Institut für Parallele und Verteilte Systeme

Abteilung Simulation großer Systeme

Universität Stuttgart
Universitätsstraße 38
D - 70569 Stuttgart

Diplomarbeit Nr. 2239

Effiziente Randbehandlung unregelmäßiger Geometrien bei Simulation von Strömungen mit symmetrieerhaltender Diskretisierung

Alexander Buck

Studiengang:	Informatik
Prüfer:	Prof. Dr. Hans-Joachim Bungartz
Betreuer:	Markus Brenk
begonnen am:	12.7.2004
beendet am:	11.1.2005
CR-Klassifikation:	F.2.1, G.1.4, G.1.8, G.1.10, I.6.3, I.6.8

Inhaltsverzeichnis

Kapitel 1

Einleitung

1.1 Motivation

Die numerische Simulation von Strömungen hat in den letzten Jahren in immer mehr Bereichen Einzug gehalten. So wurde zum Beispiel für die Entwicklung einer Autokarosse mit möglichst niedrigem c_W-Wert vor fünfzehn Jahren noch fast ausschließlich auf die Erfahrung der Ingenieure und Versuche im Windkanal zurückgegriffen. Heute dagegen ist das Erstellen von Modellen und deren Simulation im Rechner nicht mehr aus diesem Bereich wegzudenken. Das liegt vor Allem daran, dass unerwünschte Verwirbelungen bereits bei der Simulation im Rechner sichtbar werden und die Korrekturen dann nur am virtuellen Modell vorgenommen werden müssen. Dadurch wird erreicht, dass auch nur die erfolgversprechenden Modelle tatsächlich das Prototypenstadium erreichen und so die Anzahl der teuren Untersuchungen im Windkanal reduziert werden kann. Da die Produktion von Prototypen ein zeitaufwändiger Vorgang ist, geht damit als angenehmer Nebeneffekt eine Verkürzung der Entwicklungszeit einher.

An diesem Beispiel werden aber auch die Grenzen der numerischen Strömungssimulation sichtbar. Um genaue Ergebnisse zu erhalten, müssen immer noch Untersuchungen im Windkanal durchgeführt werden. Zwar geht mit dem rasanten Anstieg der Rechenleistung heutiger Computer auch eine entsprechende Verbesserung der Simulationsergebnisse einher, da die Anforderugen der numerischen Strömungssimulation an die Rechenleistung aber extrem hoch sind, kann bis heute nicht auf reale Messungen verzichtet werden.

Die Einsatzgebiete der numerischen Strömungssimulation sind vielfältig. Anwendungen ergeben sich dabei vorwiegend im technischen Bereich, wie zum Beispiel bei der Entwicklung von möglichst günstigen Schaufelprofilen für Turbinen, oder der Optimierung von Verbrennungsvorgänge in Motoren. Es sind aber auch Aufgabenstellungen in anderen Gebieten, wie zum Beispiel in der Medizin vorhanden, wo für ein besseres Verständnis der inneren Körperabläufe der Blutkreislauf simuliert werden kann.

Da bei der Betrachtung von Strömungen üblicherweise nicht die Strömung an sich von Interesse ist, sonderen die Wechselwirkung mit Objekten, hat die Behandlung dieser Grenzflächen im Simulationsalgorithmus eine besondere Bedeutung. Das Ziel bei der Entwicklung von Algorithmen zur Strömungssimulation muss also unter Anderem darin bestehen, diese

Grenzflächen möglichst gut und natürlich auch möglichst effizient zu approximieren.

1.2 Ziel der Arbeit

Wie im oberen Abschnitt schon angeklungen ist, kommt der Repräsentation von Oberflächen umströmter Objekte bei der Simulation von Strömungen eine große Bedeutung zu. Insbesondere die Bestimmung integraler Größen, wie zum Beispiel der Strömungswiderstand eines Objektes, hängt stark von den Randbedingungen ab, da diese den Einfluss des Objektes auf die Strömung beschreiben.

Das Ziel dieser Arbeit ist es nun, verschiedene Techniken zu entwickeln, die in der Lage sind, den Rand eines beliebigen Hindernisses zu approximieren. Diese Techniken werden dann in ein bereits vorhandenes Programm zur Simulation 2-dimensionaler Strömungen integriert und ihre Qualität dann anhand der Simulation verschiedener durchströmter Gebiete bestimmt. Dabei kommt zum Einen ein Kanal mit Verjüngung zum Einsatz, zum Anderen wird der sogenannte DFG-Benchmark betrachtet. Ursprünglich war auch noch eine Erweiterung des Verfahrens auf drei Raumdimensionen angedacht, die aber leider nicht durchgeführt werden konnte, da sich die Integration der Randbehandlung deutlich komplizierter gestaltete als anfangs angenommen.

1.3 Gliederung

Der Aufbau dieser Arbeit gestaltet sich folgendermaßen. Zunächst wird kurz auf die mathematischen Grundlagen der numerischen Strömungssimulation eingegangen. Anhand dieser Grundlagen, wird dann der Simulationsalgorithmus präsentiert und auf die besonderen Eigenschaften der verwendeten Diskretisierung eingegangen. Daran schließt sich eine Beschreibung der Methoden zur Randbehandlung an, gefolgt von einigen Details zur Implementierung. Danach erfolgt ein kurzer Überblick über die Gebiete, die zur Validierung der Randbehandlung durchströmt wurden, und es werden die Ergebnisse, die sich dabei ergeben haben präsentiert. Den Abschluss bildet dann eine kurze Zusammenfassung mit einem kleinen Ausblick.

Kapitel 2

Grundlagen der Strömungssimulation

2.1 Chrakterisierung von Strömungen

Die Strömungen, die sich in einem durchflossenen Gebiet ausbilden, sind sehr stark von den Eigenschaften des verwendeten Fluids abhängig. Bei einem Fluid handelt es sich dabei nach [10] um einen Stoff, der im Gegensatz zu Festkörpern Scherkräften in der Ruhelage nicht widerstehen kann. Anschaulich gesprochen ist ein Fluid also entweder eine Flüssigkeit oder ein Gas.

Wichtige charakteristische Eigenschaften eines Fluids sind Viskosität und Kompressibilität. Kompressible Fluide können zusammengedrückt werden, da ihre Dichte nicht konstant ist, sondern vom Druck abhängt. Flüssigkeiten sind üblicherweise inkompressibel, wohingegen das bei Gasen nur für niedrige Strömungsgeschwindigkeiten gilt. Bei hohen Geschwindigkeiten sind sie kompressibel. Unter der Viskosität eines Fluids kann seine Zähigkeit verstanden werden. Sie ist bestimmt durch die Reibungskräfte, die bei einer Bewegung des Fluids zwischen den einzelnen Molekülen auftreten. Beispielsweise ist die Viskosität von Honig deutlich höher als die von Wasser. Wird ein Fluid als nicht-viskos bezeichnet, dann ist damit gemeint, dass die Reibungskräfte zwischen Teilchen so gering sind, dass sie vernachlässigt werden können.

Die Eigenschaften des Fluids, sowie Fließgeschwindigkeit und Topologie des Gebietes bestimmen dann die Art der Strömung. Dabei wird zwischen laminaren und turbulenten Strömungen unterschieden. Laminare Strömungen sind dadurch charakterisiert, dass parallel zur Strömungsrichtung so gut wie keine Durchmischung auftritt. Zur Veranschaulichung kann man sich in diesem Fall das Fluid als einen Plattenstapel vorstellen, in dem sich die Platten ausschließlich längs zueinander bewegen lassen. Im Gegensatz dazu bilden sich bei einer turbulenten Strömung keine Schichten aus und es findet eine Durchmischung des Fluids in alle Richtungen statt. In der Natur findet der geordnete Fall der laminaren Strömung eher in sehr langsam fließenden Gewässern statt, wogegen kleine Gebirgsbäche vorwiegend turbulent strömen.

2.2 Die Navier-Stokes-Gleichungen

Mathematische Grundlage zur Beschreibung von Strömungen sind die nach ihren Entdeckern benannten Navier-Stokes-Gleichungen. Es handelt sich dabei um ein System partieller Differentialgleichungen. Für den hier betrachteten Fall einer 2-dimensionalen, instationären, laminaren Strömung eines viskosen, unkompressiblen Fluids, sehen diese Gleichungen folgendermaßen aus:

$$\frac{\partial}{\partial t^*}\vec{u}^* = -(\vec{u}^* \cdot \nabla^*)\vec{u}^* + \frac{\mu}{\rho_\infty}\Delta^*\vec{u}^* - \frac{1}{\rho_\infty}\nabla^* p^* + \vec{g}^* \tag{2.1}$$

$$\mathrm{div}(\vec{u}^*) = 0 \tag{2.2}$$

Diese Gleichungen beschreiben die Strömung eines Fluids mit konstanter Dichte ρ_∞ zum Zeitpunkt $t^* \in [0, t_{end}]$ in einem Gebiet $\Omega^* \subset \mathbf{R}^2$. Dabei repräsentiert $\vec{g}^* \in \mathbf{R}^2$ eine äußere Kraft, die auf das System einwirkt,

$$\vec{u}^* : \Omega^* \times [0, t_{end}] \to \mathbf{R}^2$$

das Geschwindigkeitsfeld der Strömung und

$$p^* : \Omega^* \times [0, t_{end}] \to \mathbf{R}$$

deren Druckfeld. Die Variablen sind hier alle noch dimensionsbehaftet und daher mit einem Stern gekennzeichnet. Selbiges gilt für den Nabla- und den Laplace-Operator, die sich beide auf die Variable $\vec{x}^*$ beziehen. Gleichung 2.2 aus dem obigen System wird auch als Kontinuitätsgleichung und Gleichung 2.1 als Impulsgleichung bezeichnet.

Normalerweise werden Simulationen aber nicht mit absoluten, sondern mit dimensionslosen Werten durchgeführt und das Ergebnis dann entsprechend skaliert. Dadurch wird erreicht, dass sich die Werte, mit denen gerechnet wird nicht in extremen Dimensionen bewegen. Um die dimensionsbehafteten in dimensionslose Größen umzuwandeln, können die folgenden Zusammenhänge verwendet werden:

$$\vec{x} := \frac{\vec{x}^*}{L}, \qquad t := \frac{u_\infty t^*}{L}, \qquad \vec{u} := \frac{\vec{u}^*}{u_\infty}, \qquad p := \frac{p^* - p_\infty}{\rho_\infty u_\infty^2}, \qquad \vec{g} := \frac{L}{u_\infty^2}\vec{g}^*.$$

Dabei handelt es sich bei L um den Skalierungsfaktor und bei u_∞ und p_∞ um die Bezugsgrößen für Geschwindigkeit und Druck. ρ_∞ gibt, wie schon oben erwähnt, die Dichte des Fluids an. Mit der Größe μ für die dynamische Viskosität wird dann die sogenannte Reynoldszahl Re definiert durch

$$Re := \frac{\rho_\infty u_\infty L}{\mu}$$

und die Navier-Stokes-Gleichungen können dann in ihrer dimensionslosen Form angegeben werden als

$$\frac{\partial}{\partial t}\vec{u} = -(\vec{u} \cdot \nabla)\vec{u} + \frac{1}{Re}\Delta\vec{u} - \nabla p + \vec{g}$$

$$\mathrm{div}(\vec{u}) = 0.$$

Eine Herleitung der Navier-Stokes-Gleichungen, sowie ausführlichere Informationen zu der Entdimenionalisierung finden sich unter anderem in [10].

Kapitel 3

Der Ausgangsalgorithmus

Den Ausgangspunkt für die Integration der Randbehandlung bildet der Algorithmus von Felix Müller, den er im Rahmen einer Studienarbeit [1] erstellt hat. Dieser Algorithmus löst die Navier-Stokes-Gleichungen aus dem letzten Kapitel unter Verwendung einer Finiten-Volumen-Methode und geht auf eine Veröffentlichung von R.W.C.P Verstappen und A.E.P Veldman [2] sowie die Doktorarbeit von Maximilian Emans [3] zurück. Er basiert auf einer Finiten-Volumen-Methode in Kombination mit einem teilweise versetzten Gitter. Im folgenden Kapitel wird seine Funktionsweise näher erläutert.

3.1 Diskretisierung

Zur Diskretisierung wird das 2-dimensionale Simulationsgebiet mit einem Gitter aus gleichgroßen quadratischen Zellen überzogen. Wie in Abbildung 3.1 dargestellt, wird dabei der Druck in der Zellmitte und die Geschwindgkeit in den Zellecken platziert. Das entspricht der Anordnung in einem teilweise versetzten Gitter. Die Geschwindigkeit besteht aus einer horizontalen und einer vertikalen Komponente, die jeweils entlang der Zellkanten verlaufen. Da es sich um ein quadratisches Gitter handelt, besitzen alle Zellkanten die gleiche Länge, die im Folgenden als Kantelänge h bezeichnet wird. Warum dieser Anordnung gegenüber dem sonst eher gebräuchlichen vollständig versetzten Gitter, bei dem die Geschwindigkeit auf den Kantenmitten der Zelle plaziert ist, der Vorzug gegeben wurde, ist in der Studienarbeit [1] beschrieben. Dort erfolgt auch eine kurze Vorstellung verschiedener Gittertypen.

3.2 Die Finite-Volumen-Methode

Bei der Finiten-Volumen-Methode wird versucht, die integrale Form der Navier-Stokes-Gleichungen zu erfüllen. Dafür wird zunächst die integrale Form aufgestellt, indem man die Gleichungen über das Simulationsgebiet integriert. Wenn das Simulationsgebiet in mehrere Kontrollvolumina unterteilt wird und die integrale Form der Navier-Stokes-Gleichungen für jedes Kontrollvolumen erfüllt ist, dann kann aufgrund der Integraladditivität daraus geschlossen werden, dass das auch auf dem gesamten Gebiet der Fall ist.

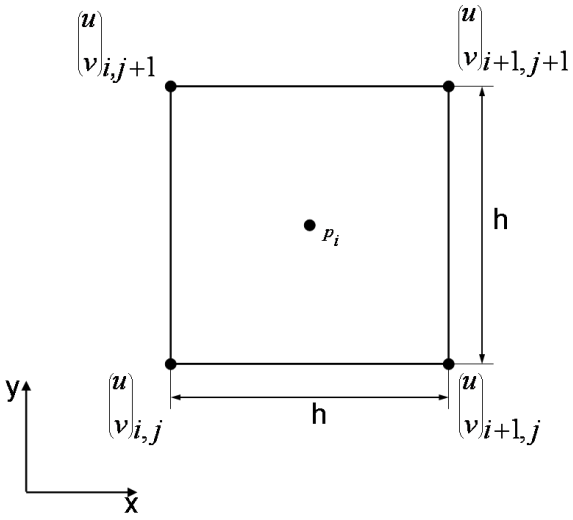

Abbildung 3.1: Anordnung von Geschwindigkeit und Druck im teilweise versetzten Gitter.

Zunächst muss also die integrale Form der Navier-Stokes-Gleichungen bestimmt werden. Im vorliegenden 2-dimensionalen Fall wird dazu nicht über ein Volumen, sondern die Gebietsfläche Ω integriert. Es ergibt sich dann für die dimensionslosen Navier-Stokes-Gleichungen aus Kapitel 2.2:

$$\begin{aligned}
\int_\Omega \frac{\partial u}{\partial t} dA &= -\int_\Omega (\frac{\partial u^2}{\partial x} + \frac{\partial (uv)}{\partial y}) dA + \frac{1}{Re} \int_\Omega (\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}) dA - \int_\Omega \frac{\partial p}{\partial x} dA + \int_\Omega g_x dA \\
\int_\Omega \frac{\partial v}{\partial t} dA &= -\int_\Omega (\frac{\partial (uv)}{\partial x} + \frac{\partial v^2}{\partial y}) dA + \frac{1}{Re} \int_\Omega (\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}) dA - \int_\Omega \frac{\partial p}{\partial y} dA + \int_\Omega g_y dA \\
\int_\Omega \text{div} \vec{u} \; dA &= 0.
\end{aligned}$$

Dabei wurde für diese Darstellung die Impulsgleichung in ihre Komponenten zerlegt. Die endgültige Form der integralen Navier-Stokes-Gleichungen erhält man, indem mehrere Umformungen wie in der Studienarbeit [1] beschrieben, durchgeführt werden. Die endgültige Formulierung lautet dann:

$$\begin{aligned}
\int_\Omega \frac{\partial u}{\partial t} dA &= -\int_{\partial\Omega} \begin{pmatrix} u^2 \\ uv \end{pmatrix} \cdot \vec{n} \; dS + \frac{1}{Re} \int_{\partial\Omega} \nabla u \cdot \vec{n} \; dS - \int_\Omega \frac{\partial p}{\partial x} dA + \int_\Omega g_x dA \\
\int_\Omega \frac{\partial v}{\partial t} dA &= -\int_{\partial\Omega} \begin{pmatrix} uv \\ v^2 \end{pmatrix} \cdot \vec{n} \; dS + \frac{1}{Re} \int_{\partial\Omega} \nabla v \cdot \vec{n} \; dS - \int_\Omega \frac{\partial p}{\partial y} dA + \int_\Omega g_y dA \\
\int_{\partial\Omega} \vec{u} \cdot \vec{n} \; dS &= 0.
\end{aligned}$$

Als nächstes müssen nun Kontrollvolumen festgelegt und die Integralgleichungen diskretisiert werden. Dabei erfolgt zunächst die räumliche Diskretisierung, beschrieben im nächsten Abschnitt und dann erst die zeitliche Diskretisierung, beschrieben im übernächsten Abschnitt.

3.3 Räumliche Diskretisiertung

Bei der räumlichen Diskretisierung bestehen große Unterschiede zwischen der Behandlung der Impulsgleichungen und der Behandlung der Kontinuitätsgleichung, da für die Diskretisierung der Kontinuitätsgleichung ein anderes Kontrollvolumen zum Einsatz kommt. Die beiden Impulsgleichungen dagegen weisen so viele Gemeinsamkeiten auf, dass deren Diskretisierung im folgenden Abschnitt nur anhand der ersten Gleichung beschrieben wird. Das Verfahren zur Diskretisierung der zweiten Gleichung kann dann problemlos daraus abgeleitet werden.

3.3.1 Diskretisierung der Impulsgleichungen

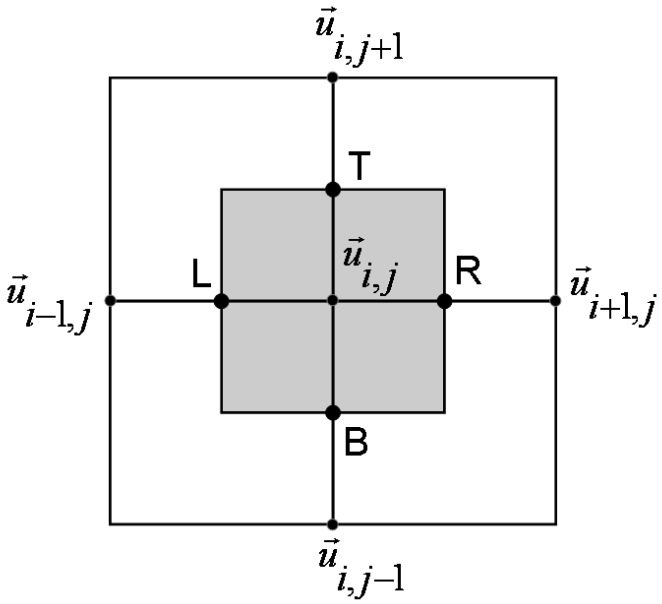

Abbildung 3.2: Lage der Kontrollflächen.

Zur Diskretisierung der Impulsgleichungen wird das Simulationsgebiet mit quadratischen Kontrollvolumen oder besser gesagt Kontrollflächen überzogen. Dabei liegen wie in Abbildung 3.2 zu sehen ist, die Knoten im Mittelpunkt der Kontrollflächen. Entsprechend ist dann die Kantenlänge der Kontrollflächen mit der der Zellen identisch. Schließlich bleibt noch zu erwähnen, dass die Kantenmittelpunkte für die folgenden Betrachtungen wie in der Abbildung mit B, R, T und L bezeichnet werden. Im Folgenden sollen nun die Impulsgleichungen schrittweise diskretisiert werden.

Der konvektive Term

Beim konvektiven Term handelt es sich um das Integral

$$-\int_{\partial\Omega} \begin{pmatrix} u^2 \\ uv \end{pmatrix} \cdot \vec{n}\, dS.$$

Geht man davon aus, dass die Werte für u^2 und uv in Kantenmittelpunkten bekannt, sowie auf dem Rand des Kontrollvolumens konstant sind, dann gilt:

$$\int_{\partial\Omega} \begin{pmatrix} u^2 \\ uv \end{pmatrix} \cdot \vec{n}\, dS = h(U_R u_R - U_L u_L + U_T v_T - U_B v_B). \tag{3.1}$$

Es müssen also die Werte für die sogenannten konvektiven Flüsse $U_R u_R$, $U_L u_L$, $U_T v_T$ und $U_B v_B$ an den entsprechenden Kantelmittelpunkten bestimmt werden. Dazu wird bei diesen Produkten der erste Faktor als transportierende Größe und der zweite Faktor als transportierte Größe aufgefasst, wobei die transportierenden Größen hier durch Großbuchstaben gekennzeichnet sind.

Zur Berechnung der transportierenden Größe in einem Kantemittelpunkt des Kontrollvolumens, wird zunächst die Geschwindigkeit in den Mittelpunkten der beiden angrenzenden Zellen benötigt. Die Geschwindigkeit im Mittelpunkt einer beliebigen Zelle i, j wird dabei getrennt für horizontale Komponente $um_{i,j}$ und vertikale Komponente $vm_{i,j}$ aus den angrenzenden Geschwindigkeitsknoten entsprechend Abbildung 3.3 bestimmt:

$$um_{i,j} = \frac{1}{4}(u_{i,j} + u_{i+1,j} + u_{i,j+1} + u_{i+1,j+1} + v_{i,j} + v_{i+1,j+1} - v_{i+1,j} - v_{i,j+1}) \tag{3.2}$$

$$vm_{i,j} = \frac{1}{4}(v_{i,j} + v_{i+1,j} + v_{i,j+1} + v_{i+1,j+1} + u_{i,j} + u_{i+1,j+1} - u_{i+1,j} - u_{i,j+1}). \tag{3.3}$$

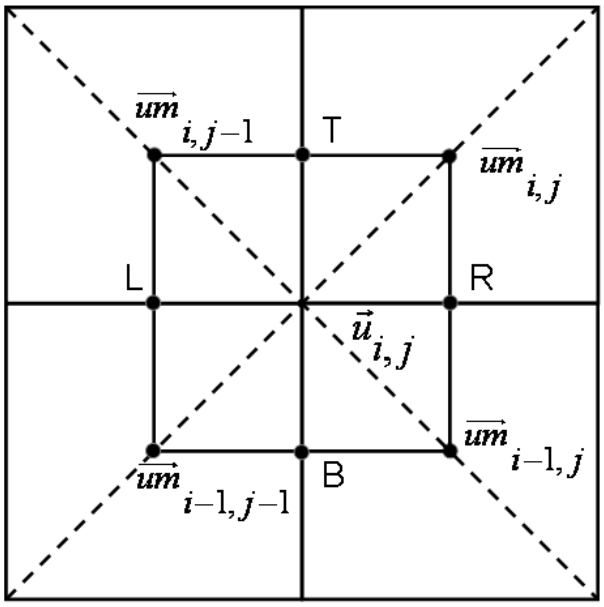

Abbildung 3.3: Knoten mit umliegenden Geschwindigkeiten

Die transportierende Größe ergibt sich dann als arithmetisches Mittel aus den Geschwindigkeiten in den Mittelpunkten der angrenzenden Zelle, sowie den Geschwindigkeiten der

beiden benachbarten Knoten. Für horizontale Größen werden dazu nur die u-Komponenten und für vertikale Größen entsprechend nur die v-Komponenten der Geschwindigkeit an den entsprechenden Punkten verwendet. Für U_R und V_T ergibt sich dann beispielsweise:

$$\begin{aligned} U_R &= \frac{1}{4}(um_{i,j} + um_{i,j-1} + u_{i,j} + u_{i+1,j}) \\ V_T &= \frac{1}{4}(vm_{i,j} + vm_{i-1,j} + v_{i,j} + v_{i,j+1}). \end{aligned}$$

Die Berechnung der transportierten Größe verläuft sehr ähnlich Der Unterschied liegt in der Bestimmung der Geschwindigkeiten in den Zellmittelpunkten. Hier wird der Wert im Mittelpunkt durch lineare Interpolation entlang der gestrichelten Linien in Abbildung 3.3 berechnet. In den Punkten B und L ergeben sich dann beispielsweise die Größen v_B und u_L zu

$$\begin{aligned} v_B &= \frac{1}{4}(\frac{1}{2}(v_{i,j} + v_{i+1,j-1}) + \frac{1}{2}(v_{i,j} + v_{i-1,j-1}) + v_{i,j} + v_{i,j-1}) \\ u_L &= \frac{1}{4}(\frac{1}{2}(u_{i,j} + u_{i-1,j+1}) + \frac{1}{2}(u_{i,j} + u_{i-1,j-1}) + u_{i,j} + v_{i-1,j}). \end{aligned}$$

Diffusiver Term

Als diffusiver Term wird das Integral

$$\int_{\partial\Omega} \nabla u \cdot \vec{n}\ dS = \int_{\partial\Omega} \begin{pmatrix} \frac{\partial u}{\partial x} \\ \frac{\partial u}{\partial y} \end{pmatrix} \cdot \vec{n}\ dS$$

bezeichnet. Auch hier werden die Ableitungen auf den Kantenmittelpunkten des Kontrollvolumens zunächst als bekannt und konstant vorausgesetzt. Die Diskretisierung erfolgt dann wie beim konvektiven Term und es ergibt sich

$$\frac{1}{Re}\int_{\partial\Omega} \nabla u \cdot \vec{n}\ dS = h \cdot (\frac{\partial u_R}{\partial x} - \frac{\partial u_L}{\partial x} + \frac{\partial v_T}{\partial y} - \frac{\partial v_B}{\partial y}).$$

Werden die Ableitungen dann durch Differenzen angenähert, erhält man schließlich den diffusiven Term für ein beliebiges Kontrollvolumen mit Mittelpunkt im Knoten i, j:

$$\begin{aligned} \frac{1}{Re}\int_{\partial\Omega} \nabla u \cdot \vec{n}\ dS &= \frac{1}{Re}(\frac{u_{i+1,j} - u_{i,j}}{h} - \frac{u_{i,j} - u_{i-1,j}}{h} + \frac{u_{i,j+1} - u_{i,j}}{h} - \frac{u_{i,j} - u_{i,j-1}}{h}) \\ &= \frac{1}{hRe}(u_{i+1,j} + u_{i-1,j} - 4u_{i,j} + u_{i,j+1} + u_{i,j-1}). \end{aligned}$$

Weitere Terme

Für den Druckterm erübrigt sich die Diskretisierung. In der Studienarbeit [1] wurde nämlich gezeigt, dass eine Berücksichtigung des Druckterms nicht notwendig ist, wenn die Chorin-Projektion durchgeführt wird, wie es hier der Fall ist.

Es bleiben dann noch zwei Terme übrig. Dabei handelt es sich zum Einen um den Term, der die Beeinflussung durch äußere Kräfte beschreibt. Hier ist die Diskretisierung einfach.

Da nämlich die äußere Kraft im gesamten Simulationsgebiet und damit auch innerhalb eines Kontrollvolumens konstant ist, gilt:

$$\int_\Omega g_x dA = h^2 g_x.$$

Bei dem letzten Term handelt es sich um die linke Seite der Impulsgleichung. Hier wird bei der Diskretisierung wie für den obigen Term verfahren. Die Ableitung nach der Zeit t nimmt man als konstant auf einem Kontrollvolumen an und verschiebt deren Diskretisierung auf später. Entsprechend ergibt sich dann für die linke Seite der Impulsgleichung:

$$\int_\Omega \frac{\partial u}{\partial t} dA = h^2 \frac{\partial u}{\partial t}.$$

3.3.2 Diskretisierung der Kontinuitätsgleichung

Wie schon erwähnt wird für die Diskretisierung der Kontinuitätsgleichung eine andere Unterteilung in Kontrollvolumen benutzt als bei den Impulsgleichungen. Hier sind Kontrollvolumen und Zellen identisch. Wird nun die Geschwindigkeit auf einer Kante des Kontrollvolumens bzw. einer Zellkante durch Mittelung der Geschwindigkeiten an den beiden Endpunkten bestimmt und weiterhin als konstant auf der entsprechenden Kante angenommen, dann können die Ableitungen im Mittelpunkt des Kontrollvolumens durch Differenzen ausgedrückt werden und für die Kontinuitätsgleichung auf einem beliebigen Kontrollvolumen i, j gilt:

$$\begin{aligned}
&\int_{\partial\Omega} \begin{pmatrix} u \\ v \end{pmatrix} \cdot \vec{n}\, dS \\
&\quad = h\left(\frac{u_{i+1,j} + u_{i+1,j+1}}{2} - \frac{u_{i,j+1} + u_{i,j}}{2} + \frac{v_{i+1,j+1} + v_{i,j+1}}{2} - \frac{v_{i,j} + v_{i+1,j}}{2}\right) \\
&\quad = \frac{h}{2}(u_{i+1,j} + u_{i+1,j+1} - u_{i,j+1} - u_{i,j} + v_{i+1,j+1} + v_{i,j+1} - v_{i,j} - v_{i+1,j}). \qquad (3.4)
\end{aligned}$$

3.3.3 Kompakte Notation

Um die Eigenschaften der oben beschriebenen Diskretisierung leichter verdeutlichen zu können, wird für die räumlich diskretisierten Navier-Stokes-Gleichungen eine kompakte Darstellung eingeführt. Diese Darstellung wird erreicht, indem zunächst die Geschwindigkeits- und Druckwerte lexikographisch zu Vektoren

$$\begin{aligned}
u_h^T &= (u_0, u_1, \ldots, u_n, v_0, v_1, \ldots, v_n)^T \qquad \text{und} \\
p_h^T &= (p_0, p_1, \ldots, p_m)^T
\end{aligned}$$

zusammengefasst werden. Dabei wird mit dem Wert am linken unteren Rand des Simulationsgebietes begonnen und die weiteren Werte dann zeilenweise, jeweils von links nach rechts, in den entsprechenden Vektor übernommen. Durch das Einführen von Matrizen können dann die räumlich diskretisierten Navier-Stokes-Gleichungen für sämtliche Kontrollvolumen einfach zusammengefasst werden und es ergibt sich:

$$\begin{aligned}
\mathbf{\Omega} \frac{\partial}{\partial t} \vec{u_h} &= -\mathbf{C}(\vec{u_h})\vec{u_h} + \mathbf{D}\vec{u_h} + \vec{g_h} \qquad (3.5) \\
\mathbf{M}\vec{u_h} &= 0. \qquad (3.6)
\end{aligned}$$

Dabei sind die Matrizen $\mathbf{C}(\vec{u_h})$ und $\mathbf{D}$ so konstruiert, dass sie die konvektiven bzw. diffusen Terme wie in den oberen Abschnitten beschrieben berechnen. $\mathbf{\Omega}$ ist eine Diagonalmatrix, in der die Diagonalelemente durch die Faktoren, die bei der Diskretisierung der linken Seite entstanden sind, bestimmt werden. Entsprechend enthält die Diagonale nur den Wert h^2. $\mathbf{M}$ beschriebt die Diskretisierung der Kontinuitätsgleichung und der Vektor $\vec{g_h}$ schließlich repräsentiert die Einwirkung von äußeren Kräften auf das System. Seine Einträge sind dabei entsprechend Abschnitt 3.3.1 je nach Impulsgleichung entweder $h^2 g_x$ oder $h^2 g_y$.

3.3.4 Eigenschaften der räumlichen Diskretisierung

Hier soll kurz verdeutlicht werden, warum die Diskretisierung so wie oben beschrieben durchgeführt wurde. Zunächst einmal sollen beim Übergang von den kontinuierlichen zu den diskretisierten Gleichungen elementare physikalische Eigenschaften des Systems erhalten bleiben. Dabei ist zum Einen darauf zu achten, dass die Impulserhaltung erfüllt ist, zum Anderen muss sichergestellt werden, dass keine Energie vom System produziert wird. Zusätzlich sollte außerdem noch eine adaptive kontinuitätserhaltende Gitterunterteilung möglich sein, um eine Verfeinerung des Simulationsgebietes auch adaptiv vornehmen zu können. Von der Möglichkeit der adaptiven Verfeinerung wird aber im Moment noch kein Gebrauch gemacht.

Der Beweis, dass die räumliche Diskretisierung all diese Anforderungen erfüllt wird hier nicht geführt. Er kann unter anderem in [2], [3] oder der Studienarbeit [1] nachgelesen werden.

3.4 Zeitliche Diskretisierung

Mit der zeitlichen Diskretisierung wird nun die Umwandlung der kontinuierlichen Navier-Stokes-Gleichungen abgeschlossen. Da zum Zeitpunkt t_0 Druck- und Geschwindigkeitsfeld, sowie deren zeitliche Änderung gegeben ist, handelt es sich hierbei offensichtlich um ein Anfangswertproblem. Anfangswertprobleme können unter anderem mit dem Euler-Verfahren gelöst werden, bei dem es sich um das wohl einfachste Verfahren zum Lösen von Anfangswertproblemen handelt. Es lässt sich folgendermaßen beschreiben: Ist zum Zeitpunkt t_i ein Funktionswert y_i sowie dessen Änderung y_i' bekannt, dann kann der Funktionswert zum Zeitpunkt t_{i+1} berechnet werden und es gilt:

$$\begin{aligned} y_{i+1} &= y_i + \delta t y_i \\ t_{i+1} &= t_i + \delta t. \end{aligned}$$

Die Konvergenzordnung des Euler-Verfahrens beträgt dabei nur $O(h)$. Im Vergleich mit anderen Verfahren, wie zum Beispiel den Runge-Kutta-Verfahren, schneidet es daher schlecht ab. Ausserdem existieren noch diverse numerische Probleme. Im vorliegenden Fall fiel die Wahl aber dennoch auf das Euler-Verfahren, da man diese Probleme hier durch eine Beschränkung der Schrittweite wie in der Studienarbeit [1] beschrieben, in den Griff bekommen kann und weil das Euler-Verfahren, wie bereits erwähnt, sehr einfach zu realisieren ist.

Überträgt man nun das Euler-Verfahren auf das aktuelle Problem, ergibt sich die folgende Formulierung. Ist zum Zeitpunkt t das Geschwindigkeitsfeld $\vec{u}_{ht}$ bekannt, dann kann mit Hilfe

der Ableitung aus Gleichung 3.5 das Feld zum Zeitpunkt $t + \delta t$ berechnet werden und es gilt:

$$\vec{\tilde{u}}_{ht+\delta t} = \vec{u}_{ht} + \delta t \cdot \mathbf{\Omega}^{-1}(-\mathbf{C}(\vec{u}_h)\vec{u}_h + \mathbf{D}\vec{u}_h + \vec{g}_h).$$

Bei $\vec{\tilde{u}}_{ht+\delta t}$ handelt es sich aber noch nicht um das endgültige Geschwindigkeitsfeld zum Zeitpunkt $t + \delta t$. Normalerweise wird nämlich bei der Durchführung eines Zeitschritts die Divergenzfreiheit des Geschwindigkeitsfeldes zerstört und muß dann erst wieder hergestellt werden. Dazu wird das Geschwindigkeitsfeld durch eine Chorin-Projektion in den Unterraum der divergenzfreien Felder projeziert. Wie das vonstatten geht, ist im nächsten Abschnitt beschrieben.

3.5 Chorin-Projektion

Die Chorin-Projektion[6] dient zum Herstellen der Divergenzfreiheit von Vektorfeldern. Das wird erreicht, indem für das nicht divergenzfreie Feld eine Projektion in den Unterraum der divergenzfreien Vektorfelder durchgeführt wird. Das Verfahren wird im Folgenden anhand von [5] kurz erläutert.

Zentraler Bestandteil der Chorin-Projektion ist das Bestimmen einer Lösung für die sogenannte Poisson-Gleichung

$$\Delta p = -\mathrm{div}\vec{u}.$$

Dabei wird für ein in der Regel nicht divergenzfreies Vektorfeld $\vec{u}$ ein Druckfeld p bestimmt. Wird nun mit Hilfe von p ein neues Vektorfeld

$$\vec{\hat{u}} = \vec{u} + \nabla p$$

berechnet, so ist dieses wegen

$$\mathrm{div}\vec{\hat{u}} = \mathrm{div}(\vec{u} + \nabla p) = \mathrm{div}\vec{u} + \mathrm{div}\nabla p = \mathrm{div}\vec{u} + \Delta p = 0$$

divergenzfrei. Wie in [4] gezeigt wird, existiert dabei für die Poisson-Gleichung immer eine Lösung, wenn die integrale Form der Kontinuitätsgleichung

$$\int_\Omega \mathrm{div}\vec{u}\ dA = 0$$

erfüllt ist. Diese Bedingung entspricht der Forderung, dass im Vektorfeld ∇p kein Massentransport über die Gebietskanten stattfindet. Bei Addition von ∇p zu $\vec{u}$ ändert sich also nichts an der Menge des herein- und herausströmenden Fluids, was durchaus sinnvoll erscheint, da so sichergestellt wird, dass das Fluid nicht aus dem Simulationsgebiet verschwindet oder sich dort ansammeln kann.

3.5.1 Diskretisierung der Poisson-Gleichung

Um eine Lösung für die Poisson-Gleichung bestimmen zu können, muss auch hier wieder diskretisiert werden. Dazu wird die Poisson-Gleichung nicht mehr für jeden Punkt des Simulationsgebietes erfüllt, sondern nur noch auf den einzelnen Zellen. Für jede Zelle muss dann also die Gleichung

$$\Delta p = -\mathrm{div}\vec{u} \qquad \Leftrightarrow \qquad \mathrm{div}\nabla p = -\mathrm{div}\vec{u}$$

erfüllt sein. Man sieht, dass sowohl auf der linken als auch auf der rechten Seite die Divergenz eines Geschwindigkeitsfeldes [1] berechnet wird. Um sich nun keine Probleme einzuhandeln, wird für die Berechnung der Divergenzen auf der linken und rechten Seite der Gleichung jeweils die gleiche Methode entsprechend Abschnitt 3.3.2 verwendet. Die rechte Seite ergibt sich dann einfach analog zu Gleichung 3.4. Dagegen müssen für die linke Seite erst noch die Werte des Geschwindigkeitsfeldes ∇p an den Gitterknoten bestimmt werden. Die horizontale

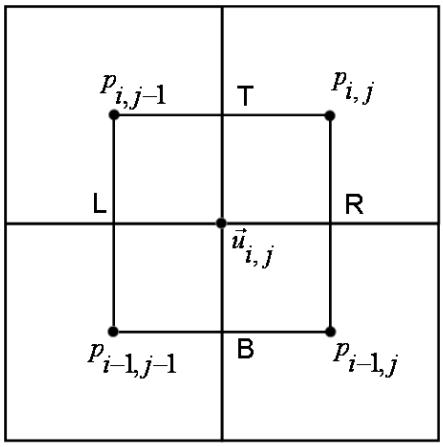

Abbildung 3.4: Knoten i, j mit umliegenden Druckwerten

Geschwindigkeitskomponente $up_{i,j}$ von ∇p in einem beliebigen Knoten i, j wird dabei entsprechend Abbildung 3.4 approximiert, indem man zuerst die Geschwindigkeit in den Punkten T und B als Ableitung des Drucks mit Hilfe der zentralen Differenzen bestimmt und dann den Mittelwert bildet. Das Vorgehen zur Bestimmung der vertikalen Komponente $vp_{i,j}$ gestaltet sich analog und für die Komponenten ergibt sich:

$$\begin{aligned} up_{i,j} &= \frac{1}{2}\left(\frac{p_{i,j} - p_{i-1,j}}{h} + \frac{p_{i,j-1} - p_{i-1,j-1}}{h}\right) \\ vp_{i,j} &= \frac{1}{2}\left(\frac{p_{i,j} - p_{i,j-1}}{h} + \frac{p_{i-1,j} - p_{i-1,j-1}}{h}\right). \end{aligned}$$

Damit können jetzt beide Seiten der Poisson-Gleichung für ein beliebiges Kontrollvolumen i, j angegeben werden. Für die rechte Seite ergibt sich dabei

$$-\mathrm{div}\vec{u} = \frac{h}{2}(u_{i+1,j} + u_{i+1,j+1} + v_{i+1,j+1} + v_{i,j+1} - u_{i,j+1} - u_{i,j} - v_{i,j} - v_{i+1,j}).$$

Die linke Seite wird zunächst zu

$$\begin{aligned} \Delta p &= \mathrm{div}(\nabla p) = \mathrm{div}\overrightarrow{up} \\ &= \frac{h}{2}(up_{i+1,j} + up_{i+1,j+1} + vp_{i+1,j+1} + vp_{i,j+1} - up_{i,j+1} - up_{i,j} - vp_{i,j} - vp_{i+1,j}). \end{aligned}$$

[1] Auch ∇p ist ein Geschwindigkeitsfeld. Die Geschwindigkeit wird nämlich durch die Druckänderungen $\frac{\partial p}{\partial x}$ und $\frac{\partial p}{\partial y}$ in x- bzw. y-Richtung beschrieben, was gerade $(\nabla p)^T = (\frac{\partial p}{\partial x}, \frac{\partial p}{\partial y})$ entspricht.

Durch Einsetzen der Geschwindigkeiten und nachfolgendes Zusammenfassen kann die linke Seite schließlich umgeformt werden zu

$$\Delta p = \frac{1}{4}(2p_{i-1,j-1} + 2p_{i+1,j-1} - 8p_{i,j} + 2p_{i-1,j+1} + 2p_{i+1,j+1})$$

und der korrespondierende Differenzenstern hat die Form

$$\frac{1}{4}\begin{pmatrix} 2 & 0 & 2 \\ 0 & -8 & 0 \\ 2 & 0 & 2 \end{pmatrix}.$$

Das setzt voraus, dass für die Berechnung von Δp in einer Zelle alle diagonalen Nachbarn existieren. Am Rand ist das natürlich nicht der Fall und es können sich dafür dann je nach Verlauf unter anderem die Differenzensterne

$$\frac{1}{4}\begin{pmatrix} \cdot & 0 & 2 \\ \cdot & -2 & 0 \\ \cdot & \cdot & \cdot \end{pmatrix}, \qquad \frac{1}{4}\begin{pmatrix} 2 & 0 & 2 \\ 0 & -4 & 0 \\ \cdot & \cdot & \cdot \end{pmatrix} \qquad \text{und} \qquad \frac{1}{4}\begin{pmatrix} 2 & 0 & 2 \\ 0 & -6 & 0 \\ \cdot & 0 & 2 \end{pmatrix}.$$

ergeben.

Berechnet man nun für jede Zelle die negative Divergenz nach Gleichung 3.7 und sortiert diese Werte wie in Abschnitt 3.3.3 beschrieben lexikographisch in einen Vektor f_h ein, dann kann man die diskrete Poissongleichung schreiben als

$$\mathbf{P}\vec{p}_h = \vec{f}_h,$$

mit der Poisson-Matrix $\mathbf{P}$, die den Laplace-Operator wie oben beschrieben diskretisiert und dem bereits bekannten Druckvektor p_h.

Bei genauerer Betrachtung der Differenzensterne fällt auf, dass die Matrix $\mathbf{P}$ dünn besetzt sein muss und sich nur auf der Haupt- sowie jeweils zwei Nebendiagonalen von Null verschiedene Einträge befinden. Der Abstand zwischen Haupt- und Nebendiagonalen kann angegeben werden, wenn bekannt ist, in wieviele Zellen x, das Simulationsgebiet in horizontaler Richtung unterteilt wurde. Der Abstand zur ersten Nebendiagonalen beträgt dann nämlich $x-1$ und der Abstand zur zweiten Nebendiagonalen $x+1$ Matrixeinträge. Das ist zum Einen interessant, da bei der Implementierung von Matrixoperationen eventuell die Diagonalgestalt ausgenutzt werden kann, zum Anderen müssen auch nur die Diagonalen und nicht die komplette Matrix gespeichert werden.

3.5.2 Spezielle Eigenschaften der Diskretisierung

Wenn man die Poisson-Gleichung wie eben beschrieben diskretisiert, ergeben sich unter anderem interessante Sachverhalte bezüglich Lösbarkeit und Gestalt, die für die Randbehandlung von besonderer Bedeutung sind.

Zerlegung des Gleichungssystems

Zunächst wird auf die Zerlegung des Gleichungssystems eingegangen. Dazu stellt man sich die Zellen im Simulationsgebiet schachbrettmusterartig in weiße und schwarze Zellen unterteilt vor. Eine Betrachtung der obigen Differenzensterne ergibt dann, dass für die Berechnung von Δp in einer schwarzen Zelle nur Werte schwarzer Zellen verwendet werden und entsprechendes auch für die Berechnung von Δp in einer weißen Zelle gilt. Das bedeutet, dass die Poissongleichung von oben aus zwei voneinander entkoppelten Linearen Gleichungssystemen besteht, die sich getrennt schreiben lassen als

$$\begin{aligned} \mathbf{P_b}\vec{p}_b &= -\overrightarrow{bdiv} \qquad \text{für die schwarzen Zellen und} \\ \mathbf{P_w}\vec{p}_w &= -\overrightarrow{wdiv} \qquad \text{für die weißen Zellen.} \end{aligned}$$

Dabei sind die Divergenzen und Druckwerte der Zellen ihrer Farbe entsprechend auf die Vektoren $\overrightarrow{bdiv}$ und $\overrightarrow{wdiv}$ bzw. $\vec{p}_b$ und $\vec{p}_w$ verteilt worden. Analog ergeben sich auch die Matrizen $\mathbf{P_b}$ und $\mathbf{P_w}$ aus den Zeilen der Poisson-Matrix $\mathbf{P}$. Gegenüber der Matrix $\mathbf{P}$ hat sich die Größe dieser Matrizen ungefähr halbiert, da prinzipiell nur jede zweite Zeile übernommen wird und aus diesen Zeilen die Einträge für die Zellen der jeweilig anderen Farbe entfernt werden. Wegen der Entkopplung werden aber automatisch nur Einträge weggelassen, die sowieso Null sind und es gehen dadurch keine Information verloren. Die beiden verkleinerten Matrizen sind dann wie die Matrix $\mathbf{P}$ ebenfalls dünn besetzt, besitzen aber im Allgemeinen ihre spezielle Diagonalgestalt nicht mehr.

Diese Aufspaltung in zwei getrennte Gleichungssysteme kann beim Lösen des Gleichungssystems unter Umständen ausgenutzt werden. Wird zur Lösung zum Beispiel der Gauss-Algorithmus verwendet, beträgt der Aufwand für n Unbekannte $O(n^3)$. Werden aber nun stattdessen zwei Gleichungssysteme mit jeweils $\frac{n}{2}$ Unbekannten gelöst, so ergibt sich der Aufwand zu

$$O(\frac{n}{2}^3) + O(\frac{n}{2}^3) = 2 \cdot O(\frac{n^3}{8}) = \frac{1}{4}O(n^3).$$

Der Aufwand zur Lösung des Gleichungssystems ist also auf ein Viertel gesunken. Fairerweise muss aber gesagt werden, dass bei einem schnelleren Verfahren, wie dem hier verwendeten Gauss-Seidel-Algorithmus, die Zerlegung keine Vorteile mehr bringt. Da die Matrizen $\mathbf{P_b}$ und $\mathbf{P_w}$ dünn besetzt sind, beträgt der Aufwand zur Lösung des großen Gleichungssystems nämlich $O(n)$ und für die beiden kleinen Gleichungssysteme dann jeweils $O(\frac{n}{2})$. Da aber aktuelle Rechner gleich mehrere Prozessorkerne auf einem Chip vereinen, oder vielleicht sogar mehrere physikalische Prozessoren besitzen, stellt eine parallele Lösung dieser beiden kleinen Gleichungssysteme kein Problem dar und man hat eine Beschleunigung um den Faktor zwei erreicht.

Lösbarkeit

Um nun Aussagen über die Lösbarkeit der beiden Gleichungssysteme zu treffen, werden zunächst deren Matrizen $\mathbf{P_b}$ und $\mathbf{P_w}$ betrachtet. Wie man sich anhand der Differenzensterne klarmachen kann, besitzen beide Matrizen jeweils einen Eigenvektor zum Eigenwert

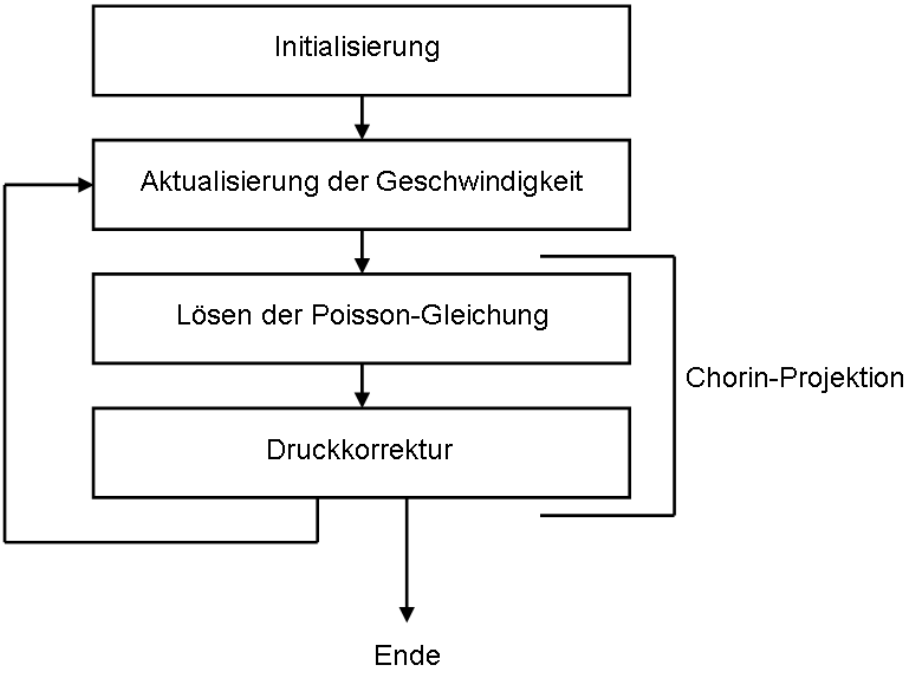

Abbildung 3.5: Grober Ablauf des Algorithmus

Null, dessen Einträge nur aus Einsen bestehen. Die Gleichungssysteme sind nun jeweils genau dann lösbar, wenn das Skalarprodukt aus Eigenvektor und rechter Seite Null ist. Da die Komponenten der Eigenvektoren alle den gleichen Wert haben, ist das gleichbedeutend mit der Forderung, dass die Summen über die Komponenten der Vektoren $\overrightarrow{bdiv}$ und $\overrightarrow{wdiv}$ verschwinden. Es muss also

$$\begin{aligned} \sum_{i=1}^{n_b} bdiv_i &= 0 \qquad \text{bzw.} \\ \sum_{i=1}^{n_w} wdiv_i &= 0 \end{aligned}$$

erfüllt sein. In Kapitel 4 wird man sehen, dass das Herstellen dieser Bedingungen den zentralen Bestandteil der Randbehandlung bildet. Für den Beweis sei auf Hackbusch, W. [4] verwiesen, er wird dort anhand eines ähnlichen Problems geführt.

3.6 Ablauf

Um nach all den obigen Details wieder den Gesamtüberblick zu gewinnnen, wird jetzt zum Abschluss des Kapitels nocheinmal der grobe Ablauf des Algorithmus beschrieben, wie er auch in Abbildung 3.5 dargestellt ist. Zunächst werden die Felder für Druck und Geschwindigkeit intitialisiert, das Ende der Simulation durch t_{end} festgelegt und die aktuelle Zeit auf

Null gesetzt. Danach erfolgt die Aktualisierung des Geschwindigkeitsfelds mit Hilfe der diskretisierten Impulsgleichungen. Das hat zur Folge, dass die Divergenzfreiheit, wie sie von der Kontinuitätsgleichung gefordert wird, nicht mehr erfüllt ist. Durch Lösen der Poisson-Gleichung während der Chorin-Projektion bestimmt man daher Druckwerte, mit deren Hilfe dann die Divergenzfreiheit des Geschwindigkeitsfeldes wiederhergestellt wird. Schließlich wird der Zeitschritt δt bestimmt und die Zeit aktualisiert. Ist dabei das Ende der Simulation noch nicht erreicht, erfolgt ein weiterer Durchlauf.

Kapitel 4

Randbehandlung

In diesem Kapitel ist die Durchführung der Randbehandlung beschrieben. Dabei wird zunächst darauf eingegangen, wie genau die Randwerte die Lösbarkeit beeinflussen. Danach wird kurz die klassische Methode erläutert und schließlich folgt eine ausführliche Beschreibung der verbesserten Randbehandlung.

4.1 Etwas Terminologie

Die im Folgenden verwendeten Begriffe lassen sich am Besten anhand von Abbildung 4.1 erläutern. Dort ist ein kleines Simulationsgebiet mit einem Hindernis im Inneren dargestellt in dem jeder Knoten- bzw. Zelltyp auftaucht. Die Zellen lassen sich dann in drei Klassen unterteilen und zwar in die weiß eingfärbten Fluidzellen, die schraffierten Randzellen, sowie die grauen Hinderniszellen. Die Randknoten sind schwarz eingefärbt und befinden sich entweder im Hindernis oder genau auf dessen Rand. Zusammen bilden sie die innere Berandung des Hindernisses, dargestellt durch die dicke schwarze Linie. Die restlichen Knoten sind nicht weiter gekennzeichnet, es handelt sich dabei entweder um Hindernis- oder Fluidknoten.

4.2 Lösbarkeit der Poisson-Gleichung

Wie bereits im letzten Kapitel gezeigt wurde, müssen zur Durchführung der Chorin-Projektion die beiden Linearen Gleichungssysteme

$$\begin{aligned} \mathbf{P}_b \vec{p}_b &= \overrightarrow{bdiv} \qquad \text{und} \\ \mathbf{P}_w \vec{p}_w &= \overrightarrow{wdiv} \end{aligned}$$

gelöst werden. Das ist aber nur möglich, wenn die Summen über die jeweiligen Divergenzen verschwinden, wenn also die Bedingungen

$$\sum_{i=1}^{n_b} bdiv_i = 0 \qquad \text{und} \tag{4.1}$$

$$\sum_{i=1}^{n_b} wdiv_i = 0 \tag{4.2}$$

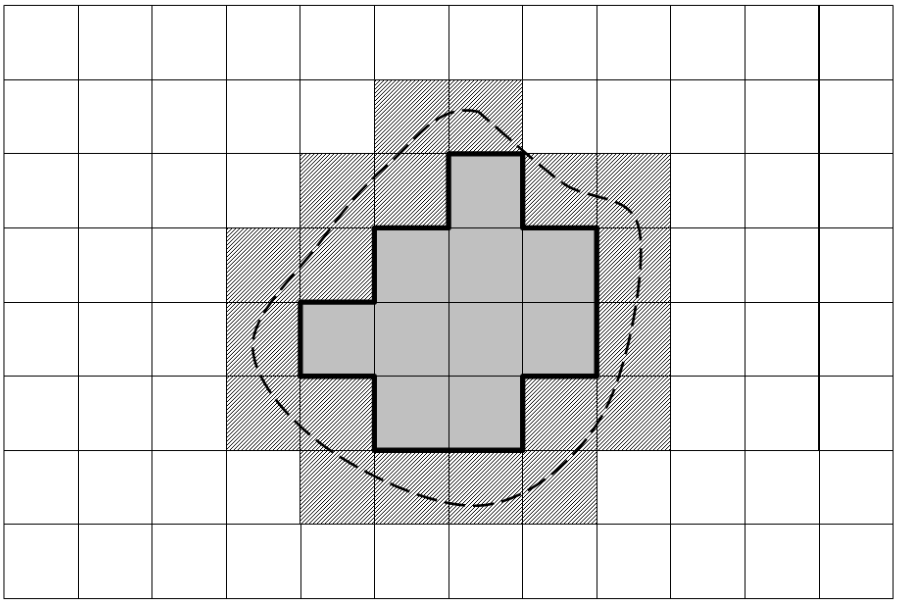

Abbildung 4.1: Simulationsgebiet mit Hindernis

erfüllt sind. Die Anforderungen an das Geschwindigkeitsfeld, die sich daraus ergeben, werden in den folgenden Abschnitten erörtet. Dabei reicht es aus, die Anforderungen beispielhaft anhand der schwarzen Zellen zu bestimmen, da sie sich unverändert auf die weißen Zellen übertragen lassen.

4.2.1 Innere Knoten

Zunächst wird er Einfluss der inneren Knoten auf Gleichung 4.1 untersucht. Wie in Abbildung 4.2 dargestellt ist, sind in einem inneren Knoten jeweils zwei gleichfarbige Zellen über Eck miteinander verbunden. Dabei beeinflusst ein innerer Knoten ausschließlich die Divergenz der beiden angrenzen Zellen. Im dargestellten Fall ergibt sich für die Summe der Divergenzen mit besonderem Augenmerk auf die Zellen $i-1, j-1$ und i, j dann

$$\sum_{i=1}^{n_b} bdiv_i =$$
$$\frac{h}{2}(\ldots + \underbrace{(-u_{i-1,j-1} + u_{i,j-1} + u_{i,j} - u_{i-1,j} - v_{i-1,j-1} - v_{i,j-1} + v_{i,j} + v_{i-1,j})}_{\text{Divergenz von Zelle i-1, j-1}} + \ldots$$
$$\ldots + \underbrace{(-u_{i,j} + u_{i+1,j} + u_{i+1,j+1} - u_{i,j+1} - v_{i,j} - v_{i+1,j} + v_{i+1,j+1} + v_{i,j+1})}_{\text{Divergenz von Zelle i, j}} + \ldots).$$

Man erkennt, dass sich die Summanden $u_{i,j}$ und $-u_{i,j}$ sowie $v_{i,j}$ und $-v_{i,j}$ gegenseitig aufheben. Da diese Summanden ausschließlich in den beiden dargestellten Divergenztermen vorkommen, ist die Gesamtsumme der Divergenzen von der Geschwindigkeit im Knoten i, j unabhängig. Diese Aussage gilt analog auch für die restlichen inneren Knoten, in denen sich die linke obere und die rechte untere Ecke zweier Zellen treffen. Zusammenfassend lässt sich also sagen, dass Knoten in denen sich zwei gleichfarbige Zellen berühren, keinen Einfluß auf die Divergenzsumme haben und daher die Forderung 4.1 nicht zerstören können.

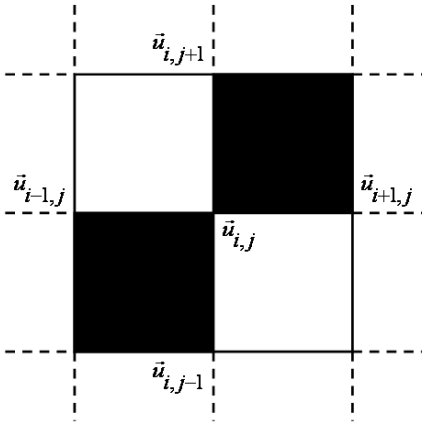

Abbildung 4.2: Zellkonstellation an einem inneren Knoten

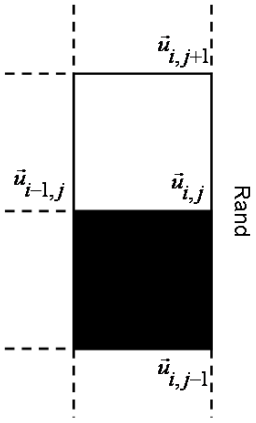

Abbildung 4.3: Zellkonstellation an einem Randknoten

4.2.2 Randknoten

Nun wird der Einfluß der Randknoten auf die Summe der Divergenzen behandelt. Da im Moment nur schwarze Zellen betrachtet werden, wird ein Randknoten entsprechend Abbildung 4.3 nur von einer Zelle berührt. Für die Zellkonstellation aus der Abbildung kann man sich wieder die Summe der Divergenzen wie oben aufschreiben und erhält dann

$$\sum_{i=1}^{n_b} bdiv_i = \frac{h}{2}(\ldots + \underbrace{(-u_{i-1,j} + u_{i,j} + u_{i,j+1} - u_{i-1,j+1} - v_{i-1,j} - v_{i,j} + v_{i,j+1} + v_{i-1,j+1})}_{\text{Divergenz von Zelle i-1, j}} + \ldots).$$

Der Knoten i, j leistet also bei der dargestellten Konstellation den Beitrag $u_{i,j} - v_{i,j}$ zur Divergenzsumme. Allgemein lässt sich feststellen, dass ein beliebiger Knoten i, j, der sich auf dem unverbundenen Eckpunkt einer Zelle befindet, den Beitrag $wu_{i,j}u_{i,j} + wv_{i,j}v_{i,j}$ zur Summe der Divergenzen leistet. Wie in Abbildung 4.4 gezeigt wird, sind die Gewichte $wu_{i,j}$ und $wv_{i,j}$ dabei von der Position des Knotens in der Zelle abhängig und ergeben sich direkt aus der Berechnungsvorschrift zur Bestimmung der Divergenz aus Abschnitt 3.3.2.

Unter Einbeziehung der Erkenntnisse für die inneren Knoten folgt, dass ausschließlich die

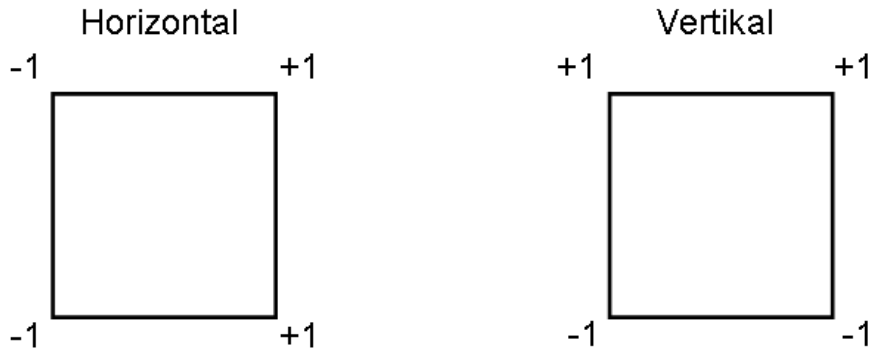

Abbildung 4.4: Gewichtung der Knoten in Abhängigkeit von der Zellposition

Geschwindigkeit in den Randknoten entscheidet, ob die Forderungen 4.1 und 4.2 für schwarze bzw. weiße Zellen erfüllt sind oder nicht.

4.3 Gebietsgrenzen

Betrachtet man ein Gebiet ohne Hindernisse im Inneren, so werden die Divergenzsummen der schwarzen und weißen Zellen, wie gerade gezeigt wurde, nur von den Knoten auf dem Gebietsrand beeinflusst. Um die Lösbarkeit der Poisson-Gleichung sicherzustellen, müssen sich die Beiträge, die diese Knoten zu den jeweiligen Divergenzsummen leisten, aufheben. Wie das geschehen kann, wird dabei kurz anhand eines horizontal durchströmten Kanals erläutert für dessen Ränder die sogenannte Haftbedingung gelten soll. Damit ist gemeint, dass am Kanalrand keine Fluidbewegung stattfindet. Die Kanalwände werden daher durch Nullsetzen der Geschwindigkeit an den oberen und unteren Begrenzungsknoten modelliert. Durch Setzen der horizontalen Komponenten am linken und rechten Rand kann man dann Fluid in den Kanal hinein- bzw. herausströmen lassen. Die vertikalen Komponenten belässt man auf Null, da der Kanal nur horizontal durchströmt werden soll. Bezeichnet man die horizontalen Komponenten am linken Rand mit i_k und die am rechten Rand mit o_k, dann gilt für die Summe der Divergenzen:

$$\begin{aligned} \sum_{k=0}^{n_i} bdiv_k &= -\sum_k i_k + \sum_k o_k \\ \sum_{k=0}^{n_o} wdiv_k &= -\sum_k i_k + \sum_k o_k. \end{aligned}$$

Die übrigen Randknoten leisten keinen Beitrag, da dort die Geschwindigkeit auf Null gesetzt wurde. Die Gewichtung der Komponenten erfolgt entsprechend Abbildung 4.4 am linken Rand negativ und am rechten Rand positiv. Daraus ergeben sich dann die obigen Summen, die in diesem Fall ausnahmsweise identisch sind. Das liegt hier an der Kombination von vertikalem Rand mit horizontaler Geschwindigkeit, im Allgemeinen sind die Summen aber verschieden.

Die Lösbarkeit der Poisson-Gleichung wird dann sinnvollerweise dadurch garantiert, dass die Summe der Einströmgeschwindigkeiten und die Summe der Ausströmgeschwindigkeiten gleich ist.

4.4 Die klassische Randbehandlung

4.4.1 Vorgehensweise

Wird ein Hindernis wie in Abbildung 4.1 in das Simulationsgebiet gesetzt, dann werden die Hinderniszellen, die dabei entstehen beim Lösen der Gleichungssysteme nicht mehr berücksichtigt. Dadurch entsteht ein neuer Rand im Gebiet, der bereits in Abschnitt 4.1 erwähnte sogenannte innere Rand. Geht man davon aus, dass die Geschwindigkeit an den Gebietsgrenzen wie oben beschrieben sinnvoll gesetzt wurde, dann kann die Forderung nach einem Verschwinden der Divergenzsummen nur noch am inneren Rand zerstört werden. Das Verschwinden der Divergenzsummen bei der klassischen Methode stellt man dann einfach dadurch sicher, dass man die Geschwindigkeit an allen inneren Knoten auf Null setzt.

4.4.2 Bewertung

Das Problem der klassischen und auch jeder anderen Art von Randbehandlung ist nun die Tatsache, dass der Rand üblicherweise nicht entlang der Zellkanten verläuft, sondern durch die Zellen hindurch geht. Beim Aktualisieren des Geschwindigkeitsfeldes führt das zu Problemen, da für die Berechnung der neuen Geschwindigkeit in einem Knoten, jeweils die Werte der umliegenden Knoten verwendet werden. Wird diese Aktualisierung nun für einen Fluidknoten in Randnähe durchgeführt, dann wird auch die Geschwindigkeit von Knoten im Hindernis benötigt. Die Kunst der Randbehandlung besteht nun darin, für diese Knoten, bei denen es sich hier um die inneren Knoten handelt, die Werte so zu setzen, dass der richtige Rand möglichst gut approximiert wird.

Wie schon erwähnt, werden die inneren Knoten bei der klassischen Randbehandlung alle auf Null gesetzt. Da aufgrund der Haftbedingung die Geschwindigkeit am Rand verschwinden soll, hat das zur Folge, dass der simuliert Rand mit dem inneren Rand zusammenfällt. Um den Fehler, der dabei entsteht, möglichst gering zu halten ist es notwendig den Randbereich höher aufzulösen. Da hier kein adaptives Verfahren verwendet wird, ist das gleichbedeutend mit der Erhöhung der Auflösung im gesamten Simulationsgebiet, was wiederum den Zeitaufwand für die Simulation stark in die Höhe treibt.

An dieser Stelle setzt nun die verbesserte Randbehandlung an. Statt die Auflösung zur Verbesserung der Simulationsergebnisse zu erhöhen wird einfach versucht den Rand besser zu approximieren. Dazu werden die Geschwindigkeitswerte an den Randknoten im Gegensatz zum klassischen Verfahren nicht auf Null gesetzt, sondern anderweitig angepasst. Durch eine geschickte Wahl dieser Werte erhofft man sich nun eine Verbesserung der Randapproximation und damit einhergehend eine Verbesserung der Simulationsergebnisse. Diese Idee der verbesserten Randapproximation ist nicht neu und wurde unter anderem schon von G.H. Shortley und R. Weller [7], H. Forrer [8] und M.F. Tome und S. McKee [9] formuliert.

4.5 Die verbesserte Randbehandlung

Die verbesserte Randbehandlung lässt sich in zwei Teilschritte zerlegen. Im ersten Schritt, der eigentlichen Randbehandlung, werden zunächst die Werte in den Randknoten so gesetzt, dass die Randapproximation möglichst gut ist. Das hat höchstwahrscheinlich zur Folge, dass die beiden Divergenzsummen nicht mehr verschwinden und somit auch für die Poisson-Gleichung keine Lösung mehr existiert. Deshalb müssen bei der Randkorrektur im zweiten Schritt die Werte auf den Randknoten so korrigiert werden, dass das Verschwinden der beiden Divergenzsummen sichergestellt ist. Dabei sollte die Änderung der Randwerte möglichst klein gehalten werden, um die Randapproximation aus dem ersten Schritt nicht zu zerstören.

4.5.1 Eigentliche Randbehandlung

Um das Verfahren einfach zu halten, werden für die Randbehandlung nur Randknoten betrachtet, die mindestens einen direkten Nachbarknoten besitzen, der innerhalb des Fluids liegt. Wie in Abbildung 4.1 zu sehen ist gibt es durchaus auch Randknoten, deren direkte Nachbarn alle innerhalb des Hindernisses liegen. Diese Knoten werden nicht weiter behandelt und ihre Geschwindigkeit auf Null gesetzt. Für die anderen Randknoten ergibt sich dann jeweils eine der fünf Konstellationsmöglichkeiten aus Abbildung 4.5. Dabei wurden Fall 2.1

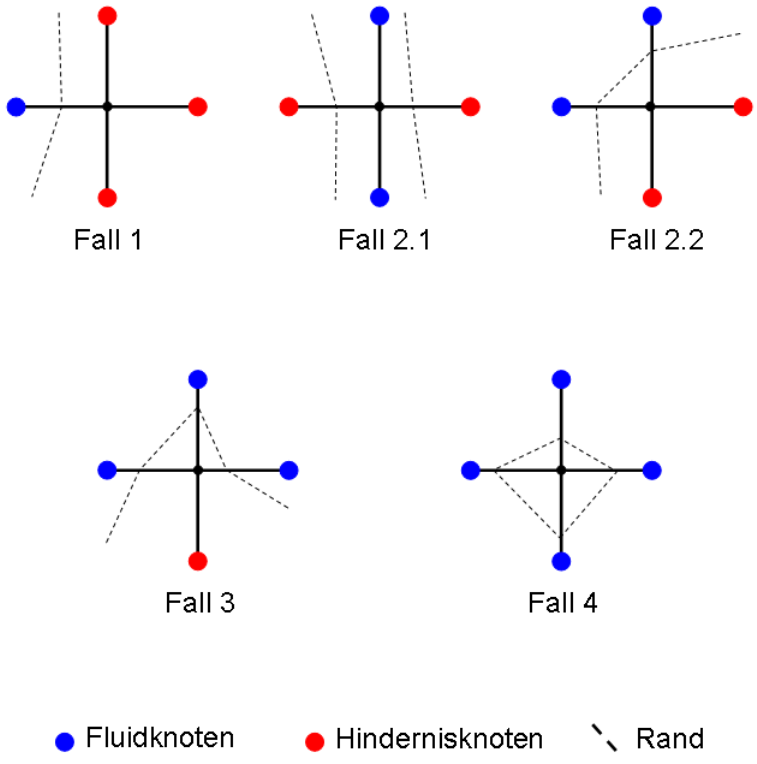

Abbildung 4.5: Konstellationsmöglichkeiten am Rand

Name	Vorschrift
ZERO	f wird ohne Beachtung der extrapolierten Werte auf Null gesetzt
AVERAGE	f wird durch das arithmetische Mittel der ex_i bestimmt
SEPARATE	Die horizontale Komponente von f wird durch Mittelung der in x-Richtung extrapolierten Werte bestimmt. Für die vertikale Komponente wird analog vorgegangen.
SWITCHED_SEPARATE	Die horizontale Komponente von f wird durch Mittelung der in y-Richtung extrapolierten Werte bestimmt. Für die vertikale Komponente wird analog vorgegangen.

Tabelle 4.1: Kombinationsmöglichkeiten zur Konfliktauflösung

und Fall 4 nur der Vollständigkeit halber aufgeführt. In der Regel sind die Simulationsgebiete nämlich so fein diskretisiert, dass diese Fälle gar nicht erst auftreten. Ausserdem hat sich gezeigt, dass der erste Fall in der Realität sehr viel häufiger anzutreffen ist als die Fälle 2.2 und 3.

Zur Bestimmung der Geschwindigkeit in einem Randknoten werden nun sämtliche Nachbarknoten betrachtet, die sich im Fluid befinden. Dazu wird für jeden dieser Knoten ein eigener Wert extrapoliert, der abhängig vom Wert des Fluidknotens und dem Schnittpunkt des Randes mit der dazugehörigen Zellkante ist. Wie genau diese Extrapolation vonstatten geht ist dabei im nächsten Abschnitt beschrieben und soll momentan noch nicht von näherem Interesse sein.

Wurde also eine Extrapolationsvorschrift angewendet, so existiert für den ersten Fall aus Abbildung 4.5 offensichtlich nur ein Wert, der unverändert am Randknoten übernommen wird. In den anderen Fällen liegen je nach Knotenkonstellation zwischen zwei und vier Werten für einen Randknoten vor. Der Wert f am Randknoten wird dann bestimmt, indem die extrapolierten Werte ex_i mit Hilfe einer der Kombinationsvorschriften aus Tabelle 4.5.1 zusammengefasst werden.

Wie stark das Simulationsergebnis dabei von der Wahl der Kombinationsvorschrift abhängt, lässt sich im Voraus nur schwer sagen. Man könnte meinen, dass so gut wie keine Beeinflussung besteht, da der erste Fall weitaus am häufigsten auftritt, die Kombinationsvorschrift aber nur in den Fällen zwei bis vier zum Einsatz kommt. Es wird sich aber zeigen, dass bereits wenige Punkte für eine starke Beeinflussung des Simulationsergebnisses ausreichen.

Schließlich bleibt noch anzumerken, dass die Vorschrift ZERO der klassischen Randbehandlung entspricht. Das ist aber nicht zu verwechseln mit dem Nullsetzen aller Randknoten, da nur diejenigen Randknoten auf Null gesetzt werden, für die mehrere Extrapolationswerte existieren.

4.5.2 Extrapolation

Wie bereits im letzten Abschnitt erwähnt wurde, wird bei der Extrapolation die Geschwindigkeit in einem Randknoten mit Hilfe des Randverlaufs und eines benachbarten Fluidknotens bestimmt. Anhand von Abbildung 4.6 wird das Problem nun genauer erörtert. Gegeben sind die Geschwindigkeit $\vec{u}_F$ im Fluidknoten F durch das Geschwindigkeitsfeld, die Geschwindigkeit $\vec{u}_S = 0$ am Rand durch die Haftbedingung und das Schnittverhältnis $w = \frac{\overline{SR}}{\overline{FR}} \in [0, 1]$ durch den Randverlauf. Gesucht wird ein Wert für die Geschwindigkeit $\vec{u}_R$ im Randknoten R. Dieser Wert kann mit Hilfe einer der beiden folgenden Extrapolationsverfahren bestimmt werden.

Lineare Extrapolation

Bei der Linearen Extrapolation wird entsprechend Abbildung 4.6 zur Bestimmung von $\vec{u}_R$ vorgegangen. Dabei ist die Geschwindigkeit in y-Richtung aufgetragen. Die Zellkante entlang der extrapoliert wird, verläuft in x-Richtung. Es wird nun der Gradient mit Hilfe der Geschwindigkeiten in den Punkten F und S bestimmt. Wird der Gradient auch über S hinaus bis zum Knoten R als konstant angenommen, so kann mit Hilfe des Strahlensatzes die Geschwindigkeit in R bestimmt werden und es gilt:

$$\frac{\vec{u}_F}{1-w} = \frac{\vec{u}_R}{w} \Rightarrow \vec{u}_R = -\frac{\vec{u}_F w}{1-w}$$

Diese Art der Geschwindigkeitsbestimmung für den Randknoten R ist nicht ganz unproblematisch, da die extrapolierten Werte sehr groß werden können, wenn der Rand nahe am Knoten F verläuft.

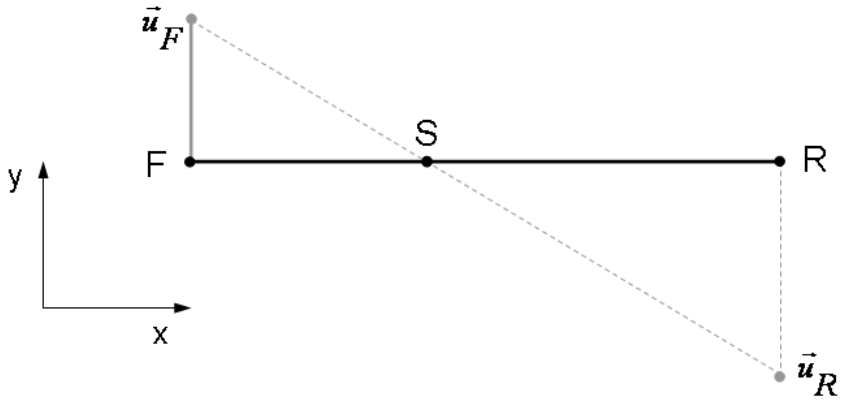

Abbildung 4.6: Lineare Extrapolation

Gradient-Extrapolation

Bei der Gradient-Extrapolation tritt die Problematik der Linearen Extrapolation nicht auf. Ihre Funktionsweise ist in Abbildung 4.7 zu sehen. Die Achsen entsprechen denen der Linearen

Extrapolation. Hier ist die Grundidee, dass der Geschwindigkeitsgradient nicht durch F und S sondern durch F und R geht. Dabei wird zunächst angenommen, dass die Geschwindigkeit im Knoten R verschwindet. Die Geschwindigkeit im Knoten F wird nun so angepasst, dass die Geschwindigkeit im Punkt S unter Verwendung dieses Gradienten Null wird. Auch hier kann jetzt u_R nach dem Strahlensatz bestimmt werden und es gilt:

$$\frac{\vec{u}_F}{1} = -\frac{\vec{u}_R}{w} \Rightarrow \vec{u}_R = -w\vec{u}_F$$

Wie die Ergebnisse in Kapitel 7 zeigen, macht es dabei keinen Sinn, den veränderten Wert im Fluidknoten auch in das Geschwindigkeitsfeld zu übernehmen.

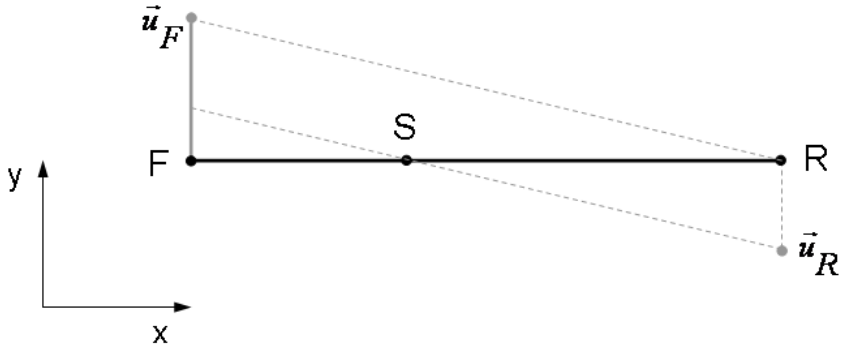

Abbildung 4.7: Gradient-Extrapolation

4.5.3 Randkorrektur

Eingangs wurde bereits erwähnt, dass die Randkorrektur dazu dient, die Divergenzfreiheit am Rand wiederherzustellen, die durch die vorausgegangene Randbehandlung zerstört wurde. Die Änderung in den Randknoten, die dafür erforderlich ist, sollte sich jedoch auf ein Mindestmaß beschränken.

Für eine mathematische Formulierung dieser Anforderung stellt man sich zunächst einmal alle n Randknoten durchnummeriert vor. Die Geschwindigkeitskomponenten an diesen Knoten werden dann anhand der Knotennummer im Vektor $\vec{r}$ zusammengefasst, so dass sich

$$\vec{r}^T = (u_1, u_2, \ldots, u_n, v_1, v_2, \ldots, v_n)^T$$

ergibt. Entsprechend werden die Ausgangswerte, die sich durch die Randbehandlung ergeben haben, in den Vektor $\overrightarrow{r0}$ eingereiht. Damit die Lösbarkeit der Poisson-Gleichung gegeben ist, müssen zunächst die bekannten Gleichungen

$$\begin{aligned} f_1(\vec{r}) &= \sum_{i=1}^{b_n} bdiv_i = \sum_{i=1}^{2n} wb_i \cdot r_i = 0 \qquad \text{und} \\ f_2(\vec{r}) &= \sum_{i=1}^{w_n} wdiv_i = \sum_{i=1}^{2n} ww_i \cdot r_i = 0 \end{aligned}$$

erfüllt werden. Dabei enthalten $\overrightarrow{wb}$ und $\overrightarrow{ww}$ die Gewichte der Komponenten entsprechend Abschnitt 4.2.2. Diesmal wird aber zusätzlich noch gefordert, dass

$$h(\vec{r}) = \sum_{i=1}^{2n} (r_i - r0_i)^2$$

minimal wird, um die Änderung in den Randknoten auf ein Minimum zu beschränken.

Zusammenfassend lässt sich also sagen, dass das Minimum von $h(\vec{r})$ unter den Nebenbedingungen $f_1(\vec{r}) = 0$ und $f_2(\vec{r}) = 0$ gesucht ist. Es handelt sich hierbei um ein mathematisches Standardproblem, das im Folgenden mit dem Verfahren nach Lagrange gelöst werden soll.

Minimierung unter Nebenbedingungen nach Lagrange

Für die Minimierung unter Nebenbedingungen nach Lagrange benötigt man zunächst den folgenden Satz. Dieser sollte eigentlich in jedem Analysis-Buch zu finden sein und wurde hier aus [11] übernommen.

SATZ: Seien $h: D \to \mathbf{R}$ und $\vec{f}: D \to \mathbf{R}^p$ stetig differenzierbare Abbildungen auf einer offenen Menge $D \subset \mathbf{R}^n$, $n > p$, wobei $\vec{f}'(\vec{r})$ für jedes $\vec{r} \in D$ den Rang p hat, dann folgt: Ist $\vec{r}_0 \in D$ eine Extremalstelle von h unter der Nebenbedingung $\vec{f}(\vec{r}) = 0$, so existieren dazu die Lagrangeschen Multiplikatoren $\lambda_1, \lambda_2, \ldots, \lambda_p$ so dass gilt:

$$\nabla h(\vec{r}_0) = \sum_{i=1}^{n} \lambda_i \nabla f_i(\vec{r}_0)$$

Damit liegen $n+p$ Gleichungen für $n+p$ Unbekannte $r_1, r_2, \ldots, r_n, \lambda_1, \lambda_2, \ldots, \lambda_p$ vor, die durch Lösen des Gleichungssystems bestimmt werden können. Der Vektor $\vec{r} = (r_1, r_2, \ldots, r_n)^T$ ist dann die gesuchte Extremalstelle unter der Nebenbedingung $h(\vec{r}) = 0$.

Lösung

Überträgt man das eben vorgestellte Verfahren auf das vorliegende Problem, so erhält man ein Lineares Gleichungssystem mit $2n + 2$ Unbekannten:

$$\begin{aligned}
2(r_1 - r0_1) &= \lambda 1 \cdot wb_1 + \lambda 2 \cdot ww_1 \\
2(r_2 - r0_2) &= \lambda 1 \cdot wb_2 + \lambda 2 \cdot ww_2 \\
&\vdots \\
2(r_{2n} - r0_{2n}) &= \lambda 1 \cdot wb_{2n} + \lambda 2 \cdot ww_{2n} \\
wb_1 \cdot r_1 + wb_2 \cdot r_2 + \ldots + wb_{2n} \cdot r_{2n} &= 0 \\
ww_1 \cdot r_1 + ww_2 \cdot r_2 + \ldots + ww_{2n} \cdot r_{2n} &= 0
\end{aligned}$$

In Matrixschreibweise ergibt sich dann dafür

$$\begin{pmatrix} 2 & 0 & 0 & \dots & 0 & wb_1 & ww_1 \\ 0 & 2 & 0 & \dots & 0 & wb_2 & ww_2 \\ 0 & 0 & 2 & \dots & 0 & wb_3 & ww_3 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & 2 & wb_{2n} & ww_{2n} \\ wb_1 & wb_2 & wb_3 & \dots & wb_{2n} & 0 & 0 \\ ww_1 & ww_2 & ww_3 & \dots & ww_{2n} & 0 & 0 \end{pmatrix} \cdot \vec{r} = \begin{pmatrix} -2r0_1 \\ -2r0_2 \\ -2r0_3 \\ \vdots \\ -2r0_{2n} \\ 0 \\ 0) \end{pmatrix}.$$

Durch mehrere Äquivalenzumformungen, kann dieses Gleichungssytem nun auf die obere Dreiecksform gebracht werden. Da aber die Terme in den unteren beiden Zeilen dann keine schöne Darstellung mehr erlauben, wird hier der letzte Umformungsschritt weggelassen und das Gleichungssystem ergibt sich zu:

$$\begin{pmatrix} 2 & 0 & 0 & \dots & 0 & wb_1 & ww_1 \\ 0 & 2 & 0 & \dots & 0 & wb_2 & ww_2 \\ 0 & 0 & 2 & \dots & 0 & wb_3 & ww_3 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & 2 & wb_{2n} & ww_{2n} \\ 0 & 0 & 0 & \dots & 0 & \sum wb_i^2 & \sum wb_i \cdot ww_i \\ 0 & 0 & 0 & \dots & 0 & \sum wb_i \cdot ww_i & \sum ww_i^2 \end{pmatrix} \cdot \vec{r} = \begin{pmatrix} -2r_1 \\ -2r_2 \\ -2r_3 \\ \vdots \\ -2r_{2n} \\ -2\sum r0_i \cdot wb_i \\ -2\sum r0_i \cdot ww_i) \end{pmatrix}.$$

Damit können dann die korrigierten Geschwindigkeiten an den Randknoten direkt berechnet werden und es erfolgt kein Umweg über das Lösen eins Linearen Gleichungssystems.

Kapitel 5

Implementierungsaspekte

5.1 Objektstruktur

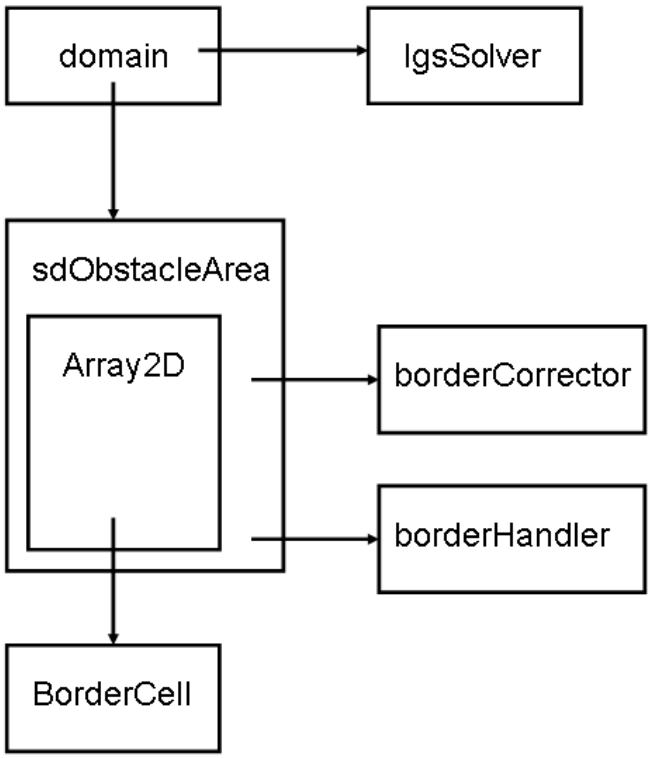

Abbildung 5.1: Strukturierung der wichtigen Klassen für die Randbehandlung

Die grobe Objektstruktur mit den wichtigsten Klassen für die Randbehandlung ist in Abbildung 5.1 dargestellt. Den Ausgangspunkt bildet wie schon in der Studienarbeit von Felix Müller[1] die Klasse domain. Dort sind die Druckwerte und die Geschwindigkeiten des recht-

eckigen Simulationsgebietes in 2-dimensionalen Feldern gespeichert und dort wird auch der Zeitschritt durchgeführt. Früher enthielt die domain-Klasse auch noch den Code zur Lösung der Poisson-Gleichung, dieser wurde aber in eine eigene Klasse ausgelagert, um das Verfahren zum Lösen des Linearen Gleichungssystems ohne größere Probleme austauschen zu können. Die Randbehandlung selbst wird in den nachfolgend beschriebenen Klassen durchgeführt.

5.1.1 Klasse sdObstacleArea

Die Klasse sdObstacleArea stellt den zentralen Punkt der Randbehandlung dar. Ihre wichtigsten Daten sind im Folgenden kurz aufgeführt, dabei wurde die Notation an die Syntax von C++ angelehnt.

```
enum nodeType {BORDER, FLUID, OBSTACLE}; // Definition des Knotentyps
enum cellType {BORDER, FLUID, OBSTACLE}; // Definition des Zelltyps

                                         // Eintrag der Hindernisliste,
struct obstacleEntry {                   // bestehend aus
  obstacle *o;                           // eigentlichem Hindernis
  borderHandler *bHandler;               // Objekt zur Randbehandlung und
  borderCorrector *bCorrector;           // Objekt zur Randkorrektur
};
                                         // Arrays zum Speichern von
nodeType *nodeData;                      // Knotentypen und
cellType *cellData;                      // Zelltypen

std::list<_obstacleEntry> obstacles;     // Liste mit den Hindernissen
array2D<borderCell> *borderCells;        // Wrapper-Objekt zur Speicherung
                                         // der Randzellen
```

Wie man sieht ist unter anderem eine Liste der Hindernisse enthalten. Ein Listeneintrag besteht dabei aus dem Hindernisobjekt selbst, einem borderHandler-Objekt, das für das eigentliche Setzen der Geschwindigkeit am Rand verantwortlich ist und einem borderCorrector-Objekt, das die Korrektur der Randwerte durchführt. Die Randbehandlung kann also für jedes Hindernis unterschiedlich sein. Weiterhin enthält diese Klasse ein 2-dimensionales Feld, das zu jedem Knoten den entsprechenden Typ enthält. Es wird hier unterschieden zwischen Fluid-, Rand- und Hindernisknoten. Ein ähnliches Feld existiert auch für die Zellen, dort findet analog eine Unterteilung in Fluid-, Rand- und Hinderniszellen statt. Schließlich wird für jede Randzelle noch ein borderCell-Objekt gespeichert, das unter anderem den Schnittverlauf in der Randzelle enthält. Da die Anzahl der Randzellen im Vergleich zur Gesamtzahl der Zellen eher gering ausfällt, wird dazu kein Array verwendet. Die Randzellen werden stattdessen in einer Wrapper-Klasse, die intern eine Liste zur Speicherung verwendet, nach aussen aber den Zugriff auf die Zellen über die Zellposition gestattet, gespeichert. Bei einem Zugriff muss dann zwar die gesamte Liste durchsucht werden, der Aufwand hierfür wird sich aber in Grenzen halten, da wie schon erwähnt üblicherweise nur wenig Randzellen existieren.

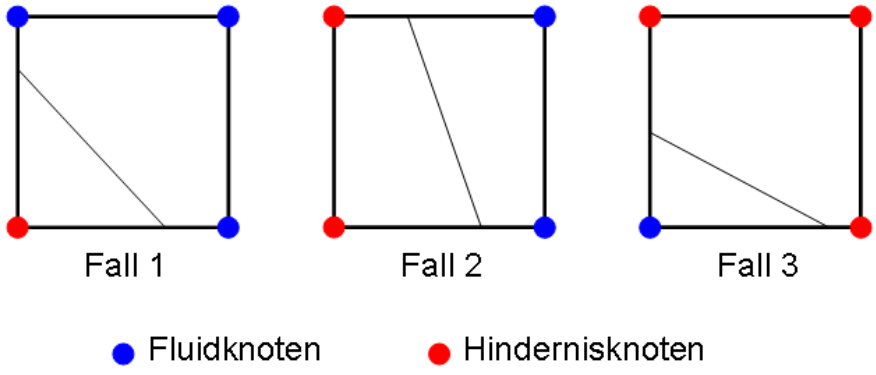

Abbildung 5.2: Basisfälle für den Schnittverlauf

Um nun ein Hindernis in das Simulationsgebiet zu setzten, wird das entsprechende Hindernisobjekt an die Klasse sdObstacleArea übergeben. Dort werden zunächst die Felder für die Zell- und Knotentypen angepasst und dann entsprechend Randzellenobjekte angelegt und gespeichert. Schließlich werden noch neue Objekte für Randbehandlung und Randkorrektur erstellt und zusammen mit dem Hindernisobjekt in der Hindernisliste abgelegt. Die Klassifikation der Zellen und Knoten, sowie das Erstellen der Randzellen leistet dabei das Hindernisobjekt.

Die Durchführung der Randbehandlung gestaltet sich deutlich einfacher. Für die Randbehandlung wird nämlich einfach über die Liste mit den Hindernissen iteriert und dabei Randbehandlung und Randkorrektur an die entsprechenden Objekte, die zusammen mit dem Hindernis gespeichert sind, weiterdelegiert.

Schließlich bleibt noch zu erwähnen, dass auf die Felder für die Knoten- und Zellentypisierung ein lesender Zugriff von aussen möglich ist. Diese Felder werden nänlich unter anderem im domain-Objekt zum Aufbau der Poisson-Matrix, sowie zur Bestimmung der Knoten, die im Zeitschritt aktualisiert werden müssen, benötigt.

5.1.2 Klasse borderCell

Die Klasse borderCell dient der Speicherung des Randverlaufs. Dazu enthält sie Objekte zunächst einmal für jede Zellkante den Schnittpunkt mit dem Rand, normiert auf das Intervall $(0, 1]$. Das Intervall wurde dabei linksseitig offen gewählt, um bei einem exakten Aufeinandertreffen von Rand und Eckpunkt Konflikte zu vermeiden. Existiert für die entsprechende Kante kein Schnittpunkt wird der Wert auf -1 gesetzt. Weiterhin wird zu jedem Eckpunkt der korrespondiere Knotentyp gespeichert. Schließlich enthält die Klasse dann noch den qualitativen Schnittverlauf in Form eines Basisfalls mit dazugehöriger Drehung entsprechend Abbildung 5.2. Der qualitative Schnittverlauf ließe sich zwar auch aus den Schnittpunkten mit den Zellkanten und den Knotentypen der Eckpunkte bestimmen, da sich diese Bestimmung aber recht aufwändig gestaltet, wird sie nur einmal bei der Erstellung des Objekts durchgeführt und das

Ergebnis dann in Form von Basisfall und Drehung gespeichert. Es muss beachtet werden, das es sich bei den drei Basisfällen aus Abbildung 5.2 nicht um alle theoretisch möglichen Fälle handelt, da eine Zelle eigentlich auch mehr als einmal geschnitten werden kann. In Bereichen, in denen sich Hindernisse derart nahe kommen, muss die Auflösung aber sowieso erhöht werden, um sinnvolle Ergebnisse zu erhalten, was zur Folge hat, dass der eben beschriebene Fall nicht mehr existiert. Es kann daher davon ausgegangen werden, dass die drei verwendeten Basisfälle in der praktischen Anwendung zur Beschreibung der Hindernisse ausreichen.

5.1.3 Klasse borderHandler

In der Klasse borderHandler wird die eigentliche Randbehandlung, wie sie in Kapitel 4 beschrieben ist, durchgeführt. Um dabei nicht bei jedem Aufruf über alle Knoten iterieren zu müssen, werden die relevanten Knoten beim Erstellen des Objekts in einer Liste gespeichert. Dazu wird für jeden Knoten innerhalb des Hindernisses überprüft, ob sich mindestens einer seiner vier direkten Nachbarknoten im Fluid befindet. Ist das der Fall, wird der Knoten inklusive der Schnittverhältnisse an den vier angrenzenden Kanten zur Liste hinzugefügt. Die Schnittverhältnisse werden dabei einfach aus den borderCell-Objekten der umliegenden Randzellen entnommen. Zum Durchführen der Randbehandlung wird dann einfach über die Liste iteriert und der Wert am aktuellen Knoten entsprechend den Vorschriften aus Abschnitt 4.5 berechnet.

5.1.4 Klasse borderCorrector

Die borderCorrector-Klasse ist verantwortlich für die Korrektur der Geschwindigkeitswerte am inneren Rand des Hindernisses. Ähnlich wie bei der borderHandler-Klasse, werden auch hier sämtliche Zellkanten des inneren Randes nur einmal im Konstruktor bestimmt und dann für die weitere Verwendung in einer Liste gespeichert. Eine Zellkante ist dabei charakterisiert durch den Anfangs- und Endpunkt, sowie die Farbe und Seite der Zelle auf der sie liegt. Um die Gewichtung eines Knotens zur Bestimmung der Divergenzsummen entsprechend Abschnitt 4.2.2 zu bestimmen, reicht es aus, die beiden angrenzenden Zellkanten des inneren Randes zu betrachten.

Dabei wird die Kantenliste aufgebaut, indem man ausgehend von einer ersten Randzelle, sukzessive die benachbarten Randzellen bestimmt und dabei die Zellkanten, die Teil des inneren Randes sind, zur Kantenliste hinzufügt. Ist man wieder bei der Startzelle angelangt, wurde das Hindernis einmal umlaufen und der innere Rand befindet sich in der Kantenliste. Da Randzellen auch über Eck verbunden sein können, wird die Bestimmung der Nachbarzellen wie folgt vorgenommen. Zunächst wird die aktuelle Randzelle markiert. Dann werden die direkten Nachbarzellen, zu denen eine Verbindung über eine Zellkante besteht, betrachtet. Befindet sich unter ihnen eine nichtmarkierte Randzelle, wird das Verfahren mit dieser fortgesetzt. Ist das nicht der Fall, muss eine Verbindung über Eck bestehen und eine nichtmarkierte Randzelle befindet sich unter den diagonalen Nachbarn.

Um die korrigierten Werte für die inneren Knoten zu bestimmen, wird jetzt einfach die Lösung für das Gleichungssystem aus Abschnitt 4.5.3 direkt übernommen, da durch Kenntnis

des inneren Randes alle benötigten Größen ermittelt werden können.

5.2 Parallelisierung

Im Verlauf der Simulation muss in jedem Zeitschritt die Poisson-Gleichung gelöst werden. Dieses Gleichungssystem kann wie in Abschnitt 3.5.2 beschrieben in zwei unabhängige Systeme aufgeteilt werden. Da zur Durchführung der Simulation auch Rechner mit zwei Prozessoren zur Verfügung standen, findet die Lösung jedes Gleichungssystems in einem getrennten Thread statt.

Diese Aufteilung bewirkte aber zunächst einen Performanceeinbruch und nicht die erhoffte Geschwindigkeitssteigerung. Es stellte sich schließlich heraus, dass dieses Problem durch die Art und Weise der Matrix-Speicherung verursacht wurde. Da die Poisson-Matrix hier nur aus einer Hauptdiagonalen und jeweils zwei Nebendiagonalen besteht, erfolgte die Speicherung ursprünglich in fünf Feldern. Da die Matrixeinträge für schwarze und weiße Zellen in den Diagonalen miteinander verschachtelt sind, wurde von den beiden Threads beim Lösen der Poisson-Gleichung parallel auf den gleichen Speicherbereich zugegriffen. Die Synchronisierung der parallelen Zugriffe, die dabei nötig war, hat dann offensichtlich den Performancevorteil, der durch die gleichzeitige Berechnung entstanden ist, zunichte gemacht.

Dieses Problem wurde nun umgangen, indem jeweils für die weißen und die schwarzen Zellen eine eigene Matrix verwendet wird. Durch die Aufteilung geht aber leider die oben beschriebene Diagonalgestalt verloren. Allerdings enthalten die beiden Matrizen pro Zeile wie die Ausgangsmatrix auch nur maximal fünf Einträge, die von Null verschieden sind. Es wurde dann auch hier wieder Platz gespart, da pro Zeile nur die fünf Einträge zusammen mit ihrer Position innherhalb der Zeile gespeichert werden.

Kapitel 6

Testszenarien

Um beurteilen zu können, wie gut der hier vorgestellte Algorithmus arbeitet, wird die Durchströmung von zwei verschiedenen Gebieten simuliert. Aus den Simulationsergebnissen werden dann die charakteristischen Größen des jeweiligen Gebiets bestimmt, die zusammen mit den entsprechenden Referenzgrößen Rückschlüsse auf die Genauigkeit der Simulation zulassen. Dazu werden in diesem Kapitel ersteinmal Topologie und charakteristische Größen, sowie deren Berechnung für beide Gebiete vorgestellt.

6.1 DFG Benchmark

Das erste Testszenario beschreibt die Umströmung eines Zylinders in einem Kanal und entspricht dem stationären Testfall für 2-dimensionale laminare Strömungen aus dem sogenannten DFG Benchmark [13]. Dabei beträgt entsprechend Abbildung 6.1 die Länge des Kanals $L = 2.2\text{m}$ und die Höhe $H = 0.41\text{m}$. Der Zylinder mit einem Durchmesser von $D = 0.1\text{m}$ wird so im Kanal platziert, dass er zum unteren und zum linken Rand des Kanals jeweils einen Abstand von $A = 0.15\text{m}$ hat. Das Fluid strömt über den linken Rand des Kanals ein und über den rechten Rand wieder heraus. Die Einströmgeschwindigkeit am linken Rand $\vec{U}(0,y)^T = (U(0,y), V(0,y))$ ist dabei gegeben durch

$$U(0,y) = \frac{4U_m(H-y)}{H^2} \qquad \text{und} \qquad V(0,y) = 0,$$

mit der Maximalgeschwindigkeit $U_m = 0.3\frac{\text{m}}{\text{s}}$. Die Ausströmgeschwindigkeit beeinflusst die charakteristischen Größen nicht und kann daher beliebig gewählt werden, natürlich unter Einhaltung der entsprechenden Bedingungen aus Abschnitt 4.3. Am oberen und unteren Rand der Kanals gilt die Haftbedingung und daher ist dort $U = V = 0$. Für die Durchströmung wird ein inkompressibles Fluid mit Dichte $\rho = 1.0\frac{\text{kg}}{\text{m}^2}$ und dynamischer Viskosität $\mu = 10^{-3}\frac{\text{m}^2}{\text{s}}$ verwendet. Mit der mittleren Geschwindigkeit

$$\bar{U} = \frac{1}{H}\int_0^H U(0,y)dy = \frac{2}{3}U_m$$

ergibt sich schließlich eine Reynoldszahl von

$$Re = \frac{\bar{U}D}{\mu} = 20.$$

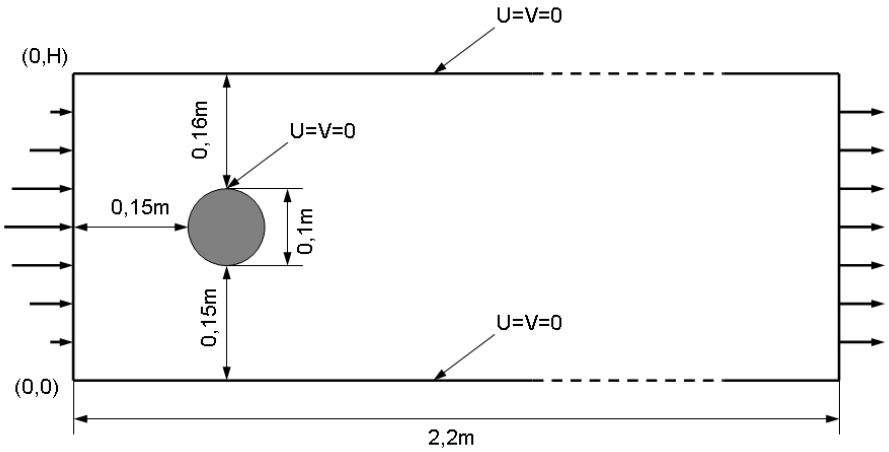

Abbildung 6.1: Anordnung von Kanal und Zylinder beim DFG Benchmark

Mit den obigen Informationen kann nun die Simulation durchgeführt werden. Die Beurteilung der Ergebnisse wird dann beim DFG Benchmark anhand von vier, in unserem Fall aber nur anhand von drei charakteristischen Größen vorgenommen. Es handelt sich dabei um den Drag-Koeffizient c_D den Lift-Koeffizient c_L sowie den Druckunterschied ΔP.

Für die Berechnung der beiden Koeffizienten benötigt man zunächst die Kräfte, die auf den Zylinder wirken. In x-Richtung ist das die Kraft F_D und in negativer y-Richtung die Kraft F_L. Sie ergeben sich zu

$$\begin{aligned} F_D &= \int_S (\rho\mu \frac{\partial v_t}{\partial n} n_y - P n_x) dS \qquad \text{und} \\ F_L &= -\int_S (\rho\mu \frac{\partial v_t}{\partial n} n_x + P n_y) dS. \end{aligned}$$

Dabei bezeichnet S den Rand des Zylinders, $\vec{n}^T = (n_x, n_y)$ den Normalen-Vektor auf S und v_t die Tangentialgeschwindigkeit in S. Mit diesen beiden Kräften lassen sich dann die Drag- und Lift-Koeffizienten berechnen und es gilt:

$$\begin{aligned} c_D &= \frac{2F_D}{\rho \bar{U}^2 D} \qquad \text{und} \\ c_L &= \frac{2F_L}{\rho \bar{U}^2 D}. \end{aligned}$$

Die Bestimmung von ΔP dagegen gestaltet sich einfacher, da nur der Druckunterschied zwischen Vorder- und Rückseite des Zylinders bestimmt werden muss. Mit den Druckwerten P_f und P_b an den entsprechenden Punkten $(0.15, 0.2)^T$ und $(0.25, 0.2)^T$ folgt dann für den Druckunterschied $\Delta P = P_f - P_b$.

Um die Qualität der Simulation schließlich auch beurteilen zu können, sind Referenzwerte für die charakteristischen Größen ΔP, c_L und c_D in Tabelle 6.1 angegeben.

Tabelle 6.1: Referenzwerte der charakteristischen Größen beim DFG Benchmark

	c_D	c_L	ΔP
Minimum	5.5700	0.0104	0.1172
Maximum	5.5900	0.0110	0.1176

6.2 Kanal mit Verengung

Beim zweiten Testszenario handelt es sich um einen Kanal mit Verengung. Dieses Szenario wurde speziell zur Untersuchung der Randbehandlung erstellt und orientiert sich sehr stark am DFG Benchmark. Entsprechend werden sämtliche Randbedingungen inklusive der Ein- und Ausströmbedingungen von oben übernommen. Es wird ausserdem das gleiche Fluid verwendet. Im Gegensatz zum obigen Fall wird aber der Zylinder weggelassen, und stattdessen entsprechend Abbildung 6.2 ein Trapez auf die Unterseite des Kanals gesetzt. Die Länge der Rampen beträgt dabei $R = 0.2$m, die Kanalhöhe wie oben $H = 0.41$m und die Kanallänge $L = 3.0$m. Die Trapezhöhe T ist abhängig von der Diskretisierung des Gebietes und beträgt das zweieinhalbfache der Kantenlänge h. Dadurch wird erreicht, dass die untere Begrenzung des verjüngten Kanals die Zellen genau in der Mitte schneidet und die Randbehandlung auch tatsächlich zum Einsatz kommt. Für die folgenden Erläuterungen wird ausserdem die Höhe des verengten Kanals mit H_V bezeichnet und die Maximalgeschwindigkeit in horizontaler Richtung mit U_{mV}. Als charakteristische Größe wird nun die horizontale Geschwindigkeit im

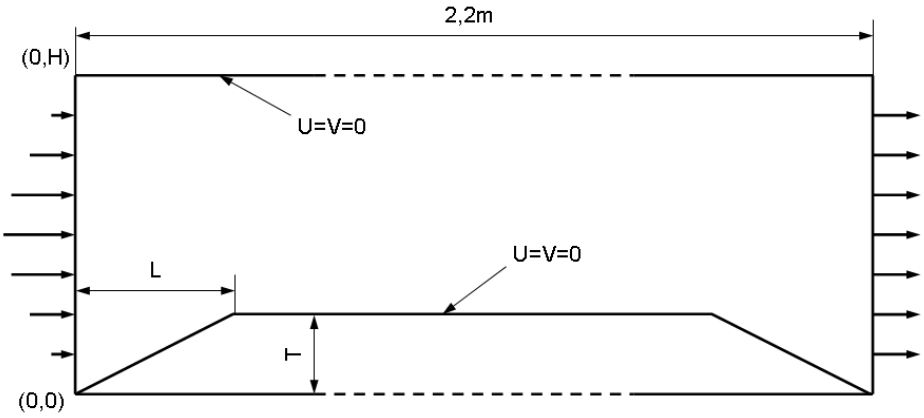

Abbildung 6.2: Kanal mit Verengung

Querschnitt des Kanals bestimmt. Der Abstand vom Kanalende beträgt dabei $Q = 0.3$m. Nach Gerthsen und Vogel [12] bildet sich dann in diesem Querschnitt ein parabolisches Profil aus, für das aufgrund der Haftbedingungen am Rand in einem beliebigen Punkt mit Abstand

y_V vom unteren Kanalrand gilt:

$$U(y_V) = \frac{4U_{Vm}y_V(H_V - y_V)}{H_V^2} \qquad \text{und} \qquad V(y_V) = 0.$$

Dort wird weiterhin erwähnt, dass bei einer Kanalverjüngung das Produkt aus Geschwindigkeit und Kanalquerschnitt konstant ist. Damit kann dann U_{Vm} in Abhängigkeit von U_m bestimmt werden da gilt:

$$U_m H = U_{Vm} H_V \Rightarrow U_{Vm} = U_M \frac{H}{H_V}.$$

Die Qualität der Simulation wird schließlich bestimmt, indem das ideale Geschwindigkeitsprofil mit dem Profil verglichen wird, das sich bei der Simulation ausgebildet hat. Da dieses Szenario nur dazu dient die generelle Funktionsweise der Randbehandlung zu überprüfen, ist die Bestimmung von quantitativen Größen überflüssig. Die Diskretisierung des Gebiets sollte allerdings nicht zu grob ausfallen, da dann wegen der Kopplung von Zellkantenlänge und Trapezhöhe eine sehr starke Verengung entsteht. Das hat zur Folge, dass an den Rampen starke Verwirbelungen entstehen, die verhindern, dass am Kanalende das parabolische Profil bereits wieder vollständig ausgebildet ist.

Kapitel 7

Ergebnisse

7.1 Integration der Randbehandlung in den Simulationsablauf

Für die Integration der Randbehandlung in den Simulationsablauf wurden hier zwei verschiedenen Fälle betrachtet. Die eigentliche Randbehandlung erfolgte dabei in jedem Fall immer vor der Aktualisierung der Geschwindigkeiten. Die nachfolgende Randkorrektur wurde im ersten Fall direkt nach der eigentlichen Randbehandlung, also auch noch vor der Geschwindigkeitsaktualisierung ausgeführt. Im zweiten Fall jedoch wurde sie hinausgezögert und zwischen die Aktualisierung der Geschwindigkeiten und die Durchführung der Chorin-Projektion gelegt. Eigentlich könnte man meinen, dass sich dadurch Vorteile gegenüber der ersten Variante ergeben, da die idealen Werte in den Randknoten für die Aktualisierung des Geschwindigkeitsfeldes herangezogen werden können. Interessanterweise war das aber nicht der Fall. Die Methode mit der verspäteten Randkorrektur lieferte nämlich schlechtere Ergebnisse. Bei allen durchgeführten Simulationen, wurde deshalb die Integration der Randbehandlung in den Algorithmus wie für den ersten Fall beschrieben durchgeführt.

7.2 Kanal mit Verengung

Um die grundlegende Funktion der Randbehandlung zu überprüfen, ist der Kanal mit Verengung, wie im letzten Kapitel definiert, simuliert worden. In Abbildung 7.1 sind dabei nur die Ergebnisse, die sich für die Gradient-Extrapolation und die klassische Methode ergeben haben, dargestellt, um das Schaubild nicht zu überladen. Die Qualität der beiden Randbehandlungsmethoden wird dann beurteilt, indem die Geschwindigkeiten, die sich für die jeweilige Methode an den Gitterknoten ergeben haben, mit dem idealen Geschwindigkeitsprofil verglichen werden. Dabei fällt auf, dass gerade im Randbereich, wo die Randbehandlung auch tatsächlich durchgefürt wird, die Kurve, die sich für die Gradient-Extrapolation ergibt, deutlich näher an der Referenzkurve liegt, als die der klassischen Methode.

Bei Verwendung der Linearen Extrapolation tritt allerdings genau der befürchtete Fall ein. Die Werte in Randnähe explodieren und es ergeben sich keine verwertbaren Ergebnisse. Es wurde daher überprüft, ob eine künstliche Einschränkung der extrapolierten Werte Abhilfe schafft. Dies hat dann zwar verhindert, dass sämtliche Geschwindigkeiten ins Unermessliche

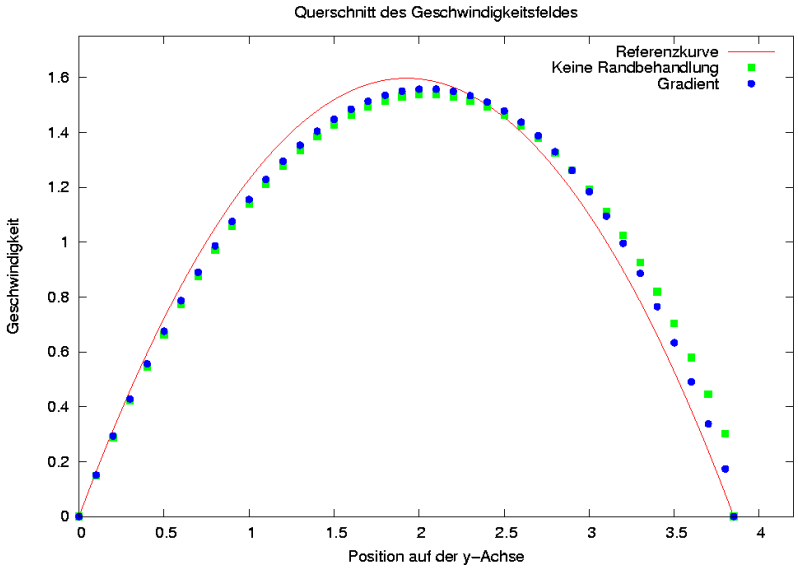

Abbildung 7.1: horizontale Geschwindigkeit im Kanalquerschnitt.

steigen, das Simulationsergebnis an sich blieb aber weiterhin unbrauchbar.

7.3 DFG-Benchmark

Wie schon beim Kanal mit Verengung lieferte auch beim DFG Benchmark die Randbehandlung mit Hilfe der Linearen Extrapolation keine brauchbaren Ergebnisse. Es wird daher in den folgenden Abschnitten nur die Randbehandlung durch die Gradient-Extrapolation genauer untersucht. Von besonderem Interesse ist dabei natürlich der Vergleich zur klassischen Randbehandlung, aber auch der Vergleich der verschiedenen Kombinationsmöglichkeiten aus Tabelle 4.5.1.

7.3.1 Werteverlauf in Abhängigkeit von der Zeit

Zur Betrachtung des Werteverlaufs wurde der DFG Benchmark viermal durchgefürt. Einmal ohne besondere Randbehandlung und dreimal mit der Gradient-Extrapolation. Für die Simulation wurde eine Auflösung des Simulationsgebiets von 880x164 Zellen verwendet, die Simulationsdauer betrug zehn Sekunden. Die Ergebnisse dieser Simulationen sind in den Abbildungen 7.2, 7.3 und 7.4 zu sehen.

Man kann problemlos erkennen, dass eine spezielle Randbehandlung durchaus Vorteile mit sich bringen kann. Dabei überzeugen vor allem die konservative Kombinationsvorschrift ZERO oder die naheliegende Vorschrift AVERAGE, wobei die Vorschrift AVERAGE im Ganzen ein wenig besser abschneidet.

Die Ergebnisse der verbliebenen Vorschrift SWITCHED_SEPARATE sind sehr durchwachsen. Zwar schneidet sie für den Drag-Koeffizienten recht gut ab, aber für Druckdifferenz

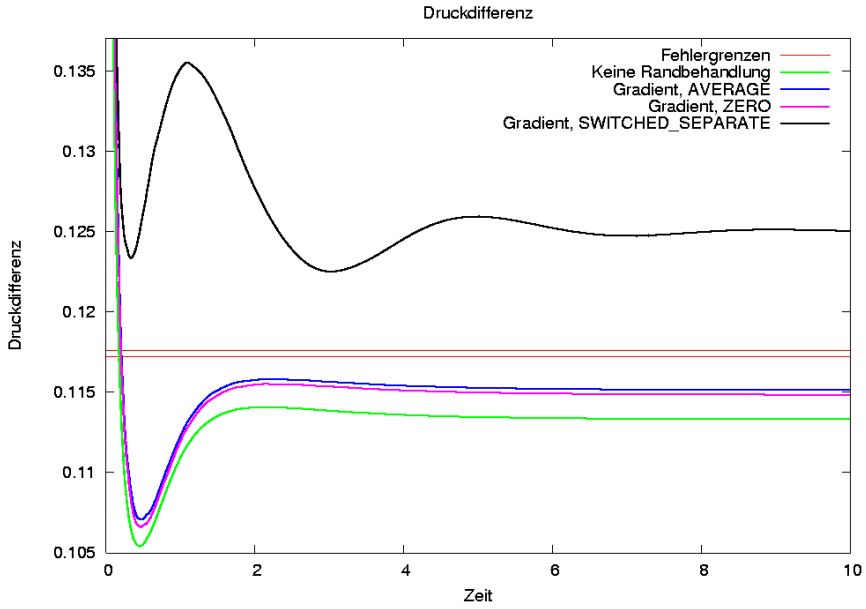

Abbildung 7.2: Verlauf des Druckunterschieds beim DFG Benchmark, bei einer Auflösung von 880x164 Zellen.

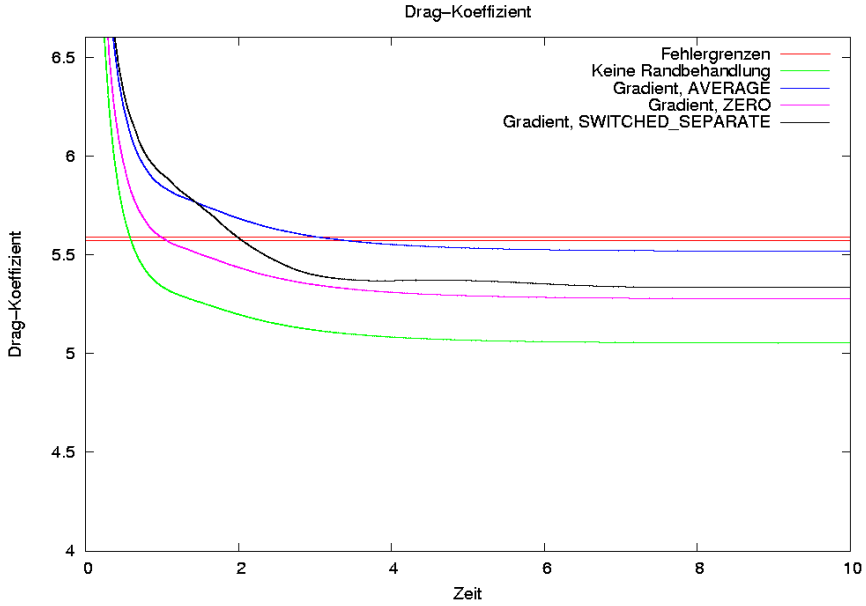

Abbildung 7.3: Verlauf des Drag-Koeffizienten beim DFG Benchmark, bei einer Auflösung von 880x164 Zellen.

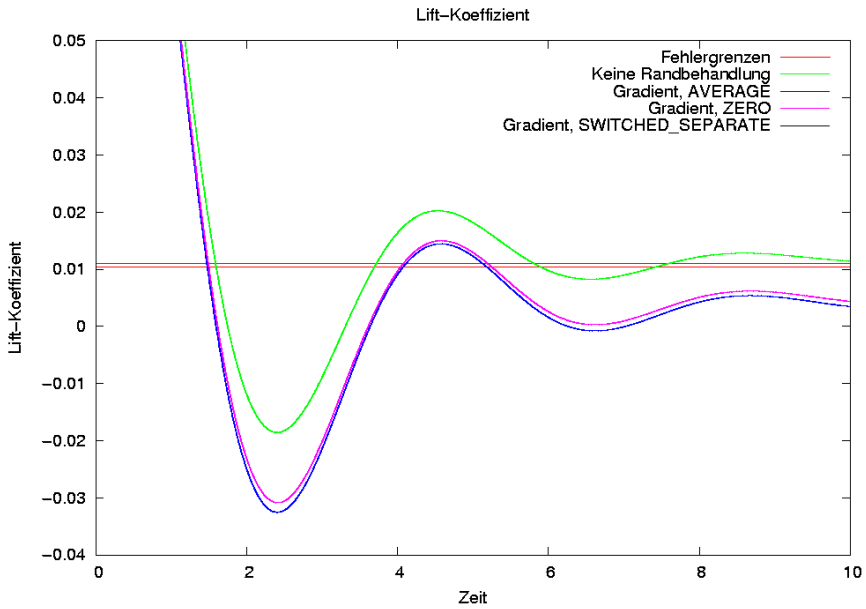

Abbildung 7.4: Verlauf des Lift-Koeffizienten beim DFG Benchmark, bei einer Auflösung von 880x164 Zellen.

und Lift-Koeffizient ist das Simulationsergebnis eher schlecht. Der Lift-Koeffizient liegt sogar derartig weit daneben, dass die Kurve gar nicht mehr im Schaubild zu sehen ist.

Der Lift-Koeffizient nimmt hier eine Sonderrolle ein. Für ihn ergeben sich die besten Werte, wenn auf eine Randbehandlung komplett vezichtet wird. Wie man aber im folgenden Abschnitt sehen wird, scheint das nur für diese eine Auflösung zuzutreffen und es handelt sich hier wohl um einen Glückstreffer.

7.3.2 Vergleich unterschiedlich feiner Diskretisierungen

Um nun auch erkennen zu können, wie sich die Methoden zur Randbehandlung in Abhängigkeit von der Diskretisierung des Gebiets verhalten, wurde der DFG Benchmark für die drei verschiedene Auflösungen 440x82, 880x164 und 1100x205 durchgeführt. Die Ergebnisse, die sich dabei für die drei charakteristischen Größen ergeben haben, sind in den folgenden drei Abbildungen 7.5, 7.6 und 7.7 dargestellt. Es wurde darauf geachtet, dass die Simulationen erst abgebrochen wurden, wenn keine Änderung der charakteristischen Größen mehr feststellbar war.

Zunächst fällt auf, dass sich für die klassische Randbehandlung eine Verschlechterung des Lift-Koeffizienten mit zunehmender Auflösung ergibt. In etwas abgeschwächter Form trifft das auch auf die anderen Methoden zu. Da leider nur drei verschiedene Aufösungen betrachtet wurden, können daher aus dem Schaubild für den Lift-Koeffizienten keine weitern Schlüsse gezogen werden. Hier würde sich eine Untersuchung des Lift-Koeffizienten für weitere

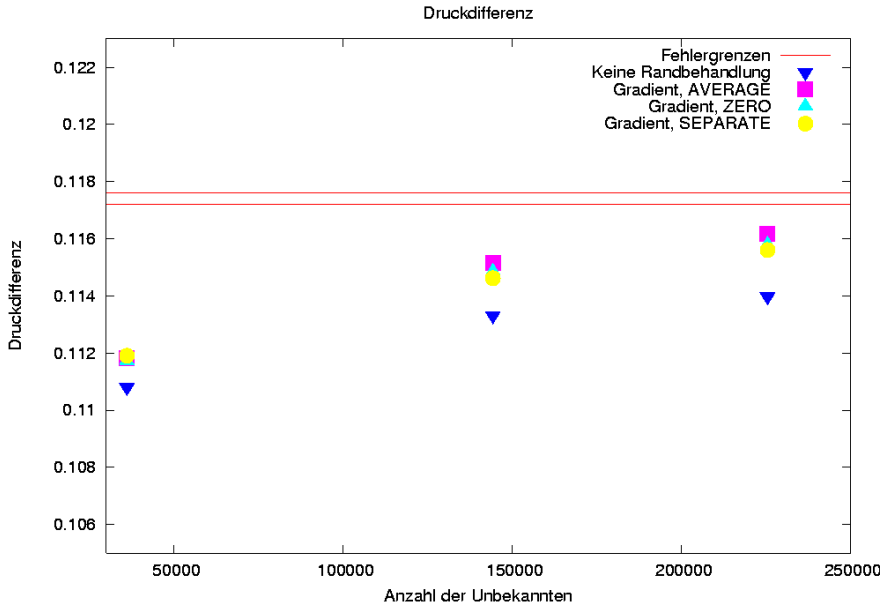

Abbildung 7.5: Druckunterschied in Abhängigkeit von der Anzahl der Unbekannten

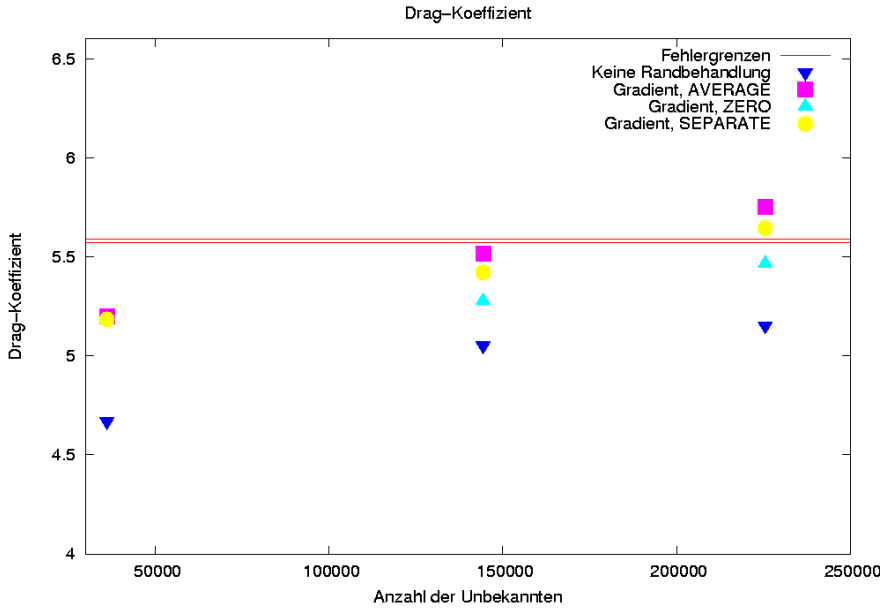

Abbildung 7.6: Drag-Koeffizient in Abhängigkeit von der Anzahl der Unbekannten

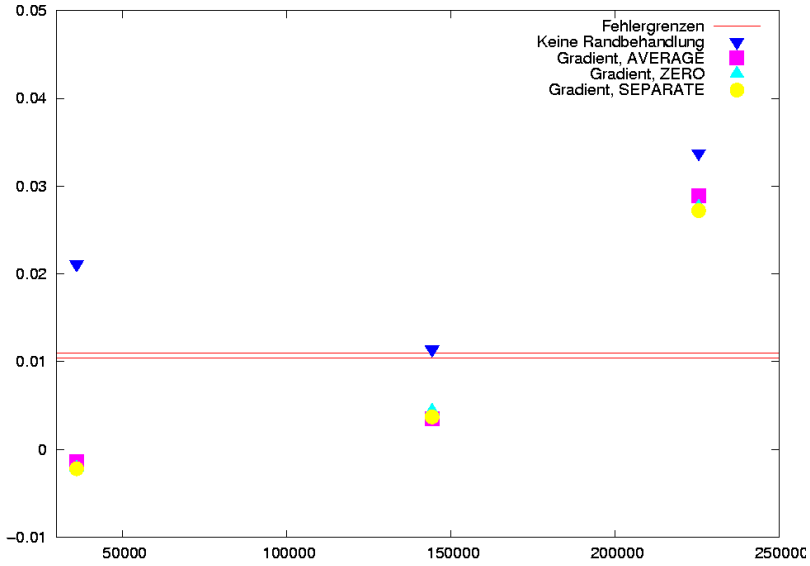

Abbildung 7.7: Lift-Koeffizient in Abhängigkeit von der Anzahl der Unbekannten

Auflösungen anbieten.

Die Werte für den Drag-Koeffizienten und die Druckdifferenz dagegen festigen den Eindruck aus der ersten Untersuchung. Hier ist ein deutlicher Vorsprung der Randbehandlungsmethoden sichtbar. Es stellt sich heraus, dass die Vorschrift AVERAGE offensichtlich unabhängig von der Auflösung die Besten Ergebnisse liefert.

7.4 Ergebnis

Abschließend lässt sich feststellen, dass durch die Randbehandlung tatsächlich eine Verbesserung der Simulationsergebnisse erreicht werden konnte. Allerdings müssen sowohl die Extrapolationsmethode als auch die Kombinationsmethode stimmen. Sonst können die Ergebnisse wie im Fall der Linearen Extrapolation schlichtweg unbrauchbar werden.

Interessant ist auch der Werteverlauf des Lift-Koeffizienten. Ob die Werte für die verschiedenen Auflösungen aber nun chaotisch um die Referenzwerte schwanken, oder ob eine Gesetzmäßigkeit dahinter steckt, können nur weitere Simulationsdurchläufe klären. Dagegen scheinen die Werte der anderen chrakteristischen Größen in Ordnung zu sein, obwohl auch hier interessant zu wissen wäre, ob der Drag-Koeffizient für die Randbehandlungsverfahren wie in Abbildung 7.6 schon angedeutet ist weiter zunimmt. Auch hier sind Spekulationen vergeblich und es können letztendlich nur weitere Simulationen Gewissheit schaffen.

Kapitel 8

Abschließende Bemerkungen

8.1 Zusammenfassung

In den letzten Kapiteln wurde ein Verfahren für eine verbesserte Randbehandlung vorgestellt und in einen Algorithmus zur Simulation laminarer Strömungen integriert. Der Algorithmus basiert dabei auf einer Finiten-Volumen-Methode kombiniert mit einem teilweise versetzten Gitter. Für die Behandlung des Randes wurde eine Vielzahl von Kombinationsmöglichkeiten vorgestellt. Wie sich herausgestellt hat, sind Verfahren, die auf einer Linearen Extrapolation beruhen so wie sie hier durchgeführt wurden nicht geeignet eine bessere Randapproximation zu erzielen. Eine bessere Randapproximation wurde nur mit den Verfahren erreicht, die auf Gradient-Extrapolation beruhen. Wie sich gezeigt hat, kann die Konfliktlösungsstrategie dabei das Ergebnis sehr stark beeinflussen in positiver wie in negativer Hinsicht. Abschließend bleibt festzuhalten, dass die Randbehandlung ein sehr komplexes Thema ist, bei dem die Ergebnisse, die durch Anwendung einer Randbehandlungsmethode erzielt werden, nicht unbedingt im Vorraus abzusehen sind. Da aber bei richtiger Anwendung ein positiver Effekt erzielt wird und der Aufwand hierfür vergleichsweise gering ist, stellt die Randbehandlung ein probates Mittel zur Verbesserung von Algorithmen zur Strömungssimulation dar.

8.2 Ausblick

In dieser Arbeit wurden nur zwei verschiedene Verfahren zur Randbehandlung vorgestellt, von denen sich eins als unbrauchbar erwiesen hat. Daher stellt sich die Frage, ob nicht noch andere Methoden gefunden werden können, die eine noch bessere Randapproximation ermöglichen. Dazu könnte die Randbehandlung dahingehend erweitert werden, dass die Geschwindigkeit am Rand nicht nur von einem einzigen Knoten und dem Schnittpunkte mit einer Zellkante abhängt, sondern auch die nähere Umgebung betrachtet wird. Ausserdem könnte auch die Randkorrektur verbessert werden, indem zum Beispiel eine andere Zielfunktion minimiert wird. Interessant ist natürlich auch eine Betrachtung der Randapproximation im Dreidimensionalen.

Es bleibt abzuwarten, wie sich das Thema Randbehandlung weiter entwickelt. Vielleicht gewinnt es zunehmend an Bedeutung, wenn sich die Rechenleistung eines Tages nicht mehr

so rasant ansteigt wie im Moment. Vielleicht wird eine spezielle Randbehandlung aber auch überflüssig, weil die Rechner so schnell geworden sind, dass die Gebiete so fein aufgelöst werden können, dass gesonderte Randbehandlung gar keine Vorteile mehr mit sich bringt. Da dieser Tag, falls er überhaupt existieren sollte, aber sicherlich noch in weiter Ferne liegt, wird die Randbehandlung beim Entwickeln von effizienten Algorithmen zur Strömungssimulation weiterhin eine bedeutende Rolle spielen.

Abbildungsverzeichnis

Tabellenverzeichnis

Literaturverzeichnis

[1] Müller, Felix: Simulation von Strömungen in unregelmäßig berandeten Gebieten mit symmetrieerhaltender Diskretisierung. Studienarbeit, Institut für Parallele und Verteilte Systeme Universität Stuttgart 2004

[2] Verstappen, R. W. C. P.; Veldman, A. E. P.: Symmetry-preserving discretization of turbulent flow, Journal of Computational Physics, 187:343-368 2002

[3] Emans, Maximilian: Numerische Simulation des unterkühlten Blasensiedens in turbulenter Strömung. Dissertation, Institut für Informatik TU München 2003

[4] Hackbusch, W.: Theorie und Numerik elliptischer Diffenrentialgleichungen. Stuttgart: Teubner 1996

[5] Zimmer, Stefan: Skript zur Vorlesung Numerische Simulation. Universität Stuttgart 2003

[6] Chorin, Alexandre J.: Numerical Solution of the Navier-Stokes Equations, Computational Mathematics, 22:745-762 1968

[7] Shortley, G. H.; Weller, R.: The Numerical Solution of Laplace's Equation, Journal of Applied Physics, 9: 1938

[8] Forrer H.: Second Order Accurate Boundary Treatment for Cartesian Grid Methods, Research Report No. 96-13, Eidgenössische Technische Hochschule Zürich 1996

[9] Tome, M. F.; McKee, S.: GENSMAC: A Computational Marker and Cell Method for Free Surface Flows in General Domains, Journal of Computational Physics, 110: 171-186 1994

[10] Griebel, M.; Dornseifer, T.; Neunhoeffer, T.: Numerische Simulation in der Strömungsmechanik. Braunschweig Wiesbaden: Vieweg 1995

[11] Burg, K.; Haf, H.; Wille, F.: Höhere Mathematik für Ingenieure. Stuttgart: Teubner 1985

[12] Gerthsen, Christian; Vogel, Helmut: Physik - Ein Lehrbuch zum Gebrauch neben Vorlesungen (17.Auflage). New York Berlin Heidelberg: Springer-Verlag 1993

[13] Schäfter, M.; Turek, S.: Benchmark Computations of Laminar Flow Around a Cylinder. Universität Erlangen-Nürnberg 1996

Erklärung

Hiermit versichere ich, diese Arbeit selbstständig verfasst und nur die angegebenen Quellen benutzt zu haben.

Unterschrift:

Stuttgart, den 11.1.2005