Optimierte Löserverfahren zur Druckberechnung in den zweiphasigen Navier-Stokes-Gleichungen

Thomas Sprügel

Geboren am 4. Mai 1987 in Köln

28. Dezember 2010

Bachelorarbeit Mathematik

Betreuer: Prof. Dr. Michael Griebel

Institut für Numerische Simulation

Mathematisch-Naturwissenschaftliche Fakultät der

Rheinischen Friedrich-Wilhelms-Universität Bonn

Inhaltsverzeichnis

Kapitel 1

Einleitung

Die numerische Strömungsmechanik ist in der heutigen Zeit von großem Interesse. Sie verbindet mathematische, physikalische, informatische und ingenieurswissenschaftliche Elemente und gehört zu dem modernen Gebiet des wissenschafltichen Rechnen. Das Ziel der Strömungsmechanik ist es Vorgänge aus der Natur mit einem möglichst genauen mathematischen und physikalischen Modell zu beschreiben und dieses dann am Computer zu simulieren, sodass möglichst realistische Annäherungen an die Naturvorgänge entstehen. Mithilfe von Strömungssimulationen können Experimente, die in der Realität nur mit großem Aufwand und mit hohen Kosten durchführbar sind, am Computer wesentlich günstiger simuliert werden. Jedoch sind Strömungssimulationen, selbst in der heutigen Zeit von modernen Hochleistungsrechnern, immer noch mit sehr langen Berechnungszeiten verbunden. Um komplexe Simulationsszenarien mit hohen Auflösungen nur wenige Sekunden zu simulieren, benötigt man mehrere Tage oder Wochen Rechenzeit. Allein durch die Weiterentwicklung von Hochleistungsrechnern können die Laufzeiten nicht angemessen reduziert werden. Daher müssen zusätzlich intelligente, problemangepasste Verfahren entwickelt werden. Die Optimierung von Lösungsverfahren im Bereich der Strömungsmechanik ist also von immenser Bedeutung. An diesem Punkt setzt die vorliegende Arbeit an.

Motivation

Im Mittelpunkt dieser Arbeit steht die Simulation von zweiphasigen Fluidströmungen. Ein *Fluid* bezeichnet dabei eine Flüssigkeit oder ein Gas. Zweiphasig bedeutet, dass zwei solche Fluide gleichzeitig simuliert werden können, wie zum Beispiel Wasser und Luft. Diese beiden Fluide sollen sich nicht mischen können und so existiert eine Grenzschicht zwischen ihnen, welche man auch mit *freier Oberfläche* oder *freiem Rand* bezeichnet. Im Fall von Wasser und Luft kann man sich die freie Oberfläche als Wasseroberfläche vorstellen.

Überall, wo die Bewegung und das Verhalten von freien Oberflächen eine wichtige Rolle spielen, kommen zweiphasige Strömungssimulationen zum Einsatz, wie beispielsweise im Bereich des Schiffbaus oder der Oberflächendynamik in Beschichtungsprozessen. Auch finden Simulationen von Tröpfchendynamik und Gasblasen breite Anwendung, beispielsweise im Bereich von Verbrennungsmotoren und Transportprozessen von Erdöl.

Abbildung 1.1: Wassersäule nach Aufprall eines Wassertropfen (Qelle [12])

Abbildung 1.2: Rutschender Tropfen auf einem Blatt (Qelle [13])

Als mathematisches Modell, welches zur Beschreibung dieser natürlichen Strömungsvorgänge dient, werden die Navier-Stokes-Gleichungen verwendet, welche ein System von partiellen Differentialgleichungen zweiter Ordnung bilden. Diese werden auf einem Gebiet Ω betrachtet. Durch Setzten von gewissen Anfangs- und Randbedingungen können verschiedenste Simulationsszenarien entstehen. Da die Navier-Stokes-Gleichungen ein kontinuierliches Modell bilden, müssen diese für eine Numerische Simulation zunächst diskretisiert werden. Dies kann zum Beispiel mit dem Ansatz der *finiten Differenzen* geschehen. Dabei werden die kontinuierlichen Differntialoperatoren der Navier-Stokes-Gleichungen durch Differenzenoperatoren ersetzt und es werden Approximationen in diskreten Punkten, welche auf einem Gitter liegen, berechnet. Eine Verbesserung der Approximation lässt sich damit durch eine Verfeinerung des entsprechenden Diskretisierungsgitters erreichen. Von besonderem Interesse sind daher Simulationen mit sehr feinen Diskretisierungsgittern und demnach hohen Auflösungen. Jedoch erhöht sich mit feinerer Auflösung der Rechenaufwand dramatisch. Um dies einzusehen, kann man ein Beispiel einer Wasserströmung in einem Kanal betrachten. Im Wasser steht ein Brückenpfeiler, um welchen das Wasser herumfließt (siehe Abbildung 1.3[1]).

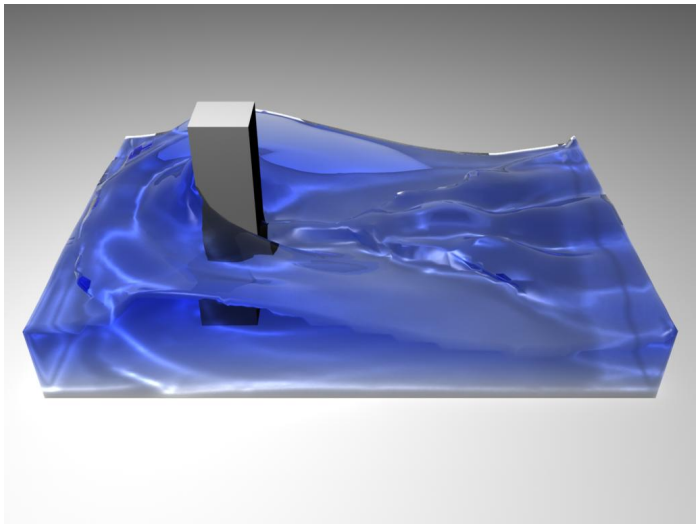

Abbildung 1.3: Strömung um ein Hindernis

[1] Quelle [16]

In einem quaderförmigen Gebiet berechnet man Approximationen in endlich vielen Punkten, welche auf einem Gitter liegen. Bei 60 Punkten in x-Richtung, 25 in y-Richtung und 40 in z-Richtung müssen also pro Schritt $60 \cdot 25 \cdot 40 = 60000$ Approximationen berechnet werden. Um eine Sekunde in dieser Auflösung zu simulieren, benötigt man eine Rechenzeit von etwa einer Minute. Vervierfacht man die Anzahl der Punkte in jeder Raumrichtung, so müssen 3840000 Approximationen pro Schritt berechnet werden. Eine Simulation von einer Sekunde derselben Aufgabe bei einer erhöhten Auflösung benötigt in etwa 12 Stunden Rechenzeit. Rechnet man die Zeiten hoch für eine Simulation von einer Minute, so würde man in der niedrigen Auflösung eine Laufzeit von etwa einer Stunde benötigen, in der hohen Auflösung eine Laufzeit von ungefähr einem Monat. Dabei wurden Testrechnungen parallel auf 32 Prozessoren mit jeweils 3.2 GHz durchgeführt. Ingesamt gilt, dass sich die Anzahl der zu berechnenden Unbekannten kubisch zum Verfeinern der Auflösung verhält. Verdoppelt man die Auflösung in jede Raumrichtung, so verachtfacht sich die Anzahl der Unbekannten. Hieran kann man erkennen, dass man beliebig viel Rechenleistung durch Verfeinern der Auflösung neutralisieren kann. Daher müssen zusätzlich die Verfahren zur Strömungssimulation optimiert werden.

In dieser Arbeit wird die Berechnung der Druckwerte optimert. Die Druckwerte gehören zu den gesuchten Größen in den Navier-Stokes-Gleichungen. Man erhält diese durch das Lösen der Druck-Poissongleichung, welche Im n-ten Zeitschritt von der Form

$$\nabla \cdot \left(\frac{1}{\rho(\phi^{n+1})} \nabla p^{n+1} \right) = rhs \tag{1.1}$$

ist, wobei p der zu bestimmende Druck und rhs eine vorher berechnete rechte Seite ist. Der Term $\rho(\phi)$ ist die Dichte des entsprechenden Fluids in Abhängigkeit der *Level-Set-Funktion* ϕ, welche den Abstand zur freien Oberfläche angibt. Da zweiphasige Strömungen betrachtet werden, können also auch zwei Fluide mit unterschiedlichen Dichten vorkommen. Simuliert man zwei Fluide mit einem großen Dichteunterschied, wie zum Beispiel Wasser und Luft (Dichteunterschied in etwa $1 : 1000$), so kommt es an der freien Oberfläche zu einem großen Dichtesprung. Wie man an der Form der Druck-Poissongleichung (1.1) sehr gut erkennen kann, geht dieser Dichtesprung unmittelbar in die Druckberechnung mit ein. Nach geeigneter Diskretisierung der Poissongleichung entsteht ein lineares Gleichungssystem der Form

$$Ax = b. \tag{1.2}$$

Jedoch bleibt der Term $\frac{1}{\rho(\phi)}$ auch in den Einträgen der Matrix A erhalten und so ist die Kondition der Matrix A abhängig von dem Dichtesprung. Je größer der Dichteunterschied der beiden Fluide ist, desto größer und damit schlechter wird die Kondition von A. Problematisch ist, dass die Konvergenzgeschwindigkeit von *Krylov-Unterraum-Verfahren*, welche in dieser Arbeit zum Lösen des Gleichungssystems (1.2) verwendet werden, entscheidend von der Kondition der Systemmatrix abhängen. Man kann also erkennen, dass das Simulieren von Zweiphasenströmungen eine zusätzliche Schwierigkeit mit sich bringt. Der Dichtesprung an der Phasengrenze wirkt sich direkt auf die Konvergenzgeschwindigkeit des Lösungsverfahren aus. Daher sind Zweiphasensysteme noch zeitaufwändiger zu berechnen als Einphasensysteme. Dies verstärkt die Notwendigkeit von optimierten Lösungsverfahren im Bereich von zweiphasigen Strömungssimulationen.

Wie allgemein bekannt, gehören Mehrgitterverfahren zu den effizientesten Verfahren zum Lösen von großen linearen Gleichungssystem. Sie sind in der Hinsicht optimal, als dass

die Konvergenzgeschwidigkeit nicht von der Maschenweite des Diskretisierungsgitters abhängt. Damit sind Mehrgitterverfahren auf lange Sicht die schnellsten und effizientesten bisher bekannten Lösungsverfahren für lineare Gleichungssysteme. Jedoch gilt dies nur auf die lange asymptotische Sicht. Da Mehrgitterverfahren zunächst aufwändige Gitterhierarchieen erstellen müssen, können andere Verfahren im präasymptotischen Sinne schneller und effizienter sein. Die Alternative zu Mehrgitterverfahren sind die vorkonditionierten Krylov-Unterraum-Verfahren. Diese haben zwar eine Konvergenzgeschwindigkeit, die von der Maschenweite abhängt, aber es müssen keine aufwändigen Gitterhierarchieen erstellt werden. Die Idee dieser Arbeit ist es, die besten Elemente aus der Klasse der Mehrgitterverfahren und der der Krylov-Unterraum-Verfahren zu vereinen. Dazu wird ein problemangepasster Vorkonditionierer entwickelt, welcher für Zweiphasensysteme besonders gut geeignet ist. Ein Vorkonditionierer hat die Aufgabe, die Konvergenzgeschwindigkeit eines Verfahrens zu erhöhen, indem die Kondition der Systemmatrix verringert wird. Dabei wird das äquivalente System

$$P^{-1}Ax = P^{-1}b \tag{1.3}$$

gelöst, wobei P der Vorkonditionierer ist. Die Kunst ist es einen Vorkonditionierer zu finden, welcher zum einen möglichst einfach zu invertieren ist und zum anderen die Kondition von $P^{-1}A$ möglichst klein werden lässt, zumindest aber wesentlich kleiner als die Kondition von A.

Der in dieser Arbeit speziell entwickelte Vorkonditionierer verwendet die Robustheit und Effizienz von Mehrgitterverfahren, sowie die schnelle Konvergenz von vorkonditionierten Krylov-Unterraum-Verfahren im präasymptotischen Sinne. Für spezielle zweiphasige Problemstellungen kann gezeigt werden, dass dieser Vorkonditionierer das Lösungsverfahren um mehr als eine Größenordnung beschleunigt und somit deutliche Geschwindigkeitsvorteile gegenüber den bisherigen Verfahren liefert. Damit ist er von erheblichem Vorteil für die zweiphasige Strömungsmechanik.

Zu dieser Arbeit

In dieser Arbeit geht es um eine Optimierung der Druckberechnung mit einer gebietsabhängigen Vorkonditionierung. Dazu wurde eine Vorkonditionierung mit der *additiven Schwarz Methode* verwendet, welche auf H. A. Schwarz [8] zurückgeht. Die oben beschriebene schlechte Kondition der Matrix A rührt von einem hohen Dichtesprung an der Phasengrenze her. Daher ist die Idee, das Gebiet Ω in einen schmalen Bereich um die freie Oberfläche Ω_f und in den übrigen Bereich Ω_1 aufzuteilen, sodass gilt

$$\Omega = \Omega_f \cup \Omega_1. \tag{1.4}$$

Die Druckberechnung muss dann abwechselnd auf beiden Gebieten durchgeführt werden. Es entstehen also zwei Unterprobleme, welche nach ihren individuellen Eigenschaften effizient gelöst werden sollen. So ist es möglich für beide Unterprobleme verschiedene Vorkonditionierer zu verwenden und so die lokalen Gebietseigenschaften auszunutzen.

Der Bereich Ω_f enthält die freie Oberfläche und somit ist das Unterproblem auf Ω_f schlecht konditioniert. Daher sollte auf diesem Gebiet ein möglichst robuster und starker Vorkonditionierer verwendet werden, welcher die Kondition der Submatrix A_{Ω_f} wesentlich

verringert, wie zum Beispiel ein algebraisches Mehrgitterverfahren. Auch wenn der Vorkonditionierer in der Präasymptotik aufwändiger zu berechnen ist, so sollte sich wegen der Tatsache, dass die freie Oberfläche niederdimensional ist und damit Ω_f klein ist im Vergleich zu Ω die Gesamtkomplexität des Verfahrens nicht wesentlich erhöhen.

Auf dem übrigen Gebiet Ω_1 wird hingegen eine relativ glatte Lösung erwartet, sodass hier eine kostengünstige Vorkonditionierungsmethode eingesetzt werden kann, wie zum Beispiel ein Jacobi-Verfahren.

Eigener Beitrag

Im Rahmen dieser Arbeit wurde das oben beschriebene Verfahren in den bestehenenden zweiphasigen Strömungslöser NaSt3DGPF der Universität Bonn implementiert. Im Folgenden sind die wesentlichen Beiträge dieser Arbeit zusammengefasst

- Vollständige Einarbeitung in den Strömungslöser NaSt3DGPF und in das *Portable, Extensible Toolkit for Scientific Computation* (PETSc) [15], welches Werkzeuge zum Lösen von großen linearen Gleichungssystemen bereitstellt.
- Implementierung einer Gebietsaufteilung mithilfe der Level-Set Funktion in Ω_f und Ω_1. Diese Aufteilung wird in geeigneten PETSc Indexmengen gespeichert.
- Zuordnung dieser Indexmengen an den PETSc-Vorkonditionierer ASM (Additive Schwarz Methode), wodurch zwei Unterprobleme entstehen.
- Auf diesen Unterproblemen sind verschiedene Vorkonditionierer einstellbar.
- Implementierung einer Eigenwertberechnung auf den Unterproblemen zur Bestimmung der Kondition.

Übersicht

Im Folgenden wird ein Überblick über die einzelnen Kapitel dieser Arbeit gegeben

Kapitel 2 erläutert den mathematischen Hintergrund von Zweiphasenströmungen. Nach einer Beschreibung der mathematischen Modellierung wird kurz auf die Parallelisierung eingegangen.

Kapitel 3 beschreibt das Prinzip der Vorkonditionierung. Nach einer Vorstellung der klassischen Verfahren wird die allgemeine alternierende Schwarz Methode beschrieben und schließlich ihre Bedeutung im Spezialfall der additiven Schwarz Methode in dieser Arbeit erklärt.

Kapitel 4 stellt die Implementierung der gebietsabhängigen Vorkonditionierung in den NaSt3DGPF dar. Zunächst wird die Diskretisierung der Druckberechnung durch die Poissongleichung und daraufhin die Vorkonditionierung mit PETSc beschrieben. Schließlich wird erklärt wie die Gebietszerlegung mithilfe der Level-Set-Funktion implementiert wurde.

Kapitel 5 dient der Präsentation der Ergebnisse von simulierten Strömungsproblemen. Nach einer Verifikation der Gebietszerlegung, werden die Kondition, die Konvergenz und

die Laufzeit des implementierten Verfahrens analysiert. Dabei können für einige Testrechnungen erhebliche Geschwindigkeitsvorteile gegenüber den bisherigen Verfahren erzielt werden.

Kapitel 6 fasst die vorliegende Arbeit zusammen und gibt einen Ausblick auf mögliche Weiterentwicklungen.

Danksagung

An dieser Stelle möchte ich mich bei allen bedanken, die zum Entstehen dieser Arbeit beigetragen haben. Professor Griebel danke ich für die Vergabe eines interessanten Themas sowie seine Anregungen bei Schwierigkeiten und Problemen. Ein großer Dank gilt meinen Betreuern Peter Zaspel und Margrit Klitz. Insbesondere bedanke ich mich bei Peter Zaspel für seine zahlreichen Hilfestellungen bei technischen Problemen und für die vielen Anregungen und Korrekturen zu dieser Arbeit.

Weiterhin möchte ich meinen Kommilitonen Wen Zeng und Christian Reinecke für das Korrekturlesen dieser Arbeit danken. Schließlich bedanke ich mich bei meiner Familie für ihre Unterstützung und ihr Verständnis.

Kapitel 2

Strömungssimulation

In diesem Kapitel geht es um die Berechnung von Zweiphasenströmungen, wie zum Beispiel Strömungen von Wasser und Luft. Dabei wird zunächst erklärt, wie man einphasige Strömungen mit einem geeigneten mathematisches Modell beschreiben kann. Dies geschieht mithilfe der Navier-Stokes-Gleichungen, welche ein System partieller Differenzialgleichungen bilden. Der Ansatz des einphasigen Modelles wird dann auf ein Zweiphasensystem erweitert. Hierbei wird für jede der beiden Phasen ein solches System aufgestellt und mittels der Oberflächenspannkraft wird ein Zusammenhang zwischen den beiden Systemen hergestellt, wodurch man eine Darstellung von Zweiphasenströmungen erhält.

Daraufhin wird eine Projektionsmethode zur Lösung des mathematischen Modellproblems erläutert. Ein wesentlicher Teilschritt dieser Methode ist die Druckberechnung, welche im Mittelpunkt dieser Arbeit steht. Dabei muss in jedem Zeitschritt eine Poissongleichung gelöst werden, was nach geeigneter Diskretisierung zu einem linearen Gleichungssystem führt. Dieses Gleichungssystem wird bei einer feinen Auflösung und damit vielen Unbekannten sehr groß, weshalb man versucht dieses Gleichungssystem auf möglichst effiziente Art zu Lösen.

Ein weiterer zentraler Punkt dieser Arbeit ist das Beschreiben der sogenannten freien Oberfläche, welche die Trennfläche der beiden Phasen bildet (zum Beispiel die Wasseroberfläche bei Wasser und Luft), mithilfe der Level-Set-Methode. Dabei ist die Idee eine möglichst glatte Funktion zu finden, deren Nullniveaumenge gleich dieser freien Oberfläche ist. Die Level-Set-Funktion ist also eine Abstandsfunktion, die jedem Punkt im Gebiet den Abstand zur freien Oberfläche zuordnet. Die zeitliche Änderung dieser und damit auch der Level-Set-Funktion wird durch die Transportgleichung beschrieben. Mithilfe der Level-Set-Funktion wird es später möglich sein, das gesamte Gebiet aufzuteilen. Einen Teil bildet ein festgelegter Bereich um die freie Oberfläche und die restlichen Gebietszellen bilden den anderen Teil. Auf beiden Teilen werden dann verschiedene Vorkonditionierer zur Druckberechung eingesetzt, um die Konvergenz zu beschleunigen.

Schließlich wird noch kurz darauf eingegangen, wie man ein Programm zur Strömungssimulation parallelisiert, das heißt auf vielen Prozessoren simultan berechenbar macht. Dies liefert gegenüber dem sequenziellen Rechnen einen enormen Geschwindigkeitsvorteil. Weiterhin ist zusammenfassend ein parallelisierter Algorithmus angegeben, mit dem es möglich ist Strömungen von zwei Phasen zu simulieren.

2.1 Das mathematische Modell

Bei einer Strömungssimulation muss man zunächst die Vorgänge aus der Natur, wie zum Beispiel eine Wasserströmung, die um einen Brückenpfeiler fließt, in ein geeignetes mathematisches Modell übersetzen. Mit diesem Modell ist es dann später möglich, Berechnungen durchzuführen, welche die natürlichen Vorgänge möglichst realistisch wiederspiegeln. In dieser Arbeit werden instationäre, also sich zeitlich ändernde, Strömungen von viskosen Fluiden behandelt. Ein Fluid bezeichnet eine Flüssigkeit oder ein Gas und die Viskosität gibt die Zähigkeit eines Fluids an. So hat Luft eine sehr geringe, Wasser eine etwas höhere und Honig eine sehr hohe Viskosität. Diese Strömungen lassen sich physikalisch mithilfe der Navier-Stokes-Gleichungen beschreiben. Betrachtet man zunächst ein Einphasensystem von solchen Strömungen, also ein System mit nur einem Fluid, so erhält man ein System partieller Differentialgleichungen zweiter Ordnung bestehend aus den Impulsgleichungen

$$\begin{aligned}
\frac{\partial u}{\partial t} + \frac{1}{\rho}\frac{\partial p}{\partial x} &= \frac{\mu}{\rho}\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2}\right) - \frac{\partial (u^2)}{\partial x} - \frac{\partial (uv)}{\partial y} - \frac{\partial (uw)}{\partial z} + g_x, \\
\frac{\partial v}{\partial t} + \frac{1}{\rho}\frac{\partial p}{\partial y} &= \frac{\mu}{\rho}\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2} + \frac{\partial^2 v}{\partial z^2}\right) - \frac{\partial (uv)}{\partial x} - \frac{\partial (v^2)}{\partial y} - \frac{\partial (vw)}{\partial z} + g_y, \\
\frac{\partial w}{\partial t} + \frac{1}{\rho}\frac{\partial p}{\partial z} &= \frac{\mu}{\rho}\left(\frac{\partial^2 w}{\partial x^2} + \frac{\partial^2 w}{\partial y^2} + \frac{\partial^2 w}{\partial z^2}\right) - \frac{\partial (uw)}{\partial x} - \frac{\partial (vw)}{\partial y} - \frac{\partial (w^2)}{\partial z} + g_z
\end{aligned}$$

und der Kontinuitätsgleichung

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} = 0, \tag{2.1}$$

welches in einem Gebiet $\Omega = [0, \text{xlength}] \times [0, \text{ylength}] \times [0, \text{zlength}] \subset \mathbb{R}^3$ und in dem Zeitintervall $[0, t_{end}]$ gelöst werden soll. Die gesuchten Größen hierbei sind die Geschwindigkeiten $u, v, w : \Omega \times [0, t_{end}] \to \mathbb{R}$, welche man auch zu einem Geschwindigkeitsvektor $\vec{u} = (u, v, w)^T$ zusammenfassen kann, sowie der Druck $p : \Omega \times [0, t_{end}] \to \mathbb{R}$.

- ρ bezeichnet dabei die Dichte des Fluids und wird bei Einphasenströmungen als konstant angenommen,
- μ ist die Viskosität des Fluids und wird ebenfalls im einphasigen Fall als konstant angenommen,
- $\vec{g} = (g_x, g_y, g_z)^T$ sind äußere Kräfte, wie zum Beispiel die Erdanziehungskraft.

Bezeichnet man mit $\vec{u} \otimes \vec{u}$ die 3×3 Matrix $\vec{u} \cdot \vec{u}^T$, so kann das obige einphasige Strömungsmodell auch verkürzt dargestellt werden als

$$\frac{\partial \vec{u}}{\partial t} + \frac{\nabla p}{\rho} = \frac{\mu}{\rho}\Delta \vec{u} - \nabla \cdot (\vec{u} \otimes \vec{u}) + \vec{g}, \tag{2.2}$$

$$\nabla \cdot \vec{u} = 0. \tag{2.3}$$

Weiterhin müssen für dieses System von partiellen Differenzialgleichungen, damit es aus mathematischer Sicht ein wohlgestelltes Problem ist, Anfangs- und Randbedingungen vorgegeben werden. Diese sind dafür da die Situation eines speziellen Problems zu Beginn und

an den Gebietsrändern zu beschreiben. Durch verschiedene Anfangs- und Randbedingungen kann man also verschiedene Strömungen simulieren. Als Anfangsbedingungen gibt man die Geschwindigkeiten $\vec{u}$ und den Druck p zum Zeitpunkt $t = 0$ an. Die Randbedingungen werden auf dem Gebietsrand $\Gamma = \partial\Omega$ für alle $t \in [0, t_{end}]$ vorgegeben. Im Fall $\Omega = [0, \text{xlength}] \times [0, \text{ylength}] \times [0, \text{zlength}]$ sind die Ränder gerade die Außenwände des Quaders. Es ist natürlich möglich auf den verschiedenen Außenwänden verschiedene Randdaten vorzugeben, wodurch jedes mal andere Simulationen entstehen. Dabei unterscheidet man zwischen verschiedenen Typen von Randbedingungen

- Haftbedingungen: Das Fluid haftet auf diesem Rand, also verschwindet dort der Geschwindigkeitsvektor $\vec{u}|_\Gamma = 0$.
- Fixe Ein- und Ausströmbedingungen: Das Fluid strömt auf diesem Rand mit einer fest vorgegebenen Geschwindigkeit $\vec{u}_0$ ein beziehungsweise aus $\vec{u}|_\Gamma = \vec{u}_0$.
- Rutschbedingungen: Das Fluid gleitet ohne Reibungsverluste auf diesem Rand, das heißt nur die tangentialen Geschwindigkeiten bleiben erhalten, welche keine Änderung in Richtung des Normalenvektors haben.
- Ausströmbedingungen: Die Fluidgeschwindigkeit auf diesem Rand ändert sich nicht in Normalenrichtung.

Weiterhin muss beim Setzen der Randbedingungen darauf geachtet werden, dass das Randintegral über die Geschwindigkeiten senkrecht zum festen Rand Null ist, denn aus (2.3) und dem Satz von Gauß folgt

$$0 = \int_\Omega \nabla \cdot \vec{u}\, d\vec{x} = \int_{\partial\Omega} \vec{u} \cdot \vec{n}\, dF. \tag{2.4}$$

Betrachtet man nun ein Zweiphasensystem, also ein System, in dem zwei Fluide simuliert werden (oft sind es Wasser und Luft), so existiert ein (unter Umständen großer) Unterschied in der Dichte und in der Viskosität der beiden Fluide. Zum Beispiel liegt bei Luft und Wasser ein Dichtesprung von 1 zu 1000 vor. Die freie Oberfläche bezeichnet die Trennfläche zwischen den beiden Phasen (wie zum Beispiel die Wasseroberfläche bei Wasser und Luft). Die Materialeigenschaften innerhalb einer Phase werden als konstant angenommen und so erhält man zunächst zwei getrennte, wie oben beschriebene Einphasensysteme. Der Zusammenhang zwischen den beiden Einphasensystemen bildet die Oberflächenspannung. Durch Ausnutzen der Randbedingungen auf dem freien Rand erhält man die integrale Darstellung der Impulsgleichung für zwei Phasen

$$\int_\Omega \frac{\partial \vec{u}}{\partial t} + \frac{\nabla p}{\rho}\, d\vec{x} = \int_\Omega \frac{\mu}{\rho}\Delta\vec{u} - \nabla \cdot (\vec{u} \otimes \vec{u}) + \vec{g}\, d\vec{x} - \int_{\Gamma_f} \frac{\sigma\kappa\vec{n}}{\rho}\, dF, \tag{2.5}$$

wobei Γ_f den freien Rand, σ den Oberflächenspannungskoeffizienten, κ den Krümmungsparameter und $\vec{n}$ den Einheitsnormalenvektor bezeichnet. Eine genaue Beschreibung zur Herleitung des Zweiphasensystems ist in [2] gegeben. Mithilfe dieser zweiphasigen Integraldarstellung und der Level-Set-Technik, welche in Abschnitt 2.3 erklärt wird, können Zweiphasenströmungen wie folgt dargestellt werden

$$\frac{\partial \vec{u}}{\partial t} + \frac{\nabla p}{\rho(\phi)} = \frac{\mu(\phi)}{\rho(\phi)}\Delta\vec{u} - \nabla \cdot (\vec{u} \otimes \vec{u}) + \vec{g} - \frac{\sigma\kappa(\phi)\delta(\phi)\nabla\phi}{\rho(\phi)}. \tag{2.6}$$

Dabei ist ϕ die sogenannte Level-Set-Funktion, welche die freie Oberfläche beschreibt, und δ ist eine Dirac-Funktion welche auf $\Gamma_f := \{\phi = 0\}$ konzentriert ist. Ein wesentlicher Aspekt des Zweiphasenmodells ist, dass die Dichte ρ und die Viskosität μ sowie der Krümmungsparameter κ nicht mehr wie im Einphasensystem konstant sind, sondern von der Level-Set-Funktion ϕ abhängen. Die genaue Dichte- und Viskositätsverteilung wird in Abschnitt 2.3 beschrieben. Die Darstellung (2.6) von Zweiphasenströmungen wird in ähnlicher Form von Croce, Griebel und Schweitzer [4], Sussman, Smereka und Osher [10] und Unverdi und Tryggvason [11] verwendet. Die Idee, die Oberflächenspannkräfte auf dem freien Rand durch die Dirac-Funktion δ zu aktivieren, ist von Brackbill, Kothe und Zemach [1].

Das Ziel ist es nun das Zweiphasensystem zu berechnen.

2.2 Die Projektionsmethode

Um die Geschwindigkeiten und den Druck im oben beschriebenen Zweiphasensystem für den nächsten Zeitschritt zu berechnen, wird eine Projektionsmethode verwendet, bei der nicht divergenzfreie Schätzgeschwindigkeiten mithilfe des Drucks in den Raum der divergenzfreien Geschwindigkeiten projeziert werden. Dabei werden zuerst die Schätzgeschwindigkeiten

$$\vec{u}^* = \vec{u}^n + \delta t \left(\frac{\mu(\phi^n)}{\rho(\phi^n)} \Delta \vec{u}^n - \nabla \cdot (\vec{u}^n \otimes \vec{u}^n) + \vec{g} - \frac{\sigma \kappa(\phi^n) \delta(\phi^n) \nabla \phi^n}{\rho(\phi^n)} \right) \tag{2.7}$$

aus den zweiphasigen Impulsgleichungen (2.6) ohne den Druckgradienten $\frac{\nabla p}{\rho(\phi)}$ bestimmt. Hierbei ist die Dichte im Gegensatz zum Einphasensystem nicht mehr konstant, sondern hängt von der freien Oberfläche ab. Die freie Oberfläche wird durch die Nullniveaumenge der Level-Set-Funktion ϕ approximiert, welche für jeden Punkt $\vec{x} \in \Omega$ den Euklidischen Abstand zum freien Rand angibt. Um den Druck und die Geschwindigkeiten für den neuen Zeitschritt zu berechnen, benötigt man auch eine neue Aprroximation der freien Oberfläche ϕ^{n+1}. Eine genaue Beschreibung für die Berechnung der Level-Set-Funktion ist in Kapitel 2.3 gegeben. Nun wird aus

$$\frac{\vec{u}^{n+1} - \vec{u}^*}{\delta t} + \frac{\nabla p^{n+1}}{\rho(\phi^{n+1})} = 0, \tag{2.8}$$

$$\nabla \cdot \vec{u}^{n+1} = 0 \tag{2.9}$$

die Poissongleichung hergeleitet und dann werden die neuen Geschwindigkeiten $\vec{u}^{n+1}$ bestimmt. Durch Anwenden der Divergenz auf Gleichung (2.8) und Ausnutzen von (2.9) folgt die Poissongleichung

$$\nabla \cdot \left(\frac{1}{\rho(\phi^{n+1})} \nabla p^{n+1} \right) = \nabla \cdot \frac{\vec{u}^*}{\delta t}. \tag{2.10}$$

Die Geschwindigkeiten $\vec{u}^{n+1}$ berechnen sich aus der Druckkorrektur von $\vec{u}^*$

$$\vec{u}^{n+1} = \vec{u}^* - \frac{\delta t}{\rho(\phi^{n+1})} \nabla p^{n+1}. \tag{2.11}$$

Damit die Poissongleichung (2.10) zu keinem unterbestimmten Gleichungssystem führt, müssen Randbedingungen für den Druck vorgegeben werden. Setzt man $\vec{u}^*|_\Gamma = \vec{u}^{n+1}|_\Gamma$ auf dem Rand $\Gamma = \partial\Omega$, so ergibt sich die Formel

$$\left.\frac{\partial p^{n+1}}{\partial \vec{n}}\right|_\Gamma = \nabla p \cdot \vec{n} = \frac{\rho(\phi)}{\delta t}(\vec{u}^*|_\Gamma - \vec{u}^{n+1}|_\Gamma) \cdot \vec{n} = 0, \tag{2.12}$$

welche homogene Randbedingungen beschreibt.

Nach einer geeigneten Diskretisierung der Poissongleichung (2.10), welche in Kapitel 4.1 beschrieben wird, ensteht ein zu lösendes lineares Gleichungssystem $Ax = b$. Die Qualität des Iterationsverfahrens zur Lösung dieses Gleichungssystems hängt entscheidend von der Kondition der Systemmatrix A ab und jene verschlechtert sich, je höher der Dichtesprung zwischen zwei Phasen ist. Um diesen Effekt der verschlechterten Kondition abzudämpfen wird das Gleichungssystem zunächst *vorkonditioniert*. Die Methode der Vorkonditionierung ist in Kapitel 3.1 beschrieben. Im Rahmen dieser Arbeit soll ein kleiner Bereich um die freie Oberfläche herausgefiltert werden, welcher den Dichtesprung der Phasen enthält. Dieser Bereich wird dann mit einem stärkeren aber auch aufwändigeren Vorkonditioner behandelt als der restliche Bereich, welcher keinen Dichtesprung der Phasen enthält. Um den Bereich um die freie Oberfläche herauszufiltern, ist die Level-Set-Methode von großer Bedeutung. Diese wird im Folgenden beschrieben.

2.3 Die Level-Set-Methode

Die Level-Set-Methode ist ein Verfahren, um geometrische Objekte und deren Bewegung approximativ zu verfolgen. Im Fall der zweiphasigen Navier-Stokes-Gleichungen wird dieses Verfahren verwendet, um die freie Oberfläche zwischen zwei Phasen zu beschreiben. Die Grundidee besteht darin, eine möglichst glatte Funktion zu konstruieren, deren Nullniveaumenge gleich der freien Oberfläche ist. Diese Idee geht auf Sussman, Smereka und Osher [10] zurück. Betrachtet man eine geschlossene Fläche Γ_f in drei Dimensionen, die zwei Gebiete voneinander trennt, so ist das Ziel die zeitliche Veränderung der Fläche unter einer gegebenen Geschwindigkeitsfunktion F in Normalenrichtung zu verfolgen. Einflüsse in tangentiale Richtung werden hierbei vernachlässigt. Sei $\Omega \subset \mathbb{R}^3$ ein festes Gebiet und $\Gamma_f(t) \subset \Omega$ für alle $t \in [0, t_{end}]$ eine sich zeitlich ändernde Fläche. Gesucht wird eine Funktion $\phi(\vec{x}, t)$ auf $\Omega \times [0, t_{end}]$, deren Nullniveaumenge $\{\phi = 0\}$ gleich $\Gamma_f(t)$ ist, also

$$\Gamma_f(t) = \{\vec{x} \,:\, \phi(\vec{x}, t) = 0\} \quad \forall t \in [0, t_{end}]. \tag{2.13}$$

Für einen Partikel $\vec{x}(t) \in \Gamma(t)$ gilt somit

$$\phi(\vec{x}(t), t) = 0 \quad \forall t \in [0, t_{end}]. \tag{2.14}$$

Differenziert man (2.14) nach t, so erhält man

$$\phi_t + \nabla\phi(\vec{x}(t), t) \cdot \vec{x}'(t) = 0. \tag{2.15}$$

Hierbei ist $\nabla\phi$ auf Γ_f orthogonal zur Nullniveaumenge $\{\phi = 0\}$ und $F := \vec{x}'(t) \cdot \vec{n}$ ist das Geschwindigkeitsfeld, das orthogonal zu den Niveaumengen der Funktion ϕ steht, wobei

$\vec{n} = \frac{\nabla\phi}{|\nabla\phi|}$ der Einheitsnormalenvektor bezüglich der Nullniveaumenge $\{\phi = 0\}$ ist. Es gilt also

$$\begin{aligned} &\vec{x}'(t) \cdot \vec{n} = F \\ \Leftrightarrow\ &\vec{x}'(t) \cdot \frac{\nabla\phi}{|\nabla\phi|} = F \\ \Leftrightarrow\ &\vec{x}'(t) \cdot \nabla\phi = F|\nabla\phi| \end{aligned} \tag{2.16}$$

Durch Einsetzen der Gleichung (2.16) in Gleichung (2.15) erhält man die Transportgleichung für ϕ in Form eines Anfangswertproblems

$$\begin{aligned} \phi_t + F|\nabla\phi| &= 0, \\ \phi(\vec{x}, 0), & \end{aligned} \tag{2.17}$$

welche die zeitliche Entwicklung der Level-Set-Funktion ϕ beschreibt. Im Fall der Navier-Stokes-Gleichungen ist ein globales Geschwindigkeitsfeld $\vec{u}$ gegeben. Da nur die Transportgeschwindigkeit in Normalenrichtung betrachtet wird, gilt

$$F = \vec{u} \cdot \vec{n} = \vec{u} \cdot \frac{\nabla\phi}{|\nabla\phi|}. \tag{2.18}$$

Durch Einsetzen von (2.18) in (2.17) ergibt sich die zeitliche Entwicklung der Level-Set-Funktion bezüglich der Strömungsgeschwindigkeit $\vec{u}$ zu

$$\begin{aligned} \phi_t + \vec{u} \cdot \nabla\phi &= 0, \\ \phi(\vec{x}, 0). & \end{aligned} \tag{2.19}$$

Weiterhin soll die Level-Set-Funktion den folgenden Bedingungen genügen

$$\begin{aligned} \phi(\vec{x}, t) \begin{cases} > 0, & \text{falls } \vec{x} \text{ in Phase 1 ist} \\ = 0, & \text{falls } \vec{x} \in \Gamma_f \\ < 0, & \text{falls } \vec{x} \text{ in Phase 2 ist} \end{cases} \\ |\nabla\phi| = 1. \end{aligned} \tag{2.20}$$

Dies kann man fordern, da lediglich die Nullniveaumenge von ϕ die relevante freie Oberfläche darstellt. Im Allgemeinen ist jedoch die sich zeitlich ändernde Level-Set-Funktion keine Abstandsfunktion mehr. Dies gilt nur dann, wenn die sogenannte Eikonalgleichung (2.20) erfüllt ist. Für die Berechnung der Oberflächenspannungskräfte ist es aber wichtig, dass die Level-Set-Funktion während der gesamten zeitlichen Entwicklung eine vorzeichenbehaftete Abstandsfunktion innerhalb einer ϵ-Umgebung des freien Randes Γ_f bleibt. Falls die Abstandseigenschaft der Level-Set-Funktion nicht aufrecht erhalten wird, entstehen mit der Zeit sehr große oder sehr kleine Geschwindigkeitsgradienten an der freien Oberfläche, die zu verzerrten Oberflächenspannungskräften oder Oszillationen führen können. Daher sollte die Level-Set-Funktion durch eine Reinitialisierung zu einer Abstandsfunktion umgewandelt werden ohne dabei die Nullniveaumenge zu verändern. Eine Möglichkeit dies umzusetzen, ist ein Verfahren von Sussman, Smereka und Osher [10]. Bei diesem ist die Idee, iterativ in künstlicher Zeit τ eine stationäre Lösung des Anfangswertproblems

$$\begin{aligned} d_\tau + \text{sign}(\phi)(|\nabla d| - 1) &= 0, \\ d(\vec{x}, 0) = \phi(\vec{x}) & \end{aligned} \tag{2.21}$$

zu berechnen, wobei die signum-Funktion durch

$$\text{sign}(\phi) := \begin{cases} -1, & \text{falls } \phi < 0 \\ 0, & \text{falls } \phi = 0 \\ 1, & \text{falls } \phi > 0 \end{cases} \tag{2.22}$$

definiert ist. Die stationäre Lösung von (2.21) ist eine vorzeichenbehaftete Abstandfunktion, deren Nullniveaumenge gleich der der Level-Set-Funktion ϕ ist, da $\text{sign}(0) = 0$ gilt. Ist die stationäre Lösung erreicht, so wird die Level-Set-Funktion ϕ durch diese stationäre Lösung d ersetzt. Diese Methode der Reinitialisierung hat den Vorteil, dass bei einer komplizierten Anfangsgeometrie die freie Oberfläche sehr einfach beschrieben werden kann, indem $\phi = -1$ in den Gitterzellen der ersten Phase und $\phi = 1$ in den Gitterzellen der zweiten Phase gesetzt wird. Daraufhin wird eine globale Reinitialisierung der Anfangsdaten vorgenommen, welche bis zur maximalen Gebietslänge τ_{max} durchgeführt wird, und man erhält die gewünschte Abstandsfunktion. Mithilfe der Abstandsfunktion kann man nun die Dichte- und Viskositätsverteilung definieren, welche für die Projektionsmethode zur Druckberechnung benötigt werden.

$$\begin{aligned} \rho(\phi) &= \rho_g + (\rho_l - \rho_g)H(\phi), \\ \mu(\phi) &= \mu_g + (\mu_l - \mu_g)H(\phi), \end{aligned} \tag{2.23}$$

wobei $\rho_g, \rho_l, \mu_g, \mu_l$ die Dichte beziehungsweise die Viskosität der entsprechenden Gas- oder Liquidphase bezeichnet. Die Heavyside'sche Sprungfunktion $H(\phi)$ ist wie folgt definiert

$$H(\phi) := \begin{cases} 0, & \text{falls } \phi < 0 \\ \frac{1}{2}, & \text{falls } \phi = 0 \\ 1, & \text{falls } \phi > 0 \end{cases} \tag{2.24}$$

Im folgenden Abschnitt ist das gesamte Verfahren zur zweiphasigen Strömungssimulation algorithmisch zusammengefasst und es wird erklärt, wie ein solcher Algorithmus parallelisiert werden kann.

2.4 Parallelisierung

In der heutigen Zeit sind Parallelrechner, welche aus einigen dutzend bis hin zu mehreren tausend leistungsfähigen Prozessoren bestehen, weit verbreitet. Mit diesen ist es möglich, auf den verschiedenen Prozessoren gleichzeitig zu rechnen und zu kommunizieren. Ein parallelisierter Algorithmus geht zunächst aus einem möglichst effizienten sequentiellen Verfahren hervor, welches dann durch die Parallelisierung weiter beschleunigt wird. In dem Fall der zweiphasigen Strömungsberechnung geschieht dies sehr intuitiv. Die zur Strömungsberechnung notwendigen partiellen Differentialgleichungen müssen alle auf dem gesamten Gebiet Ω gelöst werden. Das Gebiet wird dann in N Teilgebiete $\Omega_1, \ldots, \Omega_N$ zerlegt und auf jedem Prozessor wird ein Teilgebiet behandelt. Die erste sogenannte Gebietszerlegungsmethode wurde 1870 von H.A. Schwarz [8] entwickelt, weswegen man auch von Schwarz'schen Gebietszerlegungsmethoden spricht. Die Strategie, wie das Gebiet Ω zerlegt wird, hängt stark von der jeweiligen Problemstellung und dem verwendeten Gitter ab. Hier soll ein kartesisches Gitter und eine Zerlegung in Blöcke verwendet werden. Damit die Diskretisierungssterne bei der Berechnung nicht auf undefinierte Werte zugreifen, müssen

für jeden Block Überlappungszellen (Randbordüren) an den Rändern hinzugefügt werden. Die Werte in diesen Randbordüren sind gleich den Werten der entsprechenden inneren Gitterzellen des anliegenden Gebietes. Damit in den Randbordüren immer die aktuellen Werte zur Verfügung stehen, müssen diese an geeigneter Stelle zwischen den einzelnen Prozessoren kommuniziert werden. Abbildung 2.1 veranschaulicht, zur Übersichtlichkeit in zwei Dimensionen, die Druckkommunikation zweier benachbarter Teilgebiete.

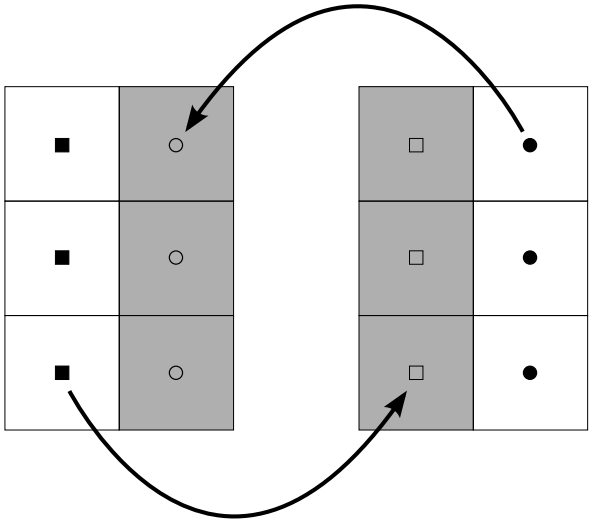

Abbildung 2.1: Druckkommunikation zweier benachbarter Teilgebiete

Die grauen Zellen sind die künstlich hinzugefügten Randbordüren, in welche die Werte des anliegenden Gebietes übergeben werden müssen. Da der Datenaustausch relativ viel Zeit im Vergleich zur Ausführung von Rechenoperationen benötigt, muss man darauf achten den Kommunikationsaufwand zu minimieren. Andernfalls kann ein erhöhter Kommunikationsaufwand den Geschwindigkeitsvorteil durch die Parallelisierung reduzieren oder sogar auslöschen. Weiterhin muss darauf geachtet werden, dass jeder Prozessor in etwa gleich belastet wird. Dies geschieht hier, indem die Blöcke, in die das Gesamtgebiet zerlegt wurde, in etwa gleich groß angelegt werden und somit annähernd gleich viele Unbekannte behandelt werden müssen. Außerdem sollten in etwa gleich viele Prozesse auf jedem Prozessor laufen.

Jedes Teilgebiet wird dann von einem Prozess behandelt, welcher einem Prozessor zur Berechnung der Unbekannten zugeordnet werden kann. Somit reduziert sich bei fester Problemgröße der Speicher und Rechenaufwand pro Prozess mit wachsender Anzahl an Teilgebieten. Bei Verwendung von P Prozessoren und P Teilgebieten wäre eine P-fache Verkürzung der Rechenzeit optimal, was jedoch wegen des Kommunikationsaufwands nicht zu realisieren ist. Dennoch werden mit einer parallelisierten Strömungssimulation erhebliche Geschwindigkeitsvorteile gegenüber dem sequentiellen Rechnen erzielt. In Algorithmus 1 ist die parallele Berechnung von Zweiphasenströmungen noch einmal zusammengefasst dargestellt

Algorithmus 1 Algorithmus zur parallelen Berechnung von Zweiphasenströmungen

Setze $t := 0$, $n := 0$
Initialisiere $\vec{u}$, p, ϕ
if $(\phi \in \{-1, 1\})$ **then**
  Setze τ_{end} :=Größte Kantenlänge des Gebietes
  Setze die Randbedingungen für die Level-Set-Funktion
  while $\tau < \tau_{end}$ **do**
    Reinitialisiere nach Abschnitt 2.3 und kommuniziere vor jedem Teilschritt die Level-Set-Werte der Randbordüren
  end while
end if
Setze die Randbedingungen für u^n
while $t \leq t_{end}$ **do**
  Bestimme den neuen Zeitschritt δt in jedem Teilgebiet und bestimme durch eine Kommunikation das Minimum
  Berechne $\vec{u}^*$ aus $\vec{u}^n$ und ϕ^n nach (2.7)
  Setze die Randbedingungen für $\vec{u}^*$
  Berechne die rechte Seite der Poissongleichung (2.10) $RHS := \nabla \cdot \frac{\vec{u}^*}{\delta t}$
  Berechne vorläufige Level-Set-Funktion ϕ^* durch Lösen der Transportgleichung (2.19)
  while $\tau < 2\epsilon$ **do**
    Berechne ϕ^{n+1} durch Reinitialisierung von ϕ^* nach Abschnitt 2.3 und kommuniziere vor jedem Teilschritt die Level-Set-Werte der Randbordüren
  end while
  while $it < it_{max}$ **or** $||r^{it}|| < \epsilon^*$ **do**
    Löse iterativ $\nabla \cdot (\frac{1}{\rho(\phi^{n+1})} \nabla p^{n+1}) = RHS$ nach geeigneter Diskretisierung und kommuniziere vor jedem Iterationsschritt die Druckwerte der Randbordüren
  end while
  Bestimme $\vec{u}^{n+1}$ über die Druckkorrektur (2.11)
  Kommuniziere die Geschwindigkeitswerte der Randbordüren
  Setze die Randbedingungen für $\vec{u}^{n+1}$
  Setze $t := t + \delta t$, $n := n + 1$
end while

Kapitel 3

Vorkonditionierung

In diesem Kapitel wird zunächst die Methode der Vorkonditionierung erklärt sowie einige wichtige Vorkonditionierer vorgestellt. Die in Kapitel 2 vorgestellte wesentliche Druckberechnung soll in dieser Arbeit weiter optimiert werden. Das Problem ist der Dichtesprung an der Phasengrenze, welcher in die Druckberechnung mit einfließt und eine schlechte Kondition des aus der Poissongleichung entstehenden Gleichungssystems bewirkt. Daher verwendet man eine Vorkonditionierung. Im Gegensatz zu den bisherigen Vorkonditionierern soll in dieser Arbeit eine gebietsabhängige Vorkonditionierung verwendet werden. Dazu wird der Ansatz der additiven Schwarz Methode eingesetzt. Diese wird in einem allgemeinen Rahmen über die alternierende Schwarz Methode in Abschnitt 3.2 erklärt. Schließlich wird in Abschnitt 3.3 beschrieben, wie die Gebietszerlegung genau durchgeführt wird und welche Vorteile diese bewirkt.

3.1 Klassische Vorkonditionierung

Ein wichtiger Teilschritt der Projektionsmethode ist die numerische Lösung der Poissongleichung (2.10). Wird diese mit zentralen Differenzen diskretisiert (siehe Kapitel 4.1) so erhält man unter Berücksichtigung der Randbedingungen (2.12) ein lineares Gleichungssystem

$$Ax = b. \tag{3.1}$$

Zur Lösung des Gleichungssystems werden oft Krylov-Unterraum-Verfahren verwendet. Die Konvergenz dieser iterativen Verfahren hängt jedoch wesentlich von der Kondition

$$\kappa(A) = ||A|| \cdot ||A^{-1}|| \tag{3.2}$$

der Matrix A ab. Bei zweiphasigen Problemen verschlechtert sich die Kondition der Matrix, je größer der Dichtesprung zwischen den beiden Phasen ist. Daher konvergiert ein Krylov-Unterraum-Verfahren wie beispielsweise das BiCGSTAB Verfahren bei einem hohen Dichtesprung (beispielsweise $1 : 1000$ bei Luft und Wasser) nur sehr langsam. Um die Konvergenz zu beschleunigen wird Gleichung (3.1) zunächst mit einer regulären Matrix P^{-1} von links multipliziert

$$P^{-1}Ax = P^{-1}b. \tag{3.3}$$

Das Ziel ist es eine reguläre Matrix P^{-1} zu finden, die mit möglichst geringem Rechen und Speicheraufwand zu bestimmen ist. Gleichzeitig sollte $P^{-1}A$ eine möglichst gute Approximation der Einheitmatrix liefern, sodass $1 \leq \kappa(P^{-1}A) \ll \kappa(A)$ gilt. Die Matrix P wird als Vorkonditionierungsmatrix bezeichnet. Beispiele für Vorkonditionierer sind

- **Jacobi Verfahren**: Dabei wird die Vorkonditionierungsmatrix $P = D = \text{diag}(A)$ gesetzt. Der Vorteil dieser Methode ist, dass sie sehr kostengünstig und leicht anwendbar ist.

- **Gauß Seidel Verfahren**: Hier wird $P = D - L$ gesetzt, wobei $D = \text{diag}(A)$ und L die linke untere Dreiecksmatrix von A ist.

- **Unvollständige LU Zerlegung**: Würde man eine Faktorisierung LU der Matrix A berechnen und $U^{-1}L^{-1}$ als Vorkonditionierer verwenden, so würde dieses Verfahren nach einem Schritt konvergieren. Leider ist im Allgemeinen die Faktorisierung von dünnbesetzten Matrizen deutlich dichter besetzt als die Matrix selbst, wodurch wesentlich mehr Speicher verwendet werden muss, was zu Geschwindigkeitsverlusten führt. Außerdem ist das explizite Berechnen der Zerlegung sehr aufwendig. Daher werden L und U durch eine unvollständige Zerlegung approximiert. Eine Möglichkeit ist es zum Beispiel, bei einer LU Zerlegung alle Elemente zu ignorieren, die kleiner als eine gegebene Toleranz sind, wodurch weniger Speicher und Rechenzeit benötigt wird. Die unvollständige LU Zerlegung wird auch kurz mit ILU (incomplete LU) bezeichnet.

- **Blockvarianten**: Erweiterungen des Jacobi und des Gauß Seidel Verfahrens sind die entsprechenden Blockvarianten dieser Verfahren. Dabei werden die Unbekannten sowie die Vorkonditionierungsmatrix in Blöcke eingeteilt und es werden jeweils gleichzeitig kleinere lineare Gleichungssysteme gelöst. Da hier jeweils mehrere kleinere Probleme gleichzeitig gelöst werden können, sind die Blockvarianten auch ideal zum parallelen Rechnen. So kann beispielsweise pro Prozessor ein kleineres Gleichungssystem gelöst werden. Die Konvergenz der Blockverfahren hängt von der Partitionierung und Anzahl der Blöcke und Prozessoren, den vorgegebenen Toleranzen und der Qualität des eingesetzten iterativen Verfahrens für die kleineren Gleichungssysteme ab. Daher ist es sehr schwer, genaue Aussagen über die Konvergenz zu treffen.

- **Additive Schwarz Methode**: Diese Methode ist eine Erweiterung der Blockvarianten, wobei jetzt die Partitionierung der Blöcke von der Gebietsstruktur abhängt. So wird das Diskretisierungsgebiet zerlegt und anhand dieser Zerlegung geschieht dann die Aufteilung der Vorkonditionierungsmatrix. Der Unterschied zu den Blockvarianten ist, dass sich die Gebiete auch überlappen können. So ist die additive Schwarz Methode ohne Überlapp äquivalent zu dem Block Jacobi Verfahren.

Da die Vorkonditionierung mithilfe der Additiven Schwarz Methode im Vordergrund dieser Arbeit steht, wird diese Methode im folgenden Abschnitt detailiert erklärt. Dazu wird zunächst auf die verallgemeinerte alternierende Schwarz Methode eingegangen.

3.2 Die alternierende Schwarz Methode

Die alternierende Schwarz Methode ist die erste bekannte Gebietszerlegungsmethode und wurde von Hermann Amandus Schwarz [8] bereits 1870 entwickelt. Heute werden die Schwarz'schen Gebietszerlegungsmethoden vor allem dazu verwendet, Randwertprobleme von partiellen Differentialgleichungen zu parallelisieren und numerisch zu lösen. Damals diente die alternierende Schwarz Methode als Hilfsmittel für einen Beweis zur Existenz harmonischer Funktionen in nicht glatten Gebieten. In dieser Arbeit soll die alternierende Schwarz Methode als Vorkonditionierer für das aus der Poissongleichung (2.10) entstehende Gleichungssystem $Ax = b$ verwendet werden und wird daher hier kurz erläutert. Gegeben sei ein Gebiet Ω und die lineare elliptische partielle Differentialgleichung

$$\begin{aligned} \mathcal{L}u &= f && \text{in } \Omega, \\ u &= g && \text{auf } \partial\Omega. \end{aligned} \tag{3.4}$$

Bei der klassischen alternierenden Schwarz Methode wird Ω zunächst in zwei sich überlappende Teilgebiete Ω_1 und Ω_2 unterteilt. Mit $\Gamma_1 = \partial\Omega_1 \cap \Omega_2$ wird der Teil des Randes von Ω_1 bezeichnet, der im Inneren von Ω_2, also nicht auf dem Rand des ursprünglichen Gebietes Ω, liegt. Hingegen ist $\partial\Omega_1 \backslash \Gamma_1$ der Teil des Randes von Ω_1, der mit dem Rand von Ω übereinstimmt. Entsprechend sind auch Γ_2 und $\partial\Omega_2 \backslash \Gamma_2$ definiert. In Abbildung 3.1 werden die Bezeichnungen noch einmal dargestellt.

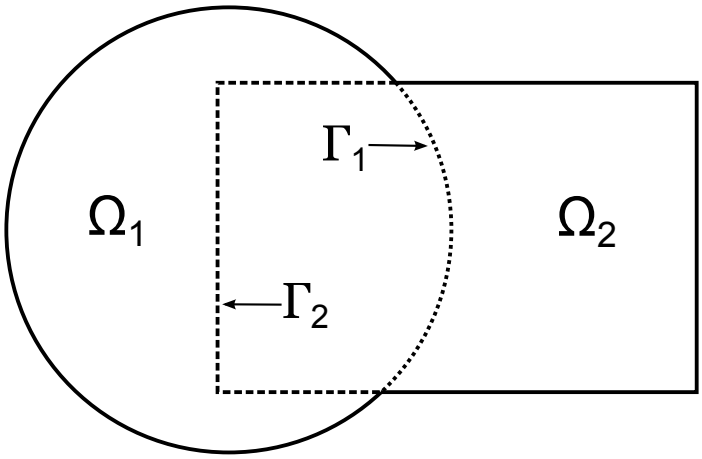

Abbildung 3.1: Gebietszerlegung in zwei sich überlappende Teilgebiete

Sind nun die Randwerte auf Γ_1 und Γ_2 bekannt, so kann das Randwertproblem (3.4) auf den Teilgebieten Ω_1 und Ω_2 gelöst werden. Dies motiviert den folgenden Algorithmus, wobei u_i^n die approximierte Lösung in Ω_i nach n Iterationen bezeichnet.

- Sei u_2^0 eine Startlösung auf Ω_2.
- Für $n = 1, 2, \ldots$ löse das Randwertproblem

$$\begin{aligned} \mathcal{L}u_1^n &= f && \text{in } \Omega_1, \\ u_1^n &= g && \text{auf } \partial\Omega_1 \backslash \Gamma_1, \\ u_1^n &= u_2^{n-1}|_{\Gamma_1} && \text{auf } \Gamma_1. \end{aligned} \tag{3.5}$$

Dann löse

$$\begin{aligned} \mathcal{L}u_2^n &= f && \text{in } \Omega_2, \\ u_2^n &= g && \text{auf } \partial\Omega_2\backslash\Gamma_2, \\ u_2^n &= u_1^n|_{\Gamma_2} && \text{auf } \Gamma_2. \end{aligned} \tag{3.6}$$

Das bedeutet, in jedem Halbschritt des Algorithmus wird das Randwertproblem (3.4) auf Ω_i mit den gegebenen Randdaten g auf dem echten Rand $\partial\Omega_i\backslash\Gamma_i$ und den vorher approximierten Randdaten auf dem künstlichen inneren Rand Γ_i gelöst. Da es im Allgemeinen nicht möglich ist, für diese Randwertprobleme (3.5) und (3.6) eine analytische Lösung zu finden, müssen die Randwertprobleme, für die Realisierung des Algorithmus, numerisch gelöst und daher zunächst auf einem geeigneten Gitter diskretisiert werden. Der Lösungsvektor

$$u_i = \begin{pmatrix} u_{\Omega_i} \\ u_{\partial\Omega_i\backslash\Gamma_i} \\ u_{\Gamma_i} \end{pmatrix} \tag{3.7}$$

besteht aus den Werten an den Gitterpunkten in dem entsprechenden Teilgebiet Ω_i, dem echten Rand $\partial\Omega_i\backslash\Gamma_i$ und dem künstlichen Rand Γ_i. Dabei sind die Werte $u_{\partial\Omega_i\backslash\Gamma_i}$ eigentlich bekannt, da sie auf dem echten Rand liegen und somit durch g gegeben sind. Sie werden hier dennoch mitgeführt um die Notation zu vereinfachen. Entsprechend zu u_i sind auch die rechte Seite f_i und die Randwerte g_i definiert.

Die Matrix A_i ist die diskrete Form des Operators $\mathcal{L}$ und bezieht sich auf das Teilgebiet Ω_i. Sie besteht aus drei Teilmatrizen. Die Matrix A_{Ω_i} stellt den Zusammenhang zwischen den inneren Knoten dar, $A_{\partial\Omega_i\backslash\Gamma_i}$ den der inneren Knoten und den Knoten auf dem echten Rand und A_{Γ_i} den der inneren Knoten und den Knoten auf dem künstlichen Rand. Damit kann A_i geschrieben werden als

$$A_i = (A_{\Omega_i}\ A_{\partial\Omega_i\backslash\Gamma_i}\ A_{\Gamma_i}). \tag{3.8}$$

Weiterhin müssen für die Berechnung der diskreten Randwerte auf den künstlichen Rändern Γ_1 und Γ_2 die Werte aus dem jeweils anderen Teilgebiet interpoliert werden. Dazu werden die Interpolationsabbildungen

$$\begin{aligned} \mathcal{I}_{1,2} &: \Omega_1 \to \Gamma_2, \\ \mathcal{I}_{2,1} &: \Omega_2 \to \Gamma_1 \end{aligned}$$

definiert. Damit haben die Randwertprobleme (3.5) und (3.6) die diskrete Form

$$\begin{aligned} A_1 u_1^n &= f_1 && \text{in } \Omega_1, \\ u_{\partial\Omega_1\backslash\Gamma_1}^n &= g_1 && \text{auf } \partial\Omega_1\backslash\Gamma_1, \\ u_{\Gamma_1}^n &= \mathcal{I}_{2,1}(u_{\Omega_2}^{n-1}) && \text{auf } \Gamma_1. \end{aligned} \tag{3.9}$$

$$\begin{aligned} A_2 u_2^n &= f_2 && \text{in } \Omega_2, \\ u_{\partial\Omega_2\backslash\Gamma_2}^n &= g_2 && \text{auf } \partial\Omega_2\backslash\Gamma_2, \\ u_{\Gamma_2}^n &= \mathcal{I}_{1,2}(u_{\Omega_1}^n) && \text{auf } \Gamma_2. \end{aligned} \tag{3.10}$$

Mit dieser Methode kann man einen Algorithmus implementieren, der das Randwertproblem (3.4) löst. In dieser Arbeit soll die Schwarz Methode allerdings als Vorkonditionierer eingesetzt werden. Dazu wird die Vorkonditionierungsmatrix

$$P^{-1} = \begin{pmatrix} A_{\Omega_1}^{-1} & 0 \\ 0 & 0 \end{pmatrix} + \begin{pmatrix} 0 & 0 \\ 0 & A_{\Omega_2}^{-1} \end{pmatrix} \tag{3.11}$$

verwendet. Falls sich die Gebiete Ω_1 und Ω_2 überlappen (andernfalls wäre die Vorkonditionierung mit P äquivalent zum Block Jacobi Verfahren) und zudem die Diskretisierungsgitter in $\Omega_1 \cap \Omega_2$ übereinstimmen, so entstehen in den Matrizen A_{Ω_1} und A_{Ω_2} gemeinsame Einträge und daraus identische, zu lösende Gleichungen. In diesem Fall spricht man auch von der *additiven Schwarz Methode*. In dieser Arbeit wird ein kartesisches Gitter und eine Zerlegung des Gebietes in Blöcke verwendet, weshalb die Diskretisierungsgitter in der Überlappungsregion übereinstimmen. Es soll also eine additive Schwarz Methode als Vorkonditionierer verwendet werden.

3.3 Gebietsabhängige Vorkonditionierung

In Abschnitt 3.1 wurde schon erwähnt, dass sich die Kondition des aus der Poissongleichung entstehenden Gleichungssystems bei einem hohen Dichtesprung zwischen den Phasen verschlechtert. Bisher wurde daher das Gleichungssystem mit einem Vorkonditionierer umgewandelt, um so ein besser konditioniertes Gleichungssystem zu erhalten, welches effizienter gelöst werden kann. Das Problem dabei ist, einen geeigneten Vorkonditionierer zu finden. Wählt man einen starken Vorkonditionierer, so kann das Gleichungssystem zwar in wenigen Iterationen gelöst werden, aber die Berechnung des Vorkonditionierers ist bereits so aufwendig, dass man in der üblicherweise vorliegenden Präasymptotik keine großen Geschwindigkeitsvorteile erzielt. Andererseits ist ein schwacher Vorkonditionierer zwar schnell berechnet, aber die Kondition des Gleichungssystems wird nicht genügend verkleinert um große Geschwindigkeitsvorteile zu erzielen.

Das Ziel dieser Arbeit ist es, dieses Verfahren zu optimieren. Der Grundgedanke dabei ist, dass der Dichtesprung zwischen den beiden Phasen, welcher für die schlechte Kondition verantwortlich ist, nur in einem kleinen Bereich des gesamten Gebietes existiert, nämlich genau an der Phasengrenze. Statt einen Vorkonditionier für das gesamte Gebiet zu verwenden, wird das Gebiet aufgeteilt. In einen ϵ-Bereich Ω_f um die freie Oberfläche Γ_f und in ein Gebiet Ω_1 für den restlichen Bereich (siehe Abbildung 3.2), sodass gilt

$$\Omega = \Omega_1 \cup \Omega_f. \tag{3.12}$$

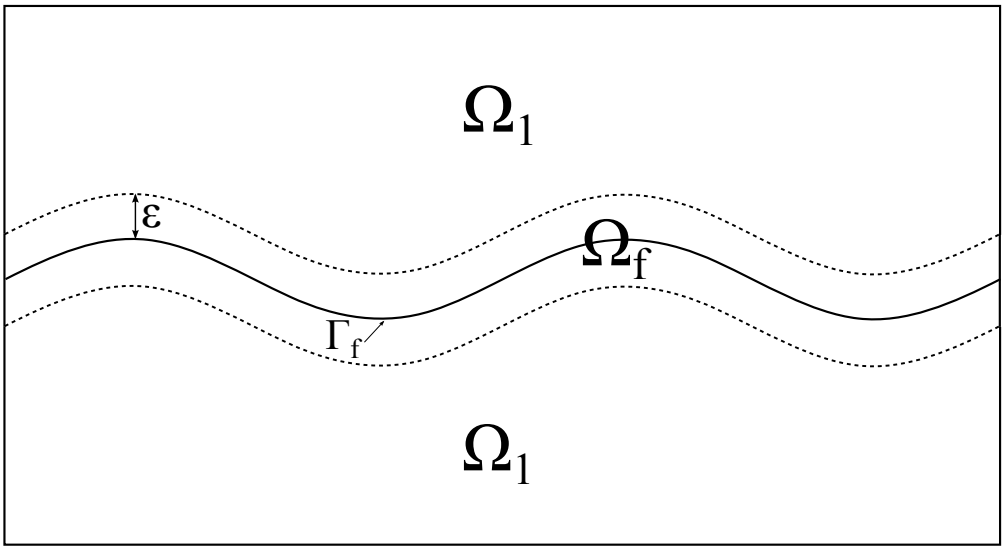

Abbildung 3.2: Aufteilung des Gebietes Ω

Der Dichtesprung ist somit nur in Ω_f enthalten. Auf Ω_1 hingegen wird eine relativ glatte Lösung erwartet, sodass es Sinn macht auf diesen Gebieten kostengünstige Varianten von Vorkonditionierern und Lösern einzusetzen, wie beispielsweise ein Jacobi-BiCGStab-Verfahren. Auf Ω_f erwartet man ein schlecht konditioniertes Problem. Hier sollte eine aufwändigere Vorkonditionierung geschehen. Da die Phasengrenze niederdimensional ist, ist Ω_f jedoch klein im Vergleich zu Ω und daher sollte die aufwändigere Vorkonditionierung die Gesamtkomplexität des Verfahrens nicht wesentlich erhöhen. Um dieses Verfahren zu realisieren, wird eine additive Schwarz Methode zur Vorkonditionierung verwendet. Die Vorkonditionierungsmatrix P hat dabei eine Blockdiagonalstruktur und besteht aus den Blöcken A_{Ω_1} und A_{Ω_f}. Überlappen die Gebiete Ω_1 und Ω_f, so entstehen auch gemeinsame Einträge in den einzelnen Blockmatrizen.

Im Rahmen dieser Arbeit ist dieses Verfahren in das bestehende Strömungssimulationsprogramm NaSt3DGPF des Numerischen Instituts der Universität Bonn implementiert worden. Mithilfe einer Gebietszerlegungsmethode ist es möglich $\Omega = \Omega_1 \cup \Omega_f$ wie oben zu zerlegen. Diese Zerlegung nutzt ein additiver Schwarz Vorkonditionierer und teilt die Druckberechnung in gebietsabhängige Teilberechnungen ein. Für diese gebietsabhängigen Teilberechungen können verschiedene Löser und Vorkonditionierer eingestellt werden. Im folgenden Kapitel ist die konkrete Implementierung beschrieben.

Kapitel 4

Implementierung

Die Druckberechnung für Zweiphasenströmungen soll in dieser Arbeit weiter optimiert werden. Daher wird zu Beginn dieses Kapitels kurz vorgestellt, wie man die kontinuierliche Druck-Poissongleichung für einen Computer berechenbar macht. Dieser Übergang von einem kontinuierlichen zu einem endlichen Modell wird Diskretisierung genannt. Mithilfe dieser Diskretisierung erhält man aus der Poissongleichung ein lineares Gleichungssystem $Ax = b$, welches dann weiter verarbeitet wird.

Im darauf folgenden Abschnitt werden die beiden wichtigen Bibliotheken MPI und PETSc kurz vorgestellt und es wird beschrieben, in welchem Rahmen sie in dieser Arbeit verwendet werden.

Am Ende des Kapitels wird gezeigt, wie die für die Vorkonditionierung mittels der additiven Schwarz Methode notwendige Gebietszerlegung implementiert wurde.

4.1 Diskretisierung der Poissongleichung

Die Hilfsgeschwindigkeiten $\vec{u}^*$ aus Abschnitt 2.2

$$\vec{u}^* = \vec{u}^n + \delta t \left(\frac{\mu(\phi^n)}{\rho(\phi^n)} \Delta \vec{u}^n - \nabla \cdot (\vec{u}^n \otimes \vec{u}^n) + \vec{g} - \frac{\sigma \kappa(\phi^n) \delta(\phi^n) \nabla \phi^n}{\rho(\phi^n)} \right) \tag{4.1}$$

sind im Allgemeinen nicht divergenzfrei, das heißt $\nabla \cdot \vec{u}^* \neq 0$. Um aber der Kontinuitätsgleichung (2.3) zu genügen, muss eine möglichst genaue Druckkorrektur vorgenommen werden. Dazu muss die Poissongleichung mit nicht konstanten Dichtekoeffizienten

$$\nabla \cdot \left(\frac{1}{\rho(\phi^{n+1})} \nabla p^{n+1} \right) = \nabla \cdot \frac{\vec{u}^*}{\delta t}. \tag{4.2}$$

möglichst effizient gelöst werden. Um diese Gleichung exakt zu lösen, muss zu gegebenen Randbedingungen eine analytische Lösung auf dem gesamten Gebiet Ω gefunden werden. Dies ist im Allgemeinen jedoch nicht möglich. Daher wird eine Näherung der Lösung in endlich vielen Punkten des Gebietes bestimmt. Dazu muss das Gebiet zunächst diskretisiert werden, das heißt es muss ein geeignetes Gitter gefunden werden, welches angibt an welchen Stellen des Gebietes eine Lösung approximiert wird. Eine Möglichkeit zur Approximation einer Lösung in den Gitterpunkten ist die Methode der *finiten Differenzen*. Die

Grundidee dabei ist, die Differentialoperatoren, welche in der zu lösenden Differentialgleichung vorkommen, durch Differenzenoperatoren zu ersetzen. Betrachtet man beispielsweise das Randwertproblem

$$\begin{aligned} \Delta u &= f \quad \text{in } \Omega = (0,1)^2, \\ u &= 0 \quad \text{auf } \partial\Omega, \end{aligned} \tag{4.3}$$

muss zuerst ein geeignetes Diskretisierungsgitter gefunden werden. Dieses sehr einfache Gebiet kann man mit einem äquidistanten Gitter der Maschenweite h beschreiben, das heißt man wählt Stützstellen $(x_i, y_j) \in \Omega$ mit

$$\begin{aligned} x_i &= ih \text{ mit } i = 0, \ldots, N \text{ und } h = \frac{1}{N}, \\ y_j &= jh \text{ mit } j = 0, \ldots, N \text{ und } h = \frac{1}{N}. \end{aligned}$$

Es werden Approximationen $u_{i,j}$ von $u(x_i, y_i)$, also genau in den Gitterpunkten, gesucht. Nun muss der kontinuierliche Differentialoperator Δ in etwas endliches umgewandelt werden. Dazu werden die Terme $u(x_{i+1}, y_j)$ und $u(x_{i-1}, y_j)$ mithilfe einer Taylorapproximation umgeformt

$$\begin{aligned} u(x_{i+1}, y_j) &= u(x_i, y_j) + \frac{\partial u}{\partial x}(x_i, y_j)h + \frac{1}{2}\frac{\partial^2 u}{\partial x^2}(x_i, y_j)h^2 + \mathcal{O}(h^3), \\ u(x_{i-1}, y_j) &= u(x_i, y_j) - \frac{\partial u}{\partial x}(x_i, y_j)h + \frac{1}{2}\frac{\partial^2 u}{\partial x^2}(x_i, y_j)h^2 + \mathcal{O}(h^3). \end{aligned}$$

Eine Addition der beiden Gleichungen liefert

$$\frac{\partial^2 u}{\partial x^2} = \frac{1}{h^2}\big(u(x_{i+1}, y_j) - 2(x_i, y_j) + u(x_{i-1}, y_j)\big) + \mathcal{O}(h). \tag{4.4}$$

Analog erhält man

$$\frac{\partial^2 u}{\partial y^2} = \frac{1}{h^2}\big(u(x_i, y_{j+1}) - 2(x_i, y_j) + u(x_i, y_{j-1})\big) + \mathcal{O}(h). \tag{4.5}$$

Also kann Δu wie folgt approximiert werden

$$\Delta u(x_i, y_j) \approx \frac{1}{h^2}\big(u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1} - 4u_{i,j}\big). \tag{4.6}$$

Da $\Delta u(x_i, y_j) = f(x_i, y_j) =: f_{i,j}$ gelten soll, müssen für alle $i, j = 0 \ldots N$ die Gleichungen

$$\begin{aligned} \frac{1}{h^2}\big(u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1} - 4u_{i,j}\big) &= f_{i,j} && \text{für } (x_i, y_j) \in \Omega, \\ u_{i,j} &= 0 && \text{für } (x_i, y_j) \in \partial\Omega \end{aligned}$$

gelöst werden. Das Lösen der Gleichungen ist äquivalent zum Lösen des Gleichungssystems

$$AU = F, \tag{4.7}$$

mit

$$U = \begin{pmatrix} u_{0,0} \\ u_{0,1} \\ \vdots \\ u_{0,N} \\ u_{1,0} \\ u_{1,1} \\ \vdots \\ u_{1,N} \\ \vdots \\ u_{N,0} \\ u_{N,1} \\ \vdots \\ u_{N,N} \end{pmatrix}, \; F = \begin{pmatrix} f_{0,0} \\ f_{0,1} \\ \vdots \\ f_{0,N} \\ f_{1,0} \\ f_{1,1} \\ \vdots \\ f_{1,N} \\ \vdots \\ f_{N,0} \\ f_{N,1} \\ \vdots \\ f_{N,N} \end{pmatrix} \tag{4.8}$$

und

$$A = \frac{1}{h^2} \begin{pmatrix} A_0 & I & & & 0 \\ I & A_0 & I & & \\ & \ddots & \ddots & \ddots & \\ & & I & A_0 & I \\ 0 & & & I & A_0 \end{pmatrix}, \; A_0 = \begin{pmatrix} -4 & 1 & & & 0 \\ 1 & -4 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & -4 & 1 \\ 0 & & & 1 & -4 \end{pmatrix},$$

wobei I die $(N+1) \times (N+1)$ Einheitsmatrix bezeichnet. Dieses Gleichungssystem muss dann möglichst effizient gelöst werden und man erhält die gewünschte Approximation der Lösung in den Gitterpunkten (x_i, y_j). Der Term

$$\begin{bmatrix} & 1 & \\ 1 & -4 & 1 \\ & 1 & \end{bmatrix} \tag{4.9}$$

aus der Matrix A_0 wird auch *Fünf-Punkte-Differenzen-Stern* genannt. Er beschreibt die Anteile der approximierten Lösung im zentralen Gitterpunkt sowie in dem umliegenden östlichen, südlichen, westlichen und nördlichen Gitterpunkt.

Im Fall der zweiphasigen Strömungssimulation soll das dreidimensionale Gebiet $\Omega = [0, \text{xlength}] \times [0, \text{ylength}] \times [0, \text{zlength}]$ behandelt werden. Auch hier wird wieder ein geeignetes Gitter benötigt. Da Ω quaderförmig ist, kann man wieder ein äquidistantes Gitter verwenden. Allerdings werden die gesuchten Größen $\vec{u}$ und p nicht in denselben Gitterpunkten approximiert, sondern in versetzten Punkten. So werden die Geschwindigkeiten $\vec{u} = (u, v, w)^T$ immer in den Mittelpunkten der Zellseitenflächen und der Druck p immer im Zellmittelpunkt approximiert (Siehe Abbildung 4.1).

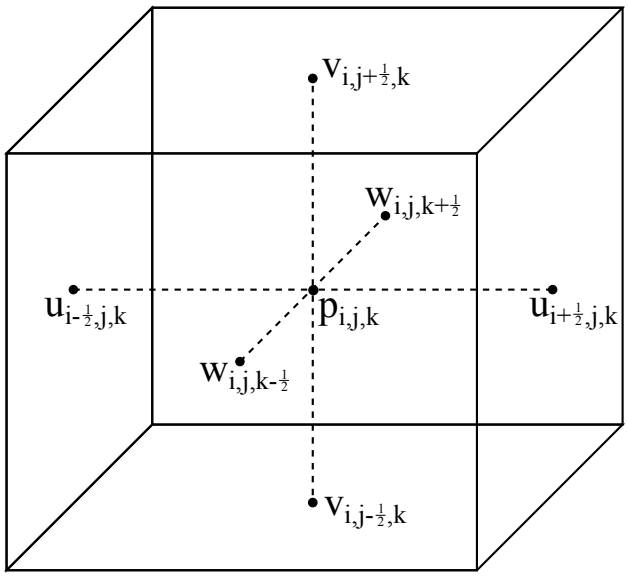

Abbildung 4.1: Eine Zelle aus dem versetzten Diskretisierungsgitter

In diesem Fall spricht man auch von einem *versetzten Diskretisierungsgitter*. Nun müssen die Differenzialoperatoren aus der Poissongleichung (4.2) durch endliche Differenzenoperatoren ersetzt werden. Die rechte Seite wird dabei mit zentralen Differenzen ausgewertet

$$[\nabla \cdot \vec{u}^*]_{i,j,k} = \frac{u^*_{i+\frac{1}{2},j,k} - u^*_{i-\frac{1}{2},j,k}}{\delta x} + \frac{v^*_{i,j+\frac{1}{2},k} - v^*_{i,j-\frac{1}{2},k}}{\delta y} + \frac{w^*_{i,j,k+\frac{1}{2}} - w^*_{i,j,k-\frac{1}{2}}}{\delta z}. \tag{4.10}$$

Dabei wird also die Approximation

$$\frac{\partial u^*_{i,j,k}}{\partial x} = \lim_{\delta x \to 0} \frac{u^*_{i+\frac{1}{2},j,k} - u^*_{i-\frac{1}{2},j,k}}{\delta x} \approx \frac{u^*_{i+\frac{1}{2},j,k} - u^*_{i-\frac{1}{2},j,k}}{\delta x} \tag{4.11}$$

verwendet, indem die Grenzwertbildung weggelassen wird. Je feiner die Maschenweite δx wird, desto genauer wird auch die Approximation. In y- und z-Richtung wird analog approximiert.

Die linke Seite der Poissongleichung wird mit einem Sieben-Punkte-Stern berechnet. Anders als bei dem Fünf-Punkte-Stern im obigen Beispiel müssen hier neben dem zentralen, östlichen, südlichen, westlichen und nördlichen Eintrag auch der vordere und hintere Eintrag mit berücksichtigt werden, da das Modell in drei Raumdimensionen betrachtet wird. Man erhält

$$\begin{aligned}
\left[\nabla \cdot \frac{1}{\rho(\phi)} \nabla p^{n+1}\right]_{i,j,k} &= \frac{1}{(\delta x)^2} \left(\left(\frac{p^{n+1}_{i+1,j,k} - p^{n+1}_{i,j,k}}{\rho(\phi_{i+\frac{1}{2},j,k})} \right) - \left(\frac{p^{n+1}_{i,j,k} - p^{n+1}_{i-1,j,k}}{\rho(\phi_{i-\frac{1}{2},j,k})} \right) \right) \\
&+ \frac{1}{(\delta y)^2} \left(\left(\frac{p^{n+1}_{i,j+1,k} - p^{n+1}_{i,j,k}}{\rho(\phi_{i,j+\frac{1}{2},k})} \right) - \left(\frac{p^{n+1}_{i,j,k} - p^{n+1}_{i,j-1,k}}{\rho(\phi_{i,j-\frac{1}{2},k})} \right) \right) \\
&+ \frac{1}{(\delta z)^2} \left(\left(\frac{p^{n+1}_{i,j,k+1} - p^{n+1}_{i,j,k}}{\rho(\phi_{i,j,k+\frac{1}{2}})} \right) - \left(\frac{p^{n+1}_{i,j,k} - p^{n+1}_{i,j,k-1}}{\rho(\phi_{i,j,k-\frac{1}{2}})} \right) \right),
\end{aligned} \tag{4.12}$$

wobei die Dichte zuvor an den passenden Stellen durch eine Lagrange-Interpolation der Level-Set-Werte bestimmt wird. Eine genaue Beschreibung dieser Interpolation ist in [2], Seite 61, gegeben.

Die Poissongleichung ist nun diskretisiert und in ein System von Differenzengleichungen umgewandelt worden, welche man zu einem linearen Gleichungssystem $Ax = b$ zusammenfassen kann. Dieses soll dann für eine gute Druckkorrektur möglichst effizient und genau gelöst werden.

4.2 MPI und PETSc

Die in Abschnitt 2.4 beschriebene Parallelisierung ist in dem zweiphasigen Navier-Stokes-Löser NaSt3DGPF des Numerischen Instituts der Universität Bonn mithilfe des *Message Passing Interface* (MPI) [14] realisiert worden. Dabei handelt es sich um eine Kommunikationsbibliothek, die sich mittlerweile zu einem Standard bei parallelen Berechnungen auf verteilten Systemen entwickelt hat. Ein verteiltes System ist ein Netzwerk von unabhängigen Computern. Da diese über keinen gemeinsamen Speicher verfügen, müssen Daten geeignet kommuniziert werden. Mit MPI ist es möglich eine Simulation auf vielen Prozessen zu starten. Zwischen den Prozessen können Daten versendet und empfangen werden.

Zur Implementierung der gebietsabhängigen Vorkonditionierung wurde das *Portable Extensible Toolkit for Scientific Computation* (PETSc) [15] verwendet. Hierbei handelt es sich um eine mächtige Bibliothek, welche eine Hilfestellung zur Lösung partieller Differentialgleichungen liefert. So beinhaltet PETSc beispielsweise sehr effizient implementierte Krylov-Unterraum-Verfahren zum Lösen von linearen Gleichungssystemen. Dabei ist es auch möglich einen Vorkonditionierer zu verwenden. In NaSt3DGPF ist bereits eine PETSc Variante zum Lösen der Druck-Poissongleichung mit einem Krylov-Unterraum-Verfahren implementiert. In dieser Arbeit soll die additive Schwarz Methode als Vorkonditionierer in Kombination mit einem Krylov-Unterraum-Verfahren verwendet werden. Dies geschieht mit dem Befehl

```
PCSetType(prec,PCASM).
```

Dabei ist `prec` ein PETSc PC Objekt und `PCASM` steht für *Preconditioner Additive Schwarz Method*. Nun werden mittels

```
PCASMSetLocalSubdomains(prec,2,is)
```

die zwei Teilgebiete auf dem aktuellen Prozess anhand des PETSc Arrays `is` von Indexmengen festgelegt. Dabei besteht `is` aus zwei PETSc Indexmengen und diese beinhalten Indizes von den Gebietszellen außerhalb und innerhalb einer ϵ-Umgebung der freien Oberfläche. Wie die Gebietszerlegung und die Aufteilung der Indexmengen funktioniert, wird in Abschnitt 4.3 erklärt. Nach einem `KSPSetup`, welches das Krylov-Unterraum-Verfahren mit Vorkonditionierer `PCASM` konfiguriert, kann man mittels

```
PCASMGetSubKSP(prec,&nlocal,&first,&subksp)
```

Zugriff auf die Krylov-Unterraum-Verfahren auf den Teilgebieten erhalten. Dabei wird in `nlocal` die Anzahl der lokalen Teilgebiete und in `first` die Nummer des ersten lokalen Teilgebietes geschrieben. Das PETSc KSP Objekt `subksp` wird für den Krylovraum-Kontext auf den Teilgebieten benötigt. Für die Unterverfahren ist es wieder möglich ein

Krylov-Unterraum-Verfahren und einen Vorkonditionierer zu bestimmen. So können auf den zwei Teilgebieten verschiedene Löser und Vorkonditionierer verwendet werden.

4.3 Gebietszerlegung mithilfe der Level-Set Funktion

In Abschnitt 3.3 wurde die gebietsabhängige Vorkonditionierung beschrieben, welche im Rahmen dieser Arbeit implementiert wurde. Dazu wird die Gebietszerlegung

$$\Omega = \Omega_1 \cup \Omega_f \tag{4.13}$$

benötigt. Um diese zu erhalten, werden zwei PETSc Indexmengen erstellt, welche später genau die zwei Teilgebiete Ω_1 und Ω_f charakterisieren sollen. Dann beginnt eine Schleife über alle Koordinaten des lokalen Gebiets. Rechnet man sequenziell, so sind hier alle Koordinaten des gesamten Gebiets gemeint. Innerhalb der Schleife wird nun abgefragt, ob es sich bei der aktuellen Zelle um eine Fluidzelle handelt, das heißt eine Zelle in der Strömungsberechnungen durchgeführt werden. Andernfalls handelt es sich um eine Hinderniszelle, das heißt eine Zelle in der keine Berechnungen stattfinden. Diese muss nicht weiter betrachtet werden. Ist es eine Fluidzelle, so wird mithilfe der Level-Set-Funktion ϕ abgefragt, ob die aktuelle Zelle weiter von der freien Oberfläche entfernt ist, als eine vorgegebene Distanz ϵ. Ist dies der Fall, so wird diese Zelle der Indexmenge 1, andernfalls der Indexmenge 2 zugeordnet. In Algorithmus 2 ist die Aufteilung der Gebiete noch

Algorithmus 2 Gebietszerlegung

```
Setze is1 := 0, is2 := 0, index = 0
Setze den Radius der ϵ-Umgebung um die freie Oberfläche
for k = kstart; k < kstart + klength; k++ do
  for j = jstart; j < jstart + jlength; j++ do
    for i = istart; i < istart + ilength; i++ do
      if Fluid flag[i][j][k] then
        if |φ[i][j][k]| > ϵ then
          indices1[is1] =index
          is1++
        else
          indices2[is2] =index
          is2++
        end if
        index++
      end if
    end for
  end for
end for
```

einmal zusammengefasst, wobei `flag` ein Feld ist, welches angibt, ob die Zelle eine Fluid oder Hinderniszelle ist. Aus den Feldern `indices1` und `indices2` werden daraufhin die PETSc Indexmengen erstellt. Mit diesen wird die gebietsabhängige Vorkonditionierung durchgeführt.

Kapitel 5

Numerische Resultate

In diesem Kapitel geht es um die numerischen Ergebnisse der im Rahmen dieser Arbeit durchgeführten Experimente zur gebietsabhängigen Vorkonditionierung. In Abschnitt 5.1 wird die implementierte Gebietszerlegung anhand von zwei Beispielen visualisiert. Abschnitt 5.2 stellt die Konditionen der Teilmatrizen A_{Ω_f} und A_{Ω_1} dar. Dabei wird deutlich das die Matrix A_{Ω_f}, welche die hohen Koeffizientensprünge enthält wesentlich schlechter konditioniert ist als A_{Ω_1}. Daraufhin wird in Abschnitt 5.3 die Konvergenz des implementierten Verfahrens diskutiert. Es werden dabei verschiedene Variationen von Dichtesprüngen und Bandbreiten von Ω_f und deren Einfluss auf die Laufzeit betrachtet. Abschließend wird in 5.4 die Laufzeit der gebietsabhängigen Vorkonditionierung analysiert und mit der bisher verwendeten Vorkonditionierung mit dem algebraischen Mahrgitter Verfahren verglichen. Dabei konnten für einige Testrechnungen deutliche Geschwindigkeitsvorteile erzielt werden.

5.1 Verifikation der Gebietszerlegung

Als Grundlage für die Vorkonditionierung mit der additiven Schwarz Methode dient die in 4.3 beschriebene Gebietszerlegung, welche Ω in den Bereich um die freie Oberläche Ω_f und in den restlichen Bereich Ω_1 aufteilt. Da sich die freie Oberfläche mit der Zeit ändert, muss die Gebietszerlegung in jedem Zeitschritt neu berechnet werden. Somit ergibt sich ein zeitlicher Verlauf der beiden Gebiete Ω_1 und Ω_f, welcher hier durch zwei Beispiele verifiziert wird.

Das erste Beispiel ist eine Simulation einer aufsteigenden Luftblase im Wasser. Das Grundgebiet ist $\Omega = [0,1]\times[0,2]\times[0,1]$. Diese Gebiet wird mit einer Auflösung von $< 32, 64, 32 >$ äquidistant diskretisiert, das heißt in x-Richtung werden 32, in y-Richtung 64 und in z-Richtung 32 Zellen mit jeweils gleicher Seitenlänge verwendet. Die Level-Set-Funktion wird so initialisiert, dass die Nullniveaumenge gleich der um den Punkt $(0.5, 0.5, 0.5)$ zentrierten Luftblase mit Radius $r = 0.25$ ist. Weiterhin ist die Dichte der Wasserphase $\rho_l = 1000$ und die Dichte der Luftphase $\rho_g = 1$. Abbildung 5.1 zeigt diese Simulation zu sechs Zeitpunkten zwischen null und einer Sekunde. Dabei wird ein Querschnitt der dreidimensionalen Simulation betrachtet. Das rote Gebiet ist der Bereich Ω_f um die freie Oberfläche, das blaue Gebiet ist der restliche Bereich Ω_1. Man kann gut erkennen, dass die simulierten Ergebnisse mit den physikalisch zu erwartenden Vorgängen übereinstimmen.

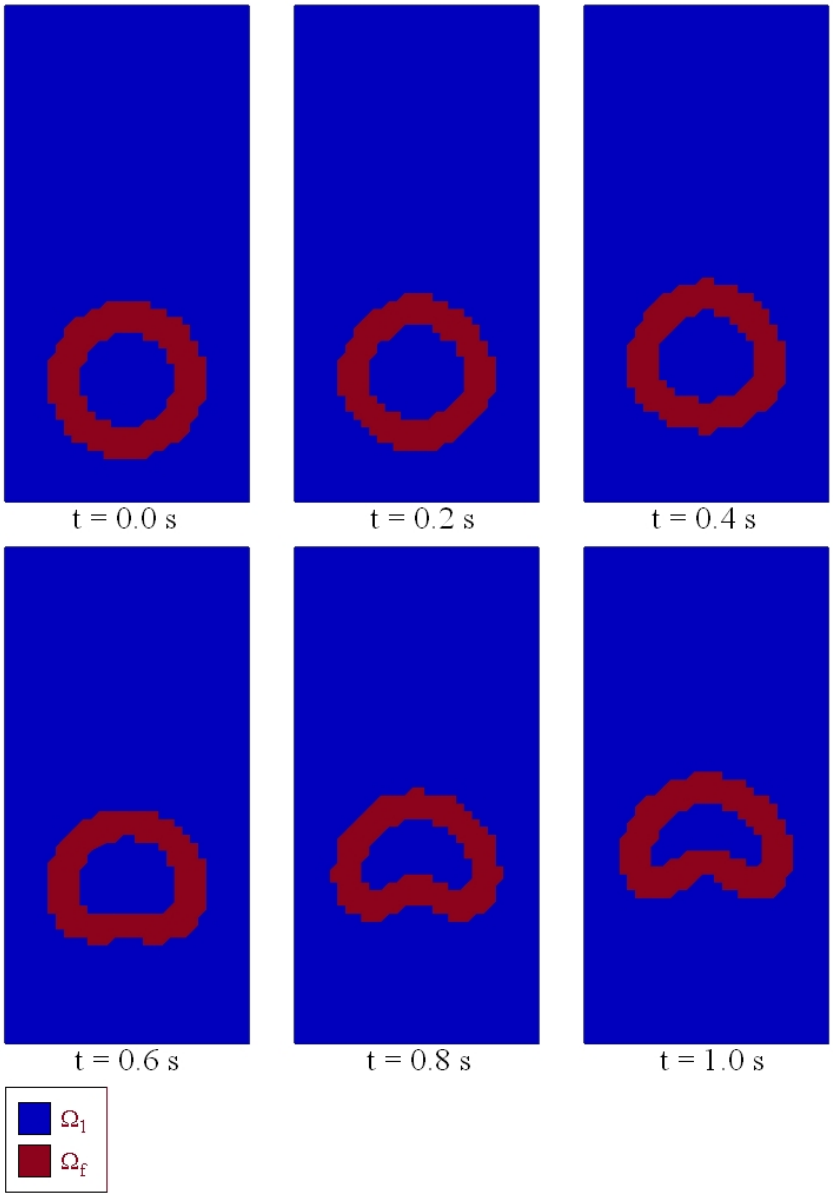

Abbildung 5.1: Gebietszerlegung bei einer aufsteigenden Luftblase im Wasser

Bei dem zweiten Beispiel handelt es sich um eine simulierte Strömung um ein Hindernis (siehe Abbildung 1.3). Dabei ist das Grundgebiet $\Omega = [0, 0.6] \times [0, 0.25] \times [0, 0.4]$ äquidistant mit einer Auflösung von $< 60, 25, 40 >$ diskretisiert. Das Hindernis ist ein quaderförmiger Pfeiler und hat die Größe $(B = 5) \times (H = 22) \times (T = 5)$. Stellt man sich Ω als einen Kanal vor, so ist dieser bis zu einer Höhe von 5 mit Wasser gefüllt. Darüber befindet sich Luft. Am linken Rand des Kanals ist eine Einströmgeschwindigkeit des Wassers von 1 gesetzt worden und am rechten Rand sind Ausströmbedingungen gesetzt worden.

Abbildung 5.2 zeigt diese Simulation zu sechs Zeitpunkten zwischen null und einer Sekun-

de. Hier gibt es ein schwarzes Gebiet, welches das Hindernis darstellt. Die Strömung fließt vom linken Gebietsrand zum rechten Gebietsrand und muss um das Hindernis herum. Auch hier stimmen die Simulationen mit den physikalischen Erwartungen überein. Der Querschnitt ist mittig durch das Hindernis.

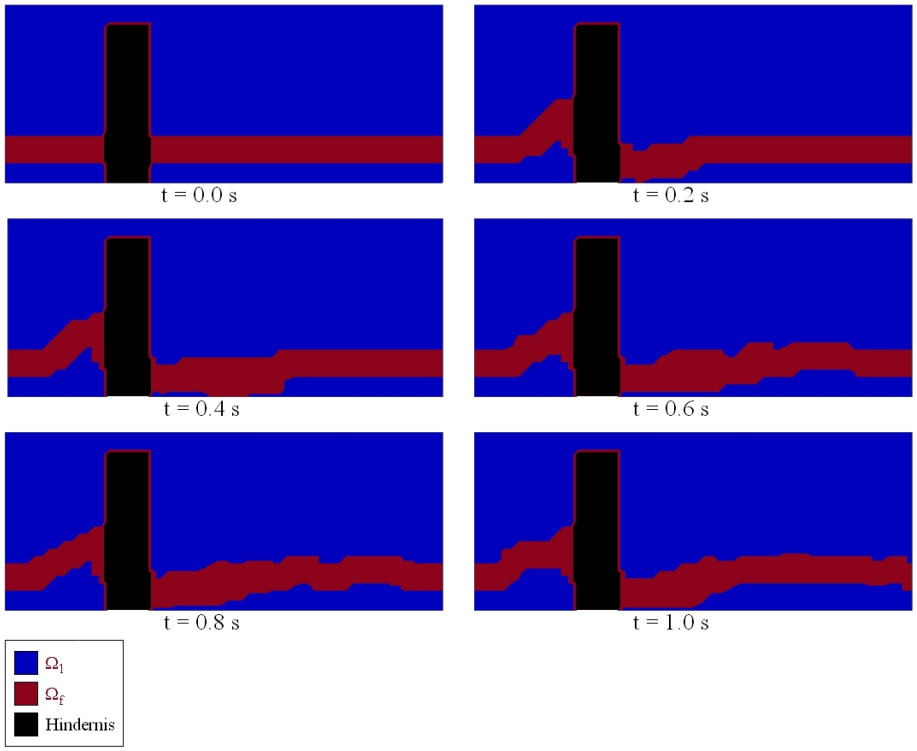

Abbildung 5.2: Gebietszerlegung einer Strömung um ein Hindernis

5.2 Kondition

Um die gesuchten Druckwerte aus den Navier-Stokes-Gleichungen zu erhalten muss die Poissongleichung

$$\nabla \cdot \left(\frac{1}{\rho(\phi^n)} \nabla p^n \right) = rhs \tag{5.1}$$

und damit das nach Diskretisierung entstehende lineare Gleichungssystem $Ax = b$ gelöst werden. Wegen des Terms $\frac{1}{\rho(\phi)}$ in der Poissongleichung geht ein Dichtesprung an der Phasengrenze also unmittelbar mit in die Druckberechnung ein. Damit ist auch die Kondition der Systemmatrix A abhängig von dem Dichteunterschied zwischen den beiden Phasen. Für symmetrische, positiv definite Matrizen A ist die Kondition gegeben durch

$$\kappa(A) = \frac{\lambda_{max}(A)}{\lambda_{min}(A)}, \tag{5.2}$$

wobei $\lambda_{max}(A)$ und $\lambda_{min}(A)$ den größten beziehungsweise den kleinsten Eigenwert von A bezeichnen.

Zum Bestimmen der Kondition auf den Gebieten Ω_f und Ω_1 wurde eine Eigenwertberechnung der Teilmatrizen A_{Ω_f} und A_{Ω_1} mithilfe von PETSc implementiert. Da die explizite Berechnung von Eigenwerten sehr aufwändg ist, wurden die Konditionen auf einem Gitter mit einer geringen Auflösung von $< 30, 12, 20 >$ und einem Gebiet $\Omega = [0, 0.6] \times [0, 0.25] \times [0, 0.4]$ berechnet. Die Maschenweite ist $h = 0.02$ und die Bandbreite des Bereichs um die freie Oberfläche beträgt $\epsilon = 4 \cdot h$. Es wurde eine Testrechnung einer Strömung um ein Hindernis durchgeführt, wobei man die folgenden Konditionen erhält

Dichtesprung 1 : 10		
$\lambda_{min}(A_{\Omega_f}) = 75.18$	$\lambda_{max}(A_{\Omega_f}) = 27233.51$	$\kappa(A_{\Omega_f}) = 362.24$
$\lambda_{min}(A_{\Omega_1}) = 133.21$	$\lambda_{max}(A_{\Omega_1}) = 28577.03$	$\kappa(A_{\Omega_1}) = 214.52$
Dichtesprung 1 : 1000		
$\lambda_{min}(A_{\Omega_f}) = 1.21$	$\lambda_{max}(A_{\Omega_f}) = 26046.02$	$\kappa(A_{\Omega_f}) = 21494.47$
$\lambda_{min}(A_{\Omega_1}) = 133.21$	$\lambda_{max}(A_{\Omega_1}) = 28577.03$	$\kappa(A_{\Omega_1}) = 214.52$
Dichtesprung 1 : 100000		
$\lambda_{min}(A_{\Omega_f}) = 0.01$	$\lambda_{max}(A_{\Omega_f}) = 25922.17$	$\kappa(A_{\Omega_f}) = 2128956.64$
$\lambda_{min}(A_{\Omega_1}) = 133.21$	$\lambda_{max}(A_{\Omega_1}) = 28577.03$	$\kappa(A_{\Omega_1}) = 214.52$

Man sieht deutlich, dass die Matrix A_{Ω_f} in allen Fällen schlechter konditioniert ist als A_{Ω_1}. Da Ω_f den Dichtesprung enthält ist dieses Verhalten auch zu erwarten. Weiter sieht man, dass mit wachsendem Dichtesprung die Kondition von A_{Ω_f} zunehmend schlechter wird, wohingegen die Kondition von A_{Ω_1} konstant bleibt. Damit wird bestätigt, dass der Bereich um die freie Oberfläche mit einem besonders robusten und starken Vorkonditionierer behandelt werden sollte.

5.3 Konvergenzanalyse

In diesem Abschnitt wird das Konvergenzverhalten des implementierten Verfahrens untersucht. Dabei handelt es sich um eine Vorkonditionierung mit der additiven Schwarz Methode ohne Überlapp. Auf den Teilgebieten Ω_f und Ω_1 wurden keine Lösungsverfahren, sondern nur Vorkonditionierer eingesetzt, wobei auf Ω_f ein algebraisches Mehrgitterverfahren (AMG [7]) und auf Ω_1 ein ILU Verfahren zur Vorkonditionierung verwendet wurde. Als Gesamtlösungsverfahren wurde ein BiCGSTAB Verfahren verwendet. Ist $Ax = b$ das nach Diskretisierung aus der Poissongleichung entstehende lineare Gleichungssystem, so wird die Residuumsnorm

$$||Ax^{it} - b||_{l^2}, \tag{5.3}$$

mit

$$||\vec{x}||_{l^2} = \sqrt{\sum_i x_i^2} \tag{5.4}$$

betrachtet.

Bei allen Testrechnungen wurden homogene Neumannrandbedingungen für den Druck vorgegeben. Um die Eindeutigkeit der Lösung sicherzustellen, wurden zusätzlich an einer

Stelle des Gebiets Ω Dirichlet Randbedingungen vorgegeben. Dies wurde implementiert, indem die erste Zeile und Spalte der Systemmatrix A auf null gesetzt wurde, außer dem linken oberen Eintrag, welcher auf eins gesetzt wurde. Weiterhin wurde die rechte Seite des Gleichungssystem zur Konvergenzanalyse auf null gesetzt. Die Matrix A ist symmetrisch und die Eigenwerte von A wurden untersucht. Diese sind alle positiv und reell. Damit ist A auch positiv definit. Die Konvergenz des Verfahrens wird anhand des in Abschnitt 5.1 beschriebenen Beispiels von einer Strömung um ein Hindernis für den ersten Zeitschritt verifiziert. Dabei werden zunächst die folgenden Einstellungen getroffen

Gebiet	$[0, 0.6] \times [0, 0.25] \times [0, 0.4]$
Auflösung	$< 60, 25, 40 >$
Maschenweite	0.01
Dichtesprung	1 : 1000

Variiert wird die Bandbreite ϵ des Bereichs um die freie Oberfläche. Abbildung 5.3 zeigt die Norm des Residuums (5.3) im zeitlichen Verlauf für drei verschiedene Bandbreiten ϵ. Die Zeitmessung startet sobald das Lösen der Poissongleichung beginnt.

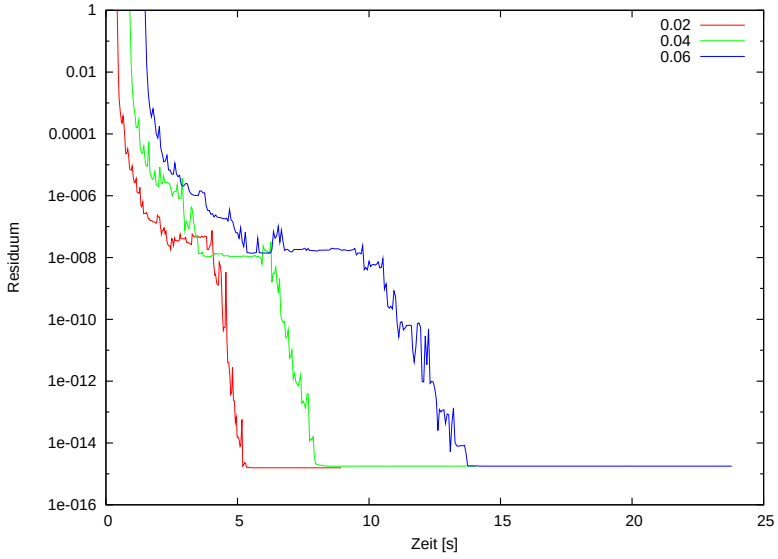

Abbildung 5.3: Konvergenz

Die Residuumsnorm wurde dabei durch Division der Residuumsnorm im ersten Iterationsschritt auf eins normiert. Man sieht deutlich, dass sich die Laufzeiten für wachsendes ϵ erhöhen. Dies liegt daran, dass mit wachsendem ϵ das Gebiet Ω_f größer wird und damit das AMG Verfahren auf einem größeren Bereich eine Gitterhierarchie erstellen muss, was zu den längeren Laufzeiten führt.

Bei der nächsten Testrechnung wurde die Bandbreite auf $\epsilon = 0.02$ festgelegt und die Dichteunterschiede der beiden Phasen wurden variiert. Alle anderen Einstellungen wurden beibehalten.

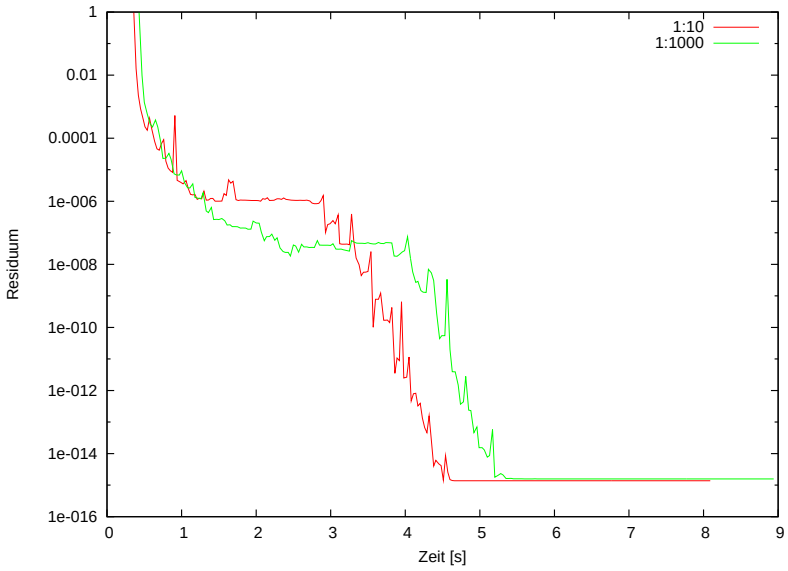

Abbildung 5.4: Konvergenz

In Abbildung 5.4 kann man erkennen, dass die Konvergenzgeschwindigkeit auch von dem Dichtesprung an der Phasengrenze abhängt. Für einen hohen Dichtesprung von $1:1000$ wird mehr Zeit benötigt, als für einen niedrigen Sprung von $1:10$, was auch mit den oben angegebenen Konditionen übereinstimmt.

5.4 Laufzeitanalyse

In diesem Abschnitt wird die Laufzeit der implementierten gebietsabhängigen Vorkonditionierung im Vergleich zu den bisherigen Vorkonditionierungsverfahren untersucht. Zu beachten ist, dass bei der Diskretisierung der Poissongleichung in Abschnitt 4.1 bereits ein Diskretisierungsfehler der Ordnung $\mathcal{O}(h^2)$ gemacht wird, wobei h die Maschenweite des Gitters ist. Bei einem Gebiet $\Omega = [0,1] \times [0,2] \times [0,1]$ und einer Auflösung von $< 32, 64, 32 >$ ist die Maschenweite $h = \frac{1}{32}$. Es wird also ein Diskretisierungsfehler der Größenordnung

$$h^2 = \frac{1}{32^2} = 9.765625 \cdot 10^{-4} \tag{5.5}$$

gemacht. Für diese Auflösung ist es nur sinnvoll das Residuum bis zur Größenordnung 10^{-4} zu betrachten, da sowieso keine höhere Exaktheit der Lösung erreicht werden kann. Dabei ist zu beachten, dass zwar ein Diskrtisierungsfehler der Ordnung $\mathcal{O}(h^2)$ gemacht wird, aber dass dieser abhängt von einer Konstanten, welche wiederum von der Bandbreite von Ω_f und dem Dichtesprung der beiden Phasen abhängt. Daher kann es in realen Anwendungen zu Schwankungen kommen, sodass das Residuum gegebenenfalls auch bis zu kleineren Größenordnungen betrachtet werden sollte.

Für die gebietsabhängige Vorkonditionierung wurde auf Ω_f ein AMG Verfahren und auf Ω_1 ein ILU Verfahren, sowie ein Jacobi Verfahren zur Vorkonditionierung eingesetzt. Im Vergleich dazu steht die Druckberechnung mit nur einem AMG Verfahren als Vorkonditionierung. Die rechte Seite des Gleichungssystems wurde wieder auf null gesetzt und das Residuum wie oben auf eins normiert. Für ein Testbeispiel mit den Parametern

Gebiet	$[0, 0.6] \times [0, 0.25] \times [0, 0.4]$
Auflösung	$< 60, 25, 40 >$
Maschenweite	0.01
Dichtesprung	$1 : 1000$
Bandbreite	0.02

ergeben sich die folgenden Kovergenzverläufe, wobei die Zeitmessung wieder ab dem Beginn des Lösens der Poissongleichung startet.

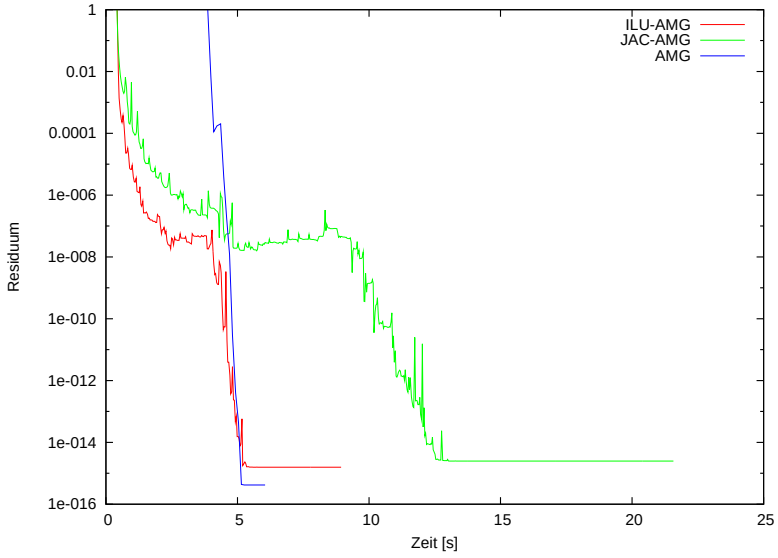

Abbildung 5.5: Konvergenz

Man sieht, dass die Itarationszyklen der Verfahren mit der gebietsabhängigen Vorkonditionierung früher starten. Dies liegt daran, dass die Gitterhierarchie des AMG Verfahrens hier nur auf dem kleinen Bereich Ω_f erstellt werden muss. Im Gegensatz dazu muss das Verfahren mit der reinen AMG Vorkonditionierung auf dem gesamten Gebiet eine Gitterhierarchie erstellen, wesshalb hier der Start verzögert ist. Weiterhin sieht man zwei Schnittpunkte der Residuumsverläufe. Bei ungefähr 4.7 Sekunden schneiden sich die Residuumsverläufe des AMG- und des Jacobi-AMG Verfahrens bei der Residuumsgrößenordnung von 10^{-8}. Das bedeutet, dass das Jacobi-AMG Verfahren Laufzeitersparnisse liefert, solange man nur bis zu einer Toleranz von 10^{-8} löst. Nach den obigen Überlegungen ist es wegen des Diskretisierungsfehlers der Druck-Poissongleichung nur sinnvoll bis zu einer Größenordnung von h^2 zu lösen. Bei dieser Testrechnung ist $h = 0.01$ und damit $h^2 = 10^{-4}$.

Löst man nur bis zu einer Toleranz von 10^{-4}, so benötigt das Jacobi-AMG Verfahren eine Laufzeit von 0.84 Sekunden und das AMG Verfahren eine Laufzeit von 4.1 Sekunden. Damit ergibt sich ein Geschwindigkeitsvorteil (Speedup) vom Faktor $\frac{4.1}{0.84} \approx 4,88$.
Der Schnittpunkt des Residdumsverlaufs der ILU-AMG Variante mit dem AMG Verfahren liegt sogar noch tiefer bei etwa 5 Sekunden und einer Residuumsgröße von 10^{-14}. Das heißt beim Lösen bis zu dieser Größenordnung ist die ILU-AMG Variante schneller als das AMG Verfahren. Löst man nur bis zu einer Toleranz von 10^{-4}, so ergibt sich ein Speedup-Faktor von $\frac{4.1}{0.54} \approx 7.59$.
In der nächsten Testrechnung wurde die Auflösung des Gebietes erhöht und die Bandbreite ϵ angepasst. Mit den Parametern

Gebiet	$[0, 0.6] \times [0, 0.25] \times [0, 0.4]$
Auflösung	$< 120, 50, 80 >$
Maschenweite	0.005
Dichtesprung	$1 : 1000$
Bandbreite	0.01

ergeben sich die folgenden Kovergenzverläufe.

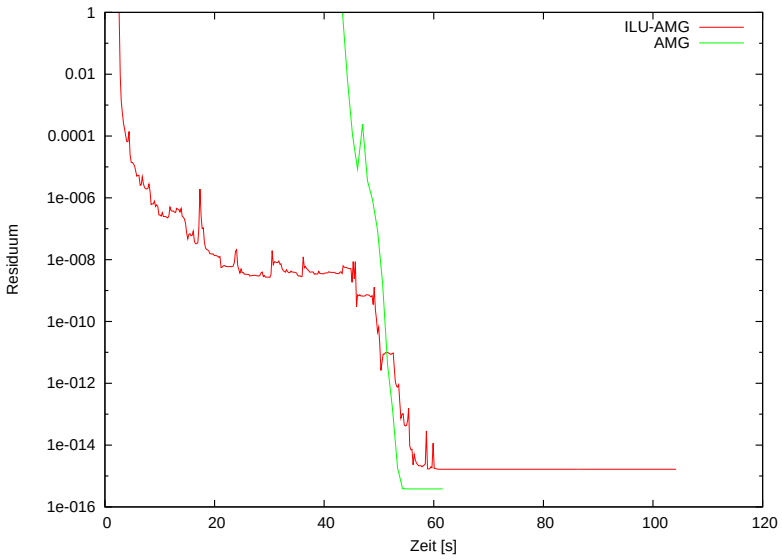

Abbildung 5.6: Konvergenz

Die Setup-Phase des AMG Verfahrens benötigt wieder eine längere Laufzeit als die des ILU-AMG Verfahrens, wesswegen die Residuumsreduktion des AMG-Verfahrens später beginnt. Die Maschenweite bei dieser Auflösung beträgt $h = 0.005$. Löst man wegen des Diskretisierungsfehlers nur bis zur Ordnung $h^2 = 25 \cdot 10^{-5}$ so ergibt sich ein Speedup-Faktor von $\frac{46.09}{4.00} \approx 11.52$. Für höhere Auflösungen ergibt sich also wegen der aufwändigen Setup-Phase des AMG Verfahrens ein noch höherer Geschwindigkeitsvorteil.

Kapitel 6

Zusammenfassung und Ausblick

Zusammenfassung

Im Rahmen dieser Arbeit wurde die Druckberechnung des zweiphasigen Navier-Stokes-Lösers NaSt3DGPF mithilfe einer gebietsabhängigen Vorkonditionierung optimiert. Die Systemmatrix des nach Diskretisierung der Poissongleichung entstandenen linearen Gleichungssystems weist eine schlechte Kondition auf, welche von dem Dichtesprung an der freien Oberfläche abhängt. Die Grundidee war es, das Gebiet in einen Bereich um die freie Oberfläche Ω_f und in den restlichen Bereich Ω_1 aufzuteilen und mithilfe einer additiven Schwarz Methode die Druckberechnung auf beiden Teilgebieten durchzuführen. Die Konditionen der Submatrizen A_{Ω_f} und A_{Ω_1} wurden untersucht, wobei die Kondition von A_{Ω_f} wesentlich schlechter als die von A_{Ω_1} war. Daher sollte auf Ω_f ein robustes AMG Verfahren zur Vorkonditionierung verwendet werden. Dieses hat zwar eine aufwändige Setup-Phase, aber da die freie Oberfläche niederdimensional ist und damit Ω_f im Vergleich zu Ω klein ist, sollte sich die Gesamtkomplexität nicht erhöhen. Auf Ω_1 hingegen wurden kostengünstige Varianten zur Vorkonditionierung verwendet, wie das Jacobi oder ILU Verfahren.

Es wurden Konvergenzanalysen mit verschiedenen Einstellungen durchgeführt, wobei sich gezeigt hat, dass die Konvergenzgeschwindigkeit des Verfahrens von dem Dichtesprung an der freien Oberfläche, der Bandbreite des Bereichs um die freie Oberfläche und der Auflösung des Diskretisierungsgitters abhängt. Es gilt jeweils nach einer Vergrößerung des Dichtesprungs, der Bandbreite oder der Auflösung, verlangsamt sich die Konvergenzgeschwindigkeit.

Insgesamt wurde die implementierte gebietsabhängige Vorkonditionierung mit der bisherigen AMG vorkonditionierten Variante verglichen. Da Mehrgitterverfahren asymptotisch die effizientesten bekannten Verfahren zum Lösen von großen linearen Gleichungssystemen sind, wird hier die Präasymptotik betrachtet. Diese liegt in heutigen Anwendungen fast immer vor, da selbst Hochleistungsrechner noch nicht in derart hohen Auflösungen Strömungen simulieren können, dass eine Asymptotik vorliegt.

Unter Berücksichtigung des Diskretierungsfehlers der Poissongleichung von $\mathcal{O}(h^2)$ hat sich gezeigt, dass mit der gebietsabhängigen Vorkonditionierung in der Präasymptotik erhebliche Geschwindigkeitsvorteile gegenüber den bisherigen Verfahren erzielt werden können.

Ausblick

Für weitere Entwicklungen zur Optimierung der Druckberechnung kann das im Rahmen dieser Arbeit implementierte Verfahren erweitert werden. Bisher wurden alle Resultate nur im sequenziellen Fall erzielt. Die Geschwindigkeitsvorteile, die bereits im sequentiellen Fall erreicht wurden, werden für ein parallelisiertes Verfahren noch verstärkt, da die gebietsabhängige Vorkonditionierung mit der additiven Schwarz Methode besonders gut zur Parallelisierung geeignet ist. Hingegen sind die Geschwindigkeitsvorteile des Verfahrens mit einer reinen AMG Vorkonditionierung durch eine Parallelisierung wesentlich geringer.

Weiterhin könnte man die Gebietsaufteilung von Ω in Ω_f und Ω_1 nicht, wie bisher in jedem Zeitschritt durchführen, sondern einen Parameter einführen, welcher angibt nach wievielen Zeitschritten die nächste Gebietsaufteilung berechnet wird. Damit kann die Laufzeit nochmals verringert werden. Dies ist insbesondere für Simulationen sinnvoll, bei denen sich die freie Oberfläche nicht allzu viel bewegt oder verformt.

Schließlich ist es eine weitere Überlegung, dass Verfahren auf andere Gleichungen mit hohen Koeffizientensprüngen anzuwenden.

Literaturverzeichnis

[1] Brackbill J.U., Kothe D.B., Zemach C.: *A Continuum Method for Modeling Surface Tension*, J. Comp. Phys., 1992.

[2] Croce R.: *Ein paralleler, dreidimensionaler Navier-Stokes-Löser für inkompressible Zweiphasenströmungen mit Oberflächenspannung, Hindernissen und dynamischen Kontaktflächen.* Diplomarbeit, Institut für Angewandte Mathematik Universität Bonn, 2002.

[3] Croce R., Engel M., Klitz M.: *NaSt3DGPF: Der lineare Löser für die Druckpoissongleichung, Vorkonditionierung und Skalierbarkeit*, Kooperation Institut für Numerische Simulation Universität Bonn und Bundesanstalt für Wasserbau.

[4] Croce R., Griebel M., Schweitzer M.A.: *Numerical simulation of bubble and droplet deformation by a level set approach with surface tension in three dimensions*, Institut für Numerische Simulation Universität Bonn, 2009.

[5] Griebel M., Dornseifer T., Neunhoeffer T.: *Numerical Simulation in Fluid Dynamics, a Practical Introduction*, SIAM, Philadelphia, 1998.

[6] Jüngel A.: *Das kleine Finite-Elemente-Skript*, Universität Mainz, 2001.

[7] Metsch B.: *Ein paralleles graphenbasiertes algebraisches Mehrgitterverfahren*, Institut für Numerische Simulation Universität Bonn, 2004.

[8] Schwarz H.A.: *Gesammelte Mathematische Abhandlungen*, Band 2, Springer Verlag, Berlin, Deutschland / Heidelberg, Deutschland / London, UK / etc., 1890.

[9] Smith B., Bjorstad P., Gropp W.: *Domain Decomposition Parallel Multilevel Methods for Elliptic Partial Differential Equatations*, University of Cambridge, 1996.

[10] Sussman M., Smereka P., Osher S.: *A Level Set Approach for Computing Solutions to incompressible Two-Phase Flow*, J. Comp. Phys., 1994.

[11] Unverdi S.O., Tryggvason G.: *A front-tracking method for viscous, incompressible, multi flows*, J. Comp. Phys., 1992.

[12] Webseite mit einem Bild von einer Wassersäule nach Aufprall eines Wassertropfen auf die Wasseroberfläche, http://www.humedica.org/e5/e768/e784/ (Letzter Zugriff 26. Dezember 2010).

[13] Webseite mit einem Bild von einem rutschenden Wassertropfen auf einem Blatt, http://www.stefan-kallinich.de/resources/ (Letzter Zugriff 26. Dezember 2010).

[14] Webseite des *Message Passing Interface Forums*, http://www.mpi-forum.org/ (Letzter Zugriff 28. Dezember 2010).

[15] Webseite des *Portable, Extensible Toolkit for Scientific Computation* (PETSc), http://www.mcs.anl.gov/petsc/petsc-as/ (Letzter Zugriff 28. Dezember 2010).

[16] Zaspel P.: *Zweiphasige Navier-Stokes Fluidsimulationen in Maya: Konfiguration, Visualisierung und Animation*, Diplomarbeit, Institut für Numerische Simulation Universität Bonn, 2009.