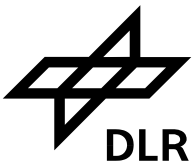



# Verifizierung und Validierung eines Lösungsverfahrens der „Volume of Fluid“ Methode für inkompressible Strömungen

**Bachelorarbeit**

für die Prüfung zum
**Bachelor of Engineering**

im Studiengang Informationstechnik
an der Dualen Hochschule Baden-Württemberg Mannheim

von
**Julian Kaping**

| | |
|---|---|
| Bearbeitungszeitraum: | 27.06.2016 bis 26.09.2016 |
| Kurs, Matrikelnummer: | MA-TINF13ITIN, 9073605 |
| Ausbildungsfirma: | Deutsches Zentrum für Luft- und Raumfahrt e. V. |
| Abteilung, Standort | Center for Computer Applications in AeroSpace Science and Engineering, Göttingen |
| Gutachter der Ausbildungsfirma: | Dr. Roland Kessler und Dr. Johannes Löwe |
| Gutachter der Studienakademie: | Prof. Dr. Harald Kornmayer |

# Eidesstattliche Erklärung

Gemäß §5 (3) der „Studien- und Prüfungsordnung DHBW Technik“ vom 22. September 2011.

Ich habe die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet.

Mannheim, den 3. November 2016

**Zusammenfassung**

Die Verifizierung und Validierung von Implementierungen, die Seiteneffekten neuerer Entwicklungen ausgesetzt werden, ist ein Thema, welches im Rahmen einer aktiven Softwareentwicklung anzutreffen ist. Diese Arbeit hat sich als Ziel gesetzt, anhand verschiedener existierender Vorgehensweisen eine Methodik zu entwickeln, die auf diesen Problemfall zugeschnitten ist. Dabei wird diese Methodik in fünf wesentliche Schritte eingeteilt: Der Erfassung von Anforderungen an die Implementierung, dem Auswählen geeigneter Testdaten, der Vorbereitung und Durchführung dieser Testfälle, der Auswertung der Ergebnisse und der parallel durchgeführten Dokumentation. Ein optionaler sechster Schritt schlägt die Einbindung kritischer Testfälle in automatische Buildprozesse vor.

Um zu zeigen, dass die entwickelte Methodik verwendbare Ergebnisse liefert, wird sie auf die Volume-of-Fluid Methode des THETA Codes angewendet. Im Rahmen der Tests wurde zur einfacheren Initialisierung des zweiten Fluids eine Methode implementiert, die es ermöglicht das Gebiet mit Hilfe eines zweiten Gitters zu definieren. Bei der Verifizierung und Validierung wurde der Fokus auf Seiteneffekte zwischen der VOF Methode und einem Mehrgitterverfahren gelegt. Die Ergebnisse zeigen, dass die VOF Methode ohne negative Seiteneffekte ein Mehrgitterverfahren zur Beschleunigung der Druckkorrekturgleichung verwenden kann. Eine mögliche Fehlerquelle wurde bei der Verwendung einer QUDS Diskretisierung gefunden, bei der die Dichte ein unerwartetes Mischungsverhältnis der beiden Fluide vorhersagt.

**Abstract**

In an active software engineering environment, older implementations are often exposed to side effects of newer implementations. This work aims to provide an approach to structure the verification and validation process of the older implementations. The approach presented structures the process into five mandatory parts: Gathering of requirements on the implementation, selection of suitable test cases, preparation and execution of those tests, evaluation of the results and the parallelized documentation of the process. An optional sixth step is proposed, where critical test cases would be added to an automated build process.

In an effort to prove the method functioning, it is used on the Volume-of-Fluid method in the THETA code. As part of the tests a method for an easier initialization of the second fluid was introduced, enabling the possibility to define the region of the second fluid using a second grid. The verification and validation process focussed particularly on side effects of the VOF method and the multigrid approach. The results show that no negative side effects between those two methods could be determined and the multigrid method is safe to use to speed up the convergence of the pressure correction. One possible error cause could be determined in the use of the QUDS discretization, where the density predicts an unexpected mixture of the two fluids.

# Inhaltsverzeichnis

# Abkürzungsverzeichnis

| | |
|---|---|
| **CAD** | Computer-Aided Design |
| **CFD** | Computational Fluid Dynamics |
| **CFL-Zahl** | Courant-Friedrichs-Lewy-Zahl |
| **CICSAM** | Compressive Interface Capturing Scheme for Arbitrary Meshes |
| **DLR** | Deutsches Zentrum für Luft- und Raumfahrt |
| **FDM** | Finite Differenzen Methode |
| **FEM** | Finite Elemente Methode |
| **FVM** | Finite Volumen Methode |
| **MARIN** | Maritime Research Institute Netherlands |
| **MPI** | Message Passing Interface |
| **QUDS** | Quadratic Upwind Difference Scheme |
| **RANS** | Reynolds-gemittelten Navier-Stokes |
| **UDS** | Upwind Difference Scheme |
| **VOF** | Volume of Fluid |
| **V&V** | Verifizierung und Validierung |

# Abbildungsverzeichnis

# 1 Einleitung

Im Rahmen großer Softwareprojekte wird parallel an verschiedenen Entwicklungen gearbeitet. Um sicherzustellen, dass diese unterschiedlichen Entwicklungen weder Seiteneffekte untereinander, noch rückwirkend auf frühere Entwicklungen besitzen, ist ein ausführlicher Testprozess notwendig. Dieser Testprozess kann hierbei in zwei unterschiedliche Phasen unterteilt werden. Zunächst ist es notwendig sicherzustellen, dass die eigentliche Entwicklung korrekt ist. Danach muss getestet werden, inwieweit die Entwicklung Seiteneffekte verursacht, die an anderen Stellen zu kritischen Fehlern führen. Die zweite Phase des Testprozesses wird auch als Integrationstest bezeichnet.

Bei Software aus der Numerischen Strömungsmechanik handelt es sich bei diesen Entwicklungsarbeiten in vielen Fällen um die Neu- oder Weiterentwicklung verschiedener numerischer Modelle. Da diese untereinander Abhängigkeiten besitzen, haben Änderungen eines Modells häufig Auswirkungen auf andere Modelle. Hierbei muss sichergestellt werden, dass diese Seiteneffekte die Ergebnisqualität nicht negativ beeinflussen.

Aus diesem Grund kann die Notwendigkeit entstehen, vorhergegangene Implementierungen ein weiteres Mal zu verifizieren und validieren. Ziel dieser Arbeit ist es, für diesen Fall eine Methodik zu erarbeiten. Diese Methodik soll es ermöglichen Implementierungen, die bewussten Seiteneffekten ausgesetzt wurden, nachträglich nochmals zu validieren und weitere Eigenschaften zu ermitteln.

Im Kapitel 2 werden grundsätzlich benötigte Grundlagen zur numerischen Strömungsmechanik aufgezeigt. Kapitel 3 entwickelt auf Basis bereits bestehender Vorgehensweisen die erwähnte Methodik. Diese Methodik soll am Beispiel der Verifizierung und Validierung (V&V) des Lösungsverfahrens der Volume of Fluid (VOF) Methode im THETA-Code des Deutschen Zentrums für Luft- und Raumfahrt (DLR) eingesetzt werden. Hierzu wird in Kapitel 4 allgemein erläutert, wie im Umfeld der numeri-

schen Strömungsmechanik Rechnungen aufgebaut werden und in Kapitel 5 wird die praktische Umsetzung gezeigt.

Ziel der V&V des Lösungsverfahrens der VOF Methode ist es, mögliche Fehler derselben aufdecken und zu ermitteln, inwieweit die Verwendung eines Mehrgitterverfahrens die Laufzeit der Rechnung verbessert. Des Weiteren sollen anhand verschiedener Testreihen gute Rechnungseinstellungen gefunden werden, die im Rahmen eines „Best Practice“ Manuals zu dokumentieren sind. Mit Hilfe des „Best Practice“ Manuals soll eine spätere Verwendung der VOF Methode vereinfacht werden.

# 2 Numerische Strömungsmechanik

Computational Fluid Dynamics (CFD), zu deutsch Numerische Strömungsmechanik, beschreibt die Untersuchung von Strömungen unter Einsatz rechnerischer Methoden. Ziel ist es Strömungen in flüssigen und gasförmigen Medien zu simulieren und somit Aussagen über reale Prozesse treffen zu können. Diese Simulationen basieren hierbei in ihrer Grundform auf drei physikalischen Prinzipien: Energieerhaltung, Massenerhaltung und Newton's zweitem Axiom [And95, S. 38].

## 2.1 Navier-Stokes-Gleichungen

Die Navier-Stokes-Gleichungen bilden das Fundament für die Numerische Strömungsmechanik. Aus den drei physikalischen Prinzipien wird ein System partieller Differentialgleichungen formuliert, welches eine Beschreibung von Strömungen ermöglicht. Dieses Differentialgleichungssystem besteht aus den Impulsgleichungen, der Kontinuitätsgleichung und der Energiegleichung. In Gleichung (2.1) sind diese gemeinsam in Vektorform dargestellt. Die Schreibweise wurde aus [Lec09, S. 22] entnommen. Für die Bestandteile des Erhaltungsvektors $\vec{U}$, der Flussvektoren $\vec{E}$, $\vec{F}$ und $\vec{G}$ und dem Quellvektor $\vec{Q}$ wird an dieser Stelle auf Anhang A.1 verwiesen.

$$\frac{\partial}{\partial t}\vec{U} + \frac{\partial}{\partial x}\vec{E} + \frac{\partial}{\partial y}\vec{F} + \frac{\partial}{\partial z}\vec{G} = \vec{Q} \tag{2.1}$$

Diese Erhaltungsgleichungen enthalten 17 Unbekannte: Die Dichte $\rho$, die drei Geschwindigkeitsvektoren $u$, $v$ und $w$, den Druck $p$, die spezifische innere Energie $e$, die spezifische Enthalpie $h$, die Temperatur $T$ und den Spannungstensor $\tau$ mit seinen neun Komponenten [Lec09, S. 23]. Um diese Unbekannten zu ermitteln sind die Erhaltungsgleichungen alleine nicht ausreichend. Daher werden zwölf weitere Beziehungen verwendet: „drei Zustandgleichungen für das Fluid und neun Stokessche

Beziehungen für die Normal- und Schubspannungen“ [Lec09, S. 24]. Diese zwölf Gleichungen können im Anhang A.2 nachgelesen werden. Erst zusammengefasst bilden diese 17 Gleichungen das Fundament für die Numerische Strömungsmechanik, die vollständigen Navier-Stokes-Gleichungen.

Bei der Verwendung der vollständigen Navier-Stokes-Gleichungen werden die Turbulenzen direkt berechnet. In vielen Fällen ist es allerdings erwünscht, stattdessen ein Turbulenzmodell zu verwenden. Ein in der Industrie verbreiteter Ansatz dies umzusetzen sind die Reynolds-gemittelten Navier-Stokes (RANS) Gleichungen. Die Turbulenzmodelle besitzen den Vorteil, dass die räumliche Diskretisierung (Vergleich 2.2) im Gegensatz zu den vollständigen Navier-Stokes-Gleichungen gröber ausfallen kann. Allerdings werden durch die Vereinfachung der Simulation durch Turbulenzmodelle die turbulenten Strömungen nur angenähert berechnet. Eine weitere Vereinfachung von den RANS Gleichungen wäre es vollständig auf die Reibung und die damit verbundenen Turbulenzen zu verzichten. Die entstehenden Gleichungen werden als Euler-Gleichungen bezeichnet. Welche Gleichungen verwendet werden ist von der Problemstellung und den zur Verfügung stehenden Ressourcen abhängig.

Ein zusätzliches Entscheidungskriterium ist, ob die Gleichungen kompressibel oder inkompressibel gelöst werden sollen. Dabei wird betrachtet, inwieweit das Medium unter Druck seine Dichte ändert. In der Realität existieren keine inkompressiblen Medien, für Simulationen kann diese Vereinfachung allerdings unter bestimmten Umständen hilfreich sein. Werden Flüssigkeiten oder langsame Strömungen (Geschwindigkeiten unter 30% der Schallgeschwindigkeit) untersucht, so ist der Einfluss der Kompressibilität so gering, dass er vernachlässigt werden kann. Der in dieser Arbeit verwendete Strömungslöser THETA verwendet den inkompressiblen Ansatz.

Zur Lösung der Gleichungen wird in THETA ein Projektionsverfahren genutzt. Hierbei werden in einem ersten Schritt zunächst die Impulsgleichungen gelöst. Das Ergebnis der Impulsgleichungen verletzt dabei die Kontinuitätsbedingung. Um dies zu beheben wird eine Druckkorrekturgleichung gelöst, die das Ergebnis der Impulsgleichungen dahingehend modifiziert, dass die Kontinuitätsbedingung erfüllt wird und gleichzeitig die Differenz zur Lösung der Impulsgleichungen möglichst gering bleibt [PF02, S.203]. Nachdem die Druckkorrektur erfolgreich durchgeführt wurde, können alle weiteren Gleichungen der verwendeten Modelle gelöst werden.

## 2.2 Diskretisierung

Das Differentialgleichungssystem kann in der Regel nicht analytisch gelöst werden. Stattdessen werden die Gleichungen sowohl zeitlich als auch räumlich diskretisiert, um Näherungslösungen mit endlich vielen Freiheitsgraden zu berechnen. Für die zeitliche Diskretisierung sind zwei unterschiedliche Simulationsarten zu unterscheiden: Die stationäre Simulation und die instationäre Simulation. Bei einer stationären Simulation handelt es sich um eine Rechnung, deren Ergebnis zeitunabhängig ist. Ein Beispiel für eine solche Rechnung wären die Strömungsbedingungen um eine Tragfläche bei konstantem Anstellwinkel und Anstromgeschwindigkeit. Reißt bei dieser Konfiguration die Strömung nicht ab, so ist es nicht notwendig, zeitliche Abhängigkeiten aufzulösen. Im Gegensatz dazu werden bei den instationären Simulationen Effekte erfasst, bei denen sich die Strömungsbedingungen mit der Zeit verändern. Ein Beispiel für eine instationäre Simulation ist das Schwappen von Flüssigkeit in einem Tank. Hierbei verändert sich die Höhe der Flüssigkeit an jedem Ort mit der Zeit, weswegen eine zeitunabhängige Rechnung nicht möglich ist.

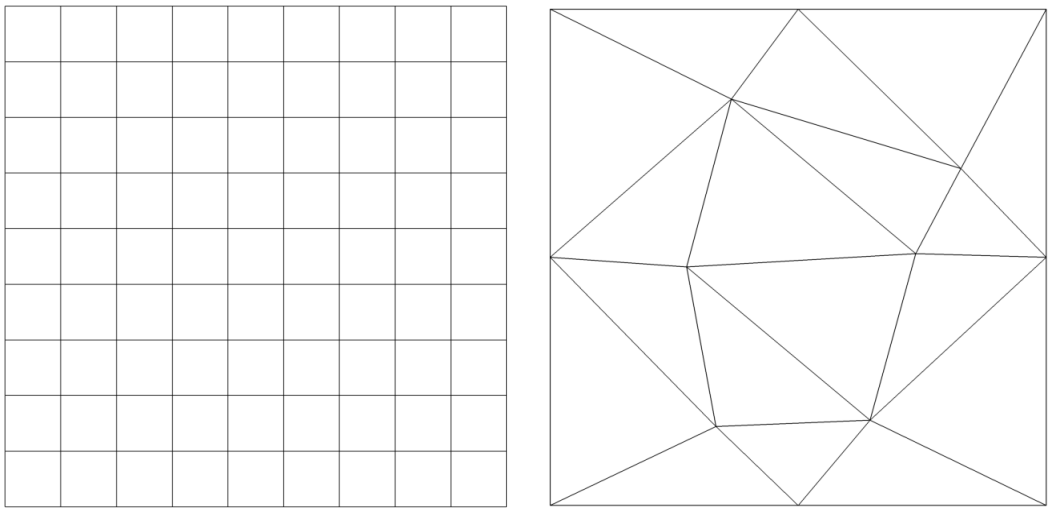
Bei beiden Arten von Simulationen wird für die Rechnung ein diskreter Zeitschritt ausgewählt. Der Unterschied besteht darin, welche physikalische Bedeutung dieser Zeitschritt für die Simulation besitzt: Während bei einer stationären Rechnung ein lokaler Pseudo-Zeitschritt für ein möglichst gutes Konvergenzverhalten ausgewählt wird, muss für die instationäre Rechnung ein Zeitschritt gewählt werden, der die physikalischen Effekte auflösen kann. Für instationäre Rechnung existiert zur Auswahl des Zeitschrittes auch die Courant-Friedrichs-Lewy-Zahl (CFL-Zahl). Bei der CFL-Zahl handelt es sich um eine lokale Größe, die näherungsweise angibt, wie viele Zellen pro Zeitschritt bei der auftretenden Geschwindigkeit durchströmt werden. Zur Berechnung der CFL-Zahl $c$ wird die Geschwindigkeit $u$, die Zeitschrittweite $\Delta t$ und die Zellbreite $\Delta x$ verwendet.

$$c = \frac{u \cdot \Delta t}{\Delta x} \tag{2.2}$$

Die Wahl eines geeigneten Zeitschrittes ist sowohl vom Gitter als auch vom physikalischen Effekt abhängig. Bei der Verwendung von expliziten Lösern ist es notwendig sicherzustellen, dass pro berechneter Zeitschritt nicht mehr als eine Zelle durch die

Strömungsgeschwindigkeit erfasst wird (CFL-Zahl $< 1$). Bei der Erfassung physikalischer Effekte verhält es sich ähnlich wie bei der Erfassung von Schwingungen. Werden zu wenig Abtastpunkte gewählt, so kann die Schwingung falsch interpretiert werden. Bei den Strömungen können durch zu große Zeitschrittweiten physikalische Effekte ebenfalls falsch oder gar nicht erfasst werden.

Mit der räumlichen Diskretisierung ist die Aufteilung des zu berechnenden Raumes gemeint. Dabei wird dieser Raum mir Hilfe eines Gitters erfasst. Ein Gitter besteht hierbei aus den einzelnen Knotenpunkten und den Verbindungen zwischen diesen. Mit diesen Punkten und Verbindungen werden einzelne geometrische Elemente beschrieben. Gitter werden hinsichtlich ihres Aufbaus in strukturierte, unstrukturierte und Hybrid-Gitter unterteilt. Als strukturierte Gitter werden Gitter bezeichnet, die einer regelmäßigen Topologie unterliegen. Dies bedeutet, dass die einzelnen Knotenpunkte des Gitters mit Hilfe von Indizes eindeutig beschrieben werden können [PF02, S. 26]. Dies bedeutet nicht, dass alle Punkte im gleichen Abstand zueinanderstehen und somit notwendigerweise ein kartesisches Netz formulieren müssen.



(a) (b)

Abbildung 2.1: Beipiele eines einfachen zweidimensionalen strukturierten (a) und unstrukturierten (b) Gitters

Ein strukturiertes dreidimensionales Gitter besteht typischerweise aus Hexaedern. Ausgehend von dieser Struktur besitzt jeder Punkt innerhalb des Gitters genau sechs Nachbarn. Durch diese eindeutige Nachbarzuordnung ist es möglich sehr effiziente

Algorithmen für Löser zu entwickeln, die nur auf strukturierte Gitter anwendbar sind. Ein Nachteil der strukturierten Gitter liegt ebenfalls in der regelmäßigen Verteilung der Punkte: Eine lokale Verfeinerung des Gitters ist nicht möglich, da sich eine Verfeinerung an einer Stelle direkt auf das gesamte Gitter auswirkt. Da der Rechenaufwand direkt mit der Feinheit des Gitters zusammenhängt, steigt hierdurch der Rechenaufwand schnell sehr stark. In einem dreidimensionalen Gitter sorgt die Verdopplung der Feinheit in jede Dimension bereits für eine Verachtfachung des Rechenaufwandes. Des Weiteren ist die Vernetzung einer komplexen Geometrie durch ein strukturiertes Gitter im Verhältnis schwieriger umzusetzen, als durch ein unstrukturiertes Gitter.

Für Verfeinerungen ist daher ein unstrukturiertes Gitter gut geeignet. Eine regelmäßige Topologie ist hierbei nicht vorhanden, und verschiedene Punkte können auch unterschiedliche Anzahlen von Nachbarn besitzen. Unstrukturierte dreidimensionale Gitter bestehen typischerweise aus Tetraedern, Dreiecksprismen, Pyramiden und Hexaedern. Die Implementierung eines Lösers ist daher im Allgemeinen schwieriger als für rein strukturierte Netze. Auch der Rechenaufwand der mit dem Lösen eines unstrukturierten Gitters verbunden ist, ist durch stetige Abfragen der Nachbarn höher, als der Aufwand für strukturierte Netze. In vielen Fällen besitzt ein unstrukturiertes Gitter für das gleiche Problem allerdings deutlich weniger Punkte, wodurch der Rechenaufwand insgesamt geringer werden kann.

Hybride Gitter bestehen sowohl aus einem strukturierten Teil als auch aus einem unstrukturierten. Der strukturierte Teil befindet sich hierbei meistens im Bereich der umströmten Geometrie (Grenzschicht) und ist dabei sehr fein aufgelöst. Der unstrukturierte Teil umfasst das restliche Gitter (Fernfeld) und wird mit größer werdenden Abstand zur Geometrie gröber, da die Strömungsverhältnisse dort in den meisten Fällen weniger turbulent sind und daher eine geringere Auflösung benötigen. Durch diese Aufteilung ist es möglich die Vorteile eines unstrukturierten Gitters hinsichtlich der Verfeinerung, bzw. Vergröberung zu nutzen, und im relevanten Teil der Strömung dennoch geordnete Schichten aufzubauen. Dadurch besitzen Hybrid-Gitter eine hohe Flexibilität und können für einzelne Bereiche Vorteile von strukturierten Blöcken nutzen.

Für die Lösung der Gleichungen auf den Gittern existieren verschiedene Lösungsan-

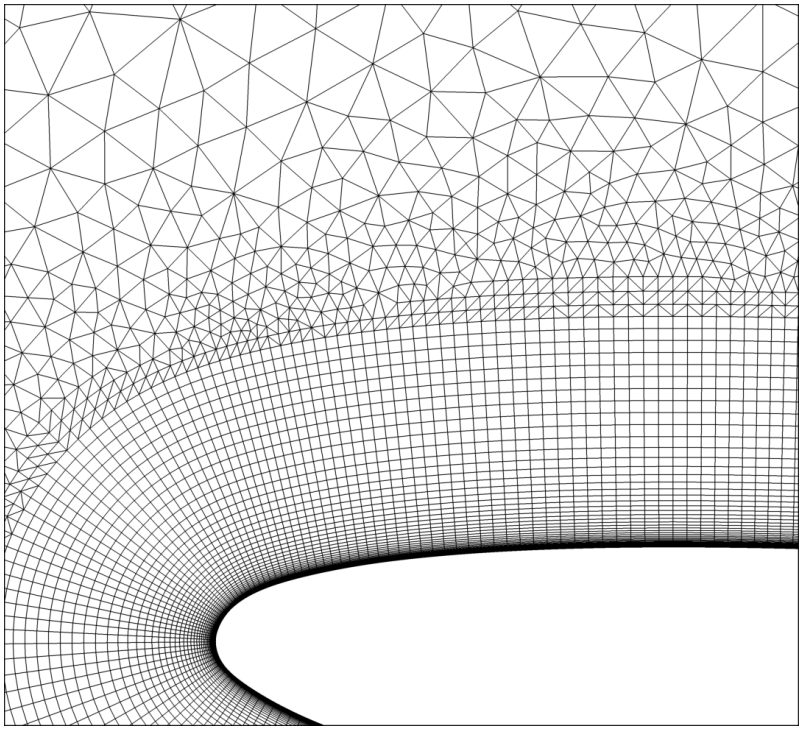


Abbildung 2.2: Ausschnitt eines Hybrid-Gitters

sätze, die unterschiedlich definieren, wie Werte auf dem Gitter verteilt sind [Lec09, S.43]. Ein möglicher Ansatz ist die Finite Differenzen Methode (FDM), bei dem die Werte direkt für die Punkte des Gitters berechnet werden. Zur Berechnung der einzelnen Werte können unterschiedliche Nachbarschaftsbeziehungen definiert werden, anhand derer die Berechnung erfolgt. Ein weiteres Verfahren ist die Finite Elemente Methode (FEM). Hierbei werden für einzelne Elemente des Gitters unterschiedliche mathematische Funktionen definiert, die bestimmen, wie sich die Werte innerhalb dieses Elementes verhalten. Für die mathematischen Funktionen der FEM können Eigenschaften festgelegt werden, die von diesen erfüllt werden müssen. So wird beispielsweise zwischen stetigen und unstetigen FEM unterschieden, wobei bei stetigen FEM die Übergänge zwischen den Elementen stetig sein müssen. Ein Spezialfall der FEM ist die Finite Volumen Methode (FVM), bei der es sich um eine unstetige FEM handelt, die als mathematische Funktionen für die Elemente nur Konstanten zulässt. Die einzelnen Elemente werden hier durch ihre Volumen gekennzeichnet, wodurch sich der Name dieses Verfahren herleitet. Der Strömungslöser THETA verwendet die FVM zur Lösung der Gleichungen.

## 2.3 Duale Gitter

Die in dem Kapitel 2.2 vorgestellten Gitter, werden im Umfeld des Strömungslösers THETA auch als Primärgitter bezeichnet. Der Löser selbst operiert allerdings nicht direkt auf Primärgittern, sondern erzeugt auf Grundlage dieser ein Duales Gitter [And95]. In den nächsten Abschnitten werden sowohl die Erzeugung als auch die Vorteile dieser Dualen Gitter näher beschrieben.

Bei der Erstellung der dualen Gitter wird jedem Punkt ein Kontrollvolumen zugeordnet. Dieses Kontrollvolumen wird auf der Grundlage der verschiedenen Primärelemente gebildet. Im Fall von THETA-Gittern werden unter Primärelementen Tetraeder, Dreiecksprismen, Pyramiden und Hexaeder verstanden. Für jedes dieser Primärelemente wird das Barycenter ermittelt. Die Koordinaten des Barycenters $\vec{X}_S$ werden mit der Anzahl der Eckpunkte $n$ und den Koordinaten der Eckpunkte $X(\vec{P}_i)$ nach folgender Formel berechnet:

$$\vec{X}_S = \frac{\sum_{i=1}^{n} X(\vec{P}_i)}{n} \tag{2.3}$$

Der somit ermittelte Punkt wird für jedes Element mit den Barycentern der einzelnen Randelementen verbunden. Dadurch entstehen sogenannte Facetten, deren Eckpunkte im dreidimenionalen Raum durch jeweils den Barycenter des Primärelementes, den Barycenter einer Seitenfläche und dem Barycenter einer Kante beschrieben werden.

Diese Facetten werden zu einzelnen Flächen (Faces) zusammengefasst. Der Strömungslöser THETA agiert somit nicht auf den Punkten des Primärgitters, sondern auf den Faces, die durch das Primärgitter beschrieben werden. Eine Face beschreibt hierbei die Grenzfläche zwischen zwei Kontrollvolumen und stellt die Nachbarschaftsbeziehung zwischen diesen dar.

Das duale Gitter zu verwenden besitzt gegenüber der Verwendung der Primärelemente als Kontrollvolumen Vorteile hinsichtlich der Robustheit und der Genauigkeit. Durch die Verwendung der dualen Zellen erhöht sich in vielen Fällen die Anzahl der Nachbarn, die ein einzelnes Kontrollvolumen besitzt. Dies ist sehr gut an einem zweidimensionalen Beispiel (Vergleich Abbildung 2.4) zu beobachten: Wird eine

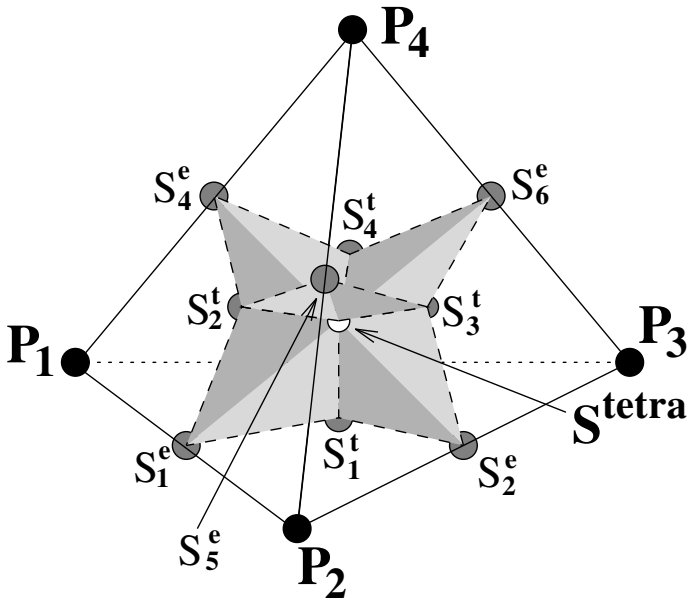



Abbildung 2.3: Generierung der Facetten am Beispiel eines Tetraeders [DLR12, S.131]

Fläche mit möglichst regelmäßigen Dreiecken ausgefüllt, so besitzt jedes Primärelement maximal drei Nachbarn, während die Anzahl für die dualen Kontrollflächen im Regelfall bei etwa sechs liegt. Eine höhere Anzahl von Nachbarn erlaubt beispielsweise häufig eine genauere Auswertung von Gradienten, was sich positiv auf die Qualität der Lösung auswirkt.

## 2.4 Mehrgitter-Verfahren

Nachdem in den vorherigen Kapiteln grundlegende Vorraussetzungen zur Durchführung einer Simulation erklärt wurden, erläutert der nächste Abschnitt, um was es sich bei einem Mehrgitterverfahren handelt. Der Einsatz eines Mehrgitter-Verfahrens besitzt als Intention, die Rechenzeit der Simulation für große Gitter zu verringern. Erreicht wird dies, indem für die gleiche Geometrie mehrere Gitter unterschiedlicher Feinheitsgrade erzeugt werden [PF02, S.348f]. Eine Möglichkeit ist es, ein feines Gitter zu erzeugen und für das nächstgröbere Gitter jeweils einige Kontrollvolumen zu einem größeren Kontrollvolumen zusammenzufassen. Für ein dreidimensionales Gitter bietet sich hierbei beispielsweise an, jeweils etwa acht Kontrollvolumen zusam-

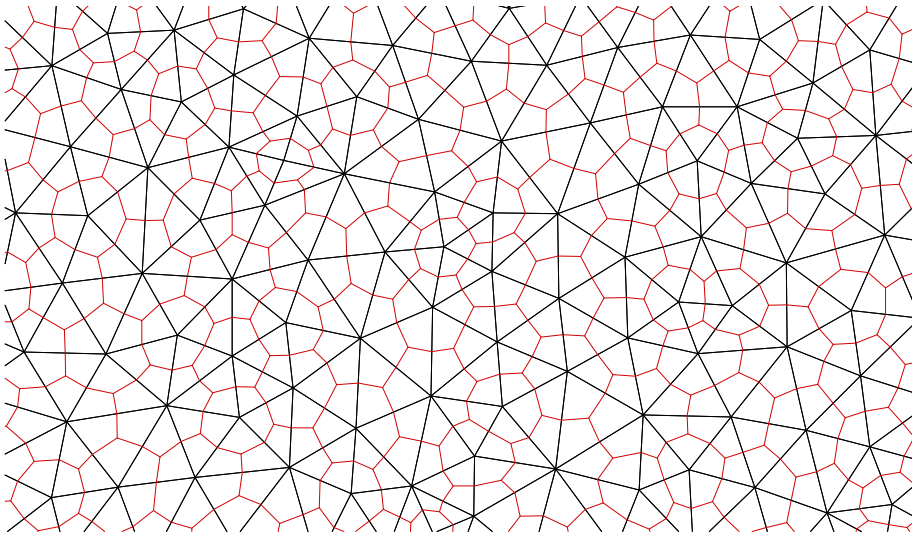


Abbildung 2.4: Primärgitter (schwarz) und Duales Gitter (rot) in einem unstrukturierten Bereich

menzufassen [PF02, S.113]. Die dadurch entstehenden Gitter sind jeweils gröber als das vorhergehende und unter Verwendung geeignet optimierter Algorithmen kann die Rechnung auf diesen Gittern gemeinsam effizienter zu einem Ergebnis führen. Dabei ist zu bemerken, dass die Qualität der Lösung von dem feinsten Gitter abhängig ist. Die Lösung bei Verwendung eines Mehrgitter-Verfahrens ist identisch mit der Lösung eines Algorithmus der nur auf dem feinen Gitter rechnet. Die groben Gitter beschleunigen ausschließlich die räumliche Informationsausbreitung.

Der verwendete Algorithmus für ein Mehrgitter-Verfahren benötigt zwei Methoden, um mit den erzeugten Leveln von Gittern rechnen zu können: Restriktion und Prolongation. Bei der Restriktion handelt es sich um eine Methode, die die Strömungsgrößen von dem feineren Gitter auf das nächstgröbere projiziert. Die Prolongation stellt den umgekehrten Prozess dar und definiert, wie die Strömungsgrößen von einem gröberen auf das nächst feinere Gitter interpoliert werden [PF02, S.112f]. Mit Hilfe dieser beiden Methoden können bei einer bestehenden Beziehung zwischen den feineren und gröberen Gitter effiziente Mehrgitter-Verfahren eingesetzt werden.

Bei der Verwendung von mehreren Gittern wird eine Routine benötigt, die entscheidet was auf welchem Gitter durchgeführt wird. Weitverbreitete Möglichkeiten sind hierbei V- und W-Zyklen. Bei beiden Möglichkeiten wird durch die verschiedenen Gitterlevel iteriert. Auf dem gröbsten Gitter werden die Gleichungen annähernd exakt

gelöst. Auf den restlichen Gitterleveln werden lediglich eine festgelegte Anzahl von Lösungsschritten durchgeführt. Dieser Prozess wird auch als Glätten bezeichnet.

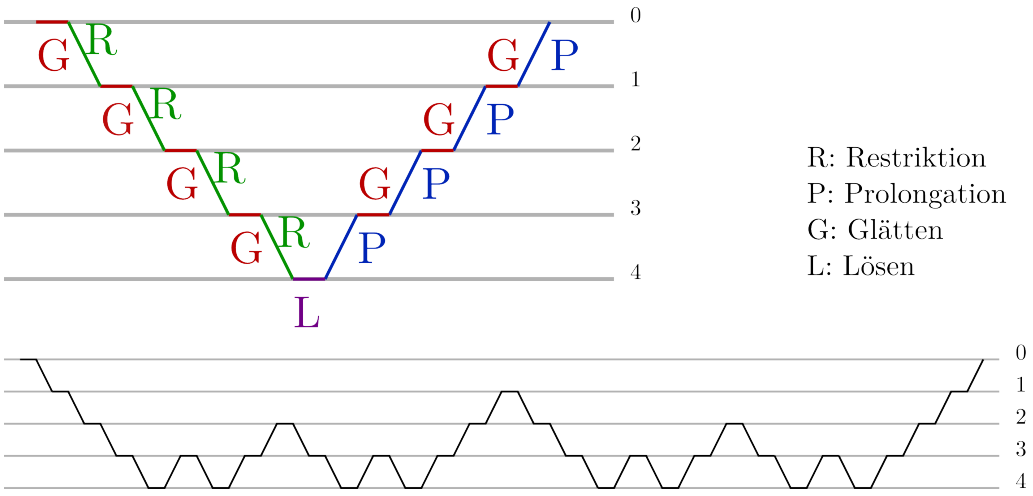



Abbildung 2.5: V- und W-Zyklen für Mehrgitteriterationen [Hac85]

Ein Vorteil der Verwendung eines Mehrgitter-Verfahrens liegt darin, dass in weniger Iterationen das gesamte Gitter von der Anfangsströmung erfasst wird. Es ist üblich als Startlösung für das Gitter an allen Punkten die Strömungsbedingungen auf Null zu setzen und lediglich am Einströmrand die Einströmgeschwindigkeit anzugeben. Bei vielen Lösungsverfahren können Strömungsinformationen nur von einem Punkt zum nächsten übertragen werden, daher ist die Anzahl der Iterationen, die benötigt wird, damit die letzten Punkte von der Strömung erfasst werden, proportional zur Anzahl der Zellen in der Breite des Gitters. Wird ein gröberes Gitter verwendet, so verläuft die allgemeine Erfassung des Gitters deutlich schneller ab, da die Anzahl der Punkte in der Breite des Gitters abnimmt. Somit kann die Anzahl der benötigten Mehrgitteriterationen für nahezu beliebig große Gitter fast konstant gehalten werden, indem mehr Gitterlevel verwendet werden.

## 2.5 Mehrphasenströmungen

Bei Mehrphasenströmungen handelt es sich um Rechnungen, bei denen zwei unterschiedliche Medien und deren Interface zwischeneinander behandelt werden. Beispiele für Mehrphasenströmungen wären ein teilweise gefüllter Treibstofftank oder auch

Wasserwellen. Bei beiden Beispielen muss die Strömung in dem gasförmigen und innerhalb des flüssigen Mediums berechnet werden. Auf Grund hoher Dichteunterschiede kann es ohne ein zusätzliches Modell dabei zu Problemen kommen. Vor allem die Erhaltung eines Interface ist kaum möglich, da es bei typischen Diskretisierungen mit der Zeit zu unphysikalischen Vermischungen der beiden Medien kommt.

Das Modell, welches innerhalb des THETA-Codes Anwendung findet, wird als Volume of Fluid (VOF) Methode bezeichnet. Dabei wird die VOF Methode zur Simulation von Strömungen mit sowohl einem Gas als auch einer Flüssigkeit verwendet. Bei der VOF Methode wird dabei für jedes Kontrollvolumen eine Strömungsgröße gespeichert, die beschreibt, in welchem Verhältnis die beiden Medien innerhalb des Kontrollvolumens auftreten. Dabei steht eine eins dafür, dass das Volumen vollständig mit der Flüssigkeit gefüllt ist, und eine null dafür, dass nur das Gas enthalten ist. Das Interface zwischen den beiden Medien wird durch diejenigen Zellen definiert, deren VOF Werte zwischen eins und null liegen. In dem Transportterm der VOF Gleichungen wird in THETA das Compressive Interface Capturing Scheme for Arbitrary Meshes (CICSAM) als Diskretisierung verwendet, das das Interface zwischen den Medien erhält, und für unterschiedliche Gitterstrukturen anwendbar ist [Gau+12].

# 3 Systematische Verifizierung und Validierung

Thema der Arbeit ist es, eine bereits existierende Implementierung eines Lösungsverfahrens der VOF Methode zu verifizieren und validieren. Das nachfolgende Kapitel hat zur Zielsetzung eine allgemeine Vorgehensweise zu spezifizieren, die für diese und vergleichbare Aufgabenstellungen verwendbar ist.

## 3.1 Begriffsabgrenzung

Die Begriffe Verifizierung und Validierung (V&V) sind im Bereich der Software Entwicklung durch den IEEE Standard 1012 [IEE04] definiert. Dieser beschreibt den Prozess einer Verifizierung als Auswertung inwieweit ein System oder eine Komponente eines Systems den Bedingungen entspricht, die am Anfang der Entwicklungsphase festgelegt worden sind. Dies bedeutet, dass während einer Verifizierung kontrolliert wird, ob die Implementierung korrekt vorgenommen wurde.

Eine Validierung wird vom IEEE 1012 Standard von einer Verifizierung abgegrenzt. Unter einer Validierung wird ein Prozess verstanden, der auswertet, inwieweit ein System oder eine Komponente den vor dem Anfang des gesamten Entwicklungsprozesses spezifizierten Anforderungen entspricht. Dies bedeutet, dass eine Validierung kontrolliert, ob ein gegebenes Produkt seine Aufgabe hinreichend erfüllt.

Wird die V&V im gesamten Prozess der Software-Entwicklung betrachtet, so sind vor allem die Begriffe des Lastenheftes und des Pflichtenheftes von Relevanz. In einem Lastenheft werden Anforderungen an die Fähigkeiten eines Produktes festgelegt, während ein Pflichtenheft die konkreten gewählten Vorgehensweisen festlegt, die zur Erreichung der Fähigkeiten durchgeführt werden sollen. An einem Beispiel bedeutet dies, dass im Lastenheft festgelegt wird, dass die Software in der Lage sein soll, Listen

effizient zu sortieren. Im Pflichtenheft wird hierfür die Verwendung eines Quicksort festgelegt. Die Verifizierung konzentriert sich darauf zu kontrollieren, inwieweit die Implementierung des Quicksorts korrekt durchgeführt worden ist, während die Validierung kontrolliert, inwieweit die vorhandene Implementierung (unabhängig vom Ausgang der Verifizierung) in der Lage ist eine Liste effizient zu sortieren.

## 3.2 Methodik einer Verifizierung und Validierung

Durch [IEE04] sind der Ablauf und die Voraussetzungen für einen allgemeinen V&V-Prozess festgelegt. Der Fokus liegt hierbei größtenteils auf der Neuentwicklung von Software im Allgemeinen. Des Weiteren beziehen sich die vorgegebenen Schritte auf allgemeine Entwicklungen von Software, während die Entwicklung und Validierung eines numerischen Modells davon zu unterscheiden ist.

Dem entgegen untersucht Sargent [Sar11] die V&V von Simulationsmodellen. Dies entspricht der gegebenen Aufgabenstellung im Ansatz etwas mehr, doch auch Sargent bezieht sich bei seiner vorgeschlagenen Methodik auf die Erstentwicklung eines solchen Modells. Ziel dieses Kapitels ist es daher, auf Basis der genannten Quellen eine Vorgehensweise zu entwickeln, die es ermöglicht bereits vergangene Entwicklungen im Kontext neuerer Entwicklungen an anderer Stelle erneut zu untersuchen. Im Fokus steht hierbei sicherzustellen, dass die neueren Entwicklungen keinen, oder einen geringen negativen Einfluss auf die Ergebnisse der zu untersuchenden Entwicklung haben. Sollten Fehler oder negative Einflüsse gefunden werden, so sollte durch die Methodik eine rekursive Vorgehensweise unterstützt werden, um die evtl. vorgenommenen Verbesserungen erneut zu prüfen.

In Abbildung 3.1 ist der vorgeschlagene Prozess dargestellt. Die nächsten Abschnitte erklären die einzelnen Schritte weiter. Während der einzelnen Kapitel werden hierbei die Schritte aus [IEE04] referenziert. Zur Übersicht sind diese in Anhang F aufgelistet.

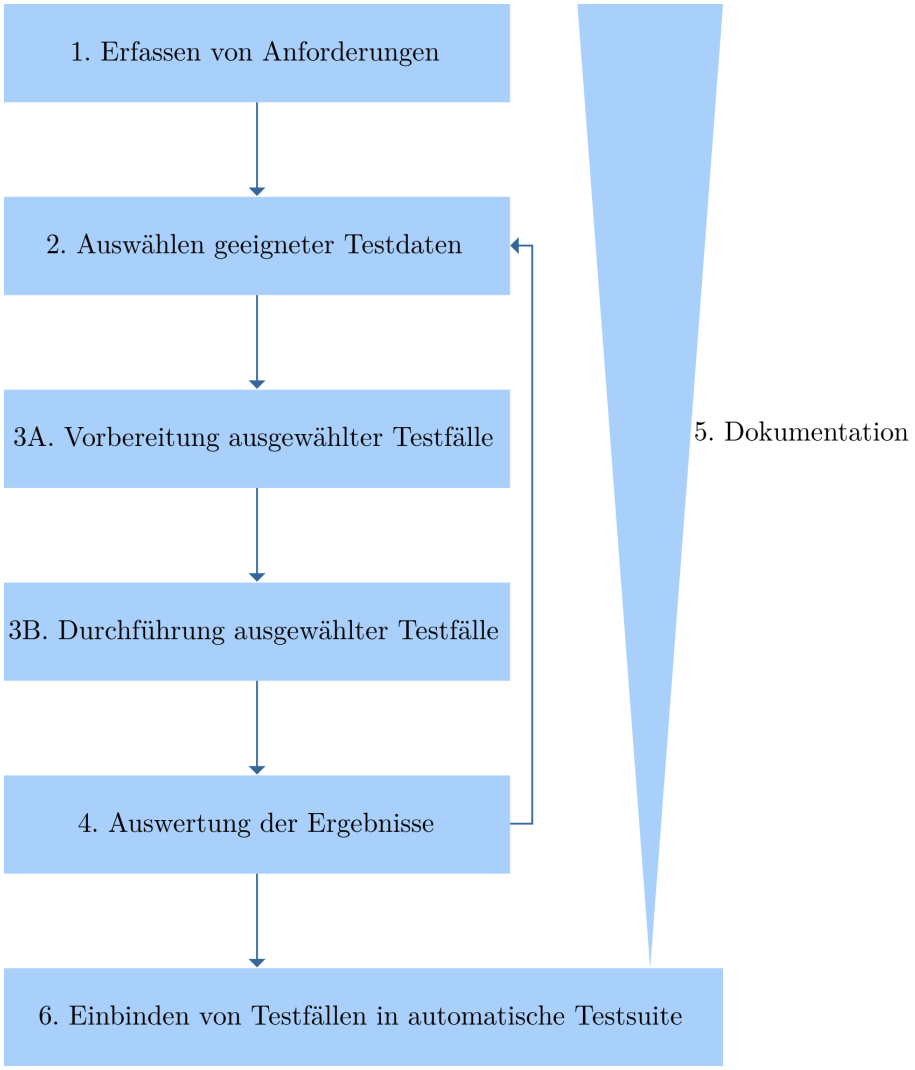



Abbildung 3.1: Ablauf eines V&V Prozesses zur nachträglichen Untersuchung einer Implementierung

### 3.2.1 Vorbereitungen

Im ersten Schritt der V&V Untersuchung müssen Bedingungen hinsichtlich der geforderten Genauigkeit und weitere Anforderungen an die bestehende Implementierung erfasst werden. Dabei entspricht dieser Schritt den ersten beiden Schritten der V&V von Simulationsmodellen [Sar11, S.194] und dem ersten Schritt einer Implementations V&V nach IEEE 1012 [IEE04, S.17]. In diesem Schritt wird die vorhergehende Dokumentation der Implementierung nach den Rahmenbedingungen untersucht und

Kernaspekte dokumentiert. Ziel dieses Schrittes ist es, für den weiteren Verlauf der V&V den Rahmen eines Erfolges und eines Misserfolges abzustecken. Desweiteren wird festgehalten welche Testmethoden für die V&V verwendet werden sollen.

Im zweiten Schritt werden geeignete Testdaten und Testfälle ausgewählt und evaluiert, welche Bereiche der V&V durch diese abgedeckt werden. Dieser Schritt umfasst die Schritte fünf bis acht der Implementations V&V nach [IEE04, S.17]. Grundlagen für diesen Schritt können hierbei vorhergehende dokumentierte Arbeiten mit der Implementierung (Vergleich mit vorhergehenden Ergebnissen), Daten von Experimenten die durch das Simulationsmodell nachgestellt werden sollen (Vergleich mit realen Experimenten), Ergebnisse von anderen Implementationen des Simulationsmodells oder anderen Simulationsmodellen mit einer ähnlichen Zielsetzung (Vergleich mit anderen Implementierungen) oder analytische Ergebnisse für abstrahierte Testfälle (Vergleich mit theoretischen Analysen) sein. Bestandteil dieses Schrittes ist es festzustellen, welcher Teil der V&V mit Hilfe welches Testfalls abgedeckt wird, und inwieweit weitere Literatur vergleichbare Ergebnisse bereitstellt. Testfälle, bei denen mit anderen Rechnungen oder Experimenten verglichen werden, müssen dabei stets unter der Annahme behandelt werden, dass auch diese Daten nicht exakt die Wirklichkeit darstellen. Daher müssen für die Testfälle geeignete Fehlermaße festgelegt werden, die hinsichtlich der Anforderungen des ersten Schrittes geeignet ausgewertet werden können.

### 3.2.2 Durchführung

Im dritten Schritt werden die Testfälle vorbereitet und durchgeführt. Dieser Schritt deckt sich mit den Schritten neun bis zwölf des Implementations V&V nach [IEE04, S.17] und dem Schritt drei der V&V von Simulationsmodellen [Sar11, S.194]. In diesem Schritt werden die nötigen Vorbereitungen und Durchführungsmaßnahmen für die Testfälle durchgeführt. Dies bedeutet, dass alle benötigten Voraussetzungen für den Verlauf des Testfalls bereitgestellt werden müssen (bsp. benötigte Ressourcen, Erzeugung von Gittern) und die Maßnahmen zur Fehleranalyse ggf. implementiert werden. Des Weiteren werden diese Tests durchgeführt und die Ergebnisse der Rechnung und Fehleranalyse dokumentiert. Für Rechnungen im Bereich der numerischen

Strömungsmechanik ist eine genauere Auflistung der benötigten Schritte für die Durchführung von Testfällen in Kapitel 4 festgehalten.

### 3.2.3 Auswertung und Dokumentation

Der vierte Schritt wertet die Ergebnisse der vorhergehenden Testfälle aus und kategorisiert mögliche auftretende Fehler. Mit diesem Schritt werden die Schritte 13 bis 15 des IEEE 1012 beschrieben. Dabei werden die Fehlermaße, die in Schritt zwei definiert und in Schritt drei implementiert wurden, dahingehend untersucht, inwieweit diese im in Schritt eins festgelegten Rahmen liegen. Sollten hierbei die festgelegten Grenzen überschritten werden, so ist über weitere Iterationen der Schritte zwei bis vier sicherzustellen und einzugrenzen, ob und worin die Fehlerursache im Rahmen der zu untersuchenden Implementierung liegt. Werden hierbei Fehler an der Implementierung festgestellt, so sind diese ausreichend zu dokumentieren. Mögliche Fehler, die hierbei auftreten können sind hinsichtlich ihrer Schwere zu untersuchen. Dabei muss untersucht werden, unter welchen Umständen der Fehler auftritt und in welchem Rahmen die Ergebnisse der Rechnung durch den Fehler beeinträchtigt werden.

Der fünfte Schritt des V&V Prozesses beschreibt die Dokumentation des selbigen und muss parallel zu den anderen vier Schritten durchgeführt werden. Die Dokumention ist der wesentliche Schritt, damit die Ergebnisse der V&V auch bei zukünftigen Entwicklungen von Nutzen sind. Im Rahmen der Dokumentation sind die festgelegten Kernaspekte der V&V festzuhalten und zu erläutern. Des Weiteren sind die ausgewählten Testfälle zu beschreiben und ihr Verwendungszweck zu dokumentieren. Für spätere Analysen und Entwicklungen sind diejenigen Mittel zur Verfügung zu stellen, die zur Rekonstruktion der Ergebnisse notwendig sind. Dies umfasst sowohl die Dokumentation der verwendeten Version der Implementierung, als auch die erzeugten Ressourcen im Rahmen der Vorbereitung für die Durchführung der Testfälle. Sollten Grafiken auf Basis von Ergebnissen erstellt worden sein, so sind auch die Ergebnisse zur Verfügung zu stellen, um die Grafiken erneut erzeugen zu können. Ziel dieser Dokumentation ist es dabei, eine erneute V&V auf Basis der gleichen Softwareversion zu verhindern und die Ergebnisse unternehmensweit veröffentlichen zu können.

### 3.2.4 Einbindung in automatische Tests

Ein optionaler sechster Schritt ist es, ausgewählte kritische Testfälle in eine automatische Testsuite einzubauen. Für große Softwareprojekte mit vielen parallelen Entwicklungsarbeiten bietet es sich zur kontrollierten Einführung neuer Funktionen an, automatisierte Tests durchzuführen, die grundlegende und sicherheitskritische Funktionen der Anwendung bei jeder Neuerung testen. Im Rahmen dieser V&V können ausgewählte kritische Testfälle analysiert werden und zu einer solchen Testsuite hinzugefügt werden.

Im Gegensatz zu Komponenten- und Integrationstests, die standardmäßig in automatisierten Testsuiten umgesetzt sind, gibt es auch die Möglichkeit hierbei ganze Testfälle auswerten zu lassen. In diesem Fall werden die Ergebnisse der Testsuite an ein Review eines zweiten Entwicklers gekoppelt. Ein Reviewer erhält hierbei unter anderem die Abweichungen der Testergebnisse von den zu erwartenden Ergebnissen und kann auf Basis der Dokumentation der zu reviewenden Änderung entscheiden, inwieweit diese Abweichungen korrekt sind oder nicht [DLR15]. Bei Abweichungen, die im Rahmen der neuen Entwicklungen dokumentiert werden, können die neuen, korrekten Ergebnisse als Vergleichsergebnisse in die Testsuite aufgenommen werden. Abweichungen, die nicht dokumentiert wurden, allerdings nach Absprache und Kontrolle korrekt sind, müssen dokumentiert werden, bevor die neuen Vergleichsergebnisse in die Testsuite übernommen werden können. Inkorrekte Änderungen sind ein Anzeichen für einen Fehler in der neuen Entwicklung und können ein Grund sein, warum das Review fehlschlägt und der zugehörige Entwickler weitere Änderungen an der Implementierung durchführen muss.

# 4 Ablauf einer Simulation im THETA-Umfeld

Nachdem im letzten Kapitel dargestellt wurde, wie eine Verifizierung und Validierung nachträglich für ehemalige Entwicklungsarbeiten durchgeführt werden kann, wird im Folgenden dargestellt, wie im Bereich der numerischen Strömungsmechanik Simulationen für beispielsweise benötigte Tests durchgeführt werden. Dabei werden sowohl allgemeine Prozesse beleuchtet, die unabhängig vom verwendeten Strömungslöser sind, als auch THETA spezifische Schritte.

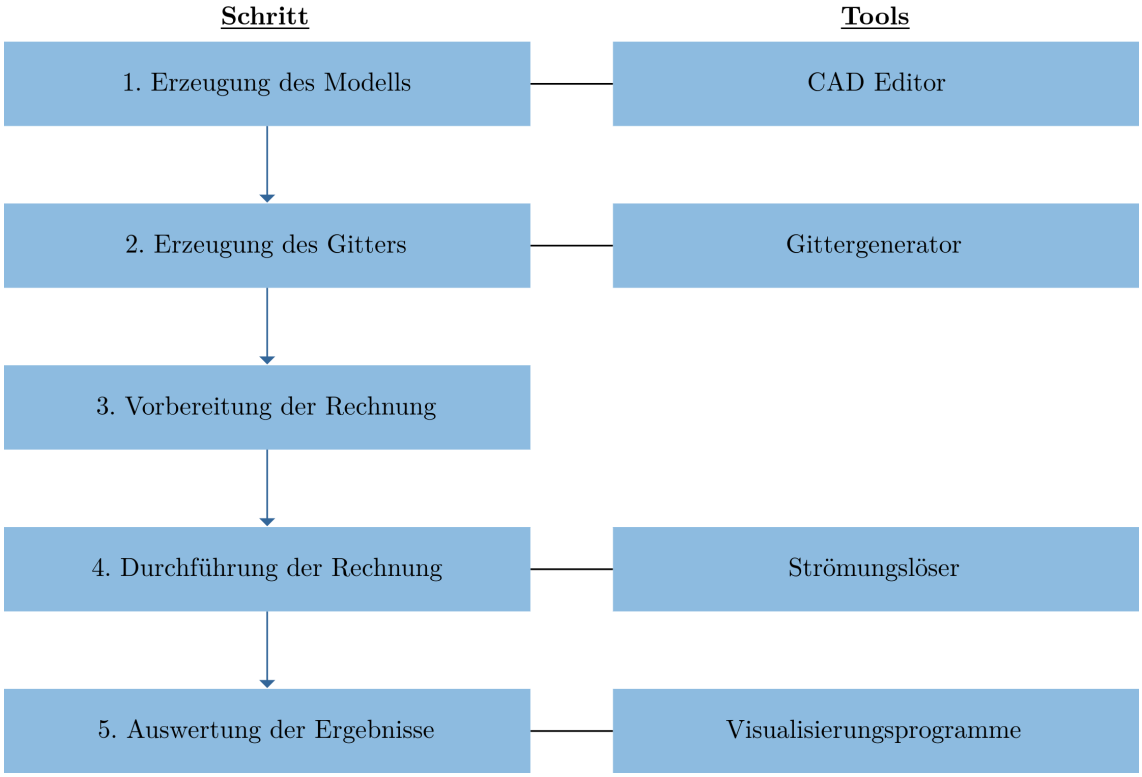



Abbildung 4.1: Ablauf zur Durchführung einer Numerischen Strömungssimulation

## 4.1 Erstellung einer Geometrie

Der erste Schritt zur Vorbereitung einer Strömungssimulation ist es, die geometrische Beschreibung des Rechengebietes zu erzeugen. Bei dieser Geometrie kann es sich beispielsweise um das Profil eines Flügels, ein komplettes Flugzeug oder auch um ein Windrad handeln. Zusätzlich sind auch die äußeren Grenzen des Rechengebietes Teil der geometrischen Beschreibung. Dieser Prozess wird auch als Computer-Aided Design (CAD) bezeichnet. Die Geometrie kann auf unterschiedliche Arten beschreiben werden. Zum einen ist es möglich, das Modell mit Hilfe von mathematischen Funktionen zu beschreiben. Dieser Ansatz besitzt den Vorteil, dass mit Hilfe der mathematischen Funktion Gitter unterschiedlicher Feinheitsgrade verhältnismäßig realisiert werden können. Dabei werden auf der Punkte die Punkte unterschiedlich fein über die definierte Geometrie verteilt. Allerdings ist es nicht für alle Geometrien möglich eine mathematische Beschreibung zu erstellen.

Ein weiterer Ansatz ist es, das Modell mit Hilfe einer Triangulierung zu definieren. Dabei wird die Oberfläche durch Dreiecksnetze approximiert . Ein Vorteil dieses Ansatzes ist es, dass er für alle Modelle verfügbar ist, da die Oberfläche durch Punkte und ihre Verbindungen definiert wird. Im Gegensatz zur mathematischen Beschreibung kann allerdings bei einer geringen Auflösung der Triangulierung die Genauigkeit niedrig sein, und somit Kanten entstehen, die ursprünglich nicht geplant waren. Des Weiteren ist die Genauigkeit des Gitters an die Genauigkeit des Modells gebunden, wenn nicht durch einfache Methoden Punkte hinzugefügt werden können.

Daten für viele verschiedene Modelle, die bereits in der Vergangenheit gut untersucht worden sind, sind häufig im Internet frei zugänglich zu finden. So ist es beispielsweise möglich für verschiedene Flügelformen unter [Air] Punkte zur Erstellung bestimmter Flügelgeometrien zu recherchieren.

## 4.2 Gittererzeugung

Nach der Erzeugung der Begrenzung und der Geometrie ist es notwendig auf Basis dieser ein Gitter für den Strömungslöser zu erzeugen. Bei diesem Schritt wird das durch die Ränder definierte Gebiet durch Volumen- oder Flächenelemente ausgefüllt.

Dabei wird entschieden, inwieweit ein strukturiertes oder ein unstrukturiertes Gitter verwendet wird, und wie fein die Auflösung sein soll.

Für die Erzeugung des Gitters werden im Rahmen von Simulationen mit THETA eigenständige Programme genutzt. Dabei ist die Wahl des Gittergenerators sowohl von der gewünschten Gitter-Topologie als auch von persönlichen Präferenzen hinsichtlich der Bedienbarkeit abhängig. Einzelne Beispiele für kommerzielle Gittergeneratoren sind CENTAUR [Cen16] oder Pointwise [Poi16].

Gitter für Strömungssimulationen sollten einige Qualitätskriterien erfüllen. Zunächst sollten sich die Größenverhältnisse zwischen benachbarten Zellen nicht zu sehr unterscheiden. Lecheler empfiehlt etwa ein maximales Längenverhältnis von 1,2 [Lec09, S.109]. Mit ausreichend kleinen Größenverhältnissen wird bei den dualen Gittern gewährleistet, dass der primäre Gitterpunkt in der Nähe des dualen Zellmittelpunktes bleibt. Dies sorgt im Allgemeinen für genauere Resultate der Rechnung.

Des Weiteren sollten degenerierte Zellen vermieden werden. Werden Hexaeder verwendet, so sollten deren Innenwinkel möglichst rechtwinklig sein. Bei unstrukturierten Gittern sollten sehr große (>170°) und sehr kleine (<10°) Winkel vermieden werden. Auch solche primären Elemente können bei der Erzeugung der dualen Zellen zu qualitativ schlechteren Ergebnissen führen. In Abbildung 4.2 sind einzelne Beispiele von Tetraedern dargestellt. Der Tetraeder mit der Bezeichnung „Round“ beschreibt dabei ein qualitativ hochwertiges Element, da keine extremen Winkel vorliegen.

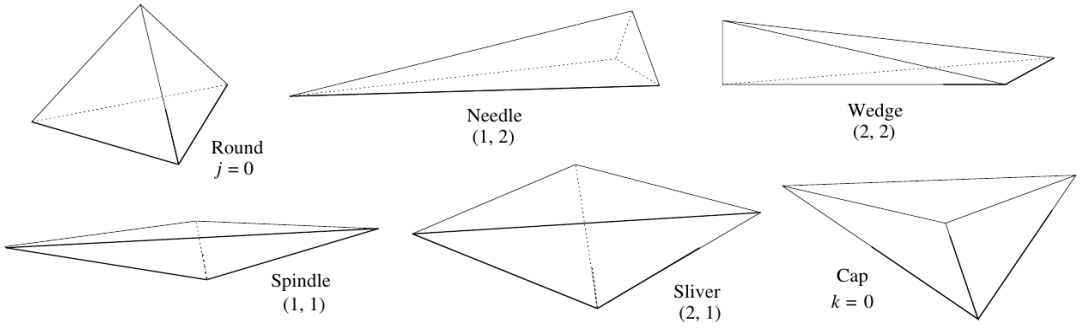



Abbildung 4.2: Beispiele Tetraeder mit unterschiedlichen Mängeln [Ber+95]

Die Gittererzeugung läuft in vielen Fällen in einzelnen Schritten ab. Zunächst werden auf den durch die geometrische Beschreibung definierten Rändern Punkte

für das Rechennetz verteilt. Als nächstes wird entschieden in welche Bereiche das Gitter eingeteilt wird. Für Simulationen um Flügelprofile kann es sich anbieten hybride Gitterstrukturen zu nutzen und dabei eine feine strukturierte Schicht um das Profil aufzubauen. Auf Grundlage dieser Entscheidungen wird dann ein Gitter erzeugt. Dieses Gitter bietet dann die Grundlage für weitere Optimierungen. So können schlechte Zellen analysiert und ausgebessert werden oder die Gitterfeinheit an Strömungsmerkmale angepasst werden.

Für komplexe Geometrien dauert die Erzeugung eines guten Gitters mehrere Stunden bis hin zu mehreren Tagen. Da die Qualität der Lösung und die benötigte Rechenzeit für ein qualitativ besseres Gitter jeweils verbessert wird, lohnt sich der Aufwand in vielen Fällen allerdings.

## 4.3 Aufbau Parameterfile und Preprocessing

Nachdem ein Gitter erstellt wurde, welches für die Rechnung genutzt werden soll, ist es notwendig einzelne Parameter festzulegen, die für die Simulation benötigt werden. Im Fall des Strömungslösers THETA werden diese Parameter mit Hilfe einer Parameterdatei festgehalten. Die Parameterdatei von THETA besitzt viele Einstellungsmöglichkeiten, die im Rahmen dieser Arbeit nicht von Relevanz sind. Für eine genaue Auflistung wird an dieser Stelle auf den UserGuide [DLR16] verwiesen und im folgenden nur einzelne relevante Einstellungen beleuchtet. Ein Beispiel für eine Parameterdatei mit den entsprechenden Einstellungen ist unter Anhang B zu finden. Im Nachfolgenden sind vor allem die Einstellungen hinsichtlich paralleler Verarbeitung, Mehrgitterverfahren, Zeitschrittweite, Residuen und der VOF Methode von Interesse.

Für die parallele Verarbeitung existiert in der Parameterdatei die Einstellung „Number of domains“. Mit Hilfe dieser Einstellung wird im Rahmen des Preprocessings entschieden auf wie viele einzelne Partitionen das Gitter zerlegt wird.

Um das Mehrgitterverfahren mit Hilfe der Parameterdatei einzustellen, muss der Löser für die gewünschte Gleichung umgestellt werden. Ebenfalls wird in der Parameterdatei

festgelegt, wie viele Mehrgitterlevel erzeugt werden sollen, und wie viele Iterationen pro Level vom Löser ausgeführt werden sollen.

Besonders wichtig für die Rechnungen ist der gewählte Zeitschritt. Dabei existieren im Rahmen von THETA zwei unterschiedliche Möglichkeiten zur Festlegung des Zeitschrittes. Eine Möglichkeit ist es einen konstanten Zeitschritt zu wählen. Wenn sich innerhalb der Rechnung die Geschwindigkeiten verändern, verändert sich somit lokal stets die Anzahl der Zellen, die pro Zeitschritt durchflossen werden. Die zweite Möglichkeit ist es eine konstante CFL-Zahl zu wählen. Die CFL-Zahl ist wie in Gleichung (2.2) beschrieben abhängig von Zellbreite, Zeitschrittweite und Geschwindigkeit. Wird die CFL-Zahl konstant gewählt, so variieren im Laufe der Rechnung bei sich verändernder Geschwindigkeit auch die Zeitschrittweiten.

Eine weitere Einstellungsmöglichkeit findet sich in den Epsilon der Residuen. Diese können für die unterschiedlichen zu lösenden Gleichungen jeweils einzeln gesetzt werden. Dabei wird in der Parameterdatei festgelegt, wie viele Größenordnungen das Residuum verkleinert werden soll. Ist diese Verkleinerung erreicht, so wird die Gleichung als ausreichend gelöst angenommen.

Zusätzlich müssen für die VOF-Methode innerhalb der Parameterdatei weitere Kenngrößen des zweiten Mediums festgelegt werden. Die Kenngrößen umfassen hierbei Dichte, Viskosität und auch Oberflächenspannung. Im Rahmen dieser Arbeit wurde an dieser zusätzlich Stelle eine Methode implementiert, die es in Zukunft erleichtert Gebiete zu definieren, in denen sich das zweite Medium zu Beginn der Simulation befindet. Hierzu ist es nun möglich in der Parameterdatei ein zweites Gitter anzugeben, welches den Bereich des Mediums definiert. Über eine Methode, die ermittelt, wo die Überschneidungen zwischen den beiden Gittern liegt [Kap15] werden diejenigen Punkte ermittelt, für die der VOF Wert am Anfang auf 1 gesetzt werden muss. Alle anderen Punkte werden mit VOF Werten von 0 initialisiert.

Sind die wesentlichen Parameter für die Rechnung festgelegt, so kann mit Hilfe dieser Einstellungen zunächst das sogenannte Preprocessing gestartet werden. Bei dem Preprocessing handelt es sich um ein Programm, welches aus den eingestellten Parametern und dem zugehörigen Primärgitter die benötigten Level und Aufteilungen von dualen Gittern erzeugt. Dies bedeutet, dass für die Verwendung eines Mehrgitterverfahrens in THETA nur das feinste Gitter durch den Prozess der Gittererzeugung

erstellt werden muss, alle weiteren Feinheitsgrade werden durch das Preprocessing auf Grundlage dieses Gitters erzeugt.

## 4.4 Vorbereitung und Durchführung der Simulation

Auf Grundlage der vom Preprocessing erzeugten dualen Gitter und den festgelegten Einstellungen in der Parameterdatei kann im nächsten Schritt die Simulation durchgeführt werden. Für den Start der Simulation wird noch der Löser selbst benötigt. Im Rahmen von THETA existiert hierbei die Vorgehensweise, für jeden Testfall einzeln den Löser zu kompilieren. Dies hat den Grund, dass über eine weitere Datei, der user.c, Testfall-spezifische Methoden in den Löser implementiert werden können. Mit Hilfe dieser Methoden können Endnutzer, ohne Änderungen am Code durchführen zu müssen, weitere Implementierungen durchführen, die für eine erfolgreiche Durchführung der einzelnen Rechnungen notwendig sind. Ein Beispiel einer user.c befindet sich in Anhang C, es werden allerdings nur im Rahmen der Arbeit verwendete Funktionen gezeigt.

Im weiteren Verlauf der Arbeit spielen vor allem zwei Methoden aus der user.c eine Rolle: `user_set_initial_values` und `user_callback`. Die erste der beiden Methoden wird aufgerufen, wenn eine Rechnung ohne vorhergehende Lösung gestartet wird. In vielen Fällen wird sie dafür genutzt zu Beginn der Rechnung lokale Werte zu setzen.

Die Methode `user_callback` wird nach jedem absolvierten Zeitschritt ausgeführt. Verwendet werden kann diese Methode beispielsweise für eine spezielle Auswertung nach jedem Zeitschritt, oder um einzelne Werte zu verändern.

Nachdem der Löser mit den implementierten user.c-Routinen kompiliert wurde, kann er auf Basis der dualen Gitter und der Parameterdatei gestartet werden. Für die parallele des Lösers Ausführung auf mehreren Partitionen wird hierbei Message Passing Interface (MPI) verwendet. Eine Simulation kann hierbei mehrere Stunden, Tage bis Wochen dauern. Auf Grund dieses Rechenaufwandes ist eine gründliche Vorbereitung der Testfälle von großer Bedeutung.

## 4.5 Postprocessing

Am Ende einer Rechnung steht die Auswertung der vom Löser produzierten Lösungen. Diese Auswertung kann durch verschiedene Nachbearbeitung unterstützt werden. Im Bereich der Numerischen Strömungsmechanik wird diese Nachbearbeitung auch als Postprocessing bezeichnet. Unter Postprocessing wird hierbei die Umwandlung der Lösungsdaten in anschauliche Graphen und Graphiken verstanden. Des Weiteren können im Postprocessing aus den vorhandenen Datensätzen weitere Strömungsgrößen berechnet und mit Daten aus Experimenten oder anderen Strömungslösern verglichen werden.

Die im folgenden wesentlichen Vorgehensweisen sind die Darstellung relevanter Kenngrößen als Diagramm (x/y-Plot) und die Darstellung von Strömungsgrößen im Feld als Isofläche oder Darstellung auf Slices. Die erste Darstellungsform wird verwendet, um beispielsweise eine Strömungsgröße an einer Stelle über die Zeit oder den Raum zu begutachten. Im Rahmen dieser Arbeit wurde für diese Darstellungsform das Programm Gnuplot [Gnu] genutzt. Ein Vorteil von Gnuplot ist, dass es sich dabei um eine Freeware handelt, die auf vielen Linux-Distributionen bereits vorinstalliert ist.

Die zweite Darstellungsform wird unter anderem dafür verwendet, um Bilder, Animationen und Videos der Strömungssimulation zu erzeugen. Ein Programm, welches für diese Auswertungsform genutzt werden kann, ist das Programm Tecplot [Tec16]. Bei dieser Auswertungsform werden die Strömungsgrößen einer Lösung über das gesamte Rechengebiet abgebildet. Im Rahmen einer Auswertung ist es dabei möglich, Anomalien im Strömungsverlauf zu ermitteln und weiter zu untersuchen. Des Weiteren eignen sich diese Darstellungen besonders für Präsentationen und Vorträge.

# 5 Verifizierung und Validierung der „Volume of Fluid“ Methode

Das folgende Kapitel stellt die Prinzipien aus Kapitel 3 in einem praktischen Umfeld dar. Als Anwendungsbeispiel wird hierbei das Lösungsverfahren der VOF Methode des Strömungslösers THETA verwendet. Ziel ist es zu verifizieren, dass die Entwicklungen seit der Implementierung der VOF Methode die korrekte Berechnung von Zweiphasenströmungen nicht beeinträchtigen. Die Implementierung der VOF Methode wurde im Rahmen der Dissertation von Markus Gauer in THETA integriert [Gau13]. Die erste Version der VOF Methode im Rahmen THETAs wurde allerdings bereits 2009 vorgestellt [GHH09]. Ein bekanntes Problem zum Endpunkt der Entwicklung war es, dass die Kombination eines Mehrgitterverfahrens und der VOF Methode keine konvergenten Ergebnisse erzeugten.

Neue Versuche die beiden Verfahren zu kombinieren haben gezeigt, dass es durch Veränderungen zur Lösung der Druckgleichungen scheinbar möglich ist, Mehrgitterverfahren zur Lösung von Zweiphasenströmungen zu verwenden. Um allerdings herauszufinden, welche Voraussetzungen dafür erfüllt sein müssen, und welche Einstellungen die zuverlässigsten und schnellsten Ergebnisse liefern, ist es notwendig genauere Tests durchzuführen.

## 5.1 Vorbereitung und Auswahl der Testfälle

Gemäß dem ersten Schritt nach 3.2.1 müssen zunächst die Rahmenbedingungen für die Testreihe definiert werden. Ziel der Testreihe ist es festzustellen, ob die VOF Methode grundsätzliche Fehler aufweist. Des Weiteren soll festgestellt werden, ob zusammen mit einem Mehrgitterverfahren identische Ergebnisse produziert und das Mehrgitterverfahren für schnellere Konvergenz sorgt. Zusätzlich soll untersucht

werden, welche Parametereinstellungen zu den besten und schnellsten Ergebnissen führen.

Bezogen auf den V&V Prozess bedeutet dies, dass es sich bei den ersten beiden Zielen, um einen Ansatz von Verifizierung handelt. Allerdings werden viele Aspekte einer reinen Verifizierung nicht betrachtet, da weder statische Analysen durchgeführt werden, noch einzelne Komponenten der Implementierung abstrahiert nach ihren Teilergebnissen kontrolliert werden. Stattdessen werden zur Untersuchung Methoden einer Validierung verwendet, indem die Ergebnisse der neuen Rechnungen mit alten Ergebnissen, Experimenten und vergleichbaren Implementierungen verglichen werden. Ähnliche Methoden werden zur Evaluierung des dritten Zieles verwendet. Zusätzlich sollen allerdings die Rechenzeiten betrachtet werden, um festzustellen, welche Parametereinstellungen für optimalen Rechenaufwand benötigt werden.

Als Testfälle werden zwei unterschiedliche Szenarien betrachtet. Zunächst sollen die Funktionalitäten an einem einfachen theoretischen Testfall kontrolliert werden. Hierzu wird der „Sloshing Tank“ verwendet.

Bei dem „Sloshing Tank“ handelt es sich um einen zweidimensionalen Tank der Ausmaße $0.1m \times 0.1m$, dessen Füllstand am Anfang mit Hilfe einer Kosinusfunktion initialisiert wird. Dieser Testfall ist dabei nur mit theoretisch vorhergesagten Ergebnissen zu vergleichen, da für die Simulation ohne Reibung und lediglich mit der VOF Methode und einem Gravitationsmodell gerechnet werden soll. Für das erste Ziel des V&V Prozesses soll festgestellt werden, ob der Füllstand mehrere Schwingungsperioden lang schwingt, ohne große Abweichungen von den analytischen Ergebnissen. Um festzustellen wie groß die Abweichungen von der Theorie sind, wurden von [RCJ95] zwei Maße definiert (entnommen aus [Gau13, S. 106]):

$$e_{temp} = 100\frac{t_s - t_t}{t_t} \tag{5.1}$$

$$e_{spatial} = \frac{100}{a\sqrt{M}}\left(\sum_{m=1}^{M}(z_s - z_t)^2\right)^{\frac{1}{2}} \tag{5.2}$$

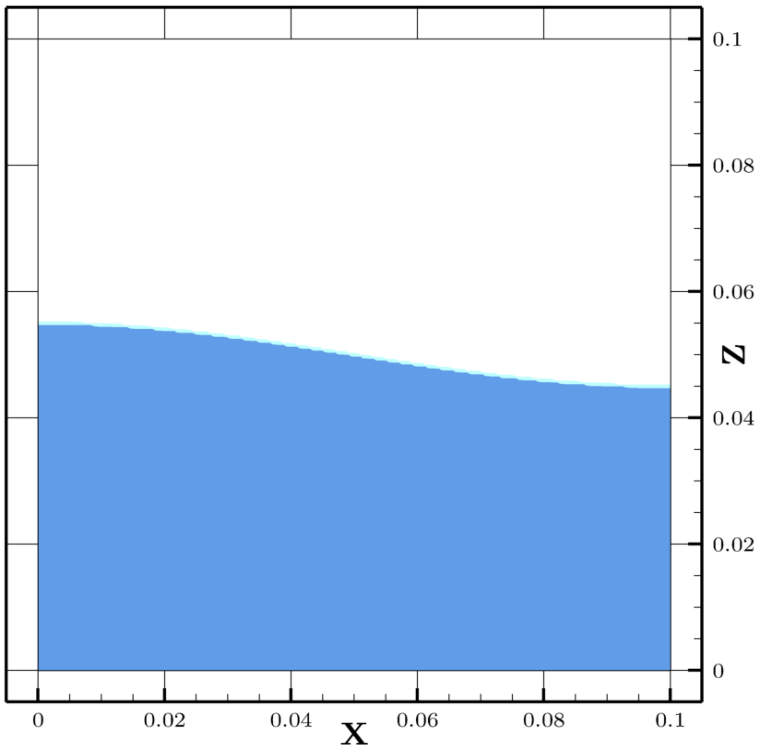



Abbildung 5.1: Aufbau des „Sloshing Tank“Testfalls

Für den zeitlichen Fehler $e_{temp}$ (5.1) wird die Zeit in der Simulation $t_s$ mit der theoretischen Zeit $t_t$ verglichen. Als Zeitpunkte werden hierbei die Amplitudenspitzen an der linken Wand betrachtet. Der räumliche Fehler $e_{spat}$ (5.2) wird über die gesamte Breite des Tanks ausgewertet. Dabei wird davon ausgegangen, dass für die Simulation ein kartesisches Gitter verwendet wird, mit einer festen Zellbreite. Die einzelnen Variablen innerhalb der Funktion beschreiben dann die Anzahl der Zellen in horizontaler Richtung ($M$), die Höhe der Welle auf Basis des initialisierten Interfaces ($a$) und die Höhen der Simulation und der Theorie ($z_s$ und $z_t$). Der räumliche Fehler wird dabei als ein Quadratisches Mittel über die gesamte Breite dargestellt.

Die Größe der Abweichungen sollte dabei im Vergleich zu den Ergebnissen aus der Dissertation von Gauer [Gau13] im gleichen Rahmen liegen. Die zeitlichen Fehler liegen bei Gauer über mehrere Perioden unter 0.5%, während der räumliche Fehler über mehrere Perioden ansteigt, aber stets unter 1% bleibt.

Gauer beschreibt bei diesem Testfall eine Anomalie, die auch in anderen Rechnungen

auftritt, dass bei den ungeraden Perioden die Höhe der theoretisch ermittelten Amplituden überschritten wird [Gau13, S. 105]. Daher werden zur Auswertung lediglich die geraden Perioden kontrolliert und die Fehler berechnet. Bei der Berechnung der Fehlerwerte gibt es zwei Probleme: Zunächst ist es nicht eindeutig, wie genau die Initialisierung der Füllhöhe durchgeführt, und wie die Füllhöhe ausgewertet wurde. Um möglichst genaue Ergebnisse zu erhalten, wurde für die neuen Rechnung die Initialisierung so genau wie möglich durchgeführt. Dies bedeutet, dass beim Übergang von Wasser zu Luft für diejenigen Zellen die nur zu einem Teil mit Wasser gefüllt sind, bei der Initialisierung berechnet wird, wie groß dieser Anteil ist. Eine ungenauere, dafür einfachere Möglichkeit wäre es gewesen, Zellen nur mit 0 und 1 zu initialisieren.

Das zweite Problem ist, dass ebenfalls nicht bekannt ist, wie die Auswertung der Fehlermaße durchgeführt wurde. Für diese gibt es ebenfalls verschiedene Möglichkeiten, von denen zwei im Rahmen dieser Arbeit betrachtet wurden. Die erste Möglichkeit wäre es, die Zellen nach derjenigen Zelle zu durchsuchen, die den Übergang von 0 zu 1 darstellt und durch diesen Übergang und den Anteil von Wasser in derjenigen Zelle zu berechnen, wie Hoch der Füllstand an der Stelle ist. Eine zweite Möglichkeit, die für die Rechnungen im Rahmen dieser Arbeit verwendet wurde, ist es für jede Spalte einzeln alle VOF Werte zu addieren und mit der Zellhöhe zu multiplizieren. Bei dieser Lösungsmöglichkeit wird bereits davon ausgegangen, dass keine Wassertropfen über der eigentlichen Füllhöhe liegen. Ein Vorteil der zweiten Auswertungsmöglichkeit gegenüber der ersten ist es, dass die Berechnung aller Höhen, die für den räumlichen Fehler benötigt werden, über eine einfache Iteration über alle Zellen durchgeführt werden kann. Insgesamt ist allerdings unklar, wie die Auswertung bei den ehemaligen Rechnungen durchgeführt wurden, weswegen es hierbei zu Abweichungen kommen kann.

Mit Hilfe dieses Szenarios können verschiedene Bereiche der Ziele dieser Testreihe abgedeckt werden. In einem ersten Testfall werden die Gegebenheiten der ehemaligen Rechnung so gut wie möglich nachgestellt und die Ergebnisse verglichen. Als zweites wird die gleiche Rechnung nochmals durchgeführt, mit der Änderung, dass ein Mehrgitterverfahren zur Konvergenzbeschleunigung verwendet wird. Zuletzt werden weitere Parametereinstellungen verändert und es wird untersucht, welche Einstellungen optimal für die Lösung dieses Problems sind.

Auf Basis der Ergebnisse wird bei einem erfolgreichen Einsatz des Mehrgitterverfahrens ein weiterer Testfall untersucht. Bei diesem Testfall handelt es sich um die Simulation eines Dammbruchs. Das Rechengitter für diesen Testfall ist ebenfalls angelehnt an Gauer [Gau13]. Dabei handelt es sich um ein grobes, unstrukturiertes Gitter, welches einen Einschnitt in Form eines Container besitzt (Vergleich Abbildung 5.2).

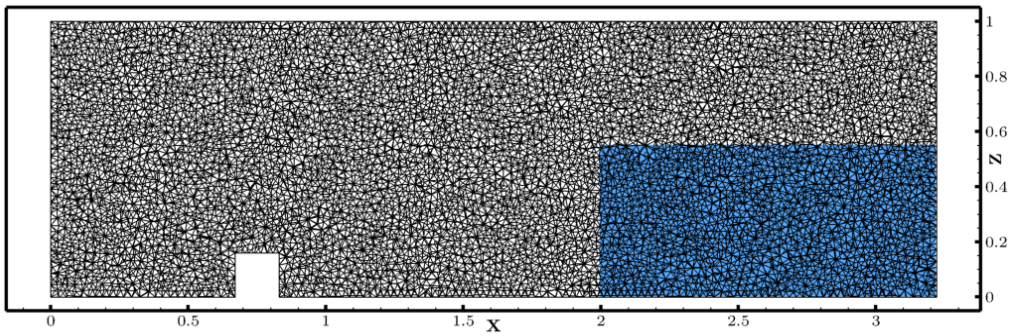



Abbildung 5.2: Zweidimensionaler Schnitt bei Y=0 durch das Gitter für den Dammbruch Testfall

Da das Originalgitter für die Testreihe nicht zur Verfügung stand, wurde ein vergleichbares Gitter mit ähnlicher Anzahl Punkte erzeugt (214.339 zu 211.264 [Gau13, S. 111]). Für die Auswertung werden verschiedene Stellen ausgewählt, für die der Wasserstand kontrolliert wird. Die Stellen wurden dabei auf Basis eines Experimentes des Maritime Research Institute Netherlands (MARIN) ausgewählt [KFV04, S. 338]. Für bestimmte Stellen wurden bei diesem Experiment Höhensensoren angebracht und die Ergebnisse veröffentlicht [Gro16]. Diese Stellen wurden ebenfalls von Gauer untersucht [Gau13, S. 114]. Abweichungen können anhand der Höhenverläufe an diesen Stellen festgestellt werden.

Auf [Gro16] wurden ebenfalls Verläufe zu Drucksensoren an dem Container veröffentlicht. Allerdings liegen hierzu keine Vergleichsdaten von früheren Rechnungen mit THETA vor. Auf Grund fehlender früherer Ergebnisse wird daher im Rahmen dieser Arbeit ebenfalls auf eine Gegenüberstellung der Druckverläufe verzichtet.

Ähnlich wie im ersten Testfall des „Sloshing Tanks“ ist auch bei diesem Testfall ein Problem, dass nicht genau bekannt ist, wie die Auswertung hinsichtlich des Füllstands an den Höhensensoren durchgeführt wurde. Auf Grund der unstrukturierten Topologie

ist die Routine aus dem ersten Testfall hierbei nicht nutzbar. Für die Rechnungen muss daher eine veränderte Auswertungsmöglichkeit entwickelt werden. Hierbei wird im Rahmen des Postprocessings auf eine Möglichkeit zurückgegriffen, Funktionen des Visualisierungsprogrammes Tecplot [Tec16] zu nutzen. Bei den durch diesen Prozess entwickelten Höhen handelt es sich allerdings um interpolierte Werte, die leichte Abweichungen besitzen können.

Mit Hilfe dieses Testfalls werden vor allem die Ziele zwei und drei weiter untersucht. Anhand eines größeren Testfalls kann eine Abschätzung hinsichtlich der Vorteile der Verwendung eines Mehrgitterverfahrens angestellt werden. Des Weiteren werden vor allem weitere Untersuchungen hinsichtlich optimaler Parametereinstellungen am Beispiel eines komplexeren Testfalls durchgeführt. Zuletzt wird durch die Verwendung eines feineren Gitters (mehr als 2 Millionen Punkte) untersucht, inwieweit die VOF Ergebnisse skalieren und durch die Verwendung feinerer Gitter präziser werden.

## 5.2 Durchführung der Testreihe „Sloshing Tank“

Für den ersten Test mussten zunächst die analytischen Vergleichswerte entwickelt werden. Der Füllstand des „Sloshing Tank“ wurde mit Hilfe der Funktion (5.3) initialisiert.

$$z(x) = 0.05m + 0.005 \cdot cos(\frac{\pi \cdot x}{0.1})m \tag{5.3}$$

Nach einer vollständigen Periode, die nach Gauer [Gau13, S. 104] eine Periodendauer $T_1 = 0.373723$ besitzt, sollte dieser Zustand wieder eingetroffen sein. Da lediglich an diesen Stellen ausgewertet wird, wie groß der räumliche Fehler ist, reicht es für den Testfall stets aus, mit dieser Verteilung zu vergleichen. Die Berechnung des räumlichen Fehlers wird somit nach jeden Zeitschritt durch einen Vergleich des simulierten Höhenverlaufs mit dem Verlauf nach der Formel (5.3) durchgeführt.

Für die Berechnung des zeitlichen Fehlers wurde ein Skript entwickelt, welches auf Basis der Ergebnisse an der linken Wand bestimmt, zu welchem Zeitpunkt das Maximum erreicht worden ist, und somit eine weitere Periode abgeschlossen ist. Das

Skript läuft hierbei auf Wanddaten, die extra nach jedem Zeitschritt mitsamt des räumlichen Fehlers durch den Löser herausgeschrieben werden. Diese Ausgabe wurde durch eine Implementierung in der user.c realisiert.

Die Einstellungen für den ersten Test können aus Tabelle 5.1 entnommen werden. Das Diskretisierungsschema Upwind Difference Scheme (UDS) wird hierbei zunächst dem Quadratic Upwind Difference Scheme (QUDS) vorgezogen, da UDS im Rahmen von Gauers Dissertation verwendet wurde. Bei QUDS handelt es sich um eine erweiterte räumliche Diskretisierung von UDS. Auf die genauen Eigenschaften der beiden unterschiedlichen Schema, wird an dieser Stelle nicht weiter eingegangen und auf Perić [PF02, S.76ff] verwiesen, da weitere Details für das Verständnis dieser Arbeit nicht von großer Relevanz sind.

| | |
|---|---|
| **Netzfeinheit** | 160 Zellen |
| **Zeitschrittweite** | 0.0007432 ($\sim$ 0.2 CFL) |
| **Mehrgitter?** | Nein |
| **Residuum** | 1e-3 |
| **Diskretisierung** | UDS |

Tabelle 5.1: Übersicht wesentlicher Einstellungen des ersten Tests „Sloshing Tank“

Mit Hilfe dieser Einstellungen wird das Ergebnis aus Grafik 5.3 erzeugt. In der Grafik ist hierbei die Füllhöhe des Tanks an der linken Wand gegenüber der Zeit aufgetragen.

Ähnlich wie bereits von Gauer [Gau13] erläutert sind die Anomalien in den ungeraden Schwingungen zu erkennen. Bei den geraden Schwingungen sind ebenfalls Abweichungen zu erkennen, allerdings sind diese bis zur letzten Periode verhältnismäßig kleiner. Vor allem der zeitliche Verlauf wird allerdings global gut getroffen. Die in 5.1 beschriebenen Auswertungsmaße sind in Tabelle 5.2 mit Werten von der früheren Rechnung verglichen. Die Werte von Raad [RCJ95], Ubbink [Ubb97] und Gauer wurden aus der Dissertation [Gau13, S.107] entnommen.

Auffällig ist bei den Werten der neuen Rechnung, dass bei den zeitlichen Fehlern eine steigende Tendenz zu erkennen ist. Dieses Muster ist bei keinem der anderen Rechnungen zu beobachten. Allerdings ist der zeitliche Fehler auch noch nach zehn

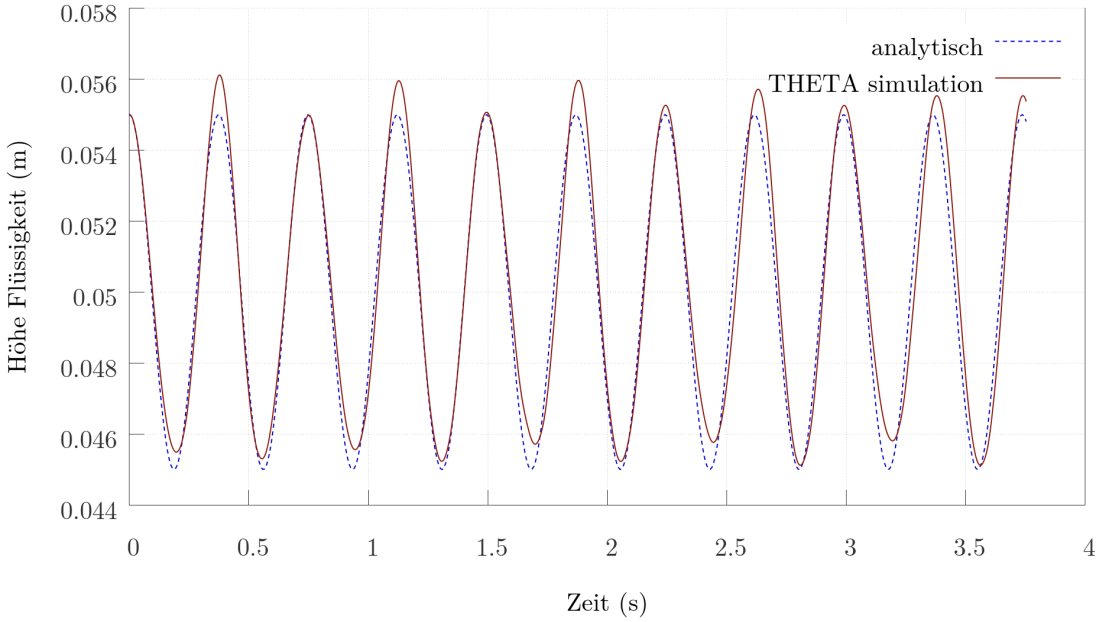



Abbildung 5.3: Vergleich analytische Lösung und Rechnung mit THETA

Perioden sehr gering ($< 0.5\%$). Insgesamt sieht der zeitliche Verlauf somit akzeptabel aus.

Anders ist es beim räumlichen Fehler. Die räumlichen Fehler verlaufen hierbei ähnlich wie bei den früheren Rechnungen mit einer steigenden Tendenz ab. Im direkten Vergleich mit der Rechnung von Gauer, dessen Implementierung für die neue Rechnung verwendet wurde, fällt auf, dass sich die Fehler um ein Vielfaches unterscheiden. Während die frühere Rechnung auch nach zehn Perioden noch unter einem Prozent lag (0.78%), ist der räumliche Fehler bei der neuen Rechnung bereits eine Größenordnung größer (7.68%).

| **Periode** | **Zeitlicher Fehler [%]** | | | | **Räumlicher Fehler [%]** | | | |
|---|---|---|---|---|---|---|---|---|
| | Raad | Ubbink | Gauer | **THETA** | Raad | Ubbink | Gauer | **THETA** |
| **2** | -0.44 | 0.00 | 0.15 | **0.18** | 2.0 | 1.9 | 0.36 | **1.70** |
| **4** | -0.39 | -0.75 | 0.07 | **0.08** | 2.8 | 2.0 | 0.57 | **2.52** |
| **6** | -0.29 | 0.04 | 0.01 | **0.16** | 3.5 | 3.7 | 0.65 | **4.00** |
| **8** | n/a | n/a | 0.00 | **0.15** | n/a | n/a | 0.72 | **6.00** |
| **10** | n/a | n/a | 0.00 | **0.20** | n/a | n/a | 0.78 | **7.68** |

Tabelle 5.2: Vergleich Zeitlicher und Räumlicher Fehler mit früheren Rechnungen

Der räumliche Fehler liegt somit deutlich außerhalb des anfangs festgelegten Akzeptanzrahmens, weswegen weitere Untersuchungen notwendig sind, welche Ursachen dieser Fehler besitzt. Wird Abbildung 5.3 etwas genauer mit Abbildungen aus der Dissertation von Gauer verglichen, sind keine großen Diskrepanzen zu erkennen. Dies legt den Schluss nahe, dass an der Auswertungsmethodik etwas nicht stimmt. Für die Auswertung wurde angenommen, dass der räumliche Fehler durch den Vergleich der theoretischen Spitze an der linken Wand mit der simulierten Spitze berechnet wird. Somit wird der zeitliche Fehler bei der Berechnung des räumlichen Fehlers so gut wie möglich ausgeschlossen. An dieser Stelle könnte eine mögliche Fehlerursache liegen, allerdings liegen nicht genügend Daten vor, um sicher zu sagen, ob ein Implementierungs- oder Auswertungsfehler vorliegt. Für den weiteren Verlauf der Testfälle wird davon ausgegangen, dass der Fehler innerhalb der Auswertungsmethodik lag und für die Ermittlung optimaler Parameterbreiten nur noch die neuen Rechnungen untereinander verglichen. Die Auswertungsmethodik wird an dieser Stelle nicht überarbeitet, da die räumlichen Fehler im Rahmen der von [RCJ95] und [Ubb97] ermittelten Werte liegt. Daher erscheint es, dass die Auswertungsmethodik nicht ganz mit der von Gauer übereinstimmt, allerdings näher an der von den anderen beiden Autoren liegt.

Für die Ermittlung optimaler Parameter wurde für die nachfolgenden Testfälle des ersten Testszenarios eine Gliederung erarbeitet: Zunächst wird über die Verwendung verschiedener Zeitschrittweiten ausgewertet, welche Zeitschrittweite für ein gutes Ergebnis bei geringem Rechenaufwand erzeugt. Im nächsten Schritt wird über eine Reihe von verschiedenen Residuen festgestellt, wie genau in diesem Testfall gearbeitet werden muss. Auch dieser Parameter ist direkt an den Rechenaufwand gekoppelt. Zuletzt soll festgestellt werden, welches der beiden Diskretisierungsverfahren UDS und QUDS besser geeignet ist.

| **Periode** | **c = 0.05** | **c = 0.1** | **c = 0.2** | **c = 0.4** | **c = 0.6** |
|---|---|---|---|---|---|
| **2** | 0.24 | 0.20 | 0.13 | 0.10 | 0.50 |
| **4** | 0.05 | 0.07 | 0.08 | 0.05 | 0.13 |
| **6** | 0.14 | 0.15 | 0.16 | 0.13 | 0.20 |
| **8** | 0.12 | 0.13 | 0.14 | 0.15 | 0.20 |
| **10** | 0.17 | 0.18 | 0.19 | 0.18 | 0.20 |

Tabelle 5.3: Zeitliche Fehler (in %) verschiedener CFL-Zahlen

| Periode | c = 0.05 | c = 0.1 | c = 0.2 | c = 0.4 | c = 0.6 |
|---|---|---|---|---|---|
| **2** | 1.74 | 1.74 | 1.70 | 1.61 | 1.49 |
| **4** | 2.47 | 2.49 | 2.52 | 2.55 | 2.28 |
| **6** | 3.92 | 4.00 | 4.11 | 4.20 | 3.83 |
| **8** | 5.99 | 6.05 | 6.11 | 6.13 | 5.77 |
| **10** | 7.68 | 7.77 | 7.66 | 7.70 | 7.62 |

Tabelle 5.4: Räumliche Fehler (in %) verschiedener CFL-Zahlen

In den Tabellen 5.3 und 5.4 sind die zeitlichen und räumlichen Fehler für verschiedene CFL-Zahlen aufgetragen. Ergebnisse für CFL-Zahl c = 0.8 sind nicht eingetragen, obwohl sie getestet worden ist, da hierfür keine Konvergenz erreicht werden konnte. Bemerkenswert ist, dass weder bei den zeitlichen Fehlern noch bei den räumlichen Fehlern große Abweichungen untereinander festzustellen sind. Gauer empfiehlt in seiner Dissertation keine CFL-Werte über 0.2 zu verwenden, dies kann aus diesen Testdaten allerdings nicht weiter erschlossen werden. Die hier angegebenen CFL-Zahlen bestimmen allerdings die Maximalwerte im Verlauf der Rechnung. Die Durchschnittswerte liegen jeweils deutlich tiefer. Da allerdings bereits ein einziger Zeitschritt mit einer CFL-Zahl $> 1$ die Rechnung divergieren lässt, ist diese Art der Betrachtung in vielen Fällen sinnvoll.

Um festzustellen, welche CFL-Zahl zuverlässige Ergebnisse bringt und dabei effizient agiert muss zusätzlich auch der Rechenaufwand betrachtet werden.

In Abbildung 5.4 werden die verschiedenen CFL-Zahlen und die CPU-Zeiten doppelt logarithmisch dargestellt. Es ist zu erkennen, dass die CPU-Zeiten direkt mit der CFL-Zahl zusammenhängen. Eine Verdopplung der CFL-Zahl sorgt somit direkt für die Verdopplung des Rechenaufwandes. Auf Basis der vorhergehenden Ergebnisse und weiteren Testfällen bezüglich der Schärfe des Interfaces von Gauer [Gau13, S. 80ff] wird eine CFL-Zahl von maximal 0.3 empfohlen. Niedrigere CFL-Zahlen können hierbei zu besseren Ergebnissen führen, steigern allerdings den Rechenaufwand deutlich. Für die weiteren Testfälle wird eine CFL-Zahl von 0.2 verwendet.

Ähnlich wie in der vorhergehenden Testreihe werden in den Tabellen 5.5 und 5.6 die zeitlichen und räumlichen Fehler für unterschiedliche Residuen gezeigt. Zu beachten ist, dass ein Residuum von 1e-2 hierbei eine ungenauere Abbruchbedingung darstellt,

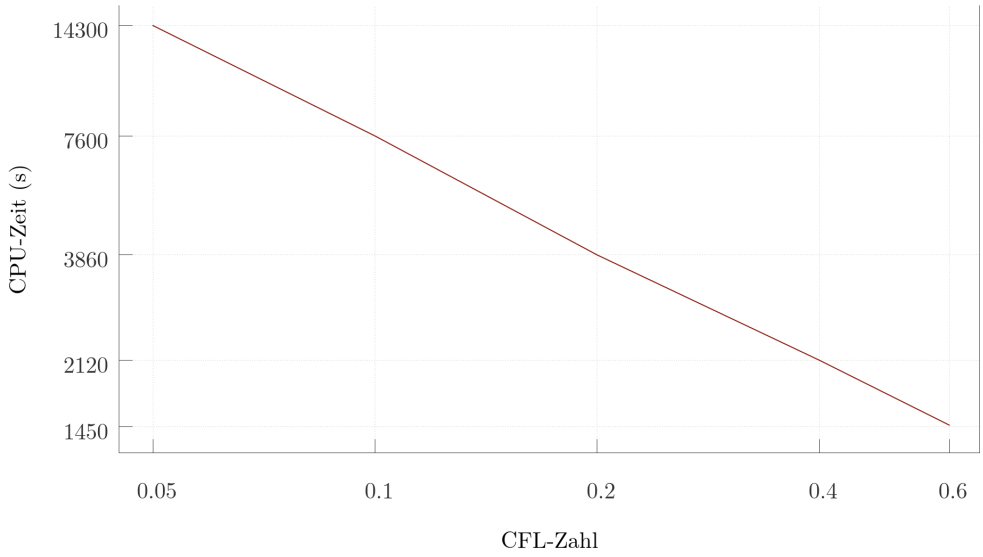



Abbildung 5.4: Rechenaufwand verschiedener CFL-Zahlen

| **Periode** | **r = 1e-2** | **r = 1e-3** | **r = 1e-4** | **r = 1e-5** | **r = 1e-6** |
|---|---|---|---|---|---|
| **2** | 0.18 | 0.18 | 0.18 | 0.13 | 0.13 |
| **4** | 0.13 | 0.08 | 0.08 | 0.08 | 0.08 |
| **6** | 0.19 | 0.16 | 0.14 | 0.16 | 0.16 |
| **8** | 0.19 | 0.15 | 0.14 | 0.14 | 0.14 |
| **10** | 0.24 | 0.20 | 0.19 | 0.19 | 0.19 |

Tabelle 5.5: Zeitliche Fehler (in %) verschiedener Residuen

| **Periode** | **r = 1e-2** | **r = 1e-3** | **r = 1e-4** | **r = 1e-5** | **r = 1e-6** |
|---|---|---|---|---|---|
| **2** | 1.72 | 1.70 | 1.70 | 1.70 | 1.70 |
| **4** | 2.42 | 2.52 | 2.52 | 2.52 | 2.52 |
| **6** | 4.00 | 4.00 | 4.11 | 4.11 | 4.11 |
| **8** | 6.06 | 6.00 | 6.11 | 6.11 | 6.11 |
| **10** | 7.71 | 7.68 | 7.78 | 7.66 | 7.66 |

Tabelle 5.6: Räumliche Fehler (in %) verschiedener Residuen

während 1e-6 weitere Lösungsiterationen fordert. Mit diesem Residuum wird die Größenordnung festgelegt, um die das Residuum beim Lösen einer Gleichung im Netz verringert werden muss, damit der Zeitschritt als abgeschlossen betrachtet wird. Wird der zeitliche Fehler betrachtet, so besitzt das Residuum 1e-3 bereits minimale

Abweichungen von den genaueren Berechnungen und 1e-2 im Verhältnis bereits große Abweichungen. Die räumlichen Fehler zeigen dagegen keine erkennbare Tendenz auf.

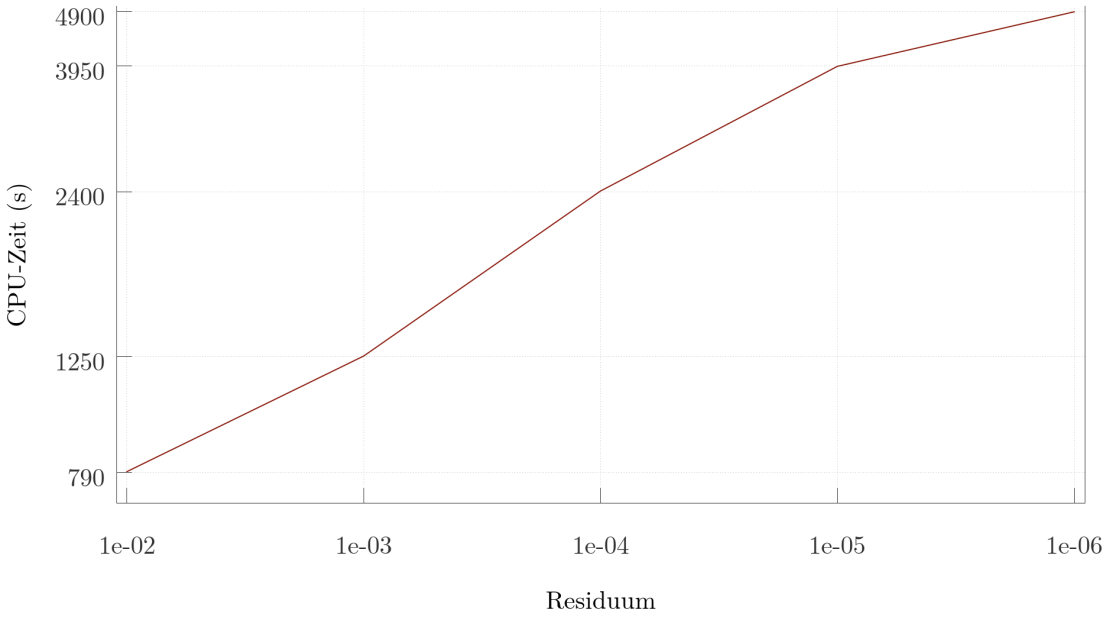



Abbildung 5.5: Rechenaufwand verschiedener Residuen

In Abbildung 5.5 wird wie in der Testreihe vorher der Rechenaufwand der einzelnen betrachteten Residuen doppelt logarithmisch aufgezeigt. Auch hier ist deutlich erkennbar, dass ein größerer Genauigkeitsgrad direkt mit dem Rechenaufwand verbunden ist. Die Ergebnisse hinsichtlich des zeitlichen Fehlers und des Rechenaufwandes legen nahe, dass kein Residuenfaktor größer 1e-3 verwendet sollte, da die Ergebnisse von 1e-2 im Verhältnis größere Abweichungen aufweisen. In diesem Testfall reicht bereits 1e-5 für das genaueste Ergebnis aus, besitzt im Verhältnis zu dem minimal schlechteren Ergebnis allerdings ein Vielfaches an Rechenzeit. Für die weiteren Testfälle wird ein Residuumsfaktor von 1e-3 verwendet.

Als letzte Testreihe im Bereich der optimalen Parametereinstellungen werden die beiden Diskretisierungsmöglichkeiten verglichen. In Tabelle 5.7 sind die zeitlichen Fehler eingetragen. Auffällig ist, dass ab Periode vier QUDS stetig bessere Ergebnisse produziert, zu Beginn jedoch eine deutlich größere Abweichung besitzt. Auf Grund der besseren Tendenz werden die letzten Tests dieses Testfalls mit QUDS durchge-

| Periode | UDS | QUDS |
|---|---|---|
| **2** | 0.18 | 0.48 |
| **4** | 0.08 | 0.00 |
| **6** | 0.16 | 0.09 |
| **8** | 0.15 | 0.07 |
| **10** | 0.20 | 0.08 |

Tabelle 5.7: Vergleich räumliche Diskretisierungen UDS und QUDS

führt, eine allgemeine Empfehlung kann hier jedoch nicht festgelegt werden. Weitere Untersuchungen zur Diskretisierung sollen daher im zweiten Testfall stattfinden.

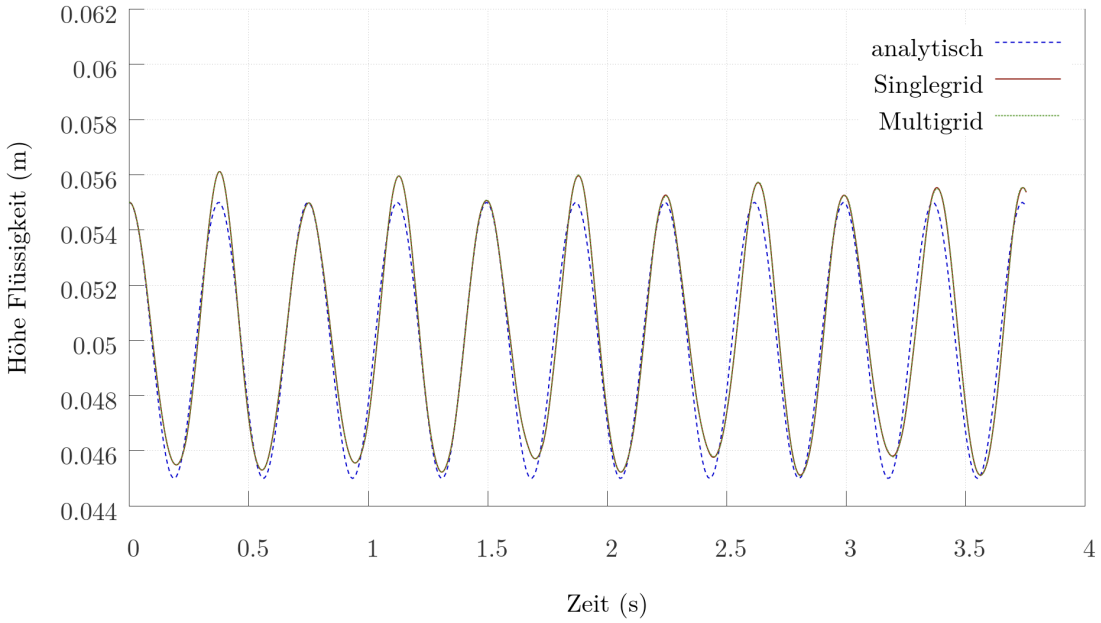



Abbildung 5.6: Vergleich Mehrgitterverfahren mit Rechnung auf einem Gitter

Ein wesentlicher Aspekt der Validierung ist es sicherzustellen, dass ein Mehrgitterverfahren zur Konvergenzbeschleunigung verwendet werden kann. In Abbildung 5.6 ist der zeitliche Verlauf der Rechnung mit und ohne einem Mehrgitterverfahren (Multigrid) gezeigt. Die Verläufe liegen direkt aufeinander, weswegen feststeht, dass ein die Verwendung eines Mehrgitterverfahrens keine negativen Seiteneffekte zu besitzen scheint. Die Rechnung ohne Mehrgitter hat etwa 1600 Sekunden gerechnet, während die Rechnung mit Mehrgitter etwa 1800 Sekunden gerechnet hat. Dieses Ergebnis ist zunächst negativ zu bewerten, ist jedoch dadurch bedingt, dass es sich bei „Sloshing

Tank“um einen einfachen zweidimensionalen Testfall handelt. Die Iterationen des Mehrgitterverfahrens sind einzeln rechenaufwändiger als die ohne dieses Verfahren, können dafür aber schneller zu einer Konvergenz führen. Dieser kleine Testfall erreicht die Konvergenz allerdings auch ohne Mehrgitterverfahren in wenigen Iterationen, weswegen sich der Einsatz in speziell diesem Testfall nicht lohnt. Allgemein wurde sichergestellt, dass keine Seiteneffekte auftreten und eine Konvergenz erreicht werden konnte.

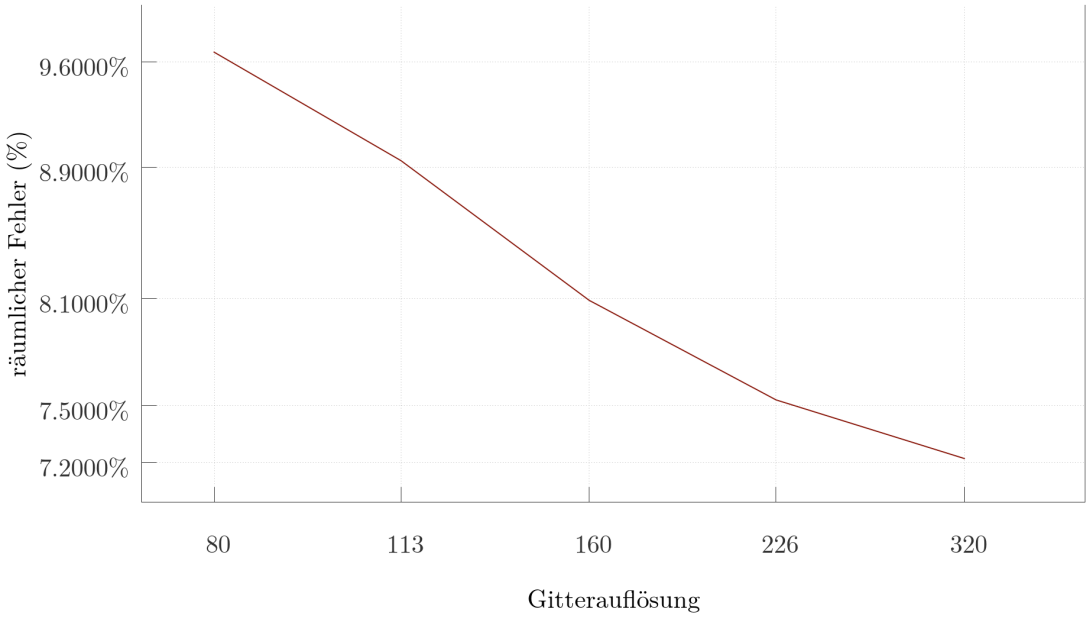



Abbildung 5.7: Räumlicher Fehler verschiedener Gitterauflösungen nach zehn Perioden

Abschließend wird noch das Skalierungsverhalten der räumlichen Diskretisierung betrachtet. Hierzu wurden verschiedene Gitterauflösungen mit identische Parametereinstellungen simuliert. Der Verlauf des räumlichen Fehlers ist in Abbildung 5.7 zu sehen. Zu erkennen ist, dass der räumliche Fehler, wie zu erwarten mit der Genauigkeit der räumlichen Diskretisierung skaliert. Allerdings ist die Verbesserung geringer als erwartet worden ist. Bei einem vierfach feineren Gitter konnte lediglich eine absolute Einsparung von 2,4% erreicht werden, was einer relativen Verbesserung von 25% entspricht. Der höhere Rechenaufwand eines feineren Rechennetzes bewirkt zwar eine Verbesserung der Genauigkeit, diese fällt allerdings im Hinblick auf den räumlichen Fehler der VOF Methode verhältnismäßig gering aus.

## 5.3 Durchführung der Testreihe „3D Dammbruch“

Im Rahmen des zweiten Testfalls, dem simulierten Dammbruch, wird die Auswertung der Wasserstände an zwei unterschiedlichen Stellen untersucht. Die erste Stelle (H2) befindet sich direkt vor dem Container, während die zweite Stelle (H4) zu Beginn in der Wassersäule steht (siehe Abbildung 5.8).

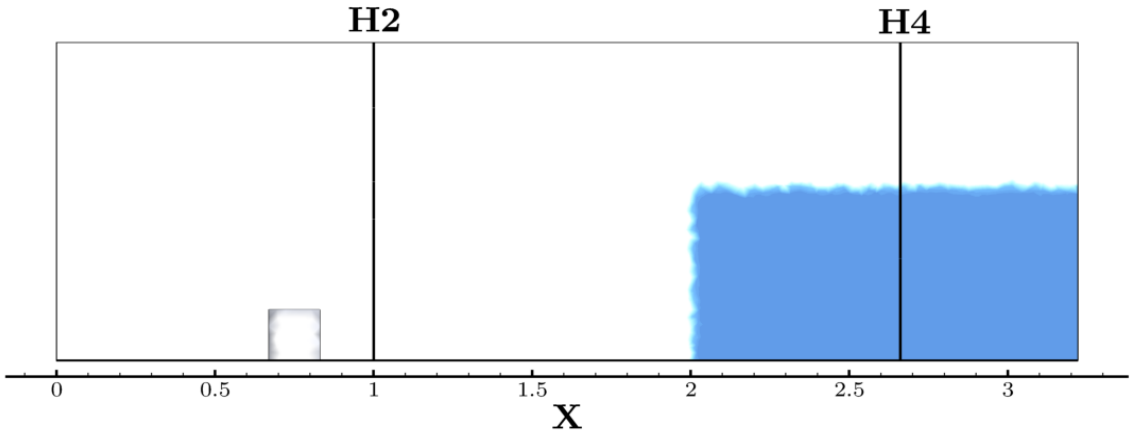



Abbildung 5.8: Positionen H2 und H4 (Y = 0)

Die Initialisierung wurde für diesen Testfall mit der neu entwickelten Methode durch ein weiteres Gitter durchgeführt. Die Auswertung der Wasserstände mit der Zeit wird über das Programm Tecplot realisiert. Dazu werden Slices an der entsprechenden x- bzw. y-Koordinate gezogen und die dadurch entstehende Schnittgerade betrachtet. An dieser Gerade wird nach einem Interfacewert von 0.5 gesucht. Diese Auswertungsart besitzt den Nachteil, dass es sich um interpolierte Werte handelt, die leichte Abweichungen von den realen Zuständen besitzen können. Des Weiteren ist es fraglich, wie der Fall, dass auf einer Gerade der Wert mehrmals auftritt, behandelt werden soll. Ein solcher Fall tritt auf, wenn von einer rücklaufenden Wasserwelle Luft eingeschlossen wird. Diese Fragestellung tritt allerdings auch bei jeder anderen Auswertungsmöglichkeit auf. In diesem Fall wurde sich auf Grund fehlender Informationen dafür entschieden die niedrigsten Werte zu verwenden, da es sich bei den höheren auch um größere Wassertropfen handeln könnte.

| | |
|---|---|
| **Zeitschrittweite** | 0.0001 (bis t = 1s) / 0.0002 (ab t = 1s) |
| **Mehrgitter?** | Ja |
| **Residuum** | 1e-5 |
| **Diskretisierung** | QUDS |

Tabelle 5.8: Einstellungen für den ersten Test des Dammbruch Testfalls

Zunächst wird ein erster Testfall durchgeführt, der die Erkenntnisse des ersten Testfalls nutzt. Dabei werden die Parameter aus Tabelle 5.8 genutzt.

Die Zeitschrittweite wurde anhand vorhergehender Probeläufe ermittelt und so festgelegt, dass zu keinem Zeitpunkt eine CFL-Zahl von 0.2 überschritten wird. Dafür wird nach der ersten simulierten Sekunde ein Restart durchgeführt und die Zeitschrittweite erhöht, da die Geschwindigkeiten geringer werden. Durch dieses Verfahren wird ein möglichst geringer Rechenaufwand sichergestellt.

Die Ergebnisse werden in Abbildung 5.9 mit den Experimentdaten von [Gro16] verglichen. Die Verwendung der Testdaten im Rahmen dieser Arbeit wurde von den Autoren der Veröffentlichung genehmigt. Ein Vergleich mit den früheren Rechnungen von Gauer war nicht möglich, da die Testdaten nicht vorlagen.

Der direkte Vergleich zeigt, dass der Verlauf getroffen wird und demnach keine schwerwiegenden Fehler im Modell vorliegen. Allerdings ist zu erkennen, dass die Rechnung gegenüber dem Experiment schneller läuft. Des Weiteren ist im Verlauf von H2 ab t = 1.55s zu erkennen, dass die Turbulenzen, die der Aufprall des Wassers erzeugt, zu deutlichen Abweichungen vom experimentellen Verlauf führen.

Um diese Abweichungen weiter zu untersuchen, werden in Abbildung 5.10 das Experiment und die simulierte Lösung zu drei unterschiedlichen Zeitpunkten gegenübergestellt. Betrachtet wird ein Zeitpunkt vor dem Aufprall des Wassers auf den Container (t = 0.4s), ein Zeitpunkt innerhalb der turbulenten Phase (t = 1.675s) und ein Zeitpunkt nachdem die Welle bereits zurückgelaufen ist (t = 4.3s). In den Vergleichen ist zu erkennen, dass die einzelnen Zustände gut erfasst werden. Die Wellenform sieht vor dem Aufprall auf den Container sehr ähnlich aus, die Turbulenzen werden nach dem Aufprall gut dargestellt und zum Zeitpunkt t = 4.3s wird ebenfalls erfasst, dass sich kein Wasser auf dem Container befindet. Insgesamt ist die räumliche

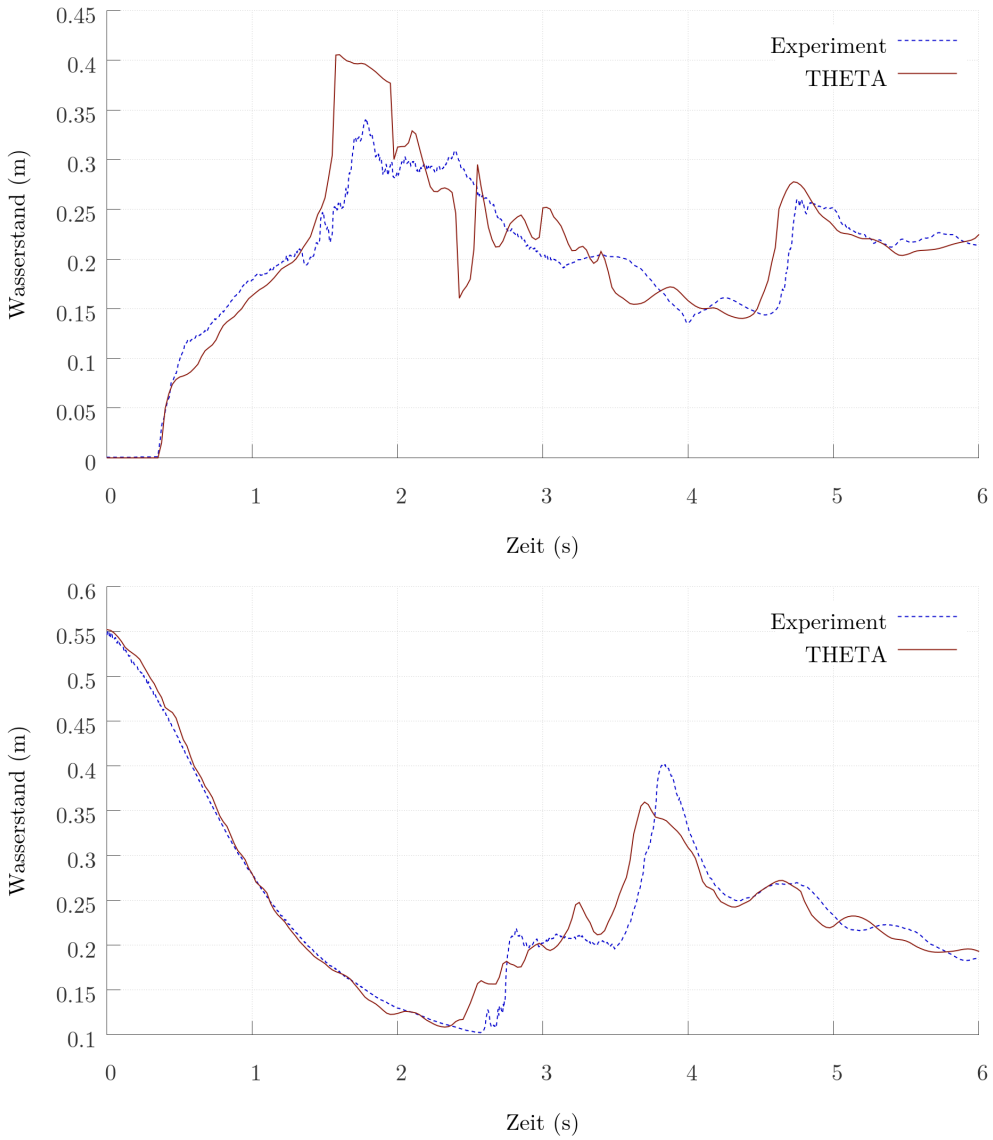



Abbildung 5.9: Zeitliche Verläufe Dammbruch an Positionen H2 (oben) und H4 (unten)

Erfassung somit ausreichend genau. Bei der zeitlichen Erfassung konnten in den Verläufen vorher Abweichungen erkannt werden. Diese sind bei diesen Vergleichen schwer zu erkennen, da das Bild des Experimentes in zwei Teile geteilt ist. Allerdings ist im letzten Bild schwach zu erkennen, dass die Welle der Simulation etwas weiter vorangeschritten ist.

Auf Grund dieser zeitlichen Verschiebung, werden als nächster Test die Auswirkungen

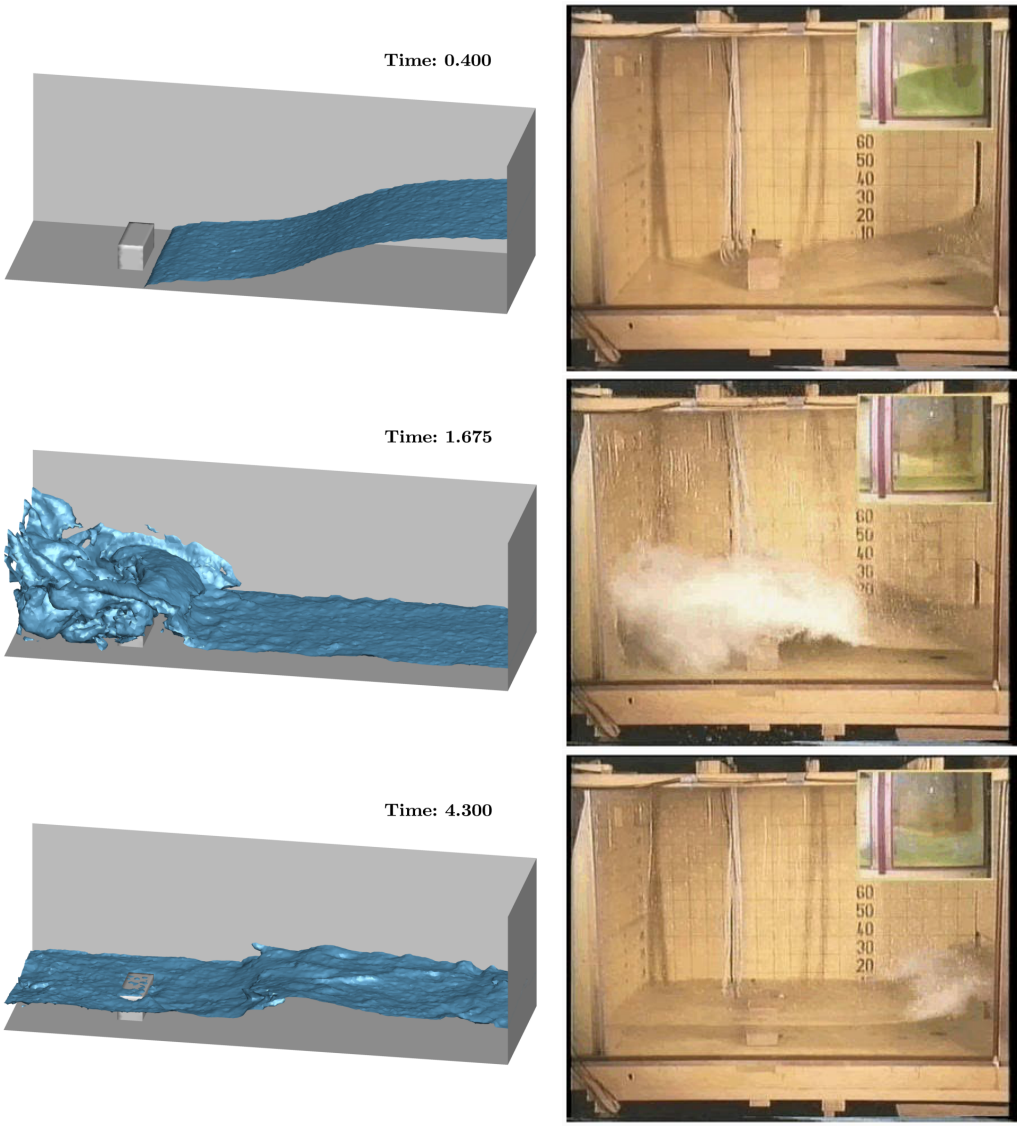



Abbildung 5.10: Dammbruchexperiment zu verschiedenen Zeitpunkten

der Zeitschrittweite ausgewertet. Unter Beachtung der Turbulenzen in H2 und der damit verbundenen Unübersichtlichkeit in einem Plot mit mehreren Verläufen wird zur Darstellung dieses Testfalls nur der Verlauf an der Position H4 genutzt. Der Verlauf ist in Abbildung 5.11 zu betrachten.

Wie im Vorfeld zu erwarten war, sorgt eine kleinere Zeitschrittweite für eine bessere Erfassung des zeitlichen Verlaufs. Beim Peak bei t = 2.8s ist der Verlauf mit der kleinsten Zeitschrittweite der einzige, der den Verlauf tatsächlich erfässt. Bei dem

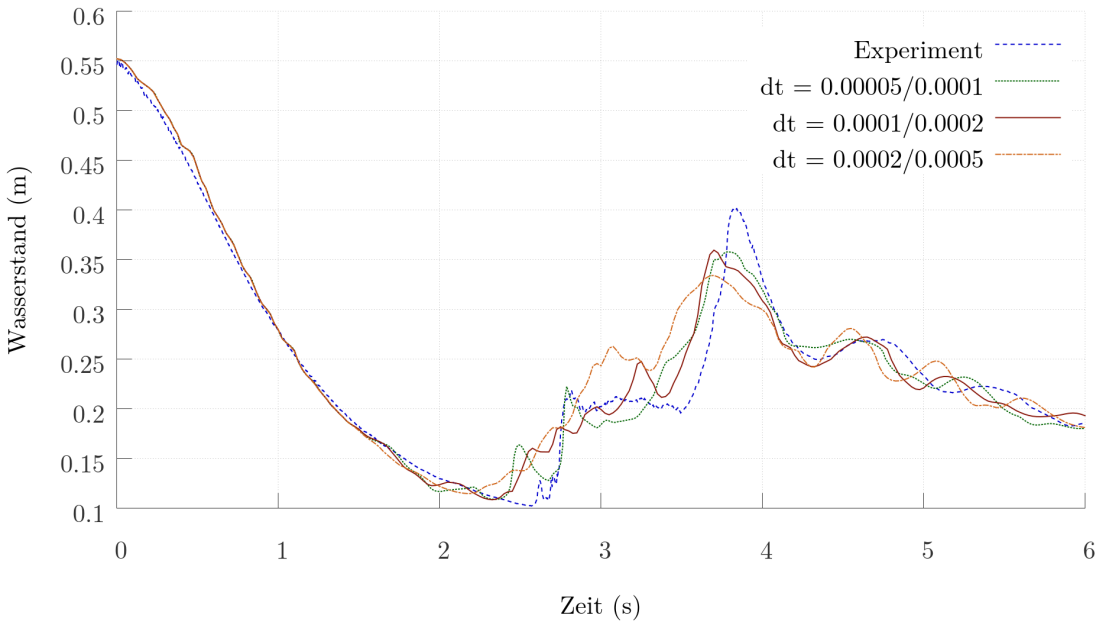



Abbildung 5.11: Zeitliche Verläufe verschiedener Zeitschrittweiten an Position H4

größeren Peak bei t = 3.85s sind die Unterschiede zwischen den beiden kleineren Zeitschrittweiten vernachlässigbar, während die gröbste Zeitschrittweite deutliche Abweichungen zeigt. Die mittlere Zeitschrittweite ist mit einer maximalen CFL-Zahl von 0.2 verbunden. Für den Verlauf reicht somit eine Zeitschrittweite, die für CFL-Zahlen unter 0.2 sorgt. Genauere Lösungen werden in komplexeren Fällen allerdings dennoch durch genauere Zeitschrittweiten erzeugt.

Nach der Untersuchung der zeitlichen Diskretisierung, wurde über ein doppelt so feines Gitter (ca. 8 mal so viele Punkte) kontrolliert, dass auch bei diesem komplexeren Testfall die Skalierung der VOF Methode funktioniert. Bei den Einstellungen wurde sich an den ersten Einstellungen orientiert, da der Rechenaufwand für eine kleinere Zeitschrittweiten deutlich größer wäre. Die Einstellungen für diesen Testfall sind aus Tabelle 5.9 zu entnehmen.

Um vergleichbare Ergebnisse zu erhalten muss auf Grund der doppelten Feinheit des Gitters in jede Dimension auch der zeitliche Schritt halbiert werden, um vergleichbare CFL-Zahlen zu erhalten.

In Abbildung 5.12 ist zu erkennen, dass auch beim feinen Gitter Abweichungen vom experimentellen Verlauf existieren. Der Verlauf an der Stelle H2 zeigt, dass die

| | |
|---|---|
| **Zeitschrittweite** | 0.00005 (bis t = 1.5s) / 0.0001 (ab t = 1.5s) |
| **Mehrgitter?** | Ja |
| **Residuum** | 1e-5 |
| **Diskretisierung** | QUDS |

Tabelle 5.9: Einstellungen für den Test des feinen Gitters des Dammbruch Testfalls

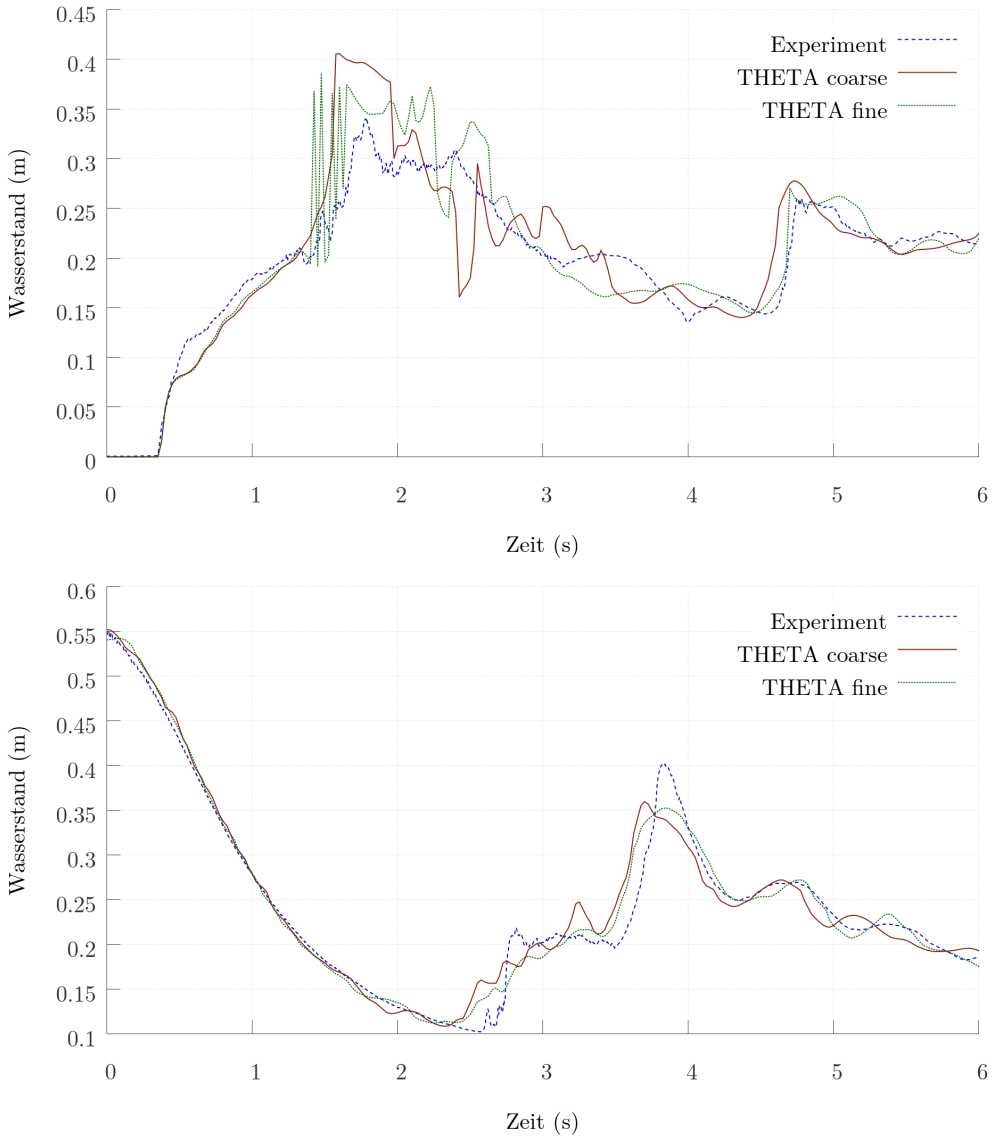



Abbildung 5.12: Vergleich Zeitliche Verläufe grobes und feines Gitter (oben H2, unten H4)

feinere räumliche Diskretisierung die Turbulenzen beim Aufprall des Wassers auf den Container stärker kennzeichnet und an der rücklaufenden Welle (t = 4.75s) den zeitlichen Verlauf genauer darstellt. Die Verbesserung des zeitlichen Verlaufs ist auch an der Stelle H4 zu erkennen. Die Spitzen der Wellen werden hierbei zeitlich genauer dargestellt, als bei der Rechnung auf dem gröberen Gitter. Anhand der gezeigten Ergebnisse kann festgestellt werden, dass die genauere räumliche Diskretisierung für verbesserte Ergebnisse sorgt.

Wie bereits im vorherigen Testfall, wurde erneut ausgewertet, inwieweit ein Mehrgitterverfahren den Rechenaufwand verringern kann. Hierfür wurden zwei Rechnungen mit identischen Einstellungen auf dem gröberen Gitter durchgeführt. Die Ergebnisse der beiden Rechnungen waren identisch, weswegen sie an dieser Stelle nicht aufgeführt werden. Die Rechenzeiten der beiden Simulationen sind in Tabelle 5.10 zu sehen. Wie deutlich zu erkennen ist, benötigt die Rechnung ohne Mehrgitterverfahren fast 1.5 mal so viel Zeit, um das identische Ergebnis zu produzieren.

| | |
|---|---|
| **Mit Mehrgitterverfahren** | 103300s |
| **Ohne Mehrgitterverfahren** | 144360s |

Tabelle 5.10: Rechenzeiten mit und ohne Mehrgitterverfahren

Im Rahmen der vorhergehenden Testfälle erfolgte zusätzlich eine Betrachtung der Dichte. Bei diesen Betrachtungen ist ein möglicher Fehler aufgefallen, der weiterer Untersuchung benötigt. In Abbildung 5.13 ist dieser mögliche Fehler dargestellt.

Die genannte Unstimmigkeit ist auf der linken Seite zu finden. Dort liegt die Dichte unter dem Interface bei etwa 600 (grüne Fläche). Die festgelegten Dichten von Luft und Wasser liegen bei etwa 1 zu 1000. Somit wird von der Simulation zu diesem Zeitpunkt ein sehr starkes Mischungsverhältnis der beiden Medien durch die Dichte vorausgesagt. Zu dem Zeitpunkt t = 5.85s sind die Turbulenzen allerdings bereits größtenteils verklungen, wodurch dieses Mischungsverhältnis die Möglichkeit eines Fehlers in der Rechnung nahelegt. Um diesen Fehler weiter zu untersuchen, und die Fehlerursache weiter einzugrenzen wird im nächsten Test die gleiche Rechnung auf dem gröberen Gitter mit UDS berechnet. Wie in Abbildung 5.14 zu erkennen ist, tritt die Dichteanomalie hierbei nicht auf.

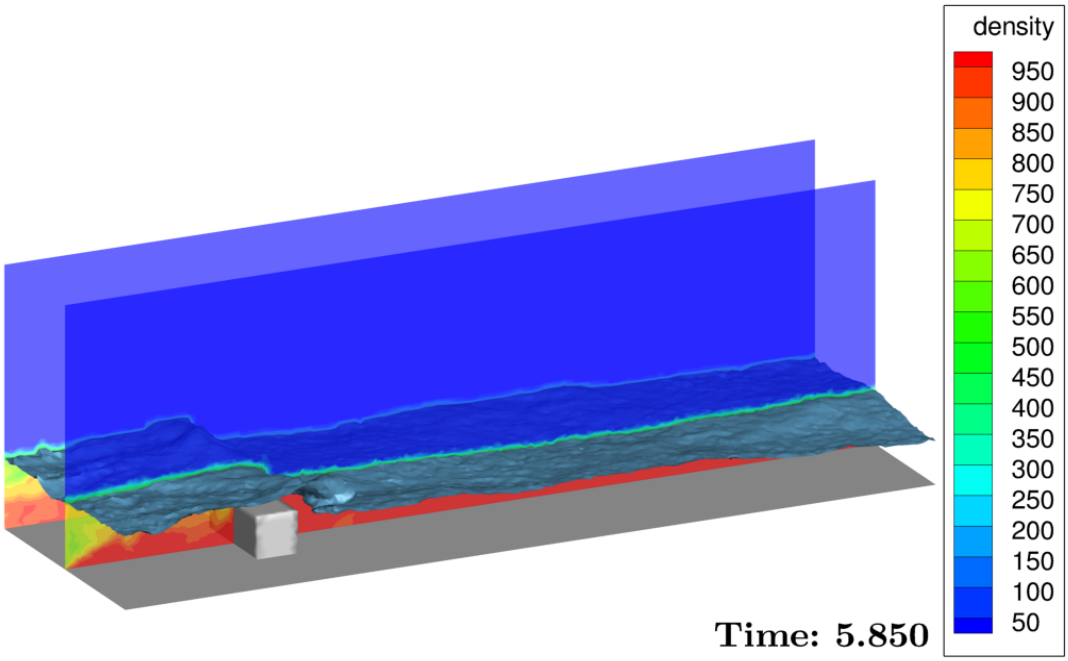



Abbildung 5.13: Darstellung der Dichte bei t = 5.85s

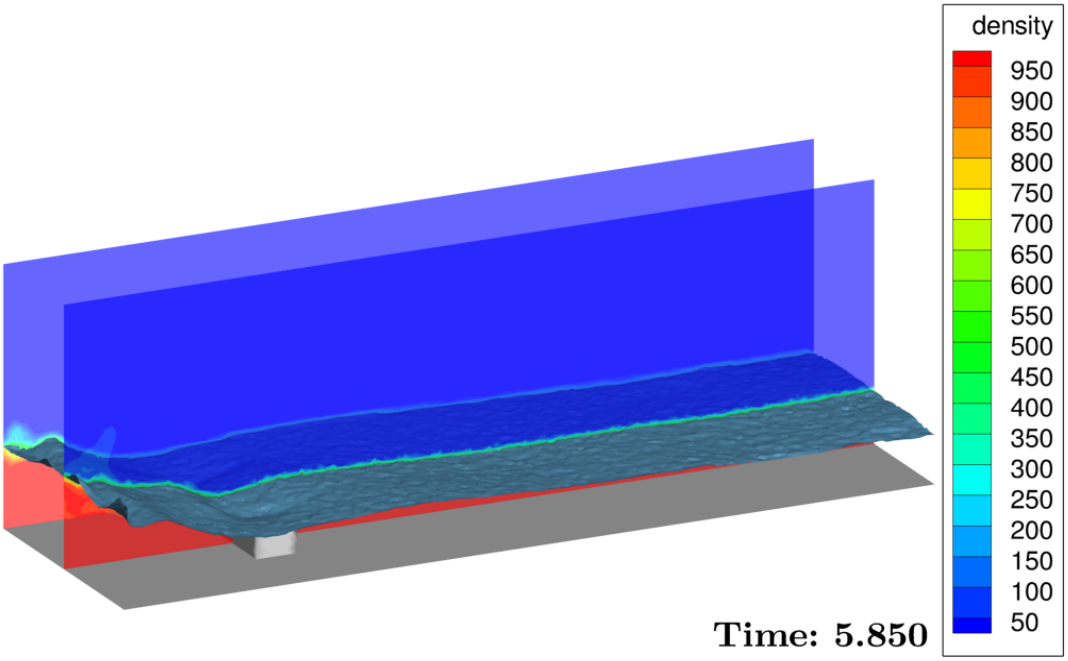



Abbildung 5.14: Vergleichsrechnung Dichtefehler mit UDS

Des Weiteren ist die Welle deutlich weniger weit vorangeschritten. Bei einem Vergleich zwischen der UDS-Rechnung und dem Experiment fällt allerdings auf, dass die UDS Rechnung zeitlich weiter vom Experiment weg ist, als die QUDS-Rechnung. Aus diesen Vergleichsrechnungen ergibt sich, dass die Seiteneffekte der Verwendung von QUDS mit VOF außerhalb dieser Arbeit weiter betrachtet werden müssen.

Zuletzt wurde noch eine weitere Anomalie innerhalb der Logdaten entdeckt. Bei einem Restart treten hier für die identischen Zeitschrittweiten unterschiedliche Konditionen auf. Auffällig sind die deutlich erhöhten Mehrgitteriterationen und die kleinere CFL-Zahl (siehe Abbildung 5.15). Auf Basis dieser Anomalie wurde ausgewertet, inwieweit sich die Unterschiede in den tatsächlichen Felddaten der Rechnung zeigen. Allerdings konnten im Rahmen dieser Arbeit keine signifikanten Unterschiede in den Felddaten festgestellt werden. Diese Anomalie sollte daher im Rahmen weiterer Untersuchungen der Implementierungen am Code korrigiert werden, besitzt den Ergebnissen entsprechend aber keine wesentlichen Auswirkungen.

```
PRES      FGMRES(MG) n: 16   r0: 4.876735e-04 rn: 2.805947e-09 r: 5.754e-06
VOF       BCGS(J)    n: 2    r0: 1.678931e-04 rn: 3.745115e-20 r: 2.231e-16
VEL       BCGS(J)    n: 1    r0: 1.572340e+01 rn: 1.285313e-08 r: 8.175e-10
step=10, dt=0.0001, t=0.001, CFL=0.000229779, D=0.000339518, r=744.306, (2.21/2.22 s)

Output file <Results/dammbruch_bug.pval.10.domain_0> written

PRES      FGMRES(MG) n: 15   r0: 4.308933e-04 rn: 4.299958e-09 r: 9.979e-06
VOF       BCGS(J)    n: 2    r0: 1.864640e-04 rn: 6.421196e-20 r: 3.444e-16
VEL       BCGS(J)    n: 1    r0: 1.572270e+01 rn: 1.585119e-08 r: 1.008e-09
step=11, dt=0.0001, t=0.0011, CFL=0.000259121, D=0.000339518, r=744.168, (2.04/2.05 s)

---

PRES      FGMRES(MG) n: 25   r0: 4.308933e-04 rn: 2.615376e-09 r: 6.070e-06
VOF       BCGS(J)    n: 2    r0: 1.864611e-04 rn: 6.441739e-20 r: 3.455e-16
VEL       BCGS(J)    n: 1    r0: 1.572270e+01 rn: 1.585063e-08 r: 1.008e-09
step=11, dt=0.0001, t=0.0011, CFL=0.000195798, D=0.000339518, r=744.168, (3.28/3.34 s)
```

Abbildung 5.15: Wesentlicher Ausschnitt zweier Logdateien (oben: kein Restart; unten: mit Restart)

## 5.4 Auswertung der Ergebnisse

Die verschiedenen Testfälle haben gezeigt, dass die VOF Methode im Allgemeinen funktional ist. Ein besonderes Augenmerk ist hierbei auf die Verwendung von Mehrgitterverfahren zu werfen, die im Rahmen der ersten Implementierung von

Gauer nicht funktionierte. Auf Basis der durchgeführten Tests weist die kombinierte Verwendung keine Probleme auf. Die Verwendung eines Mehrgitterverfahren sorgt des Weiteren bei komplexeren Fällen für eine Verringerung des Rechenaufwandes, ohne die Qualität der Lösung zu beeinträchtigen.

Während der Testfälle wurden allerdings auch einzelne mögliche Fehlerquellen ausgemacht. Die Verwendung einer QUDS Diskretisierung birgt nach den Ergebnissen des Dammbruch Testfalls die Möglichkeit, dass die Dichte einzelne Fehler aufweist. Dies könnte ein Indiz für Seiteneffekte zwischen der VOF Methode und der QUDS Diskretisierung sein, und sollte dementsprechend weiter am Code kontrolliert werden. Des Weiteren wurden einzelne fragwürdige Werte bei einem Restart der Rechnung in der ersten Iteration in den Logdateien gefunden. Bei weiterer Betrachtung der Felddaten zweier Rechnungen sind allerdings keine signifikanten Unterschiede aufgefallen, die diese Anomalie erklären würden. Da diese Werte dementsprechend keine direkten Auswirkungen auf die Lösungsqualität besitzen, handelt es sich bei dieser Anomalie zwar um eine mögliche Fehlerquelle, die Behebung dieses Problems besitzt im Verhältnis allerdings eine geringe Priorität. Von der Verwendung eines Restarts ist nach diesen Ergebnissen ebenfalls nicht abzuraten.

Anhand der einzelnen Testfälle wurden zusätzlich gute Einstellungen für zukünftige Testfälle mit der VOF Methode ermittelt. Für die Zeitschrittweitensteuerung hat sich hierbei ergeben, dass CFL-Zahlen $< 0.3$ benötigt werden, um sicherzustellen, dass die VOF Methode genaue Ergebnisse liefert. Ebenso sollte der Residuumsfaktor kleiner als 1e-3 gewählt werden, um möglichst genaue Ergebnisse zu erzielen. Größere Faktoren haben in den verwendeten Testfällen bereits zu deutlicheren Abweichungen geführt. Für komplexere Testfälle bietet es sich stets an, diese beiden Einstellungen so genau zu setzen, wie es der maximal gewünschte Rechenaufwand zulässt. Für die Diskretisierung erzeugt die Verwendung von QUDS gegenüber UDS bessere Ergebnisse. Unter Beachtung der erwähnten Dichteanomalie bietet es sich allerdings an, auf eine Verwendung von UDS solange zurückzugreifen, bis geklärt ist, inwieweit es sich bei dieser Anomalie tatsächlich um einen problematischen Seiteneffekt handelt.

# 6 Fazit und Ausblick

Im Rahmen dieser Arbeit wurde anhand existierender Standards und Vorgehensweisen eine eigene Methodik zur Verifizierung und Validierung bereits bestehender Implementierungen entwickelt und diese anhand eines praktischen Beispiels angewendet. Dabei wurde der Fokus des Prozesses auf die Auswertung numerischer Strömungsmodelle und deren Seiteneffekte zu anderen Modulen gelegt.

Die entwickelte Methodik hat am Beispiel der Verifizierung und Validierung des Lösungsverfahrens der VOF Methode des DLR THETA-Codes einzelne mögliche Fehler innerhalb dieser aufgedeckt und diese ausreichend dokumentiert, um in weiteren Schritten den Code direkt zu untersuchen. Des Weiteren ist es möglich anhand der Untersuchung festzustellen, welche Testfälle minimal abgedeckt werden müssen, um sicherzustellen, dass zukünftige Entwicklungen keine weiteren Seiteneffekte hinsichtlich dieser Implementierung besitzen. Im Rahmen von automatischen Tests, müssen diese Test einen möglichst geringen Rechenaufwand aufweisen und dennoch eine möglichst große Codeabdeckung besitzen. Für die V&V der VOF Methode bietet es sich für die Zukunft an, zwei verschiedene „Sloshing Tank“ Testfälle in die automatische Testsuite mit aufzunehmen. Dabei sollten ein Restart und die QUDS Diskretisierung mit eingeschlossen werden, deren Seiteneffekte bis jetzt nicht abgedeckt werden.

Der praktische Anwendungsfall hat für die Methodik aufgezeigt, dass der wesentliche Aspekt eine ausführliche Dokumentation des Prozesses ist. Mit Hilfe dieser kann sichergestellt werden, dass im Fall einer nachfolgenden Untersuchung sicher nachvollzogen werden kann, welche Testfälle mit welcher Intention durchgeführt wurden, und welche Ergebnisse diese produziert haben. Dabei ist vor allem eine mögliche Rekonstruktion der Testergebnisse von entscheidender Bedeutung. Eine Dokumentation die festhält, welche Programmversion, welche Parameter und welche Auswertungsmethodiken verwendet wurden ist hierbei notwendig und sollte unternehmensintern zur Verfügung stehen.

Während das theoretische Ergebnis dieser Arbeit eine Methodik zur effizienten Entwicklung einer systematischen V&V ist, liegt das praktische Ergebnis in einer Zusammenfassung möglicher Fehler und optimierter Parameter des Lösungsverfahrens der VOF Methode des THETA-Codes. Das resultierende „Best Practice“ Dokument ist in Anhang D angehängt und kann für zukünftige Rechnungen mit der VOF Methode referenziert werden. Die präsentierten Ergebnisse stimmen mit denen aus Kapitel 5 überein.

Des Weiteren wurde im Rahmen dieser Arbeit ein Parameter zur besseren Bedienung der VOF Methode hinzugefügt. Dieser ermöglicht es zur Definition des Gebietes des zweiten Mediums ein zusätzliches Gitter einzulesen. Mit Hilfe dieser Methode wurde es erleichtert, komplexe Initialisierungen durchzuführen.

# Literatur


[Air] *Airfoil Tools*. Aufgerufen in 09-2016. 2016. URL: `http://airfoiltools.com`.

[And95] John David Anderson. *Computational Fluid Dynamics*. 1995.

[Ber+95] Marschall Bern u. a. „Dihedral Bounds for Mesh Generation in High Dimensions“. In: *Proc. 6th Symp. Discrete Algorithms*. 1995.

[Cen16] CentaurSoft. *CENTAUR from CentaurSoft*. Aufgerufen in 09-2016. 2016. URL: `https://www.centaursoft.com/grid-generator`.

[DLR15] DLR. *Guidelines for Development/ Commit Process*. 2015.

[DLR12] DLR. *THETA-Code Developer's Guide*. 2012.

[DLR16] DLR. *THETA-Code User Guide*. 2016.

[Gau13] Markus Gauer. „Simulation of the sloshing behaviour of two-phase flows in cryogenic rocket upper stages“. Diss. Universität Stuttgart, 2013.

[GHH09] Markus Gauer, Volker Hannemann und Klaus Hannemann. „Implementation of the VOF method in the DLR TAU code“. In: *45th AIAA/ASME/SAE/ASEE Joint Propulsion Conference & Exhibit*. 2009.

[Gau+12] Markus Gauer u. a. „Two-phase flow modeling combining the CICSAM with the projection method“. In: *48th AIAA/ASME/SAE/ASEE Joint Propulsion Conference & Exhibit*. 2012.

[Gnu] *gnuplot homepage*. Aufgerufen in 09-2016. 2016. URL: `http://www.gnuplot.info/`.

[Gro16] University of Groningen. *ComFLOW - dambreak experiment*. Aufgerufen in 07-2016. 2016. URL: `http://www.math.rug.nl/~veldman/comflow/dambreak.html`.

[Hac85] Wolfgang Hackbusch. *Multi-Grid Methods and Applications*. Springer-Verlag, 1985.

[IEE04] IEEE. „Standard for Software Verification and Validation“. In: *IEEE Std. 1012* (2004).

[Kap15] Julian Kaping. *Bericht zum Modul Praxis II*. Techn. Ber. Duale Hochschule Baden-Württemberg, 2015.

[KFV04] K. M. T. Kleefsman, Geert Fekken und A. E. P. Veldman. „An Improved Volume-of-Fluid Method for Wave Impact Problems“. In: *The Fourteenth International Offshore and Polar Engineering Conference*. 2004, S. 334–341.

[Lec09] Stefan Lecheler. *Numerische Stroemungsberechnung*. 3. Aufl. Springer-Verlag, 2009.

[PF02] Milovan Perić und Joel H. Ferziger. *Computational Methods for Fluid Dynamics*. Springer, 2002.

[Poi16] Pointwise. *Mesh and Grid Generation Software for CFD*. Aufgerufen in 09-2016. 2016. URL: `http://www.pointwise.com/pw/`.

[RCJ95] Peter E. Raad, Shea Chen und David B. Johnson. „The introduction of micro cells to treat pressure in free surface fluid flow problems“. In: *Journal of Fluids Engineering* 117.4 (1995), S. 683–690.

[Sar11] Robert G. Sargent. „Verification and Validation of Simulation Models“. In: *Winter Simulation Conference*. 2011, S. 183–198.

[Tec16] Tecplot. *Tecplot 360 CFD post processing tools to analyze data*. Aufgerufen in 09-2016. 2016. URL: `http://www.tecplot.com/products/tecplot-360`.

[Ubb97] Onno Ubbink. „Numerical prediction of two fluid systems with sharp interfaces“. Diss. University of London, 1997.

# A Navier-Stokes-Gleichungen

## A.1 Massen-, Impuls-, und Energieerhaltungsgleichungen

Navier-Stokes-Gleichungen in Vektorform [Lec09, S.22]:

$$\frac{\partial}{\partial t}\vec{U} + \frac{\partial}{\partial x}\vec{E} + \frac{\partial}{\partial y}\vec{F} + \frac{\partial}{\partial z}\vec{G} = \vec{Q} \tag{A.1}$$

Bei den folgenden Vektoren ist in der ersten Zeile stets die Massenerhaltungsgleichung, in den folgenden drei Zeilen die Impulserhaltungsgleichungen und in der letzten Zeile die Energieerhaltungsgleichung inbegriffen.

Erhaltungsvektor $\vec{U}$ [Lec09, S.22]:

$$\vec{U} = \begin{bmatrix} \rho \\ \rho \cdot u \\ \rho \cdot v \\ \rho \cdot w \\ \rho \cdot \left[ e + \frac{1}{2}(u^2 + v^2 + w^2) \right] \end{bmatrix} \tag{A.2}$$

Quellvektor $\vec{Q}$ [Lec09, S.23]:

$$\vec{Q} = \begin{bmatrix} 0 \\ \rho \cdot g_x \\ \rho \cdot g_y \\ \rho \cdot g_z \\ \rho \cdot (u \cdot g_x + v \cdot g_y + w \cdot g_z) \end{bmatrix} \tag{A.3}$$

Flussvektoren $\vec{E}$, $\vec{F}$ und $\vec{G}$ in x-, y- und z-Richtung [Lec09, S.22]:

$$\vec{E} = \begin{bmatrix} \rho \cdot u \\ \rho \cdot u^2 + p - \tau_{xx} \\ \rho \cdot v \cdot u + p - \tau_{xy} \\ \rho \cdot w \cdot u + p - \tau_{xz} \\ \rho \cdot u \cdot \left[h + \frac{1}{2}(u^2 + v^2 + w^2)\right] - u \cdot \tau_{xx} - v \cdot \tau_{xy} - w \cdot \tau_{xz} - \lambda \cdot \frac{\partial T}{\partial x} \end{bmatrix} \tag{A.4}$$

$$\vec{F} = \begin{bmatrix} \rho \cdot v \\ \rho \cdot u \cdot v + p - \tau_{yx} \\ \rho \cdot v^2 + p - \tau_{yy} \\ \rho \cdot w \cdot v + p - \tau_{yz} \\ \rho \cdot v \cdot \left[h + \frac{1}{2}(u^2 + v^2 + w^2)\right] - u \cdot \tau_{yx} - v \cdot \tau_{yy} - w \cdot \tau_{yz} - \lambda \cdot \frac{\partial T}{\partial y} \end{bmatrix} \tag{A.5}$$

$$\vec{G} = \begin{bmatrix} \rho \cdot w \\ \rho \cdot u \cdot w + p - \tau_{zx} \\ \rho \cdot v \cdot w + p - \tau_{zy} \\ \rho \cdot w^2 + p - \tau_{zz} \\ \rho \cdot w \cdot \left[h + \frac{1}{2}(u^2 + v^2 + w^2)\right] - u \cdot \tau_{zx} - v \cdot \tau_{zy} - w \cdot \tau_{zz} - \lambda \cdot \frac{\partial T}{\partial z} \end{bmatrix} \tag{A.6}$$

## A.2 Zustands- und Beziehungsgleichungen

Thermische Zustandsgleichung [Lec09, S.24] ($R$ - Gaskonstante):

$$p = \rho \cdot R \cdot T \tag{A.7}$$

Kalorische Zustandsgleichungen [Lec09, S.24] ($c_v$ - spezifische Wärmekapazität, konstantes Volumen; $c_p$ - spezifische Wärmekapazität, konstanter Druck):

$$de = c_v \cdot dT \tag{A.8}$$

$$dh = c_p \cdot dT \tag{A.9}$$

Stokessche Beziehungen [Lec09, S.24f] ($\mu$ - dynamische Viskosität):

$$\tau_{xx} = -\frac{2}{3}\mu \cdot \left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z}\right) + 2 \cdot \mu \cdot \frac{\partial u}{\partial x} \tag{A.10}$$

$$\tau_{yy} = -\frac{2}{3}\mu \cdot \left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z}\right) + 2 \cdot \mu \cdot \frac{\partial v}{\partial y} \tag{A.11}$$

$$\tau_{zz} = -\frac{2}{3}\mu \cdot \left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z}\right) + 2 \cdot \mu \cdot \frac{\partial w}{\partial z} \tag{A.12}$$

$$\tau_{xy} = \mu \cdot \left(\frac{\partial v}{\partial x} + \frac{\partial u}{\partial y}\right) \tag{A.13}$$

$$\tau_{xz} = \mu \cdot \left(\frac{\partial u}{\partial z} + \frac{\partial w}{\partial x}\right) \tag{A.14}$$

$$\tau_{yz} = \mu \cdot \left(\frac{\partial w}{\partial y} + \frac{\partial v}{\partial z}\right) \tag{A.15}$$

$$\tau_{yx} = \tau_{xy} \tag{A.16}$$

$$\tau_{zx} = \tau_{xz} \tag{A.17}$$

$$\tau_{zy} = \tau_{yz} \tag{A.18}$$

# B Beispiel Parafile

Die folgende Parameterübersicht ist nicht vollständig und umfasst hauptsächlich in Kapitel 5 verwendete und variierte Parameter:

```
    Files/IO ------------------------------------------------: -
                                        Restart-data prefix: (none)
                                                Grid prefix: (none)
                                        Output files prefix: (none)

    Timestepping Start/Stop ---------------------------------: -
                                    Maximal time step number: 100
                                                Output period: 9999999

    Timestep Settings ---------------------------------------: -
                               CFL number or negative time step: -0.0001

    Computational quantities --------------------------------: -
        Discretization scheme for momentum (UDS/CDS/LUDS/QUDS): QUDS
          Time discretization scheme (Steady/EU_E/EU_I/TPB/CN): EU_I

    Reference quantities ------------------------------------: -
                                          Reference density: 1.29
                                       Reference velocities: 1 0 0
                                         Reference pressure: 100000
                                  Reference point coordinates: 0 0 0

    Activated models ----------------------------------------: -
                                       Names of models to use: VOF MOM_SRC

    Pressure equation solver --------------------------------: -
                           Type of solver for pressure equation: FGMRES
                                    Epsilon for pressure solver: 1e-05
                             Preconditioning of pressure solver: MG
                  MG description filename for pressure solver: v3
    Momentum equation solver --------------------------------: -
                                    Epsilon for momentum solver: 1e-05

    Scalar equation solver ----------------------------------: -
                                      Epsilon for scalar solver: 1e-05

    Two-phase flow settings ---------------------------------: -
                               Reference density for second fluid: 997.54
```

# C Beispiel user.c

Die folgende user.c-Datei wurde auf wesentliche in Kapitel 5 verwendete Funktionen gekürzt. Gezeigt ist die verwendete Datei für den zweidimensionalen „Sloshing Tank“ Testfall.

```
34 double step_size = 0.0;

36 /*****************************************************************************
37 * get_user_interface_id
38 *  - the user should specify an identification string for the interface
39 *  - this string is printed to the stdout
40 *****************************************************************************/
41 char* get_user_interface_id(void)
42 {
43   return "2D Sloshing";
44 } /* get_user_interface_id() */

56 /*****************************************************************************
57 * user_set_initial_values
58 *  - called by set_initial_values() when starting from scratch
59 *  - the user can initiate all field data which are stored
60 *    in the var_data structure
61 *  - keeping this function empty, all variables will be initialized by
62 *    their respective reference values
63 *****************************************************************************/
64 void user_set_initial_values(ThetaVar *var_data)
65 {
66   TauDualGrid         *dgrid  = var_data->dualgrid;
67   ModelModelInterface *mm_ifc = get_model_model_interface();

69   TauDouble (*pcoord)[3] = dgrid->gpdat->xx;

71   int nallpnt = dgrid->gpdat->nownpoints;
72   int ivof    = mm_ifc->get_idx_interface();

74   TauDouble *vof = var_data->var_i[ivof];
75   TauDouble *pres = var_data->var_i[IP];
76   TauDouble dens_sec = mm_ifc->get_density_second_fluid;

78   int face = 0;
79   int pnt;

81   /* Calculate step size in z-direction (equidistant)*/
```

```
82    while(step_size < 1e-9) {
83      int p0 = dgrid->gedat->fpoint[face][0];
84      int p1 = dgrid->gedat->fpoint[face][1];
85      double diff = pcoord[p0][2] - pcoord[p1][2];

87      if(diff < -1e-9  diff > 1e-9) {
88        step_size = (diff > 0) ? diff : -diff;
89        break;
90      }
91      else
92        face++;
93    }

95    /* Initialize fluids with given interface-function */
96    for(pnt = 0; pnt < nallpnt; pnt++)
97    {
98      const TauDouble z_f = 0.005 * cos(M_PI*10.0*pcoord[pnt][0]) + 0.05;

100     if(pcoord[pnt][2] + (step_size/2) < z_f) {
101       vof[pnt] = 1.0;
102       pres[pnt] = (0.1-z_f) + (z_f-pcoord[pnt][2])*dens_sec;

104     /* Interface */
105     } else if (pcoord[pnt][2] - (step_size/2) < z_f) {
106       vof[pnt] = (z_f - pcoord[pnt][2]) / step_size + 0.5;
107       if (pcoord[pnt][2] < z_f) {
108         pres[pnt] = (0.1-z_f) + (z_f-pcoord[pnt][2])*dens_sec;
109       } else {
110         pres[pnt] = (0.1-pcoord[pnt][2]);
111       }
112     } else {
113       vof[pnt] = 0.0;
114       pres[pnt] = (0.1-pcoord[pnt][2]);
115     }
116   }

118 } /* user_set_initial_values() */

164 /*****************************************************************************
165 * user_callback
166 *
167 * - called at different locations in the code
168 * - the user can monitor internal data
169 *   or carefully modify internal data and solver parameters
170 * - always check for the requested event, new events may be added at any time
171 *****************************************************************************/
172 void user_callback(ThetaVar *var_data, ThetaUserEvent event)
173 {
174   TauDualGrid         *dgrid  = var_data->dualgrid;
175   ModelModelInterface *mm_ifc = get_model_model_interface();
176   TimeStepParam   *time_param = get_timestep_para();
```

```
178   TauDouble (*pcoord)[3] = dgrid->gpdat->xx;

180   int nownpnt = dgrid->gpdat->nownpoints;
181   int ivof    = mm_ifc->get_idx_interface();

183   TauDouble *vof = var_data->var_i[ivof];
184   TauDouble vof_complete = 0.0;

186   /* Communicate number of all points */
187   int nallpnt;
188   MPI_Allreduce(&nownpnt, &nallpnt, 1, MPI_INT, MPI_SUM, MPI_COMM_WORLD);

190   /* 3D */
191   if(dgrid->dginfo->twoD_offset_vector == 0) {
192     nallpnt /= 2;
193   }

195   /* Calculate number of cells per direction (x and z have the same number) */
196   printf("\n%d", nallpnt);
197   int cells = (int)sqrt(nallpnt);
198   int i;

200   TauDouble *heights = (TauDouble *)check_malloc(cells * sizeof(TauDouble));
201   TauDouble *fin_heights = (TauDouble *)check_malloc(cells * sizeof(TauDouble));
202   for(i = 0; i < cells; i++) {
203     heights[i] = 0.0;
204     fin_heights[i] = 0.0;
205   }

207   /* Calculate interface */
208   for(i = 0; i < nownpnt; i++) {
209     if(dgrid->dginfo->twoD_offset_vector != 0  pcoord[i][1] < 1e-9) {
210       int pos = (int)(pcoord[i][0] / step_size + 0.5);
211       heights[pos] += vof[i];
212       vof_complete += vof[i];
213     }
214   }

216   /* Get all heights on mpi_rank 0 */
217   for(i = 0; i < cells; i++) {
218     MPI_Reduce(&heights[i], &fin_heights[i], 1, MPI_DOUBLE, MPI_SUM, 0,
219                MPI_COMM_WORLD);
220   }

222   TauDouble result;
223   MPI_Reduce(&vof_complete, &result, 1, MPI_DOUBLE, MPI_SUM, 0, MPI_COMM_WORLD);

225   /* Gather information which process i am */
226   int mpi_rank;
227   MPI_Comm_rank(MPI_COMM_WORLD, &mpi_rank);
```

```
229   if(mpi_rank == 0) {
230     TauDouble spatial_error = 0.0;
231
232     for(i = 0; i < cells; i++) {
233       const TauDouble z_f = 0.005 * cos(M_PI*10.0*i*step_size) + 0.05;
234
235       fin_heights[i] -= 0.5;
236       fin_heights[i] *= step_size;
237
238       spatial_error += (fin_heights[i] - z_f) * (fin_heights[i] - z_f);
239     }
240
241     spatial_error = 100/(0.005 * sqrt(cells)) * sqrt(spatial_error);
242
243     FILE *file;
244     file = fopen("monitor.dat", "a");
245
246     if(file == NULL) {
247       printf("\nCould not open file!\n");
248       return;
249     }
250
251     fprintf(file, "%f %f %f %f\n", time_param->phys_time, fin_heights[0],
252                                    spatial_error, result);
253     fclose(file);
254   }
255 } /* user_callback() */
```

# D „Best Practices“

This document aims to present optimised parameters for the usage of the VOF method in the DLR THETA environment. The parameters presented in this document are suggestions and do not claim to be the optimal parameters for each simulation. Therefore it may be necessary to adjust some parameters.

Results were obtained by evaluating the following two test cases. The first one being a simple, two-dimensional tank, the interface initialised with a cosine function. The second one is a dam break simulation, which is much more complex and needs the usage of the multigrid to operate efficiently. Images of the composition of the two test cases can be found in figure D.1.

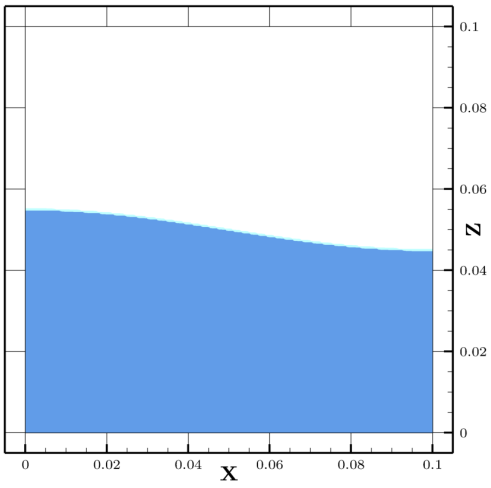
The used THETA version is 12.0 and the test cases were firstly executed in the time from the end of June to the beginning of September 2016. All changes made to the implementation after this time span will, therefore, have no effect on suggested parameters.

Tested parameters in this test cases include:

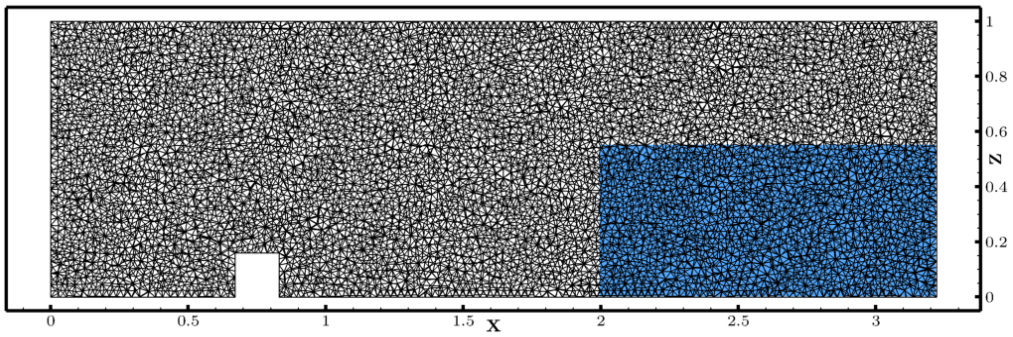
- CFL
- Epsilon for different solver
- Discretization schemes for momentum
- Usage of BCGS and FGMRES
- Scaling of spatial discretization

Additionally the 2D-functionality and parallel execution of the solver were tested.

Results for some of the executed scenarios will be presented in the following chapters. The sole recommendations can be found in table D.1.




(a) 2D Sloshing tank




(b) 3D dam break (slice at y=0)

Figure D.1: Composition of the two used testcases

| **Parameter** | **Recommended settings** |
|---|---|
| CFL / timestep | CFL should not exceed 0.3 (recommended: 0.2) |
| Epsilon | As small as necessary (maximal 1e-03) |
| Discretization schemes | QUDS or UDS (developer used UDS) |
| Multigrid? | Safe to use to speed up simulation |
| 2D? | Safe to use to speed up simulation |
| Parallel? | Safe to use to speed up simulation |
| Spatial Scaling? | Finer grids produce more accurate solutions |

Table D.1: Overview of recommended parameter settings

## D.1 Sloshing Tank

As shown in figure D.1a, the Sloshing Tank test case is a two-dimensional box ($0.1m \times 0.1m$) filled with a fluid to about its half. The interface is initialized by the function

$$z(x) = 0.05m + 0.005m \cdot \cos(\pi \frac{x}{0.1}) \tag{D.1}$$

To evaluate the quality of the results, two measures were introduced by [RCJ95] and used by [Ubb97] and [Gau13].

$$e_{temp} = 100 \frac{t_s - t_t}{t_t} \tag{D.2}$$

$$e_{spat} = \frac{100}{0.005\sqrt{M}} \left( \sum_{m=1}^{M} (z_s - z_t)^2 \right)^{\frac{1}{2}} \tag{D.3}$$

The errors were evaluated at the end of each even period number. The reasoning behind this is explained in [Gau13, p.105]. To compute both errors, it was necessary to determine when a period was over. This was achieved by checking the height of the interface at the left wall with the user.c-function `user_callback`. After each time step, the height of the interface at the left wall was written into a separate monitor file, which could be evaluated after the simulation. The spatial error would also be calculated inside of this of this function and written into the monitor file.

| **Period** | **c = 0.05** | **c = 0.1** | **c = 0.2** | **c = 0.4** | **c = 0.6** |
|---|---|---|---|---|---|
| **2** | 0.24 | 0.20 | 0.13 | 0.10 | 0.50 |
| **4** | 0.05 | 0.07 | 0.08 | 0.05 | 0.13 |
| **6** | 0.14 | 0.15 | 0.16 | 0.13 | 0.20 |
| **8** | 0.12 | 0.13 | 0.14 | 0.15 | 0.20 |
| **10** | 0.17 | 0.18 | 0.19 | 0.18 | 0.20 |

Table D.2: Temporal errors for different CFL-numbers

As shown in table D.2 and D.3 a CFL-number above 0.2 shows some varying results. At a CFL-number of 0.4 a lost sharpness of the interface is observable at the spatial

| Period | c = 0.05 | c = 0.1 | c = 0.2 | c = 0.4 | c = 0.6 |
|---|---|---|---|---|---|
| **2** | 1.74 | 1.74 | 1.70 | 1.61 | 1.49 |
| **4** | 2.47 | 2.49 | 2.52 | 2.55 | 2.28 |
| **6** | 3.92 | 4.00 | 4.11 | 4.20 | 3.83 |
| **8** | 5.99 | 6.05 | 6.11 | 6.13 | 5.77 |
| **10** | 7.68 | 7.77 | 7.66 | 7.70 | 7.62 |

Table D.3: Spatial errors for different CFL-numbers

error, while a CFL-number of 0.6 produces worse results at the temporal error. A simulation with a CFL-number greater than 0.6 did not converge in this test series.

| Period | r = 1e-2 | r = 1e-3 | r = 1e-4 | r = 1e-5 | r = 1e-6 |
|---|---|---|---|---|---|
| **2** | 0.18 | 0.18 | 0.18 | 0.13 | 0.13 |
| **4** | 0.13 | 0.08 | 0.08 | 0.08 | 0.08 |
| **6** | 0.19 | 0.16 | 0.14 | 0.16 | 0.16 |
| **8** | 0.19 | 0.15 | 0.14 | 0.14 | 0.14 |
| **10** | 0.24 | 0.20 | 0.19 | 0.19 | 0.19 |

Table D.4: Temporal errors for different Epsilon

| Period | r = 1e-2 | r = 1e-3 | r = 1e-4 | r = 1e-5 | r = 1e-6 |
|---|---|---|---|---|---|
| **2** | 1.72 | 1.70 | 1.70 | 1.70 | 1.70 |
| **4** | 2.42 | 2.52 | 2.52 | 2.52 | 2.52 |
| **6** | 4.00 | 4.00 | 4.11 | 4.11 | 4.11 |
| **8** | 6.06 | 6.00 | 6.11 | 6.11 | 6.11 |
| **10** | 7.71 | 7.68 | 7.78 | 7.66 | 7.66 |

Table D.5: Spatial errors for different Epsilon

As shown in table D.4 and D.5 the results show an increase of the temporal error beginning from an epsilon of 1e-03. At an Epsilon of 1e-02 the significance of the error was high enough to not recommend this setting further.

As shown in table D.6 the results of UDS and QUDS in hinsight of the temporal error show varying results. The tendency seems to favor QUDS as a better choice.

Tests where the usage of the 2D-option, parallel execution and Multigrid solver were compared, showed no difference, hence have no side effects on the solution and may

| Period | UDS | QUDS |
|---|---|---|
| **2** | 0.18 | 0.48 |
| **4** | 0.08 | 0.00 |
| **6** | 0.16 | 0.09 |
| **8** | 0.15 | 0.07 |
| **10** | 0.20 | 0.08 |

Table D.6: Temporal errors for discretization schemes UDS and QUDS

be used to improve convergence.

## D.2 Dam break

This test case is based on an experiment of the Maritime Research Institute Netherlands. The tank for the simulation has a size of $3.22m \times 1m \times 1m$. At the beginning of the experiment, the water is initialised at the back end of the tank beginning at $x = 2m$ with a height of $0.55m$. As an obstacle an container of the size $0.16m \times 0.4m \times 0.16m$ is placed at the position $(0.67m| - 0.2m|0m)$. Height sensors were positioned as shown in figure D.2.

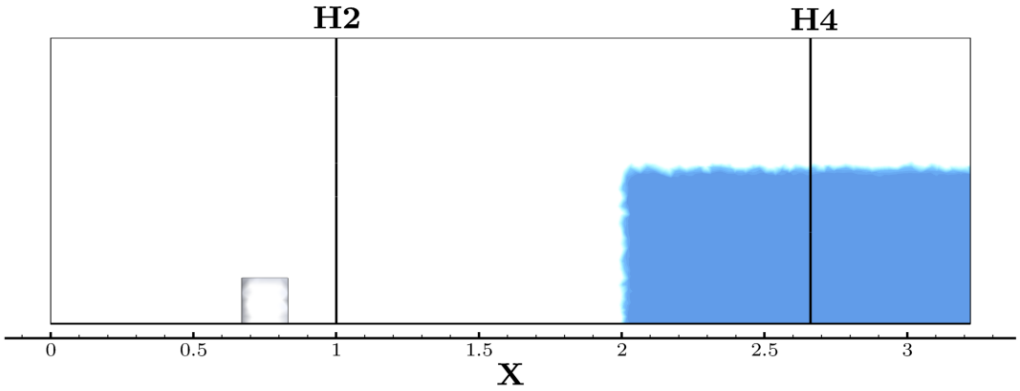



Figure D.2: Positions of H2 and H4 in the dam break experiment ($Y = 0$)

H4 has an x value of $2.66m$. Results were gained for different time steps using QUDS, an epsilon of 1e-05 on an unstructured grid with about a quarter million points. No

adaptations were used for the turbulence around the container on impact. Results of position H4 are shown in figure D.3.

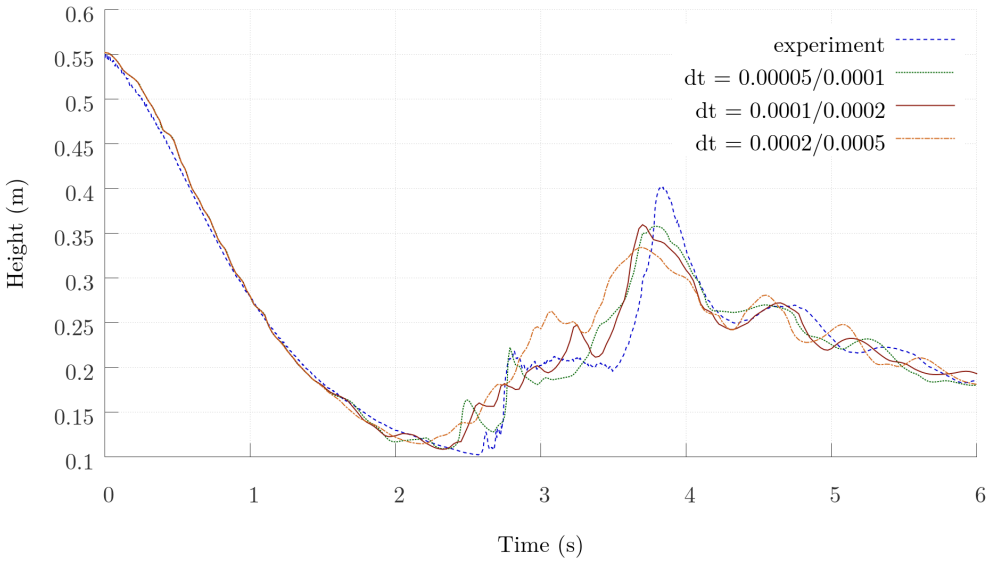



Figure D.3: Comparsion different timesteps at position H4

At the two peaks the difference in the accuracy of the time steps is visible: As the time steps get more precise the temporal error gets smaller. The difference between the smaller two time steps is a little smaller than the difference between the greater two. The time step of 0.0001/0.0002 produces CFL-numbers under 0.2, which seems to have the best quality to performance ratio. Therefore it is advised to use time steps which result in CFL-numbers equal to or smaller than 0.2.

# E Video des 3D Dammbruchs

Auf der beigelegten CD zu dieser Arbeit ist ein Video enthalten, welches die Simulation des Dammbruchs darstellt. Das Video ist im mp4-Format gespeichert und sollte von allen gängigen Media-Playern abgespielt werden können. Die Länge des Videos beträgt zwölf Sekunden, in denen auf halber Geschwindigkeit die ersten sechs Sekunden des Dammbruchs dargestellt werden. Das Video basiert auf der Rechnung auf dem feinen Gitter.

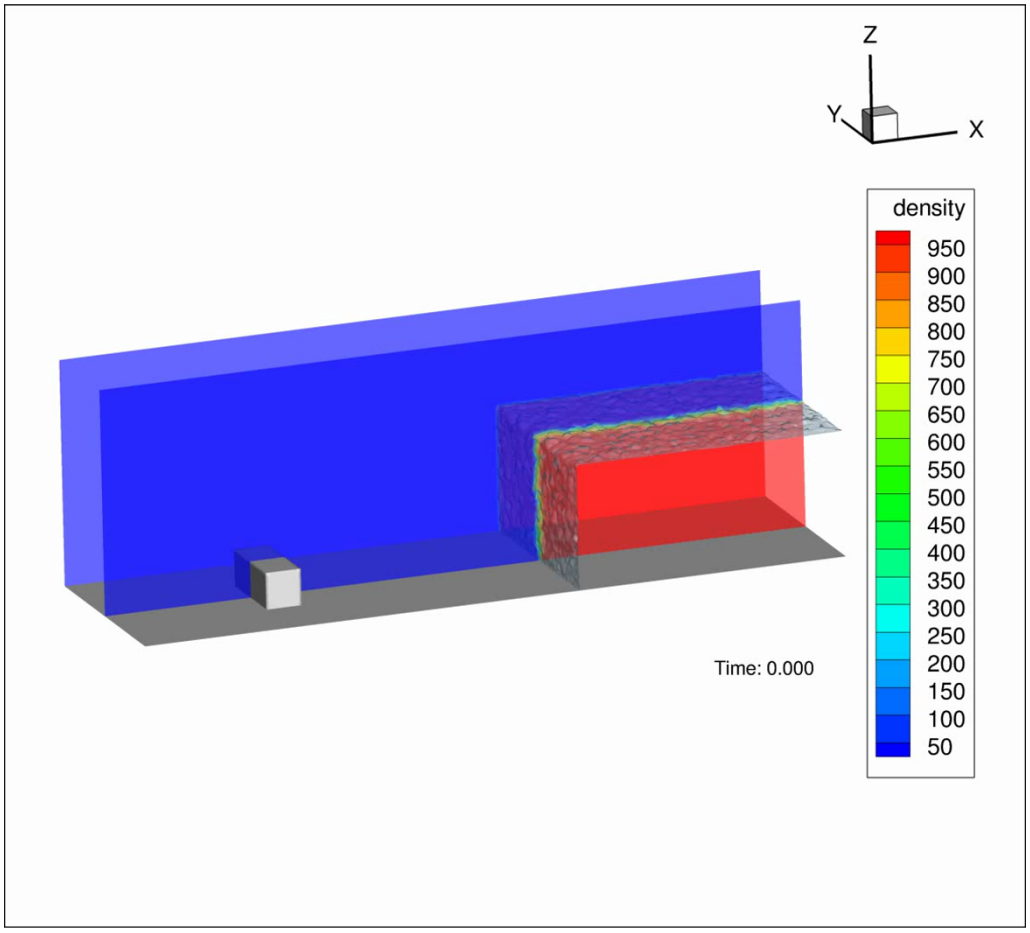



Abbildung E.1: Schnappschuss des Videos auf der beigelegten CD

# F Verifizierung und Validierung nach IEEE 1012

Entnommen aus [IEE04, S.17], 5.4.4 Implementation V&V:

1. Traceability analysis
2. Source code and source code documentation evaluation
3. Interface analysis
4. Criticality analysis
5. Component V&V test case generation
6. Integration V&V test case generation
7. System V&V test case generation
8. Acceptance V&V test case generation
9. Component V&V test procedure generation
10. Integration V&V test procedure generation
11. System V&V test procedure generation
12. Component V&V test execution
13. Hazard analysis
14. Security analysis
15. Risk analysis