Technische Universität München
Institut für Energietechnik MW7

Lehrstuhl und Laboratorium
für Hydraulische Maschinen und Anlagen

Objektorientierte Strukturen für Adaptive Multilevelverfahren zur Strömungssimulation

Andreas Müller

Vollständiger Abdruck der von der Fakultät für Maschinenwesen der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktor-Ingenieurs

genehmigten Dissertation.

Vorsitzender:	Univ.-Prof. Dr.-Ing., Dr.-Ing. habil. R. Friedrich
Prüfer der Dissertation:	
1.	Univ.-Prof. Dr.-Ing., Dr.-Ing. habil. R. Schilling
2.	Univ.-Prof. Dr. rer. nat. Chr. Zenger

Die Dissertation wurde am 25.4.2000 bei der Technischen Universität München eingereicht und durch die Fakultät für Maschinenwesen am 28.8.2000 angenommen.

Vorwort

Die vorliegende Dissertation entstand in den Jahren 1996-2000 während meiner Tätigkeit als wissenschaftlicher Mitarbeiter am Lehrstuhl für Hydraulische Maschinen und Anlagen der Technischen Universität München.

Mein Dank gilt Herrn **Univ.-Prof. Dr.-Ing., Dr.-Ing. habil. R. Schilling** für die Betreuung der Arbeit sowie die Bereitschaft, für Fragen und Diskussionen zur Verfügung zu stehen.

Herrn **Univ.-Prof. Dr. rer. nat. Chr. Zenger** danke ich nicht nur für die Übernahme des Koreferates, sondern auch für die fachliche Unterstützung. Ebenso bedanke ich mich bei Herrn **Univ.-Prof. Dr.-Ing., Dr.-Ing. habil. R. Friedrich** dafür, daß er den Vorsitz der Prüfungskommission übernommen hat.

Bedanken möchte ich mich auch bei meinen Kollegen **Dipl.-Ing. B. Szilagyi** und **Dipl.-Ing. R. Reinelt** für die Zusammenarbeit im Rahmen des Projektes mit der Firma Voith.

Abschließend möchte ich mich ganz besonders bei meiner **Familie** und meiner Freundin **Sylvia** für deren Unterstützung und Geduld bedanken.

München, September 2000

Andreas Müller

Inhaltsverzeichnis

Nomenklatur

Innerhalb dieser Dissertation gilt die Konvention, daß Vektorfelder im Gegensatz zu skalarwertigen Funktionen **fett** gedruckt sind. Vektoren sind mit Klein-, Matrizen mit Ausnahme der Matrix a in (4.3) mit Großbuchstaben bezeichnet. Im folgenden sind die wesentlichen Bezeichnungen aufgelistet. Die Bedeutung lokaler Hilfsgrößen ist dem jeweiligen Kontext zu entnehmen.

Akronyme

CONSIST	Objektorientierte Simulationsbibliothek
SIMPLE	Löser für Sattelpunktprobleme
SIP	Verfahren basierend auf unvollständiger Dreieckszerlegung
CG	Verfahren der konjugierten Gradienten
BiCGStab	stabilisiertes bi-konjugiertes Gradientenverfahren
CFD	numerische Strömungsmechanik
BPX	Multilevel Vorkonditionierer nach Bramble, Pasciak, Xu

Geometrische Größen

Ω_t	Kontrollraum im Absolutsystem
Ω	Kontrollraum im Relativsystem
$\partial\Omega$	Rand des Kontrollraums
Γ_+, Γ_-	Periodische Ränder
Γ_I, Γ_O	Ein-, Ausströmrand
Γ_B	Eulerrand
Γ_D	Dirichletrand
Γ_N	Neumannrand
$\mathbf{n}$	äußerer Normaleneinheitsvektor
Z, γ	Schaufelzahl, Teilungswinkel
$Q(\gamma)$, $Q(t)$	Drehung um den Winkel γ, zeitabhängige Drehung um die z-Achse
r	Abstand von der z-Achse
r_{ref}, u_{ref}	Referenzradius bzw. -geschwindigkeit
m, φ	Koordinaten zur Parametrisierung von Stromflächen
l, u	konforme Koordinaten zur Parametrisierung von Stromflächen
δ	Schichtdicke
$\mathbf{n}$, $\mathbf{t}$	Normal-, Tangentialvektor im $m-\varphi$-Koordinatensystem
$\boldsymbol{\nu}$, $\boldsymbol{\tau}$	Normal-, Tangentialvektor im $l-u$-Koordinatensystem
h	Maß für die Gitterweite
$\mathcal{T}_h$	Triangulation bezüglich h
τ, $\lvert\tau\rvert$	Bezeichnung für eine Zelle, deren Volumen bzw. Fläche
e, $\lvert e\rvert$	Bezeichnung für eine Kante bzw. Seite, deren Länge bzw. Fläche

Physikalische Größen

$\mathbf{c}$, c	Vektorfeld, Betrag der Absolutgeschwindigkeit
$\mathbf{w}$, w	Vektorfeld, Betrag der Relativgeschwindigkeit
$\mathbf{u}$, u	Vektorfeld, Betrag der Führungsgeschwindigkeit
$\boldsymbol{\omega}$, ω	Vektor, Betrag der Winkelgeschwindigkeit
c_u, n_u	Umfangskomponente von $\mathbf{c}$ bzw. $\mathbf{n}$
p	hydrostatischer Druck

p_t	Totaldruck
ρ	Dichte des Fluids
Q	Volumenstrom
$\dot{m}$	Massenstrom
$M_{Schaufel}$	Schaufelmoment um die z-Achse
ψ	Druckzahl für reibungsfreie Strömung, Stromfunktion
ψ_t	Druckzahl aus der Totaldruckänderung
ψ_{th}	Druckzahl aus der Umlenkung
ψ_M	Druckzahl aus dem Schaufelmoment
ψ_B	Stromfunktion entlang der Schaufel
ψ_+, ψ_-	Restriktion der Stromfunktion auf Γ_+, Γ_-
η_{num}	numerischer Wirkungsgrad
c_p	Druckkoeffizient
n_q	spezifische Drehzahl
w_m, w_φ	Geschwindigkeitskomponenten im $m-\varphi$-Koordinatensystem
w_l, w_u	Geschwindigkeitskomponenten im $l-u$-Koordinatensystem
w_φ^{in}, w_φ^{out}	Umfangskomponente der Relativgeschwindigkeiten am Ein- bzw. Ausströmrand
w_τ	Tangentialgeschwindigkeit in konformen Koordinaten
w_∞	Anströmgeschwindigkeit

Sonstige Bezeichnungen

$d\mathbf{x}$	dreidimensionales Lebesgue-Maß oder infinitesimales Volumenelement
$d\Gamma$	Lebesgue-Maß bezüglich Kurven bzw. Flächen oder infinitesimales Kurven- bzw. Flächenelement
$H^1(\Omega)$	Sobolevraum über Ω
$\mathbf{H}^1(\Omega)$	Sobolevraum vektorwertiger Funktionen über Ω
$H^1_{\Gamma_D}(\Omega)$	Sobolevraum über Ω mit verschwindenden Randwerten auf Γ_D
$L^2(M)$	Lebesgue-Raum der über M quadratisch integrierbaren Funktionen
$\mathbf{u}\cdot\mathbf{v}$	Euklidisches Skalarprodukt der Vektorfelder $\mathbf{u}$ und $\mathbf{v}$
(x,y)	Euklidisches Skalarprodukt der Vektoren x und y
$\|x\|_A$	$\sqrt{(Ax,x)}$
(f,g)	$L^2(\Omega)$-Skalarprodukt der Funktionen f und g
$< f,g >$	$L^2(\Gamma_N)$-Skalarprodukt der Funktionen f und g
$\|\|\|\psi\|\|\|$	Energienorm von ψ
P_1-Diskretisierung	Diskretisierung mit stückweise linearen Finiten Elementen
I, I_k	drei- bzw. k-reihige Einheitsmatrix
M_l	mit lumping-Techniken diagonalisierte Massenmatrix
$\mathrm{diag}(S)$	Diagonalanteil von S
$\mathcal{O}(h^\gamma)$, $\mathcal{O}(N)$	asymptotisch $\sim h^\gamma$, $\sim N$
$\kappa(A)$	spektrale Kondition der Matrix A
ψ^k	Funktionensystem auf Stufe k
ψ^k_{BPX}	Funktionensystem auf Level k
e_h	Diskretisierungsfehler
ε_h	Fehlerschätzer

Zusammenfassung

Diese Dissertation befaßt sich mit objektorientierten Strukturen, auf deren Basis sich adaptive Multilevelverfahren implementieren lassen. Zunächst werden Konzepte zur Gitterverwaltung, zum Aufbau numerischer Strukturen und zur Darstellung sowie Lösung der resultierenden Gleichungssysteme erläutert. Anschließend werden diese abstrakten Konzepte anhand zweier Beispiele konkretisiert.

Das erste Beispiel betrifft die Standard Finite Elemente Diskretisierung eines zweidimensionalen linearen elliptischen Randwertproblems 2. Ordnung. Die Netzverfeinerung geschieht adaptiv mit Hilfe eines a-posteriori Fehlerschätzers. Zur Lösung der linearen Gleichungssysteme wird ein multilevel vorkonditioniertes CG-Verfahren verwendet.

Als zweites Beispiel dient die Lösung der dreidimensionalen, stationären, inkompressiblen Euler Gleichungen im rotierenden Koordinatensystem. Die Diskretisierung erfolgt mit Hilfe eines von Rannacher und Turek entwickelten Finiten Elementes basierend auf hexalateralen Gittern. Zur Lösung der Gleichungssysteme wird ein nichtlineares Mehrgitterverfahren herangezogen. Lokale Verfeinerung sowie Multilevellösung sind vorbereitet.

Testrechnungen zeigen schließlich die Anwendbarkeit der Theorie auf praxisrelevante Fälle. Diese beziehen sich im besonderen auf Strömungen durch Laufräder hydraulischer Strömungsmaschinen.

Kapitel 1

Einführung

Numerische Simulation spielt in Wissenschaft und Technik eine immer größere Rolle. Ob in der Elektrotechnik, Thermodynamik, Struktur- oder Strömungsmechanik: Simulationsrechnungen in Verbindung mit Messungen bestimmen heute technische Entwicklungsprozesse. Eine herausragende Position nimmt dabei der CFD-Bereich (**C**omputational **F**luid **D**ynamics) ein. Viele der heute gebräuchlichen Simulationstechniken wurden im Zusammenhang mit strömungsmechanischen Problemen entwickelt oder zumindest getestet.

Anwendung finden die Erkenntnisse unter anderem in dem Industriezweig, der sich mit hydraulischen Strömungsmaschinen, wie Wasserturbinen oder Kreiselpumpen, beschäftigt. Dort wird Simulationssoftware neben der Nachrechnung sowohl zum Entwurf als auch zur Optimierung von Beschaufelungen eingesetzt.

Der Trend zur Simulation ist im Zusammenhang mit der Kostenentwicklung für reale Experimente und zweifelsohne auch mit Fortschritten in der Computer-Hardware zu sehen. Nicht zuletzt aber tragen verbesserte numerische Verfahren und daraus resultierend zuverlässigere und robustere Programme dazu bei.

Ein weiterer Aspekt betrifft die Simulationszeiten. Direkte numerische Simulation der dreidimensionalen, instationären Navier-Stokes Gleichungen ist momentan und wohl auch die nächsten Jahre industriell nicht praktikabel. Durch geeignete Modellierungsmaßnahmen, wie Reynoldsmittelung und Turbulenzmodellierung, ist es gelungen, die Rechenzeiten auf eine bis zwei Stunden auf durchschnittlichen Workstations zu reduzieren. Insgesamt jedoch stellen Simulationszeiten in dieser Größenordnung im industriellen Kontext nach wie vor einen beträchtlichen Kostenfaktor dar.

1.1 Adaptive Multilevelverfahren

In den vergangenen 20 Jahren sind erhebliche Anstrengungen unternommen worden, Rechenzeiten zu senken. Daß sich die Forschungstätigkeit weltweit dabei im wesentlichen auf die effiziente Lösung der algebraischen Gleichungssysteme konzentriert, hängt damit zusammen, daß dieser Baustein die Gesamtrechenzeit zu etwa 80% bestimmt. Die Erfolge sind beachtlich. Mit Hilfe von Mehrgitterverfahren, vgl. z.B. Brandt [13] oder Rüde und Zenger [58], die im Zuge dieser Anstrengungen entwickelt wurden, konnte das Laufzeitverhalten erheblich verbessert werden.

Gegenwärtig konzentriert sich das Interesse auf die Weiterentwicklung der Mehrgitterstrategie

auf lokal verfeinerte Netze. Diese sogenannten Multilevelverfahren verfolgen zwei Ziele: Neben der Rechenzeitersparnis ermöglichen lokal verfeinerte Gitter die Darstellung von Lösungsdetails. Genauso wie Mehrgitterverfahren anfangs mit Skepsis beurteilt wurden und heute in kommerzieller Simulationssoftware weit verbreitet sind, wird es sich auch mit Multilevelverfahren verhalten.

Besonders attraktiv ist das Konzept der lokalen Gitterverfeinerung dann, wenn es auf die Zonen des Netzes beschränkt wird, deren Verfeinerung aus Approximationsgesichtspunkten notwendig ist. Geschieht die Bestimmung dieser Zonen automatisch mit Hilfe lokaler Fehlerschätzer, spricht man von adaptiven Verfahren.

Erste Ansätze gehen auf Banks Programmpaket PLTMG Anfang der achtziger Jahre zurück, vgl. BANK [1]. Für lineare elliptische Modelle sind zuverlässige und effiziente Fehlerschätzer bereits verfügbar. Für kompliziertere Probleme, wie die Navier-Stokes Gleichungen, wird weltweit mit Hochdruck an deren Entwicklung gearbeitet, vgl. [23].

Ob die Netze nun voll adaptiv oder mit Hilfe von Steuermechanismen erzeugt werden, die Netzgenerierung ist mit der Diskretisierung sowie Lösung der resultierenden Gleichungssysteme verschmolzen. Dies steht im Widerspruch zu konventionellen Strategien, bei denen ein einzelnes Gitter vor der eigentlichen Simulationsrechnung erzeugt wird.

Hinzu kommt, daß die Aufgabe der Netzgenerierung im herkömmlichen Stil überfrachtet scheint. Dem dadurch entgegenzuwirken, daß die wesentlichen Postulate, Geometrieauflösung und Anpassung des Netzes an die zu approximierende Lösung, entkoppelt werden, ist ein zentrales Ziel des Verfeinerungskonzeptes, vgl. Abschnitt 2.2.

Nach KALLINDERIS [39] sind Netzgenerierung, Diskretisierung und Lösung der Gleichungssysteme darüber hinaus als gleichberechtigte Bausteine anzusehen. Die Netzgenerierung isoliert zu diskuttieren, macht keinen Sinn. Demzufolge hängt die Qualität eines Gitterkonzeptes im wesentlichen davon ab, ob es zu Diskretisierung und Lösung paßt oder nicht. Gleiches gilt für Diskretisierung bzw. Lösung in Bezug auf die jeweils übrigen Bausteine.

Als Beispiel seien Netze mit *hanging nodes* genannt, vgl. Abbildung 7.1. Derartige Techniken tragen erheblich zur Flexibilisierung der Netzgenerierung bei, machen jedoch nur dann Sinn, wenn geeignete Lösungsprozeduren wie Multilevelverfahren sowie flexible Datenstrukturen, die u.a. die Assemblierung von Gleichungssystemen aus lokalen Steifigkeitsmatrizen bzw. Lastvektoren, vgl. Abschnitt 2.3.2, unterstützen, zur Verfügung stehen. Darüber hinaus bereitet die numerische Behandlung der Übergangsbedingung (7.1) insbesondere dann keine wesentlichen Schwierigkeiten, wenn die Systematik von Finite Elemente Ansätzen zugrundegelegt wird.

1.2 Finite Elemente

Kommerzielle CFD-Programme stützen sich vornehmlich auf Finite Volumen Diskretisierungen. Das Prinzip dabei ist, die Erhaltungseigenschaft der kontinuierlichen Gleichungen zu nutzen. Zu diesem Zweck werden die Erhaltungsgleichungen über beliebige Kontrollvolumina integriert. Die Anwendung des Integralsatzes von Gauß reduziert das Differentialgleichungsproblem auf Flußbilanzen über Kontrollvolumina. Diese Formulierung bezeichnet man als *integrale Form der Erhaltungsgleichung.*

Anders bei Finite Elemente Diskretisierungen: Bevor über das Gesamtgebiet integriert wird, wird die Differentialgleichung mit geeigneten Testfunktionen multipliziert. Stellt die Differentialglei-

chung eine Kräftebilanz dar, entspricht dies dem Prinzip der virtuellen Arbeit bzw. Leistung. In manchen Fällen, wie etwa dem Stokes-Problem, ist die daraus resultierende *Variationsformulierung* einem Optimierungsproblem äquivalent. Finite Elemente Ansätze leiten sich also im Gegensatz zu denen mit Finiten Volumina nicht von der Erhaltungseigenschaft der kontinuierlichen Gleichungen, sondern von Variationsprinzipien ab.

Daß beide Ansätze äquivalent sind, ist z.B. bei LeVeque [43] nachzulesen. Der Vorteil der Variationsformulierung liegt in der Systematik, die sie bietet: Sind auf der Basis einer Zerlegung des Rechengebietes diskrete Ansatz- bzw Testräume definiert, ist das weitere Vorgehen „straightforward“. Dies garantiert Transparenz, die insbesondere dann von Vorteil ist, wenn Programme an andere Programmierer weitergegeben werden sollen.

Davon abgesehen läßt die Finite Elemente Strategie eine Konvergenztheorie zu, die sich auch in praxisrelevanten Anwendungen bewährt hat. Als Beispiel für die Leistungsfähigkeit der *numerischen Analysis* sei nur die *Babuska-Brezzi-Bedingung* genannt, die numerische Instabilitäten bei gemischten Ansätzen erklärbar gemacht hat, vgl. Abschnitt 5.2.1.

Was die Approximationsgüte angeht, unterscheiden sich Finite Elemente von vergleichbaren Finite Volumen Diskretisierungen nicht, vgl. Bank und Rose [2]. Dies unterstreichen auch Vergleichsrechnungen, vgl. Schäfer et al. [60], bei denen das Rannacher-Turek-Element, vgl. Abschnitt 5.2.1, im Vergleich zu anderen Diskretisierungen ausgezeichnete Ergebnisse lieferte.

Insbesondere im Zusammenhang mit Adaptivitätskonzepten jedoch weisen Finite Elemente Ansätze wesentliche Vorteile auf: Die Systematik erleichtert die Konstruktion nicht-heuristischer Fehlerschätzer, zumal im Gegensatz zu Finite Volumen Ansätzen die lokale Erhaltungseigenschaft als Kriterium für lokale Netzverfeinerung zur Verfügung steht, vgl. Abschnitt 4.6.

Von den Laufzeiten und der Systematik abgesehen spielt die Flexibilität von Codes eine zunehmend wichtige Rolle. Wenn minimale Änderungen der Aufgabenstellung dazu führen, daß Programme im Grunde neu entwickelt werden müssen, ist dies heute aus Produktivitätsgesichtspunkten nicht mehr vertretbar. Flexiblere Strukturen sind deshalb notwendig. Dies umfaßt neben algorithmischen Aspekten auch die Frage nach flexibleren Datenstrukturen sowie einer Programmiersprache, die deren Umsetzung unterstützt. Objektorientierte Programmierung trägt diesen Anforderungen Rechnung.

1.3 Objektorientierte Programmierung

Die Strategie konventioneller *prozeduraler* Programmiersprachen besteht darin, ein gegebenes Problem in mehrere Teilprobleme zu unterteilen und diesen Prozeß so lange rekursiv fortzusetzen, bis die Subprobleme einer überschaubaren Lösung zugänglich sind. *Objektorientierte* Programmiersprachen ziehen dieser *horizontalen* Sichtweise eine *vertikale* vor.

Zur Erläuterung stelle man sich ein Unternehmen vor, in dem es 3 Hierarchiestufen gibt: einen Vorstand und mehrere Abteilungen, die ihrerseits wieder unterteilt sein können, vgl. Abbildung 1.1.

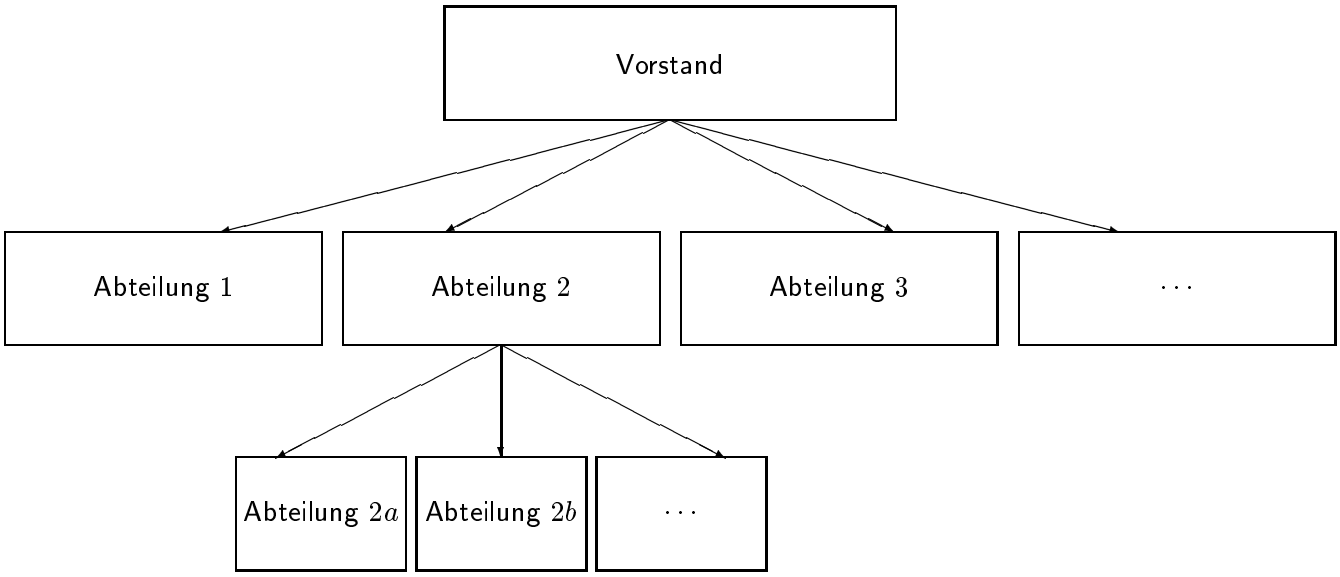

Abb. 1.1: Hierarchische Strukturierung eines Unternehmens

Der Vorstand wird sich beispielsweise nicht mit der Urlaubsplanung in Abteilung 2*b* beschäftigen. Vielmehr werden in der Vorstandsetage grundlegende Entscheidungen *abstrakter* Natur getroffen, die in den unteren Ebenen zunehmend *konkretisiert* werden.

Eine objektorientierte Programmiersprache funktioniert nach demselben Prinzip. Zunächst werden unterschiedliche Abstraktionsebenen definiert, indem Gemeinsamkeiten der zu lösenden Probleme extrahiert werden. Die Probleme werden dann auf möglichst abstrakter Ebene einer Lösung zugeführt, die nach unten hin spezialisiert werden kann. Die Kunst objektorientierter Analyse und des Designs geeigneter Strukturen besteht darin, mit Weitblick angemessene Abstraktionsniveaus zu spezifizieren entsprechend dem Grundsatz: *So konkret wie möglich, so abstrakt wie nötig.*

Ziel ist es, Software zu erstellen, die *wiederverwendbar* und *erweiterbar* ist und die den *Baustein*- gegenüber dem *System*gedanken favorisiert. Im günstigsten Fall funktioniert ein solcher Baustein als *black box*. Hilfreich ist derartige Flexibilität insbesondere im Hinblick auf den erkennbaren Trend zu interdisziplinären Fragestellungen. Man denke beispielsweise an gekoppelte Systeme der Mechatronik. Viele numerische Algorithmen sind vom speziellen Anwendungsgebiet unabhängig. Es bietet sich an, sie auch entsprechend abstrakt zu formulieren.

Im Rahmen numerischer Software entspricht die Baustein-Philosophie der Erkenntnis, daß mathematische Algorithmen immer an Voraussetzungen gebunden sind und deshalb im allgemeinen nicht — oder zumindest nicht optimal — den gesamten Anwendungsbereich abzudecken in der Lage sind, der von Software im industriellen Einsatz gefordert wird. Bei sauber konstruierten objektorientierten Programmen ist der Austausch eines Bausteins gegen einen anderen dann in sehr viel kürzerer Zeit möglich als bei prozeduraler Programmierung.

Dieser Gedanke findet häufig keine Berücksichtigung: Die Diskussion über Zeiten reduziert sich ausschließlich auf das Laufzeitverhalten. Daß man bei objektorientierter Software einen Laufzeitüberhang von durchschnittlich etwa 15% im Vergleich zu konventionellen Programmen in Kauf nehmen muß, ist unbestritten. Detaillierte Untersuchungen dazu sind bei Daehlen und Tveito [21] nachzulesen. Dieser Effekt ist auf ähnliche Reibungsverluste zurückzuführen, wie sie bei

der Strukturierung eines Unternehmens gemäß Abbildung 1.1 zwischen verschiedenen Abteilungen auftreten.

Der Grundgedanke objektorientierter Programmierung besteht somit darin, Abstraktion zu unterstützen, um Komplexität beherrschbar zu machen. Dieses Prinzip findet zunehmend Anwendung weit über die Softwareentwicklung hinaus. Dabei ist klar, daß Abstraktion alleine noch keine Probleme löst. Mit Hilfe von Abstraktion lassen sich Problemlösungsprozesse jedoch dadurch effizienter gestalten, daß grundsätzlich mehrere Probleme gleichzeitig angegangen werden. Diese Strategie auf industrielle oder auch andere Prozesse angewendet verspricht erhebliche Produktivitätssteigerung.

1.4 Zielsetzung

Unter diesen Gesichtspunkten ist die vorliegende Dissertation entstanden. Speziell sollte im Rahmen eines Optimierungstools für Wasserturbinen ein dreidimensionaler Euler-Code entwickelt werden, um einerseits im Vergleich zu einem Navier-Stokes Verfahren Rechenzeit zu sparen und andererseits im Vergleich zu einem zweidimensionalen Potentialverfahren höherwertige Ergebnisse zu erzielen. Die Optimierungssoftware als solche wurde am Lehrstuhl für Hydraulische Maschinen und Anlagen der Technischen Universität München in Zusammenarbeit mit dem Lehrstuhl V für Informatik und der Firma Voith Hydro erstellt.

Der Euler-Code setzt auf einer bereits vorher implementierten objektorientierten Simulationsbibliothek auf, zu deren Verifikation ein zweidimensionales Potentialverfahren, das die reibungsfreie Strömung zwischen den Schaufeln einer hydraulischen Strömungsmaschine simuliert, entwickelt wurde. Diese drei Entwicklungsstadien sind Gegenstand der folgenden Kapitel.

Ein Blick in die Literatur bestätigt den Eindruck, daß sich im Verlauf vieler Jahre Kommunikationsprobleme zwischen Mathematikern und Ingenieuren, was numerische Simulation angeht, angehäuft haben. Die Art der Darstellung ist so unterschiedlich, daß selbst die gemeinsame Aufgabenstellung zum Teil nicht mehr erkennbar ist. Zweifelsohne kamen und kommen die wesentlichen Impulse, was die Weiterentwicklung der Methodik angeht, aus dem Mathematikbereich. Was bisweilen fehlt, ist die Bereitschaft, diese Techniken allgemein verständlich darzustellen.

Im Rahmen dieser Dissertation kann es nicht gelingen, die Kommunikationsprobleme vollständig zu beseitigen. Es wird jedoch der Versuch unternommen, im Anschluß an formale mathematische Darstellungen, die sich in der Numerik bewährt haben, Denk- und Vorgehensweisen anhand einfacher Beispiele zu demonstrieren. Auch physikalische Interpretationen können dort, wo sie Sinn machen, zum Verständnis beitragen.

Was die Struktur dieser Dissertation anbelangt, so werden in Kapitel 2 die Konzepte der objektorientierten Simulationsbibliothek CONSIST erläutert. An eine kurze Einführung in die wesentlichen Grundlagen der Theorie der reibungsfreien Strömung in hydraulischen Maschinen in Kapitel 3 schließt sich in Kapitel 4 die numerische Lösung eines Potentialproblems für die Strömung auf vorgegebenen Stromflächen an. Kapitel 5 ist der Lösung der dreidimensionalen stationären, inkompressiblen Euler Gleichungen im rotierenden System gewidmet, die die reibungsfreie Strömung im Laufrad einer hydraulischen Strömungsmaschine modelliert, bevor in Kapitel 6 die Theorie an konkreten Beispielen überprüft wird. Abschließend werden in Kapitel 7 die wesentlichen Aspekte zusammengefaßt und durch Bemerkungen zu Entwicklungstendenzen ergänzt.

Kapitel 2

Die objektorientierte Simulationsbibliothek CONSIST

Dieses Kapitel beschäftigt sich mit der Frage, nach welchen Gesichtspunkten die Grundmodule in CONSIST konzipiert sind. Das Acronym CONSIST steht dabei für **C**omprehensive **O**bject-oriented **N**umerical **S**cientific and **I**ndustrial **S**imulation **T**oolkit. Dabei wird immer wieder folgendes Optimierungsproblem im Mittelpunkt stehen: *So konkret wie möglich, so abstrakt wie nötig.* Ist der Grad der Abstraktion zu niedrig gewählt, sind die Bausteine nicht flexibel genug einsetzbar. Ist die Software hingegen zu abstrakt formuliert, beeinträchtigt dies zum einen die Effizienz und zum anderen die Akzeptanz des Programmierers, der die Strukturen anwenden soll.

In den folgenden Abschnitten werden die wesentlichen Module erläutert. Dazu zählen die Gitter, die darauf aufsetzenden numerischen Strukturen und letztlich die Formulierung und Lösung von Gleichungssystemen. Zunächst werden zum Verständnis notwendige Grundkenntnisse über objektorientierte Programmierung bereitgestellt.

2.1 Objektorientierte Programmierung mit C++

Die Philosophie objektorientierter im Vergleich zu konventioneller Programmierung wurde bereits in Abschnitt 1.3 erläutert. C++ als *hybride* Programmiersprache kombiniert nun die prozedurale mit der objektorientierten Denkweise. Realisiert werden die Konzepte objektorientierter Programmierung im wesentlichen mittels *Klassen*, *Vererbung* und *Polymorphie*.

Klassen sind dabei nichts anderes als benutzerdefinierte Typen, die die Abstraktionsebenen festlegen. Eine Klasse umfaßt *Daten* und *Funktionen* zu deren Manipulation. Der Umgang mit Klassen unterscheidet sich von dem mit eingebauten Typen, wie *integer*, *float* oder *double*, nicht wesentlich.

Vererbung — auch *Ableitung* genannt — regelt die Relation der Klassen untereinander, was den Grad der Abstraktion angeht: Ist eine Klasse B von einer Klasse A *public abgeleitet*, so stellt B eine Spezialisierung von A dar, steht also in der Abstraktionshierarchie tiefer.

Das dritte grundlegende Konzept der *Polymorphie* oder *Vielgestaltigkeit* wird in C++ mit Hilfe *virtueller Funktionen* realisiert und funktioniert so, daß abstrakte Schnittstellen auf konkreter Ebene auf verschiedene Arten und Weisen redefiniert werden können. Diese Flexibilität ist es letztlich, die

die Implementierung von black box Bausteinen und damit Erweiter- und Wiederverwendbarkeit von Software möglich macht.

Die folgenden Abschnitte bieten eine ganze Reihe konkreter Beispiele, um die Grundbegriffe zu vertiefen. Details zu objektorientierter Programmierung mit C++ sind z.B. bei STROUSTRUP [70], LIPPMAN [44] oder MEYERS [49] nachzulesen.

2.2 Gitter

Konventionelle Netzgeneratoren erzeugen Gitter, die folgenden Anforderungen genügen sollen:

- Auflösung der geometrischen Struktur des Rechengebietes
- Anpassung des Gitters an die Strömung
- Möglichst einfach zu verwaltende und damit effiziente Datenstrukturen
- Möglichst günstige geometrische Eigenschaften der einzelnen Zellen

In Punkt 3 ist im allgemeinen an strukturierte oder zumindest blockstrukturierte Gitter gedacht. Häufig wird dabei auf die Lösung nichtlinearer elliptischer Differentialgleichungen zurückgegriffen. In vielen Anwendungsfällen erweist sich bereits Punkt 1 als so problematisch, daß die beiden zusätzlichen Vorgaben 2 und 3 zu unbefriedigenden Ergebnissen im Sinne von Punkt 4 führen.

Die Gitterstrategie, die in CONSIST hinterlegt ist, versucht nun, die Punkte 1 und 2 zu entkoppeln und auf Punkt 3 gänzlich zu verzichten. Dabei stützt sie sich auf folgende Vorgehensweise:

- Erzeugung eines Grobgitters
- Sukzessive lokale Verfeinerung des Grobgitters

Punkt zwei impliziert selbstverständlich den Spezialfall globaler Verfeinerung. Die Aufgabe der Grobgittergenerierung, vgl. Abbildung 2.1, besteht darin, ein möglichst uniformes Gitter mit einer möglichst geringen Zahl an Zellen im Sinne von Punkt 4 zu erzeugen, das die Geometrie nur andeutungsweise auflöst. Eine verbesserte Geometrieauflösung erfolgt im Zuge fortschreitender Verfeinerung, vgl. Abbildung 2.2.

Entscheidend dabei ist, daß Ecken, die bei der Verfeinerung von Randkanten bzw. Randseiten entstehen, — wie in Abbildung 2.3 illustriert — auf die Kontur verschoben werden. Zu diesem Zweck sind Randkanten bzw. -seiten mit Information über krummlinige Ränder versehen. Näheres zur Implementierung dieser Strategie ist bei REINELT [56] nachzulesen.

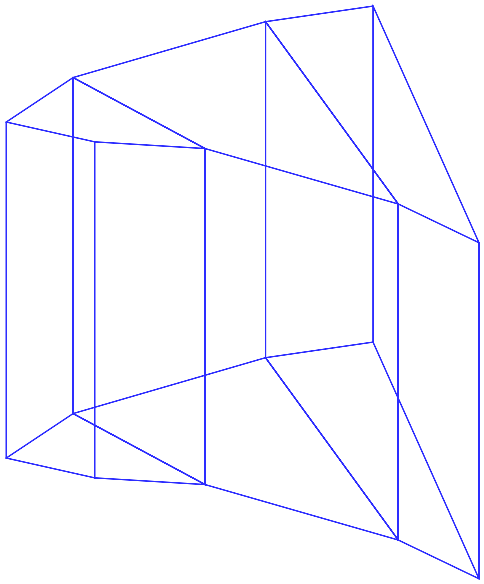

Abb. 2.1: Initiales Netz

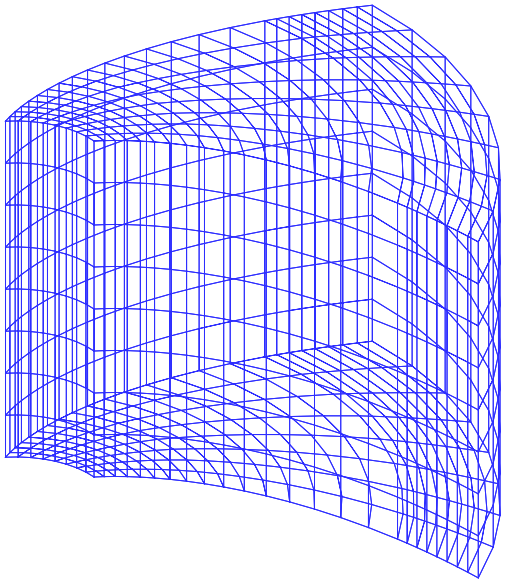

Abb. 2.2: Netz nach 3 Verfeinerungsschritten

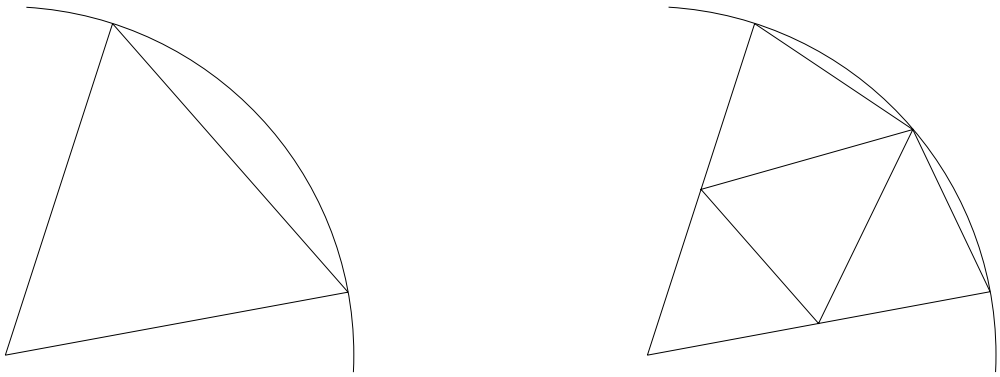

Abb. 2.3: Verfeinerung randnaher Zellen

Die Anpassung des Gitters an die zu approximierende Lösung spielt im Rahmen der Grobgittererzeugung überhaupt keine Rolle. Dies ist Aufgabe der lokalen Verfeinerung, die nach einem vorgegebenen Steuermechanismus oder im Idealfall mittels adaptiver Fehlerschätzung geschehen kann. Darüber hinaus bietet es sich an, die resultierende Hierarchie von Gittern als Grundlage für Mehrgitter- bzw. Multilevelverfahren zu nutzen.

Der fundamentale Unterschied zur herkömmlichen Netzgenerierung besteht also darin, daß die Gittererzeugung mit Diskretisierung und Lösung gekoppelt ist. Die Verwendung kommerzieller Netzgeneratoren ist demzufolge zwar möglich, jedoch nicht im Sinne des Konzeptes.

Zur Verwaltung lokal verfeinerter Netze sind dynamische und flexible Datenstrukturen notwendig. Es macht deshalb Sinn, auf unstrukturierte Gitter zurückzugreifen, zumal diese zusätzliche Flexibilität die Grobgittergenerierung erleichtert. Es hat sich ohnehin gezeigt, daß maximale Flexibilität

notwendig ist, wenn die Grobgittererzeugung bei komplexen Geometrien automatisierbar sein soll. Wie die Testrechnungen in Kapitel 6 belegen, beeinträchtigt diese Forderung die Gesamteffizienz nicht wesentlich.

2.2.1 Gitterverwaltung

Die Verwaltung strukturierter Netze geschieht üblicherweise mittels Indizierung, die unstruktierter Gitter mit Hilfe von Listen oder Feldern. Um eine durch lokale Verfeinerung entstandene Gitterhierarchie zu verwalten, bieten sich *Baumstrukturen* an. Die Entwicklung eines *quad-tree*s, der aus quadrilateralen Zellen besteht, ist Abbildung 2.4 zu entnehmen.

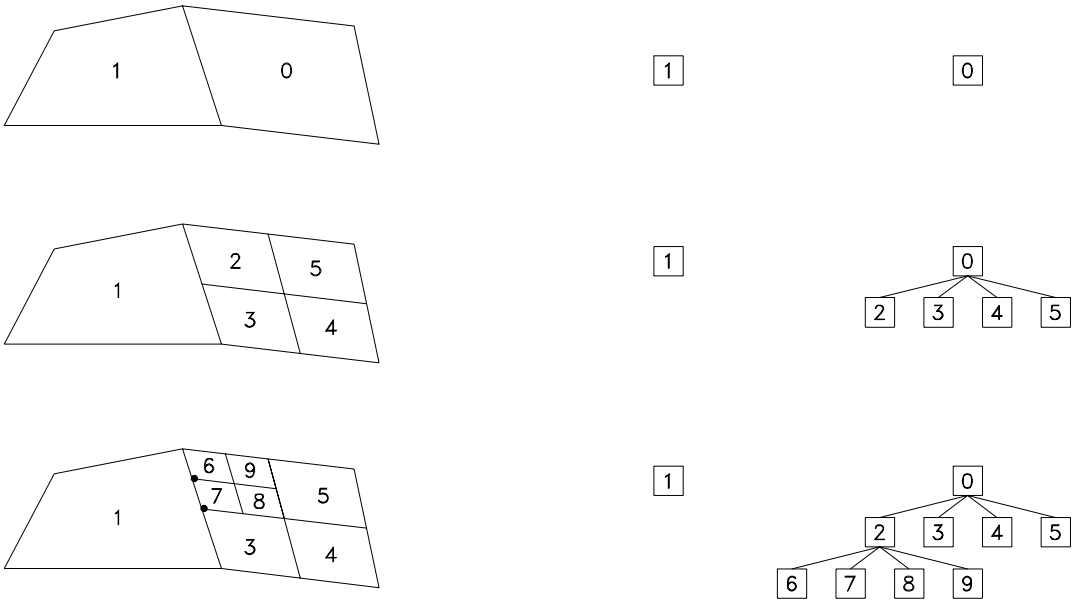

Abb. 2.4: Entwicklung eines *quad-tree*s

Die verbale Beschreibung solcher Strukturen geschieht mit Hilfe von Begriffen aus dem Familienleben. Die bei Verfeinerung einer Zelle entstehenden Zellen werden als *Kinder* dieser Zelle bezeichnet, die Ursprungszelle selber als *Vater*. Mit Hilfe von Zeigern hat eine Vaterzelle unmittelbar Zugriff auf seine Kinder und umgekehrt.

Außerdem sind im Hinblick auf Multilevelverfahren die Zellen, die zu einem *Level* gehören, in Form einer verketteten Liste organisiert. Der Begriff des Levels ist rekursiv definiert:

Definition 2.1 *Eine Zelle gehört zu Level* 0*, wenn sie Bestandteil des initialen Gitters ist. Eine Zelle gehört zu Level* k *(*$k > 0$*), wenn sie durch Verfeinerung einer Zelle auf Level* $k-1$ *entstanden ist.*

Im vorliegenden Beispiel besteht der Level 0 aus den Zellen $\{0, 1\}$, der Level 1 aus $\{2, 3, 4, 5\}$ und der Level 2 aus $\{6, 7, 8, 9\}$. Wesentlich ist im Multilevelkontext die Unterscheidung zwischen *Level* und dem, was im folgenden als *Stufe* bezeichnet wird.

Definition 2.2 *Eine Zelle gehört zu Stufe* 0*, wenn sie Bestandteil des initialen Gitters ist. Die Stufe* k *(*$k > 0$*) umfaßt neben den Zellen des Levels* k *die Zellen, deren Level kleiner als* k *ist und die nicht weiter verfeinert sind.*

Im dargestellten Fall besteht Stufe 0 aus den Zellen $\{0, 1\}$, Stufe 1 aus $\{1, 2, 3, 4, 5\}$ und Stufe 2 aus $\{1, 3, 4, 5, 6, 7, 8, 9\}$. Die feinste Stufe ist anschaulich gegeben durch die Blätter des Baumes.

2.2.2 Netzkomponenten

Netzkomponenten sind die lokalen geometrischen Objekte unterschiedlicher Dimension, aus denen das Netz aufgebaut ist: Ecken, Kanten, Seiten und Zellen.

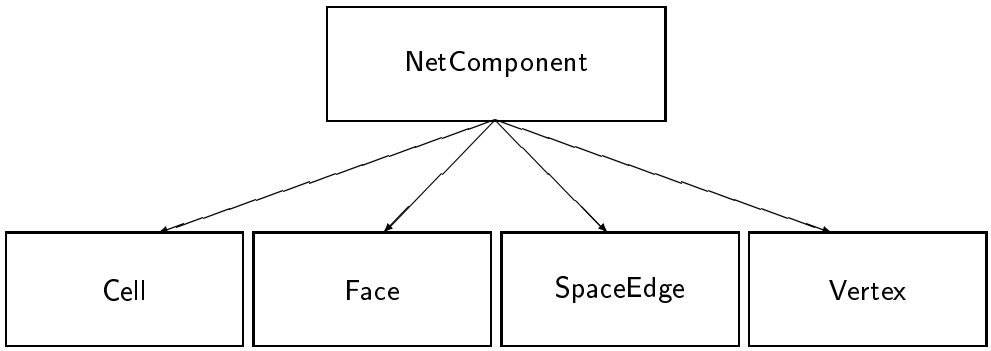

Abb. 2.5: Klassenbaum für Netzkomponenten

Die Klasse NetComponent übernimmt dabei die Funktion einer abstrakten Basisklasse. Die wesentlichen Elementfunktionen sind in der folgenden Klassendeklaration zusammengefaßt:

```
// ********************************************************************************
class NetComponent
// ********************************************************************************
{
        // ************ Public Interface ******************
public:
        virtual ~NetComponent();
        virtual const Node* accessNode(unsigned nodetype) const;
        Node* supplyNode(unsigned nodetype);
        virtual void imposeBdCond(const Data& data,unsigned nodetype) const;
protected:
        virtual Node* createNode(unsigned nodetype);
};
```

Neben einem *virtuellen Destruktor*, dessen Zweck u.a. von MEYERS [49] erläutert wird, gibt es drei Funktionen, die mit der Verwaltung von Knoten, vgl. Abschnitt 2.3.1, zu tun haben. Das Argument nodetype gibt dabei an, um welchen Knoten es sich im Falle mehrerer Variabler handelt.

- accessNode(): Gibt einen bereits existierenden Knoten zurück.
- createNode(): Erzeugt einen Knoten.

- supplyNode(): Falls bereits ein Knoten existiert, wird dieser zurückgegeben. Ansonsten wird createNode() aufgerufen.

Die *protected*-Deklaration der Funktion createNode() stellt sicher, daß ausschließlich die Klasse NetComponent und davon abgeleitete Klassen Knoten erzeugen können. Dieses Prinzip wird als *information hiding* bezeichnet und dient dem Zweck, die Gefahr nicht-lokaler Seiteneffekte zu minimieren und die Lokalisierung von Fehlern zu erleichtern.

Die Funktion imposeBdCond() stellt eine Schnittstelle zur Implementierung von Dirichletrandbedingungen zur Verfügung. Die nötige Information dafür wird durch ein Objekt vom Typ Data bereitgestellt, in dem Randbedingungen in Form von Funktionen hinterlegt sind. Abgeleitete Klassen können die virtuelle Methode redefinieren, indem sie Funktionen in Data je nach Diskretisierung in Ecken auswerten, vgl. Kapitel 4, oder Mittelwerte über Seiten bilden, vgl. Kapitel 5.

Die Klassen Vertex, SpaceEdge, Face bzw. Cell stellen Spezialisierungen von NetComponent zur Repräsentation von Ecken, Kanten im Raum, Seiten bzw. Zellen dar.

Ecken

Ecken verwalten als wesentliche Information die Koordinaten des mit ihnen assoziierten Punktes. Für Ecken im zwei- bzw. dreidimensionalen Raum geschieht dies in den Klassen PlaneVertex bzw. SpaceVertex.

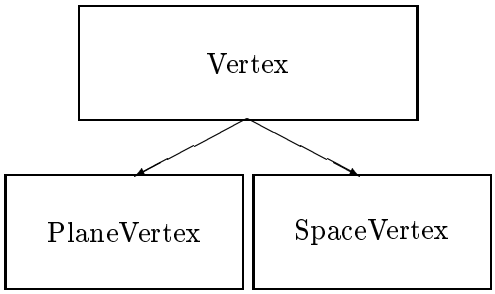

Abb. 2.6: Klassen für Ecken

Keine dieser Klassen ist mit Topologieinformation versehen. Denkbar wäre, mit jeder Ecke Zeiger auf angrenzende Kanten abzuspeichern. Da die Zahl der Kanten jedoch in einem unstrukturierten Netz a-priori nicht nach oben beschränkt ist, wird aus Speichergründen darauf verzichtet. Anders als in strukturierten Netzen steht also Information über benachbarte Ecken nicht zur Verfügung. Geeignete Techniken, wie sie beispielsweise im Zuge der Assemblierung der Gleichungssysteme, vgl. Abschnitt 2.3.2, zum Einsatz kommen, machen diese Funktionalität überflüssig.

Die Klasse Vertex verwaltet stattdessen die Zahl der die Ecke referenzierenden Kanten, vgl. Hiptmair [34]. Dies geschieht mit Hilfe des *privaten* Datenelementes refCount.

```
// ***************************************************************************
class Vertex : public NetComponent
// ***************************************************************************
```

```
{
        // ******* Internal Representation ************
private:
        unsigned refCount;

        // ******* Public Interface ******************
public:
        void incrementRefCount();
        void decrementRefCount();
};
```

Die Methoden incrementRefCount() und decrementRefCount() sind selbsterklärend. Ist der Referenzzähler gleich Null, so wird der Destruktor aufgerufen: Die Ecke wird aus dem Speicher entfernt. Dies ist insbesondere im Hinblick auf instationäre Rechnungen sinnvoll, wo neben Verfeinerungen auch Vergröberungen des Gitters sinnvoll sein können, vgl. Bey [9].

Kanten

Im Gegensatz zu Ecken sind Kanten sehr wohl mit topologischer Information versehen: Sie verwalten Zeiger auf ihre beiden Endpunkte. Eine Kante im dreidimensionalen Raum wird durch eine Klasse repräsentiert, die bereits eine ganze Reihe von Aufgaben selbständig übernehmen kann:

```
// ***************************************************************************
class SpaceEdge : public NetComponent
// ***************************************************************************
{
        // *********** Internal Representation ***************
private:
        SpaceVertex* vertex[2];
        SpaceEdge* children[2];
        unsigned refCount;

        // *********** Public Interface **********************
public:
        SpaceVertex* split();
};
```

Neben topologischer Information hat die Klasse SpaceEdge direkt Zugriff auf ihre Kinderkanten children sowie einen Referenzzähler refCount auf angrenzende Seiten. Zeiger auf diese Seiten werden nicht explizit abgespeichert. Auf die Auflistung aller topologischer sowie geometrischer Elementfunktionen wurde verzichtet.

Lediglich die Funktion split() ist erwähnt, um die Prinzipien der *Kapselung* von Daten im Zusammenhang mit *information hiding* zu erläutern. Bei prozeduraler Programmierung sind sowohl

die Daten als auch die sie manipulierenden Funktionen im wesentlichen global organisiert. Anders bei objektorientierten Sprachen: Daten und die ihnen zugeordneten Funktionen sind in Klassen zusammengefaßt. Sind Daten zudem als *private* deklariert, haben Funktionen, die keine Elementfunktionen der Klasse sind, keinen Zugriff.

Diese Form der Datenkapselung bildet die Grundlage dafür, daß Klassen eigenverantwortlich Aufgaben wahrnehmen können. Die Verfeinerung einer Kante in split() beispielsweise stützt sich ausschließlich auf Daten der Klasse. Damit gelingt es, einem Prinzip Rechnung zu tragen, das in der Politik als *Subsidiarität* bezeichnet wird: Aufgaben werden dorthin *delegiert*, wo sie am besten erledigt werden können.

Dieser Gedanke wird in Abschnitt 2.2.3 im Zusammenhang mit der Verfeinerung eines Dreiecks nochmals aufgegriffen. Kanten im Dreidimensionalen unterscheiden sich im übrigen von solchen im Zweidimensionalen im wesentlichen nur dadurch, daß letztere zusätzlich als Seiten fungieren.

Seiten

Seiten werden durch die Klasse Face repräsentiert.

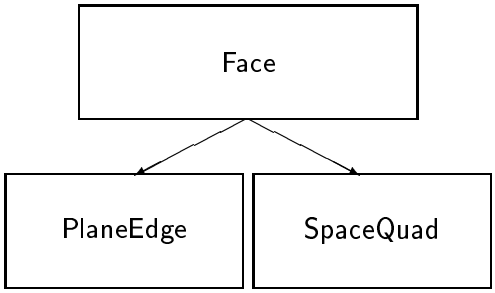

Abb. 2.7: Klassen für Seiten

Diese Klasse verwaltet Zeiger auf angrenzende Zellen. Nur Seiten haben demnach einen Bezug zu höherdimensionalen Objekten. Der Grund ist naheliegend. Ob es sich um SpaceQuads, also Seiten von Oktaedern im Dreidimensionalen, oder um PlaneEdges, also Kanten im Zweidimensionalen, handelt: Es existieren immer genau zwei Nachbarn.

Legt man die Nachbardefinition aus Abschnitt 2.2.3 zugunde, so gilt dies auch für den Fall irregulärer Ecken, die bei lokaler Verfeinerung auftreten können.

Zellen

Die Basisklasse für Zellen bildet die Klasse Cell. Die wesentlichen Daten sowie Methoden lauten wie folgt:

```
// ********************************************************************************
class Cell : public NetComponent
// ********************************************************************************
{
```

```
        // ******** Internal Representation ***************
protected:
        Patch* children;
        Patch* brothers;
        real estimatedError;

        enum {MaxNumDiffNodes = 8};
        static Array<NetComponent> ntcomp [MaxNumDiffNodes];
        static ...

        // ******** Public Interface **********************
public:
        virtual real estimateError(const Data&);
        virtual Patch* refineRegularly() = 0;
        virtual void setNodeComponents() = 0;
        virtual void setLocAssembleFcts() = 0;
};
```

Neben den Kindern children können auch die Brüder brothers einer Zelle direkt referenziert werden. Als Brüder sind dabei die Zellen zu verstehen, die mit der aktuellen die Vaterzelle gemeinsam haben. Der Vollständigkeit halber sei darauf hingewiesen, daß die Kontainerklasse Patch ein Feld von Zellen verwaltet, und real für *float* im Falle einfacher bzw. *double* bei doppelter Genauigkeit steht.

Neben der Variablen estimatedError zur Speicherung des geschätzten lokalen Fehlers im adaptiven Kontext, gibt es eine Reihe *statischer* Felder, wie z.B. ntcomp, die im Rahmen der Assemblierung von Bedeutung sind. Ihre Funktion wird zum Teil in Abschnitt 2.3.2 erläutert. Dies gilt auch für die virtuellen Funktionen setNodeComponents() sowie setLocAssembleFcts().

Die Methoden estimateError() bzw. refineRegularly() fungieren als Schnittstellen für lokale Fehlerschätzung bzw. reguläre Verfeinerung einer Zelle, vgl. S. 16. Die *rein virtuell* deklarierten Funktionen müssen in abgeleiteten Klassen, wie PlaneTriangle bzw. PlaneQuad und Hexa zur Repräsentation von Dreiecken bzw. Vierecken im Zweidimensionalen und hexalateralen Zellen im Dreidimensionalen, redefiniert werden.

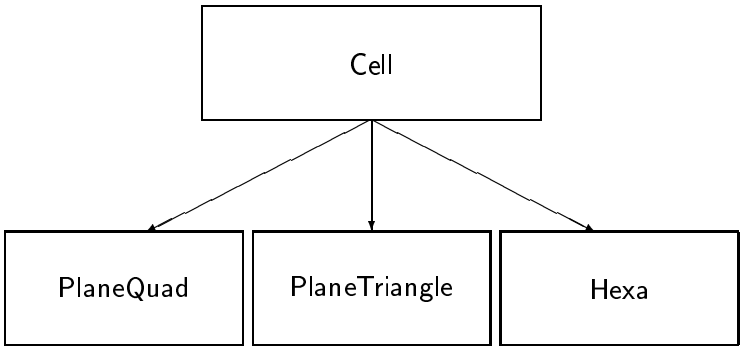

Abb. 2.8: Klassen für Zellen

2.2.3 Verfeinerung

Um die Bedeutung der Verfeinerung innerhalb der Gesamtstrategie einschätzen zu können, sei zunächst die Organisation einer Simulationsrechnung skizziert:

1. Erzeugung eines Grobgitters und Assemblierung des Gleichungssystems.
2. Lösung des algebraischen Systems.
3. Abbruch, falls beispielsweise die maximale Anzahl an Verfeinerungsschritten erreicht ist, oder der geschätzte Fehler eine gegebene Schranke unterschritten hat.
4. Markierung der zu verfeinernden Zellen mit Hilfe eines Fehlerschätzers oder nach einem vorgegebenen Steuermechanismus.
5. Verfeinerung des Gitters einschließlich Reassemblierung des Gleichungssystems und Interpolation der Lösung auf das feinere Gitter. Weiter mit Punkt 2.

Auf Techniken im Zusammenhang mit Assemblierung und Reassemblierung wird in Abschnitt 2.3.2 eingegangen. Ausgangspunkt für die Verfeinerung ist neben einem bereits existierenden Gitter eine Liste mit markierten Zellen. Der Prozeß der Verfeinerung umfaßt globale wie lokale Aspekte.

Globaler Verfeinerungsalgorithmus

Würde man die Verfeinerung auf die markierten Zellen beschränken, so wäre die Situation in Abbildung 2.4 möglich. Dargestellt ist ein Viereckgitter nach zwei Verfeinerungsschritten. Dabei wird ein Kind einer Grobgitterzelle weiter verfeinert. Das Resultat ist ein abrupter Übergang der Größe benachbarter Zellen. Zudem ist nicht ausgeschlossen, daß sich der angedeutete Prozeß fortsetzt. Um ein Mindestmaß an Glattheit des Gitters zu gewährleisten, orientiert sich der Verfeinerungsalgorithmus an der

Irreguläre-Ecken-Regel: *Verfeinere jede noch nicht verfeinerte Zelle, für die eine ihrer Seiten mehr als* 1 *irreguläre Ecke enthält.*

Eine Ecke wird als *irregulär* bezeichnet, wenn es eine angrenzende, nicht weiter verfeinerte Zelle gibt, bezüglich der sie kein Eckpunkt ist. Zur Verdeutlichung betrachte man wiederum Abbildung 2.4. Dort sind 2 irreguläre Ecken besonders hervorgehoben. Hinzu kommt die

$k_e - 1$–**Nachbar-Regel:** *Verfeinere eine Zelle, wenn* $k_e - 1$ *Nachbarn dieser Zelle bereits regulär verfeinert sind.*

Dabei ist k_e die Zahl der Seiten einer Zelle. Als *Nachbar einer Zelle* bezüglich einer Seite e ist die kleinste Zelle mit einer Seite, die e enthält, bezeichnet.

In Abbildung 2.4 ist also das Grobgitterelement 1 Nachbar der Zelle 6. Umgekehrt ist Zelle 6 jedoch nicht Nachbar von Zelle 1. Deren einziger Nachbar ist die Grobgitterzelle 0. Die *Irreguläre-Ecken-Regel* angewandt auf dieses Beispiel stellt also sicher, daß Zelle 1 verfeinert wird. Auf den Begriff der *regulären Verfeinerung* wird im nächsten Abschnitt eingegangen, vgl. Abbildung 2.9.

Sei mit $\mathcal{L}$ die Liste der markierten Zellen bezeichnet, dann lautet die globale Verfeinerungsstrategie:

Solange $\mathcal{L} \neq \emptyset$
  Sei τ das erste Element aus $\mathcal{L}$.
  Falls τ nicht regulär verfeinert ist
    Für $j = 1, \ldots, k_e$:
      Sei τ_j der Nachbar der Zelle τ bezüglich der Seite j.
      Falls τ_j nicht regulär verfeinert ist
        REFINE(τ_j), falls τ_j $k_e - 1$ oder k_e regulär verfeinerte Nachbarn hat.
        Ansonsten REFINE(τ_j), falls Level(τ) > Level(τ_j) + 1.
    Falls τ markiert ist, REFINE(τ).
  Entferne τ aus $\mathcal{L}$.

Dieser Algorithmus stellt sicher, daß die Leveldifferenz zweier benachbarter Zellen nicht größer als 1 wird, was zur *Irreguläre-Ecken-Regel* äquivalent ist. Darüber hinaus wird jede Zelle mit mindestens $k_e - 1$ Nachbarn ebenfalls regulär verfeinert. Die Verfeinerung einer Zelle in REFINE(τ) vollzieht sich in 2 Schritten:

- Verfeinere τ regulär.
- Hänge die Kinderzellen an $\mathcal{L}$ an.

Der Algorithmus in dieser Form produziert irreguläre Ecken. Während dies bei quadrilateralen bzw. hexalateralen Netzen in 2D bzw. 3D im allgemeinen in Kauf genommen wird, werden im Falle simplizialer Triangulationen, also Dreiecks- bzw. Tetraedernetzen, im Anschluß an obigen Algorithmus irreguläre Verfeinerungsschritte, vgl. Abbildung 2.10, vorgenommen. Dies geschieht, um alle irregulären Ecken zu eliminieren. Die Kandidaten für irreguläre Verfeinerung werden während der regulären Phase in einer zweiten Liste gesammelt.

Lokaler Verfeinerungsalgorithmus

Reguläre oder *rote* Verfeinerung eines Dreiecks bedeutet Zerlegung in vier kongruente Teildreiecke durch Verbinden der Kantenmittelpunkte, vgl. Abbildung 2.9.

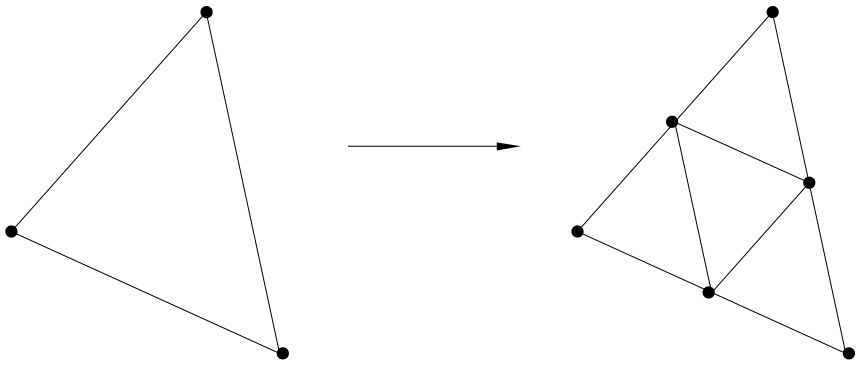

Abb. 2.9: Reguläre oder rote Verfeinerung eines Dreiecks

Da die vier neuen Dreiecke dem Ausgangsdreieck ähnlich sind, bleiben Innenwinkel erhalten. Reguläre Verfeinerung eines Dreiecks, die in der virtuellen Funktion PlaneTriangle::refineRegularly() implementiert ist, vollzieht sich im objektorientierten Kontext in drei Schritten:

- Jede der drei Makrokanten wird aufgefordert, sich selbst zu verfeinern. Dabei entstehen sechs neue Kanten und drei neue Ecken.
- Das Ausgangsdreieck erzeugt Objekte für die drei inneren Kanten.
- Das Makrodreieck instantiiert vier Mikrodreiecke.

Dieser Algorithmus knüpft an die Bemerkungen im Anschluß an die Deklaration der Klasse SpaceEdge in Abschnitt 2.2.2 an. Das Ausgangsdreieck kümmert sich im Detail nur um die Tätigkeiten, die es unmittelbar betreffen. Ansonsten übernimmt es Koordinations- und Überwachungsaufgaben. Was auf unteren Ebenen erledigt werden kann, wie etwa die Verfeinerung einer Kante, wird delegiert.

Der Vorteil dieser Vorgehensweise liegt auf der Hand: Der Code zur Verfeinerung eines Dreiecks bindet lediglich die Funktionalität *Verfeinerung einer Kante* ein, ohne von der konkreten Implementierung abhängig zu sein. Wird nun diese Implementierung modifiziert, so bleibt dies auf die Elementfunktion refineRegularly() ohne Einfluß. Objektorientierte Paradigmen tragen also dazu bei, globale Abhängigkeiten zu minimieren.

Irreguläre oder *grüne* Verfeinerung eines Dreiecks bedeutet Bisektion, vgl. Abbildung 2.10.

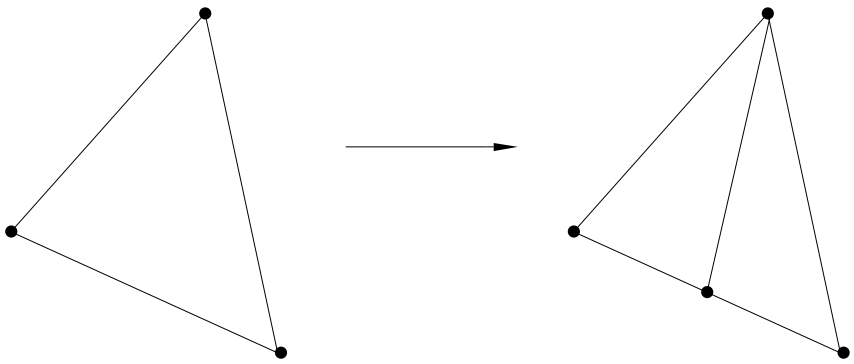

Abb. 2.10: Irreguläre oder grüne Verfeinerung eines Dreiecks

Sie dient ausschließlich dazu, eine irreguläre Ecke, die durch reguläre Verfeinerung einer Nachbarzelle entstanden ist, zu eliminieren. Im Gegensatz zu regulärer Verfeinerung wirkt sich Bisektion auf Innenwinkel negativ aus. Deshalb werden grün verfeinerte Dreiecke nicht weiter verfeinert. Sollte der globale Algorithmus dies fordern, wird zunächst die grüne Verfeinerung rückgängig gemacht und anschließend eine rote durchgeführt. Im Gegensatz zur roten Verfeinerung vollzieht sich der *grüne Abschluß* in zwei Schritten:

- Das Ausgangsdreieck erzeugt ein Objekt für die zu installierende innere Kante.
- Das Makrodreieck instantiiert zwei Mikrodreiecke.

Lediglich eine der drei Grobgitterkanten wird in diesem Zusammenhang verfeinert. Dies wiederum ist bereits bei der regulären Verfeinerung der entsprechenden Nachbarzelle geschehen.

Bei Verwendung quadrilateraler Netze beschränkt man sich im allgemeinen auf reguläre Verfeinerungsschritte. Dabei werden Vierecke durch Verbinden gegenüberliegender Kantenmittelpunkte in vier Teilvierecke zerlegt. Algorithmisch gestaltet sich dies analog zu Dreiecken. Das gilt auch für hexalaterale dreidimensionale Netze, wenn man Kanten durch Seiten ersetzt. Daß Seiten wiederum Aufgaben an Kanten delegieren, ist naheliegend. Insgesamt entstehen demnach bei der Verfeinerung einer hexalateralen Zelle acht Kinderzellen.

Dies gilt auch für die Verfeinerung von Tetraedernetzen. Erwähnenswert dabei ist, daß, was die grüne Verfeinerung von Tetraedern angeht, eine Fallunterscheidung vorgenommen werden muß. Details dazu sind z.B. bei Bey [9] nachzulesen.

2.3 Numerik

Während bisher geometrische und topologische Aspekte im Mittelpunkt standen, soll jetzt die Frage nach geeigneten numerischen Strukturen erörtert werden. In diesem Zusammenhang tritt der Zielkonflikt zwischen Flexibilität und Effizienz am offensichtlichsten zutage.

Werden beispielsweise Netze lokal verfeinert, so hat dies auch auf das algebraische Gleichungssystem lediglich lokale Auswirkungen. Deshalb sind Datenstrukturen wünschenswert, die lokale Reassemblierungsstrategien unterstützen. Auf der anderen Seite entscheidet die Effizienz bei der Lösung der Gleichungssysteme letztlich über die Laufzeit des Programms. Im Rahmen numerischer Software handelt es sich hierbei um die 20% Code, die 80% der Rechenzeit in Anspruch nehmen.

Eine Datenstruktur, die beiden Anforderungen gleichzeitig genügt, ist schwer vorstellbar. Naheliegend ist deshalb, zwei verschiedene zu verwenden: Das *Knoten*konzept, das in erster Linie mit Zeigern arbeitet und die angestrebte Flexibilität zur Verfügung stellt, einerseits und die üblichen Konzepte der linearen Algebra, *Matrizen* und *Vektoren*, die mit fixen Numerierungen arbeiten und damit die notwendige Effizienz garantieren, andererseits.

2.3.1 Knoten

Das Knotenkonzept soll folgenden Anforderungen Rechnung tragen:

1. Numerische Datenstruktur, die den flexiblen und dynamischen Multilevelstrukturen entspricht.
2. Verwaltung von linearen Operatoren und Vektoren, wie etwa Lösungsvektoren, rechten Seiten oder Diagonalvorkonditionierern.
3. Unabhängigkeit von der speziellen Lokalisierung der Freiheitsgrade.
4. Verwendbarkeit für skalare sowie Systeme von Differentialgleichungen.
5. Anwendbarkeit lokaler Assemblierungsroutinen, vgl. Abschnitt 2.3.2.
6. Anwendung als black box Baustein.

Eine naheliegende Strategie könnte darin bestehen, Netzkomponenten mit numerischer Funktionalität zu versehen. Für eine Diskretisierung wie in Kapitel 4 mit stetigen, stückweise linearen Finiten Elementen — auch konforme P_1-Diskretisierung genannt — würde das z.B. heißen, daß in jedem Objekt der Klasse Vertex Lösungswerte, rechte Seiten und Kopplungen zu Nachbarecken abgespeichert würden, vgl. Abbildung 2.11.

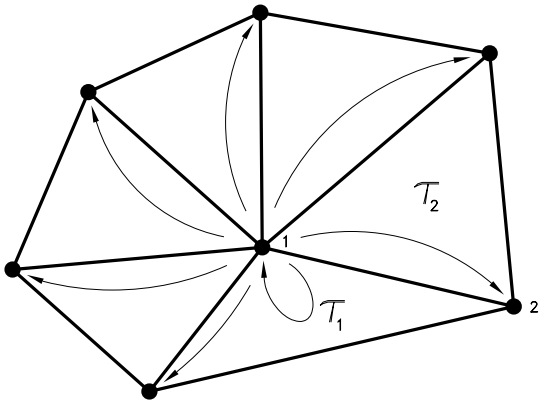

Abb. 2.11: Freiheitsgrade in den Ecken. Kopplungen sind wie in Abb. 2.12 durch Pfeile angedeutet.

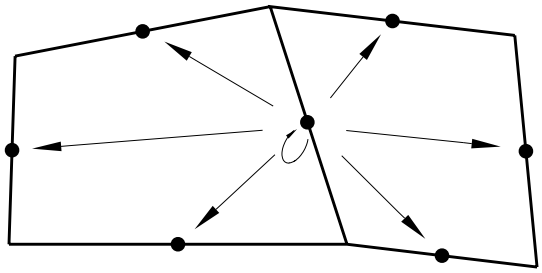

Abb. 2.12: Freiheitsgrade in den Kantenmittelpunkten

Wenn nun aber, wie bei der Diskretisierung in Kapitel 5, die Freiheitsgrade mit den Kanten assoziiert sind, vgl. Abbildung 2.12, ist die Datenstruktur unbrauchbar. Um darüber hinaus dem Grundsatz der Modularität gerecht zu werden, bietet sich eine vom Netz unabhängige Datenstruktur an.

Die Grundstruktur ist dabei die des Knotens, vgl. Abbildung 2.13.

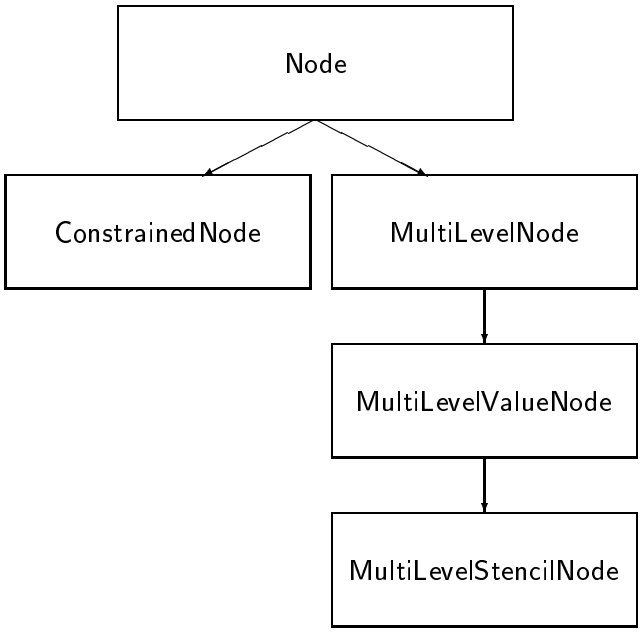

Abb. 2.13: Klassen für Knoten

Erzeugt werden diese Knoten von den zugrundeliegenden Netzkomponenten. Dafür steht die virtuelle Funktion NetComponent::createNode() zur Verfügung. Eine Referenz auf die jeweilige Netzkomponente ist in der Klasse Node niedergelegt. Jedem Knoten ist genau eine Lösungsvariable zugeordnet. Für ein Differentialgleichungssystem bedeutet das, daß eine Netzkomponente mit mehreren Knoten assoziiert sein kann.

Ein Knoten kann nun als ConstrainedNode verwendet werden, um Dirichletrandwerte zu verwalten, oder als MultiLevelNode. Letztere tragen der Tatsache Rechnung, daß eine Netzkomponente zu mehr als einem Level gehören kann. Nehmen wir als Beispiel nochmals die konforme P_1-Situation: Wird eine Zelle verfeinert, vgl. Abbildungen 2.9, 2.10, so bleiben die Ecken der Makrozelle als Feingitterecken erhalten. Sie sind aber im Gegensatz zum Grobgitter nun mit den auf den Kantenmittelpunkten neu entstandenen Ecken gekoppelt. Die geänderte Topologie wirkt sich auch auf die rechte Seite aus. Deshalb umfaßt ein MultiLevelNode mehrere Objekte der Klasse SingleLevelNode.

Um Knoten, die ausschließlich Werte verwalten, von solchen zu unterscheiden, die zusätzlich lineare Operatoren lokal darstellen, gibt es die Klassen MultiLevelValueNode sowie MultiLevelStencilNode. Analog dazu sind auch die Knoten auf den einzelnen Levels organisiert: Von SingleLevelNode ist die Klasse SingleLevelValueNode und davon wiederum SingleLevelStencilNode abgeleitet.

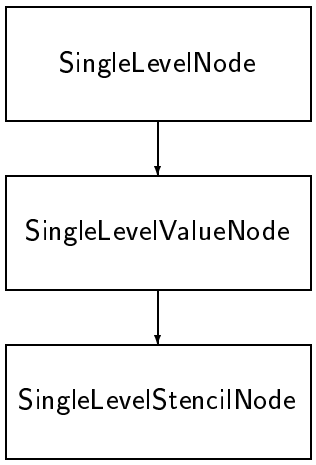

Abb. 2.14: Klassen für Knoten auf einzelnen Levels

Jeder SingleLevelStencilNode kann mehrere Objekte der Klasse Stencil enthalten. Mit Hilfe dieser Stencils oder Differenzensterne gelingt es, lineare Operatoren lokal darzustellen. Unabhängig davon, ob Finite Differenzen, Finite Volumina oder Finite Elemente zur Diskretisierung herangezogen werden, müssen zu diesem Zweck Kopplungen zu Nachbarknoten abgespeichert werden, vgl. Abbildungen 2.11, 2.12. Pro Kopplung müssen

- der Nachbarknoten und
- das Gewicht mit eben diesem Nachbarknoten

spezifiziert werden. Im Gegensatz zu strukturierten Gittern, bei denen die Nachbarrelationen klar sind und nicht explizit abgespeichert werden müssen, spricht man hier von *indirekter Adressierung*. Die Klasse Stencil ist folgendermaßen deklariert:

```
// ***************************************************************************
class Stencil
// ***************************************************************************
{
        // ******* Internal Representation *****************
private:
        Map<const Node*,MatrixEntry> connections;

        // ******* Public Interface ************************
public:
        unsigned getActualSize() const;
        const MatrixEntry& accessValue(const Node*) const;
        const Pair<const Node*,MatrixEntry>& accessPair(unsigned) const;
        MatrixEntry& supplyValue(const Node*);
        void removeCurrent();
};
```

Eine für unstrukturierte Gitter geeignete Datenstruktur ist das *assoziative Array* , auch *Map* oder *Dictionary* genannt: Während die Indizierung von Feldern üblicherweise durch Numerierung mit natürlichen Zahlen geschieht, wird die Zuordnung hier mittels Zeiger auf Nachbarknoten `const Node*` bewerkstelligt, vgl. JOSUTTIS [38]. Die Typbezeichnung `MatrixEntry` wird als Synonym für *float* bei einfacher bzw. *double* bei doppelter Genauigkeit verwendet und stellt das Gewicht der Kopplung dar. Bei Differentialgleichungssystemen kann `MatrixEntry` auch für kleine Teilmatrizen stehen, aus denen die Gesamtmatrix aufgebaut ist.

Die ersten drei der aufgelisteten Elementfunktionen modifizieren den Differenzenstern nicht und sind deshalb als `const` deklariert. Die Funktion `getActualSize()` stellt Information über die tatsächliche Zahl an Kopplungen zur Verfügung. Mit `accessValue()` bzw. `accessPair()` kann man auf vorhandene Kopplungen zugreifen. Diese Zugriffsmethoden finden beispielsweise bei der Multiplikation einer Knotenmatrix mit einem Vektor Anwendung, vgl. Abschnitt 2.4.1.

Im Gegensatz dazu dienen die beiden letzten Funktionen zur Manipulation des Differenzensterns: In `supplyValue()` wird, falls die spezifizierte Kopplung noch nicht existiert, eine neue erzeugt. Mit `removeCurrent()` wird die aktuelle Kopplung eliminiert.

2.3.2 Assemblierung

Beim Umgang mit unstrukturierten Gittern erweist sich folgende Strategie als nützlich:

- Lokalisierung der Probleme
- Lösung auf Elementebene
- Zusammensetzung der Teillösungen zu einer Gesamtlösung

Insbesondere im Zusammenhang mit der Erstellung der Gleichungssysteme hat sich diese Technik in Finite Elemente Codes bewährt, vgl. z.B. VAN KAN und SEGAL [76]. Das Prinzip dabei

ist, Integrale additiv aufzuspalten. Dies führt auf *lokale Steifigkeitsmatrizen*. Anschließend wird die *globale Steifigkeitsmatrix* dadurch erstellt, daß entsprechende Einträge der lokalen Matrizen aufsummiert werden.

In Abbildung 2.11 beispielsweise entsteht die Kopplung des Knotens 1 mit dem Knoten 2 dadurch, daß der entsprechende Eintrag der Steifigkeitsmatrix für die Zelle τ_1 zu dem für die Zelle τ_2 addiert wird. Dieser Vorgang wird im Rahmen der Finite Elemente Terminologie als *Assemblierung* bezeichnet.

In CONSIST spielen dabei folgende Funktionen eine Rolle: Durch Überladen der virtuellen Methode Cell::setNodeComponents() werden Zeiger auf die Netzkomponenten, die Knoten tragen, im *statischen* Feld ntcomp der Klasse Cell abgespeichert. Durch die Anordnung im Feld ist gleichzeitig eine lokale Numerierung der Knoten gegeben. Dieser Numerierung entsprechend werden lokale Steifigkeitsmatrizen berechnet.

Dies geschieht in von Cell abgeleiteten Klassen. Es handelt sich dabei um Funktionen der Form const DenseMatrix& fillA(const Data&) const, die unter Verwendung von Problemdaten eine Matrix zurückliefern. Im Falle der konformen P_1-Diskretisierung in Kapitel 4 wäre dies, den drei Ecken eines Dreiecks entsprechend, eine 3×3-Matrix.

Im Rahmen der Assemblierung wird die Information aus den lokalen Steifigkeitsmatrizen mit Hilfe der in setNodeComponents() gespeicherten Zeiger auf die entsprechenden Differenzensterne verteilt. Dabei wird der Differenzenstern bei jedem Eintrag daraufhin untersucht, ob die betreffende Kopplung bereits existiert. Ist dies der Fall, so wird der Eintrag addiert. Ansonsten wird in Stencil::supplyNode() eine neue Kopplung erzeugt.

Damit fillA() oder im Falle von mehr als einer Variablen weitere Funktionen dieser Art innerhalb der Assemblierungsroutine überhaupt aufgerufen werden, muß eine weitere virtuelle Funktion Cell::setLocAssembleFcts() redefiniert werden. Die Behandlung der rechten Seite(n) und eventueller zusätzlicher Vektoren, wie etwa Diagonalvorkonditionierer oder mit Hilfe von *lumping*-Techniken diagonalisierter Massenmatrizen, vgl. Abschnitt 5.3, erfolgt analog.

Diese Assemblierungstechniken sind im Zusammenhang mit unstrukturierten Gittern Standard. Bei lokal verfeinerten Gittern kommt nun ein weiterer Aspekt hinzu: Beschränkt sich die Verfeinerung auf einen kleinen Teilbereich des Gesamtgebietes, so hat dies auf Differenzensterne außerhalb dieses *aktiven Bereichs* keinerlei Einfluß. Erstrebenswert ist deshalb eine lokale Reassemblierungsstrategie. In CONSIST ist dieser Prozeß direkt in die Verfeinerung einer Zelle integriert.

Als Beispiel sei wieder die konforme P_1-Diskretisierung basierend auf einer Dreieckstriangulation herangezogen. Zum Zwecke der Reassemblierung werden unmittelbar im Anschluß an das, was auf S. 16 unter „Lokaler Verfeinerungsalgorithmus“ beschrieben ist, folgende Funktionen aufgerufen:

```
assembler->setPatch(children);
assembler->allocateNodes();
if (!greenRefined) assembler->resemble();
assembler->assemble();
```

Bei assembler handelt es sich dabei um einen Zeiger auf ein Objekt der Klasse Assembler, die für die Koordination aller Assemblierungsvorgänge zuständig ist. Im ersten Schritt muß das Assemblerobjekt mit den gerade erzeugten Kinderzellen assoziiert werden. Anschließend werden Knoten alloziert. Nehmen wir an, die Vaterzelle sei eine Level l Zelle. Die Kinder sind dann allesamt Level

$l+1$ Zellen. Für einen inneren Knoten erfolgt die Knotenallozierung an einer Ecke nach folgendem Schema:

- Ist kein MultiLevelNode vorhanden, so erzeuge einen solchen einschließlich eines SingleLevelNodes auf Level $l+1$.
- Ist ein SingleLevelNode auf Level $l+1$ nicht vorhanden, so erzeuge einen solchen durch Kopie des SingleLevelNodes auf Level l.

Wie in Abschnitt 2.2.3 bereits erwähnt, werden Dreiecke, die durch grüne Verfeinerung entstanden sind, nicht weiter verfeinert. Dem muß auch die Reassemblierung Rechnung tragen. Deshalb muß vor der roten Verfeinerung eines Dreiecks eine eventuell vorhandene grüne rückgängig gemacht werden. Zu diesem Zweck wird vor dem eben aufgelisteten Quellcode der folgende abgearbeitet:

```
if (triangle->isRefinedGreen())
{
   if (assembler)
   {
     assembler->setPatch(triangle->getChildren());
     assembler->dissemble();
   }
   triangle->undoGreenRefinement();
   greenRefined = True;
}
```

Ist ein Dreieck also grün verfeinert, so wird die Routine dissemble() aufgerufen, die die Assemblierung der Kinderdreiecke rückgängig macht. Im Gegensatz zur Assemblierung werden dabei die Einträge der lokalen Steifigkeitsmatrix subtrahiert. Anschließend werden die beiden Kinderzellen entfernt und die Variable greenRefined auf True gesetzt.

Der Wert dieser Variable entscheidet darüber, ob die Funktion resemble() aufgerufen wird. War das Dreieck nicht grün verfeinert, so muß die Assemblierung der Vaterzelle bezüglich der allozierten Knoten rückgängig gemacht werden. Während resemble() also auf die Vaterzelle angewendet wird, bezieht sich dissemble() auf die durch grüne Verfeinerung entstandenen Dreiecke.

Kopplungen, deren Gewichte im Rahmen dieser Manipulationen (im numerischen Sinne) verschwinden, werden aus dem jeweiligen Differenzenstern eliminiert. Der abschließende Aufruf der Funktion assemble() entspricht der üblichen lokalen Assemblierung.

2.4 Lineare Algebra

Wie bereits angedeutet, dient das Knotenkonzept dazu, maximale Flexibilität bei der Assemblierung und Speicherung der Gleichungssysteme zur Verfügung zu stellen. Im Hinblick auf deren Lösung sind effizientere Datenstrukturen sinnvoll. Zu diesem Zweck bietet es sich an, auf Konzepte der linearen Algebra zurückzugreifen.

2.4.1 Matrizen und Vektoren

Um die in den Knoten gespeicherten Werte bzw. Differenzensterne als Vektoren bzw. Matrizen darstellen zu können, ist eine Numerierung der Knoten notwendig. Deshalb ist jeder SingleLevelNode mit zwei Integer-Variablen ausgestattet, um neben einer stageID auch eine levelID zur Identifikation auf der feinsten Stufe bzw. dem entsprechenden Level abzuspeichern. Die Numerierung erfolgt in der Funktion LinearEquationSystem::extract(), die aufgerufen wird, wenn das Gleichungssystem im Konstruktor der Klasse LinearEquationSystem erstellt wird.

Vektoren

Vektoren dienen nicht nur dazu, rechte Seiten oder Lösungen zu repräsentieren, sondern darüber hinaus beispielsweise zur Darstellung von Diagonalmatrizen. Dazu zählen u.a. die Matrizen, die sich bei der Diskretisierung von Coriolistermen, vgl. Abschnitt 5.3, ergeben. In der Klasse Vector wird weitreichende Funktionalität zur Verfügung gestellt, die sich beim Umgang mit Vektoren als nützlich erweist. Neben den üblichen Vektorraumoperationen, wie Addition zweier Vektoren oder Multiplikation eines Vektors mit einem Skalar, sei ausdrücklich auf die Methode

```
Vector& Vector::axpy(real alpha,const Vector& x,const Vector& y);
```

verwiesen. Das Kürzel axpy entstammt den BLAS-Routinen, vgl. Huckle [36], und bedeutet, daß das α-fache des Vektors x zum Vektor y addiert und das Resultat im aufrufenden Vektorobjekt abgespeichert und zurückgegeben wird. Viele Standardlöser, wie etwa das CG-Verfahren, vgl. Abschnitt 2.4.2, greifen, was Vektoroperationen angeht, neben Skalarprodukten im wesentlichen auf diese Funktion zurück.

Matrizen

Matrizen, die aus der Diskretisierung partieller Differentialoperatoren mittels Finiter Elemente, Finiter Volumina oder Finiter Differenzen resultieren, sind im allgemeinen schwach besetzt, d.h. die Zahl der von Null verschiedenen ist verglichen mit den verschwindenden Einträgen gering.

Bei der Finite Elemente Diskretisierung des dreidimensionalen Euler Problems nach Rannacher und Turek, vgl. Abschnitt 5.2.1, stehen Geschwindigkeitsknoten im Inneren des Gebietes im allgemeinen mit zehn anderen in Kopplung. Den Zentraleintrag mitgerechnet ergeben sich unabhängig von der Feinheit des Netzes für die assoziierte Zeile der Matrix elf nichtverschwindende Einträge. Es gibt nun vielfältige Möglichkeiten, die von Null verschiedenen Einträge kompakt abzuspeichern, vgl. z.B. Huckle [36].

Die in CONSIST zum gegenwärtigen Zeitpunkt verwendeten Formate sind von der abstrakten Basisklasse

```
// ****************************************************************************
class Matrix
// ****************************************************************************
{
```

```
        // ******** Public Interface ****************************
public:
        virtual ~Matrix();
        virtual void multiply(const Vector& x,Vector& y) const = 0; // y = Ax
        virtual void multiplyAdd(const Vector& x,Vector& y) const = 0; // y = y + Ax
        virtual void multiplyTranspose(const Vector& x,Vector& y) const = 0; // y = A^T x
        virtual void multiplyTransposeAdd(const Vector& x,Vector& y) const = 0; // y = y + A^T x
};
```

abgeleitet. Diese Klasse stellt eine Schnittstelle für die wesentlichen Operationen zur Verfügung. Abgeleitete Klassen müssen diese *rein virtuellen* Funktionen redefinieren. Von fundamentaler Bedeutung sind dabei die beiden Formate NodeMatrix und CRSMatrix.

Knotenmatrizen

Die Klasse NodeMatrix definiert ein Format, das auf Differenzensternen basiert. Der Vorteil ist, daß kein wesentlicher zusätzlicher Speicheraufwand entsteht. Das Prinzip ist, Zeiger auf die numerierten Knoten in einem Feld nodes abzuspeichern:

```
// ***************************************************************************
class NodeMatrix : public Matrix
// ***************************************************************************
{
        // ******** Internal Representation *********************
private:
        unsigned nodetype;
        unsigned stencilIndex;
        Array<SingleLevelStencilNode>* nodes;

        // ******** Public Interface ****************************
public:
        void multiply(const Vector& x,Vector& y) const;
        void multiplyAdd(const Vector& x,Vector& y) const;
        void multiplyTranspose(const Vector& x,Vector& y) const;
        void multiplyTransposeAdd(const Vector& x,Vector& y) const;
};
```

Die privaten Datenelemente nodetype und stencilIndex legen bei Differentialgleichungssystemen die Variablen fest, die durch diese Matrix gekoppelt sind. In Kapitel 5 sind den Variablen u, v, w und p die Zahlen $0, 1, 2$ und 3 zugeordnet. Dementsprechend sind im Falle der Matrix, die p und u koppelt, nodetype bzw. stencilIndex gegeben durch 3 bzw. 0.

Die redefinierten Methoden multiply(), multiplyAdd(), multiplyTranspose() und multiplyTransposeAdd() nutzen die in den SingleLevelStencilNodes hinterlegte Numerierung und formulieren die Matrix-Vektor-Multiplikation direkt mit Hilfe von Differenzensternen. Zur Verdeutlichung ist die Implementierung der Funktion multiply() aufgelistet:

```
void NodeMatrix::multiply(const Vector& x,Vector& y) const
{
    const SingleLevelStencilNode* slnode;
    for (slnode = nodes->first(); slnode; slnode = nodes->next())
    {
        int id_row = slnode->getStageID();
        const Stencil& stencil = slnode->accessStencil(stencilIndex);
        y[id_row] = 0.0;
        for (register unsigned i=0; i<stencil.getActualSize(); i++)
        {
            const Pair<const Node*,MatrixEntry> & pair = stencil.accessPair(i);
            const Node* colnode = pair.getKey();
            if (colnode->isUnconstrained())
            {
                const SingleLevelNode* slnode = ((const MultiLevelNode*)colnode)->
                                                    accessNodeOnFinestStage();
                int id_col = slnode->getStageID();
                y[id_row] += pair.getValue()*x[id_col];
            }
        }
    }
}
```

Die for-Schleife läuft über alle Knoten. Im Feld nodes lassen sich im übrigen beliebige SingleLevelStencilNodes organisieren, so daß diese Struktur sowohl im Stufen- als auch im Levelkontext verwendbar ist. Hier aufgelistet ist der Quellcode für die feinste Stufe.

Der mittels stencilIndex adressierte Differenzenstern stencil entspricht einer Zeile der Matrix. Dieses *assoziative Array*, vgl. S. 21, setzt sich aus *Paar*en bestehend aus *key* und *value* zusammen. Der *Schlüssel* ist dabei jeweils die Adresse eines Nachbarknotens const Node* und der *Wert* das zugehörige Gewicht vom Typ MatrixEntry.

Das Array wird elementweise durchlaufen, wobei Dirichletrandwerte übersprungen werden. Mit Hilfe der in den Knoten gespeicherten stageID läßt sich die jeweilige Komponente des Argument- bzw. Zielvektors x bzw. y adressieren.

Matrizen im CRS-Format

Das CRS-Format (**C**ompressed **R**ow **S**torage) verzichtet gänzlich auf Zeiger und basiert stattdessen ausschließlich auf Numerierung. Insofern handelt es sich um ein FORTRAN-nahes Format.

```
// ****************************************************************************
class CRSMatrix : public Matrix
// ****************************************************************************
{
    // ******** Internal Representation *********************
```

```
private:
        unsigned nodetype;
        unsigned stencilIndex;
        unsigned NumRows;
        int* IA;
        int* JA;
        Vector A;

        // ******** Public Interface ***************************
public:
        void multiply(const Vector& x,Vector& y) const;
        void multiplyAdd(const Vector& x,Vector& y) const;
        void multiplyTranspose(const Vector& x,Vector& y) const;
        void multiplyTransposeAdd(const Vector& x,Vector& y) const;
};
```

Neben den Datenelementen nodetype und stencilIndex, vgl. Knotenmatrizen, sowie Information über die Zahl der Zeilen NumRows umfaßt die Klasse CRSMatrix zwei Integerfelder, IA und JA, sowie den Vektor A. Die von Null verschiedenen Einträge der Matrix sind zeilenweise in A gespeichert. Für einen Index $0 \leq i \leq \mathsf{NumRows} - 1$ liefert IA$[i]$ den Index, bei dem die Zeile i in A beginnt. Bei quadratischen Matrizen ist der erste Eintrag der Zeile i, A[IA$[i]$], gerade der Zentraleintrag. Das Feld JA, dessen Länge mit der von A übereinstimmt, enthält die Spaltenindizes der entsprechenden Einträge in A, vgl. z.B. RUGE und STÜBEN [59].

Die Multiplikation läßt sich mit Hilfe von Zeigerarithmetik effizient formulieren:

```
void CRSMatrix::multiply(const Vector& x,Vector& y) const
{
    register unsigned i,j;
    int* IAptr = IA;
    int* JAptr = JA;
    for (i=0; i<NumRows; i++)
    {
        y[i] = 0.0;
        for (j=*(IAptr++); j<*IAptr; j++,JAptr++)
        {
            if (*JAptr>=0) y[i] += A[j]*x[*JAptr];
        }
    }
}
```

Dirichletrandwerte werden negativ indiziert und spielen aufgrund der Abfrage if (*JAptr>=0) keine Rolle. Die aufgelistete Funktion verwendet ausschließlich Konstrukte der Standard-C-Programmierung: Felder und effiziente Methoden, um diese zu durchlaufen. Dies entspricht dem in [21] formulierten Grundsatz, objektorientierte Techniken zur Organisation einzusetzen, sich aber in laufzeitkritischen Bereichen der Software auf *low-level C code* zu beschränken.

Der Nachteil der CRS-Matrizen liegt im zusätzlichen Speicheraufwand: Während Knotenmatrizen die in Differenzensternen niedergelegte Information schlicht neu organisieren, wird bei CRS-Matrizen die Matrix einschließlich ihrer Besetzungsstruktur neu abgespeichert. Insbesondere bei nichtlinearen Problemen, bei denen ohnehin in jedem nichtlinearen Schritt eine Reassemblierung notwendig ist, kann es Sinn machen, auf Differenzensterne zu verzichten und direkt CRS-Matrizen zu assemblieren.

2.4.2 Lineare Löser

Dieser Abschnitt eignet sich besonders, um die Vorteile objektorientierter Programmierung zu verdeutlichen. Gemeint sind *Wiederverwend-* und *Erweiterbarkeit* von Software, die daraus resultieren, daß lineare Löser unabhängig von speziellen Matrixformaten formuliert werden.

Lineare Löser sind von der *abstrakten Basisklasse* LinSolver abgeleitet.

```
// ************************************************************************
class LinSolver
// ************************************************************************
{
        // ******* Public Interface **********************
public:
        virtual ~LinSolver();
        virtual void solve(const Matrix& A,Vector& x,const Vector& b) = 0;
};
```

LinSolver stellt neben einem *virtuellen Destruktor* eine Schnittstelle für die Lösung linearer Gleichungssysteme zur Verfügung. Das Gleichungssystem wird in Form einer Koeffizientenmatrix A, eines Lösungsvektors x und einer rechten Seite b übergeben. Um auch Systeme anderer Struktur subsumieren zu können, bietet es sich an, weitere Schnittstellen hinzuzufügen. Zur Lösung von Sattelpunktproblemen der Form (5.26) beispielsweise eignet sich folgende *rein virtuelle* Methode:

```
virtual void solve(const Matrix& A,const Matrix& B,Vector& x,Vector& y,
                   const Vector& f,const Vector& g) = 0;
```

Von LinSolver abgeleitet ist die Klasse IterativeSolver.

```
// ************************************************************************
class IterativeSolver : public LinSolver
// ************************************************************************
{
        // ******* Internal Representation ***************
protected:
        unsigned MaxNumIterations;
        unsigned NumIterations;
        real tolerance;
        real relTolerance;
```

```
        real convergenceRate;
        real finalResidual;

        // ******* Public Interface **********************
public:
        ...
};
```

Diese Klasse dient dazu, Charakteristika iterativer Löser zu extrahieren. Dazu zählen neben der maximal zulässigen Zahl an Iterationsschritten MaxNumIterations und der tatsächlichen NumIterations die Abbruchkriterien, ein absolutes tolerance und ein relatives relTolerance, sowie die Konvergenzrate convergenceRate als Maß für das Konvergenzverhalten des iterativen Lösers und das Residuum nach dem letzten Iterationsschritt finalResidual.

Von IterativeSolver kann man nun eine ganze Reihe von Standardlösern, wie Jacobi, Gauß-Seidel, ILU oder SIP ableiten. Die Klassendeklarationen unterscheiden sich nicht wesentlich von der des Verfahrens der konjugierten Gradienten, vgl. Abschnitt 4.5.1:

```
// ****************************************************************************
class CGSolver: public IterativeSolver
// ****************************************************************************
{
        // ******** Internal Representation *****************
protected:
        Preconditioner* preconditioner;

        // ******** Public Interface ************************
public:
        void solve(const Matrix& A,Vector& x,const Vector& b);
};
```

Die Verwendung der *abstrakten Basisklasse* LinSolver und Redefinition der virtuellen Methode solve() macht es möglich, lineare Löser gegeneinander auszutauschen. Besonders deutlich werden die Vorteile objektorientierter Programmierung, wenn man einen Blick auf die Implementierung des CG-Lösers wirft:

```
void CGSolver::solve(const Matrix& A,Vector& x,const Vector& b)
{
    const unsigned length = x.dim(); // Länge des Vektors x
    Vector r(length); // Hilfsvektor
    Vector q(length); // Hilfsvektor
    Vector p(length); // Hilfsvektor
    real initialResidual,residual,relativeTolerance;
    const real eps = 1.0e-14;
    real rho,rho_old = 1.0;
    real alpha = 1.0;
    real beta = 1.0;
```

```
    real tpq,convergenceRate = 1.0;
    A.multiply(x,r); // r = Ax
    r.axpy(-1.0,r,b); // r = b - r
    initialResidual = residual = r.norm(); // euklidische Norm von r
    relativeTolerance = initialResidual*relTolerance;
    for (NumIterations = 0; NumIterations<MaxNumIterations; NumIterations++)
    {
        if (residual<tolerance || residual<relativeTolerance)
        {
            finalResidual = residual;
            if (NumIterations) convergenceRate = pow(residual/initialResidual,1.0/NumIterations);
            return;
        }
        if (preconditioner)
        {
            preconditioner->apply(q,r); // optionale Anwendung eines Vorkonditionierers
        }
        else q = r; // Zuweisung
        rho = q*r; // Skalarprodukt ρ = (q, r)
        if (!NumIterations) p = q;
        else
        {
            beta = rho/rho_old;
            p.axpy(beta,p,q); // p = q + βp
        }
        A.multiply(p,q); // q = Ap
        tpq = p*q; // tpq = (p, q)
        if (fabs(tpq)<tolerance*eps)
        {
            finalResidual = residual;
            if (NumIterations) convergenceRate = pow(residual/initialResidual,1.0/NumIterations);
            return;
        }
        alpha = rho/tpq;
        x.axpy(alpha,p,x); // x = αp + x
        r.axpy(-alpha,q,r); // r = r - αq
        residual = r.norm(); // euklidische Norm von r
        rho_old = rho;
    }
    finalResidual = residual;
    if (NumIterations) convergenceRate = pow(residual/initialResidual,1.0/NumIterations);
}
```

Von Vektoroperationen abgesehen, wird lediglich auf Matrix-Vektor-Multiplikationen zurückgegriffen. Entscheidend dabei ist, daß der CG-Löser vom speziellen Format, in dem die Matrix A abgespeichert ist, vgl. Abschnitt 2.4.1, nicht abhängt. Bei FORTRAN-Programmierung müßte für jedes Matrixformat eine eigene CG-Version implementiert werden.

Ein Anwender kann nun eine Reihe weiterer Matrixformate generieren, ohne an der Implementierung des CG-Verfahrens irgendwelche Änderungen vornehmen zu müssen. Er muß lediglich für das von ihm favorisierte Format die Funktionalität *Multiplikation mit einem Vektor* spezifizieren und kann dann den CG-Löser als getesteten *black box* Baustein einbinden. Das Zusammenspiel von *Vererbung* und *Polymorphie* sorgt also dafür, daß Formatfragen zugunsten von Funktionalitätsaspekten an Bedeutung verlieren.

Was bei der Betrachtung des Quellcodes weiterhin auffällt, ist die kompakte Formulierung. In der Tat unterscheidet sich die Implementierung von der Formulierung des Verfahrens in der Literatur, vgl. Kommentare am Zeilenende, nicht wesentlich. In objektorientierter Terminologie ist von Verkleinerung der *semantischen Lücke* die Rede.

Den entsprechenden Programmierstil vorausgesetzt, kann die Lesbarkeit des Quellcodes selbstverständlich auch im konventionellen Kontext gewahrt werden. Im Gegensatz zu FORTRAN etwa war und ist die Verständlichkeit des Codes jedoch ein wesentlicher Aspekt bei der Entwicklung von C++. Analog dazu können auch andere objektorientierte Paradigmen mit Hilfe prozeduraler Sprachen imitiert werden. In Bezug auf den wesentlichen Aspekt jedoch gilt: **Prozedurale Programmierung läßt Abstraktion zu, objektorientierte unterstützt sie**.

Kapitel 3

Strömungstechnische Grundlagen

In diesem Kapitel sollen die Gleichungen, die die stationäre Strömung eines inkompressiblen, reibungsfreien Fluids in einer Strömungsmaschine beschreiben, abgeleitet werden. Dies geschieht ausgehend von den gewöhnlichen Euler Gleichungen durch Übergang zum Relativsystem.

3.1 Die Euler Gleichungen im rotierenden System

Betrachtet werden zunächst die Grundgleichungen instationärer, reibungsfreier, inkompressibler Strömung in kartesischen Koordinaten:

$$\frac{\partial \mathbf{c}}{\partial t} + (\mathbf{c} \cdot \nabla)\mathbf{c} + \nabla p/\rho = 0 \qquad \text{in } \Omega_t \tag{3.1}$$

$$\operatorname{div} \mathbf{c} = 0 \qquad \text{in } \Omega_t \tag{3.2}$$

$\Omega_t \subset \mathbb{R}^3$ sei dabei ein beschränktes Gebiet, das im Eulerschen Sinne mit Fluid gefüllt ist. Im Hinblick auf Strömungsmaschinen handelt es sich um einen Schaufelkanal, der sich mit konstanter Winkelgeschwindigkeit um die z-Achse dreht. Auf geeignete Randbedingungen wird in Kapitel 5 eingegangen.

Für $\mathbf{x} \in \Omega_t$ sind im Blick auf die nachfolgende Herleitung folgende Definitionen wesentlich:

$$\tilde{\mathbf{x}} := Q(t)^T \mathbf{x}(t) \tag{3.3}$$

$$\tilde{\mathbf{c}}(\tilde{\mathbf{x}}) := Q(t)^T \mathbf{c}(\mathbf{x}(t), t) \tag{3.4}$$

$$\tilde{p}(\tilde{\mathbf{x}}) := p(\mathbf{x}(t), t) \tag{3.5}$$

$Q(t)$ ist dabei eine orthogonale Transformation, die die zeitabhängige Drehung des Schaufelkanals beschreibt. $Q(t)^T = Q(t)^{-1}$ leistet demzufolge die Rückdrehung des Kanals, so daß $\tilde{\mathbf{x}}$ zeitunabhängige kartesische Koordinaten im Relativsystem sind. Die neuen Absolutgeschwindigkeitsvektoren $\tilde{\mathbf{c}}$ an den transformierten Punkten $\tilde{\mathbf{x}}$ entstehen durch Rückdrehung der ursprünglichen Absolutgeschwindigkeiten $\mathbf{c}$. Außerdem wird angenommen, daß die Strömung im Relativsystem stationär ist.

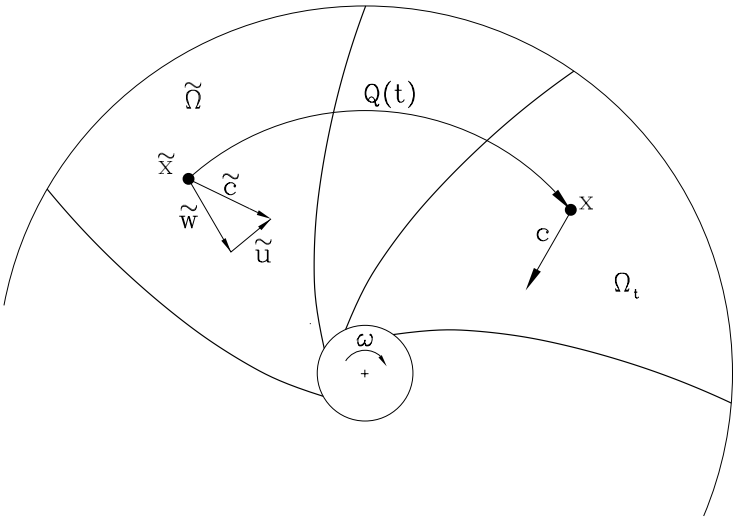

Abb. 3.1: Transformation ins Relativsystem

Der Beschleunigungsterm transformiert sich dann wie folgt:

$$\begin{aligned} Q(t)^T \frac{d\mathbf{c}}{dt}(\mathbf{x}(t),t) &= Q(t)^T \frac{d}{dt}\left(Q(t)\tilde{\mathbf{c}}(Q(t)^T\mathbf{x}(t))\right) \\ &= Q(t)^T\dot{Q}(t)\tilde{\mathbf{c}}(Q(t)^T\mathbf{x}(t)) + Q(t)^TQ(t)D\tilde{\mathbf{c}}(Q(t)^T\mathbf{x}(t)) \\ &\quad \left(\dot{Q}(t)^T\mathbf{x}(t) + Q(t)^T\dot{\mathbf{x}}(t)\right) \end{aligned} \tag{3.6}$$

Wegen $0 = \dot{I} = [Q(t)^TQ(t)\dot{]} = \dot{Q(t)}^TQ(t) + Q(t)^T\dot{Q}(t)$ oder

$$Q(t)^T\dot{Q}(t) = -\dot{Q(t)}^TQ(t)$$

ist die Matrix $Q(t)^T\dot{Q}(t)$ schiefsymmetrisch und über das Kreuzprodukt mit dem Winkelgeschwindigkeitsvektor

$$\boldsymbol{\omega} := (0,0,\omega)^T$$

assoziiert, d.h.

$$Q(t)^T\dot{Q}(t)\tilde{\mathbf{c}}(Q(t)^T\mathbf{x}(t)) = \boldsymbol{\omega}\times\tilde{\mathbf{c}}(\tilde{\mathbf{x}})$$

Mit $Q(t)^TQ(t) = I$, $\dot{\mathbf{x}}(t) = \mathbf{c}(\mathbf{x}(t),t)$ sowie den Definitionen (3.3)–(3.5) ergibt sich aus (3.6):

$$Q(t)^T \frac{d\mathbf{c}}{dt}(\mathbf{x}(t),t) \;=\; \boldsymbol{\omega} \times \tilde{\mathbf{c}}(\tilde{\mathbf{x}}) + D\tilde{\mathbf{c}}(\tilde{\mathbf{x}})\,(-\boldsymbol{\omega} \times \tilde{\mathbf{x}} + \tilde{\mathbf{c}}(\tilde{\mathbf{x}}))$$

Definiert man die Relativgeschwindigkeit wie üblich als

$$\tilde{\mathbf{w}}(\tilde{\mathbf{x}}) \;:=\; \tilde{\mathbf{c}}(\tilde{\mathbf{x}}) - \boldsymbol{\omega} \times \tilde{\mathbf{x}}$$

und beachtet, daß

$$Q(t)^T \nabla p(\mathbf{x}(t),t) \;=\; Q(t)^T Q(t) \nabla \tilde{p}(\tilde{\mathbf{x}})$$

ergibt sich aus (3.1) die folgende Formulierung der Impulsgleichungen mit Absolutgeschwindigkeiten im Relativsystem, auf die letztlich auch die numerische Lösung aufsetzt:

$$(\mathbf{w} \cdot \nabla)\mathbf{c} + \boldsymbol{\omega} \times \mathbf{c} + \nabla p/\rho \;=\; 0 \qquad \text{in } \Omega \tag{3.7}$$

Auf die Tilden wurde der Übersichtlichkeit halber verzichtet. Physikalisch bedeutet dies, daß die Änderung des Absolutgeschwindigkeitsfeldes entlang der Stromlinien des Relativgeschwindigkeitsfeldes durch den Druckgradienten und den Corioliseinfluß bestimmt wird.

Unter Verwendung der Identität $\mathrm{Spur}(AB) = \mathrm{Spur}(BA)$ für Matrizen A und B geeigneter Größe ergibt sich

$$\begin{aligned}
\operatorname{div} \mathbf{c}(\mathbf{x}(t),t) &= \mathrm{Spur}\,(D\mathbf{c}(\mathbf{x}(t),t)) \\
&= \mathrm{Spur}\,\left(D\left(Q(t)\tilde{\mathbf{c}}(Q(t)^T\mathbf{x}(t))\right)\right) \\
&= \mathrm{Spur}\,\left(Q(t)D\tilde{\mathbf{c}}(\tilde{\mathbf{x}})Q(t)^T\right) \\
&= \mathrm{Spur}\,(D\tilde{\mathbf{c}}(\tilde{\mathbf{x}})) \\
&= \operatorname{div} \tilde{\mathbf{c}}(\tilde{\mathbf{x}})
\end{aligned}$$

so daß die Kontinuitätsgleichung (3.2) nach Änderung der Bezeichnung formal unverändert bleibt:

$$\operatorname{div} \mathbf{c} \;=\; 0 \qquad \text{in } \Omega \tag{3.8}$$

3.2 Die Hauptgleichungen der Turbinentheorie

Aus der Formulierung (3.7), (3.8) leitet sich nun eine Reihe von Folgerungen ab. In der Literatur ist es üblich, von der Turbinenhauptgleichung zu sprechen. Alle Gleichungen dieses Abschnitts gelten jedoch für Pumpen in gleicher Weise.

3.2.1 Die Eulersche Turbinenhauptgleichung

Aus $\mathbf{c} = \mathbf{w} + \boldsymbol{\omega} \times \mathbf{x}$ folgt

$$\begin{aligned}(\mathbf{w} \cdot \nabla)\mathbf{c} &= D\mathbf{c} \cdot \mathbf{w} \\ &= D\mathbf{w} \cdot \mathbf{w} + \boldsymbol{\omega} \times \mathbf{w}\end{aligned}$$

Eingesetzt in (3.7) ergibt sich

$$(\mathbf{w} \cdot \nabla)\mathbf{w} + 2\boldsymbol{\omega} \times \mathbf{w} + \boldsymbol{\omega} \times (\boldsymbol{\omega} \times \mathbf{x}) + \nabla p/\rho = 0 \qquad \text{in } \Omega \tag{3.9}$$

und wegen $\operatorname{div}(\boldsymbol{\omega} \times \mathbf{x}) = 0$

$$\operatorname{div} \mathbf{w} = 0 \qquad \text{in } \Omega \tag{3.10}$$

Der Fliehkraftterm $\boldsymbol{\omega} \times (\boldsymbol{\omega} \times \mathbf{x})$ als Potentialkraftfeld läßt sich als Gradient der skalarwertigen Funktion $-\frac{1}{2}\omega^2 r^2$ darstellen, wobei r der euklidische Abstand des Ortsvektors $\mathbf{x} = (x, y, z)^T$ von der Drehachse ist, d.h. $r^2 = x^2 + y^2$. Mit der Führungsgeschwindigkeit $u := r\omega$ heißt das

$$\nabla\left(-\frac{1}{2}u^2\right) = \boldsymbol{\omega} \times (\boldsymbol{\omega} \times \mathbf{x})$$

Der konvektive Term in (3.9) läßt sich folgendermaßen zerlegen, vgl. BRONSTEIN [16]:

$$(\mathbf{w} \cdot \nabla)\mathbf{w} = \nabla(\frac{1}{2}w^2) - \mathbf{w} \times \operatorname{rot}\mathbf{w}$$

wobei der zweite Term genauso wie der Coriolisterm in (3.9) keine Leistung erbringt.

Multipliziert man nun (3.9) mit dem Relativgeschwindigkeitsfeld $\mathbf{w}$ und integriert über ein beliebiges Kontrollvolumen $V \subseteq \Omega$ mit Rand ∂V, so ergibt sich als lokale Leistungsbilanz

$$\int_V \nabla\left(\frac{p}{\rho} - \frac{1}{2}u^2 + \frac{1}{2}w^2\right) \cdot \mathbf{w}\, d\mathbf{x} = 0 \tag{3.11}$$

Wie man leicht durch Nachrechnen bestätigt, gilt für ein differenzierbares Vektorfeld $\mathbf{v}$ und eine differenzierbare skalarwertige Funktion f die Identität:

$$\operatorname{div}(f\mathbf{v}) = \nabla f \cdot \mathbf{v} + f \operatorname{div}\mathbf{v} \tag{3.12}$$

Wenn man nun über V integriert und anschließend den Integralsatz von Gauß anwendet, ergibt sich mit dem äußeren Normaleneinheitsvektor $\mathbf{n}$ die wichtige **Greensche Formel**:

$$\int_V f \mathrm{div}\mathbf{v}\, d\mathbf{x} \;=\; -\int_V \nabla f \cdot \mathbf{v}\, d\mathbf{x} + \int_{\partial V} f\mathbf{v}\cdot\mathbf{n}\, d\Gamma \tag{3.13}$$

Im Eindimensionalen stellt (3.13) gerade die Formel für partielle Integration dar. Speziell für $f := \frac{p}{\rho} - \frac{1}{2}u^2 + \frac{1}{2}w^2$ und $\mathbf{v} := \mathbf{w}$ folgt aus (3.11) wegen (3.10)

$$\int_{\partial V} \left(\frac{p}{\rho} - \frac{1}{2}u^2 + \frac{1}{2}w^2\right)(\mathbf{w}\cdot\mathbf{n})\; d\Gamma \;=\; 0$$

Berücksichtigt man noch den Kosinussatz

$$\frac{1}{2}w^2 = \frac{1}{2}c^2 + \frac{1}{2}u^2 - uc_u$$

wobei c_u die azimutale Komponente von $\mathbf{c}$ in Zylinderkoordinaten ist, folgt letztlich mit $p_t := p + \frac{1}{2}\rho c^2$

$$\int_{\partial V} (p_t - \rho u c_u)(\mathbf{w}\cdot\mathbf{n})\; d\Gamma \;=\; 0 \tag{3.14}$$

Diese Bilanz gilt für jedes Kontrollvolumen, insbesondere für den gesamten Schaufelkanal. Wegen der Randbedingungen (5.3)–(5.7) tragen in diesem Fall lediglich Ein- und Ausströmrand zum Integral in (3.14) bei. Mit Hilfe der Mittelwertbildung

$$\Delta\overline{f} \;:=\; \frac{\int_{\partial\Omega} f\, dQ}{Q} \tag{3.15}$$

wobei Q den Volumenstrom und dQ das Differential $\mathbf{w}\cdot\mathbf{n}\; d\Gamma$ bezeichnet, folgt die **Eulersche Turbinenhauptgleichung**:

$$\Delta\overline{p_t} \;=\; \rho\Delta\overline{uc_u} \tag{3.16}$$

Für $\omega = 0$ reduziert sich (3.16) auf die Bernoulli-Gleichung $\Delta\overline{p_t} = 0$.

3.2.2 Die Drehimpulsbilanz

Das Ziel dieses Abschnitts ist es, eine Beziehung zwischen der Umlenkung $\Delta\overline{rc_u}$ und dem Schaufelmoment herzuleiten. Als Ausgangspunkt dient die Impulsbilanz (3.7). Unter Zuhilfenahme der Kontinuität (3.10) erhält man mit der Verallgemeinerung

$$(\mathbf{w}\cdot\nabla)\mathbf{c} \;=\; \mathrm{div}(\mathbf{c}\cdot\mathbf{w}^T) - \mathrm{div}(\mathbf{w})\mathbf{c} \tag{3.17}$$

von Formel (3.12) als äquivalente Formulierung

$$\mathrm{div}\left(\mathbf{c}\cdot\mathbf{w}^T + \frac{p}{\rho}\cdot I\right) + \boldsymbol{\omega}\times\mathbf{c} \;=\; 0 \tag{3.18}$$

Die Anwendung des Divergenzoperators auf eine Matrix ist dabei wie üblich zeilenweise zu verstehen. Interpretiert man den Coriolisterm als Quellterm, so stellt (3.18) eine Formulierung in Erhaltungsform dar. Um zur Drehimpulsbilanz zu gelangen, wird diese Kräftebilanz mit dem Ortsvektor $\mathbf{x}$ im Sinne des Vektorprodukts multipliziert. Anschließende Integration bezüglich eines beliebigen Kontrollvolumens $V \subseteq \Omega$ mit Rand ∂V liefert:

$$\int_V \mathbf{x}\times\mathrm{div}\left(\mathbf{c}\cdot\mathbf{w}^T + \frac{p}{\rho}\cdot I\right) + \mathbf{x}\times(\boldsymbol{\omega}\times\mathbf{c})\,d\mathbf{x} \;=\; 0$$

Weil die Matrix $\left(\mathbf{c}\cdot\mathbf{w}^T + \frac{p}{\rho}\cdot I\right)$ nicht symmetrisch ist, liefert die Anwendung des Integralsatzes von Gauß, vgl. TRUESDELL [73], S. 144:

$$\int_{\partial V}\mathbf{x}\times\left(\mathbf{c}\cdot\mathbf{w}^T + \frac{p}{\rho}\cdot I\right)\cdot\mathbf{n}\,d\Gamma + \int_V \mathbf{x}\times(\boldsymbol{\omega}\times\mathbf{c}) - (\mathbf{x}\times\boldsymbol{\omega})\times\mathbf{c}\,d\mathbf{x} \;=\; 0 \tag{3.19}$$

Mit Hilfe der Grassmann-Identität $a\times(b\times c) = (a\cdot c)b - (a\cdot b)c$ ergibt sich die Jacobi-Identität:

$$\mathbf{x}\times(\boldsymbol{\omega}\times\mathbf{c}) - (\mathbf{x}\times\boldsymbol{\omega})\times\mathbf{c} \;=\; \boldsymbol{\omega}\times(\mathbf{x}\times\mathbf{c})$$

d.h. das Volumenintegral in (3.19) ist orthogonal zur z-Achse. Beachtet man weiterhin, daß für einen beliebigen Vektor $\mathbf{v}$ die z-Komponente von $\mathbf{x}\times\mathbf{v}$ gegeben ist durch rv_u mit der Azimutalkomponente v_u von $\mathbf{v}$, so ergibt sich als Drehimpulsbilanz in z-Richtung:

$$\int_{\partial V}\rho r c_u(\mathbf{w}\cdot\mathbf{n})\,d\Gamma + \int_{\partial V} p r n_u\,d\Gamma \;=\; 0 \tag{3.20}$$

Wegen der Randbedingungen (5.3)–(5.7) tragen für $V=\Omega$ zum ersten Integral ausschließlich Ein- und Ausströmrand bei. Der zweite Term stellt gerade das negative Schaufelmoment $-M_{Schaufel}$ in z-Richtung dar. Mit den Bezeichnungen des vorhergehenden Abschnittes gilt also:

$$\rho\Delta\overline{rc_u} \;=\; M_{Schaufel}/Q \tag{3.21}$$

Für unbeschaufelte Räume folgt daraus die Gleichung des Potentialwirbels: $\overline{rc_u} = \text{const.}$. Mit Hilfe eines Referenzradius r_{ref} und der Referenzgeschwindigkeit $u_{ref} := \omega r_{ref}$ lassen sich mit dem Staudruck $\frac{\rho}{2}u_{ref}^2$ für $\omega \neq 0$ dimensionslose Druckzahlen definieren:

$$\begin{aligned} \psi_t &:= \frac{\Delta\overline{p}_t}{\frac{\rho}{2}u_{ref}^2} \\ \psi_{th} &:= \frac{\Delta\overline{uc_u}}{\frac{1}{2}u_{ref}^2} \\ \psi_M &:= \frac{M_{Schaufel}\omega}{\frac{\dot{m}}{2}u_{ref}^2} \end{aligned}$$

Mit $\dot{m} := Q\rho$ ist dabei der Massenstrom bezeichnet. Mit diesen Definitionen folgen aus (3.16) in Verbindung mit (3.21) für reibungsfreie Strömung die Beziehungen

$$\psi := \psi_t = \psi_{th} = \psi_M \tag{3.22}$$

Diese identischen Größen stellen ein Maß für die Energieumsetzung in einer hydraulischen Strömungsmaschine dar. Bei einer Turbine beispielsweise quantifiziert ψ_t die auf den Volumenstrom bezogene Leistung, die dem Fluid entzogen, ψ_M hingegen diejenige, die an der Welle abgenommen wird. Beide Werte sind gemäß (3.22) mit der kinematischen Größe ψ_{th} identisch.

Kapitel 4

Die Euler Gleichungen auf Rotationsflächen

Um zu einem zweidimensionalen Modell zu gelangen, wird vorausgesetzt, daß die Strömung auf vorgegebenen Rotationsflächen verläuft. Nimmt man zusätzlich Drehungsfreiheit des Absolutgeschwindigkeitsfeldes an und eliminiert die Kontinuitätsgleichung mit Hilfe von Stromfunktionstechniken, so resultiert ein lineares elliptisches Randwertproblem zweiter Ordnung. Näheres zur Herleitung ist bei SCHILLING [62] oder FEISTAUER [22] nachzulesen.

4.1 Lineares elliptisches Randwertproblem zweiter Ordnung

Mit der Meridiankoordinate m und der Azimutalkoordinate φ lautet das Problem für die Stromfunktion ψ in differentieller Form:

$$\frac{\partial}{\partial m}\left(\frac{r}{\delta}\frac{\partial \psi}{\partial m}\right)+\frac{\partial}{\partial \varphi}\left(\frac{1}{r\delta}\frac{\partial \psi}{\partial \varphi}\right) \;=\; 2\omega r\frac{dr}{dm} \qquad \text{in } \Omega \tag{4.1}$$

oder in kompakter Divergenzschreibweise

$$-\mathrm{div}(a\nabla\psi) \;=\; f \qquad \text{in } \Omega \tag{4.2}$$

mit

$$a := \begin{pmatrix} r/\delta & 0 \\ 0 & 1/(r\delta) \end{pmatrix} \qquad \text{und} \qquad f := -2\omega r\frac{dr}{dm} \tag{4.3}$$

Dabei handelt es sich bei $r(m)$ und $\delta(m)$ um gegebene geometrische Größen. ω ist die Winkelgeschwindigkeit des Laufrades. Ein Kontrollraum Ω ist in Abbildung 4.1 schematisch skizziert.

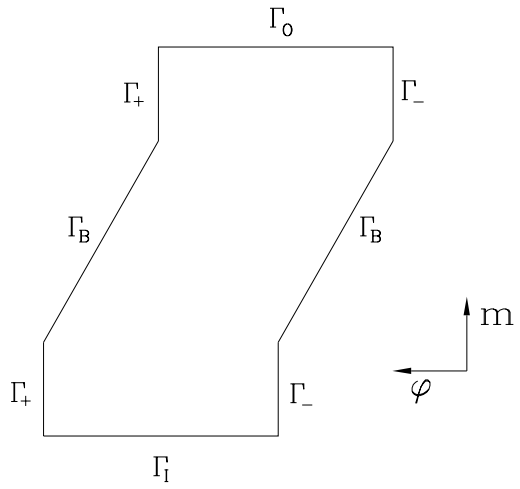

Abb. 4.1: Kontrollraum und Randbedingungen

Neben dem $m-\varphi$-Koordinatensystem werden im folgenden konforme $l-u$-Koordinaten von Bedeutung sein. Letztere haben den Vorzug, Winkel in der realen dreidimensionalen Konfiguration zu reproduzieren. Mit einem Referenzradius r_{ref} gilt zwischen den Koordinaten der Zusammenhang:

$$l := r_{ref} \int \frac{dm}{r(m)} \quad \text{und} \quad u := r_{ref}\varphi \tag{4.4}$$

Die Komponenten der Relativgeschwindigkeit in $m-\varphi$ sind mit der Stromfunktion ψ in folgender Weise korreliert, vgl. SCHILLING [62]:

$$w_m = \frac{1}{r\delta}\frac{\partial\psi}{\partial\varphi} \tag{4.5}$$

$$w_\varphi = -\frac{1}{\delta}\frac{\partial\psi}{\partial m} \tag{4.6}$$

Hinzu kommen die Randbedingungen:

$$\psi = \psi_B \quad \text{auf} \quad \Gamma_B \tag{4.7}$$

$$a\nabla\psi\cdot\mathbf{n} = rw_\varphi^{in} \quad \text{auf} \quad \Gamma_I \tag{4.8}$$

$$a\nabla\psi\cdot\mathbf{n} = -rw_\varphi^{out} \quad \text{auf} \quad \Gamma_O \tag{4.9}$$

$$\psi_+ - \psi_- = Q \quad \text{auf} \quad \Gamma_- \tag{4.10}$$

$$(a\nabla\psi\cdot\mathbf{n})_+ + (a\nabla\psi\cdot\mathbf{n})_- = 0 \quad \text{auf} \quad \Gamma_- \tag{4.11}$$

wobei mit Γ_B die Schaufeloberfläche und mit Γ_I bzw. Γ_O Ein- bzw. Ausströmrand bezeichnet sind. $\mathbf{n}$ ist der äußere Normaleneinheitsvektor und ψ_B eine gegebene Funktion, die von der Definition des Kontrollraums abhängt. Bei der in Abbildung 4.1 angedeuteten Konfiguration ist ψ_B

auf Druck- und Saugseite jeweils konstant, wobei sich die beiden Konstanten durch den Volumenstrom unterscheiden. Ist die Schaufel jedoch in der Mitte des Kontrollraums positioniert, so gilt $\psi_B = \text{const}$. In beiden Fällen ist das Niveau frei wählbar. Zur Vereinfachung der Notation sei im folgenden $\psi_B = 0$ angenommen.

Am Einströmrand ist die Normalkomponente des Flusses $a\nabla\psi$, die wegen $\mathbf{n} = \binom{-1}{0}$ auf Γ_I in Verbindung mit (4.3) und (4.6) gerade rw_φ entspricht, vorgegeben. Es ist üblich, w_φ^{in} aus c_φ^{in} mit Hilfe des Geschwindigkeitsdreiecks am Eintritt zu berechnen und als konstant anzunehmen. Unter der Annahme drallfreier Zuströmung beispielsweise ist $c_\varphi^{in} = 0$ und damit $w_\varphi^{in} = -\omega r$. Eine von φ abhängige Vorgabe ist jedoch genauso möglich. w_φ am Ausströmrand wird ebenfalls als konstant angenommen, wobei die Konstante w_φ^{out} aus der Kutta-Bedingung zu bestimmen ist.

Die Stromfunktionswerte am periodischen Rand unterscheiden sich um den Volumenstrom Q. Für eine stetige Funktion f bezeichnen $\tilde{f}_+$ bzw. f_- die Restriktion von f auf Γ_+ bzw. Γ_-, wobei Γ_+ aus Γ_- durch Verschiebung um den Teilungswinkel $2\pi/Z$ mit der Schaufelzahl Z in positive φ-Richtung hervorgeht, vgl. Abbildung 4.1. f_+ ergibt sich aus $\tilde{f}_+$ durch Verschiebung auf Γ_-, d.h.

$$f_+(m,\varphi) := \tilde{f}_+(m,\varphi + \frac{2\pi}{Z}) \quad \text{für} \quad (m,\varphi) \in \Gamma_- \tag{4.12}$$

Diese Definition gilt für die Stromfunktion genauso wie für die Normalkomponente des Flusses, deren Stetigkeit über den periodischen Rand in (4.11) postuliert wird.

4.2 Variationsformulierung

Wie in Lehrbüchern über Finite Elemente, vgl. z.B. Braess [11] oder Van Kan und Segal [76], ausführlich erläutert, wird die Differentialgleichung mit einer *geeigneten* skalarwertigen *Testfunktion* v multipliziert und über Ω integriert:

$$\int_\Omega -\text{div}\,(a\nabla\psi)\, v \; dmd\varphi \;\; = \;\; \int_\Omega f v \; dmd\varphi \tag{4.13}$$

Um den Begriff *geeignet* genauer zu spezifizieren, sind folgende Definitionen sinnvoll:

$$\tilde{\Psi} \;\; := \;\; \left\{ \psi \in H^1(\Omega) : \psi = 0 \text{ auf } \Gamma_B,\ \psi_+ - \psi_- = Q \text{ auf } \Gamma_- \right\} \tag{4.14}$$

$$\tilde{V} \;\; := \;\; \left\{ v \in H^1(\Omega) : v = 0 \text{ auf } \Gamma_B,\ v_+ = v_- \text{ auf } \Gamma_- \right\} \tag{4.15}$$

Als $H^1(\Omega)$ ist dabei der *Sobolevraum* bezeichnet, der aus dem Raum der stetig differenzierbaren Funktionen auf Ω in analoger Weise entsteht wie die Menge der reellen Zahlen aus der der rationalen Zahlen. Diese sogenannte *Vervollständigung* führt auf den Begriff der *verallgemeinerten* oder *distributionellen Ableitung*. Damit assoziiert ist eine Verallgemeinerung des Riemann-Integrals: das *Lebesgue-Integral*. Erst mit diesen Begriffsbildungen ist es möglich, eine Theorie für (partielle) Differentialgleichungen aufzustellen.

Daß sich diese zunächst rein formalen mathematischen Konstrukte in Physik und Technik durchgesetzt haben, zeigt die Bedeutung der δ-Distribution nicht nur in der Regelungstechnik, vgl. SCHILLING [61], und hat damit zu tun, daß sich Lösungen, die sich mit der klassischen Theorie nicht mehr, mit der verallgemeinerten jedoch sehr wohl beschreiben lassen, als physikalisch und technisch sinnvoll erwiesen haben.

$\tilde{\Psi}$ und $\tilde{V}$ sind nun Teilräume des H^1, in die — ebenfalls in einem verallgemeinerten Sinne — Rand- und Nebenbedingungen aufgenommen sind. Analoges gilt für den $L^2(M)$, der sich als Vervollständigung des Raums der stetigen Funktionen auf einer weitgehend beliebigen Menge M ergibt. Auf die Angabe der Normen, bezüglich derer die jeweilige Vervollständigung erfolgt, wird hier verzichtet.

Mit Hilfe dieser Räume und der Greenschen Formel (3.13) angewandt auf (4.13) mit $f := -v$ und $\mathbf{v} := a\nabla\psi$ lautet die Variationsformulierung von (4.2) in Verbindung mit den Randbedingungen (4.7)-(4.11):

Finde $\psi \in \tilde{\Psi}$, so daß

$$\int_\Omega a\nabla\psi \cdot \nabla v \; dmd\varphi \;=\; \int_\Omega fv \; dmd\varphi + \int_{\Gamma_I} rw_\varphi^{in} v \; d\Gamma - \int_{\Gamma_O} rw_\varphi^{out} v \; d\Gamma \quad \text{für alle} \quad v \in \tilde{V} \tag{4.16}$$

Eingehende Untersuchungen im Hinblick auf die eindeutige Lösbarkeit dieses Variationsproblems sind bei FEISTAUER [22] nachzulesen. Bei der Herleitung von (4.16) ist zu beachten, daß die zusätzlichen Terme

$$\int_{\Gamma_B} a\nabla\psi \cdot \mathbf{n} v \; d\Gamma + \int_{\Gamma_-} (a\nabla\psi \cdot \mathbf{n})_- v_- + (a\nabla\psi \cdot \mathbf{n})_+ v_+ \; d\Gamma \tag{4.17}$$

wegen $v = 0$ auf Γ_B bzw. $v_+ = v_-$ auf Γ_- in Verbindung mit (4.11) entfallen.

Bei dieser Formulierung des Problems gehen also alle Zwangsbedingungen in die Definition der Funktionenräume (4.14) und (4.15) ein. Eine alternative Strategie besteht darin, Nebenbedingungen mit Hilfe eines Lagrange-Parameters anzukoppeln. Führt man dies für die Periodizitätsbedingung exemplarisch durch, so ergibt sich mit dem Raum

$$V \;:=\; \left\{ v \in H^1(\Omega) : v = 0 \text{ auf } \Gamma_B \right\}$$

die gemischte Formulierung:

Finde $\psi \in V$ und $\lambda \in L^2(\Gamma_-)$, so daß

$$\begin{aligned} \int_\Omega a\nabla\psi \cdot \nabla v \; dmd\varphi + \int_{\Gamma_-} \lambda(v_+ - v_-) \; d\Gamma \;&=\; \int_\Omega fv \; dmd\varphi + \int_{\Gamma_I} rw_\varphi^{in} v \; d\Gamma \\ &\quad - \int_{\Gamma_O} rw_\varphi^{out} v \; d\Gamma \quad \text{für alle} \quad v \in V \qquad (4.18) \\ \int_{\Gamma_-} (\psi_+ - \psi_-)\mu d\Gamma \;&=\; \int_{\Gamma_-} Q\mu d\Gamma \quad \text{für alle} \quad \mu \in L^2(\Gamma_-) \qquad (4.19) \end{aligned}$$

Bemerkung 4.1 *Ein Vergleich von (4.18) mit (4.16) und (4.17) zeigt, daß der Lagrange-Parameter λ gerade der Normalkomponente des Flusses über Γ_- entspricht. Mit derselben Technik kann man auch die Dirichletbedingung $\psi = 0$ auf Γ_B ankoppeln.*

4.3 Diskrete Variationsformulierung

Im diesem Abschnitt werden Grundkenntnisse über Finite Elemente vorausgesetzt, wie sie u.a. bei Braess [11], Brenner und Scott [14], Van Kan und Segal [76] oder Schwarz [65] nachzulesen sind. Basierend auf einer simplizialen Triangulation von Ω bezeichne V_h den Raum der stetigen, stückweise linearen Funktionen, vgl. Abbildung 4.2, die auf Γ_B verschwinden, und Q_h den Raum der stetigen, stückweise linearen Funktionen auf Γ_-.

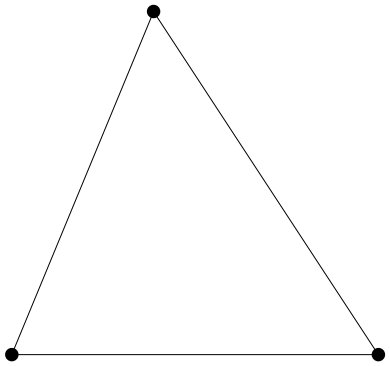

Abb. 4.2: Freiheitsgrade für die konforme P_1-Diskretisierung

Der Galerkin-Ansatz für (4.18), (4.19) lautet dann:

Finde $\psi_h \in V_h$ und $\lambda_h \in Q_h$, so daß

$$\begin{aligned} \int_\Omega a\nabla\psi_h \cdot \nabla v_h \, dmd\varphi + \int_{\Gamma_-} \lambda_h (v_{h+} - v_{h-}) \, d\Gamma &= \int_\Omega f v_h \, dmd\varphi + \int_{\Gamma_I} r w_\varphi^{in} v_h \, d\Gamma \\ &\quad - \int_{\Gamma_O} r w_\varphi^{out} v_h \, d\Gamma \quad \text{für alle } v_h \in V_h \quad (4.20) \\ \int_{\Gamma_-} (\psi_{h+} - \psi_{h-}) \mu_h \, d\Gamma &= \int_{\Gamma_-} Q \mu_h \, d\Gamma \quad \text{für alle } \mu_h \in Q_h \quad (4.21) \end{aligned}$$

oder mit Hilfe der Bilinearformen

$$a(\psi_h, v_h) := \int_\Omega a\nabla\psi_h \cdot \nabla v_h \, dmd\varphi \quad \text{für} \quad (\psi_h, v_h) \in V_h \times V_h$$

und

$$b(\psi_h, \lambda_h) := \int_{\Gamma_-} (\psi_{h+} - \psi_{h-}) \lambda_h \, d\Gamma \quad \text{für} \quad (\psi_h, \lambda_h) \in V_h \times Q_h$$

sowie der Linearformen

$$f(v_h) := \int_\Omega f v_h \, dmd\varphi + \int_{\Gamma_I} r w_\varphi^{in} v_h \, d\Gamma - \int_{\Gamma_O} r w_\varphi^{out} v_h \, d\Gamma \quad \text{für} \quad v_h \in V_h$$

und

$$g(\lambda_h) \quad := \quad \int_{\Gamma_-} Q\lambda_h \, d\Gamma \quad \text{für} \quad \lambda_h \in Q_h$$

Finde $\psi_h \in V_h$ und $\lambda_h \in Q_h$, so daß

$$a(\psi_h, v_h) + b(v_h, \lambda_h) \quad = \quad f(v_h) \quad \text{für alle} \quad v_h \in V_h \tag{4.22}$$

$$b(\psi_h, \mu_h) \quad = \quad g(\mu_h) \quad \text{für alle} \quad \mu_h \in Q_h \tag{4.23}$$

(4.22), (4.23) stellt ein Sattelpunktproblem dar, d.h. die Matrix des zugehörigen linearen Gleichungssystems ist indefinit, besitzt also positive wie negative Eigenwerte. Zu einem definiten System gelangt man, wenn es gelingt, eine Basis des Unterraums

$$\tilde{V}_h \quad := \quad \{v_h \in V_h : b(v_h, \mu_h) = 0 \quad \text{für alle} \quad \mu_h \in Q_h\} \tag{4.24}$$

und eine Partikulärlösung für (4.23) anzugeben, vgl. Brezzi und Fortin [15].

Zu diesem Zweck wird die bezüglich der Ecken der Triangulation nodale Basis von V_h mit $\{v_h^i : 1 \leq i \leq n_h\}$ bezeichnet. Die i-te Funktion v_h^i ist dabei charakterisiert durch die Maßgabe

$$v_h^i(a_j) \quad = \quad \delta_{ij}$$

wobei die a_j, $1 \leq j \leq n_h$, die Ecken der Triangulation sind. Die Basisfunktionen von V_h seien nun so numeriert, daß die ersten m_h mit den Ecken auf Γ_+ und die zweiten m_h mit denen auf Γ_- assoziiert sind. Eine Basis von $\tilde{V}_h$ ist dann gegeben durch die Funktionen

$$\{v_h^{i+m_h} + v_h^i : 1 \leq i \leq m_h\} \cup \{v_h^i : 2m_h + 1 \leq i \leq n_h\} \tag{4.25}$$

und eine Partikulärlösung von (4.23) durch

$$\psi_h^p \quad := \quad \sum_{i=1}^{m_h} Q v_h^i$$

Damit kann man nun die Periodizitätsbedingung einschließlich Lagrange-Parameter λ_h eliminieren. Übrig bleibt das folgende definite System:

Finde $\tilde{\psi}_h \in \tilde{V}_h$, so daß

$$a(\tilde{\psi}_h, \tilde{v}_h) \quad = \quad \tilde{f}(\tilde{v}_h) \quad \text{für alle} \quad \tilde{v}_h \in \tilde{V}_h \tag{4.26}$$

mit

$$\tilde{f}(\tilde{v}_h) \quad := \quad f(\tilde{v}_h) - a(\psi_h^p, \tilde{v}_h) \quad \text{für} \quad \tilde{v}_h \in \tilde{V}_h$$

4.3.1 Assemblierung des Gleichungssystems

Die Basis (4.25) mag auf den ersten Blick unhandlich erscheinen. Dies ändert sich, wenn man den Übergang von der nodalen Basis zur Basis (4.25) betrachtet. Seien zu diesem Zweck die Koeffizienten von $\tilde{v}_h \in \tilde{V}_h$ (4.25) entsprechend in einen periodischen Anteil $\tilde{\alpha}_p \in \mathbb{R}^{m_h}$ und einen inneren Anteil $\tilde{\alpha}_i \in \mathbb{R}^{n_h-2m_h}$ aufgeteilt. Stellt man $\tilde{v}_h$ bezüglich der nodalen Basis von V_h dar, so läßt sich der Koeffizientenvektor analog partitionieren: $\alpha_+ \in \mathbb{R}^{m_h}$ bzw. $\alpha_- \in \mathbb{R}^{m_h}$ seien die Anteile bezüglich der nodalen Basisfunktionen bezüglich Γ_+ bzw. Γ_-. $\alpha_i \in \mathbb{R}^{n_h-2m_h}$ bezeichne den inneren Anteil. Damit ergeben sich die Zusammenhänge:

$$\begin{aligned} \alpha_i &= \tilde{\alpha}_i \\ \alpha_+ &= \tilde{\alpha}_p \\ \alpha_- &= \tilde{\alpha}_p \end{aligned}$$

oder in Matrixschreibweise mit der k-reihigen Einheitsmatrix I_k:

$$C = \begin{pmatrix} I_{n_h-2m_h} & 0 \\ 0 & I_{m_h} \\ 0 & I_{m_h} \end{pmatrix} \tag{4.27}$$

Bezeichnet man die Matrizen, die die Bilinearform $a(.,.)$ bezüglich der nodalen Basis bzw. der Basis (4.25) darstellen, mit A bzw. $\tilde{A}$, so gilt der Zusammenhang, vgl. z.B. MEYBERG [48]:

$$\tilde{A} = C^T A C \tag{4.28}$$

Partitioniert man die Matrix A und die rechte Seite f analog zum Lösungsvektor

$$\begin{pmatrix} A_{ii} & A_{i+} & A_{i-} \\ A_{+i} & A_{++} & A_{+-} \\ A_{-i} & A_{-+} & A_{--} \end{pmatrix} \begin{pmatrix} \alpha_i \\ \alpha_+ \\ \alpha_- \end{pmatrix} = \begin{pmatrix} f_i \\ f_+ \\ f_- \end{pmatrix}$$

so ergibt sich aus (4.28) und analoger Transformation der rechten Seite mit dem Vektor $e := (1,\ldots,1)^T \in \mathbb{R}^{m_h}$

$$A_{ii}\tilde{\alpha}_i + (A_{i+} + A_{i-})\tilde{\alpha}_p = f_i - QA_{i+}e \quad (4.29)$$

$$(A_{+i} + A_{-i})\tilde{\alpha}_i + (A_{++} + A_{-+} + A_{+-} + A_{--})\tilde{\alpha}_p = f_+ + f_- - Q(A_{++} + A_{-+})e \quad (4.30)$$

Rechentechnisch bedeutet dies, daß die Matrix $\tilde{A}$ mit denselben Assemblierungsmethoden erstellt werden kann, wie dies im Finite Elemente Kontext üblich ist: Steifigkeitsmatrizen werden lokal unter der Annahme homogener Neumannscher Randbedingungen am periodischen Rand berechnet. Im Rahmen der Assemblierung wird für jedes Paar periodischer Ecken lediglich ein Freiheitsgrad gesetzt und diese periodischen Freiheitsgrade wie innere Freiheitsgrade behandelt, vgl. SEGAL ET AL. [66].

Lediglich die Q-proportionalen Terme auf der rechten Seite bedürfen besonderer Beachtung bei der Berechnung des lokalen Lastvektors. Auf die Assemblierung bleibt auch dies ohne Einfluß. Zusammenfassend kann man festhalten: Periodische Freiheitsgrade werden behandelt wie innere, Standardassemblierungsmethoden können mit minimaler Modifikation verwendet werden.

4.3.2 Folgerungen

Die Eigenschaft der Matrix a in (4.3), symmetrisch und gleichmäßig positiv definit zu sein, d.h.

$$\left(a\begin{pmatrix} m \\ \varphi \end{pmatrix}, \begin{pmatrix} m \\ \varphi \end{pmatrix}\right) = \frac{r}{\delta}m^2 + \frac{1}{r\delta}\varphi^2 \geq \alpha(m^2 + \varphi^2)$$

mit $\alpha := \min_m\{\frac{r}{\delta}, \frac{1}{r\delta}\} > 0$, überträgt sich auf die Matrix A, vgl. BRAESS [11]. Wegen (4.28) ist mit A auch $\tilde{A}$ symmetrisch und positiv definit:

$$(\tilde{A}\tilde{x}, \tilde{x}) = (C^T AC\tilde{x}, \tilde{x}) = (AC\tilde{x}, C\tilde{x}) \geq 0$$

wobei das Gleichheitszeichen genau dann gilt, wenn $C\tilde{x} = 0$. Wegen Kern $C = \{0\}$, folgt daraus $\tilde{x} = 0$ und somit die Definitheit von $\tilde{A}$.

Daß periodische Ränder keine Ränder im eigentlichen Sinne und die Freiheitsgrade darauf deshalb genauso zu behandeln sind wie innere, ist auch ohne obige Herleitung naheliegend. Interessant sind jedoch mindestens zwei Zwischenergebnisse, die sich im Hinblick auf Euler und Navier-Stokes Rechnungen verallgemeinern lassen:

- Zum einen gilt der Zusammenhang (4.28) zwischen der aus der Standarddiskretisierung resultierenden Matrix A und $\tilde{A}$, in die zusätzlich die Periodizitätsbedingungen eingearbeitet sind. In Abschnitt 5.4 wird sich zeigen, daß es bei der Diskretisierung der Euler Gleichungen Sinn macht, die faktorisierte Form (4.28) anstelle der expliziten (4.29), (4.30) zugrundezulegen.
- Zum anderen ergibt sich aus (4.20) und Bemerkung 4.1, daß der diskrete Lagrange-Parameter λ_h gerade die Normalkomponente des Flusses $a\nabla\psi$ über den periodischen Rand approximiert, die ihrerseits direkt mit der Tangentialgeschwindigkeit w_τ in konformen $l-u$-Koordinaten korreliert ist.

Der genaue Zusammenhang in Punkt 2 ergibt sich, wenn man eine Randkurve des Kontrollraums in $m - \varphi$ und deren Bild in $l - u$ betrachtet, vgl. Abbildung 4.3.

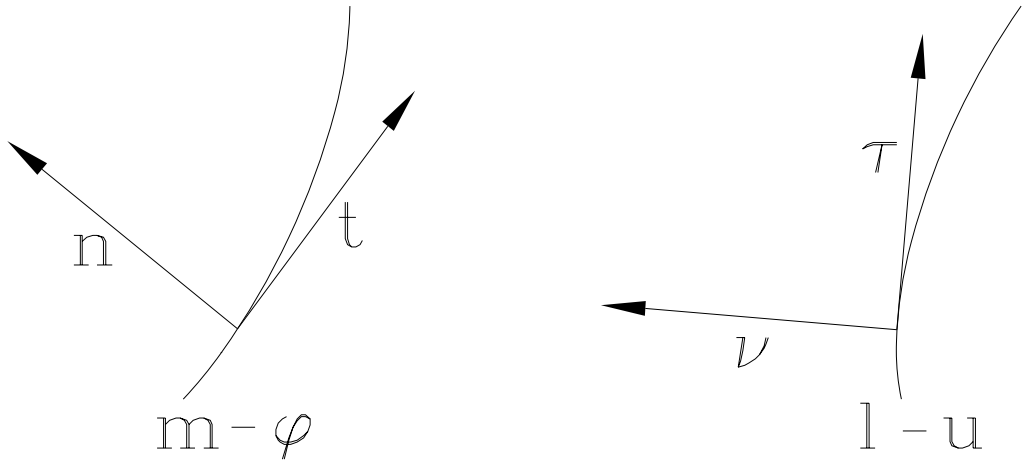

Abb. 4.3: Tangentialvektoren in $m - \varphi$ und $l - u$

Die normierten Tangentialvektoren an entsprechenden Punkten werden mit $\mathbf{t} = (t_m, t_\varphi)^T = (n_\varphi, -n_m)^T$ bzw. $\boldsymbol{\tau} = (\tau_l, \tau_u)^T$ bezeichnet. Mit der Transformation (4.4) liefert die Kettenregel den Zusammenhang zwischen den beiden Tangentialvektoren:

$$\begin{pmatrix} \tau_l \\ \tau_u \end{pmatrix} = \frac{1}{\sqrt{t_m^2 + (rt_\varphi)^2}} \begin{pmatrix} t_m \\ rt_\varphi \end{pmatrix}$$

Wegen $w_l = w_m$ und $w_u = w_\varphi$ gilt damit für w_τ:

$$\begin{aligned} w_\tau &= \begin{pmatrix} w_l \\ w_u \end{pmatrix} \cdot \begin{pmatrix} \tau_l \\ \tau_u \end{pmatrix} \\ &= \frac{w_m t_m + w_\varphi r t_\varphi}{\sqrt{t_m^2 + (rt_\varphi)^2}} \\ &= \frac{w_m n_\varphi - w_\varphi r n_m}{\sqrt{n_\varphi^2 + (rn_m)^2}} \\ &= \frac{a\nabla\psi \cdot \mathbf{n}}{\sqrt{n_\varphi^2 + (rn_m)^2}} \end{aligned} \tag{4.31}$$

Analog zu periodischen Randbedingungen verhält es sich mit der Dirichletrandbedingung $\psi = 0$ auf der Schaufeloberfläche. Üblicherweise werden Randwerte dieser Art eingearbeitet, indem die entsprechende Zeile der Matrix gelöscht und die zugehörige Spalte mit dem vorgegebenen Wert (hier 0) multipliziert auf die rechte Seite gebracht wird.

Wie in Bemerkung 4.1 bereits angedeutet, läßt sich die Behandlung von Dirichletrandbedingungen ebenfalls im Sinne einer Lagrange-Multiplikator Technik interpretieren und eine gemischte Formulierung von der Form (4.22), (4.23) ableiten. Entscheidend ist nun, daß man auf der Basis einer berechneten Stromfunktionslösung ψ_h den Lagrange-Parameter λ_h durch Lösen des Variationsproblems

$$\int_{\Gamma_B} \lambda_h \cdot v_h \, d\Gamma \quad = \quad f(v_h) - a(\psi_h, v_h) \tag{4.32}$$

ermitteln kann, wobei v_h alle die nodalen Basisfunktionen durchläuft, die mit Ecken auf der Schaufeloberfläche assoziiert sind. Dies entspricht der Diskretisierung eines eindimensionalen Problems auf der Schaufelkontur. Die Matrix, die mit (4.32) assoziiert ist, ist — bei geeigneter Numerierung der Unbekannten — eine tridiagonale Massenmatrix. Mit den üblichen *lumping*-Techniken, vgl. z.B. GROSSMANN und ROOS [29], reduziert sich die Lösung auf eine einfache Skalierung.

Wie bei Periodizitäten approximiert λ_h die Normalkomponente des Flusses und damit gemäß (4.31) die Tangentialgeschwindigkeit w_τ. Wegen $w_\nu = 0$ entlang der Schaufelkontur geht lediglich w_τ in die Berechnung des Druckkoeffizienten

$$c_p \quad = \quad 1 - \frac{w_\tau^2}{w_\infty^2} \tag{4.33}$$

ein, wobei mit w_∞ die Anströmgeschwindigkeit bezeichnet ist. Mit dieser Strategie erreicht man eine genauere Approximation des Druckes entlang der Schaufel im Vergleich zu gewöhnlicher Extrapolation, vgl. Abschnitt 6.

Die Techniken der Lagrange-Mechanik werden auch bei der Behandlung der Euler Gleichungen in Kapitel 5 eine wesentliche Rolle spielen. Eine ausführliche Diskussion der zugrundeliegenden physikalischen Prinzipien findet sich in Abschnitt 5.7.

4.4 Bestimmung der Abströmrichtung

Die Abströmrichtung w_φ^{out} und damit die Zirkulation über die Schaufel werden mit Hilfe der Kuttabedingung festgelegt. Diese besagt, daß die Abströmung an der Schaufelhinterkante tangential erfolgt. Im diskreten Kontext geht — wie ein Blick auf (4.20) zeigt — w_φ^{out} linear in die rechte Seite des Gleichungssystems ein:

$$\tilde{A}\tilde{\alpha} \quad = \quad \tilde{f} \quad \text{mit} \quad \tilde{f} = \hat{f} - \tilde{a}_{w_\varphi} w_\varphi^{out} \tag{4.34}$$

mit dem Vektor $\tilde{a}_{w_\varphi} \in \mathbb{R}^{n_h - m_h}$. Betrachtet man w_φ^{out} als zusätzliche Unbekannte, so resultiert das modifizierte Gleichungssystem:

$$\left(\tilde{A} \mid \tilde{a}_{w_\varphi}\right) \cdot \begin{pmatrix} \tilde{\alpha} \\ \\ w_\varphi^{out} \end{pmatrix} \quad = \quad \hat{f}$$

oder mit $\hat{A} := \left(\tilde{A} \mid \tilde{a}_{w_\varphi}\right)$ und $\hat{\alpha} := \left(\tilde{\alpha}, w_\varphi^{out}\right)^T$

$$\hat{A}\hat{\alpha} = \hat{f} \tag{4.35}$$

$\hat{A}$ ist eine $(n_h - m_h) \times (n_h - m_h + 1)$-Matrix vom Rang $n_h - m_h$. Der Kern hat also die Dimension 1 und die allgemeine Lösung von (4.35) lautet damit

$$\hat{\alpha}^{part} + \mathrm{Kern}\hat{A} \tag{4.36}$$

oder mit $\lambda \in \mathbb{R}$

$$\hat{\alpha}_1^{part} + \lambda\left(\hat{\alpha}_2^{part} - \hat{\alpha}_1^{part}\right)$$

Dies entspricht einer Gerade im $\mathbb{R}^{n_h - m_h + 1}$, die durch die zwei Punkte $\hat{\alpha}_1^{part}$ und $\hat{\alpha}_2^{part}$ eindeutig festgelegt ist. Das Problem ist also gelöst, wenn zwei Partikulärlösungen von (4.35) bekannt sind. Der Parameter λ läßt sich dann mit Hilfe der Kuttabedingung bestimmen.

Die beiden Partikulärlösungen erhält man, wenn man in (4.34) zwei verschiedene Werte für w_φ^{out} vorgibt und daraus jeweils $\tilde{\alpha}$ berechnet. Zusammenfassend heißt das, daß zur Erfüllung der Kuttabedingung genau zwei lineare Gleichungssysteme mit gleicher Koeffizientenmatrix und leicht modifizierter rechter Seite zu lösen sind.

Diese Strategie ist auch anwendbar, wenn der Betriebspunkt stoßfreier Anströmung gesucht ist. Zusätzlich zu w_φ^{out} tritt dann bei festgehaltener Drehzahl der Volumenstrom Q als weitere Unbekannte auf. Als allgemeine Lösung ergibt sich eine Ebene im $\mathbb{R}^{n_h - m_h + 2}$. Zu deren Bestimmung müssen drei Partikulärlösungen ermittelt werden. Eine Verallgemeinerung auf weitere lineare Nebenbedingungen ist in analoger Weise möglich.

4.5 Lösung der Gleichungssysteme

Betrachtet wird in diesem Abschnitt ein lineares Gleichungssystem $Ax = b$ mit symmetrisch, positiv definiter Matrix A wie in (4.34). Dieses Gleichungssystem resultiere aus der Diskretisierung eines elliptischen Randwertproblems 2. Ordnung mittels stückweise linearer konformer Finiter Elemente. Die zugehörige Variationsformulierung sei durch (4.26) gegeben, wobei in diesem Abschnitt auf die Tilden verzichtet wird.

Es zeigt sich, daß das Verfahren der konjugierten Gradienten zur Lösung dieses Problems geeignet ist, vorausgesetzt es stehen Vorkonditionierungsstrategien zur Verfügung stehen.

4.5.1 Das CG-Verfahren als Basislöser

Das CG-Verfahren basiert auf einer äquivalenten Formulierung des linearen Gleichungssystems

$$Ax = b \tag{4.37}$$

als Minimierungsproblem, unter der Voraussetzung, daß A symmetrisch, positiv definit ist. Das zugeordnete quadratische Funktional lautet:

$$F(x) := \frac{1}{2}(x, Ax) - (b, x) \tag{4.38}$$

Die Optimalitätsbedingung $\nabla F = 0$ stimmt mit (4.37) überein. Daß es sich bei der Lösung von (4.37) um das eindeutig bestimmte Minimum von (4.38) handelt, folgt aus der positiven Definitheit von A.

Daß sich das Verfahren der konjugierten Gradienten gegenüber dem einfachen Gradientenverfahren durchgesetzt hat, liegt nun in der Tatsche begründet, daß beim CG-Verfahren die Optimalität bezüglich vorheriger Suchrichtungen nicht verloren geht. Näheres ist u.a. bei BRAESS [11] oder HACKBUSCH [33] nachzulesen. Die Implementierung des Verfahrens in CONSIST findet sich in Abschnitt 2.4.2.

Bezeichnet man mit x_0 die Startiterierte, mit x_k die Iterierte nach dem k-ten Schritt und die exakte Lösung des Gleichungssystems mit x, so gilt die Fehlerabschätzung:

$$\|x - x_k\|_A \leq 2 \left(\frac{\sqrt{\kappa(A)} - 1}{\sqrt{\kappa(A)} + 1} \right)^k \|x - x_0\|_A \tag{4.39}$$

wobei mit der spektralen Kondition $\kappa(A)$ das Verhältnis des größten Eigenwerts der Matrix A zum kleinsten gemeint ist. Dieses Verhältnis strebt nun allerdings mit feiner werdendem Gitter gegen Unendlich. Verwendet man als Maß für die Feinheit eines Gitters den maximalen Durchmesser h aller Zellen, so gilt:

$$\kappa(A) = \mathcal{O}(h^{-2}) \tag{4.40}$$

Um ein effizientes Verfahren zu erhalten, sind Vorkonditionierungsstrategien unerläßlich. Das Prinzip dabei ist, (4.37) mit einer symmetrischen, positiv definiten Matrix C, die den beiden folgenden Anforderungen genügt, zu multiplizieren:

- Ein Gleichungssystem mit Koeffizientenmatrix C ist mit *geringem* Aufwand lösbar.
- $\kappa(C^{-1}A)$ ist von h unabhängig.

Als besonders günstig haben sich Vorkonditionierer erwiesen, die auf Mehrgitter- bzw. im lokal verfeinerten Fall auf Multilevelstrategien basieren.

4.5.2 Multilevel Vorkonditionierung

Multilevelverfahren sind Mehrgitterverfahren für den Fall lokal verfeinerter Gitter. Dabei ist die Unterscheidung der Begriffe *Level* und *Stufe* von fundamentaler Bedeutung. Die Definitionen finden sich in Abschnitt 2.2.1. Die Begriffe sind in Abbildung 4.4 illustriert.

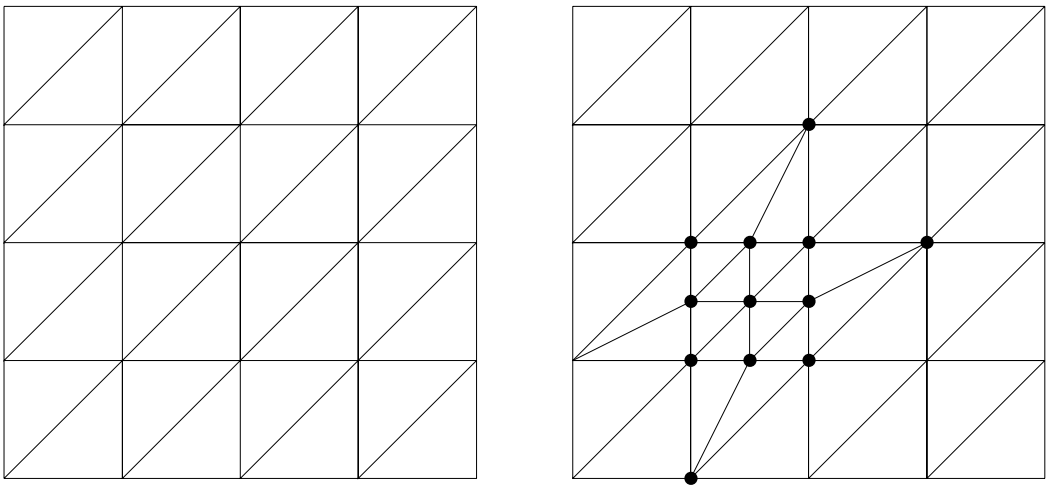

Abb. 4.4: Initiales und lokal verfeinertes Dreiecksnetz. Markiert sind die Freiheitsgrade auf Level 1.

Links ist ein initiales Dreiecksgitter auf dem Einheitsquadrat zu erkennen. Hier fallen die Begriffe Level 0 und Stufe 0 zusammen. Rechts ist dasselbe Gebiet nach einem Verfeinerungsschritt dargestellt. Dabei wurden 2 Dreiecke *rot* und 4 *grün* verfeinert. Das resultierende Gitter bildet gerade die Stufe 1. Level 1 ist lediglich eine Teilmenge von Stufe 1 und besteht aus den 16 durch Verfeinerung entstandenen Dreiecken.

Um die damit assoziierten Finite Elemente Räume zu spezifizieren, sind folgende Bezeichnungen nützlich:

$$\mathcal{N}_k = \{x_1, \ldots, x_{n_k}\}$$

bezeichne die Menge der inneren Ecken sowie Ecken auf dem Neumannrand auf Stufe $k \geq 0$ und

$$\psi^k := \{\psi_i^k : 1 \leq i \leq n_k\} \tag{4.41}$$

die mit $\mathcal{N}_k$ assoziierten nodalen Basisfunktionen auf Stufe k. Levelweise werden nun folgende Systeme von Basisfunktionen definiert:

$$\psi_{BPX}^0 = \{\psi_i^0 : x_i \in \mathcal{N}_0\} \tag{4.42}$$

und für $k \geq 1$

$$\psi_{BPX}^k = \{\psi_i^k : x_i \in \mathcal{N}_k, \text{aber } \psi_i^k \text{ ist nicht Basisfunktion auf Level } k-1\} \tag{4.43}$$

Die Bezeichnung BPX steht für **B**ramble, **P**asciak und **X**u, auf deren Arbeiten der Vorkonditionierer zurückgeht, vgl. BRAMBLE ET AL. [12]. Alternativ dazu schlägt YSERENTANT [81] vor, anstelle des Erzeugendensystems (4.42), (4.43) auf hierarchische Basen zurückzugreifen.

In obigem Beispiel umfaßt ψ^1_{BPX} gerade alle mit den markierten Ecken assoziierten Basisfunktionen, wenn man annimmt, daß auf dem unteren Rand Neumann- und auf dem linken Dirichletrandbedingungen vorgegeben sind. Die zugehörigen Räume ergeben sich als lineare Hülle der Funktionensysteme (4.41) bzw. (4.42), (4.43):

$$W_k = \operatorname{span}\{\psi^k\} \quad \text{bzw.} \quad S_k = \operatorname{span}\{\psi^k_{BPX}\}$$

Nach dem J-ten Verfeinerungsschritt ist also das (4.26) zugeordnete Stufen-Problem

Finde $\psi_J \in W_J$, so daß

$$a(\psi_J, w_J) = f(w_J) \quad \text{für alle} \quad w_J \in W_J \tag{4.44}$$

zu lösen. Im Rahmen von Multilevelverfahren wird die Lösung von (4.44) zurückgeführt auf die Lösung von Level-Problemen der Form:

Finde $\psi_k \in S_k$, so daß

$$a(\psi_k, s_k) = f(s_k) \quad \text{für alle} \quad s_k \in S_k \tag{4.45}$$

für $0 \le k \le J$. Im Zweilevel-Kontext ($J = 1$) mit Nachglättung sind Multilevelverfahren dann folgendermaßen organisiert, vgl. z.B. MCCORMICK [47]:

> **1. Schritt: Grobgitterkorrektur** Berechne das Residuum auf Stufe 1 und restringiere es auf Level 0. Löse das Level-Problem (4.45) für $k = 0$ mit dem restringierten Residuum als rechter Seite und prolongiere die Lösung auf Stufe 1. Korrigiere damit die aktuelle Lösung auf Stufe 1.
>
> **2. Schritt: Glättung** Löse das Level-Problem (4.45) für $k = 1$ approximativ mit Hilfe eines geeigneten Glättungsverfahrens.

Das Multilevelverfahren ergibt sich aus dem Zweilevelverfahren durch rekursive Fortsetzung. Im Falle eines stetigen, stückweise linearen Ansatzes verwendet man für die Prolongation üblicherweise die natürliche Inklusion und für die Restriktion den transponierten Operator.

Als Glättungsprozedur für den BPX-Vorkonditionierer ist das Jacobi-Verfahren ausreichend, um zu zeigen, daß die Konvergenzraten nicht von der Verfeinerungstiefe abhängen, und die Komplexität des Verfahrens ein optimales $\mathcal{O}(N)$-Verhalten aufweist, wobei N die Zahl der Unbekannten auf der feinsten Stufe bezeichnet. Diese und weitere theoretische Ergebnisse sowie die zugehörigen Beweise für den BPX-Vorkonditionierer sind in [50] und der darin zitierten Literatur nachzulesen.

Das Besondere an diesem Algorithmus ist die Tatsache, daß sich die Glättung in Schritt 2 lediglich auf Freiheitsgrade des Levels 1 erstreckt. Das klassische Mehrgitterverfahren angewendet auf lokal verfeinerte Gitter würde die Glättung auch auf solche Knoten auf Stufe 1 ausdehnen, die nicht zu Level 1 gehören. Da Prolongation und Restriktion nur einen Teilbereich von Level 0 — nämlich den, der weiter verfeinert wurde — betreffen, können alle Bausteine levelweise organisiert werden.

Der Stufen-Begriff spielt demzufolge bei der Implementierung keine Rolle. Im Falle global verfeinerter Gitter fallen die Begriffe Level und Stufe zusammen. In diesem Sinne sind Multilevelverfahren in der Tat eine Weiterentwicklung der klassischen Mehrgitterverfahren für den Fall lokal verfeinerter Gitter.

4.6 A-posteriori Fehlerschätzung

Uniforme Gitterverfeinerung erweist sich im Rahmen der Finite Elemente Methode in vielen Fällen als ungeeignet; denn sie trägt lokalen Eigenschaften der exakten Lösung in keiner Weise Rechnung. Bereiche, in denen die Lösung kaum variiert, werden genauso verfeinert wie solche, in denen Singularitäten auftreten.

A-priori-Fehlerabschätzungen liefern Aussagen über das asymptotische Verhalten der berechneten Näherungslösungen, falls die Gitterweite gegen Null strebt. Aus praktischer Sicht ist jedoch anzustreben, auf der Basis einer berechneten Näherungslösung Gitterzonen, die einer Verfeinerung bedürfen, von anderen, in denen der polynomiale Ansatz auf einem relativ groben Gitter bereits akzeptable Approximationsergebnisse liefert, zu unterscheiden.

Die Konstruktion eines derartigen *a-posteriori* Fehlerschätzers für die Standard Finite Elemente Diskretisierung des elliptischen Randwertproblems (4.2), der von BANK und WEISER [4] entwickelt wurde, ist das Thema dieses Abschnitts.

4.6.1 Aufgabenstellung

Betrachtet wird das Problem (4.1) in der abstrakten Formulierung (4.2):

$$L(\psi) \quad := \quad -\mathrm{div}\,(a\nabla\psi) = f \quad \text{in} \quad \Omega \tag{4.46}$$

mit a und f gemäß (4.3) und den Randbedingungen

$$\psi \quad = \quad 0 \quad \text{auf} \quad \Gamma_D \tag{4.47}$$

$$a\nabla\psi \cdot \mathbf{n} \quad = \quad g \quad \text{auf} \quad \Gamma_N \tag{4.48}$$

Mit $\Gamma_D := \Gamma_B$ bzw. $\Gamma_N := \Gamma_I \cup \Gamma_O$ sind dabei der Dirichlet- bzw. Neumannrand bezeichnet. g ist eine gegebene Funktion auf Γ_N. Mit $H^1_{\Gamma_D}(\Omega) := \{v \in H^1(\Omega) : v = 0 \text{ auf } \Gamma_D\}$ lautet die zugehörige Variationsformulierung:

Finde $\psi \in H^1_{\Gamma_D}(\Omega)$ mit

$$a(\psi, v) \quad = \quad (f, v) + < g, v > \quad \text{für alle} \quad v \in H^1_{\Gamma_D}(\Omega)$$

Mit $(f, v) := \int_\Omega fv \; dmd\varphi$ ist das L^2-Skalarprodukt auf Ω, mit $< g, v > := \int_{\Gamma_N} gv \; d\Gamma$ das auf Γ_N bezeichnet. Die Bilinearform ist gegeben durch

$$a(\psi, v) \;:=\; \int_\Omega a \nabla\psi \cdot \nabla v \; dmd\varphi \quad \text{für} \quad (\psi, v) \in H^1_{\Gamma_D}(\Omega) \times H^1_{\Gamma_D}(\Omega)$$

Falls $\Gamma_D \neq \emptyset$, ist diese Bilinearform positiv definit, definiert also ein Skalarprodukt. Die dadurch induzierte Norm wird als Energienorm bezeichnet:

$$|||\psi|||^2 \;:=\; a(\psi, \psi) \quad \text{für} \quad \psi \in H^1_{\Gamma_D}(\Omega)$$

Sei $\mathcal{L}_h$ der Raum der stetigen, stückweise linearen Finiten Elemente über einer Dreieckstriangulation $\mathcal{T}_h$ von Ω, die auf Γ_D verschwinden, dann lautet die diskrete Variationsformulierung von (4.46) unter Einbeziehung der Randwerte (4.47), (4.48):

Finde $\psi_{h,l} \in \mathcal{L}_h$ mit

$$a(\psi_{h,l}, v_{h,l}) \;=\; (f, v_{h,l}) + < g, v_{h,l} > \quad \text{für alle} \quad v_{h,l} \in \mathcal{L}_h \tag{4.49}$$

Wesentlich für die Konstruktion des Fehlerschätzers ist eine zweite Approximation von höherer Ordnung.

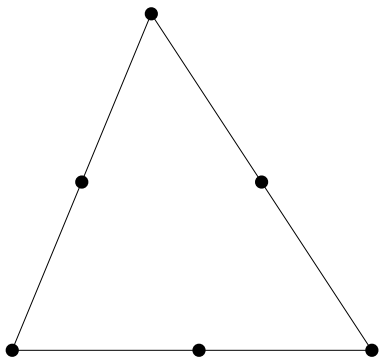

Abb. 4.5: Freiheitsgrade bei quadratischem Ansatz

Sei $\mathcal{Q}_h$ der Raum der stetigen, stückweise quadratischen Finiten Elemente über derselben Triangulation $\mathcal{T}_h$, die ebenfalls auf Γ_D verschwinden, so gilt analog zu (4.49):

Finde $\psi_{h,q} \in \mathcal{Q}_h$ mit

$$a(\psi_{h,q}, v_{h,q}) \;=\; (f, v_{h,q}) + < g, v_{h,q} > \quad \text{für alle} \quad v_{h,q} \in \mathcal{Q}_h \tag{4.50}$$

Die Freiheitsgrade für den linearen bzw. quadratischen Ansatzraum sind mit den in den Abbildungen 4.2 bzw. 4.5 markierten Punkten assoziiert.

Ziel ist es nun, den Diskretisierungfehler der linearen Näherungslösung $e_h := \psi - \psi_{h,l}$ in der Energienorm zu schätzen. Präzisiert heißt das: Gesucht ist eine mit möglichst geringem Aufwand lokal zu berechnende Größe ε_h, deren Energienorm den exakten Diskretisierungsfehler weder wesentlich

über- noch unterschätzt. In der Fehlerschätzertheorie spricht man gewöhnlich von *effizienten* und *zuverlässigen* lokalen a-posteriori Fehlerschätzern.

Dabei wird ein Fehlerschätzer dann als effizient akzeptiert, wenn er Verfeinerungsbedarf nur für die Regionen des Gebietes anzeigt, in denen Verfeinerung auch tatsächlich vonnöten ist, und somit zur Reduktion des Rechenaufwandes beiträgt. Im Gegensatz dazu verlangt man von einem zuverlässigen Schätzer, daß aus Approximationsgründen notwendige Verfeinerungen auf jeden Fall durchgeführt werden. In mathematischer Form lautet die Forderung:

$$c_1|||e_h||| \leq |||\varepsilon_h||| \leq c_2|||e_h||| \tag{4.51}$$

mit Konstanten $0 < c_1 \leq c_2$. Am günstigsten ist offenbar der Fall $c_1 = c_2 = 1$.

Die Konstruktion des Schätzers erfolgt in zwei Schritten:

- Defektkorrektur in einem Ansatzraum höherer Ordnung
- Lokalisierung durch hierarchisches Zwei-Level-Splitting

4.6.2 Approximation höherer Ordnung und Defektgleichung

Grundsätzlich erfolgt die Konstruktion des Fehlerschätzers mit derselben Technik, die sich in der Numerik gewöhnlicher Differentialgleichungen, vgl. z.B. STÖR und BULIRSCH [68], bewährt hat: Man vergleicht die berechnete Näherungslösung mit einer Approximation höherer Ordnung.

Saturationsannahme

Für die folgende Darstellung sind zusätzliche Bezeichnungen notwendig, die an dieser Stelle bereitgestellt werden sollen. E sei die Menge aller Kanten der Triangulation, E_I die der inneren und E_N die der Neumannkanten. Für ein Dreieck $\tau \in \mathcal{T}_h$ bezeichne E_τ die drei Kanten von τ und $\mathbf{n}_\tau$ den (stückweise konstanten) äußeren Normaleneinheitsvektor.

Außerdem wird für jede innere Kante eine Normalenrichtungen ausgezeichnet und der assoziierte Einheitsvektor mit $\mathbf{n}$ bezeichnet, vgl. Abbildung 4.6. Die Wahl zwischen den beiden möglichen Richtungen ist dabei beliebig. Die beiden angrenzenden Dreiecke werden mit τ_{in} und τ_{out} bezeichnet, wobei $\mathbf{n}$ von τ_{in} aus nach außen gerichtet ist. Für $e \in E_N$ bezeichnet $\mathbf{n}$ den äußeren Normaleneinheitsvektor. Auf der Basis dieser Bezeichnungen wird nun auf $e \in E_I$ die Sprungfunktion $[.]_J$

$$[v]_J(x) := v(x)|_{\tau_{out}} - v(x)|_{\tau_{in}}$$

definiert. Man beachte, daß die Größe $[\partial v/\partial \mathbf{n}]_J$ von der speziellen Wahl der Richtung von $\mathbf{n}$ unabhängig ist.

Üblicherweise wird folgende *Saturationsannahme* vorausgesetzt:

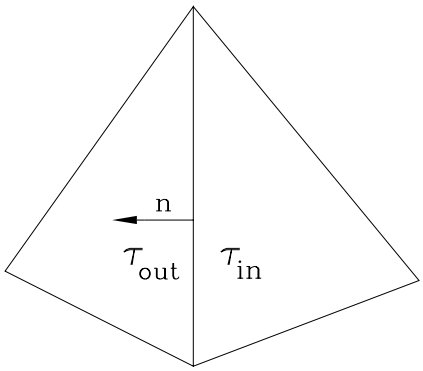

Abb. 4.6: Normalenvektor über eine innere Kante

$$|||\psi - \psi_{h,q}||| \leq \beta \, |||\psi - \psi_{h,l}||| \tag{4.52}$$

mit $0 \leq \beta < 1$. Dies bedeutet, daß man von einem Ansatz höherer Ordnung eine bessere Approximation auf demselben Gitter erwartet. Wesentlich ist, daß es sich hierbei um eine *Annahme* handelt: Die Ordnung eines Verfahren sagt grundsätzlich nichts über die Genauigkeit der Approximation auf einem einzelnen Gitter aus.

Als Fehlerschätzer setzt man nun an:

$$\varepsilon_h := \psi_{h,q} - \psi_{h,l} \tag{4.53}$$

Unter der Annahme (4.52) folgt mit der Dreiecksungleichung

$$\begin{aligned} |||\psi_{h,q} - \psi_{h,l}||| &\leq |||\psi - \psi_{h,q}||| + |||\psi - \psi_{h,l}||| \\ &\leq (1+\beta)|||\psi - \psi_{h,l}||| \end{aligned}$$

und umgekehrt

$$\begin{aligned} |||\psi_{h,q} - \psi_{h,l}||| &\geq |||\psi - \psi_{h,l}||| - |||\psi - \psi_{h,q}||| \\ &\geq (1-\beta)|||\psi - \psi_{h,l}||| \end{aligned}$$

Insgesamt ergibt sich daraus:

$$\frac{1}{1+\beta}|||\varepsilon_h||| \leq |||e_h||| \leq \frac{1}{1-\beta}|||\varepsilon_h|||$$

Dies ist in der Tat eine Abschätzung der Form (4.51). Die Berechnung von ε_h setzt jedoch die Kenntnis von $\psi_{h,q}$ voraus, so daß der Fehlerschätzer in dieser Form nicht in Frage kommt.

Fehlergleichung

Einen wesentlicher Schritt auf dem Weg zu einer mit vernünftigem Aufwand berechenbaren Größe stellt die folgende variationelle Charakterisierung für ε_h dar.

Für $v_{h,q} \in \mathcal{Q}_h$ folgt aus (4.53), (4.49), (4.50):

$$\begin{aligned}
a(\varepsilon_h, v_{h,q}) &= a(\psi_{h,q}, v_{h,q}) - a(\psi_{h,l}, v_{h,q}) \\
&= \int_\Omega f v_{h,q}\, dmd\varphi + \int_{\Gamma_N} g v_{h,q}\, d\Gamma - \int_\Omega a\nabla\psi_{h,l} \cdot \nabla v_{h,q}\, dmd\varphi \\
&= \sum_{\tau\in\mathcal{T}_h} \left(\int_\tau f v_{h,q}\, dmd\varphi - \int_\tau a\nabla\psi_{h,l} \cdot \nabla v_{h,q}\, dmd\varphi \right) + \int_{\Gamma_N} g v_{h,q}\, d\Gamma \\
&= \sum_{\tau\in\mathcal{T}_h} \left(\int_\tau f v_{h,q}\, dmd\varphi + \int_\tau \operatorname{div}(a\nabla\psi_{h,l}) v_{h,q}\, dmd\varphi - \int_{\partial\tau} a\nabla\psi_{h,l} \cdot \mathbf{n}_\tau v_{h,q}\, d\Gamma \right) \\
&\quad + \int_{\Gamma_N} g v_{h,q}\, d\Gamma \\
&= \sum_{\tau\in\mathcal{T}_h} \int_\tau \left(f - L(\psi_{h,l})\right) v_{h,q}\, dmd\varphi + \int_{\Gamma_N} \left(g - a\nabla\psi_{h,l}\cdot\mathbf{n}\right) v_{h,q}\, d\Gamma \\
&\quad + \sum_{e\in E_I} \int_e \left[a\nabla\psi_{h,l}\cdot\mathbf{n}\right]_J v_{h,q}\, d\Gamma
\end{aligned}$$

oder mit den Residuen $r := f - L(\psi_{h,l})$ und $r_N := g - a\nabla\psi_{h,l}\cdot\mathbf{n}$

$$a(\varepsilon_h, v_{h,q}) = \sum_{\tau\in\mathcal{T}_h} \int_\tau r v_{h,q} dx + \int_{\Gamma_N} r_N v_{h,q} d\Gamma + \sum_{e\in E_I} \int_e \left[a\nabla\psi_{h,l}\cdot\mathbf{n}\right]_J v_{h,q} d\Gamma \qquad (4.54)$$

Lokalisierung und hierarchisches Splitting

Zum Zwecke der Lokalisierung von (4.54) werden zunächst die Stetigkeitsbedingungen der Räume $\mathcal{L}_h$ und $\mathcal{Q}_h$ außer acht gelassen, so daß globale Kopplungen entfallen. Da außerdem der Fehlerschätzer ε_h nach (4.53) als Differenz von quadratischer und linearer Approximation angesetzt wird, bietet es sich an, den lokalen quadratischen Ansatzraum $\mathcal{Q}_h^{loc}$ analog zu zerlegen:

$$\mathcal{Q}_h^{loc} = \mathcal{L}_h^{loc} \bigoplus \tilde{\mathcal{Q}}_h^{loc}$$

Mit dem hierarchischen Überschuß $\tilde{\mathcal{Q}}_h^{loc}$ sind nun gerade die Freiheitsgrade in den Kantenmittelpunkten assoziiert, vgl. Abbildung 4.7.

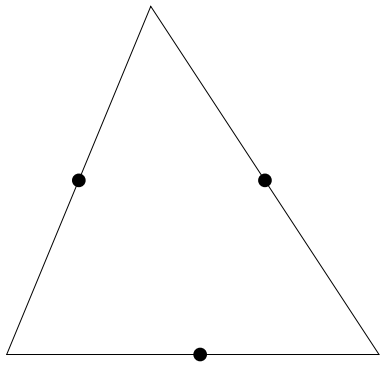

Abb. 4.7: Freiheitsgrade für den Fehlerschätzer

Zur Lokalisierung der rechten Seite wird der Sprung in der Normalenableitung von $\psi_{h,l}$ jeweils zur Hälfte auf die beiden angrenzenden Dreiecke verteilt:

$$F_\tau(v_{h,q}) \;:=\; \int_\tau r v_{h,q}\, dmd\varphi + \int_{E_\tau \cap E_N} r_N v_{h,q}\, d\Gamma + \frac{1}{2}\int_{E_\tau \cap E_I} \left[a\nabla\psi_{h,l}\cdot\mathbf{n}\right]_J v_{h,q}\, d\Gamma \qquad (4.55)$$

Damit ist der lokale Fehlerschätzer letztlich für jede Zelle $\tau \in \mathcal{T}_h$ gegeben als Lösung des folgenden Variationsproblems:

Finde $\varepsilon_h \in \tilde{\mathcal{Q}}_h^{loc}$, so daß

$$a(\varepsilon_h, v_{h,q}) \;=\; F_\tau(v_{h,q}) \quad \text{für alle} \quad v_{h,q} \in \tilde{\mathcal{Q}}_h^{loc} \qquad (4.56)$$

(4.56) ist äquivalent zu einem 3×3-Gleichungssystem mit symmetrisch positiv definiter Koeffizientenmatrix. Die theoretische Untersuchung dieses Fehlerschätzers im Hinblick auf (4.51) ist bei Bank und Weiser [4] nachzulesen.

4.6.3 Interpretation

Zur Implementierung von (4.56) bietet sich die Verwendung einer nodalen Basis von $\tilde{\mathcal{Q}}_h^{loc}$ bezüglich der Kantenmittelpunkte an. Zu diesem Zweck werden an dieser Stelle folgende Notationen bezüglich eines beliebigen Dreiecks $\tau \in \mathcal{T}_h$ vereinbart, vgl. Abbildung 4.8:

Die baryzentrischen Koordinaten λ_i $(1 \le i \le 3)$ sind nun definiert als diejenigen linearen Funktionen über τ, die den Bedingungen

$$\lambda_i(a_j) \;=\; \delta_{ij}, \quad 1 \le j \le 3$$

genügen. Mit den Bezeichnungen in Abbildung 4.8 erhält man die explizite Darstellung:

$$\lambda_i(m,\varphi) = -\frac{|e_i|}{2|\tau|}\left(\begin{pmatrix} m \\ \varphi \end{pmatrix} - \mathbf{m}_i, \mathbf{n}_i\right) \quad \text{für} \quad \begin{pmatrix} m \\ \varphi \end{pmatrix} \in \tau$$

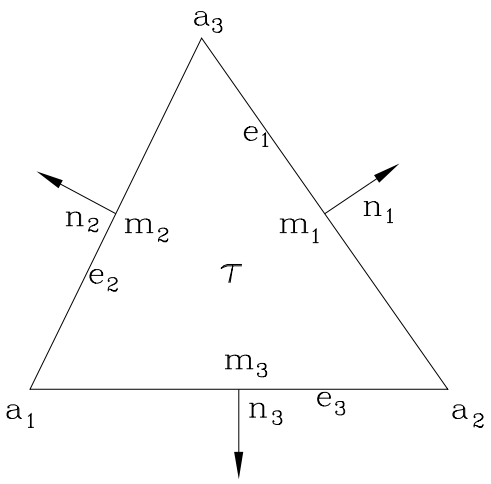

Abb. 4.8: Bezeichnungen für ein beliebiges Dreieck

Dabei ist mit e_i die Kante bezeichnet, die der i-ten Ecke gegenüberliegt, mit $|e_i|$ deren Länge, mit $|\tau|$ die Fläche des Dreiecks, mit $\mathbf{m}_i$ der Mittelpunkt der Kante i und mit $\mathbf{n}_i$ der i-te nach außen gerichteten Normalen*einheits*vektor. Die entsprechenden Vektoren, deren Länge mit der der assoziierten Kante übereinstimmt, werden mit $\mathbf{N}_i$ bezeichnet, d.h.

$$\mathbf{N}_i = |e_i|\mathbf{n}_i, \quad 1 \leq i \leq 3.$$

Eine nodale Basis, die den Bedingungen

$$\chi_i(m_j) = \delta_{ij}, \quad 1 \leq j \leq 3.$$

genügt, ist dann gegeben durch

$$\chi_i = 4\lambda_{i+1}\lambda_{i+2}, \quad 1 \leq i \leq 3. \tag{4.57}$$

Die χ_i zeichnen sich dadurch aus, daß sie auf den Kanten $i+1$ und $i+2$ verschwinden. Deshalb werden sie auch in Anlehnung an die kubischen bubble-Funktionen, die auf allen drei Kanten verschwinden und innerhalb des Elementes positive Werte annehmen, als *quadratische bubbles* bezeichnet. Zur Notation: Ein Index j in (4.57) ist als $(j-1) \bmod 3 + 1$ zu verstehen.

Bezüglich der Basis (4.57) von $\tilde{\mathcal{Q}}_h^{loc}$ läßt sich jetzt die rechte Seite von (4.56) physikalisch interpretieren. Nach (4.55) spielen dabei drei Terme eine Rolle: Das lokale Residuum, das Residuum auf dem Neumannrand und der Sprung der Normalkomponente des Flusses über innere Kanten. Für die exakte Lösung würden alle diese drei Terme entfallen. Das heißt, der Fehlerschätzer überprüft, wie gut die Näherungslösung $\psi_{h,l}$ diese Konsistenzbedingungen erfüllt.

Das lokale Residuum ergibt sich dabei, wenn man in die Erhaltungsgleichung (4.46) die diskrete Lösung einsetzt, das Residuum mit χ_i gewichtet und über die Zelle $\tau \in \mathcal{T}_h$ integriert. Eine prägnante Formulierung erhält man, wenn man beachtet, daß sich $f = -2\omega r \frac{dr}{dm}$ als Divergenz des Vektorfeldes $\begin{pmatrix} -\omega r^2 \\ 0 \end{pmatrix}$ darstellen läßt. Mit $\mathbf{F} := \begin{pmatrix} -\omega r^2 \\ 0 \end{pmatrix} + a\nabla\psi_{h,l}$ folgt dann unter Verwendung der Greenschen Formel (3.13) mit $f := \chi_i$ und $\mathbf{v} := \mathbf{F}$:

$$\begin{aligned} \int_\tau r\chi_i \, dmd\varphi &= \int_\tau \left(f + \operatorname{div}(a\nabla\psi_{h,l})\right)\chi_i \, dmd\varphi \\ &= \int_\tau \operatorname{div}(\mathbf{F})\chi_i \, dmd\varphi \\ &= \int_{e_i} \mathbf{F}\cdot\mathbf{n}\chi_i \, d\Gamma - \int_\tau \mathbf{F}\cdot\nabla\chi_i \, dmd\varphi. \end{aligned}$$

Wertet man das Randintegral mit der Simpson-Formel und das Flächenintegral mit der Quadraturformel $\int_\tau p \, dmd\varphi \approx \frac{|\tau|}{3}\sum_{k=1}^{3} p(\mathbf{m}_k)$ aus, so erhält man unter Beachtung von

$$\nabla\chi_i = \frac{1}{|\tau|}\sum_{k=1}^{3}(-1)^{1-\delta_{ik}}\mathbf{N}_k$$

die Approximation:

$$\begin{aligned} \int_\tau r\chi_i \, dmd\varphi &\approx \frac{2}{3}\mathbf{F}(\mathbf{m}_i)\cdot\mathbf{N}_i - \frac{1}{3}\sum_{k=1}^{3}(-1)^{1-\delta_{ik}}\mathbf{F}(\mathbf{m}_k)\cdot\mathbf{N}_k \\ &= \frac{1}{3}\sum_{k=1}^{3}\mathbf{F}(\mathbf{m}_k)\cdot\mathbf{N}_k. \end{aligned}$$

Mit Hilfe dieser Darstellung gelingt nun der Brückenschlag zum Finite Volumen Verfahren. Letzteres führt direkt auf die *diskrete Form der Erhaltungsgleichung*:

$$\sum_{k=1}^{3}\mathbf{F}(\mathbf{m}_k)\cdot\mathbf{N}_k = 0.$$

Die beiden anderen Summanden in (4.55) sind analog zu behandeln. Insgesamt wird also überprüft, inwieweit die Finite Elemente Lösung folgende Kriterien erfüllt, die für Finite Volumen Ansätze selbstverständlich sind:

- Lokale Erhaltungseigenschaft für jede Zelle
- Über innere Kanten stimmen die Flüsse von beiden Seiten überein
- Flußvorgaben auf dem Neumannrand werden exakt erfüllt

Von der Systematik bei der Ableitung des Fehlerschätzers abgesehen, zeigt sich an dieser Stelle, weshalb Finite Elemente Ansätze im Zusammenhang mit adaptiven Strategien im Vergleich zu Finite Volumen Verfahren Vorteile aufweisen: Während Finite Elemente Verfahren auf Variationsprinzipien und Galerkin-Ansätzen beruhen und die drei genannten Kriterien zur a-posteriori Fehlerschätzung nutzen können, werden Finite Volumen Verfahren gerade mittels dieser Kriterien konstruiert.

Kapitel 5

Die Lösung der Euler Gleichungen

In diesem Kapitel geht es um die numerische Lösung der Gleichungen, vgl. (3.7) und (3.8):

$$(\mathbf{w} \cdot \nabla)\mathbf{c} + \boldsymbol{\omega} \times \mathbf{c} + \nabla p/\rho = 0 \qquad \text{in } \Omega \tag{5.1}$$

$$\operatorname{div} \mathbf{c} = 0 \qquad \text{in } \Omega. \tag{5.2}$$

Die zweidimensionale Projektion eines zugrundeliegenden Kontrollraums Ω für eine rein radiale Maschine ist in Abbildung 5.1 schematisch skizziert. Die dreidimensionale Darstellung des Laufrades einer Francisturbine ist in Abbildung 6.30 gegeben.

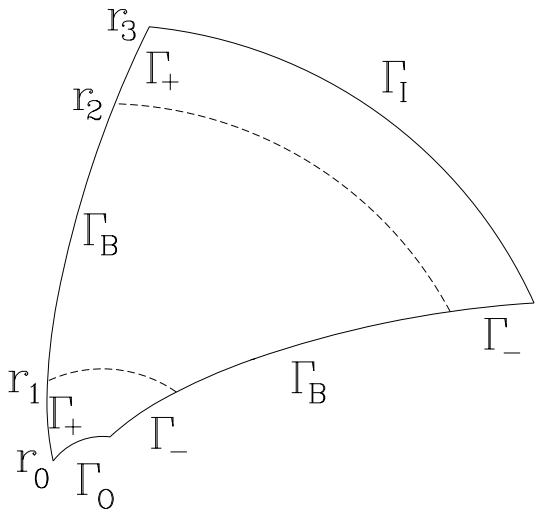

Abb. 5.1: Kontrollraum und Randbedingungen

Mit diesen Bezeichnungen lassen sich die für Strömungsmaschinen üblichen Randbedingungen folgendermaßen formulieren:

$$\begin{aligned}
\mathbf{c} &= \mathbf{c}_{in} \qquad \text{auf } \Gamma_I && (5.3)\\
p &= p_{out} \qquad \text{auf } \Gamma_O && (5.4)\\
p_+ &= p_- \qquad \text{auf } \Gamma_- && (5.5)\\
\mathbf{c}_+ &= Q(\gamma)\mathbf{c}_- \qquad \text{auf } \Gamma_- && (5.6)\\
\mathbf{c}\cdot\mathbf{n} &= \mathbf{u}\cdot\mathbf{n} \qquad \text{auf } \Gamma_B && (5.7)
\end{aligned}$$

Dabei bezeichnen Γ_I den Einström-, Γ_O den Ausström- und $\Gamma_+ \cup \Gamma_-$ den periodischen Rand, wobei sich Γ_+ ergibt, wenn man Γ_- bezüglich der z-Achse um den Teilungswinkel $\gamma = 2\pi/Z$ gegen den Uhrzeigersinn dreht. Z ist dabei die Schaufelzahl. Alle übrigen Ränder, wie Nabe, Deckscheibe und Schaufeloberfläche sind mit Γ_B bezeichnet.

Die Geschwindigkeitsverteilung am Eintritt, $\mathbf{c}_{in}$, ist genauso gegeben wie die Druckverteilung am Kontrollraumaustritt, p_{out}, und die Führungsgeschwindigkeit $\mathbf{u} = \boldsymbol{\omega} \times \mathbf{x}$. Der äußere Normaleneinheitsvektor ist mit $\mathbf{n}$ bezeichnet. $\tilde{p}_+$ bzw. p_- sind statische Drücke an sich entsprechenden periodischen Punkten, wobei sich p_+ aus $\tilde{p}_+$ analog zu Abschnitt 4.1 durch Verschiebung ergibt.

Analoge Definitionen gelten für die Geschwindigkeiten, wobei die Geschwindigkeitsvektoren $\mathbf{c}_+$ und $\mathbf{c}_-$ nicht identisch sind, sondern den Winkel γ einschließen. Mit $Q(\gamma)$ ist demzufolge die Drehung bezüglich der z-Achse um den Winkel γ gegen den Uhrzeigersinn bezeichnet.

5.1 Variationsformulierung

Analog zu Abschnitt 4.2 wird (5.1) mit einem *geeigneten* Vektorfeld $\mathbf{v}$, (5.2) mit einer *geeigneten* skalarwertigen Funktion q multipliziert und jeweils über Ω integriert:

$$\begin{aligned}
\int_\Omega \left((\mathbf{w}\cdot\nabla)\mathbf{c}\right)\cdot\mathbf{v} + \left((\boldsymbol{\omega}\times\mathbf{c})\cdot\mathbf{v}\right) + (\nabla p\cdot\mathbf{v})\; d\mathbf{x} &= 0\\
\int_\Omega \mathrm{div}\mathbf{c}\cdot q\; d\mathbf{x} &= 0
\end{aligned}$$

Hier und im folgenden wird p für p/ρ geschrieben. Dieser Ansatz entspricht in der Terminologie der klassischen Mechanik dem *Prinzip der virtuellen Leistung*. Um nun das Problem variationell formulieren zu können, sind folgende Definitionen sinnvoll:

$$\begin{aligned}
\mathbf{H} &:= \left\{\mathbf{c}\in\mathbf{H}^1(\Omega) : \mathbf{c} = \mathbf{c}_{in} \text{ auf } \Gamma_I, \mathbf{c}_+ = Q(\gamma)\mathbf{c}_- \text{ auf } \Gamma_-, \mathbf{c}\cdot\mathbf{n} = \mathbf{u}\cdot\mathbf{n} \text{ auf } \Gamma_B\right\}\\
\mathbf{V} &:= \left\{\mathbf{v}\in\mathbf{H}^1(\Omega) : \mathbf{v} = 0 \text{ auf } \Gamma_I, \mathbf{v}_+ = Q(\gamma)\mathbf{v}_- \text{ auf } \Gamma_-, \mathbf{v}\cdot\mathbf{n} = 0 \text{ auf } \Gamma_B\right\}\\
Q &:= L^2(\Omega)
\end{aligned}$$

Mit $\mathbf{H}^1(\Omega)$ ist dabei der Sobolevraum vektorwertiger Funktionen bezeichnet, deren Komponenten allesamt im $H^1(\Omega)$ enthalten sind, d.h. $\mathbf{H}^1(\Omega) := H^1(\Omega)\times H^1(\Omega)\times H^1(\Omega)$. Die Konstruktion des $H^1(\Omega)$ ist in Abschnitt 4.2 erläutert. Theoretische Ergebnisse im Zusammenhang mit inkompressiblen Euler Gleichungen sind bei Marchioro und Pulvirenti [45] nachzulesen.

Mit Hilfe dieser Räume und der Greenschen Formel (3.13) mit $f := -p$ lautet nun die Variationsformulierung von (5.1), (5.2) unter Einbeziehung der Randwerte (5.3)–(5.7):

Finde ein Paar $(\mathbf{c}, p) \in \mathbf{H} \times Q$, so daß

$$\int_\Omega ((\mathbf{w} \cdot \nabla)\mathbf{c}) \cdot \mathbf{v} + ((\boldsymbol{\omega} \times \mathbf{c}) \cdot \mathbf{v}) - p \cdot \mathrm{div}\mathbf{v}\, d\mathbf{x} + \int_{\Gamma_O} p_{out}\mathbf{v} \cdot \mathbf{n}\, d\Gamma = 0 \quad \text{für alle } \mathbf{v} \in V \quad (5.8)$$

$$\int_\Omega q \cdot \mathrm{div}\mathbf{c}\, d\mathbf{x} = 0 \quad \text{für alle } q \in Q \quad (5.9)$$

Die zusätzlichen Terme

$$\int_{\Gamma_I} p\mathbf{v} \cdot \mathbf{n}\, d\Gamma + \int_{\Gamma_B} p\mathbf{v} \cdot \mathbf{n}\, d\Gamma + \int_{\Gamma_-} p_-\mathbf{v}_- \cdot \mathbf{n}_- + p_+\mathbf{v}_+ \cdot \mathbf{n}_+\, d\Gamma$$

entfallen: der erste wegen $\mathbf{v} = 0$ auf Γ_I, der zweite wegen $\mathbf{v} \cdot \mathbf{n} = 0$ auf Γ_B und die beiden letzten wegen $p_+ = p_-$, $\mathbf{v}_+ = Q(\gamma)\mathbf{v}_-$ und $\mathbf{n}_+ = -Q(\gamma)\mathbf{n}_-$ jeweils auf Γ_-.

Die Anwendung der Greenschen Formel (3.13) reduziert zum einen die Differenzierbarkeitsanforderungen an den statischen Druck p und erlaubt zum anderen die Formulierung von Spannungsrandbedingungen. Die Druckvorgabe am Ausströmrand geht als natürliche Randbedingung in diese Formulierung ein. Gibt man einen konstanten Druck $p_{out} = 0$ vor, so verschwindet der Term $\int_{\Gamma_{out}} p_{out}\mathbf{v} \cdot \mathbf{n}\, d\Gamma$ und muß nicht weiter berücksichtigt werden. Aus diesem Grund wird diese Art der Randbedingung auch als *do nothing*-Bedingung bezeichnet.

Definiert man auf $\mathbf{H} \times Q$ die Bilinearform

$$b(\mathbf{v}, p) := -\int_\Omega p \cdot \mathrm{div}\mathbf{v}\, d\mathbf{x}$$

und auf $\mathbf{H} \times \mathbf{H} \times \mathbf{H}$ die Trilinearform

$$n(\mathbf{w}, \mathbf{c}, \mathbf{v}) := \int_\Omega ((\mathbf{w} \cdot \nabla)\mathbf{c}) \cdot \mathbf{v}\, d\mathbf{x}$$

so läßt sich (5.8), (5.9) äquivalent formulieren als

Finde ein Paar $(\mathbf{c}, p) \in \mathbf{H} \times Q$, so daß

$$n(\mathbf{w}, \mathbf{c}, \mathbf{v}) + b(\mathbf{v}, p) = 0 \quad \text{für alle } \mathbf{v} \in \mathbf{V} \quad (5.10)$$

$$b(\mathbf{c}, q) = 0 \quad \text{für alle } q \in Q \quad (5.11)$$

5.2 Diskretisierung

Zum Zwecke der Diskretisierung von (5.8) und (5.9) wird ein von Rannacher und Turek [55] entwickeltes Finites Element verwendet. Dieses Element verallgemeinert das Crouzeix-Raviart-Element für simpliziale Triangulationen, vgl. z.B. Braess [11], auf Zerlegungen des Grundgebiets Ω in quadrilaterale Elemente in zwei bzw. hexalaterale in drei Raumdimensionen.

Die Freiheitsgrade für die Geschwindigkeitsunbekannten sind dabei mit den Kanten- bzw. Seitenmittelpunkten assoziiert und ordnen jeder Ansatzfunktion den Mittelwert über die jeweilige Kante bzw. Seite zu. Der Druck ist stückweise konstant. Im Gegensatz zu einem *cell-centered* Finite Volumen Verfahren handelt es sich hier also um eine versetzte Anordnung der Freiheitsgrade.

5.2.1 Das Element von Rannacher und Turek

Im zweidimensionalen Fall ergeben sich die Ansatzfunktionen lokal aus einem *rotiert* bilinearen Ansatz für jede der 2 kartesischen Geschwindigkeitskomponenten:

$$S_h(\tau) \;:=\; \left\{\hat{s}\circ\psi_\tau^{-1} : \hat{s}\in \operatorname{span}\{1,\xi,\eta,\xi^2-\eta^2\}\right\}^2 \tag{5.12}$$

Das dreidimensionale Analogon lautet:

$$S_h(\tau) \;:=\; \left\{\hat{s}\circ\psi_\tau^{-1} : \hat{s}\in \operatorname{span}\{1,\xi,\eta,\zeta,\xi^2-\eta^2,\eta^2-\zeta^2\}\right\}^3 \tag{5.13}$$

Diese Definition entspricht der üblichen Vorgehensweise: Die Basisfunktionen werden auf dem Referenzelement $\hat{\tau} := [-1,1]^2$ bzw. $\hat{\tau} := [-1,1]^3$ spezifiziert und mittels der Transformation $\psi_\tau : \hat{\tau} \to \tau$ auf eine beliebige Zelle $\tau \in \mathcal{T}_h$ transportiert, vgl. Abbildung 5.2.

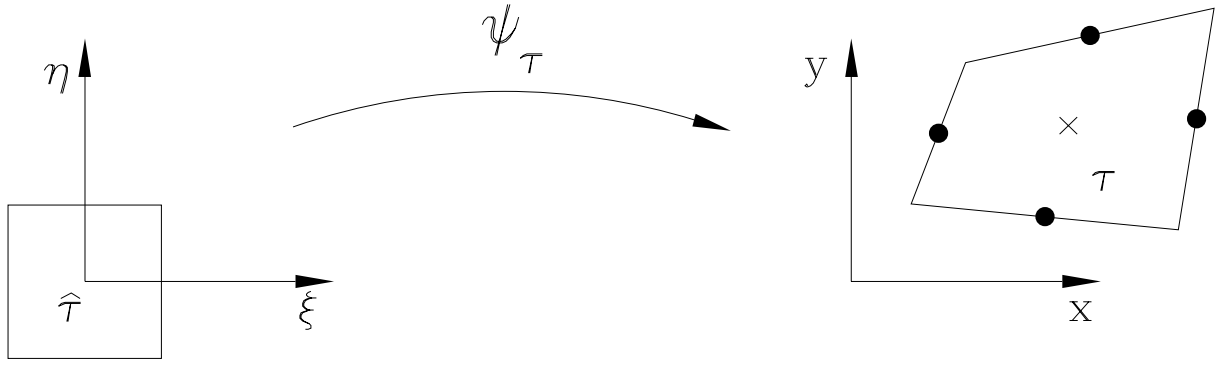

Abb. 5.2: Transformation auf das Referenzelement. Markiert sind die Freiheitsgrade für Geschwindigkeiten und Druck.

Ein alternativer lokaler Ansatzraum, der ohne Referenzelement auskommt, wird weiter unten definiert. Die Freiheitsgrade sind durch folgende Funktionale gegeben:

$$F_{e_h}(\mathbf{s}_h) \;:=\; \frac{1}{|e_h|}\int_{e_h} \mathbf{s}_h \, d\Gamma \tag{5.14}$$

Jeder Kante bzw. Seite e_h einer Zelle $\tau \in \mathcal{T}_h$ ist also der Mittelwert von $\mathbf{s}_h$ über e_h zugeordnet. Die Mittelwertbildung (5.14) wird im folgenden sowohl auf vektor- als auch auf skalarwertige Funktionen angewendet. Es stellt sich die Frage, warum gerade ein *rotiert* multilinearer Ansatz verwendet wird. Eine naheliegende Alternative im zweidimensionalen Fall wäre der bilineare Ansatz

$$S_h^{bilinear}(\tau) \;:=\; \left\{\hat{s}\circ\psi_\tau^{-1} : \hat{s}\in \mathrm{span}\{1,\xi,\eta,\xi\eta\}\right\} \tag{5.15}$$

für jede Geschwindigkeitskomponente. Der lokale Raum (5.15) in Verbindung mit den Freiheitsgraden (5.14) ist jedoch nicht *unisolvent*, d.h. es existiert keine $1-1$–Zuordnung zwischen nodalen Werten und Ansatzfunktionen. Zur Erläuterung sei angenommen, daß bezüglich des Referenzquadrates $\hat{\tau}$, vgl. Abbildung 5.2, nodale Werte, also Mittelwerte im Sinne von (5.14), berechnet sind, die allesamt verschwinden. Neben der Nullfunktion erfüllt offensichtlich auch die bilineare Funktion

$$\hat{s}(\xi,\eta) := \xi\eta \tag{5.16}$$

diese Bedingung. Welcher ist nun beispielsweise der richtige interpolierte Wert an der Stelle $(0.5, 0.5)$? Im Gegensatz zur Nullfunktion liefert die Auswertung von (5.16) 0.25. Um Mehrdeutigkeiten dieser Art zu vermeiden, werden für ein bezüglich des Referenzelementes um 45° gedrehtes Quadrat bilineare Ansatzfunktionen verwendet. Daraus ergibt sich für das Referenzquadrat dann der *rotiert* bilineare Ansatz (5.12).

Globaler Test- bzw. Ansatzraum für dreidimensionale Probleme ergeben sich damit als

$$\begin{aligned}\mathbf{V}_h \;:=\; &\{\mathbf{s}_h\in L^2(\Omega)\times L^2(\Omega)\times L^2(\Omega) : \mathbf{s}_h|_\tau \in S_h(\tau) \text{ für } \tau\in\mathcal{T}_h,\\ &\text{die linearen Funktionale (5.14) sind stetig über innere Zellflächen,}\\ &F_{e_h}(\mathbf{s}_h)=0 \text{ für Zellflächen } e_h \text{ auf } \Gamma_I\}\end{aligned}$$

bzw.

$$\begin{aligned}\mathbf{H}_h \;:=\; &\{\mathbf{s}_h\in L^2(\Omega)\times L^2(\Omega)\times L^2(\Omega) : \mathbf{s}_h|_\tau \in S_h(\tau) \text{ für } \tau\in\mathcal{T}_h,\\ &\text{die linearen Funktionale (5.14) sind stetig über innere Zellflächen,}\\ &F_{e_h}(\mathbf{s}_h)=\frac{1}{|e_h|}\int_{e_h}\mathbf{c}_{in}\,d\Gamma \text{ für Zellflächen } e_h \text{ auf } \Gamma_I\}.\end{aligned}$$

Die Definition des Ansatzraums $\mathbf{H}_h$ beispielsweise bedeutet, daß der Mittelwert der Geschwindigkeit über innere Zellflächen als stetig angesetzt wird und über Zellflächen auf dem Dirichletrand mit dem der vorgegebenen Geschwindigkeit $\mathbf{c}_{in}$ übereinstimmt.

Der Druck wird stückweise konstant approximiert:

$$Q_h \;:=\; \{q_h\in L^2(\Omega) : q_h|_\tau = \text{const. für } \tau\in\mathcal{T}_h\}.$$

Aus physikalischer Sicht ist dieser Ansatz deshalb besonders attraktiv, weil der Druck als Lagrange-Parameter die Volumenströme über die Kanten bzw. Seiten einer Zelle so kontrolliert, daß die

Kontinuität erfüllt ist. Mathematisch präzisieren läßt sich die Frage, ob Geschwindigkeits- und Druckansatz geeignet ausbalanziert sind, mit Hilfe der *Babuska-Brezzi-Bedingung*

$$\inf_{q_h \in Q_h} \sup_{\mathbf{v}_h \in \mathbf{V}_h} \frac{b_h(\mathbf{v}_h, q_h)}{||\mathbf{v}_h|| ||q_h||} \geq \beta \tag{5.17}$$

mit einer Konstanten β, die nicht von h abhängt. Die Bilinearform $b_h(.,.)$ wird im nächsten Abschnitt definiert. Auf die genaue Definition der Normen wird hier nicht eingegangen.

Die Bedingung (5.17) ist für das Rannacher-Turek-Element erfüllt. Praktisch bedeutet dies, daß im Gegensatz zu vielen anderen Geschwindigkeits-Druck-Kombinationen Instabilitäten, wie etwa das sogenannte Schachbrettverhalten, vgl. z.B. BRAESS [11], a priori ausgeschlossen sind. Auf Stabilisierungsmaßnahmen, wie sie beispielsweise bei *cell-centered* Finite Volumen Verfahren notwendig sind, kann man bei dieser Art der Finite Elemente Diskretisierung also verzichten. Ergebnisse, die bei Vergleichsrechnungen für Navier-Stokes Probleme erzielt wurden, sind bei SCHÄFER ET AL. [60] nachzulesen.

Wie das Crouzeix-Raviart- ist auch das Rannacher-Turek-Element ein *nichtkonformes*, d.h. der diskrete Geschwindigkeitsraum ist nicht Teilraum des kontinuierlichen. Von STRANG und FIX [69] als *variational crime* bezeichnet, stellen nichtkonforme Ansätze im Vergleich zu konformen zusätzliche Flexibilität zur Vermeidung von *locking*-Effekten zur Verfügung.

Unter *locking* versteht man, daß durch zu starke Gewichtung von Nebenbedingungen, wie hier der Kontinuitätsgleichung, das diskrete Problem überbestimmt ist. Näheres dazu ist beispielsweise in [50] nachzulesen. Konkrete Auswirkungen hat die Nichtkonformität des Ansatzes beispielsweise auf die Implementierung der Grobgitterkorrektur im Rahmen eines Mehrgitterverfahrens, vgl. MÜLLER and SZILAGYI [51].

Um optimale Approximationsordnung auch auf sehr verzerrten Gittern zu erhalten, schlägt TUREK [74] alternativ zur Spezifikation der Ansatzfunktionen auf dem Referenzelement vor, für jede Zelle einen separaten Referenzraum zu spezifizieren. Zu diesem Zweck wird für eine Zelle ein lokales Koordinatensystem durch Verbinden gegenüberliegender Seiten definiert. Diese Konstruktion vermittelt eine lineare Transformation der Zelle, auf deren Bild dann ein multilinearer Ansatz vom Typ (5.12) bzw. (5.13) gemacht wird. Details dazu sind bei SCHREIBER [64] nachzulesen.

Der Nachteil dieser Technik besteht darin, daß für jede Zelle im Zweidimensionalen eine 4×4-, im Dreidimensionalen sogar eine 6×6-Matrix invertiert werden muß. Solange jedoch ohne Diffusion und gleichzeitig mit Upwind-Diskretisierung, vgl. Abschnitt 5.2.2, gearbeitet wird, spielen Basisfunktionen bei der Implementierung ohnehin keine Rolle.

5.2.2 Das diskrete Problem ohne Nebenbedingungen

Wegen der Nichtkonformität des Raumes $\mathbf{V}_h$ sind zur Formulierung des diskreten Problems die Bilinearform

$$b_h(\mathbf{v}_h, p_h) \;:=\; -\sum_{\tau \in \mathcal{T}_h} p_h|_\tau \int_\tau \mathrm{div}\mathbf{v}_h \; d\mathbf{x}$$

für $\mathbf{v}_h \in V_h$ und $p_h \in Q_h$ und auf $\mathbf{V}_h \times \mathbf{V}_h \times \mathbf{V}_h$ die Trilinearform

$$n_h(\mathbf{w}_h, \mathbf{c}_h, \mathbf{v}_h) \quad := \quad \sum_{\tau \in \mathcal{T}_h} \int_\tau ((\mathbf{w}_h \cdot \nabla)\mathbf{c}_h) \cdot \mathbf{v}_h \, d\mathbf{x}$$

nützlich.

Damit lautet das diskrete Pendant zu (5.10), (5.11) **ohne Periodizitäts- und Eulersche Randbedingungen**:

Finde ein Paar $(\mathbf{c}_h, p_h) \in \mathbf{H}_h \times Q_h$, so daß

$$\begin{aligned} n_h(\mathbf{w}_h, \mathbf{c}_h, \mathbf{v}_h) + b_h(\mathbf{v}_h, p_h) &= 0 \qquad \text{für alle } \mathbf{v}_h \in \mathbf{V}_h & (5.18)\\ b_h(\mathbf{c}_h, q_h) &= 0 \qquad \text{für alle } q_h \in Q_h. & (5.19) \end{aligned}$$

Die beiden Formen $b_h(.,.)$ sowie $n_h(.,.,.)$ sollen im folgenden genauer untersucht werden.

Die diskrete Kontinuitätsgleichung

Die Diskretisierung des Divergenzoperators bereitet keine besonderen Schwierigkeiten:

$$b_h(\mathbf{c}_h, q_h) \quad = \quad -\sum_{\tau \in \mathcal{T}_h} q_h|_\tau Q_\tau(\mathbf{c}_h) \qquad \text{mit} \quad Q_\tau(\mathbf{c}_h) := \sum_{e_h \in E_\tau} |e_h| F_{e_h}(\mathbf{c}_h) \cdot \mathbf{n}_h \qquad (5.20)$$

Mit E_τ sind dabei die Randkanten bzw. -seiten der Zelle τ, mit $|e_h|$ die Länge der Kante bzw. Fläche der Seite e_h und mit $\mathbf{n}_h$ der äußere Normaleneinheitsvektor bezüglich e_h bezeichnet. Falls $\mathbf{n}_h$ über e_h nicht konstant ist, wird der Mittelwert verwendet. Gleichung (5.20) ist dann im allgemeinen nicht mehr exakt.

Weil Q_h aus stückweise konstanten Funktionen besteht, entspricht die diskrete Kontinuitätsforderung dem Prinzip der lokalen Volumenerhaltung, wie sie auch von Finite Volumen Verfahren respektiert wird: Die Summe der diskreten Volumenströme über die Zellflächen verschwindet für jede Zelle. Es sei ausdrücklich darauf hingewiesen, daß nicht alle Finite Elemente Diskretisierungen diesem Prinzip genügen.

Upwind-Diskretisierung

Zur Diskretisierung des Konvektionsoperators sind zusätzliche Stabilisierungsmaßnahmen vonnöten. Im Finite Elemente Kontext kommen dabei üblicherweise zwei Möglichkeiten in Betracht: *Stromliniendiffusion* und *Upwind*. Während die erste zusätzliche Diffusion lediglich in Strömungsrichtung addiert, geschieht dies bei der zweiten Variante in isotroper Weise. Demzufolge kommt man beim einfachen Upwind-Verfahren über die Fehlerordnung 1 nicht hinaus, vgl. Abschnitt 6.2.1. A-priori Abschätzungen im Zusammenhang mit Stromliniendiffusion zeigen eine Ordnung von 3/2. Details zum Stromliniendiffusionsverfahren sind bei Turek [74] und Schreiber [64] nachzulesen.

Die Upwind-Diskretisierung für den rotiert multilinearen Ansatz (5.12), (5.13) orientiert sich an der Vorgehensweise bei Finite Volumen Verfahren. Dazu ist eine Sekundärzerlegung des Gebietes

notwendig. In Abbildung 5.3 ist für den zweidimensionalen Fall das Kontrollvolumen R_l für einen Freiheitsgrad skizziert.

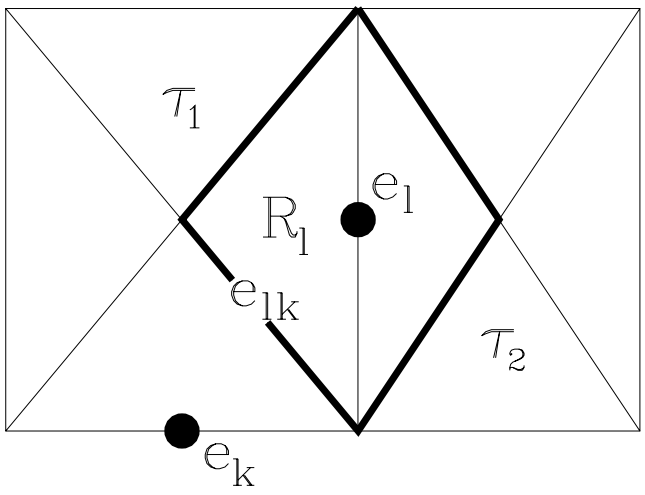

Abb. 5.3: Kontrollvolumen für die Upwind-Diskretisierung

Es entsteht, indem die Endpunkte der zugeordneten Kante mit den Mittelpunkten der angrenzenden Vierecke τ_1 und τ_2 verbunden werden. Den beiden dabei entstehenden Dreiecken entsprechen in drei Raumdimensionen Pyramiden.

Mit (3.17) kann man n_h als Differenz von

$$\begin{aligned} n_h^1(\mathbf{w}_h, \mathbf{c}_h, \mathbf{v}_h) &:= \sum_{\tau \in \mathcal{T}_h} \int_\tau \operatorname{div}(\mathbf{c}_h \mathbf{w}_h^T) \cdot \mathbf{v}_h \, d\mathbf{x} \quad \text{und} \\ n_h^2(\mathbf{w}_h, \mathbf{c}_h, \mathbf{v}_h) &:= \sum_{\tau \in \mathcal{T}_h} \int_\tau \operatorname{div}(\mathbf{w}_h) \mathbf{c}_h \cdot \mathbf{v}_h \, d\mathbf{x} \end{aligned}$$

schreiben. $\mathbf{v}_h$ in n_h^1 bzw. $\mathbf{c}_h$ und $\mathbf{v}_h$ in n_h^2 werden mit Hilfe der Mittelwertbildung (5.14) konstant approximiert. Unter Verwendung des Integralsatzes von Gauß für die Kontrollvolumina R_l ergeben sich die approximativen Trilinearformen

$$\begin{aligned} \hat{n}_h^1(\mathbf{w}_h, \mathbf{c}_h, \mathbf{v}_h) &:= \sum_l \sum_k \int_{e_{lk}} (\mathbf{w}_h \cdot \mathbf{n}_{lk}) \mathbf{c}_h^{lk} \, d\Gamma \;\; F_{e_l}(\mathbf{v}_h) \quad \text{und} \\ \hat{n}_h^2(\mathbf{w}_h, \mathbf{c}_h, \mathbf{v}_h) &:= \sum_l \sum_k \int_{e_{lk}} (\mathbf{w}_h \cdot \mathbf{n}_{lk}) \, d\Gamma \;\; F_{e_l}(\mathbf{c}_h) F_{e_l}(\mathbf{v}_h) \end{aligned}$$

mit

$$\mathbf{c}_h^{lk} := \lambda_{lk} \, F_{e_l}(\mathbf{c}_h) + (1 - \lambda_{lk}) \, F_{e_k}(\mathbf{c}_h) \quad \text{wobei} \quad \lambda_{lk} = 1 - \lambda_{kl}.$$

Mit $k = k(l)$ sind dabei die Kanten indiziert, die die Kante e_l schneiden. Mit $\hat{n}_h := \hat{n}_h^1 - \hat{n}_h^2$ erhält man insgesamt:

$$\hat{n}_h(\mathbf{w}_h, \mathbf{c}_h, \mathbf{v}_h) \;=\; \sum_l \sum_k \int_{e_{lk}} \mathbf{w}_h \cdot \mathbf{n}_{lk}\, d\Gamma \;\; (1 - \lambda_{lk})\,(F_{e_k}(\mathbf{c}_h) - F_{e_l}(\mathbf{c}_h))F_{e_l}(\mathbf{v}_h) \qquad (5.21)$$

Das *einfache* Upwind-Verfahren ergibt sich durch folgende Wahl der Gewichtungsfaktoren

$$\lambda_{lk} \;:=\; \begin{cases} 1 & \text{falls } \int_{e_{lk}} \mathbf{w}_h \cdot \mathbf{n}_{lk}\, d\Gamma \geq 0 \\ 0 & \text{sonst} \end{cases}$$

In (5.18) ist nun n_h durch $\hat{n}_h$ zu ersetzen. Seien die nodale Basis von Q_h mit $\{p_h^j\}_{1 \leq j \leq m_h}$ und die von $\mathbf{V}_h$ mit $\mathcal{B}_h := \{\mathbf{v}_h^i\}_{1 \leq i \leq n_h}$ bezeichnet, d.h.

$$F_{e_j}(v_h^{i,l}) \;=\; \delta_{ij}\delta_{kl} \quad 1 \leq k \leq 3 \qquad (5.22)$$

wobei e_j alle Kanten bzw. Seiten der Triangulation durchläuft und mit $v_h^{i,l}$, $1 \leq l \leq 3$, die kartesischen Komponenten von $\mathbf{v}_h^i$ bezeichnet sind. Dann sind (5.18), (5.19) bei festgehaltener Konvektionsrichtung $\mathbf{w}_h$, vgl. dazu Abschnitt 5.9, äquivalent zu folgendem linearen Gleichungssystem:

$$\begin{pmatrix} S(\mathbf{w}_h) & B^T \\ B & 0 \end{pmatrix} \begin{pmatrix} c \\ p \end{pmatrix} \;=\; \begin{pmatrix} f \\ g \end{pmatrix}. \qquad (5.23)$$

Dabei ergibt sich die $n_h \times n_h$-Konvektionsmatrix $S(\mathbf{w}_h)$ aus

$$S(\mathbf{w}_h)_{ij} \;=\; \hat{n}_h(\mathbf{w}_h, \mathbf{v}_h^j, \mathbf{v}_h^i)$$

und die $m_h \times n_h$-Matrix B, die als Diskretisierung des Divergenzoperators Geschwindigkeiten und Druck koppelt, aus

$$B_{ij} \;=\; b_h(\mathbf{v}_h^j, p_h^i).$$

Der diskrete Gradientoperator ist gerade der transponierte Divergenzoperator. Wie für Finite Elemente Diskretisierungen üblich erfolgt die Assemblierung dieser Matrizen mit Hilfe lokaler Steifigkeitsmatrizen, vgl. Abschnitt 2.3.2.

5.2.3 Das diskrete Problem mit Nebenbedingungen

Unter Einbeziehung der Nebenbedingungen schreiben sich der diskrete Ansatz- bzw. Testraum für die Geschwindigkeiten wie folgt:

$$\begin{aligned}\tilde{\mathbf{H}}_h &:= \{\mathbf{c_h} \in \mathbf{H}_h : \mathbf{c_{h+}} = Q(\gamma)\mathbf{c_{h-}} \text{ auf } \Gamma_-, \mathbf{c_h} \cdot \mathbf{n_h} = \mathbf{u} \cdot \mathbf{n_h} \text{ auf } \Gamma_B\} \\ \tilde{\mathbf{V}}_h &:= \{\mathbf{v_h} \in \mathbf{V}_h : \mathbf{v_{h+}} = Q(\gamma)\mathbf{v_{h-}} \text{ auf } \Gamma_-, \mathbf{v_h} \cdot \mathbf{n_h} = 0 \text{ auf } \Gamma_B\}\end{aligned}$$

Mit $\hat{n}_h$ statt n_h lautet das diskrete Pendant zu (5.10), (5.11) **mit Periodizitäts- und Eulerschen Randbedingungen**:

Finde ein Paar $(\mathbf{c}_h, p_h) \in \tilde{\mathbf{H}}_h \times Q_h$, so daß

$$\hat{n}_h(\mathbf{w}_h, \mathbf{c}_h, \mathbf{v}_h) + b_h(\mathbf{v}_h, p_h) = 0 \qquad \text{für alle } \mathbf{v}_h \in \tilde{\mathbf{V}}_h \tag{5.24}$$

$$b_h(\mathbf{c}_h, q_h) = 0 \qquad \text{für alle } q_h \in Q_h \tag{5.25}$$

Wie bei Dirichletrandwerten wirkt sich die Inhomogenität der Eulerschen Randbedingung lediglich auf die rechte Seite aus, vgl. Abschnitt 5.6.2. Ohne Einschränkung kann man deshalb $\tilde{\mathbf{H}}_h$ durch $\tilde{\mathbf{V}}_h$ ersetzen. Mit einer Basis $\tilde{\mathcal{B}}_h := \{\tilde{\mathbf{v}}_h^i\}_{1 \le i \le \tilde{n}_h}$ von $\tilde{\mathbf{V}}_h$ sind (5.24), (5.25) bei festgehaltener Konvektionsrichtung $\tilde{\mathbf{w}}_h$, vgl. Abschnitt 5.9, dann äquivalent zu folgendem linearen Gleichungssystem:

$$\begin{pmatrix} \tilde{S}(\tilde{\mathbf{w}}_h) & \tilde{B}^T \\ \tilde{B} & 0 \end{pmatrix} \begin{pmatrix} \tilde{c} \\ p \end{pmatrix} = \begin{pmatrix} \tilde{f} \\ g \end{pmatrix}. \tag{5.26}$$

p_h in (5.24) ist dabei mit p_h in (5.18) genauso wenig identisch wie p in (5.26) mit p in (5.23). Analog zum vorherigen Abschnitt ergibt sich die $\tilde{n}_h \times \tilde{n}_h$-Konvektionsmatrix $\tilde{S}(\tilde{\mathbf{w}}_h)$ aus

$$\tilde{S}(\tilde{\mathbf{w}}_h)_{ij} = \hat{n}_h(\tilde{\mathbf{w}}_h, \tilde{\mathbf{v}}_h^j, \tilde{\mathbf{v}}_h^i)$$

und die $m_h \times \tilde{n}_h$-Matrix $\tilde{B}$ aus

$$\tilde{B}_{ij} = b_h(\tilde{\mathbf{v}}_h^j, p_h^i).$$

5.3 Vergleich der beiden diskreten Formulierungen

Das lineare Gleichungssystem (5.26) hat den Vorteil, daß alle Rand- sowie Nebenbedingungen eingebaut sind und deshalb zusätzliche Iterationen zur Auflösung der Periodizitäts- sowie Eulerschen

Randbedingungen nicht notwendig sind. Der wesentliche Nachteil liegt jedoch in der Struktur der Matrix $\tilde{S}$. Die Standarddiskretisierung, vgl. Abschnitt 5.2.2, mit $\mathbf{V}_h$ bzw. $\mathbf{H}_h$ statt $\tilde{\mathbf{V}}_h$ bzw. $\tilde{\mathbf{H}}_h$ führt auf eine Matrix S, vgl. (5.23), mit folgender Blockgestalt:

$$S = \begin{pmatrix} S_{uu} & -\omega M_l & 0 \\ \omega M_l & S_{vv} & 0 \\ 0 & 0 & S_{ww} \end{pmatrix}. \tag{5.27}$$

Die Matrix ist entsprechend den drei kartesischen Richtungen zerlegt. Die Diagonalblöcke, die die interne Kopplung zwischen Geschwindigkeitskomponenten in x-, y- bzw. z-Richtung repräsentieren, sind zudem identisch, d.h.

$$S_{uu} = S_{vv} = S_{ww}.$$

Geschwindigkeitskopplung zwischen u- und v-Komponenten tritt lediglich aufgrund des Coriolisterms auf und wird durch die Diagonalmatrix M_l erfaßt. M_l bezeichnet dabei die L^2-Massenmatrix, die durch übliche *lumping*-Techniken entsteht, vgl. z.B. GROSSMANN und ROOS [29].

Der gesamte Speicheraufwand bei dieser Art der Diskretisierung beschränkt sich demzufolge auf eine schwachbesetzte Konvektionsmatrix und einen Vektor. Bei der Matrix $\tilde{S}$ in (5.26) führen die periodischen sowie die Eulerschen Randbedingungen auf zusätzliche $u-v$–Kopplung. Letztere induzieren darüber hinaus auch $u-w$– und $v-w$–Kopplungen.

Der nun folgende Abschnitt beschäftigt sich mit der Frage, wie man die voll implizite Formulierung aller Rand- und Nebenbedingungen (5.26) zu einer effizienten Lösung ausnutzen kann, ohne dabei die erstrebenswerte Speicherstruktur (5.23) aufgeben zu müssen.

5.4 Diskrete Formulierung in faktorisierter Form

Die Aufgabe reduziert sich darauf, einen Zusammenhang zwischen den Matrizen bzw. Vektoren $\tilde{S}$, $\tilde{B}$ bzw. $\tilde{f}$ in (5.26) und S, B bzw. f in (5.23) herzustellen. Wesentlich dabei ist die Übergangsmatrix C, die die Einbettung von $\tilde{\mathbf{V}}_h$ in $\mathbf{V}_h$ bezüglich der Basis $\tilde{\mathcal{B}}_h$ von $\tilde{\mathbf{V}}_h$ und der nodalen Basis $\mathcal{B}_h$ von $\mathbf{V}_h$ darstellt. Die Konkretisierung dieser $n_h \times \tilde{n}_h$-Matrix im Falle periodischer bzw. Eulerscher Randbedingungen folgt in den Abschnitten 5.6.1 bzw. 5.6.2.

Mit Hilfe dieser Übergangsmatrix gelten nun analog zu (4.28) die Relationen

$$\begin{aligned} \tilde{S} &= C^T S C \\ \tilde{B} &= BC \\ \tilde{f} &= C^T f. \end{aligned}$$

Damit schreibt sich das lineare Gleichungssystem (5.26):

$$\begin{pmatrix} C^T S C & C^T B^T \\ BC & 0 \end{pmatrix} \begin{pmatrix} \tilde{c} \\ p \end{pmatrix} = \begin{pmatrix} C^T f \\ g \end{pmatrix}. \tag{5.28}$$

Im Gegensatz zu Abschnitt 4.3.1 setzt die Lösungsprozedur nun unmittelbar auf (5.28) auf. Der Vorteil dieser faktorisierten Darstellung liegt auf der Hand: Die Matrix S ist mit Standardmethoden assemblierbar und ihre Speicherstruktur ist die aus (5.27). Dennoch sind alle Rand- und Nebenbedingungen implizit enthalten. Ein Nachteil ist, daß die Geschwindigkeitsprobleme für u, v und w simultan gelöst werden müssen.

Ein weiterer besteht darin, daß die Matrix $C^T S C$ nicht in expliziter, sondern lediglich in faktorisierter Form vorliegt. Damit sind lineare Standardlöser wie Jacobi, Gauß-Seidel oder ILU nicht anwendbar. Sehr wohl anwendbar sind hingegen Löser, die ausschließlich auf Matrix-Vektor-Multiplikationen zurückgreifen. Da die Matrix S nicht symmetrisch ist, bietet sich BiCGStab als Iterationsverfahren an.

5.5 BiCGStab als Löser für die Geschwindigkeitsprobleme

In (5.28) fällt auf, daß die Tilde nur noch im Zusammenhang mit dem Lösungsvektor c auftritt. Der BiCGStab, ein Derivat des klassischen Verfahrens der konjugierten Gradienten für nichtsymmetrische Probleme, vgl. DAEHLEN und TVEITO [21], soll nun so formuliert werden, daß ausschließlich Vektoren der Länge n_h auftreten. Es wird sich zeigen, daß daraus ein BiCGStab-Verfahren für S resultiert mit CC^T als eine Art Vorkonditionierer.

Sei $\tilde{c}^0$ gegeben.
Berechne $\tilde{r}^0 = C^T f - C^T S C \tilde{c}^0$.
Wähle $\tilde{r}'$, so daß $\rho_0 = (\tilde{r}', \tilde{r}^0) \neq 0$, z.B. $\tilde{r}' = \tilde{r}^0$.
$\rho_0 = \alpha = \omega_0 = 1$.
$\tilde{v}^0 = \tilde{p}^0 = 0$.
Für $k = 1, 2, \ldots$ berechne bis zur Konvergenz

$$\begin{aligned}
\rho_k &= (\tilde{r}', \tilde{r}^{k-1}) \\
\beta &= \rho_k \alpha / \rho_{k-1} \omega_{k-1} \\
\tilde{p}^k &= \tilde{r}^{k-1} + \beta(\tilde{p}^{k-1} - \omega_{k-1} \tilde{v}^{k-1}) \\
\tilde{v}^k &= C^T S C \tilde{p}^k \\
\alpha &= \rho_k / (\tilde{r}', \tilde{v}^k) \\
\tilde{s} &= \tilde{r}^{k-1} - \alpha \tilde{v}^k \\
\tilde{t} &= C^T S C \tilde{s} \\
\omega_k &= (\tilde{t}, \tilde{s}) / (\tilde{t}, \tilde{t}) \\
\tilde{c}^k &= \tilde{c}^{k-1} + \alpha \tilde{p}^k + \omega_k \tilde{s} \\
\tilde{r}^k &= \tilde{s} - \omega_k \tilde{t}
\end{aligned}$$

Sei c^0 gegeben.
Berechne $r^0 = CC^T(f - Sc^0)$.
Wähle r', so daß $\rho_0 = (r', r^0) \neq 0$, z.B. $r' = r^0$.
$\rho_0 = \alpha = \omega_0 = 1$.
$v^0 = p^0 = 0$.
Für $k = 1, 2, \ldots$ berechne bis zur Konvergenz

$$\begin{aligned}
\rho_k &= (r', r^{k-1}) \\
\beta &= \rho_k \alpha / \rho_{k-1} \omega_{k-1} \\
p^k &= r^{k-1} + \beta(p^{k-1} - \omega_{k-1} v^{k-1}) \\
v^k &= CC^T S p^k \\
\alpha &= \rho_k / (r', v^k) \\
s &= r^{k-1} - \alpha v^k \\
t &= CC^T S s \\
\omega_k &= (t, s) / (t, t) \\
c^k &= c^{k-1} + \alpha p^k + \omega_k s \\
r^k &= s - \omega_k t
\end{aligned}$$

Der Algorithmus links ist der Standard-BiCGStab angewandt auf das lineare Gleichungssystem in der Form (5.28). Beim Übergang von links nach rechts können alle Tilden mit Hilfe der Übergangsmatrix C eliminiert werden:

$$x \quad := \quad C\tilde{x} \quad \text{für} \quad x \in \{c^k, r^k, v^k, p^k, r', s, t\}$$

Der hier definierte Vektor c ist nicht mit dem in (5.23) zu verwechseln. Außerdem ist zu beachten, daß

$$(C^T x, C^T y) \quad = \quad (CC^T x, y)$$

und, wenn CC^T eine Projektion ist,

$$(CC^T x, y) = ((CC^T)(CC^T)x, y) = (CC^T x, CC^T y).$$

Das heißt: Wird CC^T als Projektion konstruiert, sind beide Algorithmen äquivalent. Dabei ist der Begriff Projektion hier und im folgenden bezüglich des Euklidischen Skalarproduktes zu verstehen.

Zusammenfassend läßt sich feststellen, daß es auf diese Weise gelungen ist, die Zwangsbedingungen in den Löser einzuarbeiten, ohne daß dies Auswirkungen auf Diskretisierung oder Speicherstruktur hätte. Lediglich an drei Stellen innerhalb des BiCGStab-Lösers müssen Projektionen in den durch die Zwangsbedingungen definierten Unterraum vorgenommen werden.

5.6 Projektion für periodische und Eulersche Randbedingungen

Aufgrund eines fundamentalen Ergebnisses der linearen Algebra, vgl. KOECHER [40], handelt es sich bei CC^T um eine Projektion, wenn die Spalten der Matrix C eine Orthonormalbasis des durch die Zwangsbedingungen festgelegten Unterraums bilden. Die Konstruktion einer solchen Orthonormalbasis aus einer beliebigen Basis ist mit Hilfe des Schmidtschen Orthogonalisierungsverfahrens immer möglich. Bei periodischen und Eulerschen Randbedingungen liegen die Dinge besonders einfach.

5.6.1 Periodische Randbedingungen

Es reicht, ein Paar periodischer Punkte herauszugreifen. Gemäß (5.6) sind die Geschwindigkeitsvektoren $\mathbf{c}_+$ sowie $\mathbf{c}_-$ in kartesischen Koordinaten in folgender Weise miteinander korreliert:

$$\mathbf{c}_+ \quad = \quad Q(\gamma)\mathbf{c}_- \quad \text{mit} \quad Q(\gamma) = \begin{pmatrix} \cos(\gamma) & -\sin(\gamma) & 0 \\ \sin(\gamma) & \cos(\gamma) & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

Wenn man mit Hilfe dieser drei Zwangsbedingungen $\mathbf{c}_+$ eliminiert, ergibt sich für die Übergangsmatrix lokal eine 6×3-Matrix:

$$\frac{1}{\sqrt{2}} \cdot \begin{pmatrix} \cos(\gamma) & -\sin(\gamma) & 0 \\ \sin(\gamma) & \cos(\gamma) & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

Der Faktor $1/\sqrt{2}$ ist aus Normierungsgründen notwendig. Die Orthogonalität der Spalten dieser Matrix ist automatisch gewährleistet. Insgesamt werden hier sechs lokale Freiheitsgrade durch drei unabhängige Zwangsbedingungen auf drei generalisierte reduziert.

5.6.2 Euler Randbedingungen

Wieder genügt es, das Problem lokal zu betrachten. Der Geschwindigkeitsvektor $\mathbf{c}$ an einer Eulerseite läßt sich bezüglich eines lokalen $(\mathbf{n}, \mathbf{t}_1, \mathbf{t}_2)$-Koordinatensystems folgendermaßen entwickeln

$$\mathbf{c} \quad = \quad c_{t_1} \cdot \mathbf{t}_1 + c_{t_2} \cdot \mathbf{t}_2 + c_n \cdot \mathbf{n}$$

vorausgesetzt, das Koordinatensystem ist orthonormiert. Dabei steht $\mathbf{n}$ senkrecht auf der Fläche, und c_{t_1} bzw. c_{t_2} sind die Koordinaten in der Tangentialebene.

Wegen $c_n = \mathbf{c} \cdot \mathbf{n} = \mathbf{u} \cdot \mathbf{n}$, handelt es sich hier um einen affinen Unterraum, falls $\omega \neq 0$. Dies bringt jedoch keinerlei grundsätzliche Schwierigkeiten mit sich. Wie der Volumenstrom in (4.29), (4.30) induziert der affine Anteil einen zusätzlichen Term auf der rechten Seite des Gleichungssystems (5.28) und die Spalten der lokalen 3×2-Übergangsmatrix sind gegeben durch die Tangentialvektoren $\mathbf{t}_1$ und $\mathbf{t}_2$.

5.6.3 Die divergenzfreie Basis

Eine weitere Anwendung der Projektionsstrategie besteht darin, mit einer diskret divergenzfreien Basis zu rechnen. Für den zweidimensionalen Fall soll diese Technik im folgenden untersucht werden. Dabei wird eine Funktion $\mathbf{v}_h \in \mathbf{V}_h$ als diskret divergenzfrei bezeichnet, wenn sie im Kern der Matrix B liegt, d.h. wenn

$$b_h(\mathbf{v}_h, q_h) \quad = \quad 0 \quad \text{für alle} \quad q_h \in Q_h$$

oder mit der Notation (5.20)

$$Q_\tau(\mathbf{v}_h) \quad = \quad 0 \quad \text{für alle} \quad \tau \in \mathcal{T}_h.$$

Das bedeutet, daß für jede Zelle die Summe der Volumenströme über die vier Kanten verschwindet. Sei $\mathcal{B}_\tau = \{\psi_\tau^i : 1 \le i \le 4\}$ die bezüglich der Freiheitsgrade (5.14) lokale nodale Basis, die durch $F_{e_j}(\psi_\tau^i) = \delta_{ij}$, $1 \le j \le 4$, charakterisiert ist. Für ein beliebiges Viereck $\tau \in \mathcal{T}_h$ seien dabei die Kanten mit e_j, die äußeren Einheitsnormalenvektoren mit $\mathbf{n}_i$ und die zugehörigen Einheitstangentialvektoren mit $\mathbf{t}_i$, vgl. Abbildung 5.4, bezeichnet.

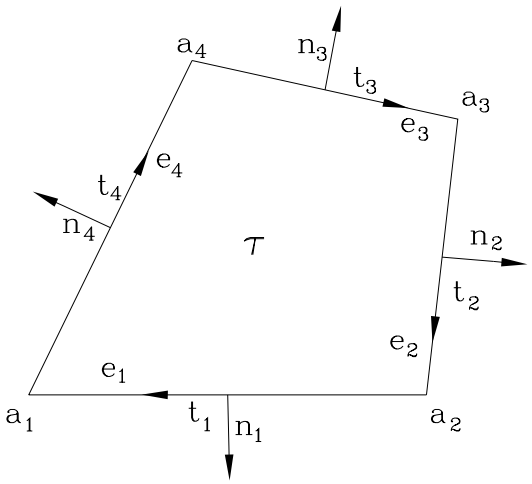

Abb. 5.4: Bezeichnungen für ein beliebiges Viereck

Es werden nun zwei Funktionensysteme definiert, die sich lokal darstellen lassen als

$$\psi_\tau^i \mathbf{t}_i \quad \text{für} \quad 1 \le i \le 4 \tag{5.29}$$

bzw.

$$\frac{\psi_\tau^k}{|e_k|}\mathbf{n}_k - \frac{\psi_\tau^j}{|e_j|}\mathbf{n}_j \quad \text{für} \quad 1 \le j \le 4,\ k = (j+2) \bmod 4 + 1. \tag{5.30}$$

Das Konstruktionsprinzip ist einfach: Die erste Gruppe von Funktionen ist mit den Kanten assoziiert und erzeugt keinen Volumenstrom über die jeweilige Kante. Die zweite Gruppe ist mit den Ecken assoziiert und balanziert die Volumenströme über die in dieser Ecke zusammenlaufenden Kanten so aus, daß für die angrenzenden Zellen das Prinzip der diskreten Volumenerhaltung erfüllt ist, vgl. Abbildung 5.5.

Was die physikalische Interpretation angeht, so approximieren die nodalen Werte bezüglich dieser Funktionen im Falle von (5.29) die Tangentialgeschwindigkeit auf der jeweiligen Kante und im Falle von (5.30) den Stromfunktionswert in der assoziierten Ecke, vgl. Turek [74]. Dreidimensionale Verallgemeinerungen sind möglich. Näheres dazu ist in Thomasset [72] nachzulesen.

Abhängig von den Randbedingungen und davon, ob das zugrundeliegende Gebiet einfach zusammenhängend ist, liefert eine leichte Modifikation dieser Funktionensysteme eine Basis des Kerns von $b_h(.,.)$, vgl. Turek [74], Thomasset [72].

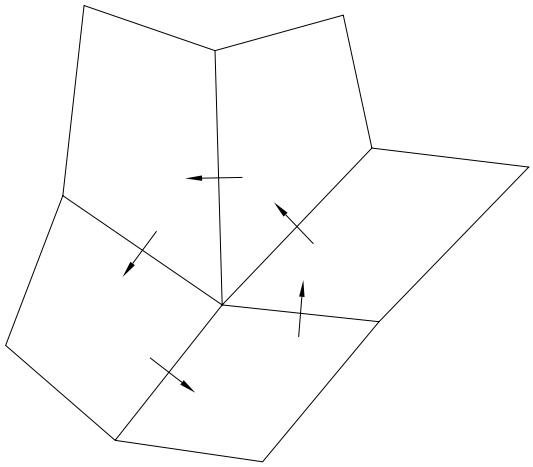

Abb. 5.5: Lokale Volumenerhaltung für das aus 5 Zellen bestehende Makroelement

Die Übergangsmatrix C erhält man, wenn man die divergenzfreie Basis (5.29), (5.30) bezüglich der nodalen Basis $\mathcal{B}_h$ des gesamten Raums darstellt. Es ergibt sich auch hier ein Gleichungssystem der Form (5.28), wobei nach Konstruktion $BC = 0$. Die Einführung einer diskret divergenzfreien Basis ermöglicht es somit, die Kontinuität einschließlich des hydrostatischen Druckes zu eliminieren. Übrig bleibt ein Gleichungssystem ausschließlich für die Geschwindigkeiten:

$$C^T S C \tilde{c} \;=\; C^T f \tag{5.31}$$

Ist die Geschwindigkeitslösung aus (5.31) bekannt, läßt sich der statische Druck ohne Lösung eines zusätzlichen Gleichungssystems ermitteln, vgl. Turek [74]. Dies entspricht der Technik in Abschnitt 4.3.2, wo der diskrete Lagrange-Parameter, der im wesentlichen die Tangentialgeschwindigkeit entlang der Schaufel approximiert, in einem analogen Postprocessing-Schritt bestimmt wird.

Abgesehen davon, daß die Verwendung einer divergenzfreien Basis eine sehr elegante Separation von Geschwindigkeit und Druck zuläßt, hat sie den wesentlichen Vorteil, daß sie auf ein Gleichungssystem mit definiter Matrix führt. Sattelpunktlöser wie der SIMPLE, vgl. Abschnitt 5.8, sind damit nicht notwendig. Ein weiterer Vorteil liegt im geringeren Speicheraufwand im Vergleich zur Standardformulierung.

Der entscheidende Nachteil ist, daß es sich bei (5.29), (5.30) nicht um eine Orthonormalbasis handelt. Eine solche zu konstruieren ist mit Hilfe des Schmidtschen Orthogonalisierungsverfahrens, vgl. z.B. Meyberg [48], weder aus mathematischer noch aus rechentechnischer Sicht ein Problem. Die daraus resultierende Basis ist jedoch von nichtlokaler Natur und die Projektion in den Raum der divergenzfreien Geschwindigkeitsfelder demzufolge zu aufwendig.

Dies beeinträchtigt das Laufzeitverhalten eines Mehrgitterverfahrens massiv, da eine derartige Projektion nach jedem Prolongations- sowie Restriktionsschritt durchzuführen ist. Da andererseits die Kondition der Matrix $C^T SC$ im Gegensatz zu (4.40) mit $\mathcal{O}(h^{-4})$ wächst, ist eine Mehrgitterstra-

tegie unverzichtbar. TUREK [74] beschreibt zwar eine lokalisierte Version der Gittertransferoperatoren für den zweidimensionalen Fall. Die angegebenen Entwicklungszeiten schrecken aber davon ab, dies auch im Dreidimensionalen zu versuchen.

5.7 Bemerkungen zur Projektionsstrategie

Die in Anschnitt 5.4 entwickelte Strategie zur Behandlung von Nebenbedingungen ist relativ universell einsetzbar. Die physikalische Situation ist dabei diejenige, die aus der *Lagrange-Mechanik* bekannt ist, vgl. z.B. PFEIFFER [53]. Die kinematischen *Neben-* oder — wie sie dort genannt werden — *Zwangsbedingungen* definieren einen Unterraum. Daß die Bewegung diesen Unterraum respektiert, wird durch eine Kraft erzwungen. Diese *Zwangskraft* ist aus mathematischer Sicht ein Lagrange-Parameter, mit dessen Hilfe die Nebenbedingung angekoppelt wird.

Nehmen wir als Beispiel den Fall Eulerscher Randbedingungen: Der mit der Bedingung $\mathbf{w} \cdot \mathbf{n} = 0$ entlang der Schaufel assoziierte Lagrange-Parameter ist der statische Druck auf der Schaufeloberfläche, der sich gemäß obiger Interpretation so einstellt, daß die kinematische Zwangsbedingung erfüllt ist. Im diskreten Kontext kann man nun eine Basis des durch die Bedingung $\mathbf{w}_h \cdot \mathbf{n}_h = 0$ für Kanten bzw. Seiten auf der Schaufelfläche festgelegten Unterraums explizit angeben, vgl. Abschnitt 5.6.2, und so die Nebenbedingung einschließlich des sie erzwingenden Lagrange-Parameters eliminieren.

Es ergibt sich ein reduziertes System mit einer geringeren Anzahl von Unbekannten, in der klassischen Terminologie als *Minimalkoordinaten* oder *generalisierte Koordinaten* bezeichnet. Dies entspricht den *Lagrange-Gleichungen zweiter Art*, wohingegen das ursprüngliche System ohne Elimination der Zwangsbedingungen gerade die *Lagrange-Gleichungen erster Art* darstellt.

Ist nun das reduzierte Problem gelöst, läßt sich anschließend mit Hilfe der Lösung die Zwangskraft ermitteln. Für den Fall der Eulerschen Randbedingung bedeutet dies analog zu Abschnitt 4.3.2 eine Alternative zur Berechnung des Druckes entlang der Schaufel, die ohne Extrapolation auskommt und optimale Ordnung garantiert, vgl. GUNZBURGER [30].

Zur weiteren Verdeutlichung soll dieser Gedankengang an einem besonders einfachen Beispiel, dem Fadenpendel, nachvollzogen werden, vgl. Abbildung 5.6. Mit der kinematischen Vorgabe $x^2 + y^2 = R^2$ und einer Transformation auf Polarkoordinaten gelingt es, die Fadenkraft als Zwangskraft zu eliminieren. Nach dem Lösen der Bewegungsgleichung in azimutaler Richtung mit φ als generalisierter Koordinate, kann man die Fadenkraft aus dem radialen Gleichgewicht berechnen.

Die vorgestellte Strategie funktioniert im Kern analog. Was noch hinzukommt, ist die Tatsache, daß aus Gründen, die in Abschnitt 5.3 ausführlich erläutert wurden, nicht mit der Minimalbasis direkt, sondern mit der nodalen Basis gerechnet wird und an genau definierten Stellen eine Projektion in den durch die Zwangsbedingungen beschriebenen Unterraum erfolgt.

Daß die Strategie ihre Grenzen hat, zeigt sich in Abschnitt 5.6.3. Für Zwangsbedingungen jedoch, die lokaler Natur sind, bietet Abschnitt 5.4 in Verbindung mit 5.5 einen Ansatz, der die Nebenbedingungsproblematik reduziert auf ein einfaches Problem der linearen Algebra: Projektion in einen Unterraum.

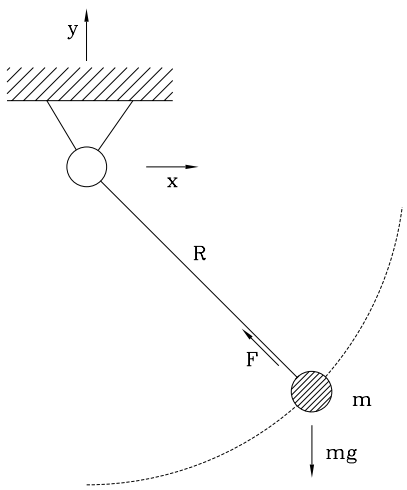

Abb. 5.6: Fadenpendel

5.8 Der erweiterte SIMPLE-Algorithmus

Die Lösung des Gleichungssystems (5.28) ist insofern problematisch, als die Matrix indefinit ist, also sowohl positive als auch negative Eigenwerte hat. Ein Lösungsansatz besteht darin, Druck und Geschwindigkeiten voneinander zu entkoppeln, indem die diskrete Impulsgleichung nach der unbekannten Geschwindigkeit aufgelöst und in die Druckgleichung eingesetzt wird. Man erhält dann:

$$\begin{pmatrix} C^T S C & C^T B^T \\ 0 & BC(C^T SC)^{-1}(BC)^T \end{pmatrix} \begin{pmatrix} \tilde{c} \\ p \end{pmatrix} = \begin{pmatrix} C^T f \\ BC(C^T SC)^{-1} C^T f - g \end{pmatrix}. \tag{5.32}$$

Bindet man diese Strategie in ein Defekt-Korrektur-Verfahren ein, vgl. Abschnitt 5.9, so spricht man gewöhnlich von einem *Druckkorrekturschema*. Der wesentliche Vorteil der Herleitung im diskreten Kontext im Vergleich zur kontinuierlichen Formulierung nach Chorin [18], Temam [71] oder Van Kan [75] besteht darin, daß weder die Diskretisierung des zugrundeliegenden Poisson-Problems für den Druck noch die Neumannschen Randbedingungen für dieses Problem explizit spezifiziert werden müssen. Beides ergibt sich durch rein algebraische Manipulationen automatisch.

Vom algebraischen Standpunkt aus handelt es sich demnach bei der Herleitung von (5.32) um Block-Gaußelimination, eine Technik, die sich weit über inkompressible Strömungen hinaus zur Lösung der Lagrange-Gleichungen erster Art bewährt hat, vgl. z.B. Pfeiffer [53].

Im Rahmen des SIMPLE-Algorithmus wird nun die Inverse von $C^T SC$ in (5.32) durch die Inverse des Diagonalanteils ersetzt. Einfach nur die Diagonale von S zu nehmen, reicht dabei nicht aus. Als nützlich erweist sich der Ansatz

$$C^T D^{-1} C \quad = \quad (C^T \mathrm{diag}(S) C)^{-1} \tag{5.33}$$

mit einer zu bestimmenden Diagonalmatrix D. Offensichtlich gehen Diagonalelemente von S, die nicht mit dem periodischen oder Eulerrand assoziiert sind, wie bei SIMPLE auch, unverändert in D ein. Setzt man die Übergangsmatrix aus Abschnitt 5.6.2 in den Ansatz (5.33) ein, so sieht man, daß dies auch für Freiheitsgrade auf Eulerseiten gilt. Lediglich periodische Freiheitsgrade induzieren in D einen Eintrag von $(s_+ + s_-)/2$, wenn s_+ und s_- die entsprechenden Diagonaleinträge von S sind.

Setzt man (5.33) für $(C^T S C)^{-1}$ in (5.32) ein, so ergibt sich als Druckmatrix

$$BC(C^T D^{-1} C)(BC)^T \quad = \quad B(CC^T) D^{-1} (CC^T) B^T. \tag{5.34}$$

Dieses Ergebnis läßt sich folgendermaßen interpretieren: Nach Anwendung des Gradient- und vor Anwendung des Divergenzoperators wird das Geschwindigkeitsfeld jeweils projeziert. Ansonsten bleibt der SIMPLE unverändert.

Der Projektionsanteil CC^T wird mit der Matrix B assembliert. Nur so gelingt es, die Matrix (5.34) explizit auszuwerten, was hinsichtlich der Effizienz des Gesamtverfahrens wesentlich ist. Wie die Ergebnisse in Kapitel 6 zeigen, funktioniert der SIMPLE in dieser Konfiguration als Glätter innerhalb eines nichtlinearen Mehrgitterverfahrens für einen weiten Anwendungsbereich ausgezeichnet.

5.9 Das nichtlineare Iterationsverfahren

Als nichtlineares Iterationsverfahren bietet sich ein Quasi-Newton-Verfahren an. *Quasi*-Newton deshalb, weil nicht die gesamte Frechet-Ableitung, vgl z.B. Bronstein [17], sondern nur ein Teil davon Verwendung findet, vgl. Turek [74]. In der Praxis heißt das, daß in jedem nichtlinearen Iterationsschritt ein lineares Konvektionsproblem von der Gestalt (5.1)–(5.7) mit gegebener Konvektionsrichtung approximativ gelöst wird. Die momentane Konvektionsrichtung entspricht dabei dem im vorherigen Iterationsschritt berechneten Geschwindigkeitsfeld. Wie das klassische Newton-Verfahren ist das nichtlineare Iterationsverfahren als *Defekt-Korrektur-Verfahren* organisiert. Im einzelnen heißt das:

1. Berechne das aktuelle Residuum.

$$\begin{pmatrix} \mathrm{res}_c \\ \\ \mathrm{res}_p \end{pmatrix} \quad = \quad \begin{pmatrix} f - S(w^n) c^n - B^T p^n \\ \\ g - B c^n \end{pmatrix} \tag{5.35}$$

2. Löse approximativ ein lineares Konvektionsproblem mit dem Residuum als rechter Seite.

$$\begin{pmatrix} C^T S(w^n) C & C^T B^T \\ BC & 0 \end{pmatrix} \begin{pmatrix} \Delta\tilde{c} \\ \Delta p \end{pmatrix} = \begin{pmatrix} C^T \mathrm{res}_c \\ \mathrm{res}_p \end{pmatrix} \tag{5.36}$$

3. Korrigiere die vorhergehende Iterierte.

$$\begin{pmatrix} c^{n+1} \\ p^{n+1} \end{pmatrix} = \begin{pmatrix} c^n \\ p^n \end{pmatrix} + \begin{pmatrix} \Delta c \\ \Delta p \end{pmatrix} \quad \text{mit} \quad \Delta c = C\Delta\tilde{c} \tag{5.37}$$

Details zur Organisation des darauf basierenden nichtlinearen Mehrgitterverfahrens sind bei Müller and Szilagyi [51] nachzulesen. Prolongation und Restriktion sind so implementiert, wie dies von Turek [74] vorgeschlagen wird. Die bisher durchgeführten Tests zeigen, daß die Konvergenzraten nicht vom Verfeinerungslevel abhängen und für Laufradrechnungen im Bereich von 0.2–0.3, für Leitradrechnungen sogar im Bereich 0.1–0.2 liegen. Die proportionale Abhängigkeit der Rechenzeiten von der Zahl der Unbekannten wird durch die Testrechnungen bestätigt, vgl. Kapitel 6.

Kapitel 6

Ergebnisse

In diesem Kapitel sollen die theoretischen Resultate der Kapitel 4 und 5 anhand numerischer Testrechnungen verifiziert werden. Zunächst geht es um Ergebnisse bezüglich des zweidimensionalen Potentialverfahrens, anschließend um dreidimensionale Euler-Rechnungen.

6.1 Potentialverfahren

Zweidimensionale Potentialverfahren spielen heute im Vergleich zu dreidimensionalen Rechnungen eine untergeordnete Rolle. Sie können allerdings dazu dienen, die in Kapitel 4 vorgestellten Techniken anhand geeigneter Beispiele zu verifizieren. Entscheidend ist, daß sich beispielweise die alternative Methode zur Berechnung der Druckverteilung entlang der Schaufel im einfacheren Kontext besser untersuchen und anschließend auf komplexere Modelle verallgemeinern lassen.

6.1.1 Busemann-Gitter

Das Busemann-Gitter ist durch logarithmisch-spiralige unendlich dünne Schaufeln charakterisiert, vgl. Abbildung 2.2. Es handelt sich um ein rein radiales Schaufelgitter. Die $m-\varphi$-Koordinaten stimmen demzufolge mit Polarkoordinaten überein. Wie bei SCHILLING [62] nachzulesen ist, erscheinen die Schaufeln in konformer Abbildung als Geraden mit Steigung $\tan\beta$, in Polarkoordinaten als Exponentialkurven $r(\varphi) = \exp(-(\tan\beta)\varphi)$, wobei β den konstanten Schaufelwinkel bezeichnet. Zu- und Nachlauf ergeben sich als deren Verlängerung.

Für eine Schichtdicke

$$\delta(r) \;:=\; \frac{1}{\omega r}\cdot\begin{cases}\tilde{\delta}(r)+3(r-r_1)^2 & \text{falls } r_0\le r\le r_1\\ \tilde{\delta}(r) & \text{falls } r_1\le r\le r_2\\ \tilde{\delta}(r)+3(r-r_2)^2 & \text{falls } r_2\le r\le r_3\end{cases} \tag{6.1}$$

mit $\tilde{\delta}(r) := \frac{Q}{\gamma r\tan\beta}$, wobei mit Q der Volumenstrom, mit $\gamma := 2\pi/Z$ der Teilungswinkel zur Schaufelzahl Z und mit ω die Winkelgeschwindigkeit bezeichnet sind, ist die Lösung des Problems (4.1)

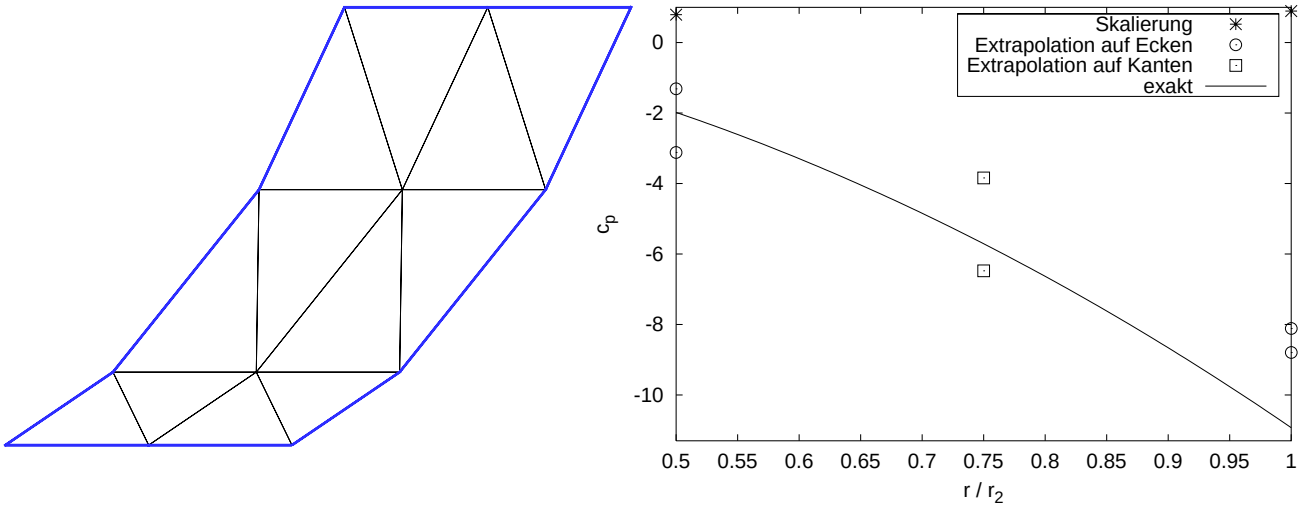

Abb. 6.1: Initiales Gitter und Druckverteilung auf Level 0

in Verbindung mit den Randbedingungen (4.7)–(4.11) unter der Annahme drallfreier Zuströmung gegeben durch die Stromfunktion:

$$\psi(r,\varphi) \quad = \quad \begin{cases} \tilde{\psi}(r,\varphi) + (r-r_1)^3 & \text{falls } r_0 \le r \le r_1 \\ \tilde{\psi}(r,\varphi) & \text{falls } r_1 \le r \le r_2 \\ \tilde{\psi}(r,\varphi) + (r-r_2)^3 & \text{falls } r_2 \le r \le r_3 \end{cases} \tag{6.2}$$

mit $\tilde{\psi}(r,\varphi) := \frac{Q}{\gamma}(\varphi + \frac{\ln r}{\tan\beta})$.

Zweifelsohne handelt es sich hierbei um einen rein akademischen Testfall. Die Umlenkung ist null, es wird keine Energie umgesetzt. Beispiele dieser Art eignen sich jedoch hervorragend, um Codes im Hinblick auf Programmierfehler zu untersuchen und Approximationsordnungen zu überprüfen. Dies bezieht sich beispielsweise auf die in Abschnitt 4.3.2 vorgeschlagene Methode zur Berechnung des statischen Drucks entlang der Schaufel.

Gerechnet wurde das Problem für $\beta = 60°$, $Q = 1m^3/s$ und $\omega = 1/s$. Die Schaufelzahl ist $Z = 8$, und die Radien sind gegeben durch: $r_0 = 0.3m$, $r_1 = 0.5m$, $r_2 = 1m$ und $r_3 = 1.5m$. Das Laufrad wird von innen nach außen durchströmt und dreht in positive Umfangsrichtung. In den Abbildungen 6.1–6.5 sind für feiner werdendes Netz, das jeweils in Polarkoordinaten dargestellt ist, die Ergebnisse verschiedener Strategien zur Druckberechnung mit der exakten Lösung verglichen.

Offensichtlich liefert die Skalierungsvariante gemäß (4.32) die besten Ergebnisse. Die beiden anderen Techniken basieren auf Extrapolation, wobei in dem einen Fall der Druckwert in einer Ecke durch gewichtete Mittelung der Drücke in den angrenzenden Dreiecken berechnet wird. Im zweiten Fall wird jeder Kante der Druck der angrenzenden Zelle zugeordnet. Aus Gründen der Übersichtlichkeit wurde in Abbildung 6.5 auf die Darstellung der extrapolierten Werte in den Ecken verzichtet. Was die Qualität der Approximation angeht, sind zwischen den beiden Extrapolationsstrategien keine wesentlichen Unterschiede festzustellen.

Interessant ist das Verhalten der Skalierungslösung an der Vorder- und Hinterkante der Schaufel,

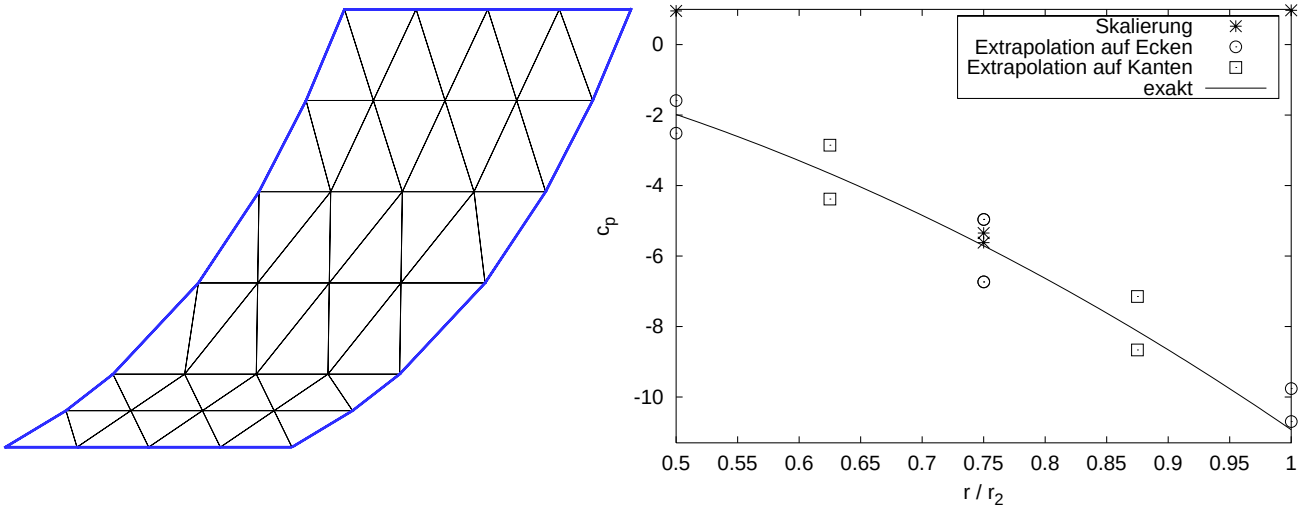

Abb. 6.2: Uniform verfeinertes Gitter und Druckverteilung auf Level 1

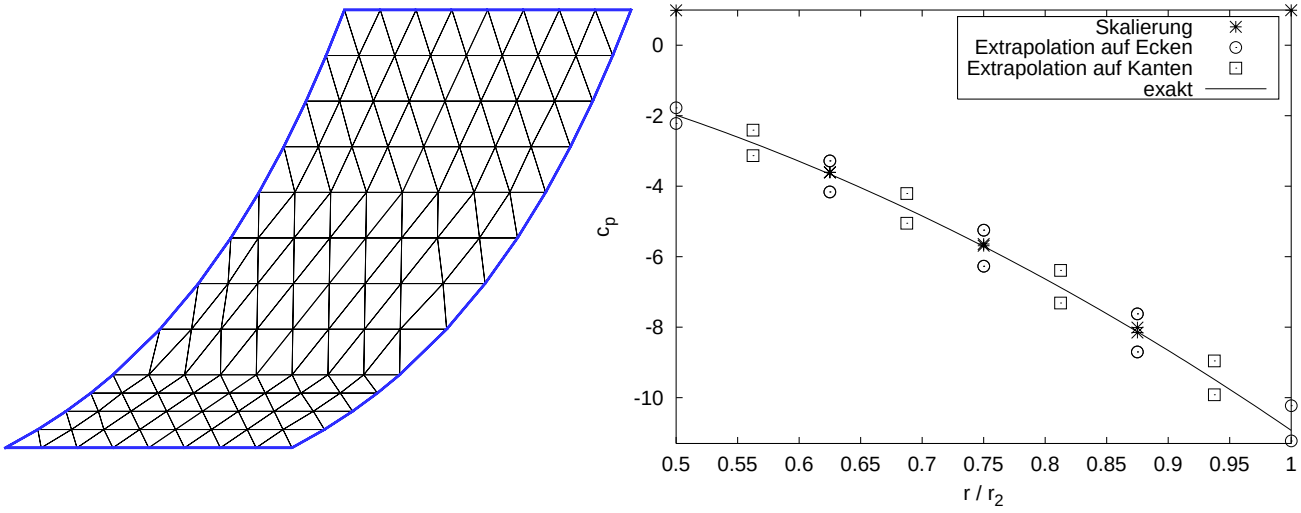

Abb. 6.3: Uniform verfeinertes Gitter und Druckverteilung auf Level 2

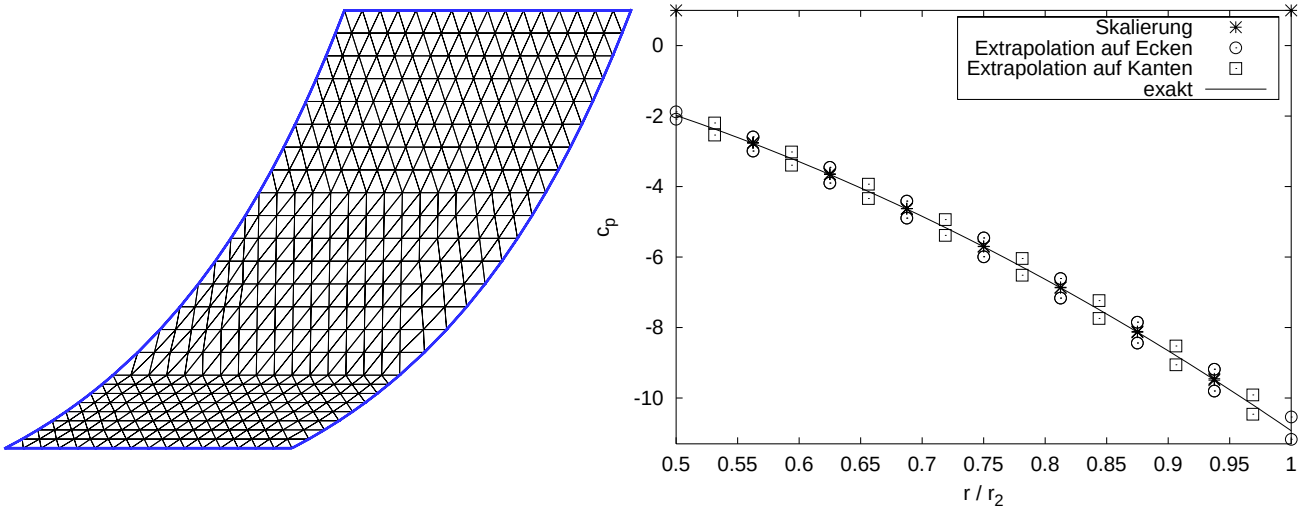

Abb. 6.4: Uniform verfeinertes Gitter und Druckverteilung auf Level 3

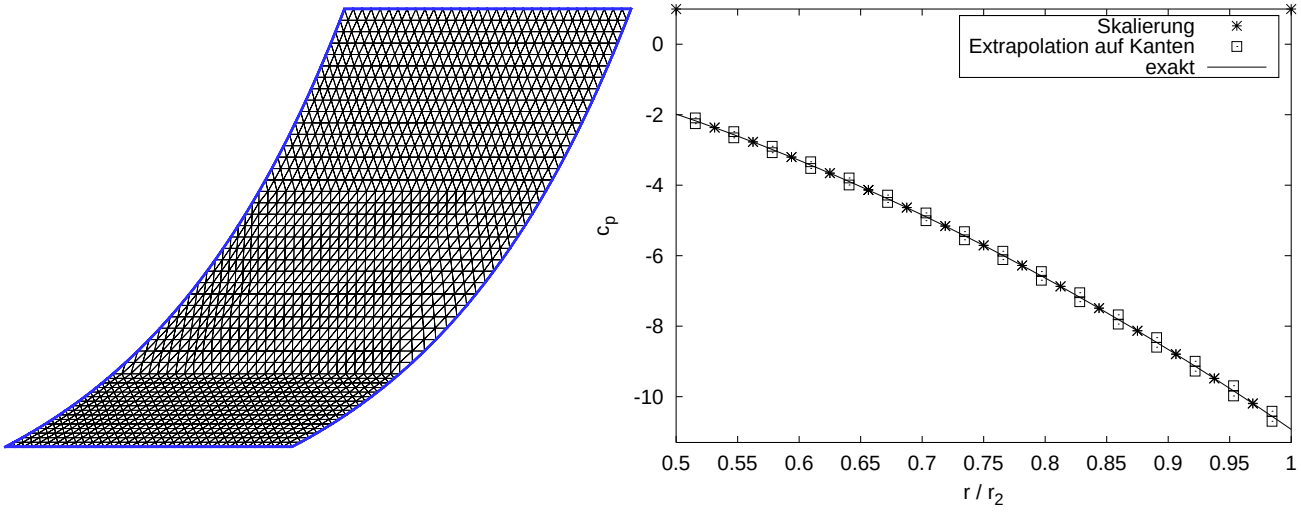

Abb. 6.5: Uniform verfeinertes Gitter und Druckverteilung auf Level 4

wo der Normalenvektor nicht definiert ist. Aufgrund der stoßfreien Anströmung bzw. der Kutta-Bedingung stellt sich an beiden Punkten als Normalkomponente des Flusses 0 ein, und es bilden sich Staupunkte aus. Dasselbe Verhalten zeigt sich an der Hinterkante des Gostelow-Profils, vgl. Abschnitt 6.1.2.

Daß der statische Druck entlang der Schaufel abfällt, ist zunächst nicht einsichtig. Schließlich wird das Rad von innen nach außen durchströmt, der Radius nimmt also zu. Der Grund ist, daß die Schichtdicke nach (6.1) im Schaufelbereich proportional zu $\frac{1}{r^2}$ und die Querschnittsfläche A demzufolge mit $\frac{1}{r}$ abnehmen. Die radiale Geschwindigkeit $w_r = \frac{Q}{A}$ nimmt dann linear mit r zu. Wegen $w_\varphi = -\omega r$ bedeutet dies, daß

$$c_p = 1 - \frac{w^2}{w_\infty^2} \tag{6.3}$$

quadratisch fällt. Für dieselbe Hierarchie uniform verfeinerter Netze sind in Tabelle 6.1 die Abweichungen der numerischen Näherungslösungen von der exakten Lösung (6.2) bezüglich der $L^2(\Omega)$- und der $H^1(\Omega)$-Norm aufgelistet.

Level	Zellen	$\|\psi - \psi_h\|_{L^2(\Omega)}$	$\|\psi - \psi_h\|_{H^1(\Omega)}$
0	12	0.048	0.276
1	48	0.0105	0.12
2	192	0.00214	0.059
3	768	0.00048	0.029
4	3072	0.00012	0.014

Tab. 6.1: Approximationsfehler bei uniformer Verfeinerung

Die Abschätzung in der $H^1(\Omega)$-Norm ist deshalb von besonderem Interesse, weil neben Funktionswerten auch Ableitungen berücksichtigt werden und letztere wegen (4.5), (4.6) direkt in die Berechnung der Geschwindigkeiten und damit auch des Druckes nach (6.3) eingehen. Konvergenzverhalten von der Ordnung 2 in der $L^2(\Omega)$- und von der Ordnung 1 in der $H^1(\Omega)$-Norm bestätigen die Konvergenztheorie stückweise linearer Elemente, vgl. z.B. Braess [11].

Das Resultat zeigt, daß — zumindest für diesen Testfall — durch die Manipulationen, die beispielsweise im Zusammenhang mit der Kuttabedingung notwendig sind, die Approximation der Stromfunktion mit Ordnung 2 nicht beeinträchtigt wird. Darüber hinaus werden die Geschwindigkeiten mit Ordnung 1 und damit der Druck gemäß (6.3) mit Ordnung 2 approximiert.

6.1.2 Gostelow-Gitter

Das Gostelow-Gitter ist eigentlich ein (in kartesischen Koordinaten) ebenes Gitter. Um es mit dem Potentialverfahren aus Kapitel 4 rechnen zu können, wird es als rein axiales Schaufelgitter mit dem Referenzradius $r_{ref} = 0.2m$ behandelt. Zur Parametrisierung der Stromfläche werden $z - \varphi$- bzw. konforme $z - r_{ref}\varphi$-Koordinaten verwendet.

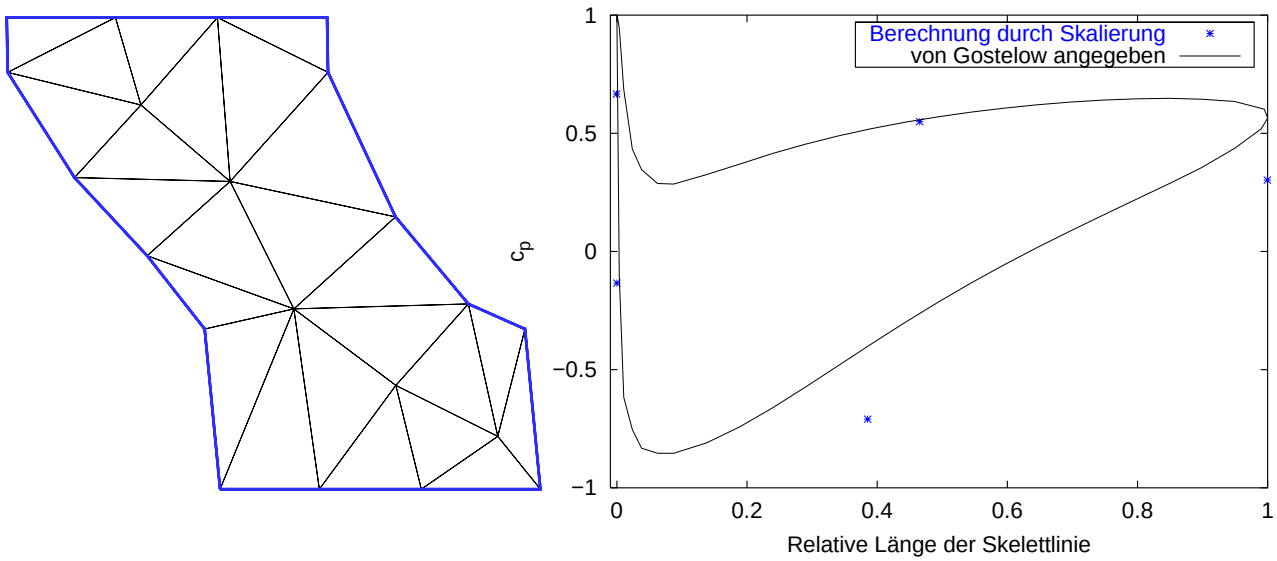

Abb. 6.6: Initiales Gitter und Druckverteilung auf Level 0

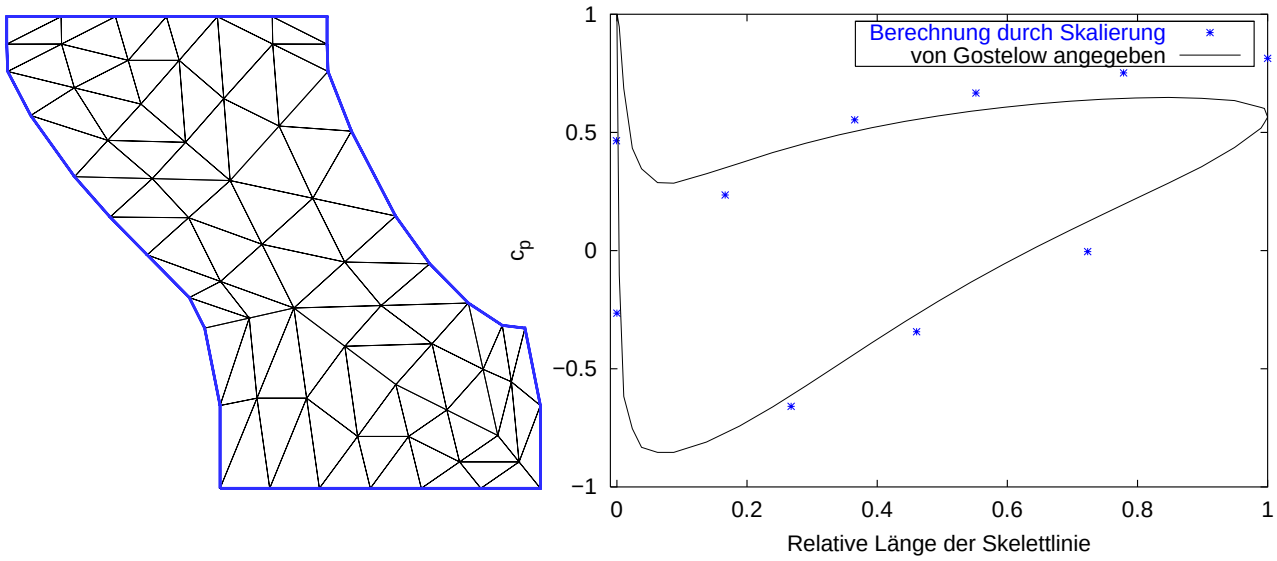

Abb. 6.7: Uniform verfeinertes Gitter und Druckverteilung auf Level 1

Ziel der Berechnungen ist es, zum einen die von GOSTELOW [26] angegebene Druckverteilung zu reproduzieren und zum anderen das Verhalten des in Abschnitt 4.6 analysierten Fehlerschätzers zu untersuchen. Die Eingabedaten sind gegeben durch: $Z = 10$, $Q = 0.0251327 m^3/s$ und $\omega = -67.571125/s$.

Den Abbildungen 6.6–6.11 sind die Entwicklung des Gitters in konformen Koordinaten bei uniformer Verfeinerung sowie die zugehörigen Druckverteilungen im Vergleich zur gegebenen zu entnehmen.

Zur Druckberechnung entlang der Schaufel wurde die Skalierungsmethode nach Abschnitt 4.3.2

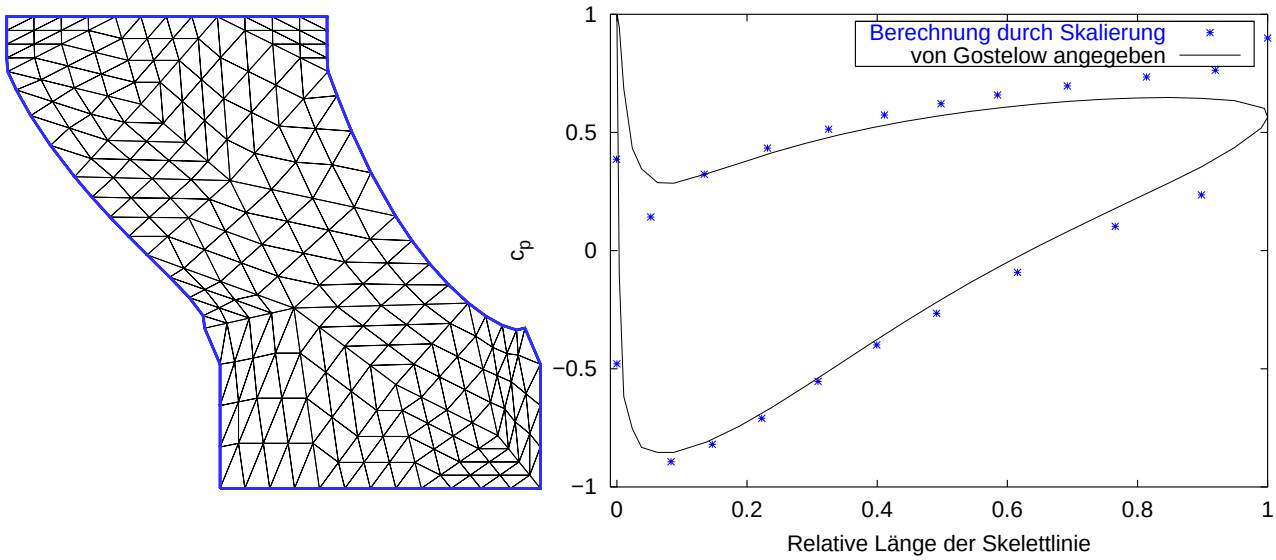

Abb. 6.8: Uniform verfeinertes Gitter und Druckverteilung auf Level 2

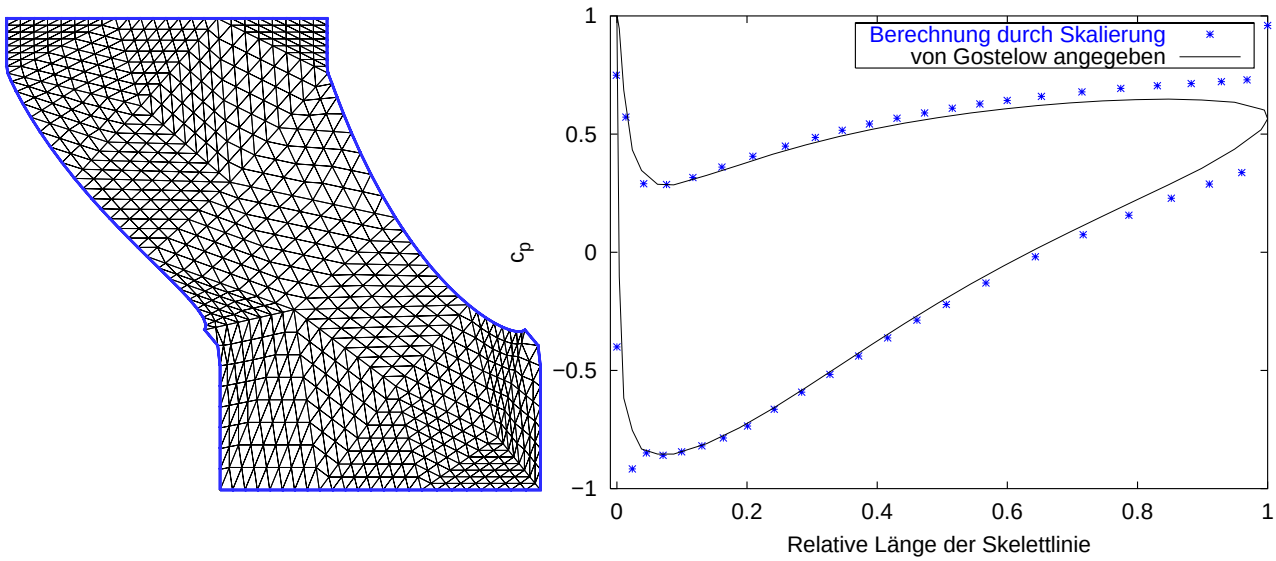

Abb. 6.9: Uniform verfeinertes Gitter und Druckverteilung auf Level 3

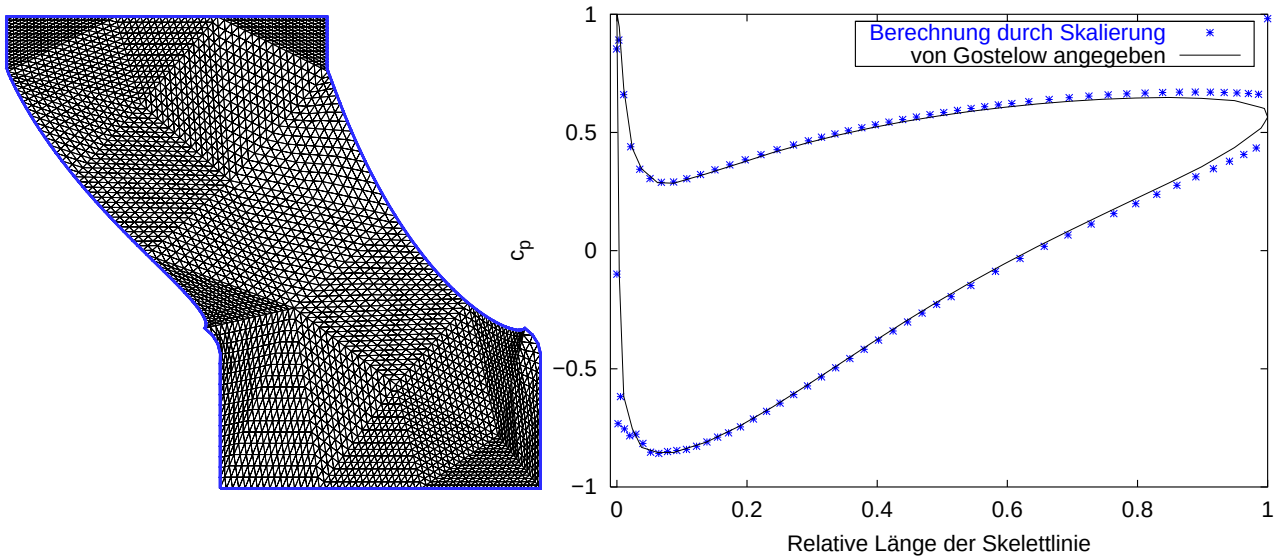

Abb. 6.10: Uniform verfeinertes Gitter und Druckverteilung auf Level 4

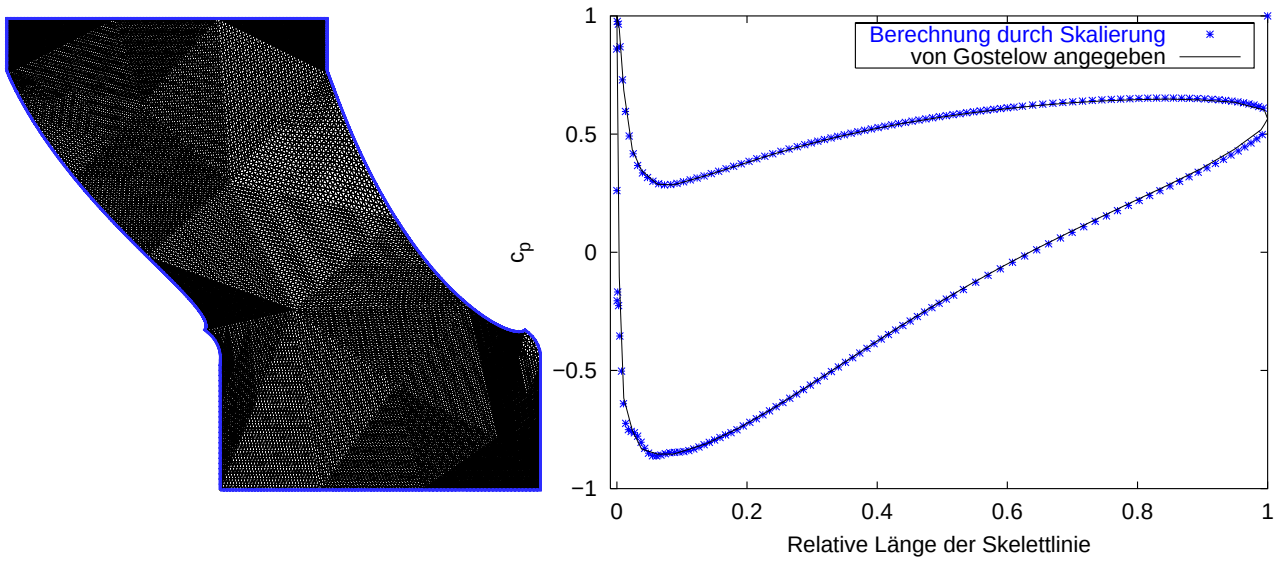

Abb. 6.11: Uniform verfeinertes Gitter und Druckverteilung auf Level 5

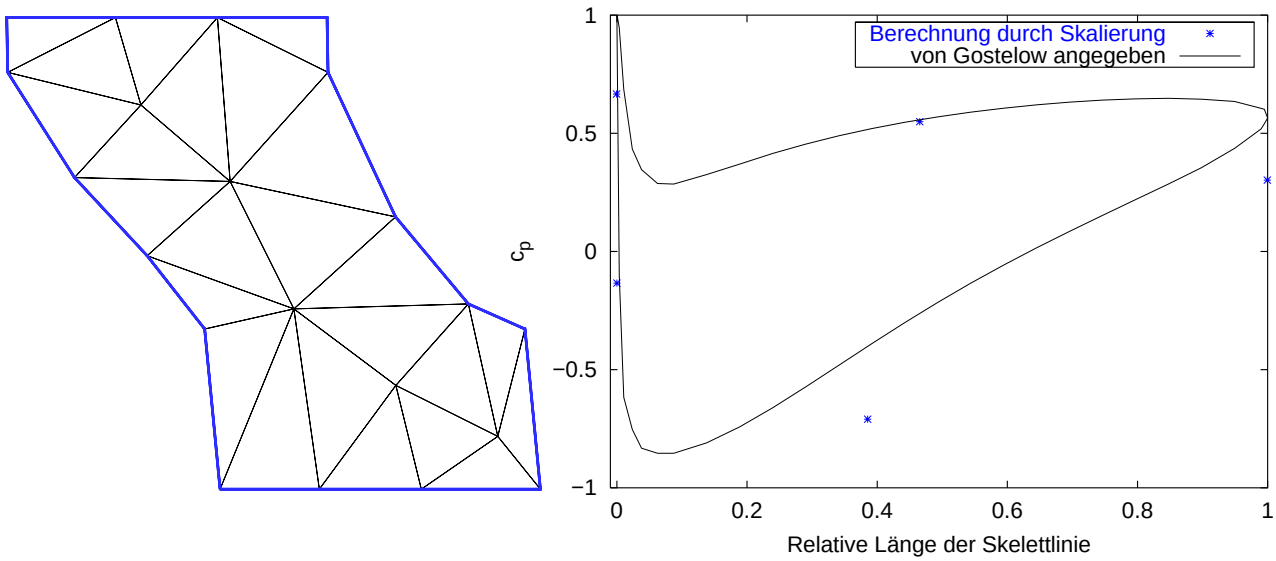

Abb. 6.12: Initiales Gitter und Druckverteilung auf Stufe 0

herangezogen. Man erkennt, daß bei feiner werdendem Gitter die Folge der berechneten Druckverteilungen tendentiell gegen die vorgegebene konvergiert. Die Wellenbewegung im Bereich der Saugspitze ist auf ein leichtes Schwingen des die Kontur beschreibenden Splines zurückzuführen.

In Tabelle 6.2 sind die Rechenzeiten in Sekunden auf einem Pentium II PC mit 200MHz mit und ohne Vorkonditionierer angegeben.

Level	Zellen	Unbekannte	Rechenzeit mit BPX	Rechenzeit ohne BPX
0	24	11	0.02	0.02
1	96	47	0.05	0.05
2	384	191	0.19	0.13
3	1536	767	1.05	0.81
4	6144	3071	4.86	6.84
5	24576	12287	21.61	46.45

Tab. 6.2: Rechenzeiten bei uniformer Verfeinerung mit und ohne Multilevel Vorkonditionierer

Man erkennt, daß die Auswirkungen der Multilevel-Vorkonditionierung (BPX) auf die Rechenzeit um so deutlicher zutage treten, je feiner das Netz ist. Auf Level 5 mit 24576 Elementen gelingt es mit Hilfe von Multileveltechniken, die Rechenzeit zu halbieren. Bei sehr groben Netzen hingegen liegen die Rechenzeiten mit Vorkonditionierer sogar etwas höher. Insgesamt ist ein von der Theorie vorhergesagtes asymptotisches $\mathcal{O}(N)$-Verhalten zu erkennen, wenn N die Zahl der Unbekannten bezeichnet.

Unter sonst unveränderten Bedingungen zeigen die Abbildungen 6.12–6.21 Gitter, die bei adaptiver Verfeinerung entstehen, sowie zugehörige Druckverteilungen. Was die Fehlerschätzung angeht, finden die Konzepte aus Abschnitt 4.6 Anwendung.

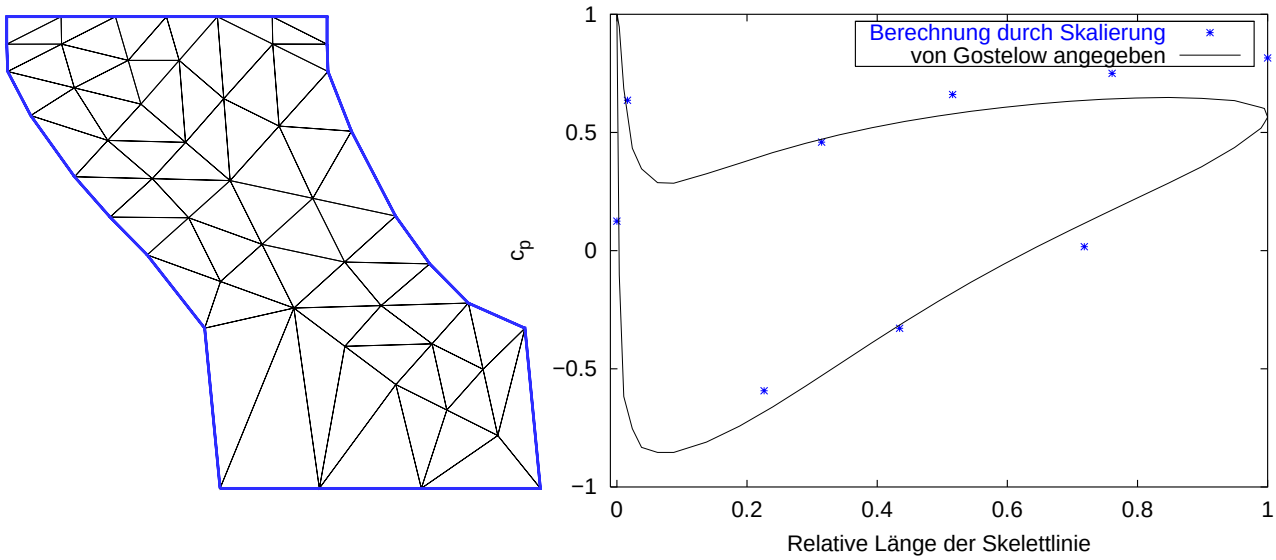

Abb. 6.13: Adaptiv verfeinertes Gitter und Druckverteilung auf Stufe 1

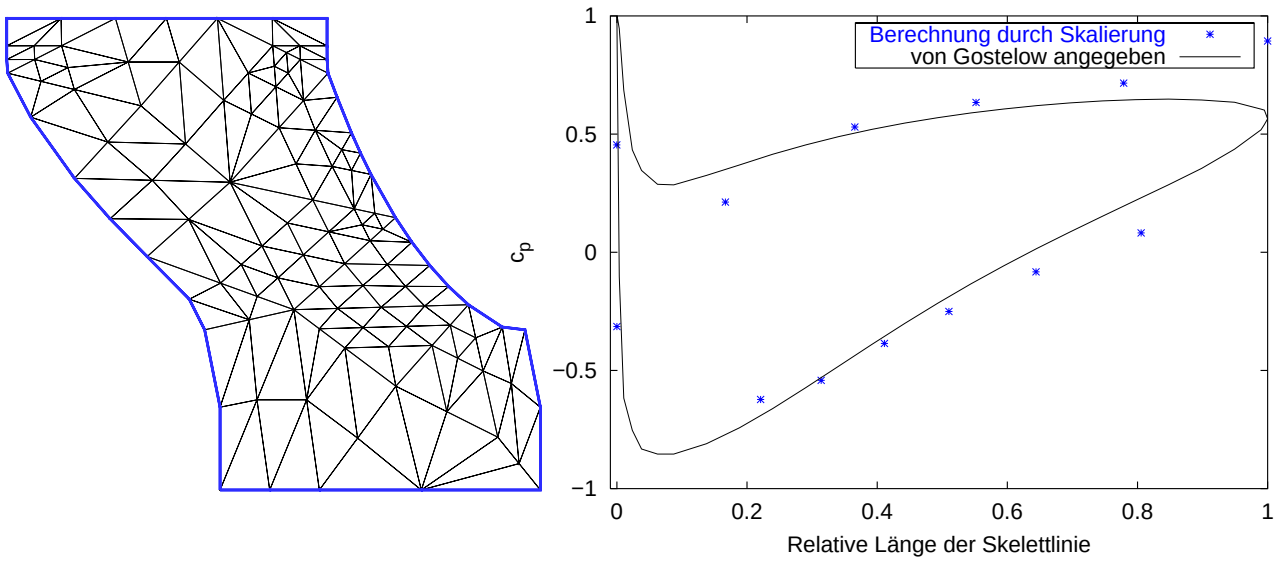

Abb. 6.14: Adaptiv verfeinertes Gitter und Druckverteilung auf Stufe 2

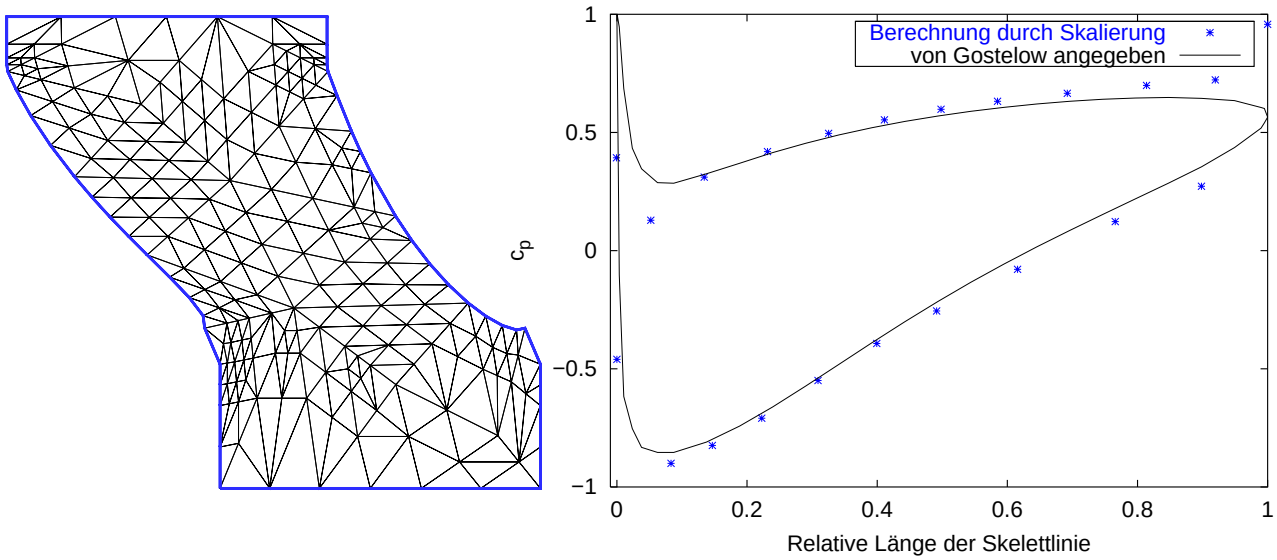

Abb. 6.15: Adaptiv verfeinertes Gitter und Druckverteilung auf Stufe 3

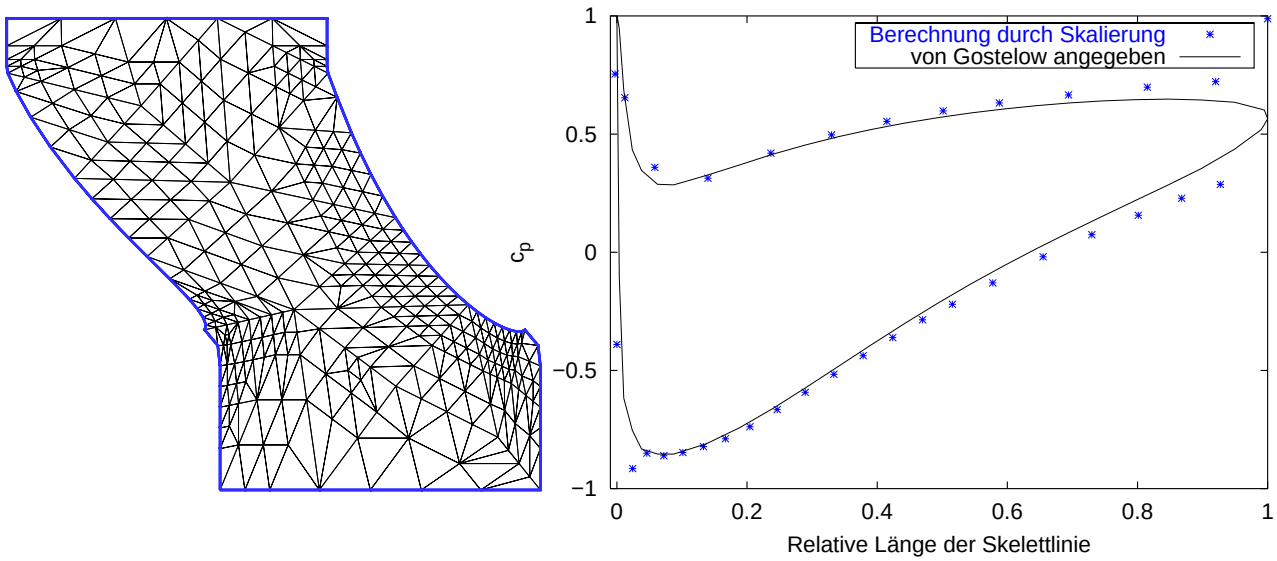

Abb. 6.16: Adaptiv verfeinertes Gitter und Druckverteilung auf Stufe 4

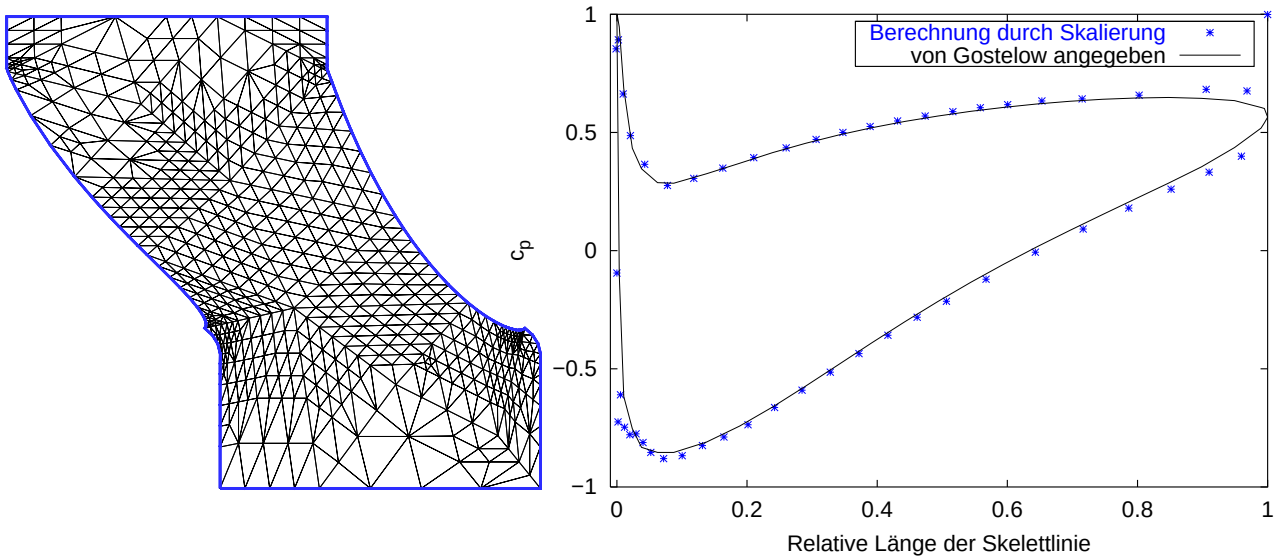

Abb. 6.17: Adaptiv verfeinertes Gitter und Druckverteilung auf Stufe 5

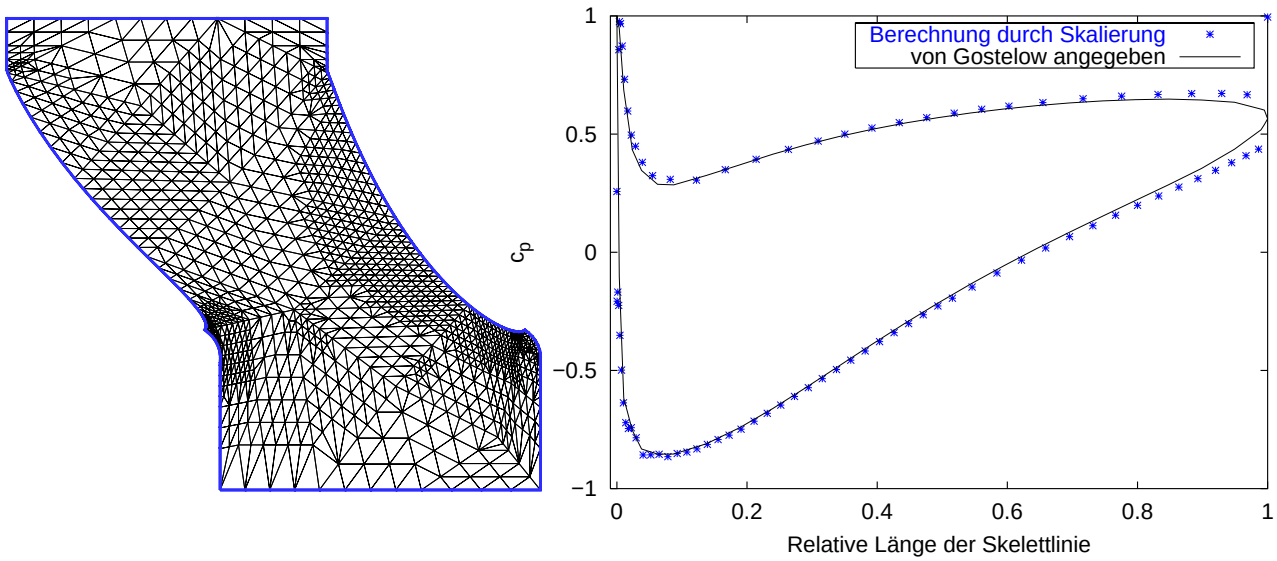

Abb. 6.18: Adaptiv verfeinertes Gitter und Druckverteilung auf Stufe 6

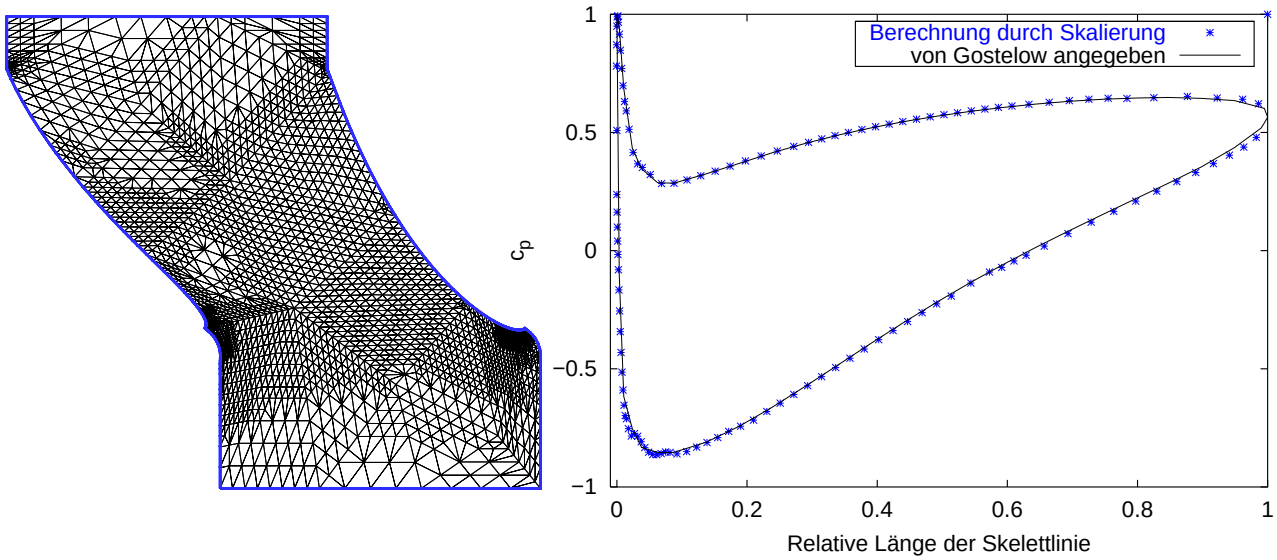

Abb. 6.19: Adaptiv verfeinertes Gitter und Druckverteilung auf Stufe 7

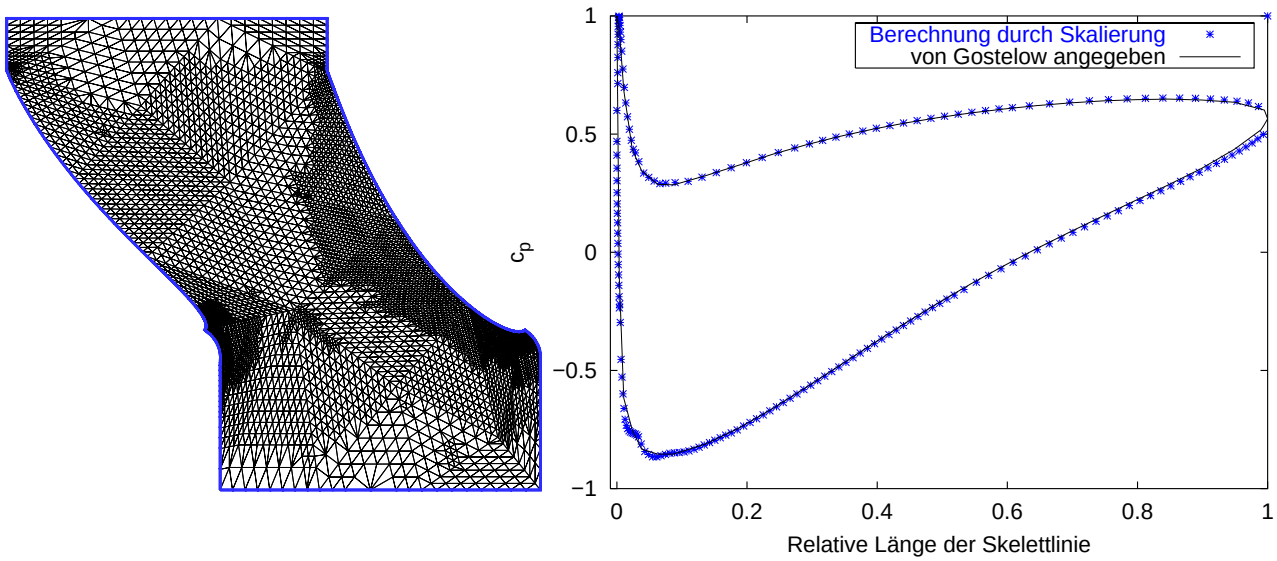

Abb. 6.20: Adaptiv verfeinertes Gitter und Druckverteilung auf Stufe 8

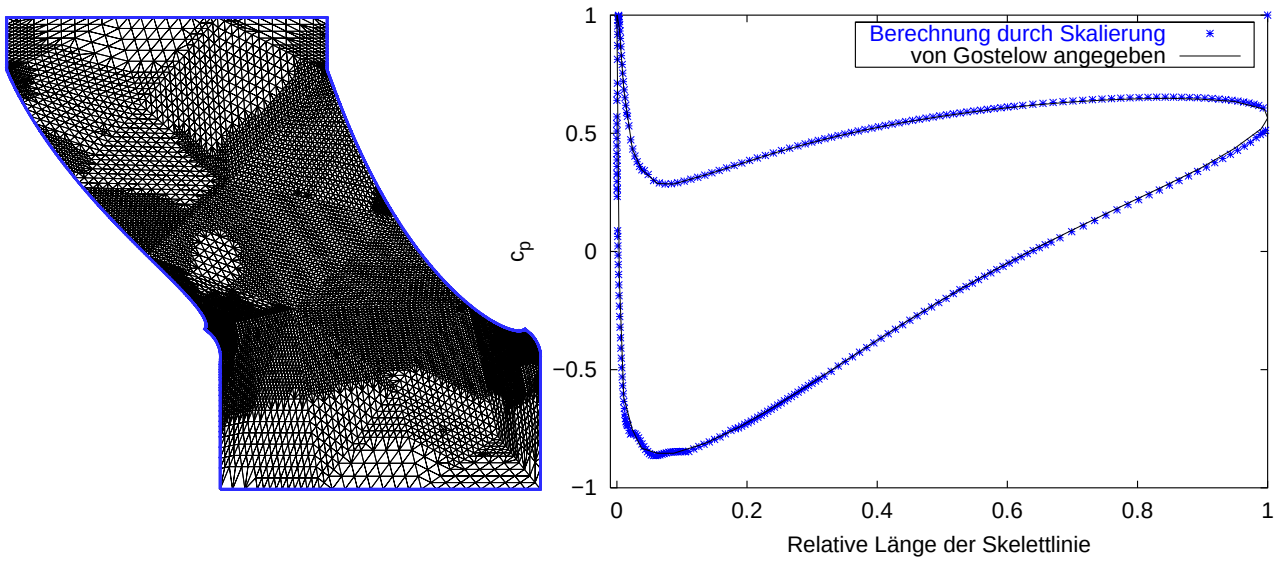

Abb. 6.21: Adaptiv verfeinertes Gitter und Druckverteilung auf Stufe 9

Man sieht, daß der Fehlerschätzer die Gitterverfeinerung im Nasenbereich und an der Schaufelhinterkante forciert. Darüber hinaus wird das Netz entlang der Schaufel — vornehmlich saugseitig — sowie zwischen den Schaufeln, wo das Fluid aufgrund der Versperrungswirkung der Schaufeln beschleunigt wird, verdichtet. Im Zu- und Nachlaufbereich ist eine mäßige Netzdichte ausreichend.

Eine akzeptable Genauigkeit der Druckapproximation stellt sich bereits nach 7 Verfeinerungsschritten auf einem Netz mit 4252 Zellen ein. Die Approximation des Druckes entlang der Schaufel ist mit der in Abbildung 6.11, die auf einem Netz mit 24576 Zellen errechnet wurde, vergleichbar. In Tabelle 6.3 sind analog zu Tabelle 6.2 die Netto-Rechenzeiten ohne Postprocessing, aber mit Netzgenerierung angegeben.

Stufe	Zellen	Unbekannte	Rechenzeit mit BPX	Rechenzeit ohne BPX
0	24	11	0.05	0.05
1	73	35	0.08	0.13
2	181	86	0.13	0.32
3	359	172	0.26	0.64
4	585	280	0.55	1.12
5	1106	536	1.14	2.10
6	2230	1084	2.58	4.34
7	4252	2083	5.59	9.45
8	8473	4165	12.31	21.69
9	16042	7922	26.69	50.23

Tab. 6.3: Rechenzeiten bei adaptiver Verfeinerung mit und ohne Multilevel Vorkonditionierer

Die Rechenzeit auf Stufe 7 des adaptiv verfeinerten Gitters liegt bei etwa 1/4 der Rechenzeit auf Level 5 des uniform verfeinerten Gitters. Daß sich kein Faktor 6 ergibt, obwohl die Zahl der Zellen

auf etwa 1/6 reduziert ist, liegt am zusätzlichen Aufwand für die Fehlerschätzung. Darüber hinaus führen Multilevelstrategien auf Stufe 7 zu einer Rechenzeitersparnis von etwa 40%.

In Abbildung 6.22 sind Geschwindigkeitsfeld im Nasenbereich der Schaufel und zugehöriges Netz auf Stufe 7 für den bisher betrachteten Betriebspunkt, vgl. linke Bilder, mit Geschwindigkeitsfeld und Netz für einen anderen Betriebspunkt verglichen.

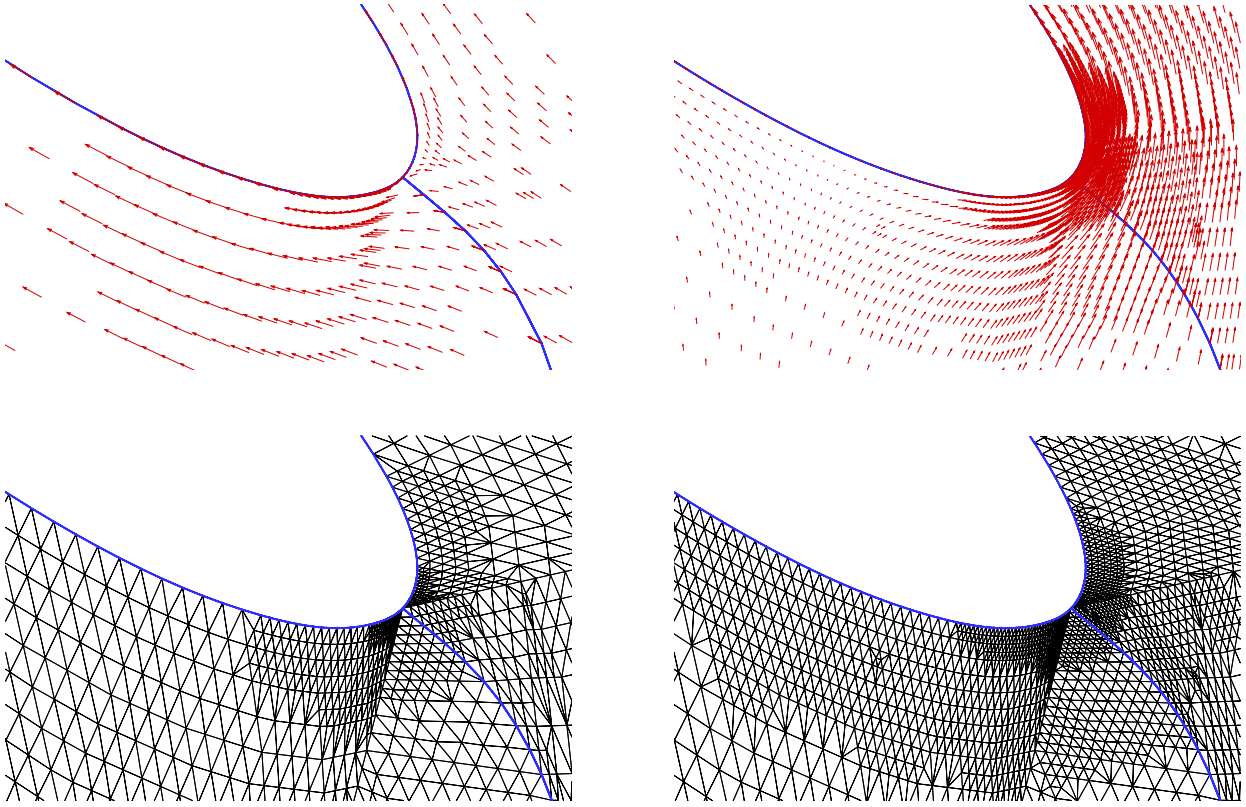

Abb. 6.22: Geschwindigkeitsfelder und Netze für 2 verschiedene Betriebspunkte

Während bei dem bisher betrachteten Betriebspunkt der Staupunkt leicht auf die Druckseite verschoben ist — die angedeutete Linie trennt als Verlängerung einer gedachten Skelettlinie Druck- und Saugseite —, stellt er sich im rechten Bild auf der Saugseite ein. Es handelt sich hierbei um eine extremen Fall von Falschanströmung. Beim Vergleich der Netze fallen 2 Aspekte auf:

- Das rechte Netz ist im Bereich des neuen Staupunktes erheblich dichter als das linke. Die Adaption des Gitters orientiert sich also nicht nur an der Geometrie, sondern auch an den Randbedingungen.
- Das rechte Netz ist insgesamt feiner. Um bei komplizierterer Strömung vergleichbare Genauigkeit sicherstellen zu können, ist ein feineres Netz nötig. Während herkömmliche Konzepte zur Netzgenerierung die Zahl der Zellen konstant halten, versuchen also adaptive, das Genauigkeitsniveau zu fixieren. Demzufolge nehmen komplexere Strömungsprobleme auch etwas höhere Rechenzeiten in Anspruch.

Abschließend soll die in Abschnitt 4.4 erläuterte Technik zur systematischen Bestimmung des Betriebspunktes stoßfreier Anströmung an diesem Beispiel verifiziert werden. Stoßfrei ist eine Anströmung dann, wenn sie tangential zu der angedeuteten Verlängerung einer gedachten Skelettlinie verläuft. In Abbildung 6.23 ist das berechnete Geschwindigkeitsfeld um die Schaufelnase herum dargestellt.

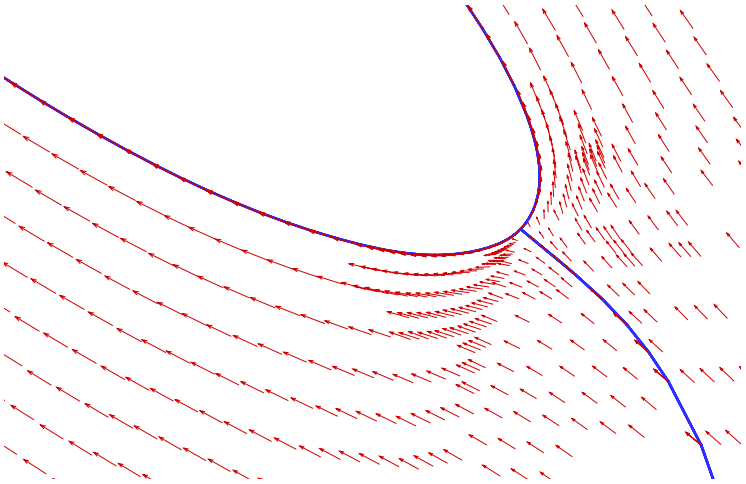

Abb. 6.23: Betriebspunkt stoßfreier Anströmung

Bei festgehaltener Drehzahl ergibt sich ein Volumenstrom von $Q = 0.0301271 m^3/s$.

6.2 Euler-Ergebnisse

Die Theorie in Kapitel 5 soll an vier exemplarischen Testfällen evaluiert werden, an einem Busemann-Gitter sowie an drei Francisturbinen, die den relevanten n_q-Bereich abdecken.

6.2.1 Busemann-Gitter

Der zugrundeliegende dreidimensionale Kontrollraum ist in Abbildung 2.2 dargestellt. Mit Hilfe der Abbildung

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} (\mu, \eta, \zeta) \quad := \quad \begin{pmatrix} \exp(-\alpha\mu)\cos(\mu+\eta) \\ \exp(-\alpha\mu)\sin(\mu+\eta) \\ \zeta \end{pmatrix}$$

gelingt die Transformation des Kontrollraums auf einen achsenparallelen Quader. Dabei gilt die Definition $\alpha := \tan\beta$, wobei β den konstanten Schaufelwinkel bezeichnet. Die Rechnungen beziehen sich allesamt auf folgende Daten: $r_0 = 0.25m$, $r_1 = 0.5m$, $r_2 = 1m$, $r_3 = 1.25m$, $Z = 6$, $\beta = 60°$. Für μ und η gilt dann:

$$-\frac{\ln(r_3/r_2)}{\alpha} \leq \mu \leq -\frac{\ln(r_0/r_2)}{\alpha}, \quad 0 \leq \eta \leq 2\pi/Z.$$

Das Rad wird von außen nach innen durchströmt. Eine schaufelkongruente Lösung für das Relativgeschwindigkeitsfeld ist gegeben durch:

$$w_x(\mu,\eta,\zeta) := -\frac{1}{\sqrt{1+\alpha^2}}(\sin(\mu+\eta)+\alpha\cos(\mu+\eta))\exp(\alpha\mu) \tag{6.4}$$

$$w_y(\mu,\eta,\zeta) := \frac{1}{\sqrt{1+\alpha^2}}(\cos(\mu+\eta)-\alpha\sin(\mu+\eta))\exp(\alpha\mu) \tag{6.5}$$

$$w_z(\mu,\eta,\zeta) := 0 \tag{6.6}$$

Wegen $w_z = 0$ und $\frac{\partial w_x}{\partial \zeta} = \frac{\partial w_y}{\partial \zeta} = 0$ handelt es sich eigentlich um eine zweidimensionale Strömung. Was den statischen Druck angeht, sind im Blick auf die *do nothing*-Randbedingungen, vgl. Abschnitt 5.1, sowie die Periodizitätsbedingungen zwei Fälle zu unterscheiden:

- **Leitrad**: Für $\omega = 0$ ist eine Drucklösung, die sowohl der Periodizitäts- als auch der Ausströmrandbedingung $p = 0$ genügt, gegeben durch:

$$p(\mu,\eta,\zeta) := \rho\left(-\frac{1}{2}\exp(2\alpha\mu)+8\right) \tag{6.7}$$

- **Laufrad**: Für $\omega \neq 0$ werden die Periodizitäts- durch Eulersche Randbedingungen ersetzt und am Ausströmrand Dirichletbedingungen vorgegeben. Eine Drucklösung ist dann gegeben durch:

$$p(\mu,\eta,\zeta) := \rho\left(-\frac{1}{2}\exp(2\alpha\mu)+\frac{2\omega\alpha}{\sqrt{1+\alpha^2}}\eta+\frac{1}{2}\omega^2\exp(-2\alpha\mu)-\tilde{p}(\omega)\right) \tag{6.8}$$

  Mit Hilfe der Verschiebung $\tilde{p}(\omega) := -1.0729591 + 0.40625\omega^2 + 6.3482974\omega$ wird das Druckniveau der Mittelwertbedingung $\int_\Omega p \; d\mathbf{x} = 0$ entsprechend fixiert, vgl. BANK ET AL. [5]. Die Tatsache, daß der Druck lediglich bis auf eine Konstante bestimmt ist, hat auch auf das diskrete System (5.28) Auswirkungen.

  Die Matrix BC hat nicht vollen Rang. Genauer gilt: $\mathrm{Kern}(BC)^T = \mathrm{span}\{(1,1,\ldots,1)^T\}$. Zur Lösung der Druckkorrekturgleichung im Rahmen des SIMPLE-Verfahrens sind demzufolge wegen der Singularität der Matrix Standardlöser wie etwa der ILU ausgeschlossen. Im Gegensatz dazu bedeuten verschwindende Eigenwerte für das CG-Verfahren kein Problem, vorausgesetzt die rechte Seite des Gleichungssystems liegt im Bild der Matrix. Dies ist erfüllt, falls die Dirichletrandwerte der Kompatibilitätsbedingung $\int_\Gamma \mathbf{c}\cdot\mathbf{n}\; d\Gamma = 0$ genügen.

  Ungenauigkeiten, die u.a. durch Interpolation der Randwerte entstehen, vgl. GLOWINSKI [25], können durch explizite Projektion der rechten Seite in das Bild der Matrix kompensiert werden. Die Normierungsbedingung $\int_\Omega p_h \; d\mathbf{x} = 0$ wird ebenfalls erzwungen. Näheres dazu ist in [50] oder bei BANK ET AL. [5] nachzulesen.

Leitrad

Für $\omega = 0$ gilt $rc_u = 1/\sqrt{1+\alpha^2}$. Der Drall rc_u ist also im gesamten Strömungsfeld konstant, es erfolgt keine Umlenkung. Demzufolge verschwindet auch das Schaufelmoment punktweise, die Isoflächen des statischen Druckes sind konzentrische Zylindermantelflächen. In Strömungsrichtung nimmt der Druck quadratisch mit dem Radius ab. In Abbildung 6.24 sind Geschwindigkeitsfeld sowie Isolinien des statischen Druckes auf dem Gitter, das in Abbildung 2.2 angedeutet ist, für einen Schnitt senkrecht zur z-Achse dargestellt.

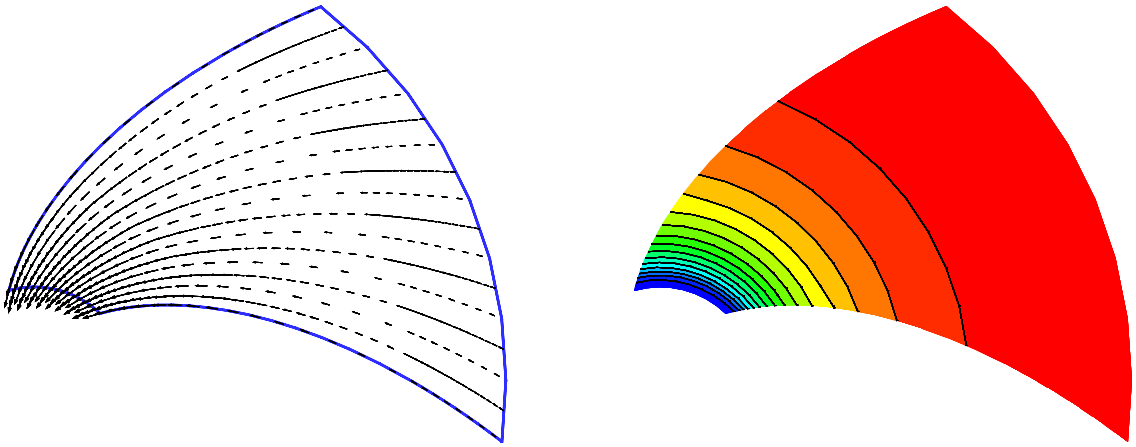

Abb. 6.24: Berechnete Lösung für das Geschwindigkeits- und Druckfeld nach 3 Verfeinerungsschritten für einen Schnitt senkrecht zur z-Achse

In Tab. 6.4 sind neben dem $L^2(\Omega)$-Fehler der berechneten Geschwindigkeit sowie des Druckes bezüglich der exakten Lösung (6.4)–(6.6), (6.7) die Rechenzeiten bei globaler Gitterverfeinerung aufgelistet.

Level	Zellen	Unbekannte	$\|\mathbf{c}-\mathbf{c}_h\|_{L^2(\Omega)}$	$\|p-p_h\|_{L^2(\Omega)}$	Rechenzeit
1	24	312	0.228	0.614	0.07
2	192	2208	0.107	0.264	0.39
3	1536	16512	0.051	0.125	4.03
4	12288	127488	0.025	0.063	39.05
5	98304	1001472	0.012	0.032	322.97

Tab. 6.4: $L^2(\Omega)$-Fehler und Rechenzeiten bei uniformer Verfeinerung

Sowohl der Geschwindigkeits- als auch der Druckfehler zeigen ein Abklingverhalten 1. Ordnung in der $L^2(\Omega)$-Norm. Dies bestätigt *a-priori*-Abschätzungen sowohl für das einfache Upwind-Verfahren als auch für stückweise konstante Approximation des Druckes, vgl. Turek [74]. Die Rechenzeiten verhalten sich asymptotisch proportional zur Zahl der Unbekannten. Ohne Mehrgitterverfahren würde man ein weit überproportionales Verhalten feststellen.

Laufrad

Für $\omega \neq 0$ induzieren die Coriolis-Kräfte ein Schaufelmoment. Die Drehrichtung des Laufrades ist die negative Umfangsrichtung. Für $\omega = -2/s$ ergibt sich mit $r_{ref} = 1.25m$ gemäß der Definitionen in Abschnitt 3.2.2 als dimensionslose Druckzahl: $\psi = 1.92$. Dieser sowie alle folgenden ψ-Werte für Turbinen stimmen bis auf das Vorzeichen mit den in Abschnitt 3.2.2 definierten Größen überein.

Abbildung 6.25 zeigt das Geschwindigkeitsfeld auf Level 3. Offenkundig ist es noch nicht optimal. Auf Level 5 hingegen sind Unterschiede zu dem Geschwindigkeitsfeld in Abbildung 6.24 nicht mehr festzustellen. Auf eine Darstellung wird deshalb verzichtet.

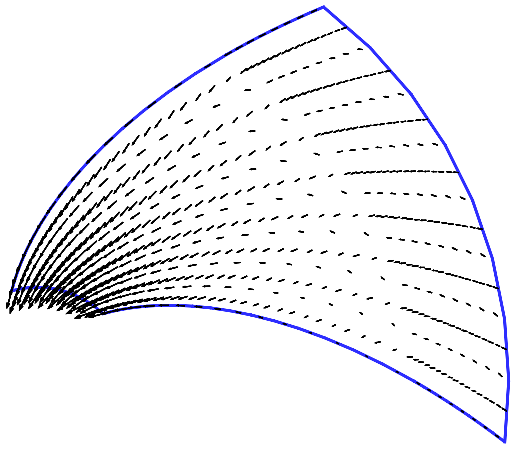

Abb. 6.25: Berechnete Lösung für das Geschwindigkeitsfeld nach 3 Verfeinerungsschritten für einen Schnitt senkrecht zur z-Achse

Abbildung 6.26 zeigt einen Vergleich der berechneten Lösungen für den statischen Druck auf Level 3 bzw. 5.

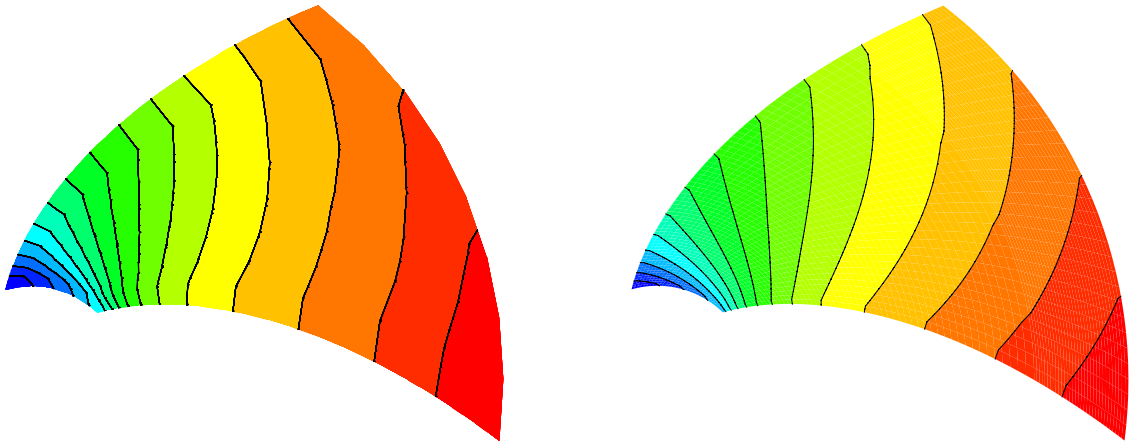

Abb. 6.26: Berechnete Lösung für den statischen Druck nach 3 (links) bzw. 5 Verfeinerungsschritten für einen Schnitt senkrecht zur z-Achse

Wenn man von den Isolinien, die auf der Saugseite beginnen, jeweils die herausgreift, die dem Einströmramd am nächsten liegt, so sieht man, daß diese Isolinie druckseitig im rechten Bild erheblich näher am Ausströmrand endet als im linken, d.h. die durch Corioliseffekte induzierte Druckdifferenz zwischen Druck- und Saugseite wird auf Level 5 wesentlich besser aufgelöst als auf Level 3. Dieses Verhalten deckt sich mit der Entwicklung der ψ_M-Werte in Tabelle 6.5.

Level	Zellen	ψ_t	ψ_{th}	ψ_M	$\|\mathbf{c}-\mathbf{c}_h\|_{L^2(\Omega)}$	$\|p-p_h\|_{L^2(\Omega)}$
3	1536	1.379	1.906	1.302	0.40	0.38
4	12288	1.646	1.916	1.6	0.21	0.19
5	98304	1.776	1.919	1.755	0.11	0.094

Tab. 6.5: ψ-Werte und $L^2(\Omega)$-Fehler bei uniformer Verfeinerung

Neben den Druckzahlen sind die $L^2(\Omega)$-Fehler der berechneten Geschwindigkeit sowie des Druckes bezüglich der exakten Lösung (6.4)–(6.6), (6.8) bei globaler Gitterverfeinerung angegeben. Sowohl die $L^2(\Omega)$-Fehler als auch die ψ-Werte zeigen ein Konvergenzverhalten 1. Ordnung. Daß die Werte für ψ_{th} den exakten Wert 1.92 auffallend gut approximieren, ist damit begründet, daß — der Vorbemerkung über die Laufradrechnung entsprechend — auf dem ganzen Rand Geschwindigkeiten vorgegeben sind, so daß bei der Berechnung von ψ_{th} ausschließlich auf vorgegebene Randwerte zurückgegriffen wird.

6.2.2 Francisturbinen

Abschließend soll der Euler-Code an drei realen Francisturbinen mit den spezifischen Drehzahlen 30, 50 und 100 validiert werden. Die Ergebnisse im optimalen Betriebspunkt werden mit denen des kommerziellen Navier-Stokes-Codes TASCflow der Firma AEA verglichen.

In allen drei Fällen werden die über dem Verfeinerungslevel aufgetragenen Druckzahlen des Euler-Codes, vgl. Abschnitt 3.2.2, den auf jeweils einem einzelnen Gitter berechneten Werten für ψ_t und ψ_{th} des Navier-Stokes-Codes gegenübergestellt. Verglichen werden weiterhin die Druckverteilungen in der Schaufelmitte sowie die rc_u-Verteilungen am Ausströmrand. Bezüglich letzterer sind in Umfangsrichtung gemittelte Werte zwischen Nabe und Deckscheibe aufgetragen. Ein Vergleich der Rechenzeiten findet sich am Ende dieses Abschnitts.

Francisturbine FT 30

Bezüglich einer langsamläufigen Francisturbine ergeben sich die in Abbildung 6.27 dargestellten Werte für ψ_t, ψ_{th} sowie ψ_M. Dem Verlauf der Kurven ist ein ψ-Wert von 1.81 im Sinne von (3.22) zu entnehmen. Die von TASCflow berechneten Werte liegen bei $\psi_t = 1.87$ bzw. $\psi_{th} = 1.75$. Der *numerische Wirkungsgrad*

$$\eta_{num} := \frac{\psi_{th}}{\psi_t} \tag{6.9}$$

der Euler-Rechnung liegt auf Level 4 bei 98%.

Wie Abbildung 6.28 zeigt, stellt sich bei feiner werdendem Gitter eine Druckverteilung entlang der Schaufel ein, die vom TASCflow-Ergebnis nur unwesentlich abweicht. Was die rc_u-Verteilung angeht, ergibt sich die in Abbildung 6.29 zu erkennende Diskrepanz. Die Unterschiede sind beispielsweise dadurch begründet, daß Grenzschichten von einem Euler-Code nicht simuliert werden können.

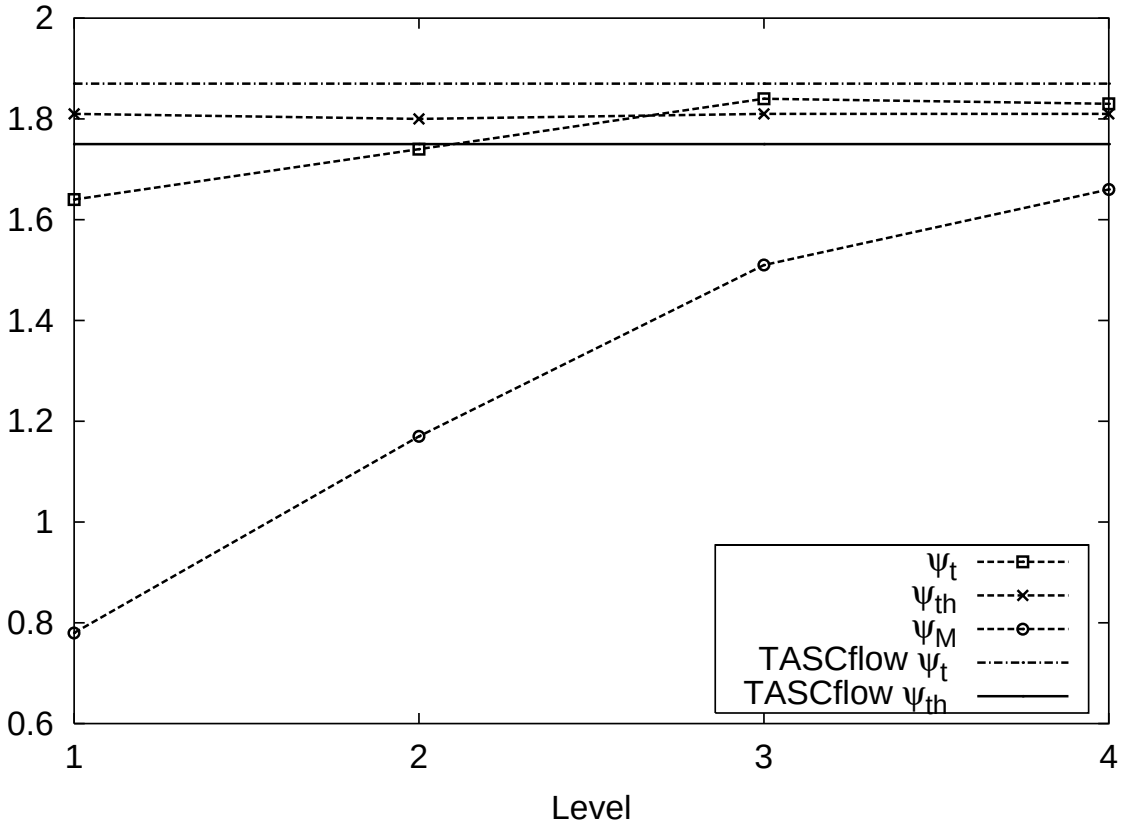

Abb. 6.27: Vergleich der Druckzahlen der FT 30

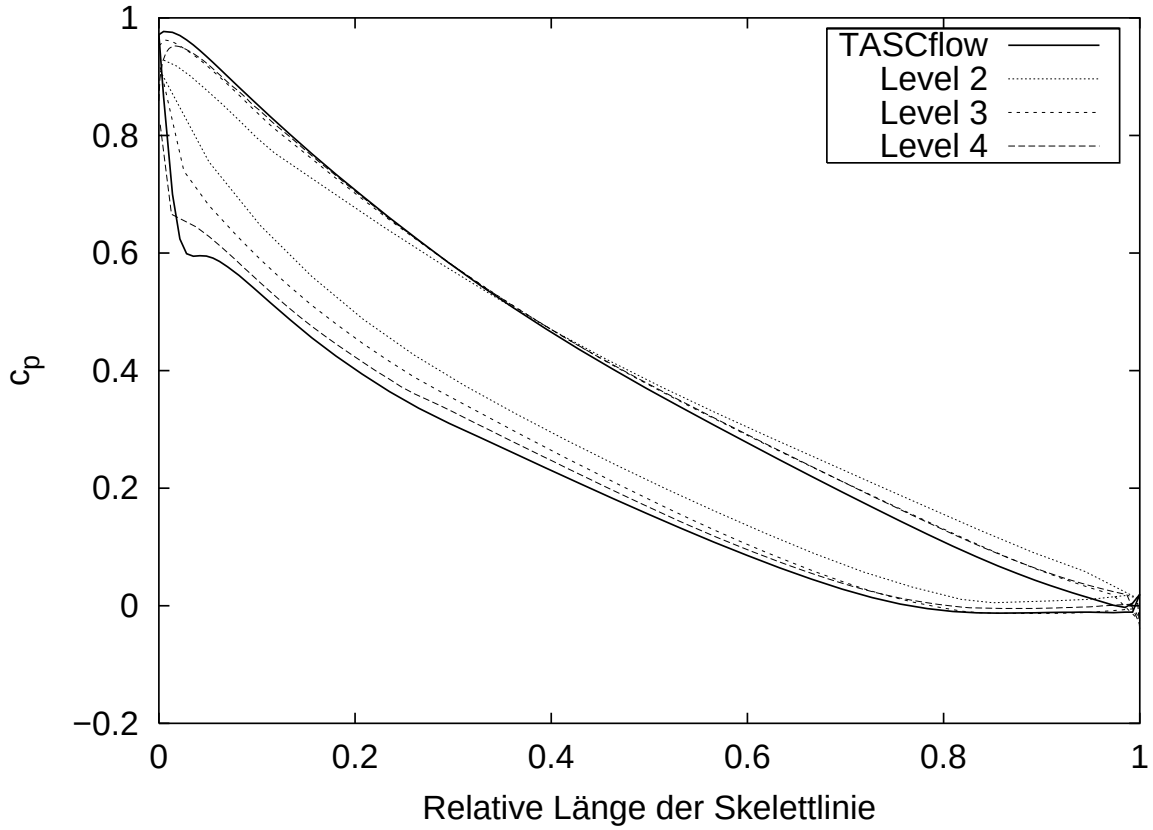

Abb. 6.28: c_p-Verteilung in der Schaufelmitte der FT 30

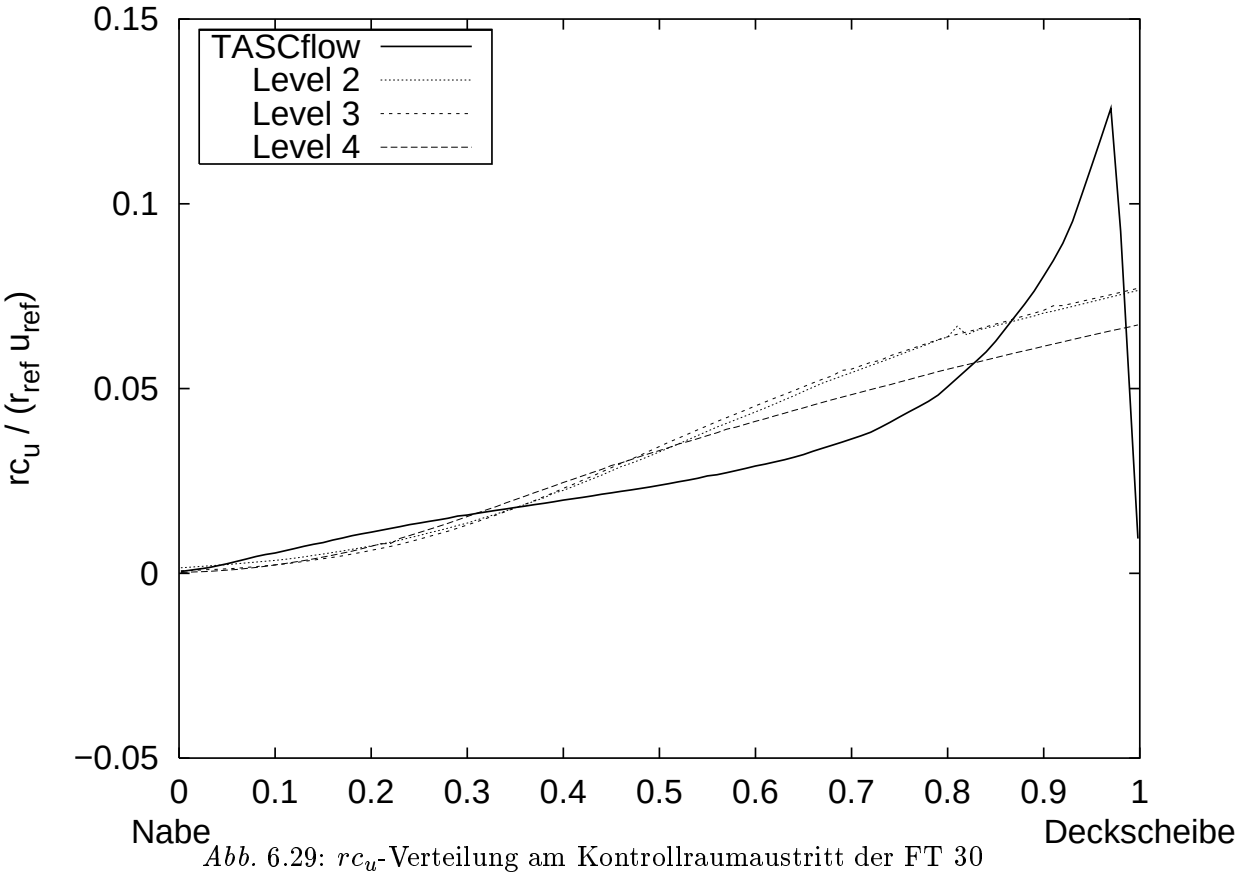

Abb. 6.29: rc_u-Verteilung am Kontrollraumaustritt der FT 30

Francisturbine FT 50

In Abbildung 6.30 ist exemplarisch die Geometrie der Laufradbeschaufelung der Turbine FT 50 dargestellt. Neben den zur Formulierung der Randbedingungen relevanten Flächen ist die Schaufelspur auf der Deckscheibe zu erkennen. Die Drehachse steht senkrecht zur Ausströmfläche. Das Wasser strömt radial in das Laufrad ein und axial aus.

Die Druckzahlen sind Abbildung 6.33 zu entnehmen. Extrapoliert man die Euler-Lösungen, so erhält man einen ψ-Wert von 1.46. Die TASCflow-Werte liegen bei $\psi_t = 1.50$ bzw. $\psi_{th} = 1.45$. η_{num} gemäß (6.9) beträgt 98.8%.

Im Vergleich zum Laufrad mit der spezifischen Drehzahl $n_q = 30\frac{1}{min}$ stimmen hier die rc_u-Verteilungen besser überein, vgl. Abbildung 6.32. Wie Abbildung 6.31 zeigt, weisen die Druckverteilungen jedoch ab der Mitte der Schaufel deutliche Unterschiede auf.

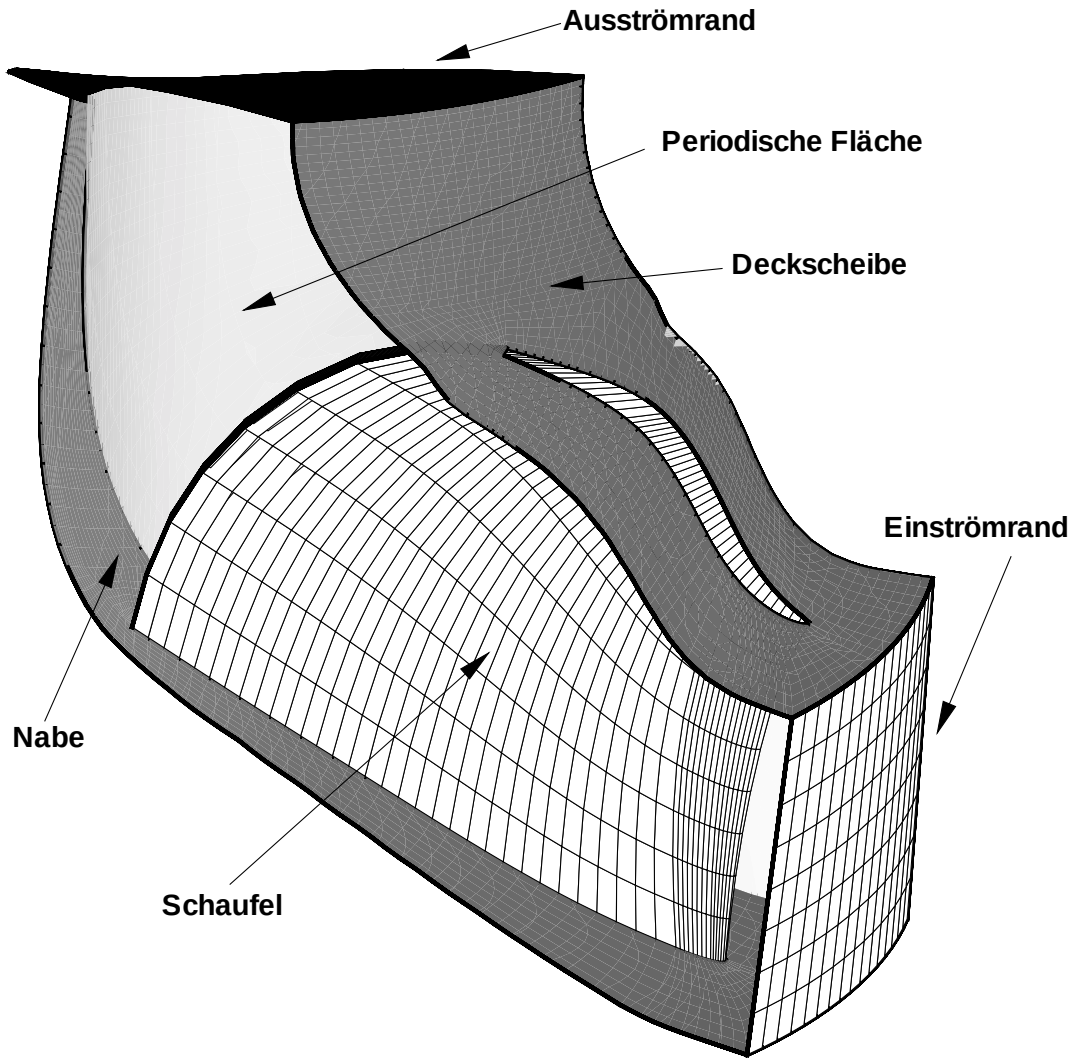

Abb. 6.30: Geometrie und Randbedingungen für das Laufrad der FT 50

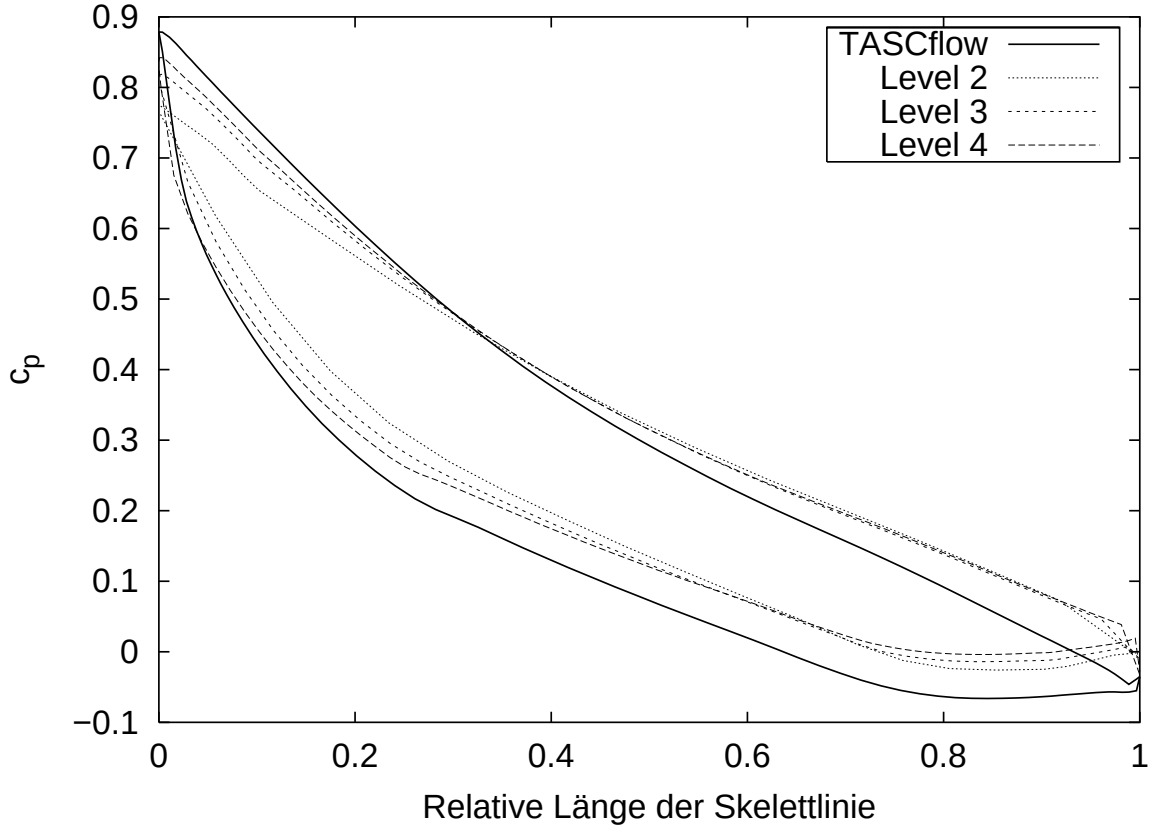

Abb. 6.31: c_p-Verteilung in der Schaufelmitte für die FT 50

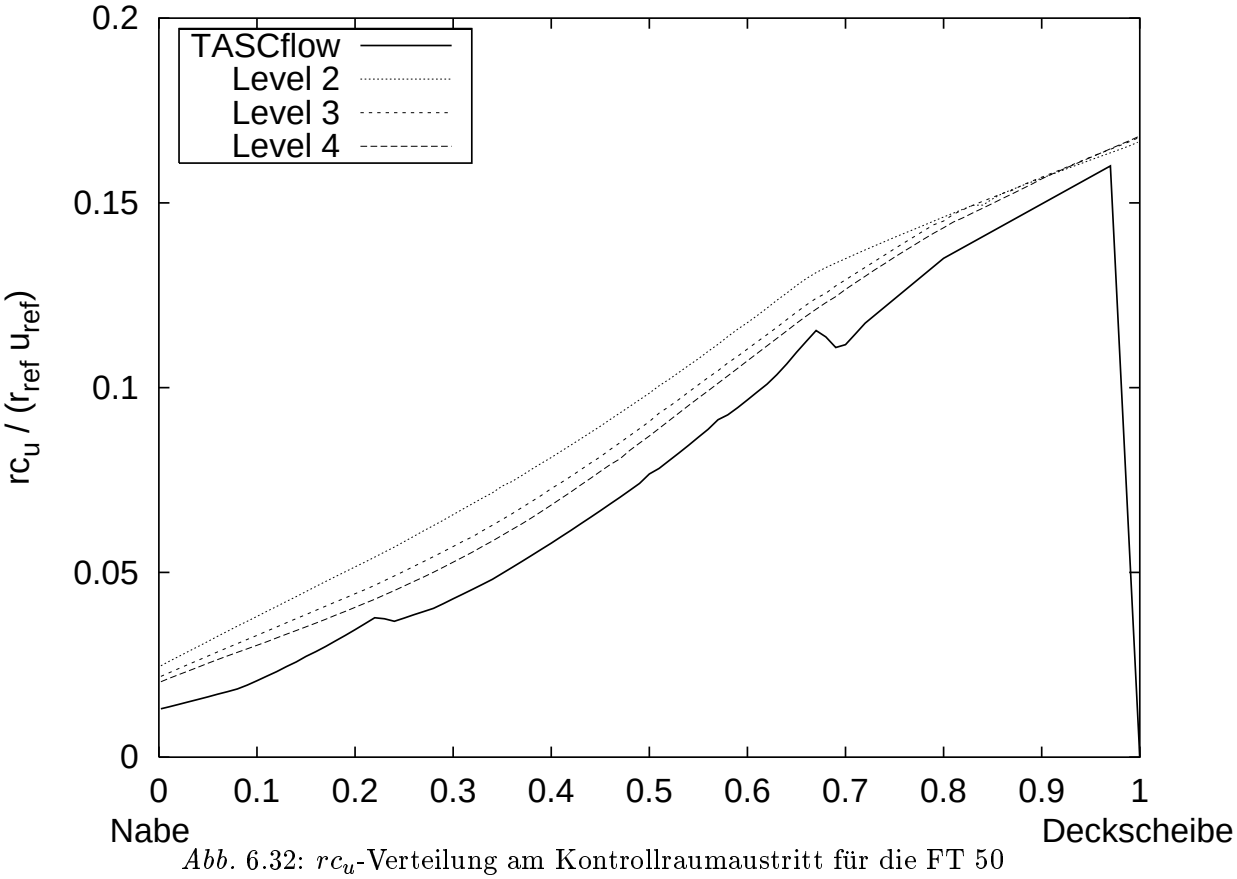

Abb. 6.32: rc_u-Verteilung am Kontrollraumaustritt für die FT 50

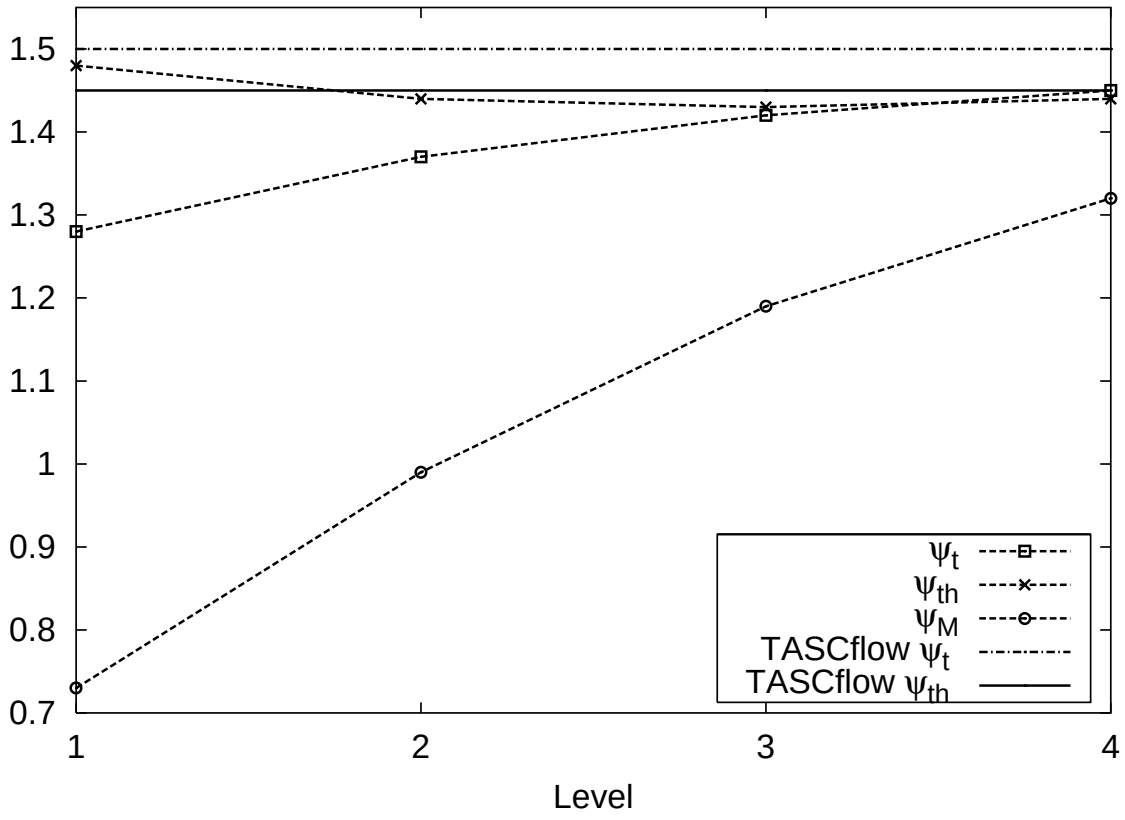

Abb. 6.33: Vergleich der Druckzahlen für die FT 50

Francisturbine FT 100

Abschließend sind Ergebnisse für eine schnelläufige Francisturbine dargestellt. Zunächst betrachte man die ψ-Werte in Abbildung 6.36. Durch Extrapolation ergibt sich für die Euler-Rechnung eine Druckzahl von 1.08. Die TASCflow-Werte liegen bei $\psi_t = 1.22$ und $\psi_{th} = 1.10$. Als numerischer Wirkungsgrad im Sinne von (6.9) ergibt sich ein Wert von 92%.

Wie Abbildung 6.34 zu entnehmen ist, stimmt — vom Nasenbereich der Saugseite abgesehen — die extrapolierte Druckverteilung mit der von TASCflow berechneten sehr gut überein. Qualitativ gilt dies auch für die in Abbildung 6.35 dargestellten rc_u-Verteilungen, wobei die Euler-Rechnung eine etwas geringere Umlenkung prognostiziert. Dieses Ergebnis ist mit den integralen ψ-Werten konsistent.

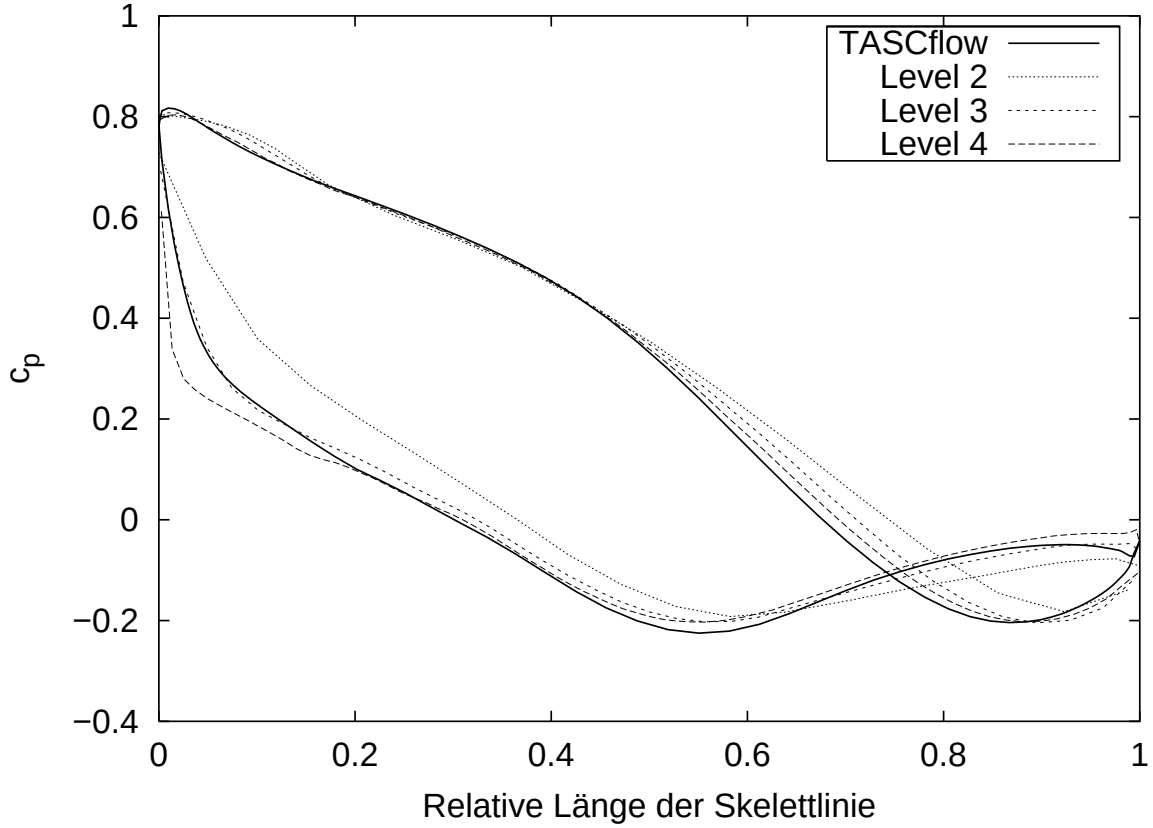

Abb. 6.34: c_p-Verteilung in der Schaufelmitte für die FT 100

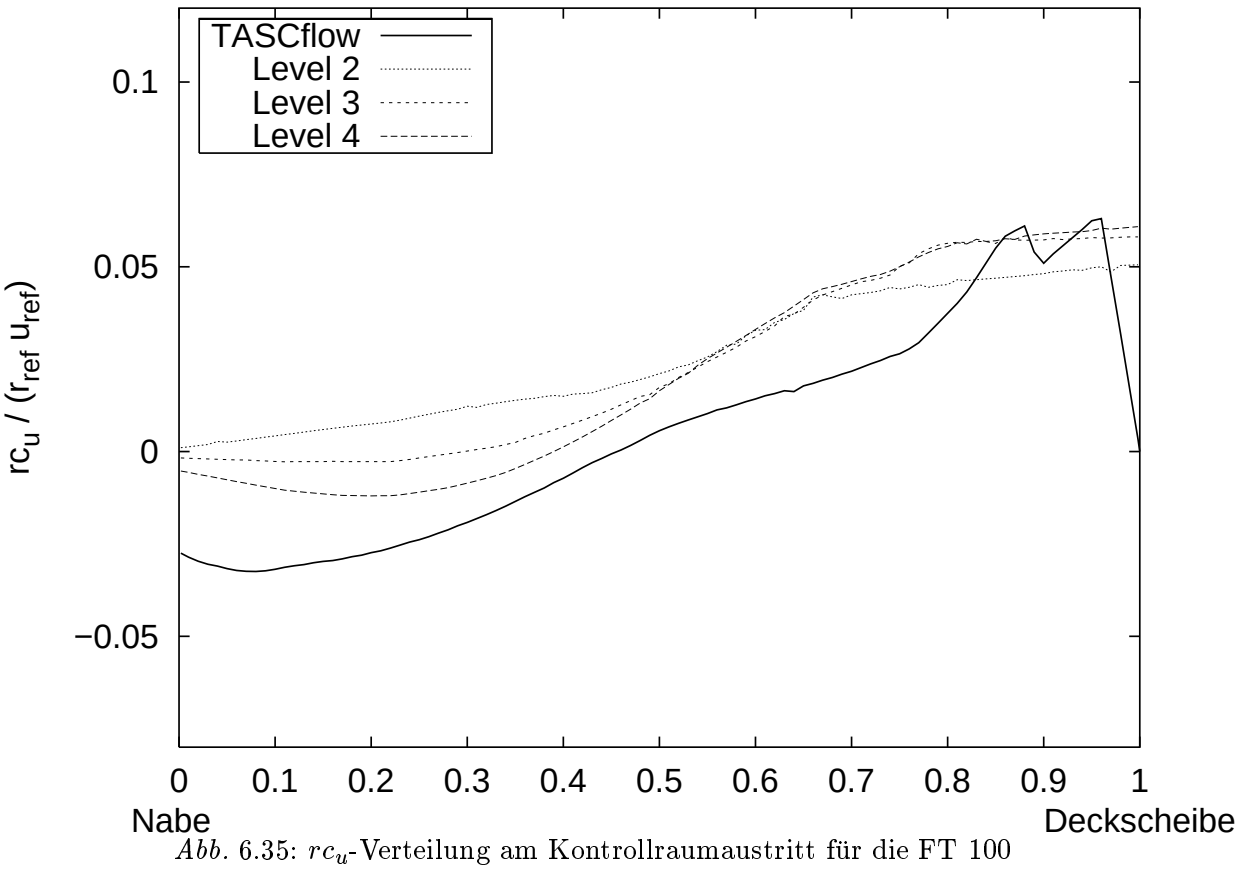

Abb. 6.35: rc_u-Verteilung am Kontrollraumaustritt für die FT 100

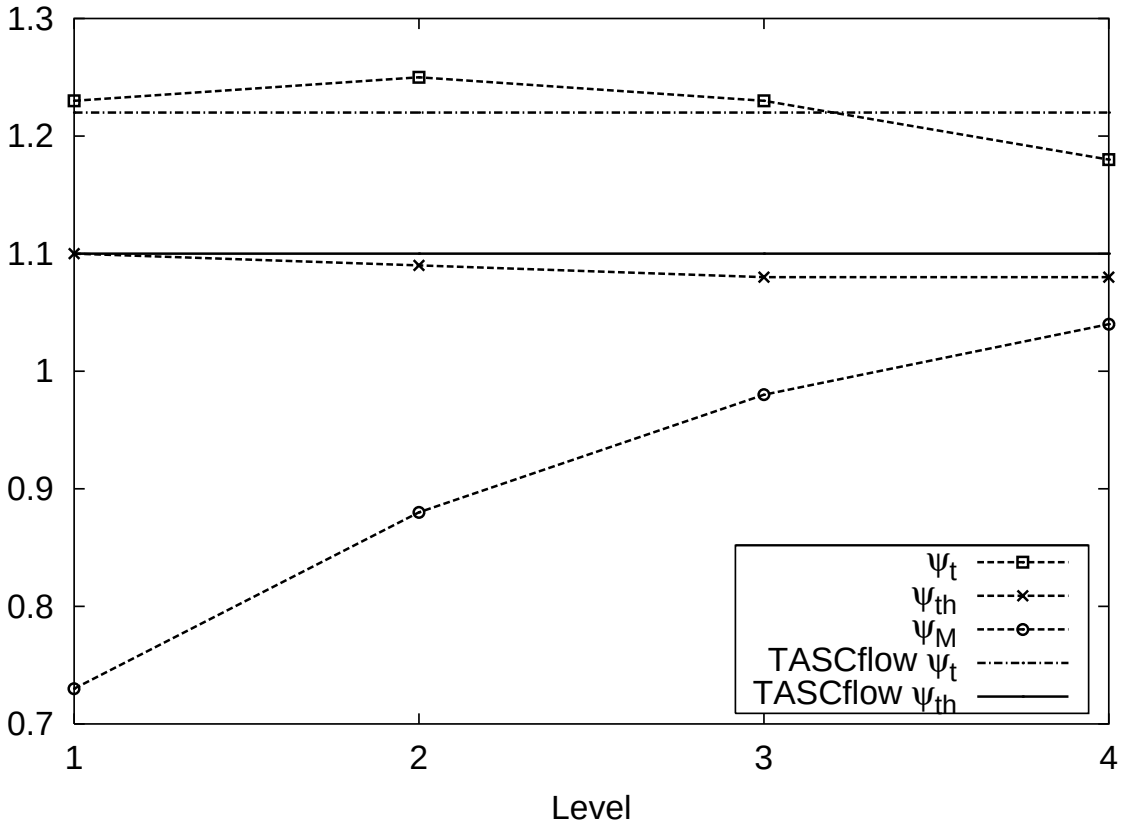

Abb. 6.36: Vergleich der Druckzahlen für die FT 100

Rechenzeiten

Für das Laufrad der FT 100 sind in Abbildung 6.37 die Rechenzeiten für unterschiedlich feine Gitter aufgetragen. Durchgeführt wurden diese Rechnungen auf einer HP 9000/785 mit 440 MHz. Für die Euler-Rechnung hat sich ein relatives Abbruchkriterium bewährt, d.h. die nichtlineare Iteration wird abgebrochen, wenn das Residuum um drei Größenordnungen reduziert ist. Bei dieser Konfiguration ergibt sich offenkundig eine proportionale Abhängigkeit der Rechenzeit von der Zahl der Unbekannten, wie man dies von Mehrgitterverfahren erwartet, vgl. Abschnitt 5.9.

Daß die Rechenzeiten für TASCflow sogar ein leicht unterproportionales Verhalten zeigen, liegt daran, daß aufgrund des dort implementierten Abbruchkriteriums die Zahl der nichtlinearen Iterationsschritte stark schwankt. Im relevanten Bereich erreicht TASCflow eine Rechengeschwindigkeit von 137, der Euler-Code hingegen 1092 Unbekannte pro Sekunde, was einem Faktor von 8 entspricht.

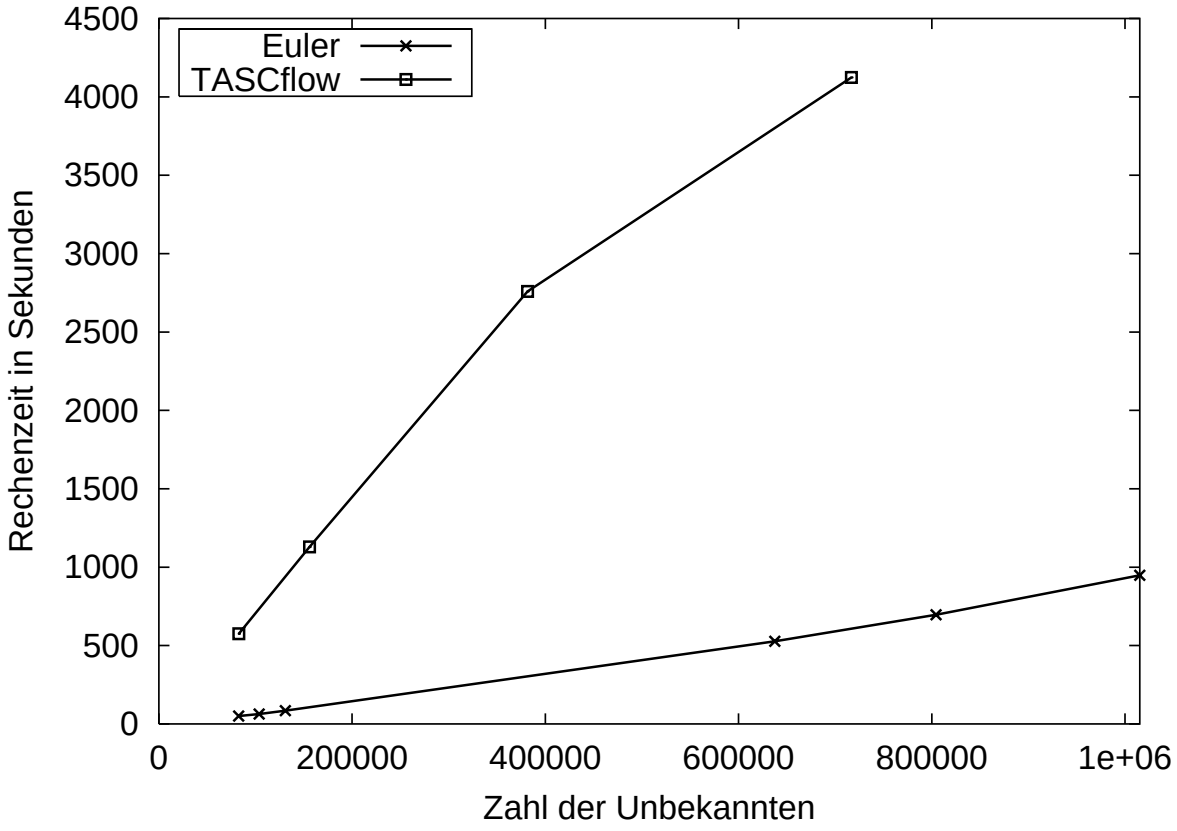

Abb. 6.37: Vergleich der Rechenzeiten für die FT 100

Zusammenfassend sind in Tabelle 6.6 die Rechenzeiten für die drei Francisturbinen auf einem geringfügig langsameren Rechner verglichen.

n_q	TASCflow Zellen	TASCflow Rechenzeit	Zellen auf Level 4	Rechenzeit auf Level 4
30	32566	1465	172032	2296
50	32566	3327	77824	980
100	54740	2798	77824	905

Tab. 6.6: Zahl der Zellen und Rechenzeiten für FT 30, 50 und 100

Auffällig ist zunächst, daß die TASCflow-Rechenzeit für die FT 50 trotz gröberen bzw. gleich feinen Netzes deutlich höher liegt als für FT 100 bzw. FT 30. Dieser Effekt ist wiederum auf die starken Schwankungen der Iterationszahl zurückzuführen, so daß es schwierig ist, für TASCflow eine repräsentative Rechenzeit anzugeben. In Bezug auf die Euler-Rechnung besteht die Problematik grundsätzlich darin, daß bei globaler Verfeinerung in jedem Verfeinerungsschritt lediglich eine Verachtfachung der Zellenzahl realisiert werden kann, so daß bei einem Netz mit 77824 Zellen das nächst gröbere aus 9728 besteht. Zwischenstufen mit 30000–40000 Zellen läßt der Code in seiner gegenwärtigen Konfiguration nicht zu.

Obwohl die Netze für Euler-Rechnungen im Vergleich zu TASCflow-Netzen feiner sind, ist der Euler-Code im Vergleich zu TASCflow etwa drei- bis viermal so schnell. Diese Aussage stützt sich auf ausführliche Testrechnungen, von denen hier nur 3 exemplarisch aufgeführt sind.

Interpretation der Ergebnisse

Die 3 beschriebenen Testfälle haben durchaus repräsentativen Charakter. Dies gilt sowohl für die Rechenzeiten, als auch für den Vergleich mit TASCflow-Ergebnissen. Sicherlich wird man keine wesentlichen Diskrepanzen zwischen reibungsfreier und reibungsbehafteter Strömung bei hohen Reynoldszahlen erwarten. Wie gut aber die Übereinstimmung im Einzelfall quantitativ tatsächlich ist, ist vom physikalischen Standpunkt aus nicht klar. Wenn dann noch numerische Unwägbarkeiten hinzukommen, ist die Vergleichbarkeit von Euler- mit Navier-Stokes-Ergebnissen durchaus in Frage zu stellen. Dies gilt umso mehr, wenn keine Abströmbedingung, wie etwa die Kutta-Bedingung, implementiert ist.

Wesentlich ist, daß die erzielten Euler-Ergebnisse in sich konsistent sind. Dies bezieht sich insbesondere auf das Konvergenzverhalten der ψ-Werte. Ziel einer Weiterentwicklung des Codes muß es sein, auf erheblich gröberen Gittern eine absolute Lösungsgenauigkeit sicherzustellen, die bei globaler Verfeinerung erst auf Level 6 oder 7 erreichbar ist. Dabei spielen im wesentlichen zwei Aspekte eine Rolle:

- Das Netz muß im Zuge der Verfeinerung an die zu approximierende Lösung angepaßt werden. Wie in Abschnitt 2.2 ausführlich erläutert, geschieht dies in CONSIST mittels lokaler Verfeinerungstechniken. Insbesondere stellt diese Strategie eine verbesserte Auflösung von Lösungsdetails, wie etwa von Saugspitzen in der Druckverteilung, in Aussicht.
- Die Upwind-Diskretisierung muß durch ein Verfahren höherer Ordnung ersetzt werden. Zu diesem Zweck bietet sich ein Stromliniendiffusionsverfahren an, vgl. Abschnitt 5.2.2.

Was die erste Strategie auszeichnet, ist die Tatsache, daß sie im Vergleich zur zweiten wesentlich mehr Variationsmöglichkeiten bietet. Die Druckapproximation in der L^2-Norm geschieht ebenfalls mit der Ordnung 1. Bei einfacher Extrapolation des Druckes auf die Schaufel sind nun Ordnungseinbußen in Kauf zu nehmen, vgl. GUNZBURGER [30]. Abhilfe kann die im Zusammenhang mit dem Potentialverfahren vorgeschlagene Strategie verschaffen, vgl. Abschnitt 6.1.1.

Gelingt es, mit Hilfe dieser Techniken die für eine Euler-Rechnung notwendige Zahl an Zellen auf 10000–15000 zu reduzieren, sind Laufzeiten deutlich unter 5 Minuten auf 450 MHz-Rechnern durchaus realistisch.

Kapitel 7

Bewertung und Ausblick

Adaptivitätskonzepte sowie Multilevellösung konnten sich bisher in kommerzieller numerischer Software nicht entscheidend durchsetzen. Angesichts der Forderung nach genauerer Auflösung bei immer kürzeren Rechenzeiten wird es mittelfristig jedoch keine Alternative dazu geben. Ähnliches gilt für abstrakte Lösungsansätze weit über die Softwareentwicklung hinaus.

7.1 Adaptive Multilevelverfahren

Die numerischen Ergebnisse in Kapitel 6 untermauern die theoretischen Aussagen in den Kapiteln 4 und 5. Daß die Rannacher-Turek-Diskretisierung für Navier-Stokes Probleme ausgezeichnete Ergebnisse liefert, zeigen ausführliche Vergleichsrechnungen von SCHÄFER ET AL. [60]. Neu ist, daß mit diesem Finiten Element auch im reinen Konvektionsfall zuverlässige Lösungen effizient berechnet werden können.

Dies ist insbesondere deshalb von Interesse, weil Finite Elemente Ansätze zur Lösung der inkompressiblen Euler Gleichungen nicht weit verbreitet sind. Lediglich 2 Literaturstellen konnten in diesem Kontext ausfindig gemacht werden. Während sich JOHNSON [37] auf zweidimensionale Probleme beschränkt und darauf Stromfunktionstechniken anwendet, beschäftigt sich QUARTAPELLE [54] im wesentlichen mit Taylor-Galerkin Ansätzen zur Behandlung instationärer Probleme.

Erste Versuche, vollständig adaptive Verfahren für Navier-Stokes Probleme zu entwickeln, wurden bereits unternommen. BECKER [8] z.B. leitet mit den in Abschnitt 4.6 beschriebenen Methoden einen Fehlerschätzer für eine gemischte Finite Elemente Diskretisierung mit bilinearem Ansatz sowohl für die Geschwindigkeiten als auch für den Druck basierend auf einem Viereckgitter ab. Einen Überblick über a-posteriori Fehlerschätzung gibt VERFÜHRT [79]. Der Zeitpunkt, an dem Fehlerschätzer verfügbar sein werden, die ein für kommerzielle Anwendungen notwendiges Maß an Robustheit garantieren, ist jedoch absehbar. Zumindest werden die Forschungsaktivitäten diesbezüglich gegenwärtig intensiviert, vgl. [23].

Solange solche Regelungsmechanismen noch nicht einsetzbar sind, kann man auf Steuerungskonzepte zur lokalen Netzverfeinerung zurückgreifen. Die Abbildungen 6.12–6.21 zeigen, daß bei reibungsfreier Strömung die Verfeinerung des Netzes dort zu forcieren ist, wo sich Staupunkte ausbilden. Bei reibungsbehafteter Strömung kommen Grenzschichtbereiche hinzu, so daß zu erwarten ist, daß insbesondere in diesem Fall die Vorzüge lokaler Netzverfeinerung deutlich zutage treten

werden.

Interessant ist die Frage, wie *hanging nodes* bei versetzten Gittern zu behandeln sind. McCormick [47] und Becker [8] beschränken ihre Betrachtungen auf konforme Ansätze, bei denen die Freiheitsgrade in den Ecken der Triangulation lokalisiert sind. Im Falle des Rannacher-Turek-Elementes bietet es sich an, mit den beiden Feingitterknoten 1 und 2, vgl. Abbildung 7.1, zu rechnen und den Grobgitterknoten m mit Hilfe der Kontinuitätsbedingung

$$\mathbf{c}_m = (\mathbf{c_1} + \mathbf{c}_2)/2 \tag{7.1}$$

zu eliminieren.

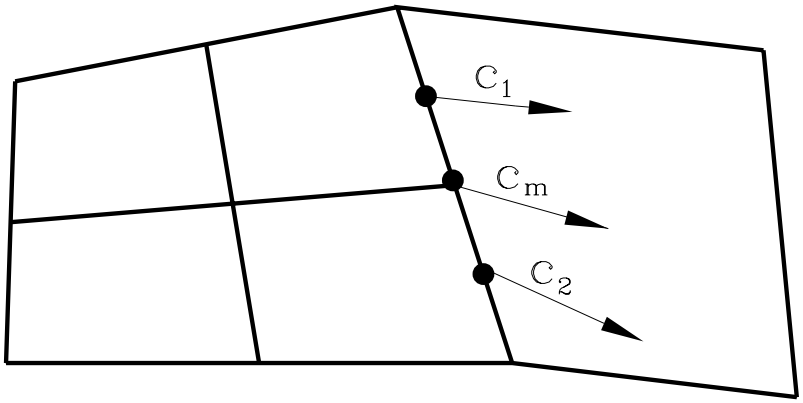

Abb. 7.1: Hanging nodes

Abbildung 7.2 zeigt das Geschwindigkeitsfeld sowie die Druckverteilung für das Busemann-Problem mit $\omega = 0$ auf einem lokal verfeinerten Gitter.

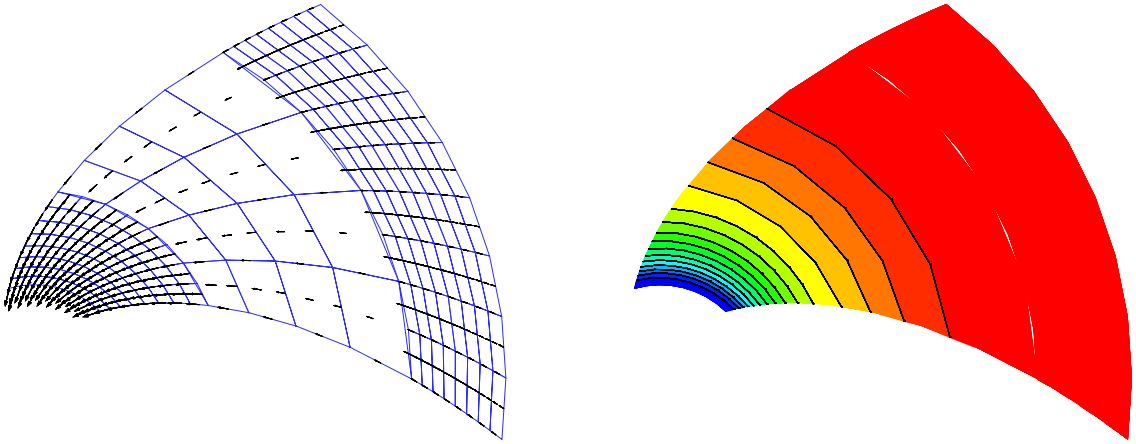

Abb. 7.2: Berechnete Lösung für das Geschwindigkeits- und Druckfeld auf einem lokal verfeinerten Gitter für einen Schnitt senkrecht zur z-Achse

Dieses Gitter wurde erzeugt, indem im ersten Verfeinerungsschritt lediglich zwei der drei in Abbildung 2.1 dargestellten Grobgitterzellen verfeinert und anschließend zwei globale Verfeinerungsschritte durchgeführt wurden. Löcher, die insbesondere im rechten Bild deutlich zu erkennen sind, resultieren daraus, daß auch im Inneren des Gebietes Punkte auf vorgegebene Konturen — hier Zylindermantelflächen — verschoben werden. In Tabelle 7.1 ist die Entwicklung des $L^2(\Omega)$-Geschwindigkeits- bzw. Druckfehlers über der Verfeinerungstiefe aufgelistet.

Level	Zellen	$\|\mathbf{c} - \mathbf{c}_h\|_{L^2(\Omega)}$	$\|p - p_h\|_{L^2(\Omega)}$
0	3	0.47	1.47
1	17	0.41	1.08
2	136	0.17	0.33
3	1088	0.074	0.14

Tab. 7.1: $L^2(\Omega)$-Fehler bei lokaler Verfeinerung

Die $L^2(\Omega)$-Fehler deuten an, daß die Übergangsbedingungen das lineare Konvergenzverhalten nicht beeinträchtigen. Weitere Tests sind aber sicherlich notwendig. Ein geeignetes Multilevelverfahren wird gemäß Abschnitt 4.5.2 organisiert sein, wobei die beiden Feingitterknoten nach einem Vorschlag von ZENGER [82] unter der Nebenbedingung (7.1) bei festgehaltenem $\mathbf{c}_m$ geglättet werden. Dabei wird die in Abschnitt 5.5 vorgestellte Projektionsstrategie innerhalb des BiCGStab-Lösers hilfreiche Dienste leisten.

Ein weiterer Vorteil des BiCGStab ist die sofortige Parallelisierbarkeit. Bei Verfahren hingegen, die wie SIP auf unvollständiger Dreieckszerlegung basieren, ist eine Rechnung auf mehreren Prozessoren nur unter Inkaufnahme zusätzlicher Iterationen möglich. Daß es auch beim BiCGStab Sinn machen kann, mehr Iterationsschritte zuzulassen, um Kommunikationszeiten für den Datentransfer zwischen den einzelnen Prozessoren und damit unter Umständen die Laufzeit insgesamt zu senken, ist eine andere Frage.

Zu implementieren bleiben ein Stromliniendiffusionsverfahren zur Erhöhung der Ordnung sowie eine alternative Strategie zur Berechnung des Druckes entlang der Schaufel in Analogie zum Potentialverfahren. Darüber hinaus bietet sich die Verwendung von Tetraedernetzen an, weil die damit assoziierte zusätzliche Flexibilität eine massive Erleichterung der Grobgittergenerierung in Aussicht stellt.

Bei den drei zuletzt genannten Aspekten handelt es sich um Bausteine, die durch andere ersetzt werden: Upwind durch Stromliniendiffusion, Extrapolation durch Skalierung, hexalaterale Elemente durch Tetraeder. Diese Bausteinphilosophie wird — wie in Abschnitt 1.3 ausführlich erläutert — von der Programmiersprache unterstützt. Es bietet sich deshalb an, weitere wesentliche Bausteine doppelt zu besetzen. Beispielsweise könnte man den SIMPLE- durch eine VANKA-Glätter, vgl. VANKA [77], ersetzen.

Sollte ein Baustein überhaupt nicht oder nicht optimal arbeiten, wird er gegen den zweiten ausgetauscht. Dieses *Rotationsprinzip* könnte man automatisieren und so der Gefahr von Programmabstürzen entgegenwirken. Schließlich ist insbesondere im industriellen Kontext ein möglichst hohes Maß an Robustheit anzustreben.

Außerdem ist es naheliegend, angesichts der zur Verfügung stehenden Gitterhierarchie auf Extrapolationsstrategien zurückzugreifen. Darüber hinaus kann die in den Abschnitten 5.4 und 5.5

erläuterte Projektionstechnik die Behandlung weiterer Zwangsbedingungen erleichtern. Als Beispiel seien nur die Übergangsbedingungen im Zusammenhang mit Rotor-Stator-Kopplung oder Fluid-Struktur-Wechselwirkung genannt. Abschließend ist noch die Weiterentwicklung im Hinblick auf Navier-Stokes Gleichungen zu erwähnen, wobei sich dies im wesentlichen auf Turbulenzmodellierung und eventuell auf die Verwendung von Hybridnetzen bezieht.

7.2 Potentialverfahren

Vor nicht allzu langer Zeit noch die einzigen Modelle, die mit der verfügbaren Rechenleistung zu bewältigen waren, haben zweidimensionale lineare Potentialverfahren mittlerweile deutlich an Bedeutung verloren. Dies gilt in gleicher Weise für das Q3D-Verfahren, das die Lösung auf Stromflächen mit der Lösung auf einer charakteristischen Meridianebene kombiniert. Näheres dazu ist bei SCHILLING [63] nachzulesen. Neue Perspektiven ergeben sich, wenn man über traditionelle Strategien hinaus umfassende Lösungskonzepte erarbeitet und Potentialverfahren darin einbindet.

Einem Vorschlag von ZENGER entsprechend können zweidimensionale lineare für dreidimensionale höherwertige Modelle dieselbe Rolle übernehmen wie grobe Gitter für feine im Rahmen einer Multilevelstrategie. Deshalb macht es auch Sinn, in diesem Zusammenhang von *Multilevel-CFD* zu sprechen.

Grundsätzlich entscheidet bei nichtlinearen Problemen die Qualität der Startwerte nicht nur über die Rechenzeit, sondern auch darüber, ob überhaupt Konvergenz eintritt. So konvergiert das Newton-Verfahren quadratisch, allerdings nur, wenn der Startwert *hinreichend* nahe bei der exakten Lösung liegt, vgl. z.B. STÖR [67]. Eine wesentliche Aufgabe besteht demzufolge darin, geeignete Startwerte zur Verfügung zu stellen. Bei der in Abschnitt 2.2.3 dargestellten Verfeinerungsstrategie geschieht dies, indem nach der Verfeinerung die Grobgitterlösung auf das feinere Gitter interpoliert wird.

Bei linearen Problemen ist die Startwertproblematik entschärft. Die Abschätzung (4.39) zeigt, daß das CG-Verfahren zur Lösung eines linearen Gleichungssystems unabhängig von der Wahl des Startwertes konvergiert, vorausgesetzt die Koeffizientenmatrix ist symmetrisch und positiv definit. Symmetrie und positive Definitheit der Matrix sind aber bei Verwendung einer Standard Finite Elemente Diskretisierung gewährleistet, vgl. Abschnitt 4.3.2.

Das bedeutet: Die Rechenzeit hängt selbstverständlich von der Wahl des Startwertes, von der Kondition der Matrix, von der Qualität des Netzes, von Problemparametern, vom Rechengebiet, von geeigneten Vorkonditionierungsstrategien u.a. ab. Von pathologischen Fällen abgesehen wird das Potentialverfahren jedoch **immer** eine Lösung liefern. Insbesondere im industriellen Kontext ist ein derartiges Verhalten von entscheidender Bedeutung. Insgesamt ist es demnach naheliegend, der Lösung eines nichtlinearen Problems die eines linearen vorzuschalten.

Neben der Startwertproblematik bieten lineare Probleme die Möglichkeit, Techniken zu entwickeln und im Hinblick auf ihre Tauglichkeit zu überprüfen, vgl. Abschnitt 6.1. Die Druckberechnung entlang der Schaufel ist ein Beispiel dafür, die lokale Verfeinerung von Netzen mit Hilfe von a-posteriori Fehlerschätzern ein anderes. Lineare Probleme lassen eine Isolierung des zu testenden Bausteins eher zu. Die Verallgemeinerung auf nichtlineare Probleme gestaltet sich dann in der Regel als nicht besonders schwierig.

7.3 Objektorientierte Paradigmen

Objektorientierte Programmiersprachen stellen die Umsetzung einer Philosophie dar, die weit über die Softwareentwicklung hinaus von Bedeutung ist.

Ein hohes Lohnniveau wird bei fortschreitender Globalisierung künftig nur zu rechtfertigen sein, wenn Arbeitsprozesse unter Ausnutzung von Synergieeffekten optimiert werden und dadurch Produktivität gesteigert wird. Die Aufgabe besteht also darin, aus Prozessen, die scheinbar nichts oder nur wenig miteinander zu tun haben, Gemeinsamkeiten *herauszufaktorisieren* und so zu behandeln, daß deren Lösung im günstigsten Fall als black box Baustein eingebunden werden kann. Dies entspricht der Aufgabenstellung bei Analyse und Design objektorientierter Strukturen, vgl. Booch [10].

Eine Logistik zu hinterlegen, ist immer mit einem gewissen Zeitaufwand verbunden. Auf der anderen Seite müssen Entwicklungszeiten reduziert werden. Insgesamt wird nur der mit der nötigen Spontanität auf Markttendenzen reagieren können, der von einem soliden Fundament aus operiert. Ist dies nicht der Fall, wird man kurzfristigen Entwicklungen erst begegnen können, wenn sie bereits von anderen abgelöst sind. Es ist also beides notwendig: Eine vernünftige Basis und darauf aufbauend schnelle Lösungen.

Unter diesen Gesichtspunkten ist die Bibliothek CONSIST entstanden. Die Grundidee ist, eine Entwicklungsumgebung zur Verfügung zu stellen, auf deren Basis neue Simulationsbausteine insbesondere im Rahmen adaptiver Multilevelstrategien mit möglichst geringem Aufwand getestet werden können. Wie die Kapitel 4 und 5 zeigen, ist es gelungen, einen weiten Bereich an Problemstellungen, Diskretisierungsschemata und Lösungsprozeduren abzudecken. Zusammenfassend kann man festhalten: **Begreift man Abstraktion als nützliches Hilfsmittel, können objektorientierte Paradigmen wesentlich zur Produktivitätssteigerung beitragen.**

Literaturverzeichnis

[1] Bank, R.: *PLTMG: A Software Package for Solving Elliptic Partial Differential Equations, User's Guide 6.0*. SIAM, 1990

[2] Bank, R. ; Rose, D.: Some Error Estimates for the Box Method. In: *SIAM J. Numer. Anal.* 24 (1987), S. 777–787

[3] Bank, R. ; Sherman, A. ; Weiser, A.: Refinement Algorithms and Data Structures for Regular Local Mesh Refinement. In: Stepleman, R. (Hrsg.): *Scientific Computing*. North-Holland, 1983, S. 3–17

[4] Bank, R ; Weiser, A.: Some a-posteriori Error Estimators for Elliptic Partial Differential Equations. In: *Math. Comp.* 44 (1985), S. 283–301

[5] Bank, R. ; Welfert, B. ; Yserentant, H.: A Class of Iterative Methods for Solving Saddle Point Problems. In: *Numerische Mathematik* 56 (1990), S. 645–666

[6] Barrett, R. ; Berry, M. ; Chan, T. F. ; Demmel, J. ; Donato, J. ; Dongarra, J. ; Eijkhout, V. ; Pozo, R. ; Romine, C. ; Van der Vorst, H.: *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods*. 2. Auflage. SIAM, 1994

[7] Bastian, P.: *Parallele Adaptive Mehrgitterverfahren*, Ruprecht-Karls-Universität Heidelberg, Diss., 1994

[8] Becker, R.: *An Adaptive Finite Element Method for the Incompressible Navier-Stokes Equations on Time-Dependent Domains*, Universität Heidelberg, Diss., 1995

[9] Bey, J.: Tetrahedral Grid Refinement. In: *Computing* 55 (1995), S. 355–378

[10] Booch, G.: *Object Oriented Analysis and Design with Applications*. 2. Auflage. Benjamin/Cummings, 1994

[11] Braess, D.: *Finite Elemente*. Springer Verlag, 1992

[12] Bramble, J. ; Pasciak, J. ; Xu, J.: Parallel Multilevel Preconditioners. In: *Math. Comp.* 55 (1990), S. 1–22

[13] Brandt, A. Multigrid Techniques 1984: Guide with Applications to Fluid Mechanics. GMD Studien 85. 1984

[14] Brenner, S. ; Scott, L.: *The Mathematical Theory of Finite Element Methods*. Springer Verlag, 1994

[15] Brezzi, F. ; Fortin, M.: *Mixed and Hybrid Finite Element Methods*. Springer Verlag, 1991

[16] Bronstein, I. N. ; Semendjajew, K. A.: *Taschenbuch der Mathematik*. Verlag Harri Deutsch, 1981

[17] Bronstein, I. N. ; Semendjajew, K. A.: *Ergänzende Kapitel zum Taschenbuch der Mathematik*. Verlag Harri Deutsch, 1991

[18] Chorin, A.: Numerical Solution of the Navier-Stokes Equations. In: *Math. Comp.* 22 (1968), S. 745–762

[19] Chorin, A. ; Marsden, J.: *A Mathematical Introduction to Fluid Mechanics*. Springer Verlag, 1993

[20] Ciarlet, P.: *The Finite Element Method for Elliptic Problems*. North-Holland, 1979

[21] Daehlen, M. (Hrsg.) ; Tveito, A. (Hrsg.): *Numerical Methods and Software Tools in Industrial Mathematics*. Birkhäuser, 1997

[22] Feistauer, M.: *Mathematical Methods in Fluid Dynamics*. Longman Scientific & Technical, 1993

[23] *GAMM Workshop: Adaptive Methods — Error Estimators.* Kiel, Jan. 2000

[24] Girault, V. ; Raviart, P.: *Finite Element Methods for Navier-Stokes Equations*. Springer Verlag, 1986

[25] Glowinski, R.: *Numerical Methods for Nonlinear Variational Problems*. Springer, 1984

[26] Gostelow, J. P.: *Cascade Aerodynamics*. 1. Auflage. Pergamon Pr., 1984

[27] Griebel, M.: *Multilevelmethoden als Iterationsverfahren über Erzeugendensystemen*. Teubner, 1994

[28] Griebel, M. ; Dornseifer, T. ; Neunhoeffer, T.: *Numerische Simulation in der Strömungsmechanik*. Vieweg, 1995

[29] Grossmann, C. ; Roos, H.: *Numerik partieller Differentialgleichungen*. 2. Auflage. Teubner, 1994

[30] Gunzburger, M.: Navier-Stokes Equations for Incompressible Flows: Finite-Element Methods. In: Peyret, R. (Hrsg.): *Handbook of Computational Fluid Mechanics*. Academic Press, 1996, S. 99–157

[31] Hackbusch, W.: *Multi-Grid Methods and Applications*. Springer Verlag, 1985

[32] Hackbusch, W.: *Theorie und Numerik elliptischer Differentialgleichungen*. Teubner Studienbücher, 1986

[33] Hackbusch, W.: *Iterative Lösung großer schwachbesetzter Gleichungssysteme*. Teubner Studienbücher, 1991

[34] Hiptmair, R.: Object Oriented Concepts for an Adaptive Finite Element Code / TU München. 1995. – Forschungsbericht

[35] Hoppe, R. W. H.: Finite Elemente / TU München. 1994. – Vorlesung

[36] Huckle, T.: Numerik auf Parallelrechnern / TU München. 1997/1998. – Vorlesung

[37] Johnson, C.: *Numerical Solution of Partial Differential Equations by the Finite Element Method*. Cambridge University Press, 1987

[38] Josuttis, N.: *Objektorientiertes Programmieren in C++*. 1. Auflage. Addison-Wesley, 1995

[39] Kallinderis: Grid Generation / TU München, Lehrstuhl A für Thermodynamik. 1999. – Vortrag

[40] Koecher, M.: *Lineare Algebra und Analytische Geometrie*. 4. Auflage. Springer Verlag, 1997

[41] Kopka, H.: *LATEX*. Bd. 1. 1. Auflage. Addison-Wesley, 1994

[42] Leinen, P.: Data Structures and Concepts for Adaptive Finite Element Methods. In: *Computing* 55 (1995), S. 325–354

[43] LeVeque, R.: *Numerical Methods for Conservation Laws*. 2. Auflage. Birkhäuser, 1992 (Lectures in Mathematics)

[44] Lippman, S.: *C++. Einführung und Leitfaden*. 2. Auflage. Addison-Wesley, 1991

[45] Marchioro, C. ; Pulvirenti, M.: *Mathematical Theory of Incompressible Nonviscous Fluids*. Springer Verlag, 1994

[46] Marsden, J ; Hughes, T.: *Mathematical Foundations of Elasticity*. Prentice Hall, 1983

[47] McCormick, S.: *Multilevel Adaptive Methods for Partial Differential Equations*. Soc. for Industrial and Applied Math., 1989 (Frontiers in Applied Mathematics 6)

[48] Meyberg, K. ; Vachenauer, P.: *Höhere Mathematik*. Bd. 1. 5. Auflage. Springer Verlag, 1999

[49] Meyers, S.: *Effektiv C++ programmieren*. 2. Auflage. Addison-Wesley, 1995

[50] Müller, A.: *Adaptive Multilevelverfahren zur Numerischen Lösung der Stokes-Gleichungen*, TU München, Diplomarbeit, 1995

[51] Müller, A. ; Szilagyi, B.: Nonconforming Finite Element Approximation of Steady, Incompressible 3D Euler Equations in a Rotating Frame of Reference / TU München, Lehrstuhl für Hydraulische Maschinen und Anlagen. 2000. – Interner Bericht

[52] Oswald, P.: *Multilevel Finite Element Approximation. Theory and Application*. Teubner, 1994 (Skripten zur Numerik)

[53] Pfeiffer, F.: *Einführung in die Dynamik*. Teubner Studienbücher, 1992 (Leitfäden der angewandten Mathematik und Mechanik 65)

[54] Quartapelle, L.: *Numerical Solution of the Incompressible Navier-Stokes Equations*. Birkhäuser, 1993

[55] Rannacher, R. ; Turek, S.: A Simple Nonconforming Quadrilateral Stokes Element. In: *Numer. Part. Diff. Equ.* 8 (1992), S. 97–111

[56] Reinelt, R.: Netzgenerierung mit Hilfe von Verfeinerungstechniken / TU München, Lehrstuhl für Hydraulische Maschinen und Anlagen. 2000. – Interner Bericht

[57] Rüde, U.: *Mathematical and Computational Techniques for Multilevel Adaptive Methods*. Soc. for Industrial and Applied Math., 1993 (Frontiers in Applied Mathematics 13)

[58] Rüde, U. ; Zenger, C.: A Workbench for Multigrid Methods / TU München, Inst. für Informatik. 1986 (TUM-I 8607). – Bericht

[59] Ruge, J. ; Stüben, K.: Algebraic Multigrid. In: McCormick, S. (Hrsg.): *Multigrid Methods*. Soc. for Industrial and Applied Math., 1987 (Frontiers in Applied Mathematics 3), S. 73–130

[60] Schäfer, M ; Turek, S. ; Durst, F. ; Krause, E ; Rannacher, R.: Benchmark Computations of Laminar Flow Around a Cylinder. In: Hirschel, E. (Hrsg.): *Notes on Numerical Fluid Mechanics* Bd. 52. Vieweg, 1996, S. 547–566

[61] Schilling, R.: Regelungstechnik. 1993/1994. – Vorlesungsskriptum

[62] Schilling, R.: Rechnergestützte Entwicklung von Strömungsmaschinen / TU München. 1998. – Vorlesungsskriptum

[63] Schilling, R. ; Watzelt, C. ; Haas, H.: A Fast CAE/CAD-Procedure for the Optimum Design of Arbitrary Impellers. In: Kim, J. H. (Hrsg.): *Proceedings of ISROMAC-3*. Honolulu : EPRI, April 1990, S. 480–494

[64] Schreiber, P.: *Eine nichtkonforme Finite-Elemente-Methode zur Lösung der inkompressiblen 3-D Navier-Stokes Gleichungen*, Ruprecht-Karls-Universität Heidelberg, Diss., 1996

[65] Schwarz, H.: *Methode der Finiten Elemente*. 3. Auflage. Teubner, 1991

[66] Segal, G. ; Vuik, K. ; Kassels, K.: On the Implementation of Symmetric and Antisymmetric Periodic Boundary Conditions for Incompressible Flow. In: *International Journal for Numerical Methods in Fluids* 18 (1994), S. 1153–1165

[67] Stör, J.: *Numerische Mathematik*. Bd. 1. 5. Auflage. Springer Verlag, 1989

[68] Stör, J. ; Bulirsch, R.: *Numerische Mathematik*. Bd. 2. 3. Auflage. Springer Verlag, 1990

[69] Strang, G ; Fix, G.: *An Analysis of the Finite Element Method*. Prentice-Hall, 1973

[70] Stroustrup, B.: *Die C++-Programmiersprache*. 3. Auflage. Addison-Wesley-Longman, 1998

[71] Temam, R.: *Theory and Numerical Analysis of the Navier-Stokes Equations*. North-Holland, 1977

[72] Thomasset, F.: *Implementation of Finite Element Methods for Navier-Stokes Equations*. Springer, 1981

[73] Truesdell, C.: *A First Course in Rational Continuum Mechanics*. Bd. 1. Academic Press, 1977

[74] Turek, S.: *Efficient Solvers for Incompressible Flow Problems. An Algorithmic and Computational Approach*. Springer, 1999

[75] Van Kan, J.: A Second-Order Accurate Pressure-Correction Scheme for Viscous Incompressible Flow. In: *SIAM J. Sci. Stat. Comp.* 7 (1986), S. 870–891

[76] Van Kan, J. ; Segal, A.: *Numerik partieller Differentialgleichungen für Ingenieure*. Teubner, 1995

[77] Vanka, S.: Implicit Multigrid Solutions of Navier-Stokes Equations in Primitive Variables. In: *J. Comp. Phys.* 65 (1985), S. 138–158

[78] Verführt, R.: Finite Element Approximation of Incompressible Navier-Stokes Equations with Slip Boundary Condition. In: *Numerische Mathematik* 50 (1987), S. 697–721

[79] Verführt, R.: *A Review of a-posteriori Error Estimation and Adaptive Mesh-Refinement Techniques*. Teubner, 1996

[80] Xu, J.: Iterative Methods by Space Decomposition and Subspace Correction. In: *SIAM Review* 34 (1992), S. 581–613

[81] Yserentant, H.: Old and New Convergence Proofs for Multigrid Methods. In: *Acta Numerica* (1993), S. 285–326

[82] Zenger, C. Persönliche Mitteilungen. 1999