^{(k)}) > 0, \quad k = 1, ..., K-1.$$

Dann existiert eine zweimal stetig differenzierbare, streng konvexe Funktion $f : \mathbb{R}^n \to \mathbb{R}$ mit

$$\nabla f(x^{(k)}) = g^{(k)}, \quad k = 1, ..., K,$$

deren Hessematrix $\nabla^2 f$ lokal Lipschitz-stetig ist.

Beweis. Betrachte zwei beliebige auf einander folgende Iterierte $x^{(k)}$ und $x^{(k+1)}$. Bezeichne

$$
\begin{aligned}
\Delta g^{(k)} &:= (g^{(k+1)} - g^{(k)}) \\
d^{(k)} &:= (x^{(k+1)} - x^{(k)}).
\end{aligned}
$$

Nach Voraussetzung gelte

$$\Delta g^{(k)T} d^{(k)} > 0 \tag{4.86}$$

und somit gilt für den Winkel $\measuredangle(\Delta g^{(k)}, d^{(k)})$ zwischen $\Delta g^{(k)}$ und $d^{(k)}$ nach dem Kosinussatz

$$\cos\left(\measuredangle(\Delta g^{(k)}, d^{(k)})\right) > 0$$

und folglich

$$0^\circ \leq \measuredangle(\Delta g^{(k)}, d^{(k)}) < 90^\circ.$$

Falls $\Delta g^{(k)}$ und $d^{(k)}$ linear unabhängig sind (das heißt $\measuredangle(\Delta g^{(k)}, d^{(k)}) \neq 0$), existieren demnach auf einander senkrecht stehende Vektoren $v_1, v_2 \in \mathbb{R}^n$ in der von $\Delta g^{(k)}$ und $d^{(k)}$ aufgespannten Ebene mit

$$
\begin{aligned}
\Delta g^{(k)} &= \sum_{i=1}^{2} a_i v_i, \\
d^{(k)} &= \sum_{i=1}^{2} b_i v_i
\end{aligned}
$$

und $a_i, b_i > 0,\ i = 1, 2$. Ergänzend seien $v_3, ..., v_n$ so gewählt, dass $v_1, ..., v_n$ eine Orthogonalbasis bilden. Betrachte die Matrix

$$\Lambda = \text{diag}\left(\frac{a_1}{b_1}, \frac{a_2}{b_2}, \lambda_3, \lambda_4, ..., \lambda_n\right)$$

mit $\lambda_3, \lambda_4, ..., \lambda_n > 0$. P bezeichne die Transformationsmatrix von der Einheitsbasis in die Basis $v_1, ..., v_n$. Die Matrix

$$A := P^T \Lambda P$$

ist symmetrisch, positiv definit und es gilt

$$Ad^{(k)} = \sum_{i=1}^{2} b_i A v_i = \sum_{i=1}^{2} a_i v_i = \Delta g^{(k)}. \tag{4.87}$$

Im Falle, in dem $\Delta g^{(k)}$ ein (wegen (4.86) notwendigerweise positives) Vielfaches von $d^{(k)}$ ist und somit keine Ebene aufgespannt wird, kann $d^{(k)}$ als ersten Basisvektor gewählt werden. Der Faktor des Vielfachen ist dann der erste Eigenwert, während alle weiteren wieder als beliebige positive Werte gewählt werden.
Ignoriert man zunächst die anderen Iterierten, so erfüllt jede Funktion $f : \mathbb{R}^n \to \mathbb{R}$ mit

$$\begin{aligned} \nabla f(x^{(k)}) &= g^{(k)} \\ \nabla^2 f(x) &= A. \end{aligned}$$

die Behauptung. Da die Funktion f jedoch zweimal stetig differenzierbar sein muss, können wir nicht für jeden Teilabschnitt

$$x^{(k)} + \alpha(x^{(k+1)} - x^{(k)}),\ \alpha \in [0, 1]$$

unterschiedliche konstante Hessematrizen definieren.
Auf dieser Basis können wir jedoch im Folgenden eine Funktion f mit einer stetigen Hessematrix konstruieren.
Seien $A^{(k)},\ k = 1, ..., K$ die gemäß der obigen Vorgehensweise definierten Matrizen. Da nach Definition alle $A^{(k)}$ positiv definit sind, existieren nach Lemma 4.5.18 Konstanten $b^{(j)} > 0,\ j = 2, ..., K$, so dass alle Matrizen

$$A^{(k)} - \sum_{j=k}^{K} \beta^{(kj)}(A^{(j+1)} - A^{(j)}), \quad \beta^{(kj)} \in [0, b_j]$$

positiv definit sind. Definiere für

$$\beta^{(kj)} \in [0, b^{(j)}),\ j = k + 1, ..., K \tag{4.88}$$

mit

$$\beta^{(k(k+1))} < \beta^{(k(k+2))} < ... < \beta^{(kK)}$$

die Matrix

$$B^{(k)} := A^{(k)} - \sum_{j=k}^{K} \beta^{(kj)}(A^{(j+1)} - A^{(j)}). \tag{4.89}$$

Offensichtlich sind die Matrizen (4.89) auch symmetrisch.
Wegen

$$\beta^{(kj)} < b^{(j)}$$

existieren $\varepsilon^{(kj)} > 0$ mit

$$\beta^{(kj)} + \varepsilon^{(kj)} < b^{(j)}.$$

Bezeichne

$$\begin{aligned} c^{(kk)} &:= \frac{\beta^{(kk)}}{1+\beta^{(kk)}} \\ c^{(kj)} &:= 2\frac{\varepsilon^{(kj)}}{\beta^{((k+1)j)} + \beta^{(kj)} + 2\varepsilon^{(kj)}} \end{aligned}$$

Damit definieren wir die integrierbaren Funktionen

$$s_{kj} : [0,1] \to \mathbb{R},\ k = 1, ..., K,\ j = k, ..., K$$

mit

$$s_{kk}(\alpha) := \begin{cases} \begin{cases} 0, & \textit{falls } \alpha < 1 - 2c^{(kk)} \\ \frac{\alpha - (1-2c^{(kk)})}{2c^{(kk)}}, & \textit{falls } \alpha \geq 1 - 2c^{(kk)} \end{cases}, & \text{falls } \beta^{(kk)} < 1 \\ \begin{cases} \frac{\alpha}{2(1-c^{(kk)})}, & \textit{falls } \alpha < 2(1-c^{(kk)}) \\ 1, & \textit{falls } \alpha \geq 2(1-c^{(kk)}) \end{cases}, & \text{falls } \beta^{(kk)} \geq 1, \end{cases}$$

$$s_{kj}(\alpha) := \begin{cases} -\frac{\alpha}{c^{(kj)}} \frac{\beta^{(kj)} + \varepsilon^{(kj)}}{\beta^{((k+1)j)} - \beta^{(kj)}}, & \alpha < c^{(kj)} \\ -\frac{\beta^{(kj)} + \varepsilon^{(kj)}}{\beta^{((k+1)j)} - \beta^{(kj)}}, & c^{(kj)} \leq \alpha < (1-c^{(kj)}) \\ -\frac{\beta^{(kj)} + \varepsilon^{(kj)}}{\beta^{((k+1)j)} - \beta^{(kj)}} + \frac{\alpha - (1-c^{(kj)})}{c^{(kj)}} \left(1 + \frac{\beta^{(kj)} + \varepsilon^{(kj)}}{\beta^{((k+1)j)} - \beta^{(kj)}}\right), & \alpha \geq (1-c^{(kj)}). \end{cases}$$

Es gilt

$$\int_0^1 s_{kk}(t)dt = \begin{cases} \int_0^{2c^{(kk)}} \frac{t}{2c^{(kk)}} dt = c^{(kk)}, & \text{falls } \beta^{(kk)} < 1 \\ \int_0^{2(1-c^{(kk)})} \frac{t}{2(1-c^{(kk)})} dt + (2c^{(kk)} - 1) = c^{(kk)}, & \text{falls } \beta^{(kk)} \geq 1 \end{cases} \tag{4.90}$$

sowie

$$\begin{aligned} \int_0^1 s_{kj}(t)dt &= -\int_0^{c^{(kj)}} \frac{t}{c^{(kj)}} \frac{\beta^{(kj)} + \varepsilon^{(kj)}}{\beta^{((k+1)j)} - \beta^{(kj)}} dt - (1-c^{(kj)}) \frac{\beta^{(kj)} + \varepsilon^{(kj)}}{\beta^{((k+1)j)} - \beta^{(kj)}} \\ &\quad + \int_0^{c^{(kj)}} \frac{t}{c^{(kj)}} \left(1 + \frac{\beta^{(kj)} + \varepsilon^{(kj)}}{\beta^{((k+1)j)} - \beta^{(kj)}}\right) dt \\ &= -\frac{1}{2} c^{(kj)} \frac{\beta^{(kj)} + \varepsilon^{(kj)}}{\beta^{((k+1)j)} - \beta^{(kj)}} - (1-c^{(kj)}) \frac{\beta^{(kj)} + \varepsilon^{(kj)}}{\beta^{((k+1)j)} - \beta^{(kj)}} \\ &\quad + \frac{1}{2} c^{(kj)} \left(1 + \frac{\beta^{(kj)} + \varepsilon^{(kj)}}{\beta^{((k+1)j)} - \beta^{(kj)}}\right) \end{aligned}$$

$$
\begin{aligned}
&= \frac{1}{2}c^{(kj)} - (1-c^{(kj)})\frac{\beta^{(kj)}+\varepsilon^{(kj)}}{\beta^{((k+1)j)}-\beta^{(kj)}} \\
&= c^{(kj)}\Big(\frac{1}{2}+\frac{\beta^{(kj)}+\varepsilon^{(kj)}}{\beta^{((k+1)j)}-\beta^{(kj)}}\Big) - \frac{\beta^{(kj)}+\varepsilon^{(kj)}}{\beta^{((k+1)j)}-\beta^{(kj)}} \\
&= c^{(kj)}\frac{\beta^{((k+1)j)}+\beta^{(kj)}+2\varepsilon^{(kj)}}{2\beta^{((k+1)j)}-2\beta^{(kj)}} - \frac{\varepsilon^{(kj)}}{\beta^{((k+1)j)}-\beta^{(kj)}} - \frac{\beta^{(kj)}}{\beta^{((k+1)j)}-\beta^{(kj)}} \\
&= \frac{2\varepsilon^{(kj)}}{2\beta^{((k+1)j)}-2\beta^{(kj)}} - \frac{\varepsilon^{(kj)}}{\beta^{((k+1)j)}-\beta^{(kj)}} - \frac{\beta^{(kj)}}{\beta^{((k+1)j)}-\beta^{(kj)}} \\
&= -\frac{\beta^{(kj)}}{\beta^{((k+1)j)}-\beta^{(kj)}}. \qquad (4.91)
\end{aligned}
$$

Definiere für $\alpha \in [0,1]$

$$
\begin{aligned}
\nabla^2 f(x^{(k)}+\alpha(x^{(k+1)}-x^{(k)})) &= B^{(k)} + s_{kk}(\alpha)(1+\beta^{(kk)})(A^{(k+1)}-A^{(k)}) \\
&\quad - \sum_{j=k+1}^{K} s_{kj}(\alpha)(\beta^{((k+1)j)}-\beta^{(kj)})(A^{(j+1)}-A^{(j)}). \qquad (4.92)
\end{aligned}
$$

Dann gilt

$$
\begin{aligned}
\nabla^2 f(x^{(k)}) &= B^{(k)}, \\
\nabla^2 f(x^{(k+1)}) &= B^{(k+1)},
\end{aligned}
$$

das heißt $\nabla^2 f$ ist stetig. Außerdem ist ∇f^2 überall symmetrisch. Der zu einer Klammer

$$(A^{(j+1)}-A^{(j)}),\ j>k$$

gehörende Koeffizient der Matrix $\nabla^2 f(x^{(k)}+\alpha(x^{(k+1)}-x^{(k)}))$ lautet

$$-\beta^{(kj)} - s_{kj}(\alpha)(\beta^{((k+1)j)}-\beta^{(kj)})$$

und es gilt

$$
\begin{aligned}
-\beta^{(kj)} - s_{kj}(\alpha)(\beta^{((k+1)j)}-\beta^{(kj)}) &\leq -\beta^{(kj)}+\beta^{(kj)}+\varepsilon^{(kj)} = \varepsilon^{(kj)} < b^{(j)} \\
-\beta^{(kj)} - s_{kj}(\alpha)(\beta^{((k+1)j)}-\beta^{(kj)}) &\geq -\beta^{(kj)} - (\beta^{((k+1)j)}-\beta^{(kj)}) = -\beta^{((k+1)j)} > -b^{(j)}.
\end{aligned}
$$

Daher ist für jedes $\alpha \in [0,1]$ sowohl die Matrix

$$B^{(k)} + \sum_{j=k+1}^{K} s_{kj}(\alpha)(-\beta^{((k+1)j)}+\beta^{(kj)})(A^{(j+1)}-A^{(j)}),$$

als auch

$$B^{(k)} + (1+\beta^{(kk)})(A^{(k+1)}-A^{(k)}) + \sum_{j=k+1}^{K} s_{kj}(\alpha)(-\beta^{((k+1)j)}+\beta^{(kj)})(A^{(j+1)}-A^{(j)})$$

positiv definit und folglich muss auch der lineare Übergang (4.92) positiv definit sein. Es folgt außerdem, dass

$$\begin{aligned}
\int_0^1 \nabla^2 f(x^{(k)} + t(x^{(k+1)} - x^{(k)}))dt &= B^{(k)} + (1+\beta^{(kk)})(A^{(k+1)} - A^{(k)}) \int_0^1 s_{kk}(t)dt \\
&\quad + \sum_{j=k+1}^{K} (-\beta^{((k+1)j)} + \beta^{(kj)})(A^{(j+1)} - A^{(j)}) \int_0^1 s_{kj}(t)dt \\
&\overset{(4.90)}{\underset{(4.91)}{=}} B^{(k)} + \frac{(1+\beta^{(kk)})\beta^{(kk)}}{1+\beta^{(kk)}}(A^{(k+1)} - A^{(k)}) \\
&\quad + \sum_{j=k+1}^{K} \frac{(-\beta^{((k+1)j)} + \beta^{(kj)})(-\beta^{(kj)})}{\beta^{((k+1)j)} - \beta^{(kj)}}(A^{(j+1)} - A^{(j)}) \\
&= B^{(k)} + \sum_{j=k}^{K} \beta^{(kj)}(A^{(j+1)} - A^{(j)}) \\
&\overset{(4.89)}{=} A^{(k)}. \qquad (4.93)
\end{aligned}$$

Unter allen Funktionen mit der zweiten Ableitung $\nabla^2 f$ aus (4.92) wähle man eine mit

$$\nabla f(x^{(1)}) = g^{(1)}.$$

Alle bei der Konstruktion von $\nabla^2 f$ auftretenden Matrizen sind beschränkt. Insofern ist die zusätzliche Forderung der lokalen Lipschitz-Stetigkeit von $\nabla^2 f$ leicht zu erfüllen.
Nach dem Mittelwertsatz gilt

$$\begin{aligned}
\nabla f(x^{(k+1)}) - \nabla f(x^{(k)}) &= \int_0^1 \nabla^2 f(x^{(k)} + t(x^{(k+1)} - x^{(k)}))dt \cdot (x^{(k+1)} - x^{(k)}) \\
&\overset{(4.93)}{=} A^{(k)}(x^{(k+1)} - x^{(k)}) \\
&\overset{(4.87)}{=} \Delta g^{(k)}
\end{aligned}$$

und somit gilt für alle $k = 1, ..., K$

$$\nabla f(x^{(k)}) = g^{(k)}.$$

□

Wir kommen zum wichtigsten Resultat dieses Kapitels.

Satz 4.5.20. *Es existieren ein $\varepsilon > 0$ und ein $\delta > 0$, sowie Funktionen $\tilde{f}_{(p)},\ p = 1, ..., M$, so dass der Algorithmus 4.5.14 für jeden Startvektor $x^{(0)} \in \mathbb{R}^n$ mit*

$$\|x^{(0)} - x^*\| < \varepsilon$$

und jede symmetrische positiv definite Startmatrizen $H_p^{(0)} \in \mathbb{R}^{n\times n},\ p = 1, ..., M$ mit

$$\|H^{(0)} - \nabla^2 \tilde{f}_{(p)}(x_p^*)\| < \delta \qquad (4.94)$$

eine Folge $(x^{(k)})_{k\in\mathbb{N}} \subset \mathbb{R}^n$ erzeugt, die superlinear gegen x^* konvergiert.

Beweis. Wegen Satz 4.5.13 wissen wir, dass für ein hinreichend kleines $\varepsilon > 0$ stets Vektoren $u_{(p)},\ p = 1, ..., M$ gemäß (4.84) konstruierbar sind mit

$$u_{(p)}^T d_p > 0, \quad p = 1, ..., M.$$

Laut Satz 4.5.19 existieren zweimal stetig differenzierbare Funktionen $f_{(p)},\ p = 1, ..., M$ mit

$$\nabla f_{(p)}(x_p^{(k+1)}) - \nabla f_{(p)}(x_p^{(k)}) = u_{(p)}^{(k)},$$

deren zweite Ableitungen $\nabla^2 f_{(p)}$ positiv definit und lokal Lipschitz-stetig sind. Die Summe

$$\tilde{f}(x) := \sum_{p=1}^{M} f_{(p)}(x_p)$$

ist zwar nicht notwendigerweise gleich der Funktion f. Es gilt jedoch, dass $\tilde{f}$ die gleiche Blockstruktur (4.62) besitzt wie f. Des Weiteren ist

$$\nabla \tilde{f}(x^*) = 0$$

und $\nabla^2 \tilde{f}(x)$ in $U_\varepsilon(x^*)$ positiv definit. Die Voraussetzungen von Satz 4.4.10 sind demnach erfüllt. Die Folge $(x^{(k)})_{k\in\mathbb{N}} \subset \mathbb{R}^n$ konvergiert daher superlinear gegen x^*. □

Bemerkung 4.5.21.

a) Man beachte, dass die Funktionen $\tilde{f}_p,\ p = 1, ..., M$ zu keinem Zeitpunkt bekannt zu sein brauchen. Der Satz 4.5.20 weist lediglich nach, dass eine solche Zerlegung in Teilfunktionen existiert, für die quasi unwissentlich ein partitioniertes BFGS-Verfahren durchgeführt wird.

b) Da die Matrizen $\nabla^2 \tilde{f}_{(p)}(x_p^*)$ genauso unbekannt sind wie die Matrix $\nabla^2 f(x^*)$, sind die Voraussetzungen im Satz 4.5.20 nicht stärker als die Voraussetzungen des Satzes 4.2.13 über die superlineare Konvergenz des klassischen BFGS-Verfahrens. In den meisten Fällen begnügt man sich in der Praxis mit der Einheitsmatrix als Startmatrix und erzielt gute Ergebnisse, wie wir in Abschnitt 5.2 sehen werden.

4.5.3 Mehrfache Überschneidungen

4.5.3.1 Methodik

Die Struktur (4.62) aus Abschnitt 4.5.2 führt häufig dazu, dass eine Vielzahl von Nulleinträgen als Nichtnulleinträge behandelt werden, was sowohl für den Speicher- als auch für den Rechenaufwand als auch für die Genauigkeit des Verfahren schädlich ist. Mit einfachen Überschneidungen kann beispielsweise eine Bandmatrix (mit Bandbreite > 2) nur dadurch dargestellt werden, dass eine Vielzahl von Nulleinträgen als Nichtnulleinträge behandelt werden. In diesem Abschnitt betrachten wir den Fall, in dem sich mehr als zwei Blöcke überschneiden dürfen und versetzen uns somit in die Lage, die tatsächliche Struktur der Matrix genauer abzubilden. Die Matrix habe folgende Struktur.

$$H = \begin{bmatrix} \begin{bmatrix} & H_1 & \\ & & \begin{bmatrix} & H_2 & \\ & & \begin{bmatrix} \cdots & \\ & H_3 \end{bmatrix} \end{bmatrix} \end{bmatrix} & & & \\ & \ddots & & \\ & & \ddots & \\ & & & \ddots \end{bmatrix} \in \mathbb{R}^{n\times n}. \tag{4.95}$$

Die Punkte rechts unten in H_1 deuten an, dass hier auch mehr als doppelte Überschneidungen betrachtet werden. Im Übrigen sei vorausgesetzt, dass kein Block vollständig in einem anderen enthalten ist. Andernfalls würde der innere Block für die Beschreibung der Matrix ohnehin keinen echten Beitrag leisten. Wie man leicht sieht, kann diese Struktur eine Bandmatrix exakt abbilden.
Im folgenden wird das Update (4.68)-(4.69) auf diese Struktur erweitert. Wir verwenden auch hier die Indexmengen I_p. Bezeichne

$$p_i^{(\min)} := \min\{p \in \{1,..,M\} : \ i \in I_p\}.$$

Für $\eta \in \mathbb{R}^{M-1}$ seien $u_p,\ p = 1,...,M$ komponentenweise erklärt durch:

$$(u_{(p)})_i := \begin{cases} \Big(\prod\limits_{r=p_i^{(\min)}}^{p-1}(1-\eta_r)\Big)y_i, & i \in I_p \setminus I_{p+1} \\ \eta_p\Big(\prod\limits_{r=p_i^{(\min)}}^{p-1}(1-\eta_r)\Big)y_i, & i \in I_p \cap I_{p+1} \end{cases}. \tag{4.96}$$

Man beachte, dass $p_i^{(\min)} = p$ möglich ist, wodurch

$$\prod_{r=p_i^{(\min)}}^{p-1}(1-\eta_r) = 1$$

gilt. Auch für die $u_{(p)},\ p = 1, ..., M$ aus (4.96) gilt der Satz 4.5.1:

Satz 4.5.22. *Für $u_{(p)},\ p = 1, ..., M$ aus (4.96) und $v_{(p)} := -H_{(p)}d_p$ erfüllt die aktualisierte Hessematrix*

$$\begin{aligned} H^+ &:= \sum_{p=1}^{M} \overline{H^+_{(p)}} \\ H^+_{(p)} &= H_{(p)} + \frac{u_{(p)}{u_{(p)}}^T}{{u_{(p)}}^T d_p} + \frac{v_{(p)}{v_{(p)}}^T}{{v_{(p)}}^T d_p} \end{aligned}$$

die Quasi-Newton-Bedingung

$$H^+ d = y.$$

Beweis. Nachzuweisen ist lediglich, dass

$$\sum_{p=1}^{M} \overline{\Big(\frac{u_{(p)}u_{(p)}^T}{u_{(p)}^T d_p}\Big)} d = y.$$

Wieder gilt:

$$\sum_{p=1}^{M} \overline{\Big(\frac{u_{(p)}{u_{(p)}}^T}{{u_{(p)}}^T d_p}\Big)} d = \sum_{p=1}^{M} \overline{\left(\frac{u_{(p)}{u_{(p)}}^T}{{u_{(p)}}^T d_p} d_p\right)} = \sum_{p=1}^{M} \overline{u_{(p)}}$$

Diese Summe wird nun komponentenweise berechnet. Für die Komponente $i \in \{1, ..., n\}$ gelte

$$i \in I_{p_1} \cap I_{p_2} \cap ... \cap I_{p_k}.$$

Dann gilt

$$\begin{aligned} \sum_{p=1}^{M} (u_{(p)})_i &= y_i \left(\sum_{j=1}^{k-1} \left(\eta_{p_j} \prod_{r=1}^{j-1} (1 - \eta_{p_r}) \right) + \prod_{r=1}^{k-1} (1 - \eta_{p_r}) \right) \\ &= y_i \left(\sum_{j=1}^{k-2} \left(\eta_{p_j} \prod_{r=1}^{j-1} (1 - \eta_{p_r}) \right) + \left(\underbrace{\big(\eta_{p_{k-1}} + (1 - \eta_{p_{k-1}})\big)}_{=1} \prod_{r=1}^{k-2} (1 - \eta_{p_r}) \right) \right) \\ &= y_i \left(\sum_{j=1}^{k-2} \left(\eta_{p_j} \prod_{r=1}^{j-1} (1 - \eta_{p_r}) \right) + \prod_{r=1}^{k-2} (1 - \eta_{p_r}) \right) \\ &= y_i \big(\eta_{p_1} + (1 - \eta_{p_1})\big) \\ &= y_i \end{aligned}$$

□

Bemerkung 4.5.23. Ist die Struktur wie in Abschnitt 4.5.2 gegeben, unterscheidet sich das dort definierte $u_{(p)}$ nicht von dem in (4.96).

Wegen

$$\sum_{p=1}^{M} u_{(p)}^T d_p = y^T d$$

leuchtet ein, dass auch im Falle mehrfacher Überschneidungen die einzelnen Blöcke nur dann positiv definit sein können, falls

$$y^T d > 0 \tag{4.97}$$

gilt. Andernfalls kann auch hier der Vektor q aus (4.13) anstatt von y verwendet werden. Wieder gilt allerdings, dass aus (4.97) noch nicht folgt, dass

$$u_{(p)}^T d_p > 0, \quad p = 1, ..., M, \tag{4.98}$$

was positiv definite Blöcke garantieren würde, wie wir aus Abschnitt 4.2 wissen.
Der folgende Satz ist das Analogon zu Satz 4.5.2 aus Abschnitt 4.5.2.

Satz 4.5.24. *Seien $q, d \in \mathbb{R}^n$ mit $q^T d > 0$. Seien*

$$I_p, \quad p = 1, ..., M,$$

die gemäß (4.63) zur Matrix H aus (4.95) definierten Indexmengen. Bezeichne

$$\tilde{p}_j \in \{1, ..., M-1\}, \quad j = 1, ..., \tilde{M}$$

diejenigen Blöcke, für die gilt

$$\forall s < \tilde{p}_j, \forall t \geq \tilde{p}_j : \quad q_{s \cap t} d_{s \cap t} = 0. \tag{4.99}$$

Diese seien aufsteigend sortiert und $\tilde{p}_0$ und $\tilde{p}_{M+1}$ so definiert, dass

$$0 =: \tilde{p}_0 < \tilde{p}_1 < \tilde{p}_2 < ... < \tilde{p}_{\tilde{M}} \leq \tilde{p}_{\tilde{M}+1} := M$$

gilt.
Dann existiert ein Vektor $\eta \in \mathbb{R}^{M-1}$, so dass (4.98) gilt, falls

$$q_{\tilde{\mathcal{J}}_j}^T d_{\tilde{\mathcal{J}}_j} > 0, \quad j = 0, ..., \tilde{M} \tag{4.100}$$

wobei

$$\tilde{\mathcal{J}}_j := \bigcup_{p=\tilde{p}_j+1}^{\tilde{p}_{j+1}} I_p, \quad j = 0, 1, 2, ..., \tilde{M}.$$

Beweis. Der Beweis verläuft sehr ähnlich wie der von Satz 4.5.2. Wieder betrachte man ein beliebiges $j \in \{0, ..., \tilde{M}\}$. Sei $v \in \mathbb{R}_+^{\tilde{p}_{j+1} - \tilde{p}_j}$ ein beliebiger Vektor positiver Zahlen, für den gilt

$$\sum_{i=1}^{\tilde{p}_{j+1} - \tilde{p}_j} v_i = q_{\tilde{\mathcal{J}}_j}^T d_{\tilde{\mathcal{J}}_j}.$$

Wir formulieren das zu (4.77)-(4.79) analoge Gleichungssystem. Aufgrund der Eigenschaft von $\tilde{p}_j$ lautet es:

$$q_{(\tilde{p}_j+1) \setminus (\tilde{p}_j+2)}^T d_{(\tilde{p}_j+1) \setminus (\tilde{p}_j+2)} + \eta_{\tilde{p}_j+1} q_{(\tilde{p}_j+1) \cap (\tilde{p}_j+2)}^T d_{(\tilde{p}_j+1) \cap (\tilde{p}_j+2)} = v_1, \tag{4.101}$$

$$\sum_{i\in I_p\setminus I_{p+1}}\Big(\prod_{r=\overline{p_i^{(\min)}}}^{p-1}(1-\eta_r)\Big)y_i^Td_i+\sum_{i\in I_p\cap I_{p+1}}\eta_p\Big(\prod_{r=\overline{p_i^{(\min)}}}^{p-1}(1-\eta_r)\Big)y_i^Td_i=v_{p-\tilde{p}_j} \tag{4.102}$$

$$\sum_{i\in I_p\setminus I_{p+1}}\Big(\prod_{r=\overline{p_i^{(\min)}}}^{p-1}(1-\eta_r)\Big)y_i^Td_i=v_{\tilde{p}_{j+1}-\tilde{p}_j} \tag{4.103}$$

mit $p=\tilde{p}_j+2,...,\tilde{p}_{j+1}-1$ und

$$\overline{p_i^{(\min)}}:=\min(\tilde{p}_j,p_i^{(\min)}).$$

Dieses Gleichungssystem kann in gleicher Weise wie (4.77)-(4.79) gelöst werden: (4.101) definiert $\eta_{\tilde{p}_j+1}$ auf eindeutige Weise. Durch sukzessive Lösung der Gleichung (4.102) für $p=\tilde{p}_j+2,...,\tilde{p}_{j+1}-1$, erhält man alle weiteren η_p. Aufgrund der Konstruktion von v, ist (4.103) folglich automatisch erfüllt.
Wendet man dieses Verfahren für alle $j=0,...,\tilde{M}$ an, so bleiben wieder nur die $\eta_{\tilde{p}_j},\ j=\{1,...,\hat{M}\}$ zunächst unbestimmt, für die

$$q^T_{\hat{p}_j\cap(\hat{p}_j+1)}d_{\hat{p}_j\cap(\hat{p}_j+1)}=0$$

gilt und die daher unerheblich sind. Die Existenz des Vektors η ist daher nachgewiesen. □

Analog zu Satz 4.5.3 formulieren wir den

Satz 4.5.25. *Sei $q^Td>0$ und $H_{(p)},\ p=1,...,M$ positiv definit. Die Bedingung (4.100) aus Satz 4.5.24 sei nicht erfüllt. Somit existiert ein Block $\tilde{p}_j$ in der Matrix so dass (4.99) gilt und (4.100) nicht gilt. Man ersetze q_p durch*

$$\tilde{q}_p:=\theta q_p+(1-\theta)H_{(p)}d_p,\quad \tilde{p}_j<p\leq\tilde{p}_{j+1} \tag{4.104}$$

mit

$$\theta=\frac{\sigma\cdot\sum_{p=\tilde{p}_j+1}^{\tilde{p}_{j+1}}d_p^TH_{(p)}d_p}{\sum_{p=\tilde{p}_j+1}^{\tilde{p}_{j+1}}\big(d_p^TH_{(p)}d_p-q_p^Td_p\big)},$$

und $\sigma\in(0,1)$. Dann gilt

$$\tilde{q}^T_{\tilde{\mathcal{J}}_j}d_{\tilde{\mathcal{J}}_j}>0.$$

Beweis. Siehe Beweis von Satz 4.5.3. □

Die Bemerkung 4.5.4 gilt auch hier. Analog zu Satz 4.5.11 gilt für die Struktur (4.95) der

Satz 4.5.26. *Seien $n_1,n_2,n_3,n_4,n_5\in\mathbb{N}$ und $A\in\mathbb{R}^{n_1\times n_1}$, $B\in\mathbb{R}^{n_1\times n_2}$, $C\in\mathbb{R}^{n_1\times n_3}$, $D\in\mathbb{R}^{n_2\times n_2}$, $E\in\mathbb{R}^{n_2\times n_3}$, $F\in\mathbb{R}^{n_3\times n_3}$, $G\in\mathbb{R}^{n_2\times n_4}$, $H\in\mathbb{R}^{n_3\times n_4}$, $J\in\mathbb{R}^{n_4\times n_4}$, $K\in\mathbb{R}^{n_3\times n_5}$,*

$L \in \mathbb{R}^{n_4 \times n_5}$, $M \in \mathbb{R}^{n_5 \times n_5}$. A, D, F, J und M seien symmetrisch. Die Matrix

$$X := \begin{bmatrix} A & B & C & 0 & 0 \\ B^T & D & E & G & 0 \\ C^T & E^T & F & H & K \\ 0 & G^T & H^T & J & L \\ 0 & 0 & K^T & L^T & M \end{bmatrix}$$

sei positiv definit. Dann existieren Matrizen $D_1, D_2, E_1, E_2, F_1, F_2, F_3, H_1, H_2, J_1, J_2$ mit den entsprechenden Dimensionen und mit

$$\begin{aligned} D_1 + D_2 &= D, \\ E_1 + E_2 &= E, \\ F_1 + F_2 + F_3 &= F, \\ H_1 + H_2 &= H, \\ J_1 + J_2 &= J, \end{aligned}$$

so dass die Matrizen

$$\begin{bmatrix} A & B & C \\ B^T & D_1 & E_1 \\ C^T & E_1^T & F_1 \end{bmatrix},$$

$$\begin{bmatrix} D_2 & E_2 & G \\ E_2^T & F_2 & H_1 \\ G^T & H_1^T & J_1 \end{bmatrix}$$

und

$$\begin{bmatrix} F_3 & H_2 & K \\ H_2^T & J_2 & L \\ K^T & L^T & M \end{bmatrix}$$

positiv definit sind.

Beweis. Nach Satz 4.5.11 können wir H_1, H_2, J_1, J_2 sowie $\tilde{F}_2$ und F_3 finden mit

$$\begin{aligned} \tilde{F}_2 + F_3 &= F, \\ H_1 + H_2 &= H, \\ J_1 + J_2 &= J \end{aligned}$$

so dass

$$\begin{bmatrix} D & E & G \\ E^T & \tilde{F}_2 & H_1 \\ G^T & H_1^T & J_1 \end{bmatrix}$$

und

$$\begin{bmatrix} F_3 & H_2 & K \\ H_2^T & J_2 & L \\ K^T & L^T & M \end{bmatrix}$$

positiv definit sind. Genauso können nach Satz 4.5.11 für die Matrix

$$\begin{bmatrix} A & B & C & 0 \\ B^T & D & E & G \\ C^T & E^T & \tilde{F}_2 & H_1 \\ 0 & G^T & H_1^T & J_1 \end{bmatrix}$$

Matrizen $D_1, D_2, E_1, E_2, F_1, F_2$ finden mit

$$\begin{aligned} D_1 + D_2 &= D, \\ E_1 + E_2 &= E \\ F_1 + F_2 &= \tilde{F}_2 \end{aligned}$$

so dass

$$\begin{bmatrix} A & B & C \\ B^T & D_1 & E_1 \\ C^T & E_1^T & F_1 \end{bmatrix}$$

und

$$\begin{bmatrix} D_2 & E_2 & G \\ E_2^T & F_2 & H_1 \\ G^T & H_1^T & F_2 \end{bmatrix}$$

positiv definit sind. □

Korollar 4.5.27. *Analog zu Korrolar 4.5.12 kann auch hier der allgemeine Fall mit beliebig vielen mehrfachen Überschneidungen gefolgert werden.*

Daher gilt auch hier:

Satz 4.5.28. *f habe die Form (4.95). Dann existiert ein $\varepsilon > 0$, so dass für alle $x^+, x \in U_\varepsilon(x^*)$ stets ein Vektor $\eta \in \mathbb{R}^{M-1}$, so dass für die Vektoren $u_{(p)},\ p = 1, ..., M$ aus (4.96)*

$$u_{(p)}^T d_p > 0$$

gilt.

Beweis. Siehe Beweis von Satz 4.5.13 □

Somit erhalten wir wieder ein dünn besetztes BFGS-artiges Update, welches stets positiv definit ist und in einer Umgebung um x^* die Quasi-Newton-Bedingung erfüllt.
Wir gehen wieder davon aus, dass $x^{(0)}$ eine hinreichend gute Startschätzung sei und folglich, nach Satz 4.5.28 die Verwendung von q beziehungsweise q_p und $\tilde{q}_p$ aus (4.104) unnötig ist. Somit formulieren wir den folgenden

Algorithmus 4.5.29.

i. Identifiziere eine Struktur der Form (4.95).

ii. Wähle $x^{(0)} \in \mathbb{R}^n$ *und* $H_{(p)}^{(0)} \in \mathbb{R}^{n_p \times n_p}$, $p = 1, ..., M$ *symmetrisch und positiv definit und setze*

$$H^{(0)} := \sum_{p=1}^{M} H_{(p)}^{(0)}.$$

Wähle außerdem $\varepsilon \geq 0$ *und setze* $k := 0$.

iii. Ist $\|\nabla f(x^{(k)})\| \leq \varepsilon$. *STOP.*

iv. Bestimme $d^{(k)}$ *aus dem Gleichungssystem*

$$H^{(k)} d^{(k)} = -\nabla f(x^{(k)}).$$

v. Setze

$$\begin{aligned} x^{(k+1)} &:= x^{(k)} + d^{(k)} \text{ und} \\ y^{(k)} &:= \nabla f(x^{(k+1)}) - \nabla f(x^{(k)}). \end{aligned}$$

vi. Bestimme $\eta^{(k)} \in \mathbb{R}^{M-1}$, *so dass für alle Blöcke*

$$u_{(p)}^{(k)T} d_p^{(k)} > 0$$

gilt, mit $u_{(p)}^{(k)}$ *aus (4.96).*

vii. Berechne

$$H_{(p)}^{(k+1)} = H_{(p)}^{(k)} + \frac{y_p^{(k)} y_p^{(k)T}}{y_p^{(k)T} d_p^{(k)}} - \frac{(H_{(p)}^{(k)} d_p^{(k)})(H_{(p)}^{(k)} d_p^{(k)})^T}{d_p^{(k)T} H_{(p)}^{(k)} d_p^{(k)}}, \quad p = 1, ..., M$$

und berechne die gesamte Matrix durch

$$H^{(k+1)} = \sum_{p=1}^{M} \overline{H_{(p)}^{(k+1)}}.$$

viii. Setze k:=k+1 und gehe zu iii.

Bemerkung 4.5.30. Schritt *vi.* kann beispielsweise wie im Beweis von Satz 4.5.24 erfolgen.

Wir haben die mehrfachen Überschneidungen eingangs unter anderem mit dem Speicher motiviert, der unnötigerweise besetzt wird, falls Nulleinträge mit in die Struktur der Blöcke aufgenommen und somit als Nichtnulleinträge behandelt werden. Der Vollständigkeit halber sei an dieser Stelle darauf hingewiesen, dass eine Struktur mit mehrfachen Überschneidungen, welche im Vergleich zu den einfachen Überschneidungen die Struktur genauer abbildet, unter Umständen sogar mehr Speicherplatz benötigt. Betrachte dazu das folgende

Beispiel 4.5.31. Sei

$$H := \begin{bmatrix} \times & \times & \times & 0 & 0 \\ \times & \times & \times & \times & 0 \\ \times & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \\ 0 & 0 & \times & \times & \times \end{bmatrix}.$$

Bei einfacher Überschneidung muss zwangsläufig eine Null in die Blockstruktur aufgenommen werden und es entstehen zwei Blöcke, einer der Dimension 4×4, einer der Dimension 3×3. Bei zugelassener mehrfachen Überschneidung, kann die Struktur in diesem Beispiel exakt abgebildet werden. Es resultieren drei 3×3 Blöcke. Wie im Algorithmus 4.5.29 beschrieben, werden im Schritt vii zunächst die einzelnen Blöcke aufaddiert. Daher ist es notwendig, die Blöcke einzeln zu speichern. Dies macht demnach insgesamt einen Speicheraufwand von 27 Werten im Falle der mehrfachen Überschneidungen und von lediglich 25 im Falle einfacher Überschneidungen.
Abgesehen davon, dass ein solches Verhältnis nicht die Regel ist, gibt es, wie ebenfalls bereits erwähnt, weitere gute Gründe, die Struktur möglichst exakt abzubilden.

4.5.3.2 Konvergenzeigenschaften des Block-BFGS-Verfahrens mit mehrfachen Überschneidungen

Die Konvergenzaussage deckt sich mit der aus Abschnitt 4.5.2.2. Liegen alle Nichtnulleinträge von $\nabla^2 f$ innerhalb einer Blockstruktur der Form (4.95), gilt unter ansonsten gleichen Voraussetzungen wie in Abschnitt 4.5.2.2 der

Satz 4.5.32. *Es existieren ein $\varepsilon > 0$ und ein $\delta > 0$, sowie Funktionen $\tilde{f}_p,\ p = 1, ..., M$, so dass der Algorithmus 4.5.14 für jeden Startvektor $x^{(0)} \in \mathbb{R}^n$ mit*

$$\|x^{(0)} - x^*\| < \varepsilon$$

und jede symmetrische positiv definite Startmatrizen $H_p^{(0)} \in \mathbb{R}^{n \times n},\ p = 1, ..., M$ mit

$$\|H^{(0)} - \nabla^2 \tilde{f}_p(x_p^*)\| < \delta \tag{4.105}$$

eine Folge $(x^{(k)})_{k \in \mathbb{N}} \subset \mathbb{R}^n$ erzeugt, die superlinear gegen x^* konvergiert.

Beweis. Im Beweis von Satz 4.5.20 tritt die spezielle Struktur (4.95) nicht auf. Daher kann der gleiche Beweis an dieser Stelle geführt werden, um Satz 4.5.32 zu beweisen. □

4.5.4 Variablen-Permutation

Die mehrfachen Überschneidungen haben wir zu Gunsten einer besseren Abbildung der Nichtnullstruktur der Originalmatrix eingeführt. Im Falle ungünstig gelegener Einträge innerhalb der Matrix, kann jedoch auch mit zugelassenen mehrfachen Überschneidungen häufig nur eine sehr schlechte Approximation der Struktur der Originalmatrix erreicht werden. Der

schlimmste Fall lautet wie folgt. Die Originalmatrix habe die Struktur

$$\nabla^2 f(x^*) = \begin{bmatrix} \times & 0 & \cdot & \cdot & \cdot & 0 & \times \\ 0 & \times & 0 & \cdot & \cdot & \cdot & 0 \\ \cdot & 0 & \times & 0 & \cdot & \cdot & 0 \\ \cdot & \cdot & 0 & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ 0 & \cdot & \cdot & \cdot & \cdot & \cdot & 0 \\ \times & 0 & 0 & \cdot & \cdot & 0 & \times \end{bmatrix}. \tag{4.106}$$

Die Diagonale ist zwangsläufig besetzt, da andernfalls die positive Definitheit nicht gegeben wäre.
Unabhängig davon, ob mehrfache Überschneidungen zugelassen werden oder nicht, entsteht eine vollbesetzte Matrix, obwohl die tatsächliche Matrix in hohem Maße dünnbesetzt ist. Dieses Problem kann dadurch bewältigt werden, dass man die Variablen permutiert. Durch Vertauschen der Indizierung der Variablen x_2 und x_n in dem zu (4.106) gehörenden Optimierungsproblem, entsteht die Hessematrix

$$\nabla^2 f(x^*) = \begin{bmatrix} \times & \times & 0 & \cdot & \cdot & \cdot & 0 \\ \times & \times & 0 & \cdot & \cdot & \cdot & 0 \\ 0 & 0 & \times & 0 & \cdot & \cdot & 0 \\ \cdot & \cdot & 0 & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 0 \\ 0 & 0 & 0 & \cdot & \cdot & 0 & \times \end{bmatrix},$$

die wiederum sowohl mit zugelassener Einfach- als auch Mehrfachüberschneidung exakt abgebildet werden kann. Tatsächlich braucht die Permutation der Variablen überhaupt nicht in der Problemformulierung zu erfolgen. Es genügt, wenn die Vertauschung lediglich innerhalb des BFGS-Updates vorgenommen wird. Es leuchtet ein, dass die optimale Permutation, die zu der minimalen Anzahl an Nulleinträgen innerhalb der Blockstruktur führt, schwer zu ermitteln ist. Dieses Problem ist vermutlich NP-vollständig (s. Abschnitt 3.2.1). Als erste Näherung könnte man die Matrix so permutieren, dass eine Bandmatrix mit möglichst geringer Bandbreite entsteht. Dazu existiert eine Vielzahl von Permutationsverfahren, die dies anstreben. Auch diese Verfahren sind jedoch lediglich heuristische Verfahren und liefern bedauerlicherweise nicht immer die optimale Bandbreite. Eines dieser Verfahren ist der *Reverse Cuthill-McKee (RCM)*-Algorithmus [20]. Für eine Start-Variable $k \in \{1, ..., n\}$ lautet dieser:

Algorithmus 4.5.33.

i. Für $i = 1, ..., n$ setze $P(i) := 0$.

ii. Setze $j := 1$.

iii. Setze $\tilde{P}(k) := j$ und $j := j + 1$.

iv. Für $i = 1, ..., n$ finde alle n_i Zeilenindizes für diejenigen Spalten i, für die $\tilde{P}(i) = 0$ gilt und sortiere sie in aufsteigender Reihenfolge in $\tilde{P}(j : j + n_i - 1)$ und setze $j := j + n_i$.

v. Für $i = 1, ..., n$ setze $P(i) = n - \tilde{P}(i) + 1$

Das resultierende P ist die zu wählende Permutationsreihenfolge. Sollte auch durch Permutation keine zufrieden stellende Abbildung der Matrixstruktur erreicht werden können, gibt es wie in Abschnitt 4.5.1 bereits angesprochen, noch die Möglichkeit, einen Nichtnulleintrag, welcher für die Dünnbesetztheit der Approximation besonders hinderlich ist, als Nulleintrag zu behandeln. Wie bereits in Abschnitt 4.5.1 erwähnt, lässt sich für ein solches Verfahren wohl keine Konvergenzaussage treffen.
Im folgenden Abschnitt stellen wir jedoch ein Verfahren vor, welches das Ignorieren von Nichtnulleinträgen überflüssig macht.

4.6 Verallgemeinertes Sparse BFGS

4.6.1 Methodik

Den Gedanken von Abschnitt 4.5 weiterführend, wird in diesem Abschnitt das Block-BFGS-Verfahren in einer Weise verallgemeinert, dass die Struktur der Hessematrix exakt abgebildet werden kann. In Abschnitt 4.5 wurden die Blöcke lediglich derartig konstruiert, dass alle Nichtnulleinträge der tatsächlichen Hessematrix innerhalb der Nichtnullstruktur der Block-BFGS-Matrix liegen, wobei es jedoch zu Gunsten der Blockstruktur zulässig war, konstante Nulleinträge als Nichtnulleinträge zu betrachten. Das Vorgehen des hier vorgestellten Ansatzes wird anhand zweier Beispiele illustriert. Zum besseren Verständnis werden die Einträge hier exakt benannt und nicht durch ein "×" symbolisiert.

Beispiel 4.6.1. Als Minimalbeispiel betrachte man die Funktion $f : \mathbb{R}^4 \to \mathbb{R}$, für deren Hessematrix

$$\nabla^2 f(x) = \begin{bmatrix} \frac{\partial^2 f(x)}{\partial x_1^2} & \frac{\partial^2 f(x)}{\partial x_1 \partial x_2} & 0 & 0 \\ \frac{\partial^2 f(x)}{\partial x_1 \partial x_2} & \frac{\partial^2 f(x)}{\partial x_2^2} & \frac{\partial^2 f(x)}{\partial x_2 \partial x_3} & \frac{\partial^2 f(x)}{\partial x_2 \partial x_4} \\ 0 & \frac{\partial^2 f(x)}{\partial x_2 \partial x_3} & \frac{\partial^2 f(x)}{\partial x_3^2} & 0 \\ 0 & \frac{\partial^2 f(x)}{\partial x_2 \partial x_4} & 0 & \frac{\partial^2 f(x)}{\partial x_4^2} \end{bmatrix}, \quad x \in \mathbb{R}^4 \tag{4.107}$$

gelte. Auch mit Permutationen kann unter Verwendung der bisher entwickelten Verfahren die Struktur nicht exakt abgebildet werden. Der verallgemeinerte Block-BFGS-Ansatz betrachtet die Blöcke

$$\begin{bmatrix} \frac{\partial^2 f(x)}{\partial x_1^2} & \frac{\partial^2 f(x)}{\partial x_1 \partial x_2} \\ \frac{\partial^2 f(x)}{\partial x_1 \partial x_2} & \frac{\partial^2 f(x)}{\partial x_2^2} \end{bmatrix},$$

$$\begin{bmatrix} \frac{\partial^2 f(x)}{\partial x_2^2} & \frac{\partial^2 f(x)}{\partial x_2 \partial x_3} \\ \frac{\partial^2 f(x)}{\partial x_2 \partial x_3} & \frac{\partial^2 f(x)}{\partial x_3^2} \end{bmatrix}$$

und

$$\begin{bmatrix} \frac{\partial^2 f(x)}{\partial x_2^2} & \frac{\partial^2 f(x)}{\partial x_2 \partial x_4} \\ \frac{\partial^2 f(x)}{\partial x_2 \partial x_4} & \frac{\partial^2 f(x)}{\partial x_4^2} \end{bmatrix}$$

und vollzieht wie in den vorherigen Abschnitten ein Update auf die einzelnen Blöcke und fügt diese Blöcke gegebenenfalls durch Summation zusammen. In diesem Beispiel muss tatsächlich nur beim Eintrag $\frac{\partial^2 f(x)}{\partial x_2^2}$ summiert werden.

Die resultierenden Blöcke haben jedoch nicht notwendigerweise die Größe 2×2, wie im Beispiel der Matrix (4.107) gegeben.

Beispiel 4.6.2. Die Funktion $f : \mathbb{R}^7 \to \mathbb{R}$ habe die Hessematrix

$$\nabla^2 f(x) = \begin{bmatrix} \frac{\partial^2 f(x)}{\partial x_1^2} & 0 & 0 & \frac{\partial^2 f(x)}{\partial x_1 \partial x_4} & \frac{\partial^2 f(x)}{\partial x_1 \partial x_5} & 0 & 0 \\ 0 & \frac{\partial^2 f(x)}{\partial x_2^2} & 0 & \frac{\partial^2 f(x)}{\partial x_2 \partial x_4} & \frac{\partial^2 f(x)}{\partial x_2 \partial x_5} & \frac{\partial^2 f(x)}{\partial x_2 \partial x_6} & 0 \\ 0 & 0 & \frac{\partial^2 f(x)}{\partial x_3^2} & 0 & \frac{\partial^2 f(x)}{\partial x_3 \partial x_5} & 0 & \frac{\partial^2 f(x)}{\partial x_3 \partial x_7} \\ \frac{\partial^2 f(x)}{\partial x_1 \partial x_4} & \frac{\partial^2 f(x)}{\partial x_2 \partial x_4} & 0 & \frac{\partial^2 f(x)}{\partial x_4^2} & \frac{\partial^2 f(x)}{\partial x_4 \partial x_5} & \frac{\partial^2 f(x)}{\partial x_4 \partial x_6} & 0 \\ \frac{\partial^2 f(x)}{\partial x_1 \partial x_5} & \frac{\partial^2 f(x)}{\partial x_2 \partial x_5} & \frac{\partial^2 f(x)}{\partial x_3 \partial x_5} & \frac{\partial^2 f(x)}{\partial x_4 \partial x_5} & \frac{\partial^2 f(x)}{\partial x_5^2} & \frac{\partial^2 f(x)}{\partial x_5 \partial x_6} & \frac{\partial^2 f(x)}{\partial x_5 \partial x_7} \\ 0 & \frac{\partial^2 f(x)}{\partial x_2 \partial x_6} & 0 & \frac{\partial^2 f(x)}{\partial x_4 \partial x_6} & \frac{\partial^2 f(x)}{\partial x_5 \partial x_6} & \frac{\partial^2 f(x)}{\partial x_6^2} & 0 \\ 0 & 0 & \frac{\partial^2 f(x)}{\partial x_3 \partial x_7} & 0 & \frac{\partial^2 f(x)}{\partial x_5 \partial x_7} & 0 & \frac{\partial^2 f(x)}{\partial x_7^2} \end{bmatrix}, \quad x \in \mathbb{R}^7$$

Betrachtet werden die folgenden Blöcke:

$$\begin{bmatrix} \frac{\partial^2 f(x)}{\partial x_1^2} & \frac{\partial^2 f(x)}{\partial x_1 \partial x_4} & \frac{\partial^2 f(x)}{\partial x_1 \partial x_5} \\ \frac{\partial^2 f(x)}{\partial x_1 \partial x_4} & \frac{\partial^2 f(x)}{\partial x_4^2} & \frac{\partial^2 f(x)}{\partial x_4 \partial x_5} \\ \frac{\partial^2 f(x)}{\partial x_1 \partial x_5} & \frac{\partial^2 f(x)}{\partial x_4 \partial x_5} & \frac{\partial^2 f(x)}{\partial x_5^2} \end{bmatrix},$$

$$\begin{bmatrix} \frac{\partial^2 f(x)}{\partial x_2^2} & \frac{\partial^2 f(x)}{\partial x_2 \partial x_4} & \frac{\partial^2 f(x)}{\partial x_2 \partial x_5} & \frac{\partial^2 f(x)}{\partial x_2 \partial x_6} \\ \frac{\partial^2 f(x)}{\partial x_2 \partial x_4} & \frac{\partial^2 f(x)}{\partial x_4^2} & \frac{\partial^2 f(x)}{\partial x_4 \partial x_5} & \frac{\partial^2 f(x)}{\partial x_4 \partial x_6} \\ \frac{\partial^2 f(x)}{\partial x_2 \partial x_5} & \frac{\partial^2 f(x)}{\partial x_4 \partial x_5} & \frac{\partial^2 f(x)}{\partial x_5^2} & \frac{\partial^2 f(x)}{\partial x_5 \partial x_6} \\ \frac{\partial^2 f(x)}{\partial x_2 \partial x_6} & \frac{\partial^2 f(x)}{\partial x_4 \partial x_6} & \frac{\partial^2 f(x)}{\partial x_5 \partial x_6} & \frac{\partial^2 f(x)}{\partial x_6^2} \end{bmatrix}$$

und

$$\begin{bmatrix} \frac{\partial^2 f(x)}{\partial x_3^2} & \frac{\partial^2 f(x)}{\partial x_3 \partial x_5} & \frac{\partial^2 f(x)}{\partial x_3 \partial x_7} \\ \frac{\partial^2 f(x)}{\partial x_3 \partial x_5} & \frac{\partial^2 f(x)}{\partial x_5^2} & \frac{\partial^2 f(x)}{\partial x_5 \partial x_7} \\ \frac{\partial^2 f(x)}{\partial x_3 \partial x_7} & \frac{\partial^2 f(x)}{\partial x_5 \partial x_7} & \frac{\partial^2 f(x)}{\partial x_7^2} \end{bmatrix}.$$

Formal definieren wir wie folgt:

Definition 4.6.3. Sei $X \in \mathbb{R}^{n \times n}$. Zu einer Indexmenge $I \subset \{1, ..., n\}$ der Kardinalität n_I ist die Matrix $X_I \in \mathbb{R}^{n_I \times n_I}$ ein *verallgemeinerter Block* falls gilt

$$(X_I)_{ij} = X_{ij}, \quad \text{falls } i, j \in I,$$

wobei die Indizierung der Matrix X_I wieder über die Indizes der Matrix X erfolge, weswegen X_I für die Indizes ij mit

$$i \notin I \vee j \notin I$$

undefiniert bleibt.

Bemerkung 4.6.4. Sei ν die Anzahl der Nichtnullelemente der Matrix $\nabla^2 f(x),\ x \in \mathbb{R}$. Für die Anzahl M der entstehenden Blöcke gilt

$$M \leq \frac{\nu - n}{2}.$$

In diesem Fall induziert jeder Nichtnulleintrag unterhalb beziehungsweise oberhalb der Diagonalen $\frac{\partial^2 f(x)}{\partial x_i \partial x_j},\ i < j$ einen eigenen 2×2-Block.

Wie wir sehen werden, ist es nicht sinnvoll, lediglich 2×2-Blöcke zu betrachten, wenn die Struktur der Matrix auch größere Unterblöcke zulässt. Zumindest ist aber klar, dass die Matrix in der beschriebenen Weise in Blöcke unterteilt werden kann, so dass alle Nichtnulleinträge und keine Nulleinträge in den Blöcken liegen.
Das Update erfolgt prinzipiell wie in Abschnitt 4.5.3. Der wesentliche Unterschied ist lediglich, dass hier

$$I_{p \cap q_1} = \emptyset$$
$$I_{p \cap q_2} \neq \emptyset$$

mit

$$p < q_1 < q_2.$$

möglich ist. Mit anderen Worten: Eine Überschneidung zwischen zwei Blöcken setzt nicht voraus, dass eine Überschneidung mit allen Blöcken "dazwischen" existiert.
Die Matrix sei in M verallgemeinerte Unterblöcke unterteilt. Wie gehabt sei

$$y := \nabla f(x^+) - \nabla f(x).$$

$u_{(p)},\ p = 1, ..., M$ sei für $i \in I_p$ komponentenweise erklärt durch:

$$(u_{(p)})_i := \begin{cases} \eta_p \Big(\prod\limits_{\{r<p:\ i \in I_r\}} (1 - \eta_r) \Big) y_i, & \text{falls } \exists q > p :\ i \in I_{q \cap p} \\ \Big(\prod\limits_{\{r<p:\ i \in I_r\}} (1 - \eta_r) \Big) y_i, & \text{sonst.} \end{cases} \tag{4.108}$$

Korollar 4.6.5. *Für $u_{(p)},\ p = 1, ..., M$ aus (4.108) und*

$$v_{(p)} := -H_{(p)} d_p$$

erfüllt die aktualisierte Hessematrix

$$\begin{aligned} H^+ &:= \sum_{p=1}^{M} \overline{H^+_{(p)}}, \\ H^+_{(p)} &= H_{(p)} + \frac{u_{(p)} {u_{(p)}}^T}{{u_{(p)}}^T d_p} + \frac{v_{(p)} {v_{(p)}}^T}{{v_{(p)}}^T d_p} \end{aligned}$$

die Sekantenbedingung $H^+ d = y$.

Beweis. Der Beweis erfolgt vollkommen analog zum Beweis von Satz 4.5.22. □

Analog zu den Sätzen 4.5.11 und 4.5.26 gilt für den Fall verallgemeinerter Blöcke der

Satz 4.6.6. *Gegeben sei eine symmetrische positiv definite Matrix $X \in \mathbb{R}^{n\times n}$. Die Indexmengen $I_p \in \mathbb{R}^{n_p\times n_p},\ p = 1, ..., M$ induzieren verallgemeinerte Blöcke X_p der Matrix X. Dann existieren symmetrische positiv definite Matrizen $Y_{(p)} \in \mathbb{R}^{n_p\times n_p}$ mit den gleichen Indexmengen I_p, so dass*

$$X := \sum_{p=1}^{M} \overline{Y_p}.$$

Beweis. Ohne Beschränkung der Allgemeinheit lassen sich die Variablen so permutieren, dass der verallgemeinerte Block X_M einen gewöhnlichen Block in der Matrix X darstellt. Nach einer solchen Permutation entsteht zwischen den beiden Matrizen

$$X_{ij}, \quad i, j \in \bigcup_{p\in\{1,...,M-1\}} I_p$$

und X_M eine dicht besetzte einfache Überschneidung, wie in Abschnitt 4.5.2 behandelt. Nach Satz 4.5.11 können wir bezüglich dieser Überschneidung zwei symmetrische positiv definite Matrizen $\tilde{Y}_{(1)}$ und $Y_{(M)}$ finden, so dass

$$\tilde{Y}_{(1)} + Y_{(M)} = X$$

gilt. Die Indexmengen $I_p,\ p = 1, ..., M-1$ induzieren jetzt verallgemeinerte Blöcke $\tilde{Y}_{1,p}$ der Matrix $\tilde{Y}_1$. In gleicher Weise lassen sich erneut die verbliebenen Variablen $i \in \{1, ..., n\} \setminus I_M$ so permutieren, dass zwischen den Matrizen

$$X_{ij}, \quad i, j \in \bigcup_{p\in\{1,...,M-2\}} I_p$$

und $Y_{1,M-2}$ eine dicht besetzte einfache Überschneidung entsteht, so dass erneut Satz 4.5.11 angewandt werden kann und so weiter. So entstehen M symmetrische positiv definite Matrizen $Y_{(p)},\ p = 1, ..., M$, welche nach entsprechender Rückpermutierung die Behauptung liefern. □

Somit sind wir in der Lage, entsprechend der Sätze 4.5.13 und 4.5.28, auch hier festzustellen:

Satz 4.6.7. *Es existiert ein $\varepsilon > 0$ so dass für $x^+, x \in U_\varepsilon(x^*)$ stets ein Vektor $\eta \in \mathbb{R}^{M-1}$ existiert, so dass für die Vektoren $u_{(p)},\ p = 1, ..., M$ aus (4.108) gilt*

$$u_{(p)}^T d_p > 0.$$

Beweis. Siehe Beweis von Satz 4.5.13 □

Für den Fall, dass die Startschätzung $x^{(0)}$ außerhalb von $U_\varepsilon(x^*)$ liegt, müssen die Vektoren $u_{(p)}$ unter Umständen wieder angepasst werden. Falls

$$y^T d \leq 0,$$

muss y in (4.108) durch q aus (4.11) ersetzt werden. Wie in den vorangegangen Abschnitten auch, garantiert dies allein jedoch noch nicht, dass

$$u_{(p)}^T d_p > 0 \tag{4.109}$$

mit geeignetem η hergestellt werden kann. Analog zu Satz 4.5.2 aus Abschnitt 4.5.2 und Satz 4.5.24 aus Abschnitt 4.5.3 formulieren wir den

Satz 4.6.8. *Seien $q, d \in \mathbb{R}^n$ mit $q^T d > 0$. Seien $I_p,\ p = 1, ..., M$, die Indexmengen der verallgemeinerten Blöcke zur Matrix H. Existieren Teilmengen von Blöcken*

$$P_1, ..., P_s \subset \{1, ..., M\}$$

mit

$$\bigcup_{i=1}^{s} P_i = \{1, ..., M\}$$

und

$$P_i \cap P_j = \emptyset, \quad i \neq j$$

so dass

$$\forall i, j \in \{1, ..., s\} \text{ mit } i \neq j: \quad q_{p\cap r}^T d_{p\cap r} = 0,\ p \in P_i, r \in P_j \tag{4.110}$$

und außerdem

$$q_{\overline{\mathcal{J}}_i}^T d_{\overline{\mathcal{J}}_i} > 0 \tag{4.111}$$

gilt, wobei

$$\overline{\mathcal{J}}_i := \bigcup_{p \in P_i} I_p, \quad i = 1, ..., s,$$

so existiert ein Vektor $\eta \in \mathbb{R}^{M-1}$, so dass (4.109) gilt.

Beweis. Analog zum Beweis von Satz 4.5.24. Die Indexmengen $\overline{\mathcal{J}_i}$ sind das Analogon zu den dort verwendeten Indexmengen $\tilde{\mathcal{J}}_j$. □

Analog zu Satz 4.5.3 und Satz 4.5.25 formulieren wir den

Satz 4.6.9. *Sei $q^T d > 0$ und $H_{(p)},\ p = 1, ..., M$ positiv definit. Die Bedingung (4.111) aus Satz 4.6.8 sei nicht erfüllt. Somit existiert eine Teilmenge P_i von Blöcken, so dass (4.110) gilt und (4.111) nicht gilt. Man ersetze $q_{\overline{p}},\ \overline{p} \in P_i$ durch*

$$\tilde{q}_{\overline{p}} := \theta q_{\overline{p}} + (1 - \theta) H_{(p)} d_{\overline{p}} \tag{4.112}$$

mit

$$\theta = \frac{\sigma \cdot \sum_{\overline{p} \in P_i} d_{\overline{p}}^T H_{(\overline{p})} d_{\overline{p}}}{\sum_{\overline{p} \in P_i} \left(d_{\overline{p}}^T H_{(\overline{p})} d_{\overline{p}} - q_{\overline{p}}^T d_{\overline{p}} \right)},$$

und $\sigma \in (0, 1)$. Dann gilt

$$\tilde{q}_{\tilde{\mathcal{J}}_j}^T d_{\tilde{\mathcal{J}}_j} > 0.$$

Beweis. Siehe Beweis von Satz 4.5.3. □

Erstmalig ist es uns gelungen, ein dünn besetztes BFGS-artiges Update zu erzeugen, welches nicht nur stets positiv definit ist und in einer Umgebung um x^* die Quasi-Newton-Bedingung erfüllt, sondern auch die vorgegebene Dünnbesetztheit exakt einhält.
Wir gehen wieder davon aus, dass $x^{(0)}$ eine hinreichend gute Startschätzung sei und folglich, nach Satz 4.6.7 die Verwendung von q beziehungsweise q_p und $\tilde{q}_p$ aus (4.112) unnötig ist. Somit formulieren wir den folgenden

Algorithmus 4.6.10.

i. Identifiziere die Indexmengen I_p, $p = 1, ..., M$, die verallgemeinerte Blöcke in der Struktur der Matrix $\nabla^2 f(\cdot)$ induzieren.

ii. Wähle $x^{(0)} \in \mathbb{R}^n$ und $H^{(0)}_{(p)} \in \mathbb{R}^{n_p \times n_p}$, $p = 1, ..., M$ symmetrisch und positiv definit und setze

$$H^{(0)} := \sum_{p=1}^{M} H^{(0)}_{(p)}.$$

Wähle außerdem $\varepsilon \geq 0$ und setze $k := 0$.

iii. Ist $\|\nabla f(x^{(k)})\| \leq \varepsilon$. STOP.

iv. Bestimme $d^{(k)}$ aus dem Gleichungssystem

$$H^{(k)} d^{(k)} = -\nabla f(x^{(k)}).$$

v. Setze

$$\begin{aligned} x^{(k+1)} &:= x^{(k)} + d^{(k)} \text{ und} \\ y^{(k)} &:= \nabla f(x^{(k+1)}) - \nabla f(x^{(k)}). \end{aligned}$$

vi. Bestimme $\eta^{(k)} \in \mathbb{R}^{M-1}$, so dass für alle Blöcke

$$u^{(k)T}_{(p)} d^{(k)}_p > 0$$

gilt, mit $u^{(k)}_{(p)}$ aus (4.108).

vii. Berechne

$$H^{(k+1)}_{(p)} = H^{(k)}_{(p)} + \frac{y^{(k)}_p y^{(k)T}_p}{y^{(k)T}_p d^{(k)}_p} - \frac{(H^{(k)}_{(p)} d^{(k)}_p)(H^{(k)}_{(p)} d^{(k)}_p)^T}{d^{(k)T}_p H^{(k)}_{(p)} d^{(k)}_p}, \quad p = 1, ..., M$$

und berechne die gesamte Matrix durch

$$H^{(k+1)} = \sum_{p=1}^{M} \overline{H^{(k+1)}_{(p)}}.$$

viii. Setze k:=k+1 und gehe zu iii.

Bemerkung 4.6.11. Schritt *vi.* kann wieder in analoger Weise wie im Beweis von Satz 4.5.24 erfolgen.

Aufgrund der bereits in Abschnitt 4.5.3.1 angesprochenen doppelten Speicherung der Werte in den Überschneidungen, ist es hinsichtlich des Speichers erstrebenswert, möglichst wenig Überschneidungseinträge zu generieren. Dies ist gleichbedeutend damit, die verallgemeinerten Blöcke so groß wie möglich, das heißt so groß, wie es die tatsächliche Struktur zulässt, zu wählen. Man beachte, dass die Bestimmung der größtmöglichen verallgemeinerten Blöcke ein sehr aufwändig zu lösendes kombinatorisches Optimierungsproblem (vermutlich NP-vollständig) darstellt, weswegen es sich bewährt, eine maximale Blockgröße zu definieren. Dies ermöglicht es, die Zeit, die ein Algorithmus bei der Suche nach möglichst großen verallgemeinerten Blöcken benötigt, geeignet zu steuern, so dass das Ergebnis innerhalb einer akzeptablen Zeit erzielt werden kann.
Obwohl das verallgemeinerte Sparse BFGS in der Lage ist, die Nichtnullstruktur exakt abzubilden, kann es unter Umständen sogar sinnvoll sein, zu Gunsten des Speichers einige Nulleinträge in die Struktur aufzunehmen.

4.6.2 Konvergenzeigenschaften des Verallgemeinerten Sparse BFGS

Die Konvergenzaussage deckt sich erneut mit denen aus den Abschnitten 4.5.2.2 und 4.5.3.2.

Satz 4.6.12. *Es existieren ein $\varepsilon > 0$ und ein $\delta > 0$, sowie Funktionen $\tilde{f}_p,\ p = 1, ..., M$, so dass der Algorithmus 4.5.14 für jeden Startvektor $x^{(0)} \in \mathbb{R}^n$ mit*

$$\|x^{(0)} - x^*\| < \varepsilon$$

und jede symmetrische positiv definite Startmatrizen $H_{(p)}^{(0)} \in \mathbb{R}^{n\times n},\ p = 1, ..., M$ mit

$$\|H_{(p)}^{(0)} - \nabla^2 \tilde{f}_p(x_p^*)\| < \delta \tag{4.113}$$

eine Folge $(x^{(k)})_{k\in\mathbb{N}} \subset \mathbb{R}^n$ erzeugt, die superlinear gegen x^* konvergiert.

Beweis. Aus dem gleichen Grunde wie beim Satz 4.5.32, kann der gleiche Beweis wie bei Satz 4.5.20 an dieser Stelle geführt werden. □

5 Numerische Ergebnisse

"Die Praxis sollte ein Ergebnis des Nachdenkens sein, nicht umgekehrt."
Herrmann Hesse, 1877-1962

5.1 CUTEr

CUTEr [45] ist eine vielseitige Testumgebung für Software zur Lösung von Problemen in der Optimierung und der Linearen Algebra. Das Paket enthält eine Sammlung von Testproblemen und weitere Routinen in Fortran77, Fortran 90/95 und Matlab, die Entwickler dabei unterstützen, neue Löser zu entwickeln, bereits existierende zu verbessern oder verschiedene miteinander zu vergleichen.
Die Testprobleme sind in einem so genannten Standard Input-Format (SIF) geschrieben. CUTEr umfasst einen Decoder, der die Dateien von diesem Format beispielsweise in Fortran 77 konvertiert. Einmal übersetzt, können diese Dateien manipuliert und somit weitere Testprobleme erzeugt werden. Des Weiteren ist CUTEr auch über die Modellierungssprache AMPL zugänglich.
CUTEr ist auf einer Vielzahl von UNIX-Plattformen, einschließlich Linux, verfügbar und ist darauf ausgerichtet, leicht zugänglich und handhabbar zu sein.
In CUTEr finden sich 920 Nichtlineare Optimierungsprobleme. In der gegebenen Form haben diese Probleme bis zu 50000 Optimierungsvariable und Nebenbedingungen. Die Probleme können aber leicht angepasst werden, so dass beliebig viel mehr Variable zu optimieren sind. Die Untersuchungen dieser Arbeit beschränken sich jedoch auf die unmodifizierten CUTEr-Testprobleme.

5.2 Auswertung

Das Ziel der in diesem Abschnitt durchgeführten numerischen Auswertung ist es, Aufschluss darüber zu erhalten, wie sehr sich die unterschiedlichen, in dieser Arbeit entwickelten Verfahren zur Approximation der Ableitungen in der Praxis bewähren. Damit sich die Effekte der unterschiedlichen Verfahren nicht vermischen, betrachten wir zunächst nur die Verfahren zur effizienten Approximation der ersten Ableitungen und anschließend diejenigen der zweiten Ableitung.
Für gewöhnlich sind die hier entwickelten Verfahren trotz aller Effizienzsteigerungen rechenintensiver als Routinen, in denen die Ableitung in analytischer Form zur Verfügung steht und unmittelbar ausgewertet werden kann. Für all unsere Testbeispiele stehen solche Routinen zur Verfügung. Wir bezeichnen Sie fortan als *analytische Routinen*. Als Vergleichswert (Benchmark) für unsere Verfahren, was Rechenzeit und Rechengenauigkeit (gemessen an der Anzahl gelöster Probleme) angeht, verwenden wir auch diese Berechnungen, obwohl in der Regel nicht zu erwarten ist, dass wir an diese Werte herankommen.
Wir unterscheiden zwischen zwei erfolgreichen Optimierungsprozessen. Einerseits das Auffinden einer optimalen und andererseits das Auffinden einer akzeptablen Lösung. Eine optimale Lösung ist gefunden, wenn die KKT-Bedingungen (2.6)-(2.8), geeignet normiert und skaliert, kleiner sind als eine Toleranzgrenze (hier: 10^{-6}).
Dauert die Lösung eines Optimierungsproblems länger als die angegebene Höchstdauer (hier: 30 Minuten) oder bedarf es mehr als eine maximale Anzahl an Hauptiterationen (hier: 10000) oder tritt irgendein erdenklicher Fehler auf, so erfolgt ein Abbruch. Wurde in irgendeine Iteration bis zu diesem Abbruch eine Lösung gefunden, die den KKT-Bedingungen mit einer höheren Toleranz (hier: 10^{-3}) genügt, so bezeichnen wir diese Lösung als akzeptabel. Der Verwendete Rechner ist ein Quad-Core AMD Opteron 8378 mit 4 Prozessoren mit jeweils 2,4 GHz, 32 GB RAM.
Mit analytischen Routinen erhalten wir das folgende Ergebnis

Testbeispiele	Erfolgreich			Erfolglos			Rechenzeit
	Optimal	Akzeptabel	Total	Timeout	Andere	Total	
920	898	9	907	3	10	13	21281s

Tabelle 5.1: Ergebnis mit analytischen Routinen.

5.2.1 Erste Ableitungen

Zunächst betrachten wir die Verfahren zur Bestimmung der ersten Ableitungen. Das bedeutet, wir vergleichen die unterschiedlichen in Abschnitt 3.3 entwickelten Gruppenmethoden miteinander und außerdem mit der Berechnung erster Ableitungen ohne Gruppenmethoden. Wie erwähnt, spielt die Gruppenmethode zur Berechnung der ersten Ableitung nur bei der Jacobimatrix der Nebenbedingungen eine Rolle. Der Gradient der Zielfunktion kann in der Regel auf herkömmliche Weise effizient bestimmt werden. Zu beachten ist dabei lediglich, dass Auswertungen der Form $f(x + \varepsilon e_i)$ dann und nur dann vorgenommen werden, falls f von x_i abhängt.

Zur isolierten Betrachtung der Gruppenmethoden wird hier der Gradient der Zielfunktion und die Hessematrix der Lagrangefunktion über eine analytische Routinen "exakt" (das heißt bis auf numerische Ungenauigkeiten) ausgerechnet und ausschließlich die Jacobimatrix der Nebenbedingungen mittels Finiter Differenzen berechnet. Infolgedessen ist es sinnvoll, hier lediglich die restringierten Testprobleme zu betrachten.
Wir unterscheiden bei der Rechenzeit diejenige, die für den Optimierungsprozess an sich benötigt wird, inklusive aller notwendigen Funktionsauswertungen und diejenige, welche für das Erstellen der Gruppen benötigt wird.
In Tabelle 5.2 finden sich die Ergebnisse der unterschiedlichen Gruppen-Methoden. $GM0$ bezeichnet hierbei die klassische Weise der Finiten Differenzen, ohne die Bildung von Gruppen. $GM1$ bezeichnet den CPR-Algorithmus, $GM2$ den LFO-Algorithmus und $GM3$ den SLO-Algorithmus. $GM4, GM5$ und $GM6$ bezeichnen die gleichen Methoden wie $GM1, GM2$ und $GM3$, jedoch inklusive Austauschmethode.

Größe \ Methode		GM0	GM1	GM2	GM3	GM4	GM5	GM6
Probleme		590	590	590	590	590	590	590
+	Optimal	556	565	566	554	540	542	524
	Akz.	9	9	10	9	10	9	9
	Total	565	574	576	563	550	551	533
-	Gr-TO	0	0	0	12	30	29	50
	Löser-TO	16	3	3	2	3	2	1
	Andere	9	13	11	13	7	8	6
	Total	25	16	14	27	40	39	57
Gruppen		843557	75499	75567	25568	15465	15457	13369
Gesp. Ausw.		0	768058	767990	767453	614638	630712	383445
Gesp. Ausw. rel.		0	0,91	0,91	0,97	0,98	0,98	0,97
Zeit	Gruppen	0	40	547	1104	6115	13214	17178
	Löser	58649	15681	16830	15728	10241	11539	8327
	Total	58649	15721	17376	16832	16356	24753	25505
Gesp. Löser-Zeit		0	9473	8325	9426	14913	13615	16828
Gesp. Löser-Zeit rel.		0	0,38	0,33	0,37	0,59	0,54	0,67

Tabelle 5.2: Restringierte Testprobleme mit Jacobimatrix mittels Finiter Differenzen.

Wie wir sehen, sind 590 der 920 Probleme restringiert. "+" bezeichnet die erfolgreich gelösten Optimierungsprobleme, unterteilt in "Optimal", Akzeptabel ("Akz.") und Insgesamt ("Total"). "-" bezeichnet entsprechend die nicht erfolgreich gelösten Probleme. Hierbei unterscheiden wir unterschiedliche Ursachen: "Gr-TO" bedeutet, dass die Bestimmung der Gruppen die Höchstzeitdauer überschritten hat, Löser-TO besagt, dass der Lösungsprozess die Höchstzeitdauer überschritten hat. Diese beiden Zeiten werden getrennt behandelt, so dass theoretisch der Fall auftreten kann, dass die gesamte Prozesszeit knapp unter der doppelten Höchstzeitdauer liegt. Alle anderen Abbrüche werden unter "Andere" zusammengefasst und sämtliche erfolglose Prozesse unter "Total" summiert.

Die Zeile “Gruppen” bezeichnet die kumulierte Anzahl der Gruppen über alle betrachteten Optimierungsprobleme. Unter $GM0$ ist diese Zahl gleichbedeutend mit der kumulierten Anzahl der Variablen. Die Zeilen “Gesp. Ausw.” beziehungsweise “Gesp. Ausw. rel.“ bezeichnen die aufgrund der jeweils verwandten Gruppen-Methode absolut und relativ gesparten Auswertungen pro Jacobimatrix-Berechnung. Hierbei ist jedoch zu beachten, dass nur die Beispiele Berücksichtigung gefunden haben, für die es keine Höchstzeitüberschreitung bei der Gruppenfindung mit der jeweiligen Methode gegeben hat.
Die gemessene Zeit ist in Sekunden angegeben. Hierbei unterscheiden wir erneut zwischen der zur Gruppenbestimmung und der zur anschließenden Lösung des Optimierungsproblems benötigten Zeit.
Die letzten beiden Zeilen geben an, wieviel Zeit zur Lösung des Problems absolut und relativ aufgrund der Gruppen gespart werden konnte.
Zunächst stellen wir fest, dass der Aufwand der Gruppenbestimmung von links nach rechts zunimmt. Ein echter Vergleich gestaltet sich mit dieser Tabelle jedoch schwierig, da mit einigen Methoden die Höchstzeit bei der Gruppenfindung häufiger überschritten wurde als bei anderen. Dies erklärt die deutlich niedrigere Zahl an Gruppen in der letzten Spalte bei ungefähr gleicher relativer Einsparung an Funktionsauswertungen.
Zur besseren Vergleichbarkeit werden in der Tabelle 5.3 nur diejenigen Testprobleme aufgeführt, bei denen keine der Methoden eine Höchstzeitüberschreitung aufweist. Naturgemäß treten solche Überschreitungen gerade bei den extrem hochdimensionalen und verhältnismäßig wenig dünnbesetzten Problemen auf. Dies erklärt die deutlich geringeren Rechenzeiten bei diesem Vergleich und die höheren anteiligen Zeit- und Auswertungseinsparungen.

Größe \ Methode		GM0	GM1	GM2	GM3	GM4	GM5	GM6
Probleme		538	538	538	538	538	538	538
+	Optimal	515	521	522	522	521	522	522
	Akz.	9	9	10	9	10	9	9
	Total	524	530	532	531	531	531	531
-	Gr-TO	0	0	0	0	0	0	0
	Löser-TO	7	2	1	1	2	1	1
	Andere	7	6	5	6	5	6	6
	Total	14	8	6	7	7	7	7
Gruppen		376814	19117	19158	19208	13353	13349	13362
Gesp. Ausw.		0	357697	357656	357606	363461	363465	363452
Gesp. Ausw. rel.		0,00	0,95	0,95	0,95	0,96	0,96	0,96
Zeit	Gruppen	0	14	155	811	2737	8383	17150
	Löser	25154	6861	6955	7776	6985	7078	7004
	Total	25154	6875	7109	8587	9722	15461	24154
Gesp. Löser-Zeit		0	18293	18200	17379	18169	18076	18151
Gesp. Löser-Zeit rel.		0,00	0,73	0,72	0,69	0,72	0,72	0,72

Tabelle 5.3: Restringierte Vergleichsprobleme mit Jacobimatrix mittels Finiter Differenzen.

Wir stellen fest, dass der LFO-Algorithmus ohne Austauschmethode in der insgesamten Gruppenanzahl keine Verbesserung liefert im Vergleich zum CPR-Algorithmus ohne Austausch. Tatsächlich liegt die Anzahl sogar geringfügig darüber. Das beste Ergebnis bei dieser Vergleichsrechnung erhalten wir jedoch beim LFO-Algorithmus mit Austauschmethode. Des Weiteren ist auffällig, wie deutlich die Anzahl der Gruppen durch die Verwendung der Austauschmethode abnimmt. Andererseits ist der beobachtbare Zeitgewinn, was die Zeit des Lösers angeht, verhältnismäßig klein. Betrachtet man die Zeit für die Gruppenfindung, so sieht man, dass der CPR-Algorithmus deutlich schneller ist als die anderen, aufwändigeren Verfahren und dass die Austauschmethode einen erheblichen Rechenmehraufwand mit sich bringt.
Trotz der recht hohen Zahl an Testbeispielen, wäre es falsch aus dieser Testrechnung einen eindeutigen Schluss zu ziehen. Insbesondere bei Problemen, deren Funktionsauswertungen extrem rechenaufwändig sind, kann es sich unter Umständen rentieren, ein sehr aufwändiges Verfahren zur Gruppenbestimmung durchzuführen, auch wenn man damit nur eine unwesentlich kleinere Anzahl von Gruppen erhält. Es ist wichtig, nochmals zu betonen, dass die Gruppenfindung nur einmalig vor Beginn der Optimierung durchgeführt werden muss, während die Jacobimatrix in jeder Iteration zu berechnen ist.
Die Verwendung aufwändigerer Gruppen-Algorithmen empfiehlt sich insbesondere dann, wenn bei einer Anwendung viele unterschiedliche Optimierungsprobleme zu lösen sind, welche jedoch alle die gleiche Nichtnullstruktur besitzen. In diesem Falle empfiehlt es sich, die aufwändig bestimmten, nahezu optimalen Gruppen zu speichern und für alle Probleme zu nutzen.
Dennoch halten wir fest, dass der CPR-Algorithmus für gewöhnlich vorzuziehen ist, da er in deutlich kürzerer Zeit nahezu gleichwertige Ergebnisse liefert.

5.2.2 Zweite Ableitungen

Zunächst untersuchen wir die für die Finiten Differenzen entwickelten Methoden zur effizienten Berechnung der zweiten Ableitung. Wie im Abschnitt 3.5 dargelegt, empfiehlt sich die Verwendung der Paargruppenmethode lediglich dann, wenn auch die Jacobimatrix mittels finiter Differenzen berechnet wird. Daher vergleichen wir hier die Paargruppen- und die Hessegruppenmethode bei gleichzeitiger Berechnung der Jacobimatrix mittels finiter Differenzen. Um den zeitlichen Aufwand zur Gruppenbestimmung gering zu halten und aufgrund der Ergebnisse aus Abschnitt 5.2.1, verwenden wir hierfür stets den CPR-Algorithmus. Als Vergleichswert ("Benchmark“) betrachten wir die Berechnung der Jacobi- und Hessematrix mittels finiter Differenzen ohne die Verwendung von Gruppen, Paargruppen und Hessegruppen. Den Gradienten der Zielfunktion berechnen wir wieder analytisch und den Beitrag der Zielfunktion für die Hessematrix ermitteln wir bei allen Methoden auf gleiche Weise, das heißt mittels finiter Differenzen unter Berücksichtigung der Dünnbesetztheit, siehe (3.5) und die Erläuterungen in Abschnitt 3.1.
Der Aufbau der Tabelle 5.4 ähnelt dem der Tabellen 5.2 und 5.3. “PG/HG-TO” bedeutet, dass die Bestimmung der Paar- beziehungsweise Hessegruppen die Höchstzeitdauer überschritten hat. Da der Vergleich zwischen der Anzahl der Paargruppen und der der Hessegruppen nicht aussagekräftig ist, vergleichen wir die für die Berechnungen notwendigen Auswertungen der Nebenbedingungen. Dies umfasst sowohl die Berechnung der Jacobima-

trix als auch die der Hessematrix, wodurch sich die Ersparnisse durch Gruppenmethode und Paargruppen- beziehungsweise Hessegruppenmethode hier kumulieren. Auffällig ist wieder

Größe \ Methode		Benchmark	PG	HG
Probleme		590	590	590
+	Optimal	531	545	544
	Akz.	17	14	16
	Total	548	559	560
-	PG/HG-TO	0	0	0
	Löser-TO	22	11	12
	Andere	20	20	18
	Total	42	31	30
Auswertungen		1758921	148310	192970
Gesp. Ausw.		0	1610611	1565951
Gesp. Ausw. rel.		0	0,92	0,89
Zeit	PG/HG	40	610	145
	Löser	108087	73277	51180
	Total	108127	73886	51325
Gesp. Löser-Zeit		0	34810	56907
Gesp. Löser-Zeit rel.		0	0,32	0,53

Tabelle 5.4: Restringierte Vergleichsprobleme mit Jacobimatrix mittels Finiter Differenzen.

die enorme Einsparung an Auswertungen durch beide Methoden. Insgesamt sparen wir bei Verwendung der Paargruppen noch etwas mehr Auswertungen als bei den Hessegruppen. Interessanterweise sind jedoch diejenigen Beispiele, bei denen die Hessegruppe mehr einspart als die Paargruppenmethode tendenziell gerade diejenigen, deren Funktionsauswertungen besonders teuer sind. Dies erklärt, warum die insgesamte Zeitersparnis bei den Hessegruppen um einiges höher ist als bei den Paargruppen. Die unterschiedliche Zeitersparnis mag an der Gestalt unserer Testbeispiele liegen, so dass daraus nicht unbedingt ein klarer Vorteil der Hessegruppen abzuleiten ist. Falls man an einer möglichst geringen Anzahl an Auswertungen interessiert ist, ist es insbesondere in Hinblick auf die recht geringe Rechenzeit, die zur Paar- beziehungsweise Hessegruppeneinteilung notwendig ist, ratsam, beide Verfahren anzuwenden und dasjenige mit der geringeren Auswertungsanzahl zu verwenden. Dies gilt wiederum um so eher, je aufwändiger die Funktionsauswertungen sind und je öfter Probleme mit der gleichen Dünnbesetztheitsstruktur zu berechnen sind.
Die Genauigkeit der beiden Methoden ist vergleichbar, genauso wie die Wahrscheinlichkeit einer numerischen Auslöschung. Dies unterstützt auch die sehr ähnliche Zahl an (optimal und akzeptabel) gelösten Problemen.
Hinsichtlich der Zeitersparnis sei darauf hingewiesen, dass zwar die relative Ersparnis unter der in den Tabellen 5.2 und 5.3 liegt, die absolute Ersparnis jedoch deutlich höher ist. Dieser Sachverhalt erklärt sich durch die unterschiedlichen Vergleichswerte. Die Berechnung der Hessematrix mittels finiter Differenzen ist extrem aufwändig. Die Ersparnisse sind zwar,

wie wir sehen, sehr hoch, jedoch verbraucht die Hessematrixberechnung mittels finiter Differenzen auch mit unseren sehr effizienten Verfahren so viel Zeit, dass die relativ eingesparte Zeit durch weniger Funktionsauswertungen geringer ausfällt. Aus der absoluten Zeitersparnis lässt sich jedoch erkennen, dass die Bedeutung der effizienten Gestaltung der Berechnung der zweiten Ableitung viel wichtiger ist als dies bei der Berechnung der Jacobimatrix der Fall ist.
Im Falle einer analytisch gegebenen Jacobimatrix, ist es nicht sinnvoll, die Paargruppenmethode zu verwenden, da bei dieser Auswertung der Form $g(x + \varepsilon e_i)$ notwendig sind, die extra für die Berechnung der Hessematrix auszuwerten wären. Da die Hessegruppenmethode unmittelbar mit den Werten der Jacobimatrix rechnen kann, leuchtet ein, dass die Hessegruppenmethode in diesem Fall klar überlegen ist.
In der Tabellen 5.5 finden sich die Ergebnisse der Hessegruppenmethode bei analytisch gegebener Jacobimatrix. Außerdem dient diese Tabelle und Tabelle 5.6 zum Vergleich zwischen effizienter Hessematrixberechnung mittels Finiter Differenzen und dem BFGS-Verfahren beziehungsweise den BFGS-ähnlichen Verfahren zur Approximation der Hessematrix. Als Vergleichswert dient die Berechnung der Hessematrix mittels finiter Differenzen ohne jegliche Gruppenverfahren bei analytischen Gradienten der Zielfunktion und Jacobimatrix der Nebenbedingungen.
Die untersuchten Update-Verfahren sind das klassische, dicht besetzte BFGS ("BFGS"), das Block-BFGS-Verfahren ohne Überschneidungen mit fester Blockgröße 10 ("fixed BS"), siehe Abschnitt 4.5.1, das BFGS-Verfahren mit zugelassenen einfachen Überschneidungen und variabler Blockgröße ("var. BS"), siehe Abschnitt 4.5.2, sowie das Sparse BFGS aus Abschnitt 4.6.1 mit maximaler Blockgröße 2 ("SBFGS 2"), 10 ("SBFGS 10") und beliebig ("SBFGS ∞"). Die in Abschnitt 4.5.3.1 eingeführten Update-Verfahren dienten eher der Vorbereitung auf das SBFGS und werden daher nicht genauer untersucht.
Die Zeile "Auswertungen" beschreibt hier die Anzahl der notwendigen Auswertungen der Jacobimatrix der Nebenbedingungen. Wir beobachten eine noch höhere relative Einsparung als in den vorhergehenden Untersuchungen. Da bei den Update-Verfahren keine Funktionsauswertungen vorzunehmen sind, bleiben die entsprechenden Zellen in der Tabelle leer.
Die Zeile "nnz" gibt an, wieviele Nichtnulleinträge sich in der unteren Hälfte der Hessematrix, inklusive Diagonale, befinden. Beim Vergleichswert sind dies genau die summierten Nichtnulleinträge aller betrachteten Testprobleme. Beim klassischen BFGS-Verfahren entsteht, wie erwähnt, eine dicht besetzte Hessematrix. Die Zahl der Nichtnulleinträge, die für sämtliche Probleme benötigt werden, finden sich in dieser Zeile. In 71 Fällen ("Out of Mem."), überschreitet der benötigte Speicherbedarf die Kapazitäten des verwendeten Computers. Wie bereits erwähnt, führt eine hohe Anzahl an Nichtnulleinträgen nicht nur zu einem erhöhten Speicher- sondern auch Rechenaufwand.
Neben den zusätzlichen Einträgen in der Hessematrix, müssen bei den unterschiedlichen Update-Verfahren oft zusätzliche Werte gespeichert werden. Wie so häufig kann ein hoher Speicheraufwand durch zusätzlichen Rechenaufwand zu einem gewissen Maße reduziert werden. Hierbei gilt es, einen guten Mittelweg zu finden. Der in unseren Implementierungen zusätzlich benötigte Speicher, inklusive der gegebenenfalls zusätzlichen Nichtnulleinträge in der Hessematrix, findet sich in Zeile "zus. Speicher". Falls sich die BFGS-Block-Struktur an der tatsächlichen Hessestruktur orientiert, ist dafür eine messbare Zeit von Nöten. Dieser Prozess wird einmal zu Beginn der Optimierung durchgeführt und wird daher sinnvollerweise mit der Zeit verglichen, die für die Hessegruppenbestimmung benötigt wird. Beim klassischen

BFGS und dem Block-BFGS mit fester Blockgröße ist die Struktur so schnell bestimmt, dass keine Zeit messbar ist.

Größe \ Methode		Benchmark	HG	BFGS	fixed BS	var. BS
Probleme		590	590	590	590	590
+	Optimal	552	557	490	566	536
	Akz.	14	11	21	22	13
	Total	566	568	511	588	549
-	Out of Mem.	0	0	71	0	26
	Löser-TO	15	12	1	2	7
	Andere	9	10	7	0	8
	Total	24	22	79	2	41
Auswertungen		843557	3443			
Gesp. Ausw.		0	840114			
Gesp. Ausw. rel.		0	0,997			
nnz		1758921	1758921	4617842066	4634696	73781912
zus. Speicher		0	0	4616083145	2875775	78543503
Zeit	HG/Block	0	80	0	0	5
	Löser	79207	51643	45939	29281	46214
	Total	79207	51723	45939	29281	46220
Gesp. Löser-Zeit		0	27644	33268	49926	32993
Gesp. Löser-Zeit rel.		0,00	0,35	0,42	0,63	0,42

Tabelle 5.5: Restringierte Vergleichsprobleme mit analytischer Jacobimatrix, Teil 1.

Als erstes fällt auf, dass das klassische BFGS-Verfahren sehr häufig bereits aufgrund des Speichers ungeeignet ist. Die Zeitangaben des klassischen BFGS sind durch die hohe Zahl an Abbrüchen aufgrund des fehlenden Speichers verfälscht, so dass diese nicht weiter beachtenswert sind. Da beim Block-BFGS mit variabler Blockgröße die Blöcke so groß gemacht werden, dass jeder Nichtnulleintrag in der Blockstruktur enthalten ist und hierbei keinerlei Permutationen durchgeführt wurden, entstehen mitunter auch hier sehr dichte Matrizen.
Grundsätzlich ist erkennbar, dass die Update-Verfahren im Vergleich zum Verfahren der Finiten Differenzen, auch mit Hessegruppen, erkennbar schneller sind. Diese Tendenz ist mit steigenden Problemdimensionen umso deutlicher zu erkennen. Die Anzahl der gelösten Probleme ist sich bei allen Verfahren recht ähnlich, berücksichtigt man die Speicherproblematik beim klassischen BFGS und beim Block-BFGS mit variabler Blockgröße.
Das beste Gesamtergebnis findet sich beim SBFGS. Anders als zu erwarten, ist der insgesamte Rechenaufwand bei unseren Testbeispielen nicht um so größer, je größer die maximale Blockgröße gewählt ist. Dies ist jedoch bei wachsender Problemgröße zu erwarten. Wie wir sehen, ist der zusätzliche Speicher umso größer, je kleiner die maximale Blockgröße gewählt ist, so dass es sich unter Speicher-Gesichtspunkten empfiehlt, die maximale Blockgröße möglichst groß zu wählen.
Bemerkenswert sind auch die guten Resultate beim Block-BFGS-Verfahren mit fester Blockgröße. Obwohl wir hierfür keine Konvergenzaussage liefern können, reichen die Ergebnisse

Größe \ Methode		Benchmark	SBFGS 2	SBFGS 10	SBFGS ∞
Probleme		590	590	590	590
+	Optimal	552	563	565	565
	Akz.	14	10	10	10
	Total	566	573	575	575
-	Out of Mem.	0	0	0	0
	Löser-TO	15	6	2	1
	Andere	9	11	13	14
	Total	24	17	15	15
nnz		1758921	1758921	1758921	1758921
zus. Speicher		0	5213156	1480881	1164573
Zeit	HG/Block	0	235	132	117
	Löser	79207	27033	24433	24155
	Total	79207	27268	24565	24273
Gesp. Löser-Zeit		0	52174	54774	55052
Gesp. Löser-Zeit rel.		0	0,66	0,69	0,70

Tabelle 5.6: Restringierte Vergleichsprobleme mit analytischer Jacobimatrix, Teil 2.

fast an die der SBFGS-Verfahren heran.
Bei Verwendung der Update-Verfahren, gibt es keinen Grund, lediglich die restringierten Probleme zu betrachten. Die Ergebnisse aller Testprobleme finden sich in den Tabellen 5.7 und 5.8.
Bei diesen Beispielen werden durch das Block-BFGS-Verfahren mit fester Blockgröße sogar am meisten Probleme gelöst. Außerdem bedarf die Struktur-Bestimmung bei beliebigen zugelassen Blockgrößen nun mehr Zeit als bei einer maximal zugelassenen Blockgröße von 10. Das SBFGS-Verfahren mit beliebiger Blockgröße weist die insgesamt geringste Rechenzeit auf.
Die Praxistauglichkeit der in dieser Arbeit entwickelten Verfahren ist somit nachgewiesen.

Größe \ Methode		Benchmark	BFGS	fixed BS	var. BS
Probleme		920	920	920	920
+	Optimal	853	764	878	833
	Akz.	28	29	33	21
	Total	881	793	911	854
-	Out of Mem	0	111	0	34
	Löser-TO	25	3	4	11
	Andere	14	13	5	55
	Total	39	127	9	66
nnz		6883501	7498842316	7111466	170404038
zus. Speicher		0	7491958815	227965	185961862
Zeit	Block	0	0	0	52
	Löser	136738	66879	50022	78011
	Total	136738	66879	50022	78062
Gesp. Löser-Zeit		0	69859	86716	58727
Gesp. Löser-Zeit rel.		0,00	0,51	0,63	0,43

Tabelle 5.7: Alle BFGS-Testprobleme mit analytischer Jacobimatrix, Teil 1.

Größe \ Methode		Benchmark	SBFGS 2	SBFGS 10	SBFGS ∞
Probleme		920	920	920	920
+	Optimal	853	849	858	863
	Akz.	28	42	26	24
	Total	881	891	884	887
-	Out of Mem	0	0	0	0
	Löser-TO	25	10	4	3
	Andere	14	19	32	30
	Total	39	29	36	33
nnz		6883501	6883501	6883501	6883501
zus. Speicher		0	11974618	8007172	10817001
Zeit	Block	0	12358	2191	5588
	Löser	136738	67351	46643	45676
	Total	136738	79709	48834	51265
Gesp. Löser-Zeit		0	69387	90095	91061
Gesp. Löser-Zeit rel.		0,00	0,51	0,66	0,67

Tabelle 5.8: Alle BFGS-Testprobleme mit analytischer Jacobimatrix, Teil 2.

6 Fazit

"Ich habe kaum jemals einen Mathematiker kennengelernt, der in der Lage war, vernünftige Schlussfolgerungen zu ziehen."
Plato 428-347 v. Chr.

In der vorliegenden Arbeit wurden verschiedene Verfahren entwickelt und analysiert zur Berechnung beziehungsweise zur Approximation von Ableitungen in der hochdimensionalen nichtlinearen Optimierung.
Zur Bestimmung der ersten Ableitung eignen sich die Finiten Differenzen, da sie eine hohe Genauigkeit liefern. Auf der anderen Seite haben sie jedoch den Nachteil der hohen Rechenzeit. Für die Ableitungsberechnung muss eine riesige Zahl an Funktionsauswertungen durchgeführt werden, die meist numerisch sehr aufwändig sind. In dieser Arbeit wurden verschiedenen Wege aufgezeigt, wie die Berechnung der Ableitungen mittels Finiter Differenzen extrem beschleunigt werden kann. Hierbei haben wir uns Resultate und Algorithmen aus der Graphentheorie zu Nutze gemacht. Über die Lösung des Graph Coloring Problems wurde ein Verfahren entwickelt, welches zahlreiche Funktionsauswertungen parallel berechnet.
Das Graph Coloring Problem besteht darin, die Ecken eines vorgegebenen Graphen so zu colorieren, dass verbundene Ecken stets unterschiedliche Farben haben und möglichst wenig Farben benötigt werden. Nachdem wir einen Transfer zwischen einer zu berechnenden Jacobimatrix und einem zu colorierenden Graphen hergestellt haben, entspricht die Anzahl der Farben im colorierten Graph der Anzahl der notwendigen Funktionsauswertungen.
Bedauerlicherweise ist das Graph Coloring Problem nicht in polynomialer Zeit exakt lösbar, so dass man sich heuristischer Verfahren bedienen muss, die eine zulässige, aber lediglich suboptimale Colorierung liefern, dafür jedoch in polynomialer Zeit enden. Wie wir an den hier betrachteten Algorithmen gesehen haben, liefern aufwändigere Heuristiken tendenziell bessere Resultate als weniger aufwändige. Insgesamt zeigte sich hier jedoch, dass der einfachste Ansatz deutlich schneller ist und nicht sehr viel schlechtere Ergebnisse liefert.
Bei der Bestimmung der zweiten Ableitungen konkurrieren die beiden hier behandelten Themenkomplexe miteinander: Die Finiten Differenzen und die Update-Techniken.
Während sich in der niedrigdimensionalen nichtlinearen Optimierung die BFGS-Update-Technik zum Standard entwickelt hat, ist man bei hochdimensionalen Problemen gezwungen, andere Wege zu gehen. Die Verwendung des klassischen BFGS-Verfahren führt zu einem zu großen Speicher- und Rechenaufwand bei hohen Dimensionen.
In dieser Arbeit wurde einerseits die Idee der Finiten Differenzen mittels Graph Coloring auf die zweiten Ableitungen erweitert. Hierbei wurden zwei unterschiedliche Konzepte entwickelt: Einerseits die Hessegruppen- und andererseits die Paargruppenmethode. Die Komplexität der Bestimmung der Hesse- und Paargruppen ist deutlich höher als die der Bestimmung der gewöhnlichen Gruppen, die bei den ersten Ableitungen benötigt werden. Beide Verfahren erzielen jedoch eine noch höhere Effizienzsteigerung als es die gewöhnlichen Gruppen bei

den ersten Ableitungen vermögen. Darüber hinaus haben wir nachgewiesen, dass Hesse- und Paargruppenmethode für gewöhnlich tatsächlich unterschiedlich vieler Funktionsauswertungen bedürfen und dass bei verschiedenen Problemen gelegentlich die eine Strategie dominiert und gelegentlich die andere. Auch die numerische Untersuchung unserer zahlreichen Beispiele gab nicht zu erkennen, dass eine der beiden Strategien insgesamt der anderen vorzuziehen sei.

Trotz der enormen Effizienzsteigerung, die wir mit Hilfe der Graphentheorie bei den Finiten Differenzen erzielen konnten, bildet der Schwerpunkt dieser Arbeit die Entwicklung eines BFGS-ähnlichen Update-Verfahrens, welches die Dünnbesetztheit berücksichtigt. Hierbei haben wir ausgehend von einem simplen Block-BFGS-Ansatz über Ansätze mit sich überschneidenden Blöcken ein Verfahren entwickelt, welches die Dünnbesetztheit erhält und lokal superlinear gegen die optimale Lösung konvergiert. Nach unserem Kenntnisstand ist dies das einzige dünnbesetzte Update-Verfahren, für welches unter solch allgemeinen Voraussetzungen, wie sie hier getroffen wurden, superlineare Konvergenz nachgewiesen werden konnte.

Die numerischen Resultate belegen, dass dieses Verfahren noch effizienter ist als die Finiten Differenzen mit den extrem beschleunigenden Gruppen-, Paargruppen- und Hessegruppenmethoden. Die Vielzahl an gelösten Problemen untermauert außerdem die zuvor theoretisch nachgewiesenen Konvergenzaussagen.

Literaturverzeichnis

[1] W. Alt. *Nichtlineare Optimierung*. Vieweg Studium: Aufbaukurs Mathematik. [Vieweg Studies: Mathematics Course]. Friedr. Vieweg & Sohn, Braunschweig, 2002. Eine Einführung in Theorie, Verfahren und Anwendungen. [An introduction to theory, procedures and applications].

[2] R. Baier, C. Büskens, I. A. Chahma, and M. Gerdts. Approximation of reachable sets by direct solution methods for optimal control problems. *Optim. Methods Softw.*, 22(3):433–452, 2007.

[3] F. L. Bauer and C. T. Fike. Norms and exclusion theorems. *Numer. Math.*, 2:137–141, 1960.

[4] D. Brélaz. New methods to color the vertices of a graph. *Comm. ACM*, 22(4):251–256, 1979.

[5] 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.

[6] A. Buckley and A. Lenir. QN-like variable storage conjugate gradients. *Math. Programming*, 27(2):155–175, 1983.

[7] A. G. Buckley. A combined conjugate-gradient quasi-Newton minimization algorithm. *Math. Programming*, 15(2):200–210, 1978.

[8] C. Büskens. Real-time solutions for perturbed optimal control problems by a mixed open- and closed-loop strategy. In *Online optimization of large scale systems*, pages 105–116. Springer, Berlin, 2001.

[9] C. Büskens. Zuwendungsantrag an das DLR für Sparse NLP Solver. 2006.

[10] C. Büskens. Optimierung, Vorlesungsskript Sommersemester 2008. Unkorrigierte Fassung, 2008.

[11] C. Büskens and K. Chudej. Parametric sensitivity analysis: a case study in optimal control of flight dynamics. In *System modeling and optimization, XX (Trier, 2001)*, volume 130 of *IFIP Int. Fed. Inf. Process.*, pages 189–197. Kluwer Acad. Publ., Boston, MA, 2003.

[12] C. Büskens and R. Griesse. Parametric sensitivity analysis of perturbed PDE optimal control problems with state and control constraints. *J. Optim. Theory Appl.*, 131(1):17–35, 2006.

[13] C. Büskens, P. Kalmbach, T. Nikolayzik, and D. Wassel. WORHP, Tätigkeitsbericht des ZeTeM. 2010.

[14] C. Büskens and M. Knauer. Higher order real-time approximations in optimal control of multibody-systems for industrial robots. *Multibody Syst. Dyn.*, 15(1):85–106, 2006.

[15] C. Büskens and H. Maurer. SQP-methods for solving optimal control problems with control and state constraints: adjoint variables, sensitivity analysis and real-time control. *J. Comput. Appl. Math.*, 120(1-2):85–108, 2000. SQP-based direct discretization methods for practical optimal control problems.

[16] C. Büskens and H. Maurer. Sensitivity analysis and real-time control of parametric optimal control problems using nonlinear programming methods. In *Online optimization of large scale systems*, pages 57–68. Springer, Berlin, 2001.

[17] C. Büskens, H. J. Pesch, and S. Winderl. Real-time solutions of bang-bang and singular optimal control problems. In *Online optimization of large scale systems*, pages 129–142. Springer, Berlin, 2001.

[18] R. H. Byrd and J. Nocedal. An analysis of reduced Hessian methods for constrained optimization. *Math. Programming*, 49(3, (Ser. A)):285–323, 1990/91.

[19] R. H. Byrd, J. Nocedal, and R. B. Schnabel. Representations of quasi-Newton matrices and their use in limited memory methods. *Math. Programming*, 63(2, Ser. A):129–156, 1994.

[20] W. M. Chan and A. George. A linear time implementation of the reverse Cuthill-McKee algorithm. *BIT*, 20(1):8–14, 1980.

[21] T. F. Coleman and J. J. Moré. Estimation of sparse Hessian matrices and graph coloring problems. *Math. Programming*, 28(3):243–270, 1984.

[22] S. A. Cook. The complexity of theorem-proving procedures. In *STOC '71: Proceedings of the third annual ACM symposium on Theory of computing*, pages 151–158, New York, NY, USA, 1971. ACM Press.

[23] A. R. Curtis, M. J. D. Powell, and J. K. Reid. On the estimation of sparse Jacobian matrices. *J. Inst. Math. Appl.*, 13:117–119, 1974.

[24] J. E. Dennis, Jr. and J. J. Moré. A characterization of superlinear convergence and its application to quasi-Newton methods. *Math. Comp.*, 28:549–560, 1974.

[25] G. Fischer. *Lineare Algebra*, volume 17 of *Grundkurs Mathematik [Foundational Course in Mathematics]*. Friedr. Vieweg & Sohn, Braunschweig, fifth edition, 1979. In collaboration with Richard Schimpl.

[26] R. Fletcher. A new approach to variable metric algorithms. *Comput. J.*, 13(3):317–322, 1970.

[27] R. Fletcher. *Practical methods of optimization.* A Wiley-Interscience Publication. John Wiley & Sons Ltd., Chichester, second edition, 1987.

[28] R. Fletcher. An optimal positive definite update for sparse Hessian matrices. *SIAM J. Optim.*, 5(1):192–218, 1995.

[29] R. Fletcher, A. Grothey, and S. Leyffer. Computing sparse Hessian and Jacobian approximations with optimal hereditary properties. In *Large-scale optimization with applications, Part II (Minneapolis, MN, 1995)*, volume 93 of *IMA Vol. Math. Appl.*, pages 37–52. Springer, New York, 1997.

[30] M. R. Garey and D. S. Johnson. *Computers and intractability*. W. H. Freeman and Co., San Francisco, Calif., 1979. A guide to the theory of NP-completeness, A Series of Books in the Mathematical Sciences.

[31] M. R. Garey, D. S. Johnson, and L. Stockmeyer. Some simplified np-complete problems. In *STOC '74: Proceedings of the sixth annual ACM symposium on Theory of computing*, pages 47–63, New York, NY, USA, 1974. ACM Press.

[32] A. H. Gebremedhin, A. Pothen, and A. Walther. Exploiting sparsity in Jacobian computation via coloring and automatic differentiation: a case study in a simulated moving bed process. In *Advances in automatic differentiation*, volume 64 of *Lect. Notes Comput. Sci. Eng.*, pages 327–338. Springer, Berlin, 2008.

[33] A. H. Gebremedhin, A. Tarafdar, A. Pothen, and A. Walther. Efficient computation of sparse Hessians using coloring and automatic differentiation. *INFORMS J. Comput.*, 21(2):209–223, 2009.

[34] C. Geiger, C. und Kanzow. *Numerische Verfahren zur Lösung unrestringierter Optimierungsaufgaben*. Springer, Berlin-Heidelberg-New York, 1999.

[35] C. Geiger, C. und Kanzow. *Theorie und Numerik restringierter Optimierungsaufgaben*. Springer, Berlin-Heidelberg-New York, 2002.

[36] M. Gerdts. Optimal control and real-time optimization of mechanical multi-body systems. *ZAMM Z. Angew. Math. Mech.*, 83(10):705–719, 2003. ECMI-Workshop "Numerical Methods in Multibody Dynamics" (Bad Herrenalb, 2001).

[37] M. Gerdts. Global convergence of a nonsmooth Newton method for control-state constrained optimal control problems. *SIAM J. Optim.*, 19(1):326–350, 2008.

[38] M. Gerdts and M. Kunkel. A nonsmooth Newton's method for discretized optimal control problems with state and control constraints. *J. Ind. Manag. Optim.*, 4(2):247–270, 2008.

[39] P. E. Gill and M. W. Leonard. Reduced-Hessian quasi-Newton methods for unconstrained optimization. *SIAM J. Optim.*, 12(1):209–237 (electronic), 2001.

[40] P. E. Gill and M. W. Leonard. Limited-memory reduced-Hessian methods for large-scale unconstrained optimization. *SIAM J. Optim.*, 14(2):380–401 (electronic), 2003.

[41] P. E. Gill and W. Murray. Numerically stable methods for quadratic programming. *Math. Programming*, 14(3):349–372, 1978.

[42] P. E. Gill, W. Murray, M. A. Saunders, and M. H. Wright. Model building and practical aspects of nonlinear programming. In *Computational mathematical programming (Bad Windsheim, 1984)*, volume 15 of *NATO Adv. Sci. Inst. Ser. F Comput. Systems Sci.*, pages 209–247. Springer, Berlin, 1985.

[43] P. E. Gill, W. Murray, and M. H. Wright. *Practical optimization*. Academic Press Inc. [Harcourt Brace Jovanovich Publishers], London, 1981.

[44] D. Goldfarb. A family of variable-metric methods derived by variational means. *Math. Comp.*, 24:23–26, 1970.

[45] N. I.M. Gould, D. Orban, and P. L. Toint. http://www.hsl.rl.ac.uk/cuter-www/, November 2010.

[46] A. Griewank. On automatic differentiation. In *Mathematical programming (Tokyo, 1988)*, volume 6 of *Math. Appl. (Japanese Ser.)*, pages 83–107. SCIPRESS, Tokyo, 1989.

[47] A. Griewank. A mathematical view of automatic differentiation. *Acta Numer.*, 12:321–398, 2003.

[48] A. Griewank and Ph. L. Toint. Local convergence analysis for partitioned quasi-Newton updates. *Numer. Math.*, 39(3):429–448, 1982.

[49] A. Griewank and J. Utke. Automatisches Differenzieren als kombinatorisches Problem. In *Jahrbuch Überblicke Mathematik, 1995*, pages 85–102. Vieweg, Braunschweig, 1995.

[50] S. P. Han. A globally convergent method for nonlinear programming. *J. Optimization Theory Appl.*, 22(3):297–309, 1977.

[51] J. Herskovits and E. Goulart. Sparse quasi-newton matrices for large scale nonlinear optimization. *6th World Congress on Structural and Multidisciplinary Optimization*, 2005.

[52] S. Hossain and T. Steihaug. Optimal direct determination of sparse jacobian matrices. Technical Report 254, Department of Informatics, University of Bergen, Norway, oct 2003. Revised version to appear in *Optimization Methods and Software.*

[53] D. S. Johnson. Worst case behavior of graph coloring algorithms. In *Proceedings of the Fifth Southeastern Conference on Combinatorics, Graph Theory and Computing (Florida Atlantic Univ., Boca Raton, Fla., 1974)*, pages 513–527. Congressus Numerantium, No. X, Winnipeg, Man., 1974. Utilitas Math.

[54] L. W. June and M. A. Hassan. Modifications of the limited memory BFGS algorithm for large-scale nonlinear optimization. *Math. J. Okayama Univ.*, 47:175–188, 2005.

[55] C. A. Kluever. Optimal feedback guidance for low-thrust orbit insertion. *Optimal Control Applications and Methods*, 16:155–173, 1995.

[56] D. C. Liu and J. Nocedal. On the limited memory BFGS method for large scale optimization. *Math. Programming*, 45(3, (Ser. B)):503–528, 1989.

[57] K. Malanowski, C. Büskens, and H. Maurer. Convergence of approximations to nonlinear optimal control problems. In *Mathematical programming with data perturbations*, volume 195 of *Lecture Notes in Pure and Appl. Math.*, pages 253–284. Dekker, New York, 1998.

[58] O. L. Mangasarian. *Nonlinear programming.* McGraw-Hill Book Co., New York, 1969.

[59] J. R. R. A. Martins, P. Sturdza, and J. J. Alonso. The complex-step derivative approximation. *ACM Trans. Math. Softw.*, 29:245–262, September 2003.

[60] D. W. Matula and L. L. Beck. Smallest-last ordering and clustering and graph coloring algorithms. *J. Assoc. Comput. Mach.*, 30(3):417–427, 1983.

[61] D. W. Matula, G. Marble, and J. D. Isaacson. Graph coloring algorithms. In *Graph theory and computing*, pages 109–122. Academic Press, New York, 1972.

[62] L. Nazareth. A relationship between the BFGS and conjugate gradient algorithms and its implications for new algorithms. *SIAM J. Numer. Anal.*, 16(5):794–800, 1979.

[63] G. N. Newsam and J. D. Ramsdell. Estimation of sparse Jacobian matrices. *SIAM J. Algebraic Discrete Methods*, 4(3):404–418, 1983.

[64] J. Nocedal. Updating quasi-Newton matrices with limited storage. *Math. Comp.*, 35(151):773–782, 1980.

[65] J. Nocedal and S. J. Wright. *Numerical optimization*. Springer Series in Operations Research and Financial Engineering. Springer, New York, second edition, 2006.

[66] A. Perry. A class of conjugate gradient algorithms with a two-step variable metric memory. Discussion Papers 269, Northwestern University, Center for Mathematical Studies in Economics and Management Science, January 1977.

[67] M. J. D. Powell. The convergence of variable metric methods for nonlinearly constrained optimization calculations. In *Nonlinear programming, 3 (Proc. Sympos., Special Interest Group Math. Programming, Univ. Wisconsin, Madison, Wis., 1977)*, pages 27–63. Academic Press, New York, 1978.

[68] M. J. D. Powell. A fast algorithm for nonlinearly constrained optimization calculations. In *Numerical analysis (Proc. 7th Biennial Conf., Univ. Dundee, Dundee, 1977)*, pages 144–157. Lecture Notes in Math., Vol. 630. Springer, Berlin, 1978.

[69] K. Schittkowski. *Nonlinear programming codes*, volume 183 of *Lecture Notes in Economics and Mathematical Systems*. Springer-Verlag, Berlin, 1980. Information, tests, performance.

[70] K. Schittkowski. The nonlinear programming method of Wilson, Han, and Powell with an augmented Lagrangian type line search function. I. convergence analysis. *Numer. Math.*, 38(1):83–114, 1981.

[71] K. Schittkowski. Organization, test, and performance of optimization programs. In *Optimization and optimal control (Proc. Conf., Math. Res. Inst., Oberwolfach, 1980)*, volume 30 of *Lecture Notes in Control and Information Sci.*, pages 109–122. Springer, Berlin, 1981.

[72] K. Schittkowski. The nonlinear programming method of Wilson, Han, and Powell with an augmented Lagrangian type line search function. II. An efficient implementation with linear. *Numer. Math.*, 38(1):115–127, 1981/82.

[73] K. Schittkowski. On the convergence of a sequential quadratic programming method with an augmented Lagrangian line search function. *Math. Operationsforsch. Statist. Ser. Optim.*, 14(2):197–216, 1983.

[74] Klaus Schittkowski, editor. *Computational mathematical programming*, volume 15 of *NATO Advanced Science Institutes Series F: Computer and Systems Sciences*, Berlin, 1985. Springer-Verlag.

[75] S. Seelecke, C. Büskens, I. Müller, and J. Sprekels. Real-time optimal control of shape memory alloy actuators in smart structures. In *Online optimization of large scale systems*, pages 93–104. Springer, Berlin, 2001.

[76] D. F. Shanno. Conditioning of quasi-Newton methods for function minimization. *Math. Comp.*, 24:647–656, 1970.

[77] D. F. Shanno. Conjugate gradient methods with inexact searches. *Math. Oper. Res.*, 3(3):244–256, 1978.

[78] D. F. Shanno. On the convergence of a new conjugate gradient algorithm. *SIAM J. Numer. Anal.*, 15(6):1247–1257, 1978.

[79] M. C. Steinbach. A structured interior point SQP method for nonlinear optimal control problems. In *Computational optimal control (Munich, 1992)*, volume 115 of *Internat. Ser. Numer. Math.*, pages 213–222. Birkhäuser, Basel, 1994.

[80] J. Stoer. Principles of sequential quadratic programming methods for solving nonlinear programs. In *Computational mathematical programming (Bad Windsheim, 1984)*, volume 15 of *NATO Adv. Sci. Inst. Ser. F Comput. Systems Sci.*, pages 165–207. Springer, Berlin, 1985.

[81] P. Stumm, A. Walther, J. Riehme, and U. Naumann. Structure-exploiting automatic differentiation of finite element discretizations. In *Advances in automatic differentiation*, volume 64 of *Lect. Notes Comput. Sci. Eng.*, pages 339–349. Springer, Berlin, 2008.

[82] Ph. L. Toint. On sparse and symmetric matrix updating subject to a linear equation. *Math. Comp.*, 31(no 140):954–961, 1977.

[83] Ph. L. Toint. Some numerical results using a sparse matrix updating formula in unconstrained optimization. *Math. Comp.*, 32(143):839–851, 1978.

[84] A. Walther. Computing sparse Hessians with automatic differentiation. *ACM Trans. Math. Software*, 34(1):Art. 3, 15, 2008.

[85] D. J. A. Welsh and M. B. Powell. An upper bound for the chromatic number of a graph and its applications to timetabling problems. *The Computer Journal*, 10:85–86, 1967.

[86] Y. X. Yuan. A modified BFGS algorithm for unconstrained optimization. *IMA J. Numer. Anal.*, 11(3):325–332, 1991.