# Simultane Optimierung unter PDE-Nebenbedingungen

## Ein Vergleich zweier One-Shot-Methoden

DIPLOMARBEIT

zur Erlangung des akademischen Grades einer
Diplom-Mathematikerin

vorgelegt am Fachbereich IV
der Universität Trier

von

Stefanie Günther
Nikolausstr. 46, 54290 Trier

Betreuer: Prof. Dr. Volker Schulz

Trier, 27. Februar 2012

## Erklärung zur Diplomarbeit

Hiermit erkläre ich, daß ich die Diplomarbeit selbständig verfaßt und keine anderen als die angegeben Quellen und Hilfsmittel benutzt und die aus fremden Quellen direkt oder indirekt übernommenen Gedanken als solche kenntlich gemacht habe. Die Diplomarbeit habe ich bisher keinem anderen Prüfungsamt in gleicher oder vergleichbarer Form vorgelegt. Sie wurde bisher auch nicht veröffentlicht.

27. Februar 2012

(Datum) (Unterschrift)

# Danksagungen

An dieser Stelle möchte ich Prof. Dr. Volker Schulz für seine beständige Unterstützung danken. Schon während meinem Studium gab er mir die Möglichkeit, an diesem sehr interessanten Thema der aktuellen Forschung im Rahmen einer hilfswissenschaftlichen Tätigkeit mitarbeiten zu können. Ihm verdanke ich auch die besonders lehrreiche und spannende Praktikumsstelle beim Deutschen Zentrum für Luft- und Raumfahrt in Braunschweig, aus dem die Zusammenarbeit mit Prof. Dr. Nicolas Gauger entstand. Es folgten Aufenthalte in Berlin und schließlich in Aachen, bei denen ich mich auf die Unterstützung durch Prof. Schulz stets verlassen konnte.

Ebenfalls möchte ich mich bei Prof. Dr. Nicolas Gauger herzlich bedanken für viele aufschlussreiche Unterrichtungen und Gespräche während der letzten 2 Jahre, sowie für die Betreuung während des Praktikums und der Aufenthalte in Berlin und Aachen. Insbesondere die Bearbeitung des numerischen Teils dieser Arbeit wurde durch diese Zusammenarbeit ermöglicht und durch die Hilfe von Emre Özkaya unterstützt.

Ein weiterer Dank gilt Dr. Stephan Schmidt für viele Diskussionen und stundenlange Hilfe bei der Implementierung der One-Shot-Algorithmen. Er hat sich stets Zeit genommen, meine Fragen zur Programmierung, wie auch zur Theorie der One-Shot-Optimierung ausführlich zu beantworten.

Meinen Eltern danke ich in besonderem Maße. Sie haben mich motiviert, moralisch und finanziell unterstützt und ein Umfeld geschaffen, das mir dieses Studium erst ermöglicht hat.

# Inhaltsverzeichnis

# Abbildungsverzeichnis

# 1 Einleitung

Viele komplexe Systeme aus den Natur- und Ingenieurwissenschaften können mathematisch durch partielle Differentialgleichungen (PDE) modelliert werden, wie beispielsweise das der Diffusion, der Wellenausbreitung oder der Fluiddynamik. Eine analytische Lösung der Gleichungen ist in vielen Fällen aufgrund von Nichtlinearität der Gleichungen oder komplexen Randbedingungen nicht möglich. Es muss daher auf numerische Verfahren zur Lösung der partiellen Differentialgleichungen (sogenannte Zustandsgleichungen) zurückgegriffen werden. Durch die ständig weiterentwickelte Computertechnik und wachsende Rechnerleistung ist es in den letzten Jahrzehnten möglich geworden, detailreiche, komplexe Systeme numerisch immer besser zu simulieren.

Die numerische Simulation der Zustandsgleichung zielt darauf ab, die partiellen Differentialgleichungen für gegebene Daten, wie z.B. Geometrie, Koeffizienten, Randbedingungen oder Anfangsbedingungen der PDE, zu lösen, um den Zustand bzw. das Verhalten des Systems zu beschreiben (z.B. Geschwindigkeitsfeld und Temperatur eines Fluids). Setzt man den Prozess der Simulation eines Systems als vorhanden voraus, so ist eine natürliche Erweiterung durch den Prozess der Optimierung gegeben. Das Ziel der Optimierung ist es, einige dieser Daten so zu adjustieren, dass gewisse Zielsetzungen (Minimierung einer Zielfunktion) erreicht und gleichzeitig Bedingungen an das Verhalten des Systems erfüllt werden. Die partiellen Differentialgleichungen treten also als Nebenbedingung eines i.A. nichtlinearen, gleichungsbeschränkten Optimierungsproblems auf und werden in diesem Zusammenhang auch als „primale Zustandsgleichungen“ bezeichnet. Gesucht wird nun ein Optimalpunkt des Optimierungsproblems, welcher die Zielfunktion in einer Umgebung des Punktes minimiert und die PDE-Nebenbedingungen erfüllt.

Häufig können die Optimierungsvariablen dabei in natürlicher Weise getrennt werden in die systembeschreibenden Zustandsvariablen, also die Lösung der partiellen Differentialgleichung, und die sogenannten Designvariablen (auch Entscheidungs- oder Kontrollvariablen), anhand derer das Verhalten des Systems beeinflusst werden kann. Betrachtet werden also Probleme der Kontrolltheorie, welche im Wesentlichen durch Lions 1971 in [Lio71] und Pironneau ([Pir73], [Pir74], [Pir82]) begründet wurde (vgl. [HSBG05]). Abhängig von der Wahl der Zielfunktion und der Designvariablen kann das PDE-beschränkte Optimierungsproblem dann beispielsweise ein Problem der optimalen Kontrolle, des optimalen Designs aus der Formoptimierung oder ein inverses Problem der Parameterschätzung darstellen. Eine Einführung in die Thematik der PDE-beschränkten Optimierung sowie eine ausführliche Betrachtung des mathematischen Hintergrundes ist z.B. in [HPUU09] zu finden.

Obwohl heutzutage hohe Rechnerleisungen und weit entwickelte PDE-Lösungsmethoden vorhanden sind, kann die Simulation der partiellen Differentialgleichung besonders für komplexe und detailreiche Systeme mit einer Vielzahl von Zustandsvariablen zu immens hohen Rechen-

zeiten führen. Selbst auf Supercomputern kann die numerische Berechnung einer Lösung auf einem feinen Rechengitter unter Umständen viele Stunden dauern. Werden dann eher heuristische, d.h. insbesondere ableitungsfreie Verfahren zur Optimierung angewendet, welche sehr viele Lösungen der PDE und Funktionsauswertungen der Zielgröße benötigen, übersteigt der Rechenaufwand schnell die realistisch durchführbaren Grenzen. Sie kommen aus diesem Grund zur Optimierung mit partiellen Differentialgleichungen zumeist nicht in Frage; stattdessen werden gradientenbasierte Optimierungsmethoden vorgezogen. Dazu wird meist aus den notwendigen Bedingungen für Optimalität des nichtlinearen, beschränkten Optimierungsproblems ein iteratives, gradientenbasiertes Verfahren hergeleitet, etwa ein Quasi-Newton-Verfahren.

Es ist dabei stets darauf zu achten, dass die Kosten einer Optimierung die einer einzelnen Simulation um nicht mehr als einen geringen Faktor übersteigen und unabhängig von der Anzahl der Optimierungsvariablen sind. Diese Anforderung richtet sich sowohl an die Optimierungsmethode selbst, als auch an die Methode der Berechnung der Ableitungen. Um gradientenbasierte Verfahren anwenden zu können, wird die totale Ableitung der Zielfunktion nach jeder Designvariable benötigt, anhand derer ein Optimierungsschritt berechnet wird. Numerisch ist die Berechnung der Ableitungen mittels Finiter Differenzen möglich, allerdings ist diese Methode sehr fehleranfällig und direkt abhängig von der Dimension des Designraumes. Sie erfüllt die obige Anforderung damit nicht.

Eine seit ihrer Herleitung aus der Kontrolltheorie durch Pironneau [Pir74] sehr etablierte Alternative ist der Ansatz über die Adjungiertenmethode. Dazu werden neben den primalen Zustandsgleichungen weitere, sogenannte adjungierte Differentialgleichungen gelöst, mit deren Hilfe die totale Ableitung der Zielfunktion effizient berechnet werden kann (vgl. [Gau03]). Der Aufwand zur Berechnung der Ableitung ist dann unabhängig von der Anzahl der Designvariablen. Man unterscheidt generell zwischen zwei Ansätzen (vgl. [Gau03]): die kontunierliche Adjungierte und die diskrete Adjungierte. Im Bereich der aerodynamischen Formoptimierung wurde die kontinuierliche Adjungierte erstmals 1988 von Jameson [Jam88] zunächst für die Potentialgleichung beschrieben, später auch für die Euler- und die Navier-Stokes-Gleichungen. Von Giles [GP00] wurde hingegen die Herleitung der diskreten Adjungierten vorangetrieben. Die Anwendung der diskreten Adjungierten lässt sich durch den Einsatz von Automatischem Differenzieren (AD) im Rückwärtsmodus automatisieren (siehe z.B. [Gri00]). Auf die kontinuierlichen, sowie die diskreten Adjungiertengleichungen wird in Kapitel 2.5 noch eingegangen.

Ist ein Algorithmus zur Lösung sowohl der Zustandsgleichung (PDE-Löser), als auch der Adjungiertengleichung zur Berechnung der benötigten Ableitungen gegeben, so stellt sich die Frage nach einem geeigneten Optimierungsalgorithmus, der die numerischen Kosten gemessen in Rechenzeit möglichst gering hält. Bei einem gewöhnlichen gradientenbasierten Verfahren, wie etwa einem Quasi-Newton-Verfahren, muss in jeder Optimierungsiteration sowohl der primale als auch der adjungierte Algorithmus auskonvergiert werden, um anschließend mit den erhaltenen Ableitungsinformationen einen Schritt im Designraum auszuführen. Man spricht in diesem Fall von einem Black-Box-Ansatz oder auch „Nested Analysis and Design (NAND)" (siehe z.B. [Haz10]). Vorteilhaft ist dabei die geringe Interaktion des Optimierungsalgorithmus mit dem PDE-Löser. Allerdings muss die zeitaufwändige Lösung der Zustandsgleichung und der Adjungiertengleichung von Grund auf für die neuen Designvariablen in jedem Schritt der Opti-

mierungsiteration durchgeführt werden.

Die Idee der simultanen Optimierung ist es, diese wiederholten und damit kostenintensiven Lösungen der Zustands- und Adjungiertengleichung zu vermeiden, indem sie simultan mit dem Optimierungsproblem gelöst werden. Die involvierten Gleichungen, resultierend aus den notwendigen Optimalitätsbedingungen, werden gleichzeitig in einer gemeinsamen Iteration gelöst, sodass die Zustands- und Adjungiertengleichung während der Optimierung zunächst inexakt gelöst sind, im Zuge der Optimierungsiterationen aber gegen die exakte Lösung konvergieren. Viele Methoden, die dieser Idee folgen, sind in den letzten Jahren entwickelt worden. In der Literatur finden sich Bezeichnungen wie „Simultaneous Optimization Approach" ([TB88]), „simultaneous analysis and design (SAND)" ([HGK90], [Haz10]), „all-at-once approach" ([FS92], [CDF$^+$94]) und der hier verwendete Begriff der „One-Shot"-Optimierung ([TKS92], [HS04], [GH11]). Durch diesen Ansatz ist es möglich, die Gesamtkosten der Optimierung auf ein geringes Vielfaches einer einzelnen Simulation der PDE zu reduzieren.

In dieser Arbeit sollen zwei dieser simultanen Optimierungsmethoden verglichen werden. Es handelt sich dabei zum einen um die One-Shot-Methode, wie sie von Schulz et al. in [HS04], [HSBG05], [IKSG10] (und weiteren) vorgestellt wird. Sie wird in Kapitel 3 mittels approximativer, reduzierter SQP-Methode hergeleitet und als präkonditioniertes Pseudo-Zeitschrittverfahren interpretiert. Zum anderen wird in Kapitel 4 die One-Shot-Methode nach Griewank et al. ([Gri06], [GGR08], [GH10], [GH11] und andere) vorgestellt. Sie basiert auf der Verwendung von Automatischem Differenzieren (AD), sodass das Update aller involvierten Variablen anhand der Informationen einer Funktionsauswertung und deren adjungierten Variablen ausgeführt wird. Ein speziell gewählter Präkonditionierer im Designraum sichert die Konvergenz des Verfahrens. Kapitel 5 stellt die vorgestellten Verfahren gegenüber. Abschließend dient Kapitel 6 dem numerischen Vergleich der Methoden an einem speziellen Testfall.

# 2 Grundlagen

In diesem Kapitel sollen einige Grundlagen wiederholt werden, die zum Verständnis der folgenden Kapitel hilfreich sind. Da hier allgemeine Hilberträume betrachtet werden, wird zunächst ein Ableitungsbegriff auf Banachräumen eingeführt, sowie die Definitionen des Gradienten und der Hessematrix. Anschließend wird das betrachtete Minimierungsproblem in Abschnitt 2.2 vorgestellt und generelle Annahmen getroffen. Abschnitt 2.3 liefert die zugehörigen notwendigen und hinreichenden Optimalitätsbedingungen, aus denen numerische Verfahren zur Lösung abgeleitet werden können. Es folgt eine Darstellung des Hauptsatzes über implizite Funktionen, welcher der Vorbereitung von Abschnitt 2.5 dient, in dem das beschränkte Optimierungsproblem auf ein unbeschränktes Problem reduziert wird.

## 2.1 Fréchet-Ableitung, Gradient und Hesseoperator

Da zur Lösung des Optimierungsproblems gradientenbasierte Verfahren angewendet werden, soll hier zunächst auf den verwendeten Ableitungsbegriff auf Banachräumen eingegangen werden. Man vergleiche zu den folgenden Ausführungen auch [HPUU09], Kapitel 1.4.

**Definition 2.1** (Fréchet-Ableitung)**.** *Seien $X, Z$ Banachräume, $U \subseteq X$ offen und $f\colon U \to Z$. $f$ heißt Fréchet-differenzierbar an $a \in U$, wenn die Richtungsableitungen an $a$*

$$Df(a)(h) \coloneqq \lim_{t \to 0^+} \frac{1}{t}(f(a+th) - f(a)) \in Z \tag{2.1}$$

*für alle $h \in X$ existieren, $Df(a)\colon X \to Z$, $h \mapsto Df(a)(h)$ beschränkt und linear ist und*

$$\|f(a+h) - f(a) - Df(a)(h)\|_Z = o(\|h\|_X) \quad \textit{für} \quad \|h\|_X \to 0 \tag{2.2}$$

*gilt. $Df(a)$ heißt (Fréchet-) Ableitung von $f$ an $a$.*

Sind $X, Y$ und $Z$ Banachräume, so sind für einen Punkt $a = (x, y) \in X \times Y$ und eine an $a$ Fréchet-differenzierbare Funktion $f\colon X \times Y \to Z$ die Abbildungen $f(\cdot, y)$ und $f(x, \cdot)$ ebenfalls Fréchet-differenzierbar an $x$ und $y$. Deren Ableitungen werden als *partielle Ableitungen* bezeichnet mit der Notation $f_x(x,y)$ bzw. $f_y(x,y)$ (auch $\frac{\partial f(x,y)}{\partial x}$ bzw. $\frac{\partial f(x,y)}{\partial y}$) und es gilt

$$Df(x,y)(h_x, h_y) = f_x(x,y)h_x + f_y(x,y)h_y \tag{2.3}$$

Die Fréchet-Ableitung ist eindeutig, sodass sie in $\mathbb{R}^n$ mit der gewohnten Ableitung (partielle Ableitungen, Jakobimatrix) übereinstimmt. Des Weiteren lassen sich die üblichen Ableitungsregeln für Fréchet-Ableitungen herleiten (vgl. auch [BF95], Kapitel 14.2). Ableitungen höherer

Ordnung können dann analog definiert werden: Ist $f\colon X \to Z$ Fréchet-differenzierbar auf einer Umgebung $V$ von $a \in X$ und ist $Df$ als Abbildung von $X$ in den Raum der stetigen, linearen Operatoren von $X$ nach $Z$ ebenfalls Fréchet-differenzierbar an $a$, so heißt $f$ zweifach Fréchet-differenzierbar an $a$ mit der Bezeichnung $D^2 f(a)$. $f$ heißt $k$-mal stetig (Fréchet-) differenzierbar, wenn $f$ $k$-mal differenzierbar und $D^k f$ stetig ist.

Die im Folgenden betrachteten Zielfunktionen sind meist reellwertig, d.h. es werden Funktionen $f\colon X \to \mathbb{R}$ vorausgesetzt. In diesem Fall ist die oben definierte Ableitung $Df(a)$ von $f$ an der Stelle $a$ ein Element aus dem Dualraum $X^* = \{T\colon X \to \mathbb{R} | T \text{ ist linear und stetig}\}$. Ist des Weiteren $X$ ein Hilbertraum mit geeignetem reellwertigen Skalarprodukt $(\cdot,\cdot)_X$, so lässt sich die Ableitung von $f$ an $a$ mit Hilfe des folgenden Satzes von Fréchet-Riesz als ein Element von $X$ selbst darstellen.

**Theorem 2.2** (Darstellungssatz von Fréchet-Riesz)**.** *Sei $X$ ein Hilbertraum mit Skalarprodukt $(\cdot,\cdot)\colon X \times X \to \mathbb{R}$. Zu jedem Element $l \in X^*$ aus dem Dualraum von $X$ existiert ein eindeutig bestimmtes Element $y \in X$ mit*

$$l(x) = (y,x) \quad \forall x \in X. \tag{2.4}$$

*$y$ heißt darstellender Vektor von $l$.*

*Beweis.* Ein Beweis ist zu finden in [Wer09], Theorem V.3.13, Seite 322. □

**Definition 2.3** (Gradient)**.** *Sei die Abbildung $f\colon X \to \mathbb{R}$ auf einem Hilbertraum $X$ mit geeignetem Skalarprodukt $(\cdot,\cdot)\colon X \times X \to \mathbb{R}$ Fréchet-differenzierbar an $a \in X$ mit der Ableitung $Df(a) \in X^*$. Dann heißt der nach dem Darstellungssatz von Fréchet-Riesz (Theorem 2.2) eindeutig existierende darstellende Vektor $y \in X$ mit*

$$Df(a)(h) = (y,h)_X \quad \forall h \in X \tag{2.5}$$

*Gradient von $f$ an $a$.*

Der Gradient von $f$ an $a$ wird im Folgenden mit $\nabla f(a)$ bezeichnet. Offenbar ist er abhängig von der Wahl eines speziellen Skalarproduktes. Betrachtet wird hier stets das euklidische Skalarprodukt, sodass

$$\nabla f(a) = [Df(a)]^T \tag{2.6}$$

geschrieben werden kann. Zur Darstellung der zweiten Ableitung von $f$ dient die folgende Definition der Hessematrix von $f$, ebenfalls in Abhängigkeit eines speziellen Skalarproduktes.

**Definition 2.4** (Hesseoperator)**.** *Ist $f\colon X \to \mathbb{R}$ auf einem Hilbertraum $X$ mit geeignetem Skalarprodukt $(\cdot,\cdot)\colon X \times X \to \mathbb{R}$ zweimal Fréchet-differenzierbar an $a$, so ist der Hesseoperator (die Hessematrix) von $f$ an der Stelle $a$ definiert durch*

$$(Hess f(a)v, w)_X = D^2 f(a)(v)(w) \quad \forall v, w \in X \tag{2.7}$$

*Bemerkung.* Der Hesseoperator wird im Folgenden mit $\nabla^2 f(a)$ bezeichnet.

## 2.2 Optimierung mit partiellen Differentialgleichungen - Problemstellung

In dieser Arbeit werden nichtlineare Optimierungsprobleme betrachtet, bei denen der Vektor der Optimierungsvariablen aufgespaltet werden kann in Zustands- und Designvariablen. Eine stationäre partielle Differentialgleichung bildet die Nebenbedingung des Problems und liefert die Verknüpfung des Zustands- und Designvektors. Es gilt dann, die beiden Vektoren so zu adjustieren, dass eine Zielfunktion abhängig von Zustands- und Designvariablen minimiert wird und die Differentialgleichung erfüllt ist. Optimierungsprobleme dieser Art werden in der Literatur häufig als „optimal control problems" bezeichnet (vgl. [HPUU09]).

Das beschränkte Optimierungsproblem wird in der allgemeinen Form definiert als

$$\begin{aligned} \min_{y,u} \quad & f(y,u) \\ \text{s.t. } & c(y,u) = 0. \end{aligned} \tag{2.8}$$

Hierbei stellt $y \in Y$ den Vektor der Zustandsvariablen des betrachteten Systems dar, das über $c(y,u) = 0$ beschrieben ist und $u \in U$ ist der Vektor der Designvariablen. $Y$, $U$ sowie deren kartesisches Produkt $Y \times U$ seien geeignete Hilberträume. Für $c\colon Y \times U \to Y$ repräsentiert $c(y,u) = 0$ die (i.A. nichtlineare) zeitunabhängige partielle Differentialgleichung mit geeigneten Randbedingungen. Sie wird auch als primale Zustandsgleichung bezeichnet, da sie für einen gegebenen Designvektor $\hat{u}$ einen Zustand $\hat{y}$ eindeutig bestimmen soll (siehe Annahme 2.5). $f\colon Y \times U \to \mathbb{R}$ ist die zu minimierende Zielfunktion. $c$ und $f$ seien zweifach stetig differenzierbar.

Aus Gründen der übersichtlicheren Notation werden hier endlichdimensionale Hilberträume betrachtet. Dann lassen sich mit $dim(Y) = n$ und $dim(U) = m$ die Koordinatenvektoren der Zustands- und Designvariablen bezüglich einer geeigneten Basis der Hilberträume mit $\mathbb{R}^n$ bzw. $\mathbb{R}^m$ identifizieren, wobei meist $n \gg m$. Aus diesem Grund wird im Folgenden statt innerer Produkte stets das euklidische Skalarprodukt verwendet und Variablen aus dem Dualraum werden mit $()^T$ als die üblichen transponierten Vektoren bezeichnet.

Eine Grundvoraussetzung für die hier betrachteten separablen Optimierungsprobleme ist, dass zu einem Designvektor $u \in U$ stets eine durch die Zustandsgleichung eindeutig definierte Zustandsvariable $y \in Y$ existiert. Es wird daher die folgende Annahme getroffen:

***Annahme*** 2.5. Die Jacobimatrix $\frac{\partial c}{\partial y}(y,u) = c_y(y,u)$ ist regulär für alle betrachteten Punkte $(y,u) \in Y \times U$, d.h. $\det(c_y(y,u)) \neq 0$.

Damit ist die Voraussetzung des Hauptsatzes über implizite Funktionen (siehe Theorem 2.10) erfüllt und die eindeutige Existenz einer Funktion $y\colon U \to Y$ gesichert, welche einem Designvektor $u$ einen Zustandsvektor $y = y(u)$ eindeutig zuordnet, sodass $c(y(u),u) = 0$ erfüllt ist (vergleiche auch Kapitel 2.4 und 2.5).

**Definition 2.6.**

- *Ein Punkt* $(y^*,u^*) \in Y \times U$ *heißt globale Lösung von (2.8), falls* $c(y^*,u^*) = 0$ *und*

$$f(y^*,u^*) \leq f(y,u) \quad \forall (y,u) \in \{(y,u) \in Y \times U | c(y,u) = 0\}. \tag{2.9}$$

- $(y^*,u^*) \in Y \times U$ *heißt lokale Lösung von (2.8), falls* $c(y^*,u^*) = 0$ *und es existiert eine offene Umgebung N von* $(y^*,u^*)$*, sodass gilt*

$$f(y^*,u^*) \leq f(y,u) \quad \forall (y,u) \in N \cap \{(y,u) \in Y \times U | c(y,u) = 0\}. \tag{2.10}$$

- $(y^*,u^*) \in Y \times U$ *heißt strikte lokale Lösung von (2.8), falls* $c(y^*,u^*) = 0$ *und es existiert eine offene Umgebung N von* $(y^*,u^*)$*, sodass gilt*

$$f(y^*,u^*) < f(y,u) \quad \forall (y,u) \in N \cap \{(y,u) \in Y \times U | c(y,u) = 0\} \setminus \{(y^*,u^*)\}. \tag{2.11}$$

Gradientenbasierte numerische Verfahren zur Lösung des Optimierungsproblems (2.8) liefern im Allgemeinen nur lokale Lösungen, während ableitungsfreie Methoden meist eine globale Lösung finden. Letztere benötigen allerdings wesentlich mehr Funktionsauswertungen und bedürfen höherer Rechnerleistung. Sie kommen aus diesem Grund zur Optimierung unter partiellen Differentialgleichungen bei hochdimensionalen Problemen zumeist nicht in Frage.

## 2.3 Notwendige und hinreichende Optimalitätsbedingungen

Häufig werden numerische Verfahren zur Lösung des Minimierungsproblems (2.8) aus den notwendigen und hinreichenden Bedingungen für eine lokale Lösung des Problems hergeleitet. Diese sollen hier zusammengefasst dargestellt und auf das Optimierungsproblem mit separabler Struktur übertragen werden. Herleitungen der folgenden Theoreme sind in der Literatur (beispielsweise [NW06]) zu finden.

Für das Optimierungsproblem (2.8) sei die folgende, sogenannte *Lagrangefunktion* definiert durch

$$\mathscr{L}(y,\bar{y},u) \coloneqq f(y,u) + \bar{y}^T c(y,u) \tag{2.12}$$

mit Lagrangemultiplikator $\bar{y} \in Y^*$.

**Theorem 2.7** (Notwendige Bedingung erster Ordnung)**.** *Sei* $(y^*,u^*)$ *lokale Lösung von (2.8). Dann existiert ein Lagrangemultiplikator* $\bar{y}^* \in Y^*$*, sodass die Karush- Kuhn- Tucker- Bedingungen (KKT-Bedingungen)*

$$\nabla_y \mathscr{L}(y^*,\bar{y}^*,u^*) = 0 \tag{2.13}$$

$$\nabla_u \mathscr{L}(y^*,\bar{y}^*,u^*) = 0 \tag{2.14}$$

$$\nabla_{\bar{y}} \mathscr{L}(y^*,\bar{y}^*,u^*) = c(y^*,u^*) = 0 \tag{2.15}$$

*erfüllt sind.*

*Beweis.* Mit der Notation $x^* = (y^*, u^*)$ ist der Beweis in [NW06], Kapitel 12.4, zu finden. Zu beachten ist lediglich, dass die dort aufgeführte LICQ (linear independence constraint qualification) - Voraussetzung, welche sicherstellt, dass $x^*$ ein regulärer Punkt ist, durch die Annahme 2.5 schon erfüllt ist. □

**Theorem 2.8** (Notwendige Bedingung zweiter Ordnung)**.** *Sei $(y^*, u^*)$ lokale Lösung von (2.8) und sei $\bar{y}^*$ der Lagrangemultiplikator, für den die KKT-Bedingungen aus Theorem 2.7 erfüllt sind. Dann gilt, dass*

$$u^T T(y^*, u^*)^T H(y^*, u^*) T(y^*, u^*) u \geq 0 \quad \forall \quad u \in U, \tag{2.16}$$

*wobei*

$$T(y^*, u^*) := \begin{bmatrix} -(c_y(y^*, u^*))^{-1} c_u(y^*, u^*) \\ I \end{bmatrix} \tag{2.17}$$

*den Tangentialraum der Nebenbedingung an $(y^*, u^*)$ aufspannt und $H(y^*, u^*)$ die Hessematrix der Lagrangefunktion an $(y^*, u^*)$ mit dem Lagrangemultiplikator $\bar{y}^*$ darstellt:*

$$H(y^*, u^*) := \begin{bmatrix} \frac{\partial^2 \mathscr{L}(y^*, \bar{y}^*, u^*)}{\partial y \partial y} & \frac{\partial^2 \mathscr{L}(y^*, \bar{y}^*, u^*)}{\partial y \partial u} \\ \frac{\partial^2 \mathscr{L}(y^*, \bar{y}^*, u^*)}{\partial u \partial y} & \frac{\partial^2 \mathscr{L}(y^*, \bar{y}^*, u^*)}{\partial u \partial u} \end{bmatrix}. \tag{2.18}$$

*$T^T H T$ wird als reduzierte Hessematrix bezeichnet.*

*Beweis.* Nach [NW06], Theorem 12.5, besagt die notwendige Bedingung zweiter Ordnung, dass die Hessematrix der Lagrangefunktion an einer lokalen Lösung positiv semidefinit auf dem Tangentialraum $T_{(y^*,u^*)}c$ der Nebenbedingung an diesem Punkt, d.h. auf dem Kern der an $(y^*, u^*)$ linearisierten Nebenbedingung sein muss. Für das Optimalitätsproblem mit separablen Variablen kann dieser Raum genau angegeben werden, wenn die Annahme 2.5 erfüllt ist und damit $(c_y(y^*, u^*))^{-1}$ existiert:

$$\begin{aligned} T_{(y^*,u^*)}c = \operatorname{kern}(Dc(y^*, u^*)) &= \{(y,u) \in Y \times U \quad | \quad c_y(y^*, u^*)y + c_u(y^*, u^*)u = 0\} && (2.19) \\ &= \{(y,u) \in Y \times U \quad | \quad y = -(c_y(y^*, u^*))^{-1} c_u(y^*, u^*)u\} && (2.20) \\ &= \{\underbrace{\begin{bmatrix} -(c_y(y^*, u^*))^{-1} c_u(y^*, u^*) \\ I \end{bmatrix}}_{=:T(y^*,u^*)} u \quad | \quad u \in U\} && (2.21) \end{aligned}$$

Durch diese Charakterisierung des Tangentialraums lässt sich die Bedingung reduzieren:

$$\begin{pmatrix} y \\ u \end{pmatrix}^T \begin{bmatrix} \frac{\partial^2 \mathscr{L}}{\partial y \partial y} & \frac{\partial^2 \mathscr{L}}{\partial y \partial u} \\ \frac{\partial^2 \mathscr{L}}{\partial u \partial y} & \frac{\partial^2 \mathscr{L}}{\partial u \partial u} \end{bmatrix} \begin{pmatrix} y \\ u \end{pmatrix} \geq 0 \quad \forall \quad (y,u) \in T_{(y^*,u^*)}c \tag{2.22}$$

$$\iff \quad u^T T(y^*, u^*)^T H(y^*, u^*) T(y^*, u^*) u \geq 0 \quad \forall \quad u \in U \tag{2.23}$$

Damit muss die reduzierte Hessematrix am lokalen Minimalpunkt positiv semidefinit auf $U$ sein. □

**Theorem 2.9** (Hinreichende Bedingung zweiter Ordnung)**.** *Sind für $(y^*,u^*)$ und $\bar{y}^*$ die KKT-Bedingungen aus Theorem 2.7 erfüllt und gilt darüber hinaus*

$$u^T T(y^*,u^*)^T H(y^*,u^*) T(y^*,u^*) u > 0 \quad \forall \quad u \in U \setminus \{0\}, \tag{2.24}$$

*d.h. die reduzierte Hessematrix an $(y^*,u^*)$ ist positiv definit auf $U$, dann ist $(y^*,u^*)$ strikte lokale Lösung des Optimierungsproblems (2.8).*

*Beweis.* Siehe [NW06], Theorem 12.6, und die entsprechenden Bemerkungen aus den Beweisen zu Theorem 2.7 und 2.8. □

Numerische Lösungsverfahren suchen häufig nach einem Punkt, welcher die KKT-Bedingungen erfüllt. Es wird generell angenommen, dass für diesen Punkt die hinreichende Bedingung zweiter Ordnung erfüllt ist, da diese Bedingung (analog dem unbeschränkten Fall) garantiert, dass sich der Lösungspunkt $(y^*,\bar{y}^*,u^*)$ eindeutig aus dem System der KKT-Bedingungen berechnen lässt. Des Weiteren ist durch diese Bedingung die Stabilität der Lösung unter kleinen Störungen der Problemdaten gesichert.

## 2.4 Der Hauptsatz über implizite Funktionen

Für die in dieser Arbeit betrachteten Optimierungsprobleme mit separabler Variablenstruktur muss stets sichergestellt sein, dass die Nebenbedingung des Problems eindeutig lösbar ist. Dies erfolgt durch Anwendung des Hauptsatzes über implizite Funktionen. Da dessen Ergebnisse auch im Folgenden eine Rolle spielen, soll er hier für Banachräume wiedergegeben werden.

**Theorem 2.10** (Hauptsatz über implizite Funktionen)**.** *Seien $X,Y,Z$ Banachräume, $G \subseteq X \times Y$ offen, die Abbildung $h\colon G \to Z$ sei zweifach stetig differenzierbar auf $G$. Sei des Weiteren $h(x_0,y_0) = 0$ für $(x_0,y_0) \in G$ und die partielle Ableitung $h_y(x_0,y_0)$ sei bijektiv.*

*Dann existiert eine offene Umgebung $V(x_0) \times W(y_0) \subseteq G$ von $(x_0,y_0)$ und eine eindeutige Abbildung $\phi\colon V(x_0) \to Y$ so, dass $\phi(x_0) = y_0$ und für alle $x \in V(x_0)$ ist ein $y \in W(y_0)$, welches $h(x,y) = 0$ löst, eindeutig durch $y = \phi(x)$ gegeben.*

*Die Abbildung $\phi\colon V(x_0) \to Y$ ist stetig differenzierbar und die Ableitung ist gegeben durch*

$$D\phi(x) = -h_y(x,\phi(x))^{-1} h_x(x,\phi(x)). \tag{2.25}$$

*Beweis.* Ein Beweis ist z.B. in [RÔ4], Satz 2.9 und Folgerung 2.15, Seite 49f., zu finden. Die Ableitung der Abbildung $\phi$ erhält man durch Differentiation und Anwendung der Kettenregel auf $h(x,\phi(x)) = 0$. □

## 2.5 Reduziertes Problem und der adjungierte Ansatz

Nach der Annahme 2.5 aus Kapitel 2.2, dass $c_y$ regulär ist, ist die Voraussetzung des Hauptsatzes über implizite Funktionen stets erfüllt. Dies sichert, dass der Zustandsvektor $y$ als abhängige

Variable $y = y(u)$ betrachtet werden kann. Für jeden Designvektor $u \in U$ existiert also ein eindeutiger Zustand $y(u)$ so, dass $c(y(u),u) = 0$ gilt. Damit lässt sich die Nebenbedingung des Optimierungsproblems (2.8) eliminieren und man erhält das folgende unbeschränkte, reduzierte Optimierungsproblem

$$\min_u \tilde{f}(u) := f(y(u),u). \tag{2.26}$$

$y(u)$ ist für jedes $u$ durch $c(y(u),u) = 0$ eindeutig festgelegt. Um gradientenbasierte Verfahren zur Lösung anwenden zu können, ist die effiziente Berechnung der Ableitung der reduzierten Zielfunktion $\tilde{f}$ notwendig.

Eine Möglichkeit der numerischen Berechnung der Ableitung ist die Verwendung von finiten Differenzen (vgl. [Haz10]). Dabei wird mit

$$\left(\frac{\partial \tilde{f}}{\partial u}\right)_i \approx \frac{f(y(u+\varepsilon e_i), u+\varepsilon e_i) - f(y(u),u)}{\varepsilon} \quad i = 1 \dots m \tag{2.27}$$

für jeden Designparameter der entsprechende Eintrag der Ableitung approximiert. Dieser Ansatz benötigt allerdings $m+1$ Systemsimulationen, d.h. Lösungen der partiellen Differentialgleichung, um $y(u)$ und $y(u+\varepsilon e_i)$ zu erhalten. Des Weiteren ist die Wahl eines geeigneten $\varepsilon$ problematisch und fehleranfällig. Nach Definition der Ableitung müsste $\varepsilon$ gegen 0 gehen, allerdings darf $\varepsilon$ nicht zu klein gewählt werden, um die bei numerischer Berechnung auftretenden Rundungsfehler gering zu halten.

Nimmt man von diesem Ansatz Abstand, so unterscheidet man generell zwischen zwei weiteren Methoden (siehe [HPUU09]): die der direkten Sensitivitäten und die der adjungierten Sensitivitäten. Beide gehen von der nach Anwendung der Kettenregel erhaltenen Formulierung der totalen Ableitung von $\tilde{f}$ nach $u$ aus:

$$D\tilde{f}(u) = \frac{\partial f}{\partial y}(y(u),u)\frac{\partial y(u)}{\partial u} + \frac{\partial f}{\partial u}(y(u),u). \tag{2.28}$$

Der Unterschied der Methoden besteht in der Berechnung des ersten Terms, welcher die Sensitivitäten $\frac{\partial y(u)}{\partial u}$ enthält. Der Hauptsatz über implizite Funktionen liefert neben der Existenz der Abbildung $y(u)$ auch dessen Ableitung (vgl. Theorem 2.10):

$$\frac{\partial y}{\partial u} = -\left(\frac{\partial c}{\partial y}\right)^{-1}\frac{\partial c}{\partial u}. \tag{2.29}$$

Die Methode der direkten Sensitivitäten berechnet diese Ableitung direkt durch Lösen des Systems

$$\frac{\partial c(y(u),u)}{\partial y}\frac{\partial y(u)}{\partial u} = -\frac{\partial c(y(u),u)}{\partial u}. \tag{2.30}$$

Dies entspricht dem $m$-maligen Lösen eines Gleichungssystems mit Systemmatrix $\frac{\partial c}{\partial y}$, dessen rechte Seite die Spalten von $-\frac{\partial c}{\partial u}$ bilden. Durch Einsetzen von $\frac{\partial y}{\partial u}$ in (2.28) erhält man die gewünschte Ableitung. Der numerische Aufwand ist zwar i. A. geringer als der der finiten Differenzen, allerdings ist er noch immer abhängig von der Dimension $m$ des Designraumes.

Durch die Anwendung des adjungierten Ansatzes lässt sich diese Abhängigkeit vermeiden. Anstelle der Berechnung der Sensivitätenmatrix $\frac{\partial y(u)}{\partial u}$ wird die sogenannte adjungierte Variable $\bar{y}$ als Lösung von

$$\left(\frac{\partial c(y(u),u)}{\partial y}\right)^T \bar{y} = -\nabla_y f(y(u),u) \tag{2.31}$$

berechnet, wobei $\nabla_y f$ der Frechét-Riesz'sche Darsteller von $\frac{\partial f}{\partial y}$ sei (vgl. Theorem 2.2). Gleichung (2.31) wird als **Adjungiertengleichung** bezeichnet. Damit lässt sich der Frechét-Riesz'sche Darsteller der totalen Ableitung von $\tilde{f}$ nach Gleichung (2.28) und (2.29) berechnen als

$$\nabla_u \tilde{f} \coloneqq [D\tilde{f}(u)]^T \tag{2.32}$$

$$= \nabla_u f(y(u),u) - \left(\frac{\partial c}{\partial u}(y(u),u)\right)^T \underbrace{\left(\frac{\partial c}{\partial y}(y(u),u)\right)^{-T} \nabla_y f(y(u),u)}_{=-\bar{y}} \tag{2.33}$$

$$= \nabla_u f(y(u),u) + (c_u(y(u),u))^T \bar{y}. \tag{2.34}$$

Dieser wird **reduzierter Gradient** der Zielfunktion $f$ genannt. Zu seiner Berechnung ist also nach dem Lösen der primalen Zustandsgleichung um $y(u)$ zu erhalten, nur das einmalige Lösen der Adjungiertengleichung nach (2.31) und einsetzen in (2.34) notwendig. Damit ist der numerische Aufwand zur Berechnung des reduzierten Gradienten unabhängig von der Anzahl der Designparameter. Des Weiteren konvergieren iterative Prozesse zur Lösung der Adjungiertengleichung mit demselben asymptotischen Faktor wie dem der primalen Iteration, da $c_y$ und $c_y^T$ dieselben Eigenwerte aufweisen.

### 2.5.1 Lagrange-basierte Sichtweise der Adjungierten

Eine äquivalente Formulierung kann mit Hilfe der in (2.12) definierten Lagrangefunktion

$$\mathscr{L}(y,\bar{y},u) \coloneqq f(y,u) + \bar{y}^T c(y,u) \tag{2.35}$$

hergeleitet werden. Für jeden Lagrangemultiplikator $\bar{y}$ gilt nach der Reduzierung der Nebenbedingung stets

$$\tilde{f}(u) = f(y(u),u) + \bar{y}^T \underbrace{c(y(u),u)}_{=0} = \mathscr{L}(y(u),\bar{y},u). \tag{2.36}$$

Differentiation nach $u$ und Zusammenfassen der Terme, die die Sensitivität $\frac{\partial y}{\partial u}$ enthalten, liefert

$$D\tilde{f}(u) = \frac{\partial f(y(u),u)}{\partial y}\frac{\partial y(u)}{\partial u} + \frac{\partial f(y(u),u)}{\partial u} + \bar{y}^T\left(\frac{\partial c(y(u),u)}{\partial y}\frac{\partial y(u)}{\partial u} + \frac{\partial c(y(u),u)}{\partial u}\right) \tag{2.37}$$

$$= \left(\frac{\partial f(y(u),u)}{\partial y} + \bar{y}^T\frac{\partial c(y(u),u)}{\partial y}\right)\frac{\partial y(u)}{\partial u} + \frac{\partial f(y(u),u)}{\partial u} + \bar{y}^T\frac{\partial c(y(u),u)}{\partial u} \tag{2.38}$$

$$= \frac{\partial \mathscr{L}(y(u),\bar{y},u)}{\partial y}\frac{\partial y(u)}{\partial u} + \frac{\partial \mathscr{L}(y(u),\bar{y},u)}{\partial u}. \tag{2.39}$$

Wählt man nun einen speziellen Lagrangemultiplikator $\bar{y}$ so, dass der erste Term verschwindet, so wird die Berechnung von $\frac{\partial y}{\partial u}$ vermieden: Wähle $\bar{y}$ so, dass

$$\nabla_y \mathscr{L}(y(u), \bar{y}, u) = 0 \quad \Longleftrightarrow \quad \left(\frac{\partial c(y(u),u)}{\partial y}\right)^T \bar{y} = -\nabla_y f(y(u),u) \tag{2.40}$$

dann ist der Frechét-Riesz'schen Darsteller der totalen Ableitung $D\tilde{f}(u)$ gegeben durch

$$\nabla_u \tilde{f}(u) = \nabla_u \mathscr{L}(y(u), \bar{y}, u) = \nabla_u f(y(u),u) + (c_u(y(u),u))^T \bar{y} \tag{2.41}$$

Dies entspricht dem reduzierten Gradienten aus (2.34). Die notwendigen Bedingungen für einen lokalen Minimalpunkt $u^*$ des unbeschränkten Problems (2.26) mit zugehörigem Zustand $y^*$ schreiben sich dann als

$$c(y^*,u^*) = 0 \quad \text{„Zustandsgleichung"} \tag{2.42}$$

$$\nabla_y f(y^*,u^*) + c_y(y^*,u^*)^T \bar{y}^* = 0 \quad \text{„Adjungiertengleichung"} \tag{2.43}$$

$$\nabla_u f(y^*,u^*) + c_u(y^*,u^*)^T \bar{y}^* = 0 \quad \text{„Designgleichung"} \tag{2.44}$$

Dies sind genau die KKT-Bedingungen aus Theorem 2.7. Der dort auftretende Lagrangemultiplikator $\bar{y}^*$ entspricht damit der adjungierte Variable aus der Adjungiertengleichung.

### 2.5.2 Diskrete oder kontinuierliche Formulierung

In der Literatur wird häufig zwischen der diskreten und der kontinuierlichen Formulierung der Adjungiertengleichung unterschieden (siehe z.B. [GP00], [Gau03], [GP08]). Der diskrete Ansatz der Adjungiertengleichung geht von der schon diskretisierten, endlichdimensionalen Zustandsgleichung aus, d.h. die Differentialoperatoren treten schon in diskretisierter Form auf. Durch Transponieren der Systemmatrix $c_y$, welches zur Berechnung eines zulässigen $y(u)$ gebraucht wird, erhält man die Vorschrift zur Berechnung der adjungierten Variable. Erweitert man das Prinzip der transponierten Matrix auf unendlich-dimensionale Räume, so gelangt man zu der sogenannten adjungierten Abbildung (vgl. [Gau03]). Anstatt die Differentialoperatoren in $c$ zuerst zu diskretisieren, um dann die adjungierte Operation auszuführen, wird im kontinuierlichen Ansatz zuerst die adjungierte Operation ausgeführt, d.h. es wird die zur kontinuierlichen Zustandsgleichung adjungierte Differentialgleichung samt Randbedingungen betrachtet. Diese wird anschließend eigenständig diskretisiert und gelöst. Für einen Differentialoperator $P$ und die Lösung $\phi$ der Differentialgleichung $P\phi = b$ in einem Gebiet $\Omega$ mit homogenen Randbedingungen ist dabei der adjungierte Differentialoperator $P^*$ unter Verwendung eines geeigneten Skalarproduktes $(\cdot,\cdot)_\Omega$ definiert durch

$$(\psi, P\phi)_\Omega = (P^*\psi, \phi)_\Omega \quad \forall \psi, \phi. \tag{2.45}$$

Löst dann die adjungierte Variable $\psi$ die adjungierte Differentialgleichung $P^*\psi = h$ in $\Omega$ mit homogenen Randbedingungen, so gilt

$$(h,\phi)_\Omega = (P^*\psi, \phi)_\Omega = (\psi, P\phi)_\Omega = (\psi, b)_\Omega \tag{2.46}$$

(vgl. [Gau03]).

Für hinreichend glatte Lösungen sind die beiden Ansätze der diskreten und kontinuierlichen Adjungierten im Grenzwert einer Folge von immer feiner werdenden Gitterauflösungen konsistent und konvergieren zu den korrekten analytischen Sensitivitäten (siehe [GP00]). Aber auch für eine endliche Gitterfolge liefern nach [NJ00] und [GP08] beide Ansätze annähernd gleiche Genauigkeiten der Sensitivitäten. Ein Nachteil des diskreten Ansatzes ist allerdings die möglicherweise umständliche Herleitung der Adjungiertengleichung. Zur Lösung der primalen Zustandsgleichung können beispielsweise Pseudo-Zeitschrittverfahren (vergleiche auch Kapitel 3.3) und ggf. Diskretisierungsverfahren höherer Ordnung angewendet werden. Zu einem solchen Verfahren muss dann die transponierte Matrix gefunden werden, was im Allgemeinen eher kompliziert erscheint. Wenn diese Adjungiertengleichung allerdings gefunden ist, so kann sie mit demselben Diskretisierungsschema gelöst werden wie die primale Zustandsgleichung. Die Problematik der Herleitung der diskreten Adjungiertengleichung kann durch die Verwendung von *Automatischem Differenzieren* (AD) im Rückwärtsmodus umgangen werden. Dabei wird durch computergesteuerte Differentiation des gesamten Algorithmus zur Lösung der primalen Zustandsgleichung nahezu automatisch eine Iteration zur Lösung der Adjungiertengleichung erzeugt, welche damit konsistent mit der primalen Iteration ist. Umfassende Informationen zu AD findet man beispielsweise in [Gri00]. Außerdem wird in Kapitel 4.5 und 6.2.1 kurz auf die Verwendung von AD zur Erzeugung der hier benötigten Ableitungen eingegangen. Die kontinuierliche Adjungiertengleichung hingegen kann unabhängig vom primalen Lösungsverfahren hergeleitet und diskretisiert werden. Dies ist beispielsweise vorteilhaft, wenn der Optimierer keine Kenntnis vom Algorithmus des PDE-Lösers hat. In praktischen Tests wurde allerdings festgestellt, dass auf eine konsistente Wahl des Diskretisierungsschemas der Adjungiertengleichung zu achten ist: dies sollte möglichst nahe an dem der primalen Zustandsgleichung sein, um höhere Sensitivitätengenauigkeiten zu erzielen (vgl. [GP08]). Insbesondere bei der Berechnung turbulenter Strömungen und deren Adjungierten wurden in [CÖGT10] Inkonsistenzen bei Verwendung des kontinuierlichen Ansatzes aufgrund von Diskretisierungsfehlern und Modellannahmen festgestellt, welche durch die Verwendung von AD vermieden werden können.

# 3 One-Shot-Optimierung mittels approximativer, reduzierter SQP-Methode

In diesem Kapitel wird die One-Shot-Optimierung vorgestellt, wie sie von Schulz et al. z.B. in [HS04], [HSBG05], [HS06], [IKSG10] vorgeschlagen wird. Der erste Abschnitt wiederholt der Vollständigkeit wegen die Voraussetzungen aus Kapitel 2.2. Kapitel 3.2 dient der Herleitung des One-Shot-Verfahrens über ein reduziertes SQP-Verfahren. Das erhaltene approximative rSQP-Verfahren wird in Abschnitt 3.3 als präkonditioniertes Pseudo-Zeitschrittverfahren interpretiert, was die Bezeichnung der simultanen One-Shot-Optimierung deutlich werden lässt. Für linear-quadratische Optimierungsprobleme wird der Konvergenzbeweis nach [Ghe07] und [IKSG10] in Abschnitt 3.4 dargestellt. Abschnitt 3.5 befasst sich mit der Behandlung zusätzlicher Nebenbedingungen im One-Shot-Verfahren.

## 3.1 Problemstellung und Voraussetzungen

Analog zu Kapitel 2.2 wird das folgende Optimierungproblem mit getrennten Variablen betrachtet:

$$\begin{aligned} &\min_{y,u} \quad f(y,u) \\ &\text{s.t. } c(y,u) = 0. \end{aligned} \tag{3.1}$$

Dabei ist wieder $f\colon Y \times U \to \mathbb{R}$ die zu minimierende Zielfunktion und $c\colon Y \times U \to Y$ die zustandsbeschreibende, stationäre partielle Differentialgleichung. $f$ und $c$ seien zweifach stetig differenzierbar.

Für die Nebenbedingung $c$ wird weiterhin die Annahme 2.5 (Seite 6) getroffen, sodass die Ableitung $\frac{\partial c}{\partial y}$ stets invertierbar ist. Der Hauptsatz über implizite Funktionen (siehe Theorem 2.10) liefert wieder die Existenz einer Funktion so, dass $y$ als Funktion von $u$ betrachtet werden kann: $y = y(u)$.

## 3.2 Herleitung der Optimierungsstrategie

### 3.2.1 Minimierung des unbeschränkten reduzierten Problems

Mit den obigen Bezeichnungen kann das Optimierungsproblem (3.1) durch Reduzierung der Nebenbedingung nach Kapitel 2.5 zu einem äquivalenten Problem umgeschrieben werden:

$$\min_u \quad \tilde{f}(u) \quad \text{mit} \quad \tilde{f}(u) := f(y(u),u). \tag{3.2}$$

Man erhält somit ein unbeschränktes Minimierungsproblem, dessen Optimierungsvariablen sich auf die Designvariable $u$ reduziert haben. Für ein gegebenes $\hat{u}$ ist eine zulässige Zustandvariable $\hat{y}$ dann durch die Bedingung $c(\hat{y},\hat{u}) \stackrel{!}{=} 0$ eindeutig definiert.

Die notwendige Bedingung für einen Minimalpunkt $u^*$ des reduzierten Problems (3.2) ist durch

$$\nabla_u \tilde{f}(u^*) = 0 \tag{3.3}$$

gegeben. Um einen solchen stationären Punkt zu erreichen wird, aufgrund der i.A. hohen Nichtlinearität der Zielfunktion $\tilde{f}$ ein (Quasi-) Newton-Verfahren (siehe z.B. [NW06]) angewendet:

$$u_{k+1} = u_k + \tau_k \Delta u_k, \tag{3.4}$$

wobei $\Delta u_k$ aus

$$\nabla_u^2 \tilde{f}(u_k)\Delta u_k = -\nabla_u \tilde{f}(u_k) \tag{3.5}$$

berechnet wird und die Schrittlänge $\tau_k \in (0,1]$ geeignet gewählt wird. Für ein Quasi-Newton-Verfahren ersetzt man die Hessematrix der reduzierten Funktion durch eine Approximation $B_k \approx \nabla_u^2 \tilde{f}(u_k)$.

**Theorem 3.1.** *Mit den Definitionen*

$$T(y,u) := \begin{bmatrix} -\left(\frac{\partial c}{\partial y}(y,u)\right)^{-1} \frac{\partial c}{\partial u}(y,u) \\ I \end{bmatrix}, \qquad R(y,u) := \begin{bmatrix} \left(\frac{\partial c}{\partial y}(y,u)\right)^{-1} \\ 0 \end{bmatrix}, \tag{3.6}$$

$$\textit{und} \quad \bar{y} := -R(y,u)^T \begin{bmatrix} \nabla_y f(y,u) \\ \nabla_u f(y,u) \end{bmatrix} = -\left(\frac{\partial c}{\partial y}(y,u)\right)^{-T} \nabla_y f(y,u) \tag{3.7}$$

*ist die erste und zweite Ableitung der Funktion $\tilde{f}(u) := f(y,u)$ gegeben durch*

$$\nabla_u \tilde{f}(u) = T(y,u)^T \begin{bmatrix} \nabla_y f(y,u) \\ \nabla_u f(y,u) \end{bmatrix} = \nabla_u f(y,u) + \left(\frac{\partial c}{\partial u}(y,u)\right)^T \bar{y} \tag{3.8}$$

$$\nabla_u^2 \tilde{f}(u) = T(y,u)^T \underbrace{\begin{bmatrix} \frac{\partial^2 (f(y,u)+c(y,u)^T\bar{y})}{\partial y \partial y} & \frac{\partial^2 (f(y,u)+c(y,u)^T\bar{y})}{\partial y \partial u} \\ \frac{\partial^2 (f(y,u)+c(y,u)^T\bar{y})}{\partial u \partial y} & \frac{\partial^2 (f(y,u)+c(y,u)^T\bar{y})}{\partial u \partial u} \end{bmatrix}}_{=:H(y,u)} T(y,u) \tag{3.9}$$

*für ein y, welches $c(y,u) = 0$ erfüllt.*

*Beweis.* Der Beweis ist in [Sch96], Lemma 3.1, S. 24f., zu finden und basiert auf dem Hauptsatz über implizite Funktionen, welcher

$$\frac{\partial y}{\partial u} = -\left(\frac{\partial c}{\partial y}\right)^{-1} \frac{\partial c}{\partial u} \tag{3.10}$$

für $c(y(u),u) = 0$ liefert. Die Behauptung folgt dann mittels Kettenregel und elementaren Umformungen wie in Kapitel 2.5. □

*Bemerkung.* $T$ aus (3.6) spannt den Kern von $\left[c_y, c_u\right]$ auf und stellt somit den Tangentialraum der Nebenbedingung $c(y,u) = 0$ dar. $R$ beschreibt die Rechtsinverse, d.h. $\left[c_y, c_u\right] R(y,u) = I$.

Nach Definition (3.7) entspricht $\bar{y}$ genau der Definition der adjungierten Variable aus Kapitel 2.5. Demnach ist $\nabla_u \tilde{f} = \nabla_u f + c_u^T \bar{y}$ gerade der *reduzierte Gradient* der Zielfunktion $f$ (vgl. Gleichung (2.34)). $\nabla_u^2 \tilde{f} = T^T H T$ wird als *reduzierte Hessematrix* bezeichnet, sie ist an einem Minimum nach den hinreichenden Bedingungen 2. Ordnung für das unbeschränkte Problem positiv definit, damit hat (3.5) in der Nähe des Minimums stets eine eindeutige Lösung.

Die durch (3.4) - (3.5) definierte Iteration wird in der Literatur aufgrund der impliziten Behandlung der Nebenbedingung auch als „Nested Analysis and Design (NAND)" oder „Black-Box-Methode" bezeichnet (siehe z.B. [Haz10], Kapitel 3.2.1).

### 3.2.2 Reduzierte SQP-Methode

Ein großer Nachteil der Iteration (3.4) - (3.5) ist, dass nach jedem Schritt in der Designvariable eine neue zulässige Zustandsvariable $y$ berechnet werden muss, die $c(y, u_{k+1}) = 0$ erfüllt. Da $c$ i. A. nichtlineare Modellgleichungen (beispielsweise die strömungsbeschreibenden Navier-Stokes-Gleichungen) beschreibt, ist hierfür meist ein eigener iterativer Prozess notwendig. Dies bedeutet numerisch einen sehr großen Aufwand. Ersetzt man allerdings diesen Prozess durch nur *einen* Newton-Schritt dieser Iteration zur Lösung von $c(y, u_{k+1}) = 0$ und definiert die zugehörigen Ausdrücke für $T$,$R$, $\bar{y}$,$\nabla_u \tilde{f}(u)$ und $\nabla_u^2 \tilde{f}(u)$ weiterhin nach Theorem 3.1 (obwohl i. A. $c(y_{k+1}, u_{k+1}) \neq 0$ und Gleichheit nur an der Lösung vorliegt), so gelangt man zum Konzept eines reduzierten Sequential-Quadratic-Programming-Verfahrens (rSQP), wie es z.B. in [Sch96] vorgestellt wird (siehe [Sch96], Algorithmus 3.3, S. 27). Dort findet man auch Ansätze zur Interpretation der auftretenden Ausdrücke aus Theorem 3.1.

Die Nebenbedingung wird in Hinblick auf den oben erwähnten Newton-Schritt mittels Taylorreihenentwicklung (siehe z.B. [NW06], Theorem 2.1, S.14) bis zu Termen erster Ordnung linearisiert:

$$c(y_{k+1}, u_{k+1}) = c(y_k + \Delta y_k, u_k + \Delta u_k) \overset{Taylor}{\approx} c(y_k, u_k) + c_y(y_k, u_k)\Delta y_k + c_u(y_k, u_k)\Delta u_k. \tag{3.11}$$

Somit liegen alle Schritte $(\Delta y, \Delta u)$ im Tangentialraum der Nebenbedingung an der aktuellen Iterierten $(y,u)$.

Die Iteration (3.4) - (3.5) zusammen mit der Approximation (3.11) und Theorem 3.1 baut sich dann aus 3 Schritten auf: Für eine aktuelle Iterierte $(y_k, u_k)$ berechne man die neue adjungierte

Variable aus (3.7). Damit erhält man aus Theorem 3.1 die Approximationen für $\nabla_u \tilde{f}(u_k)$ und $\nabla_u^2 \tilde{f}(u_k)$ und berechnet den Schritt der Designvariable $\Delta u_k$ aus (3.5). Anschließend erhält man aus (3.11) den Schritt der Zustandsvariable $\Delta y_k$. Dies entspricht der reduzierten SQP-Methode aus [Sch96] der folgenden Art:

$$\begin{pmatrix} y_{k+1} \\ u_{k+1} \end{pmatrix} = \begin{pmatrix} y_k \\ u_k \end{pmatrix} + \tau_k \begin{pmatrix} \Delta y_k \\ \Delta u_k \end{pmatrix}, \tag{3.12}$$

wobei $(\Delta y_k, \Delta u_k)$ aus dem Gleichungssystem

$$c_y(y_k,u_k)^T \bar{y}_{k+1} = -\nabla_y f(y_k,u_k) \tag{3.13}$$

$$T(y_k,u_k)^T H(y_k,u_k) T(y_k,u_k) \Delta u_k + c_u(y_k,u_k)^T \bar{y}_{k+1} = -\nabla_u f(y_k,u_k) \tag{3.14}$$

$$c_y(y_k,u_k)\Delta y_k + c_u(y_k,u_k)\Delta u_k = -c(y_k,u_k) \tag{3.15}$$

oder in Matrixschreibweise

$$\begin{bmatrix} 0 & 0 & c_y(y_k,u_k)^T \\ 0 & T(y_k,u_k)^T H(y_k,u_k) T(y_k,u_k) & c_u(y_k,u_k)^T \\ c_y(y_k,u_k) & c_u(y_k,u_k) & 0 \end{bmatrix} \begin{pmatrix} \Delta y_k \\ \Delta u_k \\ \bar{y}_{k+1} \end{pmatrix} = \begin{pmatrix} -\nabla_y f(y_k,u_k) \\ -\nabla_u f(y_k,u_k) \\ -c(y_k,u_k) \end{pmatrix} \tag{3.16}$$

mit zusätzlicher Variable $\bar{y}_{k+1}$ errechnet werden.

Zum Zweck eines Vergleich mit der vollen SQP-Methode sei die zu Problem (3.1) assoziierte Lagrangefunktion (vgl. Kapitel 2.3) definiert als

$$\mathscr{L}(y,\bar{y},u) \coloneqq f(y,u) + \bar{y}^T c(y,u). \tag{3.17}$$

Dann entspricht die Matrix $H(y,u)$ aus Theorem 3.1 genau der Hessematrix der Lagrangefunktion. Mit $\bar{y}_{k+1} = \bar{y}_k + \tau_k \Delta \bar{y}_k$ ist die obige Iteration äquivalent zu

$$\begin{pmatrix} y_{k+1} \\ u_{k+1} \\ \bar{y}_{k+1} \end{pmatrix} = \begin{pmatrix} y_k \\ u_k \\ \bar{y}_k \end{pmatrix} + \tau_k \begin{pmatrix} \Delta y_k \\ \Delta u_k \\ \Delta \bar{y}_k \end{pmatrix} \tag{3.18}$$

mit

$$\begin{bmatrix} 0 & 0 & c_y(y_k,u_k)^T \\ 0 & T(y_k,u_k)^T H(y_k,u_k) T(y_k,u_k) & c_u(y_k,u_k)^T \\ c_y(y_k,u_k) & c_u(y_k,u_k) & 0 \end{bmatrix} \begin{pmatrix} \Delta y_k \\ \Delta u_k \\ \Delta \bar{y}_k \end{pmatrix} = \begin{pmatrix} -\nabla_y \mathscr{L}(y_k,\bar{y}_k,u_k) \\ -\nabla_u \mathscr{L}(y_k,\bar{y}_k,u_k) \\ -\nabla_{\bar{y}} \mathscr{L}(y_k,\bar{y}_k,u_k) \end{pmatrix}. \tag{3.19}$$

Einen vollen SQP-Schritt hingegen erhält man, indem man auf die KKT-Bedingung $\nabla \mathscr{L} = 0$ (siehe Theorem 2.7) ein Newtonverfahren anwendet. Die Schritte $(\Delta y, \Delta u, \Delta \bar{y})$ ergeben sich dann aus dem Gleichungssystem

$$\begin{bmatrix} H_{yy} & H_{yu} & c_y^T \\ H_{uy} & H_{uu} & c_u^T \\ c_y & c_u & 0 \end{bmatrix} \begin{pmatrix} \Delta y \\ \Delta u \\ \Delta \bar{y} \end{pmatrix} = \begin{pmatrix} -\nabla_y \mathscr{L} \\ -\nabla_u \mathscr{L} \\ -\nabla_{\bar{y}} \mathscr{L} \end{pmatrix}, \tag{3.20}$$

wobei wieder alle Ausdrücke mit dem Index $k$ ausgewertet werden, was hier zur übersichtlicheren Darstellung weggelassen wurde.

Hier sieht man, dass ein rSQP-Verfahren einem SQP-Verfahren mit besonderer Hessematrixnäherung entspricht. Statt der vollen Hessematrix der Lagrangefunktion $H = \begin{bmatrix} H_{yy} & H_{yu} \\ H_{uy} & H_{uu} \end{bmatrix}$ wird für das rSQP-Verfahren „nur" die auf den Kern der linearisierten Nebenbedingung projizierte Hessematrix $T^T HT$ benötigt. Des Weiteren sei erwähnt, dass $T^T HT$ dem Schurkomplement der KKT-Matrix aus (3.20) bezüglich $y$ und $\bar{y}$ entspricht.

#### Approximation der reduzierten Hessematrix und Konvergenzeigenschaften des rSQP-Verfahrens

Für hochdimensionale Optimierungsprobleme kommt eine exakte Berechnung der reduzierten Hessematrix $T^T HT$ nicht in Betracht. Statt dessen muss numerisch eine Approximation

$$B_k \approx T^T HT = H_{uu} - H_{uy} c_y^{-1} c_u - c_u^T c_y^{-T} H_{yu} + c_u^T c_y^{-T} H_{yy} c_y^{-1} c_u \tag{3.21}$$

herangezogen werden. Alle auftretenden Ausdrücke werden am Iterationsindex $k$ ausgewertet, sodass $B_k$ in jeder Iteration variiert. Es eignen sich Update-Formeln aus der konvexen Broydenklasse (siehe [NW06]), bei denen ein symmetrisches, positiv definites $B_{k+1}$ aus einem positiv definiten $B_k$ gewonnen wird, beispielsweise ein BFGS-Update. Die Sekantenbedingung ist gegeben durch

$$B_{k+1} \Delta u_k = R_k. \tag{3.22}$$

Für $R_k$ sollen dabei nur tangentiale Informationen verwendet werden. Für Details bezüglich der konkreten Wahl der $R_k$ in einem rSQP-Verfahren sei auf [Sch96] (Kapitel 3.3.2, S.35f.) verwiesen. In jedem Schritt wird die Krümmungseigenschaft („curvature condition", siehe [NW06])

$$\Delta u_k^T R_k > 0 \tag{3.23}$$

getestet; bei Nichterfülltheit wird das Update durch Setzen von $B_{k+1} = B_k$ oder $B_{k+1} = I$ übergangen.

Die Güte der Approximation beeinflusst die Konvergenzgeschwindigkeit des rSQP-Verfahrens. Unter Verwendung der exakten reduzierten Hessematrix erzielt das rSQP-Verfahren unter milden Voraussetzungen 2-Schritt-quadratische Konvergenz in der Nähe der Lösung. Werden die reduzierten Hessematrizen hingegen durch BFGS-Update-Formeln approximiert, so kann lokal 2-Schritt-superlineare Konvergenz gezeigt werden (vgl. [Sch96], [Sch98]). Für ein volles SQP-Verfahren erhält man (vgl. [NW06]) lokal 1-Schritt-quadratische (exakte reduzierte Hessematrix) bzw. 1-Schritt-superlineare Konvergenz (approximierte reduzierte Hessematrix).

### 3.2.3 One-Shot-Verfahren als approximative rSQP-Methode

Für ein rSQP-Verfahren der Form (3.18) - (3.19) ist die Berechnung und Invertierung von $\frac{\partial c}{\partial y}$ der linearisierten Nebenbedingung notwendig. Häufig ist zur Lösung der Nebenbedingung schon ein

iterativer Algorithmus vorhanden, der zu einem festen $u$ eine zulässige Zustandsvariable liefert, etwa in der Form

$$y_{k+1} = y_k - \tau A_f^{-1} c(y_k, u), \tag{3.24}$$

wobei $A_f \approx c_y$. Um dies in das rSQP-Verfahren integrieren zu können und die Berechnung von $c_y^{-1}$ zu vermeiden, wird in einem approximativen rSQP-Verfahren nicht nur $B_k$ approximiert, sondern auch $c_y$ und $c_y^T$ der Matrix aus (3.19) durch nichtsinguläre, einfach invertierbare Matrizen $A_f \approx c_y$ und $A_a \approx c_y^T$ ersetzt (siehe [Sch98], [Ghe07], [IKSG10]).

Ein Schritt der approximativen rSQP-Methode berechnet sich dann aus dem System

$$\begin{matrix}(I)\\(II)\\(III)\end{matrix} \quad \begin{bmatrix} 0 & 0 & A_a(y_k,u_k) \\ 0 & B_k & c_u(y_k,u_k)^T \\ A_f(y_k,u_k) & c_u(y_k,u_k) & 0 \end{bmatrix} \begin{pmatrix} \Delta y_k \\ \Delta u_k \\ \Delta \bar{y}_k \end{pmatrix} = \begin{pmatrix} -\nabla_y \mathscr{L}(y_k,\bar{y}_k,u_k) \\ -\nabla_u \mathscr{L}(y_k,\bar{y}_k,u_k) \\ -\nabla_{\bar{y}} \mathscr{L}(y_k,\bar{y}_k,u_k) \end{pmatrix}. \tag{3.25}$$

Diese Iteration entspricht der One-Shot-Methode, wie sie von Schulz et al. vorgeschlagen wird ([HS04], [HSBG05], [Ghe07], [IKSG10]). Die Zustands-, Adjungierten-, sowie die Designvariable werden simultan iteriert. Man vergleiche dazu auch die äquivalente Pseudozeitschritt-Formulierung aus Kapitel 3.3, bei der für jedes Update der Designvariable genau ein Zeitschritt ausgeführt wird.

Die Indizes $f$ (für *forward*) und $a$ (für *adjoint*) der approximierenden Matrizen $A_f$ und $A_a$ deuten auf deren Verwendung im so erhaltenen One-Shot-Algorithmus hin. Dazu betrachte man die Iteration, die sich aus (3.25) berechnet (vgl. [Ghe07], Kapitel 5.1).:

1. Aus $(I)$ erhält man das Update der adjungierten Variable:

$$\bar{y}_{k+1} = \bar{y}_k - \tau_k (A_a(y_k,u_k))^{-1} (\nabla_y f(y_k,u_k) + c_y(y_k,u_k)^T \bar{y}_k). \tag{3.26}$$

   Dies entspricht einem Schritt einer Defektkorrektur-Iteration zur Lösung der Adjungiertengleichung $c_y(y_k,u_k)^T \bar{y} = -\nabla_y f(y_k,u_k)$ (siehe Kapitel 2.5).

2. Gleichung $(II)$ beschreibt das Update der Designvariable. Dieses lässt sich wegen $\bar{y}_{k+1} = \bar{y}_k + \tau_k \Delta \bar{y}_k$ zusammenfassen zu

$$u_{k+1} = u_k - \tau_k B_k^{-1} \underbrace{(\nabla_u f(y_k,u_k) + c_u(y_k,u_k)^T \bar{y}_{k+1})}_{=:\gamma_k}. \tag{3.27}$$

   Erfüllt $\bar{y}_{k+1}$ die Adjungiertengleichung $c_y(y_k,u_k)^T \bar{y}_{k+1} = -\nabla_y f(y_k,u_k)$, so ist $\gamma_k$ gerade der reduzierte Gradient der Zielfunktion an den aktuellen Iterierten $(y_k,u_k)$. Nach $(I)$ kann $\bar{y}_{k+1}$ als approximative Lösung dieser Gleichung gesehen werden, sodass $\gamma_k$ als approximativer reduzierter Gradient mit Informationen der aktuellsten Variablen auftritt.

3. Für das Update der Zustandvariable liefert Gleichung $(III)$

$$y_{k+1} = y_k - \tau_k (A_f(y_k,u_k))^{-1} (c(y_k,u_k) + c_u(y_k,u_k)\Delta u_k). \tag{3.28}$$

Im Hinblick auf die anfängliche Linearisierung der Nebenbedingung schreibt sich der eingeklammerte Term mittels Taylorreihenentwicklung in der zweiten Variable als $c(y_k, u_{k+1})$. Damit kann (3.28) als ein Schritt zur Lösung der Zustandgleichung an der neuen Designvariable interpretiert werden, sodass eine Iteration des Algorithmus zur Lösung der Zustandsgleichung nach Gleichung (3.24) angewendet werden kann für die aktuelle Designvariable $u_{k+1}$.

In numerischen Anwendungen ist häufig neben dem iterativen Algorithmus zur Berechnung einer primal zulässigen Zustandvariable auch ein adjungierter Algorithmus vorhanden, der die zugehörige adjungierte Differentialgleichung iterativ löst. Im Hinblick darauf lässt sich das Update der Adjungiertenvariable aus Gleichung (3.26) als einen Schritt dieses Algorithmus interpretieren. Der One-Shot-Algorithmus zur Lösung des Optimierungsproblems (3.1) setzt sich dann für gegebene Startwerte $(y_0, u_0, \bar{y}_0)$ aus den folgenden Schritten zusammen:

1. Führe einen Schritt der adjungierten Iteration durch und erhalte die neue adjungierte Variable $\bar{y}_{k+1}$.
2. Berechne den approximativen, reduzierten Gradienten $\gamma_k$ aus (3.27).
3. Approximiere $B_k$.
4. Berechne die neue Designvariable mit $u_{k+1} = u_k - \tau_k B_k^{-1}\gamma_k$, $\tau_k \in (0,1]$.
5. Führe einen Schritt der primalen Iteration durch und erhalte $y_{k+1}$.

### Approximation von $B$

In konkreten numerischen Testproblemen, beispielsweise in [GS05], konnte beobachtet werden, dass im Falle von $A_f \neq c_y$ und $A_a \neq c_y^T$ eine Approximation von $B$, welche konsistent zur Wahl von $A_f$ und $A_a$ ist, zu besseren Konvergenzeigenschaften führt. Nach [IKSG10] soll dementsprechend statt

$$B \approx T^T H T = H_{uu} - H_{uy}c_y^{-1}c_u - c_u^T c_y^{-T} H_{yu} + c_u^T c_y^{-T} H_{yy} c_y^{-1} c_u \tag{3.29}$$

die Approximation

$$B \approx S_A := H_{uu} - H_{uy}A_f^{-1}c_u - c_u^T A_a^{-1} H_{yu} + c_u^T A_a^{-1} H_{yy} A_f^{-1} c_u \tag{3.30}$$

vorgezogen werden. $S_A$ wird als *konsistente* reduzierte Hessematrix bezeichnet. Die obige approximative rSQP-Iteration, welche dann die Approximationen

$$A_f \approx c_y, \quad A_a \approx c_y^T, \quad B \approx S_A \tag{3.31}$$

enthält, wird auch als approximative Nullraumiteration („approximate nullspace iteration", siehe [IKSG10]) bezeichnet, da der durch $T$ aufgespannte Kern der linearisierten Nebenbedingung nur approximativ dargestellt wird durch $\tilde{T}^T = \begin{bmatrix} -c_u^T A_a^{-1}, & I \end{bmatrix}$ und $\tilde{T} = \begin{bmatrix} -A_f^{-1}c_u \\ I \end{bmatrix}$.

Die Konvergenz der obigen Iteration unter Verwendung der Approximationen (3.31) ist für linear-quadratische Optimierungsprobleme (d.h. Minimierungsprobleme mit quadratischer Zielfunktion und linearen Nebenbedingungen) in [IKSG10] gezeigt. In Kapitel 3.4 wird darauf noch genauer eingegangen.

Häufig werden zur Berechnung eines $B$, welches $S_A$ approximiert, BFGS-Updates analog zu Kapitel 3.2.2 vorgezogen. Die Sekantenbedingung ist gegeben durch

$$B_{k+1}\Delta u_k = R_k. \tag{3.32}$$

In numerischen Umsetzungen der One-Shot-Iteration wird dabei häufig $R_k = \gamma_{k+1} - \gamma_k$ gewählt (siehe z.B. [HSBG05]). Es werden also BFGS-Updates anhand der verfügbaren Information der approximativen reduzierten Gradienten durchgeführt. Dadurch ist keine zusätzlich Auswertung der Größen notwendig.

## 3.3 Interpretation als Pseudozeitschrittverfahren

Zur iterativen Lösung der stationären partiellen Differentialgleichung $c(y,u) = 0$ werden häufig Pseudozeitschrittverfahren angewendet. Dabei wird die stationäre, also zeitlich invariante Lösung der Gleichung, als stationärer Zustand eines dynamischen Systems nach seiner transienten Phase betrachtet: Nach Hinzufügen einer Pseudozeit wird das instationäre System über der Zeit integriert, bis ein stationärer Zustand erreicht wird. Um eine Einbettung eines solchen Pseudozeitschrittverfahrens zur Lösung der Nebenbedingung in den Optimierungsprozess in konsistenter Weise zu garantieren, wird in diesem Kapitel eine Formulierung des beschriebenen One-Shot-Algorithmus im Sinne eines präkonditionierten Pseudozeitschrittverfahrens vorgestellt. Man vergleiche zu den folgenden Ausführungen [HS04] und [HS06].

### Pseudozeitschrittverfahren

Für eine Abbildung $F\colon X \to Y$ soll die Nullstelle

$$F(x) = 0 \tag{3.33}$$

bestimmt werden. Durch Hinzufügen einer Pseudozeit $t$ wird $x$ als abhängig von $t$ betrachtet und die Gleichung (3.33) in eine gewöhnliche Differentialgleichung transformiert:

$$\frac{d}{dt}x(t) = -F(x(t)) \tag{3.34}$$

mit einem Anfangswert $x(0) = x_0$. Diese erreicht einen stationären Punkt $x_\infty$, falls $-F$ abklingend ist, d.h. falls alle Realteile der Eigenwerte der negativen Jakobimatrix von $F$ positiv sind. Dann gilt für diesen Punkt, dass

$$0 = \left.\frac{d}{dt}\right|_\infty x(t) = F(x_\infty), \tag{3.35}$$

sodass $x_\infty$ Gleichung (3.33) erfüllt und somit Nullstelle von $F$ ist. Um diesen Punkt zu erreichen, wird ein explizites (Pseudo-) Zeitschrittverfahren angewendet. Im Falle des expliziten Eulerverfahren erhält man

$$x_{k+1} = x_k - \Delta t F(x_k). \tag{3.36}$$

Für lineare Operatoren $F$ entspricht dies genau einem Schritt eines Richardsonverfahrens zur Lösung von $Fx = 0$. Hier erkennt man, dass (Pseudo-) Zeitschrittverfahren häufig mit iterativen Methoden zur Lösung linearer Gleichungssysteme identifiziert werden können. Um die Konvergenzgeschwindigkeit des Verfahrens zu erhöhen, kann eine Präkonditionierungsmatrix $P$ hinzugefügt werden, welche die Eigenwerte der Jakobimatrix beeinflusst:

$$x_{k+1} = x_k - \Delta t P F(x_k) \tag{3.37}$$

Dies entspricht der gewöhnlichen Differentialgleichung

$$\dot{x} = PF(x) \tag{3.38}$$

mit dem selben stationären Punkt $x_\infty$.

### Anwendung auf das Optimierungsproblem

Für das Optimierungsproblem (3.1)

$$\begin{aligned} \min_{y,u} \quad & f(y,u) \\ \text{s.t. } & c(y,u) = 0 \end{aligned}$$

sei die Lagrangefunktion wieder definiert durch

$$\mathscr{L}(y,\bar{y},u) \coloneqq f(y,u) + \bar{y}^T c(y,u) \tag{3.39}$$

Die notwendigen Bedingungen erster Ordnung (siehe Theorem 2.7) sind durch

$$\nabla_y \mathscr{L}(y,\bar{y},u) = 0 \qquad \text{(Adjungiertengleichung)} \tag{3.40}$$
$$\nabla_u \mathscr{L}(y,\bar{y},u) = 0 \qquad \text{(Designgleichung)} \tag{3.41}$$
$$c(y,u) = 0 \qquad \text{(Zustandsgleichung)} \tag{3.42}$$

gegeben. Dies entspricht einem System von nichtlinearen Gleichungen, welches zusammengefasst geschrieben werden kann als

$$F(y,\bar{y},u) = \begin{pmatrix} \nabla_y \mathscr{L}(y,\bar{y},u) \\ \nabla_u \mathscr{L}(y,\bar{y},u) \\ c(y,u) \end{pmatrix} = 0. \tag{3.43}$$

Analog zu (3.34) wird dies in ein System von gewöhnlichen Differentialgleichungen transformiert, für das ein stationärer Zustand gesucht wird:

$$\begin{aligned} \tfrac{d\bar{y}}{dt} + \nabla_y \mathscr{L}(y,\bar{y},u) &= 0 \\ \tfrac{du}{dt} + \nabla_u \mathscr{L}(y,\bar{y},u) &= 0 \\ \tfrac{dy}{dt} + c(y,u) &= 0 \end{aligned} \qquad \Longleftrightarrow \qquad \begin{pmatrix} \dot{y} \\ \dot{u} \\ \dot{\bar{y}} \end{pmatrix} = \begin{pmatrix} -\nabla_y \mathscr{L} \\ -\nabla_u \mathscr{L} \\ -c \end{pmatrix}. \tag{3.44}$$

Für das gesamte System kann ein Zeitschrittverfahren angewendet werden, um die mit Pseudozeit erweiterte Zustands-, Adjungierten- und Designgleichung simultan zu lösen, was der in [HS04] eingeführte Begriff des „simultaneous pseudo-timestepping" wiederspiegeln soll. Da dieses System von gewöhnlichen Differentialgleichungen i.A. steif ist und nicht zwingend nur abklingende Komponenten enthält (siehe z.B. [GS05]), muss ein Präkonditionierer gewählt werden, der die Konvergenz eines Zeitschrittverfahrens sichert und beschleunigt.

Betrachtet man vor diesem Hintergrund einen Schritt des in Kapitel 3.2.3 vorgestellten approximativen rSQP-Verfahrens (Gleichung (3.25))

$$\begin{pmatrix} y_{k+1} \\ u_{k+1} \\ \bar{y}_{k+1} \end{pmatrix} = \begin{pmatrix} y_k \\ u_k \\ \bar{y}_k \end{pmatrix} + \tau \begin{pmatrix} \Delta y_k \\ \Delta u_k \\ \Delta \bar{y}_k \end{pmatrix} \quad \text{mit} \quad \begin{bmatrix} 0 & 0 & A_a \\ 0 & B & c_u^T \\ A_f & c_u & 0 \end{bmatrix} \begin{pmatrix} \Delta y \\ \Delta u \\ \Delta \bar{y} \end{pmatrix} = \begin{pmatrix} -\nabla_y \mathscr{L} \\ -\nabla_u \mathscr{L} \\ -c \end{pmatrix} \tag{3.45}$$

so erkennt man, dass dies gerade dem Zeitschritt eines expliziten Eulerverfahren zur Lösung des präkonditionierten Systems

$$\begin{pmatrix} \dot{y} \\ \dot{u} \\ \dot{\bar{y}} \end{pmatrix} = \underbrace{\begin{bmatrix} 0 & 0 & A_a \\ 0 & B & c_u^T \\ A_f & c_u & 0 \end{bmatrix}^{-1}}_{=:P} \begin{pmatrix} -\nabla_y \mathscr{L} \\ -\nabla_u \mathscr{L} \\ -c \end{pmatrix} \tag{3.46}$$

entspricht.

Durch Interpretation der Variablenupdates des approximativen rSQP-Verfahren als Zeitableitungen erhält man also das präkonditionierte Pseudozeitsystem von gewöhnlichen Differentialgleichungen. Für die Präkonditionierungsmatrix $P$ gilt nach [HS06]:

$$P = \begin{bmatrix} 0 & 0 & A_a \\ 0 & B & c_u^T \\ A_f & c_u & 0 \end{bmatrix}^{-1} = \begin{bmatrix} A_f^{-1} c_u B^{-1} c_u^T A_a^{-1} & -A_f^{-1} c_u B^{-1} & A_f^{-1} \\ -B^{-1} c_u^T A_a^{-1} & B^{-1} & 0 \\ A_a^{-1} & 0 & 0 \end{bmatrix}. \tag{3.47}$$

Ein Vorteil dieser Sichtweise der Methode ist, dass keine zusätzliche Globalisierungstechnik benötigt wird. Die drei Pseudozeitdifferentialgleichungen können mit dem gleichen Zeitschritt gelöst werden. Des Weiteren wird durch die Präkonditionierung des Systems die Konvergenzgeschwindigkeit beschleunigt.

## 3.4 Konvergenzeigenschaften

Die Konvergenz des One-Shot-Verfahrens wird in [Ghe07] und [IKSG10] für linear-quadratische Minimierungsprobleme bewiesen und eine obere Schranke der Konvergenzrate abhängig von der Güte der Approximationen kann angegeben werden. Der Konvergenzbeweis soll hier zusammengefasst dargestellt werden.

Betrachtet wird ein quadratisches Minimierungsproblem mit linearen Nebenbedingungen

$$\begin{aligned} \min_{y,u} \quad & \frac{1}{2}(y^T H_y y + y^T H_{yu} u + u^T H_{uy} y + u^T H_u u) + f_y^T y + f_u^T u \\ \text{s.t} \quad & C_y y + C_u u + c = 0 \end{aligned} \tag{3.48}$$

mit den Optimierungsvariablen $y \in \mathbb{R}^{n_y}, u \in \mathbb{R}^{n_u}$, sowie den konstanten Vektoren $c, f_y \in \mathbb{R}^{n_y}, f_u \in \mathbb{R}^{n_u}$ und den Matrizen $H_u \in \mathbb{R}^{n_u \times n_u}, H_y \in \mathbb{R}^{n_y \times n_y}, H_{uy}^T = H_{yu} \in \mathbb{R}^{n_y \times n_u}, C_u \in \mathbb{R}^{n_y \times n_u}, C_y \in \mathbb{R}^{n_y \times n_y}$. Nach Annahme 2.5 soll $C_y$ invertierbar sein. Die tiefstehenden Indizes stellen hier keinesfalls partielle Ableitungen dar, sondern sollen lediglich auf deren Position in Zielfunktion und Nebenbedingung hinweisen.

Die notwendigen Bedingungen für einen Minimalpunkt von (3.48) sind durch das folgende Gleichungsystem gegeben:

$$\begin{bmatrix} H_y & H_{yu} & C_y^T \\ H_{uy} & H_u & C_u^T \\ C_y & C_u & 0 \end{bmatrix} \begin{pmatrix} y \\ u \\ \bar{y} \end{pmatrix} = - \begin{pmatrix} f_y \\ f_u \\ c \end{pmatrix}. \tag{3.49}$$

Die Anwendung des One-Shot-Verfahrens nach Kapitel 3.2.3 liefert die Iteration

$$(y_{k+1}, u_{k+1}, \bar{y}_{k+1}) = (y_k, u_k, \bar{y}_k) + (\Delta y_k, \Delta u_k, \Delta \bar{y}_k), \tag{3.50}$$

wobei nach Gleichung (3.25) $(\Delta y_k, \Delta u_k, \Delta \bar{y}_k)$ Lösung von

$$\underbrace{\begin{bmatrix} 0 & 0 & A_a \\ 0 & B & C_u^T \\ A_f & C_u & 0 \end{bmatrix}}_{=:R} \begin{pmatrix} \Delta y_k \\ \Delta u_k \\ \Delta \bar{y}_k \end{pmatrix} = - \underbrace{\begin{bmatrix} H_y & H_{yu} & C_y^T \\ H_{uy} & H_u & C_u^T \\ C_y & C_u & 0 \end{bmatrix}}_{=:K} \begin{pmatrix} y_k \\ u_k \\ \bar{y}_k \end{pmatrix} - \begin{pmatrix} f_y \\ f_u \\ c \end{pmatrix} \tag{3.51}$$

ist, mit den Approximationen $A_f \approx C_y$, $A_a \approx C_y^T$, sowie die Approximation der konsistenten reduzierten Hessematrix

$$B \approx S_A = H_u - H_{uy} A_f^{-1} C_u - C_u^T A_a^{-1} H_{yu} + C_u^T A_a^{-1} H_y A_f^{-1} C_u. \tag{3.52}$$

$S_A$ ist das Schurkomplement der KKT-Matrix $K$ nach Elimination von $y$ und $\bar{y}$ unter Verwendung der Matrizen $A_f$ und $A_a$. Die zugehörige Iterationsmatrix ist gegeben durch

$$I - R^{-1}K = \underbrace{I - R^{-1}\tilde{K}}_{=:M} + \underbrace{R^{-1}(\tilde{K} - K)}_{=:N}, \tag{3.53}$$

wobei die eingefügte Matrix $\tilde{K}$ gewählt wird als

$$\tilde{K} := \begin{bmatrix} H_y & H_{yu} & A_a \\ H_{uy} & H_u - S_A + B & C_u^T \\ A_f & C_u & 0 \end{bmatrix}. \tag{3.54}$$

Die Konvergenz ist gesichert, falls $\rho(I - R^{-1}K) = \rho(M + N) < 1$, wobei $\rho$ den Spektralradius der Matrix darstellt.

**Lemma 3.2.** *Die in (3.53) definierte Matrix M ist nilpotent zum Index* 3*, d.h.*

$$M^3 = (I - R^{-1}\tilde{K})^3 = 0. \tag{3.55}$$

*Beweis.* Nach Definition von $R$ gilt

$$R^{-1} = \begin{bmatrix} 0 & 0 & A_a \\ 0 & B & C_u^T \\ A_f & C_u & 0 \end{bmatrix}^{-1} = \begin{bmatrix} A_f^{-1}C_uB^{-1}C_u^TA_a^{-1} & -A_f^{-1}C_uB^{-1} & A_f^{-1} \\ -B^{-1}C_u^TA_a^{-1} & B^{-1} & 0 \\ A_a^{-1} & 0 & 0 \end{bmatrix}. \tag{3.56}$$

Damit lässt sich die $3 \times 3$-Blockmatrix $M = I - R^{-1}\tilde{K}$ aufstellen:

$$M = \begin{bmatrix} M_{11} & M_{12} & 0 \\ M_{21} & M_{22} & 0 \\ M_{31} & M_{32} & 0 \end{bmatrix}. \tag{3.57}$$

Analog zu [IKSG10] berechnet man die einzelnen Blöcke der Matrix $M^2$ und erkennt, dass diese von der Form

$$M^2 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ * & * & 0 \end{bmatrix} \tag{3.58}$$

ist. Damit gilt offenbar $M^3 = MM^2 = 0$. □

Für eine beliebige, durch eine Vektornorm induzierte Matrixnorm $\|\cdot\|$ gilt die Abschätzung

$$\begin{aligned} \|(M+N)^3\| &\overset{\text{L. 3.2}}{=} \|M^2N + MN^2 + NMN + N^3 + NM^2 + N^2M + MNM\| && (3.59)\\ &\leq \|(M^2 + MN + NM + N^2)N\| + \|N(M^2 + NM)\| + \|MNM\| && (3.60)\\ &\leq \underbrace{\left(\|M^2 + MN + NM + N^2\| + \|M^2 + NM\| + \|M\|^2\right)}_{=:\theta} \underbrace{\|N\|}_{=:r}. && (3.61) \end{aligned}$$

Mit dieser Abschätzung kann nun eine erste hinreichende Bedingung für Konvergenz der Iteration geliefert werden (vgl [IKSG10], Lemma 2.2 und [Ghe07], Lemma 4.2):

**Theorem 3.3.** *Seien*

$$\theta := \|M^2 + MN + NM + N^2\| + \|M^2 + NM\| + \|M\|^2 \text{ und } r := \|N\| = \|R^{-1}(\tilde{K} - K)\|. \tag{3.62}$$

*Falls* $\theta r < 1$*, so gilt*

$$\rho(I - R^{-1}K) \leq \sqrt[3]{\theta r} < 1, \tag{3.63}$$

*d.h. die One-Shot-Iteration (3.51) konvergiert mit einer durch* $\sqrt[3]{\theta r}$ *nach oben beschränkten Konvergenzrate.*

*Beweis.* Nach [HJ85], Korollar 5.6.14, gilt für den Spektralradius einer Matrix $B$, dass $\rho(B) = \lim_{n\to\infty} \|B^n\|^{\frac{1}{n}}$. Damit erhält man

$$\rho(I - R^{-1}K) = \sqrt[3]{\rho((M+N)^3)} = \sqrt[3]{\lim_{n\to\infty} \|(M+N)^{3n}\|^{\frac{1}{n}}} \tag{3.64}$$

$$\leq \sqrt[3]{\theta r} < 1. \tag{3.65}$$

□

Unter Verwendung der $l_2$-Norm kann ein Konvergenzbeweis geliefert werden, der allein von der Güte der Approximationen $A_f$, $A_a$ und $B$ abhängt:

**Theorem 3.4.** *Seien*

$$r_A^f := \|I - A_f^{-1}C_y\|_2, \quad r_A^a := \|I - A_a^{-1}C_y^T\|_2, \quad r_S := \|I - B^{-1}S_A\|_2, \tag{3.66}$$

*sowie*

$$\bar{\theta} := \theta\varphi, \text{ mit } \varphi := \left\| \begin{bmatrix} A_f^{-1}C_u B^{-1}C_u^T & -A_f^{-1}C_u & I \\ -B^{-1}C_u^T & I & 0 \\ I & 0 & 0 \end{bmatrix} \right\|_2. \tag{3.67}$$

*Gilt dann*

$$\max\{r_A^f, r_A^a, r_S\} < \frac{1}{\bar{\theta}}, \tag{3.68}$$

*so ist*

$$\rho(I - R^{-1}K) \leq \sqrt[3]{\bar{\theta}\max\{r_A^f, r_A^a, r_S\}} < 1, \tag{3.69}$$

*d.h. die One-Shot-Iteration (3.51) konvergiert mit einer durch* $\sqrt[3]{\theta\|R^{-1}(\tilde{K}-K)\|_2}$ *nach oben beschränkten Konvergenzrate.*

*Beweis.* Nach Theorem 3.3 gilt mit $\|\cdot\| = \|\cdot\|_2$ schon $\rho(I - R^{-1}K) \leq \sqrt[3]{\theta r} = \sqrt[3]{\theta\|R^{-1}(\tilde{K}-K)\|_2}$. Durch geeignetes Faktorisieren von $R^{-1}(\tilde{K}-K)$ erhält man (vgl. [IKSG10], Theorem 2.3)

$$\|R^{-1}(\tilde{K}-K)\| \leq \varphi\max\{r_A^f, r_A^a, r_S\} \tag{3.70}$$

und damit $\rho(I - R^{-1}K) \leq \sqrt[3]{\theta\varphi\max\{r_A^f, r_A^a, r_S\}} < 1$, wenn $\max\{r_A^f, r_A^a, r_S\} < 1/\bar{\theta}$. □

In [IKSG10] ist außerdem gezeigt, dass $\bar{\theta}$ nach oben beschänkt ist, wenn $A_f \to C_y$ und $A_a \to C_y^T$. Damit ist gesichert, dass die Bedingung (3.68) erfüllt werden kann durch geeignete Wahl von $A_f$ und $A_a$ nahe genug an $C_y$ und $C_y^T$. Des Weiteren zeigt der obige Beweis, dass die Approximation von $B \approx S_A$ als *konsistente* reduzierte Hessematrix gegenüber einer Approximation der exakten reduzierten Hessematrix $T^T H T$ vorzuziehen ist, was in numerischen Implementationen der One-Shot-Iteration wie in [GS05] schon beobachtet werden konnte.

## 3.5 Erweiterung für zusätzliche Zustandsgleichungen

In praktischen Optimierungsanwendungen müssen häufig zusätzliche Zustandsgleichungen betrachtet werden. Beispielsweise soll während einer aerodynamischen Formoptimierung von Flugzeugflügeln bezüglich Widerstandsreduktion gewährleistet sein, dass dieser nicht an Auftriebskraft verliert. Eine andere Nebenbedingung wäre die Einhaltung eines bestimmten Volumens oder eine gewisse Flügeldicke.

In diesem Kapitel soll die auf einem rSQP-Verfahren basierende One-Shot-Iteration auf Probleme mit einer zusätzlichen, skalaren Nebenbedingung erweitert werden. In [Sch98] wird dazu das Konzept einer partiell reduzierten SQP-Methode (prSQP) vorgestellt, welches hier aufgegriffen wird.

Das zu betrachtende Optimierungsproblem schreibt sich in der folgenden Art:

$$\begin{aligned} \min \quad & f(y,u) \\ \text{s.t. } & c(y,u) = 0 \\ & h(y,u) = 0 \end{aligned} \tag{3.71}$$

wobei $h : Y \times U \to \mathbb{R}$ eine weitere Zustandsbedingung beschreibt, welche sich nicht reduzieren lässt. Um die Erweiterung des Algorithmus vorzunehmen, betrachte man die Schritte eines rSQP-Verfahrens aus Gleichung (3.16):

$$\begin{bmatrix} 0 & 0 & c_y^T \\ 0 & T^T H T & c_u^T \\ c_y & c_u & 0 \end{bmatrix} \begin{pmatrix} \Delta y \\ \Delta u \\ \bar{y} \end{pmatrix} + \begin{pmatrix} \nabla_y f \\ \nabla_u f \\ c \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}. \tag{3.72}$$

Dieses Gleichungssystem entspricht genau den notwendigen Bedingungen des folgenden linearquadratischen Minimierungsproblems:

$$\begin{aligned} \min_{(\Delta y, \Delta u)} \quad & \frac{1}{2} \begin{pmatrix} \Delta y \\ \Delta u \end{pmatrix}^T \begin{bmatrix} 0 & 0 \\ 0 & T^T H T \end{bmatrix} \begin{pmatrix} \Delta y \\ \Delta u \end{pmatrix} + \begin{pmatrix} \nabla_y f \\ \nabla_u f \end{pmatrix}^T \begin{pmatrix} \Delta y \\ \Delta u \end{pmatrix} \\ \text{s.t.} \quad & c_y \Delta y + c_u \Delta u + c = 0 \end{aligned} \tag{3.73}$$

Die adjungierte Variable $\bar{y}$ tritt dabei als Lagrangemultiplikator der linearisierten Nebenbedingung für $c$ auf. In jedem Iterationsschritt des rSQP-Verfahrens wird also das obige linearquadratische Minimierungsproblem gelöst. Eine Erweiterung für die zusätzliche Nebenbedingung $h$ erhält man durch Hinzufügen der linearisierten Gleichung:

$$\min_{(\Delta y, \Delta u)} \quad \frac{1}{2} \begin{pmatrix} \Delta y \\ \Delta u \end{pmatrix}^T \begin{bmatrix} 0 & 0 \\ 0 & T^T H T \end{bmatrix} \begin{pmatrix} \Delta y \\ \Delta u \end{pmatrix} + \begin{pmatrix} \nabla_y f \\ \nabla_u f \end{pmatrix}^T \begin{pmatrix} \Delta y \\ \Delta u \end{pmatrix} \tag{3.74}$$

$$\text{s.t.} \quad c_y \Delta y + c_u \Delta u + c = 0 \tag{3.75}$$

$$h_y \Delta y + h_u \Delta u + h = 0 \tag{3.76}$$

Bezeichnet man mit $\mu$ den Lagrangemultiplikator zur Bedingung (3.76), so sind die notwendigen Bedingungen zur Lösung von (3.74) - (3.76) durch das folgende Gleichungssystem gegeben:

$$\begin{bmatrix} 0 & 0 & h_y^T & c_y^T \\ 0 & T^T H T & h_u^T & c_u^T \\ h_y & h_u & 0 & 0 \\ c_y & c_u & 0 & 0 \end{bmatrix} \begin{pmatrix} \Delta y \\ \Delta u \\ \mu \\ \bar{y} \end{pmatrix} = \begin{pmatrix} -\nabla_y f \\ -\nabla_u f \\ -h \\ -c \end{pmatrix} \tag{3.77}$$

$$\iff \begin{matrix} (I) \\ (II) \\ (III) \\ (IV) \end{matrix} \quad \begin{bmatrix} 0 & 0 & h_y^T & c_y^T \\ 0 & T^T H T & h_u^T & c_u^T \\ h_y & h_u & 0 & 0 \\ c_y & c_u & 0 & 0 \end{bmatrix} \begin{pmatrix} \Delta y \\ \Delta u \\ \Delta \mu \\ \Delta \bar{y} \end{pmatrix} = \begin{pmatrix} -\nabla_y \mathscr{L} \\ -\nabla_u \mathscr{L} \\ -h \\ -c \end{pmatrix}. \tag{3.78}$$

wobei die Lagrangefunktion $\mathscr{L}$ hier um den Term $h(y,u)\mu$ erweitert ist, d.h. $\mathscr{L}(y,u,\bar{y},\mu) = f(y,u) + \bar{y}^T c(y,u) + \mu h(y,u)$. Anstelle der Berechnung von $c_y$, $c_y^T$ und $T^T H T$ aus der obigen Matrix werden nun wieder gemäß Kapitel 3.2.3 die Approximationen $A_f \approx c_y$, $A_a \approx c_y^T$, sowie die konsistente Approximation $B \approx S_A = \tilde{T}^T H \tilde{T}$ verwendet.

Das System (3.78) kann dann analog zu [HS06] (Lemma 3.1) partiell reduziert werden durch Elimination von $\Delta y$ und $\Delta \bar{y}$:

$$(IV) \quad \Delta y = -A_f^{-1}(c_u \Delta u + c) \quad \Longrightarrow \quad (III) \quad (h_u - h_y A_f^{-1} c_u)\Delta u = -h + h_y A_f^{-1} c \tag{3.79}$$

und

$$\begin{aligned} (I) \quad & \Delta \bar{y} = -A_a^{-1}(\nabla_y \mathscr{L} + h_y^T \Delta \mu) \\ & \Longrightarrow \quad (II) \quad B\Delta u + (h_u^T - c_u^T A_a^{-1} h_y^T)\Delta \mu = -\nabla_u \mathscr{L} + c_u^T A_a^{-1} \nabla_y \mathscr{L} \end{aligned} \tag{3.80}$$

Defniert man nun durch

$$g_h^T := h_u - h_y A_f^{-1} c_u \tag{3.81}$$

den konsistenten reduzierten Gradienten der Nebenbedingung $h$, so schreibt sich das reduzierte System als

$$\begin{bmatrix} B & g_h \\ g_h^T & 0 \end{bmatrix} \begin{pmatrix} \Delta u \\ \mu \end{pmatrix} = \begin{pmatrix} -\nabla_u \mathscr{L} + c_u^T A_a^{-1} \nabla_y \mathscr{L} \\ -h + h_y A_f^{-1} c \end{pmatrix}. \tag{3.82}$$

Der obere Term der rechten Seite dieses Gleichungssystems ist gerade der konsistente reduzierte Gradient der Zielfunktion $f$. Dieser wird analog zu Kapitel 3.2.3 mit Hilfe der approximativen, aktuellen adjungierten Variable berechnet. Zur Berechnung des konsistenten reduzierten Gradienten der Nebenbedingung $h$ ist ebenfalls ein Schritt eines adjungierten Algorithmus notwendig, der anstelle von $f_y$ die rechte Seite $h_y$ enthält. Sind diese Informationen vorhanden, so kann durch Lösen von (3.82) ein Schritt der Designvariable $\Delta u$ berechnet werden. Anschließend erfolgt das Update der Zustandsvariable $y$ analog zu Kapitel 3.2.3.

*Bemerkung.* Ist $B$ eine symmetrische und positiv definite Approximation, so ist das Gleichungssystem (3.82) äquivalent zur Lösung des linear-quadratischen Minimierungsproblems

$$\begin{aligned} \min_{\Delta u} \frac{1}{2}\Delta u^T B \Delta u + g^T \Delta u \\ \text{s.t.} \quad g_h^T \Delta u = -h + h_z A_f^{-1} c \end{aligned} \tag{3.83}$$

wobei $g \coloneqq \nabla_u \mathscr{L} - c_u^T A_a^{-1} \nabla_y \mathscr{L}$ den approximativen, reduzierten Gradienten der Zielfunktion darstellt.

Zur Interpretation der obigen Reduktion wird sie gemäß Kapitel 3.3 in einem Pseudozeitschrittverfahren angewendet. Das zu lösende präkonditionierte Pseudozeitsystem ist gegeben durch (vergleiche Gleichung (3.78))

$$\begin{bmatrix} 0 & 0 & h_y^T & A_a \\ 0 & B & h_u^T & c_u^T \\ h_y & h_u & 0 & 0 \\ A_f & c_u & 0 & 0 \end{bmatrix} \begin{pmatrix} \dot{y} \\ \dot{u} \\ \dot{\mu} \\ \dot{\bar{y}} \end{pmatrix} = \begin{pmatrix} -\nabla_y \mathscr{L} \\ -\nabla_u \mathscr{L} \\ -h \\ -c \end{pmatrix}. \tag{3.84}$$

Nach der Reduktion aus Gleichung (3.79) und (3.80) sowie der obigen Bemerkung wird dieses System zu dem linear-quadratischen Minimierungsproblem

$$\min_{\dot{u}} \frac{1}{2}\dot{u}^T B \dot{u} + g^T \dot{u} \tag{3.85}$$

$$\text{s.t.} \quad g_h^T \dot{u} = -h + h_z A_f^{-1} c \tag{3.86}$$

reduziert. Die Reduktion lässt sich damit als Projektion der Designgeschwindigkeit $\dot{u}$ des nicht erweiterten Pseudozeitsystems (3.46) in Richtung der linearisierte Nebenbedingung $h$ interpretieren (vergleiche [HS06]).

Aufgrund der Nichtlinearität des Problems erzeugt ein solcher Schritt in der Designvariable eine Abweichung von der zusätzlichen Zustandsgleichung $h$. Aus diesem Grund soll nach [HS06] und [Haz10] in jedem Optimierungsschritt nach der Berechnung eines vorläufigen Schrittes $u_{k+1}^0$ mit (3.85) - (3.86) ein weiterer Korrekturschritt in der Designvariable durchgeführt werden, mit dem der Abstand zur der für $h$ zulässigen Mannigfaltigkeit $S = \{u \mid h(y(u)) = 0\}$ minimiert wird. Es soll also das Minimierungsproblem

$$\min \quad \frac{1}{2}\|u - u_{k+1}^0\|^2 \tag{3.87}$$

$$\text{s.t. } h(y(u)) = 0 \tag{3.88}$$

gelöst werden. Dazu wird ein Schritt eines verallgemeinerten Gauß-Newton-Verfahrens (siehe z.B. [NW06]) angewendet. Dies resultiert in einem Korrekturschritt der folgenden Art:

$$u_{k+1} = u_{k+1}^0 + \frac{h - h_z A_a^{-1} c}{g_h^T g_h} g_h \tag{3.89}$$

(vergleiche dazu [HS06], Kapitel 3.1).

Ein Verfahren zur Lösung von (3.71) ist demnach durch die folgenden Schritte gegeben:

1. Führe einen Schritt des adjungierten Algorithmus für $f$ aus und erhalte $\bar{y}_{k+1}$ zur Berechnung des approximativen, reduzierten Gradienten der Zielfunktion $f$.
2. Führe einen Schritt des adjungierten Algorithmus für $h$ aus und berechne den approximativen, reduzierten Gradienten der Nebenbedingung $h$.
3. Approximiere $B_k \approx S_A$.
4. Löse das quadratische Minimierungsproblem (3.85) - (3.86) und berechne $u_{k+1}^0 = u_k + \Delta t \dot{u}$.
5. Berechne $u_{k+1}$ mit der Korrektur (3.89).
6. Führe einen Schritt des Algorithmus zur Lösung der Zustandgleichung für $c$ aus und erhalte $y_{k+1}$.

In [Haz10] wird dieses Verfahren auch für zwei zusätzliche skalare Nebenbedingungen der aerodynamischen Formoptimierung erläutert.

Zur Konvergenz des erweiterten Verfahrens sei auf [Ghe07] verwiesen. Für linear-quadratische Minimierungsprobleme mit skalaren sowie mit vektorwertigen zusätzlichen Nebenbedingung wird dort die Konvergenz der Iteration bewiesen. Zur Generalisierung auf nichtlineare Probleme wird dort eine algorithmische Formulierung vorgestellt, welche zwar nicht die exakte Umsetzung der Methode darstellt, allerdings bewiesen ist, dass sie asymptotisch gleich und konvergent zur selben Lösung ist. In einem aufgeführten numerischen Testfall entspricht der Aufwand der so erhaltenen Optimierung mittels One-Shot-Methode etwa dem 7-fachen Aufwand einer einzelnen Simulation der partiellen Differentialgleichung (vgl. [Ghe07]). In [HS06] wird von einem 4 bis 8-fachen Anfwand einer Simulation gesprochen. Dies entspricht einer drastischem Reduktion des Aufwands verglichen mit einem herkömmlichen Gradientenverfahrens, wie beispielsweise dem NAND- bzw. Black-Box-Ansatz aus Kapitel 3.2.1.

# 4 One-Shot-Optimierung mittels zweifach erweiterter Lagrangefunktion

In diesem Kapitel wird die One-Shot Methode vorgestellt, wie sie von A. Griewank et al. vorgeschlagen wird ([Gri06], [GGR08], [GH10], [GH11]). Vorausgesetzt wird dabei wieder ein iterativer Algorithmus zur Lösung der primalen Zustandsgleichung. Durch Anwendung von Automatischem Differenzieren (AD) wird ein adjungierter Algorithmus erzeugt, der in konsistenter Weise die Adjungiertengleichung iterativ löst und die benötigten Ableitungen bereitstellt. Da es in der Natur des Automatischen Differenzierens liegt, die Zielgrößen und zugehörigen Adjungierten an den selben Stellen auszuwerten, wird in Kapitel 4.2 eine Optimierungsstrategie nach [Gri06] und [GH11] vorgestellt, welche ein Update aller Variablen auf der Basis der Informationen eines Punktes ausführt. Dabei können die primale und die adjungierte Variable simultan und ohne Präkonditionierung in der sogenannten Piggy-Back-Iteration iteriert werden (Kapitel 4.3). Ziel von Kapitel 4.4 ist es dann, einen Präkonditionierer des Designraumes so zu wählen, dass die Iteration der Designvariable in diese Piggy-Back-Iteration integriert werden kann und die Konvergenz gesichert ist. Bevor diese in Kapitel 4.6 nach [GH11] bewiesen wird, geht Kapitel 4.5 noch kurz auf die Verwendung des Automatischen Differenzierens zur Erzeugung der benötigten ersten und zweiten Ableitungen ein.

## 4.1 Problemstellung und Voraussetzungen

Betrachtet wir das gleichungsbeschränkte Optimierungsproblem mit getrennten Variablen $(y,u) \in X = Y \times U$ aus Kapitel 2.2:

$$\begin{aligned} &\min_{y,u} \quad f(y,u) \\ &\text{s.t. } c(y,u) = 0. \end{aligned} \tag{4.1}$$

Dabei ist $f\colon Y \times U \to \mathbb{R}$ die zu minimierende Zielfunktion und $c\colon Y \times U \to Y$ die zustandsbeschreibende stationäre partielle Differentialgleichung. $f$ und $c$ seien wieder zweifach stetig differenzierbar.

Für $f$ wird des Weiteren die folgende Annahme getroffen:

***Annahme*** 4.1. Für die Zielfunktion $f$ gilt

$$\lim_{\|y\|+\|u\|\to\infty} f(y,u) = +\infty. \tag{4.2}$$

Diese Annahme stellt sicher, dass alle Niveaumengen von $f$ beschränkt sind und die Werte für $y$ und $u$ während der Optimierung in einem physikalisch sinnvollen Bereich bleiben.

Analog zu Abschnitt 2.2 wird weiterhin für $c$ die Annahme 2.5 getroffen, sodass $\frac{\partial c}{\partial y}$ stets invertierbar ist. Dann folgt mit Hilfe des Hauptsatzes über implizite Funktionen (siehe Theorem 2.10), dass für jede Designvariable $u$ genau eine Zustandsvariable $y$ existiert, die $c(y,u) = 0$ löst.

Da die Gleichungsbeschränkung $c(y,u) = 0$ eine partielle Differentialgleichung darstellt, beispielsweise die strömungsbeschreibende Navier-Stokes-Gleichung um eine gegebene Form $u$, wird angenommen, dass zur Lösung dieser PDE eine Fixpunktiteration bereitgestellt ist, etwa der Form

$$y_{k+1} = G(y_k, u) \quad \text{mit} \quad G\colon Y \times U \to Y, \tag{4.3}$$

welche zu einem festen $u$ eine zulässige Zustandsvariable $y^* = lim_{k\to\infty} y_k$ iteriert, für die $c(y^*,u) = 0$ gilt. Sie wird im Folgenden als *primale* Iteration bezeichnet. Um die Konvergenz dieser Iteration sicherzustellen, wird folgende Annahme getroffen:

***Annahme*** 4.2. Für $G\colon Y \times U \to Y$, $G$ zweifach stetig differenzierbar, gilt

$$\|G_y(y,u)\| = \|G_y^T(y,u)\| \leq \rho < 1 \tag{4.4}$$

für alle betrachteten Punkte $(y,u)$.

Dabei bezeichne $\|\cdot\|$ eine durch eine Vektornorm induzierte Matrixnorm. Dann folgt nach Anwendung des Mittelwertsatzes (siehe z.B. [BF95]), dass $G$ eine kontrahierende Funktion mit

$$\|G(y_1,u) - G(y_2,u)\| \leq \rho \|y_1 - y_2\| \tag{4.5}$$

und $\rho < 1$ ist.

Der Banachsche Fixpunktsatz (siehe z.B. [BF95], Kapitel 14.5) stellt dann die Konvergenz der primalen Iteration (4.3) sicher. D.h. für ein gegebenes Design $u$ kann eine zulässige Zustandsvariable $y^*$ mittels (4.3) iteriert werden, für die $y^* = G(y^*,u)$ und damit $c(y^*,u) = 0$ gilt. Die Konvergenzgeschwindigkeit ist dabei linear mit Rate $\rho$.

Mithilfe dieser Formulierung kann das zu lösende Optimierungsproblem umgeschrieben werden zu

$$\min_{y,u} \quad f(y,u) \quad s.t. \quad y = G(y,u). \tag{4.6}$$

## 4.2 Optimierungsstrategie

Die zu (4.6) assoziierte Lagrangefunktion (siehe Kapitel 2.3) ist definiert durch

$$\mathscr{L}(y,\bar{y},u) \coloneqq f(y,u) + (G(y,u) - y)^T \bar{y} \tag{4.7}$$

$$= N(y,\bar{y},u) - y^T \bar{y}. \tag{4.8}$$

$N$ ist der sogenannte „*shifted Lagrangian*" (vgl. z.B. [Gri06])

$$N(y,\bar{y},u) := f(y,u) + G(y,u)^T\bar{y}. \tag{4.9}$$

Man beachte, dass für die ersten Ableitungen

$$N_y = \mathscr{L}_y + \bar{y}, \qquad N_{\bar{y}} = \mathscr{L}_{\bar{y}} + y, \quad \text{und} \quad N_u = \mathscr{L}_u$$

gilt.

$N_u = f_u + \bar{y}^T G_u$ stellt dabei gerade den reduzierten Gradienten der Zielfunktion dar (vgl. Kapitel 2.5).

Nach Kapitel 2.3 muss ein stationärer Punkt $(y^*,\bar{y}^*,u^*)$ des Minimierungsproblems (4.6) die KKT-Bedingungen erfüllen (Theorem 2.7). Diese sind gegeben durch

$$\nabla_{\bar{y}}\mathscr{L} = 0 \qquad \Leftrightarrow \qquad y^* = G(y^*,u^*) \tag{4.10}$$

$$\nabla_y\mathscr{L} = 0 \qquad \Leftrightarrow \qquad \bar{y}^* = N_y(y^*,\bar{y}^*,u^*)^T = f_y(y^*,u^*)^T + G_y(y^*,u^*)^T\bar{y}^* \tag{4.11}$$

$$\nabla_u\mathscr{L} = 0 \qquad \Leftrightarrow \qquad 0 = N_u(y^*,\bar{y}^*,u^*)^T = f_u(y^*,u^*)^T + G_u(y^*,u^*)^T\bar{y}^* \tag{4.12}$$

Um einen solchen Punkt zu erreichen, soll die folgende gekoppelte *single-step One-Shot Iteration* durchgeführt werden (vergleiche [Gri06] und [GH11]):

$$\begin{bmatrix} y_{k+1} \\ \bar{y}_{k+1} \\ u_{k+1} \end{bmatrix} = \begin{bmatrix} G(y_k,u_k) \\ N_y(y_k,\bar{y}_k,u_k)^T \\ u_k - B_k^{-1}N_u(y_k,\bar{y}_k,u_k)^T \end{bmatrix} \begin{matrix} \text{„primale Iteration"} \\ \text{„adjungierte Iteration"} \\ \text{„Design-Iteration"} \end{matrix} \tag{4.13}$$

$B_k$ stellt dabei eine symmetrische, positiv definite Präkonditionierungsmatrix dar, welche die Konvergenz der Iteration (4.13) sicherstellt. Auf die Wahl eines geeigneten Präkonditionierers wird in Kapitel 4.4 genauer eingegangen, in dem eine zweifach erweiterte Lagrangefunktion betrachtet wird. Zur Konvergenz der ersten beiden Gleichungen von (4.13) ist keine Präkonditionierung notwendig. Sie ist Gegenstand des folgenden Abschnitts.

## 4.3 *Piggy-Back* Iteration

In [GF02] führt Griewank den Begriff *Piggy-Back* (dt. „*Huckepack*") ein, um die simultane Iteration der primalen und adjungierten Variable ohne Änderungen der Designvariable zu bezeichnen:

$$\begin{bmatrix} y_{k+1} \\ \bar{y}_{k+1} \end{bmatrix} = \begin{bmatrix} G(y_k,u) \\ N_y(y_k,\bar{y}_k,u)^T \end{bmatrix} = \begin{bmatrix} G(y_k,u) \\ f_y(y_k,u)^T + G_y(y_k,u)^T\bar{y}_k \end{bmatrix}. \tag{4.14}$$

Nach Annahme 4.2 ist $G$ eine kontrahierende Funktion, sodass mit dem Banachschen Fixpunktsatz (siehe z.B. [BF95], Kapitel 14.5) die lineare Konvergenz der primalen Iteration $y_{k+1} = G(y_k,u)$ zum eindeutigen Fixpunkt $y^*$ mit $y^* = G(y^*,u)$ für ein festes $u$ folgt (siehe Abschnitt 4.1). Da aber die Iterationsmatrix von $\bar{y}_{k+1} = N_y(y_k,\bar{y}_k,u)^T$ gerade $G_y(y_k,u)^T$ ist und

$\|G_y^T\| \leq \rho < 1$ gilt, ist auch $N_y$ kontrahierend und es folgt wieder die eindeutige Existenz eines Fixpunktes $\bar{y}^*$. Da $G$ und $f$ mindestens einmal stetig differenzierbar sind, sind $G_y$ und $f_y$ stetig und es folgt, dass $G_y(y_k,u) \to G_y(y^*,u)$ sowie $f_y(y_k,u) \to f_y(y^*,u)$ konvergieren für $y_k \to y^*$. Damit konvergiert die adjungierte Iteration simultan mit der primalen zu der eindeutigen Lösung

$$\bar{y}^* = f_y(y^*,u)^T + G_y(y^*,u)^T \bar{y} \quad \Leftrightarrow \quad 0 = f_y(y^*,u)^T + c_y(y^*,u)^T \bar{y}. \tag{4.15}$$

Dies entspricht genau der Adjungiertengleichung wie sie in Kapitel 2.5 vorgestellt wird (siehe Gleichung (2.31)). Durch Differenziation nach $y$ kann also eine Fixpunktiteration erzeugt werden, die eine Lösung der adjungierten Differentialgleichung liefert.

Obwohl beide, die primale und die adjungierte Variable, mit demselben asymptotischem Faktor konvergieren (vergleiche [GF02])

$$\limsup_{k\to\infty} \sqrt[k]{\|y_k - y^*\|} \quad \leq \quad \rho \quad \geq \quad \limsup_{k\to\infty} \sqrt[k]{\|\bar{y}_k - \bar{y}^*\|}, \tag{4.16}$$

ist aufgrund der Abhängigkeit der Adjungiertengleichung von der Lösung $y^*$ und der Ungenauigkeit der akutellen Variable $y_k$ beim Update für $\bar{y}_{k+1}$ in (4.14) eine Verzögerung der Konvergenz der adjungierten Variable zu erwarten. Dies wurde in [GK05] nachgewiesen; Griewank bezeichnet die Zeitverzögerung der adjungierten Iterierten relativ zur primalen Iterierten dort als „dual retardation" (dt. „duale Verzögerung"):

$$\|\bar{y}_k - \bar{y}^*\| \sim k\|y_k - y^*\|. \tag{4.17}$$

## 4.4 Bedingungen an einen geeigneten Präkonditionierer

In diesem Abschnitt wird der Frage nachgegangen, wie ein geeigneter Präkonditionierer $B$ gewählt werden muss, um ein Update der Designvariable $u$ in die Piggy-Back-Iteration (4.14) zu integrieren, sodass $u$ simultan mit der primalen und der adjungierten Variable iteriert werden kann (Iteration (4.13)).

In [Gri06] wird die Jakobimatrix $J^*$ der One-Shot-Iteration (4.13) am stationären Punkt betrachtet. Um die Kontraktivität der Iteration sicherzustellen, muss für den Spektralradius der Matrix die Bedingung $\rho(J^*) < 1$ erfüllt sein. Hierfür konnten notwendige Bedingungen an den Präkonditionierer $B$ hergeleitet werden; hinreichende Bedingungen, welche dann die lokale Konvergenz sichern würden, allerdings nicht.

Stattdessen wird in [GH10] und [GH11] nach Abstiegsrichtungen der zweifach erweiterten Lagrangefunktion

$$L^a(y,\bar{y},u) := \frac{\alpha}{2}\|G(y,u) - y\|^2 + \frac{\beta}{2}\|N_y(y,\bar{y},u)^T - \bar{y}\|^2 + N(y,\bar{y},u) - \bar{y}^T y \tag{4.18}$$

gesucht, wobei $\alpha, \beta > 0$ zu wählen sind. Sie setzt sich zusammen aus den gewichteten primalen und dualen Residuen sowie der Lagrangefunktion aus (4.7)

### 4.4.1 Exakte Penaltyfunktion

**Theorem 4.3.** *Ist die Bedingung*

$$\alpha\beta\Delta G_y^T \Delta G_y \succ I + \beta N_{yy} \qquad mit \qquad \Delta G_y = I - G_y \tag{4.19}$$

*erfüllt, so ist ein lokales Minimum des Optimierungsproblems (4.6) auch ein lokales Minimum der erweiterten Lagrangefunktion $L^a$, d.h. $L^a$ ist eine exakte Penaltyfunktion.*

*Bemerkung.* Das Symbol „$\succ$" wird in folgender Weise für quadratische Matrizen verwendet: $A \succ B :\Leftrightarrow A - B \succ 0 :\Leftrightarrow A - B$ ist positiv definit.

*Beweis.* Ein ausführlicher Beweis ist in [GH10] zu finden (Proposition 3.1 und Korollar 3.2, 3.3) und wird hier zusammengefasst dargestellt.

Der Gradient der erweiterten Lagrangefunktion $L^a$ lässt sich nach elementaren Umformungen schreiben als

$$\nabla L^a = \begin{bmatrix} \nabla_y L^a \\ \nabla_{\bar{y}} L^a \\ \nabla_u L^a \end{bmatrix} = -Ms(y, \bar{y}, u), \tag{4.20}$$

wobei

$$M = \begin{bmatrix} \alpha(I - G_y)^T & -I - \beta N_{yy} & 0 \\ -I & \beta(I - G_y) & 0 \\ -\alpha G_u^T & -\beta N_{yu}^T & B \end{bmatrix} \qquad \text{und} \qquad s(y, \bar{y}, u) = \begin{bmatrix} G(y,u) - y \\ N_y(y, \bar{y}, u)^T - \bar{y} \\ -B^{-1} N_u(y, \bar{y}, u)^T \end{bmatrix}. \tag{4.21}$$

Dabei stellt $s$ gerade den Inkrementvektor der Iteration (4.13) dar. Nach Blockelimination sieht man, dass $M$ regulär ist, falls die Voraussetzung (4.19) erfüllt ist. Damit ist ein stationärer Punkt der Minimierung von $L^a$ genau dann erreicht, wenn $0 = -Ms \Leftrightarrow s = 0$ gilt und dies ist genau die Bedingung an einen stationären Punkt des Optimierungsproblems (4.6). Anschließend wird mittels Blockdiagonalisierung der Hessematrix $\nabla^2 L^a$ gezeigt, dass die positive Definitheit dieser Matrix genau dann eintritt, wenn die reduzierte Hessematrix des Optimierungsproblems (4.6) positiv definit ist. Damit liegt an einem lokalen Minimum von (4.6) unter dieser Voraussetzung auch ein lokales Minimum von $L_a$ vor und $L_a$ ist eine exakte Penaltyfunktion (Informationen bezüglich Penaltyfunktionen siehe z.B. [NW06]). □

Die Bedingung (4.19) wird auch als **Korrespondenzbedingung** („correspondence condition", vgl. [GH11]) bezeichnet, um auf die 1 zu 1 Korrespondenz der stationären Punkte von $L^a$ und den stationären Punkten von (4.6) unter dieser Voraussetzung hinzuweisen.

### 4.4.2 Abstiegsrichtung der Penaltyfunktion

Wegen Theorem 4.3 können nun Bedingungen an den Präkonditionierer $B$ und weitere Bedingungen an $\alpha$ und $\beta$ aus der Suche nach Abstiegsrichtungen für die Penaltyfunktion $L^a$ hergeleitet werden.

Mit den Bezeichnungen aus (4.21) handelt es sich bei $s$ um eine Abstiegsrichtung für $L^a$, falls

$$s^T \nabla L^a = -s^T M s < 0 \tag{4.22}$$

gilt. Es genügt also zu prüfen, unter welchen Bedingungen die Matrix $M$ positiv definit ist. Dazu wird ihr symmetrischer Anteil

$$M_S = \frac{1}{2}(M^T + M) \tag{4.23}$$

betrachtet.

**Theorem 4.4.** *Wenn*

$$\alpha\beta\Delta\bar{G}_y \succ (I + \frac{\beta}{2}N_{yy})(\Delta\bar{G}_y)^{-1}(I + \frac{\beta}{2}N_{yy}) \quad \textit{mit} \quad \Delta\bar{G}_y = \frac{1}{2}(\Delta G_y + \Delta G_y^T) \tag{4.24}$$

*erfüllt ist, so ist $M_S$ positiv definit für alle großen $B$.*

*Beweis.* Der Beweis ist zu finden in [GH10], Proposition 3.4. □

Daraus folgt, dass unter der Voraussetzung (4.24) der Inkrementvektor $s$ aus (4.21) eine Abstiegsrichtung für alle $B$, welche „groß genug" (d.h. $B \succeq B_0$ für ein noch zu spezifizierendes $B_0$) gewählt werden, liefert. Die Bedingung (4.24) wird darum auch als **Abstiegsbedingung** („descent condition", vgl. [GH10]) bezeichnet.

Um sowohl die Korrespondenzbedingung (4.19), als auch die Abstiegsbedingung (4.24) zu erfüllen, genügt es nach [GH10], die Gewichte $\alpha$ und $\beta$ so zu setzen, dass

$$\sqrt{\alpha\beta}(1-\rho) > 1 + \frac{\beta}{2}\theta \quad \text{mit} \quad \theta = \|N_{yy}\| \tag{4.25}$$

gilt.

Zusammenfassend gilt dann: Werden $\alpha$ und $\beta$ nach (4.25) gewählt, so ist $L^a$ eine exakte Penaltyfunktion, für die $s$ eine Abstiegsrichtung für alle großen $B$ darstellt.

An der Abstiegsbegingung (4.24) erkennt man, dass eine Wahl großer Gewichte den Abstieg von $L^a$ sichert. Um allerdings eine möglichst schnelle Konvergenz der gesamten One-Shot-Iteration (4.13) zu erreichen, sollten sie möglichst klein gewählt werden. Einen Kompromiss zwischen beiden Überlegungen erhält man beispielsweise durch die Minimierung von $\alpha$ unter $\beta$, sodass die Bedingung (4.25) noch erfüllt ist (vlg. [GH10]):

$$\min_{\beta} \quad \sqrt{\alpha} = \frac{1 + \frac{\theta}{2}\beta}{(1-\rho)\sqrt{\beta}}. \tag{4.26}$$

Ein solches Minimum wird erreicht für

$$\beta = \frac{2}{\theta} \quad \Rightarrow \quad \alpha = \frac{2\theta}{(1-\rho)^2}. \tag{4.27}$$

### 4.4.3 Die Wahl von $B$

Da Theorem 4.4 keine konkrete Bedinung an $B$ liefert, wird in [GH11] zunächst eine untere Schranke für den Präkonditionierer angegeben, die die positive Definitheit von $M_S$ garantiert:

**Theorem 4.5.** *Erfüllen $\alpha$ und $\beta$ die Ungleichung (4.25), so gilt für alle $B$ mit*

$$B = B^T \succeq B_0 = \frac{1}{\sigma}(\alpha G_u^T G_u + \beta N_{yu}^T N_{yu}), \text{ wobei } \sigma = 1 - \rho - \frac{(1+\frac{\theta}{2}\beta)^2}{\alpha\beta(1-\rho)} > 0, \tag{4.28}$$

*dass der Inkrementvektor $s$ eine Abstiegsrichtung für $L^a$ ist.*

*Beweis.* Siehe [GH11] Kapitel 3.1, insbesondere Proposition 3.1 und 3.2. □

$B$ tritt im Inkrementvektor $s$ nur im Update der Designvariable $u$ auf, daher ist es naheliegend ein geeignetes $B$ aus

$$\min_{\Delta u} \quad L^a(y+\Delta y, \bar{y}+\Delta\bar{y}, u+\Delta u) \tag{4.29}$$

abzuleiten (vgl. [GÖ09] und [GH11]). Dabei bezeichnet $(\Delta y, \Delta\bar{y}, \Delta u)^T$ den durch $s$ in (4.21) definierten Schrittvektor, also $s = (\Delta y, \Delta\bar{y}, \Delta u)^T$ mit

$$\Delta y := G(y,u) - y, \tag{4.30}$$
$$\Delta\bar{y} := N_y(y,\bar{y},u)^T - \bar{y}, \tag{4.31}$$
$$\Delta u := -B^{-1} N_u(y,\bar{y},u)^T. \tag{4.32}$$

Durch eine quadratische Approximation der Zielfunktion in (4.29) gelangt man zu

$$\min_{\Delta u} \quad s^T \nabla L^a(y,\bar{y},u) + \frac{1}{2} s^T \nabla^2 L^a(y,\bar{y},u) s \tag{4.33}$$
$$\Leftrightarrow \quad \min_{\Delta u} \quad \psi(\Delta u), \tag{4.34}$$

wobei

$$\psi(\Delta u) := \Delta u^T (\nabla_u L^a + \nabla^2_{uy} L^a \Delta y + \nabla^2_{u\bar{y}} L^a \Delta\bar{y}) + \frac{1}{2}\Delta u^T \nabla^2_{uu} L^a \Delta u \tag{4.35}$$
$$\approx \Delta u^T \nabla_u L^a(y+\Delta y, \bar{y}+\Delta\bar{y}, u) + \frac{1}{2}\Delta u^T \nabla^2_{uu} L^a \Delta u. \tag{4.36}$$

Das Minimum dieser Approximation kann, unter der Voraussetzung, dass $\nabla^2_{uu} L^a$ positiv definit ist, angegeben werden als

$$\Delta u = -(\nabla^2_{uu} L^a(y,\bar{y},u))^{-1} \nabla_u L^a(y+\Delta y, \bar{y}+\Delta\bar{y}, u). \tag{4.37}$$

Da der durch $s$ definierte Schritt in der Designvariable gerade $\Delta u = -B^{-1} N_u^T$ ist, muss daher in der Nähe der Lösung, d.h. für $\Delta y \to 0$ und $\Delta\bar{y} \to 0$, $B$ so gewählt werden, dass $B \approx \nabla^2_{uu} L^a$ gilt.

Betrachtet man die zweite Ableitung der erweiterten Lagrangefunktion nach $u$, die gegeben ist durch

$$\nabla^2_{uu} L^a = \alpha G_u^T G_u + \beta N_{yu}^T N_{yu} + N_{uu} + \alpha \Delta y^T G_{uu} + \beta \Delta \bar{y}^T N_{yuu}, \tag{4.38}$$

so sieht man, dass sich (4.37) für primal und adjungiert zulässige Punkte, d.h. $\Delta y = 0$ und $\Delta \bar{y} = 0$ (und damit $N_u(y, \bar{y}, u) = L^a_u(y + \Delta y, \bar{y} + \Delta \bar{y}, u)$), sogar vereinfacht zu

$$\Delta u = -B^{-1} N_u(y, \bar{y}, u)^T, \text{ mit } B := \alpha G_u^T G_u + \beta N_{yu}^T N_{yu} + N_{uu} \tag{4.39}$$

Wählt man also die Gewichte $\alpha$ und $\beta$ nach (4.27), sowie $B = \alpha G_u^T G_u + \beta N_{yu}^T N_{yu} + N_{uu}$, so ergibt sich unter der Annahme $N_{uu} \succeq 0$, dass dieses $B$ der unteren Schranke $B_0$ aus Theorem 4.5 genügt

$$B = B^T = B_0 + N_{uu} \succeq B_0, \tag{4.40}$$

das heißt, der Inkrementvektor $s$ liefert Abstieg für $L^a$. Des Weiteren hat man $B \approx \nabla^2_{uu} L^a$, sowie an zulässigen Punkten sogar $B = \nabla^2_{uu} L^a$ und da $L^a$ eine exakte Penaltyfunktion ist, ist $\nabla^2_{uu} L^a$ positiv definit in der Nähe eines lokalen Minimums.

### 4.4.4 BFGS-Update zur Approximation von $B$

Um die aufwendige Berechnung der Matrizen in der Definition von $B$ aus (4.39) zu vermeiden, wird numerisch eine BFGS-Methode (siehe [NW06]) zur Approximation von $B$ vorgezogen (vlg. [GH11]). Dabei wird in jeder Iteration ein Update des aktuellen $B$ durchgeführt, sodass die gewünschte Eigenschaft $B \approx \nabla^2_{uu} L^a$ erreicht wird. Wegen

$$\nabla^2_{uu} L^a \approx \nabla_u L^a(y, \bar{y}, u + \Delta u) - \nabla_u L^a(y, \bar{y}, u) \tag{4.41}$$

ist die Sekantenbedingung des BFGS-Updates für $B_{k+1}$ gegeben durch

$$B_{k+1} \Delta u_k = R_k \quad \text{mit} \quad R_k := \nabla_u L^a(y_k, \bar{y}_k, u_k + \Delta u_k) - \nabla_u L^a(y_k, \bar{y}_k, u_k). \tag{4.42}$$

Für ein positiv definites $B_{k+1}$ kann die Sekantenbedingung nur dann eine Lösung haben, falls die sogenannte Krümmungsbedingung („curvature condition", vgl. [NW06])

$$\Delta u_k^T R_k > 0 \tag{4.43}$$

erfüllt ist. Diese Bedingung ist in der Nähe eines Minimums stets erfüllt, da dort $\nabla^2_{uu} L^a$ positiv definit ist. Andernfalls kann sie beispielsweise durch geeignete Bedingungen an die Liniensuche gesichert, oder durch setzen von $B_{k+1} = I$ oder $B_{k+1} = B_k$ im Falle der Nichterfülltheit ersetzt werden.

## 4.5 Berechnung der Ableitungen

Bei der Berechnung der auftretenden Ableitungen muss stets beachtet werden, dass die Kosten des Optimierungprozesses proportional zu den Kosten einer Lösung der partiellen Differentialgleichung sein sollten, man spricht dann von *„bounded retardation"* (dt. „beschränkte Verzögerung") (vergleiche auch [GGR08]). Dazu ist es notwendig, dass der Aufwand der Berechnungen der Ableitungen unabhängig von der Anzahl der Parameter, d.h. von der Größe des Vektors $u$, ist. Um dies zu erreichen, wird von Griewank et al. die Verwendung des *Automatischen (auch: Algorithmischen) Differenzierens* (AD) empfohlen ([GF02], [Gri03]). Dabei muss die zu differenzierende Funktion als Algorithmus in einer geeigneten Programmiersprache vorliegen. Durch die Anwendung einer AD-Software wird nach Zerlegung der Funktion in elementare algorithmische Operationen, Anwendung der bekannten Ableitungsregeln darauf und anschließender Verwendung der Kettenregel eine Funktion erzeugt, die die Ableitung der Funktion bestimmt. Informationen zur Automatischen Differenziation und eine Übersicht verfügbarer AD-Software sind online z.B. unter [Uni11] verfügbar. Eine ausführliche Betrachtung ist beispielsweise in [Gri00] nachzulesen. Man unterscheidet zwischen dem Vorwärtsmodus („*forward*" oder auch „*tangent*") und dem Rückwärtsmodus („*reverse*"). Kurz gesprochen liefert der Vorwärtsmodus die Richtungsableitung der abhängigen Variablen der Funktion nach gegebener Variation der unabhängigen Variablen, während der Rückwärtsmodus den mit einer gegebenen Variation der abhängigen Variablen gewichteten Gradienten der Funktion berechnet. Eine kurze Einführung in die beiden Modi ist auch in [NW06] Kapitel 8.2 zu finden. Da der Vorwärtsmodus Richtungsableitungen liefert, muss also zur Berechnung des reduzierten Gradienten der *reverse*-Modus angewendet werden, um die Abhängigkeit von der Größe des Vektors $u$ zu vermeiden.

### Berechnung von $N_y$ und $N_u$

Ist ein Algorithmus zur Durchführung der Fixpunktiteration der primalen Variable und Auswertung der Zielfunktion gegeben, so kann nach den obigen Ausführungen durch Algorithmisches Differenzieren im *reverse*-Modus nach den Variablen $y$ und $u$ ein Algorithmus erzeugt werden, welcher sowohl die zum Update der adjungierten Variable benötigte Ableitung $N_y$ als auch den reduzierten Gradienten $N_u$ simultan auswertet. D.h., für die im Optimierungprozess aktuellen Variablen $(y_k, \bar{y}_k, u_k)$ liefert ein Aufruf des so erzeugten Algorithmus gerade die Ableitungen

$$\begin{bmatrix} N_y(y_k, \bar{y}_k, u_k)^T \\ N_u(y_k, \bar{y}_k, u_k)^T \end{bmatrix}. \tag{4.44}$$

Wie schon in Kapitel 4.3 angemerkt, wird also die adjungierte Iteration automatisch durch Differenziation nach $y$ erzeugt, wodurch die Konsistenz der adjungierten zur primalen Iteration gesichert ist.

### Berechnung von $\nabla_u L^a$

Zur Approximation von $B$ aus (4.39) werden nach Kapitel 4.4.4 BFGS-Updates auf dem Gradienten der erweiterten Lagrangefunktion nach $u$ berechnet. Mit den Bezeichnungen aus (4.30) -

(4.32) ist dieser Gradient gegeben durch

$$\nabla_u L^a = \alpha \Delta y^T G_u + \beta \Delta \bar{y}^T N_{yu} + N_u. \tag{4.45}$$

Im Gegensatz zu $B$ sind hier alle auftretenden Terme Vektoren oder Matrixvektorprodukte. Die Algorithmen zur Berechnung der ersten Ableitungen lassen sich wie im vorigen Abschnitt mittels Automatischen Differenzierens im *reverse*-Modus erzeugen. Um das Produkt $\beta \Delta \bar{y}^T N_{yu}$ zu berechnen, wird der Algorithmus zuerst im *tangent*-Modus nach $y$ differenziert und die Richtungsableitung in Richtung $\beta \Delta \bar{y}$ erzeugt, anschließend wird dies im *reverse*-Modus nach $u$ differenziert.

## 4.6 Globale Konvergenz

Durch die Wahl von $B$ aus (4.39) gilt

$$\nabla^2_{uu} L^a = \alpha G_u^T G_u + \beta N_{yu}^T N_{yu} + N_{uu} + \underbrace{\alpha \Delta y^T G_{uu}}_{\xrightarrow[y \to y^*]{} 0} + \underbrace{\beta \Delta \bar{y}^T N_{yuu}}_{\xrightarrow[\bar{y} \to \bar{y}^*]{} 0} \tag{4.46}$$

$$\approx B \tag{4.47}$$

$$\tag{4.48}$$

in der Nähe eines Minimums, sodass die Konvergenz der Iteration (4.13) dort zu erwarten ist. Um allerdings für einen beliebigen Startpunkt $(y_0, \bar{y}_0, u_0)$ Konvergenz zu sichern, muss eine Liniensuche (vgl. [NW06]) angewendet werden, welche für die Abstiegsrichtung $s$ aus (4.21) Schrittlängen bestimmt, die monotonen Abstieg der exakten Penaltyfunktion $L^a$ liefern. In [GH10] und [GH11] wird dazu beispielsweise eine standardmäßige Backtracking-Liniensuche (vgl. [NW06]) auf einer quadratischen Approximation von $L^a$ vorgeschlagen. Für den Startpunkt $(y_0, \bar{y}_0, u_0)$ liegen dann alle Iterierten der One-Shot-Iteration (4.13) in der unteren Niveaumenge $\mathcal{N}_0$ von $L^a$, definiert durch

$$\mathcal{N}_0 := \{(y, \bar{y}, u) \quad | \quad L^a(y, \bar{y}, u) \leq L^a(y_0, \bar{y}_0, u_0)\} \tag{4.49}$$

Das folgende Theorem (entnommen aus [GH11]) liefert die Beschränktheit aller Niveaumengen von $L^a$:

**Theorem 4.6.** *Wenn* $\lim_{\|y\|+\|u\| \to \infty} f(y,u) = +\infty$ *und*

$$\liminf_{\|y\|+\|u\| \to \infty} \frac{f}{\|\nabla_y f\|^2} > 0 \tag{4.50}$$

*gilt, dann existieren* $\alpha$ *und* $\beta$*, welche (4.25) erfüllen und*

$$\lim_{\|y\|+\|\bar{y}\|+\|u\| \to \infty} L^a(y, \bar{y}, u) = +\infty \tag{4.51}$$

*gilt. Ist des Weiteren der Grenzwert von (4.50) gegeben durch* $+\infty$*, so gilt (4.51) ohne weitere Beschränkungen an* $\alpha$ *und* $\beta$*.*

*Beweis.* Siehe [GH11], Theorem 2.1. □

Betrachtet man nun für ein $(y, \bar{y}, u)$ aus einer beliebige Niveaumenge $\mathcal{N}$ den Winkel $\gamma$ zwischen der Richtung des steilsten Abstiegs $-\nabla L^a(y, \bar{y}, u)$ und der Schrittrichtung $s(y, \bar{y}, u)$ aus (4.21), so kann unter der Annahme, dass $N_{uu}$ positiv semidefinit ist (dies wird benötigt, damit das gewählte $B$ Theorem 4.5 genügt) gezeigt werden, dass

$$\cos\gamma = -\frac{s^T \nabla L^a}{\|\nabla L^a\| \|s\|} \geq C > 0 \quad \text{für alle} \quad (y, \bar{y}, u) \in \mathcal{N} \tag{4.52}$$

gilt für ein von $(y, \bar{y}, u)$ unabhängiges $C$ (vgl. [GH11], Proposition 5.1).

Da nach [NW06], Theorem 3.2, für die Abstiegsiteration unter Verwendung der Backtracking-Liniensuche und den obigen Voraussetzungen

$$\sum_{k \geq 0} cos^2 \gamma_k \|\nabla L^a(y_k, \bar{y}_k, u_k)\|^2 < \infty \tag{4.53}$$

gilt, folgt schon, dass

$$cos^2 \gamma_k \|\nabla L^a(y_k, \bar{y}_k, u_k)\|^2 \to 0 \quad (\text{für } k \to \infty) \tag{4.54}$$

und wegen (4.52) muss schließlich

$$\lim_{k \to \infty} \|\nabla L^a(y_k, \bar{y}_k, u_k)\| = 0 \tag{4.55}$$

gelten, womit die globale Konvergenz gezeigt ist.

Zusammenfassend sind die zu erfüllenden Voraussetzungen für globale Konvergenz der One-Shot-Iteration (4.13) demnach gegeben durch

- Annahmen aus Kapitel 4.1:
  - $f$ und $G$ sind zweifach stetig differenzierbar
  - $\lim_{\|y\|+\|u\| \to \infty} f(y, u) = +\infty$
  - $\|G_y\| \leq \rho < 1$
- Voraussetzung für Theorem 4.6:

$$\liminf_{\|y\|+\|u\| \to \infty} \frac{f}{\|\nabla_y f\|^2} = +\infty$$

- Wahl von $\alpha$ und $\beta$ so, dass Ungleichung (4.25) erfüllt ist:

$$\sqrt{\alpha\beta}(1-\rho) > 1 + \frac{\beta}{2}\|N_{yy}\|$$

- $N_{uu} \succeq 0$ und Wahl von $B$ aus Definition (4.39):

$$B := \alpha G_u^T G_u + \beta N_{yu}^T N_{yu} + N_{uu}$$

- Anwendung einer Liniensuche (z.B. Backtracking)

# 5 Vergleich der vorgestellten One-Shot-Methoden

Die Schritte der One-Shot-Iteration bestehen aus den Updateformeln für die primale Variable, die adjungierte Variable und die Designvariable. Diese werden nun im Hinblick auf die Unterschiede der vorgestellen Methoden aus Kapitel 3 und 4 zusammenfassend beschrieben.

- Die **primale Iteration** ist ein iterativer Prozess zur Lösung der Zustandsgleichung. Beide vorgestellten Verfahren setzen das Vorhandensein eines solchen Algorithmus voraus und verwenden ihn in äquivalenter Weise. Für eine gegebene Designvariable $u$ wird durch eine Fixpunktiteration, etwa der Form

$$y_{k+1} = y_k - A_f^{-1} c(y_k, u), \tag{5.1}$$

  eine zulässige Zustandsvariable $y^* = lim_{k\to\infty} y_k$ iteriert, für die die Zustandsgleichung erfüllt ist, für die also $c(y^*, u) = 0$ gilt. Für nichtlineare Probleme kann $c$ zuvor geeignet linearisiert werden. Wird ein Newton-Verfahren zur Lösung der Zustandsgleichung angewendet, so ist $A_f = c_y$ zu wählen. Da die Berechnung und Invertierung der Jakobimatrix der Nebenbedingung häufig nicht in Frage kommt, werden Quasi-Newton-Verfahren in Betracht gezogen, d.h. $A_f \approx c_y$. Dabei sind alle Präkonditionierer möglich, welche sicherstellen, dass der Spektralradius der Iterationsmatrix kleiner als 1 ist und somit die Konvergenz der Iteration gewährleistet ist mit einer Konvergenzrate $\rho < 1$. Wenn die Konvergenzrate der Iteration nahe bei 1 liegt, also von einer eher langsam konvergierenden Iteration ausgegangen wird, so spricht Griewank in [Gri06] von einer *Pseudo-Newton-Iteration* statt von einer Newton-ähnlichen Iteration. In seiner Schreibweise fasst er das Update der primalen Variable allgemein durch

$$y_{k+1} = G(y_k, u) \tag{5.2}$$

  zusammen, um die Festlegung auf einen bestimmten Algorithmus zu vermeiden. In den beiden vorgestellten One-Shot-Verfahren wird in jeder Optimierungsiteration genau ein Update der primalen Iteration nach (5.1) bzw. (5.2) durchgeführt.

- Die **Iteration der adjungierten Variable** liefert für zulässige Punkte $(y^*, u^*)$ mit $c(y^*, u^*) = 0$ einen Lagrangemultiplikator $\bar{y}^* = \lim_{k\to\infty} \bar{y}_k$, der die Adjungiertengleichung nach Kapitel 2.5, Gleichung (2.31), löst und mit dessen Hilfe der reduzierte Gradient $f_u^T + c_u^T \bar{y}^*$ an der Stelle $(y^*, u^*)$ berechnet werden kann. Da im Rahmen eines One-Shot-Verfahrens in jeder Optimierungsiteration nur ein Schritt zur Lösung der Zustandsgleichung ausgeführt wird, gilt bei Durchführung eines Updates der adjungierten Variable im Allgemeinen nur $c(y^*, u^*) \approx 0$ und man erhält lediglich eine Approximation an den reduzierten Gradienten, welche dann für das Update der Designvariable verwendet wird.

An dieser Stelle besteht ein entscheidender Unterschied zwischen den vorgestellten One-Shot-Methoden, welcher aus der unterschiedlichen Herleitung der Adjungiertengleichung hervorgeht.

Für die auf einem approximativen rSQP-Verfahren basierende One-Shot-Methode nach Kapitel 3 wird die kontinuierliche Adjungiertengleichung mittels Defektkorrektur - Verfahren iterativ gelöst. Nach einem Update der primalen Iteration wird an der neuen Variable $y_{k+1}$ ein Update der adjungierten Variable durchgeführt und mit dieser neuen Variable $\bar{y}_{k+1}$ wird dann der approximative reduzierte Gradient $\gamma_k$ berechnet. Mit $A_a \approx c_y^T$ schreibt sich dies nach dem Update der primalen Variable als

$$\bar{y}_{k+1} = \bar{y}_k - A_a^{-1}(f_y(y_{k+1},u_k)^T + c_y(y_{k+1},u_k)^T\bar{y}_k) \tag{5.3}$$

$$\gamma_k = f_u(y_{k+1},u_k)^T + c_u(y_{k+1},u_k)^T\bar{y}_{k+1}. \tag{5.4}$$

Das in Kapitel 4 vorgestellte One-Shot-Verfahren nach Griewank et al. basiert hingegen auf der Verwendung des Automatischen Differenzierens (AD) im Rückwärtsmodus zur Erzeugung der adjungierten Iteration und des approximativen reduzierten Gradienten. Dabei wird der Algorithmus zur iterativen Lösung der diskretisierten Zustandsgleichung sowie zur Berechnung der diskretisierten Zielfunktion algorithmisch differenziert (siehe Abschnitt 4.5). Die zu lösende Adjungiertengleichung entspricht somit der diskretisierten Formulierung aus Kapitel 2.5. Vorteilhaft ist dabei, dass die Lösung der Adjungiertengleichung dann konsistent zum Diskretisierungsschema der Zustandsgleichung ist. Zu beachten ist allerdings die Stelle der Auswertung der so berechneten Ableitungen: Durch Verwendung des Rückwärtsmodus von AD können simultan zur Auswertung der Updateformel $G$ der primalen Iteration und Berechnung der Zielgröße $f$ die zugehörige adjungierte Variable und der approximative Gradient berechnet werden. Neben der Auswertung von $[G(y_k,u_k), f(y_k,u_k)]$ erhält man also simultan $[N_y(y_k,\bar{y}_k,u_k), N_u(y_k,\bar{y}_k,u_k)]$ an den selben Stellen $(y_k,\bar{y}_k,u_k)$. Die Berechnung von $N_u$ an der Stelle $(y_{k+1},\bar{y}_{k+1},u_k)$ wie in der One-Shot-Methode nach Schulz et al. (Gleichung (5.4)) würde eine komplett neue Auswertung derselben Größen an dieser Stelle erfordern und wird daher für die One-Shot-Iteration nach Griewank et al. nicht verwendet (vgl. [Gri06]). Zum übersichtlicheren Vergleich der Verfahren sei das Update der adjungierten Variable nach Griewank et al. (Kapitel 4) in diesem Kapitel in der Schreibweise aus [GF02] dargestellt: Für ein primales Update der obigen Form $y_{k+1} = y_k - A_f^{-1}c(y_k,u)$ liefert der Rückwärtsmodus von AD das Update der zugehörigen adjungierten Variable sowie den reduzierten Gradienten $\gamma_k$ der folgenden Art:

$$\bar{y}_{k+1} = \bar{y}_k - A_a^{-1}(f_y(y_k,u_k)^T + c_y(y_k,u_k)^T\bar{y}_k) \tag{5.5}$$

$$\gamma_k = f_u(y_k,u_k)^T + c_u(y_k,u_k)^T\bar{y}_k. \tag{5.6}$$

- Für das **Designupdate** wird bei beiden One-Shot-Methoden der jeweilige approximierte, reduzierte Gradient (Gleichungen (5.4) bzw. (5.6)) präkonditioniert. Für die auf einem rSQP-Verfahren basierende One-Shot-Methode (Kapitel 3) muss der Präkonditionierer die konsistente reduzierte Hessematrix approximieren. Das Update der Designvariable ist dann gegeben durch

$$u_{k+1} = u_k - B_S^{-1}(f_u(y_{k+1},u_k)^T + c_u(y_{k+1},u_k)^T\bar{y}_{k+1}), \tag{5.7}$$

wobei $B_S \approx \tilde{T}^T H \tilde{T}$. Numerisch werden zu dessen Approximation BFGS-Updates auf dem approximativen reduzierten Gradienten durchgeführt. Für das Verfahren nach Kapitel 4 (Griewank et al.) wird ein Präkonditionierer mittels zweifach erweiterter Lagrangefunktion $L^a$ hergeleitet, welcher die Konvergenz des Verfahrens in Verbindung mit einer Liniensuche sichert. Das Update der Designvariable schreibt sich dort als

$$u_{k+1} = u_k - B_G^{-1}(f_u(y_k,u_k)^T + c_u(y_k,u_k)^T \bar{y}_k), \tag{5.8}$$

wobei $B_G$ nach Kapitel 4.4.3, Gleichung (4.39), gewählt wird. In der Nähe eines Lösungspunktes gilt dann $B_G \approx \nabla^2_{uu} L^a$. Auch hier werden BFGS-Updates zur Approximation von $B_G$ herangezogen, allerdings erfolgen diese anhand des Gradienten der erweiterten Lagrangefunktion, wie in Kapitel 4.4.4 vorgestellt. Zur Berechnung der Sekantenbedingung nach Gleichung (4.42) ist allerdings die Auswertung dieses Gradienten an zwei verschiedenen Stellen notwendig.

Für einen direkten Vergleich der One-Shot-Schritte ist die Notation mit Hilfe der Lagrangefunktion

$$\mathscr{L}(y,\bar{y},u) = f(y,u) + \bar{y}^T c(y,u) \tag{5.9}$$

sinnvoll. Die Schritte des auf der rSQP-Methode basierenden One-Shot-Verfahrens nach Kapitel 3 berechnen sich aus dem System

$$\begin{bmatrix} 0 & 0 & A_a \\ 0 & B_S & c_u^T \\ A_f & c_u & 0 \end{bmatrix} \begin{pmatrix} \Delta y_k \\ \Delta u_k \\ \Delta \bar{y}_k \end{pmatrix} = \begin{pmatrix} -\nabla_y \mathscr{L}(y_k,\bar{y}_k,u_k) \\ -\nabla_u \mathscr{L}(y_k,\bar{y}_k,u_k) \\ -\nabla_{\bar{y}} \mathscr{L}(y_k,\bar{y}_k,u_k) \end{pmatrix}. \tag{5.10}$$

Nach der obigen Beschreibung und Kapitel 3.2.3 liefert dies die Iteration

$$\begin{aligned} y_{k+1} &= y_k - A_f^{-1} \nabla_{\bar{y}} \mathscr{L}(y_k,\bar{y}_k,u_k) \\ \bar{y}_{k+1} &= \bar{y}_k - A_a^{-1} \nabla_y \mathscr{L}(y_{k+1},\bar{y}_k,u_k) \\ u_{k+1} &= u_k - B_S^{-1} \nabla_u \mathscr{L}(y_{k+1},\bar{y}_{k+1},u_k). \end{aligned} \tag{5.11}$$

Demgegegenüber schreibt sich die auf der erweiterten Lagrangefunktion basierende One-Shot-Iteration aus Kapitel 4 in obiger Notation als

$$\begin{aligned} y_{k+1} &= y_k - A_f^{-1} \nabla_{\bar{y}} \mathscr{L}(y_k,\bar{y}_k,u_k) \\ \bar{y}_{k+1} &= \bar{y}_k - A_a^{-1} \nabla_y \mathscr{L}(y_k,\bar{y}_k,u_k) \\ u_{k+1} &= u_k - B_G^{-1} \nabla_u \mathscr{L}(y_k,\bar{y}_k,u_k) \end{aligned} \tag{5.12}$$

oder in Matrixschreibweise

$$\begin{bmatrix} 0 & 0 & A_a \\ 0 & B_G & 0 \\ A_f & 0 & 0 \end{bmatrix} \begin{pmatrix} \Delta y_k \\ \Delta u_k \\ \Delta \bar{y}_k \end{pmatrix} = \begin{pmatrix} -\nabla_y \mathscr{L}(y_k,\bar{y}_k,u_k) \\ -\nabla_u \mathscr{L}(y_k,\bar{y}_k,u_k) \\ -\nabla_{\bar{y}} \mathscr{L}(y_k,\bar{y}_k,u_k) \end{pmatrix} \tag{5.13}$$

(vgl. auch [IKSG10]). An (5.11) und (5.12) erkennt man sofort den zentralen Unterschied der Methoden: In (5.4) werden die neuen Informationen sofort, d.h. noch in derselben Iteration, verwendet, in (5.12) hingegen erst im folgenden Iterationszyklus. Schematisch kann dies in der folgenden Art dargestellt werden:

$$(5.11): \quad \cdots \to \text{primal} \to \text{adjungiert} \to \text{design} \to \cdots$$
$$(5.12): \quad \cdots \to [\text{primal, adjungiert, design}] \to \cdots$$

Hier wird auch die von Griewank in [GH11] verwendete Bezeichnung „single-step"-One-Shot für die Iteration (5.12) deutlich. Das Update der drei Variablen wird in einem Schritt gemeinsam auf der Basis der Informationen an $(y_k, \bar{y}_k, u_k)$ ausgeführt.

Für lineare Gleichungssysteme, wie sie etwa bei linear-quadratischen Minimierungsproblemen auftreten, entspricht dies gerade dem Unterschied eines Gauß-Jakobi-Verfahrens zu einem Gauß-Seidel-Verfahren (für Informationen bezüglich Gauß-Jakobi-/Gauß-Seidel-Verfahren siehe z.B. [Her06]). Die Konvergenzrate des Gauß-Seidel-Verfahrens, welches die neusten Informationen sofort bei Verfügbarkeit verwendet, ist dabei nach [Her06] häufig kleiner als die des Gauß-Jakobi-Verfahren. Auch im Hinblick auf die Herleitung des One-Shot-Verfahrens aus Kapitel 3 als approximatives rSQP-Verfahren können gute Konvergenzeigenschaften dieses Verfahrens (5.11) aufgrund der superlinearen Konvergenz von rSQP-Verfahren erwartet werden. Bezüglich der Vereinfachung, dass ein Updatezyklus nach (5.12) stets den Auswertungspunkt $(y_k, \bar{y}_k, u_k)$ verwendet, statt der Auswertung des reduzierten Gradienten an $(y_{k+1}, \bar{y}_{k+1}, u_k)$, bemerkt Griewank in [Gri06] hingegen: „While this simplification may cause the loss of superlinear convergence for an RSQP like method, we believe that it makes very little difference for the convergence speed in a pseudo-Newton setting". Der in Kapitel 6 betrachtete numerische Testfall bestätigt diese These insofern, als dass dort kein signifikanter Unterschied in den Konvergenzraten der beiden One-Shot-Verfahren festgestellt werden konnte.

Für ebendiese Klasse der linear-quadratischen Minimierungsprobleme, d.h. Minimierung einer quadratischen Zielfunktion unter einer linearen Nebenbedingung, ist ein Konvergenzbeweis der Iteration (5.11) nach Schulz et al. vorhanden (vgl. Kapitel 3.4). Für allgemeine nichtlineare Minimierungsprobleme kann die Konvergenz der Iteration demnach lokal, d.h. für hinreichend nahe an dem lokalen Minimalpunkt liegende Startpunkte angenommen werden, da dort die nichtlineare Zielfunktion durch eine quadratische Zielfunktion hinreichend gut approximiert und die nichtlineare Nebenbedingung durch eine linearisierte Nebenbedingung ersetzt werden kann.

Die Konvergenz der Iteration (5.12) nach Griewank et al. ist hingegen für beliebige Startpunkte gesichert durch die Wahl eines speziellen Präkonditionierers des Designraumes (vgl. Kapitel 4.4 und 4.6). Es wird demnach die globale Konvergenz der Iteration gegen einen lokalen Minimalpunkt des allgemeinen Optimierungsproblems erreicht.

# 6 Numerische Ergebnisse

## 6.1 Der Testfall - Ein inverses Designproblem

Zum Zweck eines numerischen Vergleichs der beiden One-Shot Verfahren dient ein inverses Designproblem. Die Nebenbedingungen des Optimierungproblems sind dabei gegeben durch die Navier-Stokes-Gleichungen. Durch Variation an den Randbedingungen soll ein vorgegebenes Strömungsfeld reproduziert werden.

Betrachtet wird die Strömung eines inkompressiblen Fluids im Inneren einer zweidimensionalen, quadratischen, abgeschlossenen Box, welche durch die stationären Navier-Stokes-Gleichungen beschrieben wird. Eine ausführliche Herleitung dieser Gleichungen sowie deren Vereinfachungen ist in vielen Lehrbüchern der Fluiddynamik verfügbar, siehe z.B. [FP02]. Am oberen und unteren Rand der Box erfolgt kein Wärmeaustausch mit der Umgebung, es handelt sich also um adiabatische Wände. An den anderen, isothermalen Rändern werden Temperaturen vorgegeben: die linke Seite bleibt konstant kalt, während an dem rechten Rand wärmere Temperaturen variabel gesetzt werden können. Die Temperaturverteilung der rechten Wand steuert das Strömungsfeld im Inneren der Box; diese Parameter dienen im Testfall als Optimierungsvariablen.

Die Lösung der stationären, inkompressiblen Navier-Stokes-Gleichungen erfolgt, nach Hinzufügen einer Pseudozeit, durch ein implizites Finite-Volumen-Verfahren unter Verwendung der Druck-Korrektur-Methode (SIMPLE-Vefahren), welches in [FP02], Kapitel 7.3.4, Seite 188ff. hergeleitet wird. Ein entsprechender Strömungslöser ist im Internet unter [Per11] verfügbar und wurde zur Lösung der Nebenbedingung des Optimierungsproblems verwendet. Das Gebiet wurde dazu in $64 \times 64$ finite Volumen eingeteilt. Abbildung 6.1 auf Seite 47 stellt die Gegebenheiten schematisch dar.

Durch die Wahl der Temperaturen an der rechten Wand wird die Strömung gesteuert. Das Fluid heizt sich an der rechten, wärmeren Wand auf und wird durch Auftrieb nach oben steigen. Links hingegen wird es gekühlt und fällt ab. So entsteht eine Strömung gegen den Uhrzeigersinn entlang der Wände. Für den Testfall wurde eine zu rekonstruierende Temperaturverteilung am rechten Rand entsprechend Abbildung 6.2 (Seite 47) gewählt. Abbildung 6.3 (Seite 48) zeigt das Residuum der primalen Zustandsgleichung während der Lösung der Gleichungen mit diesen Randbedingungen durch das SIMPLE-Verfahren. Die zugehörige Strömung (Temperatur und Geschwindigkeitsfeld im Inneren) nach Auskonvergieren des Lösungsverfahrens ist in Abbildung 6.4 (Seite 48) dargestellt. Man erkennt deutlich die kreisförmige Bewegung entlang des Randes sowie ein fast stagnierendes Gebiet in der Mitte der Box. Durch geeignete Wahl der zu minimierenden Zielfunktion soll dieses Referenzströmungsfeld im Zuge der Optimierung durch Variation der Temperatur am rechten Rand rekonstruiert werden.

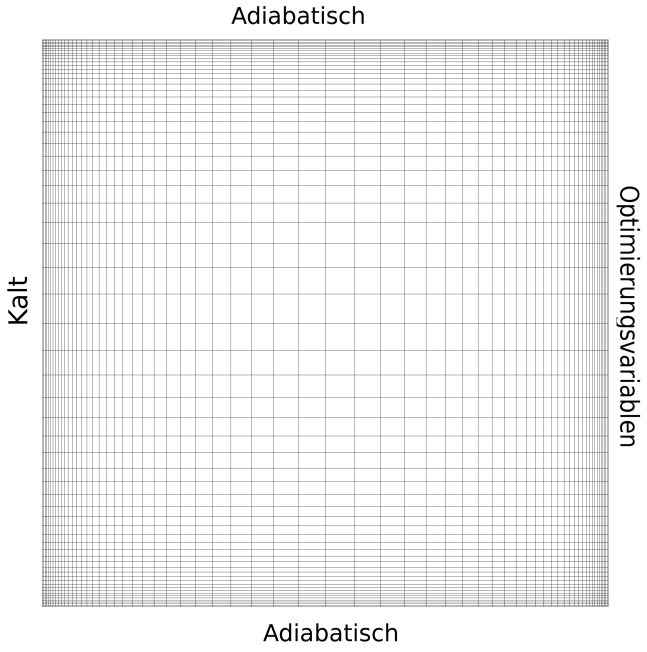


Abbildung 6.1: Diskretisiertes Gebiet (64x64 Elemente) und Randbedingungen

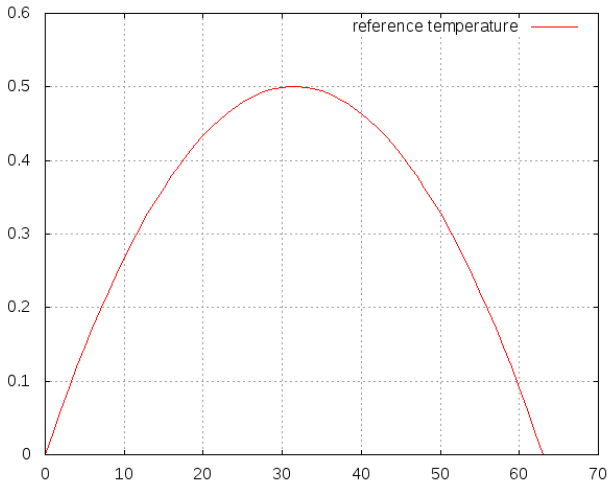


Abbildung 6.2: Temperaturverteilung am rechten Rand (64 Kontrollvariablen) zur Erzeugung einer Referenzströmung

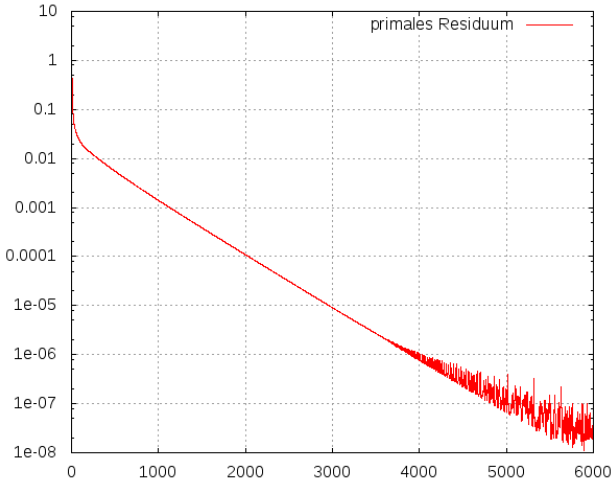


Abbildung 6.3: Residuum des Strömungslösers bei Lösung der inkompressiblen Navier-Stokes-Gleichungen mit den Randbedingungen aus Abbildungen 6.1 und 6.2

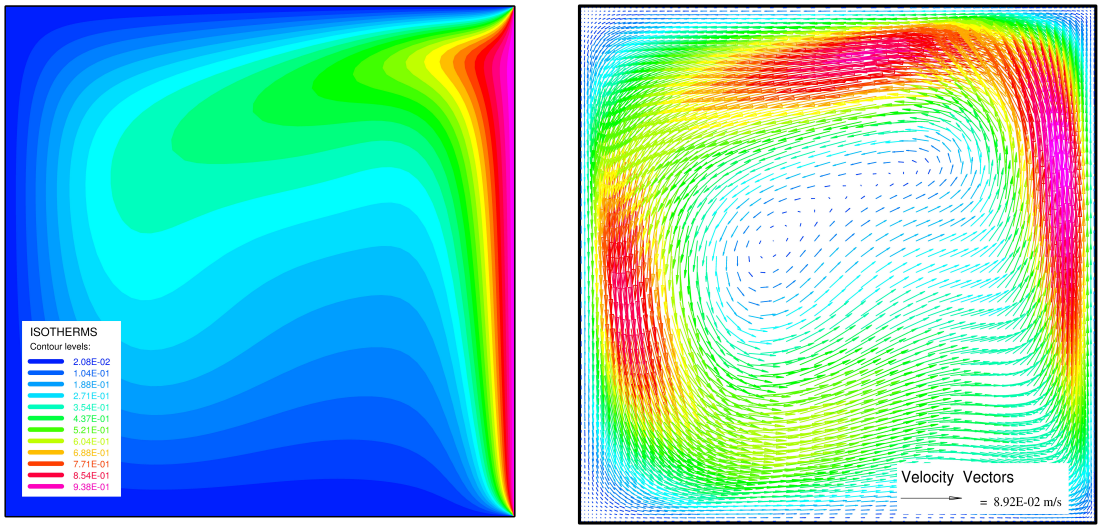


Abbildung 6.4: Referenz-Temperaturverteilung (links) und Referenz-Geschwindigkeitsfeld (rechts) im Inneren des Gebiets

Bezeichne $y = \left[v_1, v_2, p, T\right]^T$ den Vektor der Zustandvariablen, bestehend aus den Geschwindigkeiten $v_1, v_2$ in die Koordinatenrichtungen $x_1, x_2$, dem Druck $p$, sowie der Temperatur $T$ im Inneren der Box. Die Optimierungsvariablen, d.h. die Temperaturen am rechten Rand des Gebietes seien mit $u = [T_1, \ldots, T_{64}]^T$ bezeichnet. Zu Beginn der Optimierung wird $u_0 = 0$ gesetzt. Aufgrund der Schlechtgestelltheit des inversen Designproblems muss die Zielfunktion um einen Regularisierungsterm erweitert werden (siehe z.B. [Gro84]). Hier wurde eine Tikhonov-Regularisierung mit dem Parameter $\mu = 0.001$ gewählt. Unter Verwendung der diskreten 2-Norm lautet die zu minimierende Zielfunktion dann

$$f(y,u) := \|y - y_{Referenz}\|_2^2 + \mu \|u\|_2^2. \tag{6.1}$$

Dabei bezeichnet $y_{Referenz}$ das in Abbildung 6.4 dargestellte, durch die Temperaturverteilung aus Abbildung 6.2 erzeugte Strömungsfeld. Mittels Minimierung der Zielfunktion soll dieses Strömungsfeld durch Variation der Optimierungsvariablen $u$ rekonstruiert werden.

## 6.2 Implementatorische Aspekte

### 6.2.1 Automatisches Differenzieren (AD)

Zur Erzeugung der auftretenden Ableitungen wurde der gesamte Computercode zur Lösung der PDE mittels Automatischem Differenzieren abgeleitet. Dazu wurde das AD-Tool *TAPENADE* verwendet. Unter [oISA11] ist eine ausführliche Dokumentation der Software verfügbar, sowie ein sehr übersichtliches Tutorial zur Verwendung der Modi *tangent* und *reverse*. Hier soll kurz auf die Verwendung von *TAPENADE* zur Erzeugung eines adjungierten Codes, sowie zur Berechnung der ersten und zweiten Ableitungen eingegangen werden.

#### Erste Ableitungen

Analog zu den Bezeichnungen aus Kapitel 4 lässt sich die Routine zur Berechnung eines primalen Updates der Zustandsvariablen $y$ und anschließender Auswertung der Zielfunktion $f$ schematisch wie folgt darstellen:

**FUNCTION** *objective*
**input:** $y, u$
$y \leftarrow G(y,u)$
$f \leftarrow f(y,u)$
**output:** $y, f$

Die Notation „ $a \leftarrow b$“ bezeichne dabei den Ausdruck „ $a$ wird gesetzt auf $b$“. Wählt man also die aktuellen Variablen $y_k, u_k$ als Input, so berechnet die Routine *objective* sowohl das Update $y_{k+1} = G(y_k, u_k)$ als auch die Auswertung der Zielfunktion $f = f(y_{k+1}, u_k)$.

Durch automatisches Differenzieren der abhängigen Outputvaribalen $y$ und $f$ dieser Routine nach den unabhängigen Inputvariablen $y$ und $u$ im *reverse*-Modus von *TAPENADE* erhält man eine Routine *objective_b* der folgenden Art (vgl. [oISA11]):

**FUNCTION** *objective_b*

**input:** $yb, fb$

$$\begin{bmatrix} yb \\ ub \end{bmatrix} \leftarrow \underbrace{\begin{bmatrix} \frac{\partial G}{\partial y}^T & \frac{\partial f}{\partial y}^T \\ \frac{\partial G}{\partial u}^T & \frac{\partial f}{\partial u}^T \end{bmatrix}}_{=J^T} \times \begin{bmatrix} yb \\ fb \end{bmatrix}$$

**output:** $yb, ub$

Dabei stellt $J^T$ die transponierte Jakobimatrix der partiellen Abeitungen jeder abhängigen Variablen bezüglich jeder unabhängigen Variablen dar. Wählt man nun den Input $yb = \bar{y}_k$, die aktuelle adjungierte Variable, sowie $fb = 1.0$, so erhält man als Output von *objective_b* gerade den Gradienten des *shifted Lagrangian N* nach $y$ und $u$:

$$\begin{bmatrix} yb \\ ub \end{bmatrix} = \begin{bmatrix} G_y^T \bar{y}_k + f_y^T \\ G_u^T \bar{y}_k + f_u^T \end{bmatrix} = \begin{bmatrix} N_y^T(y_k, \bar{y}_k, u_k) \\ N_u^T(y_k, \bar{y}_k, u_k) \end{bmatrix}.$$

Diese Routine kann also genutzt werden, um gleichzeitig ein Update der adjungierten Variable $\bar{y}_{k+1} = N_y^T(y_k, \bar{y}_k, u_k)$ durchzuführen und den reduzierten Gradienten $N_u^T(y_k, \bar{y}_k, u_k)$ zu berechnen.

Da beide Ableitungen simultan berechnet und somit an den gleichen Argumenten ausgewertet werden, ist es für die Implementierung der One-Shot-Schritte aus einem rSQP-Ansatz, wie in Kapitel 3 vorgestellt, notwendig, die Routine *objective_b* nach dem ersten Aufruf und Update der adjungierten Variable $\bar{y}_{k+1}$ ein zweites Mal aufzurufen, um den reduzierten Gradienten an dieser neuen Variable auszuwerten: $N_u^T(y_k, \bar{y}_{k+1}, u_k)$.

Zu Verifizierung des mittels Automatischen Differenzieren berechneten reduzierten Gradienten $N_u$ sei Abbildung 6.5 betrachtet, welche den Vergleich von $N_u$ mit der über Finite Differenzen approximierten totalen Ableitung der reduzierten Zielfunktion $f(y(u), u)$ nach $u$ zeigt. Diese wird für ein kleines $\varepsilon > 0$ berechnet als

$$\frac{f(y(u+\varepsilon), u+\varepsilon) - f(y,u)}{\varepsilon}. \tag{6.2}$$

Die durch AD berechnete Ableitung $N_u$ wurde einerseits mit der aus Kapitel 4.3 beschriebenen Piggy-Back-Iteration erzeugt, andererseits über einen Black-Box-Ansatz, bei dem zuerst die primale Iteration und im Anschluss daran die adjungierte Iteration auskonvergiert wird. Abbildung 6.5 zeigt die erwartete Übereinstimmung dieser Ansätze, sowie die Verifizierung durch den Finite-Differenzen-Ansatz.

### Gradient der erweiterten Lagrangefunktion und zweite Ableitungen

Zur Präkonditionierung des Designupdates aus der One-Shot-Methode nach Griewank et al. (Kapitel 4) ist die Berechnung des Gradienten der erweiterten Lagrangefunktion $L^a$ nach $u$ not-

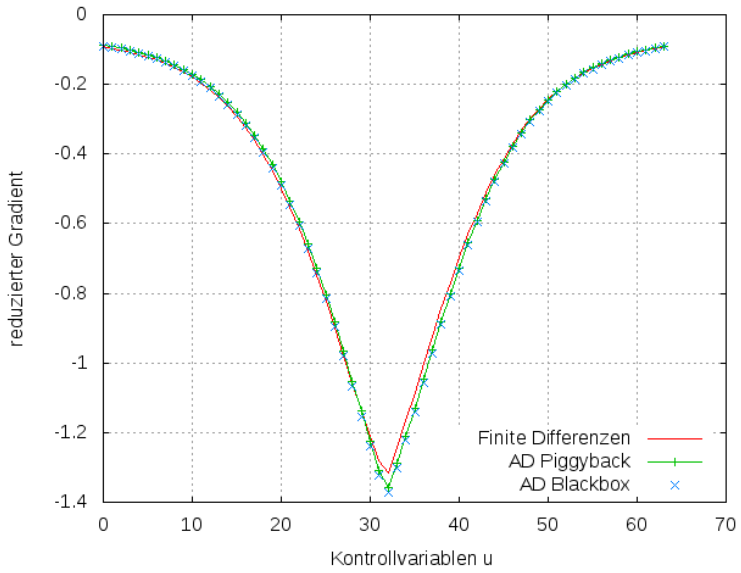


Abbildung 6.5: Verifizierung des durch Automatisches Differenzieren erzeugten Gradienten im Piggyback- und Blackbox -Ansatz mittels Finite Differenzen

wendig. Dieser ist gegeben durch

$$\nabla_u L^a(y_k, \bar{y}_k, u_k) = \alpha \Delta y_k^T G_u(y_k, u_k) + \beta \Delta \bar{y}_k^T N_{yu}(y_k, \bar{y}_k, u_k) + N_u(y_k, \bar{y}_k, u_k). \tag{6.3}$$

Dabei ist wieder $\Delta y_k = y_{k+1} - y_k$ und $\Delta \bar{y}_k = \bar{y}_{k+1} - \bar{y}_k$. Der letzte Summand ist genau der reduzierte Gradient, seine Berechnung wurde schon im vorhergehenden Abschnitt beschrieben. Der erste Summand kann durch die Wahl geeigneter Inputvariablen ebenfalls mittels der oben beschriebenen Routine *objective_b* berechnet werden. Setzt man vor dem Aufruf der Routine $yb = \alpha \Delta y_k$ und $fb = 0.0$, so liefert sie als Outputvariable gerade den gesuchten Term $ub = \alpha \Delta y_k^T G_u$.

Es verbleibt der mittlere Summand, der die zweite Ableitung des shifted Lagrangian $N_{yu}$ enthält. Zur Berechnung dieses Terms wird die Routine *objective* zweimal differenziert. Zunächst differenziert man die abhängigen Variablen $y$ und $f$ der Routine nach der unabhängigen Variable $y$ im *tangent*-Modus von *TAPENADE* (vergleiche wieder [oISA11]). Man erhält eine Routine der Form

**FUNCTION** *objective_d*

**input:** $yd$

$$\begin{bmatrix} yd \\ fd \end{bmatrix} \leftarrow \underbrace{\begin{bmatrix} \frac{\partial G}{\partial y} \\ \frac{\partial f}{\partial y} \end{bmatrix}}_{=J} \times \begin{bmatrix} yd \end{bmatrix}$$

**output:** $yd, fd$

Dabei ist $J$ wiederum die Jakobimatrix der partiellen Ableitungen der abhängigen Variablen bezüglich allen unabhängigen Variablen. Wählt man die Inputvariable $yd = \beta \Delta \bar{y}$, so sieht man

im Output $yd = \beta G_y \Delta\bar{y}$ sowie $fd = \beta f_y \Delta\bar{y}$. Differenziert man anschließend diese abhängigen Variablen $yd$ und $fd$ nach der unabhängigen Variable $u$ im *reverse*-Modus, so erhält man die Routine zur Berechnung der zweiten Ableitung:

**FUNCTION** *objective_d_b*

**input:** $ydb, fdb$

$$ub \leftarrow \underbrace{\begin{bmatrix} \frac{\partial yd}{\partial u} & \frac{\partial fd}{\partial u} \end{bmatrix}}_{J^T} \times \begin{bmatrix} ydb \\ fdb \end{bmatrix} = \begin{bmatrix} \beta\Delta\bar{y}^T G_{yu} & \beta\Delta\bar{y}^T f_{yu} \end{bmatrix} \times \begin{bmatrix} ydb \\ fdb \end{bmatrix}$$

**output:** $ub$

Zur Berechnung des gesuchten Terms setzt man also die Inputvariable $ydb = \bar{y}$, sowie $fdb = 1.0$. Dann liefert ein Aufruf der Routine *objective_d_b* genau $ub = \beta\Delta\bar{y}^T N_{yu}$.

### 6.2.2 BFGS-Updates

Statt der Berechnung der auftretenden Präkonditionierer $B_k$ werden in beiden Implementierungen der One-Shot-Iterationen BFGS-Updates nach [NW06], Kapitel 6.1, durchgeführt. Dabei wird ein positiv definites $B_{k+1}$ aus einem positiv definiten $B_k$ gewonnen. Die Sekantenbedingung

$$B_{k+1}\Delta u_k = R_k \tag{6.4}$$

kann dann nur erfüllt sein, wenn die sogenannte Krümmungsbedingung („curvature condition", siehe [NW06])

$$\Delta u_k^T R_k > 0 \tag{6.5}$$

erfüllt ist. Diese wird in jeder Iteration getestet und bei Nichterfülltheit wird das BFGS-Update durch setzen von $B_{k+1} = I$ übergangen.

Für die One-Shot-Methode nach Schulz et al. (Kapitel 3) wird

$$R_k = \gamma_k - \gamma_{k-1} \tag{6.6}$$

gewählt, wobei $\gamma_k$ den approximativen reduzierten Gradienten darstellt und somit $B_k \approx \tilde{T}^T H \tilde{T}$ gilt.

Für die One-Shot-Methode nach Griewank et al. (Kapitel 4) soll hingegen

$$R_k = \nabla_u L^a(y_k, \bar{y}_k, u_k + \Delta u_k) - \nabla_u L^a(y_k, \bar{y}_k, u_k) \tag{6.7}$$

gewählt werden (siehe Kapitel 4.4.4, Gleichung (4.42)), sodass $B_k \approx L^a_{uu}$ gilt. Dies erfordert allerdings die Auswertung des Gradienten der erweiterten Lagrangefunktion an zwei unterschiedlichen Stellen in jeder Optimierungsiteration. Da aber eigentlich $B_k \stackrel{!}{=} \alpha G_u^T G_u + \beta N_{yu}^T N_{yu} + N_{uu}$ nach der Definition eines geeigneten Präkonditionierers aus (4.39) gelten soll und an zulässigen

Punkten ($\Delta y_k = 0$ und $\Delta \bar{y}_k = 0$) schon $\alpha G_u^T G_u + \beta N_{yu}^T N_{yu} + N_{uu} = L_{uu}$ gilt, wird in der vorliegenden Implementierung die folgende Wahl der $R_k$ getroffen

$$R_k = \nabla_u L^a(y_k + \Delta y_k, \bar{y}_k + \Delta \bar{y}_k, u_k + \Delta u_k) - \nabla_u L^a(y_k, \bar{y}_k, u_k), \tag{6.8}$$

um diese zweifache Auswertung zu vermeiden.

Im Update der Designvariablen wird $B_k^{-1}$ benötigt. Anstatt $B_{k+1}$ mittels Updateformel zu berechnen und diese anschließend zu invertieren, kann das BFGS-Update nach [NW06] auch direkt auf die inverse Matrix $H_k = B_k^{-1}$ angewendet werden. Ist die Krümmungsbedingung erfüllt, so schreibt sich das BFGS-Update dann als

$$H_{k+1} = (I - r_k \Delta u_k R_k^T) H_k (I - r_k R_k \Delta u_k^T) + r_k \Delta u_k \Delta u_k^T \quad \text{mit} \quad r_k = \frac{1}{R_k^T \Delta u_k} \tag{6.9}$$

(vgl. [NW06], Kapitel 6.1, Gleichung (6.17)).

### 6.2.3 Algorithmen in Pseudocode

Die implementierten One-Shot-Algorithmen werden in diesem Kapitel schematisch in Pseudocode wiedergegeben. Über den Parameter $m$ der Schleife 2 bis 5 kann bei beiden Algorithmen entschieden werden, inwieweit ein initiales Konvergieren der primalen und adjungierten Variable statt finden soll.

---

**Algorithmus 1.** ***One-Shot-Implementierung nach Kapitel 3 (Schulz et al.)***

1: *initialize* $y_0, \bar{y}_0, u_0$
2: ***for*** $k = 0$ ***to*** $m$ ***do*** *{converge first primal and adjoint variable}*
3: *call function objective* $y_{k+1} = G(y_k, u_k)$
4: *call function objective_b* $\bar{y}_{k+1} = N_y(y_{k+1}, \bar{y}_k, u_k)$
5: ***end for***
6: ***while*** $\|N_u\| \geq \varepsilon$ ***do***
7: ***call function*** *objective:*

$$update \quad y_{k+1} \leftarrow G(y_k, u_k)$$

8: ***call function*** *objective_b:*

$$update \quad \bar{y}_{k+1} \leftarrow N_y(y_{k+1}, \bar{y}_k, u_k)$$

9: ***call function*** *objective_b:*

$$read\ gradient \quad N_u(y_{k+1}, \bar{y}_{k+1}, u_k)$$

10: $r_k \leftarrow N_u(y_{k+1}, \bar{y}_{k+1}, u_k) - N_u(y_k, \bar{y}_k, u_{k-1})$
11: $s_k \leftarrow u_k - u_{k-1}$
12: ***if*** $r_k^T s_k > 0$ ***then***

13: *BFGS-Update* $H_{k+1} = (I - r_k \Delta u_k R_k^T) H_k (I - r_k R_k \Delta u_k^T) + r_k \Delta u_k \Delta u_k^T \quad with \quad r_k = \frac{1}{R_k^T \Delta u_k}$

14: ***else***

15: $H_k = I$

16: ***end if***

17: *update*

$$u_{k+1} = u_k - \tau H_{k+1} N_u(y_{k+1}, \bar{y}_{k+1}, u_k)$$

18: ***end while***

---

**Algorithmus 2.** ***One-Shot-Implementierung nach Kapitel 4 (Griewank et al.)***

1: *initialize* $y_0, \bar{y}_0, u_0$

2: ***for*** $k = 0$ ***to*** $m$ ***do*** *{converge first primal and adjoint variable}*

3: *call function objective* $y_{k+1} = G(y_k, u_k)$

4: *call function objective_b* $\bar{y}_{k+1} = N_y(y_{k+1}, \bar{y}_k, u_k)$

5: ***end for***

6: ***while*** $\|N_u\| \geq \varepsilon$ ***do***

7: ***call function*** *objective:*

$$compute \quad \Delta y_k \leftarrow y_{k+1} - y_k = G(y_k, u_k) - y_k$$

8: ***call function*** *objective_b:*

$$\begin{aligned} &(a)\ compute \quad \Delta \bar{y}_k \leftarrow \bar{y}_{k+1} - \bar{y}_k = N_y(y_k, \bar{y}_k, u_k) - \bar{y}_k \\ &(b)\ read\ gradient \quad N_u(y_k, \bar{y}_k, u_k) \end{aligned}$$

9: *compute* $\nabla_u L^a(y_k, \bar{y}_k, u_k)$

10: $r_k \leftarrow \nabla_u L^a(y_k, \bar{y}_k, u_k) - \nabla_u L^a(y_{k-1}, \bar{y}_{k-1}, u_{k-1})$

11: $s_k \leftarrow u_k - u_{k-1}$

12: ***if*** $r_k^T s_k > 0$ ***then***

13: *BFGS-Update* $H_{k+1} = (I - r_k \Delta u_k R_k^T) H_k (I - r_k R_k \Delta u_k^T) + r_k \Delta u_k \Delta u_k^T \quad with \quad r_k = \frac{1}{R_k^T \Delta u_k}$

14: ***else***

15: $H_k = I$

16: ***end if***

17: *update*

$$\begin{aligned} y_{k+1} &= y_k + \Delta y_k \\ \bar{y}_{k+1} &= y_k + \Delta \bar{y}_k \\ u_{k+1} &= u_k - \tau H_{k+1} N_u(y_k, \bar{y}_k, u_k) \end{aligned}$$

18: ***end while***

---

## 6.3 Vergleich der Iterationsprozesse

Um den numerischen Vergleich der beiden One-Shot-Verfahren durchzuführen, wurde für beide Verfahren das einheitliche Abbruchkriterium $\|N_u\| < \varepsilon$ implementiert, wobei $\varepsilon > 0$ eine wählbare Konstante darstellt. In der vorliegenden Implementierung ist $\varepsilon = 0.00007$ gewählt. Des Weiteren wurde eine konstante Schrittweite von $\tau = 0.0008$ für beide Verfahren gewählt, um die zusätzlich benötigten Funktionsauswertungen einer Liniensuche zu vermeiden. Die konkrete Wahl der Gewichte $\alpha$ und $\beta$ der erweiterten Lagrangefunktion aus dem Verfahren nach Kapitel 4 erfolgte ebenfalls konstant (vgl. [GÖ09]) mit $\alpha = 100$ und $\beta = 2$.

Den lokalen Konvergenzeigenschaften eines rSQP-Verfahrens entsprechend wurde vor Beginn der Optimierungsiteration bei beiden Verfahren mittels $m = 1500$ die primale und die adjungierte Variable vorkonvergiert.

Abbildung 6.6 zeigt sowohl die Zielfunktionswerte $f(y_k, u_k)$ beider One-Shot-Verfahren während der Optimierung, als auch die Differenz des aktuellen Strömungsfeldes zum Referenzströmungsfeld in der diskreten 2-Norm ($||y - y_{Referenz}||_2^2$) in logarithmischer Skalierung. Der anfängliche Zielfunktionswert von 1.7174 wird von beiden Verfahren bis zu dem optimierten Zielfunktionswert von 0.008126 minimiert. Der Wert $||y - y_{Referenz}||_2^2$ beläuft sich bei beiden Verfahren auf etwa 0.00023. Die Steuerungen am rechten Rand des Gebiets nach der Optimierung sind in Abbildung 6.7 dargestellt.

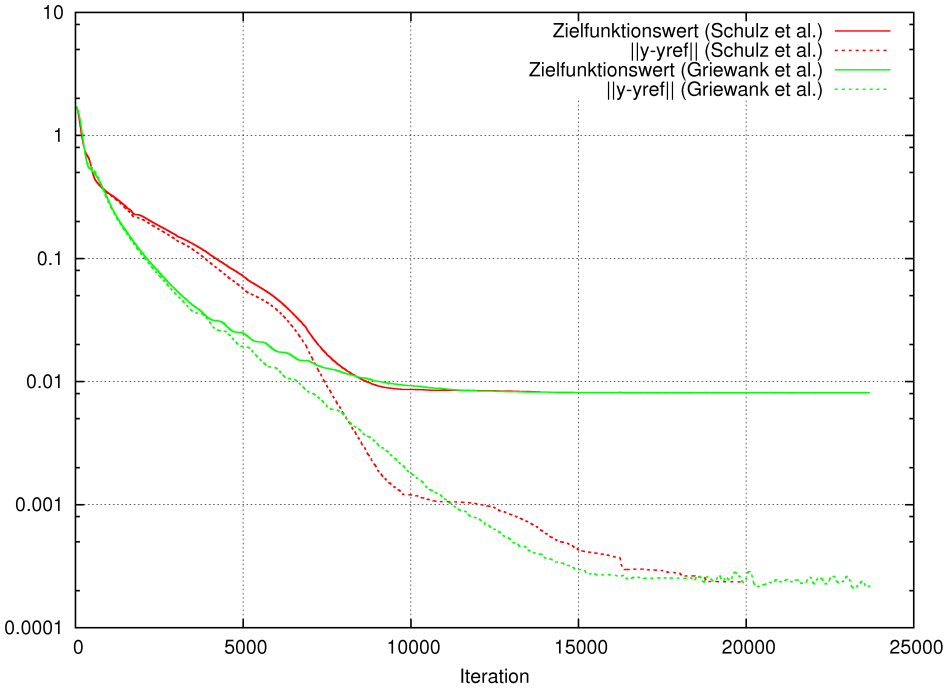


Abbildung 6.6: Vergleich der Zielfunktionswerte $f(y_k, u_k)$ sowie $||y - y_{Referenz}||_2^2$ der beiden One-Shot-Methoden während des Optimierungsprozesses

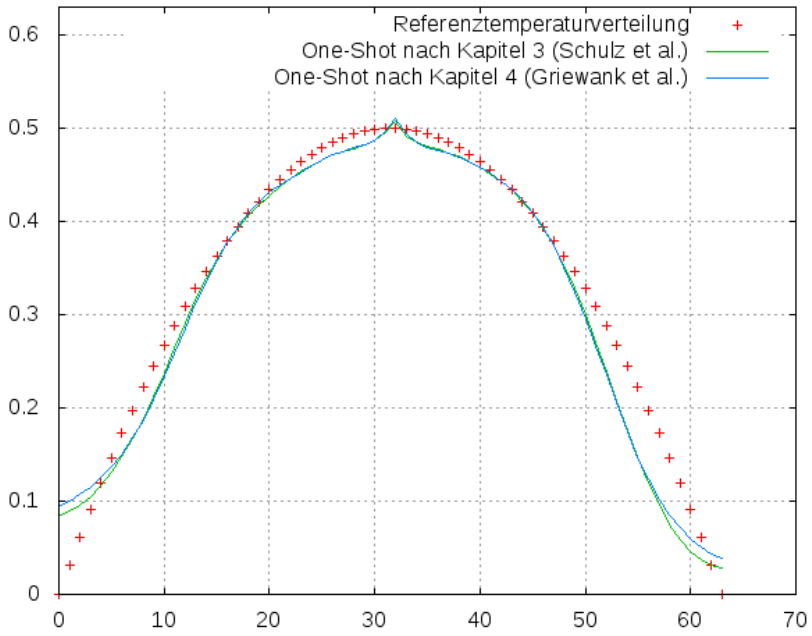


Abbildung 6.7: Vergleich der optimierten Steuerungen der One-Shot-Methoden mit der Referenztemperatur an der rechten Wand des Gebietes

Ein signifikanter Unterschied im Iterationsprozess ist kaum zu erkennen. Während die Zielfunktion der Methode nach Griewank et al. zwar anfangs etwas niedriger ist, erreicht die Methode nach Schulz et al. das Minimum letztlich etwas schneller. Das Abbruchkriterium ist für die Methode nach Schulz et al. bei der Iteration $i = 19439$ erfüllt. Das primale Residuum beläuft sich bei dieser Iteration auf $1.8e-5$. Dieses Level wird nach Abbildung 6.3 für die reine Simulation der Strömung etwa bei $i = 2700$ erreicht. Daraus ergibt sich ein Faktor der Optimierung relativ zur Simulation von etwa 7.2. Für das Verfahren nach Griewank et al. endet die Iteration bei $i = 23686$. Das primale Residuum erreicht dort $9.3e-6$. Die reine Simulation erreicht dieses Level etwa bei $i = 3000$, woraus sich ein Faktor von 7.9 errechnet.

Letzlich sei erwähnt, dass das Verfahren nach Schulz et al. in diesem speziellen Testfall auch für nicht vorkonvergierte primale und duale Startpunkte (d.h. $m = 0$) gegen den obigen Optimalpunkt konvergiert, obwohl die Konvergenztheorie dieses Verfahren nur lokale Konvergenzeigenschaften liefert.

# 7 Zusammenfassung

In der vorliegenden Arbeit wurden zwei One-Shot-Verfahren zur simultanen Optimierung unter partiellen Differentialgleichungen vorgestellt und verglichen. Die erste betrachtete Methode nach Schulz et al. basiert auf einem reduzierten SQP-Verfahren. Die Jacobimatrix der Nebenbedingungen und ihre Transponierte werden dabei approximativ dargestellt im Sinne eines Defektkorrektur-Verfahrens zur Lösung der primalen Zustandsgleichung und der kontinuierlichen Adjungiertengleichung. Nach jedem Update der Designvariable erfolgt je ein Schritt zur Lösung dieser Gleichungen. Für linear-quadratische Minimierungsprobleme ist die Konvergenz des Verfahrens bewiesen. Die zweite Methode wurde entwickelt von Griewank et al . Dabei wird durch die Verwendung von Automatischem Differenzieren auf die Lösungsiteration der diskretisierten Zustandsgleichung eine konsistente Iteration zur Lösung der Adjungiertengleichung automatisch erzeugt. Die primalen und adjungierten Variablen können dadurch in der Piggyback-Iteration simultan iteriert werden. Um ein Update der Designvariablen in die Piggyback-Iteration zu integrieren, wird ein geeigneter Präkonditionierer des Designraumes hergeleitet, der die Konvergenz der gesamten Iteration sichert.

Der entscheidende Unterschied der Verfahren liegt im Zeitpunkt der Verwendung der neusten verfügbaren Variableninformationen. Da das Verfahren nach Griewank et al. auf Automatischem Differenzieren basiert, bei dem das Update der adjungierten Variable sowie die Berechnung des reduzierten Gradienten durch einen Aufruf des mittels AD erhaltenen Algorithmus berechnet wird, wird bei dieser Methode das Update aller Variablen in einer Optimierungsiteration aufgrund der Informationen eines gemeinsamen Punktes ausgeführt. So wird das Update für den Punkt $(y_{k+1}, \bar{y}_{k+1}, u_{k+1})$ aus den Informationen an $(y_k, \bar{y}_k, u_k)$ errechnet, obwohl z.B. nach dem Update der adjungierten Variable schon die neuere Information $\bar{y}_{k+1}$ zur Berechnung des approximativen reduzierten Gradienten an dieser neuen Stelle vorhanden wäre. Dies würde allerdings einen erneuten Aufruf des mittels AD erzeugten Algorithmus verlangen und wird aus diesem Grund nicht sofort, sondern erst in der nächsten Optimierungsiteration verwendet. Die auf einem rSQP-Verfahren basierende One-Shot-Methode hingegen geht von der kontinuierlichen Adjungiertengleichung aus, für welche ein iteratives Lösungsverfahren vorhanden sein soll. Die Berechnung des reduzierten Gradienten soll anschließend durch Einsetzen der entsprechenden Adjungiertenvariable erfolgen, sodass die Trennung eines Adjungiertenupdates und der Berechnung des reduzierten Gradienten möglich ist. In einer Optimierungsiteration kann dementsprechend nach dem Update der adjungierten Variable sogleich der reduzierte Gradient an dieser neuen Stelle ausgewertet und für das Update der Designvariable verwendet werden.

Aus diesem zentralen Unterschied in der Verwendung der neusten Informationen leiten sich die unterschiedlichen Präkonditionierer des Designraumes der jeweiligen Verfahren ab. Die One-Shot-Methode nach Schulz et al. verwendet entsprechend einem rSQP-Verfahren die konsisten-

te reduzierte Hessematrix zur Präkonditionierung des Designupdates. Für linear-quadratische Probleme ist die Konvergenz des Verfahrens bewiesen, sodass lokale Konvergenz des One-Shot-Verfahrens zu erwarten ist. Um die Konvergenz der One-Shot-Methode nach Griewank et al. zu sichern, werden Bedingungen an den Präkonditionierer aus der Suche nach Abstiegsrichtungen der zweifach erweiterten Lagrangefunktion hergeleitet. Die globale Konvergenz ist unter Verwendung einer Liniensuche bewiesen.

Aufgrund der lokal superlinearen Konvergenz eines rSQP-Verfahrens liegt die Vermutung nahe, dass die auf diesem Verfahren basierende One-Shot-Methode gute lokale Konvergenzeigenschaften aufweist, welche denen der One-Shot-Methode basierend auf AD überlegen sein könnten. Diese Vermutung konnte durch den hier betrachteten Testfall nicht bestätigt werden. Dieser bestand aus einem inversen Designproblem, bei dem eine Referenzströmung im Inneren einer zweidimensionalen Box durch Variation der Temperaturverteilung am rechten Rand reproduziert werden sollte. Die beobachteten Zielfunktionswerte während der Optimierung unterscheiden sich nur geringfügig. Das auf der rSQP-Methode basierende One-Shot-Verfahren erreicht dort den Faktor 7.2 der Optimierung relativ zur Simulation. Für das Verfahren basierend auf AD und der erweiterten Lagrangefunktion wird ein Faktor von 7.9 erreicht.

# Literaturverzeichnis


[BF95] M. Barner and F. Flohr. *Analysis 2*. Walter de Gruyter, 3rd edition, 1995.

[CDF+94] E.J. Cramer, J.E. Dennis, P.D. Frank, R.M. Lewis, and G.R. Shubin. On alternative problem formulations for multidisciplinary design optimization. *SIAM Journal on Optimization*, 4:754–776, 1994.

[CÖGT10] A. Carnarius, E. Özkaya, N. Gauger, and F. Thiele. Adjoint approaches for optimal flow control. *AIAA Paper 5088-2010*, 2010.

[FP02] J.H. Ferziger and M. Peric. *Computational Methods for Fluid Dynamics*. Springer-Verlag Berlin Heidelberg, 3rd edition, 2002.

[FS92] P.D. Frank and G.R. Shubin. A comparison of optimization-based approaches for a model computational aerodynamic design problem. *Journal of Computational Physics*, 98:74–89, 1992.

[Gau03] N. R. Gauger. *Das Adjungiertenverfahren in der aerodynamischen Formoptimierung*. PhD thesis, Technische Universität Braunschweig, 2003.

[GF02] A. Griewank and C. Faure. Reduced functions, gradients and hessians from fixed-point iterations for state equations. *Numerical Algorithms*, 30:113–139, 2002.

[GGR08] N. Gauger, A. Griewank, and J. Riehme. Extension of fixed point pde solvers for optimal design by one-shot method - with first applications to aerodynamic shape optimization. *European Journal of Computational Mechanics (REMN)*, 17:87–102, 2008.

[GH10] A. Griewank and A. Hamdi. Properties of an augmented lagrangian for design optimization. *Optimization Methods and Software*, 25(4):645–664, 2010.

[GH11] A. Griewank and A. Hamdi. Reduced quasi-newton method for simultaneous design and optimization. *Computational Optimization and Applications*, 49(3):521–548, 2011.

[Ghe07] I. Gherman. *Approximate Partially Reduced SQP Approaches for Aerodynamic Shape Optimization Problems*. PhD thesis, Universität Trier, 2007.

[GK05] A. Griewank and D. Kressner. Time-lag in derivative convergence for fixed point iterations. *ARIMA*, pages 87–102, 2005.

[GÖ09] N. Gauger and E. Özkaya. Single-step one-shot aerodynamic shape optimization. *International Series of Numerical Mathematics*, 158:191–204, 2009.

[GP00] M. B. Giles and N. A. Pierce. An introduction to the adjoint approach to design. *Flow, Turbulence and Combustion*, 65:393–415, 2000.

[GP08] K.C. Giannakoglou and D.I. Papadimitriou. Adjoint methods for shape optimization. In D. Thévenin and G. Janiga, editors, *Optimization and Computational Fluid Dynamics*, chapter 4, pages 79–108. Springer-Verlag Berlin-Heidelberg, 2008.

[Gri00] A. Griewank. *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*. Society for Industrial and Applied Mathematics, 2000.

[Gri03] A. Griewank. A mathematical view of automatic differentiation. *Acta Numerica*, 12:312–398, 2003.

[Gri06] A. Griewank. Projected hessians for preconditioning in one-step one-shot design optimization. *Large Scale Nonlinear Optimization*, 83:151–171, 2006.

[Gro84] C. W. Groetsch. The theory of tikhonov regularization for fredholm equations of the first kind. In *Research Notes in Mathematics*, volume 105. Pitman Advanced Publishing Program, 1984.

[GS05] I. Gherman and V. Schulz. Preconditioning of one-shot pseudo-timestepping methods for shape optimization. *Proceedings in Applied Mathematics and Mechanics*, 5(1):741–742, 2005.

[Haz10] S.B. Hazra. *Large-Scale PDE-Constrained Optimization in Applications*, volume 49 of *Lecture notes in applied and computational mechanics*. Springer-Verlag Berlin Heidelberg, 2010.

[Her06] M. Hermann. *Numerische Mathematik*. Oldenbourg Wissenschaftsverlag GmbH, 2nd edition, 2006.

[HGK90] R.T. Haftka, Z. Gurdal, and M.P. Kamat. *Elements of structural optimization*. Kluwer Academic Publishers, 1990.

[HJ85] R.A. Horn and C.R. Johnson. *Matrix Analysis*. Cambridge University Press, 1985.

[HPUU09] M. Hinze, R. Pinnau, M. Ulbrich, and S. Ulbrich. *Optimization with PDE Constraints*. Springer Science+Business Media, 2009.

[HS04] S.B. Hazra and V. Schulz. Simultaneous pseudo-timestepping for pde-model based optimization problems. *BIT Numerical Mathematics*, 44:457–472, 2004.

[HS06] S.B. Hazra and V. Schulz. Simultaneous pseudo-timestepping for aerodynamic shape optimization problems with state constraints. *SIAM Journal on Scientific Computing*, 28(3):1078–1099, 2006.

[HSBG05] S.B. Hazra, V. Schulz, J. Brezillon, and N. Gauger. Aerodynamic shape optimization using simultaneous pseudo-timestepping. *Journal of Computational Physics*, 204:46–64, 2005.

[IKSG10] K. Ito, K. Kunisch, V. Schulz, and I. Gherman. Approximate nullspace iterations for kkt systems. *SIAM Journal on Matrix Analysis and Applications*, 31(4):1835–1847, 2010.

[Jam88] A. Jameson. aerodynamic design via control theory. *Journal on Scientific Computing*, 3:233–260, 1988.

[Lio71] J.L. Lions. *Optimal Control of Systems Governed by Partial Differential Equations*, volume 107 of *Die Grundlehren der mathematischen Wissenschaft*. Springer-Verlag Berlin Heidelberg New York, 1971.

[NJ00] S.K. Nadarajah and A. Jameson. A comparison of the continuous and discrete adjoint approach to automatic aerodynamic optimization. *AIAA Paper 2000-0667*, 2000.

[NW06] J. Nocedal and S.J. Wright. *Numerical Optimization*. Springer Science+Business Media, 2nd edition, 2006.

[oISA11] TROPICS (Research Team of INRIA Sophia-Antipolis). Tapenade 3.0, on-line automatic differentiation engine. http://www-sop.inria.fr/tropics/tapenade.html, September 2011. contact: tropics@sophia.inria.fr.

[Per11] M. Peric. Caffa - computer-aided fluid flow analysis, version 1.5. ftp://ftp.springer.de/pub/technik/peric/2dgl/sg/, September 2011. contact: peric@schiffbau.uni-hamburg.de.

[Pir73] O. Pironneau. On optimum profiles in stokes flow. *Journal of Fluid Mechanics*, 59:117–128, 1973.

[Pir74] O. Pironneau. On optimum design in fluid mechanics. *Journal of Fluid Mechanics*, 64:97–110, 1974.

[Pir82] O. Pironneau. *Optimal Shape Design for Elliptic Systems*. Springer-Verlag New York, 1982.

[RŮ04] M. Růžička. *Nichtlineare Funktionalanalysis*. Springer-Verlag Berlin Heidelberg, 2004.

[Sch96] V.H. Schulz. *Reduced SQP methods for large-scale optimal control problems in DAE with application to path planning problems for satellite mounted robots*. PhD thesis, Universität Heidelberg, 1996.

[Sch98] V. H. Schulz. Solving discretized optimization problems by partially reduced SQP methods. *Computing and Visualization in Science*, 1(2):83–96, 1998.

[TB88] I.-B. Tjoa and L. Biegler. Simultaneous solution and optimization strategies for parameter estimation of differential-algebraic systems. *Mathematics of Computation*, 51:181–202, 1988.

[TKS92] S. Ta'asan, G. Kuruvila, and M.D. Salas. Aerodynamic design and optimization in one shot. *30th Aerospace Sciences Meeting, Reno, NV, AIAA Paper 92-0025*, 1992.

[Uni11] RWTH Aachen University. Autodiff.org - community portal for automatic differentiation. http://www.autodiff.org/, September 2011.

[Wer09] D. Werner. *Einführung in die höhere Analysis*. Springer-Verlag Berlin Heidelberg, 2nd edition, 2009.