Institut für Mathematik und Informatik

Ernst-Moritz-Arndt-Universität Greifswald

Numerische Lösung der Advektionsgleichung mittels logarithmischer Rekonstruktion

Wissenschaftliche Arbeit zur Erlangung des akademischen Grades

„Diplom-Biomathematikerin“

eingereicht von

Katharina Elsen

Gutachter

Prof. Dr. Bernd Kugelmann
Dr. Oswald Knoth

Greifswald, den 12. April 2012

Inhaltsverzeichnis

1. Einleitung

Differentialgleichungen finden ein breites Anwendungsspektrum in der Modellierung verschiedenster Naturphänomene. Mit ihrer Hilfe kann beispielsweise das Wachstum von Populationen, die Ausbreitung von Infektionen, die Bewegung von Himmelskörpern oder auch das Verhalten von Strömungen beschrieben werden. Letzteres ist Gegenstand der (numerischen) Strömungsmechanik, in der insbesondere **partielle Differentialgleichungen (PDEs)** Anwendung finden.

Eine in diesem Zusammenhang wichtige Gleichung ist die **Advektionsgleichung**. Diese ist selbst wiederum Teil der **Navier-Stokes-Gleichungen** und gehört zu den hyperbolischen PDEs. Mittels der Advektionsgleichung, die als solche einen festen Platz innerhalb der Atmosphärenphysik einnimmt, kann unter anderem den Transport von Luftbeimengungen (Aerosole) beschrieben werden.
Gesucht ist dabei eine Funktion $u(\mathbf{x}, t)$ (diese beschreibt z.B. Dichte oder Konzentration eines solchen Aerosols), die folgende Gleichung (Advektionsgleichung) erfüllt:

$$\frac{\partial}{\partial t} u(\mathbf{x}, t) + \frac{\partial}{\partial \mathbf{x}} (\mathbf{v}(\mathbf{x}, t) u(\mathbf{x}, t)) = 0.$$

Dabei ist $\mathbf{v}(\mathbf{x}, t)$ ein gegebenes Geschwindigkeitsfeld.
Da Dichte, Masse oder auch Stoffkonzentrationen nicht negativ werden können, muss für eine physikalisch richtige Lösung immer gelten:

$$u(\mathbf{x}, t) \geq 0,$$

d.h. die Positivität der Lösung muss gegeben sein.
Wie für hyperbolische PDEs typisch, treten auch in den Anwendungsbeispielen der Advektionsgleichung häufig starken Gradienten auf, beispielsweise an Wolkenrändern oder in der Umgebung von Punktquellen. Diese Gradienten sollen in der Modellierung möglichst gut erhalten werden. Wie für die meisten Differentialgleichungen ist auch für die allgemeine Advektionsgleichung keine explizite Lösung bekannt. Dies gilt allerdings nicht für den Spezialfall der (diffusi-

onsfreien) linearen Advektion:

$$\frac{\partial}{\partial t}u(\mathbf{x},t) + \frac{\partial}{\partial \mathbf{x}}(\mathbf{v} \cdot u(\mathbf{x},t)) = 0,$$

mit konstanter Geschwindigkeit $\mathbf{v} = const.$ und Anfangswert $u(\mathbf{x}, 0) = u_0(\mathbf{x})$. Die Lösung lautet in diesem Fall

$$u(\mathbf{x},t) = u_0(\mathbf{x} - \mathbf{v}t).$$

Dies kann man sich vorstellen als eine Welle der Form $u_0(\mathbf{x})$, die sich mit konstanter Geschwindigkeit fortbewegt. Betrachtet man diese Welle auf einer Kugeloberfläche, bzw. einem Gebiet mit periodischen Randbedingungen, so gilt nach 2π Zeiteinheiten:

$$u(\mathbf{x}, 2\pi) = u(\mathbf{x}, 0).$$

Diese Tatsache macht man sich in der Validierung numerischer Verfahren zu Nutze. Anhand der "Differenz" zwischen der exakten und der numerischen Lösung lässt sich verhältnismäßig einfach die Güte einer Methode testen. Diese "Differenz" kann durch verschiedene Fehlernormen bestimmt werden. Die Gebräuchlichsten sind dabei der L_1-, L_2- und der L_∞-Fehler. Durch Verfeinerung des Gitters kann mithilfe dieser Fehlernormen zusätzlich die Konvergenzordnung ermittelt werden.
Des Weiteren stehen zur Validierung numerischer Lösungsverfahren in der Literatur verschiedenste Testfälle komplexerer (Anwendungs-)Beispiele zur Verfügung mit Hilfe derer die Lösungen unterschiedlicher Verfahren miteinander verglichen werden können.

Mittlerweile gibt es – auch dank der rasanten Entwicklung in der Computertechnik – eine ganze Reihe etablierter Verfahren auch zur Lösung hyperbolischer Differentialgleichungen, insbesondere der Advektionsgleichung. Zu den Standard-Verfahren gehören dabei unter anderem:

- Flux-Limiter-Methoden
- ENO/Weno-Verfahren

Auf die Flux-Limiter-Verfahren wird in Kapitel 4 noch genauer eingegangen werden.

All diese Verfahren beruhen auf der Idee aus den gegebenen (dikreten) Daten den tatsächlichen Verlauf der Funktion zu rekonstruieren. Dies gelingt umso besser, je genauer die der Rekonstuktion zugrunde liegende Funktion in der Lage ist, den realen Funktionsverlauf zu approximieren. Dazu gehört unter anderem auch die Vermeidung unnatürlicher Oszillationen. Oder, wie Rice [5] es bereits 1964 in *"The Approximation of Functions"* beschrieben hat:

> *"The key to efficient approximation is to find an approximation function which can take on the same nature or behaviour as $f(x)$."*
> (zitiert nach [1])

In diesem Zusammenhang wurden weitere Verfahren entwickelt, die anstatt auf einen polynomialen Ansatz auf *parabolische* [2] beziehungsweise *logarithmische* [1] Funktionen zurückgreifen.
Eines dieser Nicht-Standard-Verfahren ist die **Local Double Logarithmic Reconstruction (LDLR)** [1]. Dieses Verfahren bietet unter anderem eine Limiter-freie Rekonstruktion, deren lokale Variation (vgl. Kapitel 4) aber dennoch beschränkt ist, d.h. sie neigt nicht zu unerwünschtem Oszillationsverhalten. Das Verfahren ist allerdings auf logische Rechteck- und Dreiecksgitter beschränkt.

Das Ziel dieser Arbeit ist die Entwicklung eines auf der LDLR aufbauenden Finite-Volumen-Verfahrens zur numerischen Lösung der Advektionsgleichung auf beliebigen unstrukturierten Gitter. Die allgemeine Form der rekonstruierenden Funktion soll dabei von der Geometrie des Gitters unabhängig sein soll, während die entscheidenden Eigenschaften der LDLR erhalten bleiben. Dazu soll mittels logarithmischer Ansatzfunktionen $\varphi : \mathbb{R}^n \to \mathbb{R}$ eine Funktion $r : \mathbb{R}^n \to \mathbb{R}$, $n \in \{1, 2, 3\}$ berechnet werden anhand derer wiederum die benötigten numerischen Flüsse bestimmt werden können.
Die Motivation für eine Erweiterung der LDLR auf unstrukturierte (nicht-rechteckige) Gitter liegt auf der Hand. Unstrukturierte Gitter sind in der Diskretisierung komplexerer Geometrien oder gekrümmter Oberflächen wie der Sphäre unabdingbar. Darüber hinaus werden sie auch für die Darstellung der Orographie und im Rahmen von Gitterverfeinerungen benötigt. Auf Rechteckgittern wäre dies nicht, beziehungsweise nur schwer möglich.

In **Kapitel 2** soll im Folgenden zunächst auf das Modellproblem – die Advektionsgleichung – und seine Herleitung aus den Erhaltungssätzen eingegangen werden.

Anschließend werden in **Kapitel 3** die zur räumlichen Diskretisierung benötigten Gitter, die verschiedene Gittertypen sowie ihre Vor- und Nachteile erläutert. Außerdem sollen kurz die benötigten Bezeichnungen eingeführt und die entsprechende Implementierung beschrieben werden.

In **Kapitel 4** werden zunächst verschiedene Verfahren zur Berechnung der Flüsse im Rahmen Finiter-Volumen-Verfahren dargestellt. Dies sind im einzelnen das Upwind-Verfahren, Flux-

Limiter-Verfahren sowie die ebenfalls bereits erwähnte Local-Double-Logarithmic-Reconstruction.

Schließlich wird in **Kapitel 5** die oben erwähnte Methode zur Rekonstruktion auf unstrukturierten Gittern erläutert. Dazu wird zunächst die logarithmische Ansatzfunktion und anschließend die rekonstruierende Funktion hergeleitet. Davon ausgehend werden die Funktionsparamter bestimmt und eine Möglichkeit zur Integration der Funktion vorgestellt. Im Anschluss sollen die Eigenschaften der Rekonstruierenden verifiziert werden.

In **Kapitel 6** soll dann die neu entwickelte Methode in verschiedenen Test validiert und für den Fall eines Rechteckgitters mit der Local Double Logarithmic Rekonstruktion (LDLR) von R.Artebrant verglichen werden.

Abschließend soll in **Kapitel 7** ein Zusammenfassung der Ergebnisse sowie ein kurzer Ausblick auf die Erweiterung der Methode auf drei Dimensionen gegeben werden.

2. Modellproblem

Sämtliche Verfahren werden im Folgenden am Modellproblem der ein- bzw. zweidimensionalen Advektion getestet (vgl. Gleichung 2.2.6 und 2.2.7). Es handelt sich hierbei um eine hyperbolische Differentialgleichung. Eine typische Eigenschaft hyperbolischer partieller Differentialgleichungen besteht im Auftreten von Schocks beziehungsweise Kontaktunstetigkeiten. Solche Funktionen können aber nicht Lösung einer partiellen Differentialgleichung sein, da an diesen Stellen keine Ableitungen definiert sind. Zur Lösung dieser Gleichungen greift man daher häufig auf die sogenannten Integralgleichungen zurück. Für die hier verwendeten Finite-Volumen-Verfahren werden die Partiellen Differentialgleichungen also in ihrer integralen Form benötigt. Die benötigten Gleichungen werden dazu aus den Erhaltungsgleichungen hergeleitet.

2.1. Erhaltungsgleichungen

Diese Erhaltungsgleichungen haben die allgemeine Form:

$$u_t + \nabla \cdot f(u) = 0 \tag{2.1.1}$$

Die zugrundeliegende Erhaltungsgleichung der Advektion ist die Massenerhaltung. Sei M ein spezifischer Stoff mit Dichte ρ. Dann lautet die entsprechende Erhaltungsgleichung in differentieller Form:

$$\rho_t + \nabla \cdot (\rho v) = 0. \tag{2.1.2}$$

Diese besagt, dass eine Änderung der Masse m des betrachteten Stoffes M innerhalb eines definierten (zeitinvarianten) Volumens Ω, dem sogenannten Kontrollvolumen, nur stattfinden kann, wenn es einen Fluss über seinen Rand gibt; sie kann weder entstehen noch verschwinden. Die Masse m eines Stoffes innerhalb eines solchen Volumens Ω kann dabei als das Integral der Dichte ρ des betrachteten Stoffes über diesem Volumen betrachtet werden. Masseerhaltung ist in diesem Zusammenhang also gleichbedeutend mit Dichteerhaltung. Dem Nettofluss aus diesem Volumen entspricht das Integral der Flüsse f in Normalenrichtung über seinen Rand $\partial\Omega$. Für

eine Geschwindigkeit v eines kleinen Massevolumens wird $\rho v = f_m$ auch als *Konvektionsterm* bezeichnet. Entsprechend lautet deshalb die integrale Form der Massenerhaltung:

$$\frac{\partial}{\partial t}\int_{\Omega}\rho d\Omega = -\int_{\partial\Omega} f_m \cdot \vec{n}\, dS \tag{2.1.3}$$

wobei ρ und f_m als stetig differenzierbar angenommen werden.
Unter der Annahme der stetigen Differenzierbarkeit und falls das Kontrollvolumen Ω zeitinvariant ist, lassen sich nun Integration und Differentation vertauschen und mit dem Satz von Gauß gilt dann:

$$\int_{\Omega}\frac{\partial}{\partial t}\rho d\Omega = -\int_{\Omega}\nabla\cdot f_m d\Omega \;\Leftrightarrow\; \int_{V}\rho_t + \nabla\cdot f_m dV = 0. \tag{2.1.4}$$

Unter obiger Annahme der stetigen Differenzierbarkeit ist also die integrale Form der Massenerhaltung äquivalent mit der differentiellen Form der Massenerhaltung [6].

2.2. Advektionsgleichung

Für die Advektionsgleichung betrachten wir nun die Konzentration c eines Stoffes in einem Trägermedium. Diese Konzentration lässt sich mithilfe seiner Dichte ρ und dem Mischungsverhältnis zwischen Stoff und Trägermedium μ berechnen. Es gilt: $c = (\rho \cdot \mu)$. Gegeben sei im Folgenden ein Gebiet Ω mit einer Zerlegung $\Omega = \{\Omega_1, \Omega_2, ..., \Omega_n\}$ und für jede Gitterzelle $\Omega_{i,\ i=1,...,n}$ (Kontrollvolumen) der zugehörige Mittelwert $c(x,t)_i$ sowie der Geschwindigkeitsvektor $\mathbf{v}$. Die Advektionsgleichung sagt nun nichts weiter, als dass die zeitliche Änderung der Konzentration c innerhalb eines Kontrollvolumens äquivalent mit der Summe der Nettoflüsse $f_m = (vc)$ bzw. $f_m = (\rho v \mu)$ über seinen Rand ist. Dies lässt sich folgendermaßen formulieren:

$$\frac{\partial(\rho\mu)}{\partial t} = -\nabla(\rho v \mu) \tag{2.2.1}$$

bzw.:

$$\frac{\partial c}{\partial t} = -\nabla(vc). \tag{2.2.2}$$

Unter der Annahme, dass die Kontrollvolumina Ω_i zeitinvariant sind, lässt sich dies wieder in der integralen Form formulieren:

$$\int_{\Omega_i} \frac{\partial c}{\partial t} = -\int_{\Omega_i} \nabla(vc), \tag{2.2.3}$$

mit den Anfangs- und Randwerten

$$c(\mathbf{x}, 0) = c_0(\mathbf{x}) \tag{2.2.4}$$

$$c(\mathbf{x}, t) = c(t), \ \forall \mathbf{x} \in \partial\Omega. \tag{2.2.5}$$

Für die numerischen Tests (vgl. Kapitel 6) werden später – wie bereits oben erwähnt – die ein- und zweidimensionale Advektionsgleichung benötigt. Diese sind für eine gegebene Stoffkonzentration $c(\mathbf{x}, t)$ – je einmal in ihrer differentiellen und integralen Form – in den Gleichungen (2.2.6) bzw. (2.2.7) dargestellt.

Eindimensionale Advektionsgleichung:

$$\begin{aligned} c(x,t)_t &= -\nabla(v(x,t)c(x,t)), \\ \int_{\Omega_i} \frac{\partial}{\partial t} c(x,t) &= -\int_{\Omega_i} \nabla(v(x,t)c(x,t)). \end{aligned} \tag{2.2.6}$$

Zweidimensionale Advektionsgleichung:

$$\begin{aligned} c(x,y,t)_t &= -\nabla(v(x,y,t)c(x,y,t)), \\ \int_{\Omega_i} \frac{\partial}{\partial t} c(x,y,t) &= -\int_{\Omega_i} \nabla(v(x,y,t)c(x,y,t)). \end{aligned} \tag{2.2.7}$$

Die Anfangs- und Randwerte lauten für beide Fälle wie in (2.2.4) bzw. (2.2.5).

3. Gitter

Für die numerische Diskretisierung muss das Rechengebiet Ω zunächst in kleinere Teilgebiete Ω_k zerlegt werden. Das heißt:

$$\bigcup_{k=1}^{m} \Omega_k = \Omega$$

mit

$$\Omega_k \cap \Omega_l = \emptyset \; \forall \; k, l \in \{1, \ldots, m\}.$$

Es sollen im Weiteren ausschließlich Zerlegungen in konvexe Polygone betrachtet werden. Die einzelnen Gitterzellen Ω_k bilden dabei die bereits erwähnten Kontrollvolumina des sogenannten **primalen Gitters** Ω.

Im folgenden Abschnitt soll zunächst eine allgemeine Notation für diese Gitterzellen vereinbart und einige wichtige Kenngrößen zu deren Beschreibung definiert und berechnet werden. Anschließend wird auf die Eigenschaften und Anwendungsgebiete strukturierter bzw. unstrukturierter Gitter, deren Vor- und Nachteile sowie den Sonderfall logischer Rechteckgitter eingegangen.

3.1. Polygonale Gitterzellen

Wir wollen nun eine polygonale Gitterzelle $\Omega_k \subset \Omega$ betrachten. Konvexe Polygone sind durch ihre Eckpunkte (Knoten) $(\mathbf{v}_i^k,\ i = 1, ..., n)$ eindeutig definiert mit n = #Knoten von Ω_k:

$$\Omega_k = [\mathbf{v}_1^k, \mathbf{v}_2^k, ..., \mathbf{v}_n^k], \text{ bzw. kürzer } \Omega_k = [\mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_n].$$

Hierbei seien $\mathbf{v}_1\mathbf{v}_2$, $\mathbf{v}_2\mathbf{v}_3$,...,$\mathbf{v}_m\mathbf{v}_1$ die Kanten des Polygons und der Durchlauf der Punkte entgegen dem Uhrzeigersinn. Zur Vereinfachung der Darstellung gelte die zusätzliche Bezeichnung $\mathbf{v}_0 = \mathbf{v}_n$. Für die weiteren Betrachtungen benötigen wir zusätzliche Kenngrößen des Polygons, welche aus den gegebenen Eckpunkten berechnet werden können. Insbesondere sei x_M der baryzentrische Mittelpunkt einer Gitterzelle:

$$\mathbf{x}_M = \frac{1}{n} \sum_i \mathbf{v}_i.$$

Die Kanten des Polygons seien mit $\mathbf{e}_i = \overline{\mathbf{v}_{i-1}, \mathbf{v}_i},\ i = 1, ..., n$ bezeichnet. Mit jeder Kante $\mathbf{e}_i$ verbinden wir ihren Mittelpunkt $\mathbf{x}_i$, sowie deren Normale $\mathbf{n}_i$ und Tangente $\mathbf{t}_i$. Soweit nicht anders angegeben, handelt es sich bei den Normalen und Tangenten immer um die auf *eins* normierten Vektoren. Die Normalen sind bezüglich der Gitterzelle nach außen orientiert. Die Orientierung der Tangenten erfolgt gegen den Uhrzeigersinn. Vergleiche dazu auch Abb. (1.1). Weiterhin bezeichne A die Fläche des Polygons und h_i die Länge der Kante $\mathbf{e}_i$. Außerdem sei für jede dieser Kanten der Abstand zwischen dem Zellmittelpunkt und dem Kantenmittelpunkt mit h_i^d bezeichnet.

An dieser Stelle soll kurz auf die verwendeten Datenstrukturen zur Speicherung derartiger Gitter

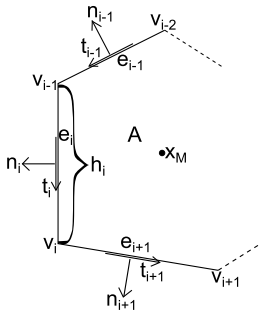

Abbildung 1.1.: Ausschnitt einer primalen Gitterzelle

eingegangen werden. Neben den oben eingeführten Größen, müssen diese auch die notwendigen Nachbarschaftsrelationen berücksichtigen. In Fortran2000 nutzen wir dazu die Typen **Cell_T** (primale Zellen), **Edge_T** (Kanten) und **Node_T** (Knoten). Koordinatenpunkte sind im Typen (**Point_T**) definiert. Dabei bezeichnet (**Point_T**) ein Tupel $\mathbf{x} = (x, y)$ (bzw. im dreidimensionalen Fall ein Tripel $\mathbf{x} = (x, y, z)$) (vgl. 3.1). Die Typen $Cell\_T$, $Edge_T$ und $Node_T$ besitzen ihrerseits wieder verschiedene Instanzen (vgl. (3.1)), insbesondere kann jeder (primalen und dualen) Zelle, jeder Kante und jedem Knoten ein eindeutiger Index (z.b. *C_number*) zugeordnet werden. All diese Beziehungen werden auch als Nachbarschaftsrelationen bezeichnet.

Die Indizes aller primalen Gitterzellen vom Typen $Cell\_T$ sind nun im Vektor $Cells$ gespeichert. Analog dazu existieren auch die Felder $Edges$ und $Nodes$ für die Kanten- und Knotenindizes. Diese Felder sind im Typen $\mathbf{Grid_T}$, der dem Gitter Ω entspricht zusammengefasst.

Möchte man nun beispielsweise auf die x-Koordinate der Normale n_2 der Gitterzelle Ω_{10} zugreifen, so erhält man zunächst den Index der zugehörigen Kante e_2 aus der Kantenliste (Instanz) der betrachteten Gitterzelle (Typ). Mit diesem Index kann man wiederum im Typen $Edge\_T$ auf die (Instanz) Normale zugreifen. Diese ist wiederum vom Typen $Point\_T$, so dass man letzendlich

den entsprechenden Vektor (x, y) erhält. Auf die Instanz "Neighbour" wird später noch genauer eingegangen.

Listing 3.1: Fortran Sourcecode: Datenstruktur zur Speicherung des Gitters

```
TYPE pCell_T
  INTEGER                          :: pC_number
  REAL(KIND=8)                     :: area
  TYPE(Point_T)                    :: mid
  TYPE(Point_T),DIMENSION(4)       :: neighbour
  REAL(KIND=8),DIMENSION(4)        :: hd
  INTEGER,DIMENSION(4)             :: Edge
  INTEGER,DIMENSION(4)             :: Node
  INTEGER                          :: material !handelt es sich um eine Randzelle oder innere Zelle
END TYPE

TYPE dCell_T
  INTEGER                          :: dC_number
  REAL(KIND=8)                     :: area
  REAL(KIND=8),DIMENSION(4,4)      :: T !T-Matrix zur Gradientenberechnung mittels MPFA-Methode
  INTEGER,DIMENSION(4)             :: pCell
END TYPE

TYPE Edge_T
  INTEGER,DIMENSION                :: E_number
  TYPE(Point_T)                    :: normal,tangential,x_bar !Kantenmittelpunkt
  REAL(KIND=8)                     :: h
  INTEGER,DIMENSION(2)             :: Node
  INTEGER                          :: material
END TYPE

TYPE Node_T
  INTEGER,DIMENSION                :: N_number
  TYPE(Point_T)                    :: P
END TYPE

TYPE(pCell_T),POINTER,DIMENSION(:)   :: pCells
TYPE(dCell_T),POINTER,DIMENSION(:)   :: dCells
TYPE(Edge_T),POINTER,DIMENSION(:)    :: Edges
TYPE(Node_T),POINTER,DIMENSION(:)    :: Nodes
```

3.2. Unstrukturierte Gitter

Klassische Beispiele unstrukturierter Gitter in der Ebene sind Dreicksgitter und logische Rechteckgittergitter. Während bei ersterem das gesamte Gitter mit Dreiecken ausgefüllt ist, sind dies im zweiten Fall Vierecks. Das bedeutet, dass alle Gitterzellen die gleiche Anzahl Knoten besitzen. Dies gilt beispielsweise nicht für mittels Voronoi-Zerlegung (vgl. Abb. (2.1)) generierte Gitter. Dreiecksgitter bieten aufgrund ihrer Flexibilität große Vorteile in der Diskretisierung komplexer Geometrien und sind Standard im Rahmen von Gitterverfeinerungen (vgl. Abb. (3.1)). Wie bereits offensichtlich geworden ist, sind sie bezüglich ihrer Erstellung und Speicherung mit einem nicht unerheblichen Aufwand verbunden. Unter anderem ist der Zugriff auf Nachbarzellen nicht mehr trivial, so dass zusätzlich eine ganze Reihe an Nachbarschaftsrelationen

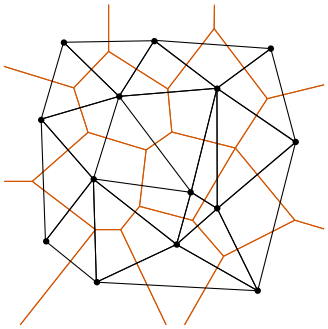

Abbildung 2.1.: Durch Voronoi-Zerlegung erzeugtes Gitter

abgespeichert werden muss.

Für die Generierung von Dreiecksgittern gibt es eine Vielzahl von Programmen. Neben der

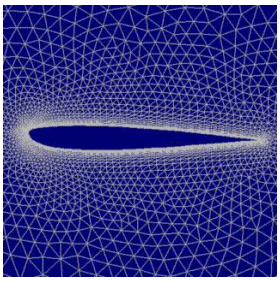

Abbildung 2.2.: Unstrukturiertes Dreiecksgitter mit Gitterverfeinerung zur Modellierung eines Tragflügels

guten Approximation des Randes des Gebiets sollen die generierten Dreiecke einer gewissen ”Güte” genügen. Mögliche Kriterien für diese sind: Nicht zu spitze Winkel, gleichmäßige Form, innenliegender Umkreismittelpunkt. Derartige Eigenschaften führen zu besseren Approximationseigenschaften zu diskretisierender Differetialoperatoren oder möglichst großer Zeitschrittweiten bei expliziten Zeitintegrationsverfahren. So würde etwa das rechte Dreieck in Abb. (2.3) zu extrem kleinen Zeitschritten führen. Gegenüber Dreiecksgittern bieten logische Rechteckgitter einen verhältnismaßig einfachen Zugriff auf Nachbarzellen. Man spricht dabei von einem logische Rechteckgitter, wenn man die Knoten des Gitters auf ein kartesisches Gitter abbilden kann. Dies ermöglicht eine Traversierung mittels einfacher Schleifenkonstrukte. Der Zugriff auf Nachbarzellen ist durch einfache Indexrechnung möglich. Diese Gitter zeichnen sich durch einen geringeren Erstellungs- und Verwaltungsaufwand aus, sind aber schwerer adaptierbar und auf verhältnismäßig einfache Geometrien beschränkt [6]. Ein typisches Beispiel logischer Recht-

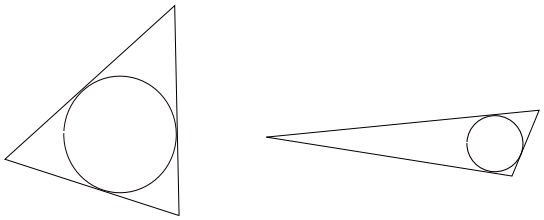

Abbildung 2.3.: Mögliche Gitterweiten bei wenig und stark verzerrten Gitterzellen

eckgitter sind sogenannte **bodenfolgende Koordinaten**. Numerische Modelle zur Wettervorhersage nutzen derartige Koordinaten zur Berücksichtigung von Orographie (vgl. Kapitel 1 und Abb. (3.1)). In diesem Zusammenhang sei erwähnt, dass spezielle Voronoi-Triangulierungen

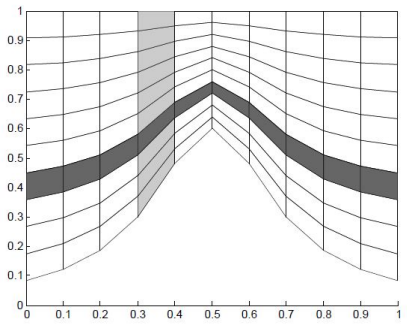

Abbildung 2.4.: Gitter mit bodenfolgenden Koordinaten

der Sphäre als mögliche Gitter für die nächste Generation von Wettervorhersage-Modellen in Betracht gezogen werden. Insbesondere erlaubt es diese Art der Triangulierung Gebiete mit verschiedener räumlicher Auflösung zu erstellen. Die dabei erzeugten polygonalen Zellen sind im allgemeinen Sechsecke, es treten vereinzelt aber auch Fünf- und Siebenecke auf (vgl. Abb. (**??**)). (In Ausblick Adaptierung auf diese Gitter beschreiben.)

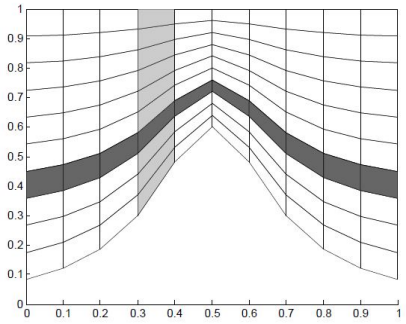

Abbildung 2.5.: Voronoi-Triangulierung für die Sphäre

4. Finite-Volumen-Verfahren: Rekonstruktion und Flussberechnung

Neben Finite-Differenzen-Verfahren (FDM) und Finite-Elemente-Verfahren (FEM) gehören die Finite-Volumen-Verfahren zu den Standardverfahren in der numerischen Strömungsmechanik. Sie beruhen auf einer direkten Approximation der integralen Erhaltungsgleichung (vgl. Kapitel 2), bei der der approximierte Funktionsverlauf die Differentialgleichung innerhalb einer Gitterzelle im Mittel erfüllt. Es ist dabei nicht gesagt, an welcher Stelle die Differentialgleichung erfüllt ist, die Abweichungen innerhalb der Gitterzelle summieren sich jedoch zu *null*.
Genau diese Eigenschaft macht man sich nun in der Diskretisierung der Advektionsgleichung (bzw. allgemein vor allem hyperbolischer DGL) zu Nutze. Dazu wird die PDE als Erhaltungsgleichung formuliert und anschließend mittels Finiter-Volumen-Verfahren diskretisiert.
Durch Integration der Erhaltungsgleichungen über den einzelnen Gitterzellen – den sogenannten Kontrollvolumina – entstehen Bilanzgleichungen, die eine konservative Diskretisierung gewährleisten [1]. Veraussetzung dafür ist jedoch, dass die Kontrollvolumina zeitinvariant sind.
Im Gegensatz zu den Finite-Differenzen-Verfahren können Finite-Volumen-Verfahren nicht nur leicht auf beliebige Gitter angewandt werden, auch große Gradienten bzw. Unstetigkeitsstellen – typisch für hyperbolische PDEs – stellen, wie bereits erwähnt, für sie kein Problem dar. Im Vergleich zu Finite-Elemente-Verfahren sind sie außerdem leichter zu implementieren [6].

Das Verfahren lässt sich allgemein in 3 Schritte unterteilen [6]:

Schritt 1: Räumliche Diskretisierung

Schritt 2: Rekonstruktion der Daten auf dem Rand des Gebietes

Schritt 3: Berechnung der Flüsse

Wie bereits in der Einleitung erwähnt ist das Oszillationsverhalten ein wichtiges Kriterium für Lösungsverfahren insbesondere hyperbolischer partieller Differentialgleichungen (PDEs). Hy-

[1] Strömungsmechanik: Grundlagen- Grundgleichungen- Lösungsmethoden

perbolische PDEs weisen typischerweise nur eine geringe bis keine Dämpfung auf. Dieses Verhalten soll auch in der numerischen Lösung erhalten werden. Das Ziel ist also Verfahren zu konstruieren, die keine übermäßig starke Dämpfung aufweisen, dabei aber kein unnatürliches Oszillationsverhalten zeigen.
Im Folgenden soll zunächst eine diskretisierte Form der Advektionsgleichung hergeleitet und die für explizite Zeitintegrationsverfahren nötige CFL-Bedingung erläutert werden.
Anschließend werden drei Verfahren näher erläutert und ihre Vor- und Nachteile aufgezeigt werden. In diesem Zusammenhang werden außerdem Definitionen gegeben, mit deren Hilfe das Oszillationsverhalten von Funktionen gemessen werden kann.

4.1. Räumliche Diskretisierung

Für die numerische Lösung der Advektionsgleichung wird zunächst eine Diskretisierung des gegebenen Problems benötigt. Wir betrachten im Folgenden die Advektionsgleichung

$$\frac{\partial c}{\partial t} = -\nabla(vc), \tag{4.1.1}$$

über einem Gebiet Ω. $\mathbf{v}(\mathbf{x}, \mathbf{t})$ sei ein gegebenes Geschwindigkeitsfeld. c bezeichne die (unbekannte) Funktion einer Stoffkonzentration.
Sei nun $\Omega = [\Omega_1, \Omega_2, ..., \Omega_m]$ eine gegebene zeitinvariante Zerlegung des betrachteten Gebietes und sei n_k die Anzahl der Knoten von Ω_k. Durch Integration von (4.1.1) über den Kontrollvolumina Ω_k erhält man dann:

$$\frac{d}{dt}\bar{c}_k(t) = -\frac{1}{|\Omega_k|}\int_{\Omega_k} \nabla(vc) d\Omega_k. \tag{4.1.2}$$

Dabei bezeichnet $\bar{c}_k(t)$ den Zellmittelwert der Konzentration c über dem Kontrollvolumen Ω_k:

$$\bar{c}_k(t) = \frac{1}{|\Omega_k|}\int_{\Omega_k} c_k d\Omega.$$

Damit lautet (4.1.1) in diskretisierter Form:

$$\int_{\Omega_k} \nabla(vc) d\Omega_k = \sum_{j=1}^{n_k} \hat{f}_j(c, \nabla c) \tag{4.1.3}$$

wobei die $\hat{f}_j$ die numerisch berechneten Flüsse über die n_k Kanten der Gitterzelle Ω_k sind.

4.2. CFL-Bedingung

Bei der numerischen Lösung zeitabhängiger PDEs ist die Stabilität expliziter Zeitintegrationsverfahren entscheidend abhängig von der Wahl des Zeitschritts $\triangle t$. Dieser lässt sich für eine gegebene Gitterweite $\triangle x$ und einen Geschwindigkeitsvektor v prinzipiell berechnen, als:

$$\triangle t = \frac{\triangle x}{v}.$$

In der Praxis wird dies meist noch mit einem Faktor, der sogenannten CFL-Zahl multipliziert. Für den Zeitschritt gilt dann mit $CFL > 0$:

$$\triangle t = CFL \cdot \frac{\triangle x}{v}. \tag{4.2.1}$$

So ist beispielsweise das explizite Euler-Verfahren nur für $CFL < 1$ stabil. Im allgemeinen Fall muss dieser Faktor im numerischen Experiment ermittelt werden. [6]
Da häufig die Geschwindigkeit v und auf nicht-kartesischen Gittern auch $\triangle x$ nicht konstant sind, muss für jeden Iterationsschritt die maximale Zeitschrittweite $\triangle t^{global}$ berechnet werden. Dazu wird auf die in Kapitel 3 beschriebene duale Halbkanten-Länge zurückgegriffen und für jede Gitterzelle Ω_k die Gitterweite $\triangle x_k$ folgendermaßen bestimmt:

> *Sei $h_i^d,\ i = 1, ...n$ für jede Kante $\mathbf{e}_i$ einer (polygonalen) Gitterzelle Ω_k mit n Kanten die Distanz zwischen dem Kantenmittelpunkt $\mathbf{x}_i$ und dem Zellmittelpunkt $\mathbf{x}_M$. Sei h_{min}^d das Minimum aus diesen Distanzen, also*
>
> $$h_{min}^d = \min_i (h_i^d | i = 1, ..., n).$$
>
> *Dann wird die Gitterweite $\triangle x_k$ wie folgt definiert: $\triangle x_k = 2 \cdot \hat{h}^d$.*

Sei weiter v_k^{max} die betragsmäßig größte Geschwindigkeit bezüglich der Gitterzelle Ω_k. Dann gilt:

$$\triangle t^{global} = CFL \cdot \min_k \left(\frac{\triangle x_k}{v_k^{max}} \right). \tag{4.2.2}$$

4.3. Upwind-Verfahren erster Ordnung

Dieses Verfahren entspricht einer stückweise konstanten Rekonstruktion. Hierbei werden die Flüsse lediglich durch den Mittelwert der aktuell betrachteten Zelle approximiert, also:

$$\hat{f}_{i+1/2} = \bar{c}_i, \tag{4.3.1}$$

falls $u > 0$. In Abbildung (3.1) ist die gegebene Funktion (hier schwarz) sowie die Mittelwerte (blaue Punkte) über den eizelnen Gitterzellen (Gitterweite $h = 1$) und die Rekonstruktionen (rot) abgebildet.

Das Upwind-Verfahren ist zwar sehr einfach und gewährleistet außerdem Positivität, es ist aber

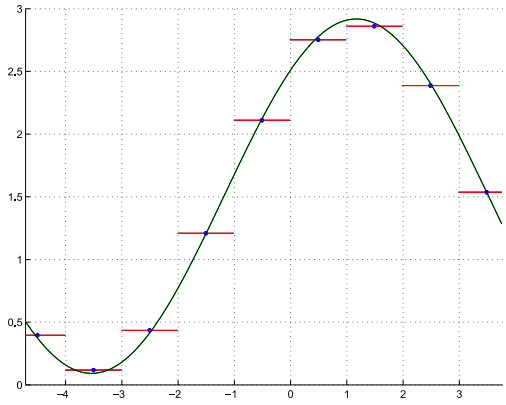

Abbildung 3.1.: Upwind-Verfahren: Stückweise konstante Rekonstruktion

auch sehr diffusiv, da die tatsächlichen Flüsse nur sehr ungenau approximiert werden (vgl. auch Abb.(3.1)).

Dieses Verfahren lässt sich auch allgemeiner definieren, als:

$$\hat{f}_{i+1/2} = max(v, 0) \cdot \bar{c}_i + min(v, 0) \cdot \bar{c}_{i+1}. \tag{4.3.2}$$

Analog zu (4.3.1) lässt sich auch das Downwind-Verfahren für den Fall $u < 0$ definieren:

$$\hat{f}_{i-1/2} = \bar{c}_i. \tag{4.3.3}$$

[4] Um die Güte der numerischen Lösung zu erhöhen, wurden verschiedene weitere Verfahren entwickelt. Dazu gehören unter anderem die Flux-Limiter-Verfahren.

4.4. Flux-Limiter-Verfahren

Die Idee der Flux-Limiter-Verfahren ist es Verfahren verschiedener Ordnung in einer Methode zu vereinen. Dazu werden die numerischen Flüsse als gewichtete Summe mehrerer Flüsse berechnet, welche einmal mit einem (monotonen) Schema erster Ordnung und einmal mit einem Schema höherer Ordnung berechnet wurden. Der Limiter berechnet dabei die Gewichte mit denen die verschiedenen Flüsse in die Summe eingehen. So kann bei genügend glattem Funktionsverlauf mit (räumlichen) Ableitungen hoher Ordnung gerechnet werden, während in der Umgebung von Sprungstellen (Schocks) mit erster Ordnung gerechnet wird und damit Oszillationen vermieden werden.
Der Limiter ist dabei eine Funktion der lokalen Lösung des vorhergehenden Zeitschrittes und soll so gewählt sein, dass die Approximation bezüglich einer eindimensionalen Erhaltungsgleichung in ihrer Totalen Variation beschränkt ist. Dabei heißt ein Verfahren *total variation nonincreasing* bzw. *total variation diminishing (TVD)*, falls gilt:

$$TV(f^{n+1}) \leq TV(f^n).$$

Diese Bedingung ist eine Abschwächung der Forderung, dass ein Verfahren monoton sein soll. Zusätzlich soll das Schema – außer in der Umgebung lokaler Extrema – mindestens zweiter Ordnung sein.[3]
Prinzipiell lassen sich diese Verfahren auch auffassen, als ein Upwind-Schema mit einem zusätzlichen Korrekturterm höherer Ordnung, z.B.:

$$\begin{aligned}\hat{f}_{i+1/2} =& v(c_j + \frac{1}{2}(c_{j+1} - c_j)) \\ =& \frac{1}{2}v(c_j + c_{j+1}) \quad \text{second-order central flux,}\end{aligned} \tag{4.4.1}$$

oder

$$\begin{aligned}\hat{f}_{i+1/2} =& v(c_j + (\frac{1}{3} + \frac{1}{6}r_j)(c_{j+1} - c_j)) \\ =& \frac{1}{6}v(-c_{j-1} + 5c_j + 2c_{j+1}) \quad \text{third-order upwind-biasd flux.}\end{aligned} \tag{4.4.2}$$

Dabei beschreibt r_j das Verhältnis der Ableitungen an den beiden Zellkanten:

$$r_j = \frac{c_j - c_{j-1}}{c_{j+1} - c_j}.$$

Diese Schemata führen im Falle vorhandener starker Gradienten allerdings zu unerwünschtem Oszillationsverhalten. Es ist daher nötig, den ”Korrekturterm” zu limitieren. Diese sogenannten

Tabelle 4.1.: Mögliche Flux-Limiter Funktionen, vgl. Abb.(4.1)

- "minmod" (Roe,1986)
  $\psi(r) = \max[0, \min(1, r)]$
- "superbee" (Roe,1986)
  $\psi(r) = \max[0, \min(1, 2r), \min(2, r)]$
- "van Leer" (van Leer,1974)
  $\psi(r) = \frac{r+\|r\|}{1+\|r\|}$
- "monotonized centered (MC)" (van Leer,1977)
  $\psi(r) = \max[0, \min(2r, \frac{1+r}{2}, 2)]$
- "Koren" (Koren,1993)
  $\psi(r) = \max[0, \min(2r, (1 + 2r)/3, 2)]$
- "Osher" (Chatkravathy und Osher,1983)
  $\psi(r) = \max[0, \min(r, \beta)], \quad (1 \leq \beta \leq 2)$

Flux-Limiter-Verfahren lassen sich in allgemeiner Form folgendermaßen aufschreiben:

$$\hat{f}_{i+1/2} = v[c_j + \psi(r_j)(c_{j+1} - c_j)], \quad v \geq 0. \tag{4.4.3}$$

Dabei ist $\psi(r_j)$ ein multiplikativer Limiter. Dieser ist maßgeblich für die Güte des Verfahrens verantwortlich [4]. In (4.1) sind einige dieser Limiter zusammengestellt und in (4.1) inclusive ihrem *2nd-order TVD*-Bereich grafisch dargestellt.

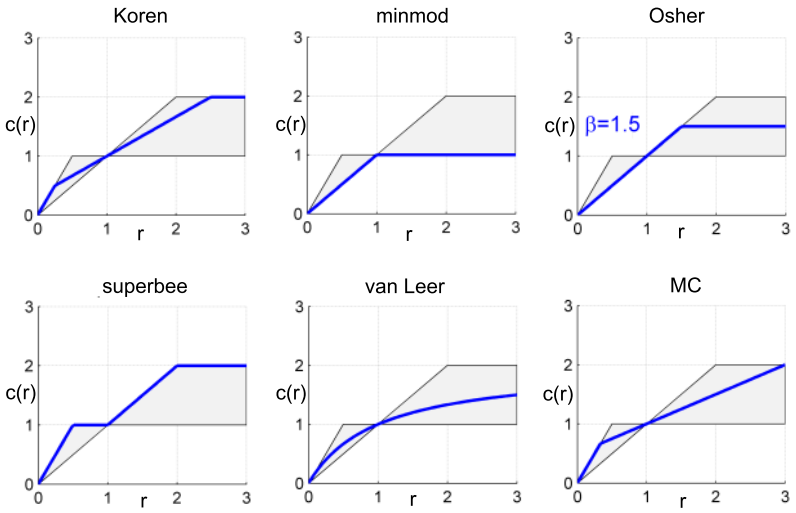

Abbildung 4.1.: Flux-Limiter Funktionen $\psi(r)$ (blau) und *2nd-order TVD* Bereiche (grau)

4.5. Local Double Logarithmic Reconstruction

Die Local-Double-Logarithmic-Reconstruction (LDLR) von R. Artebrant und H.J. Schroll ist eine Rekonstruktion dritter Ordnung, die ohne Limiter auskommt und nur einen kleinen Stencil, bestehend aus den direkten Nachbarn der betrachteten Zelle, benötigt. Trotzdem werden lokale Extrema erhalten, während die lokale Variation (s.u.) beschränkt bleibt.

Betrachtet wird ein Gitter mit $x_i = ih$, $i = 1, ..., n$, Gitterweite $h > 0$ und den Gitterzellen $C_i = [x_{i-1/2}, x_{i+1/2}]$, sowie dem Zellmittelpunkt $x_0 = \frac{x_{i+1/2}+x_{i-1/2}}{2}$ vgl. Abbildung (5.1). Die re-

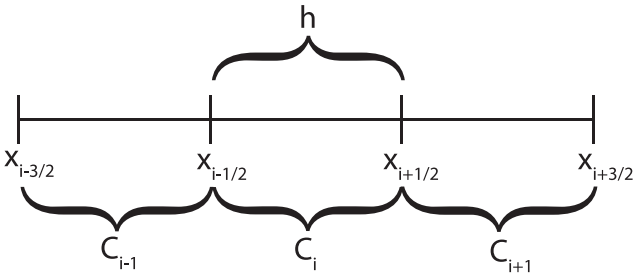

Abbildung 5.1.: Gitter

konstruierende Funktion, die für jede einzelne Zelle berechnet wird, habe die allgemeine Form:

$$r_0 \sim A + B \cdot log(x + C) + D \cdot log(x + E) \tag{4.5.1}$$

Betrachte die aktuelle Gitterzelle C_0. Für $x \in C_0$ wird nun folgende Ansatzfunktion definiert:

$$\varphi_0(x) = -\frac{ch}{a} log[x - x_0 - \frac{h}{2}(\frac{2}{a} - 1)] - \frac{dh}{b} log[x - x_0 - \frac{h}{2}(\frac{2}{b} - 1)], \tag{4.5.2}$$

mit Parametern a, b, c, d. Man könnte sagen, dass je ein logarithmischer Ausdruck die Funktion auf je einer Hälfte der Gitterzelle rekonstruiert. Durch die Subtraktion des Zellmittelpunktes x_0 wird die Zelle quasi auf eine Einheitszelle transformiert. Die Rekonstruktion hängt nun lediglich noch von den drei Zellmittelwerten ab.
Die rekonstruierende Funktion lautet dann:

$$r_0(x) = v_0 + \phi_0 - \frac{1}{h} \int_{C_0} \phi_0(\xi) d(\xi) \tag{4.5.3}$$

wobei v_0 der bekannte Mittelwert der Gitterzelle ist. Damit hat r_0 innerhalb der Gitterzelle C_0 den Mittelwert v_0.
Folgende Bedingungen sollen nun von r_0 erfüllt werden:

$$\frac{1}{h} \int_{C_0} r_0(x) dx = v_0, \tag{4.5.4}$$

$$r_0'(x_0 - \frac{h}{2}) = d_1, \tag{4.5.5}$$

$$r_0'(x_0 + \frac{h}{2}) = d_2. \tag{4.5.6}$$

Diese Bedingungen besagen nicht anderes, als dass zum einen die rekonstruierende Funktion im Mittel den gegebenen Zellmittelwert annimmt. Zum anderen sollen die Ableitungen auf den Rändern der Zelle mit den berechneten Finiten Differenzen d_1 und d_2 übereinstimmen. Letztere können aus den drei gegebenen Mittelwerten berechnet werden.
Aus Symmetriegründen wird außerdem gefordert, dass:

$$0 = r_0'(x_0)|_{d_1 = -d_2} \tag{4.5.7}$$

gilt. Diese Bedingung stellt sicher, dass im Falle symmetrischer Ableitungen, also falls gilt $d_1 = -d_2$ das Maximum bzw. Minimum genau in der Mitte der Zell angenommen wird, die Funktion also symmetrisch zum Mittlepunkt der Zelle verläuft. Aus den vier gegebenen Bedingungen können nun die Parameter b, c und d berechnet werden. Unter anderem ergibt sich dabei für den Parameter b:

$$b = \frac{a}{a - 1} \tag{4.5.8}$$

In der obigen Grafik sind blau der Funktionsverlauf der Rekonstruierenden, rot die Zellmit-

telwerte und grün die Ableitungen zu sehen. Die Rekonstruktion wurde über drei Gitterzellen, jeweils mit der Gitterweite $h = 1$ geplottet.

Schließlich muss noch der Parameter a bestimmt werden. Dazu wird noch einmal die Ansatzfunktion betrachtet und angenommen, der linke Term sei verantwortlich für die Rekonstruktion auf der linken Seite der Gitterzelle.
Sei nun angenommen die linke Ableitung sei in $O(\frac{1}{h})$, also sehr groß (Sprungstelle). In diesem Fall würde man erwarten, dass die rekonstruierende Funktion in der Umgebung des Zellrandes einen "scharfen Knick" macht. Die Singularität des Logarithmus müsste also ein kleines Stück links von $x_{i-1/2}$ liegen. Setzt man nun $a = 1$ so erhält man für das Argument des Logarithmus $(x - x_0 - \frac{h}{2})$, also eine Singularität genau auf dem linken Rand $(x = x_0 - \frac{h}{2})$ der Zelle. Wählt man $(a = 1 - \epsilon)$ so liegt die Singularität knapp außerhalb der Gitterzelle, womit diese Bedingung erfüllt ist.
Außerdem ist wünschenswert, dass die rekonstruierende Funktion in der Lage ist einen linearen Werteverlauf zu erhalten, dass sich also für den Fall $d_1 \approx d_2$ ein linearer Funktionsverlauf ergibt. Dazu müssen beide Singularitäten möglichst weit "nach Außen geschoben werden". Dies geschieht offensichtlich, wenn $a \approx 0$ gewählt wird. Mit 4.5.8 gilt dann auch $b \approx 0$, was schließlich zur gewünschten Eigenschaft führt.

Um eine reellwertige, glatte Funktion zu erhalten, muss also offensichtlich gelten:

$$a \in (0, 1).$$

Zusätzlich soll für $d_1 \approx d_2$ gelten:

$$a(d_1, d_2) \approx 0. \tag{4.5.9}$$

In den folgenden Abbildung wurde die Rekonstruierende für den Fall $d_1 \approx d_2$ geplottet. Abbildung 5.3 zeigt die Funktion über der aktuellen und den beiden umgebenden Zellen. Abbildung 5.4 wurde über einen größeren Definitionsbereich von etwa 20 Gitterzellen geplottet. Man kann gut erkennen, dass die Rekonstruktion in der direkten Umgebung der Gitterzelle tatsächlich linear verläuft. Dies wird wie oben beschrieben erreicht, indem die Singularität genügend weit "hinausgeschoben"wird.

Zusätzlich zu den bereits genannten Bedingungen soll die LVB-Bedingung von Marquina erfüllt werden. Diese garantiert, dass bei der Rekonstruktion keine Oszillationen, also neue Extrema entstehen. Insbesondere sollen auch Over- und Undershoots möglichst gering gehalten werden.

Definition: *Lokale Variation (Marquina)*

Für ein Rechengebiet Ω mit einer Zerlegung $\Omega = \{\Omega_1, \Omega_2, ..., \Omega_n\}$ und eine Funktion $f_i : \Omega_i \to R$ mit $\Omega_i \in R$ wird die *lokale Variation* definiert durch:

$$LV(f_i) = TV(f)|_{\Omega_i}. \tag{4.5.10}$$

Eine Funktion heißt *local variation bounded* (LVB) in Ω_i falls gilt

$$LV(f_i) = O(h_i),$$

wobei h die Gitterweite der Zelle Ω_i ist.

Dabei ist die Totale Variation einer Funkiton wie folgt definiert:
Definition: *Totale Variation*

Für eine Funktion $f : \Omega = [a, b] \to R$ mit $\Omega \in R$ und eine beliebig feine Zerlegung $a \leq t_1^{(n)} < t_2^{(n)} < ... < t_n^{(n)} \leq b$ ist die *totale Variation* (TV) definiert als:

$$TV|_\Omega := sup\{\sum_{k=1}^{n-1} |f(t_{k+1}^{(n)}) - f(t_k^{(n)})| | n \in N, a \leq t_1^{(n)} < t_2^{(n)} < ... < t_n^{(n)} \leq b\} \tag{4.5.11}$$

Ist die Funktion auf dem Intervall $[a, b]$ zusätzlich differenzierbar, so wird definiert:

$$TV|_\Omega := \int_\Omega |f'(x)|dx. \tag{4.5.12}$$

Die totale Variation beschreibt also in gewissem Sinne das Oszillationsverhalten einer Funktion. Sie ist beschränkt, wenn die Funktion nicht beliebig stark oszilliert. Ist nun die LVB-Bedingung für alle $\Omega_i \in \Omega$ erfüllt, so ist sichergestellt, dass die rekonstruierende Funktion kein unerwünschtes Oszillationsverhalten aufweist.
Diese Bedingungen werden für die folgende Wahl von a erfüllt:

$$a = (1 - TOL)(1 + TOL - \frac{2|d_1|^q|d_2|^q + TOL}{|d_1|^2 q + |d_2|^2 q + TOL}). \tag{4.5.13}$$

Für eine exakte Herleitung dieser Formel und der Wahl der Toleranz "TOL" sei auf [**?**] verwiesen.
Die Toleranz wird demnach in Abhängigkeit der Gitterweite folgendermaßen bestimmt:

$$TOL = 0.1h^q \quad \text{mit} \quad q = 1.4. \tag{4.5.14}$$

Je nach Anwendung kann eine abweichende Wahl von q sinnvoll sein. Wählt man den Faktor q größer wird die LV kleiner. In Abbildung (5.5) ist die stückweise logarithmische Rekonstruktion mittels LDLR geplottet. Die zugrunde liegende Funktion (schwarz) ist identisch mit der in Abb. (3.1). Es wurden zusätzlich die Mittelwerte innerhalb der einzelnen Gitterzellen (blau) und die Rekonstruktionen (rot) geplottet. Die Flüsse $\hat{f}_{i+1/2}$ und $\hat{f}_{i-1/2}$ erhält man nun durch Auswerten der Funktion r_0 an den Stellen $(x_0 \pm h/2)$. [**?**]

Tabelle 5.1.: Algorithmus zur Berechnung einer Rekonstruktion mittels LDLR

- Bestimmung der Ableitungen d_1 und d_2
- Berechnen der Parameter a und b

$$a = (1 - TOL)(1 + TOL - \frac{2|d_1|^q|d_2|^q + TOL}{|d_1|^2q + |d_2|^2q + TOL})$$

$$b = \frac{a}{a-1}$$

- Berechnen der Parameter c und d

$$c = \frac{(a-1)(d_2(1-b) - d_1)}{b-a}$$

$$d = d_1 - c$$

- Berechnung des Mittelwerts der Funktion φ_0 über der Gitterzelle C_0. In der Implementierung lässt sich diese Integration vermeiden, indem man die entsprechenden Werte von r_0 an den Stellen $(x_0 \pm \frac{h}{2})$ folgendermaßen bestimmt:

$$r_0\left(x_0 \pm \frac{h}{2}\right) = v_0 + ch\eta^{\pm}(a) + dh\eta^{\pm}(b)$$

mit

$$\eta^{+}(t) = -\frac{log(1-t) + t}{t^2}$$

und

$$\eta^{-}(t) = \frac{(t-1)log(1-t) - t}{t^2}.$$

- Berechnen der Flüsse $\hat{f}_{i+1/2}$ und $\hat{f}_{i-1/2}$.

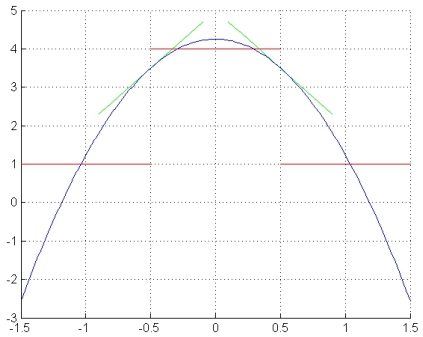

Abbildung 5.2.: Rekonstruktion bei symmetrischen Ableitungen

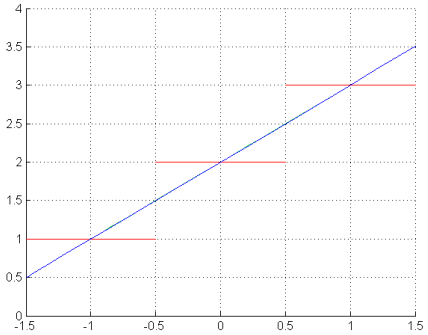

Abbildung 5.3.: lokal lineare Rekonstruktion

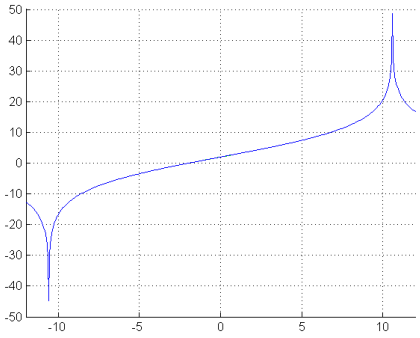

Abbildung 5.4.: Singularitäten bei lokal lineare Rekonstruktion

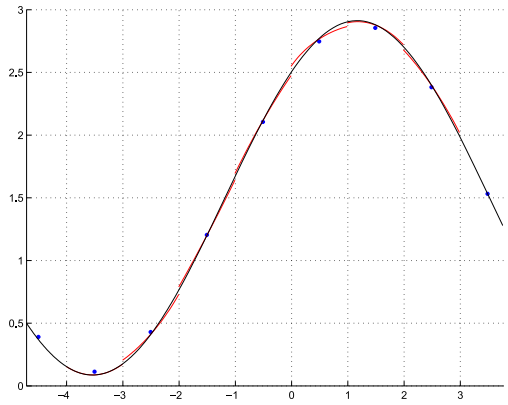

Abbildung 5.5.: stückweise logarithmische Rekonstruktion

A. Anhang

Tabelle 0.1.: Gewichte ω_i für die Finiten Differenzen in (5.2); $\omega_1 = -\omega 2,\ \omega_4 = -\omega_3$

	u_x^{+p}	u_x^{+m}	u_x^{-p}	u_x^{-m}
FD1				
ω_2	$\alpha = \frac{\sqrt{3}}{6h}$	β	α	β
ω_3	$\beta = \frac{6-\sqrt{3}}{6h}$	α	β	α
stencil	N,NE,E,O	O,E,SE,S	NW,N,O,W	W,O,S,SW
FD2				
ω_2	$\gamma = \frac{6+\sqrt{3}}{6h}$	δ	γ	δ
ω_3	$\delta = -\frac{\sqrt{3}}{6h}$	γ	δ	γ
stencil	O,E,SE,S	N,NE,E,O	W,O,S,SW	NW,N,O,W

Tabelle 0.2.: Gewichte ω_i für die Finiten Differenzen in (5.2); $\omega_4 = -\omega 1,\ \omega_3 = -\omega_2$

	u_y^{+p}	u_y^{+m}	u_y^{-p}	u_y^{-m}
FD1				
ω_1	$\beta = \frac{6-\sqrt{3}}{6h}$	β	α	α
ω_2	$\alpha = \frac{\sqrt{3}}{6h}$	α	β	β
stencil	N,NE,E,O	O,E,SE,S	NW,N,O,W	W,O,S,SW
FD2				
ω_1	$\delta = -\frac{\sqrt{3}}{6h}$	δ	γ	γ
ω_2	$\gamma = \frac{6+\sqrt{3}}{6h}$	γ	δ	δ
stencil	NW,N,O,W	W,O,S,SW	N,NE,E,O	O,E,SE,S

Abbildungsverzeichnis

Tabellenverzeichnis

Literaturverzeichnis

[1] H.J. Artebrant, R.; Schroll. Limiter-free third order logarithmic reconstruction. *SIAM J.SCI.COMPUT.*, 28(1):359–381, 2006.

[2] R. Artebrant. Third order accurate non-polynomial reconstruction on rectangular and triangular meshes. *SIAM Journal of Scientific Computing*, 30(2):193–221, 2007.

[3] D.R. Durran. *Numerical Methods for Fluid Dynamics*. Springer New York, 2 edition, 2010.

[4] Verwer J.G. Hundsdorfer, W.H. *Numerical Solution of time-dependent advection-diffusion-reaction equations*. Springer, 1 edition, 2003.

[5] J. R. Rice. *The Approximation of Functions*. Addison-Wesley, 1964.

[6] A.; Bernsdorf J.; et al. Roller, S.; Beck. Introduction to computational fluid dynamics, 9 2011.

[7] M. Schaefer. *Numerik im Maschinenbau*. Springer, 1 edition, 1999.

B. Erklärung

Hiermit versichere ich, daß ich die vorliegende Diplomarbeit selbstständig verfasst und keine anderen Hilfsmittel als die angegebenen verwendet habe. Die Stellen, die anderen Werken dem Wortlaut oder dem Sinne nach entnommen sind, habe ich in jedem Falle durch Angaben der Quelle, auch der Sekundärliteratur, als Entlehnung kenntlich gemacht.

Greifswald, den 12. April 2012