Gewichtete wesentlich nicht-oszillierende Verfahren auf unstrukturierten Gittern

Dissertation
zur Erlangung des Doktorgrades
des Fachbereichs Mathematik
der Universität Hamburg

vorgelegt von

Oliver Friedrich

aus Warburg

Hamburg 1999

Als Dissertation angenommen vom Fachbereich
Mathematik der Universität Hamburg

auf Grund der Gutachten von Prof. Dr. Thomas Sonar
und Prof. Dr. Klaus Glashoff

Hamburg, den 15. Juli 1999

Prof. Dr. Hans Daduna
Dekan des Fachbereichs Mathematik

für Alexandra

Inhaltsverzeichnis

Vorwort

Die vorliegende Arbeit entstand am Fachbereich Mathematik der Universität Hamburg. Bei ihrer Erstellung habe ich von vielen Seiten persönliche und fachliche Unterstützung erhalten, für die ich mich hiermit bedanken möchte.
Mein größter Dank gilt Professor Dr. Thomas Sonar, der meine wissenschaftliche Laufbahn wesentlich unterstützt hat. Ich möchte ihm hiermit herzlich für die schönen, gemeinsamen Jahre in Göttingen und Hamburg danken. Seine vielen Anregungen und die unangenehmen, aber sehr hilfreichen Aufforderungen, das eine oder andere Teilergebnis doch endlich einmal aufzuschreiben, haben mir immer wieder neuen Antrieb gegeben und damit viel zur Erstellung dieser Arbeit beigetragen.
Herrn Professor Dr. Klaus Glashoff danke ich für die freundliche Übernahme des Korreferates.
Allen Kollegen vom Fachbereich Mathematik danke ich für die Unterstützung meiner Arbeit und für die angenehme Zeit am Institut für Angewandte Mathematik während der Entstehung.
Daniel Hempel danke ich für die erfreuliche Zusammenarbeit in Göttingen und in Hamburg. Er hat insbesondere die Adaptionsroutinen in den gemeinsam entwickelten parallelen, adaptiven Strömungslöser eingebracht, die ich hervorragend für meine Berechnungen einsetzen konnte. Außerdem danke ich ihm für das kritische Korrekturlesen dieser Arbeit.
Arne Ahrend und Friederike Schröder-Pander danke ich für die vielen kleinen Anregungen und Bemerkungen, die mir in der Summe viel geholfen haben.
Ein besonderer Dank gilt meiner Frau Alexandra, der ich diese Arbeit gewidmet habe. Während der Entstehung der Arbeit hatte sie immer wieder darunter zu leiden, wenn es nicht zu meiner Zufriedenheit weiterging. In diesen Phasen war das Zusammenleben mit mir sicher nicht leicht. Vielen Dank für die Geduld.

Oliver Friedrich

Einleitung

Durch den Einsatz von numerischen Verfahren auf Computern ist es möglich Erhaltungsgleichungen, wie sie insbesondere in der Strömungsmechanik vorkommen, approximativ zu lösen. Man kann von *numerischen Experimenten* im Gegensatz zu Experimenten wie Messungen an einem Modell im Windkanal sprechen. Damit die numerischen Experimente konkurrenzfähig sind, müssen die verwendeten Simulationsprogramme möglichst genau und schnell sein.
In dieser Arbeit beschäftigen wir uns mit der numerischen Lösung von Erhaltungsgleichungen durch Finite-Volumen-Verfahren. Wir beschränken uns dabei vollständig auf den ebenen Fall. Alle Konzepte, bis auf die Schablonenauswahl in Abschnitt 2.2.3, lassen sich unmittelbar auf den dreidimensionalen Fall übertragen.
Finite-Volumen-Verfahren sind ein etabliertes Mittel zur Diskretisierung von Erhaltungsgleichungen. Durch die räumliche Diskretisierung mit sogenannten unstrukturierten Gittern erhält man ein hohes Maß an Flexibilität.
Um die Genauigkeit zu erhöhen, die beim Grundverfahren für die Anwendungen üblicherweise nicht ausreichend ist, werden Rekonstruktionsalgorithmen eingesetzt. Mit deren Hilfe werden aus den vorhandenen Daten Funktionen, in unserem Fall immer stückweise polynomial, rekonstruiert. Da die Lösungen bei den betrachteten Gleichungen unstetig sein können, müssen Vorkehrungen getroffen werden um Oszillationen der rekonstruierten Funktionen zu vermeiden.
Eine spezielle Klasse von Rekonstruktionsalgorithmen sind die sogenannten *wesentlich nicht oszillierenden Verfahren (ENO-Verfahren*[1]*).* Dabei werden zur Berechnung einer lokalen Rekonstruktionsfunktion mehrere Kandidaten berechnet um denjenigen auszuwählen, der am wenigsten oszilliert. Diese Verfahren können dadurch verbessert werden, dass die digitale Auswahl durch eine Konvexkombination aller Kandidaten ersetzt wird. In dieser Konvexkombination werden die einzelnen Rekonstruktionsfunktionen so gewichtet, dass diejenigen mit hoher Oszillation deutlich geringer eingehen als die-

[1] essentially non-oscillatory schemes

jenigen mit niedriger Oszillation. Man nennt diese Verfahren *gewichtete wesentlich nicht oszillierende Verfahren (WENO-Verfahren*[2]*).*
WENO-Verfahren wurden für den eindimensionalen Fall in [LOC94] und [JS96] vorgestellt und von uns auf den Fall von unstrukturierten Gittern übertragen (siehe auch [Fri98]).
Insgesamt ergeben sich so aus den Finite-Volumen-Verfahren sehr robuste und genaue Verfahren, die leider mit steigender Genauigkeit sehr rechenaufwendig sind.
Es ist aber nicht für das gesamte Rechengebiet dieselbe Feinheit der Diskretisierung erforderlich. Die Lösungen sind üblicherweise in großen Bereichen des Rechengebietes regulär. In diesen Bereichen ist kein besonders feines Gitter nötig.
Diesen Sachverhalt machen sich *adaptive Verfahren* zunutze, die — möglichst automatisch — das Rechengitter dort verfeinern, wo es zu grob ist, und es dort vergröbern, wo es unnötig fein ist.
Ein alternativer Ansatz, die *verallgemeinerte Mehrskalenanalyse*, wurde in [Har91] für den eindimensionalen Fall vorgestellt. Dabei wird ein global feines Gitter verwendet. Zusätzlich wird eine Folge von sukzessive gröberen Gittern eingesetzt. Vor jeder Flussberechnung des Finite-Volumen-Verfahrens werden die diskreten Daten, die auf dem feinen Gitter vorliegen, im Hinblick auf die erforderliche Diskretisierungsfeinheit analysiert. In Bereichen, in denen ein gröberes Gitter ausreicht, wird das aufwendige WENO-Verfahren auf einem der groben Gitter verwendet und das Ergebnis durch eine wenig aufwendige Operation auf das feine Gitter prolongiert. Nur in den Gebieten, in denen die groben Gitter nicht ausreichen, wird das WENO-Verfahren direkt auf dem feinen Gitter angewandt.
Die vorliegende Arbeit untergliedert sich in vier Kapitel. Im ersten Kapitel werden zunächst die betrachteten Erhaltungsgleichungen vorgestellt. Dann wird die Konstruktion eines Finite-Volumen-Verfahrens auf unstrukturierten Gittern beschrieben. Im zweiten Kapitel wird die Technik der polynomialen Rekonstruktion dargestellt und analysiert. Dabei wird insbesondere die algorithmische Realisierung der WENO-Rekonstruktion ausführlich beschrieben. Im dritten Kapitel werden die angesprochenen Beschleunigungstechniken dargestellt und in Bezug auf ihre Effizienz gegenübergestellt. Ein prinzipielles Problem bei der verallgemeinerten Mehrskalenanalyse wird analysiert. Im vierten Kapitel werden numerische Ergebnisse präsentiert. Dabei wird der Einfluss verschiedener Parameter des Verfahrens untersucht und seine Leistungsfähigkeit demonstriert.

[2] weighted essentially non-oscillatory schemes

Kapitel 1

Finite-Volumen-Verfahren

1.1 Betrachtete Gleichungen

Wie bereits erwähnt, beschäftigen wir uns mit Erhaltungsgleichungen in zwei Raumdimensionen. Eine *Erhaltungsgröße* oder *Zustandsgröße* u ist als Funktion von Raum und Zeit gegeben:

$$u : \Omega \times \mathbb{R}_0^+ \to S \subset \mathbb{R}^m.$$

Dabei ist $x \in \Omega$, $\Omega \subset \mathbb{R}^2$ der Ort und $t \in \mathbb{R}_0^+$ die Zeit. Ω ist ein beschränktes Gebiet mit Lipschitz-stetigem Rand $\partial\Omega$. S nennt man den Zustandsraum.
Eine Erhaltungsgleichung kann man so interpretieren, dass sie, ausgehend von einem Startzustand $u(x, 0)$, die Änderung von u im Verlauf der Zeit beschreibt. Die Änderung wird mittels sogenannter *Kontrollvolumina* beschrieben. Ein Kontrollvolumen $C \subset \Omega$ ist ein beschränktes Gebiet mit Lipschitz-stetigem Rand ∂C.
Die zeitliche Veränderung von u auf einem Kontrollvolumen C wird als Informationsfluss über den Rand von C dargestellt:

Definition 1.1.1. Eine Gleichung der Form

$$\frac{d}{dt}\int_C u(x,t)dx = -\int_{\partial C} f(u(x,t)) \cdot n \; ds \tag{1.1}$$

heißt *Erhaltungsgleichung (in integraler Form).* $n = (n_1, n_2)^T$ ist dabei der äußere Einheitsnormalenvektor auf dem Rand des Kontrollvolumens C. $f = (f_1, f_2)$ heißt *Flusstensor*, die Komponenten $f_1, f_2 \in C^1(S; \mathbb{R}^m)$ heißen *Flussfunktionen.* Im Fall $m = 1$ spricht man von *skalaren Gleichungen*, sonst von *Systemen.*

Zusätzlich zu Gleichung (1.1) werden für $x \in \partial\Omega$ Randbedingungen an $u(x,t)$ gestellt. Darauf gehen wir später genauer ein.
Für Systeme wird die folgende Begriffsbildung eingeführt:

Definition 1.1.2. Die Erhaltungsgleichung (1.1) heißt *hyperbolisch*, wenn für alle $u \in S$ und $\nu \in \mathbb{R}^2$ gilt, dass die Matrix

$$A(u,\nu) := \nabla_u f(u) \cdot \nu = \nabla_u f_1(u)\nu_1 + \nabla_u f_2(u)\nu_2 \subset \mathbb{R}^{m\times m}$$

m reelle Eigenwerte $\lambda_1(u,\nu) \leq \lambda_2(u,\nu) \leq \ldots \leq \lambda_m(u,\nu)$, die stetig von u abhängen, und m linear unabhängige Eigenvektoren $r_j(u,\nu)$ besitzt, d.h.:

$$A(u,\nu)r_j(u,\nu) = \lambda_j(u,\nu)r_j(u,\nu), \quad j = 1,\ldots,m.$$

Hyperbolische Erhaltungsgleichungen zeichnen sich unter anderem dadurch aus, dass ihre Lösungen Unstetigkeiten ausbilden und transportieren können. In dieser Arbeit werden konkret drei verschiedene Erhaltungsgleichungen betrachtet:

Lineare Advektionsgleichung Eine sehr einfache skalare Gleichung ist die *lineare Advektionsgleichung*. Die Flussfunktionen sind gegeben durch

$$f_1(u) = a_1 \cdot u, \quad f_2(u) = a_2 \cdot u \tag{1.2}$$

mit $a_1, a_2 \in \mathbb{R}$. Für die Rechnungen wurde jeweils $a_1 = a_2 = 1$ gesetzt.

Burgers Gleichung Eine weitere skalare Gleichung ist *Burgers Gleichung*. Die Flussfunktionen sind hier gegeben durch

$$f_1(u) = a_1 \cdot u^2, \quad f_2(u) = a_2 \cdot u^2 \tag{1.3}$$

mit $a_1, a_2 \in \mathbb{R}$. Für die Rechnungen wurde hier $a_1 = \frac{1}{2}$ und $a_2 = 0$ gesetzt.

Euler-Gleichungen Das sicherlich interessanteste Beispiel ist die Betrachtung der Bewegungsgleichungen kompressibler Fluide unter Vernachlässigung der Reibung, die *Euler-Gleichungen*. Hier handelt es sich um ein hyperbolisches System mit vierdimensionalem Zustandsraum. Der Vektor der Zustandsvariablen ist gegeben durch

$$u = \begin{pmatrix} \rho \\ \rho \mathsf{v}_1 \\ \rho \mathsf{v}_2 \\ \rho \mathsf{E} \end{pmatrix}.$$

Dabei heißt ρ die Dichte des Fluids, $\mathsf{v} = (\mathsf{v}_1, \mathsf{v}_2)^T$ die Geschwindigkeit und E die Totalenergie. Die Flussfunktionen sind gegeben als

$$f_1(u) = \begin{pmatrix} \rho\mathsf{v}_1 \\ \rho\mathsf{v}_1^2 + \mathsf{p} \\ \rho\mathsf{v}_1\mathsf{v}_2 \\ (\rho\mathsf{E} + \mathsf{p})\mathsf{v}_1 \end{pmatrix}, \quad f_2(u) = \begin{pmatrix} \rho\mathsf{v}_2 \\ \rho\mathsf{v}_1\mathsf{v}_2 \\ \rho\mathsf{v}_2^2 + \mathsf{p} \\ (\rho\mathsf{E} + \mathsf{p})\mathsf{v}_2 \end{pmatrix}, \tag{1.4}$$

wobei p der Druck ist. Der Druck ist durch die Zustandsgleichung für ideales Gas gegeben:

$$\mathsf{p} = (\gamma - 1)\rho(\mathsf{E} - \frac{|\mathsf{v}|^2}{2}). \tag{1.5}$$

Die Konstante γ wird unter der Annahme trockener Luft auf 1.4 gesetzt.

Bevor wir die Eigenwerte und Eigenvektoren von $A(u, \nu)$ für die Euler-Gleichungen angeben ist es sinnvoll zwei weitere Größen zu definieren: Die durch

$$\mathsf{a} := \sqrt{\gamma \frac{\mathsf{p}}{\rho}} \tag{1.6}$$

gegebene Größe heißt *Schallgeschwindigkeit*, die Größe

$$\mathsf{Ma} := \frac{|\mathsf{v}|}{\mathsf{a}} \tag{1.7}$$

nennt man *Machzahl*.

Bemerkung 1.1.3. Für die Flussfunktionen der Euler-Gleichungen (1.4) lässt sich die Matrix

$$A(u, \nu) := \nabla_u f_1(u)\nu_1 + \nabla_u f_2(u)\nu_2$$

durch die Transformation

$$\Lambda(u, \nu) := P^{-1}(u, \nu)A(u, \nu)P(u, \nu)$$

diagonalisieren. Bei der Angabe der Eigenwerte und der Diagonalisierungsmatrizen haben wir aus Gründen der Übersichtlichkeit angenommen, dass $\nu = n$ mit $|n| = 1$.
Die Diagonalmatrix $\Lambda(u, n)$ lautet:

$$\Lambda(u, n) = \mathrm{diag}(\mathsf{v} \cdot n, \mathsf{v} \cdot n, \mathsf{v} \cdot n + \mathsf{a}, \mathsf{v} \cdot n - \mathsf{a}),$$

die Matrix P und ihre Inverse lauten:

$$P(u,n) = \begin{pmatrix} 1 & 0 & \frac{\rho}{2\mathsf{a}} & \frac{\rho}{2\mathsf{a}} \\ \mathsf{v}_1 & \rho n_2 & \frac{\rho}{2\mathsf{a}}(\mathsf{v}_1 + \mathsf{a}n_1) & \frac{\rho}{2\mathsf{a}}(\mathsf{v}_1 - \mathsf{a}n_1) \\ \mathsf{v}_2 & -\rho n_1 & \frac{\rho}{2\mathsf{a}}(\mathsf{v}_2 + \mathsf{a}n_2) & \frac{\rho}{2\mathsf{a}}(\mathsf{v}_2 - \mathsf{a}n_2) \\ \frac{|\mathsf{v}|^2}{2} & \rho(\mathsf{v}_1 n_2 - \mathsf{v}_2 n_1) & \frac{\rho}{2\mathsf{a}}(\mathsf{E} + \frac{\mathsf{p}}{\rho} + \mathsf{a}\mathsf{v}\cdot n) & \frac{\rho}{2\mathsf{a}}(\mathsf{E} + \frac{\mathsf{p}}{\rho} - \mathsf{a}\mathsf{v}\cdot n) \end{pmatrix}$$

und

$$P^{-1}(u,n) = \begin{pmatrix} 1 - \frac{\gamma-1}{2}\mathsf{Ma}^2 & (\gamma-1)\frac{\mathsf{v}_1}{\mathsf{a}^2} & (\gamma-1)\frac{\mathsf{v}_2}{\mathsf{a}^2} & \frac{1-\gamma}{a^2} \\ \frac{1}{\rho}(\mathsf{v}_2 n_1 - \mathsf{v}_1 n_2) & \frac{n_2}{\rho} & -\frac{n_1}{\rho} & 0 \\ \frac{\mathsf{a}}{\rho}(\frac{\gamma-1}{2}\mathsf{Ma}^2 - \frac{\mathsf{v}\cdot n}{\mathsf{a}}) & \frac{1}{\rho}(n_1 - (\gamma-1)\frac{\mathsf{v}_1}{\mathsf{a}}) & \frac{1}{\rho}(n_2 - (\gamma-1)\frac{\mathsf{v}_2}{\mathsf{a}}) & \frac{\gamma-1}{\rho\mathsf{a}} \\ \frac{\mathsf{a}}{\rho}(\frac{\gamma-1}{2}\mathsf{Ma}^2 + \frac{\mathsf{v}\cdot n}{\mathsf{a}}) & -\frac{1}{\rho}(n_1 + (\gamma-1)\frac{\mathsf{v}_1}{\mathsf{a}}) & -\frac{1}{\rho}(n_2 + (\gamma-1)\frac{\mathsf{v}_2}{\mathsf{a}}) & \frac{\gamma-1}{\rho\mathsf{a}} \end{pmatrix}$$

Wir wollen uns in dieser Arbeit nicht mit Fragen zur Existenz oder Eindeutigkeit der behandelten Gleichungen beschäftigen. Stattdessen werden wir uns immer auf den Standpunkt stellen, dass für die betrachteten Beispiele eine eindeutige Lösung existiert.

Bemerkung 1.1.4. Bereits bei einfachen Beispielen wie Burgers Gleichung können sich nach endlicher Zeit t aus glatten Anfangsdaten $u(x,0)$ Daten $u(x,t)$ mit Unstetigkeiten bezüglich x ergeben (siehe zum Beispiel [MM94]).

Dies muss bei der Entwicklung von numerischen Verfahren berücksichtigt werden.
Die zur numerischen Behandlung erforderliche Diskretisierung wird in Raum- und Zeitkomponente getrennt durchgeführt.

1.2 Räumliche Diskretisierung

1.2.1 Geometriebeschreibung

Der Rand $\partial\Omega$ von Ω sei ab jetzt stets polygonal.
Bei der folgenden Begriffsbildung orientieren wir uns an der Finite-Elemente Literatur (siehe [Cia78]):

Definition 1.2.1. Ein *Gitter*[1] $\mathcal{G}$ von Ω ist eine endliche Menge

$$\mathcal{G} = \{C_1, \ldots, C_{\#\mathcal{G}}\}$$

von abgeschlossenen Teilmengen $C_i \subset \bar{\Omega}, i = 1, \ldots, \#\mathcal{G}$, für die gilt:

[1] In [Cia78] wird so der Begriff *Triangulierung* statt *Gitter* definiert. Wir wollen unter einer *Triangulierung* eine Menge von Dreiecken verstehen (siehe Definition 1.2.2)

1. $\bar{\Omega} = \bigcup_{i=1,\dots,\#\mathcal{G}} C_i$.

2. Jedes C_i hat ein nicht-leeres Inneres.

3. Für je zwei C_i und C_j mit $i \neq j$ gilt: $\overset{\circ}{C}_i \cap \overset{\circ}{C}_j = \emptyset$.

4. Der Rand ∂C_i aller C_i ist polygonal[2]. Das heißt, zu jedem C_i gibt es einen geschlossenen Polygonzug von endlich vielen Liniensegmenten Γ_{ij}, so dass

$$\partial C_i = \bigcup_{j=1}^{\#\Gamma_i} \Gamma_{ij}.$$

Die C_i nennen wir *Zellen*.
Für jede Zelle C_i bezeichnen wir mit $h(C_i)$ den *Durchmesser* und mit $|C_i|$ den *Flächeninhalt*:

$$h(C_i) := \sup_{x,y \in C_i} |x - y|, \tag{1.8}$$

$$|C_i| := \int_{C_i} 1 dx. \tag{1.9}$$

Die Zellen eines Gitters bilden die diskreten Kontrollvolumina.

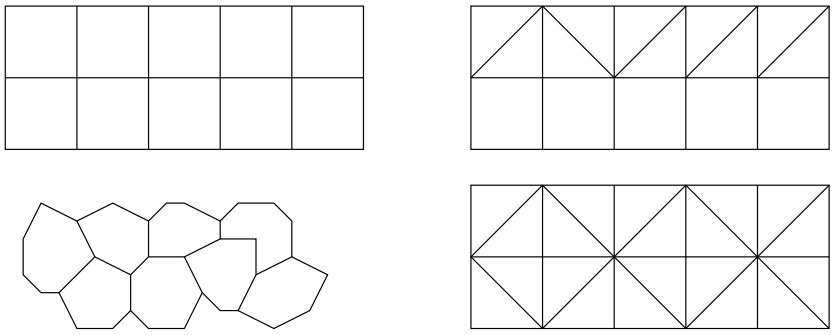

Abbildung 1.1: Beispiele für Gitter.

Ein wichtiger Spezialfall von Gittern sind die *Triangulierungen*:

Definition 1.2.2. Eine *Triangulierung* $\mathcal{T}$ von Ω ist eine endliche Menge von Dreiecken $T_i \subset \bar{\Omega}, i = 1, \dots, \#\mathcal{T}$, für die gilt:

[2] Diese Einschränkung wird aus praktischen Gründen vorgenommen. In [Cia78] wird nur verlangt, dass ∂C_i Lipschitz-stetig ist.

1. $\bar{\Omega} = \bigcup_{i=1,\dots,\#\mathcal{T}} T_i$.

2. Jedes Dreieck T_i hat ein nicht-leeres Inneres.

3. Für je zwei Dreiecke T_i und T_j mit $i \neq j$ gilt: $\overset{\circ}{T}_i \cap \overset{\circ}{T}_j = \emptyset$.

4. Eine Kante K eines Dreiecks T_i ist entweder Teilmenge von $\partial\Omega$ oder es gibt genau ein Dreieck T_j, $j \neq i$, so dass K Kante von T_j ist.

Die letzte Bedingung heißt *Konformitätsbedingung* (siehe Abbildung 1.2).

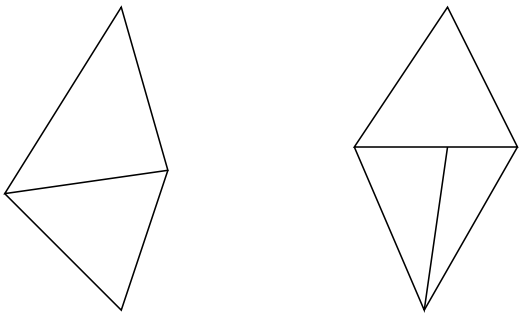

Abbildung 1.2: Konforme (links) und nicht-konforme (unzulässige) Nachbarschaft von Dreiecken.

In der Ingenieursliteratur ist die Verwendung einer anderen Art von Gittern, welche aus Triangulierungen konstruiert werden, üblich. Die Konstruktionsidee beruht auf klassischen topologischen Betrachtungen (siehe dazu [AH35]):

Definition 1.2.3. Zu jedem Dreieckspunkt P_i einer Triangulierung konstruiert man eine Zelle C_i wie folgt: Man verbindet die Mittelpunkte aller Kanten, die in P_i enden, mit den Schwerpunkten der an die Kante angrenzenden Dreiecke. Für solche Kanten, an die nur ein Dreieck angrenzt, verbindet man den Mittelpunkt der Kante mit P_i. Der so entstandene geschlossene Polygonzug bildet den Rand von C_i. Das durch die Menge der C_i gegebene Gitter heißt *baryzentrische Unterteilung der Triangulierung.*

Bemerkung 1.2.4. Die baryzentrische Unterteilung einer Triangulierung hat eine Eigenschaft, die vielen anderen Gittern, insbesondere den Triangulierungen, fehlt. Im Allgemeinen gibt es bei Gittern zwei verschiedene Arten von Zellnachbarschaften: Zwei Zellen mit nicht-leerem Schnitt können zum einen eine gemeinsame Kante besitzen, zum anderen nur einen gemeinsamen Punkt (siehe Abbildung 1.4). Bei den baryzentrischen Unterteilungen von

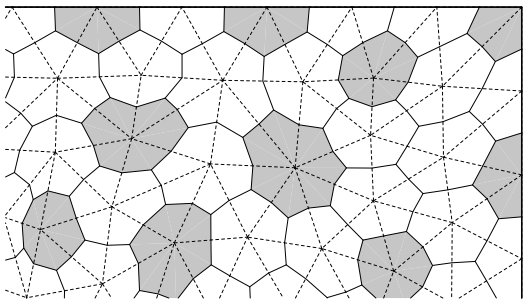

Abbildung 1.3: Triangulierung (gestrichelt) und baryzentrische Unterteilung. Einige Zellen wurden zur Verdeutlichung grau gefüllt.

Triangulierungen ist dies nicht so. Für je zwei Zellen C_i und C_j, $j \neq i$, gilt nämlich: Ist $C_i \cap C_j \neq \emptyset$, so besitzen C_i und C_j mindestens eine gemeinsame Kante.

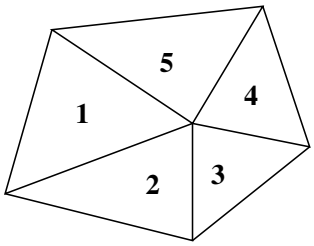

Abbildung 1.4: Zwei Arten von Nachbarschaften: Dreiecke 1 und 2 haben eine gemeinsame Kante, 1 und 3 haben nur einen gemeinsamen Punkt.

Wir führen für die Beschreibung der zwei Arten von Zellnachbarschaften die folgenden Begriffe ein:

Definition 1.2.5. Seien C_i, C_j mit $i \neq j$ zwei Zellen eines Gitters $\mathcal{G}$.

- Ist $C_i \cap C_j \neq \emptyset$, so sagen wir, *die Zellen berühren sich.*
- Gibt es ein Randsegment Γ_{ik} von Zelle C_i, das gleichzeitig Randsegment $\Gamma_{j\ell}$ von C_j ist, so nennen wir C_i und C_j *Kantennachbarn.* Den Index j bezeichnen wir dann auch als $\mathcal{N}(i,k)$ und umgekehrt i als $\mathcal{N}(j,\ell)$.

Mit diesen Begriffen können wir Bemerkung 1.2.4 wie folgt formulieren: Berühren sich zwei Zellen C_i und C_j der baryzentrischen Unterteilung einer Triangulierung, so sind C_i und C_j bereits *Kantennachbarn.*
Die Index-Zuordnung $\mathcal{N}(i,k)$ ist nicht unbedingt injektiv. Wie man in Abbildung 1.3 sehen kann, kann die Grenze zwischen zwei Zellen C_i und

C_j aus mehreren Liniensegmenten bestehen. Dann gibt es $k \neq \ell$ mit $\mathcal{N}(i,k) = \mathcal{N}(i,\ell) = j$.

Definition 1.2.6. Wir sagen, ein Gitter ist vom *Box-Typ*, wenn alle Zellen des Gitters, die sich berühren, bereits Kantennachbarn sind.

1.2.2 Zellmittelungsoperator

Fundamental für die Diskretisierung einer Erhaltungsgleichung durch ein Finite-Volumen-Verfahren ist der folgende Begriff (siehe [Son97a]):

Definition 1.2.7. Für ein Kontrollvolumen $C \subset \Omega$ und eine Funktion $u \in L^\infty(\Omega \times \mathbb{R}_0^+; S)$ bzw. $u \in L^\infty(\Omega; S)$ definieren wir den Operator $\mathcal{A}(C)(\cdot)$ durch

$$\begin{aligned} \mathcal{A}(C)u(t) &:= \frac{1}{|C|}\int_C u(x,t)dx \quad \text{bzw.} \\ \mathcal{A}(C)u &:= \frac{1}{|C|}\int_C u(x)dx. \end{aligned} \tag{1.10}$$

$|C|$ ist dabei der Flächeninhalt des Kontrollvolumens C. $\mathcal{A}(C)(\cdot)$ heißt *Zellmittelungsoperator*.

Durch Einsetzen von Zellen C_i eines Gitters als Kontrollvolumina und Division durch die Flächeninhalte $|C_i|$ kommt man von einer Erhaltungsgleichung in integraler Form (1.1) zur sogenannten semidiskreten Finite-Volumen-Formulierung:

$$\frac{d}{dt}\mathcal{A}(C_i)u(t) = -\frac{1}{|C_i|}\int_{\partial C_i} f(u(x,t))\cdot n ds, \quad i = 1,\dots,\#\mathcal{G}. \tag{1.11}$$

Setzt man zur Abkürzung

$$\mathcal{L}_{C_i}(u(\cdot,t)) := -\frac{1}{|C_i|}\int_{\partial C_i} f(u(x,t))\cdot n ds, \tag{1.12}$$

so erhält man für (1.11) die Kurzform

$$\frac{d}{dt}\mathcal{A}(C_i)u(t) = \mathcal{L}_{C_i}(u(\cdot,t)), \quad i = 1,\dots,\#\mathcal{G}. \tag{1.13}$$

Die weitere Diskretisierung der rechten Seite von (1.13) untergliedert sich in vier Aspekte. Zum Ersten muss die Randintegration, welche sich in $\mathcal{L}_{C_i}$

verbirgt, numerisch approximiert werden. Zum Zweiten liefert (1.13) nur ein Gesetz zur Veränderung der Zellmittelwerte $\mathcal{A}(C_i)u(t)$ mit der Zeit. Da der Zellmittelungsoperator nicht invertierbar ist (siehe [Son97b]), muss also eine Technik eingeführt werden um aus den Zellmittelwerten $\mathcal{A}(C_i)u(t)$ zu einer bestimmten Zeit t eine Approximation an $u(x,t)$ auf den C_i zu berechnen. Der dritte Aspekt ergibt sich aus den ersten beiden (siehe unten), der vierte Aspekt ist die Behandlung des Randes $\partial\Omega$.

1.2.3 Randintegration

Zur Umsetzung der Randintegration aus $\mathcal{L}_{C_i}$ nutzen wir aus, dass wir nur polygonale Zellen C_i verwenden. Nach Definition 1.2.1 kann der Rand ∂C_i jeder Zelle C_i als Vereinigung von endlich vielen Liniensegmenten dargestellt werden:

$$\partial C_i = \bigcup_{j=1}^{\#\Gamma_i} \Gamma_{ij}. \tag{1.14}$$

Die Integration der Flussfunktionen auf dem Rand einer Zelle C_i wird einzeln in den Zustandskomponenten durchgeführt und in die einzelnen Liniensegmente zerlegt:

$$\mathcal{L}_{C_i}(u(\cdot,t))_k = -\frac{1}{|C_i|}\sum_{j=1}^{\#\Gamma_i}\int_{\Gamma_{ij}} f_{\cdot,k}(u(x,t))\cdot n_j ds, \quad k=1,\dots,m. \tag{1.15}$$

Man beachte, dass $n_j = (n_{j,1}, n_{j,2})^T$ auf jedem Liniensegment konstant ist. Die Integrale in (1.15) sind bestimmte Integrale in einer Raumdimension. Diese werden mittels klassischer Quadraturverfahren numerisch approximiert. Wir verwenden das Gaußsche Integrationsverfahren (siehe [Sto89]) mit N_G Stützstellen.
Seien $G_1,\dots,G_{N_G}$ und $w_1,\dots,w_{N_G}$ die Gaußknoten und -Gewichte für ein Γ_{ij}. Damit können die Integrale aus (1.15) approximiert werden als

$$\int_{\Gamma_{ij}} f_{\cdot,k}(u(x,t))\cdot n_j ds = |\Gamma_{ij}|\left(\sum_{\ell=1}^{N_G} w_\ell f_{\cdot,k}(u(G_\ell,t))\cdot n_j + \mathcal{O}(h^{2N_G}(C_i))\right), \tag{1.16}$$

wenn $f_{\cdot,k}(u(x,t))\cdot n_j$ $2N_G$-mal stetig differenzierbar auf Γ_{ij} ist.
Für unsere Anwendungen haben wir die Verfahren für $N_G = 1$ und $N_G = 2$ verwendet. Zur Angabe der Formeln nehmen wir an, dass P_1 und P_2 die

Endpunkte des Liniensegmentes Γ_{ij} sind.
Für $N_G = 1$ ist der Gaußknoten durch den Mittelpunkt gegeben:

$$G_1(P_1, P_2) = \tfrac{1}{2}(P_1 + P_2)$$

und $w_1 = 1$. Für $N_G = 2$ gilt:

$$G_1(P_1, P_2) = \alpha P_1 + (1-\alpha)P_2, \quad G_2(P_1, P_2) = (1-\alpha)P_1 + \alpha P_2,$$

wobei $\alpha = \frac{\sqrt{3}+1}{2\sqrt{3}}$ und $w_1 = w_2 = \frac{1}{2}$.

1.2.4 Rekonstruktion

Der zweite genannte Aspekt der Diskretisierung ist die Notwendigkeit aus den Zellmittelwerten $\mathcal{A}(C_i)u(t)$ zu einer bestimmten Zeit t eine Approximation an $u(x,t)$ auf den C_i zu berechnen. Die erforderliche Prozedur ist ein *Rekonstruktionsalgorithmus* (siehe [Son97b]). Die vorhandenen Daten sind die Zellmittelwerte $\mathcal{A}(C_i)u(t)$ aller Zellen C_i zur Zeit t.

Definition 1.2.8. Sei $\mathcal{G}^\ell = \{C_1^\ell, \ldots, C_{\#\mathcal{G}^\ell}^\ell\}$, $\ell = 1, 2, \ldots$ eine Folge von Gittern für $\Omega \subset \mathbb{R}^2$. Ein Algorithmus $(\mathcal{R})$ heißt *Rekonstruktionsalgorithmus der (Fehler-)Ordnung $r > 0$ für die Folge $\mathcal{G}^\ell$, $\ell = 1, 2, \ldots$*, wenn gilt:
Für alle Funktionen $u \in C^r(\operatorname{conv}\Omega; \mathbb{R}^m)$ und alle ℓ berechnet $(\mathcal{R})$ aus den Daten $\{\mathcal{A}(C_k^\ell)u : k = 1, \ldots, \#\mathcal{G}^\ell\}$ Funktionen $\phi_u^{\ell,i} \in C^r(C_i^\ell; \mathbb{R}^m)$, $i = 1, \ldots, \#\mathcal{G}^\ell$ mit

$$|\phi_u^{\ell,i}(x) - u(x)| \leq K_1 \cdot K(u) \cdot h^r(C_i^\ell) \quad \forall x \in C_i^\ell.$$

Die Konstante K_1 hängt von der Folge der Gitter ab, ist aber für die gesamte Folge fest.

Den Gitterindex lassen wir ab jetzt immer weg, da wir uns immer ein festes Gitter $\mathcal{G} = \mathcal{G}^\ell$ auswählen.
Ein trivialer Rekonstruktionsalgorithmus der Ordnung $r = 1$ ist gegeben durch

$$\phi_u^i(x) \equiv \mathcal{A}(C_i)u. \tag{1.17}$$

Eine ausführliche theoretische und praktische Betrachtung von Rekonstruktionsalgorithmen findet man in [Son97b]. Wir werden uns in Abschnitt 2.2 ausführlich mit polynomialer Rekonstruktion, insbesondere mit *gewichteten wesentlich nicht oszillierenden Verfahren,* beschäftigen.
Ab jetzt gehen wir davon aus, dass ein (zunächst abstrakter) Rekonstruktionsalgorithmus $(\mathcal{R})$ der Ordnung $r > 0$ gegeben ist. Die damit berechneten Funktionen $\phi_{u(\cdot,t)}^i$ bzw. die Werte $\phi_{u(\cdot,t)}^i(G_\ell)$ wollen wir zur Approximation an die nicht bekannten Werte $u(G_\ell, t)$ in (1.16) einsetzen.

1.2.5 Numerische Flussfunktion

Nach Bemerkung 1.1.4 müssen wir davon ausgehen, dass die gesuchte Lösung Unstetigkeiten ausbildet. Deshalb haben wir von dem Rekonstruktionsalgorithmus auch nicht die Stetigkeit der ϕ_u^i über die Zellgrenzen hinaus gefordert. Dies führt uns zum dritten Aspekt der räumlichen Diskretisierung. Ist $\Gamma_{ij} \subset \partial\Omega$, so tritt die Randbehandlung in Kraft (siehe nächster Abschnitt). Ist hingegen $\Gamma_{ij} \not\subset \partial\Omega$, dann grenzt die Zelle C_i über das Liniensegment Γ_{ij} an eine andere Zelle, nämlich $C_{\mathcal{N}(i,j)}$. Damit ist $\phi_{u(\cdot,t)}^{\mathcal{N}(i,j)}(G_\ell)$ eine weitere Approximation an $u(G_\ell, t)$, die im Allgemeinen von $\phi_{u(\cdot,t)}^{i}(G_\ell)$ verschieden sein wird.
Um dieses Dilemma aufzulösen wird eine sogenannte numerische Flussfunktion $\tilde{f} : S \times S \times \mathbb{R}^2 \to \mathbb{R}^m$ eingeführt, welche zumindest Lipschitz-stetig ist, und für die für alle $u, v \in S$ und $n \in \mathbb{R}^2$ gilt:

$$\tilde{f}(u, u; n) = f(u) \cdot n_j, \tag{1.18}$$
$$\tilde{f}(v, u; -n) = -\tilde{f}(u, v; n). \tag{1.19}$$

(1.18) heißt *Konsistenzbedinung*, (1.19) heißt *Konservativitätsbedingung*. Dann verwendet man $\tilde{f}$ statt f in (1.16)

$$f_{\cdot,k}(u(G_\ell, t)) \cdot n_j \approx \tilde{f}_k(\phi_{u(\cdot,t)}^{i}(G_\ell), \phi_{u(\cdot,t)}^{\mathcal{N}(i,j)}(G_\ell); n_j). \tag{1.20}$$

Eine große Sammlung von numerischen Flussfunktionen findet man in [Hir90]. Für die skalaren Gleichungen verwenden wir daraus den numerischen Fluss von Engquist und Osher. Für die Euler-Gleichungen verwenden wir zwei verschiedene Realisierungen des *lokalen Lax-Friedrichs* Flusses sowie zu Vergleichszwecken den numerischen Fluss von Osher und Solomon (siehe [OS82]).

Der lokale Lax-Friedrichs Fluss

Der numerische Fluss von Lax und Friedrichs ist für eine skalare Gleichung gegeben als

$$\tilde{f}(u, v; n) = \frac{1}{2}\big((f(u) + f(v)) \cdot n - \Phi \cdot (v - u)\big). \tag{1.21}$$

Dabei ist $\Phi > 0$ eine globale Konstante. Der Term $\frac{1}{2}(f(u) + f(v))$ ist der zentrale Fluss. Er allein genommen führt zu einem instabilen Verfahren. Zur Stabilisierung dient der Term $\frac{\Phi}{2}(v - u)$, den man als Diffusionsterm ansehen kann. Um ein stabiles Verfahren zu erhalten müssen wir $\Phi \geq \max_u |df/du|$ wählen (siehe [Hir90]). Dieses Verfahren lässt sich dadurch verbessern, dass

die globale Konstante Φ durch eine lokale ersetzt wird. Dieses Vorgehen wird in [LOC94] für den eindimensionalen Fall beschrieben. Dort wird in Gleichung (1.21)

$$\Phi = \Phi(u,v) = \max_{\min(u,v)\le w\le\max(u,v)} \big|\frac{d}{dw} f(w)\big| \tag{1.22}$$

gesetzt. Für den Fall des Systems der Euler-Gleichungen in zwei Raumdimensionen haben wir ausgehend von [SO88] und [HS99] die folgenden Realisierungen gewählt:

Definition 1.2.9. Die erste Realisierung verwendet einen skalarwertigen Diffusionsfaktor:

$$\Phi_{slLF}(u,v;n) := \max_{w\in\{u,v,\frac{u+v}{2}\}} |\mathsf{v}(w)\cdot n| + \mathsf{a}(w). \tag{1.23}$$

Die zweite Realisierung verwendet einen matrixwertigen Diffusionsfaktor. Dazu werden die Diagonalisierungsmatrizen aus Bemerkung 1.1.3, S. 15, verwendet. Wir definieren

$$|\Lambda|(w;n) := \mathrm{diag}(|\mathsf{v}(w)\cdot n|, |\mathsf{v}(w)\cdot n|, |\mathsf{v}(w)\cdot n + \mathsf{a}(w)|, |\mathsf{v}(w)\cdot n - \mathsf{a}(w)|)$$

und setzen

$$\Phi_{mlLF}(u,v;n) := P(\tfrac{u+v}{2};n)\big(\max_{w\in\{u,v,\frac{u+v}{2}\}} |\Lambda|(w;n)\big)P^{-1}(\tfrac{u+v}{2};n), \tag{1.24}$$

wobei das Maximum komponentenweise genommen wird.
Die resultierenden numerischen Flussfunktionen bezeichnen wir mit $\tilde{f}_{slLF}$ beziehungsweise $\tilde{f}_{mlLF}$.

1.2.6 Randbehandlung

Der letzte Aspekt der räumlichen Diskretisierung ist die Behandlung des Randes von Ω. Es gibt zwei grundsätzlich unterschiedliche Arten von Rändern, was wir am Beispiel eines Gebietes Ω um ein Flügelprofil verdeutlichen wollen (siehe Abbildung 1.5).

Physikalische Ränder Der innere Rand von Ω ist durch das Flügelprofil gegeben. Hier gelten physikalische Bedingungen: Das Fluid strömt nicht durch das Flügelprofil.

Numerische Ränder Der äußere Rand ist nicht physikalisch motiviert. Dieser Rand wird eingefügt, damit das Gebiet Ω beschränkt ist.

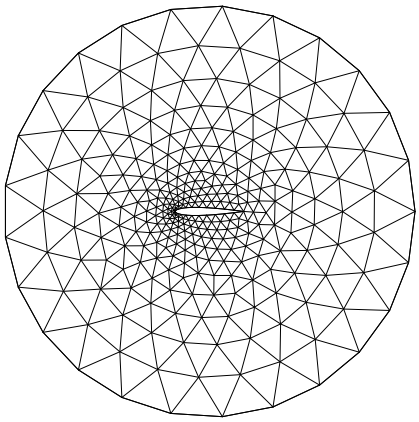

Abbildung 1.5: Triangulierung eines Gebietes Ω um ein Flügelprofil.

Physikalische Ränder kommen bei uns nur für die Euler-Gleichungen vor. Hier lautet die physikalische Bedingung an einer *festen Wand* $\mathsf{v} \cdot n = 0$, wobei n der äußere Einheitsnormalenvektor auf $\partial\Omega$ ist. Für den Flusstensor f der Euler-Gleichungen gilt

$$f \cdot n = \begin{pmatrix} \rho \mathsf{v} \cdot n \\ \rho \mathsf{v}_1 \mathsf{v} \cdot n + \mathsf{p} n_1 \\ \rho \mathsf{v}_2 \mathsf{v} \cdot n + \mathsf{p} n_2 \\ (\rho \mathsf{E} + \mathsf{p}) \mathsf{v} \cdot n \end{pmatrix}. \tag{1.25}$$

Aus der Bedingung $\mathsf{v} \cdot n = 0$ folgt

$$f_{\text{feste Wand}} \cdot n = \begin{pmatrix} 0 \\ \mathsf{p} n_1 \\ \mathsf{p} n_2 \\ 0 \end{pmatrix}. \tag{1.26}$$

Bei der numerischen Umsetzung wird p dabei aus $\phi^i_{u(\cdot,t)}(G_\ell)$ berechnet.
Für die numerischen Ränder ist die Situation vielfältig. Hier sind Kenntnisse über die physikalischen Gegebenheiten hilfreich:

- Ist der exakte Wert von $u_{\text{außen}}$ bekannt, so kann wie im Inneren von Ω die numerische Flussfunktion verwendet werden:

  $$f_{\text{Rand}}(u(G_\ell, t)) \cdot n_j := \tilde{f}(\phi^i_{u(\cdot,t)}(G_\ell), u_{\text{außen}}; n_j).$$

  Dieser Fall ist bei uns bei allen Testbeispielen für die skalaren Gleichungen gegeben. Bei Flügelprofilumströmungen ist diese Randbehandlung auch sinnvoll, wenn sichergestellt wird, dass der äußere Rand weit genug vom Flügelprofil entfernt ist.

- Ist bekannt, dass am Rand eine Einströmung mit $\mathsf{Ma} > 1$ stattfindet, so ergibt sich der Fluss am Rand allein aus dem Außenzustand:

$$f_{\text{Rand}}(u(G_\ell, t)) \cdot n_j := f(u_{\text{außen}}) \cdot n_j.$$

- Liegt dagegen eine Ausströmung mit $\mathsf{Ma} > 1$ vor, so ergibt sich der Fluss am Rand allein aus dem Innenzustand:

$$f_{\text{Rand}}(u(G_\ell, t)) \cdot n_j := f(\phi^i_{u(\cdot,t)}(G_\ell)) \cdot n_j.$$

Wir kommen für unsere Testfälle mit den oben genannten Randbehandlungen aus. Es gibt allerdings Fälle, bei denen über den physikalischen Zustand am Rand zu wenig bekannt ist um eine dieser Randbehandlungen einsetzen zu können. In diesen Fällen werden sogenannte *charakteristische Randbehandlungen* verwendet (siehe [Hir90]).

1.2.7 Finite-Volumen-Approximation

Fasst man die vorgenommenen Schritte zusammen, so erhält man für $\mathcal{L}_{C_i}(u(\cdot,t))$ die Approximation[3]

$$\tilde{\mathcal{L}}_{C_i}(\{\mathcal{A}(C_k)(u(\cdot,t))\}) := -\frac{1}{|C_i|}\sum_{j=1}^{\#\Gamma_i} |\Gamma_{ij}| \sum_{\ell=1}^{p} w_\ell \tilde{f}(\phi^i_{u(\cdot,t)}(G_\ell), \phi^{\mathcal{N}(i,j)}_{u(\cdot,t)}(G_\ell); n_j). \quad (1.27)$$

Definition 1.2.10. Die Anfangswertaufgabe

$$\begin{aligned} \frac{d}{dt}\bar{u}_i(t) &= \tilde{\mathcal{L}}_{C_i}(\{\bar{u}_k(t)\}), \qquad i = 1, \ldots, \#\mathcal{G}, \\ \bar{u}_i(0) &= \mathcal{A}(C_i)(u(\cdot,0)), \qquad i = 1, \ldots, \#\mathcal{G} \end{aligned}$$

heißt *Finite-Volumen-Approximation*. Wird der triviale Rekonstruktionsalgorithmus (1.17) verwendet, so spricht man von der *Basisdiskretisierung*.

[3] In den Operator $\mathcal{L}_{C_i}$ geht für jedes i die Menge aller Zellmittelwerte $\{\mathcal{A}(C_k)(u(\cdot,t)) : k = 1, \ldots, \#\mathcal{G}\}$ ein (in der Praxis werden meist nur die $\mathcal{A}(C_k)(u(\cdot,t))$ aus einer nahen Umgebung von C_i benötigt). Wir schreiben diese Menge immer abkürzend als $\{\mathcal{A}(C_k)(u(\cdot,t))\}$.

1.3 Zeitliche Diskretisierung

Die Finite-Volumen-Approximation aus Definition 1.2.10 ist ein endliches System gewöhnlicher Differentialgleichungen. Zur Diskretisierung verwenden wir für die Basisdiskretisierung das Polygonzugverfahren von Euler (siehe [SB90]), ansonsten die expliziten Runge-Kutta-Verfahren von Shu und Osher (siehe [SO88]) mit zwei[4] bzw. drei Stufen. Alle drei Verfahren bilden als Einschrittverfahren Algorithmen zur Berechnung der $\bar{u}_i(t+\Delta t)$ aus den $\bar{u}_i(t)$. Zur besseren Übersichtlichkeit definieren wir $\bar{u}_i^{(0)} := \bar{u}_i(t)$. Das Polygonzugverfahren von Euler ist gegeben durch

$$\bar{u}_i(t+\Delta t) := \bar{u}_i^{(0)} + \Delta t \tilde{\mathcal{L}}_i(\{\bar{u}_k^{(0)}\}). \tag{1.28}$$

Das zweistufige Verfahren von Shu und Osher ist gegeben durch

$$\begin{aligned} \bar{u}_i^{(1)} &:= \bar{u}_i^{(0)} + \Delta t \tilde{\mathcal{L}}_i(\{\bar{u}_k^{(0)}\}), \\ \bar{u}_i(t+\Delta t) &:= \tfrac{1}{2}\bar{u}_i^{(0)} + \tfrac{1}{2}\bar{u}_i^{(1)} + \tfrac{1}{2}\Delta t \tilde{\mathcal{L}}_i(\{\bar{u}_k^{(1)}\}). \end{aligned} \tag{1.29}$$

Das dreistufige Verfahren von Shu und Osher lautet:

$$\begin{aligned} \bar{u}_i^{(1)} &:= \bar{u}_i^{(0)} + \Delta t \tilde{\mathcal{L}}_i(\{\bar{u}_k^{(0)}\}), \\ \bar{u}_i^{(2)} &:= \tfrac{3}{4}\bar{u}_i^{(0)} + \tfrac{1}{4}\bar{u}_i^{(1)} + \tfrac{1}{4}\Delta t \tilde{\mathcal{L}}_i(\{\bar{u}_k^{(1)}\}), \\ \bar{u}_i(t+\Delta t) &:= \tfrac{1}{3}\bar{u}_i^{(0)} + \tfrac{2}{3}\bar{u}_i^{(2)} + \tfrac{2}{3}\Delta t \tilde{\mathcal{L}}_i(\{\bar{u}_k^{(2)}\}). \end{aligned} \tag{1.30}$$

Bei beiden Verfahren müssen die Berechnungen in jeder Stufe für alle Zellen $i = 1, \ldots, \#\mathcal{G}$ durchgeführt werden, bevor zur nächsten Stufe übergegangen werden kann, da das System über die $\tilde{\mathcal{L}}_i$ gekoppelt ist. Für die algorithmische Umsetzung ist es sehr nützlich, dass bei der dritten Stufe des dreistufigen Verfahrens die $\bar{u}_i^{(1)}$ nicht mehr eingehen. Deshalb ist zur Implementierung des dreistufigen Verfahrens nicht mehr Speicherplatz erforderlich als zur Implementierung des zweistufigen Verfahrens.
Alle drei Verfahren sind *explizit*, im Gegensatz zu impliziten Verfahren, bei denen zur Berechnung von $\bar{u}_i(t+\Delta t)$ formal bereits $\tilde{\mathcal{L}}_i(\{\bar{u}_k(t+\Delta t)\})$ eingeht. Die Umsetzung impliziter Verfahren erfordert einen hohen Aufwand bei der Implementierung. Insbesondere muss in jedem Zeitschritt ein großes, dünnbesetztes Gleichungssystem gelöst werden. Der Vorteil der impliziten Verfahren liegt darin, dass es im Gegensatz zu den expliziten Verfahren keine Beschränkung der Zeitschrittgröße aus Stabilitätsgründen gibt.

[4] Das zweistufige Verfahren von Shu und Osher entspricht dem Verfahren von Heun (siehe [SB90]).

In dieser Arbeit beschränken wir uns auf die angegebenen expliziten Zeitschrittverfahren. Zur Umsetzung von impliziten Zeitschrittverfahren zur Lösung hyperbolischer Erhaltungsgleichungen sei auf die Ausführungen in [Mei96] verwiesen.

Definition 1.3.1. Die Finite-Volumen-Approximation zusammen mit einer zeitlichen Diskretisierung ist ein Verfahren zur näherungsweisen Berechnung von $\mathcal{A}(C_i)u(\cdot,t)$ für eine hyperbolische Erhaltungsgleichung (1.1) auf einem Gitter $\mathcal{G}$. Es wird *Finite-Volumen-Verfahren* genannt.

1.3.1 Die CFL-Bedingung

Der Nachteil der expliziten Zeitschrittverfahren liegt, wie erwähnt, in der Beschränkung der Zeitschrittgröße (siehe [CFL28]). Sie lautet

$$\Delta t \leq \frac{\Delta x}{\max\limits_{\substack{k=1,\dots,m \\ |\nu|=1}} |\lambda_k(u,\nu)|}. \tag{1.31}$$

Dabei sind die $\lambda_k(u,\nu)$ die Eigenwerte der Matrix $\nabla_u f(u)\cdot\nu$ (siehe Definition 1.1.2). Δx ist die *Größe* einer Zelle. Bedingung (1.31) nennt man die *CFL-Bedingung*. Es handelt sich um eine notwendige Bedingung. Mitunter muss der Zeitschritt sogar kleiner gewählt werden.

1.3.2 Die Längendimension bei der CFL-Bedingung

Es ist nicht unmittelbar klar, welcher Wert für Δx in zwei Raumdimensionen zu nehmen ist. Wir verwenden den Flächeninhalt einer Zelle geteilt durch den Durchmesser, $\Delta x := \frac{|C_i|}{h(C_i)}$. Diese Wahl leiten wir anhand eines Modellproblems her:
Wir betrachten die lineare Advektionsgleichung, wobei ohne Einschränkung der Allgemeinheit der Transport parallel zur x-Achse verlaufe, das heißt $f_1(u) = a_1 \cdot u$, $f_2(u) = 0$, $a_1 > 0$ (siehe auch S. 14).
Wir betrachten eine konvexe Zelle C. Innerhalb der Zelle sei u konstant gleich u_i, außerhalb sei u konstant gleich $u_a \neq u_i$ (siehe Abbildung 1.6). Als numerische Flussfunktion wird der *Upwind-Fluss* genommen:

$$\tilde{f}(u_i,u_a,n) := \begin{cases} a_1 n_1 u_i & \text{falls } n_1 > 0, \\ a_1 n_1 u_a & \text{sonst,} \end{cases}$$

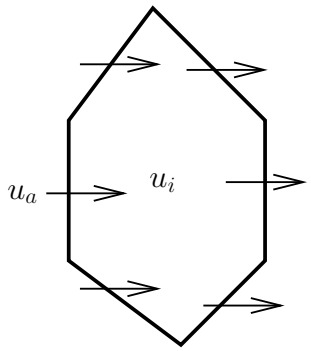

Abbildung 1.6: Modellzelle mit innerem und äußerem Zustand, beide als konstant angenommen.

wobei n die äußere Einheitsnormale an die Zelle C ist. Aufgrund der Konvexität von C können wir

$$\tilde{\mathcal{L}}_C := -\frac{1}{|C|}\int_{\partial C} \tilde{f}(u_i, u_a, n)ds \tag{1.32}$$

geschlossen angeben:

$$\tilde{\mathcal{L}}_C = \frac{a_1}{|C|}(u_a - u_i)\big(\max_{x\in C} x_2 - \min_{x\in C} x_2\big).$$

Bei den vorhandenen Daten muss für eine stabile Lösung zumindest gefordert werden, dass

$$u_i + \Delta t\tilde{\mathcal{L}}_C \in [\min\{u_i, u_a\}, \max\{u_i, u_a\}]\,.$$

Dies ist wegen $\Delta t \geq 0$ äquivalent zu

$$\Delta t|\tilde{\mathcal{L}}_C| \leq |u_a - u_i|. \tag{1.33}$$

Bei Drehung der Zelle C um einen Winkel φ gilt

$$\begin{aligned} \max_{\varphi} |\tilde{\mathcal{L}}_C| &= \frac{a_1}{|C|}|u_a - u_i|h(C), \quad \text{da} \\ \max_{\varphi}\big(\max_{x\in C} x_2 - \min_{x\in C} x_2\big) &= h(C). \end{aligned}$$

Setzen wir dies in (1.33) ein, so erhalten wir die Bedingung

$$\Delta t \leq \frac{|C|}{h(C)a}.$$

Dies entspricht der CFL-Bedingung (Gleichung (1.31)) mit $\Delta x = \frac{|C|}{h(C)}$.

1.4 Algorithmische Umsetzung

1.4.1 Datenstrukturen

Wir verwenden für alle Daten, die abstrakt als Mengen aufgefasst werden können, Arrays. Folgende Daten eines Gittters $\mathcal{G}$ werden für ein Basisverfahren benötigt[5]:

Integer	*#Zellen*
Integer	*#innereKanten*
Integer	*#Randkanten*
Real	*Flächeninhalt*(*#Zellen*)
Real	*Durchmesser*(*#Zellen*)
Integer	*innereKanteZelle*(*#innereKanten*)(2)
Coord	*innereKantePunkt*(*#innereKanten*)(2)
Integer	*RandkanteZelle*(*#Randkanten*)
Coord	*RandkantePunkt*(*#Randkanten*)(2)

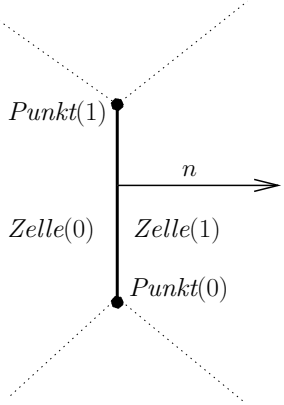

Abbildung 1.7: Orientierung von Punkten und Zellen an einer Kante.

Die Kantendaten ...*KanteZelle* und ...*KantePunkt* sind so orientiert, wie in Abbildung 1.7 zu sehen: Setzt man $P = Punkt(0)$ und $Q = Punkt(1)$, so ist

[5] Koordinaten von Kantenpunkten (*innereKantePunkt, RandkantePunkt*) kommen immer mindestens zweimal vor. Daher werden in der Implementierung (um Speicherplatz zu sparen) die Arrays *innereKantePunkt* und *RandkantePunkt* indirekt realisiert: Die Koordinaten aller Randpunkte werden in einem Array Coord *Kantenpunkt*(*#Kantenpunkte*) abgespeichert. Weiterhin werden Arrays von Indizes gespeichert, nämlich Integer *innereKantePunktIdx*(*#innereKanten*)(2) und Integer *RandkantePunktIdx*(*#Randkanten*)(2). Dann erhält man z.B. die Koordinaten *innereKantePunkt*(i)(j) durch *KantenPunkt*(*innereKantePunktIdx*(i)(j)).

der Vektor $n = (Q(1) - P(1), P(0) - Q(0))^T / |Q - P|$ der (normierte) äußere Einheitsnormalenvektor von *Zelle*(0) auf der gegebenen Kante.
Bei dem Spezialfall der baryzentrischen Unterteilung einer Triangulierung ist es nicht erforderlich die inneren Kanten explizit abzuspeichern. Stattdessen ist insbesondere für die Umsetzung der WENO-Rekonstruktion (siehe Abschnitt 2.2, S. 49) die zugrundeliegende Dreiecks-Topologie hilfreich. In diesem Fall speichern wir die folgenden Daten der Triangulierung $\mathcal{T}$:

Integer	*#Punkte* (=*#Zellen*)
Integer	*#Dreiecke*
Integer	*#Randkanten*
Integer	*DreieckspunktIdx*(*#Dreiecke*)(3)
Coord	*Punkt*(*#Punkte*)
Real	*Flächeninhalt*(*#Zellen*)
Real	*Durchmesser*(*#Zellen*)
Integer	*RandkanteZelle*(*#Randkanten*)
Coord	*RandkantePunkt*(*#Randkanten*)(2)

1.4.2 Algorithmen

Prozedur 1.1: HAUPTPROGRAMM()

Eingabe: $\mathcal{G}$: Gitter,
u: Array von Zellmittelwerten der Erhaltungsgrößen zur Zeit $t = 0$,
T: Zeit, zu der eine Näherung von u berechnet werden soll.

Ausgabe: u: Array von Zellmittelwerten der Erhaltungsgrößen zur Zeit T.

Start

- EINGABE($\mathcal{G}, u, T$)
- $t := 0$
- SOLANGE $t < T$:
  - $\Delta t := \theta \cdot$MAXIMALEZEITSCHRITTWEITE($\mathcal{G}, u$)
  - ZEITINTEGRATION($\mathcal{G}, u, \Delta t$)
  - $t := t + \Delta t$
- AUSGABE(u)

Ende.

Mit dem Parameter $\theta \in (0, 1]$ kann die Zeitschrittweite, wenn nötig, unter den durch die CFL-Bedingung vorgegebenen Wert reduziert werden.

Prozedur 1.2: MaximaleZeitschrittweite$(\mathcal{G}, u)$

Eingabe: $\mathcal{G}$: Gitter, u: Array von Zellmittelwerten der Erhaltungsgrößen.

Ausgabe: Δt: Maximal erlaubte Zeitschrittgröße.

Start

- $\Delta t := +\infty$
- Für i von 0 bis (#Zellen-1)
  - $\Delta x :=$*Flächeninhalt*$(i)/$*Durchmesser*(i)
  - Falls $\Delta x/|\lambda|_{\max}(u(i)) < \Delta t$
    - $\Delta t := \Delta x/|\lambda|_{\max}(u(i))$
- Rückgabe Δt

Ende.

Prozedur 1.3: Zeitintegration$(\mathcal{G}, u, \Delta t)$

Eingabe: $\mathcal{G}$: Gitter,
u: Array von Zellmittelwerten der Erhaltungsgrößen zur Zeit t,
Δt: Zeitschrittweite.

Ausgabe: u: Array von Mittelwerten der Erhaltungsgrößen zur Zeit $t + \Delta t$.

Start

- Für i von 0 bis *#Zellen*-1)
  - $u_0(i) := u(i)$
- Für k von 0 bis (*#Stufen*-1)
  - MittlereFlussdivergenz$(\mathcal{G}, u, \overline{\nabla \cdot f})$
  - Für i von 0 bis (*#Zellen*-1)
    - $u(i) := u_0(i) + so(k)\left(u(i) - u_0(i) - \Delta t \overline{\nabla \cdot f}(i)\right)$

Ende.

Das Koeffizienten-Array *so* ist für das Polygonzugverfahren von Euler und für die Runge-Kutta-Verfahren von Shu und Osher wie folgt definiert:

#Stufen	*so*
1	1
2	$1, \frac{1}{2}$
3	$1, \frac{1}{4}, \frac{2}{3}$

Prozedur 1.4: MittlereFlussdivergenz($\mathcal{G}, u, \overline{\nabla \cdot f}$)

Eingabe: $\mathcal{G}$: Gitter,
u: Array von Zellmittelwerten der Erhaltungsgrößen.

Ausgabe: $\overline{\nabla \cdot f}$: Array der $-\tilde{\mathcal{L}}_i(\{u_j\})$.

Start

- Rekonstruktion($\mathcal{G}, u, \phi$)
- Für i von 0 bis ($\#Zellen-1$)
  - $\overline{\nabla \cdot f}(i) := 0$
- Für i von 0 bis ($\#innereKanten-1$)
  - $Z_0 := innereKanteZelle(i)(0), \quad Z_1 := innereKanteZelle(i)(1)$
  - $P_0 := innereKantePunkt(i)(0), \quad P_1 := innereKantePunkt(i)(1)$
  - $n := \frac{1}{|P_1 - P_0|} \begin{pmatrix} P_1(1) - P_0(1) \\ P_0(0) - P_1(0) \end{pmatrix}$
  - Für j von 1 bis $\#Gaußknoten$
    - $\tilde{f}$:=NumerischerFluss($\phi(Z_0)(G_j(P_0, P_1)), \phi(Z_1)(G_j(P_0, P_1)), n$)
    - $\overline{\nabla \cdot f}(Z_0) := \overline{\nabla \cdot f}(Z_0) + w_j \cdot |P_1 - P_0| \cdot \tilde{f}$
    - $\overline{\nabla \cdot f}(Z_1) := \overline{\nabla \cdot f}(Z_1) - w_j \cdot |P_1 - P_0| \cdot \tilde{f}$
- Für i von 0 bis ($\#Randkanten-1$)
  - $Z_0 := RandkanteZelle(i)$
  - $P_0 := RandkantePunkt(i)(0), \quad P_1 := RandkantePunkt(i)(1)$
  - $n := \frac{1}{|P_1 - P_0|} \begin{pmatrix} P_1(1) - P_0(1) \\ P_0(0) - P_1(0) \end{pmatrix}$

- Für j von 1 bis *#Gaußknoten*
  - $\tilde{f}$:=RandFluss($\phi(Z_0)(G_j(P_0, P_1)), n$)
  - $\overline{\nabla \cdot f}(Z_0) := \overline{\nabla \cdot f}(Z_0) + w_j \cdot |P_1 - P_0| \cdot \tilde{f}$

- Für i von 0 bis (*#Zellen*-1)
  - $\overline{\nabla \cdot f}(i) := \overline{\nabla \cdot f}(i)/$*Flächeninhalt(i)*

Ende.

Verwenden wir als Gitter die baryzentrische Unterteilung einer Triangulierung, so können wir, wie bei den Datenstrukturen beschrieben, die Triangulierung speichern. Dann stehen uns die inneren Kanten nicht unmittelbar zur Verfügung. Sie werden stattdessen immer wieder neu berechnet. Dazu wird in der obigen Prozedur die Schleife über die inneren Kanten durch eine Schleife über die Dreiecke ersetzt:

- Für i von 0 bis (*#Dreiecke*-1)
  - $PIdx := DreieckspunktIdx(i)$
  - $B := \frac{1}{3}\big(Punkt(PIdx(0)) + Punkt(PIdx(1)) + Punkt(PIdx(2))\big)$
  - Für k_1 von 0 bis 2
    - $k_2 := (k_1 + 1) \mod 3$
    - $Z_0 := PIdx(k_1), \quad Z_1 := PIdx(k_2)$
    - $P_0 := \frac{1}{2}(Punkt(Z_0) + Punkt(Z_1)), \quad P_1 := B$
    - $n = \dfrac{1}{|P_1 - P_0|} \begin{pmatrix} P_1(1) - P_0(1) \\ P_0(0) - P_1(0) \end{pmatrix}$
    - Für j von 1 bis *#Gaußknoten*

      ...

Damit n jeweils der äußere Einheitsnormalenvektor an Zelle Z_0 ist, müssen die Punkte der Dreiecke positiv orientiert sein (siehe Abbildung 1.8).
Die Prozedur Rekonstruktion ist für die Basisdiskretisierung trivial. Ein nicht-trivialer Rekonstruktionsalgorithmus wird in Abschnitt 2.2 beschrieben.
Die Prozeduren NumerischerFluss() und RandFluss() ergeben sich unmittelbar aus den Ausführungen in den Abschnitten 1.2.5 und 1.2.6.

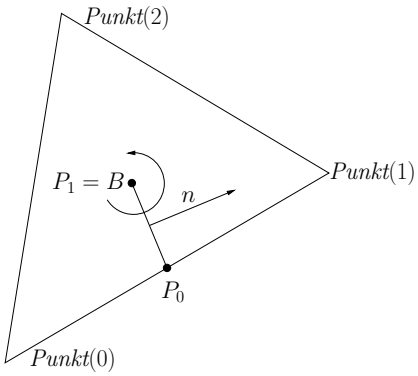

Abbildung 1.8: Konstruierte Kante und Normale für $k_1 = 0$.

1.4.3 Aufwand der Algorithmen

Wir wollen uns ansehen, welchen numerischen Aufwand die einzelnen Prozeduren verursachen. Alle Berechnungen, deren Aufwand weniger als linear mit $\#\mathcal{G}$ wächst, werden wir dabei als unwesentlich vernachlässigen. Im Folgenden bezeichnen wir mit Aufwand(Prozedur()) den wesentlichen Aufwand für die einmalige Ausführung von Prozedur().

Hauptprogramm(): Wesentlicher Aufwand entsteht nur durch die Unterprozeduren. Beachtet man, dass die Zahl der erforderlichen Zeitschritte umgekehrt proportional zum Durchmesser der kleinsten Zelle ist, so erhält man einen Gesamtaufwand, der proportional zu

$$\frac{1}{\min_i h(C_i)}\big(\ \text{Aufwand}(\textsc{MaximaleZeitschrittweite}()) + \text{Aufwand}(\textsc{Zeitintegration}())\big)$$

ist.

MaximaleZeitschrittweite():

$$K_{\Delta t} \cdot \#\mathcal{G}.$$

Zeitintegration():

$$K_{ZI} \cdot \#\mathcal{G} + \text{Aufwand}(\textsc{MittlereFlussdivergenz}()).$$

MittlereFlussdivergenz(): Der erste Schritt dieser Prozedur ist die Rekonstruktion. Für alle von uns betrachteten Rekonstruktionsalgorithmen $\mathcal{R}$ ist Aufwand(Rekonstruktion()) $= K_{\mathcal{R}} \cdot \#\mathcal{G}$.

Die Anzahl der inneren Kanten ist proportional zu $\#\mathcal{G}$, die Anzahl der Randkanten ist üblicherweise nicht wesentlich: Im Fall von gleichmäßig feinen Gittern ist die Anzahl der Randkanten nur proportional zu $1/h$ während $\#\mathcal{G}$ und die Zahl der inneren Kanten proportional zu $1/h^2$ ist. Wir erhalten also für die Berechnung der numerischen Flüsse den Aufwand $K_{\tilde{f}} \cdot \#\mathcal{G}$ und damit für diese Prozedur den Aufwand

$$K_{\tilde{\mathcal{L}}} \cdot \#\mathcal{G}, \quad \text{wobei } K_{\tilde{\mathcal{L}}} := K_{\mathcal{R}} + K_{\tilde{f}}.$$

Sieht man sich die Art der Berechnungen in den einzelnen Prozeduren an, so erkennt man, dass $K_{\Delta t} \ll K_{\tilde{\mathcal{L}}}$ und $K_{ZI} \ll K_{\tilde{\mathcal{L}}}$. Der Gesamtaufwand pro Zeitschritt wird also durch $K_{\tilde{\mathcal{L}}} \cdot \#\mathcal{G}$, den Aufwand von MittlereFlussdivergenz(), dominiert. Dabei ist bei der Verwendung der WENO-Rekonstruktion (siehe Abschnitt 2.2) mit Polynomgrad $Q = 2$ wiederum $K_{\mathcal{R}}$ deutlich größer als $K_{\tilde{f}}$.
Bei einem gleichmäßig feinen Gitter ist $\#\mathcal{G}$ proportional zu $1/h^2$, wobei $h \approx h(C_i)$ für alle Zellen C_i ist. Damit ergibt sich in dem Fall der Berechnung einer Näherungslösung zur festen Zeit $T > 0$ ein Aufwand, der proportional zu $1/h^3$ ist.

1.4.4 Parallelisierung, Vektorisierung

Neben den Anstrengungen die Algorithmen an sich zu verändern um eine Verringerung des Aufwands und damit eine Beschleunigung der Berechnungen zu erreichen, lohnt es sich auch auf der informatischen Seite für eine Beschleunigung zu sorgen. Dabei ist die Anpassung der Algorithmen an Parallel- und Vektorrechner hervorzuheben. Es ist mit moderatem Programmieraufwand möglich die beschriebenen Algorithmen auf Parallelrechner mit gemeinsamem Speicher der Prozessoren sowie auf Vektorrechner zu übertragen.
Auf Parallelrechnern mit gemeinsamem Speicher der Prozessoren steht üblicherweise das *Thread*-Konzept zur Verfügung. Damit ist es möglich verschiedene Prozeduren oder auch dieselbe Prozedur mit verschiedenen Daten parallel auszuführen. Der Aufwand bei den obigen Prozeduren kommt immer durch das Abarbeiten von Schleifen zustande. Realisiert man jede der Schleifen als eigene Prozedur und wählt einen Anfangs- und Endindex als Parameter, so kann man die Schleifen jeweils in mehrere Teilschleifen zerlegen und diese parallel abarbeiten.
Auf Vektorrechnern ist keine Unterteilung der Schleifen nötig. Paralleles Abarbeiten von Schleifen über Arrays ist gerade das Konzept dieser Rechner.
Für beide Arten von Rechnern muss allerdings sichergestellt werden, dass bei der parallelen Bearbeitung der Schleifen nicht mehrere Prozessoren konkurrierend dieselbe Speicheradresse beschreiben. Diese Gefahr besteht in der

Schleife über die inneren Kanten sowie bei der Schleife über die Randkanten in der Prozedur MittlereFlussdivergenz(), da verschiedene Kanten einen gemeinsamen Anfangs- oder Endpunkt haben können.

Einfärben

Eine Technik dieses Problem zu umgehen bezeichnet man als *Einfärben*. Wir erläutern das Konzept am Beispiel der inneren Kanten: Die Menge der inneren Kanten wird disjunkt in Teilmengen aufgeteilt, so dass innerhalb einer Teilmenge kein Punkt mehrfach adressiert wird. Ist zum Beispiel die Kante mit den Punkten P_0, P_1 in einer Teilmenge, so dürfen all die anderen Kanten, die einen der Punkte P_0, P_1 als Anfangs- oder Endpunkt haben, nicht in derselben Teilmenge enthalten sein. Der Begriff des Einfärbens kommt dadurch zustande, dass man symbolisch jeder Teilmenge eine andere Farbe zuweist. Die Zahl der erforderlichen Teilmengen ist durch die Maximalzahl von Kanten gegeben, die denselben Punkt enthalten.
Die Teilmengen werden dann sequentiell nacheinander abgearbeitet, wobei innerhalb der Teilmengen ohne das Risiko des konkurrierenden Schreibzugriffs parallelisiert oder vektorisiert werden kann.

Kapitel 2

Polynomiale Rekonstruktion

In diesem Kapitel beschäftigen wir uns mit der Rekonstuktion von Funktionen aus Zellmittelwerten mit Hilfe von Polynomen. Dabei gehen wir zunächst von skalarwertigen Funktionen $u : \mathbb{R}^2 \to \mathbb{R}$ aus. Vektorwertige Funktionen werden komponentenweise rekonstruiert, wobei wir auch den Einfluss von Variablentransformationen diskutieren.

2.1 Grundlagen

Im Folgenden wird es wichtig sein, dass die Zellen von Gittern nicht entartet sind und dass die Zellgrößen beschränkt variieren.

Definition 2.1.1. Wir sagen, eine Menge von Gittern ist *regulär*, wenn Konstanten K_1 und K_2 existieren, so dass gilt:

1. Für alle Zellen C dieser Gitter ist $h(C) \leq K_1 \cdot \sqrt{|C|}$.
2. Für jede Zelle C dieser Gitter gilt: Für jede Zelle C', die C berührt, ist $|C'| \leq K_2 \cdot |C|$.

Wir werden ab jetzt annehmen, dass das verwendete Gitter $\mathcal{G}$ aus einer solchen regulären Menge ist.
Wie bereits im vorigen Kapitel erläutert, verzichten wir auf Glätteforderungen für die Rekonstruktionsfunktionen über die Zellgrenzen hinaus. Innerhalb der Zellen verwenden wir Polynome vom Grad kleiner oder gleich Q, die wir in lokalen Koordinaten darstellen. Das heißt, für jede Zelle C_i eines Gitters $\mathcal{G}$ berechnen wir ein

$$\phi_u^i(x) = p_i(x) \in \Pi^Q(\mathbb{R}^2, \mathbb{R}) := \Big\{ \sum_{|\alpha| \leq Q} a_\alpha (x - \overline{x}_{C_i})^\alpha : a_\alpha \in \mathbb{R} \Big\}. \tag{2.1}$$

Dabei sei $\overline{x}_{C_i}$ der Schwerpunkt der Zelle C_i:

Definition 2.1.2. Für jede Zelle C eines Gitters $\mathcal{G}$ definieren wir den *Schwerpunkt* $\overline{x}_C \in \mathbb{R}^2$ durch

$$\overline{x}_C := \mathcal{A}(C)(x \mapsto x) = \frac{1}{|C|}\Big(\int\limits_C x_1 dx, \int\limits_C x_2 dx\Big)^T.$$

Für die lokalen Koordinaten $x - \overline{x}_C$ einer Zelle C definieren wir eine eigene Funktion

$$X_C : \mathbb{R}^2 \to \mathbb{R}^2, \quad X_C(x) := x - \overline{x}_C. \tag{2.2}$$

Da die Berechnung von p_i für jede Zelle C_i auf die gleiche Art geschieht und die p_i in keiner Weise gekoppelt sind, die Berechnungen für die einzelnen Zellen also unabhängig voneinander vorgenommen werden können, reicht es aus, dass wir die Rekonstruktion für eine Zelle betrachten. Dies sei ohne Beschränkung der Allgemeinheit die Zelle C_1. Wir haben also folgende

Aufgabenstellung: Gegeben sei ein Gitter $\mathcal{G}$ eines beschränkten Gebietes $\Omega \subset \mathbb{R}^2$ mit Zellen C_i. Ferner die Mittelwerte $\{\overline{u}_i := \mathcal{A}(C_i)u : i = 1, \ldots, \#\mathcal{G}\}$ einer Funktion $u : \Omega \to \mathbb{R}$ sowie der Polynomgrad Q.
Zu bestimmen ist ein Polynom

$$p(x) = \sum_{|\alpha| \le Q} a_\alpha (x - \overline{x}_{C_1})^\alpha = \sum_{|\alpha| \le Q} a_\alpha X_{C_1}^\alpha(x), \tag{2.3}$$

welches u auf C_1 (punktweise) gut approximiert.
Die Anzahl der zu bestimmenden Polynomkoeffizienten a_α ist offensichtlich gleich $\mathcal{D}(Q) := (Q+1)(Q+2)/2$, der Dimension von $\Pi^Q(\mathbb{R}^2, \mathbb{R})$. Wir benötigen mindestens diese Anzahl von Bedingungen um die a_α bestimmen zu können. Da wir nur die Mittelwerte $\overline{u}_i$ auf den Zellen C_i kennen, nehmen wir zur Zelle C_1 nahegelegene Zellen dazu:

Definition 2.1.3. Sei

$$\mathcal{S} = \{C_{k_1} = C_1, C_{k_2}, \ldots, C_{k_{\#\mathcal{S}}}\},$$

$\mathcal{D}(Q) \le \#\mathcal{S} \le K(Q)$ eine Auswahl von paarweise verschiedenen Zellen. Wir fassen $\mathcal{S}$ gleichzeitig als Punktmenge auf:

$$x \in \mathcal{S} :\Leftrightarrow x \in \bigcup_{j=1,\ldots,\#\mathcal{S}} C_{k_j}.$$

Der Durchmesser von $\mathcal{S}$ sei durch $F(Q) \cdot h(C_1)$ nach oben beschränkt. Dann nennen wir $\mathcal{S}$ eine *Schablone*.

Für eine gegebene Schablone $\mathcal{S}$ betrachten wir die $\#\mathcal{S}$ Interpolationsbedingungen

$$\begin{aligned} \mathcal{A}(C_{k_j})p &= \overline{u}_{k_j}, \quad j = 1, \ldots, \#\mathcal{S} \quad \text{bzw.} \\ \sum_{|\alpha| \le Q} a_\alpha \mathcal{A}(C_{k_j}) X_{C_1}^\alpha &= \overline{u}_{k_j}, \quad j = 1, \ldots, \#\mathcal{S}. \end{aligned} \tag{2.4}$$

Aufgrund der Linearität der Operatoren $\mathcal{A}(C_{k_j})$ handelt es sich um ein lineares Gleichungssystem für die a_α. Wir wollen den Fall überbestimmter Systeme ($\#\mathcal{S} > \mathcal{D}(Q)$) nicht ausschließen. Für diesen Fall hat das System (2.4) im Allgemeinen keine Lösung. Vielmehr müssen wir uns dann damit begnügen die Bestapproximation im Sinne kleinster Quadrate zu bestimmen. Dies führt auf ein lineares Ausgleichsproblem, das wir in Matrixform als $\|\boldsymbol{A}a - \overline{u}\|_2 = \min$ schreiben können, wobei a der Vektor der gesuchten a_α, $\overline{u}$ der Vektor der $\overline{u}_{k_j}$ und

$$\boldsymbol{A}_{j,\alpha} := \mathcal{A}(C_{k_j}) X_{C_1}^\alpha, \quad j = 1, \ldots, \#\mathcal{S}, \quad |\alpha| \le Q, \quad \text{ist.} \tag{2.5}$$

Definition 2.1.4. Wir nennen eine Schablone $\mathcal{S}$ *zulässig*, wenn für sie die Matrix $\boldsymbol{A}$ vollen Rang $\mathcal{D}(Q)$ hat, sonst nennen wir sie *unzulässig*.

Da wir eine Approximation für Zelle C_1 bestimmen wollen, zeichnen wir diese Zelle in $\mathcal{S}$ aus und erzwingen, dass für sie die Interpolationsbedingung (2.4) exakt erfüllt wird. Dazu fordern wir auch im überbestimmten Fall, dass $\mathcal{A}(C_1)p = \overline{u}_1$. Um dies umzusetzen, nutzen wir aus, dass $\mathcal{A}(C_\ell) X_{C_1}^{(0,0)} = 1$ für alle Zellen C_ℓ gilt. Damit formen wir die erste Gleichung um zu

$$a_{(0,0)} = \overline{u}_1 - \sum_{1 \le |\alpha| \le Q} a_\alpha \mathcal{A}(C_1) X_{C_1}^\alpha. \tag{2.6}$$

Durch Einsetzen von $a_{(0,0)}$ in die restlichen Zeilen erhalten wir das reduzierte Ausgleichsproblem

$$\|\boldsymbol{B}a^r - \Delta\overline{u}\|_2 = \min, \tag{2.7}$$

wobei a^r der reduzierte Vektor der a_α,

$$\begin{aligned} \Delta\overline{u} &:= (\overline{u}_{k_2} - \overline{u}_1, \ldots, \overline{u}_{k_{\#\mathcal{S}}} - \overline{u}_1)^T \qquad \text{und} \\ \boldsymbol{B}_{j,\alpha} &:= \mathcal{A}(C_{k_j}) X_{C_{k_j}}^\alpha - \mathcal{A}(C_1) X_{C_1}^\alpha, \end{aligned}$$

$j = 2, \ldots, \#\mathcal{S}$, $1 \le |\alpha| \le Q$, ist.

Bemerkung 2.1.5. Hat die Matrix $\boldsymbol{A}$ vollen Rang $\mathcal{D}(Q)$, so hat die Matrix $\boldsymbol{B}$ vollen Rang $\mathcal{D}(Q) - 1$.

Zur Berechnung von p kann jetzt (2.7) gelöst werden und $a_{(0,0)}$ wird anschließend durch (2.6) berechnet. Dabei muss in der Praxis aber sehr sorgfältig vorgegangen werden, da das Ausgleichsproblem zu (2.4) und das Ausgleichsproblem (2.7) für kleine $h := h(C_1)$ schlecht konditioniert sind.

Definition 2.1.6. Die *Konditionszahl* eines linearen Ausgleichsproblems wird definiert als

$$\mathrm{cond}\left(\|\boldsymbol{A}x - b\|_2 = \min\right) := \sqrt{\mathrm{cond}_2\left(\boldsymbol{A}^T\boldsymbol{A}\right)}.$$

Für eine schlechte Kondition des Problems gibt es zwei mögliche Gründe. Zum einen kann die Schablone $\mathcal{S}$ (fast) unzulässig sein, zum anderen wächst bei Verfeinerung der Diskretisierung die Kondition des Ausgleichsproblems zu (2.4) mit $h^{-Q}(C_1)$ und die von (2.7) immer noch mit $h^{1-Q}(C_1)$.
Ersteres kann man dadurch umgehen, dass man bei der numerischen Umsetzung (fast) unzulässige Schablonen vermeidet. Der zweite Grund liegt immanent in der Aufgabenstellung verborgen und kann nur teilweise umgangen werden, wie wir im folgenden Abschnitt ausführen werden.

2.1.1 Behandlung der schlechten Kondition des Interpolationsproblems

Die grundlegende Problematik bei der Interpolationsaufgabe liegt in der unterschiedlichen Skalierung der Spalten des Systems durch das Auftreten unterschiedlich hoher Potenzen von $(x - \overline{x}_{C_1})$. Dies kann durch Verwendung eines geeigneten lokalen Koordinatensystems umgangen werden.

Interpolation in baryzentrischen Koordinaten

In [Abg94b] werden baryzentrische Koordinaten eingeführt. Dazu werden aus $\mathcal{S}$ neben C_1 zwei weitere Zellen ausgewählt, so dass die Schwerpunkte der drei Zellen nicht kollinear sind. Der Einfachheit halber wollen wir annehmen, dass dies die Zellen C_2 und C_3 sind. Dann werden drei Polynome Λ_1, Λ_2 und Λ_3 vom Grad eins definiert durch die Bedingungen

$$\mathcal{A}(C_i)\Lambda_j = \begin{cases} 1, \text{ falls } i = j \\ 0, \text{ sonst} \end{cases} \qquad 1 \le i \le 3, 1 \le j \le 3.$$

Es gilt $\Lambda_1 + \Lambda_2 + \Lambda_3 = 1$. Die drei Polynome sind die baryzentrischen Koordinaten des Dreiecks, das von den Schwerpunkten der Zellen C_1, C_2 und

C_3 aufgespannt wird. Unter Verwendung von Λ_2 und Λ_3 erhält Abgrall eine neue Repräsentation des Polynoms p anstelle von Darstellung (2.3):

$$p = \sum_{|\alpha| \le Q} \hat{a}_\alpha \Lambda_2^{\alpha_1} \Lambda_3^{\alpha_2}.$$

Das zugehörige Ausgleichsproblem zur Berechnung der $\hat{a}_\alpha$ hat eine Konditionszahl, die von h unabhängig ist (siehe [Abg94b]).

Interpolation in skalierten Koordinaten

Wir bevorzugen eine einfachere Technik zur Bestimmung eines lokalen Koordinatensystems. Dazu definieren wir einen lokalen Skalierungsfaktor

$$s := \frac{1}{\sqrt{|C_1|}}$$

und transformieren die Darstellung (2.3) in

$$p(x) = \sum_{|\alpha| \le Q} \tilde{a}_\alpha s^{|\alpha|} (x - \overline{x}_{C_1})^\alpha. \tag{2.8}$$

Zur Berechnung der $\tilde{a}_\alpha$ ergibt sich hier das volle Ausgleichsproblem

$$\|\tilde{\boldsymbol{A}} \tilde{a} - \overline{u}\|_2 = \min, \tag{2.9}$$

wobei $\tilde{a}$ der Vektor der unbekannten $\tilde{a}_\alpha$ ist und

$$\tilde{\boldsymbol{A}}_{j,\alpha} := s^{|\alpha|} \mathcal{A}(C_{k_j}) X_{C_1}^\alpha, \quad j = 1, \ldots, \#\mathcal{S}, \quad |\alpha| \le Q. \tag{2.10}$$

Wie im Fall der unskalierten Darstellung können wir hier die erste Gleichung abspalten und kommen zu dem reduzierten Ausgleichsproblem

$$\|\tilde{\boldsymbol{B}} \tilde{a}^r - \Delta \overline{u}\|_2 = \min, \tag{2.11}$$

wobei $\tilde{a}^r$ der reduzierte Vektor der $\tilde{a}_\alpha$ ist und

$$\tilde{\boldsymbol{B}}_{j,\alpha} := s^{|\alpha|} \big(\mathcal{A}(C_{k_j}) X_{C_{k_j}}^\alpha - \mathcal{A}(C_1) X_{C_1}^\alpha \big), \tag{2.12}$$

$j = 2, \ldots, \#\mathcal{S}$, $1 \le |\alpha| \le Q$.

Satz 2.1.7. *Bei exakter Arithmetik erhält man durch Lösen von* (2.11) *und die Formel* $a_\alpha = s^{|\alpha|} \tilde{a}_\alpha$ *dieselben Ergebnisse wie durch Lösen von* (2.7).

Beweis. Sei $\boldsymbol{S}$ die Diagonalmatrix der $s^{|\alpha|}$. Offensichtlich gilt $\tilde{\boldsymbol{B}} = \boldsymbol{B}\boldsymbol{S}$. Auflösen der Normalengleichungen $\boldsymbol{B}^T\boldsymbol{B}a^r = \boldsymbol{B}^T\Delta\overline{u}$ für (2.7) liefert

$$a^r = \left(\boldsymbol{B}^T\boldsymbol{B}\right)^{-1}\boldsymbol{B}^T\Delta\overline{u}.$$

Auflösen der Normalengleichungen $\tilde{\boldsymbol{B}}^T\tilde{\boldsymbol{B}}\tilde{a}^r = \tilde{\boldsymbol{B}}^T\Delta\overline{u}$ für (2.11) liefert

$$\begin{aligned}\tilde{a}^r &= \left(\tilde{\boldsymbol{B}}^T\tilde{\boldsymbol{B}}\right)^{-1}\tilde{\boldsymbol{B}}^T\Delta\overline{u} = \left(\boldsymbol{S}\boldsymbol{B}^T\boldsymbol{B}\boldsymbol{S}\right)^{-1}\boldsymbol{S}\boldsymbol{B}^T\Delta\overline{u} \\ &= \boldsymbol{S}^{-1}\left(\boldsymbol{B}^T\boldsymbol{B}\right)^{-1}\boldsymbol{B}^T\Delta\overline{u} = \boldsymbol{S}^{-1}a^r.\end{aligned}$$

Da $a_\alpha = s^{|\alpha|}\tilde{a}_\alpha$ in Matrixschreibweise nichts anderes bedeutet als $a^r = \boldsymbol{S}\tilde{a}^r$, ist damit die Behauptung bewiesen. □

Die entsprechende Aussage gilt für die nicht-reduzierten Probleme (2.4) und (2.9). Der Beweis geht genauso wie der Beweis für die reduzierten Systeme.

Bemerkung 2.1.8. Wir haben also insgesamt folgende Technik zur Berechnung der Koeffizienten a_α von p erhalten:

1. Berechne die $\tilde{a}_\alpha$, $1 \le |\alpha| \le Q$ durch Lösen von (2.11).
2. Für $1 \le |\alpha| \le Q$ setze $a_\alpha := s^{|\alpha|}\tilde{a}_\alpha$.
3. Berechne $a_{(0,0)}$ mittels Gleichung (2.6).

Satz 2.1.9. *Die Interpolationsmatrix $\tilde{\boldsymbol{A}}$ des vollen Ausgleichsproblems* (2.9) *sowie die Interpolationsmatrix $\tilde{\boldsymbol{B}}$ des reduzierten Ausgleichsproblems* (2.11) *sind invariant unter Gitterskalierung. Das heißt, gegeben seien eine affine Transformation Ψ der Form*

$$\Psi : \mathbb{R}^2 \to \mathbb{R}^2, \quad x \mapsto fx + y, \quad 0 < f \in \mathbb{R}, \quad y \in \mathbb{R}^2,$$

und transformierte Zellen $\Psi(C_\ell)$. Dann gilt für die Matrizen $\tilde{\boldsymbol{A}}^\Psi$ und $\tilde{\boldsymbol{B}}^\Psi$, die zu der transformierten Geometrie gehören, dass

$$\tilde{\boldsymbol{A}}^\Psi = \tilde{\boldsymbol{A}} \text{ sowie } \tilde{\boldsymbol{B}}^\Psi = \tilde{\boldsymbol{B}}.$$

Insbesondere ist also die Konditionszahl der skalierten Ausgleichsprobleme (2.9) *und* (2.11) *unabhängig von $h(C_1)$.*

Beweis. Für den Skalierungsfaktor s^Ψ der transformierten Geometrie erhalten wir

$$s^\Psi = \frac{1}{\sqrt{|\Psi(C_1)|}} = \frac{1}{\sqrt{f^2|C_1|}} = \frac{s}{f}.$$

Ferner ist

$$\begin{aligned}\mathcal{A}(\Psi(C_\ell))(X_{\Psi(C_1)} \circ \Psi)^\alpha &= \frac{1}{|\Psi(C_\ell)|} \int_{\Psi(C_\ell)} \left(\Psi(x) - \overline{x}_{\Psi(C_1)}\right)^\alpha d\Psi(x) \\ &= \frac{1}{f^2|C_\ell|} \int_{C_\ell} (fx - f\overline{x}_{C_1})^\alpha \, f^2 dx \\ &= f^{|\alpha|} \mathcal{A}(C_\ell) X_{C_1}^\alpha.\end{aligned}$$

Setzt man die beiden Beziehungen in die Matrizen $\tilde{\boldsymbol{A}}^\Psi$ und $\tilde{\boldsymbol{B}}^\Psi$ ein, so folgen unmittelbar die behaupteten Gleichungen. □

Abschließende Betrachtungen zum Konditionsproblem

Durch die Einführung lokaler Koordinatensysteme — wie das Konzept der Skalierung — erhält man Ausgleichsprobleme, deren Konditionszahl unabhängig von der Feinheit von $\mathcal{G}$ ist. Diese Ausgleichsprobleme können, wie erwähnt, noch dadurch schlecht konditioniert sein, dass die gewählte Schablone (fast) unzulässig ist. Da dies aber jetzt der einzige mögliche Grund für eine schlechte Kondition ist, kann man bei der numerischen Umsetzung solche Schablonen durch Messen der Konditionszahl erkennen und dann verwerfen.

Definition 2.1.10. Wir nennen eine Schablone *numerisch zulässig*, wenn die Konditionszahl des skalierten Problems (2.9) kleiner als eine vorgegene Schranke ist. Sonst nennen wir sie *numerisch unzulässig*.

Da durch das Skalieren alle Matrixeinträge der Interpolationsmatrix auf die Größenordnung eins gebracht werden, sind also für eine numerisch zulässige Schablone $\|\tilde{\boldsymbol{B}}^T\tilde{\boldsymbol{B}}\|_2$ und $\|(\tilde{\boldsymbol{B}}^T\tilde{\boldsymbol{B}})^{-1}\|_2$ in der Größenordnung eins.
Wir wollen uns nun ansehen, was mit den Koeffizienzen a_α und mit p geschieht, wenn die Daten $\overline{u}_\ell$ gestört sind.

Satz 2.1.11. *$\mathcal{G}$ sei aus einer Menge regulärer Gitter und $\mathcal{S}$ sei eine numerisch zulässige Schablone für Zelle C_1. $p(x) = \sum_{|\alpha|\le Q} a_\alpha (x - \overline{x}_{C_1})^\alpha$ sei die exakte Lösung des linearen Ausgleichsproblems (2.6), (2.7). $\overline{u}_{\varepsilon,\ell}$ mit $|\overline{u}_{\varepsilon,\ell} - \overline{u}_\ell| = \mathcal{O}(\varepsilon)$ seien gestörte Daten. $p_\varepsilon(x) = \sum_{|\alpha|\le Q} a_{\varepsilon,\alpha}(x - \overline{x}_{C_1})^\alpha$ sei die exakte Lösung zu den $\overline{u}_{\varepsilon,\ell}$. Dann gilt*

1. *$|a_\alpha - a_{\varepsilon,\alpha}| = \mathcal{O}(\varepsilon) \cdot \mathcal{O}(h^{-|\alpha|}(C_1))$,*
2. *$|p(x) - p_\varepsilon(x)| = \mathcal{O}(\varepsilon)$ gleichmäßig für alle $x \in C_1$.*

Beweis. Wir berechnen die a_α und die $a_{\varepsilon,\alpha}$ über den Umweg des skalierten Ausgleichsproblems (2.11). Da die Schablone $\mathcal{S}$ numerisch zulässig ist, ist

das skalierte Problem gut konditioniert, $\|(\tilde{\boldsymbol{B}}^T\tilde{\boldsymbol{B}})^{-1}\|_2 = \mathcal{O}(1)$ und es gilt $|\tilde{a}_\alpha - \tilde{a}_{\varepsilon,\alpha}| = \mathcal{O}(\varepsilon)$. Also gilt

$$|a_\alpha - a_{\varepsilon,\alpha}| = s^{|\alpha|}|\tilde{a}_\alpha - \tilde{a}_{\varepsilon,\alpha}| = \mathcal{O}(\varepsilon)\cdot\frac{1}{(\sqrt{|C_1|})^{|\alpha|}} = \mathcal{O}(\varepsilon)\cdot\mathcal{O}(h^{-|\alpha|}(C_1)).$$

Für die zweite Aussage verwenden wir, dass $|x-\overline{x}_{C_1}| \le h(C_1)$ für alle $x \in C_1$. Also ist für alle $x \in C_1$

$$\begin{aligned} |p(x) - p_\varepsilon(x)| &= \left|\sum_{|\alpha|\le Q}(a_\alpha - a_{\varepsilon,\alpha})(x-\overline{x}_{C_1})^\alpha\right| \le \sum_{|\alpha|\le Q}|a_\alpha - a_{\varepsilon,\alpha}||x-\overline{x}_{C_1}|^\alpha \\ &\le \sum_{|\alpha|\le Q}|a_\alpha - a_{\varepsilon,\alpha}|h^{|\alpha|}(C_1) = \mathcal{O}(\varepsilon)\cdot\mathcal{O}(h^{-|\alpha|}(C_1))\cdot h^{|\alpha|}(C_1) \\ &= \mathcal{O}(\varepsilon). \end{aligned}$$

□

Bei feinen Diskretisierungen hängen also insbesondere die höheren Koeffizienten von p sehr empfindlich von den Daten $\overline{u}_\ell$ ab. Für $x \in C_1$ sind die Werte $p(x)$ hingegen (bei sorgfältiger Vorgehensweise) unempfindlich gegenüber gestörten Daten.

2.1.2 Die Approximationseigenschaft der Polynom-Rekonstruktion

Ziel dieses Abschnitts ist es nachzuweisen, dass — unter geeigneten Voraussetzungen an die verwendeten Gitter und Schablonen — die Polynom-Rekonstruktion aus Bemerkung 2.1.8 ein *Rekonstruktionsalgorithmus der Ordnung* $Q+1$ gemäß Definition 1.2.8 (siehe S. 22) ist.

Lemma 2.1.12. *$\mathcal{G}$ sei aus einer Menge regulärer Gitter und $\mathcal{S}$ sei eine numerisch zulässige Schablone für Zelle C_1. $q(x) \in \Pi^Q(\mathbb{R}^2,\mathbb{R})$ sei ein beliebiges Polynom vom Grad kleiner oder gleich Q und die Daten $\overline{u}_i := \mathcal{A}(C_i)q$ seien gegeben. Dann gilt für die Lösung p, die man mit der Technik aus Bemerkung* 2.1.8 *erhält, dass $p = q$.*

Beweis. Offensichtlich ist $p = q$ eine Lösung von System (2.4). Da $\mathcal{S}$ zulässig ist, hat das System vollen Rang und die Lösung ist eindeutig. Da das volle System exakt lösbar ist, erhält man über das reduzierte System (2.6), (2.7) dieselbe Lösung. Nach Satz 2.1.7 erhält man also dieselbe Lösung auch durch die Technik aus Bemerkung 2.1.8. □

Damit haben wir alle Vorbereitungen zum Beweis des Hauptsatzes dieses Kapitels getroffen.

Satz 2.1.13. *$\mathcal{G}$ sei aus einer Menge regulärer Gitter für das beschränkte Gebiet $\Omega \subset \mathbb{R}^2$ und $\mathcal{S}$ sei eine numerisch zulässige Schablone für Zelle C_1. $u \in C^{Q+1}(\operatorname{conv}\Omega, \mathbb{R})$ sei eine $Q+1$ mal stetig differenzierbare Funktion auf der konvexen Hülle von*[1] *Ω und die Daten $\overline{u}_i := \mathcal{A}(C_i)u$ für alle Zellen C_i von $\mathcal{G}$ seien gegeben. Dann gilt für die Lösung p, die man mit der Technik aus Bemerkung* 2.1.8 *erhält*

$$\max_{x \in C_1} |p(x) - u(x)| \leq K_G \cdot K(u) \cdot h^{Q+1}(C_1).$$

Die Konstante K_G hängt von der Menge der betrachteten Gitter ab, ist aber innerhalb der Menge fest. $K(u)$ hängt nur von der $(Q+1)$-ten Ableitung von u ab.

Beweis. Wir betrachten die Taylor-Entwicklung von u zum Entwicklungspunkt $\overline{x}_{C_1}$:

$$u(x) = q(x) + \sum_{|\alpha|=Q+1} \frac{D^\alpha u\big(\overline{x}_{C_1} + \theta(x)(x - \overline{x}_{C_1})\big)}{(Q+1)!}(x - \overline{x}_{C_1})^\alpha$$

mit dem Polynom

$$q(x) := \sum_{|\alpha|\leq Q} \frac{D^\alpha u(\overline{x}_{C_1})}{|\alpha|!}(x - \overline{x}_{C_1})^\alpha$$

und $\theta(x) \in [0,1]$. Nach Definition ist der Durchmesser von $\mathcal{S}$ durch $F(Q) \cdot h(C_1)$ nach oben beschränkt. Daher läßt sich

$$\max_{x \in \mathcal{S}} |x - \overline{x}_{C_1}| \leq K_a h(C_1)$$

mit K_a unabhängig von C_1 und $\mathcal{G}$ abschätzen. Also ist

$$\max_{x \in \mathcal{S}} |u(x) - q(x)| \leq \frac{K_a}{Q!} \max_{\substack{x \in \operatorname{conv}\mathcal{S} \\ |\alpha|=Q+1}} |D^\alpha u(x)| \cdot h^{Q+1}(C_1).$$

Damit ist auch

$$\max_{j=1,\dots,\#\mathcal{S}} |\mathcal{A}(C_{k_j})u - \mathcal{A}(C_{k_j})q| \leq \frac{K_a}{Q!} \max_{\substack{x \in \operatorname{conv}\mathcal{S} \\ |\alpha|=Q+1}} |D^\alpha u(x)| \cdot h^{Q+1}(C_1). \qquad (2.13)$$

[1] Es ist ausreichend, wenn u eine $Q+1$ mal stetig differenzierbare Funktion auf der Vereinigung der konvexen Hüllen aller betrachteten Schablonen $\mathcal{S}$ ist.

Nach Lemma 2.1.12 wird das Polynom q aus den Daten $\mathcal{A}(C_{k_j})q$ mit der Technik aus Bemerkung 2.1.8 exakt rekonstruiert. Die Daten $\overline{u}_{k_j} = \mathcal{A}(C_{k_j})u$ können wir nach Gleichung (2.13) als gestörte Daten der $\mathcal{A}(C_{k_j})q$ auffassen. Also ist nach Satz 2.1.11

$$\max_{x \in C_1} |p(x) - u(x)| = \mathcal{O}(h^{Q+1}(C_1)).$$

Es ist

$$K(u) = \max_{\substack{x \in \operatorname{conv} \mathcal{S} \\ |\alpha| = Q+1}} |D^\alpha u(x)|.$$

K_G ergibt sich aus K_a und der Norm der Matrix $(\tilde{\boldsymbol{B}}^T \tilde{\boldsymbol{B}})^{-1}$, die wegen der numerischen Zulässigkeit der Schablone global durch eine Konstante nach oben abgeschätzt werden kann. □

2.1.3 Berechnung der Matrix-Einträge

Um das Ausgleichsproblem (2.11) praktisch lösen zu können müssen wir die Einträge von $\tilde{\boldsymbol{B}}$ berechnen. Dazu müssen die Integrale $\int_{C_\ell} (x - \overline{x}_{C_1})^\alpha dx$ für alle α mit $|\alpha| \leq Q$ berechnet werden. Da die C_ℓ beliebige Polygone sind, können wir nicht hoffen diese Integrale für $|\alpha| > 1$ direkt durch eine Quadraturformel lösen zu können. Für $|\alpha| \leq 1$ können die Mittelwerte geschlossen angegeben werden:

$$\begin{aligned} \mathcal{A}(C_\ell) X_{C_1}^{(0,0)} &= 1, \\ \mathcal{A}(C_\ell) X_{C_1}^{(1,0)} &= (\overline{x}_{C_\ell} - \overline{x}_{C_1})_1, \\ \mathcal{A}(C_\ell) X_{C_1}^{(0,1)} &= (\overline{x}_{C_\ell} - \overline{x}_{C_1})_2. \end{aligned} \tag{2.14}$$

Für $|\alpha| > 1$ kommen zwei Lösungsmöglichkeiten in Betracht. Zum einen können die Zellen in Dreiecke subdividiert werden und auf den Dreiecken kann eine Quadraturformel hinreichend hoher Ordnung verwendet werden. Zum anderen kann man für jedes $(x - \overline{x}_{C_1})^\alpha$ eine *Stammfunktion* $\mathcal{X}^\alpha$ bestimmen, so dass $\operatorname{div} \mathcal{X}^\alpha = (x - \overline{x}_{C_1})^\alpha$ und kann mit Hilfe des Gaußschen Integralsatzes die Flächenintegrale auf Randintegrale zurückführen (siehe dazu auch [HMS96]):

$$\int_{C_\ell} (x - \overline{x}_{C_1})^\alpha dx = \int_{C_\ell} \operatorname{div} \mathcal{X}^\alpha dx = \int_{\partial C_\ell} \mathcal{X}^\alpha \cdot n ds.$$

Diese können wir exakt lösen, da die Ränder der Zellen polygonal sind. Wir haben uns für die zweite Möglichkeit entschieden, da sie genau zur Philosophie der Finite-Volumen-Verfahren passt. Eine *Stammfunktion* zu $(x-\overline{x}_{C_1})^\alpha$ kann man zum Beispiel durch

$$\mathcal{X}^\alpha := \Big(\int (x-\overline{x}_{C_1})^\alpha dx_1, 0\Big)^T \tag{2.15}$$

bestimmen. Leider ist der Aufwand zur Berechnung der Randintegrale sehr hoch. Andererseits verbietet es sich aus Gründen des Speicheraufwands alle benötigten $\mathcal{A}(C_\ell)X^\alpha_{C_i}$ abzuspeichern. Diese Probleme können wir umgehen, indem wir die Integrale umschreiben. So erhalten wir für $|\alpha| = 2$

$$\mathcal{A}(C_\ell)X^\alpha_{C_i} = (\overline{x}_{C_\ell} - \overline{x}_{C_1})^\alpha + \mathcal{A}(C_i)X^\alpha_{C_i}. \tag{2.16}$$

Wir speichern die $\mathcal{A}(C_i)X^\alpha_{C_i}$ für alle Zellen C_i ab und kommen so zu effizienten Berechnungen bei akzeptablem Speicheraufwand. Für beliebige $|\alpha|$ lässt sich eine Rekursionsformel herleiten (siehe [Hem99]).

2.2 Gewichtete wesentlich nicht-oszillierende Verfahren

2.2.1 Motivation

Die Verwendung der Rekonstruktions-Technik aus dem letzten Abschnitt als Rekonstruktionsalgorithmus für unser Finite-Volumen-Verfahren ist nicht sinnvoll, was wir an einem Modellbeispiel zeigen wollen. Wir betrachten die lineare Advektionsgleichung (siehe S. 14) auf dem Einheitsquadrat $[0,1]\times[0,1]$ und setzen

$$u(x,0) := \begin{cases} 1, & \text{falls } |x-(0.3,0.3)^T| < 0.25, \\ 0, & \text{sonst.} \end{cases} \tag{2.17}$$

$u(\cdot,0)$ ist also unstetig auf dem Rand des Kreises mit Radius 0.25 und Zentrum $(0.3,0.3)^T$. Die lineare Advektionsgleichung mit dem von uns verwendeten Flusstensor beschreibt die Verschiebung von u in Richtung des Vektors $(1,1)^T$. Die exakte Lösung zur Zeit $T = 0.3$ ist also gegeben durch

$$u(x,0.3) = \begin{cases} 1, & \text{falls } |x-(0.6,0.6)^T| < 0.25, \\ 0, & \text{sonst.} \end{cases} \tag{2.18}$$

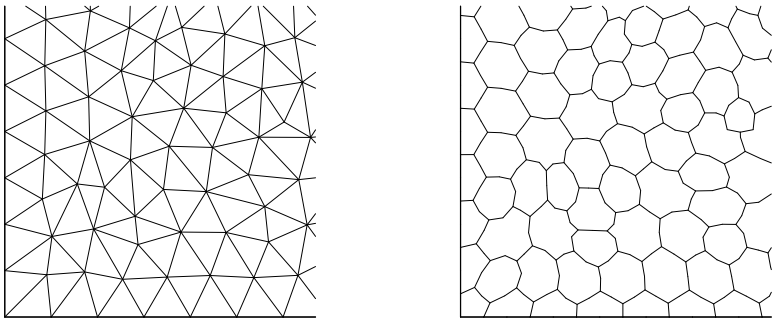

Abbildung 2.1: Ausschnitt $[0, 0.1] \times [0, 0.1]$ aus der verwendeten Triangulierungen und der baryzentrischen Unterteilung.

Das Gebiet wird durch die baryzentrische Unterteilung (siehe S. 18) einer Triangulierung diskretisiert. Der Ausschnitt $[0, 0.1] \times [0, 0.1]$ der Triangulierung und der baryzentrischen Unterteilung sind in Abbildung 2.1 zu sehen.
Als Rekonstruktionsalgorithmus wird die im letzten Abschnitt beschriebene Technik verwendet um Polynome des Grads eins zu rekonstruieren. Als Schablonen werden zu jeder Zelle jeweils genau alle Kantennachbarn genommen. Die Zeitdiskretisierung wird mit dem zweistufigen Verfahren von Shu und Osher vorgenommen. In Abbildung 2.2 ist die berechnete Näherungslösung entlang der Diagonalen von $(0, 0)^T$ nach $(1, 1)^T$ zusammen mit der exakten Lösung sowie der Näherungslösung der Basisdiskretisierung zu sehen.

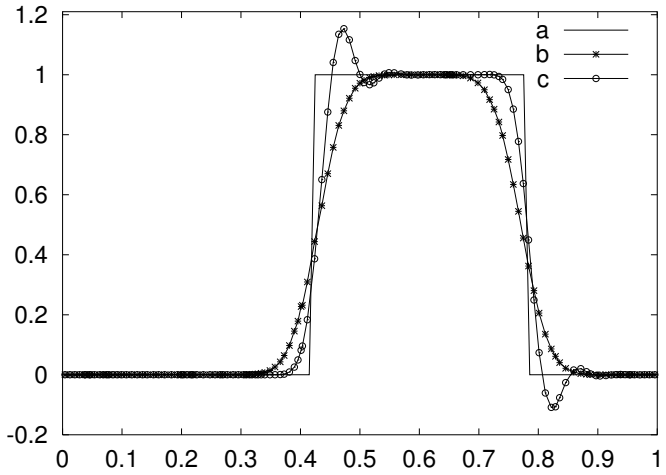

Abbildung 2.2: a) Exakte Lösung, b) Näherungslösung aus der Basisdiskretisierung sowie c) mit Polynom-Rekonstruktion zur Zeit $T = 0.3$.

Die Oszillationen an den Unstetigkeitsstellen sind nicht zu übersehen. Sie

entstehen durch die Interpolation über die Unstetigkeitsstellen hinweg. Aus der Literatur sind zwei grundsätzlich verschiedene Techniken bekannt um dieses Oszillationsproblem zu lösen. Die ältere Technik ist die Verwendung von sogannten *Limitern*. Hierbei wird üblicherweise in jedem Rekonstruktionsschritt die Variation der Rekonstruktionsfunktion ϕ^i auf einer Zelle C_i mit der Variation der umliegenden Mittelwerte verglichen. Ist die Variation von ϕ^i zu hoch, wird ϕ^i mit einem Dämpfungsfaktor $\theta \in [0,1]$, dem *Limiter*, der trivialen Rekonstruktion $\phi^{t,i} \equiv \overline{u}_i$ angenähert:

$$\tilde{\phi}^i := \theta\phi^i + (1-\theta)\phi^{t,i}.$$

Für $\theta = 1$ findet keine Dämpfung statt, $\tilde{\phi}^i = \phi^i$. Im anderen Extremfall $\theta = 0$ wird ϕ^i komplett ausgeschaltet, es ist dann $\tilde{\phi}^i = \phi^{t,i}$. Zu näheren Einzelheiten über diese Verfahren sei auf [Hem99] verwiesen. Dort wird auch eine weitere Technik eingeführt, die Limiter unabhängig von der Variation der umliegenden Mittelwerte zu berechnen. In diesem Abschnitt werden wir uns mit der zweiten Technik, den sogenannten *wesentlich nicht oszillierenden Verfahren (ENO-Verfahren)* und deren Erweiterung auf *gewichtete wesentlich nicht oszillierende Verfahren (WENO-Verfahren)* beschäftigen.
ENO-Verfahren auf Triangulierungen und auf allgemeineren Typen von Gittern wurden in [HC91] vorgestellt und später in [Son97a], [Abg94a] und [Abg94b] weiterentwickelt und analysiert. In [LOC94] und [JS96] wurden gewichtete ENO-Verfahren für eine Raumdimension vorgestellt. Diese Verfahren wurden von uns auf den Fall von unstrukturierten Gittern übertragen (siehe [Fri98]).

2.2.2 Das Prinzip der WENO-Rekonstruktion

Das Prinzip des Rekonstruktionsalgorithmus' für die Berechnung eines Rekonstruktionspolynoms[2] p für eine Zelle C_1 kann wie folgt beschrieben werden:

1. Bestimme eine Menge von numerisch zulässigen Schablonen $\mathcal{S}_1, \ldots, \mathcal{S}_m$.
2. Für jede dieser Schablonen $\mathcal{S}_i$ berechne das Rekonstruktionspolynom p_i.
3. Für jedes der Polynome p_i bestimme einen *Oszillationsindikator* $OI(p_i)$.

[2] Das Grundprinzip ist nicht auf die Rekonstruktion von Polynomen beschränkt. In [Son97b] wird es in einem wesentlich allgemeineren Kontext angewandt.

4. In Abhängigkeit von den $OI(p_i)$ bestimme positive Gewichte w_i, deren Gesamtsumme eins ist. w_i soll klein sein, wenn $OI(p_i)$ groß ist und groß, wenn $OI(p_i)$ klein ist.

5. Setze $p := \sum_{i=1}^{m} w_i p_i$.

Den klassischen Fall des (digitalen) ENO-Verfahrens erhält man dadurch, dass genau ein $w_{i^\star}$ auf eins gesetzt wird und alle anderen w_i auf null, wobei $i^\star$ durch die Bedingung $OI(p_{i^\star}) = \min_{i=1,\dots,m} OI(p_i)$ festgelegt wird.
Die Motivation für die Auswahl mehrerer Schablonen für die Rekonstruktion auf einer Zelle besteht darin die Interpolation über Unstetigkeitsstellen von u hinweg zu umgehen: Befindet sich unter den ausgewählten Schablonen eine, die keine Unstetigkeitsstelle von u enthält, so wird die *Oszillation* des zugehörigen Polynoms deutlich kleiner sein als die *Oszillation* von Polynomen, die über Unstetigkeitsstellen von u hinweg interpoliert wurden. Durch die ensprechende Gewichtung hat dann auch das resultierende Polynom p eine geringe *Oszillation*, selbst wenn Unstetigkeitsstellen in der Nähe der betrachteten Zelle C_1 sind.
Der zweite Schritt des Algorithmus, die Berechnung der p_i, wird gemäß Bemerkung 2.1.8 durchgeführt, der fünfte Schritt ist trivial. Alle anderen Schritte erfordern genauere Betrachtungen, die wir in den folgenden Abschnitten durchführen.

Satz 2.2.1. *Der oben beschriebene Rekonstruktionsalgorithmus ist unter den Voraussetzungen von Satz* 2.1.13 *ein Rekonstruktionsalgorithmus der Fehlerordnung* $Q+1$.

Beweis. p ist eine Konvexkombination der p_i. Also gilt

$$\begin{aligned} |p(x) - u(x)| &= \Big|\sum_{i=1}^{m} w_i p_i(x) - \sum_{i=1}^{m} w_i u(x)\Big| \\ &= \Big|\sum_{i=1}^{m} w_i (p_i(x) - u(x))\Big| \;\le\; \sum_{i=1}^{m} w_i |p_i(x) - u(x)| \\ &\le \max_{i=1,\dots,m} |p_i(x) - u(x)|. \end{aligned}$$

Da alle p_i nach Satz 2.1.13 die behauptete Fehlerordnung $Q+1$ haben, folgt die Behauptung. □

2.2.3 Auswahl von Schablonen

Die Auswahl von Schablonen hängt stark von der Art der verwendeten Gitter $\mathcal{G}$ und vom Polynomgrad Q ab. Wir beschränken uns in diesem Abschnitt auf

die Polynomgrade $Q = 1$ und $Q = 2$ und betrachten zum einen Gitter vom Box-Typ (siehe Definition 1.2.6, S. 20). Der für uns wichtigste Spezialfall solcher Gitter ist die baryzentrische Unterteilung einer Triangulierung. Zum anderen betrachten wir den Fall, in dem $\mathcal{G}$ eine Triangulierung ist.
Wie in [Abg94a, Abg94b, Son97a] verwenden wir keine überbestimmten Schablonen. Wir setzen also $K(1) := \mathcal{D}(1) = 3$ und $K(2) := \mathcal{D}(2) = 6$ (siehe Definition 2.1.3, S. 40).

Polynomgrad eins

Für den Polynomgrad $Q = 1$ müssen wir zur Zelle C_1 noch zwei Zellen hinzunehmen um eine Schablone zu erhalten.

Gitter vom Box-Typ Für Gitter vom Box-Typ wählen wir alle Paare von Kantennachbarn von C_1, die selbst untereinander Kantennachbarn sind (siehe Abbildung 2.3).

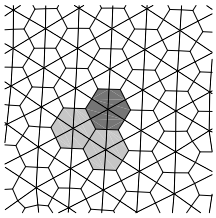

Abbildung 2.3: Schablone für Polynomgrad eins.

Verwenden wir die baryzentrische Unterteilung einer Triangulierung, so können wir die topologischen Informationen der Triangulierung ausnutzen um eben diese Schablonen zu erhalten. Nach Konstruktion gibt es eine bijektive Beziehung zwischen den Eckpunkten der Dreiecke und den Zellen der baryzentrischen Unterteilung. Die drei Zellen, die den Eckpunkten jedes Dreiecks zugeordnet sind, definieren jeweils eine Schablone.

Triangulierungen Für Triangulierungen verwenden wir die Schablonen, die in [HS99] angegeben sind. Dies sind die zentral ausgerichteten Schablonen $\{T_1, T_i, T_j\}$, $\{T_1, T_j, T_k\}$ und $\{T_1, T_k, T_i\}$ sowie die stärker einseitig ausgerichteten Schablonen $\{T_1, T_i, T_{i_a}\}$, $\{T_1, T_i, T_{i_b}\}$, $\{T_1, T_j, T_{j_a}\}$, $\{T_1, T_j, T_{j_b}\}$, $\{T_1, T_k, T_{k_a}\}$ und $\{T_1, T_k, T_{k_b}\}$ (siehe Abbildung 2.4).

Bemerkung 2.2.2. Für den Polynomgrad eins können wir eine geometrische Bedingung für die Zulässigkeit einer Schablone angeben: Eine Schablone für Polynomgrad eins aus drei Zellen ist genau dann zulässig, wenn die Schwerpunkte der drei Zellen nicht kollinear sind.

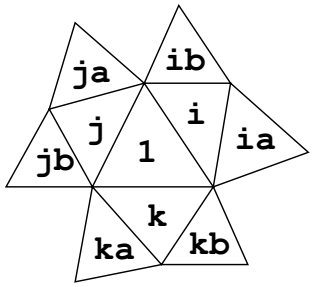

Abbildung 2.4: Kantennachbarn von Dreieck T_1 sowie deren Kantennachbarn.

Polynomgrad zwei

Für den Polynomgrad $Q = 2$ müssen wir zur Zelle C_1 noch fünf Zellen `a`, `b`, `c`, `d`, `e` hinzunehmen um eine Schablone zu erhalten.

Gitter vom Box-Typ Für Gitter vom Box-Typ sind unterschiedlich stark zentral oder einseitig ausgerichtete Schablonen denkbar. Wir haben zunächst vier verschiedene Typen betrachtet, die in Abbildung 2.5 dargestellt sind.

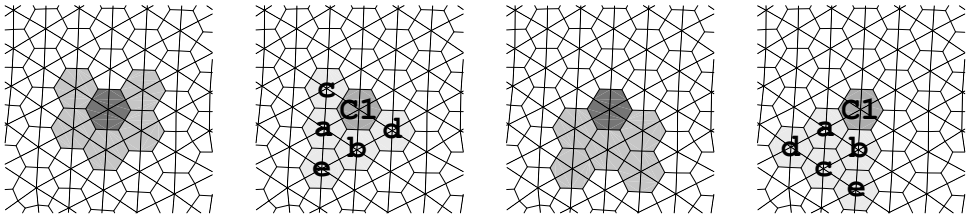

Abbildung 2.5: Typ 1–4 von Schablonen für Polynomgrad zwei.

Numerische Experimente haben dann ergeben, dass nicht die volle Auswahl erforderlich ist. Typ 2 als zentral ausgerichteter Typ und Typ 4 als einseitig ausgerichteter Typ reichen aus um sowohl Testfälle mit starken Unstetigkeiten als auch *glatte* Testfälle gut rechnen zu können. In keinem Fall hat die zusätzliche Verwendung der Typen 1 und 3 zu einer wesentlichen Veränderung der Ergebnisse geführt.
Wir beschreiben die Konstruktion der Schablonen wiederum zunächst ohne die Annahme, dass eine baryzentrische Unterteilung einer Triangulierung verwendet wird und dann unter Ausnutzung einer Triangulierung:
Die Zellen `a` und `b` sind wie beim Polynomgrad 1 ein Paar von Kantennachbarn von C_1, so dass `a` und `b` selbst Kantennachbarn sind. Daraus werden Typ 2 und Typ 4 wie folgt aufgebaut:

Typ 2:
- Zelle `c` ist Kantennachbar von Zelle C_1 und von Zelle `a`,
- Zelle `d` ist Kantennachbar von Zelle C_1 und von Zelle `b`,
- Zelle `e` ist Kantennachbar von Zelle `a` und von Zelle `b`.

Typ 4:
- Zelle `c` ist Kantennachbar von Zelle `a` und von Zelle `b`,
- Zelle `d` ist Kantennachbar von Zelle `a` und von Zelle `c`,
- Zelle `e` ist Kantennachbar von Zelle `b` und von Zelle `c`.

Nutzt man die primären Dreiecke einer baryzentrischen Unterteilung aus, so kann man diese Schablonen wie folgt erhalten (siehe auch Abbildung 2.5):

Typ 2: Zu jedem Dreieck D_1, von dem ein Eckpunkt der Punkt P_1 ist, der der Zelle C_1 zugeordnet ist, nehme man die drei Nachbardreiecke hinzu und bilde eine Schablone aus den Zellen, die den Eckpunkten dieser vier Dreiecke zugeordnet sind.

Typ 4: Zu jedem Dreieck D_1, von dem ein Eckpunkt der Punkt P_1 ist, der der Zelle C_1 zugeordnet ist, nehme man das Nachbardreieck D_2 hinzu, das dem Punkt P_1 gegenüber liegt. Dazu addiere man die zwei von D_1 verschiedenen Nachbarn von D_2 und bilde eine Schablone aus den Zellen, die den Eckpunkten dieser insgesamt vier Dreiecke zugeordnet sind.

Triangulierungen Für den Fall von Triangulierungen verwenden wir wiederum die Schablonen, die für den Polynomgrad $Q = 2$ in [HS99] angegeben sind. Die bis zu sechs Schablonen sind gleichartig, es gibt keine Abstufung von verschieden stark zentral ausgerichteten Schablonen: $\{T_1, T_i, T_{i_a}, T_{i_b}, T_k, T_{k_b}\}$, $\{T_1, T_i, T_{i_a}, T_{i_b}, T_j, T_{j_a}\}$, $\{T_1, T_j, T_{j_a}, T_{j_b}, T_i, T_{i_b}\}$, $\{T_1, T_j, T_{j_a}, T_{j_b}, T_k, T_{k_a}\}$, $\{T_1, T_k, T_{k_a}, T_{k_b}, T_j, T_{j_b}\}$ und $\{T_1, T_k, T_{k_a}, T_{k_b}, T_i, T_{i_a}\}$ (siehe wiederum Abbildung 2.4).

2.2.4 Oszillationsindikatoren für Polynome

Für ein gegebenes Polynom p vom Grad Q für Zelle C_1 wird ein geeigneter Oszillationsindikator $OI(p)$ gesucht.
In [Abg94b] wird nachgewiesen, dass, wie im eindimensionalen Fall, die Betragssumme der führenden Polynomkoeffizienten

$$OI_A(p) := \sum_{|\alpha|=Q} |a_\alpha| \tag{2.19}$$

die Eigenschaft hat gegen unendlich zu gehen, wenn die Feinheit $h(C_1)$ gegen null geht und eine k-te Ableitung mit $k < Q$ von u in der jeweiligen Schablone unstetig ist. Sind hingegen alle k-ten Ableitungen mit $k < Q$ stetig, so bleibt $OI_A(p)$ beschränkt. Mit dieser Begründung wird OI_A in [Abg94b] als Oszillationsindikator verwendet.
Leider stellte sich bei unseren Testrechnungen heraus, dass für $Q = 2$ dieser Indikator zu teilweise stark oszillierenden Lösungen führt. Als mögliche Begründung kann Satz 2.1.11 herangezogen werden. Dort haben wir nachgewiesen, dass insbesondere die führenden Koeffizienten von p sehr empfindlich von den Daten $\overline{u}_\ell$ abhängen. Daher ist es nicht überraschend, wenn der Indikator OI_A zu empfindlich auf Störungen in den Daten reagiert und die Gewichtung oder Auswahl der Polynome p_i dadurch völlig chaotisch wird.
Versteht man unter der Oszillation von p die Totalvariation von p auf C_1 so kommt man zu dem Indikator

$$OI_{L_1}(p) := \|\nabla p\|_{L_1(C_1)}. \tag{2.20}$$

Da die Normen in Funktionenräumen üblicherweise nur für skalare Funktionen definiert werden, müssen wir klarstellen, was wir unter der L_1-Norm des Gradienten verstehen wollen. Zum einen kann man darunter die L_1-Norm einer Vektornorm von ∇p verstehen, zum anderen kann man umgekehrt darunter verstehen die L_1-Norm jeder Komponente von ∇p zu berechnen und dann eine Vektornorm des resultierenden Normen-Vektors zu nehmen. Nimmt man als Vektornorm in beiden Fällen die Betragssummennorm, so erhält man in beiden Fällen dieselbe Norm:

$$\begin{aligned}
\int_{C_1} \|\nabla p(x)\|_1 dx &= \int_{C_1} \left|(\nabla p(x))_1\right| + \left|(\nabla p(x))_2\right| dx \\
&= \int_{C_1} \left|(\nabla p(x))_1\right| dx + \int_{C_1} \left|(\nabla p(x))_2\right| dx \\
&= \left\| \left(\left\|(\nabla p(x))_1\right\|_{L_1(C_1)}, \left\|(\nabla p(x))_2\right\|_{L_1(C_1)} \right)^T \right\|_1.
\end{aligned}$$

Da alle Vektornormen des $\mathbb{R}^d$ äquivalent sind, sind alle Normen, die man durch die oben beschriebenen Techniken erhält, äquivalent. Wir haben für unsere Rechnungen

$$\|\nabla p\|_{L_1(C_1)} := \sqrt{\|(\nabla p)_1\|^2_{L_1(C_1)} + \|(\nabla p)_2\|^2_{L_1(C_1)}}$$

gesetzt.
Leider ist die Berechnung der L_1-Norm auf unseren Zellen sehr aufwendig, da beim Integrieren die Zellen an der Linie $\nabla p \equiv 0$ geteilt werden müssen.

Wegen des Schnitts, der von p abhängig ist, müssen die Integrale jedesmal neu berechnet werden. Deshalb haben wir den Indikator

$$OI_{L_2}(p) := \|\nabla p\|_{L_2(C_1)} := \left(\int_{C_1} (\nabla p)_1^2 + (\nabla p)_2^2 \, dx \right)^{\frac{1}{2}} \tag{2.21}$$

definiert. Dieser Indikator lässt sich zumindest für die Polynomgrade $Q = 1$ und $Q = 2$ mit relativ geringem Aufwand berechnen. Für $Q = 1$ ist $\nabla p = (a_{(1,0)}, a_{(0,1)})$, also

$$OI_{L_2}(p) = \left(|C_1| (a_{(1,0)}^2 + a_{(0,1)}^2) \right)^{\frac{1}{2}}.$$

Für $Q = 2$ ist $(\nabla p)_1 = a_{(1,0)} + 2a_{(2,0)}(x - \overline{x}_{C_1})_1 + a_{(1,1)}(x - \overline{x}_{C_1})_2$ und $(\nabla p)_2 = a_{(0,1)} + a_{(1,1)}(x - \overline{x}_{C_1})_1 + 2a_{(0,2)}(x - \overline{x}_{C_1})_2$, also

$$\begin{aligned} OI_{L_2}(p) = \Big(|C_1| \big(& a_{(1,0)}^2 + a_{(0,1)}^2 + (a_{(1,1)}^2 + 4a_{(2,0)}^2) \mathcal{A}(C_1) X_{C_1}^{(2,0)} \\ & + (a_{(1,1)}^2 + 4a_{(0,2)}^2) \mathcal{A}(C_1) X_{C_1}^{(0,2)} \\ & + 4a_{(1,1)}(a_{(2,0)} + a_{(0,2)}) \mathcal{A}(C_1) X_{C_1}^{(1,1)} \big) \Big)^{\frac{1}{2}}. \end{aligned}$$

Da wir die benötigten $\mathcal{A}(C_1) X_{C_1}^{\alpha}$ als Gitterdaten speichern, ist der Rechenaufwand gering.
Bei keiner der durchgeführten Vergleichsrechnungen waren deutliche Unterschiede zwischen den Ergebnissen mit OI_{L_1} und OI_{L_2} zu erkennen.
Ein weiterer Oszillationsindikator aus [Fri98] ist die Übertragung eines Vorschlags aus [JS96] auf den zweidimensionalen Fall:

$$OI_{JS}(p) := \left(\sum_{1 \le |\alpha| \le Q} \int_{C_1} h^{2|\alpha|-4}(C_1) \big(D^{\alpha} p(x) \big)^2 \, dx \right)^{\frac{1}{2}}. \tag{2.22}$$

Mit diesem Oszillationsindikator erhält man sehr ähnliche Ergebnisse wie mit OI_{L_2}. In [Fri98] haben wir noch OI_{JS} bevorzugt, da wir für den Ringleb-Testfall (siehe Abschnitt 4.2) mit OI_{JS} etwas geringere Fehler erhielten als mit OI_{L_2}. Bei Testfällen mit starken Unstetigkeiten in den Daten zeigt sich allerdings, dass die Ergebnisse mit OI_{JS} stärker zu Irregularitäten neigen als mit OI_{L_2}. Das kann man bereits beim Stoßrohrproblem erkennen (siehe Abschnitt 4.1). Deshalb haben wir bei allen Rechnungen für diese Arbeit außer den Vergleichstests immer mit dem Oszillationsindikator OI_{L_2} gerechnet.

2.2.5 Berechnung der Gewichte

Wir haben oben schon angedeutet, in welcher Weise die Gewichte w_i aus den $OI(p_i)$ berechnet werden sollen:

Digitales ENO-Verfahren

Bei den klassischen ENO-Verfahren wird aus den berechneten p_i dasjenige $p := p_{i^\star}$ ausgewählt, das die geringste Oszillation hat. Aufgrund dieser digitalen Auswahl sprechen wir auch vom *digitalen ENO-Verfahren*. Die Auswahl erreicht man durch die Gewichte

$$w_i := \begin{cases} 1, & \text{falls } OI(p_i) = \min\limits_{j=1,\dots,m} OI(p_j), \\ 0, & \text{sonst.} \end{cases}$$

Dabei wird stillschweigend vorausgesetzt, dass das Minimum nur für ein i angenommen wird.
Man kann sich an einem eindimensionalen Beispiel leicht überlegen, dass die Berechnung von p durch ein digitales ENO-Verfahren kein sachgemäß gestelltes Problem ist, das heißt p hängt nicht stetig von den Daten ab:

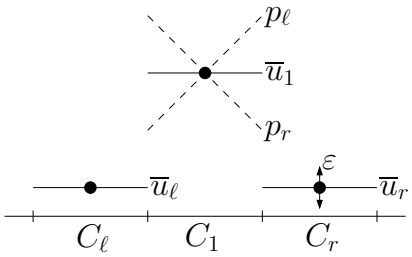

Abbildung 2.6: Beispielproblem für das digitale ENO-Verfahren.

Wir nehmen die Situation aus Abbildung 2.6 an. Zu Zelle C_1 sind die zwei Schablonen $\mathcal{S}_\ell := \{C_1, C_\ell\}$ und $\mathcal{S}_r := \{C_1, C_r\}$ gegeben sowie die Mittelwerte $\overline{u}_1 \neq \overline{u}_\ell = \overline{u}_r$. Als Rekonstruktionspolynome ($Q = 1$) ergeben sich p_ℓ und p_r, $p_\ell \neq p_r$. Nimmt man als Oszillationsindikator den Betrag der Steigung, so ergibt sich keine eindeutige Auswahl von p_ℓ oder p_r, da die Beträge der Steigungen übereinstimmen. Stört man den Wert $\overline{u}_r$ um ein beliebig kleines ε, so springt die Auswahl je nach Vorzeichen von ε zwischen p_ℓ und p_r hin und her. In zwei Raumdimensionen und für höhere Polynomgrade sind weit mehr Situationen möglich, in denen die Oszillationsindikatoren von verschiedenen Polynomen übereinstimmen.
Das Problem lässt sich dadurch umgehen, dass man auf die digitale Auswahl eines Polynoms verzichtet und stattdessen eine Konstruktion vornimmt, die p in stetiger Abhängigkeit von den Daten berechnet.

Gewichtetes ENO-Verfahren

Für $\beta > 0$ und $\varepsilon > 0$ definieren wir

$$w_i := \frac{w_t(\mathcal{S}_i) \cdot (\varepsilon + OI(p_i))^{-\beta}}{\sum_{j=1}^{m} w_t(\mathcal{S}_i) \cdot (\varepsilon + OI(p_j))^{-\beta}}. \tag{2.23}$$

Der einzige Sinn von ε ist, in obigem Ausdruck die Division durch null zu verhindern. Wir haben ε auf die Rechnergenauigkeit gesetzt ($\approx 10^{-16}$). Durch β wird gesteuert, wie empfindlich die Gewichte von den Oszillationen abhängen. Man überlegt sich leicht, dass für $\beta \to 0$ die Abhängigkeit der w_i von den Oszillationen abgeschaltet wird, also alle p_i gleichgewichtet werden. Für $\beta \to \infty$ geht das gewichtete ENO-Verfahren formal in das digitale ENO-Verfahren über.
In [LOC94] wird eine *ENO Eigenschaft* definiert. Diese erfordert $\beta \geq m$ zu setzen. Uns scheint diese Begriffsbildung sehr willkürlich zu sein, weshalb wir, wie auch Jiang und Shu in [JS96], auf diese Eigenschaft verzichten. Verschiedene Testrechnungen haben gezeigt, dass zumindest für $Q \leq 2$ die Wahl $\beta = 4$ meistens ausreichend ist. Um sicherzugehen haben wir $\beta := 8$ gesetzt.
Die $w_t(\mathcal{S}_i)$ in (2.23) sind unabhängig von der Oszillation der p_i. Sie können dazu verwendet werden unterschiedliche Typen von Schablonen unterschiedlich stark zu gewichten. Bei Gittern vom Box-Typ gewichten wir damit bei Polynomgrad $Q = 2$ die zentral ausgerichteten Schablonen vom Typ 2 zwölfmal stärker als die stark einseitig ausgerichteten Schablonen vom Typ 4. Dies führt zu einer höheren Regularität der Lösungen als die Gleichgewichtung. Die Qualität der Lösungen ist für Übergewichtungen zwischen fünf und zwanzig sehr ähnlich. Die Wahl des Faktors zwölf in diesem Bereich ist willkürlich.
In Abbildung 2.7 sind die Näherungslösungen mit dem gewichteten ENO-Verfahren für unser Modellbeispiel dargestellt. Im Vergleich mit Abbildung 2.2 sieht man, dass die mit der *naiven* polynomialen Rekonstruktion aufgetretenen Oszillationen jetzt verschwunden sind.

2.3 Übertragung auf die Euler-Gleichungen

In den bisherigen Abschnitten dieses Kapitels haben wir die Rekonstruktion von skalarwertigen Funktionen aus Zellmittelwerten beschrieben. Bei Systemen von Erhaltungsgleichungen, insbesondere bei den Euler-Gleichungen, ist u allerdings vektorwertig.

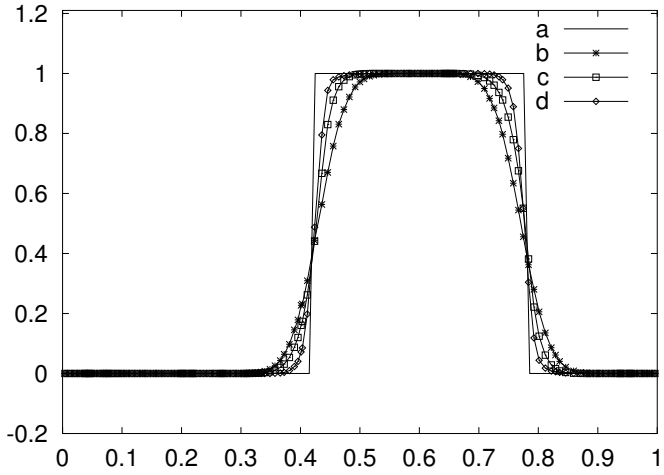

Abbildung 2.7: a) Exakte Lösung, b) Näherungslösung aus der Basisdiskretisierung, c) mit WENO-Rekonstruktion ($Q = 1$) und d) mit WENO-Rekonstruktion ($Q = 2$) zur Zeit $T = 0.3$.

2.3.1 Rekonstruktion in Zustandsvariablen

Wie bereits angedeutet, lösen wir die Rekonstruktion von vektorwertigen Funktionen, indem wir die oben beschriebenen Techniken zur Rekonstruktion skalarwertiger Funktionen aus Zellmittelwerten auf die einzelnen Komponenten von u anwenden.
Es ist in den Anwendungen der Euler-Gleichungen allerdings üblich, die Rekonstruktion nicht in den Komponenten der Zustandsvariablen vorzunehmen sondern aus diesen zunächst einen anderen Variablensatz zu bestimmen und die Komponenten dieses Satzes zu rekonstruieren.

2.3.2 Rekonstruktion in primitiven Variablen

Die Größen ρ, v_1, v_2 und p werden auch die *primitiven Variablen* genannt. Wir fassen diese Variablen in dem Vektor

$$\sigma := (\rho, \mathsf{v}_1, \mathsf{v}_2, \mathsf{p})^T$$

zusammen. Die Komponenten von u sind

$$u := (\rho, \rho\mathsf{v}_1, \rho\mathsf{v}_2, \rho\mathsf{E})^T$$

und der Druck ist durch die Zustandsgleichung für idealas Gas,

$$\mathsf{p} = (\gamma - 1)\rho\big(\mathsf{E} - \frac{|\mathsf{v}|^2}{2}\big),$$

gegeben (siehe S. 15). Da die Dichte immer positiv und insbesondere von null verschieden ist, kann zwischen den beiden Variablensätzen bijektiv tranformiert werden:

$$\sigma = \begin{pmatrix} u_1 \\ u_2/u_1 \\ u_3/u_1 \\ (\gamma-1)(u_4 - (u_2^2+u_3^2)/(2u_1)) \end{pmatrix} \tag{2.24}$$

und

$$u = \begin{pmatrix} \sigma_1 \\ \sigma_1\sigma_2 \\ \sigma_1\sigma_3 \\ \sigma_4/(\gamma-1) + \frac{1}{2}\sigma_1(\sigma_2^2+\sigma_3^2) \end{pmatrix}. \tag{2.25}$$

Für die Rekonstruktion werden nun aus den Näherungswerten für die Mittelwerte von u mittels Gleichung (2.24) Näherungswerte für die Mittelwerte von σ berechnet. Daraus werden komponentenweise Rekonstruktionspolynome berechnet, deren Werte an den Gaußknoten mittels Gleichung (2.25) wieder in Erhaltungsvariable umgerechnet werden können.
Aufgrund der Nichtlinearität der Transformationen ergibt sich dabei aber ein Konsistenzproblem, da die Transformationen nicht mit der Mittelwertbildung vertauschbar sind.

Satz 2.3.1. *Seien ρ, v_1, v_2 zweimal stetig differenzierbar auf Ω. Alle drei Größen seien beschränkt. Ferner gelte für die Dichte $\rho \geq K > 0$ mit einer globalen Konstante K. Dann gilt für jede Zelle C eines Gitters $\mathcal{G}$ von Ω*

$$\frac{\mathcal{A}(C)(\rho\mathsf{v}_j)}{\mathcal{A}(C)\rho} = \mathcal{A}(C)\mathsf{v}_j + \mathcal{O}(h^2(C)), \quad j = 1,2 \quad \textit{und}$$

$$(\gamma-1)\Big(\mathcal{A}(C)(\rho\mathsf{E}) - \frac{\sum_{j=1,2}(\mathcal{A}(C)(\rho\mathsf{v}_j))^2}{2\mathcal{A}(C)\rho}\Big) = \mathcal{A}(C)\mathsf{p} + \mathcal{O}(h^2(C)).$$

Zumindest die erste Abschätzung ist scharf.

Zum Beweis benötigen wir das folgende Lemma, das in [Son97b] bewiesen wird.

Lemma 2.3.2. *Für den Zellmittelungsoperator gilt im Schwerpunkt eine besondere Approximationseigenschaft: Ist C eine Zelle eines Gitters und $f : C \to \mathbb{R}$ zweimal stetig differenzierbar, so gilt*

$$\mathcal{A}(C)f = f(\overline{x}_C) + \mathcal{O}(h^2(C)).$$

Beweis des Satzes. O.B.d.A. sei $\overline{x}_C = 0$. Für die erste Abschätzung definieren wir die Funktionen

$$\begin{aligned}\tilde{\rho}(x) &:= \rho(x) - \mathcal{A}(C)\rho, \quad \text{und}\\ \tilde{\mathsf{v}}_j(x) &:= \mathsf{v}_j(x) - \mathcal{A}(C)\mathsf{v}_j.\end{aligned}$$

Per definitionem ist $\int_C \tilde{\rho}(x)dx = 0$ und $\int_C \tilde{\mathsf{v}}_j(x)dx = 0$. Damit gilt:

$$\begin{aligned}\frac{\mathcal{A}(C)(\rho\mathsf{v}_j)}{\mathcal{A}(C)\rho} &= \frac{\frac{1}{|C|}\int_C \big(\mathcal{A}(C)\rho + \tilde{\rho}(x)\big)\big(\mathcal{A}(C)\mathsf{v}_j + \tilde{\mathsf{v}}_j(x)\big)\,dx}{\mathcal{A}(C)\rho}\\ &= \mathcal{A}(C)\mathsf{v}_j + \frac{\mathcal{A}(C)(\tilde{\rho}\tilde{\mathsf{v}}_j)}{\mathcal{A}(C)\rho}.\end{aligned}$$

Nach Voraussetzung lässt sich $\mathcal{A}(C)\rho$ nach unten durch K abschätzen. Wir müssen also jetzt noch $\mathcal{A}(C)(\tilde{\rho}\tilde{\mathsf{v}}_j)$ abschätzen. Dazu betrachten wir die Taylor-Entwicklung von $\tilde{\rho}$ und von $\tilde{\mathsf{v}}_j$ um 0. Für $x \in C$ ist

$$\begin{aligned}\tilde{\rho}(x) &= \big(\rho(0) - \mathcal{A}(C)\rho\big) + \sum_{|\alpha|=1} D^\alpha\rho(0)\cdot x^\alpha + \mathcal{O}(h^2(C)) \quad \text{und}\\ \tilde{\mathsf{v}}_1(x) &= \big(\mathsf{v}_1(0) - \mathcal{A}(C)\mathsf{v}_1\big) + \sum_{|\alpha|=1} D^\alpha\mathsf{v}_1(0)\cdot x^\alpha + \mathcal{O}(h^2(C)).\end{aligned}$$

Die geklammerten Differenzen können mit Hilfe des Lemmas jeweils durch $\mathcal{O}(h^2(C))$ abgeschätzt werden. Multipliziert man aus und beachtet dabei, dass die Mittelwerte der x^α für $|\alpha| = 1$ null sind, so erhält man

$$\begin{aligned}\mathcal{A}(C)(\tilde{\rho}\tilde{\mathsf{v}}_j) &= \quad \partial_1\rho(0)\cdot\partial_1\mathsf{v}_1(0)\cdot\mathcal{A}(C)\big(X_C^{(2,0)}\big)\\ &\quad +\big(\partial_1\rho(0)\cdot\partial_2\mathsf{v}_1(0) + \partial_2\rho(0)\cdot\partial_1\mathsf{v}_1(0)\big)\mathcal{A}(C)\big(X_C^{(1,1)}\big)\\ &\quad +\partial_2\rho(0)\cdot\partial_2\mathsf{v}_1(0)\cdot\mathcal{A}(C)\big(X_C^{(0,2)}\big) + \mathcal{O}(h^4(C)).\end{aligned}$$

Da für $|\alpha| = 2$

$$\mathcal{A}(C)\big(X_C^\alpha\big) = \mathcal{O}(h^2(C))$$

ist, erhält man insgesamt die erste Abschätzung. Wenn man davon ausgeht, dass eines der Produkte $\partial_k\rho(0)\cdot\partial_\ell\mathsf{v}_1(0)$ nicht verschwindet, so sieht man, dass die Abschätzung scharf ist. Bei der zweiten Abschätzung müssen wir nur die nichtlinearen Terme betrachten. Diese können wir durch wiederholtes Anwenden des Lemmas nachweisen:

$$\begin{aligned}\frac{\big(\mathcal{A}(C)(\rho\mathsf{v}_j)\big)^2}{\mathcal{A}(C)\rho} &= \frac{\big(\rho(0)\mathsf{v}_j(0) + \mathcal{O}(h^2(C))\big)^2}{\rho(0) + \mathcal{O}(h^2(C))}\\ &= \frac{\rho^2(0)\mathsf{v}_j^2(0)}{\rho(0)} + \mathcal{O}(h^2(C)) = \mathcal{A}(C)(\rho\mathsf{v}_j^2) + \mathcal{O}(h^2(C)).\end{aligned}$$

Dabei wurde wiederum die Beschränktheit von ρ und v_j sowie $\rho \geq K$ ausgenutzt. □

Aufgrund dieses Satzes sollte man für Polynomgrade $Q \geq 2$ nicht in primitiven Variablen rekonstruieren, da die Mittelwerte der primitiven Variablen nicht mit der erforderlichen Genauigkeit bestimmt werden können. In allen von uns durchgeführten Vergleichsrechnungen zeigte sich für $Q = 2$ aber kein Ordnungs-Einbruch. Es ist ungeklärt, woran dies liegt. Die Ergebnisse mit Rekonstruktion in primitiven Variablen sind sogar meistens besser als die mit Rekonstruktion in den Zustandsvariablen. Trotz des Satzes bevorzugen wir deshalb auch für $Q = 2$ die primitiven Variablen vor den Zustandsvariablen.

2.3.3 Rekonstruktion in entkoppelten Variablen

Zur Motivation einer völlig anderen Wahl des Koordinatensystems, die in [HS99] beschrieben wird, müssen wir uns anschauen, wozu wir die Rekonstruktionspolynome in unserem Finite-Volumen-Verfahren einsetzen: Wir werten sie an den Gaußknoten der Zellränder aus um mit Hilfe einer numerischen Flussfunktion $f(u) \cdot n$ zu approximieren.
Für die Flussfunktionen der Euler-Gleichungen gilt eine besondere Eigenschaft:

Lemma 2.3.3. *Die Flussfunktionen f_1 und f_2 der Euler-Gleichungen sind homogen vom Grad eins, das heißt, es gilt*

$$f_i(u) = \nabla_u f_i(u) u, \quad i = 1, 2.$$

Der Beweis geht durch Nachrechnen.
Wegen der Homogenität der Flussfunktionen ist also $f(u) \cdot n = A(u, n)u$ mit der Matrix $A(u, n) = \nabla_u f(u) \cdot n$. $A(u, n)$ lässt sich durch $\Lambda(u, n) = P^{-1}(u, n)A(u, n)P(u, n)$ diagonalisieren (siehe Abschnitt 1.1, S. 13ff.). Es ist also

$$A(u, n) = P(u, n)\Lambda(u, n)P^{-1}(u, n)$$

und daher

$$f(u) \cdot n = P(u, n)\Lambda(u, n)P^{-1}(u, n)u.$$

Dies führt zu dem Ansatz für jede Kante einer Zelle eine mittlere Jacobi-Matrix $\tilde{A}(u, n)$ zu bestimmen, für diese Matrix die Diagonalisierungsmatrizen

$\tilde{P}$ und $\tilde{P}^{-1}$ zu berechnen und diese für eine lineare Koordinatentransformation im Zustandsraum zu verwenden. Wir berechnen also aus den vorhandenen Daten $\overline{u}_i$ die Daten

$$\overline{\chi}_i := \tilde{P}^{-1}\overline{u}_i, \tag{2.26}$$

rekonstruieren diese Daten komponentenweise und rechnen dann bei der Auswertung durch Linksmultiplikation mit $\tilde{P}$ zurück. Da der Zellmittelungsoperator linear ist, entsteht durch diese lineare Koordinatentransformation kein Konsistenzfehler, es ist $\mathcal{A}(C)(\tilde{P}^{-1}u) = \tilde{P}^{-1}\mathcal{A}(C)u$.
Wir verwenden für eine Kante im Inneren von Ω

$$\tilde{A}(u,n) := A\big(\tfrac{1}{2}(\overline{u}_\ell + \overline{u}_r), n\big),$$

wobei $\overline{u}_\ell$ und $\overline{u}_r$ die Mittelwerte auf den angrenzenden Zellen sind und für Randkanten

$$\tilde{A}(u,n) := A(\overline{u}_i, n),$$

wobei $\overline{u}_i$ der Mittelwert auf der angrenzenden Zelle ist.
Die Matrizen $\tilde{P}$ und $\tilde{P}^{-1}$ werden gemäß Bemerkung 1.1.3, S. 15, berechnet. Durch Verwendung dieser Transformation erhält man insbesondere bei Strömungsfällen mit starken Unstetigkeiten in den Daten deutlich bessere Ergebnisse als mit primitiven Variablen oder Zustandsvariablen.
Die Verwendung dieser Transformation hat aber einen großen Nachteil: Die Transformationen sind für jede Kante einer Zelle unterschiedlich. Deswegen kann nicht, wie sonst, ein einziges Rekonstruktionspolynom pro Zelle berechnet werden, sondern pro Kante müssen die Rekonstruktionspolynome für die beiden angrenzenden Zellen berechnet werden. Bei der baryzentrischen Unterteilung einer Triangulierung besteht der Rand der meisten Zellen aus zwölf Kanten. Für diese Zellen muss also zwölf Mal statt sonst nur ein Mal rekonstruiert werden. Da ab dem Polynomgrad $Q = 2$ der Rechenaufwand des Finite-Volumen-Verfahrens vom Aufwand der Rekonstruktionsberechnung dominiert wird, ergibt sich eine drastische Erhöhung des Gesamtaufwands. Verwendet man zur räumlichen Diskretisierung als Gitter eine Triangulierung, so ist das Verhältnis nicht ganz so dramatisch, da hier der Rand jeder Zelle nur aus drei Kanten besteht. Es ergibt sich also nur ein Faktor drei.

2.3.4 Algorithmische Umsetzung

Zur Auswahl der Schablonen ist es erforderlich den in Abschnitt 1.4, S. 30ff., definierten Daten eines Gitters $\mathcal{G}$ noch für jede Zelle die Indizes der Kantennachbarn hinzuzufügen. Diese Daten kann man aus dem Array *innereKanteZelle()()* bestimmen. Ferner werden die Schwerpunkte $\overline{x}_{C_j}$ aller Zellen C_j

von $\mathcal{G}$ sowie für Polynomgrad $Q = 2$ die $\mathcal{A}(C_j)X^{\alpha}_{C_j}$, $|\alpha| = 2$, aller Zellen C_j von $\mathcal{G}$ benötigt.

Prozedur 2.1: WENO-Rekonstruktion()

Eingabe: $\mathcal{G}$: Gitter (plus Kantennachbarn, Schwerpunkte und ($Q = 2$) Mittelwerte der Monome vom Grad zwei),
u: Array von Zellmittelwerten,
i: Index der Zelle, für die ein Rekonstruktionspolynom berechnet werden soll.

Ausgabe: p: Rekonstruktionspolynom für Zelle C_i.

Start

- $\{\mathcal{S}_j, j = 1, \ldots, m\}$:=Schablonen$(\mathcal{G}, i)$
- Für j von 1 bis m:
  - p_j :=RekonstruktionsPolynom$(\mathcal{G}, \mathcal{S}_j, u)$
  - o_j :=OszillationsIndikator(p_j)
- $p := 0$
- Für j von 1 bis m:
  - $p := p$+Gewicht$(j, \{o_j\})\cdot p_j$

Ende.

Die Hilfsprozeduren ergeben sich wie folgt aus dem Text dieses Kapitels:

Schablonen$(\mathcal{G}, i)$ Siehe Abschnitt 2.2.3.

RekonstruktionsPolynom$(\mathcal{G}, \mathcal{S}_j, u)$ Siehe Bemerkung 2.1.8.

OszillationsIndikator(p_j) Siehe Abschnitt 2.2.4.

Gewicht$(j, \{o_j\})$ Siehe Abschnitt 2.2.5.

Kapitel 3

Beschleunigung des Finite-Volumen-Verfahrens

3.1 Beschleunigung durch lokale Gitteradaption

Eine wichtige Technik zur Beschleunigung ist die sogenannte h- bzw. Gitteradaptivität. Dabei wird versucht die Gitterweite h lokal an die Erfordernisse der Lösung angepasst zu wählen. Man erreicht eine Beschleunigung, da insgesamt eine kleinere Zellenzahl erforderlich ist als bei einem global feinen Gitter.
Die Umsetzung dieses Konzepts beinhaltet zwei Teilaufgaben. Zum einen muss möglichst automatisch aus einer Näherungslösung ermittelt werden, wo das Gitter feiner aufgelöst werden muss und in welchen Bereichen möglicherweise eine gröbere als die aktuelle Gitterweite ausreichen würde. Zum anderen muss dann die lokale Verfeinerung oder Vergröberung des Gitters geometrisch umgesetzt werden.

3.1.1 Adaptionsindikator

Gegeben sei ein Gitter $\mathcal{G}$ sowie eine Näherungslösung $\bar{u}(t)$ von Mittelwerten der Zustandsgrößen auf den Gitterzellen zur Zeit t.
Grundsätzlich kann man die in der Literatur beschriebenen Adaptionsindikatoren in zwei Kategorien einteilen:
In der Ingenieursliteratur wird oft versucht Strömungscharakteristika wie Stöße und Kontaktunstetigkeiten zu detektieren und diese dann möglichst fein aufzulösen (siehe z.B. [RBY92]). Dazu werden je nach gesuchten Merkmalen die Gradienten unterschiedlicher Strömungsvariablen gemessen und

die Gitter dann dort verfeinert, wo die Gradienten besonders steil sind.
In der zweiten Kategorie wird stattdessen versucht den Fehler der Lösung zu kontrollieren: Das Gitter soll so an die Lösung angepasst werden, dass mit möglichst geringem Aufwand, also wenig Gitterzellen, einige Iterationsschritte weitergerechnet werden kann. Dabei soll gleichzeitig der Fehler möglichst klein sein. Dazu müsste im Idealfall der Fehler gemessen werden, der in den einzelnen Zellen entsteht, was üblicherweise nicht möglich ist. Es ist jedoch möglich Abschätzungen des Fehlers vorzunehmen und diese als Adaptionsindikatoren einzusetzen (siehe [SS94, SW96]).
Praktisch vergleichbare Ergebnisse erhält man durch eine wesentlich weniger aufwendige Technik, die sich insbesondere für baryzentrische Unterteilungen von Triangulierungen gut einsetzen lässt (siehe [Son93]): Man interpretiert die Lösung als stückweise linear auf den Dreiecken. Diese lineare Lösung setzt man nun auf jedem Dreieck in die Erhaltungsgleichung ein (für die Zeitableitung benötigt man eine zweite Lösung $\bar{u}(t + \Delta t)$). Die Größe des resultierenden Residuums verwendet man dann als Adaptionsindikator.
Mit diesem Indikator kann man sehr gute Ergebnisse erzielen. Es ist allerdings nicht ganz einfach bei einer gegebenen Konfiguration den Schwellwert des Indikators zu wählen, ab dem verfeinert bzw. vergröbert werden soll.
Wir haben den Adaptionsindikator verwendet, der in [Hem99] beschrieben wird. Bei diesem Indikator fällt die Wahl der Schwellwerte deutlich leichter. Auch hierbei beschränken wir uns auf den Fall, dass $\mathcal{G}$ die baryzentrische Unterteilung einer Triangulierung ist: Für die m Komponenten von $\bar{u}(t)$ wird

$$d_k := \max_{i=1,\dots,\#\mathcal{G}} \bar{u}_{i,k} - \min_{i=1,\dots,\#\mathcal{G}} \bar{u}_{i,k}, \quad k = 1, \dots, m, \tag{3.1}$$

berechnet. Für jedes Dreieck T seien T^1, T^2 und T^3 die Indices der drei Eckpunkte von T. Wir nutzen wieder aus, dass jeder Punkt der Triangulierung einer Zelle von $\mathcal{G}$ entspricht und berechnen für das Dreieck den Indikatorwert

$$AI(T) := \max_{k=1,\dots,m} \frac{\max\limits_{j=1,2,3} \bar{u}_{T^j,k} - \min\limits_{j=1,2,3} \bar{u}_{T^j,k}}{d_k + \varepsilon}. \tag{3.2}$$

$\varepsilon := 2 \cdot 10^{-16}$ dient nur dazu eine Division durch null zu vermeiden.
Zusätzlich zu diesem Adaptionsindikator kontrollieren wir beim Adaptieren die Flächeninhalte der Dreiecke. Es wird ein minimaler Flächeninhalt der Dreiecke vorgegeben und sichergestellt, dass unabhängig vom Wert des Adaptionsindikators beim Verfeinern keine Dreiecke mit kleinerem Flächeninhalt erzeugt werden.

3.1.2 Gitterverfeinerung und -vergröberung

Für die Veränderung des Gitters gibt es verschiedene Techniken (siehe z.B. [PVMZ87, Bän91, Hem96]). Wir verwenden die Implementierung der Rot-Grün-Verfeinerung/-Vergröberung von D. Hempel (siehe [Hem96, Hem99]). Bei diesem Verfahren wird jedes zur Verfeinerung markierte Dreieck in vier isotrope Dreiecke unterteilt (man nennt diese Unterteilung auch *Rot-Verfeinerung*). Bei Nachbardreiecken von verfeinerten Dreiecken entstehen dadurch sogenannte hängende Knoten, die in unseren Triangulierungen nicht zulässig sind (siehe Abbildung 3.1 links). Dreiecke mit mehr als einem hängenden Knoten werden nun sukzessive auch *rot* verfeinert. Alle Dreiecke mit genau einem hängenden Knoten werden zum Abschluss *grün* verfeinert. Dazu wird der hängende Knoten mit dem gegenüber liegenden Eckpunkt verbunden (siehe Abbildung 3.1 rechts).

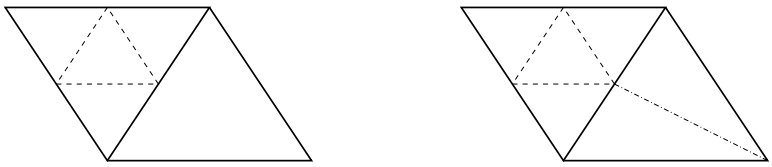

Abbildung 3.1: Rot-Verfeinerung (links) und Grün-Verfeinerung (rechts).

Wird eine bereits so verfeinerte Triangulierung weiter verfeinert, dann werden zunächst alle Grün-Verfeinerungen rückgängig gemacht. Dadurch wird sichergestellt, dass selbst bei beliebig großer Zahl von sukzessiven Verfeinerungen die Winkel der resultierenden Dreiecke nach unten beschränkt bleiben.
Das verwendete Verfahren beinhaltet auch eine Technik zum lokalen Vergröbern von vorher verfeinerten Triangulierungen. Die gröbste dabei zu erreichende Triangulierung ist die Ausgangstriangulierung.
Verbunden mit dem Verfeinerungs- und Vergröberungsalgorithmus ist ein Interpolationsverfahren, durch das die Daten $\bar{u}(t)$ bei Erhaltung des Gesamtintegrals auf das veränderte Gitter übertragen werden. Dabei wird das auch für die Flussberechnungen eingesetzte Rekonstruktionsverfahren eingesetzt um die formale Genauigkeit der Daten zu erhalten.

3.1.3 Rechenaufwand

Der Rechenaufwand für die Durchführung sowohl des Indikators als auch der Verfeinerungen und Vergröberungen zusammen mit der Dateninterpolation ist proportional zur Zellenzahl $\#\mathcal{G}$. Das heißt, der gesamte Aufwand für eine

Gitteradaption ist $K_A \cdot \#\mathcal{G}$. Die Größe von K_A ist etwa vergleichbar mit der Größe von $K_{\tilde{f}}$ (siehe Abschnitt 1.4.3, S. 35ff.).
Will man den Aufwand für eine Berechnung mit Gitteradaption mit dem Aufwand für die Berechnung auf einem global feinen Gitter $\mathcal{G}_{fein}$ vergleichen, so muss man ermitteln, auf welchen Teilmengen des Gebietes Ω eine hohe Auflösung nötig ist und auf welchen nicht. Typisch für die von uns betrachteten hyperbolischen Erhaltungsgleichungen sind Lösungen, die Unstetigkeiten in u oder in einer Ableitung von u auf (eindimensionalen) Kurven haben, die dazwischen aber glatt sind. Im glatten Bereich der Lösung ist eine relativ grobe Auflösung des Gitters ausreichend, so dass die maximale Auflösung nur entlang der genannten Kurven erforderlich ist.
Bezeichnen wir den minimalen Zelldurchmesser mit h_{fein}, so ergeben sich für ein global feines Gitter $\mathcal{O}(h_{fein}^{-2})$ Zellen und für ein lokal adaptiertes Gitter nur $\mathcal{O}(h_{fein}^{-1})$ Zellen.
Je kleiner h_{fein} wird, desto größer wird der Rechenzeitgewinn durch den Einsatz der Gitteradaption.

3.1.4 Speicheraufwand

Der Rechenzeitgewinn wird durch Verwendung von angepassten Gittern und Vermeidung von unnötig feinen Auflösungen in weiten Bereichen des Gebietes erreicht. Man spart im selben Maße Speicherplatz ein, da natürlich auch nur noch ein Gitter mit deutlich weniger Zellen abgespeichert werden muss und die Dimension der Daten $\bar{u}$ entsprechend kleiner ist.

3.2 Beschleunigung durch verallgemeinerte Mehrskalenanalyse

In [Har91] wurde eine grundsätzlich andere Art von Adaptivität vorgestellt. Im Gegensatz zur Gitteradaption, bei der die räumliche Diskretisierung nur dort fein vorgenommen wird, wo dies nötig ist, und sonst möglichst grob, diskretisiert Harten in einer Raumdimension ein Intervall gleichmäßig fein durch ein Gitter $\mathcal{G}^0$. Zusätzlich zu $\mathcal{G}^0$ werden gröbere Gitter $\mathcal{G}^1, \ldots, \mathcal{G}^L$ hinzugenommen. Die Zellen der Gitter sind dabei geschachtelt, das heißt jede Zelle C_ℓ^{k+1} eines groben Gitters $\mathcal{G}^{k+1}$ ist Vereinigung von Zellen des Gitters $\mathcal{G}^k$. Die Menge der Indices von Zellen aus $\mathcal{G}^k$, deren Vereinigung C_ℓ^{k+1} ergibt, wird mit $\mathcal{I}_\ell^k$ bezeichnet:

$$C_\ell^{k+1} = \bigcup_{j \in \mathcal{I}_\ell^k} C_j^k. \tag{3.3}$$

Bei jeder Berechnung der $\tilde{\mathcal{L}}_i(\{\mathcal{A}(C_k)u(\cdot,t)\})$ im Finite-Volumen-Verfahren wird eine Analyse der Daten $\{\mathcal{A}(C_k)u(\cdot,t)\}$ vorgenommen um festzustellen, welches grobe Gitter eine hinreichend genaue Diskretisierung von u ermöglicht. Überall dort, wo ein grobes Gitter ausreicht, wird $\tilde{\mathcal{L}}_i(\{\mathcal{A}(C_k)u(\cdot,t)\})$ nicht auf $\mathcal{G}^0$ berechnet, sondern durch Extrapolation der Berechnung vom groben Gitter bestimmt. Nur in Bereichen, in denen die Diskretisierung auf einem gröberen Gitter nicht hinreichend genau ist, wird $\tilde{\mathcal{L}}$ auf dem feinen Gitter ausgewertet. Da ein gröberes Gitter deutlich weniger Zellen hat als ein feines und die Extrapolation deutlich weniger aufwendig ist als die Berechnung von $\tilde{\mathcal{L}}$, erhält man insgesamt eine Beschleunigung des Finite-Volumen-Verfahrens. Harten motiviert diese Art der Adaptivität dadurch, dass er den relativ großen algorithmischen Aufwand zur Umsetzung der Gitteradaptivität vermeiden will.

3.2.1 Mehrskalendarstellung von diskreten Daten

Wie in [Har96] wollen wir zunächst abstrakt die Mehrskalendarstellung von diskreten Daten einführen. Dazu definieren wir zunächst:

Definition 3.2.1. Seien L sowie $N(0) > N(1) > \ldots > N(L)$ natürliche Zahlen. Für $0 \leq k < L$ seien die Operatoren D_k^{k+1} und P_{k+1}^k mit den folgenden drei Eigenschaften gegeben:[1]

1. $D_k^{k+1} : \mathbb{R}^{N(k)} \to \mathbb{R}^{N(k+1)}$, D_k^{k+1} sei linear,
2. $P_{k+1}^k : \mathbb{R}^{N(k+1)} \to \mathbb{R}^{N(k)}$,
3. $D_k^{k+1} P_{k+1}^k = Id_{k+1}$, Id_{k+1} sei die Identitäts-Abbildung des $\mathbb{R}^{N(k+1)}$.

D_k^{k+1} heißt *Dezimierungsoperator* und P_{k+1}^k heißt *Prolongationsoperator*.

Für Daten $d^0 \in \mathbb{R}^{N(0)}$ werden Daten $d^k \in \mathbb{R}^{N(k)}$, $1 \leq k \leq L$ durch sukzessives Dezimieren berechnet:

$$d^k := D_{k-1}^k d^{k-1}, \quad 1 \leq k \leq L.$$

Nun kann für jedes $0 \leq k < L$ der Operator P_{k+1}^k verwendet werden um ein

$$\tilde{d}^k := P_{k+1}^k d^{k+1}, \quad 0 \leq k < L,$$

zu berechnen. Es ergibt sich dabei der *Prolongationsfehler*

$$e^k := d^k - \tilde{d}^k, \quad 0 \leq k < L.$$

[1] Harten lässt auch unendlich-dimensionale Räume zu. Für die praktische Anwendung zur Beschleunigung von Finite-Volumen-Verfahren ist dies nicht erforderlich. Ferner indizieren wir die Räume in der umgekehrten Reihenfolge von [Har96]. Unsere Reihefolge stimmt mit der aus [Har91] überein.

Lemma 3.2.2. *e^k liegt im Kern von D_k^{k+1}.*

Beweis. $D_k^{k+1}e^k = D_k^{k+1}d^k - D_k^{k+1}P_{k+1}^k d^{k+1} = d^{k+1} - d^{k+1} = 0$ □

Da d^k durch $d^k = P_{k+1}^k d^{k+1} + e^k$ bestimmt werden kann, erhalten wir eine bijektive Zuordnung

$$d^k \leftrightarrow \{d^{k+1}, e^k\}$$

und durch sukzessive Anwendung des Arguments

$$d^0 \leftrightarrow \{d^L, e^{L-1}, \dots, e^0\}.$$

Definition 3.2.3. Die Darstellung

$$MD(d^0) := \{d^L, e^{L-1}, \dots, e^0\}$$

heißt *Mehrskalendarstellung* oder *Mehrskalenzerlegung* von d^0.

Die Mehrskalendarstellung von d^0 kann nun dafür verwendet werden die Darstellbarkeit der Daten d^0 auf gröberen Diskretisierungen zu analysieren. Dazu betrachtet man sukzessive die Komponenten von e^k, $k = 0, \dots, L-1$: Ist ein e_ℓ^k klein, so wird d_ℓ^k bereits gut durch die gröbere Diskretisierung dargestellt, es ist

$$d_\ell^k = (P_{k+1}^k d^{k+1})_\ell + e_\ell^k \approx (P_{k+1}^k d^{k+1})_\ell.$$

3.2.2 Mehrskalendarstellung von Zellmittelwerten

Zur Durchführung einer Mehrskalenzerlegung von Zellmittelwerten auf einem Gitter $\mathcal{G}$ muss eine Folge von Diskretisierungen festgelegt werden. Ferner müssen geeignete Dezimierungs- und Prolongationsoperatoren definiert werden.
Wir gehen davon aus, dass $\mathcal{G} = \mathcal{G}^0$ ein feines Gitter ist. Ferner seien $\mathcal{G}^1, \dots, \mathcal{G}^L$ sukzessive gröbere Gitter, deren Zellen geschachtelt sind. Das heißt, jede Zelle C_i^k eines groben Gitters $\mathcal{G}^k$, $1 \le k \le L$ sei Vereinigung von Zellen C_j^{k-1} des nächst-feineren Gitters $\mathcal{G}^{k-1}$.
Die Daten d^0 seien gegeben als die Zellmittelwerte einer Funktion $u : \mathbb{R}^2 \to \mathbb{R}$:

$$d_\ell^0 := \mathcal{A}(C_\ell^0)u, \quad 1 \le \ell \le \#\mathcal{G}^0.$$

Auf den Gittern $\mathcal{G}^k$, $0 \le k \le L$ seien Rekonstruktionsoperatoren $\mathcal{R}_k$ gegeben, welche aus den Zellmittelwerten auf $\mathcal{G}^k$, den Daten d^k, Funktionen $\mathbb{R}^2 \to \mathbb{R}$ rekonstruieren, deren Zellmittelwerte wiederum die d_k sind, d.h.

$\mathcal{A}(C_\ell^k)(\mathcal{R}_k d^k) = d_\ell^k$. Als $\mathcal{R}_k$ können wir die polynomiale Rekonstruktion aus Kapitel 2 verwenden. Die rekonstruierten Funktionen sind dann zellweise polynomial und die Mittelwerte der Polynome auf den Zellen sind wie gefordert die d_k.
Mit Hilfe der $\mathcal{R}_k$ und des Mittelwertoperators definieren wir für $0 \leq k < L$ die Operatoren

$$D_k^{k+1} \;:\; \mathbb{R}^{\#\mathcal{G}^k} \to \mathbb{R}^{\#\mathcal{G}^{k+1}}, \quad (D_k^{k+1} d^k)_\ell := \mathcal{A}(C_\ell^{k+1})(\mathcal{R}_k d^k), \tag{3.4}$$

$$P_{k+1}^k \;:\; \mathbb{R}^{\#\mathcal{G}^{k+1}} \to \mathbb{R}^{\#\mathcal{G}^k}, \quad (P_{k+1}^k d^{k+1})_j := \mathcal{A}(C_j^k)(\mathcal{R}_{k+1} d^{k+1}). \tag{3.5}$$

Satz 3.2.4. *Die durch* (3.4) *und* (3.5) *definierten Operatoren sind Dezimierungs- und Prolongationsoperatoren gemäß Definition* 3.2.1. *Die* D_k^{k+1} *sind unabhängig von den* $\mathcal{R}_k$ *wohldefiniert. Mit Gleichung* (3.3) *ist*

$$(D_k^{k+1} d^k)_\ell = \frac{1}{|C_\ell^{k+1}|} \sum_{j \in \mathcal{I}_\ell^k} |C_j^k| \cdot d_j^k. \tag{3.6}$$

Beweis. Zunächst können wir uns unmittelbar aus der Definition des Zellmittelungsoperators klarmachen, dass

$$\mathcal{A}(C_\ell^{k+1}) = \frac{1}{|C_\ell^{k+1}|} \sum_{j \in \mathcal{I}_\ell^k} |C_j^k| \cdot \mathcal{A}(C_j^k). \tag{3.7}$$

Es ist

$$\begin{aligned}(D_k^{k+1} d^k)_\ell &\overset{\text{Def.}}{=} \mathcal{A}(C_\ell^{k+1})(\mathcal{R}_k d^k) \overset{(3.7)}{=} \frac{1}{|C_\ell^{k+1}|} \sum_{j \in \mathcal{I}_\ell^k} |C_j^k| \cdot \mathcal{A}(C_j^k)(\mathcal{R}_k d^k) \\ &= \frac{1}{|C_\ell^{k+1}|} \sum_{j \in \mathcal{I}_\ell^k} |C_j^k| \cdot d_j^k.\end{aligned}$$

Damit gilt die behauptete Darstellung (3.6), D_k^{k+1} ist linear und unabhängig von $\mathcal{R}_k$ wohldefiniert. Nun müssen wir noch zeigen, dass $D_k^{k+1} P_{k+1}^k = Id_{k+1}$:

$$\begin{aligned}(D_k^{k+1} P_{k+1}^k d^{k+1})_\ell &\overset{\text{Def.}}{=} \mathcal{A}(C_\ell^{k+1})(\mathcal{R}_k P_{k+1}^k d^{k+1}) \\ &\overset{(3.7)}{=} \frac{1}{|C_\ell^{k+1}|} \sum_{j \in \mathcal{I}_\ell^k} |C_j^k| \cdot \mathcal{A}(C_j^k)(\mathcal{R}_k P_{k+1}^k d^{k+1}) \\ &= \frac{1}{|C_\ell^{k+1}|} \sum_{j \in \mathcal{I}_\ell^k} |C_j^k| \cdot (P_{k+1}^k d^{k+1})_j \\ &= \frac{1}{|C_\ell^{k+1}|} \sum_{j \in \mathcal{I}_\ell^k} |C_j^k| \cdot \mathcal{A}(C_j^k)(\mathcal{R}_{k+1} d^{k+1}) \\ &\overset{(3.7)}{=} \mathcal{A}(C_\ell^{k+1})(\mathcal{R}_{k+1} d^{k+1}) \;=\; d_\ell^{k+1}.\end{aligned}$$

Damit sind die drei Eigenschaften aus Definition 3.2.1 nachgewiesen. □

3.2.3 Konstruktion der groben Gitter

In einer Raumdimension ist die Konstruktion der groben Gitter trivial: Man lässt einfach jeden zweiten Punkt aus einem feinen Gitter weg.
Aus einem feinen unstrukturierten Gitter $\mathcal{G}^k$ in zwei Raumdimensionen das nächstgröbere $\mathcal{G}^{k+1}$ zu erzeugen ist dagegen deutlich aufwendiger. Aus der Bedingung, dass die Zellen geschachtelt sein sollen, ergibt sich bereits das Grundprinzip der Vergröberung: Gruppen von Zellen aus $\mathcal{G}^k$ müssen zu Zellen von $\mathcal{G}^{k+1}$ zusammengefügt werden. Das eigentliche Zusammenfügen ist bei der Verwendung von geeigneten Datenstrukturen eine einfache Aufgabe. Es ist allerdings ein Problem *geeignete* Gruppierungen von Zellen aus $\mathcal{G}^k$ zu bilden. In [AH94] wird ein sehr einfacher Algorithmus angegeben und für Verbesserungen auf die Literatur zu Mehrgitterverfahren [LSD92] verwiesen. Wir haben uns an den Beschreibungen in [VM94] orientiert, wobei wir allerdings nicht zwischen unterschiedlichen Rand-Typen unterscheiden. In einem Punkt haben wir den Algorithmus modifiziert.

Prozedur 3.1: VERGRÖBERUNG

Eingabe: $\mathcal{G}^k$: Gitter (plus Kantennachbarn).

Ausgabe: $\mathcal{G}^{k+1}$: Vergröbertes Gitter.

Start

- $N := 0$
- $\mathit{Front}_{\mathit{Rand}} := \emptyset, \quad \mathit{Front}_{\mathit{Innen}} := \emptyset$
- Solange Zellen aus $\mathcal{G}^k$ noch nicht zu Zellen aus $\mathcal{G}^{k+1}$ zusammengefasst sind:
  - $C^k_\star :=$STARTZELLE$(\mathcal{G}^k,\ \mathit{Front}_{\mathit{Rand}},\ \mathit{Front}_{\mathit{Innen}})$
  - $\{C^k_{j_1}, \ldots, C^k_{j_n}\} :=$HINZUGEFÜGTEKANTENNACHBARN$(\mathcal{G}^k,\ C^k_\star)$
  - $N := N + 1$
  - $C^{k+1}_N := \bigcup\{C^k_\star, C^k_{j_1}, \ldots, C^k_{j_n}\}$
  - entferne (falls enthalten) $C^k_\star, C^k_{j_1}, \ldots, C^k_{j_n}$ aus $\mathit{Front}_{\mathit{Rand}}$ bzw. $\mathit{Front}_{\mathit{Innen}}$
  - füge die Kantennachbarn von $C^k_\star, C^k_{j_1}, \ldots, C^k_{j_n}$, die noch nicht zu Zellen aus $\mathcal{G}^{k+1}$ zusammengefasst sind, in $\mathit{Front}_{\mathit{Rand}}$ bzw. $\mathit{Front}_{\mathit{Innen}}$ ein.

Ende.

Die Hilfsprozedur StartZelle($\mathcal{G}^k$, $Front_{Rand}$, $Front_{Innen}$) wird in [VM94] wie folgt realisiert: Ist in $Front_{Rand}$ eine Zelle enthalten, so wird die erste Zelle aus dieser Menge genommen. Ansonsten wird die erste Zelle aus $Front_{Innen}$ genommen, wenn diese Menge nicht leer ist. Sind beide Mengen leer (beim Start der Vergröberung), dann wird irgendeine Zelle aus $\mathcal{G}^k$ genommen, die am Rand von Ω liegt.
Für HinzugefügteKantenNachbarn($\mathcal{G}^k$, $C_\star^k$) werden dort zwei Varianten angegeben: Bei der *isotropic* Version werden grundsätzlich alle Kantennachbarn von $C_\star^k$, die noch nicht zu Zellen von $\mathcal{G}^{k+1}$ verschmolzen wurden, mit $C_\star^k$ verschmolzen. Bei der *semi-coarsened* Version wird aus der Menge der noch nicht verschmolzenen Kantennachbarn von $C_\star^k$ diejenige Teilmenge ausgewählt, bei der für die resultierende Zelle C_N^{k+1} das Verhältnis $|C_N^{k+1}|/(Umfang(C_N^{k+1}))^2$ maximal ist. Die Teilmenge darf nicht leer sein. Durch Experimente zeigt sich, dass die semi-coarsened Version auch im Fall von sehr regelmäßigen Gittern zu weniger verzerrten groben Zellen führt als die isotropic Version. Außerdem ist die Vergröberungsrate der isotropic Version deutlich größer als die der semi-coarsened Version.
In Abbildung 3.2 sind mit den beiden Versionen vergröberte Gitter dargestellt. $\mathcal{G}^0$ ist die baryzentrische Unterteilung einer Triangulierung aus gleichseitigen Dreiecken, $\#\mathcal{G}^0 = 46\,950$. Links ist das Gitter dargestellt, das man durch fünfmaliges Anwenden der isotropic Version erhält. Die Zellenzahl ist 60. Die semi-coarsened Version muss man siebenmal anwenden um eine ähnliche Zellenzahl (61) zu erhalten. Nach fünf Vergröberungen war hier die Zellenzahl noch 383. Das Gitter nach sieben Anwendungen ist in der Mitte dargestellt. Rechts sieht man das Gitter, das man durch siebenmalige Anwendung einer modifizierten semi-coarsened Version erhält. Hier ist die Zellenzahl 43, nach fünf Vergröberungen war die Zellenzahl noch 305.
Die Modifikation wird in der Prozedur StartZelle($\mathcal{G}^k$, $Front_{Rand}$, $Front_{Innen}$) durchgeführt. Wir nehmen als Startzelle immer die schlechteste Zelle aus den angegebenen Mengen $Front_{Rand}$ und $Front_{Innen}$, wobei die Qualität wie schon oben durch das Verhältnis $|C_\star^k|/(Umfang(C_\star^k))^2$ definiert wird. Durch diese Modifikation werden die Verzerrungen der groben Zellen nochmals kleiner. Außerdem sinkt die Abhängigkeit der Vergröberung von der Sortierung der Zellen in $\mathcal{G}^0$.
Wie auch in [AH94] wird bei uns sichergestellt, dass keine trivialen Vergröberungen vorgenommen werden. Eine Zelle aus $\mathcal{G}^{k+1}$ ist immer Vereinigung von mindestens zwei Zellen aus $\mathcal{G}^k$.
Die Vergröberungsrate variiert von Gitter zu Gitter und ist, wie obiges Beispiel zeigt, auch von der Variante des Vergröberungsalgorithmus abhängig. Wir haben bei allen Rechnungen den modifizierten *semi-coarsened* Algorith-

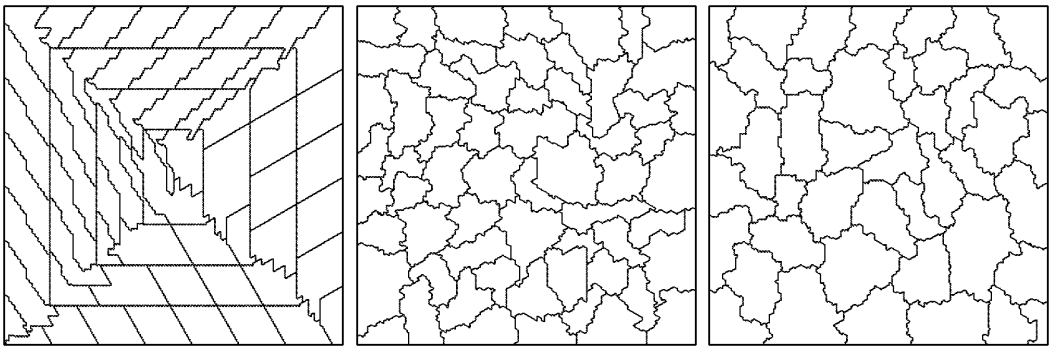

Abbildung 3.2: Vergröberte Gitter konstruiert mit dem isotropic (fünfmal vergröbert) und dem semi-coarsened Algorithmus aus [VM94] (siebenmal vergröbert) sowie mit dem modifizierten semi-coarsened Algorithmus (siebenmal vergröbert).

mus verwendet. Als Gitterdimensionen erhalten wir damit üblicherweise

$$\begin{aligned} \#\mathcal{G}^k &\approx \left(\frac{1}{3}\right)^k \cdot \#\mathcal{G}^0, \\ \text{Anzahl innere Kanten } \mathcal{G}^k &\approx \left(\frac{2}{3}\right)^k \cdot \text{ Anzahl innere Kanten } \mathcal{G}^0, \\ \text{Anzahl Randkanten } \mathcal{G}^k &= \text{ Anzahl Randkanten } \mathcal{G}^0. \end{aligned}$$

Aus den unterschiedlichen Raten bei Zellenzahl und Kantenzahl kann man erkennen, dass die Komplexität der Zellränder beim Vergröbern zunimmt: Die Zahl der Kanten je Zelle verdoppelt sich ungefähr bei jedem Gitterlevel.

3.2.4 Algorithmische Beschreibung

Beim Finite-Volumen-Verfahren soll nun der Teil beschleunigt werden, der den wesentlichen Aufwand bedeutet, nämlich die Berechnung der $\tilde{\mathcal{L}}_i$ (siehe Prozedur 1.4, S. 33). Dazu müssen neben dem feinen Gitter $\mathcal{G}^0$ auch grobe Gitter $\mathcal{G}^1, \ldots, \mathcal{G}^L$ bereitgestellt werden.
Die folgende Prozedur ersetzt die Prozedur 1.4 für alle Gitter $\mathcal{G}^k$, für die ein gröberes Gitter $\mathcal{G}^{k+1}$ vorhanden ist. Einige Teile ergeben sich unmittelbar aus den entsprechenden Teilen der Prozedur 1.4. Sie werden hier nicht nochmals detailiert ausgeführt. Das Vorgehen für eine ganze Sequenz von Gittern $\mathcal{G}^0, \ldots, \mathcal{G}^L$ ergibt sich unmittelbar aus der Prozedur.

Prozedur 3.2: MittlereFlussdivergenz-MS$(\mathcal{G}^k, \mathcal{G}^{k+1}, u^k, \varepsilon_k, \overline{\nabla \cdot f}^k)$

Eingabe: $\mathcal{G}^k, \mathcal{G}^{k+1}$: Feines und grobes Gitter,
u^k: Array von Zellmittelwerten der Erhaltungsgrößen auf $\mathcal{G}^k$,
ε_k: Toleranzwert.

Ausgabe: $\overline{\nabla \cdot f}^k$: Array der $-\tilde{\mathcal{L}}_i(\{u_j^k\})$.

Start

- $u^{k+1} := D_k^{k+1} u^k$ (Dezimierung)
- $e^k := u^k - P_{k+1}^k u^{k+1}$ (Prolongationsfehler)
- Berechne $\overline{\nabla \cdot f}^{k+1}$, die $-\tilde{\mathcal{L}}_i(\{u_j^{k+1}\})$ auf $\mathcal{G}^{k+1}$, durch rekursiven Aufruf dieser Prozedur oder (falls $k+1 = L$) durch Prozedur 1.4.
- Berechne Rekonstruktionsfunktionen ϕ_i^k nur für die Zellen C_i^k von $\mathcal{G}^k$ bei denen $|e_i^k| > \varepsilon_k$ oder $|e_j^k| > \varepsilon_k$ für einen Kantennachbarn C_j^k von C_i^k ist.
- Berechne die numerischen Flüsse für alle inneren Kanten von $\mathcal{G}^k$, bei denen $|e_j^k| > \varepsilon_k$ für eine der zwei angrenzenden Zellen C_j^k ist.
- Berechne die Randflüsse für alle Randkanten von $\mathcal{G}^k$, bei denen $|e_j^k| > \varepsilon_k$ für die angrenzende Zelle C_j^k ist.
- Für alle Zellen C_i^k von $\mathcal{G}^k$, für die $|e_j^k| \leq \varepsilon_k$ ist, setze $\overline{\nabla \cdot f}_i^k := (P_{k+1}^k \overline{\nabla \cdot f}^{k+1})_i$.

Ende.

Für die Berechnung der Rekonstruktionsfunktionen ϕ_i^k verwenden wir das WENO-Verfahren aus Abschnitt 2.2.
Für die Berechnung der D_k^{k+1} nutzen wir die Gleichung (3.6), S. 73, aus. In der Anwendung von P_{k+1}^k verbirgt sich eine Rekonstruktion der Daten auf Gitter $\mathcal{G}^{k+1}$. Um den Aufwand klein zu halten nehmen wir für diese Rekonstruktionen nicht die relativ aufwendige WENO-Rekonstruktion sondern die deutlich weniger aufwendige lineare polynomiale Rekonstruktion gemäß Bemerkung 2.1.8, S. 44, mit möglichst kleinen und zentral ausgerichteten Schablonen. Wir nehmen aber denselben Polynomgrad wie bei der WENO-Rekonstruktion um bei der Prolongation dieselbe formale Approximationsordnung wie bei der Berechnung von $\tilde{\mathcal{L}}$ ohne Mehrskalenanalyse zu erhalten.

3.2.5 Fehlerbetrachtung

Durch $\overline{\nabla \cdot f}_i^k := (P_{k+1}^k \overline{\nabla \cdot f}^{k+1})_i$ ergibt sich ein Fehler gegenüber der Berechnung von $\overline{\nabla \cdot f}_i^k$ auf $\mathcal{G}^k$. Um eine Abschätzung dieses Fehlers vornehmen zu können verlangt Harten, dass die Norm von $\mathcal{L}$ bzw. von $\tilde{\mathcal{L}}$ beschränkt ist. Da auch die Norm des zu P_{k+1}^k gehörenden Rekonstruktionsalgorithmus $\mathcal{R}_k$ beschränkt ist, erhält man insgesamt aus $|u_i^k - (P_{k+1}^k u^{k+1})_i| = |e_i^k| \leq \varepsilon_k$ die Abschätzung

$$\left|\overline{\nabla \cdot f}_i^k - (P_{k+1}^k \overline{\nabla \cdot f}^{k+1})_i\right| \leq K \cdot \varepsilon_k$$

(siehe [Har91]).

3.2.6 Rechenaufwand

Um die Effizienz des Konzepts zu untersuchen, müssen wir den Aufwand von Prozedur 3.2 bestimmen und ihn mit dem von Prozedur 1.4 vergleichen. Dazu definieren wir für $0 \leq k < L$

$$\mathcal{G}_{\varepsilon_k}^k := \{C_i^k \in \mathcal{G}^k : e_i^k > \varepsilon_k\}. \tag{3.8}$$

In Prozedur 1.4 teilt sich der Aufwand in den Aufwand für die Rekonstruktion und den Aufwand für die numerischen Flussberechnungen auf. Der Aufwand für die numerischen Flussberechnungen ist dort $K_{\tilde{f}} \# \mathcal{G}^0$. Die Konstante $K_{\tilde{f}}$ beinhaltet dabei die mittlere Anzahl von Kanten der Zellen aus $\mathcal{G}^0$.
Berücksichtigen wir, dass sich diese Anzahl beim Vergröbern pro Gitterlevel etwa verdoppelt, so erhalten wir bei rekursiver Anwendung von Prozedur 3.2 und der Verwendung von Prozedur 1.4 für Gitter $\mathcal{G}^L$ für die numerischen Flussberechnungen den Aufwand

$$K_{\tilde{f}}\Big(2^L \# \mathcal{G}^L + \sum_{k=0}^{L-1} 2^k \# \mathcal{G}_{\varepsilon_k}^k\Big).$$

Bei der WENO-Rekonstruktion ist der Aufwand unabhängig von der Komplexität der Zellen. Wir erhalten also für die Rekonstruktionen jetzt den Aufwand

$$K_{\mathcal{R}}\Big(\# \mathcal{G}^L + \sum_{k=0}^{L-1} \# \mathcal{G}_{\varepsilon_k}^k\Big).$$

Zusätzlich zum Aufwand für die numerischen Flussberechnungen und für die Rekonstruktionen sind hier noch die Dezimierungen D_k^{k+1} sowie die Prolongationen P_{k+1}^k zu berechnen. Für die Dezimierungen erhalten wir den Aufwand

$$K_D \sum_{k=0}^{L-1} \# \mathcal{G}^k$$

und für die Prolongationen

$$K_P\Big(\sum_{k=0}^{L-1}\#\mathcal{G}^k+\sum_{k=0}^{L-1}(\#\mathcal{G}^k-\#\mathcal{G}^k_{\varepsilon_k})\Big).$$

Die Berechnung der D_k^{k+1} mit Gleichung (3.6), S. 73, ist nicht sehr aufwendig. K_D ist also relativ klein. Die lineare polynomiale Rekonstruktion der Daten auf Gitter $\mathcal{G}^{k+1}$, die bei der Anwendung von P_{k+1}^k berechnet werden muss, ist deutlich weniger aufwendig, als die sonst verwendete WENO-Rekonstruktion. K_P ist also deutlich kleiner als $K_\mathcal{R}$ aber (leider) nicht verschwindend klein. Bei unserer Implementierung liegt zwischen den beiden Konstanten ein Faktor von ungefähr 10.
Nutzt man aus, dass $K_\mathcal{R} \gg K_D$ und $K_\mathcal{R} \gg K_P$, so sieht man, dass der Gesamtaufwand umso kleiner ist, je kleiner die $\#\mathcal{G}^k_{\varepsilon_k}$ sind. Man kann den Aufwand also dadurch nach unten abschätzen, dass man alle $\#\mathcal{G}^k_{\varepsilon_k}$ auf null setzt. Dann erhält man

$$\begin{aligned}\text{Aufwand(Prozedur 3.2)} &\geq (K_{\tilde{f}}2^L+K_\mathcal{R})\#\mathcal{G}^L+(K_D+2K_P)\sum_{k=0}^{L-1}\#\mathcal{G}^k\\ &\geq (K_D+2K_P)\#\mathcal{G}^0.\end{aligned}$$

Betrachtet man die Relation zum Aufwand von Prozedur 1.4 ohne Einsatz von Mehrskalendarstellungen, so kommt man zu dem folgenden ernüchternden Ergebnis:

Satz 3.2.5. *Die mögliche Beschleunigung des Finite-Volumen-Verfahrens durch die Verwendung von Mehrskalendarstellungen der Daten ist nach oben beschränkt durch*

$$\frac{K_\mathcal{R}+K_{\tilde{f}}}{K_D+2K_P}.$$

Bei unserer Implementierung ergibt sich damit eine theoretische Maximalbeschleunigung von etwa fünf. Wir sind leider bei keiner Testrechnung in die Nähe des Faktors fünf gekommen.

3.2.7 Speicheraufwand

Die Adaptivität einer Mehrskalenanalyse besteht gerade darin bewusst das Gebiet Ω überall so fein zu diskretisieren, wie es für die Auflösung der größten Variationen der Lösung wie Unstetigkeiten erforderlich ist und über die Mehrskalendarstellung in den hoffentlich großen Teilgebieten von Ω, in denen

diese Auflösung nicht nötig ist die aufwendigen Berechnungen auf gröberen Gittern durchzuführen.
Es fällt also der Speicheraufwand für die Diskretisierung von Ω durch ein größtenteils viel zu feines Gitter $\mathcal{G}^0$ an. Auch die Daten u^0 müssen passend zu diesem feinen Gitter abgespeichert werden.
Bei einem global feinen Gitter $\mathcal{G}^0$ mit Zelldurchmessern h_{fein} ergibt sich eine Zellenzahl von $\mathcal{O}(h_{fein}^{-2})$. Dies ist, verglichen mit den $\mathcal{O}(h_{fein}^{-1})$ Zellen bei der Verwendung der Gitteradaption, ein gewaltiger Speicheraufwand.
Bei heutigen Rechnern ist die Frage des Speicheraufwands für Rechnungen in einer Raumdimension sicherlich nicht mehr relevant. Auch für unsere durchgeführten Rechnungen in zwei Raumdimensionen ist die Frage nach dem Speicheraufwand nur von akademischem Interesse. Will man die Konzepte allerdings für praktische Anwendungen in drei Raumdimensionen umsetzen, so spielt die Frage nach dem Speicheraufwand eine große Rolle. Ein global feines Gitter um ein komplettes Flugzeug, das fein genug ist auch relativ kleine Details, wie die Aufhängungen der Triebwerke, hinreichend gut aufzulösen, wird man selbst auf der nächsten Generation von Supercomputern nicht abspeichern können.

3.2.8 Ein grundsätzliches Problem

Neben der mäßigen Effizienz der beschriebenen Technik ergibt sich noch ein grundsätzliches Problem bei der Verwendung der Mehrskalenanalyse zur Beschleunigung von Finite-Volumen-Verfahren für hyperbolische Erhaltungsgleichungen.
Vor der Berechnung der $\tilde{\mathcal{L}}_i(\{u_j^k(t)\})$ wird der Prolongationsfehler $e^k = u^k(t) - P_{k+1}^k D_k^{k+1} u^k(t)$ bestimmt. Aus e^k wird dann abgelesen, wo sich Variationen in $u^k(t)$ befinden, die auf $\mathcal{G}^k$ aufgelöst werden können, auf $\mathcal{G}^{k+1}$ aber nicht. Ist $|e_j^k| < \varepsilon_k$ in einem Teilgebiet von Ω, so wird in diesem Teilgebiet $\tilde{\mathcal{L}}_i(\{u_j^k(t)\})$ nicht auf $\mathcal{G}^k$ berechnet sondern durch $P_{k+1}^k \tilde{\mathcal{L}}_i(\{u_j^{k+1}(t)\})$ approximiert. Da ein sinnvoller Prolongationsoperator P_{k+1}^k nicht die Variation erhöht und $\tilde{\mathcal{L}}_i(\{u_j^{k+1}(t)\})$ Daten auf $\mathcal{G}^{k+1}$ sind, enthalten die so approximierten $\tilde{\mathcal{L}}_i(\{u_j^k(t)\})$ keine Variationen, die auf $\mathcal{G}^{k+1}$ nicht aufgelöst werden können. Bei der nächsten Analyse sind also in $u^k(t + \Delta t)$ keine zusätzlichen Variationen enthalten, die nicht auf $\mathcal{G}^{k+1}$ aufgelöst werden können. Auf demselben Teilgebiet ist also auch dann $|e_j^k| < \varepsilon_k$.
Das Verfahren ist also nicht in der Lage auf neu entstehende Variationen wie Unstetigkeiten zu reagieren.

Für den Fall eines linearen Prolongationsoperators[2] lässt sich die Aussage wie folgt präzisieren:

Satz 3.2.6. *Der verwendete Prolongationsoperator sei linear. Wir betrachten ein Paar von zwei Gittern $\mathcal{G}^k$ und $\mathcal{G}^{k+1}$. Für ein $t \geq 0$ sei für alle $C_i^k \in \mathcal{G}^k$ der Prolongationsfehler $|e_i^k(t)| < \varepsilon_k$, wobei*

$$e^k(t) := u^k(t) - P_{k+1}^k D_k^{k+1} u^k(t).$$

Entsprechend werden alle $\overline{\nabla \cdot f}_i^k = (P_{k+1}^k \overline{\nabla \cdot f}^{k+1})_i$ durch Berechnung auf $\mathcal{G}^{k+1}$ und Prolongation nach $\mathcal{G}^k$ gemäß Prozedur 3.2 berechnet. Dann gilt mit jedem der von uns in Abschnitt 1.3 angegebenen Zeitintegrationsverfahren für den Prolongationsfehler $e^k(t + \Delta t)$ zur Zeit $t + \Delta t$,

$$e^k(t + \Delta t) := u^k(t + \Delta t) - P_{k+1}^k D_k^{k+1} u^k(t + \Delta t),$$

dass $e^k(t + \Delta t) = e^k(t)$. Insbesondere ist für alle Zellen $C_i \in \mathcal{G}^k$ auch $|e_i^k(t + \Delta t)| < \varepsilon_k$. Die Mehrskalenanalyse ermittelt also für alle späteren Zeitschritte, dass alle Variationen der Daten bereits auf Gitter $\mathcal{G}^{k+1}$ hinreichend genau diskretisiert werden können.

Beweis. Für das Polygonzugverfahren von Euler ist

$$u^k(t + \Delta t) = u^k(t) - \Delta t \overline{\nabla \cdot f}^k = u^k(t) - \Delta t P_{k+1}^k \overline{\nabla \cdot f}^{k+1}.$$

Also ist

$$\begin{aligned} D_k^{k+1} u^k(t + \Delta t) &= D_k^{k+1} u^k(t) - \Delta t D_k^{k+1} P_{k+1}^k \overline{\nabla \cdot f}^{k+1} \\ &= D_k^{k+1} u^k(t) - \Delta t \overline{\nabla \cdot f}^{k+1}. \end{aligned}$$

Aufgrund der Linearität von P_{k+1}^k erhalten wir damit

$$\begin{aligned} e^k(t + \Delta t) &:= u^k(t + \Delta t) - P_{k+1}^k D_k^{k+1} u^k(t + \Delta t) \\ &= \quad \left(u^k(t) - \Delta t P_{k+1}^k \overline{\nabla \cdot f}^{k+1}\right) \\ &\quad -\left(P_{k+1}^k D_k^{k+1} u^k(t) - \Delta t P_{k+1}^k \overline{\nabla \cdot f}^{k+1}\right) \\ &= u^k(t) - P_{k+1}^k D_k^{k+1} u^k(t) \;=\; e^k(t). \end{aligned}$$

Für die mehrstufigen Verfahren erhält man den Beweis durch wiederholtes Anwenden derselben Argumente. □

[2] In unserem Fall ist der Prolongationsoperator linear, da der enthaltene Rekonstruktionsoperator linear ist.

3.3 Exemplarischer Vergleich von Gitteradaption und Mehrskalenanalyse

Wir wollen uns an zwei Beispielen ansehen, wie gut die beiden Beschleunigungstechniken Gitteradaption und Mehrskalenanalyse funktionieren.
Das erste Beispiel ist ein Fall für Burgers Gleichung, bei dem eine glatte Startlösung so gesetzt wird, dass sich nach endlicher Zeit eine Unstetigkeit ausbildet.
Das Gebiet Ω ist gegeben durch $\Omega := [0,2] \times [0,0.7]$. Zur Zeit $t = 0$ wird auf Ω die Anfangsfunktion $u(x,0) := \sin(\pi x_1)$ gesetzt. Der linke und rechte Rand sowie der obere und untere Rand werden jeweils periodisch fortgesetzt. Bereits zur Zeit $T = 0.4$ liegt bei $x_1 \equiv 1$ eine Unstetigkeit von u vor. In x_2-Richtung bleibt u für festes x_1 konstant.
Für die gitteradaptive Rechnung starten wir mit einem relativ groben Gitter mit gleichmäßigen Zelldurchmessern $h \approx 0.08$. Die Schwellwerte für den Adaptionsindikator werden so eingestellt, dass Dreiecke T mit $AI(T) > 0.05$ verfeinert und mit $AI(T) < 0.025$ vergröbert werden. Durch ein Flächenlimit wird sichergestellt, dass durch das Adaptieren die kleinsten Zellen einen Durchmesser $h_{\min} \geq 0.02$ haben. Nach jedem zweiten Zeitschritt wird adaptiert. Nach einigen Zeitschritten steilt sich die Lösung in der Nähe der Linie $x_1 \equiv 1$ soweit auf, dass der Schwellwert für die Verfeinerung erreicht wird und dieser Bereich des Gitters dann verfeinert wird. Für die Vergleichslösung rechnen wir auf einem global feinen Gitter $\mathcal{G}^0$ mit $h \approx 0.02$. Die adaptierte Triangulierung $\mathcal{T}^A$ zur Zeit $T = 0.4$ ist in Abbildung 3.3 neben der Triangulierung dargestellt, aus der $\mathcal{G}^0$ durch baryzentrische Unterteilung konstruiert wurde. Die Zellenzahl der baryzentrischen Unterteilung von $\mathcal{T}^A$ ist 527 gegenüber $\#\mathcal{G}^0 = 4271$.

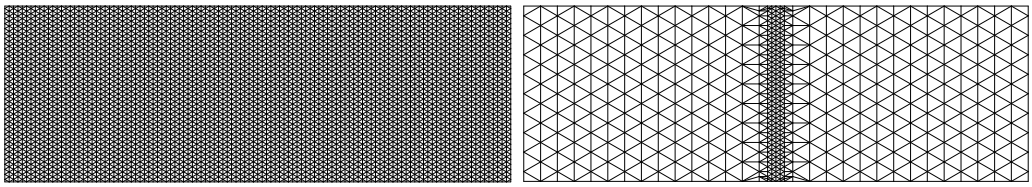

Abbildung 3.3: Triangulierung zu $\mathcal{G}^0$ sowie die adaptierte Triangulierung.

Bei der Rechnung mit Mehrskalenanalyse wird $\mathcal{G}^0$ als feinstes Gitter eingesetzt. Wir verwenden insgesamt vier Gitterlevel ($L = 3$), $\#\mathcal{G}^1 = 1473$, $\#\mathcal{G}^2 = 550$ und $\#\mathcal{G}^3 = 211$. In Abbildung 3.4 ist $\mathcal{G}^3$ dargestellt.
Eine Mehrskalenanalyse der (glatten) Anfangsfunktion ergibt $\max_i e_i^0 \approx 0.00018$, $\max_i e_i^1 \approx 0.0013$ und $\max_i e_i^2 \approx 0.0042$. Für die Lösung auf dem

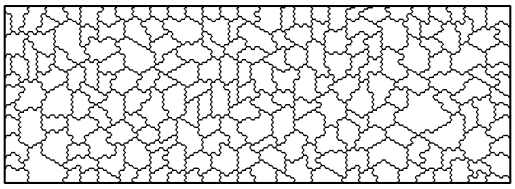

Abbildung 3.4: $\mathcal{G}^3$.

feinen Gitter zur Zeit $T = 0.4$ erhält man $\max_i e_i^0 \approx 0.13$, $\max_i e_i^1 \approx 0.25$ und $\max_i e_i^2 \approx 0.24$. Starten wir eine Rechung mit Mehrskalenanalyse und setzen $\varepsilon_0 := 0.002$, $\varepsilon_1 := 0.004$ sowie $\varepsilon_2 := 0.008$, so haben wir für die Anfangslösung die Situation von Satz 3.2.6 sowohl für das Paar $\mathcal{G}^0, \mathcal{G}^1$ als auch für $\mathcal{G}^1, \mathcal{G}^2$ und für $\mathcal{G}^2, \mathcal{G}^3$ gegeben.
In Abbildung 3.5 ist links die Näherungslösung entlang $x_2 \equiv 0.35$ zur Zeit $T = 0.4$ für die Rechnung auf dem feinen Gitter $\mathcal{G}^0$ ohne Verwendung von Gitteradaption und Mehrskalenanalyse zu sehen. In der Mitte ist die Näherungslösung aus der Rechnung mit Gitteradaption und rechts die mit den angegebenen Parametern unter Verwendung der Mehrskalenanalyse erhaltene Näherungslösung zur selben Zeit dargestellt.

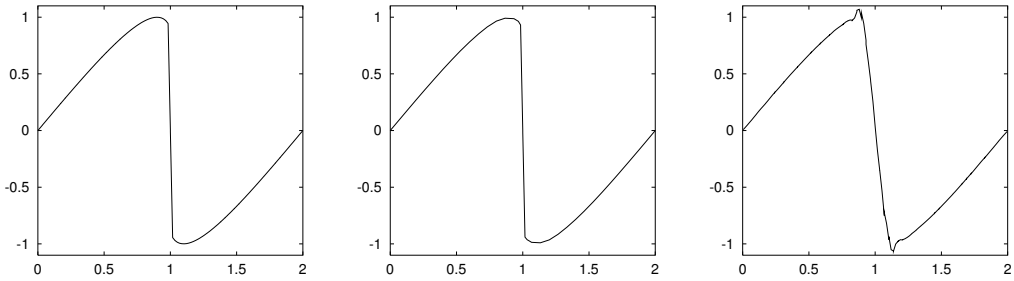

Abbildung 3.5: Näherungslösung aus der unbeschleunigten Rechnung (links), mit Gitteradaption (mitte) und mit Mehrskalenanalyse (rechts).

Man sieht keinen Unterschied zwischen der Lösung aus der unbeschleunigten Rechnung und der Lösung aus der Rechnung mit Gitteradaption. Die Lösung aus der Rechnung mit Mehrskalenanalyse ist dagegen deutlich schlechter. Dieselbe Lösung erhält man auch, wenn man eine Berechnung allein auf Gitter $\mathcal{G}^3$ durchführt und das Ergebnis auf $\mathcal{G}^2$, dann auf $\mathcal{G}^1$ und dann auf $\mathcal{G}^0$ prolongiert.
Die Rechenzeit für die Rechnung ohne Beschleunigung beträgt 256 Sekunden, bei Verwendung der Gitteradaption beträgt sie 23 Sekunden und mit Mehrskalenanalyse 66 Sekunden auf einem PC mit 333 MHz PentiumII Prozessor.

Als Zweites haben wir das Transportbeispiel aus Abschnitt 2.2 mit Gitteradaption sowie mit Beschleunigung durch Mehrskalenanalyse (vier Gitterlevel) gerechnet. Dabei wird auf dem Einheitsquadrat $[0,1] \times [0,1]$ die unstetige Startlösung

$$u(x,0) := \begin{cases} 1, & \text{falls } |x - (0.3, 0.3)^T| < 0.25, \\ 0, & \text{sonst} \end{cases}$$

gesetzt und diese mit der linearen Advektionsgleichung entlang des Vektors $(1,1)^T$ fortbewegt.
Bei der gitteradaptiven Rechnung verwenden wir als Grundtriangulierung eine gleichmäßig grobe Triangulierung mit Kantenlängen $h \approx 0.08$. Für die (unstetige) Startlösung bestimmen wir keine speziell angepasste Triangulierung, sondern verfeinern die Grundtriangulierung dreimal global. Dies führt zu einer gleichmäßig feinen Triangulierung mit Kantenlängen $h \approx 0.01$. Die Schwellwerte für den Adaptionsindikator werden so eingestellt, dass Dreiecke T mit $AI(T) > 0.05$ verfeinert und mit $AI(T) < 0.025$ vergröbert werden. Durch ein Flächenlimit wird sichergestellt, dass die Zellen auch nach dem Adaptieren einen Durchmesser $h \geq 0.01$ haben. Dies entspricht dem Zelldurchmesser der gleichmäßig feinen Starttriangulierung. Es wird nach jedem zweiten Zeitschritt adaptiert.
Da die Lösung $u(x,t)$ im Inneren des Kreises mit Zentrum $(0.3+t, 0.3+t)^T$ und Radius 0.25 sowie außerhalb dieses Kreises konstant ist, ist der Adaptionsindikator auf diesen Teilmengen klein und nur auf dem Kreisrand groß. Bei den ersten drei Adaptionen wird die feine Starttriangulierung im Inneren sowie außerhalb des Kreises vergröbert. Die resultierende Triangulierung nach sechs Zeitschritten ist in Abbildung 3.6 links dargestellt. Im Verlauf der Rechnung wird die Triangulierung durch lokales Verfeinern und Vergröbern mit der Lösung mitgeführt. Die Triangulierung am Ende der Berechnung zur Zeit $T = 0.3$ ist rechts in Abbildung 3.6 dargestellt. $\mathcal{G}^0$, die baryzentrische Unterteilung der feinen Starttriangulierung, hat 11 876 Zellen, das Gitter nach sechs Zeitschritten hat 1682 Zellen und das Gitter zur Zeit $T = 0.3$ hat 2120 Zellen.
Für die Rechnung mit Mehrskalenanalyse ist $\mathcal{G}^0$ wie bei der gitteradaptiven Rechnung die baryzentrische Unterteilung der gleichmäßig feinen Starttriangulierung mit $\#\mathcal{G}^0 = 11\,876$. Die groben Gitter haben $\#\mathcal{G}^1 = 3992$, $\#\mathcal{G}^2 = 1515$ und $\#\mathcal{G}^3 = 548$ Zellen.
Die Mehrskalenanalyse der Startlösung detektiert die Unstetigkeit in u. Gerechnet wird mit $\varepsilon_0 = \varepsilon_1 = \varepsilon_2 = 0.001$. In Abbildung 3.7 sind für die Gitter $\mathcal{G}^0$ bis $\mathcal{G}^2$ jeweils die Zellen C_i^k eingefärbt, für die die Berechnung von $\overline{\nabla \cdot f}_i^k$ beim ersten Zeitschritt durchgeführt werden muss. Auf $\mathcal{G}^3$ müssen natürlich

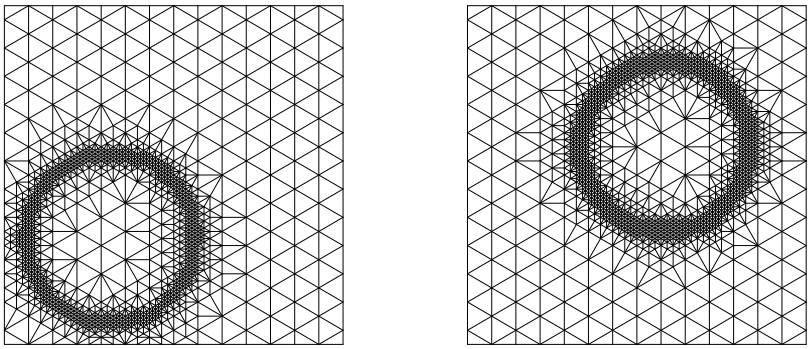

Abbildung 3.6: Adaptierte Triangulierung nach sechs Zeitschritten (links) sowie am Ende der Berechnung (rechts).

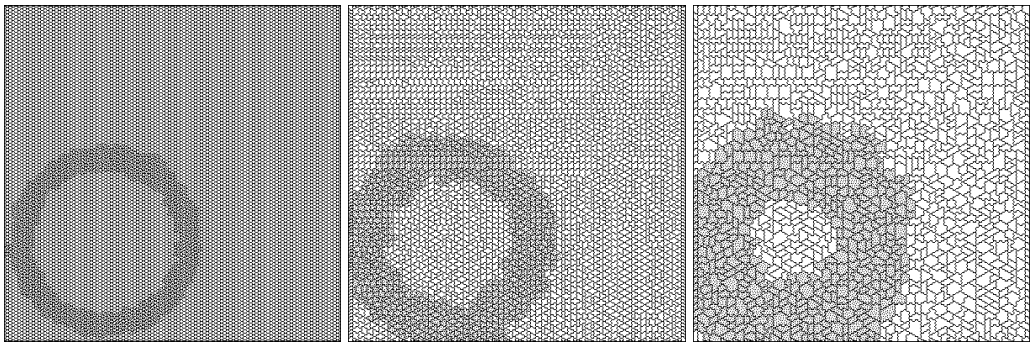

Abbildung 3.7: Ergebnis der Mehrskalenanalyse für die Startlösung. Eingefärbt sind die $\mathcal{G}^0_{\varepsilon_0}$ bis $\mathcal{G}^2_{\varepsilon_2}$.

alle $\overline{\nabla \cdot f_i}^3$ berechnet werden.
Die Teilmengen $\mathcal{G}^k_{\varepsilon_k}$ enthalten $\#\mathcal{G}^0_{\varepsilon_0} = 1423$, $\#\mathcal{G}^1_{\varepsilon_1} = 829$, $\#\mathcal{G}^2_{\varepsilon_2} = 497$ und $\#\mathcal{G}^3_{\varepsilon_3} = 548$ Zellen. Es müssen also 3297 statt 11 876 (ohne Mehrskalenanalyse) Berechnungen von $\overline{\nabla \cdot f_i}^k$ durchgeführt werden.
In Abbildung 3.8 kann man gut erkennen, dass die Mehrskalenanalyse die Lösung verfolgt hat. Die Teilmengen $\mathcal{G}^k_{\varepsilon_k}$ enthalten jetzt $\#\mathcal{G}^0_{\varepsilon_0} = 2225$, $\#\mathcal{G}^1_{\varepsilon_1} = 1040$, $\#\mathcal{G}^2_{\varepsilon_2} = 571$ und $\#\mathcal{G}^3_{\varepsilon_3} = 548$ Zellen. Es müssen also 4384 statt 11 876 (ohne Mehrskalenanalyse) Berechnungen von $\overline{\nabla \cdot f_i}^k$ durchgeführt werden. Obwohl die Anzahl der Berechnungen von $\overline{\nabla \cdot f_i}^k$ mit Mehrskalenanalyse im Mittel nur ungefähr ein Drittel der Anzahl von Berechnungen beträgt, die ohne Beschleunigungstechniken erforderlich sind, ist die Rechenzeit nur auf zwei Drittel gefallen.
In Abbildung 3.9 ist links die berechnete Näherungslösung der gitteradaptiven Rechnung dargestellt und rechts die berechnete Näherungslösung mit Mehrskalenanalyse. Zum Vergleich sind jeweils die exakte Lösung sowie die

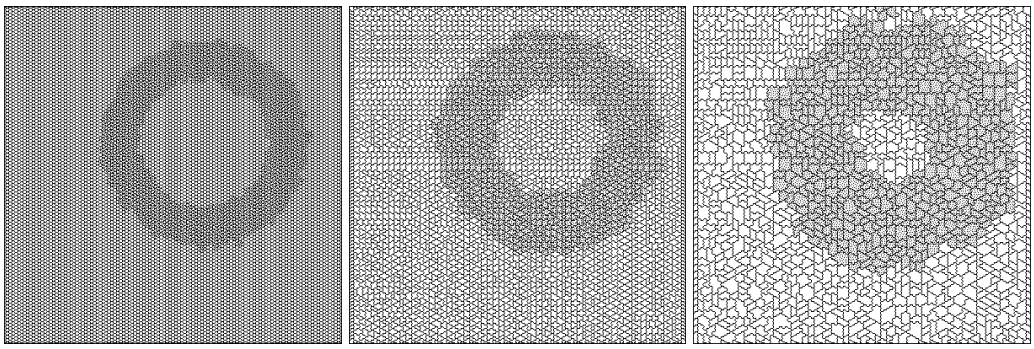

Abbildung 3.8: Ergebnis der Mehrskalenanalyse für den letzten Zeitschritt. Eingefärbt sind die $\mathcal{G}^0_{\varepsilon_0}$ bis $\mathcal{G}^2_{\varepsilon_2}$.

Lösung aus der unbeschleunigten Rechnung auf $\mathcal{G}^0$ mit dargestellt. In allen Fällen handelt es sich um die Daten entlang der Diagonalen von $(0,0)^T$ nach $(1,1)^T$.

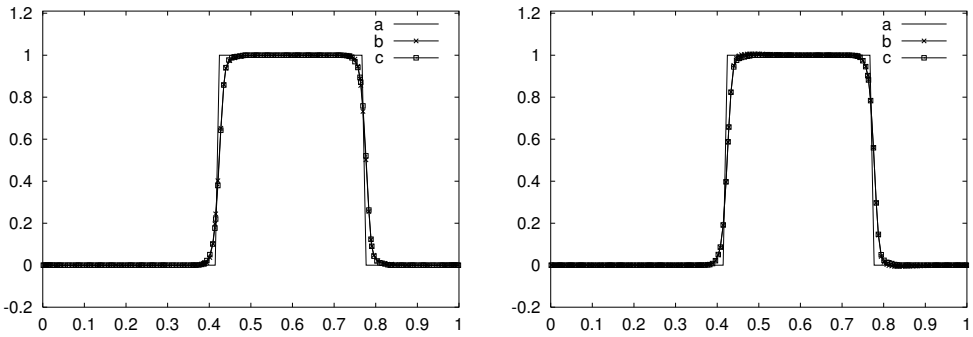

Abbildung 3.9: Links: a) Exakte Lösung, b) Näherungslösung mit Gitteradaption und c) Näherungslösung auf $\mathcal{G}^0$.
Rechts: a) Exakte Lösung, b) Näherungslösung mit Mehrskalenanalyse und c) Näherungslösung auf $\mathcal{G}^0$.

Alle Näherungslösungen stimmen gut überein. Die Lösungsqualität der unbeschleunigten Rechnung bleibt bei beiden Beschleunigungstechniken erhalten. Bei den Rechenzeiten ergeben sich in diesem Fall noch deutlichere Unterschiede als beim ersten Beispiel. Die unbeschleunigte Rechnung dauert 576 Sekunden, die gitteradaptive Rechnung dauert 44 Sekunden und die Rechnung mit Mehrskalenanalyse dauert 381 Sekunden auf einem PC mit 333 MHz PentiumII Prozessor. Der Rechenzeitgewinn durch die Mehrskalenanalyse ist enttäuschend klein.
Die sehr kurze Rechenzeit der gitteradaptiven Rechnung bedarf einer näheren

Erläuterung. Die Zellenzahl wird durch die Gitteradaption im Mittel ungefähr um einen Faktor sechs reduziert, wobei beim Start der Rechnung durch die Verwendung eines global feinen Startgitters Rechenzeit verschwendet wird.
Zusätzlich ergeben sich deutlich vergrößerte Zeitschritte, so dass die Gitteradaption nur 64 Zeitschritte gegenüber 282 Zeitschritten der unbeschleunigten Rechnung benötigt. Die deutlich kleineren Zeitschritte der unbeschleunigten Rechnung ergeben sich durch die CFL-Beschränkung. Bei der baryzentrischen Unterteilung einer gleichmäßig feinen Triangulierung entstehen am Rand des Gebietes Ω nämlich deutlich kleinere Zellen als im Inneren (siehe Abbildung 1.3, S. 19). Da die Dreiecke am Rand alle vergröbert werden (siehe Abbildung 3.6), sind die kleinsten Zellen der adaptierten Gitter deutlich größer als die kleinsten Zellen des unadaptierten Gitters.
Insgesamt kann man sagen, dass die theoretischen Betrachtungen durch die Testbeispiele bestätigt werden: Die Beschleunigung durch Mehrskalenanalyse ist kein adäquater Ersatz für die Gitteradaption. Außerdem ist die Motivation von Harten für die Einführung der Mehrskalenanalyse als Ersatz für die Gitteradaption, dass ein großer Aufwand in der Geometriebehandlung vermieden werden soll, auf unstrukturierten Gittern fraglich. Der Aufwand zur Generierung der groben Gitter sowie zur Beschreibung der Dezimierungs- und Prolongationsoperatoren ist nicht klein und durchaus vergleichbar mit dem Aufwand für die Realisierung der Gitteradaption.

3.4 Mehrgitterverfahren

Mehrgitterverfahren sind seit einigen Jahren ein bekanntes Mittel zur schnellen Lösung großer Gleichungssysteme. Für Gleichungssysteme, die aus der Diskretisierung von elliptischen Gleichungen stammen, gibt es eine ausgebaute Theorie zu Mehrgitterverfahren (siehe [Hac85]).
Obwohl diese Theorie bei der Diskretisierung von hyperbolischen Erhaltungsgleichungen nicht anwendbar ist, zeigt sich in der Praxis (siehe zum Beispiel [Jam83, LSD92, VM94]), dass mit Hilfe von Mehrgitterverfahren Berechnungen für stationäre Fälle drastisch beschleunigt werden können.

Definition 3.4.1. Gilt für die Lösung u einer hyperbolischen Erhaltungsgleichung auf einem Gebiet Ω, dass $u(\cdot, t)$ auf Ω für $t \to \infty$ gegen $u_\infty(\cdot)$ konvergiert und sind wir an $u_\infty(\cdot)$ interessiert, so sprechen wir von einem *stationären Fall*.

Bei den folgenden Ausführungen orientieren wir uns an [Mav95]. Wir gehen dabei zunächst von einem Zweigitterverfahren mit feinem Gitter $\mathcal{G}^0$ und gro-

bem Gitter $\mathcal{G}^1$ aus und erläutern später, wie man durch rekursive Anwendung eines Zweigitterverfahrens zum Mehrgitterverfahren kommt. Wir betrachten auf dem feinen Gitter $\mathcal{G}^0$ das diskrete Problem

$$L^0 u^0 = b^0 \tag{3.9}$$

mit dem (abstrakten) Operator L^0 und der gegebenen rechten Seite b^0. u^0 ist die gesuchte Lösung.

3.4.1 Einfaches Korrekturschema

Sei $\tilde{u}^0$ eine Näherungslösung für u^0, die mit einem iterativen Verfahren bestimmt wurde. Normalerweise wird $\tilde{u}^0 \neq u^0$ sein. Daher definiert man das *Residuum* der Gleichung (3.9)

$$r^0 := L^0 \tilde{u}^0 - b^0. \tag{3.10}$$

Das Ziel des Mehrgitterverfahrens ist es eine Korrektur v^0 zu berechnen, so dass möglichst

$$u^0 = \tilde{u}^0 + v^0 \tag{3.11}$$

gilt. Dazu werden zunächst die Gleichungen (3.9) und (3.10) addiert. Daraus erhält man

$$L^0 u^0 - L^0 \tilde{u}^0 = -r^0. \tag{3.12}$$

Falls der Operator L^0 linear ist, so kann man durch Einsetzen von Gleichung (3.11) in Gleichung (3.12) eine Darstellung für die exakte Korrektur v^0 bekommen:

$$L^0 v^0 = -r^0. \tag{3.13}$$

Die Idee des Mehrgitterverfahrens basiert nun auf der Annahme, dass das verwendete Iterationsverfahren zur Berechung von $\tilde{u}^0$ besonders gut hochfrequente Anteile des Fehlers eliminiert und andererseits langwellige Anteile des Fehlers nur sehr schlecht. In diesem Fall ist die erforderliche Korrektur v^0 relativ glatt und kann gut auf einem gröberen Gitter approximiert werden. Dazu verwendet man einen Dezimierungsoperator D_0^1 (siehe Abschnitt 3.2, S. 70ff.) sowie einen Grobgitteroperator L^1 und löst damit (exakt oder approximativ) das Problem

$$L^1 v^1 = -D_0^1 r^0. \tag{3.14}$$

Mit Hilfe eines Prolongationsoperators P_1^0 (siehe ebenfalls Abschnitt 3.2) erhält man eine Approximation $P_1^0 v^1 \approx v^0$. Damit setzt man

$$\tilde{u}^0_{neu} := \tilde{u}^0 + P_1^0 v^1. \tag{3.15}$$

Aus $\tilde{u}^0_{neu}$ werden nun erneut die hochfrequenten Fehleranteile durch Anwenden des Iterationsverfahrens auf dem feinen Gitter eliminiert um dann wiederum eine Korrektur auf dem groben Gitter zu bestimmen. Diese Technik wird solange wiederholt, bis r^0 hinreichend klein ist.

3.4.2 Schema mit Transfer der vollen Approximation

Die Darstellung (3.13) für die Korrektur v^0 basiert auf der Linearität von L^0. Ist L^0 nicht linear, wie in unserem Fall des Finite-Volumen-Verfahrens, so ist diese Darstellung nicht möglich. Stattdessen wird auf dem groben Gitter die Größe

$$\tilde{u}^1 := D_0^1 \tilde{u}^0 + v^1 \tag{3.16}$$

definiert. Damit wird Gleichung (3.12) auf dem groben Gitter durch

$$L^1 \tilde{u}^1 = L^1(D_0^1 \tilde{u}^0) - D_0^1 r^0 \tag{3.17}$$

approximiert. Durch die Definition von $\tilde{u}^1$ gemäß Gleichung (3.16) ist Gleichung (3.17) im linearen Fall äquivalent zu Gleichung (3.14). Hier muss nun nicht nur das Residuum r^0, sondern auch die Näherungslösung $\tilde{u}^0$ auf das grobe Gitter übertragen werden.
Bei der Lösung von (3.17) ist $\tilde{u}^1$ gesucht. Die Größen auf der rechten Seite dieser Gleichung sind fest.
Entsprechend zum linearen Fall wird Gleichung (3.17) (exakt oder approximativ) gelöst und mittels Gleichung (3.15) die neue Approximation $\tilde{u}^0_{neu}$ berechnet. Nach Gleichung (3.16) ist $v^1 = \tilde{u}^1 - D_0^1 \tilde{u}^0$. Damit erhält man hier

$$\tilde{u}^0_{neu} := \tilde{u}^0 + P_1^0 v^1 = \tilde{u}^0 + P_1^0(\tilde{u}^1 - D_0^1 \tilde{u}^0). \tag{3.18}$$

Dann wird genauso wie im linearen Fall mit einigen Iterationen auf dem feinen Gitter fortgefahren um hochfrequente Fehleranteile zu reduzieren. Danach wird erneut eine Korrektur auf dem groben Gitter berechnet. Dieser Zyklus wird wiederum solange wiederholt, bis r^0 hinreichend klein ist.

Bemerkung 3.4.2. Es ist zweckmäßig die iterative Lösung der Grobgittergleichung (3.17) mit dem Wert $\tilde{u}^1_{start} := D_0^1 \tilde{u}^0$ zu starten. In diesem Fall ist bei Erreichen der Lösung auf dem feinen Gitter ($r^0 = 0$) die Funktion $\tilde{u}^1_{start}$ exakte Lösung von Gleichung (3.17) und entsprechend ist die berechnete Korrektur $P_1^0(\tilde{u}^1 - D_0^1 \tilde{u}^0)$ gleich null.

3.4.3 Erweiterung des Zweigitterverfahrens zum Mehrgitterverfahren

Ist das Gitter $\mathcal{G}^0$ sehr fein, so ist auch das gröbere Gitter $\mathcal{G}^1$ noch relativ fein. Das Lösen von (3.17) ist also auch noch sehr aufwendig. Mit Hilfe eines nochmals gröberen Gitters $\mathcal{G}^2$ kann man zum Lösen ein Zweigitterverfahren mit dem feinen Gitter $\mathcal{G}^1$ und dem groben Gitter $\mathcal{G}^2$ verwenden. Dies kann man solange fortsetzen, bis man insgesamt eine Sequenz von $L+1$ Gittern $\mathcal{G}^0, \ldots, \mathcal{G}^L$ hat, so dass das Lösen der Grobgittergleichung für Gitter $\mathcal{G}^L$ hinreichend wenig aufwendig ist. Auf dem feinsten Gitter $\mathcal{G}^0$ ist (3.9) zu lösen und das Residuum r^0 ergibt sich aus (3.10). Für alle anderen Gitter ist entsprechend zu Gleichung (3.17)

$$L^k \tilde{u}^k = L^k(D_{k-1}^k \tilde{u}^{k-1}) - D_{k-1}^k r^{k-1} \tag{3.19}$$

zu lösen. Dabei ergibt sich r^k für $k \geq 1$ als Residuum von (3.19):

$$r^k := L^k \tilde{u}^k - \big(L^k(D_{k-1}^k \tilde{u}^{k-1}) - D_{k-1}^k r^{k-1}\big). \tag{3.20}$$

Beim Zusammenfügen der Sequenz von Zweigitterverfahren gibt es verschiedene Möglichkeiten. Die Gesamtzyklen werden rekursiv aufgebaut.
Am natürlichsten erscheint die Verwendung von sogenannten *V-Zyklen*:

Prozedur 3.3: V-ZYKLUS$(k_0, \tilde{u}^{k_0})$

Eingabe: Sequenz von Gittern $\mathcal{G}^0, \ldots, \mathcal{G}^L$,
k_0: Level des aktuell feinsten Gitters (Rekursionstiefe),
$\tilde{u}^{k_0}$: Näherungslösung auf Gitter $\mathcal{G}^{k_0}$.

Ausgabe: $\tilde{u}^{k_0}$: Neue Näherungslösung auf Gitter $\mathcal{G}^{k_0}$.

Start

- FALLS $k_0 = 0$
  - Führe einige Iterationsschritte für Gleichung (3.9) durch und berechne damit eine neue Approximation von $\tilde{u}^{k_0}$.
- Berechne r^{k_0} (siehe oben) sowie $\tilde{u}^{k_0+1} := D_{k_0}^{k_0+1} \tilde{u}^{k_0}$ und $L^{k_0+1}(D_{k_0}^{k_0+1} \tilde{u}^{k_0}) - D_{k_0}^{k_0+1} r^{k_0}$.
- Führe einige Iterationsschritte für Gleichung (3.19) ($k = k_0+1$) durch und berechne damit eine neue Approximation von $\tilde{u}^{k_0+1}$.

- FALLS $k_0 + 2 = L$
  - Berechne r^{k_0+1} (siehe oben) sowie $\tilde{u}^{k_0+2} := D_{k_0+1}^{k_0+2}\tilde{u}^{k_0+1}$ und $L^{k_0+2}(D_{k_0+1}^{k_0+2}\tilde{u}^{k_0+1}) - D_{k_0+1}^{k_0+2}r^{k_0+1}$.
  - Führe einige Iterationsschritte für Gleichung (3.19) ($k = k_0 + 2$) durch und berechne damit eine neue Approximation von $\tilde{u}^{k_0+2}$.
  - Setze $\tilde{u}^{k_0+1} := \tilde{u}^{k_0+1} + P_{k_0+2}^{k_0+1}(\tilde{u}^{k_0+2} - D_{k_0+1}^{k_0+2}\tilde{u}^{k_0+1})$.
- SONST V-ZYKLUS$(k_0 + 1, \tilde{u}^{k_0+1})$.
- Führe einige Iterationsschritte für Gleichung (3.19) ($k = k_0 + 1$) mit Startwert $\tilde{u}^{k_0+1}$ durch[3] und berechne damit eine neue Approximation von $\tilde{u}^{k_0+1}$.
- Setze $\tilde{u}^{k_0} := \tilde{u}^{k_0} + P_{k_0+1}^{k_0}(\tilde{u}^{k_0+1} - D_{k_0}^{k_0+1}\tilde{u}^{k_0})$.

Ende.

Wesentlich bessere Konvergenz erhalten wir bei unserem Finite-Volumen-Verfahren allerdings durch den Einsatz von sogenannten *W-Zyklen*:

Prozedur 3.4: W-ZYKLUS$(k_0, \tilde{u}^{k_0})$

Eingabe: Sequenz von Gittern $\mathcal{G}^0, \ldots, \mathcal{G}^L$,
k_0: Level des aktuell feinsten Gitters (Rekursionstiefe),
$\tilde{u}^{k_0}$: Näherungslösung auf Gitter $\mathcal{G}^{k_0}$.

Ausgabe: $\tilde{u}^{k_0}$: Neue Näherungslösung auf Gitter $\mathcal{G}^{k_0}$.

Start

- FALLS $k_0 = 0$
  - Führe einige Iterationsschritte für Gleichung (3.9) durch und berechne damit eine neue Approximation von $\tilde{u}^{k_0}$.
- Berechne r^{k_0} (siehe oben) sowie $\tilde{u}^{k_0+1} := D_{k_0}^{k_0+1}\tilde{u}^{k_0}$ und $L^{k_0+1}(D_{k_0}^{k_0+1}\tilde{u}^{k_0}) - D_{k_0}^{k_0+1}r^{k_0}$.
- Führe einige Iterationsschritte für Gleichung (3.19) ($k = k_0 + 1$) durch und berechne damit eine neue Approximation von $\tilde{u}^{k_0+1}$.

[3] Dieser Schritt kann auch entfallen. Dann spricht man vom *Sägezahn-V-Zyklus*.

- FALLS $k_0 + 2 = L$
  - Berechne r^{k_0+1} (siehe oben) sowie $\tilde{u}^{k_0+2} := D_{k_0+1}^{k_0+2}\tilde{u}^{k_0+1}$ und $L^{k_0+2}(D_{k_0+1}^{k_0+2}\tilde{u}^{k_0+1}) - D_{k_0+1}^{k_0+2}r^{k_0+1}$.
  - Führe einige Iterationsschritte für Gleichung (3.19) ($k = k_0 + 2$) durch und berechne damit eine neue Approximation von $\tilde{u}^{k_0+2}$.
  - Setze $\tilde{u}^{k_0+1} := \tilde{u}^{k_0+1} + P_{k_0+2}^{k_0+1}(\tilde{u}^{k_0+2} - D_{k_0+1}^{k_0+2}\tilde{u}^{k_0+1})$.
- SONST W-ZYKLUS$(k_0 + 1, \tilde{u}^{k_0+1})$.
- Führe einige Iterationsschritte für Gleichung (3.19) ($k = k_0 + 1$) mit Startwert $\tilde{u}^{k_0+1}$ durch und berechne damit eine neue Approximation von $\tilde{u}^{k_0+1}$.
- FALLS $k_0 + 2 = L$
  - Berechne r^{k_0+1} (siehe oben) sowie $\tilde{u}^{k_0+2} := D_{k_0+1}^{k_0+2}\tilde{u}^{k_0+1}$ und $L^{k_0+2}(D_{k_0+1}^{k_0+2}\tilde{u}^{k_0+1}) - D_{k_0+1}^{k_0+2}r^{k_0+1}$.
  - Führe einige Iterationsschritte für Gleichung (3.19) ($k = k_0 + 2$) durch und berechne damit eine neue Approximation von $\tilde{u}^{k_0+2}$.
  - Setze $\tilde{u}^{k_0+1} := \tilde{u}^{k_0+1} + P_{k_0+2}^{k_0+1}(\tilde{u}^{k_0+2} - D_{k_0+1}^{k_0+2}\tilde{u}^{k_0+1})$.
- SONST W-ZYKLUS$(k_0 + 1, \tilde{u}^{k_0+1})$.
- Führe einige Iterationsschritte für Gleichung (3.19) ($k = k_0 + 1$) mit Startwert $\tilde{u}^{k_0+1}$ durch und berechne damit eine neue Approximation von $\tilde{u}^{k_0+1}$.
- Setze $\tilde{u}^{k_0} := \tilde{u}^{k_0} + P_{k_0+1}^{k_0}(\tilde{u}^{k_0+1} - D_{k_0}^{k_0+1}\tilde{u}^{k_0})$.

Ende.

In Abbildung 3.10 sind der V-Zyklus und der W-Zyklus für 5 Gitterlevel ($L = 4$) grafisch dargestellt.

3.4.4 Verwendung beim Finite-Volumen-Verfahren

Wie bereits oben erwähnt, wollen wir Mehrgitterverfahren zur Beschleunigung des Finite-Volumen-Verfahrens für stationäre Fälle einsetzen. Hier ist die gesuchte Lösung u_∞ nicht mehr von der Zeit abhängig, also

$$\frac{d}{dt}u_\infty = 0.$$

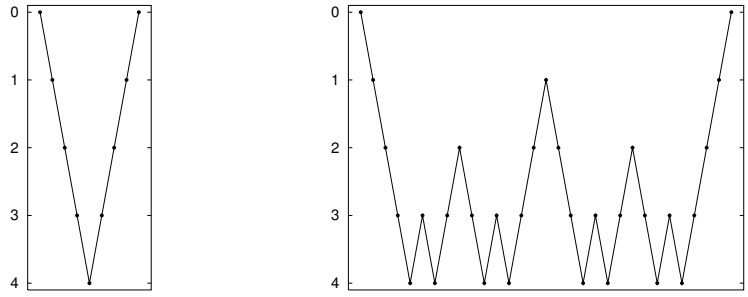

Abbildung 3.10: Fünf Level V-Zyklus (links) und W-Zyklus. Das jeweils verwendete Gitterlevel ist im zeitlichen Ablauf dargestellt.

Im diskreten Fall der Finite-Volumen-Approximation (siehe Definition 1.2.10, S. 26) heißt das

$$0 = \frac{d}{dt}\bar{u}_{\infty,i}(t) = \tilde{\mathcal{L}}_{C_i}(\bar{u}_\infty(t)), \qquad i = 1, \dots, \#\mathcal{G}.$$

Zur Bestimmung von $\bar{u} = \bar{u}_\infty$ haben wir also die diskrete Operatorgleichung

$$\tilde{\mathcal{L}}(\bar{u}) = 0 \tag{3.21}$$

zu lösen. Als Iterationsverfahren verwenden wir dabei, ausgehend von einer Startlösung $\bar{u}_0$, das Finite-Volumen-Verfahren, wobei wir für die Zeitdiskretisierung das Polygonzugverfahren von Euler mit lokalen Zeitschritten nehmen. Auf dem feinsten Gitter $\mathcal{G}^0$ haben wir also das Iterationsverfahren

$$\bar{u}_i^+ := \bar{u}_i + \Delta t_i \tilde{\mathcal{L}}_{C_i^0}(\bar{u}), \quad 1 \le i \le \#\mathcal{G}^0. \tag{3.22}$$

Die lokale Zeitschrittweite Δt_i wird jeweils so gewählt, dass sie die CFL-Bedingungen für Zelle C_i erfüllt (siehe Abschnitt 1.3.1, S. 28ff.).
Das Residuum r^0 ergibt sich unmittelbar aus Gleichung (3.21):

$$r^0 := \tilde{\mathcal{L}}(\bar{u}(t)).$$

Für die groben Gitter $\mathcal{G}^k$, $k \ge 1$, ist die Operatorgleichung

$$\tilde{\mathcal{L}}^k \bar{u}^k = \tilde{\mathcal{L}}^k (D_{k-1}^k \bar{u}^{k-1}) - D_{k-1}^k r^{k-1} \tag{3.23}$$

zu lösen. Ihr Residuum ergibt sich in natürlicher Weise als

$$r^k := \tilde{\mathcal{L}}^k \bar{u}^k - \big(\tilde{\mathcal{L}}^k (D_{k-1}^k \bar{u}^{k-1}) - D_{k-1}^k r^{k-1}\big). \tag{3.24}$$

Als Iterationsverfahren auf den groben Gittern $\mathcal{G}^k$, $k \geq 1$ erhalten wir

$$\bar{u}_i^{+k} := \bar{u}_i^k + \Delta t_i \Big(\tilde{\mathcal{L}}^k_{C_i^k}(\bar{u}) - \big(\tilde{\mathcal{L}}^k (D^k_{k-1} \bar{u}^{k-1}) - D^k_{k-1} r^{k-1} \big)_i \Big), \tag{3.25}$$

$1 \leq i \leq \#\mathcal{G}^k$. Für die Grobgitteroperatoren $\tilde{\mathcal{L}}^k$, $k \geq 1$, verwenden wir eine modifizierte Basisdiskretisierung auf den $\mathcal{G}^k$ (siehe unten). Die Dezimierungsoperatoren D_k^{k+1} sowie die Prolongationsoperatoren P_{k+1}^k definieren wir gemäß den Gleichungen (3.4), (3.5) (S. 73), wobei wir für den im Prolongationsoperator enthaltenen Rekonstruktionsalgorithmus $\mathcal{R}_{k+1}$ für $k \geq 1$ die triviale Rekonstruktion (siehe Abschnitt 1.2.4, S. 22) und für $k = 0$ die lineare polynomiale Rekonstruktion mit Polynomgrad eins (siehe Bemerkung 2.1.8, S. 44) verwenden.
Bei den Prozeduren V-Zyklus und W-Zyklus haben wir nicht näher angegeben, wieviele Iterationen auf einem bestimmtem Gitter $\mathcal{G}^k$ jeweils durchgeführt werden sollen. Wir machen die Anzahl vom Gitterlevel k abhängig. Auf den feineren Gittern sind die Iterationen noch relativ aufwendig. Je größer k ist, desto weniger aufwendig sind die Iterationen. Deshalb sind wir mit der Iterationszahl auf den gröberen Gittern großzügiger als auf den feineren Gittern. Auf $\mathcal{G}^0$ führen wir jeweils zwei Iterationsschritte durch. Auf $\mathcal{G}^k$, $k \geq 1$ führen wir jeweils $k+1$ Schritte durch.

3.4.5 Aufwand der Mehrgitterzyklen

Bei einem V- oder einem W-Zyklus entsteht der hauptsächliche Rechenaufwand durch die Durchführung der angegebenen Anzahl von Iterationen auf den verschiedenen Gittern. Beim V-Zyklus sieht man direkt, dass auf dem feinsten und dem gröbsten Gitter jeweils einmal iteriert werden muss und auf allen anderen Gittern jeweils zweimal. Beim W-Zyklus muss auf Gitter $\mathcal{G}^0$ ebenfalls einmal iteriert werden, auf dem gröbsten Gitter $\mathcal{G}^L$ muss insgesamt 2^{L-1} Mal iteriert werden und auf den dazwischenliegenden Gittern $\mathcal{G}^k$, $1 \leq k < L$, muss jeweils $3 \cdot 2^{k-1}$ Mal iteriert werden. In jedem Fall muss daher pro Mehrgitterzyklus dreimal der Feingitteroperator $\tilde{\mathcal{L}}$ ausgewertet werden, nämlich zweimal für die zwei Iterationen auf dem feinen Gitter und dann noch einmal für die Berechnung von r^0. Weiterhin ergibt sich für V- und W-Zyklen eine unterschiedliche Anzahl von Auswertungen der Grobgitteroperatoren $\tilde{\mathcal{L}}^k$, $1 \leq k \leq L$.
Beim V-Zyklus sind dies auf Gitter $\mathcal{G}^k$, $1 \leq k < L$, insgesamt $2(k+1)+1$ Auswertungen für die zweimal $k+1$ Iterationsschritte sowie für die einmalige Berechnung von r^k. Auf dem gröbsten Gitter $\mathcal{G}^L$ muss kein Residuum berechnet werden, hier ergeben sich $L+1$ Auswertungen für die $L+1$ Iterationen.

Beim W-Zyklus ergeben sich auf den Gittern $\mathcal{G}^k$, $1 \leq k < L$, insgesamt $3 \cdot 2^{k-1}(k+1) + 2^k$ Auswertungen, denn es wird $3 \cdot 2^{k-1}$ Mal iteriert und es muss 2^k Mal das Residuum r^k berechnet werden. Auf $\mathcal{G}^L$ müssen keine Residuen berechnet werden. Dort ergeben sich $2^{L-1}(L+1)$ Auswertungen.
Bei der Basisdiskretisierung auf den groben Gittern entsteht der Aufwand bei der Berechnung von $\tilde{\mathcal{L}}^k$ durch die numerischen Flussberechnungen über die inneren Kanten von $\mathcal{G}^k$ sowie durch die Berechnungen der Randflüsse. Wir wollen uns den Aufwand in Abhängigkeit von $\#\mathcal{G}^0$, der Zellenzahl des feinsten Gitters, ansehen:
In Abschnitt 3.2.3 (S. 74ff.) haben wir angegeben, wie groß die Zahl der inneren Kanten und der Randkanten der $\mathcal{G}^k$ ist. Dabei haben wir bereits festgestellt, dass die Anzahl der inneren Kanten beim Vergröbern deutlich langsamer fällt, als die Zahl der Zellen. Die Zahl der Randkanten ist bei den groben Gittern dieselbe wie bei $\mathcal{G}^0$. Dies liegt daran, dass die Zellränder beim Vergröbern immer komplexere Polygonzüge werden.

Modifizierte Basisdiskretisierung für die groben Gitter

Den numerischen Fluss zwischen zwei Zellen C_ℓ und C_r mit der Basisdiskretisierung kann man als Summe der numerischen Flüsse über die einzelnen Kanten

$$\sum_{j \in \mathcal{J}} l_j \cdot \tilde{f}(u_\ell, u_r, n_j) \tag{3.26}$$

schreiben, wobei l_j die Länge der j-ten Kante und n_j die äußere Normale an C_ℓ über die j-te Kante ist. Für die komplexen Ränder der Grobgitterzellen modifizieren wir die Basisdiskretisierung, indem wir diese Summe von numerischen Flussauswertungen zu einer einzigen Flussauswertung zusammenfassen (siehe Abbildung 3.11). Dazu ersetzen wir Gleichung (3.26) durch

$$\bar{l} \cdot \tilde{f}(u_\ell, u_r, \bar{n}), \tag{3.27}$$

mit

$$\bar{l} := \Big|\sum_{j \in \mathcal{J}} l_j \cdot n_j\Big| \quad \text{und} \quad \bar{n} := \frac{1}{\bar{l}} \sum_{j \in \mathcal{J}} l_j \cdot n_j.$$

Die $\bar{l}$ und $\bar{n}$ werden einmalig berechnet und als Teil der Gitter-Daten gespeichert. Die Kanten am Rand von Ω werden entsprechend zusammengefasst.
Wäre $\tilde{f}$ linear in der n-Komponente, so würde diese Modifikation die Ergebnisse nicht verändern. Da dies aber nicht der Fall ist, führt die Modifikation zu einer Änderung der Basisdiskretisierung. Dies führt aber nicht zu

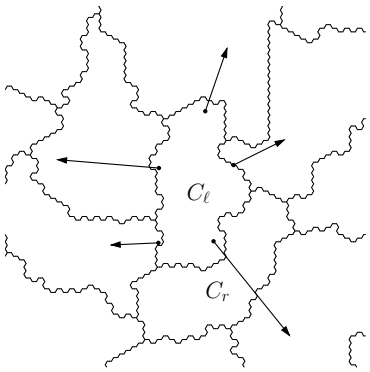

Abbildung 3.11: Zusammenfassung der Flussauswertungen an den Zellrändern. Es sind die Zellrändern und die $\bar{l} \cdot \bar{n}$ dargestellt.

Änderungen in den Ergebnissen für die (auskonvergierte) Feingitterlösung, da die Grobgitterkorrekturen und damit die Einflüsse der Grobgitteroperatoren im Limes verschwinden (siehe Bemerkung 3.4.2).
Der Aufwand für die so modifizierte Basisdiskretisierung hängt nicht mehr von der Zahl der Kanten ab, sondern nur noch von der Zahl der Gitterzellen. Die Zahl $\#\mathcal{G}^k$ der Zellen von $\mathcal{G}^k$ ist ungefähr $\#\mathcal{G}^0/3^k$. Setzen wir dies ein und verwenden die oben gezählte Anzahl von Auswertungen, so erhalten wir für einen L-Level V-Zyklus einen Aufwand proportional zu

$$\#\mathcal{G}^0 \cdot \left(\sum_{k=0}^{L-1} (2(k+1)+1) \frac{1}{3^k} + (L+1) \frac{1}{3^L} \right)$$

für die Berechnungen aller Grobgitteroperatoren.
Für einen L-Level W-Zyklus erhalten wir einen Aufwand proportional zu

$$\#\mathcal{G}^0 \cdot \left(\sum_{k=0}^{L-1} (3 \cdot 2^{k-1}(k+1) + 2^k) \frac{1}{3^k} + (2^{L-1}(L+1)) \frac{1}{3^L} \right)$$

für die Berechnungen aller Grobgitteroperatoren.
Man sieht unmittelbar, dass die als Faktor von $\#\mathcal{G}^0$ auftretenden Summen in beiden Fällen für $L \to \infty$ konvergent sind. Der Aufwand für die Durchführung eines V-Zyklus oder eines W-Zyklus ist also unabhängig von L durch $K \cdot \#\mathcal{G}^0$ nach oben beschränkt.
Zur Demonstration der Effizienz des Mehrgitterverfahrens haben wir eine Flügelprofilumströmung gerechnet. Das verwendete Flügelprofil ist das sogenannte NACA0012 Profil. Das Profil wird mit Hilfe der Funktion

$$f(x) := 0.6 \cdot (0.2969\sqrt{x} - 0.126x - 0.3516x^2 + 0.2843x^3 - 0.1015x^4)$$

definiert. Oberer und unterer Rand des Profils sind symmetrisch durch die Kurven $(x, f(\kappa x)/\kappa)^T$ und $(x, -f(\kappa x)/\kappa)^T$, $0 \leq x \leq 1$ mit $\kappa := 1.008930411365$ gegeben.
In Abbildung 3.12 ist die Triangulierung dargestellt, aus der $\mathcal{G}^0$ durch baryzentrische Unterteilung gebildet wurde. Die Triangulierung wurde mit dem Verfahren aus [Fri93] erzeugt.

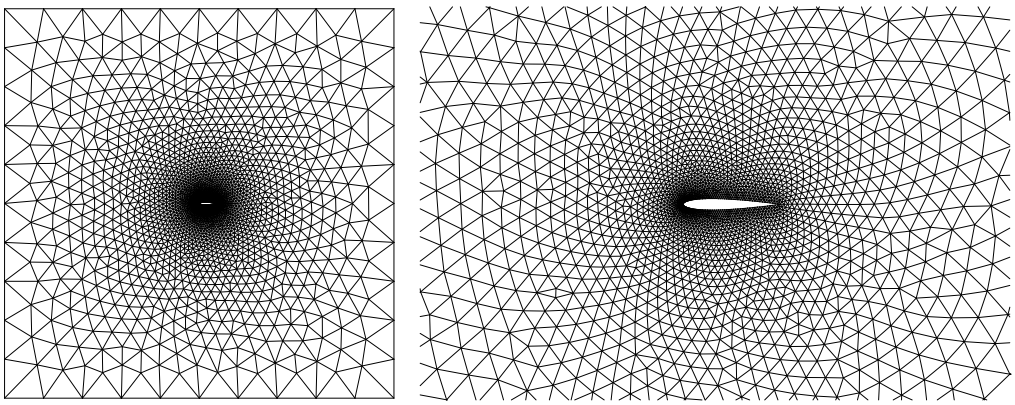

Abbildung 3.12: Verwendete Triangulierung (rechts ein Ausschnitt) für die Umströmung des NACA0012 Profils.

Das Profil wird mit einer Geschwindigkeit von $|\mathsf{v}| = 0.8$ und einem Winkel von $\alpha = 1.25°$ bei einem Druck $\mathsf{p} = 1$ und einer Dichte $\rho = 1.4$ angeströmt (siehe auch [RBY92]). Die Mach-Zahl der Anströmung ist also $\mathsf{Ma}_\infty = 0.8$. Als Startlösung wird die Parallelströmung mit den Daten der Anströmung gesetzt. Es bildet sich eine stationäre Strömung mit einem starken Stoß auf der Oberseite und einem schwächeren Stoß auf der Unterseite des Profils aus (siehe Abbildung 3.13, links).
Auf dem feinen Gitter wurde mit Polynomen vom Grad eins rekonstruiert. Gerechnet wurde zum Vergleich der Rechenzeit mit und ohne Mehrgitterbeschleunigung. Bei der Mehrgitterrechnung wurde der beschriebene sechs Level W-Zyklus verwendet. Die Rechenzeit auf einem PC betrug ohne Mehrgitterbeschleunigung 0.35 Sekunden pro Iterationsschritt, mit dem 6 Gitter W-Zyklus 2.56 Sekunden pro Zyklus. In Abbildung 3.14 ist in logarithmischer Darstellung die diskrete L_2-Norm von $d\rho/dt$ über der CPU-Zeit dargestellt. An dieser Größe kann man sehr gut die Konvergenz gegen den stationären Zustand ablesen. Man sieht, dass mit Mehrgitterbeschleunigung eine deutlich schnellere Konvergenz erfolgt als ohne.

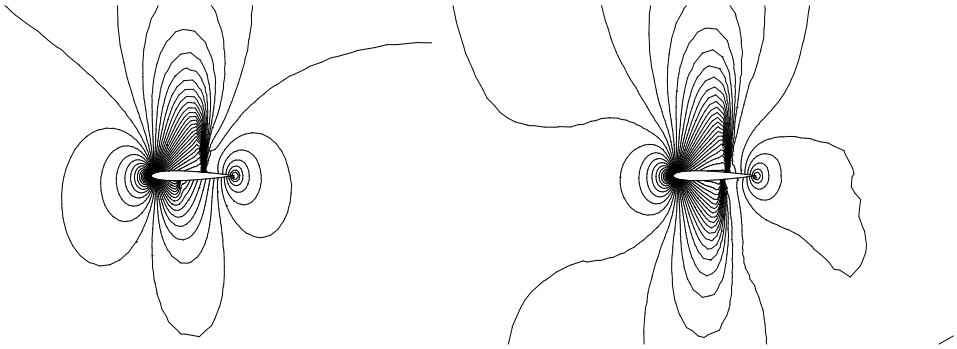

Abbildung 3.13: 50 äquidistante Isolinien des Drucks von 0.5 bis 1.5. Links ist die auskonvergierte Lösung, rechts eine Zwischenlösung ohne Mehrgitterbeschleunigung nach der Rechenzeit, nach der die Mehrgitterrechnung auskonvergiert ist, dargestellt. Die Zwischenlösung unterscheidet sich noch deutlich von der auskonvergierten Lösung.

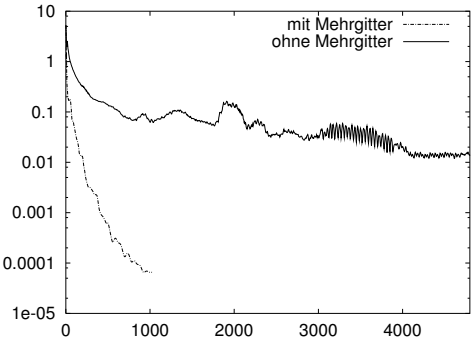

Abbildung 3.14: Diskrete L_2-Norm von $d\rho/dt$ über der CPU-Zeit.

Kapitel 4

Numerische Berechnungen

In diesem Kapitel werden einige Ergebnisse von Berechnungen für die Euler-Gleichungen präsentiert. In den ersten beiden Abschnitten wird die Qualität der numerischen Lösungen anhand von Beispielen, zu denen exakte Lösungen bekannt sind, überprüft. Dabei wird auch demonstriert, wie sich die Wahl der Parameter *numerische Flussfunktion* (siehe Abschnitt 1.2.5, S. 23ff.), *Oszillationsindikator* (siehe Abschnitt 2.2.4, S. 55ff.) und *Variablensatz für die Rekonstruktion* (siehe Abschnitt 2.3, S. 59ff.) auf die Lösungen auswirkt. Um die Darstellung überschaubar zu halten wird aber nicht das Kreuzprodukt aller möglichen Parametereinstellungen kombiniert. Stattdessen wird immer nur ein Parameter variiert, während für die anderen Parameter die von uns favorisierten Einstellungen festgehalten werden.
Nach vielen Vergleichsrechnungen favorisieren wir die folgenden Einstellungen: Als numerische Flussfunktion den lokalen Lax-Friedrichs Fluss mit matrixwertigem Diffusionsfaktor $\tilde{f}_{mlLF}$, als Oszillationsindikator OI_{L_2}, und als Variablensatz für die Rekonstruktion die primitiven Variablen. Die Wahl des Variablensatzes ist ein Kompromiss zwischen Rechenaufwand und Qualität der Ergebnisse. Durch Rekonstruktion in entkoppelten Variablen erhält man teilweise deutlich bessere Ergebnisse als in primitiven Variablen. Der Rechenaufwand ist aber so viel höher, dass wir die Rekonstruktion in primitiven Variablen bevorzugen.
Wann immer bei der folgenden Präsentation der Ergebnisse nichts anderes notiert ist, wird mit WENO-Rekonstruktion (Polynomgrad zwei) und den *favorisierten Einstellungen* gerechnet, als Zeitdiskretisierung wird das dreistufige Verfahren von Shu und Osher eingesetzt und als Gitter werden die baryzentrischen Unterteilungen von Triangulierungen verwendet.

4.1 Stoßrohrprobleme

Bei den zwei betrachteten sogenannten Stoßrohrproblemen handelt es sich um eindimensionale Testfälle, die bei uns zweidimensional gerechnet werden. Als Gitter wird jeweils die baryzentrische Unterteilung einer Triangulierung verwendet. Triangulierung und Gitter sind in Abbildung 4.1 dargestellt. Die Feinheit des Gitters entspricht den einhundert Zellen, die für diese Testfälle für eindimensionale Rechnungen üblicherweise verwendet werden, das Gebiet ist definiert durch $\Omega := [0, 1] \times [-0.05, 0.05]$.
Bei den Rechnungen werden keine Sonderbehandlungen durchgeführt um die Strömung eindimensional zu halten. Die Startlösungen enthalten keine Geschwindigkeitsanteile in y-Richtung und am oberen sowie unteren Rand wird die Randbehandlung für *feste Wände* verwendet. Für beide Beispiele wird bei den numerischen Ergebnissen jeweils die Dichte entlang der Linie $y \equiv 0$ dargestellt. Die Dichte ist hier, wie bei vielen anderen Beispielen auch, die Größe, an der man am besten die Qualität der Ergebnisse beurteilen kann. Zum Vergleich der Ergebnisse sei für beide Fälle auf [Bot95] verwiesen.

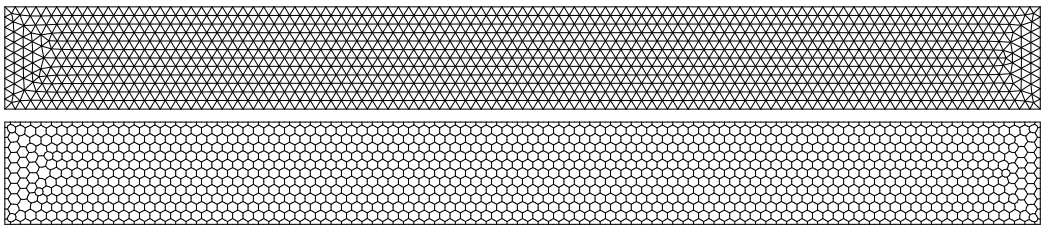

Abbildung 4.1: Triangulierung und baryzentrische Unterteilung für die Stoßrohrprobleme.

4.1.1 Das Beispiel von Sod

Das Testbeispiel von Sod wird durch die Anfangsdaten

$$(\rho, \mathsf{v}_1, \mathsf{v}_2, \mathsf{p})(x, 0) = \begin{cases} (1, 0, 0, 1) & \text{falls } x_1 \leq 0.5, \\ (0.125, 0, 0, 0.1) & \text{falls } x_1 > 0.5, \end{cases}$$

definiert. Gerechnet wird bis zur Zeit $T = 0.18$.
In Abbildung 4.2 sind die Ergebnisse für den lokalen Lax-Friedrichs Fluss mit matrixwertigem ($\tilde{f}_{mlLF}$) sowie mit skalarem Diffusionsfaktor ($\tilde{f}_{slLF}$) dargestellt. Für die Berechnungen mit der WENO-Rekonstruktion (links) sind kaum Unterschiede zwischen den Ergebnissen zu sehen. Verwendet man allerdings die Basisdiskretisierung, so kann man deutlich erkennen, dass die

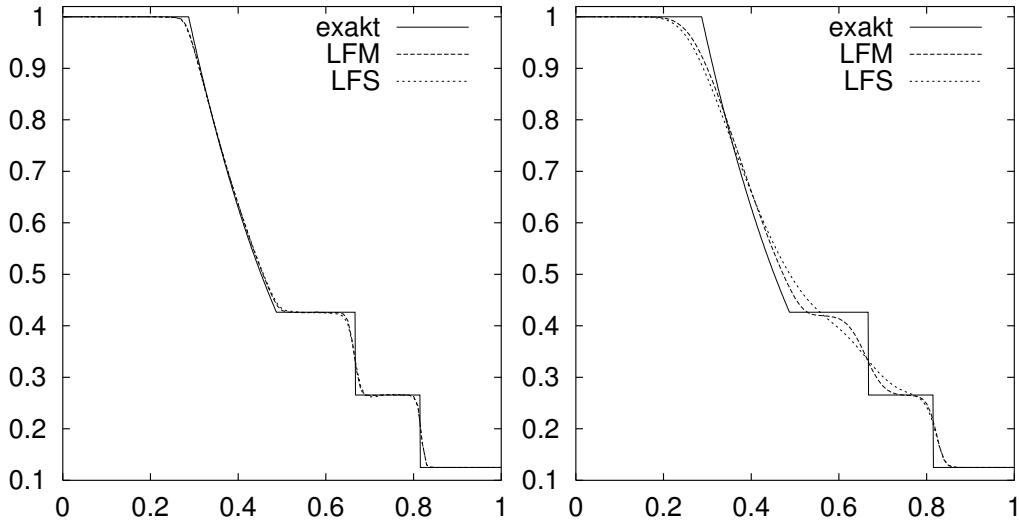

Abbildung 4.2: Beispiel von Sod. Ergebnisse für die numerischen Flussfunktionen $\tilde{f}_{mlLF}$ (LFM) und $\tilde{f}_{slLF}$ (LFS). Links Berechnungen mit WENO-Rekonstruktion, rechts mit Basisdiskretisierung.

Näherungslösung mit $\tilde{f}_{slLF}$ viel stärker verschmiert wird als mit $\tilde{f}_{mlLF}$. Wir haben dieses Beispiel auch mit dem numerischen Fluss von Osher und Solomon gerechnet, auf die Darstellung des Ergebnisses aber verzichtet, da keine Unterschiede zu den Ergebnissen mit $\tilde{f}_{mlLF}$ erkennbar sind.

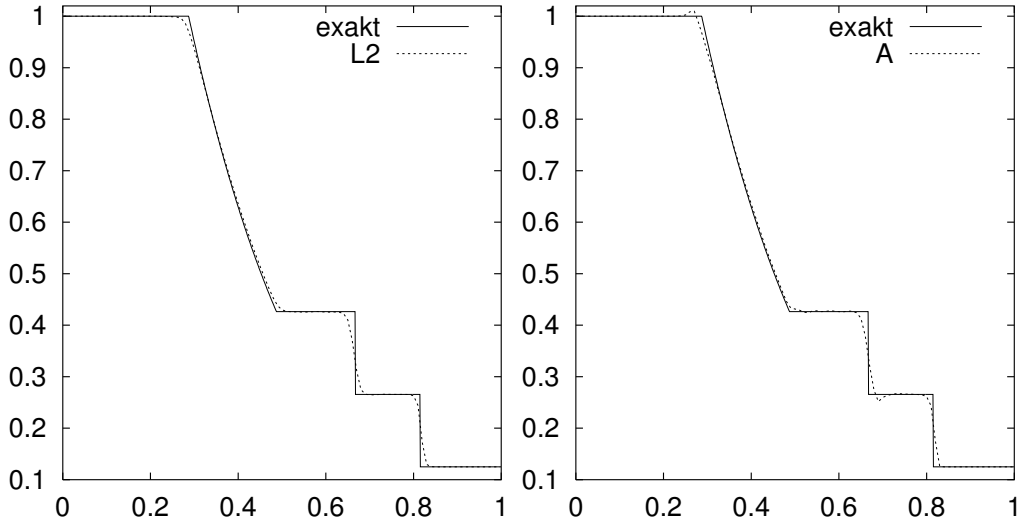

Abbildung 4.3: Beispiel von Sod. Ergebnisse für die Oszillationsindikatoren OI_{L_2} (links) und OI_A (rechts).

Beim Vergleich der Oszillationsindikatoren (siehe Abbildung 4.3) fällt bei diesem Beispiel nur der Indikator OI_A heraus. Bei Verwendung von OI_A zeigen sich deutliche Über- und Unterschießer der numerischen Lösung. Die Ergebnisse mit den anderen Oszillationsindikatoren sind kaum voneinander unterscheidbar. Auf deren Darstellung wurde deshalb verzichtet.
Für die drei von uns betrachteten Möglichkeiten zur Wahl des Variablensatzes ergeben sich bei diesem Testfall keine sichtbaren Unterschiede.

4.1.2 Das Beispiel von Lax

Das Testbeispiel von Lax wird durch die Anfangsdaten

$$(\rho, \mathsf{v}_1, \mathsf{v}_2, \mathsf{p})(x,0) = \begin{cases} (0.445, 0.698, 0, 3.528) & \text{falls } x_1 \leq 0.5, \\ (0.5, 0, 0, 0.571) & \text{falls } x_1 > 0.5, \end{cases}$$

definiert. Gerechnet wird bis zur Zeit $T = 0.1445$.

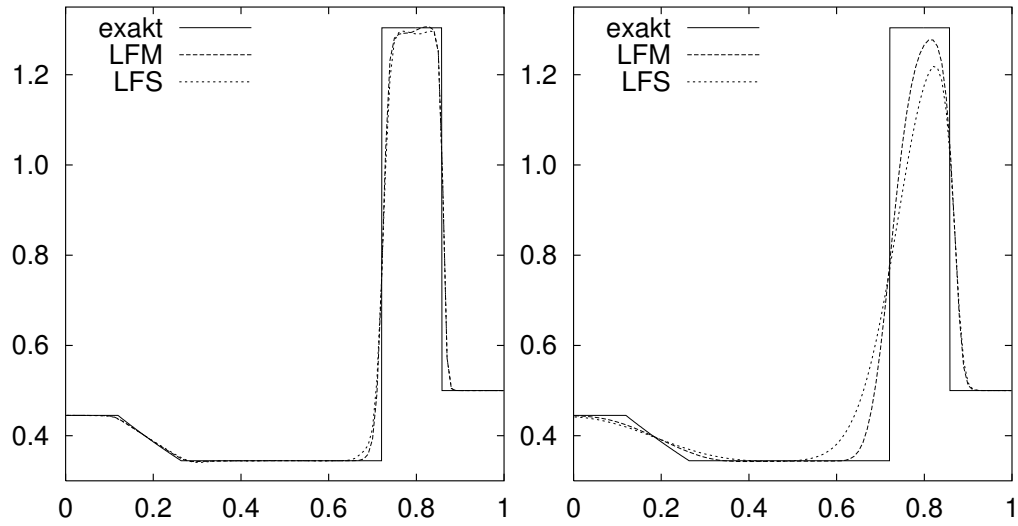

Abbildung 4.4: Beispiel von Lax. Ergebnisse für die numerischen Flussfunktionen $\tilde{f}_{mlLF}$ (LFM) und $\tilde{f}_{slLF}$ (LFS). Links Berechnungen mit WENO-Rekonstruktion, rechts mit Basisdiskretisierung.

Auch bei diesem Beispiel sieht man bei den Berechnungen mit WENO-Rekonstruktion kaum den Einfluss der numerischen Flussfunktionen (siehe Abbildung 4.4 links). Bei genauem Hinsehen kann man erkennen, dass die numerische Lösung mit $\tilde{f}_{slLF}$ im Bereich der Kontaktunstetigkeit ($x \approx 0.7$) etwas mehr verschmiert ist als mit $\tilde{f}_{mlLF}$. Bei den Ergebnissen mit der Basisdiskretisierung ist dies wiederum viel deutlicher zu erkennen. Auch hier

bestätigt sich, dass die Ergebnisse mit dem numerischen Fluss von Osher und Solomon sowie mit $\tilde{f}_{mlLF}$ gleichwertig sind. Diese Übereinstimmung haben wir bei allen durchgeführten Vergleichsrechnungen festgestellt.

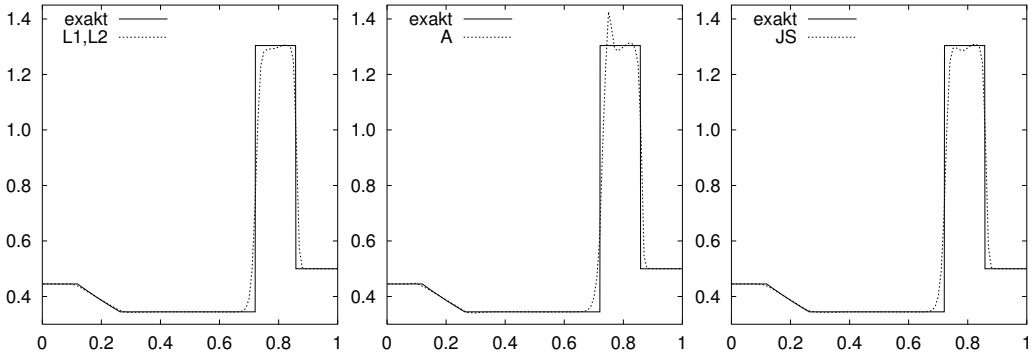

Abbildung 4.5: Beispiel von Lax. Ergebnisse für die Oszillationsindikatoren OI_{L_2}, OI_A und OI_{JS}.

Beim Vergleich der Oszillationsindikatoren (siehe Abbildung 4.5) fällt bei diesem Beispiel der Indikator OI_A extrem heraus: Die numerische Lösung zeigt einen deutlichen Überschießer. Die Ergebnisse mit OI_{L_1} und OI_{L_2} sind nicht zu unterscheiden. Mit OI_{JS} erhält man vergleichbare Ergebnisse. Es ist allerdings ein kleines zusätzliches Maximum bei $x \approx 0.75$ zu erkennen.

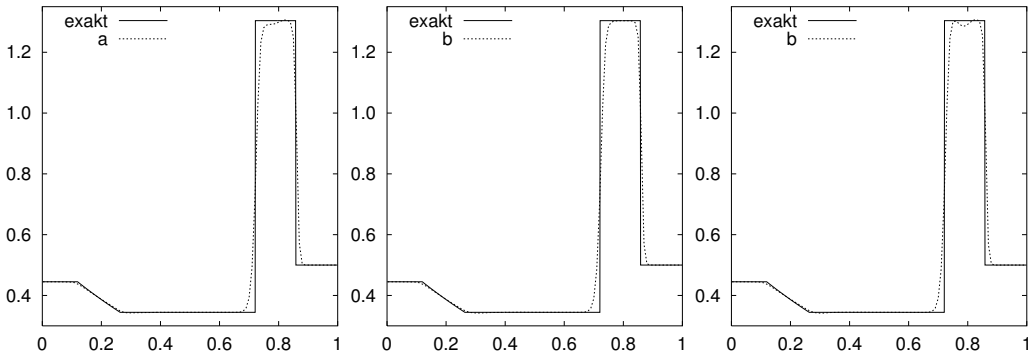

Abbildung 4.6: Beispiel von Sod. Ergebnisse für die unterschiedlichen Variablensätze zur Rekonstruktion. a: Rekonstruktion in primitiven Variablen, b: Rekonstruktion in entkoppelten Variablen und c: Rekonstruktion in Zustandsvariablen.

In Abbildung 4.6 sind die numerischen Ergebnisse zum Vergleich der Variablensätze für die Rekonstruktion dargestellt. Hier sind deutliche Unterschiede zu erkennen. Die beste Lösung erhält man mit Rekonstruktion in

entkoppelten Variablen. Eine sichtbare Verschlechterung ist bereits bei der Lösung mit Rekonstruktion in primitiven Variablen zu erkennen. Rekonstruiert man direkt in den Zustandsvariablen, dann bildet sich sogar ein kleines zusätzliches Maximum bei $x \approx 0.75$ aus.

4.2 Die Ringleb-Strömung

Zur numerischen Überprüfung der Approximationsordnung verwenden wir einen Testfall aus [AR-211], die Ringleb-Strömung. Diese Strömung ist durch Stromlinien definiert. Um eine Strömung von links nach rechts zu erhalten wird die Strömung um 90° gedreht. In [Fri98] haben wir ein Gebiet verwendet, dessen oberer und unterer Rand durch Stromlinien gegeben war. Da die Randbehandlung dort ohnehin durch numerische Flüsse mit den exakten Außenzuständen realisiert wurde und da wir die Tatsache, dass der Rand durch Stromlinien definiert war, in keiner Weise ausgenutzt haben, trennen wir uns hier von diesem Gebiet und verwenden einen rechteckigen Ausschnitt $\Omega := [-3.89711, 0] \times [1.1, 2.6]$. Das hat den Vorteil, dass Gitter mit gleichmäßigen Zelldurchmessern h verwendet werden können.
In Abbildung 4.7 ist die Dichteverteilung der exakten Lösung dargestellt.

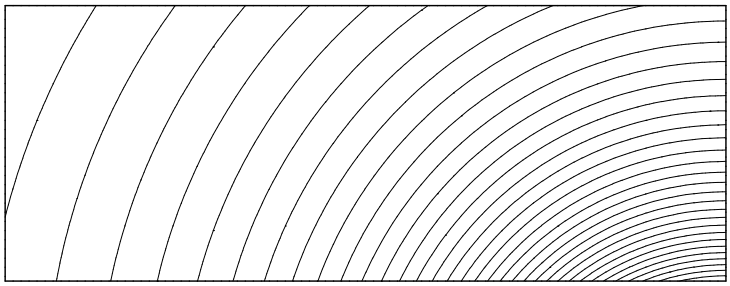

Abbildung 4.7: Exakte Lösung der Ringleb-Strömung: Isolinien der Dichte.

Als Startlösung wird bei allen Berechnungen die Parallelströmung mit den Daten

$$(\rho, \mathsf{v}_1, \mathsf{v}_2, \mathsf{p}) = (0.95, 0.3, 0, 0.65)$$

gesetzt. Dies entspricht bis auf die Richtung der Geschwindigkeit ungefähr der exakten Lösung in der linken oberen Ecke von Ω.
Da es sich bei diesem Beispiel um eine stationäre Strömung handelt, wird das Mehrgitterverfahren zur Beschleunigung eingesetzt. Dies führt bei diesem Beispiel zu einer massiven Beschleunigung der Berechnungen. In Abbildung 4.8 ist der Verlauf von $d\rho/dt$ der Berechnung auf dem Gitter mit

$h = 0.0375$ und WENO-Rekonstruktion mit quadratischen Polynomen über der CPU-Zeit dargestellt. Die Rechnungen wurden auf einem PC mit 400 MHz PentiumII Prozessor durchgeführt.

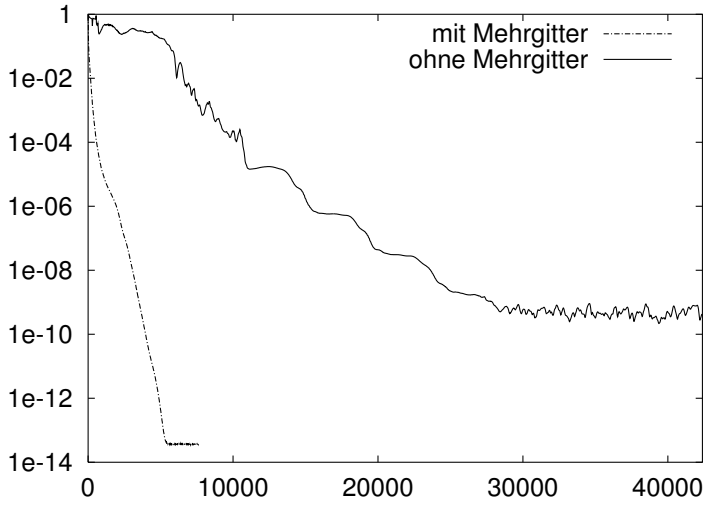

Abbildung 4.8: $d\rho/dt$ (logarithmisch) über der CPU-Zeit in Sekunden.

Misst man die Rechenzeit, die zum Erreichen von $d\rho/dt = 10^{-9}$ erforderlich ist, so ergibt sich auf diesem Gitter durch die Verwendung des Mehrgitterverfahrens eine Beschleunigung um etwa den Faktor acht.
Zur Sicherheit werden bei allen Berechnungen 500 Mehrgitterzyklen gerechnet. Bereits nach etwa 100 Mehrgitterzyklen sind keine wesentlichen Änderungen der Fehler mehr festzustellen.
Aus den Ergebnissen der Berechnungen für Gitter mit unterschiedlichen Zelldurchmessern h kann man die numerisch erhaltene Approximationsordnung bestimmen. Wir geben die Ergebnisse nur für die Dichte an. Für die anderen Komponenten der Lösungen erhält man vergleichbare Approximationsordnungen. Gemessen wird die Differenz zwischen den numerisch berechneten und den exakten Zellmittelwerten auf den jeweils verwendeten Gittern in der diskreten L_1- und L_∞-Norm.
Die folgenden Verfahren werden betrachtet:

a: Basisdiskretisierung,

b: Digitale ENO-Rekonstruktion (Polynomgrad eins),

c: WENO-Rekonstruktion (Polynomgrad eins),

d: Digitale ENO-Rekonstruktion (Polynomgrad zwei),

e: WENO-Rekonstruktion (Polynomgrad zwei) mit Rekonstruktion in primitiven Variablen,

f: WENO-Rekonstruktion (Polynomgrad zwei) mit Rekonstruktion in Zustandsvariablen und

g: WENO-Rekonstruktion (Polynomgrad zwei) mit Rekonstruktion in entkoppelten Variablen.

In Abbildung 4.9 sind die Fehler, gemessen in der diskreten L_1-Norm, grafisch dargestellt. Die Fehler in der diskreten L_∞-Norm sind in Tabelle 4.1

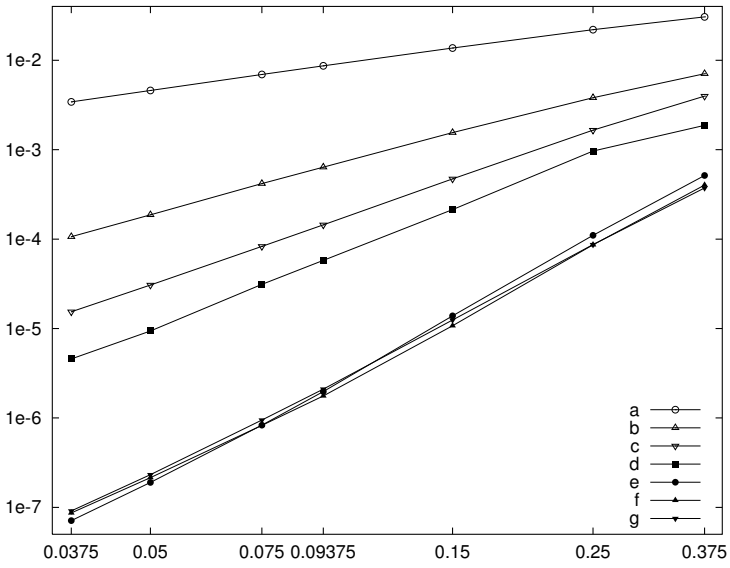

Abbildung 4.9: Fehler der Dichte-Komponente in der diskreten L_1-Norm über den Zelldurchmessern (jeweils logarithmisch).

aufgelistet. Aus den Fehlern kann man die numerische Approximationsordnung durch Messen der Steigung in der log-log-Darstellung ermitteln: Kennt man für h_1 und h_2 die Fehler e_1 und e_2, so ergibt sich

$$\text{numerische Fehlerordnung} = \frac{\ln e_2 - \ln e_1}{\ln h_2 - \ln h_1}.$$

Die so ermittelten Fehlerordnungen sind für die gemessenen Fehler in Tabelle 4.2 aufgelistet. Dabei sind die Fehlerordnungen zu jeweils aufeinander folgenden Zelldurchmessern h_1 und h_2 immer dem kleineren der beiden h_j zugeordnet worden.
Man sieht beim Vergleich der Fehler die deutliche Überlegenheit der gewichteten ENO-Verfahren über die digitalen ENO-Verfahren: Die Fehler für (c)

h	a	b	c	d	e	f	g
0.375	2.3e-2	5.4e-3	5.4e-3	2.1e-3	1.9e-3	1.3e-3	1.6e-3
0.25	1.7e-2	3.3e-3	2.8e-3	1.5e-3	5.8e-4	3.3e-4	4.0e-4
0.15	1.2e-2	1.6e-3	1.1e-3	3.6e-4	1.2e-4	4.9e-5	6.4e-5
0.09375	7.7e-3	7.8e-4	4.8e-4	1.2e-4	2.6e-5	1.3e-5	1.3e-5
0.075	6.3e-3	5.0e-4	3.3e-4	6.9e-5	1.3e-5	7.7e-6	6.9e-6
0.05	4.3e-3	2.2e-4	1.7e-4	2.9e-5	3.6e-6	2.8e-6	2.6e-6
0.0375	3.3e-3	1.4e-4	1.0e-4	1.7e-5	1.4e-6	1.3e-6	1.3e-6

Tabelle 4.1: Fehler der Dichte-Komponente in der diskreten L_∞-Norm.

h	a	b	c	d	e	f	g
0.25	0.8 0.7	1.5 1.2	2.2 1.7	1.6 0.9	3.8 3.0	3.8 3.5	3.6 3.4
0.15	0.9 0.8	1.8 1.4	2.5 1.8	3.0 2.8	4.1 3.2	4.1 3.7	3.8 3.6
0.09375	1.0 0.9	1.9 1.5	2.5 1.8	2.8 2.4	4.2 3.2	3.8 2.9	3.8 3.4
0.075	1.0 0.9	1.9 2.0	2.5 1.7	2.8 2.4	3.9 3.2	3.4 2.3	3.6 2.9
0.05	1.0 0.9	2.0 2.0	2.5 1.7	3.0 2.1	3.6 3.2	3.3 2.5	3.5 2.4
0.0375	1.0 0.9	2.0 1.6	2.4 1.7	2.5 1.8	3.4 3.2	3.2 2.6	3.3 2.6

Tabelle 4.2: Approximationsordnung für die L_1-Norm (jeweils links) und die L_∞-Norm (jeweils rechts).

sind deutlich geringer als die für (b) und die Fehler für (e), (f) und (g) sind deutlich geringer als die für (d).
Etwas überraschend sind die geringen Fehler und die hohen Fehlerordnungen von (e) gegenüber (f) und (g). Nach Satz 2.3.1, S. 61, hätte die Fehlerordnung für die Rekonstruktion in primitiven Variablen auf zwei einbrechen müssen. Weiterhin ist erstaunlich, dass die Fehlerordnung in der L_1-Norm bei WENO-Rekonstruktion mit Polynomen vom Grad eins über zwei und bei WENO-Rekonstruktion mit Polynomen vom Grad zwei über drei liegt. Im Gegensatz zu den in [LOC94, JS96, HS99] beschriebenen Verfahren wird bei unserem Verfahren nämlich nicht versucht durch Bedingungen an die Gewichte die Ordnung zu erhöhen.
Neben den deutlich höheren Fehlern kann man bei diesen Berechnungen ein weiteres Problem der digitalen ENO-Verfahren beobachten: Im Gegensatz zu den gewichteten ENO-Verfahren findet bei den einzelnen Berechnungen keine numerische Konvergenz gegen den stationären Zustand statt. Während bei den gewichteten ENO-Verfahren du/dt bis auf *numerisch null* zurückgeht, fällt diese Größe bei den digitalen ENO-Verfahren nach dem Start der Berechnungen nur um wenige Größenordnungen ab.

4.3 Doppel-Mach-Reflektion eines starken Stoßes

Dieses Beispiel stammt aus [WC84]. Zur Zeit $t = 0$ trifft ein Stoß mit Machzahl 10 in einem Winkel von 60° auf eine (reflektierende) feste Wand. Vor dem Stoß herrscht ein Ruhezustand mit Dichte $\rho = 1.4$ und Druck $\mathsf{p} = 1$. Hinter dem Stoß ergibt sich ein Zustand mit $\rho = 8$, $|\mathsf{v}| = 8.25$ und $\mathsf{p} = 116.5$. Die Wand befindet sich am Boden ($y = 0$) des Gebiets $\Omega := [0, 3] \times [0, 0.8]$ und beginnt bei $x = 1/6$.
Die Gitter der ersten Berechnungen sind die baryzentrischen Unterteilungen von gleichmäßigen Triangulierungen. Betrachtet werden die Zelldurchmesser $h = 1/100$ sowie $h = 1/200$. Danach werden Ergebnisse präsentiert, bei denen als Gitter direkt die beiden Triangulierungen verwendet werden. In allen Fällen sind die Ergebnisse zur Zeit $T = 0.2$ dargestellt.

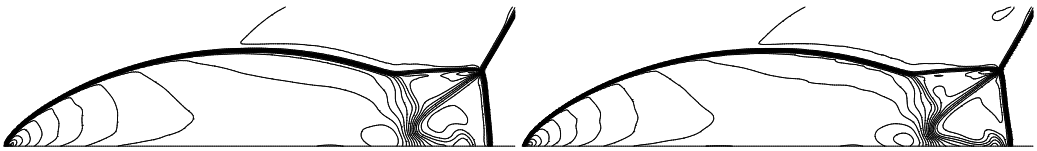

Abbildung 4.10: Digitale ENO-Rekonstruktion (links) und WENO-Rekonstruktion (rechts), jeweils Polynomgrad eins, $h = 1/100$. 35 Isolinien der Dichte ab 1.39 mit Abstand 0.6.

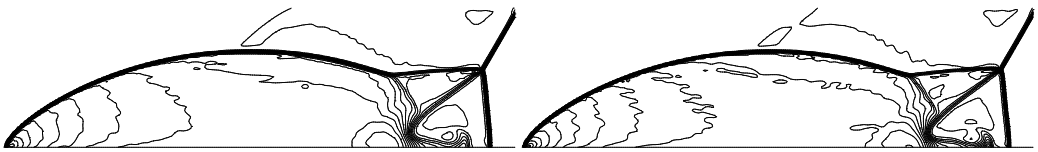

Abbildung 4.11: Wie in Abbildung 4.10, jedoch mit Polynomgrad zwei.

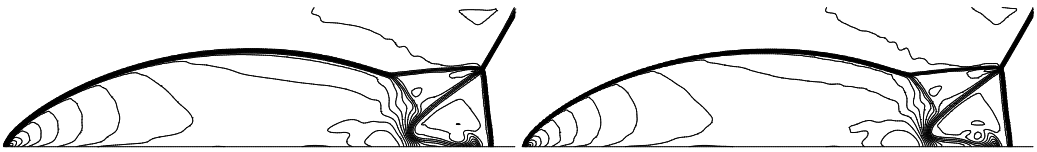

Abbildung 4.12: WENO-Rekonstruktion mit Polynomgrad zwei. Links: Lokaler Lax-Friedrichs Fluss mit skalarwertigem Diffusionsfaktor. Rechts: Rekonstruktion in entkoppelten Variablen. $h = 1/100$. 35 Isolinien der Dichte ab 1.39 mit Abstand 0.6.

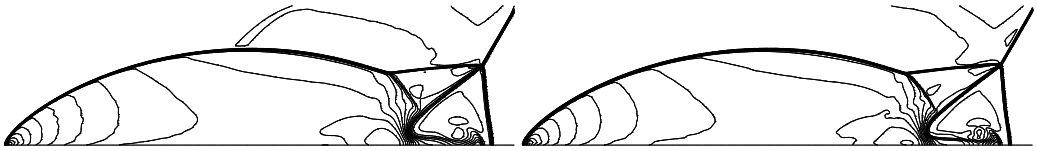

Abbildung 4.13: Links: WENO-Rekonstruktion mit Polynomgrad eins. Rechts: WENO-Rekonstruktion in entkoppelten Variablen mit Polynomgrad zwei. $h = 1/200$. 35 Isolinien der Dichte ab 1.39 mit Abstand 0.6.

In den Abbildungen 4.10 und 4.11 ist deutlich zu erkennen, dass die Ergebnisse mit WENO-Rekonstruktion weniger verschmiert sind, als die Ergebnisse mit digitaler ENO-Rekonstruktion. Außerdem sieht man den Auflösungsgewinn beim Übergang von Rekonstruktionspolynomen des Grades eins zum Grad zwei. Leider zeigen sich insbesondere in Abbildung 4.11 rechts deutliche Oszillationen in der Lösung. Diese verschwinden aber vollständig, wenn man statt des matrixwertigen Diffusionsfaktors den skalarwertigen Diffusionsfaktor beim lokalen Lax-Friedrichs Fluss verwendet (siehe Abbildung 4.12 links). Durch die erhöhte Diffusion werden allerdings die Unstetigkeiten im Ergebnis etwas stärker verschmiert. Zum Vergleich ist rechts in Abbildung 4.12 das Ergebnis mit lokalem Lax-Friedrichs Fluss mit matrixwertigem Diffusionsfaktor aber mit Rekonstruktion in entkoppelten Variablen dargestellt. Hier treten trotz der geringeren Diffusion keine Oszillationen auf. Bei Fällen mit derart starken Stößen muss man sich also entscheiden, ob man eine etwas erhöhte Diffusion akzeptieren kann, oder ob man das sehr aufwendige Verfahren mit Rekonstruktion in entkoppelten Variablen einsetzen muss oder eben die Oszillationen in den Ergebnissen hinnimmt.
In Abbildung 4.14 sind die Ergebnisse für Berechnungen dargestellt, bei denen als Gitter nicht die baryzentrischen Unterteilungen der Triangulierungen verwendet wurden, sondern unmittelbar die Triangulierungen.

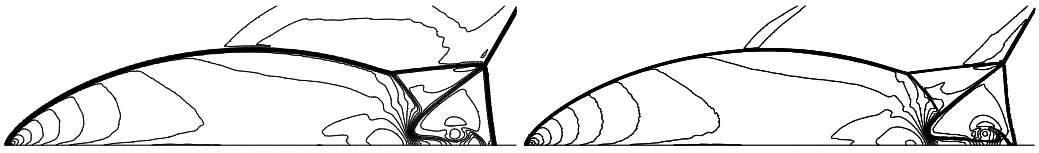

Abbildung 4.14: WENO-Rekonstruktion in entkoppelten Variablen mit Polynomgrad zwei, lokaler Lax-Friedrichs Fluss mit skalarwertigem Diffusionsfaktor. Gitter sind die Triangulierungen. Links $h = 1/100$, rechts $h = 1/200$. 35 Isolinien der Dichte ab 1.39 mit Abstand 0.6.

4.4 Strömung um eine rückwärts gerichtete Stufe

Ein optisch interessanter Fall ist die Strömung um eine rückwärts gerichtete Stufe. Eine eindrucksvolle Sequenz von Bildern zu diesem Fall findet man in [Dyk82]. Bei diesem Fall läuft ein Stoß mit Machzahl 2.4 über eine rückwärts gerichtete Stufe. Dadurch bildet sich eine komplexe Strömung aus. Der Rand von Ω ist durch den geschlossenen Polygonzug mit den Ecken $(-0.1, 0), (0, 0), (0, -1), (1.6, -1), (1.6, 1), (-0.1, 1)$ gegeben. Zur Zeit $t = 0$ tritt der Stoß bei $x = -0.1$ in Ω ein. Vor dem Stoß befindet sich ruhende Luft mit Dichte $\rho = 1.4$ und Druck $\mathsf{p} = 1$.
In Abbildung 4.15 sind numerische Lösungen zu den Zeiten $t = 0.2$, $t = 0.4$ und $T = 0.65$ dargestellt. Die Berechnungen wurden auf einem Gitter mit gleichmäßigen Zelldurchmessern $h = 0.0125$ durchgeführt, $\#\mathcal{G} = 24\,674$.

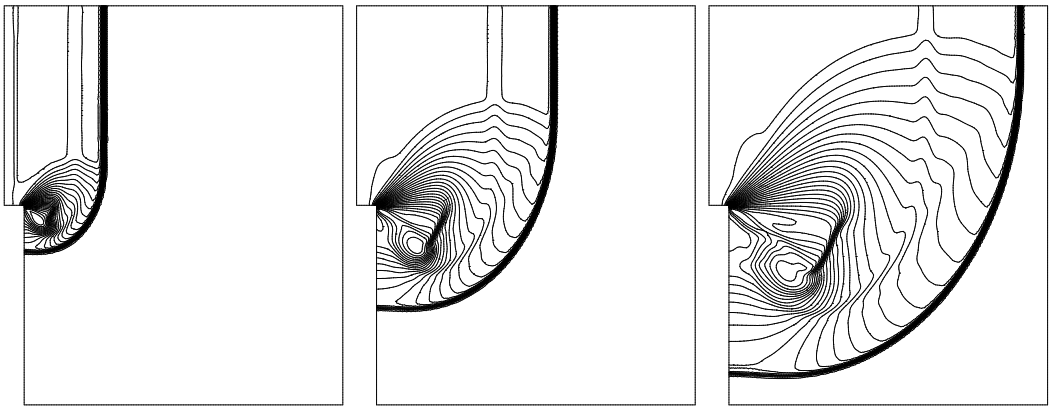

Abbildung 4.15: 35 Isolinien der Dichte ab 0 mit Abstand 0.135, WENO-Rekonstruktion mit Polynomgrad zwei, lokaler Lax-Friedrichs Fluss mit skalarwertigem Diffusionsfaktor auf einem gleichmäßig feinen Gitter mit $h = 0.0125$.

Um die Strömung besser aufzulösen wurde das Adaptionsverfahren verwendet. Dabei wurde alle fünf Zeitschritte adaptiert. Das Gitter zur Zeit $T = 0.65$ hat 262 563 Zellen bei $h_{\min} = 0.0015625$. Zur Zeit $t = 0.2$ hatte das Gitter noch unter 150 000 Zellen. Ein mit $h_{\min}$ global fein aufgelöstes Gitter hätte etwa 1.6 Millionen Zellen gehabt. Durch die Verwendung der Adaption wurde die Rechenzeit ungefähr um den Faktor zehn verringert.

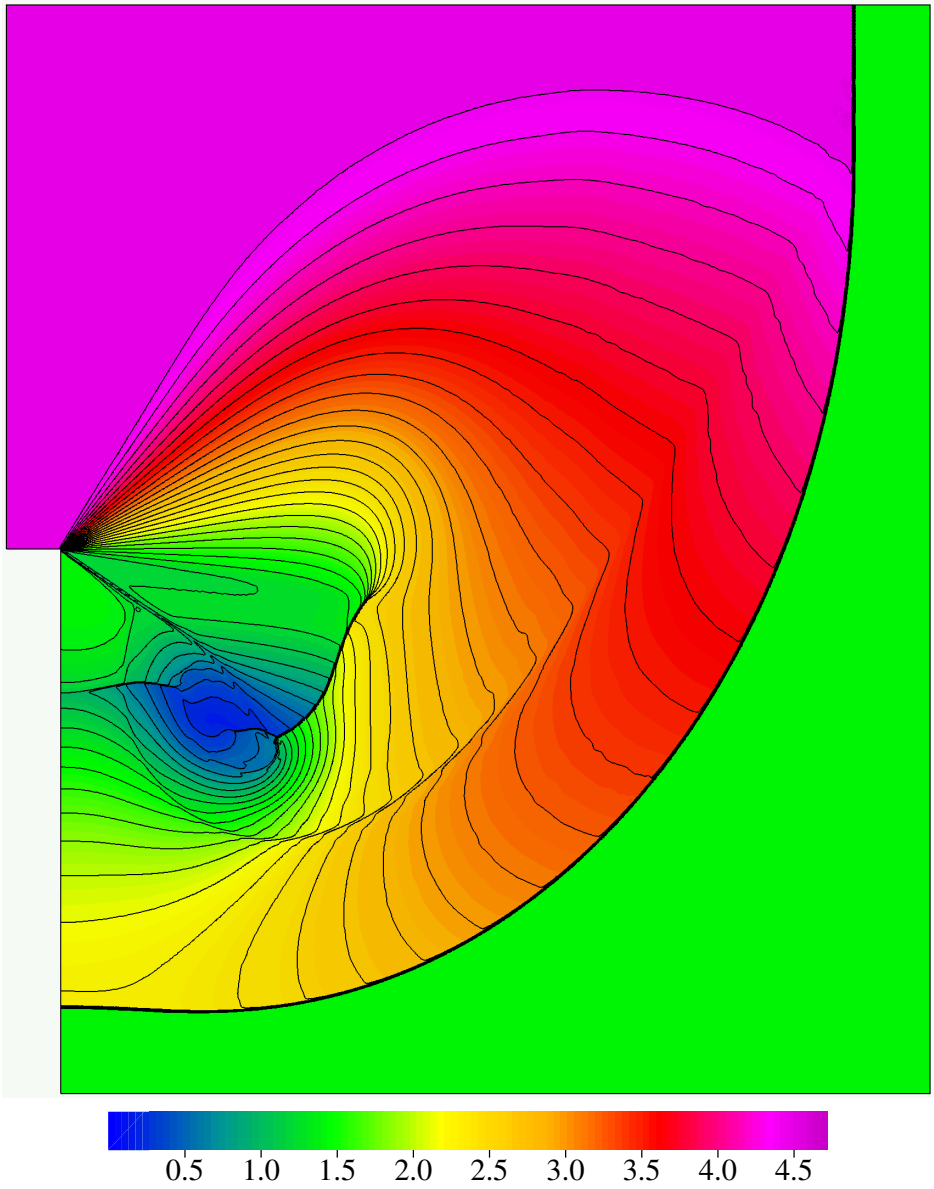

Abbildung 4.16: Dichteverteilung. WENO-Rekonstruktion in entkoppelten Variablen mit Polynomgrad zwei, lokaler Lax-Friedrichs Fluss mit matrixwertigem Diffusionsfaktor auf einem dynamisch adaptierten Gitter mit $h_{\min} = 0.0015625$.

4.5 Strömung über eine Keilspitze

Ein weiterer interessanter Strömungsfall wird in [Qui92] beschrieben. Hier befindet sich am Boden eines Kanals die Spitze eines Keils mit Winkel 90°. Der linke Rand des Gebiets ist bei $x = -0.3$, der untere Rand des Kanals liegt bei $y = 0$. Der Keil beginnt am Boden bei $x = 0$, die Spitze befindet sich am Punkt $(1/\sqrt{2}, 1/\sqrt{2})$ und der Keil endet bei $x = \sqrt{2}$. Zur Zeit $t = 0$ tritt ein Stoß mit Machzahl 2.8 bei $x = -0.3$ in Ω ein. In Ω befindet sich vor dem Einfall des Stoßes ruhende Luft mit Dichte $\rho = 1.4$ und Druck $\mathsf{p} = 1$. Gerechnet wird bis zur Zeit $T = 0.65$.
Auch bei diesem Fall haben wir das Adaptionsverfahren eingesetzt um die Strömung möglichst gut aufzulösen. Es wurde ebenfalls alle fünf Zeitschritte adaptiert. Das Gitter am Ende der Berechnung hat 192 490 Zellen. Ein mit $h_{\min} = 0.0015625$ global fein aufgelöstes Gitter hätte etwa 1.4 Millionen Zellen gehabt. Hier hatte das Gitter zur Zeit $t = 0.316$ noch 30 020 Zellen und zur Zeit $t = 0.434$ noch 72 989. Die Zellenzahl ist also erst kurz vor Erreichen von $T = 0.65$ drastisch angestiegen. Der geschätzte Rechenzeitgewinn beträgt hier mehr als ein Faktor zwanzig.

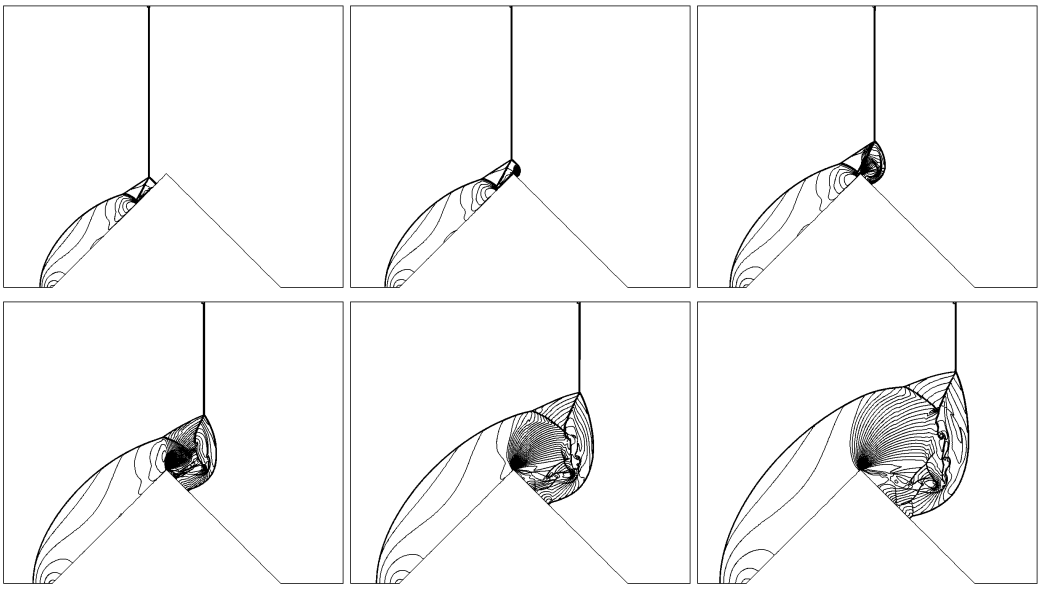

Abbildung 4.17: Isolinien der Dichte zu den Zeiten $t = 0.316$, $t = 0.349$, $t = 0.385$, $t = 0.435$, $t = 0.497$ und $t = 0.56$. WENO-Rekonstruktion in primitiven Variablen mit Polynomgrad zwei, lokaler Lax-Friedrichs Fluss mit skalarwertigem Diffusionsfaktor auf einem dynamisch adaptierten Gitter mit $h_{\min} = 0.0015625$.

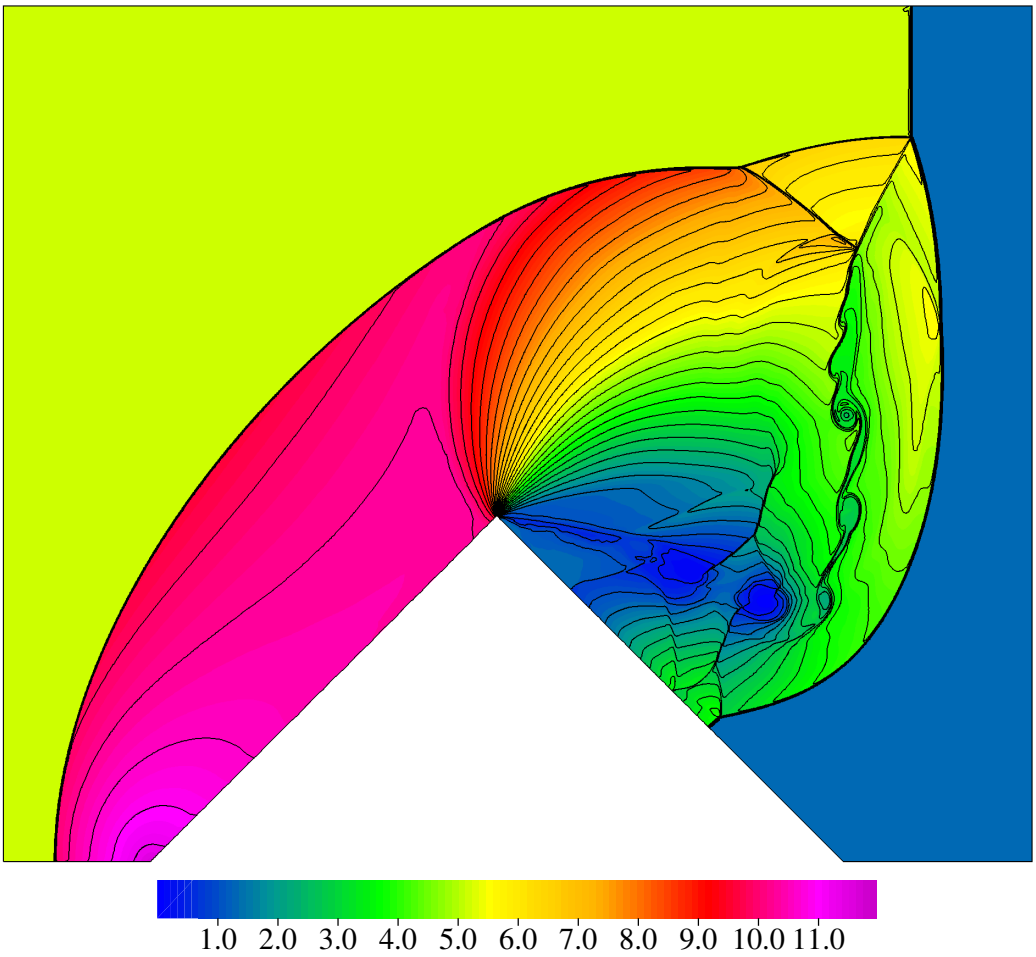

Abbildung 4.18: Dichteverteilung der numerischen Lösung zur Zeit $T = 0.65$. WENO-Rekonstruktion in primitiven Variablen mit Polynomgrad zwei, lokaler Lax-Friedrichs Fluss mit skalarwertigem Diffusionsfaktor auf einem dynamisch adaptierten Gitter mit $h_{\min} = 0.0015625$.

Zusammenfassung und Ausblick

In der vorliegenden Arbeit werden Finite-Volumen-Verfahren auf unstrukturierten Gittern behandelt. Dabei wird auf zwei Aspekte besonders eingegangen, nämlich zum einen auf den Rekonstruktionsalgorithmus und zum anderen auf Beschleunigungstechniken. Beide Aspekte werden algorithmisch umgesetzt.
Beim Rekonstruktionsalgorithmus wird die polynomiale Rekonstruktion dargestellt und analysiert. Die Idee der WENO-Rekonstruktion wird vom eindimensionalen Fall auf den Fall unstrukturierter Gitter in zwei Raumdimensionen übertragen. Die resultierende Verbesserung der Ergebnisse gegenüber der (digitalen) ENO-Rekonstruktion wird demonstriert. Im glatten Bereich der Lösungen führt die WENO-Rekonstruktion zu deutlich besseren Ergebnissen als die ENO-Rekonstruktion. Gleichzeitig bleibt die Qualität der Lösungen in der Nähe von Unstetigkeiten erhalten. Für stationäre Strömungen erhält man mit dem WENO-Verfahren auch ein stationäres Verhalten der numerischen Lösung, was beim ENO-Verfahren nicht der Fall ist.
Als Beschleunigungstechniken werden die Gitteradaption, die verallgemeinerte Mehrskalenanalyse und für stationäre Strömungen die Mehrgittertechnik behandelt. Bei der Effizienzanalyse der ersten beiden Techniken erhalten wir das klare Ergebnis, dass die verallgemeinerte Mehrskalenanalyse kein adäquater Ersatz für die Gitteradaption ist: Die Rechenzeit kann mit Hilfe der verallgemeinerten Mehrskalenanalyse nur um einen beschränkten Faktor reduziert werden, während bei der Gitteradaption die Beschleunigung beliebig groß werden kann, wenn das Gitter immer feiner wird. Bei der Gitteradaption wird proportional zur Rechenzeit auch Speicherplatz eingespart, während sich der Speicherplatzbedarf bei der verallgemeinerten Mehrskalenanalyse nicht verringert.
Es wird bewiesen, dass die verallgemeinerte Mehrskalenanalyse bei der Entstehung von Unstetigkeiten versagen kann. Dieses Versagen wird an einem Beispiel demonstriert.
Die Mehrgittertechnik für stationäre Strömungen wird erläutert und an einigen Beispielen wird gezeigt, dass damit die Rechenzeiten erheblich reduziert

werden können.
Offen bleibt die Frage, warum im Fall der Euler-Gleichungen die Rekonstruktion in primitiven Variablen nicht zum erwarteten Einbruch in der Genauigkeit führt. Beim Variablensatz für die Rekonstruktion stellt sich außerdem die Frage, wie man die Rekonstruktion in entkoppelten Variablen mit geringerem Rechenaufwand realisieren kann.
Im zweidimensionalen Fall kann man noch untersuchen, wie sich die Lösungsqualität ändert, wenn man auf die Regularitätsannahme für die Gitter verzichtet.
Für eine mögliche Implementierung der WENO-Rekonstruktion in drei Raumdimensionen müssen noch geeignete Schablonen ausgewählt werden.

Symbole und Notation

Symbole

α	Multi-Index (siehe S. 119).
Γ_{ij}	$(j = 1, \ldots, \#\Gamma_i)$ Randsegmente von Zelle C_i (Definition 1.2.1, S. 16).
γ	Verhältnis der spezifischen Wärmen ($\gamma = 1.4$) (Abschnitt 1.1, S. 15).
Δt	Zeitschrittweite.
$\Lambda(u, n)$	Diagonalmatrix der Eigenwerte von $A(u, n)$ (Bemerkung 1.1.3, S. 15).
$\Pi^Q(\mathbb{R}^2, \mathbb{R})$	Raum der Polynome vom Grad kleiner oder gleich Q auf dem $\mathbb{R}^2$.
Φ	Diffusionsparameter des numerischen Flusses von Lax und Friedrichs sowie von Varianten (Abschnitt 1.2.5, S. 23).
ρ	Dichte (Abschnitt 1.1, S. 15).
Ω	Beschränktes Gebiet im $\mathbb{R}^2$.
$\mathcal{A}(C)(\cdot)$	Zellmittelungsoperator (Definition 1.2.7, S. 20).
$A(u, \nu)$	Jacobimatrix von $f \cdot \nu$ (Definition 1.1.2, S. 14).
C_i	$(i = 1, \ldots, \#\mathcal{G})$ Zellen eines Gitters $\mathcal{G}$.
$\lvert C \rvert$	Flächeninhalt von Zelle C (Definition 1.2.1, S. 16).
$\operatorname{cond}_2 A$	Kondition der Matrix A $(= \lVert A \rVert_2 \cdot \lVert A^{-1} \rVert_2)$.
$\operatorname{conv} M$	Konvexe Hülle der Menge M.

D_k^{k+1}	Dezimierungsoperator (Definition 3.2.1, S. 71).
$\mathcal{D}(Q)$	Dimension von $\Pi^Q(\mathbb{R}^2, \mathbb{R})$.
$\mathrm{diag}(\lambda_1, \ldots, \lambda_n)$	Diagonalmatrix mit den Diagonaleinträgen $\lambda_1, \ldots, \lambda_n$.
E	Totalenergie (Abschnitt 1.1, S. 15).
$F(Q)$	Maximales Verhältnis zwischen dem Durchmesser einer Schablone für Polynomgrad Q und $h(C_1)$ (Definition 2.1.3, S. 40).
f	(f_1, f_2) Flusstensor, Flussfunktionen (Definition 1.1.1, S. 13).
$\tilde{f}(u_\ell, u_r, n)$	Numerischer Fluss (Abschnitt 1.2.5, S. 23).
$\mathcal{G}$	Gitter (Definition 1.2.1, S. 16).
$\#\mathcal{G}$	Anzahl der Zellen von $\mathcal{G}$.
G_i	$(i = 1, \ldots, N_G)$ Gaußknoten (Abschnitt 1.2.3, S. 21).
N_G	Zahl der Stützstellen bei der Gauß-Integration.
$h(C)$	Durchmesser von Zelle C (Definition 1.2.1, S. 16).
$\mathcal{I}(\alpha)$	Position von Multi-Index α (siehe S. 119).
$\mathcal{I}_\ell^k$	Menge der Indices von Zellen aus $\mathcal{G}^k$, deren Vereinigung C_ℓ^{k+1} ergibt (Gleichung (3.3)).
$K(Q)$	Maximale Kardinalität einer Schablone für Polynomgrad Q (Definition 2.1.3, S. 40).
$\ker D$	Kern (Nullraum) der linearen Abbildung D.
$\mathcal{L}_{C_i}$	Räumlicher Anteil der semidiskreten Finite-Volumen Formulierung (Gleichung (1.12), S. 20).
$\tilde{\mathcal{L}}_{C_i}$	Approximation für $\mathcal{L}_{C_i}$ (Abschnitt 1.2.7, S. 26).
$MD(d)$	Multiskalendarstellung von d (Definition 3.2.3, S. 72).
$\mathcal{N}(i, k)$	Index der Nachbarzelle von Zelle C_i über das Segment Γ_{ij} (Definition 1.2.5, S. 19).

$OI_{???}(p)$	Verschiedene Oszillationsindikatoren für Polynom p (Abschnitt 2.2.4, S. 55).
p	Druck (Abschnitt 1.1, S. 15).
P_{k+1}^k	Prolongationsoperator (Definition 3.2.1, S. 71).
$P(u,n), P^{-1}(u,n)$	Diagonalisierungsmatrizen für $A(u,n)$ (Bemerkung 1.1.3, S. 15).
Q	Maximaler Polynomgrad.
$\operatorname{rank} A$	Rang der Matrix A.
$\mathcal{S}$	Schablone (Definition 2.1.3, S. 40).
S	Zustandsraum (Abschnitt 1.1, S. 13).
$\mathcal{T}$	Triangulierung (Definition 1.2.2, S. 17).
T_i	$(i = 1, \ldots, \#\mathcal{G})$ Dreiecke einer Triangulierung $\mathcal{T}$.
v	$(\mathsf{v}_1, \mathsf{v}_2)$ Geschwindigkeit (Abschnitt 1.1, S. 15).
$\overline{x}_C$	Schwerpunkt der Zelle C (Definition 2.1.2, S. 40).
X_C	Lokale Koordinaten (Gleichung (2.2), S. 40).

Notation

Euklidische Norm

Für die euklidische Länge von Vektoren im $\mathbb{R}^n$ verwenden wir Betragsnotation:

$$|x| := \sqrt{x_1^2 + \ldots + x_n^2}.$$

Multi-Indizes

Bei polynomialen Ausdrücken im $\mathbb{R}^2$ verwenden wir Multi-Indizes. Für $\alpha = (\alpha_1, \alpha_2)$, $\alpha_i \in \{0, 1, 2, \ldots\}$ und $x \in \mathbb{R}^2$ definieren wir in der üblichen Weise

$$\begin{aligned} |\alpha| &:= \alpha_1 + \alpha_2, \\ x^\alpha &:= x_1^{\alpha_1} x_2^{\alpha_2} \quad \text{und} \\ D^\alpha &:= \frac{\partial^\alpha}{\partial x^\alpha} := \frac{\partial^{|\alpha|}}{\partial x_1^{\alpha_1} \partial x_2^{\alpha_2}}. \end{aligned}$$

Die Menge aller Multi-Indizes ordnen wir wie folgt:

$$\mathcal{I}(\alpha) := |\alpha|(|\alpha| + 1)/2 + \alpha_2.$$

Das heißt, $(0, 0)$ hat den Index 0 und die Multi-Indizes können wie folgt aufgezählt werden:

$$(0,0), (1,0), (0,1), (2,0), (1,1), (0,2), \ldots$$

Algorithmen

Bei der Beschreibung von Algorithmen bedienen wir uns einer Pseudo-Programmiersprache. Dabei verwenden wir die folgenden Konventionen:

1. Das Zeichen # verwenden wir als Abkürzung für *Anzahl*. Zum Beispiel wird *#Dreiecke* als *Anzahl Dreiecke* gelesen.

2. Folgende primitive Datentypen werden verwendet:
   Integer Ganze Zahl,
   Real Fließkommazahl,
   Coord Array von 2 Fließkommazahlen (z.B. Koordinaten im $\mathbb{R}^2$).

3. Arrays werden mit runden Klammern adressiert, die Indizierung beginnt mit 0. Beispiel:

   FÜR i VON 0 BIS $n - 1$
   Mittelwert(i):=*Integral*(i)/*Flächeninhalt*(i)

4. Mehrdimensionale Arrays werden durch mehrfache Verwendung von runden Klammern angesprochen:
   Real *Matrix*(n)(m) ist ein zweidimensionales Array der Größe $n \times m$. *Matrix*(i)(j) ist ein Eintrag, *Matrix*(i) ist ein eindimensionale Array (Länge m).

Literaturverzeichnis

[AH35] P. Alexandroff, H. Hopf, „Topologie I“, *Springer Verlag, Die Grundlehren der Mathematischen Wissenschaften in Einzeldarstellung, Band 45* (1935)

[Abg94a] R. Abgrall, „An essentially non-oscillatory reconstruction procedure on finite-element type meshes: application to compressible flows“, *Comput. Methods Appl. Mech. Engrg.* **116**, 95–101 (1994)

[Abg94b] R. Abgrall, „On Essentially Non-oscillatory Schemes on Unstructured Meshes: Analysis and Implementation“, *J. Comput. Phys.* **114**, 45–58 (1994)

[AH94] R. Abgrall, A. Harten, „Multiresolution Representation in Unstructured Meshes: I. Preliminary Report“, *CAM Report 94-20, Department of Mathematics, University of California, Los Angeles* (1994)

[AR-211] „Test Cases for Inviscid Flow Field Methods“, *AGARD ADVISORY REPORT No. 211*

[Bän91] E. Bänsch, „Local mesh refinement in 2 and 3 dimensions“, *Impact Comput. Sci. Engrg.* **3**, 181–191 (1991)

[Bot95] N. Botta, „Numerical investigation of two-dimensional Euler flows: cylinder at transonic speed“, *Dissertation 10852 an der ETH Zürich* (1995)

[Cia78] P. G. Ciarlet, „The Finite Element Method for Elliptic Problems“, *North-Holland* (1978)

[CFL28] R. Courant, K. Friedrichs, H. Lewy, „Über die partiellen Differenzengleichungen der mathematischen Physik“, *Math. Annalen* **100**, 32–74 (1928)

[Dyk82] M. Van Dyke, „An Album of Fluid Motion“, *The Parabolic Press, Stanford* (1982)

[Fri93] O. Friedrich, „A New Method for Generating Inner Points of Triangulations in Two Dimensions“, *Comput. Methods Appl. Mech. Engrg.* **104**, 77–86 (1993)

[Fri98] O. Friedrich, „Weighted Essentially Non-Oscillatory Schemes for the Interpolation of Mean Values on Unstructured Grids“, *J. Comput. Phys.* **144**, 194–212 (1998)

[FHMS96] O. Friedrich, D. Hempel, A. Meister, Th. Sonar, „Adaptive Computation of Unsteady Flow Fields with the DLR-τ-Code.“, *AGARD-CP-578*, 37-1–37-11 (1996)

[FSPS98] O. Friedrich, F. Schröder-Pander, Th. Sonar, „Generalized Multiresolution Analysis on Unstructured Grids“, *Hamburger Beiträge zur Angewandten Mathematik, Reihe F* (1998)

[Hac85] W. Hackbusch, „Multi-Grid Methods and Applications.“, *Springer Verlag* (1985)

[Har91] A. Harten, „Multi-Resolution Analysis for ENO Schemes“, *ICASE Report No. 91-77* (1991)

[Har93] A. Harten, „Multiresolution Algorithms for the Numerical Solution of Hyperbolic Conservation Laws“, *CAM Report 93-03, Department of Mathematics, University of California, Los Angeles* (1993)

[Har94] A. Harten, „Adaptive Multiresolution Schemes for Shock Computations“, *J. Comput. Phys.* **115**, 319–338 (1994)

[Har96] A. Harten, „Multiresolution Representation of Data: A General Framework“, *SIAM J. Numer. Anal.* **33**, No. 3, 1205–1256 (1996)

[HC91] A. Harten, S. R. Chakravarthy, „Multi-Dimensional ENO Schemes for General Geometries“, *ICASE Report No. 91-76* (1991)

[Hem96] D. Hempel, „Isotropic refinement and recoarsening in two dimensions“, *Numerical Algorithms* **13**, 33–43 (1996)

[Hem99] D. Hempel, „Rekonstruktionsverfahren auf unstrukturierten Gittern zur numerischen Simulation von Erhaltungsprinzipien“, *Dissertation am Fachbereich Mathematik der Universität Hamburg* (eingereicht 1999)

[HMS96] D. Hietel, A. Meister, Th. Sonar, „On the Comparision of Four Different Implementations of an Implicit Third-Order ENO Scheme of Box Type for the Computation of Unsteady Compressible Flow“, *Numerical Algorithms* **13**, 77–105 (1996)

[Hir90] C. Hirsch, „Numerical Computation of Internal and External Flows“, Vol. 2, *John Wiley & Sons* (1990)

[HS99] C. Hu, C.-W. Shu, „Weighted Essentially Non-oscillatory Schemes on Triangular Meshes“, *J. Comput. Phys.* **150**, 97–127 (1999)

[Jam83] A. Jameson, „Solution of the Euler equations for two dimensional transonic flow by a multigrid method.“, *Appl. Math. Comp.* **13**, 327–356 (1983)

[JS96] G.-S. Jiang, C.-W. Shu, „Efficient Implementation of Weighted ENO Schemes“, *J. Comput. Phys.* **126**, 202–228 (1996)

[LSD92] M.-H. Lallemand, H. Steve, A. Dervieux, „Unstructured Multigriding by Volume Agglomeration: Current Status“, *Computers and Fluids* **21**, 397–433 (1992)

[LOC94] X.-D. Liu, S. Osher, T. Chan, „Weighted Essentially Non-oscillatory Schemes“, *J. Comput. Phys.* **115**, 200–212 (1994)

[Mav95] D.J. Mavriplis, „Multigrid Techniques for Unstructured Meshes“, *ICASE TR-95-27*, 1–59 (1995)

[Mei96] A. Meister, „Zur zeitgenauen numerischen Simulation reibungsbehafteter, kompressibler, turbulenter Strömungsfelder mit einer impliziten Finite-Volumen-Methode vom Box-Typ“, *Dissertation D17 am Fachbereich Mathematik der Technischen Hochschule Darmstadt* (1996)

[MM94] K. W. Morton, D. F. Mayers, „Numerical Solution of Partial Differential Equations“, *Cambridge University Press* (1994)

[OS82] S. Osher, F. Solomon, „Upwind Difference Schemes for Hyperbolic Conservation Laws“, *Math. Comp.* **38**, 339–374 (1982)

[PVMZ87] J. Peraire, M. Vahdati, K. Morgan, O.C. Zienkiewicz, „Adaptive Remeshing for Compressible Flow Computations“, *J. Comput. Phys.* **72**, 449–466 (1987)

[Qui92] J.J. Quirk, „A Contribution to the Great Riemann Solver Debate“, *ICASE Report No. 92-64* (1992)

[RBY92] R.D. Rausch, J.T. Batina, H.T.Y. Yang, „Spatial Adaption of Unstructured Meshes for Unsteady Aerodynamic Flow Computations“, *AIAA Journal* **30**, No. 5, 1243–1251 (1992)

[SO88] C.-W. Shu, S. Osher, „Efficient Implementation of Essentially Non-Oscillatory Shock-Capturing Schemes“, *J. Comput. Phys.* **77**, 439–471 (1988)

[Son93] Th. Sonar, „Strong and weak norm refinement indicators based on the finite element residual for compressible flow computations“, *Impact of Computing in Science and Engineering* **5**, 111–127 (1993)

[Son97a] Th. Sonar, „On the Construction of Essentially Non-Oscillatory Finite Volume Approximations to Hyperbolic Conservation Laws on General Triangulations: Polynomial Recovery, Accuracy and Stencil Selection“, *Comput. Methods Appl. Mech. Engrg.* **140**, 157–181 (1997)

[Son97b] Th. Sonar, „Mehrdimensionale ENO-Verfahren“, *B. G. Teubner Stuttgart* (1997)

[SS94] Th. Sonar, E. Süli, „A Dual Graph Norm Refinement Indicator for Finite Volume Approximations of the Euler Equations“, *Oxford University Computing Laboratory Report No. 94/9* (1994)

[SW96] Th. Sonar, G. Warnecke, „On Finite Difference Error Indication for Adaptive Approximations of Conservation Laws“, *Hamburger Beiträge zur Angewandten Mathematik, Preprint 109* (1996)

[Sto89] J. Stoer, „Numerische Mathematik I“, 5. Auflage, *Springer-Verlag* (1989)

[SB90] J. Stoer, R. Bulirsch, „Numerische Mathematik II“, 3. Auflage, *Springer-Verlag* (1990)

[VM94] V. Venkatakrishnan, D.J. Mavriplis, „Agglomeration Multigrid for the Three-Dimensional Euler Equations“, *ICASE Report No. 94-5* (1994)

[WC84] P. Woodward, Ph. Colella, „The Numerical Simulation of Two-Dimensional Fluid Flows with Strong Shocks“, *J. Comput. Phys.* **54**, 115-173 (1984)

Zusammenfassung

In der vorliegenden Arbeit werden Finite-Volumen-Verfahren auf unstrukturierten Gittern behandelt. Dabei wird zum einen auf den Rekonstruktionsalgorithmus und zum anderen auf Beschleunigungstechniken besonders eingegangen. Beide Aspekte werden algorithmisch umgesetzt.
Beim Rekonstruktionsalgorithmus wird die polynomiale Rekonstruktion dargestellt und analysiert. Die Idee der WENO-Rekonstruktion wird vom eindimensionalen Fall auf den Fall unstrukturierter Gitter in zwei Raumdimensionen übertragen. Die resultierende Verbesserung der Ergebnisse gegenüber der (digitalen) ENO-Rekonstruktion wird demonstriert. Im glatten Bereich der Lösungen führt die WENO-Rekonstruktion zu deutlich besseren Ergebnissen als die ENO-Rekonstruktion. Gleichzeitig bleibt die Qualität der Lösungen in der Nähe von Unstetigkeiten erhalten. Für stationäre Strömungen erhält man mit dem WENO-Verfahren auch ein stationäres Verhalten der numerischen Lösung, was beim ENO-Verfahren nicht der Fall ist.
Als Beschleunigungstechniken werden die Gitteradaption, die verallgemeinerte Mehrskalenanalyse und für stationäre Strömungen die Mehrgittertechnik behandelt. Bei der Effizienzanalyse der ersten beiden Techniken erhalten wir das klare Ergebnis, dass die verallgemeinerte Mehrskalenanalyse kein adäquater Ersatz für die Gitteradaption ist: Die Rechenzeit kann mit Hilfe der verallgemeinerten Mehrskalenanalyse nur um einen beschränkten Faktor reduziert werden, während bei der Gitteradaption die Beschleunigung beliebig groß werden kann, wenn das Gitter immer feiner wird. Bei der Gitteradaption wird proportional zur Rechenzeit auch Speicherplatz eingespart, während sich der Speicherplatzbedarf bei der verallgemeinerten Mehrskalenanalyse nicht verringert.
Es wird bewiesen, dass die verallgemeinerte Mehrskalenanalyse bei der Entstehung von Unstetigkeiten versagen kann. Dieses Versagen wird an einem Beispiel demonstriert.
Die Mehrgittertechnik für stationäre Strömungen wird erläutert und an einigen Beispielen wird gezeigt, dass damit die Rechenzeiten erheblich reduziert werden können.

Lebenslauf

Name	Oliver Friedrich
Geburtsdatum	30. Januar 1968
Geburtsort	Warburg (Westf.)

Schulausbildung

8. 1974 – 7. 1978	Grundschule Liebenau
8. 1978 – 7. 1984	Gesamtschule Hofgeismar
8. 1984 – 5. 1987	Gymnasiale Oberstufenschule Hofgeismar
Abschluss	Allgemeine Hochschulreife

Grundwehrdienst

10. 1987 – 10. 1988	Bundeswehr in Warburg, Arolsen und Hofgeismar

Studium

10. 1988 – 6. 1994	Universität Göttingen
Abschluss	Diplom-Mathematiker (Nebenfach Informatik)

Beruf

4. 1991 – 6. 1994	Software-Entwicklung als Nebentätigkeit bei der Deutschen Forschungsanstalt für Luft- und Raumfahrt e.V. in Göttingen
7. 1994 – 7. 1996	Wissenschaftlicher Mitarbeiter bei der Deutschen Forschungsanstalt für Luft- und Raumfahrt e.V. in Göttingen
8. 1996 –	Wissenschaftlicher Mitarbeiter am Institut für Angewandte Mathematik der Universität Hamburg