AG-Turbo Interimsphase, Projekt 1.110:

Ein mathematisches Verfahren zur schnellen Formoptimierung von Turbinenschaufeln

Schlußbericht

Thomas Dreyer, Volker Schulz, Hans Georg Bock

Dezember 1995

Interdisziplinäres Zentrum für Wissenschaftliches Rechnen

UNIVERSITÄT HEIDELBERG

Im Neuenheimer Feld 368

69120 Heidelberg

Berichtsblatt

1. ISBN	2. Berichtsart	3.

4. Titel des Berichts

Ein mathematisches Verfahren zur schnellen Formoptimierung von Turbinenschaufeln

5. Autor(en) (Name, Vorname(n))

Dr. Dreyer, Thomas; Schulz, Volker;
Prof. Dr. Bock, Hans Georg

6. Abschlußdatum des Vorhabens
31.08.1995

7. Veröffentlichungsdatum

8. Durchführende Institution(en) (Name, Adresse)

Interdisziplinäres Zentrum
für Wissenschaftliches Rechnen
der Universität Heidelberg
Im Neuenheimer Feld 368
D-69120 Heidelberg

9. Ber. Nr. Durchführende Institution

10. Förderkennzeichen
0326821 B

11. Seitenzahl
62

12. Literaturangaben
24

13. Fördernde Institution (Name, Adresse)

Bundesministerium für Bildung, Wissenschaft, Forschung und Technologie (BMBF)

53170 Bonn

14. Tabellen
7

15. Abbildungen
20

16. Zusätzliche Angaben

17. Vorgelegt bei (Titel, Ort, Datum)

18. Kurzfassung

Die optimale Auslegung von Strömungsmaschinen erfordert schnelle Algorithmen zur numerischen Optimierung von Parametern, die die Strömung beeinflussen. In vorliegendem Bericht wird ein Prototyp eines Optimierungsprogramms vorgestellt, das automatisch und schnell die optimale Auslegung von 2D-Turbinenschaufelprofilen ermöglicht. Hierzu wird ein nichtlineares Optimierungsproblem mit Nebenbedingungen formuliert und gelöst. Das zu minimierende Zielfunktional wird dadurch definiert, daß eine um das Profil vorgegebene Lavalzahlverteilung möglichst gut approximiert werden soll. Die Strömungsgleichungen – basierend auf einer Stromfunktionsmodellierung – und zusätzliche Restriktionen stellen die Nebenbedingungen dar.

Im Unterschied zu bisherigen Lösungsansätzen wurde eine simultane Optimierungsstrategie implementiert, die gleichzeitig mit der Lösung der Strömungsgleichungen die Optimierungsaufgabe löst. Als Optimierungsalgorithmus wurde ein spezielles partiell reduziertes SQP-Vefahren (Sukzessive Quadratische Programmierung) entwickelt. Dieses beruht darauf, das nichtlineare Optimierungsproblem iterativ durch die Lösung reduzierter quadratischer Teilprobleme zu lösen. Zur Lösung linearer Teilprobleme wurden spezielle Mehrgittermethoden entwickelt.

Die aus dem Simultanoptimierungsansatz resultierende drastische Rechenzeiteinsparung gegenüber bisher verwendeten einfachen Kopplungsansätzen (z.B. evolutionäre Algorithmen) wird in numerischen Tests dargelegt. Hierdurch wird eine Grundlage für die Entwicklung ähnlicher Algorithmen für verfeinerte Strömungsmodelle geschaffen.

19. Schlagwörter

Formoptimierung, Mehrgittermethoden

20. Verlag	21. Preis

Document Control Sheet

1. ISBN	2. Type of Report	3.

4. Report Title

A computational method for the fast shape optimization of turbine blades

5. Author(s) (Family Name, First Name(s))

Dr. Dreyer, Thomas; Schulz, Volker;
Prof. Dr. Bock, Hans Georg

6. End of Project

31.08.1995

7. Publication Date

8. Performing Organization(s) (Name, Address)

Interdisziplinäres Zentrum
für Wissenschaftliches Rechnen
der Universität Heidelberg
Im Neuenheimer Feld 368
D-69120 Heidelberg

9. Originator's Report No.

10. Reference No.

0326821 B

11. No. of Pages

62

12. No. of References

24

13. Sponsoring Agency (Name, Adress)

Bundesministerium für Bildung, Wissenschaft,
Forschung und Technologie (BMBF)

53170 Bonn

14. No. of Tables

7

15. No. of Figures

20

16. Supplementary Notes

17. Presented at (Title, Place, Date)

18. Abstract

Optimal turbomachinery design requires fast algorithms for the numerical optimization of design parameters that influence the flow in the turbomachine. In this report a prototype of an optimization algorithm is presented which can be used for the automatic and fast optimal design of 2D blade profiles. A nonlinear constrained optimization problem is formulated and solved. The objective functional to be minimized measures the deviation from a prescribed Laval number distribution around the blade profile. The flow equations—based on a stream function model—and additional restrictions define the constraints.

In contrast to other approaches a simultaneous optimization strategy is implemented which produces the solution of the flow problem and the solution of the optimization problem simultaneously and permits intermediate infeasibilities. The optimization algorithm developed is a special partially reduced SQP-method (Successive Quadratic Programming). It solves the nonlinear optimization problem iteratively solving reduced quadratic subproblems in each iteration. For the solution of arising linear subproblems special multigrid methods are developed.

The drastic savings in computation time resulting from the simultaneous optimization approach—compared with simple coupling strategies (e.g., evolutionary algorithms)—are demonstrated in numerical tests. Thus the ground is laid for the development of similar algorithms for refined flow models.

19. Keywords

shape optimization, multigrid methods

20. Publisher	21. Price

Inhaltsverzeichnis

Kapitel 1

Einleitung

1.1 Aufgabenstellung

Beim Entwurf der Beschaufelung einer Turbomaschine tritt folgende in sich abgeschlossene Teilaufgabe auf:

Als Ergebnis der – vorangegangenen – Meridionalströmungsauslegung liegen in der Ein- und Austrittsebene der Gitter die Zu- und Abströmbedingungen vor. Zu diesen Randbedingungen ist nun eine Schaufelform zu finden, die die aerodynamischen Anforderungen möglichst gut erfüllt.

Dabei ist die (dreidimensionale) Schaufel das Resultat einer radialen Auffädelung von zweidimensionalen Profilschnitten; gesucht sind also die optimalen Formen des jeweiligen Schaufelschnittes. Beurteilungsgrundlage ist dabei das Ergebnis der Berechnung der Strömung entlang der Schaufel. Die hohe Anzahl von Freiheitsgraden jeder Profilkontur macht eine automatische Optimierungsstrategie dringend erforderlich.

Die *Aufgabenstellung* des Projektes bestand in der Entwicklung eines Prototyps eines Optimierungsprogramms, das automatisch und schnell die optimale Auslegung von 2D-Turbinenschaufelprofilen ermöglicht.

Hierzu wurde in einem Simultanoptimierungsansatz ein nichtlineares Optimierungsproblem mit Nebenbedingungen formuliert und gelöst. Das zu minimierende Zielfunktional wurde dadurch definiert, daß eine um das Profil vorgegebene Lavalzahlverteilung möglichst gut approximiert werden sollte. Jedoch sollte der zu entwickelnde Code so allgemein gehalten werden, daß dieses Zielfunktional ohne größeren Mehraufwand durch ein geeignetes anderes ersetzt werden kann. Die Strömungsgleichungen und zusätzliche Geometrierestriktionen stellten die Nebenbedingungen dar. Es wurde der Unterschallfall und eine Strömungsmodellierung mit dem Stromfunktionsverfahren entsprechend [6] betrachtet.

Die so definierte Optimierungsaufgabe sollte mit einer speziellen reduzierten SQP-Methode (Sukzessive Quadratische Programmierung) gelöst werden.

Die Entwicklung des Optimierungsprogramms sollte aufbauen auf einem Simulationsmodul [19] der Firma MTU München, das mit Hilfe eines Mehrgitterverfahrens die diskretisierten Strömungsgleichungen löst.

1.2 Planung und Ablauf des Vorhabens

Das Projekt wurde im Zeitraum 01. Juli 1994 bis 31. August 1995 am Interdisziplinären Zentrum für Wissenschaftliches Rechnen (IWR) der Ruprecht-Karls-Universität Heidelberg durchgeführt. Es wurde barbeitet von Dr. Th. Dreyer. Die Projektleitung wurde von Prof. Dr. H.G. Bock und V. Schulz übernommen. Die beteiligten Projektpartner waren MTU München (Ansprechpartner: Dr. Speer) und ABB Baden (Ansprechpartner: Hr. Kappis). Das Projekt wurde erfolgreich abgeschlossen.

Der Projektplan umfaßte die Erfüllung der folgenden Teilaufgaben:

1) **Projektdefinition:** Es wurden die von MTU München erbrachten Vorleistungen [19] aufbereitet, darauf aufbauend das zu lösende Optimierungsproblem im Detail formuliert und Lösungsvorschläge für problematische Punkte erarbeitet. Als Ergebnis dieser Projektphase wurde ein Lastenheft erstellt, in dem das weitere mathematische und programmiertechnische Vorgehen detailliert dargelegt wurde.

   Für diese erste Projektphase waren im Projektantrag 2 Mannmonate eingeplant worden. Dies erwies sich im Projektverlauf als Fehleinschätzung. Es war ein erheblicher Mehraufwand nötig für

   - die Dokumentation und das Austesten der als Vorleistung gelieferten Software (ca. 6 Wochen),
   - eine Umformulierung der das Optimierungsproblem im Detail beschreibenden Gleichungen, so daß sie sich zur Verwendung in einer gradientenbasierten Optimierungsstrategie eignen (ca. 2 Wochen).

   Dadurch wurde eine Laufzeitverlängerung des Projektes um 2 Mannmonate nötig, die im Änderungsbescheid vom 11.04.95 vom Projektträger BMBF/BEO (Geschäftszeichen 413 - 4003 - 036821B) bewilligt wurde. Diese Laufzeitverlängerung war kostenneutral möglich.

2) **Implementation der Optimierungsmodule:** In dieser Phase wurden die wesentlichen Bausteine des Optimierungsprogramms entwickelt:

   - ein Mehrgitterverfahren für das linearisierte Vorwärtsproblem,
   - ein Mehrgitterverfahren für das adjungierte Problem,
   - ein RSQP-Optimierungsverfahren,
   - Module zur Behandlung der auftretenden zusätzlichen Geometriebedingungen.

   Die im Projektplan hierfür veranschlagten 2.5 Mannmonate reichten genau aus zur Bearbeitung der genannten Punkte.

3) **Integration und Validierung:** Hier wurden die in Phase 2) erstellten Module zu einem Optimierungspaket zusammengefügt. Es wurden verschiedene Programmvarianten getestet. Das Ergebnis ist unten in Kapitel 2 dargestellt. Eine wesentliche Einsicht war in dieser Projektphase, daß die zunächst im Lastenheft avisierte Behandlung der Geometriebedingungen durch Reduktion auf den Kern aller linearisierten Nebenbedingungen nicht zuverlässig funktionierte. Stattdessen wurden nun *Partiell Reduzierte SQP-Methoden* [17]

zum Einsatz gebracht, die eine stabile Behandlung der geometrischen Nebenbedingungen ermöglichten.

Die geplante Protoyp-Demo fand am 23.08.95 bei einem gemeinsamen Treffen mit den Projektpartnern statt.

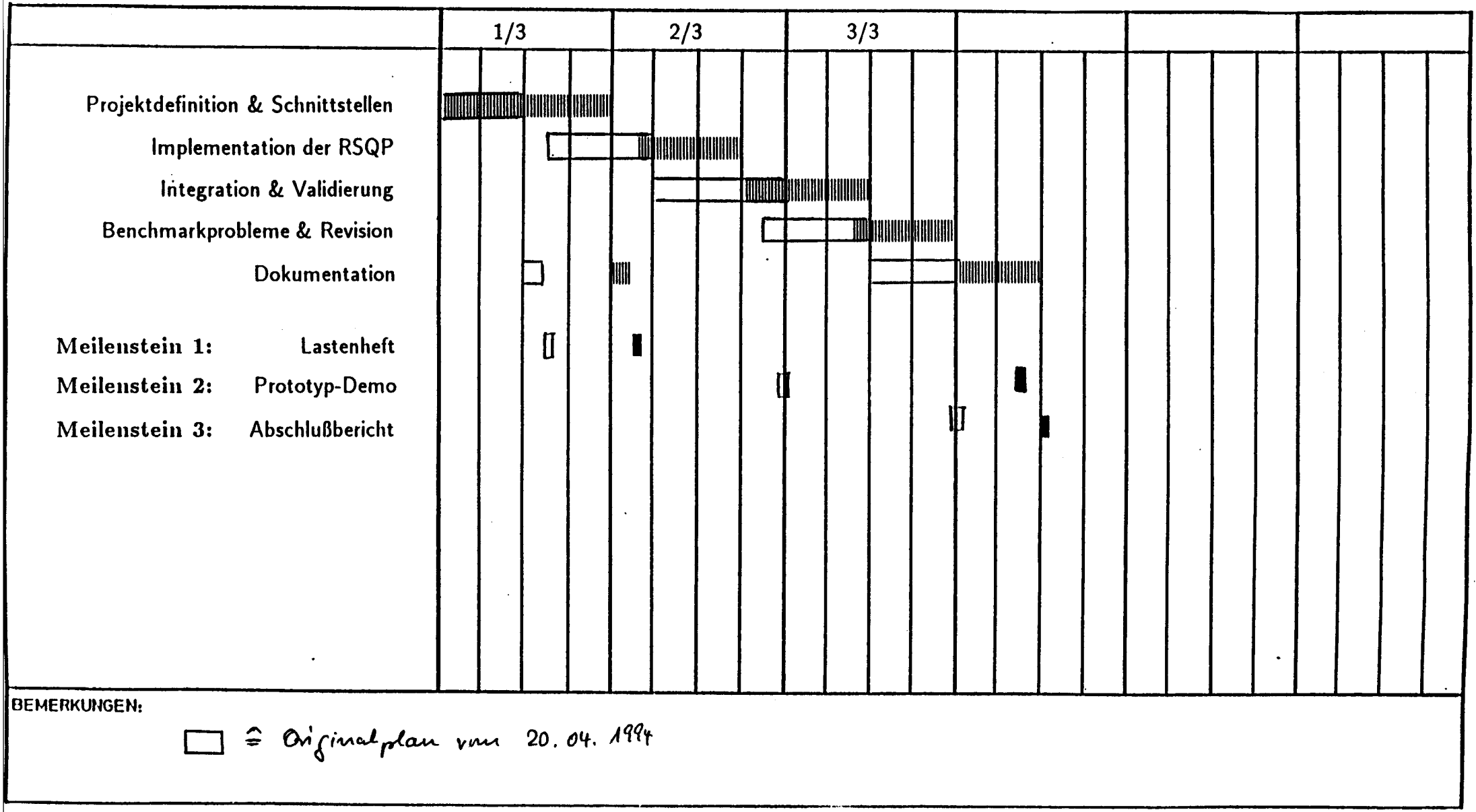
Die im Projektplan für diese Phase veranschlagten 3 Mannmonate reichten genau aus zur Bearbeitung der genannten Punkte.

4) **Benchmarkprobleme und Revision:** Es wurden realitätsnahe Beispielprobleme gerechnet und daran das Verhalten des entwickelten Optimierungscodes getestet. Die Ergebnisse sind in Abschnitt 2.7 dargestellt. Hierfür wurden genau die im Projektplan veranschlagten 2.5 Mannmonate benötigt.

5) **Dokumentation:** Die Dokumentation der Ergebnisse dieses Projektes fand in Form eines Zwischenberichts (Lastenheft) für die beteiligten Projektpartner und des vorliegenden Abschlußberichts statt, der die Ergebnisse des Zwischenberichts einschließt. Hierfür wurden die im Projektplan veranschlagten 2.5 Mannmonate aufgewendet.

Einen Überblick über den geplanten und tatsächlichen zeitlichen Ablauf des Projekts gibt der Balkenplan auf der folgenden Seite.

Es bleibt noch festzuhalten, daß die Zusammenarbeit mit Dr. Speer als Ansprechpartner bei MTU München und Hr. Kappis als Ansprechpartner bei ABB Baden reibungslos funktionierte. Insbesondere unterstützte Dr. Speer den Fortgang des Projekts durch die bereitwillige Beantwortung etlicher Fragen zur Implementierung des MTU-Simulationscodes.


1/3
2/3
3/3
Projektdefinition & Schnittstellen
Implementation der RSQP
Integration & Validierung
Benchmarkprobleme & Revision
Dokumentation
Meilenstein 1: Lastenheft
Meilenstein 2: Prototyp-Demo
Meilenstein 3: Abschlußbericht
BEMERKUNGEN:
≙ Originalplan vom 20.04.1994

Balkenplan

1.3 Stand der Technik zu Beginn des Vorhabens

Mathematisch gesehen handelt es sich bei der Aufgabenstellung des Projekts um ein Problem der Gestaltoptimierung (optimal shape design), bei dem eine nichtlineare partielle Differentialgleichung auf einem variablen Gebiet zu berechnen ist. Das Zielfunktional ist eine gewichtete Summe mehrerer von der Strömung (also der Lösung der Differentialgleichung) und der Profilform abhängender Beurteilungskritereien. Weiterhin treten Nebenbedingungen auf.

Nach Einführung von ca. 24 Designparametern, welche eine Klasse glatter Profile beschreiben, und der Diskretisierung der Differentialgleichung auf einem Rechennetz erhält man ein Problem der mehrdimensionalen Optimierung unter Nebenbedingungen.

Zur Lösung solcher Probleme ist die Methode der Sukzessiven Quadratischen Optimierung gut geeignet. Ausführliche Informationen dazu finden sich in den Aufsätzen von Stoer [21] und Schittkowski [16] und besonders für die Berücksichtigung strukturierter Hessematrizen in dem Artikel von Bock und Plitt [2].

Die Literatur zur Optimierung von Triebwerksschaufeln und Tragflügeln (einem benachbarten Gebiet) mit stochastischen oder deterministischen Verfahren ist nicht sehr umfangreich; dargestellt werden einige neuere Ansätze:

In mehreren Arbeiten, z. B. Pfeil [13], wurden bei MTU München Verdichter- und Turbinenschaufeln mit Hilfe einer Evolutionsstrategie (dazu siehe Rechenberg [14]) optimiert. Eine einfache Variante der Evolutionsmethode benutzten Gregg und Misegades [8] zur Optimierung eines Tragflügels.

Die Evolutionsmethode erlaubt eine nur sehr lose Kopplung zwischen Optimierungs- und Strömungsberechnungsprogramm. Sie erfordert allerdings eine immense Anzahl von Strömungsberechnungen (mehrere tausend) und ist daher für den beabsichtigten Einsatzzweck nicht geeignet.

In Zusammenarbeit mit MTU München wurden von Schwarz [18] am Lehrstuhl für Flugantriebe der TU München sehr unterschiedliche Optimierungsstrategien mit einem eindimensionalen Strömungsmodell für Radialverdichter gekoppelt. Da in der dort verwendeten Formulierung der Optimierungsaufgabe jedoch in jedem Iterationsschritt die Strömung erneut berechnet werden muß, kann diese Vorgehensweise auf das vorliegende zweidimensionale Problem nur unter ganz erheblicher Rechenzeitverlängerung übertragen werden.

Zur Optimierung von Triebwerksschaufeln benutzten Tong und Gregory [22] ein Programm von Vanderplaats [23], der sich seit 1975 mit diesem Gebiet befaßt. Es basiert auf einer Approximation zweiter Ordnung der Zielfunktion, welche durch Berechnung finiter Differenzen gewonnen wird, wozu sehr viele aerodynamische Berechnungen notwendig sind. Darüber hinaus berichten Tong und Gregory von Konvergenzschwierigkeiten und von starker Abhängigkeit der Methode von Startwerten und internen Parametern.

Einen anderen Ansatz verfolgen Orozco und Ghattas [12] bei der Optimierung von Tragflügeln: Die partielle Differentialgleichung wird gleichzeitig mit der Variation des Profils gelöst, so daß erst am Ende der Berechnung sowohl die beste Profilform als auch die dazu passende Strömung vorliegen. Dazu verwenden sie eine reduzierte SQP-Methode (sukzessiv Quadratische Optimierung), die bei ihnen wesentlich die symmetrische Struktur der dort auftretenden linearen Systeme nutzt. Außerdem wird die Variation des Gebietes, auf dem die Differentialgleichung definiert ist, in eine Variation der Koeffizienten der Differentialgleichung auf einem festen Rechtecksgebiet transformiert. Da die wiederholte Strömungsberechnung wegfällt, ist diese Methode

wesentlich schneller. Allerdings sind hierbei das SQP-Programm und das Strömungsberechnungsprogramm stärker miteinander verzahnt.

Für die Optimierung kann prinzipiell ein beliebiges geeignetes Strömungsmodell verwendet werden (Stromfunktions- oder Potentialverfahren, Euler- oder Navier-Stokes-Gleichungen). Da das Hauptgewicht des vorliegenden Vorhabens auf der Optimierung liegt, sei bezüglich des Strömungsmodells auf die Spezialliteratur verwiesen. Ein einfaches Modell für reibungsfreie Strömung auf einer Rotationsstromfläche stellt das von Fritsch [6] bei MTU München implementierte Stromfunktionsverfahren dar.

Kapitel 2

Ergebnisse des Forschungsvorhabens

2.1 Einführung

In diesem Kapitel wird ein Algorithmus zur Formoptimierung von Turbinenschaufeln vorgestellt und das dazu entwickelte Programmpaket dokumentiert.

Nach der von Wu in [24] dargestellten Methode ist es möglich, eine dreidimensionale Strömung zwischen Turbinenschaufeln (blade to blade Rechnung) auf zweidimensionale Strömungen auf sogenannten Stromflächen zurückzuführen, die in geeigneter Weise miteinander gekoppelt sind.

Man unterscheidet Stromflächen erster Art, die von einer Schaufel zur anderen verlaufen und die mit S_1 bezeichnet werden, von Stromflächen zweiter Art, die in radialer Richtung verlaufen und mit S_2 bezeichnet werden. Die vorliegende Arbeit befaßt sich mit dem Designproblem für S_1–Stromflächen. Dabei wird die Stromfläche als rotationssymmetrisch angenommen, man spricht von einer *Rotationsstromfläche.*

Zur Beschreibung eignen sich am besten Zylinderkoordinaten (r, φ, x), wobei r den Abstand eines Punktes P von der Achse der Turbomaschine angibt, φ den Azimutwinkel und x die Koordinate längs der Achse ist. Eine S_1–Stromfläche läßt sich allgemein beschreiben durch eine Gleichung

$$S_1(r, \varphi, x) = 0, \tag{1.1}$$

wodurch implizit r als Funktion von φ und x gegeben ist. Bei Rotationsstromflächen hängt r nur von x ab. Eine weitere Koordinate m, die sogenannte *Meridionalkoordinate*, mißt die Bogenlänge von Kurven auf der Stromfläche mit konstantem φ. Es ergibt sich für eine einfache Kegelgeometrie (vgl. Abbildung 2.1):

$$r(x) = r_0 + \alpha(x - x_0), \tag{1.2}$$

$$m(x) = m_0 + \sqrt{1 + \alpha^2}(x - x_0). \tag{1.3}$$

Modellierung Wir verwenden das von Fritsch in [6] dargestellte Strömungsmodell, das als Variablen die Stromfunktion u und die Dichte ρ benutzt. Für die Stromfunktion wird in [6] eine partielle Differentialgleichung zweiter Ordnung abgeleitet, für die Dichte wird eine thermodynamische Beziehung herangezogen, die für Durchströmungsprozesse durch Düsen gültig ist.

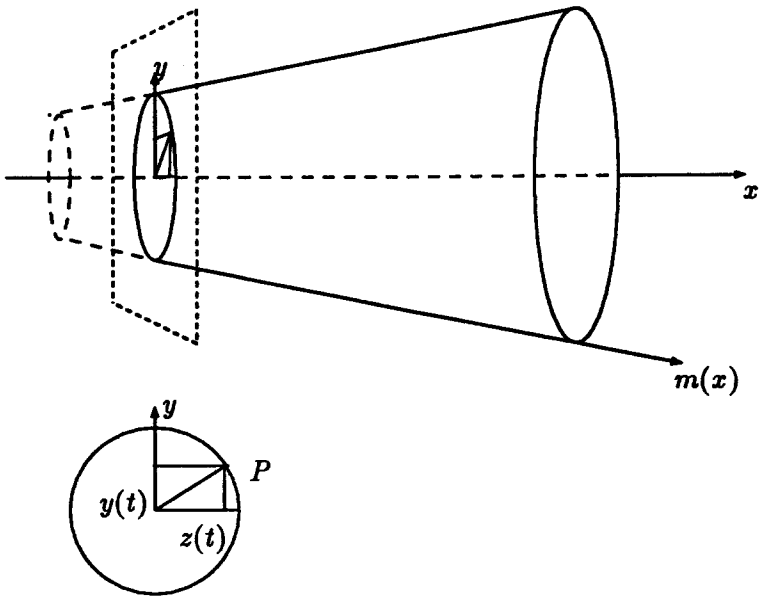


Abbildung 2.1: Kegel und Querschnitt

Da keine Ableitungen der Dichte auftreten, liegt ein System aus einer partiellen Differentialgleichung und einer algebraischen Gleichung vor. (Die Dichtegleichung enthält allerdings auch erste Ableitungen von u.)

Ein rechteckiges „Rechengebiet" mit einem kartesischen Gitter wird durch eine Transformation auf ein konturangepaßtes H–Netz zwischen den Turbinenschaufeln abgebildet. Das Schaufelprofil selbst wird durch eine Parametrisierung mit 2×12 Basis–Splines vom Grad 5 dargestellt. Für das Schaufelprofil sind die Lage des Leading Edge und die Länge (bezüglich der Meridionalkoordinate m) vorgeschrieben (Geometriebedingungen). Weitere geometrische Restriktionen wurden in dieser Version nicht gefordert.

Optimierung Die diskretisierten partiellen Differentialgleichungen sowie die Geometriebedingungen ergeben die Nebenbedingungen des Optimierungsproblems. Die Koeffizienten der Basis–Splines sind die Designparameter, die „eigentlichen" Freiheitsgrade des Optimierungsproblems. Als Zielfunktional wird eine Least–Squares–Approximation an eine vorgegebene Geschwindigkeitsverteilung entlang der Profilkonturen verwendet.

Zur Lösung des Optimierungsproblems wird ein PRSQP–Algorithmus formuliert (Partially Reduced Sequential Quadratic Programming), eine Methode, die von Schulz in [17] für ein Problem der optimalen Steuerung entwickelt wurde. Die darin auftretenden großen linearen Gleichungssysteme werden mit Mehrgitterverfahren approximativ gelöst. Die Systeme für die adjungierten Variablen entstehen durch Transposition aus den linearisierten Diskretisierungsgleichungen. Da diese transponierten Systeme nicht selbst als Diskretisierung einer partiellen Differentialgleichung verstanden werden können, mußten bestimmte Mehrgitterkomponenten, insbesondere die Gittertransferoperationen, neu konstruiert werden.

Ergebnisse Mit dem implementierten PRSQP–Mehrgitterverfahren läßt sich die zweidimensionale Formoptimierung von Turbinenschaufeln effizient vornehmen. An verschiedenen Beispielen wird gezeigt, daß der Algorithmus sowohl größere Profiländerungen bei Startdaten weit von der Lösung errechnet, als auch feine Korrekturen findet, die sich ergeben, wenn an der Lavalzahlverteilung nur bestimmte Teile geringfügig verändert werden.

2.2 Koordinatensysteme und Gitter

2.2.1 Die Koordinatensysteme

Von großer Bedeutung ist die Darstellung der vier verschiedenen verwendeten Koordinatensysteme (s,z), (ξ,η), (m,φ) und (x,y).
1) (s,z) bezeichnet das „Numerische Gitter", auf dem sämtliche Gleichungen diskretisiert werden.
2) (ξ,η) bezeichnet das „Čebyshev–Gitter", das lokale Verfeinerungen aufweist.
3) (m,φ) ist das „Physikalische Gitter", auf dem die kontinuierlichen Gleichungen zunächst formuliert werden.
4) (x,y) schließlich ist ein Koordinatensystem, in dem die Profilkurven für die Turbinenschaufeln in parametrisierter Form vorgelegt werden.
Zur Transformation der Gleichungen und zur Beschreibung der Abhängigkeit von den Einflußparametern, den B–Spline–Koeffizienten für das Schaufelprofil, werden nun die Beziehungen zwischen den verschiedenen Systemen inklusive deren erste Ableitungen genannt.

$F:(s,z)\mapsto(\xi,\eta)$: ξ hängt nur von s und η nur von z ab.
Für ein vorgegebenes $M\in\mathbb{N}$ (Anzahl der Gitterpunkte in der z- bzw. η-Koordinatenrichtung) definiere die Hilfspunkte

$$\tau_j := \frac{\pi}{2}\frac{2j-1}{M} \qquad \text{für } j=1,2,\ldots,M, \tag{2.1}$$

die das Intervall $(0,\pi)$ äquidistant unterteilen wie in Abbildung 2.2 gezeigt.

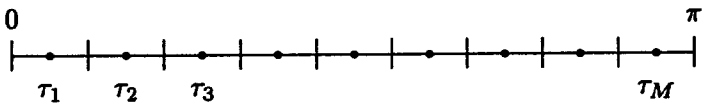


Abbildung 2.2: Die Punkte τ_j für $M=9$

Die äquidistanten Gitterpunkte

$$z_j := \frac{j-1}{M-1} \qquad \text{für } j=1,2,\ldots,M \tag{2.2}$$

werden vermöge der Funktion

$$\eta(z) := \frac{\cos\tau_1 - \cos(\tau_1 + z(\tau_M-\tau_1))}{\cos\tau_1 - \cos\tau_M} \tag{2.3}$$

auf die Čebyšev–Punkte

$$\eta_j := \eta(z_j) = \frac{\cos\tau_1 - \cos\tau_j}{\cos\tau_1 - \cos\tau_M} \in [0,1] \tag{2.4}$$

abgebildet, die im Intervall $[0,1]$ liegen, siehe Abbildung 2.3. Die beiden Randpunkte sind die Bilder von τ_1 und τ_M, so daß sich die Anzahl der Teilintervalle um eins vermindert. In Abbildung 2.3 wird $M=17$ gewählt, um die lokale Verdichtung der Gitterpunkte besser sichtbar zu machen. Es ist

$$\eta'(z) = \frac{\tau_M-\tau_1}{\cos\tau_1 - \cos\tau_M}\sin(\tau_1 + z(\tau_M-\tau_1)). \tag{2.5}$$

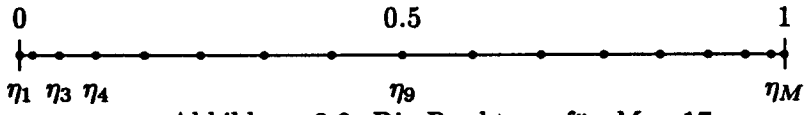


Abbildung 2.3: Die Punkte η_j für $M = 17$

Die Abbildung von s auf ξ wird mit Hilfe derselben Abbildung η vorgenommen, die aber nur auf Teilintervalle angewendet wird: Die Punkte $s = \frac{1}{4}$ und $s = \frac{3}{4}$ bleiben fest, d.h. werden auf $\xi = \frac{1}{4}$ und $\xi = \frac{3}{4}$ abgebildet. Das Intervall $[0, \frac{1}{4})$ enthält auf diesen Gittern den Vorlauf, das Intervall $[\frac{1}{4}, \frac{3}{4}]$ den Profilbereich und das Intervall $(\frac{3}{4}, 1]$ den Nachlauf. Vor– und Nachlauf entstehen jeweils durch Spiegelung des halben Profilgitters an den Punkten $\frac{1}{4}$ und $\frac{3}{4}$. Man erreicht damit eine Verdichtung der Gitterpunkte um das Leading Edge und um das Trailing Edge, vgl. Abbildung 2.4:

$$\begin{aligned}
\xi(s) &:= \tfrac{1}{4} + \tfrac{1}{2}\eta(2s - \tfrac{1}{2}) && \text{für } s \in [\tfrac{1}{4}, \tfrac{3}{4}], && (2.6)\\
\xi(s) &:= 2\xi(\tfrac{1}{4}) - \xi(\tfrac{1}{2} - s) && \text{für } s \in [0, \tfrac{1}{4}) && (2.7)\\
&= \tfrac{1}{4} - \tfrac{1}{2}\eta(\tfrac{1}{2} - 2s), && && (2.8)\\
\xi(s) &:= 2\xi(\tfrac{3}{4}) - \xi(\tfrac{3}{2} - s) && \text{für } s \in (\tfrac{3}{4}, 1] && (2.9)\\
&= \tfrac{5}{4} - \tfrac{1}{2}\eta(\tfrac{5}{2} - 2s). && && (2.10)
\end{aligned}$$

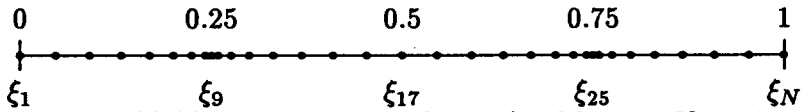


Abbildung 2.4: Die Punkte ξ_i für $M = 17$, $N = 33$

Für die Ableitungen erhält man:

$$\begin{aligned}
\xi'(s) &= \eta'(2s - \tfrac{1}{2}) && \text{für } s \in [\tfrac{1}{4}, \tfrac{3}{4}], && (2.11)\\
\xi'(s) &= \xi'(\tfrac{1}{2} - s) && \text{für } s \in [0, \tfrac{1}{4}) && (2.12)\\
&= \eta'(\tfrac{1}{2} - 2s), && && (2.13)\\
\xi'(s) &= \xi'(\tfrac{3}{2} - s) && \text{für } s \in (\tfrac{3}{4}, 1] && (2.14)\\
&= \eta'(\tfrac{5}{2} - 2s). && && (2.15)
\end{aligned}$$

Die Jacobimatrix von F ist diagonal:

$$DF = \begin{pmatrix} \frac{\partial \xi}{\partial s} & 0 \\ 0 & \frac{\partial \eta}{\partial z} \end{pmatrix} = \begin{pmatrix} \xi' & 0 \\ 0 & \eta' \end{pmatrix}. \qquad (2.16)$$

$G : (\xi, \eta) \mapsto (m, \varphi)$: m hängt nur von ξ ab, aber φ von ξ *und* η.
Das physikalische Modellierungsgebiet in meridionaler Richtung werde durch den kleinsten Wert m_W und durch den größten Wert m_O abgegrenzt. Die Indizes stehen dabei für Westen und Osten. Das ξ-Gitter wird linear auf das m–Intervall $[m_W, m_O]$ übertragen. Dabei spielt der Profilverlauf gar keine Rolle:

$$m(\xi) := (1 - \xi)m_W + \xi m_O \qquad \text{und daher} \qquad (2.17)$$

$$\frac{\partial m}{\partial \xi} = m'(\xi) = m_O - m_W, \tag{2.18}$$

$$\frac{\partial m}{\partial \eta} = 0. \tag{2.19}$$

Entsprechend wird auch das η-Gitter auf die Winkelkoordinate φ übertragen, aber nicht zwischen konstanten Werten φ_S und φ_N — Süden und Norden —, sondern zwischen ***Funktionen*** $m \mapsto \varphi_S(m), \varphi_N(m)$, die als Berandungskurven des physikalischen Gebiets dienen, siehe die Abbildung 2.5.

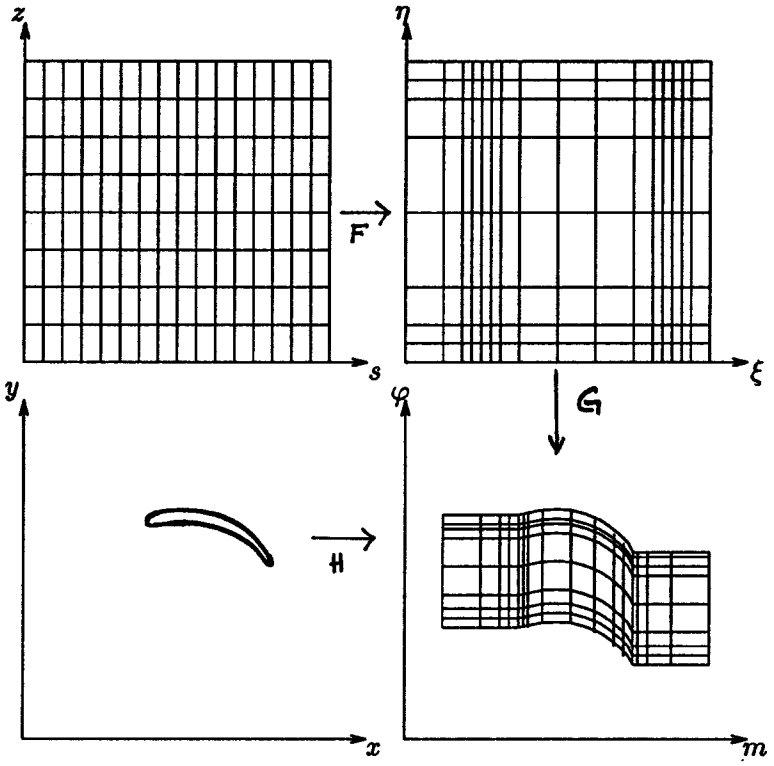


Abbildung 2.5: Die vier Koordinatensysteme des Stromfunktionsverfahrens

$$\varphi(\xi,\eta) = (1-\eta)\,\varphi_S(m(\xi)) + \eta\,\varphi_N(m(\xi)). \tag{2.20}$$

Für die Ableitungen erhält man:

$$\frac{\partial \varphi}{\partial \xi}(\xi,\eta) = \left[(1-\eta)\frac{\partial \varphi_S}{\partial m}(m(\xi)) + \eta\frac{\partial \varphi_N}{\partial m}(m(\xi))\right](m_O - m_W), \tag{2.21}$$

$$\frac{\partial \varphi}{\partial \eta}(\xi,\eta) = \varphi_N(m(\xi)) - \varphi_S(m(\xi)), \qquad \text{unabhängig von } \eta. \tag{2.22}$$

Die φ–Differenz in der zweiten Gleichung ist außerhalb des Profils immer konstant, gleich dem Quotienten aus 2π und der Anzahl der Schaufeln. Für die Hintereinanderausführung $G \circ F$:

$(s,z) \mapsto (m,\varphi)$ ergibt sich insgesamt die folgende Ableitung:

$$J := D(G \circ F) \equiv \begin{pmatrix} \frac{\partial m}{\partial s} & \frac{\partial m}{\partial z} \\ \frac{\partial \varphi}{\partial s} & \frac{\partial \varphi}{\partial z} \end{pmatrix} = \begin{pmatrix} \frac{\partial m}{\partial \xi}\frac{\partial \xi}{\partial s} & 0 \\ \frac{\partial \varphi}{\partial \xi}\frac{\partial \xi}{\partial s} & \frac{\partial \varphi}{\partial \eta}\frac{\partial \eta}{\partial z} \end{pmatrix}. \tag{2.23}$$

Zur Definition der Berandungskurven φ_S und φ_N wird das vierte Koordinatensystem (x,y), in dem das Profil gegeben ist, benötigt. Und *nur da* wird es gebraucht. Auf folgende Weise hängt es mit (m,φ) zusammen: Zunächst wird eine Radiusfunktion $x \mapsto r(x)$ eingeführt, im einfachsten Fall linear (für kegelförmige Stromflächen),

$$r(x) := r_0 + r_1 x, \qquad r'(x) = r_1. \tag{2.24}$$

Damit wird festgelegt: $H : (x,y) \mapsto (m,\varphi)$: m hängt nur von x ab, aber φ von x *und* y.

$$m(x) = \sqrt{1 + (r'(x))^2}\, x = \sqrt{1 + r_1^2}\, x, \tag{2.25}$$

$$\varphi(x,y) = \arcsin \frac{y}{r(x)}. \tag{2.26}$$

Vgl. Abbildung 2.1.
Für die Ableitungen erhält man:

$$\frac{\partial m}{\partial x} = \sqrt{1 + r_1^2}\,, \tag{2.27}$$

$$\frac{\partial m}{\partial y} = 0, \tag{2.28}$$

$$\frac{\partial \varphi}{\partial x} = \frac{-y r_1}{r\sqrt{r^2 - y^2}}, \tag{2.29}$$

$$\frac{\partial \varphi}{\partial y} = \frac{1}{\sqrt{r^2 - y^2}}. \tag{2.30}$$

Das Profil selbst ist mit Hilfe von 12 Basissplines N_j, $(j = 1,2,\ldots,12)$ der Ordnung 6 parametrisiert (vgl. [4]):

$$x(t) = \sum_{j=1}^{12} a_j N_j(t), \tag{2.31}$$

$$y(t) = \sum_{j=1}^{12} b_j N_j(t). \tag{2.32}$$

Die Koeffizienten a_j und b_j, für die wir auch die vereinheitlichende Schreibweise

$$q_j = \begin{cases} a_j & \text{für } 1 \le j \le 12 \\ b_{j-12} & \text{für } 13 \le j \le 24 \end{cases} \tag{2.33}$$

einführen, sind die *Einflußparameter* des Optimierungsproblems.

Die Berandungskurven φ_S und φ_N werden bei gegebenem Parametersatz q über die folgende Kette konstruiert, siehe die Abbildung 2.5:

$$m \mapsto x \mapsto t_S, t_N \mapsto y_S, y_N \mapsto \varphi_S(m), \varphi_N(m)$$

x ergibt sich aus m so:

$$x = \frac{1}{\sqrt{1+r_1^2}}\, m. \tag{2.34}$$

t_S und t_N sind die beiden Nullstellen der Gleichung

$$\sum_{j=1}^{12} a_j N_j(t) - x = 0, \qquad \text{wobei } t_S < t_N \tag{2.35}$$

sein soll, vgl. die Abbildung 2.6.

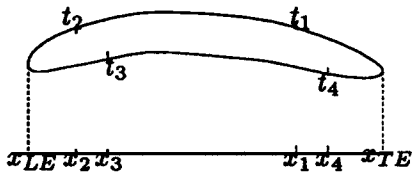


Abbildung 2.6: Profilparametrisierung

Dann ist

$$y_S := y(t_S) = \sum_{j=1}^{12} b_j N_j(t_S) \qquad \text{der Wert auf der Südseite und} \tag{2.36}$$

$$y_N := y(t_N) = \sum_{j=1}^{12} b_j N_j(t_N) \qquad \text{der Wert auf der Nordseite.} \tag{2.37}$$

Nun erhält man

$$\varphi_S(m) := \arcsin \frac{y_S}{r(x)}, \tag{2.38}$$

$$\varphi_N(m) := \arcsin \frac{y_N}{r(x)} \tag{2.39}$$

mittels Koordinatentransformation.

Um die Ableitung nach den Einflußparametern zu finden, müssen für sämtliche Teilschritte die Ableitungen nach den q_j gebildet werden. Dies ist beim Schritt von x auf t_S und t_N nur implizit möglich.

2.2.2 Gitterzellen und Numerierung

Sämtliche Diskretisierungen werden im „numerischen Gitter", also in den Koordinaten s und z, vorgenommen.

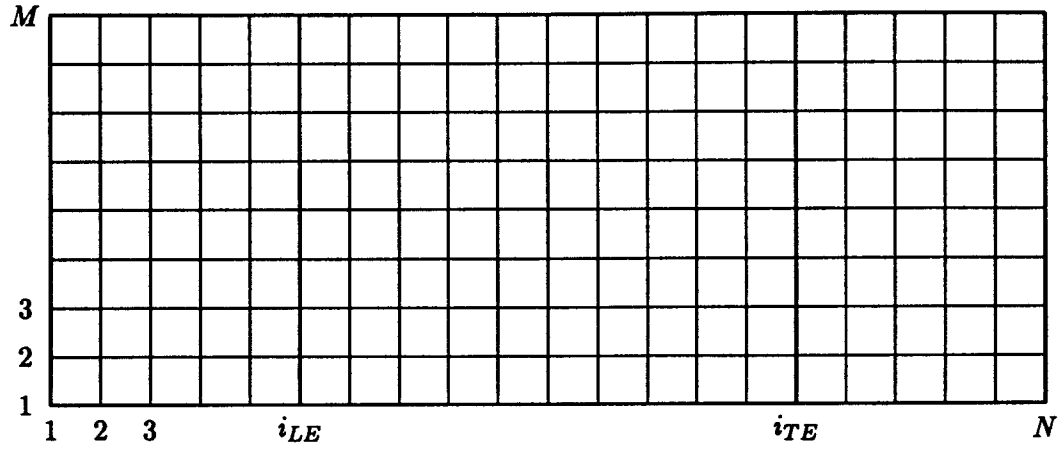


Abbildung 2.7: Numerisches Gitter in den Koordinaten s und z

Anzahl der Gitterpunkte: $N = 21$, $M = 9$

Zur Umsetzung in ein Programm benötigt man verschiedene Numerierungen:
1) Eine *komponentenweise* Numerierung: Mit der Koordinate s wird der Gitterindex i verbunden, der von 1 bis N laufen soll. Besonders hervorgehoben werden die Indizes i_{LE} und i_{TE}, die Leading Edge und Trailing Edge des Schaufelprofils markieren. Mit der Koordinate z wird der Gitterindex j verbunden, der von 1 bis M laufen soll. Eine Darstellung des Gitters mit $N = 21$ und $M = 9$, des zweitgröbsten im Code verwendeten Gitters, ist in Abbildung 2.7 gezeigt. Die Koordinatenwerte sind durch

$$s_i := (i-1)\Delta s \quad \textit{und} \tag{2.40}$$
$$z_j := (j-1)\Delta z \tag{2.41}$$

gegeben, wobei Δs und Δz die Schrittweiten in s- und z–Richtung sind. Abbildung 2.8 a) zeigt einen Ausschnitt aus dem (s,z)–Gitter.
2) Eine *globale* Numerierung mit *einem* Index: Es ist oft notwendig, die Gitterpunkte mit einem einzigen Index k durchzuzählen. Wir verwenden die *lexikographische* Numerierung, die zeilenweise vorgeht:

$$k = (i-1)N + j. \tag{2.42}$$

So erhält z.B. der Gitterpunkt $(i,j) = (3,3)$ in Abbildung 2.7 die Nummer $k = 2 * 21 + 3 = 45$.
3) Eine *lokale* Numerierung für Flächenmitten: Die Stromfunktion u wird immer in den Knoten (Schnittpunkte zweier Gitterlinien) ausgewertet, so daß die Gitterindizierung einfach übernommen werden kann, siehe Abbildung 2.8 b):

$$u_{ij} := u(s_i, z_j). \tag{2.43}$$

Die Dichte ρ dagegen wird immer in den Flächenmittelpunkten ausgewertet, so daß halbzahlige Indizes benötigt würden. Stattdessen werden aber die vier Flächenmittelpunkte, die

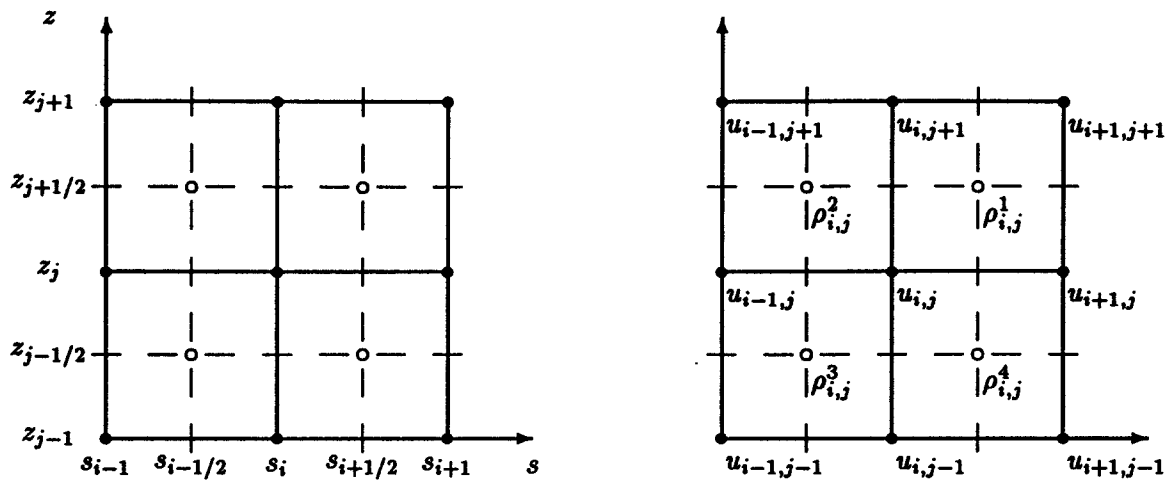


Abbildung 2.8: Vier Gitterzellen

a) (s,z)–Koordinatensystem b) Indizes für die Variablen u und ρ

einen Knoten (i,j) umgeben, lokal numeriert mit „Exponenten“ 1,2,3 und 4. So wählen wir die Bezeichnungen

$$\rho^1_{i,j} := \rho(s_{i+1/2}, z_{i+1/2}), \tag{2.44}$$
$$\rho^2_{i,j} := \rho(s_{i-1/2}, z_{i+1/2}), \tag{2.45}$$
$$\rho^3_{i,j} := \rho(s_{i-1/2}, z_{i-1/2}), \tag{2.46}$$
$$\rho^4_{i,j} := \rho(s_{i+1/2}, z_{i-1/2}). \tag{2.47}$$

Für $\rho^1_{i,j}$ wird auch $\rho_{i,j}$ geschrieben, wenn die anderen Werte von ρ nicht gebraucht werden. Im ρ–Gitter laufen die Indizes i und j nur bis $N-1$ bzw. $M-1$. Gleiche Indizes für in den Knoten gegebene Größen und für in den Flächenmittelpunkten gegebene Größen bezeichnen also unterschiedliche geometrische Punkte, die jeweils um halbe Schrittweiten gegeneinander verschoben sind.

Der Versatz der beiden Gitter hat für die Diskretisierungen zwei Konsequenzen:
1) Um Auswertungen von u und ρ in denselben Punkten zu ermöglichen, muß jeweils eine der Größen gemittelt werden, wodurch sich das Abhängigkeitsmuster erweitert.
2) Eine angenehme Folge jedoch ist, daß für die ersten Ableitungen von u immer zentrale Differenzen genommen werden können, die die Konsistenzordnung 2 aufweisen.

2.3 Die Nebenbedingungen

2.3.1 Die Stromfunktionsgleichung

Die Stromfunktionsgleichung lautet:

$$\nabla \cdot (A(\rho,q)\nabla u + C(q)) = 0 \qquad \text{mit} \tag{3.1}$$

$$A(\rho,q) = \begin{pmatrix} b(\rho,q) & -a(\rho,q) \\ -a(\rho,q) & c(\rho,q) \end{pmatrix}$$
$$= \frac{1}{\rho}\frac{1}{B_n r \det J}\begin{pmatrix} r^2\left(\frac{\partial\varphi}{\partial z}\right)^2 & -r^2\frac{\partial\varphi}{\partial s}\frac{\partial\varphi}{\partial z} \\ -r^2\frac{\partial\varphi}{\partial s}\frac{\partial\varphi}{\partial z} & r^2\left(\frac{\partial\varphi}{\partial s}\right)^2+\left(\frac{\partial m}{\partial s}\right)^2 \end{pmatrix}, \tag{3.2}$$

$$C(q) = \begin{pmatrix} -d(q) \\ e(q) \end{pmatrix} = \frac{\omega r^2}{\dot{M}}\begin{pmatrix} -\frac{\partial\varphi}{\partial z} \\ \frac{\partial\varphi}{\partial s} \end{pmatrix}. \tag{3.3}$$

Ausgeschrieben lautet die Gleichung (3.1):

$$\frac{\partial}{\partial s}\Big(b\frac{\partial u}{\partial s}-a\frac{\partial u}{\partial z}\Big)+\frac{\partial}{\partial z}\Big(-a\frac{\partial u}{\partial s}+c\frac{\partial u}{\partial z}\Big)=\frac{\partial d}{\partial s}-\frac{\partial e}{\partial z}. \tag{3.4}$$

Diskretisiert man mit zentralen Differenzen, die wegen der versetzten Gitter (Abbildung 2.8) gemittelt werden müssen, so erhält man für die einzelnen Terme die unten folgenden Formeln, jeweils für $i = 1,\ldots,N$ und $j = 1,\ldots,M$. Dabei sind die Koeffizienten a, b, c, d und e immer in den Flächenmitten auszuwerten und die Stromfunktion u in den Knoten. Die Bezeichnungen und Numerierungen werden wie in Abschnitt 2.2.2 gewählt.

$$\frac{\partial}{\partial s}\Big(b\frac{\partial u}{\partial s}\Big)(s_i,z_j) \approx \frac{1}{2\triangle s^2}\Big[(b^1_{ij}+b^4_{ij})u_{i+1,j} \tag{3.5}$$
$$-(b^1_{ij}+b^4_{ij}+b^2_{ij}+b^3_{ij})u_{ij}+(b^2_{ij}+b^3_{ij})u_{i-1,j}\Big],$$
$$\frac{\partial}{\partial z}\Big(c\frac{\partial u}{\partial z}\Big)(s_i,z_j) \approx \frac{1}{2\triangle z^2}\Big[(c^1_{ij}+c^2_{ij})u_{i,j+1} \tag{3.6}$$
$$-(c^1_{ij}+c^2_{ij}+c^3_{ij}+c^4_{ij})u_{ij}+(c^3_{ij}+c^4_{ij})u_{i,j-1}\Big],$$

$$-\Big[\frac{\partial}{\partial s}\Big(a\frac{\partial u}{\partial z}\Big)+\frac{\partial}{\partial z}\Big(a\frac{\partial u}{\partial s}\Big)\Big](s_i,z_j) \approx \frac{-1}{2\triangle s\triangle z}\Big[(-a^1_{ij}+a^2_{ij}-a^3_{ij}+a^4_{ij})u_{ij}$$
$$+a^1_{ij}\,u_{i+1,j+1}-a^2_{ij}\,u_{i-1,j+1}+a^3_{ij}\,u_{i-1,j-1}-a^4_{ij}\,u_{i+1,j-1}\Big], \tag{3.7}$$

$$\frac{\partial d}{\partial s}(s_i,z_j) \approx \frac{1}{2\triangle s}(d^1_{ij}+d^4_{ij}-d^2_{ij}-d^3_{ij}), \tag{3.8}$$
$$\frac{\partial e}{\partial z}(s_i,z_j) \approx \frac{1}{2\triangle z}(e^1_{ij}+e^2_{ij}-e^3_{ij}-e^4_{ij}). \tag{3.9}$$

Bemerkung: Wegen der Vertauschbarkeit der zweiten Ableitungen von φ nach s und z kann man die rechte Seite der Stromfunktionsgleichung analytisch vereinfachen:

$$\frac{\partial d}{\partial s}-\frac{\partial e}{\partial z}=\frac{2\omega r}{\dot{M}}\frac{\partial r}{\partial s}\frac{\partial\varphi}{\partial z} \tag{3.10}$$

Dabei wird auch $\frac{\partial r}{\partial z}=0$ verwendet. Im Code wird jedoch von dieser Möglichkeit bisher kein Gebrauch gemacht.

2.3.2 Die Dichtegleichung

Aus der technischen Thermodynamik werden die folgenden Begriffe übernommen:
1) Der *statische Zustand* eines Gases ist der in einem Raumpunkt der Strömung herrschende Zustand, der durch Zustandsgrößen vollständig beschrieben wird. Wir verwenden die Zustandsgrößen p (Druck), T (absolute Temperatur) und ρ (Dichte).
2) Der *Gesamtzustand* oder *Totalzustand* ist der Zustand, den das Gas im Zustand der Ruhe gehabt hat, bevor es unter Annahme isentroper Zustandsänderung auf den vorliegenden Bewegungszustand expandiert worden ist.
3) Den *Laval-Zustand* nimmt das Gas an, wenn es aus dem Gesamtzustand isentrop bis zur Schallgeschwindigkeit expandiert.

Größen, die den Totalzustand beschreiben, kennzeichen wir mit dem Index t, Größen, die den Laval-Zustand beschreiben, mit dem Index La.

Für die Lavalzahl La, die als Quotient von aktueller Geschwindigkeit $w = |\underline{w}|$ und der Laval-Geschwindigkeit a_{La} definiert ist,

$$La = \frac{w}{a_{La}}, \tag{3.11}$$

erhält man die folgende Abhängigkeit von Dichte und Totaldichte:

$$La = \sqrt{\frac{\kappa+1}{\kappa-1}\left(1-\left(\frac{\rho}{\rho_t}\right)^{\kappa-1}\right)}\,. \tag{3.12}$$

Dabei ist κ der Quotient aus den Wärmekapazitäten und hat für Luft einen Wert von $\kappa = 7/5$.

Die Geschwindigkeitskomponenten lassen sich durch die Stromfunktion ausdrücken, nach deren Definition:

$$w_m = \frac{\dot{M}}{B_n \rho}\frac{1}{r}\frac{\partial u}{\partial \varphi}, \tag{3.13}$$

$$w_\varphi = -\frac{\dot{M}}{B_n \rho}\frac{\partial u}{\partial m}. \tag{3.14}$$

Die Ableitungen von u nach m und φ müssen mittels der Koordinatentransformation $G \circ F$ durch Ableitungen nach s und z ersetzt werden:

$$w_m = \frac{\dot{M}}{B_n \rho}\frac{1}{r}\frac{1}{\det J}\frac{\partial m}{\partial s}\frac{\partial u}{\partial z}, \tag{3.15}$$

$$w_\varphi = -\frac{\dot{M}}{B_n \rho}\frac{1}{\det J}\left(\frac{\partial \varphi}{\partial z}\frac{\partial u}{\partial s} - \frac{\partial \varphi}{\partial s}\frac{\partial u}{\partial z}\right). \tag{3.16}$$

Für das Geschwindigkeitsquadrat ergibt sich

$$w^2 = w_m^2 + w_\varphi^2 = \frac{1}{\rho^2}\frac{\dot{M}^2}{B_n r}\frac{1}{\det J}(\nabla u)^{\mathsf{T}} A_0(q)(\nabla u), \tag{3.17}$$

wobei

$$A_0(q) := \rho A(\rho, q) \tag{3.18}$$

mit dem $A(\rho, q)$ aus Gleichung (3.2) definiert wird und nicht mehr von der Dichte abhängt. Dies ist für die Dichtegleichung übersichtlicher. Auch die Koeffizienten werden entsprechend definiert:

$$a_0(q) := \rho\, a(\rho, q), \tag{3.19}$$
$$b_0(q) := \rho\, b(\rho, q), \tag{3.20}$$
$$c_0(q) := \rho\, c(\rho, q), \tag{3.21}$$

so daß „wieder“

$$A_0(q) = \begin{pmatrix} b_0(q) & -a_0(q) \\ -a_0(q) & c_0(q) \end{pmatrix} \tag{3.22}$$

gilt. Aus den beiden Darstellungen von w^2 ergibt sich die *Dichtegleichung*

$$a_{La}^2 \frac{\kappa+1}{\kappa-1}\left(1-\left(\frac{\rho}{\rho_t}\right)^{\kappa-1}\right) = w^2 = \frac{1}{\rho^2}\frac{\dot{M}^2}{B_n r}\frac{1}{\det J}(\nabla u)^{\mathsf{T}} A_0(q)(\nabla u). \tag{3.23}$$

Die Abhängigkeit von u soll festgehalten werden:

$$(\nabla u)^{\mathsf{T}} A_0(q)(\nabla u) = b_0(q)\left(\frac{\partial u}{\partial s}\right)^2 - 2a_0(q)\frac{\partial u}{\partial s}\frac{\partial u}{\partial z} + c_0(q)\left(\frac{\partial u}{\partial z}\right)^2. \tag{3.24}$$

Diese Ausdrücke müssen in den Flächenmittelpunkten ausgewertet werden, was eine Diskretisierung mit gemittelten zentralen Differenzen nahelegt, vgl. Abbildung 2.12:

$$\left(\frac{\partial u}{\partial s}\right)(s_{i+1/2}, z_{j+1/2}) \approx \frac{1}{2\triangle s}(u_{i+1,j} + u_{i+1,j+1} - u_{ij} - u_{i,j+1}), \tag{3.25}$$
$$\left(\frac{\partial u}{\partial z}\right)(s_{i+1/2}, z_{j+1/2}) \approx \frac{1}{2\triangle z}(u_{i,j+1} + u_{i+1,j+1} - u_{ij} - u_{i+1,j}). \tag{3.26}$$

Zur Vereinfachung wird im folgenden die Geschwindigkeit w als Variable genommen — im Code muß sie mit Hilfe von u errechnet werden. Zur Abkürzung der Dichtegleichung wird die Funktion

$$\tilde{\Gamma}(u, \rho, q) := 1 - \left(\frac{\rho}{\rho_t}\right)^{\kappa-1} - \frac{\kappa-1}{\kappa+1}\left(\frac{w}{a_{La}}\right)^2 \tag{3.27}$$

definiert, die zu null werden soll, d.h. die Dichtegleichung ist äquivalent mit $\tilde{\Gamma}(u, \rho, q) = 0$. Im Code wird aber die Lavalzahl $\frac{w}{a_{La}}$ durch 1 begrenzt, indem die Funktion

$$\mu(x) := \min\{x, 1\} \tag{3.28}$$

zum „Abschneiden“ verwendet wird: Die modifizierte Dichtegleichung

$$\frac{\kappa+1}{\kappa-1}\left(1-\left(\frac{\rho}{\rho_t}\right)^{\kappa-1}\right) = \mu\left(\left(\frac{w}{a_{La}}\right)^2\right) \tag{3.29}$$

ist nun äquivalent mit $\Gamma(u, \rho, q) = 0$, wobei

$$\Gamma(u, \rho, q) := 1 - \left(\frac{\rho}{\rho_t}\right)^{\kappa-1} - \frac{\kappa-1}{\kappa+1}\mu\left(\left(\frac{w}{a_{La}}\right)^2\right) \tag{3.30}$$

definiert wird. Die Änderungen werden wirksam, sobald Überschallgebiete auftreten.

Für Zwecke der differenzierbaren Optimierung wurde eine glatte „Abschneidekurve" ν verwendet, die als Hermite–Interpolationspolynom an μ gewonnen wird. Für ein vorgegebenes ϵ (z.B. $\epsilon = 0.1$) werden die Bedingungen

$$\nu(1-\epsilon) = 1-\epsilon, \qquad \nu'(1-\epsilon) = 1, \tag{3.31}$$

$$\nu(1+\epsilon) = 1, \qquad \nu'(1+\epsilon) = 0 \tag{3.32}$$

durch das quadratische Polynom

$$\nu(x) = 1 - \frac{\epsilon}{4} + \frac{1}{2}(x-1) - \frac{1}{4\epsilon}(x-1)^2 \tag{3.33}$$

mit der Ableitung

$$\nu'(x) = \frac{1}{2} - \frac{1}{2\epsilon}(x-1) \tag{3.34}$$

erfüllt. μ und ν werden in Abbildung 2.9 dargestellt. In Gleichung (3.30) wird einfach μ durch ν ersetzt. Für die Optimierung brauchen wir auch die Ableitung.

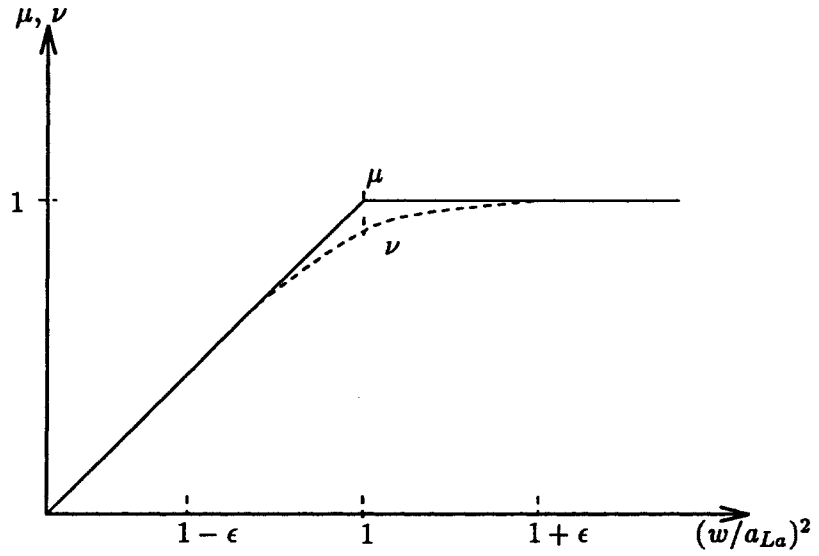


Abbildung 2.9: Glatte Begrenzung der Lavalzahl

Ableitungen der Dichte ρ tauchen nicht auf, so daß (außer der einfachen Auswertung) *keine Diskretisierung* erforderlich ist. Es werden auch *keine Randbedingungen* für die Dichte benötigt.

2.3.3 Randbedingungen

Die Abbildung 2.10 zeigt drei Arten von Gitterpunkten:
1) Die mit ⊙ gekennzeichneten Punkte sind Knoten, in denen die Stromfunktionsgleichung mit dem oben ermittelten 9–Punkt–Stern diskretisiert wird. Damit dies in Randnähe möglich ist, werden Randbedingungen benötigt.
2) Die mit □ gekennzeichneten Punkte sind Knoten, in denen eine Dirichlet–Bedingung (vorgegebene Werte von $\dot{u}$, längs der Profilkurven) bzw. eine periodische Randbedingung vorliegt

(außerhalb des Profilbereichs).
3) Die mit ◇ gekennzeichneten Punkte sind Knoten, in denen eine Winkelbedingung für die Einströmung bzw. die Ausströmung des Gases vorliegt (am Eintritts- und am Austrittsrand).

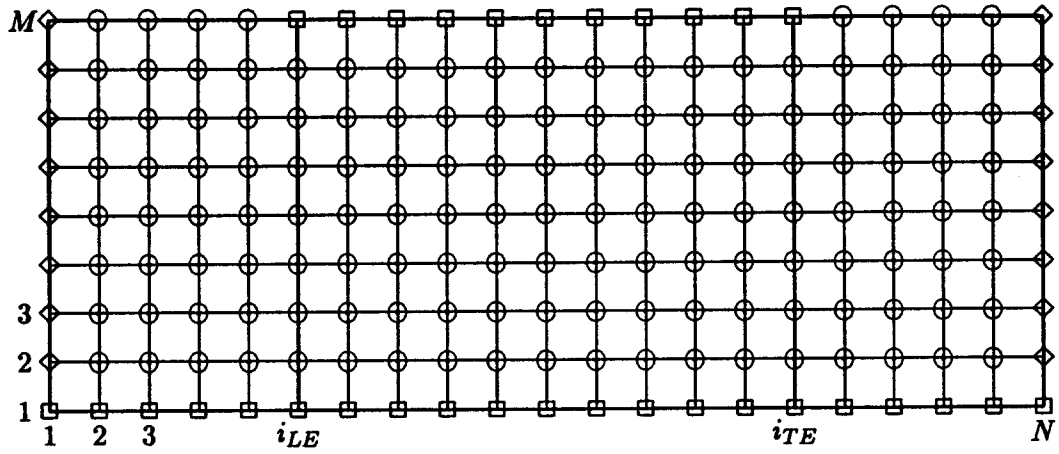


Abbildung 2.10: Numerisches Gitter mit Randpunkten
Anzahl der Gitterpunkte: $N = 21$, $M = 9$

1) Die ⊙–Punkte wurden bereits behandelt.
2) Randbedingungen für □–Punkte:
Die Dirichlet–Randbedingungen

$$u(s,z) = 0 \quad \text{für } s \in [\tfrac{1}{4},\tfrac{3}{4}],\ z = 0 \quad \text{Profil, Südseite,} \tag{3.35}$$
$$u(s,z) = 1 \quad \text{für } s \in [\tfrac{1}{4},\tfrac{3}{4}],\ z = 1 \quad \text{Profil, Nordseite} \tag{3.36}$$

werden diskretisiert durch

$$u_{i,1} = 0 \quad \text{für } i = i_{LE},\ldots,i_{TE}, \tag{3.37}$$
$$u_{i,M} = 1 \quad \text{für } i = i_{LE},\ldots,i_{TE}, \tag{3.38}$$

wobei i_{LE} und i_{TE} diejenigen Werte von i sind, bei denen das Profil anfängt bzw. endet. N bezeichnet die Anzahl der Gitterpunkte in s–Richtung, M die in z–Richtung. Die periodischen Randbedingungen

$$u(s,0) = u(s,1) - 1 \quad \text{für } s \in [0,\tfrac{1}{4}) \cup (\tfrac{3}{4},1] \quad \text{Vorlauf und Nachlauf,} \tag{3.39}$$
$$\frac{\partial u}{\partial z}(s,0) = \frac{\partial u}{\partial z}(s,1) \quad \text{für } s \in [0,\tfrac{1}{4}) \cup (\tfrac{3}{4},1] \quad \text{Vorlauf und Nachlauf} \tag{3.40}$$

werden diskretisiert durch

$$u_{i,M} - u_{i,1} = 1 \quad \text{für } i = 1,\ldots,i_{LE}-1,\ i_{TE}+1,\ldots,N \tag{3.41}$$
$$\frac{u_{i,2} - u_{i,1}}{\Delta z} = \frac{u_{i,M+1} - u_{i,M}}{\Delta z} \quad \text{für } i = 1,\ldots,i_{LE}-1,\ i_{TE}+1,\ldots,N. \tag{3.42}$$

$u_{i,M+1}$ bezeichnet den Wert in einem Fortsetzungspunkt, nach dem man aber wegen der ersten Bedingung (3.39) sofort auflösen kann:

$$u_{i,M+1} - u_{i,2} = 1 \qquad \text{für } i = 1,\ldots,i_{LE},\ i_{TE},\ldots,N. \tag{3.43}$$

3) Randbedingungen für ◇–Punkte:
Die Winkelbedingungen ergeben sich aus der Vorgabe eines Einströmwinkels β_1 und eines Ausströmwinkels β_2. Für diese beiden Winkel gilt:

$$\tan\beta = \frac{w_m}{w_\varphi} \qquad \text{für } \beta \in \{\beta_1, \beta_2\}. \tag{3.44}$$

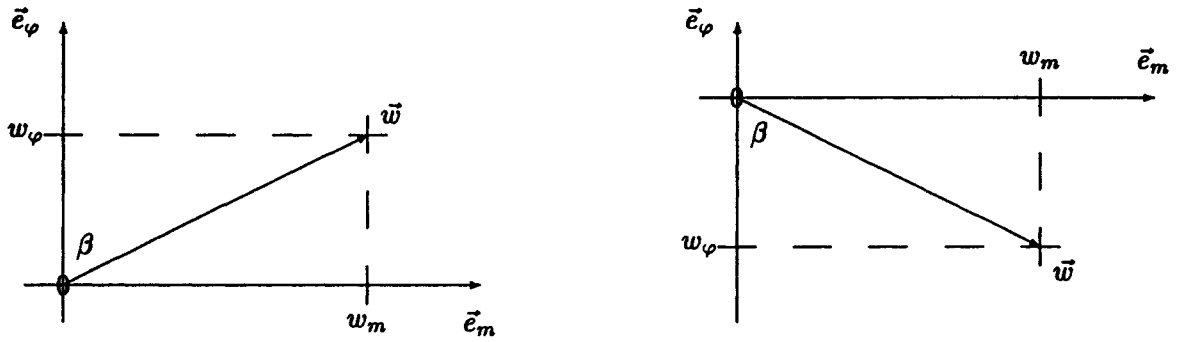


Abbildung 2.11: Zur Definition des Strömungswinkels β

a) Beispiel für $\tan\beta > 0$ b) Beispiel für $\tan\beta < 0$

In der Abbildung 2.11 ist jeweils ein Beispiel für einen positiven und einen negativen Wert von $\tan\beta$ gegeben. Aus den Gleichungen (3.13) und (3.14) erhält man als Winkelbedingung

$$r\frac{\partial u}{\partial m}\tan\beta = -\frac{\partial u}{\partial \varphi} \qquad \text{für } \beta \in \{\beta_1, \beta_2\}. \tag{3.45}$$

Für die Diskretisierung müssen die Ableitungen nach m und φ durch Ableitungen nach s und z ersetzt werden. Man erhält nach Koordinatentransformation die beiden äquivalenten Gleichungen

$$\Big(r\frac{\partial \varphi}{\partial z}\tan\beta - \frac{\partial m}{\partial z}\Big)\frac{\partial u}{\partial s} + \Big(-r\frac{\partial \varphi}{\partial s}\tan\beta + \frac{\partial m}{\partial s}\Big)\frac{\partial u}{\partial z} = 0, \tag{3.46}$$

$$\Big(r\frac{\partial s}{\partial m}\tan\beta + \frac{\partial s}{\partial \varphi}\Big)\frac{\partial u}{\partial s} + \Big(r\frac{\partial z}{\partial m}\tan\beta + \frac{\partial z}{\partial \varphi}\Big)\frac{\partial u}{\partial z} = 0. \tag{3.47}$$

Die zweite Version wird im Programm diskretisiert. Für die verwendeten Koordinatensysteme ist immer $\frac{\partial m}{\partial z} = 0$ und entsprechend $\frac{\partial s}{\partial \varphi} = 0$. Speziell im Vor– und Nachlauf hängt φ auch nicht von s ab, so daß $\frac{\partial \varphi}{\partial s} = 0$ und $\frac{\partial z}{\partial m} = 0$ gelten. Die Gleichungen (3.46) und (3.47) lauten somit:

$$\Big(r\frac{\partial \varphi}{\partial z}\tan\beta\Big)\frac{\partial u}{\partial s} + \Big(\frac{\partial m}{\partial s}\Big)\frac{\partial u}{\partial z} = 0, \tag{3.48}$$

$$\Big(r\frac{\partial s}{\partial m}\tan\beta\Big)\frac{\partial u}{\partial s} + \Big(\frac{\partial z}{\partial \varphi}\Big)\frac{\partial u}{\partial z} = 0. \tag{3.49}$$

Für die Gleichung (3.49) werden die Koeffizienten der partiellen Ableitungen von u in den Flächenmitten ausgewertet. Im Hinblick auf die Diskretisierung wird gleich durch $2\Delta s$ bzw. $2\Delta z$ dividiert:

$$\mathtt{cfx}_{ij} := \frac{1}{2\Delta s}\Big(r\frac{\partial s}{\partial m}\tan\beta\Big)(s_{i+1/2}, z_{j+1/2}), \tag{3.50}$$

$$\mathtt{cfy}_{ij} := \frac{1}{2\Delta z}\Big(\frac{\partial z}{\partial \varphi}\Big)(s_{i+1/2}, z_{j+1/2}), \tag{3.51}$$

jeweils für $i = 1$, $\beta = \beta_1$ und $j = 1, \ldots, M-1$ oder für $i = N-1$, $\beta = \beta_2$ und $j = 1, \ldots, M-1$. Die mit gemittelten zentralen Differenzen (siehe Abbildung 2.12) diskretisierte Winkelbedingung lautet:

$$\begin{aligned} &(\mathtt{cfx}_{ij} + \mathtt{cfy}_{ij})u_{i+1,j+1} + (-\mathtt{cfx}_{ij} + \mathtt{cfy}_{ij})u_{i,j+1} \\ +\ &(\mathtt{cfx}_{ij} - \mathtt{cfy}_{ij})u_{i+1,j} + (-\mathtt{cfx}_{ij} - \mathtt{cfy}_{ij})u_{ij} = 0 \\ &\qquad \text{für } i \in \{1, N-1\},\ j = 1, \ldots, M-1. \end{aligned} \tag{3.52}$$

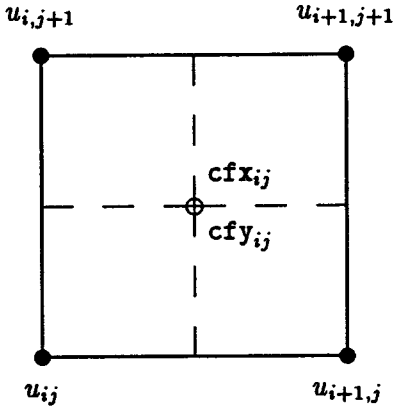


Abbildung 2.12: Eine Gitterzelle

2.3.4 Die Geometriebedingungen

Die Splineparameter, die das Profil beschreiben, legen auch Lage und Größe des Profils fest. Die optimale Form ist allerdings unabhängig von der Lage des Profils, weshalb das Optimum nicht eindeutig definiert ist. Eindeutig wird die Lösung des Optimierungsproblems erst, wenn man als zusätzliche Bedingung die Lage des Profils im Raum vorschreibt. Darüberhinaus soll das Lösungsprofil eine vorgegebene Länge haben. Insgesamt ergeben sich dadurch drei zusätzliche Gleichungsbedingungen an die Profilparameter. Zur konkreten Realisierung dieser Bedingungen gibt es viele Möglichkeiten. Der vorliegende (Simulations-)Mehrgitteralgorithmus baut allerdings darauf auf (oder sorgt im Preprocessing dafür), daß gilt:

$$\left.\begin{aligned} x(t_{LE}) &= x_{LE} \\ y(t_{LE}) &= y_{LE} \end{aligned}\right\} \text{ gegebene Position des } \textit{leading edge},$$

$$x(t_{TE}) = x_{TE} \quad \text{gegebene x-Position des } \textit{trailing edge}.$$

Um den vorhandenen (Simulations-)Mehrgitteralgorithmus weitgehend auszunutzen für den Optimierungsalgorithmus, ist es notwendig, diese Bedingungen **während der gesamten Optimierung zulässig** zu halten. Um entsprechende Differenzierbarkeitseigenschaften während der Optimierung aufrecht zu halten (s.u.), wurden in der ersten Programmversion drei Parameter $a_{j_1}, b_{j_1}; a_{j_2}$ ausgewählt, die mit Hilfe obiger Geometriebedingungen als Funktionen der anderen dargestellt werden können (vgl. Abb. 2.13). Die Auswahl wird mit Hilfe der Startparameter

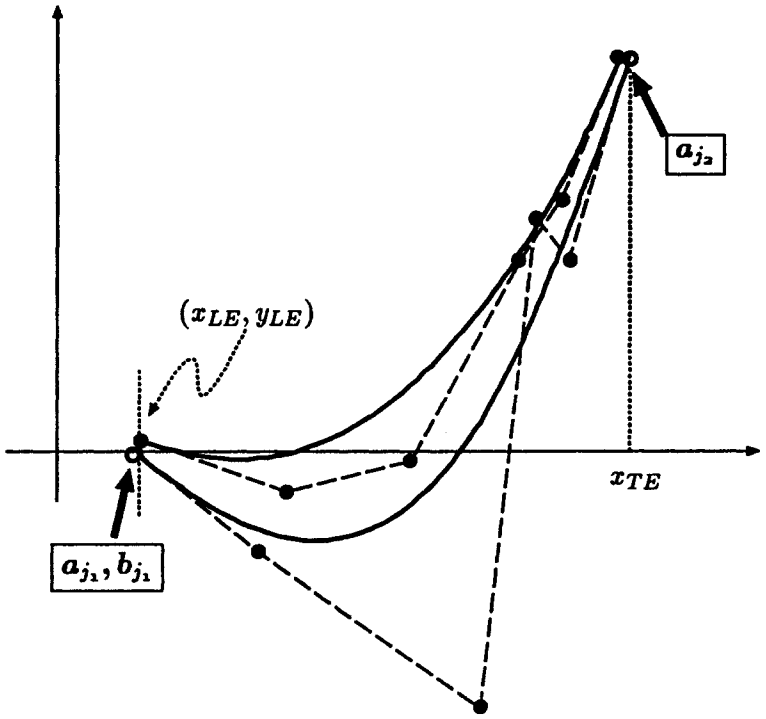


Abbildung 2.13: Profil mit Kontrollpolygon - abhängige Parameter

vorgenommen, die als zulässig angenommen werden, und zwar in der Art, daß j_1 der Index des minimalen a_j ist und j_2 der Index des maximalen a_j. Ziel ist hierbei, eine globale Parametrisierung des gesamten Zulässigkeitsbereiches während der Optimierung zu erhalten.

In der aktuellen Programmversion wird dieses Ziel einfacher durch Translation und Skalierung erreicht. Folgende Bedingungen und Definitionen müssen dazu erfüllt werden:

$$\sum_{i=1}^{m_q} a_i \dot{N}_i(t_{LE}) = 0 \quad \text{(Definition von } t_{LE}) \tag{3.53}$$

$$\sum_{i=1}^{m_q} a_i \dot{N}_i(t_{TE}) = 0 \quad \text{(Definition von } t_{TE}) \tag{3.54}$$

$$\sum_{i=1}^{m_q} b_i N_i(t_{LE}) = y_{LE} \tag{3.55}$$

$$\sum_{i=1}^{m_q} a_i N_i(t_{LE}) = x_{LE} \tag{3.56}$$

$$\sum_{i=1}^{m_q} a_i N_i(t_{TE}) \;=\; x_{TE} \tag{3.57}$$

wobei $m_q := n_q/2$ (n_q=#q). Offensichtlich ist es nötig, noch weitere Variablen einzuführen, nämlich t_{LE} und t_{TE}. Da diese Größen in die Diskretisierungsgleichungen nicht eingehen, sind die entsprechenden Ableitungen der Diskretisierungsgleichungen nach t_{LE} und t_{TE} null.

Ordnet man die Freiheitsgrade in der Reihenfolge $(a_1,\ldots,a_{m_q},b_1,\ldots,b_{m_q},t_{LE},t_{TE})$, so ergibt sich folgende Jacobimatrix der fünf Geometriebedingungen:

$$J = \begin{bmatrix}
\dot N_1(t_{LE}) & \ldots & \dot N_{m_q}(t_{LE}) & 0 & \ldots & 0 & \sum_{i=1}^{m_q} a_i \ddot N_i(t_{LE}) & 0 \\
\dot N_1(t_{TE}) & \ldots & \dot N_{m_q}(t_{TE}) & 0 & \ldots & 0 & 0 & \sum_{i=1}^{m_q} a_i \ddot N_i(t_{TE}) \\
0 & \ldots & 0 & N_1(t_{LE}) & \ldots & N_{m_q}(t_{LE}) & \sum_{i=1}^{m_q} b_i \dot N_i(t_{LE}) & 0 \\
N_1(t_{LE}) & \ldots & N_{m_q}(t_{LE}) & 0 & \ldots & 0 & \sum_{i=1}^{m_q} a_i \dot N_i(t_{LE}) & 0 \\
N_1(t_{TE}) & \ldots & N_{m_q}(t_{TE}) & 0 & \ldots & 0 & 0 & \sum_{i=1}^{m_q} a_i \dot N_i(t_{TE})
\end{bmatrix}$$

Bezeichnet man die Bedingungen (3.53)–(3.57) zusammenfassend mit $g(q,t_{LE},t_{TE}) = 0$, ergibt sich mit den Definitionen $q_1 := (a_{j_1},a_{j_2},b_{j_1})^\top$, $\hat q_1 := (q_1^\top,t_{LE},t_{TE})$, q_2 :=Rest von q, eine Aufspaltung der Jacobimatrix in einen regulären Teil $\partial g/\partial q_1$, der zu den abhängigen Parametern korrespondiert und in einen Teil $\partial g/\partial q_2$, der zu den echten Freiheitsgraden des Gesamtsystems korrespondiert. Beide seien der Vollständigkeit halber nochmal aufgeführt:

$$G_{\hat q_1} := \frac{\partial g}{\partial \hat q_1} \;=\; \begin{bmatrix}
\dot N_{j_1}(t_{LE}) & \dot N_{j_2}(t_{LE}) & 0 & \sum_{i=1}^{m_q} a_i \ddot N_i(t_{LE}) & 0 \\
\dot N_{j_1}(t_{TE}) & \dot N_{j_2}(t_{TE}) & 0 & 0 & \sum_{i=1}^{m_q} a_i \ddot N_i(t_{TE}) \\
0 & 0 & N_{j_1}(t_{LE}) & \sum_{i=1}^{m_q} b_i \dot N_i(t_{LE}) & 0 \\
N_{j_1}(t_{LE}) & N_{j_2}(t_{LE}) & 0 & \sum_{i=1}^{m_q} a_i \dot N_i(t_{LE}) & 0 \\
N_{j_1}(t_{TE}) & N_{j_2}(t_{TE}) & 0 & 0 & \sum_{i=1}^{m_q} a_i \dot N_i(t_{TE})
\end{bmatrix}$$

$$G_{q_2} := \frac{\partial g}{\partial q_2} \;=\; \begin{bmatrix}
\dot N_1(t_{LE}) & \ldots <j_1,j_2> \ldots & \dot N_{m_q}(t_{LE}) & 0 & \ldots\ldots\ldots\ldots & 0 \\
\dot N_1(t_{TE}) & \ldots <j_1,j_2> \ldots & \dot N_{m_q}(t_{TE}) & 0 & \ldots\ldots\ldots\ldots & 0 \\
0 & \ldots\ldots\ldots\ldots & 0 & N_1(t_{LE}) & \ldots <j_1> \ldots & N_{m_q}(t_{LE}) \\
N_1(t_{LE}) & \ldots <j_1,j_2> \ldots & N_{m_q}(t_{LE}) & 0 & \ldots\ldots\ldots\ldots & 0 \\
N_1(t_{TE}) & \ldots <j_1,j_2> \ldots & N_{m_q}(t_{TE}) & 0 & \ldots\ldots\ldots\ldots & 0
\end{bmatrix}$$

wobei eingeklammerte Indizes („<>“) weggelassene Matrixeinträge bedeuten.

2.4 Das Optimierungsproblem

Das Ziel des Optimierungsverfahrens ist es, Splineparameter dergestalt zu finden, daß ein vorgegebener Lavalzahlverlauf längs des Profils realisiert wird. Idealerweise soll also ein Zielfunktional der Form

$$\begin{aligned} f(u,\rho,q) \;:=\; & \frac{1}{2}\int_{t_{TE}}^{t_{LE}} \left[La(u_S(t),\rho_S(t),q) - \widehat{La}_S(t)\right]^2 \omega_S(t)dt \\ & + \frac{1}{2}\int_{t_{LE}}^{t_{TE}} \left[La(u_N(t),\rho_N(t),q) - \widehat{La}_N(t)\right]^2 \omega_N(t)dt \end{aligned}$$

minimiert werden. Hierbei bezeichnen die Indizes N und S wieder die Nord- bzw. Südseite. Die Funktionen $\omega_S(t)$ und $\omega_N(t)$ sind Gewichtsfunktionen und dienen dazu, einzelne numerisch schlecht aufgelöste Bereiche auszublenden oder andere Bereiche stärker zu gewichten. Sie werden a priori festgelegt und bleiben während der Optimierung unverändert. Die Zielgrößen $\widehat{La}_S(t)$ und $\widehat{La}_N(t)$ liegen nur als diskrete Größen an den Gitterpunkten des feinsten Gitters längs des Profils vor. Dies legt nahe, als diskretes Zielfunktional

$$\begin{aligned} f_h(u_h,\rho_h,q) \;:=\; & \frac{1}{2}\sum_{i=i_{LE}}^{i_{TE}} \left[La(u_{S,i},\rho_{S,i},q) - \widehat{La}_{S,i}\right]^2 \omega_{S,i} \\ & + \frac{1}{2}\sum_{i=i_{LE}}^{i_{TE}} \left[La(u_{N,i},\rho_{N,i},q) - \widehat{La}_{N,i}\right]^2 \omega_{N,i} \end{aligned} \tag{4.1}$$

zu betrachten. Der Standardfall ist $\omega_{S,i} = 1$, $\omega_{N,i} = 1$. Somit ist f_h von der Anzahl der Summanden und damit von der Schrittweite h abhängig und keine Approximation an die Integrale oben. Dieses Funktional soll minimiert werden unter den folgenden Nebenbedingungen:

(1) $c_U(u_h,\rho_h,q) = 0$, diskretisierte Stromfunktionsgleichung mit Randbedingungen

(2) $c_D(u_h,\rho_h,q) = 0$, diskretisierte Dichtegleichung

(3) Geometriebedingungen an das Profil

Diese Bedingungen sind im vorigen Abschnitt eingehend erläutert worden.

2.5 Der Optimierungsalgorithmus

2.5.1 Der RSQP–Algorithmus

In einer ersten Programmversion wurde ein reduziertes SQP-Verfahren verwendet. Diese Verfahren beruhen darauf, nur die auf den Kern der linearisierten Gleichungsnebenbedingungen projizierte Hessematrix aufzustellen und auch nur dort ein quadratisches Programm zu lösen. Faßt man die diskretisierte Stromfunktion, die diskretisierte Dichte und die abhängigen Splineparameter (vgl. Abschnitt 2.3) zusammen in der Variable x und die restlichen Splineparameter in der Variablen q und sammelt man alle Beschränkungen (1)–(3) in der Funktion $c(x,q)$, so läßt sich das zu lösende Optimierungsproblem abstrakt schreiben als:

$$\begin{aligned} &\min f(x,q) \\ &\text{unter } c(x,q) = 0 \end{aligned}$$

Für dieses abstrakte Problem wird der vorgeschlagene RSQP–Algorithmus formuliert. Es werden folgende Notationen verwendet: $C_x^k := \partial c(x^k,q^k)/\partial x$, $C_q^k := \partial c(x^k,q^k)/\partial q$, $\nabla_x f^k := (\partial f(x^k,q^k)/\partial x)^\mathsf{T}$, $\nabla_q f^k := (\partial f(x^k,q^k)/\partial q)^\mathsf{T}$; Indizes k bedeuten Auswertung in x^k.

RSQP-Algorithmus:

(0) Start in x^0, q^0, B^0, $k := 0$

(1) Berechne die *adjungierten Variablen* und den *reduzierten Gradienten*:
$\lambda^k := \left(C_x^k\right)^{-\mathsf{T}} \nabla_x f^k$, $\gamma^k := \nabla_q f^k - \left(C_q^k\right)^\mathsf{T} \lambda^k$

(2) Löse: $B^k \Delta q^k = -\gamma^k$

(3) Bestimme *Schritt* im Bildraum: $\Delta x^k = -\left(C_x^k\right)^{-1}\left(C_q^k \Delta q^k + c^k\right)$

(4) Linesearch $\longrightarrow \alpha^k$

(5) Bestimme Zwischenwerte für Update
$q^{k+} := q^k + \alpha^k \Delta q^k$, $x^{k+} := x^k - \alpha^k \left(C_x^k\right)^{-1} C_q^k \Delta q^k$, $\lambda^{k+} := \left(C_x^{k+}\right)^{-\mathsf{T}} \nabla_x f^{k+}$
$\gamma^{k+} := \nabla_q f^{k+} - \left(C_q^{k+}\right)^\mathsf{T} \lambda^{k+}$ („$()^{k+}$“ ≙ Auswertung in (x^{k+}, q^{k+}))

(6) Führe *Update der reduzierten Hessematrix* durch:
$B^{k+1} := B^k + U^{BFGS}(B^k, \alpha^k \Delta q^k, \gamma^{k+} - \gamma^k)$

(7) Führe *Gesamtschritt* aus: $x^{k+1} := x^k + \alpha^k \Delta x^k$, $q^{k+1} := q^k + \alpha^k \Delta q^k$

(8) $k := k + 1$, fahre fort bei (1), bis Konvergenz erreicht ist.

Für diesen Algorithmus läßt sich unter moderaten Regularitätsannahmen lokale 2-Schritt-superlineare Konvergenz zeigen. Globale Konvergenzeigenschaften werden durch die Linesearch bewirkt. Es lassen sich auch lokal 1-Schritt-superlinear konvergente Varianten davon formulieren. Jedoch wird die Linesearch dann sehr viel aufwendiger. Schritt (5) und der Update stellen den wesentlichen Unterschied zum Vorgehen von Nocedal/Overton [11] dar. Im Schritt (5)

wird etwa die Hälfte der gesamten Rechenzeit pro Iteration verbraucht. Allerdings weist der Algorithmus von Nocedal/Overton starke numerische Instabilitäten auf. Auch diese Möglichkeit, den Algorithmus zu verändern, wurde ausgetestet.

2.5.2 Der PRSQP–Algorithmus

Bei der praktischen Anwendung des RSQP–Algorithmus auf die Schaufeloptimierung haben sich Probleme ergeben: In Schritt (2) des Algorithmus wird nur nach den Änderungen der unabhängigen Designparameter aufgelöst, in unserem Fall sind das $24 - 3 = 21$ Variablen. Die übrigen Designparameter (3 Stück) und die Kurvenparameter t_{LE} und t_{TE} für das Leading Edge (LE) und das Trailing Edge (TE) werden mit Hilfe eines Newton–Verfahrens aus den Geometriebedingungen bestimmt. (Diese sind ja bezüglich t_{LE} und t_{TE} nichtlinear). Diese Auflösung ist schlecht konditioniert: Selbst kleine Parameteränderungen Δq_2^k können zu großen Änderungen Δq_1^k führen, so daß völlig unbrauchbare Profile errechnet werden. Die Profilnormierung sollte nicht mit wenigen, sondern mit *allen* Designparametern vorgenommen werden.

Zu diesem Zweck kann man ein partiell reduziertes SQP–Verfahren formulieren, siehe Schulz [17]. Das Verfahren bezeichnen wir mit PRSQP. Ein Teil der Nebenbedingungen wird nicht bei der Linearisierung eliminiert, sondern beibehalten. Dementsprechend arbeitet man auch mit mehr Variablen. In unserem Fall wird nicht mehr auf 21 Variablen reduziert wie bei der RSQP–Methode, sondern auf 26 Variablen (24 Designparameter und die zwei Kurvenparameter $t_{\mathrm{LE}}, t_{\mathrm{TE}}$). Die fünf Geometriebedingungen werden als Nebenbedingungen *mitgeführt.* Das PRSQP–Verfahren liegt gewissermaßen *zwischen* der RSQP–Methode und der SQP–Methode.

2.5.2.1 Asymptotisch korrekter BFGS–Update der reduzierten Hessematrix

Die erste PRSQP–Variante errechnet eine Zwischenstelle (x^+, q^+), die nur für den BFGS–Update verwendet wird. Der Begriff „asymptotisch korrekt" bezieht sich hierbei darauf, daß im Gegensatz zu der Update-Strategie in [11] die Sekantenbedingung für die Größen, für die sie verwendet wird, in der Nähe der Lösung auch gilt. Mit dieser Variante wurden auch die später aufgeführten numerischen Ergebnisse produziert.

PRSQP-Algorithmus, Update asymptotisch korrekt:

(0) Start in x^0, q^0, B^0, $k := 0$

(1) Berechne die *adjungierten Variablen* und den *partiell reduzierten Gradienten*:
$\lambda^k := (C_x^k)^{-\mathsf{T}} \nabla_x f^k$, $\gamma^k := \nabla_q f^k - (C_q^k)^{\mathsf{T}} \lambda^k$

(2) Löse QP: $\begin{bmatrix} B^k & (G_q^k)^{\mathsf{T}} \\ G_q^k & 0 \end{bmatrix} \begin{bmatrix} \Delta q^k \\ -\nu^k \end{bmatrix} = \begin{bmatrix} -\gamma^k \\ -g^k \end{bmatrix}$

(3) Bestimme *Schritt* im Bildraum: $\Delta x^k := -(C_x^k)^{-1} \left(c^k + C_q^k \Delta q^k \right)$

(4) Linesearch $\longrightarrow \alpha^k$

(5) Bestimme *Zwischenwerte* für Update:

$$\Delta x^{k+} := -(C_x^k)^{-1} C_q^k \Delta q^k$$

$$\begin{aligned} x^{k+} &:= x^k + \alpha^k \omega_{\text{upd}} \Delta x^{k+} \\ q^{k+} &:= q^k + \alpha^k \omega_{\text{upd}} \Delta q^k \\ \lambda^{k+} &:= (C_x^{k+})^{-\mathsf{T}} \nabla_x f^{k+} \\ \gamma^{k+} &:= \nabla_q f^{k+} - (C_q^{k+})^{\mathsf{T}} \lambda^{k+} \end{aligned}$$

(6) Führe *Update der partiell reduzierten Hessematrix* durch:
$B^{k+1} := B^k + U^{BFGS}\left(B^k, q^{k+} - q^k, \gamma^{k+} - (G_q^{k+})^{\mathsf{T}} \nu^k - \left(\gamma^k - (G_q^k)^{\mathsf{T}} \nu^k\right)\right)$

(7) Führe *Gesamtschritt* aus: $x^{k+1} := x^k + \alpha^k \Delta x^k$; $\quad q^{k+1} := q^k + \alpha^k \Delta q^k$

(8) $k := k + 1$, fahre fort bei (1), bis Konvergenz erreicht ist.

Bei der Anwendung dieses Algorithmus auf die Schaufeloptimierung ist noch darauf zu achten, daß nach jeder Parameteränderung um Δq der neue Satz von Designparametern normiert werden muß. Die dazu nötige Verschiebung (LE in den Ursprung des Koordinatensystems) und Skalierung (Profillänge ist vorgegeben) bringt aber nur kleine Änderungen. Diese Operationen ergeben auch keine Instabilitäten wie beim RSQP–Algorithmus. Hier beim PRSQP–Algorithmus werden alle Parameter gleichmäßig behandelt und keine ausgezeichnet.

Der Parameter ω_{upd} wurde zu Zwecken des Fine-Tunings des Update eingeführt. Die Differenz von reduzierten Gradienten ergibt nur dann eine sinnvolle Approximation für die zweite Ableitung, wenn die Stellen, an denen ausgewertet wird, nicht zu weit entfernt sind. In der praktischen Durchführung hat sich $\omega_{\text{upd}} = 0.1$ als ein guter Wert erwiesen.

2.5.2.2 Update–Strategie gemäß Nocedal/Overton

In dem im vorigen Abschnitt vorgestellten PRSQP–Algorithmus ist die Berechnung der Zwischenstelle (x^{k+}, q^{k+}) genauso aufwendig wie die Berechnung von λ^k und Δx^k, weil ebenfalls zwei große lineare Systeme zu lösen sind. Die Idee von Nocedal und Overton in [11] besteht darin, die Differenz der reduzierten Gradienten für den Hessematrix–Update in Schrittrichtung $\alpha^k(\Delta x^k, \Delta q^k)$ vorzunehmen. Wenn man keine Vollschritte geht, sondern wieder den Parameter ω_{upd} einführt, erhält man den folgenden Algorithmus, der die Auflösung von nur noch drei (statt vier) linearen Gleichungssystemen pro Optimierungsschritt erfordert:

PRSQP-Algorithmus, Update mit Nocedal/Overton–Strategie:

(0) Start in x^0, q^0, B^0, ν^0, $k := 0$

(1) Berechne die *adjungierten Variablen* und den *partiell reduzierten Gradienten*:
$\lambda^k := (C_x^k)^{-\mathsf{T}} \nabla_x f^k$, $\quad \gamma^k := \nabla_q f^k - (C_q^k)^{\mathsf{T}} \lambda^k$

(2) Löse QP: $\begin{bmatrix} B^k & (G_q^k)^{\mathsf{T}} \\ G_q^k & 0 \end{bmatrix} \begin{bmatrix} \Delta q^k \\ -\nu^{k+1} \end{bmatrix} = \begin{bmatrix} -\gamma^k \\ -g^k \end{bmatrix}$

(3) Falls $k \geq 2$ ist, führe den *Update der partiell reduzierten Hessematrix* durch:

$$\begin{aligned} B^{k+1} := B^k + U^{BFGS}\Big(B^k, q^{(k-1)+} - q^{k-1}, \gamma^{(k-1)+} - (G_q^{(k-1)+})^{\mathsf{T}} \nu^k \\ - \left(\gamma^{(k-1)} - (G_q^{(k-1)})^{\mathsf{T}} \nu^k\right)\Big) \end{aligned}$$

(4) Bestimme *Schritt*: $\Delta x^k := -(C_x^k)^{-1}\left(c^k + C_q^k \Delta q^k\right)$

(5) Linesearch $\longrightarrow \alpha^k$

(6) Bestimme die *adjungierten Variablen* der Geometriebedingungen:
$\nu^{(k+1)} := \nu^k + \alpha^k\left(\nu^{k+1} - \nu^k\right)$
Bestimme *Zwischenschritt* für den Update:

$$\begin{aligned} x^{k+} &:= x^k + \alpha^k \omega_{\text{upd}} \Delta x^k \\ q^{k+} &:= q^k + \alpha^k \omega_{\text{upd}} \Delta q^k \\ \lambda^{k+} &:= (C_x^{k+})^{-\mathsf{T}} \nabla_x f^{k+} \\ \gamma^{k+} &:= \nabla_q f^{k+} - (C_q^{k+})^{\mathsf{T}} \lambda^{k+} \end{aligned}$$

(7) Führe *Gesamtschritt* aus: $x^{k+1} := x^k + \alpha^k \Delta x^k$; $\quad q^{k+1} := q^k + \alpha^k \Delta q^k$

(8) $k := k + 1$, fahre fort bei (1), bis Konvergenz erreicht ist.

Bei der Messung der Rechenzeiten für die einzelnen Programmteile hat sich herausgestellt, daß das Aufstellen der linearen Gleichungssysteme, also die Ableitungsberechnung für C_q^k, sehr aufwendig ist, so daß die Ersparnis gegenüber dem asymptotisch korrekten BFGS–Update aus dem vorigen Abschnitt gar nicht so groß ist. Da sich die asymptotisch korrekte Variante auch als stabiler und zuverlässiger erwiesen hat, geben wir ihr eindeutig den Vorzug. Was den Rechenaufwand angeht: Wenn man auf den Parameter ω_{upd} verzichten könnte, dann würde man eine komplette Linearisierung und ein weiteres lineares Gleichungssystem pro Optimierungsschritt einsparen, was den Aufwand deutlich reduzieren würde. Aber die Voraussetzung dazu ist in unserem Fall nicht gegeben, weil mit $\omega_{\text{upd}} = 1.0$ die Schrittlängen $q^k - q^{k-1}$ zu groß werden, um noch sinnvoll numerische Differenzen zur Approximation der zweiten Ableitung bilden zu können.

2.5.3 Die asymptotisch korrekte PRSQP–Methode zur Schaufeloptimierung im Detail

Im folgenden wird der implementierte Schaufeloptimierungsalgorithmus, der die Geometriebedingungen durchgehend zulässig hält, ausführlich dargestellt. Größen wie u oder ρ sind immer als diskretisierte Größen auf dem feinsten Gitter zu verstehen. Der übersichtlicheren Schreibweise wegen wird ein Index h weggelassen.

Der PRSQP–Algorithmus zur Schaufeloptimierung:

(0) Start : $k := 0$. Gegeben: u^0, ρ^0, zulässige Splineparameter q^0. Initialisiere die reduzierte Hessematrix B^0 mit einer geeigneten Diagonalmatrix.

(1a) Erzeuge das Gitter und berechne die Ableitungen in der Matrix

$$\begin{bmatrix} U_u^k & U_\rho^k & U_q^k \\ D_u^k & D_\rho^k & D_q^k \\ 0 & 0 & G_q^k \end{bmatrix}.$$

(1b) Berechne den Gradienten des (regularisierten) Zielfunktionals: $(\nabla_u f^k, \nabla_\rho f^k, \nabla_q f^k)$. Werte die Residuen der Gleichungsnebenbedingungen aus: c_U^k, c_D^k, g^k.

(1c) Berechne die *adjungierten Variablen* als Lösung des Systems:
$$\begin{bmatrix} U_u^k & U_\rho^k \\ D_u^k & D_\rho^k \end{bmatrix}^\top \begin{bmatrix} \lambda^k \\ \mu^k \end{bmatrix} = \begin{bmatrix} \nabla_u f^k \\ \nabla_\rho f^k \end{bmatrix}$$

(1d) Berechne den *partiell reduzierten Gradienten*:
$$\gamma^k := \nabla_q f^k - (U_q^k)^\top \lambda^k - (D_q^k)^\top \mu^k$$

(2) Löse das partiell reduzierte *Karush–Kuhn–Tucker–System* KKT:
$$\begin{bmatrix} B^k & (G_q^k)^\top \\ G_q^k & 0 \end{bmatrix} \begin{bmatrix} \Delta q^k \\ -\nu^k \end{bmatrix} = \begin{bmatrix} -\gamma^k \\ -g^k \end{bmatrix}$$
(*Trust–Region–Variante:* Addiere in diesem System zu B^k hinreichend große positive Vielfache der Einheitsmatrix, bis für die Lösung Δq^k die Ungleichung $\left\|\Delta q^k\right\|_2 \leq r_{\mathrm{TR}}$ mit einem vorgegebenen Trust–Region–Radius r_{TR} erfüllt ist.)

(3) Bestimme den *Schritt* in den diskretisierten Variablen als Lösung des Gleichungssystems:
$$\begin{bmatrix} U_u^k & U_\rho^k \\ D_u^k & D_\rho^k \end{bmatrix} \begin{bmatrix} \Delta u^k \\ \Delta \rho^k \end{bmatrix} = - \begin{bmatrix} c_U^k + U_q^k \Delta q^k \\ c_D^k + D_q^k \Delta q^k \end{bmatrix}$$

(4) Berechne den Linesearch–Parameter α^k

(5) Bestimme Zwischenwerte für den Update:

(5a) Bestimme den *Zwischenschritt* in den diskretisierten Variablen als Lösung des Gleichungssystems
$$\begin{bmatrix} U_u^k & U_\rho^k \\ D_u^k & D_\rho^k \end{bmatrix} \begin{bmatrix} \Delta u^{k+} \\ \Delta \rho^{k+} \end{bmatrix} = - \begin{bmatrix} U_q^k \Delta q^k \\ D_q^k \Delta q^k \end{bmatrix}$$

(5b) Berechne die Zwischenwerte u^{k+}, ρ^{k+}, q^{k+}:
$$\begin{aligned} u^{k+} &= u^k + \alpha^k \, \omega_{\mathrm{upd}} \, \Delta u^{k+} \\ \rho^{k+} &= \rho^k + \alpha^k \, \omega_{\mathrm{upd}} \, \Delta \rho^{k+} \\ q^{k+} &= q^k + \alpha^k \, \omega_{\mathrm{upd}} \, \Delta q^k \end{aligned}$$

(5c) Erzeuge das Gitter und berechne die Ableitungen in der Matrix
$$\begin{bmatrix} U_u^{k+} & U_\rho^{k+} & U_q^{k+} \\ D_u^{k+} & D_\rho^{k+} & D_q^{k+} \\ 0 & 0 & G_q^{k+} \end{bmatrix}$$

(5d) Berechne den Gradienten des (regularisierten) Zielfunktionals: $\left(\nabla_u f^{k+}, \nabla_\rho f^{k+}, \nabla_q f^{k+}\right)$.

(5e) Berechne die *adjungierten Variablen* an der Zwischenstelle als Lösung des Systems:
$$\begin{bmatrix} U_u^{k+} & U_\rho^{k+} \\ D_u^{k+} & D_\rho^{k+} \end{bmatrix}^\top \cdot \begin{bmatrix} \lambda^{k+} \\ \mu^{k+} \end{bmatrix} = \begin{bmatrix} \nabla_u f^{k+} \\ \nabla_\rho f^{k+} \end{bmatrix}$$

(5f) Berechne den *partiell reduzierten Gradienten* an der Zwischenstelle:
$\gamma^{k+} := \nabla_q f^k - (U_q^{k+})^\top \lambda^{k+} - (D_q^{k+})^\top \mu^{k+}$

(6) Führe den *Update der partiell reduzierten Hessematrix* durch:
$B^{k+1} := B^k + U^{BFGS}\left(B^k, q^{k+} - q^k, \gamma^{k+} - (G_q^{k+})^\top \nu^k - \left(\gamma^k - (G_q^k)^\top \nu^k\right)\right)$

(7a) Führe den Gesamtschritt aus:

$$\begin{aligned} u^{k+1} &:= u^k + \alpha^k \Delta u^k \\ \rho^{k+1} &:= \rho^k + \alpha^k \Delta \rho^k \\ q^{k+1} &:= q^k + \alpha^k \Delta q^k \end{aligned}$$

(7b) Mache q^{k+1} zulässig bzgl. der Geometriebedingungen: Durch Verschiebung und Skalierung.

(8) $k := k + 1$, fahre fort bei (1), bis Konvergenz erreicht ist.

Der Übersichlichkeit wegen wurde die Notation vereinfacht, indem die Randbedingungen nicht mehr explizit mit eigenen Operatoren aufgeführt werden. Die Randbedingungen für die Stromfunktion u sind in den Blöcken U_u, U_ρ, U_q und in den entsprechenden rechten Seiten mit berücksichtigt. Die Dichte ρ hat ohnehin keine Randbedingungen. Diese Darstellung entspricht auch der Implementierung, bei der die Randpunkte wie die inneren Gitterpunkte lexikographisch mit numeriert werden.

Die rechenzeitintensivsten Teile des RSQP-Algorithmus sind die Aufstellung der linearisierten Systeme, was einen Aufruf des Netzgenerators (d.h. der Routine `INITGR`, die das Gitter und die Ableitungen der Transformationen berechnet) bedeutet, die Berechnung der Ableitungen nach den Designparametern, und die Lösung des linearisierten Systems und des transponierten linearisierten Systems, die den Aufruf eines Mehrgitterverfahrens implizieren. Deshalb werden diese Komponenten des Algorithmus im folgenden jeweils extra erwähnt.

Die Schritte (1c), (3), (5a) und (5e) werden durch einen oder mehrere lineare Mehrgitterschritte näherungsweise gelöst. Die Systemmatrizen in (3) und (5a) stimmen mit der Matrix C_x^k im abstrakten Algorithmus überein („Vorwärtssystem“) und die Matrizen in (1c) und (5e) mit $\left(C_x^k\right)^\top$ (transponiertes System).

Man kann die Schritte (1c) (Berechnung der adjungierten Variablen), (2) (KKT–System) und (3) (Schritt in den diskretisierten Variablen) äquivalent umformulieren in:

$$\begin{bmatrix} U_u & U_\rho & U_q \\ D_u & D_\rho & D_q \\ 0 & 0 & G_q \end{bmatrix} \begin{bmatrix} \Delta u \\ \Delta \rho \\ \Delta q \end{bmatrix} = - \begin{bmatrix} c_U \\ c_D \\ g \end{bmatrix},$$

$$\begin{bmatrix} U_u & U_\rho & U_q \\ D_u & D_\rho & D_q \\ 0 & 0 & G_q \end{bmatrix}^\top \begin{bmatrix} \lambda \\ \mu \\ \nu \end{bmatrix} = \begin{bmatrix} \nabla_u f \\ \nabla_\rho f \\ \nabla_q f + B\Delta q \end{bmatrix}.$$

Das erste Gleichungssystem enthält die linearisierten Nebenbedingungen, das zweite die Definition der adjungierten Variablen.

Zu den verwendeten linearen Mehrgitteralgorithmen sind Einzelheiten im Abschnitt 2.6.2 festgehalten.

Die Trust–Region–Variante wurde eingeführt, weil die zunächst errechneten Parameteränderungen Δq als Lösung von (2) in sehr vielen Fällen zu groß wurden, so daß unbrauchbare Profile errechnet wurden und auch unphysikalische Zwischenlösungen (z.B. negative Dichtewerte oder zu hohe Lavalzahlen) entstanden. Wichtig ist noch, daß B^k selbst in (2) unverändert bleibt. Die Diagonalmatrizen werden nur temporär aufaddiert, das KKT–System erneut gelöst, die Größe von $\|\Delta q\|_2$ geprüft usw. Anschaulich bedeutet die Addition großer Diagonalterme, daß die Suchrichtung *in Richtung des reduzierten Gradienten gedreht* wird. In den ersten Iterationen wird die Trust–Region–Begrenzung häufig aktiv, in späteren Iterationen immer seltener (oder gar nicht mehr).

2.5.4 Die Merit–Funktion

Motivation: In diesem und im nächsten Abschnitt wird die Line-search spezifiziert. Zur Vorbereitung brauchen wir den Begriff der *Merit–Funktion.*

Bei der Optimierung sind zwei Ziele im Auge zu behalten:

1. Die vorgegebene Zielfunktion soll minimiert werden.
2. Dabei sind die Nebenbedingungen zu beachten. Spätestens in der „abgelieferten" Lösung müssen sie erfüllt sein.

Wenn man nun unter verschiedenen Approximationen die beste heraussuchen soll — so etwas geschieht bei der Linesearch in einem eindimensionalen Teilraum —, dann reicht es nicht, nur den Wert der Zielfunktion als Kriterium heranzuziehen, weil die Nebenbedingungen in hohem Maße verletzt sein könnten. Während man bei linearen Nebenbedingungen grundsätzlich die Möglichkeit hat, die Suchrichtung so zu wählen, daß alle Punkte zulässig sind, falls man auch zulässig gestartet ist, geht dies bei nichtlinearen Nebenbedingungen i.a. nicht. Jede Suchrichtung führt aus der (gekrümmten) zulässigen Mannigfaltigkeit heraus, selbst wenn zulässig gestartet wurde. Aber der hier beschriebene Algorithmus startet nicht einmal zulässig.

Merit–Funktionen berücksichtigen also außer dem Zielfunktional auch die Nebenbedingungen. Eine häufig verwendete Merit–Funktion ist die von *Powell:*

$$\zeta(x,q,\lambda,\mu) = f_r(x,q) + \sum_{l \text{ Knotenindex}} |\lambda_l c_{U,l}| + \sum_{l \text{ Zentrumindex}} |\mu_l c_{D,l}| \tag{5.1}$$

Zu der regularisierten Zielfunktion wird das Produkt der Absolutwerte der nichtlinearen Defekte und der zugehörigen adjungierten Variablen addiert. Die adjungierten Variablen sorgen dabei für eine angemessene Gewichtung: λ_l ist groß, wenn die l–te Komponente der Stromfunktionsgleichung besonders aktiv ist, d.h. einer weiteren Verminderung der Zielfunktion im Wege steht. Entsprechendes gilt für μ_l.

Die Powellsche Merit–Funktion ist als Unterprogramm `ZETA2` im Directory `zielfkt` implementiert. Mit ihr wurden alle Beispiele, die im Abschnitt 2.7 aufgeführt sind, gerechnet.

Als Alternative, für die aber keine Vergleichszahlen zu den aktuellen Ergebnissen vorliegen, haben wir die *Merit–Funktion von Fletcher* als Unterprogramm `ZETA3` implementiert. Sie ist gegeben durch:

$$\zeta(x;q,\lambda,\mu,\theta) = f_r(x,q) - c_U^T(x)\lambda - c_D^T(x)\mu + \theta\left(c_U^T c_U + c_D^T c_D\right). \tag{5.2}$$

Dabei ist θ ein bei jedem Aufruf vorab zu bestimmender positiver Parameter. Bei hinreichend genauer Berechnung der Suchrichtung liefert der verwendete Algorithmus eine Abstiegsrichtung für den nichtlinearen Defekt, so daß der letzte Term in (5.2) in Suchrichtung abnimmt. θ wird nun (in der Linesearch–Routine) so bestimmt, daß die Merit–Funktion abnimmt. Falls die ersten Terme in (5.2) — sie bilden gerade die regularisierte Lagrange–Funktion — abnehmen, wird θ nicht erhöht.

2.5.5 Die Linesearch

Die Linesearch hat die Aufgabe, entlang einer Suchrichtung, die in unserem Fall durch $(\Delta x, \Delta q)^T$ gegeben ist, den Parameter $\alpha \in [0,1]$ zu finden, für den $(x + \alpha\Delta x, q + \alpha\Delta q)^T$ den minimalen Zielfunktionswert bzw. den minimalen Wert der Merit–Funktion ergibt. Wenn die Ableitung der Merit–Funktion in Suchrichtung negativ ist, d.h. wenn

$$\frac{d}{d\alpha}\zeta(x + \alpha\Delta x, q + \alpha\Delta q) < 0 \tag{5.3}$$

ist, ergibt sich $\alpha > 0$. Die Bestimmung von α erfordert die Lösung eines *eindimensionalen Optimierungsproblems*, für das eine Reihe von Algorithmen zur Verfügung steht. Da die Auswertung der Merit–Funktion in unserem Fall aufwendig ist (die Defektauswertung erfordert z.B. die Aufstellung der geänderten PDGL–Koeffizienten auf dem *ganzen* Gitter), beschränken wir uns auf eine Näherungslösung. Zunächst brauchen wir ein Kriterium dafür, wann die Merit–Funktion hinreichend abgenommen hat. Mit den Abkürzungen

$$y := \begin{pmatrix} x \\ q \end{pmatrix} \qquad \text{und} \qquad \Delta y := \begin{pmatrix} \Delta x \\ \Delta q \end{pmatrix} \tag{5.4}$$

und der Bezeichung $\nabla_y\zeta$ für die Ableitung nach x und q ergibt sich für den Abstieg der im Punkt y durch die Linearisierung ersetzten Merit–Funktion der Wert

$$\Delta\zeta := (\nabla_y\zeta)^T \Delta y. \tag{5.5}$$

Ferner wird eine Zahl $\beta \ll 1$ festgelegt. Im Programm wurde $\beta = 0.01$ gewählt.

Ein „vorgeschlagener" Wert α für den Linesearch–Parameter wird akzeptiert, wenn

$$\zeta(y + \alpha\Delta y) \leq \zeta(y) + \alpha\beta\Delta\zeta \tag{5.6}$$

gilt, sonst wird α mit einem festen Faktor verkleinert und der nächste Versuch gestartet. Als Startwert wird $\alpha = 1.0$ genommen. Um das Verfahren zu terminieren, wird bei Werten $\alpha < \alpha_{\min}$ (im Programm $\alpha_{\min} = 0.05$) die Linesearch mit $\alpha = \alpha_{\min}$ beendet. Dieses Vorgehen in der Linesearch wird auch als Armijo/Goldstein-Strategie bezeichnet.

Wenn überhaupt keine Abstiegsrichtung gefunden wurde, wird ein negativer Wert für α vorgeschrieben (im Programm $\alpha = -0.1$).

Die genannte Linesearch–Strategie ist recht einfach und kann verbessert werden. Da die Merit–Funktion, beurteilt nach den Erfahrungen mit Programmläufen, in unserem Fall nicht überall glatt zu sein scheint, sollte man dafür eine effiziente Methode mit einer robusten Methode verbinden: Z.B. kann man die Merit–Funktion im Parameterintervall $[0,1]$ oder auch auf Teilintervallen durch eine Parabel approximieren und deren Minimum bestimmen. Dies funktioniert gut für glatte Funktionen. Bei Versagen dieser Methode schaltet man z.B. um auf einen Suchalgorithmus, der nur Funktionsauswertungen (und keine Ableitungen) benutzt, z.B. die Methode des Goldenen Schnitts.

2.6 Der Mehrgitteralgorithmus

2.6.1 Das nichtlineare Mehrgitterverfahren

In diesem Abschnitt skizzieren wir kurz das nichtlineare Mehrgitterströmungssimulationsverfahren, das den Ausgangspunkt des Projekts darstellt (kurz „MTU-Programmpaket" o.ä. genannt). Sehr umfassend sind Mehrgitterverfahren in [9] beschrieben. Das hier verwendete Mehrgitterverfahren geht auf eine Darstellung in [10] zurück. Zur Darstellung des Mehrgitterverfahrens gehen wir die Gleichungen noch einmal durch und führen kurze Bezeichnungen ein.

Für die Stromfunktionsgleichung (3.1) definieren wir Differentialoperator und rechte Seite:

$$L(\rho,q)\,u := \nabla\cdot(A(\rho,q)\nabla u) \quad \text{und} \tag{6.1}$$

$$f(q) := -\nabla\cdot C(q). \tag{6.2}$$

Die diskretisierte Stromfunktionsgleichung hat dann die Gestalt

$$L_h(\rho_h,q)\,u_h = f_h(q), \tag{6.3}$$

wobei die Diskretisierungen der Formeln (3.5) bis (3.9) einzusetzen sind.

Die verschiedenen Randbedingungen lassen sich mit Hilfe eines Randoperators (analog zum 9–Punkt–Differenzenoperator für die PDGL) schreiben: Die (diskreten) Dirichlet–Bedingungen (3.37) und (3.38) sind 1–Punkt–Sterne, die periodischen Randbedingungen (3.41) und (3.43) sind 2–Punkt–Sterne und die Winkelbedingungen (3.52) 4–Punkt–Sterne. Die Dichte und die Spline–Parameter kommen in keiner dieser Bedingungen vor, und alle Bedingungen sind linear in u_h, so daß die Schreibweise

$$R_h u_h = k_h \tag{6.4}$$

mit einer geeigneten rechten Seite k_h für die Gesamtheit aller diskretisierten Randbedingungen möglich ist. In der folgenden Beschreibung des Mehrgitterverfahrens werden die Randbedingungen nicht mehr explizit genannt.

Für die Dichtegleichung (3.23) diskretisieren wir die Funktion Γ aus (3.30) und definieren eine rechte Seite g:

$$\Gamma_h(u_h,\rho_h,q) := 1-\Big(\frac{\rho_h}{\rho_{t,h}}\Big)^{\kappa-1} - \frac{\kappa-1}{\kappa+1}\,\mu\Big(\Big(\frac{w_h}{a_{La,h}}\Big)^2\Big), \tag{6.5}$$

$$g_h := 0. \tag{6.6}$$

Bei der Rekursion innerhalb des Mehrgitterverfahrens treten auch nichttriviale rechte Seiten auf. Die modifizierte Dichtegleichung hat dann die Gestalt

$$\Gamma_h(u_h,\rho_h,q) = g_h, \tag{6.7}$$

wobei die Diskretisierungen der Formeln (3.25) und (3.26) einzusetzen sind.

Das Mehrgitterverfahren werde mit $\texttt{mgm}(L_h,u_h,f_h,\Gamma_h,\rho_h,g_h)$ bezeichnet, wobei in der Argumentliste für beide Gleichungen der Operator, die „gesuchte" Variable und die rechte Seite aufgezählt sind. Die Kopplungen zwischen den Gleichungen sind den ausführlichen Definitionen zu entnehmen.

(0) Falls $h = h_{\max}$ ist (gröbstes Gitter), einen exakten Löser aufrufen oder ausreichend viele Relaxationen rechnen: `CALL exact`. Weiter mit (5).
Sonst: Anfangen mit (1).

(1) Vorglättung (ν_1 Iterationen)

$$u_h := \texttt{relax}(L_h, u_h, f_h, \rho_h) \quad \text{Linien–GS in den Knoten;} \tag{6.8}$$
$$\rho_h := \texttt{relax}(\Gamma_h, \rho_h, g_h, u_h) \quad \text{Punktweise Lösung in den Zentren;} \tag{6.9}$$

(2) Defektberechnung

$$d_h := f_h - L_h u_h \quad \text{Defekt für die Stromfunktion;} \tag{6.10}$$
$$\delta_h := g_h - \Gamma_h(\rho_h) \quad \text{Defekt für die Dichte;} \tag{6.11}$$

(3) Restriktionen

$$u_{2h} := I_h^{2h} u_h \quad \text{und} \quad d_{2h} := I_h^{2h} d_h \quad \text{in den Knoten;} \tag{6.12}$$
$$\rho_{2h} := I_h^{2h} \rho_h \quad \text{und} \quad \delta_{2h} := I_h^{2h} \delta_h \quad \text{in den Zentren;} \tag{6.13}$$

(4) Rekursiver i_g-facher Mehrgitteraufruf

$$f_{2h} := d_{2h} + L_{2h} u_{2h}; \tag{6.14}$$
$$g_{2h} := \delta_{2h} + \Gamma_{2h}(\rho_{2h}); \tag{6.15}$$

`CALL mgm`$(L_{2h}, \hat{u}_{2h}, f_{2h}, \Gamma_{2h}, \hat{\rho}_{2h}, g_{2h})$, i_g-fach;
(Dies ist ein Sprung nach (0).)

(5) Berechnung und Interpolation der Korrektur

$$v_h := I_{2h}^h(\hat{u}_{2h} - u_{2h}); \tag{6.16}$$
$$\sigma_h := I_{2h}^h(\hat{\rho}_{2h} - \rho_{2h}); \tag{6.17}$$

(6) Korrektur der aktuellen Iterierten

$$u_h := u_h + v_h; \tag{6.18}$$
$$\rho_h := \rho_h + \sigma_h; \tag{6.19}$$

(7) Nachglättung (ν_2 Iterationen)

$$u_h := \texttt{relax}(L_h, u_h, f_h, \rho_h) \quad \text{Linien–GS in den Knoten;} \tag{6.20}$$
$$\rho_h := \texttt{relax}(\Gamma_h, \rho_h, g_h, u_h) \quad \text{Punktweise Lösung in den Zentren;} \tag{6.21}$$

(8) Abfrage und Verzweigung:
Vorgegebene Schrittzahl erreicht? Falls nein, gehe nach (1). Falls ja:
Aktuelles Gitter = feinstes Gitter? Falls nein, gehe nach (5). Falls ja:
Fertig!

Die Relaxationen

Für die Stromfunktionsgleichung wird eine blockweise Gauß–Seidel–Relaxation verwendet, die alternierend x-Gitterlinien (d.h. Linien, parallel zur x-Achse) und y-Gitterlinien (d.h. Linien, parallel zur y-Achse) zu Blöcken zusammenfaßt.

Für x-Gitterlinien ist es günstig, das 9–Punkt–Schema folgendermaßen als Blocktridiagonalschema mit tridiagonalen Blöcken zu schreiben:

$$L_h(\rho_h, q) =: \begin{pmatrix} T_1 & C_2 & & & \\ B_1 & T_2 & C_3 & & \\ & B_2 & \ddots & \ddots & \\ & & \ddots & \ddots & C_M \\ & & & B_{M-1} & T_M \end{pmatrix}. \tag{6.22}$$

Im ersten Fall, in dem Gitterzeilen zusammengefaßt werden, ergibt sich eine $M \times M$–Blockstruktur mit $N \times N$–Blöcken. Die u_h–Variablen und die rechte Seite f_h sollen auch zeilenweise zu Blöcken zusammengefaßt werden:

$$\bar{u}_j := (u_{1j}, u_{2j}, \ldots, u_{Nj}) \qquad \text{für } j = 1, 2, \ldots, M, \tag{6.23}$$

$$\bar{f}_j := (f_{1j}, f_{2j}, \ldots, f_{Nj}) \qquad \text{für } j = 1, 2, \ldots, M. \tag{6.24}$$

Dann lassen sich die diskretisierten Stromfunktionsgleichungen (und Randbedingungen), die zur j-ten Gitterzeile gehören, so schreiben:

$$B_{j-1}\bar{u}_{j-1} + T_j\bar{u}_j + C_{j+1}\bar{u}_{j+1} = \bar{f}_j\,. \tag{6.25}$$

Das x–Linien–Gauß–Seidel–Verfahren löst nun zeilenweise im j-ten Schritt nach dem j-ten Diagonalblock auf, wobei die im vorigen Schritt ($\bar{u}_{j-1}^{\text{alt}} \to \bar{u}_{j-1}^{\text{neu}}$) errechneten Werte gleich wieder eingesetzt werden:

$$\bar{u}_j^{\text{neu}} := T_j^{-1}\left(\bar{f}_j - B_{j-1}\bar{u}_{j-1}^{\text{neu}} - C_{j+1}\bar{u}_{j+1}^{\text{alt}}\right) \qquad (j\text{-te Gitterzeile}). \tag{6.26}$$

Im zweiten Fall, in dem die Gitterlinien parallel zur y-Achse zu Blöcken zusammengefaßt werden, liegen die für einen Block benötigten Matrixeinträge von L_h und die Vektorkomponenten von u_h nicht mehr hintereinander, sondern „verstreut". Daher bietet sich zur Darstellung des Algorithmus folgendes Vorgehen an: Wir transformieren die Stromfunktionsgleichung mit einer Permutation P, so daß in

$$v_h := Pu_h \qquad \text{und in} \qquad g_h := Pf_h \tag{6.27}$$

die Komponenten entlang der y-Richtung hintereinander stehen und wenden den Algorithmus (6.26) an. Man erhält:

$$M_h v_h = g_h \qquad \text{mit} \tag{6.28}$$

$$M_h = PL_hP^{\mathsf{T}}. \tag{6.29}$$

Jetzt entsteht eine $N \times N$–Blockstruktur mit $M \times M$–Blöcken, wieder blocktridiagonal mit tridiagonalen Blöcken:

$$M_h(\rho_h, q) =: \begin{pmatrix} S_1 & E_2 & & & \\ D_1 & S_2 & E_3 & & \\ & D_2 & \ddots & \ddots & \\ & & \ddots & \ddots & E_N \\ & & & D_{N-1} & S_N \end{pmatrix}. \tag{6.30}$$

Mit den Abkürzungen

$$\bar{v}_l := (u_{l1}, u_{l2}, \ldots, u_{lM}) \quad \text{für } l = 1, 2, \ldots, N, \tag{6.31}$$

$$\bar{g}_l := (f_{l1}, f_{l2}, \ldots, f_{lM}) \quad \text{für } l = 1, 2, \ldots, N, \tag{6.32}$$

mit denen Gitterspalten zu Blöcken zusammengefaßt werden, lautet (6.28):

$$D_{l-1}\bar{v}_{l-1} + S_l\bar{v}_l + E_{l+1}\bar{v}_{l+1} = \bar{g}_l\,. \tag{6.33}$$

Die y–Linien–Gauß–Seidel–Iteration ist gegeben durch

$$\bar{v}_l^{\text{neu}} := S_l^{-1}\left(\bar{g}_l - D_{l-1}\bar{v}_{l-1}^{\text{neu}} - E_{l+1}\bar{v}_{l+1}^{\text{alt}}\right) \qquad (l\text{-te Gitterspalte}). \tag{6.34}$$

Für die Dichtegleichung wird punktweise die folgende Zuweisung gerechnet:

$$\rho_h^{\text{neu}} := \rho_{t,h}\left[1 - \frac{\kappa - 1}{\kappa + 1}\,\mu^2\left(\frac{w_h^{\text{neu}}}{a_{La,h}}\right) - g_h\right]^{\frac{1}{\kappa-1}}. \tag{6.35}$$

Grobgitterlöser
Block–Gauß–Seidel–Verfahren mit zwei großen Blöcken für u und ρ.

Gittertransfer
Siehe Abschnitt 2.6.2.

2.6.2 Das lineare Mehrgitterverfahren

Motivation: Das lineare Mehrgitterverfahren wurde zusätzlich zum bereits existierenden nichtlinearen Mehrgitterverfahren programmiert, weil der verwendete Optimierungsalgorithmus die Linearisierung des Problems einsetzt. Die Methode, durch Bestimmung der adjungierten Variablen den reduzierten Gradienten zu bilden (also die totale Ableitung der Zielfunktion nach den Designparametern bei eliminierten Nebenbedingungen), benötigt ferner das adjungierte System. Dies ist nach einer Linearisierung einfacher zu bilden.

Als roter Faden bei der Konstruktion des linearen Mehrgitterverfahrens diente der Gedanke, sich möglichst eng an den nichtlinearen Algorithmus anzulehnen. Dabei mußte grundsätzlich über *alle* Komponenten eines Mehrgitterverfahrens nachgedacht werden, das sind:

1. Gittertransfer
2. Erzeugung von Grobgittermatrizen
3. Grobgitterlöser
4. Relaxationsverfahren als Glätter

Außerdem ist zu beachten, daß für die transponierten Gleichungen ein passender Algorithmus gefunden wird, der zu der Lösung des Vorwärtssystems korrespondiert. Diese Kopplung wird nahegelegt durch den folgenden Zusammenhang:

Wir definieren

$$p := \Delta q \qquad \text{und} \qquad \tilde{f}(p) := f(x - C_x^{-1} C_q p, q + p) \tag{6.36}$$

für einen beliebigen Punkt (x, q). Dann kann der reduzierte Gradient als Ableitung geschrieben werden,

$$\gamma^\top = \frac{\partial}{\partial p} \tilde{f}(p). \tag{6.37}$$

Nun kann man die Technik der *adjungierten Differentiation* anwenden, die in [1, 3] benutzt wurde, um adjungierte Schemata zur numerischen Integration von Systemen gewöhnlicher Differentialgleichungen herzuleiten. Es gilt:

Satz 2.6.1 *Das reguläre lineare System $C_x \Delta x + c = 0$ werde iterativ gelöst mit der Iteration $\Delta x^k = \Delta x^{k-1} - B^k(C_x \Delta x^{k-1} + c)$, $k = 1, \ldots, n$. Auf diese Weise wird ein approximativer Lösungsoperator $\tilde{C}_x^{-1}$ definiert. Dann gilt für den approximativen reduzierten Gradienten, der durch $\tilde{\gamma}^\top := \partial f(x - \tilde{C}_x^{-1} C_q p, q + p)/\partial p$ definiert wird, die folgende Gleichung:*

$$\tilde{\gamma} = \tilde{\gamma}_n := \nabla_q f - C_q^\top \lambda^1 \qquad \textit{mit} \tag{6.38}$$

$$\lambda^{k-1} = \lambda^k - \left(B^k\right)^\top \left(C_x^\top \lambda^k - \nabla_x f\right) \qquad \textit{für} \quad k = n, \ldots, 2, \tag{6.39}$$

$$\lambda^n = (B^n)^\top \nabla_x f. \tag{6.40}$$

(6.39) und (6.40) definieren eine Rückwärtsrekursion für die Folge $\{\lambda^k\}_{k=1}^n$.

Beweis: Siehe [17]. ∎

Es hat sich herausgestellt, daß man am besten das Mehrgitterverfahren für das Vorwärtssystem transponiert, um ein gutes Verfahren für das transponierte System zu erhalten. Alle Operationen werden in umgekehrter Reihenfolge durchlaufen, jede Operation für sich wird transponiert. Bei genauerem Hinsehen ist diese Wahl gar nicht so klar, wie sie jetzt erscheinen mag. Nun sollen die beiden Algorithmen vorgestellt werden.

2.6.2.1 Mehrgitterverfahren für das linearisierte System

Die Ableitungen der Stromfunktionsgleichung nach der Stromfunktion und nach der Dichte bezeichnen wir mit U_u und U_ρ, die Ableitungen der Dichtegleichung analog mit D_u und D_ρ. Die nichtlinearen Defekte (Residuen), also die Differenz aus dem Term mit dem Differentialoperator und der rechten Seite, bezeichnen wir mit c_U und c_D. Damit läßt sich *ein Newton–Verfahren* zur Lösung des nichtlinearen Simulationsproblems schreiben als:

(0) Starte in u^0, ρ^0; setze $k := 0$.

(1a) Berechne die *Linearisierung* in (u^k, ρ^k):
$$\begin{bmatrix} U_u^k & U_\rho^k \\ D_u^k & D_\rho^k \end{bmatrix}$$
Werte die Residuen aus: c_U^k, c_D^k

(1b) Berechne den *Schritt* $(\Delta u^k, \Delta \rho^k)$ als Lösung des Systems:
$$\begin{bmatrix} U_u^k & U_\rho^k \\ D_u^k & D_\rho^k \end{bmatrix} \begin{bmatrix} \Delta u^k \\ \Delta \rho^k \end{bmatrix} = - \begin{bmatrix} c_U^k \\ c_D^k \end{bmatrix}$$

(2) Addiere die Korrektur zur alten Iterierten:
$u^{k+1} := u^k + \Delta u^k$
$\rho^{k+1} := \rho^k + \Delta \rho^k$

(3) Falls die Residuen noch nicht klein genug sind, setze $k := k + 1$ und gehe nach (1).

Zur approximativen Lösung des Systems in (1b) wird das lineare Mehrgitterverfahren gebraucht.

1. Gittertransfer Im MTU–Programmpaket waren relativ aufwendige Interpolationen p und Restriktionen r implementiert. Da der Differentialoperator für die Stromfunktion u von 2. Ordnung ist und die Dichte ρ nur algebraisch vorkommt („0. Ordnung"), würde man als Ansatzräume $H^1(\Omega)$ für u und $L^2(\Omega)$ für ρ in der schwachen Formulierung der Gleichungen vorsehen. Dafür reichen dann aber die *bilineare* Interpolation für u (definiert in den Knoten) und die *konstante* Interpolation für ρ (definiert in den Zentren). Praktische Rechnungen bestätigen dies. Die Auswahl möglichst einfacher Transferoperationen bietet zwei Vorzüge: Einerseits wird die Programmierung erheblich vereinfacht, besonders für randnahe Punkte. Zum anderen ist die Konstruktion adjungierter Operatoren viel einfacher. Dagegen spielt der verringerte Rechenaufwand nur eine untergeordnete Rolle.

Während die Wahl der Operatoren für Zentren ganz einfach ist (keine Randdaten), sind für die in den Knoten definierten Größen besondere Überlegungen erforderlich:

1. Ein wichtiges Unterscheidungsmerkmal für verschiedene Varianten eines Gittertransferoperators ist die *Behandlung der Randpunkte und der randnahen Punkte*, die einen Randpunkt als nächsten Nachbarn haben. Wir unterscheiden zwischen Operatoren, die *„Bereiche trennen"* und solchen, die *„Bereiche nicht trennen"*. Mit „Bereich" meinen wir eine Menge von Gitterpunkten, in denen eine bestimmte Gleichung gilt: So gilt etwa in allen inneren Gitterknoten die Stromfunktionsgleichung, aber in den Randknoten am Eintritts– und am Austrittsrand die Winkelbedingung. Jedem Gleichungstyp ordnen wir so einen Gitterbereich zu. Wir sagen, daß ein Operator die Bereiche trennt, wenn beim Gittertransfer nicht Funktionswerte aus verschiedenen Bereichen zur Berechnung eines Wertes auf dem anderen Gitter verwendet werden. Sonst sagen wir, daß ein Operator die Bereiche nicht trennt oder sie vermischt. Zwei Beispiele:

a. Die bilineare Interpolation der Stromfunktion u verwendet auf dem ganzen Gitter denselben Stern, so daß die randnahen Gitterpunkte des feinen Gitters aus Randpunkten *und* aus randnahen Gitterpunkten des groben Gitters errechnet werden: Diese Interpolation trennt

die Bereiche *nicht*. Das ist auch sinnvoll, weil die physikalische Bedeutung der Variable u dieselbe ist, unabhängig vom Ort des Gitterpunkts.

b. Die Restriktion (Gittertransfer vom feinen zum groben Gitter) dagegen wird auf den Defekt c_U angewendet. Da der Gleichungstyp sich am Rand ändert, empfiehlt es sich, bei der Restriktion die Bereiche zu trennen. Der 9–Punkt–Stern im Inneren wird dann durch einen 3–Punkt–Stern am Rand ersetzt, der nur Randwerte verwendet.

Mit dieser Vorgehensweise werden die besten Ergebnisse erzielt.

2. Ein weiteres Unterscheidungsmerkmal ist die *Normierung*. Wir ziehen die Zeilensummennorm $\|\cdot\|_\infty$ heran, für lineare Operatoren p und r. Bei allen hier auftretenden Operatoren treten nur nichtnegative Einträge auf. p hat genau dann in jeder Zeile die Zeilensumme 1, wenn alle konstanten Gitterfunktionen wieder *auf dieselbe Konstante abgebildet werden*. Insbesondere ist dann $\|p\|_\infty = 1$. Entsprechendes gilt für r.

3. Das letzte Unterscheidungsmerkmal ist die Eigenschaft der Periodizität. Da der Nordrand des Rechengebietes durch eine periodische Randbedingung mit dem Südrand verbunden ist, kann ein Restriktionsoperator definiert werden, der auch Punkte aus dem Fortsetzungsgebiet erfaßt. Für die Anwendung auf den Defekt ist dies eine mögliche Variante. Bei der Anwendung auf die Stromfunktion u muß der Periodenversatz zwischen Nord– und Südrand berücksichtigt werden, so daß ein solcher Operator nicht mehr linear ist.

Welche Merkmale auf die von uns verwendeten Operatoren zutreffen, wird im nächsten Unterabschnitt 2.6.2.2 behandelt: Wenn man nur das „Vorwärtssystem“ mit einem Mehrgitteralgorithmus lösen will, findet man viele Varianten. Durch das Studium des transponierten Systems ergeben sich jedoch interessante Rückschlüsse, die eine Auswahl ermöglichen.

2. Erzeugung von Grobgittermatrizen Im Programm werden die Grobgittermatrizen durch sukzessive Linearisierung berechnet. Dazu werden die aktuellen Variablen u und ρ von dem feinsten Gitter auf alle gröberen Gitter restringiert und als Auswertestellen für die Linearisierung benutzt. Bei dieser Vorgehensweise konvergiert das lineare Mehrgitterverfahren und auch das Newton–Verfahren.

Eine Standardmöglichkeit zur Erzeugung der Grobgittermatrizen bei linearen Problemen ist der *Galerkin–Ansatz*, den wir zunächst in Betracht gezogen haben und der für das vorliegende System die Form

$$\begin{bmatrix} U_{u,2h} & U_{\rho,2h} \\ D_{u,2h} & D_{\rho,2h} \end{bmatrix} = \begin{bmatrix} r_{u,h} & 0 \\ 0 & r_{\rho,h} \end{bmatrix} \begin{bmatrix} U_{u,h} & U_{\rho,h} \\ D_{u,h} & D_{\rho,h} \end{bmatrix} \begin{bmatrix} p_{u,2h} & 0 \\ 0 & p_{\rho,2h} \end{bmatrix} \tag{6.41}$$

annimmt. Dabei bezieht sich der Gitterindex h bzw. $2h$ bei den Transferoperationen immer auf den Urbildraum, also bedeutet $p_{u,2h} : \Omega_{2h} \to \Omega_h$, $u_{2h} \mapsto u_h$ eine Interpolation vom groben Knotengitter (Index u) auf das feine Knotengitter, $r_{u,h} : \Omega_h \to \Omega_{2h}$, $f_h \mapsto f_{2h}$ eine Restriktion vom feinen Knotengitter auf das grobe. Entsprechendes gilt für die Transferoperationen zwischen den zentrierten Gittern.

Der einfacheren Programmierung wegen haben wir jedoch der erstgenanten Möglichkeit den Vorzug gegeben.

3. Grobgitterlöser Als Grobgitterlöser dient ein Block–Gauß–Seidel–Verfahren, zur Abkürzung Block–GS–Verfahren, das Gleichungen vom Typ (1b) aus dem Newton–Verfahren auf folgende Weise iterativ löst (aber eben nur auf groben Gittern):

(0) Starte mit $\Delta u^0 = 0$, $\Delta \rho^0 = 0$, setze $i := 0$.

(1) Löse $U_u \Delta u^{i+1} = -c_U - U_\rho \Delta \rho^i$,
löse $D_\rho \Delta \rho^{i+1} = -c_D - D_u \Delta u^i$.

(2) Falls das lineare Residuum
$$\begin{bmatrix} U_u & U_\rho \\ D_u & D_\rho \end{bmatrix} \begin{bmatrix} \Delta u^{i+1} \\ \Delta \rho^{i+1} \end{bmatrix} + \begin{bmatrix} c_U \\ c_D \end{bmatrix}$$
noch nicht klein genug ist, setze $i := i + 1$ und gehe nach (1).

Der Index k der äußeren Newton–Iteration wurde weggelassen, und der Iterationszähler i für die Block–GS–Schritte eingeführt.

In dieser Grundversion konvergiert der Grobgitterlöser noch nicht, und er entspricht auch nicht dem nichtlinearen Löser aus dem MTU–Programmpaket. Dort wird nämlich die Dichtegleichung mit der Fixpunktiteration (6.35) relaxiert. Dies kann auf folgende Weise im Linearen nachvollzogen werden:

Aus der zweiten Gleichung in (3.23) ergibt sich der Zusammenhang

$$\frac{w^2}{a_{\mathrm{La}}^2} = \frac{1}{\rho^2}\eta(u,q), \tag{6.42}$$

wobei η eine Funktion ist, die von u und q, aber nicht von ρ abhängt. Die modifizierte Dichtegleichung $\Gamma(u,\rho,q) = g$ läßt sich jetzt mit

$$\Gamma(u,\rho,q) = 1 - \left(\frac{\rho}{\rho_t}\right)^{\kappa-1} - \frac{\kappa-1}{\kappa+1}\mu\left(\frac{\eta(u,q)}{\rho^2}\right) \tag{6.43}$$

schreiben, vgl. (3.30). Aus dieser Form entnimmt man bequem die funktionale Abhängigkeit der Funktion Γ von ρ. Definiert man eine Hilfsfunktion

$$\hat{\Gamma}(u,\rho_1,\rho_2,q) := 1 - \left(\frac{\rho_1}{\rho_t}\right)^{\kappa-1} - \frac{\kappa-1}{\kappa+1}\mu\left(\frac{\eta(u,q)}{\rho_2^2}\right) \tag{6.44}$$

in der die beiden Terme mit ρ künstlich unterschieden werden, so kann man zwei partielle Ableitungen nach „der“ Dichte bilden:

$$\frac{\partial\hat{\Gamma}}{\partial\rho_1}(u,\rho_1,\rho_2,q) = -\frac{\kappa-1}{\rho_1}\left(\frac{\rho_1}{\rho_t}\right)^{\kappa-1} \quad \text{und} \tag{6.45}$$

$$\frac{\partial\hat{\Gamma}}{\partial\rho_2}(u,\rho_1,\rho_2,q) = 2\frac{\kappa-1}{\kappa+1}\frac{1}{\rho_2}\frac{\eta(u,q)}{\rho_2^2}\mu'\left(\frac{\eta(u,q)}{\rho_2^2}\right) \tag{6.46}$$

Damit erhält man die folgende Aufspaltung der diagonalen Ableitungsmatrix D_ρ:

$$D_\rho = \frac{\partial\Gamma}{\partial\rho}(u,\rho,q) = \frac{\partial\hat{\Gamma}}{\partial\rho_1}(u,\rho,\rho,q) + \frac{\partial\hat{\Gamma}}{\partial\rho_2}(u,\rho,\rho,q). \tag{6.47}$$

Die beiden Terme auf der rechten Seite der Gleichung bezeichnen wir mit $D_{\rho,1}$ und $D_{\rho,2}$. Nun kann die nichtlineare Fixpunktiteration nachgebildet werden, indem man im Block–GS–Verfahren (1) ersetzt durch

(1') Löse $U_u \Delta u^{i+1} = -c_U - U_\rho \Delta \rho^i$,
löse $D_{\rho,1} \Delta \rho^{i+1} = -c_D - D_u \Delta u^i - D_{\rho,2} \Delta \rho^i$.

Mit dieser Modifikation konvergiert das Block–GS–Verfahren. Wir haben eine so ausführliche Darstellung gewählt, weil dieser Punkt eine der Hürden war, die beim Finden und Implementieren des Algorithmus auftraten.

4. Relaxationsverfahren als Glätter Den verwendeten Glättern liegt ebenfalls die Aufspaltung $D_\rho = D_{\rho,1} + D_{\rho,2}$ zugrunde. Der Rahmen ist dasselbe Block–GS–Verfahren wie beim Grobgitterlöser auch. Nur wird jetzt das erste lineare Gleichungssystem mit der Matrix U_u nicht mehr exakt gelöst, sondern erneut mit einem Block–GS–Verfahren mit kleineren Blöcken: Erst werden Gitterzeilen zu Blöcken zusammengefaßt, dann Gitterspalten. Für jede Zeile oder Spalte ist ein tridiagonales System zu lösen. All dies geschieht genauso, wie es im Abschnitt 2.6.1 für die nichtlinearen Gleichungen beschrieben wurde.

Das zweite Gleichungssystem mit $D_{\rho,1}$ ist ohnehin diagonal und kann sofort korrekt gelöst werden.

Trotzdem konnte der vorhandene Block–GS–Glätter nicht ohne Modifikation übernommern werden: Da der äußere Rahmen durch ein Newton–Verfahren gegeben ist, beziehen sich alle Gleichungen auf *Korrekturen* von Variablen und nicht auf die Variablen selbst. So entfällt z.B. der Periodenversatz der Stromfunktion bzw. ist durch das Residuum der periodischen Randbedingungen zu ersetzen.

2.6.2.2 Mehrgitterverfahren für das transponierte System

Der Übergang von der Systemmatrix C_x auf die transponierte Matrix C_x^T ist nicht so harmlos, wie er zunächst aussieht. Dies gilt besonders für den Anteil U_u der Stromfunktion, in dem mit einem 9–Punkt–Stern diskretisiert wird. Was geschieht, kann man am besten an den Randbedingungen verfolgen, die nur im U_u–Block auftreten. Z.B. betrachtet man eine Zeile von U_u, in der eine periodische Randbedingung gilt: Zwei Einträge in dieser Zeile verbinden je einen u–Wert am Nord– und Südrand miteinander. Nach dem Transponieren landen diese Einträge aber in verschiedenen Zeilen, weil sie vorher in verschiedenen Spalten gestanden haben. Es gibt also keine Gleichung mehr, die diese beiden Punkte periodisch verbindet oder, anders ausgedrückt: Gleichungen, die sich als periodische Randbedingung interpretieren lassen, treten in U_u^T gar nicht auf. Stattdessen gibt es eine Kopplung mit Variablen, deren Einträge bei U_u in derselben *Spalte* standen. So werden jetzt völlig verschiedene Skalierungen (Abhängigkeiten von der Schrittweite h) in den neuen Gleichungen vermischt. *U_u^T ist nicht mehr als Diskretisierung eines Randwertproblems interpretierbar.* Dies hat Konsequenzen für die Wahl der Mehrgitterkomponenten, insbesondere für den Gittertransfer.

1. Gittertransfer Im vorigen Abschnitt wurde in der entsprechenden Rubrik erläutert, daß die für Systeme C_x benutzte Interpolation p die Bereiche nicht trennt und daß die Restriktion r die Bereiche trennt. Durch Adjungieren (Kennzeichnung der Operatoren mit *) entsteht aus

p eine Restriktion $r' := p^*$, die die Bereiche nicht trennt, und aus r entsteht eine Interpolation $p' := r^*$, die die Bereiche trennt. Die Striche ($'$) sollen Transferoperatoren für das transponierte System $C_x^\top$ kennzeichnen. Dieses Verhalten von r' und p' ist auch angemessen: In $U_u^\top$ haben alle Gleichungen dieselbe Skalierung (h^{-2}), da in jeder Gleichung Terme aus der Diskretisierung des Differentialoperators vorkommen. Daher „darf man" für C_x bei der Restriktion alle Gleichungen mischen (und sollte dies auch tun). Und bei der Interpolation der adjungierten Variablen ist zu beachten, daß diese den Gleichungen des Vorwärtssystems zugeordnet sind, also in Randnähe nicht miteinander gemischt werden dürfen. *Mit einem konventionellen Gittertransfer konvergiert das Mehrgitterverfahren für das transponierte System nicht.*

Die *Bildung des adjungierten Operators* soll noch genauer erläutert werden. Dazu muß auf jedem Gitter ein *Skalarprodukt* festgelegt werden. Wir definieren für Gittervektoren v, w

$$(v, w)_h := h^2 \sum_{l \in I_l} v_l \, w_l, \tag{6.48}$$

wobei I_l die Indexmenge der Gitterpunkte ist, die zur Diskretisierung Ω_h der Maschenweite h gehören. Werden v und w als kontinuierliche Funktionen über dem Gebiet Ω aufgefaßt, so approximiert $(.,.)_h$ das $L^2(\Omega)$–Skalarprodukt

$$(v, w) := \int_\Omega v(x) \, w(x) \, dx. \tag{6.49}$$

Die adjungierten Operatoren $p^* : \Omega_h \to \Omega_{2h}$ und $r^* : \Omega_{2h} \to \Omega_h$ werden durch die folgenden Gleichungen definiert:

$$(p^* u, v)_{2h} = (u, pv)_h \qquad \forall u \in \Omega_h \quad \forall v \in \Omega_{2h}, \tag{6.50}$$

$$(r^* v, u)_h = (v, ru)_{2h} \qquad \forall u \in \Omega_h \quad \forall v \in \Omega_{2h}. \tag{6.51}$$

Einsetzen der Einheitsvektoren für u und v ergibt sofort:

$$p^* = \frac{1}{4} p^\top, \quad r^* = 4 r^\top. \tag{6.52}$$

Für die Norm $\|.\|_\infty$ dieser Operatoren ergeben sich ebenfalls unerwartete Konsequenzen. Aus einem einfachen Abzählargument folgt, daß nicht gleichzeitig p und p^* normiert sein können, ebensowenig r und r^*: $n_{x,h}$ bezeichne die Anzahl der Gitterpunkte in x–Richtung, $n_{y,h}$ die Anzahl der Gitterpunkte in y–Richtung und $n_h := n_{x,h} \cdot n_{y,h}$ die Gesamtzahl der Gitterpunkte von Ω_h. Mit den Anzahlen auf dem nächstgröberen Gitter (Ω_{2h}) ergibt sich der Zusammenhang:

$$n_{x,h} = 2 \cdot n_{x,2h} - 1, \tag{6.53}$$

$$n_{y,h} = 2 \cdot n_{y,2h} - 1, \tag{6.54}$$

$$n_h = 4 n_{2h} - 2(n_{x,2h} + n_{y,2h}) + 1 < 4 n_{2h}. \tag{6.55}$$

Hat p_{2h} in *jeder* Zeile die Zeilensumme 1, dann addieren sich sämtliche Matrixeinträge von p_{2h} zu n_h. Für $p_{2h}^* = \frac{1}{4} p_{2h}^\top$ ist die Summe über alle Einträge $n_h/4 < n_{2h}$. Also muß p_{2h} mindestens eine (es sind natürlich mehrere) Zeilen haben mit einer Zeilensumme < 1. Entsprechend gilt: Hat r_h in *jeder* Zeile die Zeilensumme 1, dann gibt es in $r_h^* = \frac{1}{4} r_h^\top$ Zeilen mit Zeilensumme > 1.

Die Trennung der Bereiche und die Wahl der Norm sind wesentlich für die Konstruktion des Mehrgitterverfahrens für $C_x^\top$. Dagegen hat die Frage nach der Periodizität keine signifikante Rolle gespielt, so daß wir sie hier nicht weiter behandeln.

2. Erzeugung von Grobgittermatrizen Die Grobgittermatrizen werden durch Transponieren der Grobgittermatrizen des Vorwärtssystems übernommen. Sie entstehen also durch sukzessive Linearisierung nach Restriktion von u und ρ.

3. Grobgitterlöser Es wird wieder das Block–GS–Verfahren aus dem letzten Abschnitt verwendet. Da D_ρ diagonal ist, kann im transponierten System genau dieselbe Aufspaltung $D_\rho = D_{\rho,1} + D_{\rho,2}$ verwendet werden.

4. Relaxationsverfahren als Glätter Auch hier wird das Verfahren für das Vorwärtssystem transponiert. Einige Tests wurden vorgenommen, um eine günstige Reihenfolge von Gitterzeilen und Gitterspalten zu finden, nach denen im Glätter aufgelöst wird (jeweils ein tridiagonales System). In der aktuellen Version wird mit den Gitter*zeilen* begonnen, in der Reihenfolge `j = npy-1,npy-2,...,2,1,npy`. Dann kommen die Gitterspalten `i = npx,npx-1,...,2,1` sowie einige Extra–Relaxationen am Profilanfang und am Profilende, wie es auch im MTU–Programmpaket gemacht wurde. (`npx` = Anzahl der Gitterpunkte in x–Richtung, `npy` entsprechend.) Zum Finden dieser Reihenfolge wurde im Besetzungsmuster von U_u^T nachgesehen, welche Variablen welche anderen Variablen beeinflussen, so daß eine Folge von tridiagonalen Systemen entsteht, in der die gerade errechneten Werte gleich im nächsten System Verwendung finden. Auswahlkriterien waren dann die erzielten Konvergenzraten insgesamt und das Verhalten in den ersten Schritten, weil ja immer nur wenige Iterationen mit dem Relaxationsverfahren gerechnet werden.

Zwischendurch wurde auch immer wieder das Vorwärtssystem mit der umgekehrten Reihenfolge der Teiloperationen relaxiert, um sicherzustellen, daß die Glätter für *beide* Systeme konvergieren. Schließlich ist noch zu bemerken, daß die vorgestellte Reihenfolge nicht die einzige gute Lösung ist.

2.7 Numerische Ergebnisse

In diesem Abschnitt sollen die mit dem Optimierungsverfahren erzielten Ergebnisse dargestellt werden. Zu Beginn der Rechnung mußten wir uns für ein „Modell" entscheiden: Damit die Nebenbedingungen hinreichend oft differenzierbar sind, mußte die Begrenzung der Lavalzahlauswertung, die im MTU-Simulationsprogrammpaket durch Minimumbildung mit 1.0 vorgenommen wurde, mit Hilfe einer glatten Funktion geschehen. Dazu wurde eine dreimal stetig differenzierbare Funktion verwendet, die stückweise aus Polynomen zusammengesetzt ist. Es haben sich im Vergleich verschiedener Varianten dieser Begrenzungsfunktion bei der Simulation kaum Unterschiede ergeben. Die Lavalzahlplots für die Strömungsgeschwindigkeiten an den Schaufeln waren kaum voneinander zu unterscheiden. Auch das ursprünglich bei MTU verwendete Modell mit dem Maximum wurde mit in diesen Vergleich eingeschlossen.

In jedem Beispiel wird eine Lavalzahlverteilung am Schaufelprofil entlang vorgegeben. Im Laufe des partiell reduzierten SQP–Verfahrens werden die Designparameter des Startprofils so verändert, daß die tatsächliche Lavalzahlverteilung der vorgegebenen möglichst nahe kommt im Sinne der durch das Zielfunktional (4.1) gegebenen Least–Squares–Approximation.

Die gewählten Beispiele lassen sich zunächst einteilen in Nullresiduum–Probleme und Nicht–Nullresiduum–Probleme. Im ersten Fall gibt es ein Profil, dessen zugehöriger Strömungsverlauf genau mit der Sollvorgabe übereinstimmt. Konstruiert werden solche Nullresiduum–Probleme durch Vorgabe des „gewünschten" Profils und Berechnung der Strömung mit Hilfe der Simulation. Die Bedeutung liegt vor allem in der Testmöglichkeit für einen Algorithmus.

Gibt man aber irgendeine Lavalzahlverteilung vor, so gibt es im allgemeinen kein Profil, das genau diese Verteilung erzeugt, so daß auch der Minimalwert des Zielfunktionals von null verschieden ist. Wir nennen das ein Nicht–Nullresiduum–Problem.

2.7.1 Rechnungen für Nullresiduum–Probleme

Als Startprofil wird immer das von der Firma MTU gelieferte Profil aus Abbildung 2.14a) benutzt. Das „Zielprofil" aus Abbildung 2.14b) ist durch willkürliche Veränderung dreier Designparameter daraus entstanden.

Tabelle 2.1: Nullresiduum–Problem ohne Regularisierung

k	$\|c_U\|_\infty$	f	f_r	$\Delta\zeta$	$\|\Delta q\|_2$	$\|\nabla_q L\|_2$
0/1	.225E+04	.356E-02	.356E-02	-.322E-01	.199E-02	.132E+02
10	.100E+05	.278E-02	.278E-02	-.452E-02	.247E-02	.168E+02
20	.315E+04	.106E-02	.106E-02	-.125E-02	.122E-02	.664E+01
30	.742E+03	.368E-03	.368E-03	-.531E-03	.774E-03	.934E+01
60	.997E+02	.177E-03	.177E-03	-.517E-04	.656E-03	.197E+01
100	.127E+04	.178E-03	.178E-03	-.677E-04	.235E-02	.304E+01

Tabelle 2.1 zeigt in Abhängigkeit von der Schrittzahl k den nichtlinearen Defekt $\|c_U\|_\infty$ der Stromfunktionsgleichung, das Zielfunktional f, das regularisierte[1] Zielfunktional f_r, den „mög-

[1] Zur Regularisierung siehe unten.

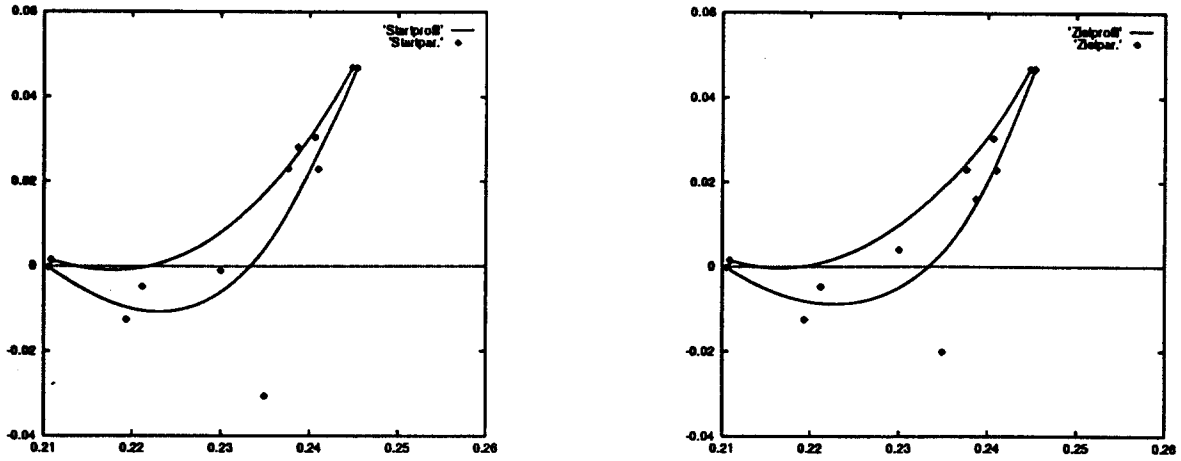


Abbildung 2.14: Schaufelprofile

a) Startprofil b) Zielprofil

lichen Abstieg" $\Delta\zeta$ der Merit–Funktion ζ, die euklidische Norm $\|\Delta q\|_2$ des Inkrements in den Designparametern und die euklidische Norm $\|\nabla_q L\|_2$ des Gradienten der Lagrange–Funktion.

Die verwendeten Größen sind im Programm wie folgt codiert:

$$\begin{aligned}
\texttt{itno} &:= k \\
\texttt{defmax(1)} &:= \|c_U\|_\infty \\
\texttt{zfs} &:= f \\
\texttt{zreg} &:= f_r = f + \frac{1}{2}\alpha_z \|q - \hat{q}\|_2^2 \\
\texttt{zetas} &:= \Delta\zeta \\
\texttt{dqn} &:= \|\Delta q\|_2 \\
\texttt{dgr2n} &:= \|\nabla_q L\|_2
\end{aligned}$$

Das verwendete Optimierungsverfahren läßt sich durch die folgenden Angaben charakterisieren:

- Asymptotisch korrekte Strategie zum Update der Hesse-Matrix
- Wahl der Startapproximation für die Hessematrix: $B = 500 \cdot I$
- Verwendung des BFGS–Updates mit Powell–Strategie
- Schrittweite für den Zwischenschritt $\texttt{omupd} = \omega_{\text{upd}} = 0.1$
- Abschneiden der Zielfunktion findet nicht statt: $\texttt{div1} = -1$, $\texttt{div2} = -1$
- Regularisierungsparameter $\texttt{az} = \alpha_z = 0.0$ (Regularisierung nicht aktiv)
- Glattheit des Lavalzahl–Limiters: C^3, also $\texttt{smooth} = 3$ mit Kurvenparameter $\texttt{ep} = 0.3$

- Trust Region Methode mit festem Radius `trmax` $= 3 \cdot 10^{-3}$
- Verwendung der Linesearch mit der Merit–Funktion von Powell, d.h. Aufruf der Routine `LSRCH2`
- Anzahl der linearen Mehrgitterschritte zur näherungsweisen Lösung der linearen Systeme (mit den Matrizen C_x unt C_x^T): `itmax` $= 1 \ldots 2$
- Beispielnummer `swpre` $= 13$

Der Auswahlparameter `swpre` codiert hierbei die Nummer des Files, das sie Soll-Lavalzahlkurve enthält.

1) Nach 100 SQP–Iterationen ist die vorgegebene Lavalzahlverteilung sehr gut approximiert, vgl. Abbildung 2.15a)[2]. Auch das Profil wird gut erreicht, siehe Abbildung 2.15b).

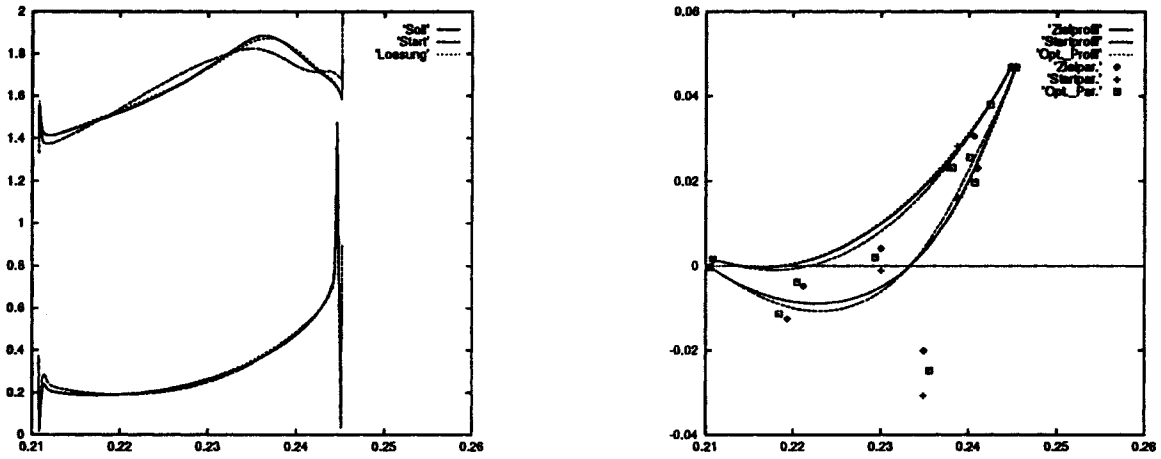


Abbildung 2.15: Nullresiduum–Problem, ohne Regularisierung

a) Lavalzahlverlauf b) Profilkurven

Es fällt auf, daß die Designparameter keineswegs mit den Designparametern des Sollprofils übereinstimmen.

Bei einer anderen Wahl der Steuergrößen des Algorithmus[3] wird das Profil nach 100 Schritten noch besser getroffen, so daß die Plots fast ununterscheidbar sind, aber die Designparameter liegen deutlich getrennt. Dies bedeutet, daß die Parameter *schwach nichteindeutig* bestimmt sind. Wir geben folgende Erklärung: Das Intervall zur Parametrisierung der Profilkurve ist in vier Teilintervalle zerlegt. Es werden 12 Basissplines zur Darstellung der Komponenten $x(t)$ und $y(t)$ verwendet, so daß je 3 Basissplines N_{j+1}, N_{j+2}, N_{j+3}, $j \in \{0, 3, 6, 9\}$ denselben Träger (immer zwei Teilintervalle) haben. Sie sind zwar linear unabhängig, können aber offenbar im Sinne einer Approximation gegen eine andere Konfiguration ausgetauscht werden, ohne den Kurvenverlauf deutlich zu verändern. Auch die 6 „Nachbarsplines", deren Träger jeweils auf einem Teilintervall mit den Trägern von N_{j+1}, N_{j+2}, N_{j+3} übereinstimmen, können dazu beitragen. In

[2]In dieser und in den weiteren Lavalzahlplots ist zur besseren Übersichtlichkeit zur Lavalzahl der Saugseite künstlich 1 hinzuaddiert.

[3]Insbesondere bei Verwendung einer verfeinerten Trust–Region–Strategie, die zur Bestimmung des Trust–Region–Radius Krümmungsinformation des Lösungsweges in den Designparametern ausnutzt

diesem Sinne sind die Designkoeffizienten *nicht eindeutig identifizierbar*, was auch einen Einfluß auf die Effizienz des Optimierungsverfahrens hat.

2) Das vorliegende Strömungsmodell führt am Profilanfang und besonders am Profilende auf stark variierende Lavalzahlen, die nicht die tatsächlich beobachtete Strömung wiedergeben. Von diesen willkürlichen „Ausschlägen" sollten die Designparameter aber nicht abhängig gemacht werden. Deshalb wurde im Programm die Möglichkeit vorgesehen, die ersten und letzten Profilpunkte aus dem Zielfunktional herauszunehmen (Summationsgewicht 0.0).

Wenn man bei dem beschriebenen Nullresiduum–Problem jeweils 10 Gitterpunkte am Leading Edge und am Trailing Edge „abschneidet"(`div1` = 10, `div2` = 10), dann werden im Optimierungslauf Profile mit Selbstdurchdringung in der Nähe des Trailing Edge erzeugt, und das Verfahren bricht ab wegen unphysikalischer Werte (negative Dichtewerte).

3) Eine angemessene Behandlung bestünde darin, zusätzliche Ungleichungsnebenbedingungen (z.B. für eine Mindestprofildicke) aufzustellen. Dies könnte in einer weiteren Phase des

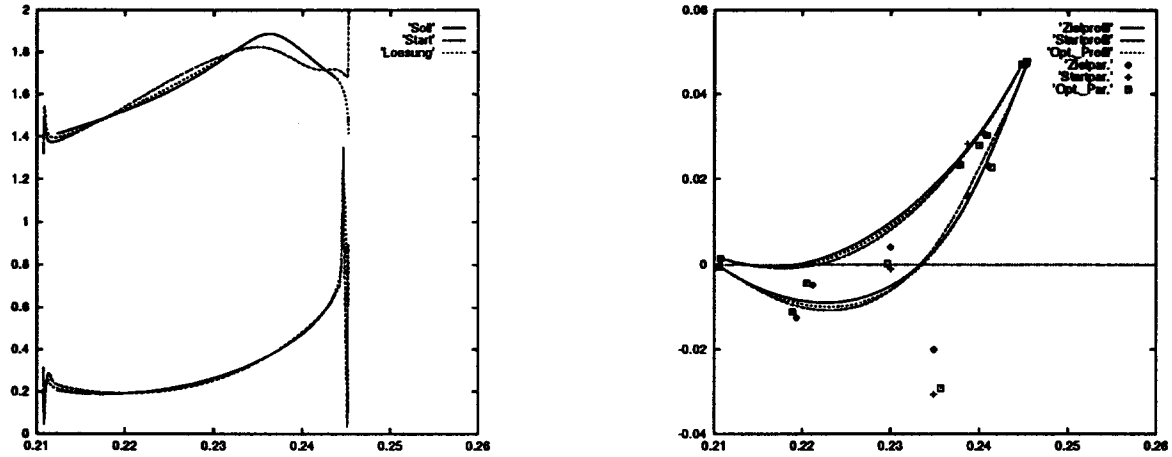


Abbildung 2.16: Nullresiduum–Problem mit Regularisierung

a) Lavalzahlverlauf b) Profilkurven

Projekts geschehen. Wir haben im vorliegenden Programm einen *Regularisierungsparameter* α_z eingeführt. Die regularisierte Zielfunktion

$$f_r(u,\rho,q) = f(u,\rho) + \frac{1}{2}\alpha_z \|q - \hat{q}\|_2^2 \tag{7.1}$$

nimmt zu, wenn der aktuelle Parametervektor q von einem vorgegebenen Vektor $\hat{q}$ abweicht. Wir haben für $\hat{q}$ immer die Startdaten eingesetzt. Für $\alpha_z = 50.0$ erhält man den in Tabelle 2.2 dargestellten Konvergenzverlauf. Dabei reichen 30 SQP–Schritte aus, um das Profil nach Abbildung 2.16b) zu erreichen. Nach 100 Schritten ergibt sich praktisch dieselbe Kurve. Die Größenordnung der charakteristischen Werte, insbesondere die Normen des partiell reduzierten Gradienten und des Gradienten der reduzierten Lagrangefunktion fallen höher aus als erwartet. Die dargestellten Tabellen haben auch den Zweck, die erreichten Daten als Erfahrungswerte für spätere Berechnungen (Anwendungen) zur Orientierung festzuhalten.

Tabelle 2.2: Nullresiduum–Problem mit Regularisierung

k	$\|c_U\|_\infty$	f	f_r	$\Delta\zeta$	$\|\Delta q\|_2$	$\|\nabla_q L\|_2$
0/1	.225E+04	.640E-03	.640E-03	-.873E-03	.174E-02	.864E+00
10	.548E+04	.115E-03	.211E-03	-.156E-03	.279E-03	.444E+00
20	.306E+04	.667E-04	.156E-03	-.164E-05	.133E-03	.858E-01
30	.309E+03	.527E-04	.153E-03	-.436E-05	.467E-03	.405E-01
60	.154E+04	.708E-04	.161E-03	-.112E-04	.132E-02	.632E-01
100	.147E+03	.572E-04	.153E-03	-.112E-06	.122E-03	.128E-01

2.7.2 Rechnungen für Nicht–Nullresiduum–Probleme

In diesem Abschnitt werden die Rechenergebnisse für vier verschiedene Nicht–Nullresiduum–Probleme dargestellt. Es wurden jeweils 100 SQP–Iterationen durchgeführt und dann nachgesehen, ob sich das Ergebnis im wesentlichen auch schon mit weniger (30) Schritten erreichen läßt. In drei von vier Fällen verhält es sich so, nur in einem Fall werden tatsächlich 100 Schritte benötigt.

Gegenüber den in Abschnitt 2.7.1 für das Nullresiduum–Problem festgehaltenen Steuergrößen und Varianten des Optimierungsalgorithmus wurden hier in allen Beispielen folgende Veränderungen vorgenommen:

a. Es wurde immer die Regularisierungsstrategie verwendet, mit Parameter $\alpha_z = 10.0$

b. Die Hesse-Matrix wurde als Diagonalmatrix gestartet mit einer Diagonale, die in einem Optimierungslauf nach mehreren Updates errechnet worden ist. (Insbesondere wird dabei das unterschiedliche Gewicht der verschiedenenen Designparameter schon berücksichtigt.)

c. Schließlich wurde nach je 5 SQP–Schritten ein nichtlinearer Mehrgitterzyklus aufgerufen, der zu einer Defektreduktion und zu einer deutlichen Verbesserung des gesamten Optimierungsverlaufs geführt hat.

1) Das erste Beispiel ist konstruiert: Es wurde eine Konvexkombination aus dem Lavalzahlverlauf der gegebenen MTU–Profilkurve und einem linear ansteigenden Lavalzahlverlauf (in derselben Größenordnung) als Sollverlauf gebildet. Dies führt im Vergleich zu den weiteren Beispielen mit „realistischen“ Vorgaben zu größeren Profiländerungen, die zeigen, daß der Optimierungsalgorithmus auch größere Abweichungen vom optimalen Verlauf verarbeiten kann. Tabelle 2.3 zeigt die charakteristischen Daten des Optimierungslaufes. Der Lavalzahlverlauf ist nach 30 Schritten gut approximiert, und die Profilkurve ist gefunden. Parameter in **SETPAR**: `swpre = 4, tau = 0.6`. Die Ergebnisse sind in Abbildung 2.17a)b) dargestellt.

2) Bei der zweiten Lavalzahl–Sollvorgabe wurde — ausgehend vom Simulationsergebnis des MTU–Startprofils — das Lavalzahl–Maximum der Saugseite Richtung Leading Edge verschoben. Die Geschwindigkeitsverteilung an der Druckseite wurde nicht verändert. Dieses Beispiel wurde erzeugt mit dem Parameter `swpre` = 16. Die Ergebnisse in Tabelle 2.4 zeigen, daß man in diesem Fall mit 30 Iterationen das Ziel noch nicht erreicht (Kriterium: Weitere Abnahme der Zielfunktion f), sondern 100 Schritte braucht. Es wurde noch bis zur Schrittzahl 200 weitergerechnet,

Tabelle 2.3: Lavalzahl–Verlauf mit linearem Anstieg konvex kombiniert

k	$\|c_U\|_\infty$	f	f_r	$\Delta\zeta$	$\|\Delta q\|_2$	$\|\nabla_q L\|_2$
0/1	.225E+04	.195E-02	.195E-02	-.467E-02	.241E-02	.209E+01
10	.648E+04	.229E-03	.274E-03	-.266E-03	.890E-03	.193E+00
20	.998E+03	.118E-03	.186E-03	-.117E-03	.158E-02	.262E+00
30	.240E+04	.775E-04	.160E-03	-.872E-05	.335E-03	.160E+00
60	.210E+03	.775E-04	.158E-03	-.120E-06	.291E-03	.894E-01
100	.274E+02	.775E-04	.158E-03	-.160E-06	.298E-03	.929E-01

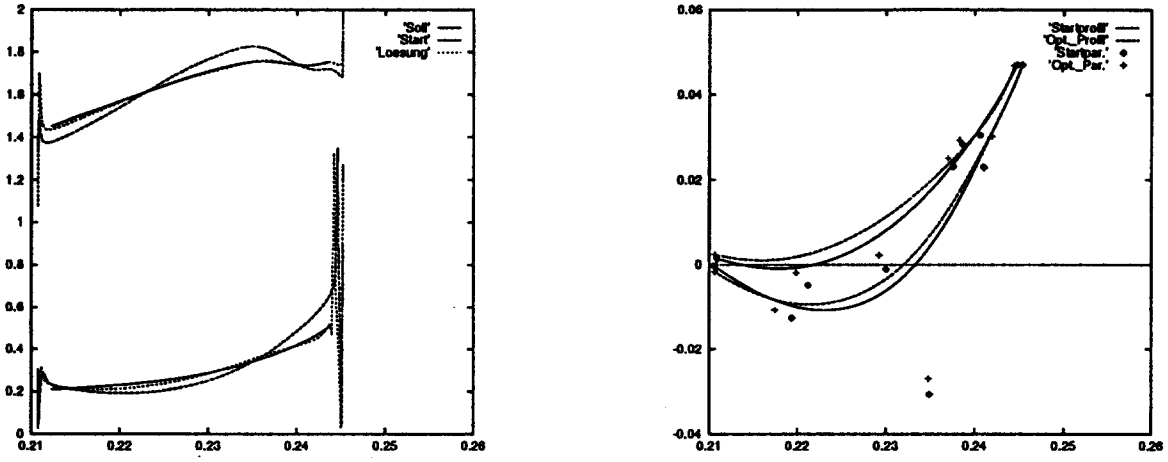


Abbildung 2.17: Nicht–Nullresiduum–Problem, Beispiel 1

a) Lavalzahlverlauf b) Profilkurven

ohne weitere Verbesserung. Die Plots in Abbildung 2.18a)b) zeigen wieder Lavalzahlverlauf und Schaufelprofil, dieses Mal nach 100 SQP–Iterationen.

3) Im dritten Beispiel wurde das Lavalzahl–Maximum an der Saugseite nicht verschoben, aber der Verlauf von dort bis in die Nähe des Trailing Edge begradigt. Technisch bedeutet dies, daß eine Wiederbeschleunigungsphase vermieden werden sollte. Die Optimierungsdaten sind in Tabelle 2.5 zusammengefaßt, die Plots sind in Abbildung 2.19 dargestellt. Dieses Beispiel wurde mit **swpre** = 6 errechnet.

Das Zielfunktional wurde noch einmal deutlich reduziert, obwohl es auch am Start schon recht klein ist. Dazu reichen auch 30 Schritte wieder aus. Im optimierten Lavalzahlverlauf tritt dann die Wiederbeschleunigung doch auf: Es ist eine charakteristische Eigenschaft der Lösung eines Optimierungsproblems mit Least–Squares–Zielfunktional, daß sie um die Zieldaten oszilliert. Ganz anschaulich deshalb, weil mehrere kleine Abweichungen ein kleineres Gewicht haben als wenige große.

4) Im vierten und letzten Beispiel schließlich wurde das Lavalzahl–Maximum an der Saugseite Richtung Trailing Edge verschoben. Tabelle 2.6 zeigt das Konvergenzverhalten des SQP–Algorithmus. Nach 30 Iterationen ist die Lösung recht gut erreicht. Abbildung 2.20 zeigt die

Tabelle 2.4: Lavalzahl–Maximum Richtung LE verschoben

k	$\|c_U\|_\infty$	f	f_r	$\Delta\zeta$	$\|\Delta q\|_2$	$\|\nabla_q L\|_2$
0/1	.225E+04	.225E-03	.225E-03	-.189E-03	.127E-02	.325E+00
10	.105E+03	.196E-03	.196E-03	-.306E-04	.248E-02	.312E+00
20	.607E+02	.193E-03	.196E-03	-.693E-05	.225E-02	.214E+00
30	.409E+02	.189E-03	.196E-03	-.159E-05	.203E-02	.163E+00
60	.133E+02	.160E-03	.193E-03	-.684E-05	.863E-03	.513E-01
100	.250E+03	.998E-04	.164E-03	-.285E-05	.201E-02	.179E+00
150	.130E+02	.972E-04	.162E-03	-.217E-08	.745E-03	.135E+00
200	.244E+02	.921E-04	.162E-03	+.175E-05	.817E-03	.151E+00

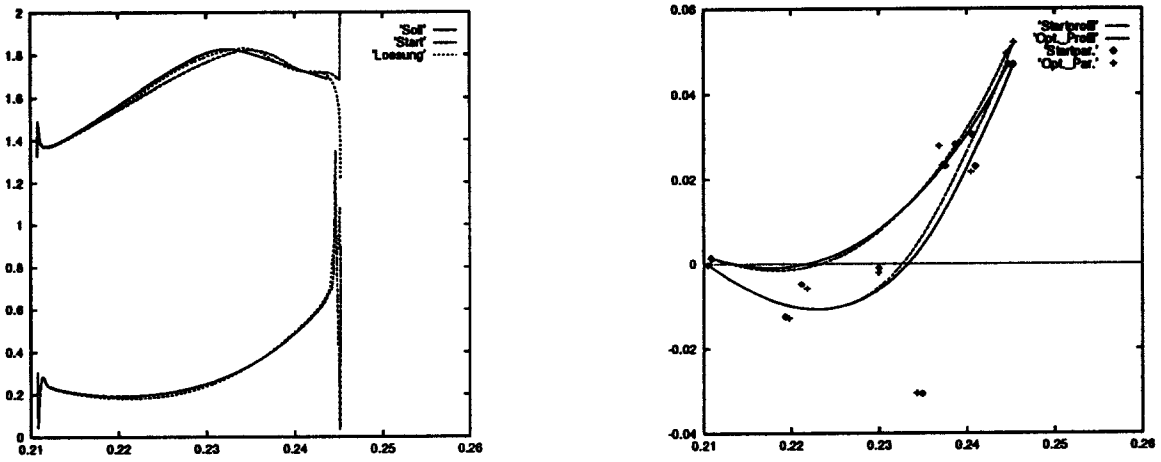


Abbildung 2.18: Nicht–Nullresiduum–Problem, Beispiel 2

a) Lavalzahlverlauf b) Profilkurven

Lavalzahlverteilung und die Veränderung des Schaufelprofils. In diesem Beispiel ist `supre` = 7.

2.7.3 Erforderliche Genauigkeit bei der Lösung linearer Teilprobleme

Die Anzahl der Nebenbedingungen bei der Schaufeloptimierung ist gegeben durch die Diskretisierung der die Strömung modellierenden partiellen Differentialgleichung. Sie liegt etwa bei 20 000, so daß wir iterative Löser für die linearen Systeme (mit den Matrizen C_x und C_x^T) benutzen. Bei vielen anderen Optimierungsproblemen kann dagegen mit direkten Lösern gearbeitet werden, so daß die Frage, *wie genau* die linearen Systeme eigentlich gelöst werden müssen, gar nicht untersucht werden muß. Das ist hier grundsätzlich anders. Für das Nullresiduum–Problem, das auch schon im Abschnitt 2.7.1 behandelt wurde, haben wir dazu vier Läufe mit jeweils 30 SQP–Schritten verglichen, in denen die Anzahl der Mehrgitterzyklen für jedes lineare System auf 1, 2, 3 oder 5 festgelegt wurde. Die Ergebnisse, die nach 10, nach 20 und nach 30 SQP-Iterationen erzielt wurden, sind in Tabelle 2.7 zusammengestellt. Die Variante, bei der der niedrigste Zielfunktionswert f auftritt, ist jeweils mit einem Sternchen * versehen. Es zeigt sich,

Tabelle 2.5: Lavalzahl–Verlauf ohne Wiederbeschleunigung

k	$\|c_U\|_\infty$	f	f_r	$\Delta\zeta$	$\|\Delta q\|_2$	$\|\nabla_q L\|_2$
0/1	.225E+04	.245E-04	.245E-04	-.596E-04	.480E-03	.181E+00
10	.134E+03	.149E-04	.161E-04	-.844E-05	.537E-03	.522E-01
20	.467E+02	.138E-04	.158E-04	-.101E-05	.455E-03	.323E-01
30	.533E+02	.135E-04	.157E-04	-.765E-06	.435E-03	.290E-01
60	.540E+02	.130E-04	.155E-04	-.294E-07	.391E-03	.236E-01
100	.510E+02	.130E-04	.153E-04	-.468E-05	.454E-03	.269E-01

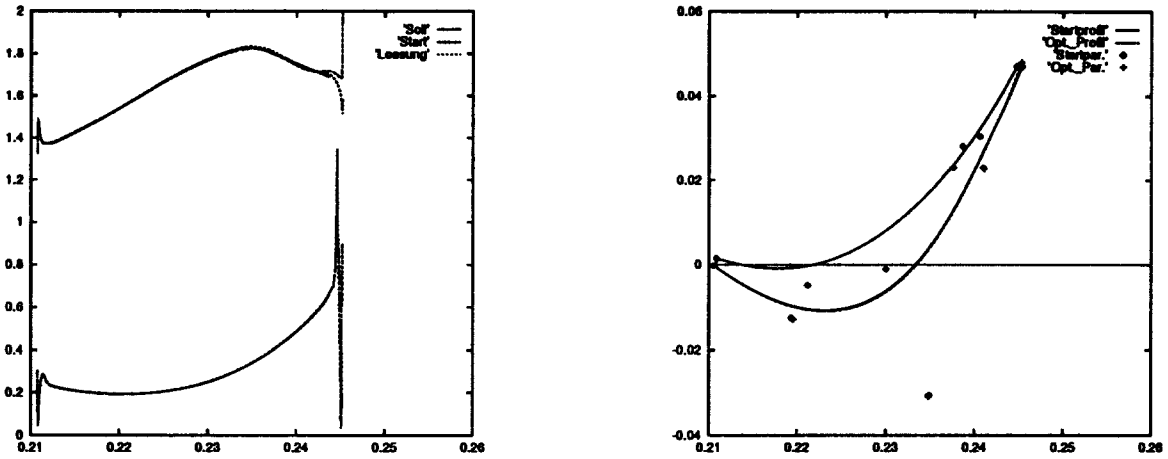


Abbildung 2.19: Nicht–Nullresiduum–Problem, Beispiel 3

a) Lavalzahlverlauf b) Profilkurven

daß weniger genaue Rechnungen hier zu Beginn zu einer größeren Reduktion des Zielfunktionals führen, daß aber im weiteren Verlauf eine höhere Rechengenauigkeit bessere Ergebnisse bringt. — In anderen Beispielen wurde auch sehr deutlich eine Asymptotik sichtbar, d.h. es war z.B. egal, ob man mit 3, 4 oder 5 Mehrgitteraufrufen arbeitet.

2.7.4 Formulierung eines Abbruchkriteriums

Aus dem in den vorigen Abschnitten gezeigten Tabellen mit charakteristischen Größen des Optimierungsverfahrens geht auch hervor, daß es keine festen Schranken gibt, die für ein Abbruchkriterium herangezogen werden könnten. Die Daten sind sehr vom Problem abhängig. Auch die relativen Änderungen der verschiedenen Größen sind vom Problem und obendrein noch von den Startwerten abhängig. Die Größe $\Delta\zeta$ (im Programm `zetas`) wird oft (im Vergleich zu den anderen Größen) recht deutlich reduziert, so daß wir folgendes „relative Abbruchkriterium" formuliert haben: Stop des Algorithmus nach Iteration k, falls

$$\Delta\zeta^{(k)} \leq \theta\ \Delta\zeta^{(1)} \tag{7.2}$$

Tabelle 2.6: Lavalzahl–Maximum Richtung TE verschoben

k	$\|c_U\|_\infty$	f	f_r	$\Delta\zeta$	$\|\Delta q\|_2$	$\|\nabla_q L\|_2$
0/1	.225E+04	.185E-03	.185E-03	-.743E-04	.107E-02	.321E+00
10	.631E+03	.160E-03	.162E-03	-.176E-04	.247E-02	.230E+00
20	.122E+03	.139E-03	.143E-03	-.320E-06	.803E-03	.187E+00
30	.907E+03	.124E-03	.127E-03	-.640E-04	.299E-02	.207E+00
60	.525E+03	.101E-03	.107E-03	-.122E-04	.233E-02	.217E+00
100	.446E+03	.948E-04	.104E-03	-.171E-04	.230E-02	.190E+00

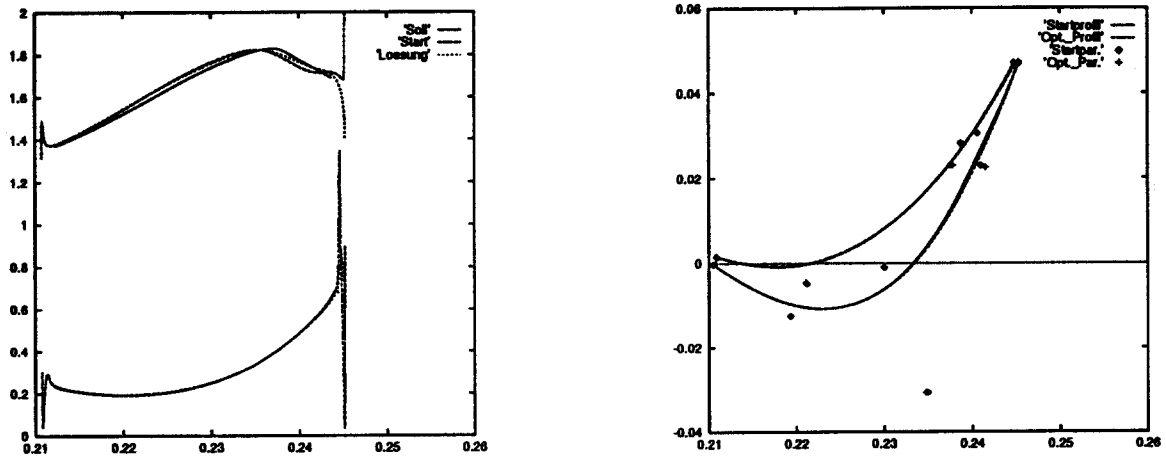


Abbildung 2.20: Nicht–Nullresiduum–Problem, Beispiel 4

a) Lavalzahlverlauf b) Profilkurven

Tabelle 2.7: Profil 002, Null-Residuum

k	itmax	$\|c_U\|_\infty$	f	f_r	$\Delta\zeta$	$\|\Delta q\|_2$	$\|\nabla_q L\|_2$
10	1	.964E+2	.672E-3	.672E-3	-.317E-3	.138E-3	2.31
	*2	.393E+2	.557E-3	.557E-3	-.506E-4	.109E-3	4.08
	3	.177E+2	.617E-3	.617E-3	-.699E-4	.128E-3	1.68
	5	.329E+1	.643E-3	.643E-3	-.329E-4	.722E-4	1.71
20	1	.200E+3	.278E-3	.278E-3	-.326E-4	.170E-3	0.630
	2	.191E+3	.164E-3	.164E-3	-.105E-3	.350E-3	2.44
	*3	.603E+2	.147E-3	.147E-3	-.190E-3	.797E-3	2.79
	5	.111E+3	.167E-3	.167E-3	-.213E-3	.132E-2	4.32
30	1	.569E+2	.707E-4	.707E-4	-.421E-4	.679E-3	1.19
	2	.402E+2	.555E-4	.555E-4	-.436E-4	.411E-3	2.09
	3	.198E+2	.270E-4	.270E-4	-.270E-4	.677E-3	0.634
	*5	.417E+2	.160E-4	.160E-4	-.108E-4	.460E-3	0.226

erfüllt ist. Da kleine Werte von $\Delta\zeta$ auch zwischendurch, vor Erreichen eines Minimums auftreten können, haben wir zusätzlich gefordert, daß das Kriterium in mehreren SQP–Schritten hintereinander erfüllt sein soll. — Dies ist natürlich sehr heuristisch, und es müssen auch erst wieder Erfahrungswerte gewonnen werden. So kann man z.B. bei den Nicht–Nullrediduum–Problemen mit $\theta = 0.02$ gut arbeiten.

Allgemein läßt sich sagen, daß in dem verwendeten Modell sehr verschiedene Größenordnungen auftreten. Die Ableitung der Stromfunktionsgleichung nach den Designparametern erreichen Werte um 10^9, so daß winzige Änderungen der Designparameter große Änderungen des nichtlinearen Defekts hervorrufen. Deswegen sollte bei einer Fortsetzung des Projekts auch über eine verbesserte Modellierung des Strömungsproblems nachgedacht werden, bzw. mindestens über eine geeignete Skalierung.

2.7.5 Rechenzeitmessungen

Die IBM RS6000 hat zwischen 39 und 50 Sekunden für einen SQP–Schritt gebraucht, wenn die asymptotisch korrekte Update–Technik verwendet wurde. Pro SQP–Schritt sind hier vier große lineare Systeme approximativ zu lösen.

Zwei Zeitvergleiche sind von besonderem Interesse:
1) Der Aufwand der SQP–Iteration *im Verhältnis zur Startphase*, bis der erste nichtlineare Mehrgitterschritt auf dem feinsten Gitter gerechnet ist. (Oder auch allgemeiner im Verhältnis zum Aufwand der Simulation, bis ein vergleichbarer nichtlinearer Defekt erzielt ist.) Ein Überschlag ergibt sich aus einer Messung, nach der der Startprozeß etwa 16s dauert. Bei 48s pro SQP–Schritt und 30 Schritten insgesamt erhält man den $3 \cdot 30 = 90$–fachen Aufwand für eine Profiloptimierung.

Bei einem Rechenzeitvergleich in verschiedenen Beispielen bewegte sich der Aufwand für die Optimierung zwischen dem 50– und 200–fachen des Simulationsaufwandes. Die Zahlen kann man aber nur als grobe Anhaltspunkte verstehen, weil ein scharfes Abbruchkriterium fehlt. Oft kann in der Linesearch nur ein kleiner Schrittlängenparameter α gefunden werden, so daß eine ganze Reihe von SQP–Schritten notwendig sind. Ein kleines α bedeutet anschaulich, daß man nicht weit in Abstiegsrichtung gehen kann, bevor die Zielfunktion (genauer die Merit–Funktion) wieder ansteigt. Die Täler des Zielfunktionsgebirges sind also in Suchrichtung des Algorithmus oft sehr schmal. Wir vermuten auch, daß die berechnete Hesse-Matrix nicht so gut bestimmt ist. Es werden — in der Zeilennummernorm $\|.\|_\infty$ gemessen — oft sehr große Änderungen durch das BFGS–Update vorgenommen. Dies weist darauf hin, daß auch die Hesse–Matrix sehr empfindlich (d.h. mit großen Änderungen) auf Designparameter– und/oder Variablenänderungen reagiert.

2) Zum zweiten ist der relative Aufwand der verschiedenen Bestandteile einer SQP–Iteration, der in Tabelle 2.8 für die einzelnen Beispiele aufgelistet wurde, von Interesse.
Die Beispiele sind durch den Auswahlparameter `supre` und ggf. durch einen weiteren Parameter gekennzeichnet. Die Reihenfolge entspricht genau der Reihenfolge, in der wir die Rechnungen vorgestellt haben, d.h. die ersten beiden Zeilen beziehen sich auf das Nullresiduum–Problem, die Zeilen 3 – 6 auf die vier Nicht–Nullresiduum–Probleme und die Zeilen 7 – 10 auf das Nullresiduum–Problem mit mehreren Mehrgitterschritten. Die ersten sechs Beispiele wurden auf einer IBM RS6000 gerechnet, die anderen auf einer Silicon Graphics INDY mit einem MIPS R4600–Prozessor. Dabei wurden die Rechenzeiten zur Aufstellung der Ableitungen von Stromfunktions– und Dichtegleichung nach den Designparametern, zur approximativen Mehr-

Tabelle 2.8: Rechenzeit pro SQP–Iteration [s]

swpre	Parameter	$\sum$ Rechenzeit [s]	pro SQP-It.	Berechn. Abl. C_q	System C_x	System C_x^T	Linesearch
13		4472.99	45	36.7%	5.8%	22.8%	18.8%
13	$\alpha_x = 50$	1170.05	39	41.9%	4.3%	16.7%	17.9%
4	$\tau = 0.6$	1283.85	43	38.1%	4.5%	17.5%	20.7%
16		5047.59	50	32.5%	6.2%	24.2%	20.9%
6		1494.66	50	32.8%	6.1%	23.5%	20.7%
7		1423.53	47	34.6%	5.7%	22.2%	19.8%
13	itmax = 1	1911.56	64	42.3%	10.3%	14.8%	10.5%
13	itmax = 2	2358.07	79	34.3%	16.0%	23.9%	7.9%
13	itmax = 3	2807.05	94	28.8%	20.0%	30.0%	6.2%
13	itmax = 5	3719.03	124	21.7%	24.8%	37.6%	4.5%

gitterlösung des linearisierten und transponierten Systems sowie für die Linesearch gemessen und durch die Gesamtrechenzeit dividiert. Es fällt besonders der sehr hohe Anteil der Ableitungsberechnung auf: Bei den auf der IBM durchgeführten Rechnungen lag er zwischen 32.5 und 41.9%. Pro Gleichung sind 24 Gittervektoren aus partiellen Ableitungen zu bilden. Im Vorlauf und Nachlauf verschwinden die Ableitungen systematisch, so daß vom Umfang her 12 vollbesetzte Gittervektoren berechnet werden müssen. Das ist das $\frac{4}{3}$–fache des 9–Punkt–Sterns, der in der Diskretisierung der Stromfunktionsgleichung auftritt. In vielen Fällen, in denen SQP–basierte Optimierungsalgorithmen zur Anwendung kommen, ist die Rechenzeit zur Ableitungsberechnung vernachlässigbar, so daß wir mit diesem Aufwand nicht gerechnet hatten.

Der Aufwand zur Lösung der linearen Systeme wiederum ist unterschiedlich für „Vorwärtssysteme“ und transponierte Systeme. Das liegt an der Darstellung der Matrix im Programm: Während die nichttrivialen Diagonalen von C_x komplett abgespeichert sind, wird C_x^T aus C_x immer zeilenweise erzeugt, was die Adressierung der Elemente teurer macht. Offenbar reagiert die IBM RS6000 erheblich empfindlicher als die Silicon Graphics INDY. Mit so einem Effekt hatten wir nicht gerechnet. An dieser Stelle läßt sich ein Teil der Rechenzeit sparen durch komplette Abspeicherung von C_x^T.

Der Aufwand für die Linesearch ist nicht vernachlässigbar, weil zur wiederholten Auswertung der Merit–Funktion Operationen auf dem gesamten Gitter (und nicht nur in Profilnähe) vorgenommen werden müssen. Powells Merit–Funktion verwendet nämlich den nichtlinearen Defekt. Die Anzahl der Auswertungen ist nicht vorhersagbar, aber in der Implementierung durch die Konstruktion nach oben beschränkt.

Kapitel 3

Zusammenfassung und Ausblick

Im Projekt 1.110 der Turbotech-Interimsphase wurde ein neues Verfahren zur Optimierung von Turbinenschaufeln entwickelt, das im vorliegenden Schlußbericht eingehend dargestellt und analysiert wird. Dieses Optimierungsverfahren erweist sich als deutlich effizienter als vergleichbare Verfahren zur Lösung derselben Problemstellung (z.B. genetische Algorithmen). Der Hauptgrund hierfür liegt in der simultanen Behandlung des Optimierungsproblems: erst in der Lösung des Optmierungsproblems wird auch Zulässigkeit erreicht.

In diesem Projekt wurde ein Prototyp eines praktisch einsetzbaren Optimierungscodes zur Berechnung optimaler Turbinen- und Verdichterschaufelprofile entwickelt. Die Performance des Codes wird in Abschnitt 2.7 eingehend analysiert.

Vom wissenschaftlichen Standpunkt betrachtet stellt das entwickelte Verfahren einen Durchbruch innerhalb der Mathematischen Optimierung dar, da es sich hierbei um das erste nichtlineare Optimierungsverfahren überhaupt handelt, das die auftretenden unsymmetrischen linearen Teilsystem mit Hilfe von Mehrgitterverfahren iterativ löst. Es wurden grundlegende Resultate für adjungierte Mehrgitterverfahren erzielt, für die überraschenderweise noch keine theoretischen Untersuchungen vorlagen. Darüberhinaus erwiesen sich die parallel zu diesem Projekt in [17] entwickelten partiell reduzierten SQP Methoden als essentiell für die Behandlung der auftretenden geometrischen Nebenbedingungen.

Folgende Veröffentlichungen sind im Rahmen dieses Projektes enstanden:

- V. H. Schulz, Th. Dreyer, Th. Speer and H. G. Bock: *Optimum shape design of turbine blades.* In: Proceedings of the Symposium on Operations Research, University of Passau, Sept. 13-15, 1995. (erscheint)

- V. H. Schulz: *Numerical Optimization of the Cross-Sectional Shape of Turbine Blades.* In: Proceedings of ICIAM 95, Special Issue of Zeitschrift für Angewandte Mathematik und Mechanik, Akademie-Verlag, Berlin, 1995. (eingereicht)

Die in diesem Projekt geleistete Arbeit versetzt nun die daran Beteiligten in die Lage, weiterführende Detailfragen zu formulieren, die ein weiteres Effizienzpotential über den erreichten Stand hinaus erkennen lassen:

- Eine Änderung des Diskretisierungsgitter scheint ratsam. Es könnte etwa ausgetauscht werden durch ein O- oder ein C-Netz.

- Ein kompletter Trust-Region Ansatz könnte eine noch stabilere Alternative zu der im vorliegenden Projekt verfolgten Line-Search sein.
- Operationen mit der transponierten Systemmatrix sind in der vorliegenden Code-Version noch nicht in optimaler Weise implementiert. Hierbei besteht Verbesserungspotential.
- Der Aufwand zur Berechnung der Ableitung der Systemgleichungen nach den Splineparametern läßt sich möglicherweise reduzieren durch Einführen weiterer Zwischenvariablen längs der Profilkontur.

Es besteht die berechtigte Hoffnung, daß in einem Anschlußprojekt im Rahmen von Turbotech II diese Punkte aufgegriffen und in die Bearbeitung des Projektes nutzbringend einbezogen werden können.

Literaturverzeichnis

[1] H.G. Bock. Randwertproblemmethoden zur Parameteridentifizierung in Systemen nichtlinearer Differentialgleichungen. newblock *Bonner Mathematische Schriften 183*, Bonn, 1987.

[2] H.G. Bock, K.J. Plitt: A multiple shooting algorithm for the direct solution of constrained optimal control problems, Proceedings 9th IFAC World Congress Automatic Control, Pergamon Press, 1985.

[3] H.G. Bock, J.P. Schlöder und V.H. Schulz. Numerik großer Differentiell-Algebraischer Gleichungen – Simulation und Optimierung. In H. Schuler (Hrsg.), *Prozeßsimulation*, S. 35–80. VCH Verlagsgesellschaft mbH, Weinheim, 1994.

[4] C. de Boor: A Practical Guide to Splines. Springer, Berlin, New York, 1978.

[5] C. A. J. Fletcher: Computational Techniques for Fluid Dynamics I, II. Springer Berlin, New York, 1988.

[6] G. Fritsch: Berechnung der zweidimensionalen Umfangslösung entlang einer beliebigen „Rotationsstromfläche“ erster Art mit Hilfe eines Stromfunktionsverfahrens in konservativer Formulierung. Diplomarbeit am Institut für Thermische Strömungsmaschinen und Maschinenlaboratorium, Universität Stuttgart, 1987.

[7] Ph. E. Gill, W. Murray, M. H. Wright: Practical Optimization. Academic Press, London, New York, 1981.

[8] R. Gregg, K. Misegades: Transonic Wing Optimization Using Evolution Theory. AIAA paper 87-0520; 1987.

[9] W. Hackbusch: Multi-grid gethods and applications. Springer, Berlin, New York, 1985.

[10] W. Hackbusch, U. Trottenberg (Hrsg.): Multigrid methods. Lecture Notes in Mathematics No. 960, Springer, Berlin, New York, 1982.

[11] J. Nocedal, M. L. Overton: Projected hessian updating algorithms for nonlinearly constrained optimization. SIAM J. Num. Anal. 22 (1985) 821-850.

[12] C. E. Orozco, O. N. Ghattas: Massively parallel aerodynamic shape optimization. Computing Systems in Engineering Vol. 3, S. 311-320, 1992.

[13] B. Pfeil: Aerodynamische Optimierung von Turbinenprofilen mit evolutionären Algorithmen. Diplomarbeit am Institut für Luftfahrtantriebe, Universität Stuttgart, 1994.

[14] I. Rechenberg: Evolutionsstrategie - Optimierung technischer Systeme nach Prinzipien der Biologischen Evolution. Friedrich Frommann Verlag, Stuttgart, 1973.

[15] E. Schmidt, K. Stephan, F. Mayinger: Technische Thermodynamik. Grundlagen und Anwendungen. Band 1, Einstoffsysteme. Springer, Berlin, 1975.

[16] K. Schittkowski: NLPQL: A FORTRAN subroutine solving constrained nonlinear programming problems. Annals of Operations Research 5 (1985/86), S. 485-500, 1986.

[17] V. Schulz: Reduced SQP methods for large scale optimal control problems in DAE with application to path planning problems for satellite mounted robots. Dissertation, Universität Heidelberg, 1995.

[18] R. Schwarz: Einsatz numerischer Optimierungsverfahren bei der aerothermodynamischen Auslegung von Radialverdichterstufen. Dissertation TU München, 1992.

[19] Th. Speer: Arbeitsunterlagen und Fortran-Programm zur 2D-Strömungsberechnung um Turbinenschaufeln. Firma MTU, München, 1994.

[20] Karl Stephan, Franz Mayinger: Thermodynamik: Grundlagen und technische Anwendungen. Springer, Berlin, New York, 1986. (Dieses Buch ist eine neuere Auflage von [15].)

[21] J. Stoer: Principles of sequential quadratic programming methods for solving nonlinear programs. Computational Mathematical Programming (Schittkowski, ed.), Springer, Berlin, Heidelberg, New York, Tokyo, 1985.

[22] S. S. Tong, B. A. Gregory: Turbine preliminary design using artificial intelligence and numerical optimization techniques. J. of Turbomachinery, Jan. 1992, Vol. 114, S. 1-7.

[23] G. N. Vanderplaats: Lecture Notes on Computer Aided Optimization in Engineering Design, Union College Continuing Education course, July 27 -31, 1987.

[24] C.H. Wu: A general theory of three-dimensional flow in subsonic and supersonic turbomachines of axial-, radial- and mixed-flow type. NACA TN 2604, 1952.

AG-Turbo Interimsphase, Projekt 1.110:

Ein mathematisches Verfahren zur schnellen Formoptimierung von Turbinenschaufeln

Erfolgskontrollbericht

Thomas Dreyer, Volker Schulz, Hans Georg Bock

Dezember 1995

Interdisziplinäres Zentrum für Wissenschaftliches Rechnen
UNIVERSITÄT HEIDELBERG
Im Neuenheimer Feld 368
69120 Heidelberg

---

1 Erfolg des Vorhabens

Im Rahmen dieses Projektes wurde ein neues Verfahren zur Optimierung von Turbinenschaufeln entwickelt. Dieses Optimierungsverfahren erweist sich als deutlich effizienter als vergleichbare Verfahren zur Lösung derselben Problemstellung (z.B. genetische Algorithmen). Der Hauptgrund hierfür liegt in der simultanen Behandlung des Optimierungsproblems: erst in der Lösung des Optmierungsproblems wird auch Zulässigkeit erreicht.

In diesem Projekt wurde ein Prototyp eines praktisch einsetzbaren Optimierungscodes zur Berechnung optimaler Turbinen- und Verdichterschaufelprofile entwickelt. Die Performance des Codes wird in Abschnitt 2.7 des Schlußberichts eingehend analysiert.

Vom wissenschaftlichen Standpunkt betrachtet stellt das entwickelte Verfahren einen Durchbruch innerhalb der Mathematischen Optimierung dar, da es sich hierbei um das erste nichtlineare Optimierungsverfahren überhaupt handelt, das die auftretenden unsymmetrischen linearen Teilsystem mit Hilfe von Mehrgitterverfahren iterativ löst. Es wurden grundlegende Resultate für adjungierte Mehrgitterverfahren erzielt, für die noch keine theoretischen Untersuchungen vorlagen. Darüberhinaus erwiesen sich die parallel zu diesem Projekt in der Dissertation von V. Schulz (vgl. Quelle [17] im Schlußbericht) entwickelten partiell reduzierten SQP-Methoden als essentiell für die Behandlung der auftretenden geometrischen Nebenbedingungen.

2 Einhaltung des Finanzierungs- und Zeitplans

Die Projektdauer wurde während der Projektlaufzeit kostenneutral um zwei Monate verlängert (siehe dazu Abschnitt 1.2 des Schlußberichts). Der resultierende Zeitplan und der Finanzierungsplan wurden eingehalten.

3 Verwertung der Ergebnisse

Eine Lizenzvergabe für das entwickelte Programmpaket ist nicht vorgesehen.

4 Schutzrechte

Während der Projektlaufzeit waren weder eigene noch fremde Schutzrechte berührt, noch wurden neue Schutzrechte angemeldet.

5 Arbeiten, die zu keiner Lösung geführt haben

Alle vorgenommenen Arbeiten wurden durchgeführt, um die geplanten Teilziele des Projektes zu erfüllen. Arbeiten, die zu keiner Lösung geführt hätten, wurden nicht durchgeführt.