Ausarbeitung der Pfropfenströmung zum effektiven Extraktionswerkzeug in der Mikroverfahrenstechnik

Zur Erlangung des akademischen Grades eines
Dr.-Ing.
von der Fakultät Bio- und Chemieingenieurwesen
der Technischen Universität Dortmund

genehmigte Dissertation

vorgelegt von
M.Sc. Christian Andreas Schwarz
aus Bochum, Deutschland

Tag der mündlichen Prüfung: 26.10.2021
1. Gutachter: Prof. Dr. David W. Agar
2. Gutachter: Prof. Dr. Peter Ehrhard

Veröffentlichung als Dissertation der Fakultät für Bio- und Chemieingenieurwesen der Technischen Universität Dortmund.
Dissertationsort: Dortmund, Deutschland

Danksagung

Es gibt eine Vielzahl von Menschen, ohne deren Unterstützung die vorliegende Arbeit nicht im aktuellen Maße hätte entstehen können. All diesen Personen möchte ich für ihren Beistand danken, sei er durch Taten, Worte, Fürsorge oder anderweitiges Wirken entstanden. Nicht alle können namentlich erwähnt werden, doch seien alle Personen angesprochen, denen ich das Gelingen der letzten Jahre zu verdanken habe. Spezielle Personen sollen dennoch hervorgehoben werden.

In erster Linie danke ich Prof. Dr.-Ing. D.W. Agar für die Betreuung meiner Promotionszeit und die Aufnahme am Lehrstuhl für chemische Verfahrenstechnik. Er setzte viel Vertrauen in mich und stand in Problemfällen stets für ein offenes Gespräch zur Verfügung. Weitere fachliche Unterstützung erhielt ich von den Mitarbeitern der mechanischen und elektronischen Werkstätten sowie der Glasbläserei und Michael Schlüter. Allen Personen sei für ihre Beratung bei der Umsetzung der experimentellen Arbeiten gedankt.

Weiterhin möchte ich auch den Studenten danken, die ich während meiner Zeit am Lehrstuhl betreuen durfte, und die im Rahmen ihrer Abschluss- oder studentischen Hilfsarbeiten zum Gelingen der Experimente beigetragen haben. Auch den Mitdoktoranden, ganz speziell meinen Bürokollegen, sei für die anregenden Fachdiskussionen, Tipps und das angenehme, motivierende Arbeitsumfeld gedankt.

Neben den fachlichen Komponenten trägt jedoch auch das persönliche Umfeld zum Gelingen einer solchen Arbeit bei. Allen voran möchte ich meinen Eltern, Ursula und Manfred Schwarz, und dem Rest meiner Familie für den Glauben an mich, den Beistand und den guten Zuspruch danken, den ich von ihrer Seite erhalten habe. Auch meinen Freunden danke ich für ihr aufgebrachtes Verständnis und die vielen schönen gemeinsamen Stunden, in denen ich Zuversicht und Kraft sammeln konnte. Besonders hervorheben möchte in diesem Zusammenhang Sascha Vetter, sowie Jan und Sarah Hülsewig. In ihnen fand ich ganz besondere Ansprechpartner voller Vertrauen in allen Momenten und Lebenslagen. Für ihre Geduld, Rücksicht und den emotionalen Beistand, auch schon weit vor Beginn der Promotionszeit, danke ich in ganz besonderem Maße.

Zusammenfassung

Der allgemeine Trend zur Prozessintensivierung lässt die Mikroverfahrenstechnik aufgrund ihrer charakteristischen kurzen Transportwege für Wärme- und Stoffströme in den Interessensfokus rücken. Speziell für rein transportlimitierte Prozesse wie die Extraktion kann eine Miniaturisierung erhebliche Verbesserungen hervorbringen.

Im Rahmen dieser Arbeit wurde daher die flüssig-flüssig Pfropfenströmung als Extraktionswerkzeug weiterentwickelt. Zum einen wurde dabei die Beseitigung bisheriger Nachteile beim Einsatz der Pfropfenströmung fokussiert. Hierzu zählen die noch nicht abschließend erforschten Vorgänge bei der Phasentrennung, sowie der noch erhebliche Aufwand bei der Parallelisierung. Zum anderen wurde die Pfropfenströmung in Anwendungsgebieten untersucht, die für bisherige Extraktionsapparate noch große Herausforderungen darstellen.

Die Vorgänge während der Phasentrennung wurden durch numerische Simulationen näher untersucht. Hierfür wurde als ein Hauptpunkt der Arbeit eine numerische Methode mit besonders geringen parasitären Strömungen entwickelt. Bei hohen Volumenströmen wurden dabei instationäre Beschleunigungsdruckverluste in den Fluidausgängen des Trennapparates als signifikanter Einfluss identifiziert. Daraufhin wurden experimentell Trennweisen untersucht, die derartige Beschleunigungseffekte auf das Volumen des Trennapparates begrenzen können. Die Reduzierung der Druckverluste und der Verzicht auf feinporige Membranen erlaubten den Einsatz von Ferrofluiden als disperse Phase, wodurch magnetische Spulen als Sensor eingesetzt werden konnten, die in Folgearbeiten gleichzeitig als Aktuator wirken können.

Parallel wurde die Pfropfenströmung bei der Extraktion aus hochviskosen Medien heraus charakterisiert, da die bisher konventionell eingesetzten Extraktionskolonnen bei geringen Dichtedifferenzen, hohen Grenzflächenspannungen und viskosen Fluiden entweder keinen stabilen Gegenstrombetrieb mehr erlauben oder starke Einbußen im Stofftransport verzeichnen. Um die Pfropfenströmung unter diesen Bedingungen effektiv parallelisieren zu können, wurde ein Verschaltungskonzept erarbeitet und in kleinen Dimensionen experimentell plausibilisiert.

Die Pfropfenströmung wird auf Basis der erarbeiteten Ergebnisse für die Anwendung auf herkömmliche Extraktionsaufgaben nicht als konkurrenzfähig bewertet. Stattdessen konnte die Pfropfenströmung als vielversprechendes Werkzeug überzeugen, wenn die Dichtedifferenz beider Phasen besonders klein ist, die Grenzflächenspannung hoch ist, oder der Wertstoff aus einer hochviskosen Phase herausextrahiert werden soll.

Abstract

The overall trend for process intensification leads to an increased focus on micro process engineering due to the enhanced heat and mass transfer. Especially transfer limited processes like extraction can massively benefit from miniaturization.

The scope of this work was to extend the liquid-liquid slug flow as an extraction tool. Therefore, elimination of existing disadvantages of slug flow were focused. Incomplete model accuracy for prediction of clean phase splitting and huge parallelization effort count to these drawbacks. Beside these developments slug flow was characterized in application areas that are still challenging for typical extraction devices.

Phase split phenomena were investigated numerically. As main part of this work a numerical method was developed that exhibits extremely low parasitic currents. The simulations show significant instationary acceleration effects of the fluids in the surrounding piping. As a result other phase splitting methods were investigated experimentally that are able to limit those effects to the inner device volume. The reduction of pressure drop and the removal of membranes allowed the use of ferrofluids as disperse phase. These ferrofluidic slugs were detectable for monitoring reasons by inductors that may be used as actuators as well in following studies.

Parallel to these investigations the liquid-liquid slug flow was evaluated during extraction form highly viscous media. Typical extraction devices are still challenged by small density differences, high surface tensions or high viscosities due to inhibited counter-current flow or reduced mass transfer. To parallelize slug flow extraction units under these circumstances a new parallelization concept was developed and showed plausible results.

Overall, slug flow is evaluated to be not competitive to state of the art extraction devices for common extraction tasks. Instead slug flow turned out to be a highly recommended tool for extractions incorporating small density differences, big surface tensions or high viscosities.

Deklaration bereits veröffentlichter Inhalte und Reproduktion anderer Arbeiten

Elemente und Ergebnisse der vorliegenden Arbeit sind bereits vorab veröffentlicht worden. Weitere Teile beruhen auf Ergebnissen, die im Rahmen betreuter studentischer Arbeiten und Tätigkeiten am Lehrstuhl für Chemische Verfahrenstechnik (Fakultät BCI, TU Dortmund) entstanden sind oder führen diese weitere. Speziell die experimentell erfassten Rohdaten, welche im Rahmen dieser Arbeit in Kapitel 4 ausgewertet und interpretiert werden, entstammen Arbeiten anderer Autoren wie folgt:

Teil der Arbeit	basierend auf	vorveröffentlicht in
Kapitel 1		[A]
Abschnitt 3.2		[B]
Abschnitt 3.3		[B]
Abschnitt 3.4		[B]
Abschnitt 4.2	[a], [b]	
Abschnitt 4.3	[c]	
Abschnitt 4.4	[d]	
Abschnitt 4.5	[e]	[C]

Vorveröffentlichte Inhalte

[A] Schwarz, Christian Andreas; Kaske, Florian; Arsenjuk, Linda; Agar, David Wiliam, *Die flüssig-flüssig Pfropfenströmung als effektives Werkzeug in der Mikroextraktion*, Posterbeitrag ProcessNet Frankfurt, 2015

[B] Schwarz, Christian Andreas; Agar, David Wiliam, *Reduzierung räumlicher Diskretisierungsfehler in der Volume-of-Fluid-Methode durch analytische Integration der Grenzflächenformfunktion*, Posterbeitrag ProcessNet Bremen, 2018

[C] Schwarz, Christian Andreas; Mendelawi, Mehdy; Agar, David Wiliam, *Kreuz-Gegenstrom-Verschaltung zum Numbering-up der Pfropfenströmung zu Extraktionszwecken*, Kurzartikel Chemie-Ingenieur-Technik 2021, DOI: 10.1002/cite.202100060

Datenerhebung in betreuten studentischen Abschlussarbeiten

[a] He, Yijing, *Vermessung und Weiterentwicklung eines Phasentrenners für die flüssig/flüssig Pfropfenströmung nach dem Parallelitätsprinzip.* Bachelorarbeit, 2017, TU Dortmund

[b] Tasdemir, Bilal, *Stabilitätsuntersuchung einer Parallelströmung in oberflächenmodifizierten rechteckigen Mikrokanälen*, Masterarbeit, 2018, TU Dortmund

[c] Schrief, Julian, *Untersuchungen zur Detektion von Pfropfen in Mikrokapillaren durch Magnetfelder*, Bachelorarbeit, 2018, TU Dortmund

[d] Yuan, Wangbin, *Vermessung von Stofftransportkoeffizienten in viskosen Pfropfenströmungen*, Masterarbeit; 2019, TU Dortmund

[e] Mendelawi, Mehdy, *Experimentelle Untersuchung zur Kreuzgegenstrom-Verschaltung als Numbering-Up Konzept für die Extraktion in Mikrokapillaren durch Pfropfenströmung*, Bachelorarbeit, 2019, TU Dortmund

[f] Raffel, Carola, *Optimierung einer Analytik zur enantioselektiven Extraktion von (R,S)-Phenylsuccinylsäure*, Bacherlorarbeit, 2016, TU Dortmund

[g] Li, Bo, *Charakterisierung von Pfropfenströmungen in parallelisierten Mikrokanälen zu Extraktrionszwecken*, Bacherlorarbeit, 2019, TU Dortmund

Weitere Unterstützung leisteten Frau Carola Raffel und Frau Tanita Six im Rahmen ihrer Forschungspraktika und Folgeanstellungen als studentische Hilfskraft bei Rechercheaufgaben und durch Beihilfe in der Durchführung von Voruntersuchungen zu genannten Arbeiten.

Inhaltsverzeichnis

Kapitel 1

Einleitung

Die Mikroverfahrenstechnik besitzt das Potential zur massiven Prozessintensivierung durch die herausragenden Transporteigenschaften für Wärme- und Stoffströme. [1] Durch die Miniaturisierung der Apparate steigt auf der einen Seite die spezifische Oberfläche innerhalb des Apparates, die für den Transportvorgang zur Verfügung steht, während auf der anderen Seite die Länge des zu überwindenden Transportweges abnimmt. Beide Effekte sind für die Anwendung von wärmeübergangs- oder stofftransportlimitierten Unit Operations in mikroverfahrenstechnischen Apparaten besonders interessant. Ein Beispiel hierfür ist die flüssig-flüssig Extraktion, welche bisher häufig in Extraktionskolonnen durchgeführt wird. Die Kolonnen sind mit Füllkörpern oder strukturierten Packungen gefüllt sind, um die interne Austauschfläche der Phasen zu steigern. Ebenso werden Pulsationen und andere Möglichkeiten der externen Energiezufuhr genutzt, um den Stofftransport weiter zu verbessern. [2] [3] [4] [5] Zu evaluieren ist daher, in welchem Maße und für welche Anwendungsfälle die Mikroverfahrenstechnik gegenüber den makroskopischen Extraktionsapparaten überlegen ist.

Auf dieser Grundlage untersucht die vorliegende Arbeit die flüssig-flüssig Pfropfenströmung als Werkzeug für Extraktionsaufgaben und entwickelt diese weiter. Hierzu wird zunächst der aktuelle Stand des Wissens zu bisher etablierten Extraktionsapparaten in Abschnitt 2.1 zusammengefasst. Im Vergleich dazu werden in Abschnitt 2.2 die bisherigen Fortschritte und Schwierigkeiten in der Anwendung der Pfropfenströmung als Extraktionswerkzeug dargestellt. Besonders die Parallelisierung mehrerer Extraktionsstränge zur Steigerung der Produktionsmenge und eine kostengünstige Implementierung eines einzelnen Strangs sind dabei relevante Themen.

Aus dem aktuellen Stand der Technik resultierend werden in dieser Arbeit zwei potentielle Etablierungsmöglichkeiten für die Pfropfenströmung als Extraktionswerkzeug verfolgt. Auf der einen Seite kann die Pfropfenströmung in ihrer Anwendung auf gängige Extraktionsaufgaben derartig weiterentwickelt werden, dass sie im Vergleich zum Bau bisheriger Extraktionsapparate keine nennenswerten Nachteile mehr aufweist und lediglich durch ihre guten Stofftransportleistungen überzeugt. Auf der anderen Seite kann gezielt nach Anwendungsgebieten gesucht werden, welche mit bisherigen Extraktionsapparaten nur schwer oder praktisch gar nicht erschlossen werden können. Unabhängig von der Etablierungsmöglichkeit ist die Grundlage eines funktionierenden Extraktionsstrangs das stetige Kontaktieren und Trennen der fluiden Phasen. Während die Pfropfenerzeugung bereits gut charakterisiert und er-

forscht ist, weisen bisherige Modelle zur Vorhersage einer stabilen Phasentrennung nachweislich noch Fehler auf. So wird nach der Modellvorstellung eine stabile und reine Phasentrennung vorhergesagt, welche in der Praxis jedoch nicht vorgefunden wird. [6] Dem kann aktuell nur mit besonders kleinen Abmessungen der Trenneinheit und einem dadurch gesteigerten Kapillardruck entgegengewirkt werden, was jedoch unnötige Druckverluste verursacht und dadurch einige Förder- und Parallelisierungskonzepte unattraktiv werden lässt. Die Phasentrennung einer flüssig-flüssig Pfropfenströmung wird daher numerisch in Abschnitt 4.1 untersucht und in einem Trennapparat nachverfolgt, welcher einen geringen Druckverlust aufweist, jedoch wegen unsauberer Trennleistung bisher wenig Anwendung findet. Die Simulationen sollen dabei besonders arm an parasitären Störströmungen sein, welche durch Diskretisierungsfehler entstehen und die Grenzfläche unphysikalisch bewegen oder die Simulation sogar divergieren lassen können. [7] Die hierfür notwendige numerische Methode wird als Hauptbestandteil der Arbeit ausführlich in Kapitel 3 erarbeitet und charakterisiert. Numerische Simulationen der Phasentrennung zeigen dabei die Gründe für das Versagen des Trennapparates auf.

Folgend wird zur Ausarbeitung der ersten Etablierungsmöglichkeit versucht, die Schwierigkeiten in der Anwendung der Pfropfenströmung zu beseitigen. Eine alternative Art der Phasentrennung mit kleinen auftretenden Druckverlusten wird in Abschnitt 4.2 experimentell untersucht. Entwicklungsarbeiten zu einem integrierten Sensor- und Aktuatorkonzept über Magnetfelder und ferrofluidische Pfropfen zur einfacheren und kostengünstigeren Parallelisierung werden in Abschnitt 4.3 gezeigt.

Der alternativen Etablierungsmöglichkeit folgend wird die Stofftransportleistung der Pfropfenströmung in Abschnitt 4.4 an einem Stoffsystem untersucht, welches eine besonders kleine Dichtedifferenz, eine hohe Viskosität und eine ausgeprägte Grenzflächenspannung besitzt. Jedes dieser Kriterien konnte Abschnitt 2.1 folgend als mögliche Limitierung in der Anwendung bisheriger Extraktionsapparate identifiziert werden. Da weiterhin eine anwendbare Parallelisierungsstrategie zur Steigerung der Produktionsmenge fehlt, stellt Abschnitt 4.5 eine Möglichkeit zur drastischen Senkung des Investitionsaufwandes dar.

Abschließend werden in Kapitel 5 die erarbeiteten Resultate verglichen und eine Prognose aufgestellt, welche Anwendungsgebiete die Pfropfenströmung als Extraktionswerkzeug fokussieren sollte.

Kapitel 2

Stand der Technik

Innerhalb dieses Kapitels wird der aktuelle Stand des Wissens im Bezug auf die im Rahmen dieser Arbeit relevanten Themengebiete dargestellt. Abschnitt 2.1 gibt dabei zunächst einen zusammenfassenden Einblick in die Methoden und Apparate zur Bewältigung bisheriger Extraktionsaufgaben. Nachfolgend führt Abschnitt 2.2 in die Mikroverfahrenstechnik ein und setzt den Fokus dabei auf die flüssig-flüssig Pfropfenströmung als Extraktionswerkzeug und ihre Stofftransportleistung, sowie die genutzten Apparate. Weiterführend schafft Abschnitt 2.3 einen Einblick in das relevante Gebiet der Simulation von kapillarkraftdominierten Mehrphasenströmungen, da hierauf die in Kapitel 3 erarbeitete Methode aufbaut.

2.1 Typische Extraktionsapparate

Die Extraktion findet in der Prozesssynthese oft erst dann Anwendung, wenn andere Trennverfahren alleine nicht die gewünschte Trennwirkung erzielen, da der Wertstoff immer nur in eine andere Phase überführt wird. Dies erfordert den anschließenden Einsatz eines weiteren Trennverfahrens, um das Wertprodukt in Reinform zu gewinnen. Dies ist nur dann sinnvoll, wenn die Abtrennung aus der neuen Phase wesentlich einfacher möglich ist. Daher sind die häufigsten Anwendungsgebiete der Extraktion die Auftrennung von Gemischen mit kleiner Siedetemperaturdifferenz, das Aufbrechen von Azeotropen, die Aufreinigung von temperaturempfindlichen Stoffen oder das gleichzeitige Abtrennen mehrerer Komponenten, die sich besonders stark in ihren Siedepunkten unterscheiden. [2] [3] [8] Für die Durchführung der Extraktion stehen eine Vielzahl unterschiedlicher Apparate zur Verfügung. Die am häufigsten verwendeten sollen in diesem Abschnitt grob vorgestellt und charakterisiert werden, um den aktuellen Stand der Technik darzustellen. Üblicherweise werden Extraktionen dabei selten mit mehr als 10 theoretische Trennstufen realisiert. [4]

2.1.1 Mixer-Settler

Eine besonders einfache Realisierung der Extraktion ist in Mixer-Settler-Anlagen möglich. Namensgebend ist das wiederholte intensive Mischen von Extraktionsmittel und Wertstoffphase in jeder Stufe mit anschließender Einleitung der Dispersion in eine Ruhezone zur Phasentrennung. Schematisch ist diese Betriebsweise in Abbildung 2.1 dargestellt. Die Phasentrennung erfolgt lediglich auf Grund der Dichtedifferenz $\Delta\rho$ in Kombination mit der durch die Grenzflächenspannung σ angetriebene

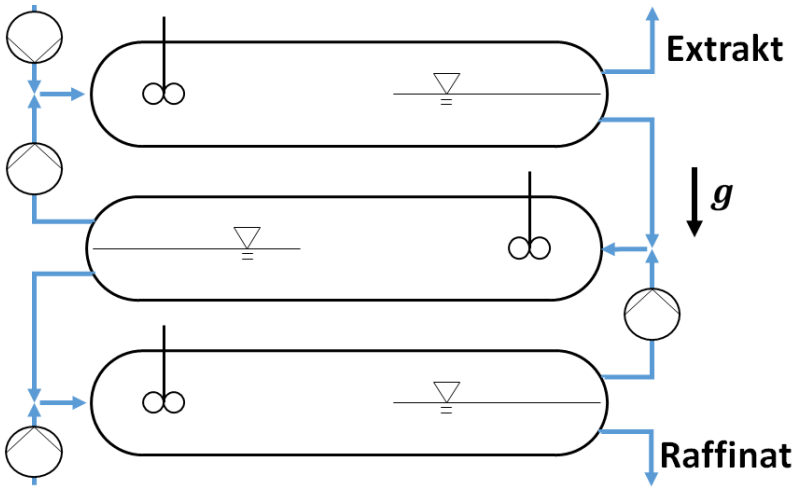


Abbildung 2.1: Schematische Darstellung einer dreistufigen Mixer-Settler-Anlage mit 4 Pumpen. Nachgezeichnet aus [4].

Koaleszenz der Tropfen. Die Koaleszenz in der Ruhezone wird durch den Einbau von Drahtgewebe oder Füllkörpern, welche von der disperse Phasen benetzt werden, gefördert. Dies verkleinert das notwendige Apparatevolumen, steigert jedoch den Druckverlust und die Betriebskosten. [3] Bei korrekter Auslegung kann eine theoretische Trennstufe pro Extraktionsraum realisiert werden. Allgemein benötigen Mixer-Settler-Anlagen allerdings viel Platz und weisen einen großen Flüssigkeitsinhalt auf. [2] [3]. Selten werden daher mehr als 4 Trennstufen realisiert. [3] Vorteile einer Mixer-Settler-Anlage sind jedoch die praktisch freie Wahl des Phasenverhältnisses und die einfache Erweiterbarkeit, falls durch geänderte Prozessbedingungen nachträglich weitere theoretische Trennstufen notwendig sind. [2]

2.1.2 Extraktionskolonnen

Für die Umsetzung einer Extraktionsaufgabe werden Kolonnen am häufigsten eingesetzt, da sie durch ihre variablen Gestaltungsmöglichkeiten einen breiten Anwendungsbereich bieten und mittlerweile gut untersucht und charakterisiert sind. Die einzelnen Varianten von Extraktionskolonnen werden in diesem Unterabschnitt mit ihren jeweiligen Anwendungsgebieten beschrieben und zusammengefasst.

Allgemein lassen sich Extraktionskolonnen in zwei Kategorien einteilen: Kolonnen ohne externe Energiezufuhr (z.B. Siebbodenkolonne, Füllkörperkolonne, Packungskolonnen) und Kolonnen mit externer Energiezufuhr (z.B. Sprühkolonne, pulsierte Siebbodenkolonne, pulsierte Füllkörperkolonne, pulsierte Packungskolonnen, Rotating Disc Contactor, Kühni-Kolonne). Die externe Energiezufuhr wird zur Steuerung der Tropfengröße im Inneren der Kolonne eingesetzt und verbessert dadurch die Extraktionsleistung.[1] Angestrebt werden meist Tropfengrößen im Bereich von 1 mm bis 4 mm ([4]), deren Realisierung aber besonders bei Systemen mit hoher Grenzflächenspannung nur durch externe Energiezufuhr möglich ist ([2]). Die Energiezufuhr erfolgt meist durch Rührorgane oder eine Auf- und Abwärtsbewe-

[1] Die Sprühkolonne nimmt dabei eine gewissen Sonderstellung ein. Es erfolgt zwar eine erhebliche Energiezufuhr zur Verteilung der dispersen Phase als Spray, jedoch geht die eingestellte Größenverteilung schnell wieder verloren. Primäres Ziel der Energiezufuhr ist hingegen die Steuerung der Tropfengröße in der gesamten Kolonne.

gung der enthaltenen Fluide (Pulsation). Seltener wird eine Pulsation der Einbauten vorgenommen, doch auch dies ist bereits realisiert worden (Karr-Kolonne). Im allgemeinen Fall werden in der Kolonne die kontinuierliche und die disperse Phase im Gegenstrom aneinander vorbeigeführt, indem die schwere Phase am Kopf und die leichte Phase am Sumpf der Kolonne aufgegeben wird. Der dabei stattfindende Stofftransport beeinflusst stark die lokalen Stoffwerte, wie Dichte, Viskosität und Grenzflächenspannung, und nimmt dadurch massiven Einfluss auf die Hydrodynamik, welche wiederum den Stofftransport beeinflusst. Diese interne Kopplung gestaltet die quantitative Berechnung von Extraktionskolonnen daher schwierig. Sie werden den Mixer-Settler-Anlagen aus wirtschaftlichen Gründen jedoch oft vorgezogen. [3] Welches Fluid die kontinuierliche Phase in der Kolonne bildet wird häufig durch das Anfahrverhalten beeinflusst. Für einen optimalen Betrieb der Kolonne, sollten durch die Wahl der kontinuierlichen Phase dabei möglichst alle der folgenden Punkte erfüllt sein:

- Der Stofftransport sollte von der kontinuierlichen Phase in die disperse Phase erfolgen. Andernfalls begünstigen Marangoni-Konvektionen, welche durch die Konzentrationsabhängigkeit der Grenzflächenspannung hervorgerufen werden, eine Koaleszenz der Tropfen. [2] [3] [4]

- Das Fluid, welches die Einbauten der Kolonne und die Kolonnenwand besser benetzt, sollte als kontinuierliche Phase gewählt werden. Eine gut benetzende disperse Phase führt zu einer Koaleszenz der Tropfen. [2] [3] [4]

- Es sollte ein Flüssigkeitsanteil an disperser Phase von ca 5-15 % [4], maximal 30 % [3] angestrebt werden. Demnach sollte die Phase mit dem größeren Volumenstrom als kontinuierliche Phase gewählt werden.

Ein Großteil der Extraktionsaufgaben lässt sich durch Extraktionskolonnen bewältigen. Typische Kolonnenarten sind dabei Sprühkolonnen, Siebbodenkolonnen, Füllkörperkolonnen, Packungskolonnen und gerührte Kolonnen, die jeweils pulsiert oder nicht pulsiert betrieben werden können. Beschreibungen dieser Bauarten folgen in anschließenden Unterabschnitten. Praktisch werden in diesen Apparaten bis zu 15 theoretische Trennstufen realisiert. [2] [3] [4] Oft lassen sich allerdings auch durch exotische Materialien im Apparatebau nicht alle genannten Kriterien gleichzeitig erfüllen, was Raum für die Anwendung anderer Extraktionsapparate lässt.

2.1.2.1 Sprühkolonnen

Sprühkolonnen stellen die einfachste Art von Extraktionskolonnen dar. Ohne weitere Einbauten im Inneren wird die Tropfenphase in der Kolonne dispergiert. Nach wenigen Tropfendurchmessern erreichen die Tropfen ihre Endgeschwindigkeit [4] und interagieren frei miteinander, wodurch schnell die aufgegebene Tropfengrößenverteilung verloren geht. Da die Strömung der Phasen nur durch die Schwerkraft angetrieben wird, sind Sprühkolonnen nur bei Stoffsystemen mit einer Dichtedifferenz $\Delta\rho \geq 150\,\mathrm{kg/m^3}$ effektiv einsetzbar. [4] Die freie Strömung im Inneren ermöglicht zudem die Ausbildung eines radialen Geschwindigkeitsprofils, was die axiale Dispersion der Phasen verstärkt und somit die Trennleistung negativ beeinflusst. [8] Sprühkolonnen werden daher nur bei einfachen Trennaufgaben eingesetzt.

2.1.2.2 Bodenkolonnen

Im Gegensatz zu Sprühkolonnen sind in Bodenkolonnen Lochplatten eingezogen, welche typischerwiese Löcher in der Größe von 2-4 mm [4], maximal 6 mm [3], enthalten. Durch die Löcher treten aufgrund der Schwerkraft abwechselnd kontinuierliche und disperse Phase hindurch. Die hierfür zur Verfügung stehende freie Fläche beeinflusst dabei stark den maximalen Durchsatz der Kolonne und beträgt bei einer Bodenplatte in der Regel ungefähr 25 % [4]. Die Durchtrittsöffnungen können dabei als einfache Löcher oder mit weiteren Aufsätzen, wie z.B. Glockenköpfen, ausgeführt sein. Die Verwendung solcher Einbauten mindert den maximalen Durchsatz durch den erhöhten Druckverlust, verbessert aber über eine gezielte Steuerung der Tropfengrößen den Stofftransport. So erreichen normale Glockenböden typischerweise einen Bodenwirkungsgrad von 0,1 - 0,2, während Siebböden auch Bodenwirkungsgrade von 0,3 - 0,6 ([2]) oder sogar 0,7 ([8]) erreichen können. Werden Bodenkolonnen pulsiert betrieben, kann die Trennleistung weiter verbessert werden. Pulsierte Bodenkolonnen benötigen dabei eine Dichtedifferenz $\Delta\rho$ von mindestens 70 $\mathrm{kg/m^3}$ und können dabei disperse Phasen mit einer Viskosität η von bis zu 100 mPa · s handhaben [3]. Der Pulsationshub a nimmt dabei häufig Werte im Bereich von 0,8-1,2 cm an, während die Pulsationsfrequenz f so gewählt wird, dass die Pulsationsintensität $a \cdot f$ Werte zwischen 1-2,5 cm/s annimmt [4].

2.1.2.3 Füllkörper- und strukturierte Packungskolonnen

Eine Alternative zu Lochplatten als Einbauten bieten strukturierte Packungen und Füllkörper, die in ihrer Form zur Steuerung der Hydrodynamik und in der Materialwahl zur Beeinflussung der Benetzungseigenschaften frei gestalten werden können. Beliebte Füllkörperformen sind z.B. Raschigringe, Berlsättel, Pallringe aus Metall, Keramik und Kunststoffen. [9] Die Füllkörper oder Packungen nehmen dabei einen großen Teil des Innenraumes ein. Die Ausbildung von Kanalströmungen in der Schüttung ist beim Einsatz von Füllkörpern möglichst zu vermeiden, was den Aufbau solcher Kolonnen herausfordernd macht. Für die Grenze der maximal möglichen Flüssigkeitsbelastung ist in der Literatur eine Beziehung in der Form von Gl. 2.1 gängig, wie sie auch in den Untersuchungen von Watson, McNeese und Carroad zur Beschreibung verwendet wird. [10]

$$\sqrt{w_{kont.}} + K_1 \cdot \sqrt{w_{disp.}} = K_2 \tag{2.1}$$

Die Parameter K_1 und K_2 sind dabei lediglich von Stoffdaten und Geometrieparametern abhängig. Für einen stabilen Gegenstrombetrieb ergibt sich aus einer gewählten Geschwindigkeit w_i dann die maximal mögliche Geschwindigkeit der anderen Phase. Eine Übersicht verschiedener weiterer Korrelationen stellten Houlihan und Landau zusammen. [11] Auch Mackowiak stellt diverse Beschreibungen gegenüber und entwickelte eigene Ansätze zur Charakterisierung. [9] Besonders gute Stofftransportleistungen werden dann erzielt, wenn die Geschwindigkeit der kontinuierlichen Phase $w_{kont.}$ 80 % der maximal zulässigen Geschwindigkeit ([2] [3]) beträgt und für die Geschwindigkeit der dispersen Phase $w_{disp.}$ 60 - 85 % ([2]), bzw. ebenfalls genau 80 % ([3]), des Maximums angesetzt werden. Eine Pulsation der Fluide bewirkt dabei wieder eine Steigerung des Stofftransportes. Dies senkt die Höhe einer theoretischen Trennstufe, wodurch die Kolonne kleiner gebaut werden kann. Eine zu starke Pulsation lässt jedoch die Belastungsgrenze der Kolonne stark absinken.

[4] Hierdurch sinkt die zulässige Geschwindigkeit in der Kolonne und für das Erreichen eines geforderten Volumenstroms werden wieder breitere und größerer Apparat benötigt. Es ist daher ein Kompromiss zwischen gesteigertem Stofftransport und verkleinerter Belastungsgrenze der Kolonne zu finden. Ähnliche Grenzen und Bedingungen gelten für Packungskolonnen, die laut Pfennig bis zu einer minimalen Dichtedifferenz $\Delta\rho$ von $70\,\mathrm{kg/m^3}$ betrieben werden können. [3] Für Medien mit hoher Viskosität η sind Packungskolonnen hingegen schlecht geeignet. [8] Dafür erbringen pulsierte Packungskolonnen besonders bei sehr kleinen Grenzflächenspannungen σ außerordentlich gute Ergebnisse und erreichen spezifische Oberflächen von bis zu $500\,\mathrm{m^2/m^3}$. [3] Durch ihre mittlerweile sehr gute Beschreibbarkeit sowie ihre Flexibilität im Betrieb und der Auslegung werden pulsierte Packungskolonnen für Standardextraktionsaufgaben[2] daher oft in Betracht gezogen und stellen häufig die günstigste Alternative dar. [3] Da die internen Strukturen von Füllkörper- und Packungskolonnen aus Wiederholungseinheiten bestehen, weist die Hydrodynamik im Inneren kaum eine Abhängigkeit vom Durchmesser der Kolonne auf. Erst im Randbereich wird die Hydrodynamik durch die Wandnähe beeinflusst. Füllkörper- und Packungskolonnen sind daher wesentlich einfacher als Sprühkolonnen zu skalieren ([3]) und auch bei Teillast gut zu betreiben ([8]).

2.1.2.4 Gerührte Extraktionskolonnen

Neben der Pulsation von Fluidraum oder Einbauten kann auch durch Rührorgane ein Energieeintrag in der Kolonne stattfinden. Als Rührorgane werden dabei je nach Anwendung die unterschiedlichsten Geometrien eingesetzt. Für hochviskose disperse Phasen eignen sich schnell rotierende Scheiben (Rotating Disc Contactor), welche die Tropfen redispergieren sollen ([3]), während unter anderen Bedingungen auch normale Blattrührer zum Einsatz kommen können ([4]). Zwischen den Rührorganen werden dabei Statorbleche installiert, welche die Rührorgane darin unterstützen, die axiale Rückvermischung der Kolonne zu unterdrücken. [4] [5] Neben der Homogenisierung der Hydrodynamik beeinflussen die Rührorgane ebenso die Tropfengröße, indem sie große Tropfen zerteilen. Eine höhere Drehzahl erzeugt dabei umso kleinere Tropfen. [3] [4] [5] Besonders kleine Tropfen senken allerdings die maximale Belastbarkeit der Kolonne. [2] Für einen geforderten Durchsatz sind dann wiederum besonders breite Kolonnen notwendig. Diese Kopplung bedingt, dass technisch bisher nur gerührte Kolonnen mit bis zu 3 m Durchmesser realisiert worden sind. [3] Eine Kombination aus Rührelementen und Pulsation ist aktueller Gegenstand der Forschung. So untersuchen Soboll et al. in einer Kolonne mit einem Durchmesser von 15 mm die Kopplung von Rührelementen mit verschiedenen Pulsationsmethoden und erreichen Trennleistungen von bis zu 25 theoretischen Trennstufen pro Meter. [5] Für die einfachere und kostensparendere Bestimmung des charakteristischen Flutpunkts existieren nach Schmalenberg et al. ebenfalls bereits Strategien. [12]

Am Beispiel des Stoffsystems Toluol(d)/Aceton(←)/Wasser(c) gibt Abbildung 2.2 eine Übersicht der jeweiligen Belastungsbereiche und Trennleistungen der vorgestellten Extraktionskolonnen. Die Siebbodenkolonne weist die schlechteste Trennleistung auf, bietet jedoch den größten Belastungsbereich, während strukturierte

[2] Unter einer Standardextraktionsaufgabe wird in diesem Zusammenhang eine Extraktion ohne herausragende Besonderheiten in den Stoffwerten verstanden. $\Delta\rho > 50\,\mathrm{kg/m^3}$; $\eta \simeq 1\,\mathrm{mPa \cdot s}$; $\sigma > 5\,\mathrm{mN/m}$; $n_{th} < 10$ [3]

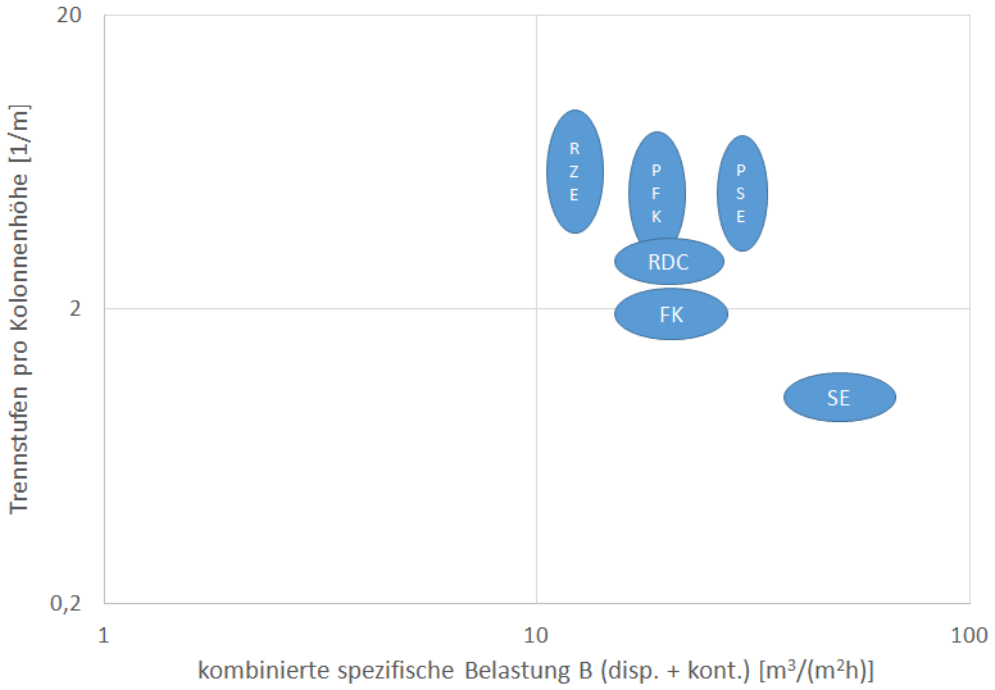


Abbildung 2.2: Einordnung der Belastungsbereiche und Trennleistungen ausgewählter Extraktionskolonnen für das Stoffsystem Toluol(d)/Aceton(←)/Wasser(c) nach Mersmann: Siebbodenkolonne (SE), pulsierte Siebbodenkolonne (PSE), Füllkörperkolonne (FK), pulsierte Füllkörperkolonne (PFK), Rotating Disc Contactor (RDC), Rührzellenextraktor (RZE). Nachgestellt aus [4].

Kolonnen mit Einbauten die Trennleistung auf Kosten der Belastbarkeit steigern. Allgemein steigert ein Energieeintrag jeglicher Form die Trennleistung und senkt die Belastbarkeit. Verursacht wird dies durch wesentlich kleinere Tropfen, die mehr Austauschfläche bieten, aber auch langsamer aufsteigen/absinken. [4]

2.1.3 Zentrifugalextraktoren

Da die Phasentrennung in Extraktionskolonnen und Mixer-Settler-Anlagen primär durch die Dichtedifferenz $\Delta\rho$ angetrieben wird, kann dies zu extrem großen Apparaten führen, wenn die Dichten beider Phasen sehr ähnlich sind. Zentrifugalextraktoren können durch Zentrifugieren der Dispersion auch bei Dichtedifferenzen in der Größenordnung von 30 kg/m^3 noch eine saubere Phasentrennung zu realisieren. [2] Dabei können in einem Apparat bis zu 30 theoretische Trennstufen realisiert werden ([2]), was jedoch durch hohe Kosten in der Anschaffung und dem Betrieb erkauft werden muss ([3]). Diese Umstände machen Zentrifugalextraktoren zu vergleichsweise selten eingesetzten Apparaten. [4] Bei kleinen Dichtedifferenzen sind sie dennoch das Mittel der Wahl. [8]

2.2 Die Anwendung der Mikroverfahrenstechnik zu Extraktionszwecken

Die Mikroverfahrenstechnik beschäftigt sich allgemein mit der Miniaturisierung verfahrenstechnischer Apparate, um pro Volumen viel Austauschfläche zur Verfügung zu stellen. Dies begünstigt besonders solche Verfahrensschritte, die durch Stoff- und Wärmetransport limitiert sind. Ein Vielzahl von Anwendungen sind denkbar, wie Kenig et al. in ihrer Arbeit zeigen. [13] Im Rahmen dieser Arbeit soll speziell die flüssig-flüssig Extraktion als verfahrenstechnische Grundeinheit betrachtet und hierfür der Einsatz der Pfropfenströmung als Strömungsform in Mikroapparaten weiterentwickelt werden. Daher begrenzt sich die folgende Darstellung hauptsächlich auf diese beiden Gebiete und deren Kombination. Für Anwendungen als Reaktor sei auf Kashid et al. ([1]) und Bobers et al. ([14]) verwiesen. Eine Übersicht im Bezug auf Gas-Flüssig-Strömungen erarbeiteten Sobieszuk et al. ([15]).

2.2.1 Strömungsformen in Mikroapparaten

Die Pfropfenströmung zeichnet sich durch die langgezogenen Blasen (gas/flüssig) bzw. Tropen (flüssig/flüssig) der dispersen Phase aus, die in der Mikrokapillare fast den gesamten Durchmesser ausfüllen. [16] Zwischen den Pfropfen befindet sich die kontinuierliche Phase, welche die disperse Phase je nach Benetzbarkeit der beteiligten Phasen vollständig umhüllt. In diesem Fall spricht man von einer Strömung mit Wandfilm. Das Auftreten der Pfropfenströmung in Mikrokanälen hängt von den Betriebsbedingungen, den Stoffparametern und der Kanalgeometrie ab. Holbach et al. unterscheiden zwischen der Parallelströmung, der Blasenströmung, der Pfropfenströmung und der unregelmäßigen dispersen Strömung in Form einer Art Emulsion und untersuchten den Einfluss der Strömungsgeschwindigkeiten auf die resultierende Strömungsform für Kombinationen von Wasser mit n-Heptan, n-Nonan und n-Undecan. [17] Weiterhin ist die Ringströmung als mögliche Strömungsform zu nennen. Die Grenzen für das Auftreten genannter Strömungsformen untersuchten Kashid et al. ([16]) für das Stoffsystem Toluol/Wasser in verschiedenen Pfropfenerzeugern. Auch Biswas et al. ([18]) untersuchten die Strömungsformen für dieses Stoffsystem, während Scheiff et al. ([19]) das Auftreten von Pfropfenströmung beim Einsatz von ionischen Flüssigkeiten als disperse Phase betrachteten. Allgemein ist erkennbar, dass die Pfropfenströmung bei niedrigen Strömungsgeschwindigkeiten und dominierender Grenzflächenspannung auftritt. Zur Charakterisierung der Strömung haben sich in der Literatur die Reynolds-Zahl, die Kapillar-Zahl und die Weber-Zahl etabliert (Gl. 2.2, Gl. 2.3, Gl. 2.4)

$$\mathrm{Re} = \frac{\rho \cdot u \cdot L}{\eta} \tag{2.2}$$

$$\mathrm{Ca} = \frac{u \cdot \eta}{\sigma} \tag{2.3}$$

$$\mathrm{We} = \frac{\rho \cdot u^2 \cdot L}{\sigma} \tag{2.4}$$

Sie geben das Verhältnis von Trägheitskräften zu viskosen Kräften (Re), viskosen Kräften zu Grenzflächenkräften (Ca) und Trägheitskräften zu Grenzflächenkräften

an (We). In Mehrphasenströmungen ist im Falle von Mehrdeutigkeiten jeweils die Bezugsphase anzugeben. Ein allgemeingültiges Kriterium für das Auftreten von Pfropfenströmung konnte in der Literatur nicht gefunden werden. Kleine Kapillar-Zahlen begünstigen jedoch bei ausreichend unterschiedlichen Benetzungseigenschaften der Flüssigkeiten stets die Ausbildung einer Pfropfenströmung. Hohe Reynolds-Zahlen begünstigen hingegen das Auftreten einer Ringströmung. Sofern Pfropfenströmung auftritt, bilden sich innerhalb der kontinuierlichen und dispersen Phase charakteristische Wirbelmuster aus, die für eine intensive Durchmischung der Phasen sorgen. Dittmar untersuchte den Einfluss der Parameter Re, Ca und We auf die Topologie der Wirbelmuster. [20] [21]

2.2.2 Eingesetzte Apparate der Mikroextraktion

Die Erzeugung einer Pfropfenströmung, die Verweilzeitstrecke und die anschließende Phasentrennung der Pfropfenströmung bilden eine µ-Mixer-Settler-Einheit, welche nur im Gleichstrom betrieben werden kann. Zur Steuerung der Pfropfeneigenschaften, des Stofftransportes und für die effektive Phasentrennung haben sich verschiedene Apparate etabliert.

2.2.2.1 Pfropfenerzeuger und Extraktionsstrecke

Eine sehr einfache Möglichkeit zur Erzeugung einer Pfropfenströmung ist eine einfache Zusammenführung der beiden Flüssigkeiten. Je nach eingeschlossenem Winkel zwischen den Flüssigkeitszuläufen und der angeschlossenen Extraktionsstrecke spricht man auch von T- und Y-Mischern. Diese Mischer weisen jedoch allgemein eine Abhängigkeit der Pfropfenlänge von der Strömungsgeschwindigkeit und dem Phasenverhältnis auf. Eine bessere Kontrollierbarkeit der Pfropfenlänge erreichten Kaske et al. mit Hilfe einer mechanischen Stellschraube zur künstlichen und variablen Verengung des Kontaktraums beider Phasen. [22] [23]. Für das Stoffsystem Wasser/Toluol erreichten sie Pfropfenlängen im Bereich von 2 mm - 13 mm. Eine noch bessere Kontrollierbarkeit erzielten Arsenjuk et al. mit einem koaxialen Mischer, in dem die kontinuierliche Phase ringförmig um eine Nadel herumgeführt wird. [24] An der Nadelspitze wächst die disperse Phase bis zu einer durch den Kontaktraum begrenzten Größe an und wird abgeschert. Die Position der Nadel ist variabel und beeinflusst damit das Volumen des Kontaktraums und die Größe der abgescherten Tropfen. Neben den invasiven Eingriffen in die Mischergeometrie existiert zudem die Möglichkeit, ein elektrisches Wechselfeld im Bereich der Pfropfenerzeugung anzulegen, welche von Antweiler et al. genutzt wurde, um die Pfropfenlänge in gewissen Grenzen zu steuern. [25]

Im Anschluss an die Pfropfenerzeugung folgt die Verweilzeitstrecke. Über die Benetzungseigenschaften des Wandmaterials kann darin eine Phaseninversion erzwungen werden, welche den Stofftransport kurzzeitig intensiviert. Kontaktwinkel in der Größenordnung von 90° begünstigen dabei meist eine Strömung ohne Wandfilm, sobald die disperse Phase erst einmal Wandkontakt erhalten hat. Durchmesser und geometrischer Verlauf der Verweilzeitstrecke bestimmen dabei maßgeblich den stattfindenden Stofftransport (siehe Unterabschnitt 2.2.3).

2.2.2.2 Phasentrenner

Im Anschluss an die Verweilzeitstrecke erfolgt die Trennung der kontaktierten Phasen. In makroskopischen Apparaten kann nach Eigenschaften wie Siede- und Schmelzpunkt oder der Dichte getrennt werden. In mikroskopischen Bauteilen können hingegen nur schwer Temperaturgradienten aufrecht erhalten werden und die Grenzflächenspannung dominiert als Oberflächenkraft häufig über die Schwerkraft. Daher bietet sich zur Aufreinigung in Mikroapparaten ein weiteres Trennkriterium an, welches auf der Benetzbarkeit der Fluide an verschiedenen Oberflächen beruht. In Grenzflächen aus Fluiden untereinander oder an Feststoffen ist Energie enthalten, die nach Minimierung strebt. Daraus resultieren die Ausbildung eines Meniskus mit stoffsystemspezifischem Kontaktwinkel und eines nach Gl. 2.5 gegebenen Kapillardrucks.

$$\Delta p_{cap} = \sigma \cdot \kappa \tag{2.5}$$

Hierin ist σ die Grenzflächenspannung der Fluide zueinander und κ die mittlere Krümmung der Grenzfläche an betrachteter Stelle. Diese Druckdifferenz Δp_{cap} ist zu überwinden, wenn Fluid 1 ein besser benetzendes Fluid 2 von einer Oberfläche verdrängen soll. Auf diesem Trennprinzip sind bereits diverse Mirkoapparate zur Phasentrennung realisiert worden. Abbildung 2.3 gibt hierzu eine Übersicht der verschiedenen Bauformen von Phasentrennern, die bereits zur Aufreinigung von Pfropfenströmungen eingesetzt wurden. So entwarfen Kashid et al. einen Y-Splitter, in

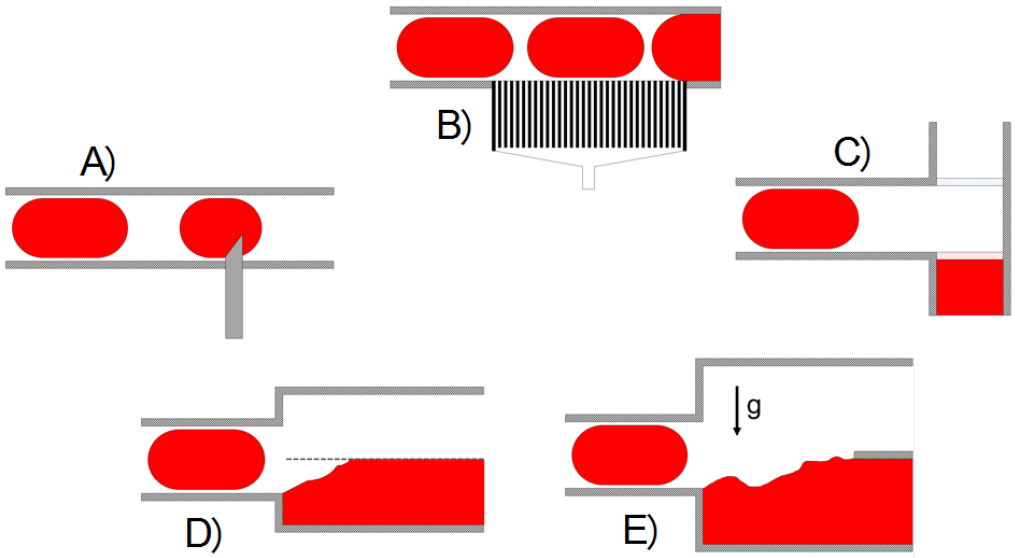


Abbildung 2.3: Schematische Darstellung verschiedener Trennapparate zur Aufreinigung von Pfropfenströmungen: A) direkter Seitenabzug, B) Porenkämme und Siebe, C) Membranen, D) Umwandlung in eine Paralellströmung, E) Abtrennung durch Schwerkraft in Mikrodekantern.

dem die einlaufende Strömung entweder in eine angeschlossene Stahl- oder Teflonkapillare eintreten muss. Der zu überwindende Kapillardruck verhindert den Eintritt der wässrigen Phase in die Teflonkapillare, während die organische Phase einen gewissen Kapillardruck beim Eintritt in die Stahlkapillare erfährt und daher bevorzugt durch die Teflonkapillare abfließt. [26] Mit abweichenden Materialien nutzte auch Susanti et al. diese Art von Y-Phasentrenner. [27] Ähnlich konstruierten Scheiff et al.

([6]) und stachen mit einer Stahlnadel in die selbst hergestellte Kapillaren aus Polyolefin, während Peroni et al. ([28]) die Geometrie mit beschichteten Glasskapillaren realisierten (jeweils Trennprinzip A). Ufer nutzte ebenso eine Y-Geometrie nach Trennprinzip A zur Phasentrennung seiner Pfropfenströmung. [29] Weitere Apparate erweitern die Vorgehensweise des seitlichen Phasenabzugs und gruppieren eine Vielzahl seitlicher Abflusskanäle zu einer Art Porenkamm (Trennprinzip B). Beispiele hierzu finden sich bei Bussmann ([30]), Castell et al. ([31]), Fries et al. ([32]) und Guenther et al. ([33]). Werden die Trennwände der seitlich angeordneten Poren aufgebrochen, so entstehen langgezogene Schlitze, durch welche nur die jeweils besser benetzende Phase abfließen kann. Die Apparatur von Gaakeer et al. verfolgt diesen Ansatz und bietet für die organische und die wässrige Phase jeweils einen gut benetzbaren Abflusschlitz. [34] Je größer die Abflussöffnungen sind, desto geringer ist der hydrodynmaische Druckverlust beim Duchtritt des Fluids. Jedoch fällt auch der zur Trennung verfügbare Kapillardruck ab, da große Öffnungen auch Phasengrenzflächen mit kleineren Krümmungswerten κ zulassen. Den mit Abstand größten Kapillardruck weisen daher Trennapparate mit porösen Membranen auf, die anstatt der Porenkämme seitlich in Flussrichtung platziert werden (Kombination Trennprinzip B + C). Diese Konstruktionsweise verfolgten Adamo et al. ([35]) und Kralj et al. ([36]) in ihren Arbeiten. Durch den hohen Kapillardruck ist eine seitliche Anordnung der Membranen nicht unbedingt notwendig (Trennprinzip C). Einen Kompromiss stellt der Einsatz von Siebmaterial dar, welches der durchtretenden Strömung wenig Widerstand bietet, aber trotzdem nur relativ kleine Öffnungen aufweist. Holbach und Kurt konstruierten in ihren Arbeiten jeweils Phasentrenner nach diesem Ansatz (je Trennprinzip C).[17] [37]. Neben der direkten Auftrennung einer Pfropfenströmung kann jedoch auch zunächst eine Umwandlung in eine Parallelströmung erfolgen. Die Auftrennung von Parallelströmungen in sehr engen Kanälen ist vergleichsweise einfach, wie Aota et al. ([38]), Kolehmainen et al. ([39]) und Fries et al. ([32]) zeigten (Trennprinzip D). Jede dieser Konstruktionen kann dabei mit der korrekten Anordnung im Schwerefeld kombiniert und verbessert werden (Trennprinzip E). Eine Kombination aus dem Y-Splitter nach Kashid et al. ([26]) und einer Unterstützung durch die Trennung nach Dichtedifferenz findet sich z.B. bei Kaske et al.. [22] [23] Weitere Anwendungen von Phasentrennern finden sich in der Arbeit von Kenig et al. und den dort angegebenen Quellen. [13] Je nach Bauweise der Trennapparate werden, wenn überhaupt, unterschiedliche Kriterien für die korrekte Funktionsweise der Apparate angegeben, die sich jedoch auf Gl. 2.6 reduzieren lassen, sofern sie auf beide Fluide angewendet wird.

$$\Delta p_{cap,i} \geq \Delta p_{hydr,i} \tag{2.6}$$

Dabei ist $\Delta p_{cap,i}$ der Kapillardruck, den Fluid i erfährt, wenn es in den Auslass der anderen Phase einzudringen versucht und $\Delta p_{hydr,i}$ der Druckverlust, den Fluid i beim Verlassen der Trenneinheit durch den dafür jeweils vorgesehenen Auslass verursacht. Erfährt nur eine Phase, beispielsweise Fluid 1, einen Kapillardruck beim Eindringen in den nicht vorgesehenen Auslass, wie es z.B. bei den bisherigen Anwendungen von Porenkämmen und Membranen der Fall ist, gilt ein leicht modifiziertes Kriterium nach Gl. 2.7, um den Durchtritt der abzutrennenden Phase 2 auch zu erzwingen.

$$\Delta p_{cap,1} \geq \Delta p_{hydr,1} > \Delta p_{hydr,2} \tag{2.7}$$

Ist $\Delta p_{hydr,2}$ kleiner als $\Delta p_{hydr,1}$ wird sich der Durchfluss durch Auslass 2 immer weiter erhöhen. Dies ist solange möglich, bis der gesamte Anteil $\dot{V}_2$ am Gesamtvo-

lumenstrom $\dot{V}_{ges}$ auch durch Auslass 2 abfließt oder eine unerwünschte Gleichheit $\Delta p_{hydr,1} = \Delta p_{hydr,2}$ und damit eine Verletzung von Gl. 2.7 erreicht wird. Die Arbeit von Kralj et al. diskutiert dabei nach Kenntnis des Autors als einzige Arbeit, dass die angegebenen Ungleichungen zu allen Zeitpunkten erfüllt sein müssen. [36] Wird z.B. durch ein sehr langes Segment kontinuierlicher Phase der gesamte Trennapparat mit Fluid 1 geflutet, existiert kein Volumenstrom in Auslass 2. Stattdessen strömt lediglich Fluid 1 durch Auslass 1. Dies geschieht dann jedoch aus Kontinuitätsgründen mit einem Gesamtvolumenstrom von $\dot{V}_{ges} = \dot{V}_1 + \dot{V}_2$. Dies stellt höhere Anforderungen an $\Delta p_{cap,1}$ als der eigentlich vorgesehene Betriebspunkt mit $\dot{V}_1 < \dot{V}_{ges}$ in Auslass 1. Auch bei kompletter Flutung des Trennapparates mit Fluid 2 soll die Trennung weiterhin gewährleistet sein. Hierzu muss trotz fehlenden Durchflusses in Auslass 1 ebenfalls $\Delta p_{hydr,1} > \Delta p_{hydr,2}$ gelten, was Adamo et al. z.B. automatisch durch das gedehnte Diaphragma in der genutzten Trenneinheit gewährleisten konnten. [35] Die Druckverluste $\Delta p_{hydr,i}$ beinhalten dabei explizit alle Beiträge, also sowohl das Durchfließen der Trennstruktur selbst, wie auch der nachfolgenden Leitungen. Trotz Einhaltung von Gl. 2.6 erreichen etliche auf Trennprinzip A und dessen Varianten basierende Arbeiten ([26], [6], [34]) jedoch keine gute Modellübereinstimmung. Die Annahme einer bereits vollzogenen perfekten Trennung zur Auslegungen der notwendigen Kapillardrücke kann dies jedoch nur bedingt erklären. Gaakeer et al. merkten hierzu an, dass die eintretende Strömungsform an sich unregelmäßig ist. Demnach kann nicht garantiert werden, dass die Geschwindigkeiten in den Trennöffnungen und den Auslässen zeitlich konstant sind, was Störungen verursachen kann. [34] Scheiff untersuchte die Phasentrennung zudem numerisch, nutzte dabei jedoch ein Rechengitter, welches die Höhe und Breite des genutzten Rechteckkanals in maximal 40 Zellen auflöste. [40] Eine korrekte Abbildung des Wandfilms und aller von Dittmar ([21]) nachgewiesenen internen Wirbelstrukturen ist daher fraglich. Er klärte jedoch auf, dass der Kapillardruck Ungleichheiten in den hydrodynamischen Druckverlusten beider Ausgänge ausgleicht. Die von ihm teilweise verzeichnete unsaubere Phasentrennung führt er auf ein zu langsames Eindringen des dispersen Pfropfens in den vorgesehen Auslass und allgemein zu große Druckverluste in den zu langen Auslässen zurück. [40] Eine genaue Untersuchung des instationären Verhaltens konnte in der Literatur darüber hinaus nicht gefunden werden. Die Modelle zur Vorhersage eines stabilen und sauberen Trennverhaltens sind daher noch unzureichend. Dem kann bisher nur durch besonders kleine Spalt- und Porendurchmesser entgegengewirkt werden, was unnötige Druckverluste erzeugt und zudem den Einsatz einiger Stoffsysteme, wie z.B. Suspensionen, wegen Verblockungsgefahr einschränkt.

2.2.3 Stofftransport in flüssig-flüssig Pfropfenströmungen

Hauptgrund für den Einsatz der Mikroverfahrenstechnik sind die herausragenden Eigenschaften bezüglich des Stoff- und Wärmetransportes innerhalb der Apparate. Speziell der Stofftransport in flüssig-flüssig Pfropfenströmungen ist bereits vielfach untersucht worden. Kashid et al. zeigten, dass kleinere Kapillardurchmesser den Stofftransport stark verbessern und höhere Pfropfengeschwindigkeiten zwar den Stofftransportkoeffizienten steigern, die verkürzte Verweilzeit bei konstanter Länge der Extraktionsstrecke allerdings eine niedrigere Extraktionsleistung verursacht. [26]. Die Ergebnisse von Kaske et al. bestätigen diese Befunde und zeigen zudem, dass eine Zunahme der Pfropfenlänge einen Abfall des Stofftransportes bewirkt. [22]

[23] Kurt untersuchte den Einfluss einer gewundenen Extraktionsstrecke im Coiled Flow Inverter und zeigte eine Steigerung des Stofftransports von fast 50 % im Vergleich zu geradlinigen Kapillaren, da die Kapillarwindungen eine Sekundärströmung in den Pfropfen induzieren, welche die Totzonen der Wirbelstruktur aufbricht. [37] Dabei wurden jedoch oft nur niedrigviskose Stoffsysteme untersucht. Scheiff et al. experimentierten mit ionischen Flüssigkeiten als disperse Phase ([19]), doch finden sich sonst vergleichsweise wenige Stofftransportmessungen mit viskosen Systemen in der Literatur, wie auch Tsaoulidis und Angeli in ihrer Untersuchung zur Extraktion von UO_2^{2+} aus Salpetersäure in eine ionische Flüssigkeiten hinein schreiben. [41] Tabelle 2.1 gibt eine Übersicht ausgewählter Literaturquellen mit den darin gefundenen Stofftransportkoeffizienten. Neben den experimentellen Arbeiten ist der Stofftransport auch numerisch bereits untersucht worden. Kashid et al. berechneten für eine Strömung ohne Wandfilm den Verlauf der Extraktion von Essigsäure aus Kerosin in Natronlauge. Numerische Untersuchungen zum Stofftransport bei Strömung mit Wandfilm wurden von Heckmann et al. vorgenommen. [45] [44] Heckmann zeigte, dass die Peclet-Zahl (Gl. 2.8) als Verhältnis von konvektivem zu diffusivem Stofftransport einen entscheidenden Einfluss auf den Stofftransportkoeffizienten besitzt.

$$\mathrm{Pe} = \frac{u \cdot L}{D} \tag{2.8}$$

Gebildet mit der Wandfilmdicke h zwischen Pfropfen und Wand sowie der Relativgeschwindigkeit $u_{rel} = u_{Pfropfen} - \dot{V}_{ges}/A$ zwischen Pfropfen und Kernströmung in der Kapillare konnte Heckmann zeigen, dass ein Grenzwert von $\mathrm{Pe} = 10$ mindestens eingehalten werden sollte, um einen guten Stofftransport gewährleisten zu können. [44] Zudem konnte Heckmann zeigen, dass längere Pfropfen einen kleineren Stofftransportkoeffizienten herbeiführen und ein Anstieg der Kapillar-Zahl Ca einen positiven Einfluss auf den Stofftransport besitzt. [44] In Summe kann allen genannten Quellen entnommen werden, dass höhere Strömungsgeschwindigkeiten den Stofftransport begünstigen, der Extraktionsgrad bei konstanter Länge der Extraktionsstrecke durch eine Verkürzung der Verweilzeit aber dennoch absinkt. Ebenso begünstigen kleine Kapillardurchmesser und kurze Pfropfenlängen den Stofftransport.

2.2.4 Anwendungen der Mikroextraktion

Neben der Bestimmung von Stofftransportkoeffizienten finden sich in der Literatur auch zahlreiche Anwendungen der Pfropfenströmung als Extraktionswerkzeug. So nutzten Peroni et al. die Pfropfenströmung zur Aufreinigung von Proben für die Gaschromatografie. [28] Adamo et al. demonstrierten im Labor den Lösemittelwechsel durch kontinuierliche Überführung von Benzoesäure aus Ethylacetat über Wasser in Toluol. [35] Weiterführend finden sich Anwendungen in den Übersichtsarbeiten von Assmann et al. ([46]), Santana et al. ([47]) und zu Teilen bei Xie et al. ([48]). Strömungsmechanisch ist die Pfropfenströmung jedoch auf den Gleichstrombetrieb limitiert, während eine Parallelströmung nach Aota et al. auch im Gegenstrombetrieb möglich ist. [49] Daraus resultiert für die Pfropfenströmung die Notwendigkeit zur Verschaltung mehrerer Extraktionsstrecken in Gegenstromfahrweise, die jedoch jede für sich im Gleichstrom betrieben werden. Eine solche Verschaltung wurde jeweils von Adamo et al. ([35]), Holbach et al. ([17])) und Kaske ([23]) realisiert. Wie es auch bei makroskopischen Mixer-Settler-Anlagen der Fall ist, steigt der apparative Aufwand bei µ-Mixer-Settler-Anlagen mit wachsender Zahl an

Tabelle 2.1: Übersicht ausgewählter Literaturquellen zur Bestimmung von Stofftransportkoeffizienten in flüssig-flüssig Pfropfenströmungen.

Autor	Stoffsystem	$k_l \cdot a$ $[1/s]$	d_i $[mm]$	$\dot{V}_{ges}$ $[ml/min]$	Anmerkung
Burns und Ramshaw [42]	Kerosin[a]/Essigsäure(→)/OH^-_{aq}	$k_l = 0,16 - 0,56\frac{mm}{s}$	0,38 x 0,38	0,04 - 0,3	Rechteck, a ohne Wandfilm
Kashid et al. [26]	Kerosin/Iod(←)/Wasser	0,13 - 0,98	0,5 0,75 1,0	0,2 - 2,8	$k_l \cdot a_{max}$ bei kleinstem d_i
	Kerosin/Essigsäure(→)/Wasser	0,4 - 1,5	0,5 0,75 1,0	0,2 - 2,8	$k_l \cdot a_{max}$ bei kleinstem d_i
	Butanol/Bernsteinsäure(←)/Wasser	0,03 - 0,3	0,5 0,75 1,0	0,2 - 2,8	$k_l \cdot a_{max}$ bei kleinstem d_i
Fries et al. [32]	Toluol/Vanilin(←)/Wasser	max 5,3	0,38 x 0,18	0,2 - 2,8	Rechteck
Dessimoz [43]	OH^-_{aq}/TCA[b]/Hexan	0,2 - 0,5	0,4 x 0,4	2 - 12	Rechteck
Scheiff et al. [19]	Heptan/Essigsäure(→)/IL[c]	0,004 - 0,014	-	2	Ionische Flüssigkeit
Tsaoulidis und Angeli [41]	HNO_{3aq}/UO_2^{2+}(→)/TBP + IL[d]	0,049 - 0,29	0,5 1,0 2,0	0,12 - 5,67	Ionische Flüssigkeit
Susanti et al. [27]	Octanol/Acetanilide(←)/Wasser	0,01 - 0,08	0,8	0,08 - 0,42	
	TOA[e]/Milchsäure(←)/Wasser	0,01 - 0,07	0,8	0,08 - 0,42	Reaktivextraktion
Kurt [37]	Butylacetat/Aceton(←)/Wasser	0,356 - 0,557	1	1-6	geradlinige Kapillare
	Butylacetat/Aceton(←)/Wasser	0,505 - 1,024	1	1-6	Coiled Flow Inverter
Kaske et al. [22]	Wasser/Aceton(→)/Toluol	0,1 - 0,3	1	0,17 - 1,67	
	Kerosin/Essigsäure(→)/Wasser	0,1 - 0,3	1	0,17 - 1,67	
Kaske [23]	Wasser/Al^{3+}-DHAB[f](→)/Hexanol	0,08 - 0,23	-	1 - 5	Fluoreszenzmessung
Heckmann [44]	Butylacetat/Aceton(←)/Wasser	0,031 - 0,038	1	2-6	Numerik + Experiment
	Kerosin/Essigsäure(→)/NaI_{aq}	0,11 - 0,16	1	2-4	Numerik + Experiment

[a] gemischt mit Silikonöl
[b] TCA: Trichloressigsäure
[c] IL: Ionische Flüssigkeit, [EMIM][EtSO4] 1-Ethyl-3-methylimidazolium ethylsulfate mit $\eta = 93\,\mathrm{mPa \cdot s}$, η fällt während Extraktion auf $15\,\mathrm{mPa \cdot s}$
[d] TBP: Tributylphospat, IL: Ionische Flüssigkeit $[C_4mim][NTf_2]$ 1-methyl-3-butyl-imidazolium bistriflimide mit $\eta = 52\,\mathrm{mPa \cdot s}$
[e] TOA: tri-n-octylamine gelöst in Octanol
[f] Al^{3+}-DHAB: Chelat aus Al^{3+} und 2,2-Dihydroaxobenzoyl

Extraktionsstufen linear an, während Extraktionskolonnen für eine Steigerung der Trennstufenzahl lediglich in der Höhe vergrößert werden müssen (siehe Abschnitt 2.1). Bei µ-Mixer-Settler-Anlagen bestimmt jedoch nicht nur die geforderte Reinheit die Mindestzahl an Extraktionsapparaten, sondern ebenso der zu bewältigende Volumenstrom. Da Mikroapparate in ihrem Durchsatz begrenzt sind, können größere Produktionsmengen nur durch den parallelen Betrieb mehrerer Extraktionsapparate erreicht werden. Da jeder einzelne Apparat Pumpen und Überwachungssensoren benötigt, steigen ohne gute Parallelisierungsstrategien die Investitionskosten stark und schnell an.

2.2.5 Parallelisierungsstrategien für Mikroapparate

Für den Betrieb von n im Gegenstrom verschalteten Mixer-Settler-Stufen sind allgemein $n+1$ Pumpen notwendig (vgl. Abbildung 2.1). Kann in einer Mixer-Settler-Stufe der Volumenstrom nicht weiter erhöht werden, weil dann keine stabile Strömung oder Phasentrennung mehr möglich ist, kann in der Mikroverfahrenstechnik kein Scale-Up vorgenommen werden. Hierdurch gingen die Vorteile der Mikroverfahrenstechnik verloren. Eine Steigerung der Produktionsmenge muss stattdessen durch ein Numbering-Up erfolgen, indem m Extraktionsstränge parallel betrieben werden. Jeder Strang benötigt ohne angemessene Strategie der Parallelisierung jeweils $n+1$ Pumpen, welche den größten Anteil der apparativen Kosten ausmachen. In der Literatur lassen sich daher verschiedene Arbeiten finden, welche eine Reduzierung der notwendigen Pumpenzahl anstreben. Allgemein unterscheiden Kashid et al. zwischen dem externem Numbering-Up, bei dem nach oben beschriebenem Verfahren jeder Extraktionsstrang mit allen notwendigen Pumpen versehen wird, und dem internem Numbering-Up, bei dem die Pfropfenströmung zunächst erzeugt und danach auf die Kanäle verteilt wird. [50] Erfolgreiche Realisierungen des internen Numbering-Up für die Pfropfenströmung sind in der Literatur nach Kenntnis des Autors nicht bekannt. Kashid et al. nutzten das interne Numbering-Up jedoch für die einphasigen Zuleitungen zu den jeweiligen Pfropfenerzeugern und schaffte hierdurch eine Parallelisierung von bis zu 6 Mikroreaktoren mit lediglich einer Pumpe für Wasser und einer Pumpe für Kerosin. Eine einheitliche Verteilung auf die Kanäle wird lediglich durch die Symmetrie der Verteilerstruktur erreicht und durch die Druckverluste der angeschlossenen Mischelemente erzeugt. [50] Allgemein wird die Gleichmäßigkeit der Verteilung auf die Mikrokanäle durch die großen Fertigungstoleranzen gestört. Zur Verbesserung der Gleichmäßigkeit existieren verschiedene Ansätze:

- Die Mikrokanäle selbst können sehr genau gefertigt werden. Schenk et al. zeigten dies experimentell. [51] [52] Die notwendigen Toleranzen ließen sich industriell jedoch nicht mit akzeptablen Aufwand umsetzen.

- Den Mikrokanälen kann ein Druckverlust vorgeschaltet werden, welcher den Druckverlust der nachfolgenden Bauteile um ein Vielfaches übersteigt und wiederum besonders exakt gefertigt werden muss. Al-Rawashdeh et al. entwickelten einen Konstruktionsleitfaden und untersuchten dieses Konzept experimentell, wie auch numerisch. Es wird angegeben, dass der vorgeschaltete Druckverlust etwa 4-25 Mal so groß gewählt werden muss, wie der Druckverlust des nachfolgenden Mikrokanals. [53] [54] [55] Der Kosten für den Betrieb

der Anlage steigen hierdurch jedoch ebenfalls um mindestens diesen Faktor, da unnötig Energie dissipiert wird.

- Die Schwankungen der Druckverluste können manuell durch iteratives Anpassen der Kapillarlängen ausgeglichen werden. Mendorf et al. parallelisierten hierdurch sehr erfolgreich einphasige Verteilerstrukturen, während die Ergebnisse für mehrphasige Strömungen im Vergleich dazu schlechter ausfallen. [56] Bei einer großen Anzahl an parallelisierten Strängen ist ein manuelles Nachjustieren aller Druckverluste hingegen sehr aufwändig.

- Neben dem iterativen Anpassen vor der Inbetriebnahme können unterschiedliche Druckverluste auch durch Aktuatoren nachgeregelt werden. Hierfür sind nicht nur Aktuatoren sondern auch Sensoren in den Kanälen notwendig, welche die Strömungsverhältnisse erfassen. Antweiler et al. setzten sowohl für Beeinflussung der Strömung im Bereich der Pfropfenerzeugung als auch für die Detektion der Strömung im Kanal elektrische Felder ein. Mit Hilfe der allgemein temperaturabhängigen Viskosität beeinflussten sowohl Antweilter et al. als auch Arsenjuk et al. durch gezielte Aufheizung oder Abkühlung die Druckverluste in den einzelnen Mikrokanälen. [25] [57] Die Notwendigkeit einer Vielzahl an Pumpe wird hierbei durch die Notwendigkeit vieler Sensoren und Aktuatoren ersetzt, welche ebenfalls Kosten verursachen.

Alle aufgeführten Methoden konnten die apparativen Kosten bisher noch nicht so stark absenken, dass die Mirkoverfahrenstechnik eine echte Konkurrenz zu den makroskopischen Apparaten in herkömmlichen Anwendungsfällen darstellt, wenn große Mengen produziert werden sollen. Zudem liefern nicht alle Methoden die geforderte Gleichheit zwischen den parallelisierten Kanälen, was weiteres Entwicklungspotential darstellt.

2.3 Numerische Simulation von Mehrphasenströmungen

Numerische Simulationen nehmen in der Prozessauslegung einen immer größeren Stellenwert ein, da sie das Systemverhalten ohne aufwändige Versuche voraussagen können und Einblicke ermöglichen, die experimentell teilweise gar nicht zugänglich sind. Im Bezug auf diese Arbeit soll das dynamische Verhalten eines Phasentrenners abgebildet und damit a priori berechnet werden, was mit bisherigen numerischen Methoden nicht zufriedenstellend möglich ist. Als Grundlage für die Erarbeitung der numerischen Methode dieser Arbeit soll der aktuelle Stand des Wissens im Bezug auf die Simulation von Mehrphasenströmungen rekapituliert werden. Dabei soll auf die Volume-Tracking-Methoden von Olsson et al. ([58]), Jacqmin ([59], [60]), sowie Xiao et al. ([61]) und Xie et al. ([62], [63]) im Besonderen eingegangen werden, da sie den Ansatzpunkt zur Entwicklung der eigenen numerischen Methode in Kapitel 3 darstellen.

2.3.1 Die Finite-Volumen-Methode

Zur numerischen Lösung von Differentialgleichungen existieren verschiedene Methoden. Die größte Bedeutung besitzen hierbei das Finite-Elemente-Verfahren, das

Finite-Differenzen-Verfahren und die Finite-Volumen-Methode. [64] Für die numerische Berechnung von Mehrphasenströmungen kommt dabei der Finite-Volumen-Methode die größte Bedeutung zu, da sie im Gegensatz zu z.B. der Finite-Differenzen-Methode in Kombination mit einem Volume-of-Fluid-Ansatz direkt die Massenerhaltung der Phasen garantieren kann. Auf diesem Ansatz basiert auch die in dieser Arbeit selbst entwickelte numerische Methode, weshalb in diesem Abschnitt exemplarisch die Finite-Volumen-Methode am Beispiel der Konvektions-Diffusions-Gleichung einer Komponente c im Geschwindigkeitsfeld $\vec{u}$ dargestellt wird.

$$\frac{\partial c}{\partial t} = -\vec{\nabla} * (\vec{u} \cdot c) + \vec{\nabla} * \left(D \cdot \vec{\nabla} c\right) \tag{2.9}$$

Zunächst wird das Rechengebiet in nicht überlappende Kontrollvolumina unterteilt, die beliebig geformt sein können und das Rechengebiet vollständig ausfüllen. Jede Zelle (cell) besteht aus beliebig vielen Zellwänden (faces). Die Zellwände wiederum bestehen aus einzelnen Kannten (edges), die jeweils zwei Eckpunkte der Zelle (vertices) miteinander verbinden. Diese Eckpunkte (vertices) sind nicht mit den Zellzentren (cellcentre) oder Zellwandzentren (facecentre) zu verwechseln. Über das Volumen einer einzelnen Zelle kann nun Gl. 2.9 integriert und mit Hilfe des Gaußschen Integralsatzes ([65]) umgeformt werden.

$$\begin{aligned} \iiint \frac{\partial c}{\partial t} \, \mathrm{d}V &= -\iiint \vec{\nabla} * (\vec{u} \cdot c) \, \mathrm{d}V + \iiint \vec{\nabla} * \left(D \cdot \vec{\nabla} c\right) \, \mathrm{d}V \\ \frac{\partial}{\partial t} \iiint c \, \mathrm{d}V &= -\oiint_{\partial V} (\vec{u} \cdot c) \, \mathrm{d}\vec{A} + \oiint_{\partial V} \left(D \cdot \vec{\nabla} c\right) \, \mathrm{d}\vec{A} \end{aligned} \tag{2.10}$$

Auf dieser Umformung basiert der Grundansatz der Finite-Volumen-Methode. Eine Rechenzelle wird dabei als Bilanzraum angesehen, in dem eine Erhaltungsgröße wie die Komponente c über die Zeit nur durch Zu- und Abflüsse über die Zellwände variieren kann. Eine Zellwand gehört dabei stets zu zwei Zellen, sofern sie nicht einer der Randbedingungen zugeordnet ist. Das Geschwindigkeitsfeld $\vec{u}$ erzeugt einen Fluss durch die Zellwand, welcher der einen anliegenden Zelle abgezogen und zugleich der anderen anliegenden Zelle in gleicher Größe zugeschrieben wird. Der Betrag und die Richtung des Flusses können dabei durch Modell- oder Diskretisierungsfehler fehlerbehaftet sein, doch bleibt die Menge an Komponente c bei korrekter Implementierung dabei unverändert. Diesen Vorteil der inhärenten Massenerhaltung nutzen eine Vielzahl an Volume-of-Fluid-Methoden (siehe Paragraph 2.3.2.1). Um den erhaltenen integralen Ausdruck in ein algebraisches Gleichungssystem zu überführen, müssen Näherungen an die Volumen- und Oberflächenintegrale eingeführt werden, welche einen entsprechenden Diskretisierungsfehler mit sich bringen. Gängige Praxis sind dabei folgende Näherungen, welche jeweils der Anwendung einer Mittelpunktsregel im Ort entsprechen ([64]):

$$\iiint c \, \mathrm{d}V \simeq c_{cellcentre} \cdot V_{cell} \tag{2.11}$$

$$\iint (\vec{u} \cdot c) \, \mathrm{d}\vec{A} \simeq c_{facecentre} \cdot A_{face} \cdot \vec{u}_{facecentre} * \vec{n}_{face} \tag{2.12}$$

$$\iint \left(D \cdot \vec{\nabla} c\right) \, \mathrm{d}\vec{A} \simeq D \cdot \frac{\partial c}{\partial \vec{n}_{face}} \cdot A_{facecentre} \tag{2.13}$$

Die Werte für $\vec{u}_{facecentre}$, $c_{facecentre}$ und $\frac{\partial c}{\partial \vec{n}_{face}}$ sind dabei unbekannt und müssen durch Interpolation oder Differenzenverfahren aus den Werten der anliegenden Zellzentren berechnet werden. Die Wahl der Interpolations- und Integrationsverfahren

zur Berechnung der Volumen- und Flächenintegrale in Gl. 2.10 beeinflusst dabei maßgeblich die Größe des eingebrachten numerischen Fehlers und ist Grund für die Vielzahl an Verfahren, die hierfür bereits entwickelt worden sind. Zusätzlichen zu der räumlichen Diskretisierung ist die Transportgleichung auch zeitlich noch zu diskretisieren. In der Praxis wird dabei meist Gl. 2.10 als Anfangswertproblem betrachtet und mit Hilfe von Runge-Kutta-Methoden hoher Ordnung näherungsweise gelöst ([64], [66]), was jedoch einen weiteren Diskretisierungsfehler einbringt. [3]

2.3.2 Darstellungsmöglichkeiten einer zweiten Phase

Zur Berechnung von mehrphasigen Strömungen mit CFD-Programmen existieren diverse Methode, die jeweils verschiedene Vor- und Nachteile aufweisen, und in Volume-Tracking- und Surface-Tracking-Verfahren eingeteilt werden können. In diesen Unterabschnitt soll eine kurze Übersicht über die verschiedenen Methoden gegeben werden. Anschließend wird auf die Volume-of-Fluid-Methode (VOF) nach dem Volume-Tracking- und die Levelset-Methode als Surface-Tracking-Verfahren noch detaillierter eingegangen, da diese Methoden derzeit am häufigsten angewendet werden und essentielle Grundbausteine für die eigene Methode liefern. Darüber hinaus soll dargestellt werden, auf welche Weise die Oberflächenspannung als zusätzlicher Term in die zu lösende Impulsgleichung eingebunden wird. Dabei ist stets Gl. 2.14 als Transportgleichung für den Impuls zu lösen ([67], [68]), doch existieren je nach Methode verschiedene Ansätze zur Behandlung des letzten Terms, da die sprunghaft auftretende Grenzflächenkraft zu Problemen in der Implementierung führt. Auch die Dichte ρ und die Viskosität η unterscheiden sich im Allgemeinen in beiden Phasen.

$$\frac{\partial\left(\rho\cdot\vec{u}\right)}{\partial t}+\left(\rho\cdot\vec{u}\cdot\vec{\nabla}\right)\vec{u}=-\vec{\nabla}p+\vec{\nabla}*\left(\eta\cdot\left(\left(\vec{\nabla}\vec{u}\right)+\left(\vec{\nabla}\vec{u}\right)^{T}\right)\right)+\delta\cdot\sigma\cdot\kappa\cdot\vec{n} \quad (2.14)$$

Einfluss auf den Transport des Impulses $\rho\cdot\vec{u}$ nehmen die Konvektion, die viskosen Reibungskräfte, das Druckfeld und der durch die Grenzflächenspannung eingebrachte Quellterm. Die Beschreibung einer zweiten Phase kann grundsätzlich auf zwei verschiedene Arten erfolgen. Der erste Ansatz beruht auf dem Verfolgen der Phasengrenzfläche und dessen exakter Position (front-tracking). Nach dem Ausschlussverfahren liegt das Fluid 1 auf der einen Seite und das Fluid 2 auf der anderen Seite der Phasengrenzfläche. Alternativ kann auch das Fluid selbst verfolgt werden (volume-tracking). Überall, wo sich Fluid 1 nicht befindet, wird automatisch Fluid 2 angenommen. Dazwischen liegt gezwungenermaßen die Phasengrenzfläche.

Zur Verfolgung der Phasengrenzfläche werden im Front-Tracking-Verfahren häufig Markerpartikel eingesetzt. Diese spannen eine Art Netz auf, welches die Phasengrenzfläche repräsentiert und mit der Strömung mittransportiert wird. Dies erfüllt automatisch die kinematische Randbedingung für freien Grenzflächen. Dehnung und Stauchung des Netzes erfordert stellenweise das Einfügen neuer Partikel und auch die Zuordnung der Konnektivität zwischen den Partikeln stellt eine große Herausforderung dar. Koaleszenz oder der Abriss einzelner Tropfen können daher nur schwer dargestellt werden. Glimm et al. diskutieren Ansätze zur Lösung einiger dieser Probleme. [69] Auf diesem Verfahren basierend führten Unverdi and Tryggvason die Berechnung von aufsteigenden Blasen in Flüssigkeit durch. [70] Sie implementierten

[3] Im mathematischen Sinn gehören auch die explizite und implizite Euler-Methode, wie auch das Crank-Nicholson-Verfahren, zu den Runge-Kutta-Verfahren.

die Grenzflächenspannung, indem sie Polynome an das Netz aus Markerpartikeln angepassten und aus diesen die lokale Krümmung der Grenzfläche ermittelten. Die Grenzflächenkraft wurde daraufhin auf zugehörigen Nachbarpunkte verteilt und in die Kräftebilanz eingebunden. Auch Popinet und Zaleski nutzten Markerpartikel zur direkten Verfolgung der Grenzfläche und berechneten durch angepasste Splines die Krümmung mit zugehörigem Drucksprung. [71]

Ist die Grenzfläche selbst ein Teil des Rechengitters, können derartige Probleme umgangen werden. Muzaferija und Peric, wie auch Tukovic und Jasak, demonstrierten in ihren Arbeiten die Darstellung der zweiten Phase jeweils durch ein direkt an die Grenzfläche angepasstes Rechengitter, welches als Teil der Lösung eine Gitterdeformationen durchführt, um diesen Zustand aufrecht zu erhalten. [72] [73]

2.3.2.1 Die Volume-of-Fluid-Methode

Als einen Vorgänger der Volume-of-Fluid Methode kann der Ansatz von Harlow und Welch gesehen werden (Marker-and-Cell-Methode). Sie markierten in ihren Rechnungen nach dem Volume-Tracking-Verfahren eine der beiden Phasen (Fluid 1) durch Partikel, welche der Strömung folgen. [74] Alle Zellen, die im nächsten Zeitschritt mindestens einen Partikel enthielten, wurden als Zellen mit Fluid 1 angesehen und entsprechend mit zugehöriger Dichte und Viskosität versehen. Eine Weiterentwicklung dieser Idee durch Hirt und Nichols ist die Angabe einer lokalen Funktion, die den Füllgrad F einer Zelle beschreibt. [75] Eine Zelle ist dann nicht wie in der Marker-and-Cell-Methode komplett gefüllt oder vollständig leer, sondern zu einem Prozentsatz gefüllt. Die zeitliche Entwicklung des Füllgrades F erfolgt nach der allgemeinen Transportgleichung (Gl. 2.15).

$$\frac{\partial F}{\partial t} + \vec{\nabla} * (\vec{u} \cdot F) = 0 \tag{2.15}$$

Dabei nutzen Hirt und Nichols zur Berechnung der Flüsse durch die Zellwände (Gl. 2.12) geometrische Überlegungen statt numerischen Differenzenverfahren, indem sie eine Rekonstruktion der Grenzfläche in jeder Zelle um eine Wegstrecke $\vec{u} \cdot \Delta t$ verschieben. [75] Da F nur von einer Zelle in die andere verschoben wird und selbst mit der Masse an Fluid 1 identifiziert werden kann, garantiert dies die Massenerhaltung.

2.3.2.2 Die Levelset-Methode

Die Entstehung der Levelset-Methode Methode führen Sussmann et al. ([76]) und Ferziger ([64]) auf Osher und Sethian ([77]) zurück. Für die Beschreibung von Mehrphasenströmungen mit Kapillarkräften erarbeiteten Sussmann et al. eine Formulierung mit Hilfe der heute noch weitgehend genutzte vorzeichenbehaftete skalare Abstandsfunktion Φ. Sie ist im gesamten Rechengebiet definiert und besitzt direkt an er Grenzfläche den Wert 0, weshalb sie den Surface-Tracking-Verfahren zugeschrieben wird. Der Gradient $\vec{\nabla}\Phi$ hat nach Definition die Länge 1. Der Wert von Φ gibt jeweils den minimalen Abstand zur Grenzfläche an und ist in Fluid 1 positiv, in Fluid 2 hingegen negativ. Die Bewegung der Grenzfläche durch das Gitter erfolgt, wie auch für den Füllgrad F in der Volume-of-Fluid-Methode, durch eine generische Transportgleichung (Gl 2.16).

$$\frac{\partial \Phi}{\partial t} + \vec{\nabla} * (\vec{u} \cdot \Phi) = 0 \tag{2.16}$$

Um numerische Stabilität zu garantieren werden Dichte und Viskosität nicht sprunghaft an der Grenzfläche geändert, sondern kontinuierlich variiert, wie an der Dichte ρ beispielhaft in Gl. 2.17 gezeigt ist [58]:

$$\rho(\Phi) = \begin{cases} \rho_1 & \Phi < -\varepsilon \\ \rho_1 + (\rho_2 - \rho_1) \cdot \left(\frac{1}{2} + \frac{\Phi}{2 \cdot \varepsilon} + \frac{1}{2 \cdot \pi} \cdot \sin\left(\frac{\pi \cdot \Phi}{\varepsilon} \right) \right) & |\Phi| \leq \varepsilon \\ \rho_2 & \varepsilon < \Phi \end{cases} \tag{2.17}$$

Der Parameter ε besitzt dabei die Größenordnung der Gitterweite Δx und beschreibt die Größe des Übergangsbereichs von einer Phase in die andere. Der Normalenvektor $\vec{n}$ auf die Grenzfläche und die daraus resultierende Krümmung κ werden nach Gl. 2.18 und Gl. 2.19 berechnet.

$$\vec{n} = \frac{\vec{\nabla}\Phi}{|\vec{\nabla}\Phi|} \tag{2.18}$$

$$\kappa = -\vec{\nabla} * \vec{n} \tag{2.19}$$

Die Implementierung des Kapillardrucks in Gl. 2.14 erfolgt nach einem Ansatz von Brackbill et al., bei dem die Flächenkraft der Grenzflächenspannung als Grenzwert einer Volumenkraft angesehen wird, die auf einem Volumen endlicher Dicke wirkt. [68] Im Grenzwert hat das Volumen eine Dicke von 0 und das Modell bildet den unstetigen Drucksprung ab, der durch die Delta-Distribution, auch Dirac-Funktion genannt, vorgesehen ist. Unter Verwendung einer endliche Dicke lässt sich Gl. 2.20 formulieren, was zu einer wesentlich einfacheren numerischen Implementierung führt.

$$\sigma \cdot \kappa \cdot \delta \cdot \vec{n} = \sigma \cdot \kappa \cdot \frac{\vec{\nabla}\rho\left(\Phi\right)}{\rho_2 - \rho_1} \tag{2.20}$$

Für eine akkurate Berechnung des Normalenvektors $\vec{n}$ und der Krümmung κ, sowie die korrekte Zuweisung der Stoffdaten, sollte Φ die Eigenschaft einer Abstandsfunktion behalten, was durch Anwendung von Gl. 2.16 nicht garantiert ist. Zur Lösung dieses Problems implementieren Sussmann et al. einen Reinitialisierungsschritt nach Gl. 2.21, der nach jedem Zeitschritt gelöst werden muss. [76]

$$\frac{\partial \Phi}{\partial \tau} = \text{sign}\left(\Phi\right) \cdot \left(1 - |\vec{\nabla}\Phi|\right) \tag{2.21}$$

Dabei bezeichnet τ explizit nicht die Zeitskala t der Simulation, sondern eine Zwischenskala, die lediglich innerhalb eines Zeitschrittes Δt existiert. Eine Massenerhaltung der Methode ist dann gegeben, wenn die durch das Niveau $\Phi = 0$ eingeschlossene Fläche erhalten bleibt. Die Reinitialisierung nach Gl. 2.21 lässt das Nullniveau unberührt. Ein Transport durch Gl. 2.16 garantiert allerdings lediglich die Erhaltung von Φ selbst, nicht jedoch der im Nullniveau eingeschlossenen Fläche. Die Levelset-Methode ist daher nicht automatisch massenerhaltend, wie auch Tornberg und Engquist zeigten. [78] Durch Hinzufügen von Reinitialisierungsschritten ähnlich zu Gl. 2.21 nach anderen Kompressionsgleichungen konnten Zhang et al. die Massenerhaltung verbessern. [79] Ihre Methode ist jedoch nur dann vollständig massenerhaltend, wenn die zusätzlichen Reinitialisierungsschritte in jedem einzelnen Zeitschritt vollständig bis in den stationären Zustand gelöst werden. Jedes Restresiduum führt zu einer Änderung der von Φ eingeschlossenen Fläche.

2.3.2.3 Die modifizierte Levelset-Methode nach Olsson und Kreiss

Während die Volume-of-Fluid-Methode massenerhaltend ist, aber Probleme bei der Rekostruktion der Grenzflächenposition hat, überzeugt die Levelset-Methode gerade dadurch, ist aber nicht konservativ. Daher liegt das Bestreben nahe, beide Vorteile zu verbinden. Olsson und Kreiss entwickelten den Ansatz, direkt eine Indikatorfunktion c ähnlich zu Gl. 2.17 zu nutzen. Die Grenzfläche definiert sich dann durch das Niveau $c = 1/2$, während Fluid 1 mit $c = 1$ und Fluid 2 mit $c = 0$ identifiziert werden.[4] Da die allgemeine Transportgleichung massenerhaltend ist, wird dadurch das Gesamtverfahren konservativ. Im Vergleich zur reinen Levelset-Methode treten durch den nicht linearen Verlauf von c jedoch verstärkt Interpolations- und Diskretisierungsfehler bei der Advektion auf, weshalb Olsson und Kreiss ebenfalls ein intermdediären Kompressionsschritt nach Gl. 2.22 einführten, der ebenso bis in den stationären Zustand zu lösen ist. [58]

$$\frac{\partial c}{\partial \tau} + \vec{\nabla} * \vec{k}(c) = \varepsilon \cdot \Delta c \tag{2.22}$$

$$\vec{k}(c) = c \cdot (1 - c) \cdot \vec{n} \tag{2.23}$$

$$\vec{n} = \frac{\vec{\nabla} c}{|\vec{\nabla} c|} \tag{2.24}$$

Dabei bezeichnet τ abermals nicht die normale Zeitskala t, sondern existiert wiederum nur innerhalb eines Zeitschrittes Δt. Das Gleichgewicht ist erreicht, sobald der komprimierende Fluss von $\vec{k}(c)$ und der entgegenwirkende Diffusionsterm $\varepsilon \cdot \Delta c$ sich bis auf eine vorher definierte Toleranz gegenseitig kompensieren. Der Parameter ε steuert dabei, welche Ausdehnung der Gleichgewichtszustand besitzt. Für den räumlich eindimensionalen Fall besitzt Gl. 2.22 eine analytische Lösung, in welcher der Einfluss von ε gut erkennbar ist.

$$c(x) = \frac{1}{2} \cdot \left(1 + \tanh\left(\frac{x}{2 \cdot \varepsilon}\right)\right) \tag{2.25}$$

Damit der Übergang von einer Phase in die andere Phase möglichst schnell abläuft und dadurch die Modellfehler einer verschmierten Phasengrenzfläche klein bleiben, ist ε möglichst klein zu wählen. Demgegenüber steht die Zunahme von numerischer Diffusion bei sehr kleinem ε und der resultierenden Notwendigkeit von vielen Kompressionsschritten. Ollson und Kreiss geben an, dass ε in der Größenordnung der Gitterweite Δx gewählt werden sollte und bei Gitterverfeinerung über einen Exponenten $d \leq 1$ an diese zu koppeln ist. [58] An diesem Punkt verschwimmen die Ansätze von Volume-of-Fluid und Levelset-Methode. Die Funktion c ist keine strikte Levelset-Funktion, da sie nicht die Eigenschaft einer Abstandsfunktion besitzt, beschreibt allerdings auch nicht direkt den Füllgrad F einer Zelle und ist damit im ursprünglichen Sinne keine Volume-of-Fluid-Funktion. Dennoch wird c häufig als Volume-of-Fluid-Funktion bezeichnet und auch so verwendet. Anwendung findet diese Art von Kompressionsterm an verschiedensten Stellen, unter anderem bei Dittmar ([21]) und Shams et al. ([80]). Eine Variation von Gl. 2.22 wird außerdem in OpenFOAM im Solver für Mehphasenströmungen interFoam verwendet, wo jedoch auf den Diffusionsterm verzichtet wird und stattdessen die komprimierende Rückstellkraft von $\vec{k}(c)$ an die lokale Geschwindigkeit $\vec{u}$ gekoppelt wird. [81] Hierdurch

[4] Zu beachten ist die angepasste Variablendeklaration, um innerhalb der hier vorliegenden Arbeit konsistente Bezeichnungen verwenden zu können. z.B. $c \rightarrowtail \Phi$

soll numerischer Diffusion im gleichen Maße entgegengewirkt werden, wie sie entsteht, was diesen Zusammenhang aber als bekannt voraussetzt und zudem keinerlei Kompression an nicht transportierten Grenzflächen vornimmt. Die Kompression der Grenzfläche wird in OpenFOAM außerdem nicht auf einer gesonderten Zeitskala bis in den stationären Zustand gelöst, sondern direkt in die Transportgleichung der Indikatorfunktion eingebracht.

2.3.3 Kapillarer Drucksprung und parasitäre Strömungen

Die bisher genannten Methoden implementieren den Drucksprung an der Phasengrenzfläche durch einen Quellterm in der Kräftebilanz (Gl. 2.14). Da die Kapillarkraft prinzipiell eine Oberflächenkraft ist, die Kräfte im Fluidraum aber häufig volumenspezifisch bilanziert werden, ist eine Modellierung als Volumenkraft notwendig. Einen Ansatz hierzu, welcher bereits in Unterabschnitt 2.3.2.2 beschrieben worden ist, lieferten Brackbill et al. (Gl. 2.20, [68]). Diese Umformulierung selbst kann bereits zu unphysikalische Kräften an der Grenzfläche zweier Fluide führen, welche sich in unphysikalischen Geschwindigkeiten äußern, den sogenannten parasitären Strömungen. Zudem bringen Diskretisierungs- und Modellfehler weitere Ungenauigkeiten in die Kräftebilanz ein, welche parasitäre Strömungen erzeugen oder weiter verstärken können. Zu nennen sind hier unter Anderem die Berechnung des Normalenvektors $\vec{n}$ und der Krümmung κ. Früh wurde erkannt, dass parasitäre Strömungen am liegenden Tropfen aus einem Ungleichgewicht zwischen Druckgradient und Grenzflächenspannung resultieren. [82] Obwohl eine analytische Lösung ohne parasitäre Strömungen für einen ruhenden Tropfen existiert, ist ein exaktes Ausbalancieren der beiden Kräfte in den diskretisierten Gleichungen oft nicht möglich. Dieses Ungleichgewicht führt zu einer resultierenden Kraft, die das Fluid antreibt. So zeigte die Arbeit von Popinet und Zaleski ebenfalls die Notwendigkeit, dass Druckgradient und Oberflächenspannung auch in den diskretisierten Gleichungen die Möglichkeit benötigen, sich exakt aufzuheben, da sonst parasitäre Strömungen entstehen können. [71] Jamet et al. erweiterten das Verständnis und zeigten, dass effektiv eine strikte Energieerhaltung in den diskretisierten Gleichungen ausreichend ist. [83] Francois et al. führen hierzu einen Ansatz an, bei dem Druckgradient und Grenzflächenkraft bereits auf den Zellwänden gegeneinander bilanziert werden. Nur die resultierende Kraft wird in einen Vektor umgewandelt und in das Lösen der Kräftebilanz eingebracht. [84] Die Möglichkeit zur exakten Balance beider Kräfte ist auf den Zellwänden wesentlich einfacher zu realisieren, was die parasitären Strömungen am liegenden Tropfen nahezu vollständig verschwinden lässt.

Eine Anwendung dieser Erkenntnisse auf bewegte Grenzflächen bringt parasitäre Strömungen jedoch nicht zwangsläufig zum Erliegen. Kein Advektionsschema ist frei von Diskretisierungfehlern, sodass in jedem Zeitschritt eine unphysikalisch Auslenkung der Grenzfläche auftritt. Diese Diskretisierungfehler führen zu einem Verschmieren der Indikatorfunktion und bringen Fehler in die Krümmungsberechnung ein, welche sich wiederum in parasitären Strömungen äußern. Um diesen Fehlern entgegenzuwirken werden Kompressionsterme, wie sie in Paragraph 2.3.2.2 und 2.3.2.3 beschrieben sind, in die Transportgleichung der Indikatorfunktion eingefügt. Diese stabilisieren den Transport, aber verursachen wiederum Probleme am liegenden Tropfen. Neuste Arbeiten von Aboukhedr et al. zeigen unter Nutzung des Solvers interFoam aus OpenFOAM, dass auch ein liegender Tropfen ohne Einfluss von

äußeren Kräften unter parasitären Strömungen leidet, die von der Stärke des Kompressionsterms abhängig sind. [85] Eine Interpretation dieses Ergebnisses ist, dass von Kompressionsterm und Druckgleichung unterschiedliche Gleichgewichtszustände angestrebt werden. Abbildung 2.4 veranschaulicht diese Problematik schematisch. In analytischer Form mögen Druckfeld und Kompressionsterm den gleichen Zustand

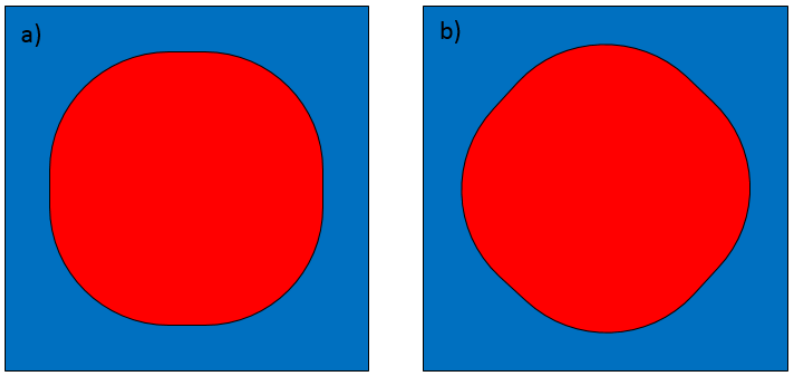


Abbildung 2.4: Schematische Darstellung der angestrebten Gleichgewichtszustände von Kompressionsterm (a) und berechnetem Druckfeld (b). Zustände sind frei erfunden und dienen lediglich der Veranschaulichung.

anstreben, doch können sich die angestrebten Gleichgewichtszustände in den diskretisierten Gleichungen unterscheiden. Während der Kompressionsterm versucht, die Phasengrenzfläche in Zustand a) zu versetzen, wird das berechnete Druckfeld durch die Krümmungsberechnung eine Ungleichmäßigkeit feststellen und eine Geschwindigkeit erzeugen, die Zustand b) anstrebt. Diese resultierende Geschwindigkeit wird als parasitäre Strömung wahrgenommen und mit U_{fehl} bezeichnet. Sie tritt auch dann auf, wenn Grenzflächenspannung und Druckgradient nach Jamet et al. und Francois et al. korrekt diskretisiert werden. Solange der Kompressionsterm stets gegen diese Geschwindigkeit arbeitet, kann die Geschwindigkeit U_{fehl} nicht zum Erliegen kommen. Wird die Korrekturkraft des Kompressionsterms erhöht, so verstärkt sich dieser Effekt, weil der Kompressionsterm die Grenzfläche nun noch weiter vom Gleichgewichtszustand der Druckgleichung entfernt und Zustand a) angenommen wird. Beträgt die Kompressionsstärke 0, dominiert das berechnete Druckfeld und die parasitäre Strömung kann zum Erliegen kommen, da Zustand b) angenommen wird. Dieses Ungleichgewicht kann am liegenden Tropfen z.B. dadurch kaschiert werden, dass die Stärke des Kompressionsterms mit der aktuellen Geschwindigkeit der Phasengrenzfläche skaliert wird. Die lokale Kompressionsstärke wird immer schwächer, je kleiner die lokale Geschwindigkeit U_{fehl} wird. Die ursprüngliche Problematik wird hierdurch jedoch nicht gelöst, sondern im Falle einer nicht transportierten Grenzfläche lediglich kaschiert. Bei transportierten Grenzflächen kommt das Ungleichgewicht weiterhin zum Tragen und erzeugt parasitäre Strömungen. So zeigen Shams et al., dass am liegenden Tropfen zwar parasitäre Strömungen zum Erliegen kommen können, sie beim konvektiven Transport aber weiterhin bestehen. [80] Aus dieser Grundlage erschließt sich, dass ein Kompressionsterm konstruiert werden muss, der den Ergebnissen von Francois et al. ([84]) folgend auch in diskretisierter Form möglichst exakt im Einklang mit der Krümmungsberechnung steht.

2.3.3.1 Die kapillare Zeitschrittweitenlimitierung

Durch eine Implementierung der Grenzflächenspannung über eine Formulierung nach Brackbill et al. entsteht eine zusätzliche Limitierung der Zeitschrittweite. Neben der bekannten Limitierung über die Courant-Zahl Co, nach der sich Informationen pro Zeitschritt höchstens eine Gitterzelle weit bewegen dürfen ($\mathrm{Co} = u \cdot \Delta t / \Delta x \leq 1$), ist ein weiteres Stabilitätskriterium nach Gl. 2.26 einzuhalten. [68]

$$\Delta t_{cap} < \sqrt{\frac{(\rho_1 + \rho_2) \cdot \Delta x^3}{4 \cdot \pi \cdot \sigma}} \tag{2.26}$$

Physikalisch fordert diese Kriterium, dass Beschleunigungen, die aus numerischen Fehlern resultieren, die Grenzfläche nur so weit auslenken, dass sich der Fehler nicht weiter verstärkt. Je träger ein Fluid ist, desto unempfindlicher reagiert es auf solche Beschleunigungen. Galusinski et al. konnten dieses Kriterium erweitern und zu Gl. 2.27 weiterentwickeln, indem sie nicht nur die Trägheit, sondern auch dämpfende Effekte der Viskosität η in Betracht zogen. [86]

$$\Delta t_{cap} < \frac{1}{2} \cdot \left(C_2 \cdot \frac{\eta \cdot \Delta x}{\sigma} + \sqrt{\left(C_2 \cdot \frac{\eta \cdot \Delta x}{\sigma} \right)^2 + 4 \cdot C_1 \cdot \frac{\rho}{\sigma} \cdot \Delta x^3} \right) \tag{2.27}$$

Die Konstanten C_1 und C_2 hängen dabei nicht von den physikalischen Eigenschaften der Fluide, sondern lediglich von den numerischen Eigenschaften des Lösungsalgorithmus ab. Bei immer feiner werdenden Gittern, gewinnt die kapillare Zeitschrittweite dabei immer mehr an Bedeutung, da sie im Vergleich zur Limitierung durch die Courant-Zahl mit $\Delta x^{\frac{3}{2}}$ skaliert. Eine Möglichkeit, diese Limitierung zu umgehen stellten Sussman et al. vor. Hierbei wird die Indikatorfunktion der Grenzfläche nicht konvektiv transportiert, sondern durch eine alternative Beschreibung über die mittlere Krümmung der Grenzfläche in der Zeit transportiert. Die Notwendigkeit des erneuten Lösens der Kräftebilanz nach Δt_{cap} wird hierbei umgangen, jedoch auf Kosten von oft mehreren wesentlich kleineren notwendigen Zeitschritten für den Transport der Indikatorfunktion. [87] Hierdurch findet diese Methode in der Praxis kaum Anwendung.

2.3.4 Das Phase-Field-Modell nach Jacqmin

Das Phase-Field-Modell stellt eine weitere Möglichkeit zur Modellierung einer Mehrphasenströmung dar. Es weist viele Aspekte bereits genannter Methoden auf, aber auch etliche Unterschiede. Da es Grundbausteine für die entwickelte Methode dieser Arbeit liefert, wird es separat dargestellt und diskutiert.

Die Phase-Field-Methode kann am ehesten der Gruppe der modifizierten Levelset-Verfahren zugeschrieben werden (siehe Paragraph 2.3.2.3) und geht auf Jacqmin zurück. Im Phase-Field-Modell versuchen sich zwei Komponenten aufgrund eines Potentials an freier Energie F zu entmischen, wodurch die verschmierte Grenzfläche direkter Bestandteil der theoretischen Grundüberlegung ist. [60] [59] Die zwei Komponenten werden durch ein Konzentrationsfeld c dargestellt. Die Modellierung der freien Energiedichte f erfolgt nach Gl. 2.28

$$f = \frac{1}{2} \cdot \varepsilon \cdot \sigma \cdot \alpha \cdot \left| \vec{\nabla} c \right|^2 + \frac{\sigma \cdot \beta}{\varepsilon} \cdot \Psi(c) \tag{2.28}$$

Die freie Energiedichte f setzt sich dabei aus einem Gradiententerm $|\vec{\nabla}c|^2$ und einem Potentialterm $\Psi(c)$ zusammen. Der Gradiententerm strebt nach der Vermeidung eines zu steilen Überganges von c zwischen den Phase. Der Potentialterm nimmt nur im Inneren der Kernphasen einen Wert von 0 an und liefert sonst stets einen positiven Beitrag zu f, wodurch Mischzustände sowie Über- bzw. Untersättigung unattraktiv werden. Die gesamte freie Energie F des Systems ist dann durch Gl. 2.29 definiert und strebt ein Gesamtminimum an, was einem Variationsproblem bezüglich c entspricht.

$$F = \int f(c, |\vec{\nabla}c|)\mathrm{d}V = \mathbf{min!} \tag{2.29}$$

Einfluss auf die Lösung des Variationsproblems haben die Parameter α und β, indem sie die Beiträge des Gradienten- und des Potientialterms unterschiedlich gewichten. Die Grenzflächenspannung zwischen den beiden Fluiden fließt durch σ ein. [5] Die Lösung von Problemen dieser Art wird typischerweise durch das Lösen einer Eulerschen Differentialgleichung erreicht (Gl. 2.30). [65]

$$\frac{\partial f}{\partial c} - \frac{\mathrm{d}}{\mathrm{d}x}\left(\frac{\partial f}{\partial |\vec{\nabla}c|}\right) = 0 \tag{2.30}$$

Lösungen dieser Gleichung lassen den Integralausdruck in Gl. 2.29 extremal werden und beschreiben den angestrebten Gleichgewichtszustand von c. Ist dieser Zustand nicht gegeben, lässt sich lokal an jedem Ort ein chemisches Potential Φ definieren, welches das Bestreben beschreibt, diesen Zustand zu erreichen.

$$\Phi = \frac{\delta F}{\delta c} = \frac{\sigma \cdot \beta}{\varepsilon} \cdot \Psi'(c) - \varepsilon \cdot \sigma \cdot \alpha \cdot \Delta c \tag{2.31}$$

Zu beachten ist hierbei die Variationsableitung (siehe [65]) von F nach c, welche nicht mit einer partiellen Ableitung verwechselt werden darf. Der Term $\Psi'(c)$ stellt hingegen die partielle Ableitung von Ψ nach c dar. Das zeitliche Verhalten von c lässt sich dann ähnlich zu Diffusionsprozessen über Gradienten des chemischen Potentials beschreiben. Jacqmin führt dies auf Cahn zurück. [59] [60]

$$\frac{\partial c}{\partial t} = D \cdot \vec{\nabla} * \left(\vec{\nabla}\Phi\right) = D \cdot \Delta\Phi \tag{2.32}$$

Die Wahl des Potentialterms $\Psi(c)$ beeinflusst die Form des Gleichgewichtszustandes. Jacqmin wählte einen Ansatz für $\Psi(c)$ nach Gl. 2.33 wodurch in der Kernphase von Fluid 1 der Wert $c = -1/2$ und in der Kernphase von Fluid 2 entsprechend $c = +1/2$ gilt. [59] [60] Im eindimensionalen Fall existiert durch diese Wahl von $\Psi(c)$ eine analytische Lösung von Gl. 2.30, welche ähnlich zu Olsson und Kreiss ([58], siehe Paragraph 2.3.2.3) ebenfalls die Form eines tangens hyperbolicus besitzt (Gl. 2.34).

$$\Psi(c) = \left(c - \frac{1}{2}\right)^2 \cdot \left(c + \frac{1}{2}\right)^2 \tag{2.33}$$

$$c(x) = \frac{1}{2} \cdot \tanh\left(\frac{x}{\varepsilon} \cdot \sqrt{\frac{\beta}{2 \cdot \alpha}}\right) \tag{2.34}$$

[5] Damit die Energie, welche durch die Grenzflächenspannung σ bedingt in der Grenzfläche zweier Fluiden enthalten ist, akkurat beschrieben wird, ergibt sich eine Randbedingung bezüglich der Wahl von α und β in Abhängigkeit von σ, auf die hier jedoch nicht näher eingegangen werden soll. Nähere Informationen sind in [60] und [88] zu finden.

Der Diffusionprozess sorgt automatisch dafür, dass die Form des tangens hyperbolicus wieder hergestellt wird, falls durch numerische Diffusion oder Koaleszenz zweier Tropfen lokal eine Abweichung vom Gleichgewichtszustand entsteht. Für die Beschreibung einer mehrphasigen Strömung mit Einfluss von Kapillarkräften durch das Phase-Field-Modell ist eine Einbindung der Kapillarkraft in die Kräftebilanz und der zusätzliche Transport von c durch Konvektion notwendig. Hierfür führt Jacqmin Gl. 2.35 und Gl. 2.36 an ([59] [60]):

$$\rho \cdot \frac{\mathcal{D}u_i}{\mathcal{D}t} = -\vec{\nabla}S + \eta \cdot \Delta u_i - c \cdot \vec{\nabla}\Phi + g_i \cdot \rho(c) \tag{2.35}$$

$$\frac{\partial c}{\partial t} = -\vec{\nabla} * (\vec{u} \cdot c) + D \cdot \Delta\Phi \tag{2.36}$$

Die Variable S repräsentiert nicht direkt den Druck p, sondern garantiert lediglich die Inkompressibilität $\vec{\nabla} * \vec{u} = 0$. Die Umrechnungsmöglichkeiten sind in [59] gegeben. Die direkte Kopplung des Drucksprungs an das chemische Potential erlaubt dabei die Konstruktion von Methoden, die auch in numerisch diskretisierter Form vollständig energieerhaltend sind. Kinetische Energie wird auch in den numerisch diskretisierten Gleichungen direkt in Grenzflächenenergie umgewandelt bzw. in der Gegenrichtung ausgetauscht. [59] Die Menge an ausgetauschter Energie mag durch Diskretisierungsfehler fehlerbehaftet sein, doch wird bei korrekter Implementierung weder Energie erzeugt noch vernichtet. Erfolgt kein Energieaustausch mit der Umgebung bleibt die Gesamtenergie des Systems konstant und kann höchstens durch Dissipation innerhalb des Systems abnehmen. Zu nennen sind hier viskose Reibung durch η und der Diffusionsvorgang durch Φ. Ein solches System ist zwangsläufig frei von parasitären Strömungen [83], was als einer der Hauptvorteile des Phase-Field-Modells angesehen werden kann. Jacqmin verwendet in allen seinen Berechnungen das Ψ, welches in Gl. 2.33 gegeben ist. [59] [60] Prinzipiell sind auch andere Potentialformen möglich, wie z.B. Nochetto et al. ([89]) zeigten. Dies beeinflusst entscheidend die Form des Gleichgewichtszustandes (Gl. 2.34) und der tangens hyperbolicus ist nicht mehr die analytische Lösung von Gl. 2.30. Hierauf wird in Abschnitt 3.3 genauer eingegangen werden.

Den genannten Vorteilen des Phase-Field-Modells steht jedoch ein Nachteil gegenüber, welcher beim Transport der Grenzfläche durch das Rechengitter auftreten kann. Ist die Grenzfläche in ihrem Gleichgewichtszustand, so ist Φ konstant. Eine Auslenkung von c durch numerische Fehler führt jedoch zu einer Auslenkung von Φ, welche sich direkt auf S und damit den Druck p auswirkt und so das Fluid abbremst. [59] Es ist daher möglich, dass Druckverluste nicht korrekt dargestellt werden. Tropfen werden durch dieses Phänomen unter Umständen langsamer transportiert, als durch das umströmende Fluid vorgegeben würde. Auch Druckverluste bei der Durchströmung von Bauteilen werden überschätzt und falsch abgebildet, was die Methode in der vorliegenden Form ungeeignet für die Simulation einer auf Druckverlusten basierenden Phasentrennung macht. Anwendung findet das Phase-Field-Modell daher unter Anderem in der Berechnung von Pfropfenströmungen, die sich nicht durch das Rechengitter bewegen. So nutzten He und Kasagi einen mit dem Pfropfen bewegten Beobachter und berechneten mit dieser Methode die Luft-Wasser-Strömungen in Kanälen mit 600 µm Durchmesser. Sie zeigten, dass im Vergleich zur sonst verwendeten CSF-VOF-Methode von Brackbill et al. (siehe Unterabschnitt 2.3.3) die parasitären Strömungen deutlich reduziert werden können. [90] Durch den mitbewegten Beobachter stand der Pfropfen effektiv im Rechengitter still und die

Phasengrenzfläche wurde keinen numerischen Diffusionseffekten ausgesetzt. Ganapathy et. al. untersuchten in ihren Arbeiten die Pfropfenerzeugung ([88]) und die Hydrodynamik innerhalb der Pfropfen ([91]). Auch sie verzeichneten einen massiven Rückgang parasitärer Strömungen, aber eine starke Abhängigkeit der Ergebnisse vom gewählten Mobilitätsparameter D. Die Abhängigkeit vom Mobilitätsparameter D, welcher keinerlei physikalischen Stoffdaten entnommen werden kann, stellt einen weiteren Nachteil der Methode dar.

2.3.5 Das THINC-Schema zur Advektion von Grenzflächen

Für die Reduktion parasitärer Strömungen ist neben der genauen Berechnung der Krümmung κ ebenso ein möglichst genauer Transport der Grenzfläche selbst notwendig. [92] Eine spezielle Art der VOF-Methode mit diesem Ziel stellt das THINC-Verfahren dar. Typischerweise wird in VOF-Verfahren die Position der Grenzfläche explizit, z.B. als lineare Ebene, rekonstruiert und geometrisch um die Distanz $\vec{u} \cdot \Delta t$ transportiert (siehe Paragraph 2.3.2.1). Aus der geometrischen Verschiebung ergeben sich die Flüsse der Indikatorfunktion c über die Zellwände, welche zur Berechnung von c für den nächsten Zeitschritt genutzt werden. Im Kontrast dazu stehen die algebraischen VOF-Methoden, welche auf eine geometrische Rekonstruktion verzichten und für den Transport stattdessen Gl. 2.37 über spezielle Diskretisierungsverfahren lösen.

$$\frac{\partial c}{\partial t} + \vec{\nabla} * (\vec{u} \cdot c) = 0 \tag{2.37}$$

Um numerische Diffusion so gut wie möglich zu unterbinden wird besonders berücksichtigt, dass die Indikatorfunktion einen Phasenanteil darstellt und die Grenzfläche direkt am Ort des Phasenwechsels innerhalb der Zelle einen steilen Übergang von $c = 0$ auf $c = 1$ erfordert. Der Verlauf der Indikatorfunktion innerhalb einer Zelle wird daher durch einen tangens hyperbolicus nach Gl. 2.38 angenähert. [61]

$$c = \frac{1}{2} \cdot (1 + \tanh(\beta * (P(x, y, z) + d))) \tag{2.38}$$

Der Parameter β reguliert, wie schnell der Übergang erfolgt. Je größer β zu Beginn der Simulation gewählt wird, desto besser wird der instantane Übergang wiedergegeben. Der Gesamtterm $P(x, y, z) + d$ stellt ein Polynom beliebiger Ordnung mit Absolutglied dar, dessen Nullniveau die Position der Grenzfläche repräsentiert. Das Polynom $P(x, y, z)$ wird so bestimmt, dass deren Ableitungen möglichst gut die Normalenvektoren $\vec{n}$ in den Zellecken repräsentieren, während d iterativ so bestimmt wird, dass bei einmalig bestimmten $P(x, y, z)$ möglichst gut das Volumenintegral nach Gl. 2.39 erfüllt wird.

$$\bar{c} = \frac{1}{V} \cdot \iiint c \, \mathrm{d}V = \frac{1}{V} \cdot \iiint \frac{1}{2} \cdot (1 + \tanh(\beta * (P(x, y, z) + d))) \, \mathrm{d}V \tag{2.39}$$

In Weiterentwicklungen der Methode wird nicht mehr strikt gefordert, dass c direkt den Phasenanteil repräsentiert, wie in der ursprünglichen VOF-Methode intendiert war ([75]). Die Grenze zwischen algebraischen VOF-Methoden und der modifizierten Levelset-Methode nach Olsson und Kreiss ([58], siehe Paragraph 2.3.2.3) verschwimmt dadurch. Um den integralen Mittelwert nach Gl. 2.39 zu berechnen, werden Zellgeometrie und Normalenvektoren zunächst auf eine Einheitszelle transformiert. Derartige Transformationen existieren z.B. für Vierecke und Dreiecke im Zweidimensionalen, sowie für Hexaeder, Tetraeder, Prismen und Pyramiden im Dreidimensionalem ([62], [63]), und gestalten die Berechnung des Polynoms besonders einfach. Zudem ist die Volumenintegration auf Einheitszellen wesentlich einfacher durchzuführen und kann zumindest in einer Raumrichtung analytisch erfolgen, sofern sich $P(x, y, z)$ in dieser Raumrichtung linear verhält. Die anderen Raumrichtungen müssen numerisch integriert werden, was großen Rechenaufwand bedeutet und nach Gauss'schen Quadraturformeln erfolgt. [62] [63] Nach der Bestimmung von $P(x, y, z)$ und d ist der Verlauf von c in der Zelle und auf den Zellwänden gut

bekannt, wodurch die notwendigen Flüsse zur Lösung von Gl. 2.37 sehr genau berechenbar sind. Die zeitliche Integration erfolgt häufig über ein TVD-Runge-Kutta-Verfahren. Zhao et al. setzten in ihrer Arbeit hingegen die einmalige analytische Integration in der Zeit ein und führten dafür alle räumlichen Integrationen numerisch durch. Hierdurch entstanden jedoch ebenfalls starke Einschränkungen bzgl. der maximalen Ordnung von $P(x, y, z)$. [93] Jüngste Arbeiten von Qian et. al. koppeln die THINC-Methode mit einer Levelset-Methode und ermöglichen dadurch auch eine akkurate Berechnung von kapillarkraftdominierten Strömungen. [94] Darüber hinaus ist die Methode in weiteren Varianten auch zur Berechnung von konvektiv dominierten Strömungen geeignet, wie in zahlreichen Simulationen des Dammbruchproblems gezeigt wurde. [95] [96] [97]

Das THINC-Verfahren stellt damit bereits einen guten Ansatz zum genauen Transport von Grenzflächen dar, benötigt für eine hohe Genauigkeit jedoch viele Funktionsauswertung zur Berechnung der Volumen- und Flächenintegrale über Gauß'sche Quadraturformeln. Zudem ist die Anwendung auf Zellgeometrien limitiert, welche auf eine Standardzelle mit vorhandener Quadraturformel transformiert werden können.

Kapitel 3

Entwicklung der numerischen Methode

In diesem Kapitel wird auf die Entwicklung und Herleitung der numerischen Methode dieser Arbeit eingegangen, da bisherige Standardmethoden ungeeignet für die angestrebte Simulation der Phasentrennung sind. Zunächst wird in Abschnitt 3.1 allgemein erläutert, warum die Entwicklung einer verbesserten Simulationsmethode für notwendig erachtet wird. Die Entstehung parasitärer Strömungen als Folge von Diskretisierungsfehlern wurde diesbezüglich bereits in Unterabschnitt 2.3.3 diskutiert. Die entwickelte Methode soll besonders arm an parasitären Strömungen sein, ohne dabei das Rechengitter unnötig stark zu verfeinern. Essentielle Punkte werden hierzu in Abschnitt 3.2 erläutert. Daraus abgeleitet wird die Notwendigkeit, die Diskretisierungsfehler bei der Berechnung von Oberflächen- und Volumenintegralen nach der Finite-Volumen-Methode (siehe Unterabschnitt 2.3.1) möglichst stark zu reduzieren. Genannte Integrale werden momentan über numerische Näherungsformeln ausgewertet (siehe Unterabschnitt 2.3.5). Eine analytische Integration kann an dieser Stelle sowohl die Geschwindigkeit wie auch die Genauigkeit der Simulation verbessern. Um die analytische Flächen- und Volumenintegration zu ermöglichen muss bezüglich der Grenzflächenformfunktion zunächst vom bisher gängigen tangens hyperbolicus abgewichen werden, da dieser nur einmal geschlossen integrierbar ist. Wichtig ist dabei die Existenz eines Kompressionsterm für die Grenzfläche, welche die gewählte Formfunktion auch als Gleichgewichtszustand anstrebt. Die zugehörigen Analysen hierzu erfolgen in Abschnitt 3.3. Die durchzuführende Oberflächen- und Volumenintegration soll dabei auf beliebigen Zellgeometrien analytisch möglich sein. Ein Algorithmus hierzu wird in Abschnitt 3.4 erarbeitet. Die Parameter der Formfunktion werden daraufhin in Abschnitt 3.5 optimiert und das numerische Verhalten der entstandenen Methode in Abschnitt 3.6 charakterisiert. Auf die konsistente Implementierung eines vorgegebenen Kontaktwinkels bei Kontakt der Grenzfläche mit einer Wand wird in Abschnitt 3.7 eingegangen. Abschließend wird die numerische Methode in Abschnitt 3.8 mit Literaturdaten validiert und in Abschnitt 3.9 mit dem bisherigen Solver für Mehrphasenströmung interFoam verglichen.

3.1 Motivation zur Weiterentwicklung des numerischen Verfahrens

Bisherige Methoden zur Simulation vor kapillarkraftdominierten, mehrphasigen Strömungen leiden unter dem Phänomen von parasitären Strömungen, welche teilweise die gleiche Größenordnung erreichen können, wie die zu untersuchende Hauptströmung. Hierdurch können Stofftransportraten überschätzt werden, weil zusätzliche Wirbel eine stärkere Durchmischung suggerieren. In anderen Fällen können Tropfen durch parasitäre Strömungen abgelenkt werden, was zu einer veränderten Position innerhalb des Rechengebietes führt. So kann ein liegender Tropfen, der eigentlich keinerlei äußeren Kräften ausgesetzt ist, sich bei schlecht gewählter numerischer Methode durch parasitäre Strömungen dennoch in Bewegung setzen (siehe Abschnitt 3.9). Die Eindämmung von parasitären Strömungen für liegende Tropfen gelingt seit der Arbeit von Francois et al. bereits sehr gut durch das exakte Ausgleichen von Drucksprung und Grenzflächenspannung in den bereits diskretisierten Gleichungen. [84] Für bewegte Phasengrenzflächen werden parasitäre Strömungen jedoch kaum betrachtet. Shams et al. zeigen zumindest, das bei Anwendung ihrer Methode der Transport des untersuchten Tropfens nur unter vergleichsweise kleinen parasitären Strömungen leidet. [80] Für eine numerische Untersuchung zur Phasentrennung einer flüssig-flüssig-Pfropfenströmung, wie sie in dieser Arbeit erfolgen soll, wird jedoch eine verlässliche Berechnung von Drucksprung, Grenzflächenspannung und Position der Phasengrenzfläche benötigt, da bereits wenige Pascal Druckunterschied über eine erfolgreiche Phasentrennung entscheiden können. Daher soll eine numerische Methode entwickelt werden, die auch beim konvektiven Transport von Phasengrenzflächen zur Abbildung einer flüssig-flüssig Pfropfenströmung besonders arm an parasitären Strömungen ist. Eine Beschleunigung der Berechnungen und Verbesserung der Genauigkeit sind dabei ebenso ein Ziel dieser Arbeit.

3.2 Diskretisierungsfehler und Kernpunkt der entwickelten Methode

Hauptansatz der hier entwickleten Methode ist die starke Reduktion der entscheidenden Diskretisierungsfehler, wofür die Volumen- und Flächenintegration der Finite-Volumen-Methode (Gl. 2.11 und Gl. 2.12) als Hauptquelle identifiziert wurden (siehe Unterabschnitt 2.3.5). Da numerische Diskretisierungsfehler nur eingedämmt aber nicht komplett eliminiert werden können, wird ein passender Kompressionsterm entwickelt, welcher die Akkumulation solcher Fehler ausgleichen soll, ohne dabei weitere Fehler in die Ergebnisse der Simulation einzubringen. Ansätze hierfür werden den Arbeiten von Olsson und Kreiss ([58]) und Jacqmin ([60], [59]) entnommen. Bisherige Arbeiten zur Verringerung des örtlichen Diskretisierungsfehler ohne Verwendung von Levelsetfunktionen konzentrierten sich auf die Anwendung von Methoden hoher Ordnung, die jedoch nicht auf allen Gitterarten anwendbar sind und auch nur dann effektiv sind, wenn die Grenzfläche über viele Zellen hinweg aufgelöst wird. Beispielhaft soll dies an der Interpolation von c mit einem Profil in Form des tangens hyperbolicus diskutiert werden, wie es auch in den Methoden von Olsson und

Kreiss ([58]) und Jacqmin ([59], [59]) ähnlich vorkommt.

$$c = \tanh\left(\frac{x}{\varepsilon}\right) \tag{3.1}$$

Interpolationen von c durch Polynome erreichen bei Verwendung eines Polynoms vom Grad m eine Ordnung von $m+1$. Der Fehler kann nach einer Taylor-Reihen-Entwicklung ähnlich zu Gl. 3.2 ausgedrückt werden, wobei C_m eine Konstante ist, die nur noch von m und nicht mehr von x abhängig ist.

$$c_{int.} - c_{exakt} = C_m \cdot \frac{\partial^{(m+1)} c}{\partial x^{(m+1)}}(\xi) \cdot (\Delta x)^{(m+1)} \tag{3.2}$$

Dabei ist die m-fache Ableitungen von c jedoch proportional zu $1/\varepsilon^m$. Sind ε und die Gitterweite Δx in der gleichen Größenordnung entsteht dadurch kaum ein Zugewinn an Genauigkeit. Lediglich durch ein stetes Absinken der Konstante C_m oder anderweitiges Abfallen der höheren Ableitungen von c entstehen durch Verwendung von Polynomen höherer Ordnung auch höhere Genauigkeiten.[1] Beispielhaft ist dieser Effekt in Anhang A.1 in Tabelle A.1 dargestellt. Ein starker Abfall des Inteprolationsfehlers mit steigender Polynomordnung m ist nur dann zu beobachten, wenn das Verhältnis $\Delta x/\varepsilon$ Werte kleiner als 1 annimmt. Wird stattdessen eine Unterfunktion L eingeführt, die eine Transformation von c darstellt, kann stattdessen auch L interpoliert werden. Dies bietet dann Vorteile, wenn die höheren Ableitungen von L betragsmäßig wesentlich kleiner sind als die höheren Ableitungen von c. Im hier konstruierten Beispiel würde die Definition von L nach Gl. 3.3 den Interpolationsfehler direkt vollständig eliminieren, da $L = x/\varepsilon$ exakt interpoliert werden kann.

$$c = \tanh(L) \quad \text{mit} \quad L = \operatorname{arctanh}(c) = \frac{x}{\varepsilon} \tag{3.3}$$

Ist L selber nicht perfekt linear, so gilt für den Interpolationsfehler von $L_{int.}$ ebenfalls die Abschätzung aus Gl. 3.2 bezogen auf L statt auf c. Der Interpolationsfehler für c sinkt dadurch allerdings erheblich, wie Gl. 3.4 verdeutlicht.

$$\begin{aligned} c_{int.} &= \tanh(L_{int.}) \\ &= \tanh\left(L_{exakt} + C_m \cdot \frac{\partial^{(m+1)} L}{\partial x^{(m+1)}}(\xi) \cdot (\Delta x)^{(m+1)}\right) \\ &= \tanh(L_{exakt}) + \frac{\partial \tanh(L)}{\partial L}(\chi) \cdot C_m \cdot \frac{\partial^{(m+1)} L}{\partial x^{(m+1)}} \cdot (\xi) \cdot (\Delta x)^{(m+1)} \\ &= c_{exakt} + \mathcal{O}(1) \cdot C_m \cdot \frac{\partial^{(m+1)} L}{\partial x^{(m+1)}} \cdot (\xi) \cdot (\Delta x)^{(m+1)} \end{aligned} \tag{3.4}$$

Zunächst kann $L_{int.}$ durch eine Taylor-Reihen-Entwicklung um den exakten Wert L_{exakt} herum dargestellt werden. Anschließend kann auch der tangens hyperbolicus in einer Taylor-Reihe dargestellt werden, wodurch eine Abschätzung für $c_{exakt} = \tanh(L_{exakt})$ entsteht. Die höheren Ableitungen von L sind im Allgemeinen nur proportional zu $1/\varepsilon$ und nicht zu $1/\varepsilon^m$. Die Ableitung des tangens hyperbolicus bezüglich L besitzt eine Größenordnung von 1. Der Interpolationsfehler sinkt mit wachsendem Polynomgrad m also um den Faktor Δx statt um den Faktor $\Delta x/\varepsilon$, falls

[1] Die Konstante C_m skaliert in etwa invers mit der Fakultät $m!$. Durch die anspruchsvolle Implementierung werden jedoch selten Werte $m > 4$ verwendet.

die Transformation $c \rightarrow L$ eine nahezu lineare Funktion L entstehen lässt. Die Definition von L erinnert dabei stark an die Einführung einer Levelset-Funktion nach Sussmann et al. ([76]) bzw. Osher und Sethian ([77]). Neben den vielen Gemeinsamkeiten existieren allerdings auch Unterschiede, die an dieser Stelle angesprochen werden sollen.

- Eine echte Levelset-Funktion ist darüber definiert, dass sie an jedem Ort $\vec{x}$ den kürzesten Abstand zur Phasengrenzfläche anzeigt. Dem gegenüber ist die hier eingeführte Unterfunktion L direkt als Transformation $L = f^{-1}(c)$ definiert. Die Eigenschaft einer Abstandsfunktion ist daher nicht gegeben. Dafür enthält L jedoch implizit den genauen Verlauf von c, was in gängigen CLS-VOF-Methoden nicht gegeben ist.

- Für eine Levelset-Funktion wird gefordert, dass ihr Gradient stets einen Betrag von 1 haben soll. Dem gegenüber existiert für die hier eingeführte Unterfunktion L keinerlei Anforderung solcher Art. Auch der lineare Verlauf von L ist nicht garantiert. Wenn jedoch die Form des tangens hyperbolicus durch numerische Diffusion nicht zu stark ausgelenkt wird, oder die Form durch adäquate Kompressionsterme wiederhergestellt wird, gilt in guter Näherung $|\vec{\nabla} L| \simeq 1/\varepsilon$ und L besitzt dadurch nahezu lineare Eigenschaften.

Die nahezu linearen Eigenschaften von L können darüber hinaus zur Berechnung der Differentialgrößen $\vec{\nabla} c$ und Δc von $c = f(L)$ im Zellzentrum genutzt werden.

$$\vec{\nabla} c = \vec{\nabla} f(L) = \frac{\partial f}{\partial L}(L) \cdot \vec{\nabla} L \tag{3.5}$$

$$\Delta c = \vec{\nabla} * \left(\vec{\nabla} c\right) = \frac{\partial f}{\partial L}(L) \cdot \Delta L + \frac{\partial^2 f}{\partial L^2}(L) \cdot \left|\vec{\nabla} L\right|^2 \tag{3.6}$$

Zeitliche Diskretisierungsfehler treten ebenso auf, sind bei Verwendung von Runge-Kutta-Verfahren, dessen Diskretisierungsfehler proportional zur m-fachen zeitlichen Ableitung von c ist, jedoch vernachlässigbar klein.

$$c = \tanh\left(\frac{x - u \cdot t}{\varepsilon}\right) \tag{3.7}$$

$$\frac{\partial^m c}{(\partial t)^m} \cdot \Delta t^m = \mathcal{O}(1) \cdot \left(\frac{-u \cdot \Delta t}{\varepsilon}\right)^m \tag{3.8}$$

Durch die Einschränkung der Zeitschrittweite Δt über die Courant-Zahl Co und die Grenzflächenspannung nach Gl. 2.27, sind zeitliche Diskretisierungsfehler in der Größenordnung von Gl. 3.8 oft akzeptabel klein genug, wie auch Abschnitt 3.8 noch zeigt.

Ansätze dieser Art wurden bereits in den Arbeiten von Qian et al. ([94]), Xiao et al. ([61]) und Xie et al. ([62], [63]) verwendet (siehe Unterabschnitt 2.3.5), doch sind Transportvorgänge niemals komplett frei von Diskretisierungsfehlern. Die Form des tangens hyperbolicus wird daher beim Transport der Phasengrenzfläche über große Distanzen gestört und muss durch adäquate Kompressionsterme wieder hergestellt werden. Dies erfolgte, nach Kenntnis des Autors, bisher in keiner vorliegenden Arbeit mit anschließender Analyse der parasitären Strömungen. Die Verbesserung in

der Berechnung von Flächen- und Volumenintegralen soll durch eine analytische Integration erfolgen. Im Gegensatz zu den bisherigen numerischen Integrationsmethoden ist eine analytische Integration schneller und in den meisten Fällen auch genauer durchführbar. Dies erfordert auf der einen Seite eine Formfunktion für c, die im Gegensatz zum tangens hyperbolicus mindestens drei mal geschlossen analytisch integrierbar ist (siehe Abschnitt 3.3), und auf der anderen Seite einen passenden Integrationsalgorithmus, der im besten Fall auf beliebigen Gittergeometrien anwendbar ist (siehe Abschnitt 3.4).

3.3 Neudesign der Grenzflächenformfunktion $f(L)$

Ziel dieses Abschnittes ist die Definition einer neuen Grenzflächenformfunktion, die den Verlauf von c als Grenzfläche beschreibt. Dabei werden diverse Forderungen an die neue Formfunktionen $f(L)$ gestellt.

- Die Formfunktion $f(L)$ soll mindestens 3 mal geschlossen analytisch integrierbar sein, um eine Stammfunktion für den im Folgenden noch näher beschriebenen Integrationsalgorithmus angeben zu können.
- Die Funktionsauswertungen von $f(L)$ und den Stammfunktionen $F_1(L)$, $F_2(L)$ und $F_3(L)$ sollen schnell erfolgen können und keine unendlichen Reihenauswertungen mit langsamer Konvergenzgeschwindigkeit beinhalten.
- Es muss ein Kompressionsterm angegeben werden können, dessen Lösung im Gleichgewichtszustand genau $f(L)$ entspricht.
- Für die effektive Anwendung von Runge-Kutta-Verfahren muss $f(L)$ ausreichend oft stetig differenzierbar sein.
- Die Funktion $f(L)$ soll einfach und eindeutig [2] invertierbar sein, um die Transformierte $L = f^{-1}(L)$ schnell berechnen zu können.
- $f(L)$ soll einen möglichst raschen Übergang zwischen den Phasen beschreiben und für große Werte von $|L|$ möglichst schnell den Grenzwert in der Kernphasen annehmen.

Nicht allen Forderungen kommt dabei die gleiche Wichtigkeit zu und einige Kriterien widersprechen sich gegenseitig. Die größte Wichtigkeit besitzen die analytische Integrierbarkeit und die Existenz eines Kompressionsterms, der $c = f(L)$ als Gleichgewichtszustand anstrebt. Hierzu wird zunächst eine Analyse der bisherigen Kompressionsterme durchgeführt und der entscheidende Faktor identifiziert, welcher den tangens hyperbolicus als bisherige Formfunktion vorgegeben hat. Darauf folgt die Ausarbeitung der konkreten Wahl für $f(L)$.

3.3.1 Analyse bekannter Kompressionsterme zur Neudefinition von $f(L)$

In der Arbeit von Olsson und Kreiss ([58]) definiert die Funktion $\vec{k}(c)$ in Gl. 2.22 die Stärke des komprimierenden Geschwindigkeitsfeldes. Der Gleichgewichtszustand

[2] Gefordert wird die Bijektivität der Abbildung $c = f(L)$

wiederum definiert sich durch das Gegenspiel von konvektiver Kompression und diffusiver Glättung. Im stationären Endzustand des eindimensionalen Falles gilt daher Gl. 3.9

$$\cancelto{0}{\frac{\partial c}{\partial \tau}} + \frac{\partial}{\partial x}\left(k(c)\right) = \varepsilon \cdot \frac{\partial^2 c}{\partial x^2} \tag{3.9}$$

Die Wahl von $k(c)$ definiert demnach den Gleichgewichtszustand. Eine Integration nach x zeigt, dass Lösungen dieser Differentialgleichung ebenfalls Gl. 3.10 erfüllen.

$$k(c) = \varepsilon \cdot \frac{\partial c}{\partial x} \tag{3.10}$$

Zu gegebener Funktion $c(x)$ kann daher berechnet werden, welche Wahl von $k(c)$ notwendig ist, um $c(x)$ als Gleichgewichtszustand zu erhalten. Dabei sollten die Ableitungen von $c(x)$ die Ursprungsfunktion $c(x)$ wieder selbst enthalten. Existiert eine explizite Inverse, ist dies sehr einfach möglich. Wird beispielhaft für $c(x)$ die Form eines arcus tangens angestrebt, ergeben sich folgende Überlegungen:

$$c(x) = \frac{2}{\pi} \cdot \arctan\left(\frac{x}{\varepsilon}\right) \tag{3.11}$$

$$\begin{aligned} k(c) &\overset{\text{Gl.3.10}}{=} \varepsilon \cdot \frac{\partial}{\partial x}\left(\frac{2}{\pi} \cdot \arctan\left(\frac{x}{\varepsilon}\right)\right) = \varepsilon \cdot \frac{2}{\pi} \cdot \frac{1}{1+\left(\frac{x}{\varepsilon}\right)^2} \cdot \frac{1}{\varepsilon} \\ &\overset{\text{Gl.3.11}}{=} \frac{2}{\pi} \cdot \frac{1}{1+\tan^2\left(\frac{\pi}{2} \cdot c\right)} \qquad = \frac{2}{\pi} \cdot \cos^2\left(\frac{\pi}{2} \cdot c\right) \end{aligned} \tag{3.12}$$

Ein Nachteil dieser Methode liegt jedoch in der schwierig einzuhaltende Massenerhaltung in Wandnähe im mehrdimensionalen Fall, wo nach Gl. 3.13 Normalenvektor und Laplace-Operator benötigt werden.

$$\cancelto{0}{\frac{\partial c}{\partial \tau}} + \vec{\nabla} * (\vec{n} \cdot k(c)) = \varepsilon \cdot \Delta c \tag{3.13}$$

Damit der Laplace-Operator keinen effektiven Fluss der Komponente c über die Wand erzeugt ist für c die Randbedingung zeroGradient zu wählen. Diese beeinflusst jedoch auch die Berechnung von $\vec{\nabla} c$ und damit von $\vec{n}$. Der gebildete Normalenvektor würde stets einen Kontaktwinkel von 90° anstreben. Wird nicht die Randbedingung zeroGradient gewählt ist stattdessen auf andere Weise sicher zu stellen, dass der diffusive Fluss von Δc exakt durch einen konvektiven Gegenfluss über $\vec{\nabla} * (\vec{n} \cdot k(c))$ kompensiert wird.

Neben der Arbeit von Olsson und Kreiss ([58]) kann auch aus dem Phase-Field-Ansatz nach Jacqmin ([59], [60]) ein Kompressionsterm abgeleitet werden, der auf einem chemischen Potential Φ basiert. [60] Der Kompressionsterm folgt dabei Gl. 3.14.

$$\cancelto{0}{\frac{\partial c}{\partial \tau}} = D \cdot \Delta \Phi \tag{3.14}$$

$$\Phi(c) = \frac{\Psi'(c)}{\varepsilon} - \varepsilon \cdot \Delta c \tag{3.15}$$

Im eindimensionalen Fall wird wieder deutlich, dass der Gleichgewichtszustand durch das chemische Potential Φ definiert und damit direkt über die Wahl von $\Psi(c)$ bestimmt wird. Gl. 3.14 zeigt, dass Φ im Gleichgewichtszustand überall konstant sein muss, der Absolutwert jedoch keine Rolle spielt. Dieser kann nach Gl. 3.15 auch 0 betragen, wenn $\Psi(c)$ passend gewählt wird. Daraus lassen sich analoge Überlegungen wie sie bei der Variation des Kompressionsterms nach Olsson und Kreiss ([58]) herleiten:

$$c(x) = \frac{2}{\pi} \cdot \arctan\left(\frac{x}{\varepsilon}\right) \tag{3.16}$$

$$\begin{aligned} \Psi'(c) \overset{\text{Gl.3.15}}{=} \varepsilon^2 \cdot \frac{\partial^2 c(x)}{\partial x^2} &= \varepsilon^2 \cdot \frac{\partial}{\partial x}\left(\frac{2}{\pi \cdot \varepsilon} \cdot \cos^2\left(\frac{\pi}{2} \cdot c\right)\right) \\ &= -\frac{4}{\pi}\cos^3\left(\frac{\pi}{2} \cdot c\right) \cdot \sin\left(\frac{\pi}{2} \cdot c\right) \end{aligned} \tag{3.17}$$

Durch gezielte Wahl von $\Psi'(c)$ lassen sich demnach auch aus dem Phase-Field-Ansatz Kompressionsterme erstellen, die eine bestimme Zielfunktion $c(x)$ als angestrebten Gleichgewichtszustand besitzen.[3] Wieder ist dieser Vorgang besonders einfach zu realisieren, wenn eine explizite Inverse von $c(x)$ existiert und genutzt werden kann. Die Übertragung des Phase-Field-Ansatzes auf den mehrdimensionalen Fall bedarf allerdings noch weiteren Überlegungen, um den Phänomen der Übersättigung $|c| > 1$ in den beiden Kernphasen entgegen zu wirken. Der Phase-Field-Ansatz basiert auf der Minimierung der freien Energie F (siehe Gl. 2.29) eines Systems. Yue et al. analysierten in ihrer Arbeit bereits das Zusammenschrumpfen eines Tropfens durch Bilanzierung der freien Energie F. Energiebeiträge entstehen auf der einen Seite durch $|\vec{\nabla}c|$ im Bereich der Grenzfläche und sind proportional zur Oberfläche eines Tropfens, da nur hier $|\vec{\nabla}c|$ nennenswert groß ist. Auf der anderen Seite entstehen Energiebeiträge durch Abweichungen des Feldes c von den Potentialminima von $\Psi(c)$. Da die Minimalwerte von $\Psi(c)$ bei $c = \pm 1$ liegen, liefert jede Abweichung $|c| \neq 1$ einen Energiebeitrag der proportional zum Tropfenvolumen ist. Yue et al. wiesen nach, dass ein Tropfen durch Schrumpfen stets Gesamtenergie freisetzen kann, da hierdurch die energieenthaltende Oberfläche verkleinert wird. [98] Der Tropfen schrumpft solange weiter, bis die daraus resultierende Übersättigung im Tropfeninneren ($c > 1$) bei gleichzeitiger Untersättigung im Außenraum ($c > -1$) mehr Energie benötigen würde, als durch ein weiteres Schrumpfen freigesetzt werden kann. Ein solches Schrumpfen des Tropfens verschiebt den Gleichgewichtszustand des Kompressionsterms in Abhängigkeit der verwendeten Simulationsparameter ($R_{Tropfen}$, ε, $V_{ges.}$) und der Anzahl an Raumdimensionen. Werden Werte $c > 1$ angestrebt, fällt dies jedoch aus dem Definitionsbereich der Transformation $L = f^{-1}(c)$. Die Transformation ist dann nicht mehr anwendbar. In den Übergangsbereichen $|c| \simeq 1$ verliert L zudem die annähernd linearen Eigenschaften, sodass die Berechnung von $\vec{\nabla}c$ und Δc über L durch Anwendung von Gl. 3.5 und Gl. 3.6 zu größeren Fehlern führen kann als die direkte Diskretisieurung von c selbst.

Zur Lösung dieses Problems wurde in dieser Arbeit ein Ansatz entwickelt, welcher die Abhängigkeit der Cahn-Hillard-Gleichung (Gl. 3.14) von der Anzahl der Raumdimensionen aufheben soll. Hierfür wird der Laplace-Operator durch eine Ableitung

[3] Es fällt die Analogie auf, dass stets eine Ableitung des Feldes c mit ihrem Sollwert im Gleichgewichtszustand verglichen wird. Olsson und Kreiss ([58]) vergleichen $\vec{\nabla}c$ mit $k(c)$, während im Phase-Field-Ansatz eine Übereinstimmtung von Δc mit $\Psi'(c)$ angestrebt wird.

des Gradientenbetrags in Normalenrichtung der Grenzfläche ersetzt.

$$\Delta c \rightarrow \vec{n} * \vec{\nabla} \left(|\vec{\nabla} c| \right) \tag{3.18}$$

Diese zunächst willkürlich erscheinende Substitution bietet den Vorteil, dass der erhaltene Wert unabhängig von der Krümmung κ der Oberfläche und den verwendeten Raumdimensionen wird. Während im eindimensionalen Fall zwangsläufig nur eine Raumrichtung in Betracht gezogen wurde, wird im mehrdimensionalen eine gezielte Richtungsableitung in die einzige Richtung von Interesse, in Richtung des Normalenvektors, gebildet. Eine effektive Berechnung der numerisch sonst nur schwer zugänglichen Ableitung in Normalenrichtung ist mit Gl. 3.19 und Gl. 3.20 möglich.

$$\kappa = -\vec{\nabla} * \vec{n} = -\vec{\nabla} * \left(\frac{\vec{\nabla} c}{|\vec{\nabla} c|} \right) = -\frac{\Delta c \cdot |\vec{\nabla} c| - \vec{\nabla} c * \vec{\nabla} \left(|\vec{\nabla} c| \right)}{|\vec{\nabla} c|^2} \tag{3.19}$$

$$\vec{n} * \vec{\nabla} \left(|\vec{\nabla} c| \right) = \Delta c + \kappa \cdot |\vec{\nabla} c| \tag{3.20}$$

Die Addition von $\kappa \cdot |\vec{\nabla} c|$ projeziert demnach auf einfache Weise den unerwünschten Anteil aus Δc heraus. Vorteil dieser Abänderung ist eine vollständige Eliminierung der Übersättigung und eine Wiederherstellung der annähernd linearen Eigenschaften von L. Die erhaltene modifizierte Cahn-Hillard-Gleichung repräsentiert nun nicht mehr den Ansatz einer Energieminimierung, was bei einer einfachen Nutzung als Kompressionsterm in Mehrphasensimulationen jedoch nicht ins Gewicht fällt, sofern das chemische Potential Φ nicht zur Berechnung des Drucksprungs an der Phasengrenze verwendet wird (Gl. 2.35).

Die Erstellung eines adäquaten Kompressionsterms zu gegebener Formfunktion $f(L)$ ist demnach sowohl mit dem Ansatz nach Olsson und Kreiss ([58]) als auch mit Jacqmins Ansatz ([59], [60]) möglich. Trotz der wesentlich weiteren Verbreitung der Methode nach Olsson und Kreiss ([58]), wird hier dennoch ein Kompressionsterm nach dem Phase-Field-Ansatz gewählt, da dieser Kompressionsterm bereits den empfindlichen Parameter κ enthält. Eine fehlerbehaftete Berechnung der Krümmung κ wird in der Literatur häufig als Ursache parasitärer Strömungen angesehen. [92] Ebenso ist bekannt, dass Kompressionsterm und berechnetes Druckfeld durchaus leicht verschiedene Gleichgewichtszustände eines Tropfens anstreben können. Diskutiert wurde dieser Punkt bereits in Unterabschnitt 2.3.3. Ein Kompressionsterm, dessen Gleichgewichtszustand sich bereits über die Krümmung κ berechnet, scheint ein vielversprechender Kandidat zu sein, um die unterschiedlichen Gleichgewichtszustände aus Abbildung 2.4 möglichst ähnlich zu gestalten. Je ähnlicher die beiden Gleichgewichtszustände sind, desto geringer sind die parasitären Strömungen eines liegenden Tropfen bei voller Kompressionsstärke. Im folgenden Abschnitt wird auf dieser Basis eine konkrete Formfunktion $f(L)$ definiert und der zugehörige Kompressionsterm nach dem Phase-Field-Ansatz konstruiert.

3.3.2 Ausarbeitung der Formfunktion $f(L)$ und Konstruktion des zugehörigen Kompressionsterms

Neben der Existenz eines zugehörigen Kompressionsterms sind zu Beginn dieses Abschnittes weitere Anforderungen an $f(L)$ gestellt worden, aus denen eine konkrete Formfunktion hergeleitet werden soll. Im Mittelpunkt stehen dabei die analytische

Integrierbarkeit und das Grenzwertverhalten für große Werte von L. Funktionen, wie der beispielhaft diskutierte arcus tangens, lassen sich zwar drei mal analytisch integrieren, weisen jedoch eine extrem langsame Annäherung an die Grenzwerte von $c = \pm 1$ auf. Prädestiniert hierfür sind hingegen Exponentialfunktionen, die sich sehr schnell aber nur einseitig an einem Grenzwert annähern. Um dieses Verhalten nutzen zu können, wird $f(L)$ stückweise definiert. Dies ist zulässig, solange die analytische Integrierbarkeit erhalten bleibt und die Übergangsstellen zur Anwendung von Runge-Kutta-Methoden hoher Ordnung ausreichend oft stetig differenzierbar sind. Zwischen den beiden Exponentialfunktionen, die das gewünschte Grenzwertverhalten für große Werte von $|L|$ einbringen, muss demnach eine Zwischenfunktion eingefügt werden, die sich leicht an die vorliegenden Stetigkeitsbedingungen anpasst und trotzdem analytisch integrierbar ist. Hierfür werden Polynome in Betracht gezogen. Zusätzlich werden in den Exponentialfunktionen weitere Variationsparameter (α, β) eingeführt, um zusätzliche Freiheitsgrade zu gewinnen und die Polynomordnung m möglichst niedrig zu halten. Die aus numerischen Gründen um $L = 0$ herum punktsymmetrisch[4] und noch weiter auszuarbeitende Formfunktion $f(L)$ ist daher zunächst in Gl. 3.21 definiert.

$$f(L) = \begin{cases} \operatorname{sign}(L) \cdot (1 - \exp(\alpha \cdot |L| + \beta)) & \text{für } |L| \geq L_{inner} \\ p(L) = \sum_i a_i \cdot L^i & \text{für } |L| < L_{inner} \end{cases} \tag{3.21}$$

Die Werte L_{inner}, α, β und der Polynomgrad m mit den zugehörigen Polynomkoeffizienten a_i ermöglichen nun die Definition der Stetigkeitsbedingungen für den Funktionswert und die ersten fünf Ableitungen an der Übergangsstelle L_{inner}. An dieser Stelle wird eine Stetigkeit bis zur fünften Ableitung gefordert, um in der Wahl des Runge-Kutta-Verfahrens noch nicht eingeschränkt zu sein.[5] Zunächst werden α, β und L_{inner} als bekannt angesehen und dadurch die ersten vier Ableitungen inklusive Funktionswert durch die Koeffizienten a_i des Polynoms neunten Grades abgestimmt. Anschließend erfolgt die Bestimmung von β zum Abgleich der fünften Ableitung. Die mathematische Herleitung des resultierenden Gleichungssystems ist in Anhang A.2 dargestellt. Wie in Unterabschnitt 3.3.1 bereits am Beispiel des arcus tangens ausgearbeitet wurde, erfolgt die Konstruktion des zugehörigen Kompressionsterms durch Variation von $\Psi'(c)$ nach Gl. 3.22.

$$\Psi'(c) = f''(f^{-1}(c)) \tag{3.22}$$

Die hierfür notwendige Inverse f^{-1} ist, wie $f(L)$ selbst, nur stückweise definiert und kann nur im äußeren Bereich explizit angegeben werden. Im inneren Bereich wird eine Iteration mit Hilfe des Newton-Verfahrens verwendet.[6]

$$f^{-1}(c) = \begin{cases} \operatorname{sign}(c) \cdot \frac{\ln(1-|c|)-\beta}{\alpha} & \text{für } |c| \geq f(L_{inner}) \\ L_{n+1} = L_n - \frac{p(L_n)-c}{p'(L_n)} & \text{für } |c| < f(L_{inner}) \end{cases} \tag{3.23}$$

[4]Eine Punktsymmetrie um $L = 0$ herum erlaubt eine übersichtlichere Definition der Gesamtmethode und erlaubt eine effizientere Auswertung des Polynoms über das Horner-Schema.

[5]Je höher die Ordnung des gewählten Runge-Kutta-Verfahrens ist, desto mehr Ableitungen müssen an der Übergangsstelle L_{inner} übereinstimmen, damit die zusätzlichen Funktionsauswertungen auch tatsächlich eine Erhöhung der Konvergenzordnung mit sich bringen.

[6]Eine explizite Angabe der Umkehrfunktion ist nur bei Polynomen bis zum Grad $m = 4$ möglich und erfordert zudem mehrfaches Radizieren. Das Newton-Verfahren ist durch Nutzung des Horner-Schemas für die Auswertungen von $p(L)$ und $p'(L)$ schnell und konvergiert mit dem Startwert $L = 0$ sicher für positive, wie negative c mit $|c| < f(L_{inner})$

Neben der Verwendung in $\Psi'(c)$ wird L auch zur genaueren Berechnung von $\vec{\nabla}c$ und Δc verwendet (siehe Gl. 3.5 und Gl. 3.6). Der verwendete Kompressionsterm hat dadurch die in Gl. 3.24 und Gl. 3.25 gegebene Gestalt

$$\frac{\partial c}{\partial \tau} = D \cdot \Delta\Phi \tag{3.24}$$

$$\begin{aligned}\Phi &= \frac{\Psi'(c)}{\varepsilon} - \varepsilon \cdot (\Delta c + \kappa \cdot |\vec{\nabla}c|) \\ &= \frac{\partial^2 f}{\partial L^2}(L) \cdot \left(\frac{1}{\varepsilon} - \varepsilon \cdot |\vec{\nabla}L|^2\right) - \varepsilon \cdot \frac{\partial f}{\partial L}(L) \cdot \left(\Delta L + \kappa \cdot |\vec{\nabla}L|\right)\end{aligned} \tag{3.25}$$

Eine Berechnung von L ist allerdings nur in Zellen mit $|c| < 1$ möglich. Werte von $|c| \geq 1$ sind zwar unphysikalisch, können aber bei großen Courant-Zahlen Co durch die Advektion gelegentlich auftreten. Darüber hinaus ist eine Berechnung von L auch nicht in allen Zellen sinnvoll, da für Werte $|c| \simeq 1$ die direkte Berechnung von $\vec{\nabla}c$ und Δc bereits sehr genau ist und ein Umweg über L nur unnötige Rechenzeit beanspruchen würde. Daher wird ein Grenzwert L_{outer} definiert, ab dem die Transformierte L nicht mehr berechnet oder genutzt wird. Der genaue Wert für L_{outer} wird im späteren Verlauf noch genauer bestimmt (siehe Abschnitt 3.5).

3.4 Algorithmus zur analytischen Volumenintegration der Formfunktion $f(L)$ und Verfahren zur impliziten Bestimmung von L

Qian, Xie und Xiao nutzten in ihren Methoden die bekannte Formfunktion eines tangens hyperbolicus und berechneten die Transformierte L, sodass in jeder Zelle der Grenzfläche Gl. 3.26 erfüllt. ([94], [99], [100], [63], [62], [61])

$$\bar{c} = \frac{1}{V} \cdot \iiint c \, \mathrm{d}V \; = \frac{1}{V} \cdot \iiint \frac{1}{2} \cdot (1 + \tanh(L)) \; \mathrm{d}V \tag{3.26}$$

Dabei wurde L je nach vorliegender Arbeit als lineare Funktion oder durch ein Polynom höherer Ordnung beschrieben. [99] Die Berechnung des Volumenintegrals in Gl. 3.26 erfordert die Anwendung numerischer Integrationsverfahren. Diese erfolgen nach linearer Transformation des Integrationsgebietes auf eine formentsprechende Einheitszelle, was die Anwendung auf Zelltypen limitiert, für die eine solche Transformation bekannt ist ([62], [63]). Die numerischen Integrationsverfahren sind Gauss'sche Quadraturformeln optimaler Ordnung, was allerdings bereits bei 3 Stützpunkten pro Raumdimension zu 27 Funktionsauswertungen pro Zelle führt. Da der Wert von L im Zellzentrum zunächst nur geschätzt werden kann und daraufhin iterativ verbessert werden muss, ist die Auswertung des Volumenintegrals mehrmals erforderlich. Über ein Newton-Verfahren wird L im Zellzentrum so variiert, dass der nach Gl. 3.26 berechnete Wert mit dem in der Zelle vorliegendem Wert von $\bar{c}$ übereinstimmt. Hierbei halten alle genannten Autoren die Ableitungen von L über die Iterationen hinweg konstant und variieren lediglich das Absolutglied des beschreibenden Polynoms. In diesem Punkt liegt bereits ein gravierender Unterschied zu der in dieser Arbeit vorgestellten Methode.

Jeder Iterationsschritt des Newton-Verfahrens benötigt dabei eine Auswertung des Volumenintegrals und eine Abschätzung der Änderungsrate $\partial\bar{c}/\partial L$. Eine Verringerung der hierfür notwendigen Funktionsauswertungen und eine Beschleunigung des Gesamtverfahrens wäre durch analytische Integration der Formfunktion möglich. Xie et al. verfolgten diesen Ansatz bereits ([62]), indem sie den tangens hyperbolicus zumindest in Richtung des steilsten Gradienten von L analytisch integrierten. Die restlichen Raumdimensionen werden numerisch integriert, da die höheren Stammfunktionen nicht mehr adäquat auszuwerten sind. In diesem Abschnitt soll daher erläutert werden, wie die Auswertung des Volumenintegrals in Gl. 3.26 beschleunigt und gleichzeitig auf beliebige Zellgeometrien erweitert werden kann. Hierzu wird die analytische Integrierbarkeit der neuen Grenzflächenformfunktion $f(L)$ in Kombination mit dem Integralsatz von Gauss (Gl. 3.27) ausgenutzt.

$$\iiint \vec{\nabla} * \vec{g} \, \mathrm{d}V = \oiint_{\partial V} \vec{g} \, \mathrm{d}\vec{A} \tag{3.27}$$

Wird L in jeder Zelle linear angenähert (Gl. 3.28), lässt sich zu jeder integrierbaren Funktion $f(L)$ nach Gl. 3.29 ein Vektorpotential $\vec{g}$ definieren, dessen Divergenz wiederum $f(L)$ entspricht. Die Übereinstimmung von $\vec{\nabla} * \vec{g}_1(L)$ und $f(L)$ wird in Gl. 3.30 gezeigt. $F_1(L)$ ist dabei Stammfunktion zu $f(L)$.

$$\vec{\nabla} L = \begin{pmatrix} L_x \\ L_y \\ L_z \end{pmatrix} = \textbf{const.}, \text{ da } L \text{ linear} \tag{3.28}$$

$$\vec{g}_1(L) = F_1(L) \cdot \frac{\vec{\nabla} L}{\left|\vec{\nabla} L\right|^2} \tag{3.29}$$

$$\vec{\nabla} * \vec{g}_1(L) = \left(\vec{\nabla} F_1(L)\right) * \frac{\vec{\nabla} L}{\left|\vec{\nabla} L\right|^2} = \left(f(L) \cdot \vec{\nabla} L\right) * \frac{\vec{\nabla} L}{\left|\vec{\nabla} L\right|^2} = f(L) \tag{3.30}$$

Eingesetzt in Gl. 3.27 lässt sich dadurch das Volumenintegral auf die Berechnung von Flächenintegralen über das Vektorpotentials $\vec{g}_1$ auf den Zellwände zurückführen. Voraussetzung hierfür ist ein linearer Verlauf von L innerhalb der Zelle und ein nicht verschwindender Gradient $|\vec{\nabla} L| \neq 0$.

Die Flächenintegrale über die einzelnen Zellwände können jeweils getrennt voneinander berechnet werden. Im Allgemeinen ist jede Zellwand eine ebene Fläche, wodurch der Normalenvektor $\vec{n}_{face}$ konstant ist. Das Produkt aus Vektorpotential $\vec{g}_1$ und infinitesimalem Flächenvektor $\mathrm{d}\vec{A}_i$ kann daher in einen konstanten Vorfaktor und ein Residuum innerhalb des Integrals umgeformt werden (Gl. 3.31).

$$\iiint f(L) \, \mathrm{d}V = \oiint_{\partial V} \vec{g} \, \mathrm{d}\vec{A} = \sum_{face,i} \left(\frac{\vec{\nabla} L * \vec{n}_{face,i}}{\left|\vec{\nabla} L\right|^2} \cdot \iint F_1(L) \, \mathrm{d}A_i \right) \tag{3.31}$$

Die Flächenintegrale innerhalb der Summe können entweder numerisch oder wiederum analytisch berechnet werden. Welche Methode Anwendung findet, wird im weiteren Verlauf noch erläutert werden. Zunächst werden die Integralwerte als gegeben angesehen und die Vorfaktoren der Flächenintegrale genauer betrachtet. Die Vorfaktoren stellen Wichtungsfaktoren dar, welche in ihrer Größenordnung stark

schwanken und sowohl positiv als auch negativ sein können. Der Betrag des Gradienten $\vec{\nabla}L$ sollte stets eine Größenordnung von $1/\varepsilon$ besitzen, kann bei starker Verformung der Grenzfläche, zu schwacher Kompressionskraft oder in Zellen mit $|c| \simeq 1$ auch Werte in einer Größenordnung bis hinunter zu $|\vec{\nabla}L| \simeq \mathcal{O}(0)$ annehmen. Für besonders kleine Werte von $|\vec{\nabla}L|$ entstehen sehr große Wichtungsfaktoren, die bei der Summenauswertung durch die begrenzte Zahl an gespeicherten Nachkommastellen zu Auslöschungseffekten und Ungenauigkeiten führen können. Für den Fall $|\vec{\nabla}L| = 0$ ist die Anwendung der Umformung erst gar nicht möglich, da das Vektorpotential $\vec{g}_1(L)$ nicht definiert ist. Demnach muss im Vorfeld abgeschätzt werden, in welchen Fällen Gl. 3.31 und wann stattdessen ein einfaches numerisches Integrationsverfahren (in diesem Fall die Mittelpunktsregel Gl. 2.11) angewendet werden sollte. Für die Anwendung der Mittelpunktsregel auf eine würfelförmige Zelle wurde der Fehlerschätzer e in Gl. 3.32 hergeleitet.

$$e_{cell,num.} = \left| \iiint f(L)\ \mathrm{d}V - f(L_{center}) \cdot V \right| \simeq \frac{V}{6} \cdot \frac{\partial^2 f}{\partial L^2}(\xi) \cdot \left|\vec{\nabla}L\right|^2 \cdot \left(\frac{V}{8}\right)^{\frac{2}{3}} \tag{3.32}$$

Die Kantenlänge Δx des Würfels, die typischerweise bei Abschätzungen durch Taylor-Reihen-Entwicklung auftritt, wurde durch das Volumen V der Zelle ausgedrückt. Wird dieser Fehlerschätzer auf nicht würfelförmige Zellen angewendet, entstehen Ungenauigkeiten in der Schätzung des Fehlers, die jedoch vernachlässigbar klein sind. Dies wurde im Vergleich zum Würfel auf einem volumengleichem Parallelepiped und einer Prismengeometrie verifiziert. Eine genaue Betrachtung ist in Anhang C.1 enthalten. Der Gesamtfehler $e_{cell,anal.}$ durch Anwendung von Gl. 3.31 wird, wie die Bildung der Summe $\sum_i$ über die Flächenintegrale auch, sukzessiv berechnet. Zunächst starten die Summe $\sum_i$ und der Fehlerschätzer $e_{cell,anal.}$ mit einem Wert von 0, welcher als exakt angesehen wird. Nach Hinzufügen jedes weiteren Summanden wird der nun vorliegende Fehler $e_{cell,anal.}$ nach Gl. 3.33 aktualisiert.

$$e_{cell,anal.} = \max\left(e_{cell,anal.}\ ;\ 10^{-16} \cdot \left| \sum_{face,i} \right|\ ;\ \left| \frac{\vec{\nabla}L * \vec{n}_{face,i}}{\left|\vec{\nabla}L\right|^2} \cdot e_{face,i} \right| \right) \tag{3.33}$$

In der Bildung des Maximalwertes sorgt der erste Eintrag dafür, dass der Fehlerschätzer monoton steigt und nicht durch Hinzufügen eines weiteren, sehr genau berechneten Summanden, sinken kann. Der zweite Eintrag entspricht dem Phänomen der numerischen Auslöschung. Der aktuelle Summenwert $|\sum_i|$ wird stets nur mit 16 gültigen Stellen gespeichert, wodurch ein Fehler in der Größenordnung von $10^{-16} \cdot |\sum_i|$ entsteht, der auch dann noch bestehen bleibt, wenn die Größenordnung der Gesamtsumme durch Hinzufügen weiterer negativer Summanden wieder abnimmt.[7] Der letzte Beitrag wird hier mit $e_{face,i}$ bezeichnet und entspricht dem Fehler, welcher dem Wert des gerade hinzuaddierten Flächenintegrals $\iint F_1(L)\ \mathrm{d}A_i$ anhaftet. Er ist noch mit dem Wichtungsfaktor des zugehörigen Flächenintegrals zu versehen, da auch das Flächenintegral selber gewichtet in die Gesamtsumme eingeht. Der endgültige Wert von $e_{cell,anal.}$ wird mit $e_{cell,num.}$ verglichen und zur Volumenintegration das Verfahren mit kleinerem Fehler e angewendet.

[7] Beispiel: Hat der Summenwert $\sum_i$ erst einmal einen Wert von 10^6 erreicht, ist er bei lediglich 16 gespeicherten Stellen mit einem Fehler von 10^{-10} behaftet. Auch wenn der Gesamtsummenwert im späteren Verlauf 10^1 betragen sollte, liegt der Fehler dennoch bei 10^{-10}, da Beträge in der Größenordnung von 10^{-11} im vorherigen Verlauf durch Auslöschung verloren gingen.

Für dieses Verfahren müssen jedoch die Fehler $e_{face,i}$ bekannt sein, welche den Flächenintegralen $\iint F_1(L)\ \mathrm{d}A_i$ anhaften. Auf die Integralauswertung und den dadurch entstehenden Fehler wird nun genauer eingegangen. Das Flächenintegral kann, wie bei der Volumenintegration auch, numerisch oder analytisch berechnet werden. Es sind allerdings nicht für alle Flächengeometrien numerische Integrationsverfahren hoher Ordnung mit akzeptablen Aufwand bekannt.[8] Es wird daher nur die Anwendung der universell einsetzbaren Mittelpunktsregel mit der analytischen Integration verglichen und das Verfahren mit dem kleineren Fehler e_{face} angewendet. Für die Abschätzung des Fehlers bei Anwendung der Mittelpunktsregel wird Gl 3.34 verwendet.

$$e_{face,num.} = \left| \iint F_1(L)\ \mathrm{d}A - F_1(L_{face}) \cdot A \right| \simeq \frac{A}{6} \cdot \frac{\partial^2 F_1(\xi)}{\partial L^2} \cdot \left|\vec{\nabla} L_{face}\right|^2 \cdot \left(\frac{A}{4}\right) \quad (3.34)$$

Die zu integrierende Fläche wurde als flächengleiches Quadrat approximiert und eine Taylor-Reihen-Entwicklung auf $F_1(L)$ angewendet. Auftretende Kantenlängen wurden durch die Fläche A des Quadrates ausgedrückt. Wieder führt die Anwendung des Fehlerschätzers auf nicht quadratische Zellwände zu Ungenauigkeiten, die sich jedoch als vernachlässigbar klein erweisen (siehe Anhang C.1) Zu beachten ist die Verwendung von $\vec{\nabla} L_{face}$, welcher sich nach Gl. 3.35 berechnet.

$$\vec{\nabla} L_{face} = \vec{\nabla} L - \vec{n}_{face} \cdot \left(\vec{n}_{face} * \vec{\nabla} L\right) \quad (3.35)$$

Die analytische Flächenintegration beinhaltet erneut ein Vektorpotential $\vec{g}_2(L)$, dessen Divergenz nun $F_1(L)$ beträgt.[9] $F_2(L)$ ist dabei Stammfunktion zu $F_1(L)$.

$$\vec{g}_2(L) = F_2(L) \cdot \frac{\vec{\nabla} L_{face}}{\left|\vec{\nabla} L_{face}\right|^2} \quad (3.36)$$

$$\vec{\nabla} * \vec{g}_2(L) = \left(\vec{\nabla} F_2(L)\right) * \frac{\vec{\nabla} L_{face}}{\left|\vec{\nabla} L_{face}\right|^2} = \left(F_1(L) \cdot \vec{\nabla} L_{face}\right) * \frac{\vec{\nabla} L_{face}}{\left|\vec{\nabla} L_{face}\right|^2} = F_1(L) \quad (3.37)$$

Analog zur Volumenintegration wird das Flächenintegral über den Gauss'schen Integralsatzes in Integrale niedrigerer Dimension über die Kanten der Zellwand zerlegt.

$$\begin{aligned} \iint F_1(L)\ \mathrm{d}A &= \iint \vec{\nabla} * \vec{g}_2(L)\ \mathrm{d}A = \oint_{\partial A} \vec{g}_2(L) * \mathrm{d}\vec{s} \\ &= \sum_{edge,i} \left(\frac{\vec{\nabla} L_{face} * \vec{n}_{edge,i}}{\left|\vec{\nabla} L_{face}\right|^2} \cdot \int F_2(L)\ \mathrm{d}s_i \right) \end{aligned} \quad (3.38)$$

Analog zu $\vec{n}_{face}$ für das dreidimensionale Zellvolumen stellt hier $\vec{n}_{edge,i}$ den nach außen orientierten Normalenvektor der zweidimensionalen Zellwand dar. Abbildung 3.1 veranschaulicht die Definition der einzelnen Normalenvektoren. Zur Berechnung

[8] Jede ebene Zellwand kann in Dreiecke und Vierecke zerlegt werden, für die ein Verfahren hoher Ordnung existiert. Dies führt jedoch zu einer Vielzahl notwendiger Funktionsauswertungen pro Zellwand, die diese Methode unattraktiv machen.

[9] Mathematisch handelt es sich hierbei um ein zweidimensionales Vektorpotential $\vec{g}_2(L)$ und einen zweidimensionalen Differentialoperator $\vec{\nabla}$, da ein Flächenintegral zu transformieren ist. Die Ergebnisse haben jedoch Entsprechungen im dreidimensionalen Raum (z.B. $\vec{\nabla} L_{face}$), welche direkt in die angegebene Gleichung eingesetzt worden sind.

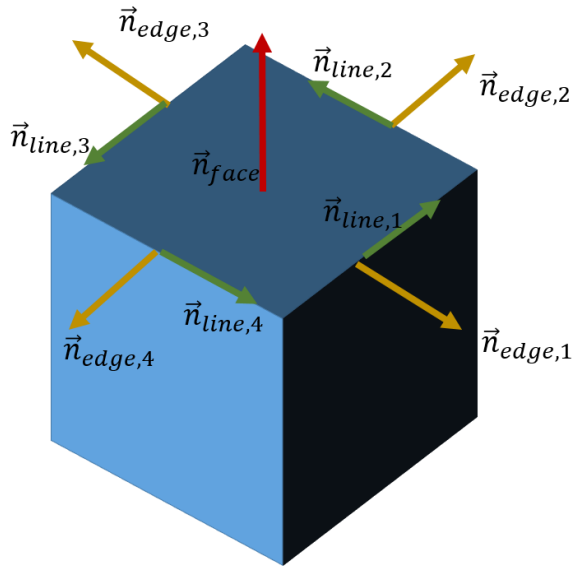


Abbildung 3.1: Definition der Normalenvektoren $\vec{n}_{face}$ (rot), $\vec{n}_{edge,i}$ (gelb) und $\vec{n}_{line,i}$ (grün) am Beispiel einer quaderförmigen Rechenzelle. Während $\vec{n}_{face}$ den nach außen orientierte Normalenvektor der Gesamtzelle darstellt, ist $\vec{n}_{edge;i}$ der nach außen orientierte Normalenvektor der Zellwand.

des Fehlers $e_{face,anal.}$ der analytischen Integration wird das gleiche Verfahren wie bei $e_{cell,anal.}$ angewendet. Mit einem Startwert von 0 für Summe und Fehlerwert wird nach Hinzufügen jedes weiteren Summanden in Gl. 3.38 der Fehlerschätzer nach Gl. 3.39 aktualisiert.

$$e_{face,anal.} = \max\left(e_{face,anal.}\ ;\ 10^{-16}\cdot\left|\sum_i\right|\ ;\ \left|\frac{\vec{\nabla}L_{face}*\vec{n}_{edge,i}}{\left|\vec{\nabla}L_{face}\right|^2}\cdot e_{edge,i}\right|\right) \tag{3.39}$$

Es wird wieder gefordert, dass der Fehlerschätzer durch Hinzufügen weiterer Summanden nicht abfallen kann, dass Auslöschungseffekte durch die begrenzte Anzahl an gespeicherten Stellen abgeschätzt werden und, dass auch die Auswertung der Kantenintegrale $\int F_2(L)\ \mathrm{d}s_i$ einem noch zu wichtenden Fehler $e_{edge,i}$ unterliegen kann. Auch die eindimensionalen Kantenintegrale können analytisch oder numerisch über Gauss'sche Quadraturformeln aus der Literatur ([65], [101]) berechnet werden.[10] An dieser Stelle wird die Quadraturformel mit zwei Stützstellen angewendet, welche Polynome dritten Grades exakt integriert. Nur im Falle verschwindend kleiner Gradienten entlang der Kante ($|\vec{\nabla}L_{edge}| \simeq 10^{-16}$) wird die Mittelpunktsregel angewendet, um die zusätzliche Funktionsauswertung zu sparen. Für die analytische Berechnung ergeben sich Gl. 3.40 und die zugehörige Fehlerabschätzung Gl. 3.41.

$$\int_{x_1}^{x_2} F_2(L)\ \mathrm{d}x = \frac{1}{|\vec{\nabla}L_{edge}|}\cdot\left[F_3(L(x_2)) - F_3(L(x_1))\right] \tag{3.40}$$

$$e_{edge,anal.} = \frac{10^{-16}}{|\vec{\nabla}L_{edge}|}\cdot\max\left(|F_3(L(x_2))|\ ;\ |F_3(L(x_1)|\right) \tag{3.41}$$

[10] Die Mittelpunktsregel gehört im mathematischen Sinn auch zu den Gauss'schen Quadraturformeln mit optimaler Ordnung. Sie integriert mit einer einzelnen Funktionsauswertung ($n = 1$) Polynome vom Grad $2 \cdot n - 1 = 1$ exakt.

In dieser Fehlerabschätzung wird ausgenommen, dass die Funktionsauswertungen $F_3(L(x_i))$ mit einer Ungenauigkeit in der Größenordnung von $10^{-16} \cdot |F_3(L(x_i))|$ behaftet sind, welche noch durch einen Vorfaktor $1/|\vec{\nabla} L_{edge}|$ verstärkt werden können. Im Gegensatz dazu erfolgt die numerische Integration nach Gl. 3.42 mit dem Fehlerschätzer aus Gl. 3.43.

$$\int_{x_1}^{x_2} F_2(L)\ \mathrm{d}x = \frac{x_2 - x_1}{2} \cdot (F_2(L_-) + F_2(L_+)) \tag{3.42}$$

mit

$$L_- = L(x_1) \cdot \frac{3-\sqrt{3}}{6} + L(x_2) \cdot \frac{3+\sqrt{3}}{6}$$
$$L_+ = L(x_1) \cdot \frac{3+\sqrt{3}}{6} + L(x_2) \cdot \frac{3-\sqrt{3}}{6}$$

$$e_{edge,num.} = 10^{-16} \cdot |x_2 - x_1| + \frac{1}{4320} \cdot \frac{\partial^4 F_2(\xi)}{\partial L^4} \cdot |\vec{\nabla} L_{edge}|^4 \cdot |x_2 - x_1|^5 \tag{3.43}$$

Der Gradient $\vec{\nabla} L_{edge}$ wird über Gl. 3.44 berechnet.

$$\vec{\nabla} L_{edge} = \vec{\nabla} L_{face} - \vec{n}_{edge} \cdot \left(\vec{n}_{edge} * \vec{\nabla} L_{face}\right) \tag{3.44}$$

Der rechte Summand der Fehlerabschätzung aus Gl. 3.43 geht dabei auf Assion ([101]) zurück, der eine allgemeinen Fehlerbetrachtung für Gauss'sche Quadraturformeln durchführte. Da die Funktionsauswertungen $F_2(L_-)$ und $F_2(L_+)$ in Gl. 3.42 selbst nur auf 16 Stellen genau sind und danach noch mit der Kantenlänge $|x_2 - x_1|$ multipliziert werden, liegt der Mindestfehler der numerischen Integration folglich bei $10^{-16} \cdot |x_2 - x_1|$ und wird zu Assions Fehlerabschätzung hinzuaddiert. Prinzipiell ist noch die Größenordnung der Funktionsauswertungen $F_2(L_-)$ und $F_2(L_+)$ einzubeziehen, da ein Funktionswert von beispielhaft $F_2(L_+) = 10$ mit 16 gültigen Nachkommastellen effektiv einen Fehler von lediglich 10^{-15} aufweist. Für eine a-priori Fehlerabschätzung wird hier jedoch zur Vermeidung zusätzlicher Funktionsauswertungen in der Intervallmitte die Annahme $F_2(L) \simeq \mathcal{O}\ (1)$ eingeführt.

Nach Berechnung von $e_{edge,anal.}$ und $e_{edge,num.}$ wird für jedes Linienintegral das Verfahren mit dem kleineren Fehler $e_{edge,i}$ angewendet. Dies ermöglicht die Berechnung von e_{face} und hierdurch letztlich e_{cell}. Es wird deutlich, dass die Anwendung des Verfahrens auf der Ebene der Zellpunkte beginnen muss, um darauf folgend die Integrale über die Kanten, die Zellwände und zuletzt den gesamten Zellkörper durchzuführen. Aus Gründen der Darstellbarkeit und Verständlichkeit wurde der Algorithmus jedoch mit der Volumenintegration beginnend dargestellt.

Für eine Implementierung in OpenFOAM 2.1.1 werden zur Beschleunigung der Berechnungen noch einige weitere Annahmen und Vereinfachungen getroffen:

- Die Fehlerabschätzungen der numerischen Integrationsverfahren enthalten noch Funktionsauswertungen von Ableitungen an unbestimmten Stellen ξ.
    - $\frac{\partial^2 F_1}{\partial L^2}(\xi) = \frac{\partial f}{\partial L}(\xi)$ wird an allen Stellen durch $\frac{\partial f}{\partial L}(L_{centre})$ angenähert und nur einmalig berechnet, solange L_{centre} sich nicht verändert.

  - $\frac{\partial^4 F_2}{\partial L^4}(\xi) = \frac{\partial^2 f}{\partial L^2}(\xi)$ wird an allen Stellen durch $\frac{\partial^2 f}{\partial L^2}\left(\max\left(L_{maxf''}\ ;\ |L_{centre}|\right)\right)$ angenähert und nur einmalig berechnet, solange L_{centre} sich nicht verändert. $L_{maxf''}$ bezeichnet dabei den L-Wert, bei dem $f''(L)$ maximal wird. Hierdurch wird verhindert, dass im Bereich um $L_{centre} \simeq 0$ ein zu geringer Wert $\frac{\partial^2 f}{\partial L^2}$ in der Fehlerabschätzung angesetzt wird.

- In der Fehlerabschätzungen für $e_{edge,anal.}$ wird an allen Stellen ein globaler Maximalwert $F_3(L_{outer})$ verwendet, der nicht in jeder Zelle neu berechnet werden muss, sondern zu Beginn der Simulation während der Konstruktion der Profilklasse in OpenFOAM erzeugt und abgespeichert wird.

- Auf ebenen Zellwänden mit geradlinigen Kanten bilden $\vec{n}_{face}$, $\vec{n}_{edge}$ und $\vec{n}_{line}$ stets ein Orthonormalsystem. Dadurch lässt sich $\vec{\nabla}L$ sehr einfach in seine Betragsanteile α, β und γ zerlegen. Diese Anteile werden einmalig berechnet und können sowohl in den Fehlerschätzern selber als auch in der Berechnung der zugehörigen Wichtungsfaktoren der Integrale wiederverwendet werden. $\alpha = \vec{\nabla}L * \vec{n}_{face} \quad \beta = \vec{\nabla}L * \vec{n}_{edge} \quad \gamma = \vec{\nabla}L * \vec{n}_{line}$

- Zu Beginn einer jeden Integralauswertung wird überprüft, ob $\vec{\nabla}L$, $\vec{\nabla}L_{face}$ oder $\vec{\nabla}L_{edge}$ betragsmäßig 0 sind. In diesen Fällen wird der zugehörige Fehler der analytischen Integration direkt zu 10^{17} gesetzt und so automatisch das numerische Integrationsverfahren durchgeführt, was besonders in zweidimensionalen Simulationen Rechenzeit einspart.

Der resultierende Algorithmus ist in Abbildung 3.2 dargestellt. Die Implementierung kann im Anhang B eingesehen werden. Eine Betrachtung des Aufwandes für dieses

Berechne an allen Eckpunkten $\vec{x}_i$ der Zelle $L_i = L_{cellcentre} + \vec{\nabla}L * (\vec{x}_i - \vec{x}_{cellcentre})$ und den zugehörigen Wert von $F_3(L_i)$

↓

Berechne und vergleiche für jede Kante die Fehler von numerischer und analytischer Integration (Gl. 3.43 und Gl. 3.41). Führe das Verfahren mit kleinerem Fehler e_{edge} durch und speichere Integrations- und Fehlerwert der Kannte.

↓

Berechne und vergleiche für jede Zellwand die Fehler von numerischer Integration und Summation über die gewichteten Kantenintegrale (Gl. 3.34 und Gl. 3.39). Führe das Verfahren mit kleinerem Fehler e_{face} durch und speichere Integrations- und Fehlerwert der Zellwand.

↓

Berechne und vergleiche für die Gesamtzelle die Fehler von numerischer Integration und Summation über die gewichteten Zellwandintegrale (Gl. 3.32 und Gl. 3.33). Führe das Verfahren mit kleinerem Fehler e_{cell} durch und speichere Integrations- und Fehlerwert der Zelle.

Abbildung 3.2: Algorithmus zur Volumenintegration einer Funktion $f(L)$ über eine beliebige Zellgeometrie.

Verfahren zeigt, dass im besten Fall lediglich an jedem Eckpunkt einer Zelle der Wert von $F_3(L_i)$ zu berechnen ist. Hinzu kommen die Berechnungen von zwei Fehlerschätzern pro Kante, zwei Fehlerschätzern pro Zellwand und zwei Fehlerabschätzungen für das Volumenintegral zum Schluss. Diese erfordern für ihrer Berechnung jedoch wesentlich weniger Aufwand als die Funktionsauswertungen von $F_3(L_i)$ bzw. $f(L_i)$. Im Falle zu kleiner Gradienten $\vec{\nabla}L$ sind noch Werte von F_2 auf den Kanten und F_1 auf den Zellwänden zu berechnen. Am Beispiel einer häufig verwendeten quaderförmigen Zelle lässt sich der Aufwand der üblicherweise verwendeten numerischen Integration mit 3 Stützstellen pro Raumdimension mit dem Aufwand des hier vorgestellten Algorithmus vergleichen (siehe Tabelle 3.1). Durch den verwendeten Kompressionsterm

Tabelle 3.1: Vergleich des numerischen Aufwandes bei der Anwendung verschiedener Integrationsverfahren.

analytischer Algorithmus	numerische Integration
8 Funktionsauswertungen von $F_3(L_i)$	27 Funktionsauswertungen von $f(L_i)$
24 Fehlerschätzer e_{edge}	
12 Fehlerschätzer e_{face}	
2 Fehlerschätzer e_{cell}	
u.U. 12 Funktionsauswertungen von $F_2(L_i)$	
u.U. 6 Funktionsauswertungen von $F_1(L_i)$	

zum Ausgleich numerischer Diffusionseffekte wird c im Regelfall einen sehr raschen Übergang von -1 nach $+1$ durchlaufen und steile Gradienten mit $|\vec{\nabla}L| \simeq \mathcal{O}(1/\varepsilon)$ liefern. Die zusätzlichen Funktionsauswertungen von $F_2(L_i)$ und $F_1(L_i)$ kommen daher erfahrungsgemäß nur in Bereichen großer L-Werte ($L \simeq L_{outer}$) oder in vereinzelten Zellen vor, in denen $\vec{\nabla}L$ innerhalb der Zelle unglücklich orientiert ist. Neben der beschleunigten Berechnung sticht zudem die Genauigkeit des analytischen Algortihmus hervor. Während für besonders steile Gradienten die numerische Integration einen immer größer werden Verfahrensfehler macht, wird die analytische Integration immer genauer, da der Verfahrensfehler bei analytischer Integration 0 beträgt und auftretende Auslöschungseffekte bei großen Werten von $\vec{\nabla}L$ immer weniger ins Gewicht fallen. Zudem ist der vorgestellte Algorithmus auf beliebige Zellgeometrien anwendbar, während Gauss'sche Quadraturformeln nur bei bestimmten Zellgeomtrien genutzt werden können (Übersicht z.B. in [62] und [63]).[11]

Zu gegebenen Werten von L und $\vec{\nabla}L$ ist demnach der resultierende Wert $\bar{c}$ hinreichend genau berechenbar. L ist nun in jeder Zelle so zu variieren, dass berechneter und vorhandener Wert von $\bar{c}$ möglichst genau übereinstimmen. Die hierfür verwendete Iteration nach Gl. 3.45 ist dem Newton-Verfahren ähnlich.

$$L_{n+1} = L_n - \frac{\frac{1}{V} \cdot \iiint f(L)\ \mathrm{d}V - \bar{c}}{\frac{1}{V} \cdot \frac{\partial}{\partial L_n} \iiint f(L)\ \mathrm{d}V} \tag{3.45}$$

Die bekannte quadratische Konvergenzgeschwindigkeit wird jedoch nur dann erreicht, wenn neben der Volumenintegration eine ebenso genaue Berechnung der im

[11] Wie bei der Auswertung der Flächenintegrale, kann jede Zellgeometrie in Untergeometrien aufgespalten werden, für die wiederum Gauss'sche Quadraturformeln anwendbar sind. Die Anzahl notwendiger Funktionsauswertungen $f(L_i)$ steigt dabei jedoch unverhältnismäßig stark an.

Nenner dargestellten Änderungrate des Integrals erfolgt. Gl. 3.31 verdeutlicht jedoch, dass der berechnete Wert von $\bar{c}$ nicht nur von L_n, sondern auch von $\vec{\nabla}L_n$ abhängig ist. Die Änderung von $\vec{\nabla}L_n$ bezüglich L_n kann jedoch bei der Ableitung nicht in notwendiger Weise berücksichtigt werden. Zusätzlich wird $\vec{\nabla}L_n$ von den L_n-Werten der Nachbarzellen und dem gewählten Diskretisierungsschema zur Bildung von $\vec{\nabla}L_n$ mitbestimmt. Nur unter Berücksichtigung aller Faktoren ließe sich die quadratische Konvergenzgeschwindigkeit beibehalten. Es entstünde ein nichtlineares Gleichungssystem, welches selbst nur durch Linearisierung und Iteration lösbar wäre. Zuletzt müsste auch die Auswertung der Änderungsrate nach dem dargestelltem Algorithmus zur Volumenintegration erfolgen.[12] Die hierfür notwendigen zusätzlichen Funktionsauswertungen sollen vermieden werden, weshalb die quadratische Konvergenzgeschwindigkeit der Iteration aufgegeben wird. Die Änderungsrate von $\bar{c}$ wird stattdessen über die Ableitung von $f(L)$ (Gl. 3.21) bezüglich L approximiert. Dies entspricht einer Auswertung des Integrals im Nenner nach der Mittelpunktsregel. Die dadurch entstehende Iterationsvorschrift ist in Gl. 3.46 dargestellt.

$$L_{n+1} = L_n - \frac{\frac{1}{V} \cdot \iiint f(L_n)\ \mathrm{d}V - \bar{c}}{\frac{\partial}{\partial L_n} f(L_n)} \tag{3.46}$$

Die Implementierung dieser Herangehensweise zeigt in den Rechnungen aus Abschnitt 3.6 eine lineare Konvergenzgeschwindigkeit mit einer Reduktion des Residuums um etwa eine Größenordnung pro Iterationsschritt. Dabei wird die Iteration abgebrochen, sobald 10 Iterationen durchgeführt wurden oder das Residuum der Volumenintegration in jeder Rechenzelle kleiner als 10^{-10} ist. Dieses Residuum kann nicht immer erreicht werden, doch betrifft dies stets nur einzelne Zellen und tritt vermehrt im Übergangsbereich auf, wo sich $|L|$ dem im Unterabschnitt 3.3.2 definierten Grenzwert L_{outer} annähert. Dies ist darauf zurückzuführen, dass für die Auswertung der Volumenintegrale der Wert von $\vec{\nabla}L$ benötigt wird und $\vec{\nabla}L$ wiederum auf die L-Werte der benachbarten Zellen zurückgreift. Es existieren demnach zwangsläufig Zellen, in denen zwar L benötigt wird, aber kein sinnvoller Wert für $\vec{\nabla}L$ definiert werden kann. In diesen Zellen kann die iterative Bestimmung von L nicht angewendet werden. Hier wird stattdessen die direkte Inverse der Formfunktion verwendet ($L = f^{-1}(\bar{c})$), was einen weiteren kleinen Fehler einführt. Zusätzlich ist auch die Annahme eines linearen Verlaufs von L in der Zelle zwar oft gerechtfertigt, aber keineswegs mathematisch korrekt. Beide Annahmen führen kleine Fehler in die Berechnung ein und sind für ein Restresiduum der Iteration mitverantwortlich, wodurch die Angabe einer maximalen Iterationszahl zwingend erforderlich ist. Das Restresiduum wird nach Abschluss der Iteration zwischengespeichert und steht am Ende der Simulation für eine abschließende Fehlerbewertung zur Verfügung.

An dieser Stelle soll darauf hingewiesen werden, dass $\vec{\nabla}L$ in bisherigen Arbeiten über die Iterationen hinweg als konstant angesehen worden ist ([94], [99], [100], [63], [62], [61]). Ein Aktualisierung von $\vec{\nabla}L$ mit den neu iterierten L_{n+1}-Werten wird jedoch für essentiell wichtig erachtet, da eine Berechnung des Gradienten, des Normalenvektors und damit der Krümmung über $\bar{c}$ nach den Überlegungen aus

[12] Betrachtet man vereinfacht nur den komplett analytischen Fall der Berechnung von $\bar{c}$, so gilt $\bar{c} = \sum_i \left(\omega_i \cdot F_3(L_{cellcentre} + \vec{\nabla}L * (\vec{x}_i - \vec{x}_{cellcentre}))\right)$. Demnach wäre die exakte Änderungsrate nach $\partial\bar{c}/\partial L_{cellcentre} = \sum_i \left(\omega_i \cdot F_2(L_{cellcentre} + \vec{\nabla}L * (\vec{x}_i - \vec{x}_{cellcentre}))\right)$ zu berechnen, wodurch weitere i Funktionsauswertungen notwendig würden.

Abschnitt 3.2 mit einem Diskretisierungsfehler der Größenordnung $\mathcal{O}((\Delta x)^2/\varepsilon^2)$ behaftet ist, welcher über die Iteration hinweg beibehalten würde. Eine Aktualisierung des Gradienten während der Iteration ermöglicht eine Bestimmung des Gradienten mit einem Diskretisierungsfehler der Größenordnung $\mathcal{O}((\Delta x)^2/\varepsilon)$, was einen großen Zugewinn an Genauigkeit bedeutet, da in der Regel $\mathcal{O}(\varepsilon) = \mathcal{O}(\Delta x)$ gilt.

Neben der Volumenintegration (vgl. Gl. 2.11) kann dieser Algorithmus auch zur direkten Flächenintegration genutzt werden und so die Flüsse durch die Zellwände (vgl. Gl. 2.12, $\iint (\vec{u} \cdot c)\, \mathrm{d}\vec{A}$)) bei der Advektion von Grenzflächen sehr genau bestimmen. Die Geschwindigkeit $\vec{u}$ wird hierfür über die Zellwand als konstant angenommen und vor das Integral gezogen. Die restliche Integration erfolgt ähnlich zu Gl. 3.38 je nach Fehlerabschätzung analytisch oder numerisch mit $c = f(L)$ und Werten für L_{face} und $\vec{\nabla} L_{face}$, die aus den anliegenden Zellen interpoliert werden.

3.5 Bestimmung des optimalen Parametersatzes

Die bisher hergeleitete Formfunktion $f(L)$ besitzt noch einige Freiheitsgrade. Die freien Parameter α, L_{inner}, L_{outer} und ε sind noch so zu wählen, dass ein rascher Übergang von $c = -1$ nach $c = 1$ erfolgt, eine möglichst geringe parasitäre Strömung U_{Fehl} im Gleichgewichtszustand hervorgerufen wird und eine möglichst große Kompressionskraft D gewählt werden kann. Die Limitierung der maximal zulässigen Kompressionskraft D ist durch die explizite Implementierung von Gl. 3.24 bedingt. Durch die nichtlineare Form der Gleichung gestalten sich implizite Verfahren, die z.B. von Vollmayr-Lee und Rutenberg vorgeschlagen werden ([102]), sehr schwierig und werden hier nicht angewendet. Zur Bestimmung der maximalen Zeitschrittweite Δt bei gegebenem D wird eine lineare Stabilitätsanalyse nach von Neumann durchgeführt, wie auch Vollmayr-Lee und Rutenberg sie ähnlich durchführten ([102]). Hierbei wird das Feld c um eine kleine Störung δc ausgelenkt und auf analytische Weise nachvollzogen, ob sich die Störung δc mit der Zeit abbaut oder anwächst und das Verfahren damit instabil werden lässt. Zeitlich wird in der Analyse das explizite Euler-Verfahren genutzt[13], während alle örtlichen Diskretisierungen nach Zentraldifferenzen erfolgen.

$$\left(c^{n+1} + \delta c^{n+1}\right) = (c^n + \delta c^n) + \Delta t \cdot D \cdot \Delta \left(\frac{\Psi'(c^n + \delta c^n)}{\varepsilon} - \varepsilon \cdot \Delta \left(c^n + \delta c^n\right) \right) \quad (3.47)$$

Subtrahiert man von Gl. 3.47 die nicht gestörte Gleichung, erhält man nach einer Linearisierung von Ψ' eine Beschreibung der zeitlichen Entwicklung des Fehlers.

$$\begin{aligned} \delta c^{n+1} &= \delta c^n + \Delta t \cdot D \cdot \Delta \left(\frac{\Psi'(c^n + \delta c^n) - \Psi'(c^n)}{\varepsilon} - \varepsilon \cdot \Delta \left(\delta c^n\right) \right) \\ &= \delta c^n + \Delta t \cdot D \cdot \Delta \left(\frac{\Psi''(c^n) \cdot \delta c^n + \Psi'''(\xi) \cdot (\delta c^n)^2}{\varepsilon} - \varepsilon \cdot \Delta \left(\delta c^n\right) \right) \end{aligned} \quad (3.48)$$

Für Stabilität darf der numerische Fehler δc in der Zeit nicht anwachsen. Zur Beurteilung wird hierfür die Fourrierreihe $\delta c = \sum \delta \tilde{c} \cdot \exp\left(i \cdot \vec{k} * \vec{x}\right)$ betrachtet. Der Fehler

[13] Der Kompressionsterm wird auf einer separaten Zeitskala betrachtet, wie auch Olsson und Kreiss ([58]) es eingeführt hat. Das für den Kompressionsterm verwendete Euler-Verfahren ist daher nicht mit dem Runge-Kutta-Verfahren zum Transport der Grenzfläche auf der echten Zeitskala zu verwechseln.

wächst genau dann nicht an, wenn dies für alle Koeffizienten $\delta\tilde{c}$ der Fourrierreihe nachgewiesen werden kann. Die Koeffizienten $\delta\tilde{c}$ sind räumlich betrachtet jedoch Konstanten. Die räumliche Entwicklung wird stattdessen durch die Exponentialterme und die Summierung über alle Wellenzahlvektoren $\vec{k}$ beschrieben. Hierdurch kann der Laplace-Operator Δ in diskreter Form direkt auf die Exponentialterme $\exp\left(i \cdot \vec{k} * \vec{x}\right)$ angewendet und durch einen algebraischen Faktor λ_k ersetzt werden, welcher von der Gitterweite in allen Raumrichtungen (Δx, Δy, Δz), der Geometrie der Rechenzelle und den betrachteten Wellenzahlen k_i abhängt. Für die hier beispielhaft betrachteten zweidimensionalen kartesischen Rechengitter ist λ_k in Gl. 3.49 gegeben.

$$\lambda_k = -\frac{2 - 2 \cdot \cos\left(k_1 \cdot \Delta x\right)}{\Delta x^2} - \frac{2 - 2 \cdot \cos\left(k_2 \cdot \Delta y\right)}{\Delta y^2} \leq 0 \tag{3.49}$$

Für einen konkreten Koeffizienten $\delta\tilde{c}$ gilt demnach Gl. 3.50

$$\delta\tilde{c}^{n+1} = \delta\tilde{c}^{n} \cdot \left(1 + \Delta t \cdot D \cdot \lambda_k \cdot \left(\frac{\Psi''(c^n)}{\varepsilon} - \varepsilon \cdot \lambda_k\right)\right) \tag{3.50}$$

Nachzuweisen ist dann, dass unabhängig von den Wellenzahlen k_i keiner der Koeffizienten $\delta\tilde{c}$ zeitlich anwächst.

$$\left|\frac{\delta\tilde{c}^{n+1}}{\delta\tilde{c}^{n}}\right| \leq 1 \tag{3.51}$$

$$\Rightarrow \left|1 + \Delta t \cdot D \cdot \lambda_k \cdot \left(\frac{\Psi''(c^n)}{\varepsilon} - \varepsilon \cdot \lambda_k\right)\right| \leq 1 \tag{3.52}$$

Nach Auflösung der Betragsstriche ($\lambda_k \leq 0$) erhält man Gl. 3.53 als notwendige Bedingung für Stabilität.

$$\frac{2}{\varepsilon \cdot {\lambda_k}^2 - \lambda_k \cdot \frac{\Psi''(c^n)}{\varepsilon}} > \Delta t \cdot D \tag{3.53}$$

Die größte Einschränkung liefert λ_k für $k_1 = \pi/\Delta x$ und $k_2 = \pi/\Delta y$. Zur Veranschaulichung, welchen Einfluss $\Psi''(c^n)$ besitzt, sind für das bisherige tanh-Profil und die neue Formfunktion die Verläufe von $\Psi'(c^n)$ nach Gl. 3.22 berechnet und in Abbildung 3.3 gegenübergestellt. Verwendet wurden exemplarisch die Werte $\alpha = -2$ und $L_{inner} \approx 0,82$. Aus Symmetriegründen des Potentials $\Psi(c)$ ist die Darstellung auf positive c-Werte beschränkt. Die Einhaltung der Stabilitätsbedingung aus Gl. 3.53 wird von der zweiten Ableitung $\Psi''(c)$ des eigentlichen Energiepotentials $\Psi(c)$ mitbestimmt. Dementsprechend ist nach dem größten Steigungswert von $\Psi'(c)$ in Abbildung 3.3 zu suchen, welcher für beide Profile bei $|c| = 1$ zu finden ist. Eine analytische Fortsetzung der beiden Verläufe am Punkt $c = 1$ liefert Gl. 3.54 und Gl. 3.55.

$${\Psi''}_{\tanh}(c) = 6 \cdot c^2 - 2 \rightarrow \max_{c \in [-1;1]} \left({\Psi''}_{\tanh}\right) = 4 \tag{3.54}$$

$${\Psi''}_{f(L),|c|>f(L_{inner})}(c) = \operatorname{sign}(c) \cdot \alpha^2 \rightarrow \max_{c \in [-1;1]} \left({\Psi''}_{f(L)}\right) = \alpha^2 \tag{3.55}$$

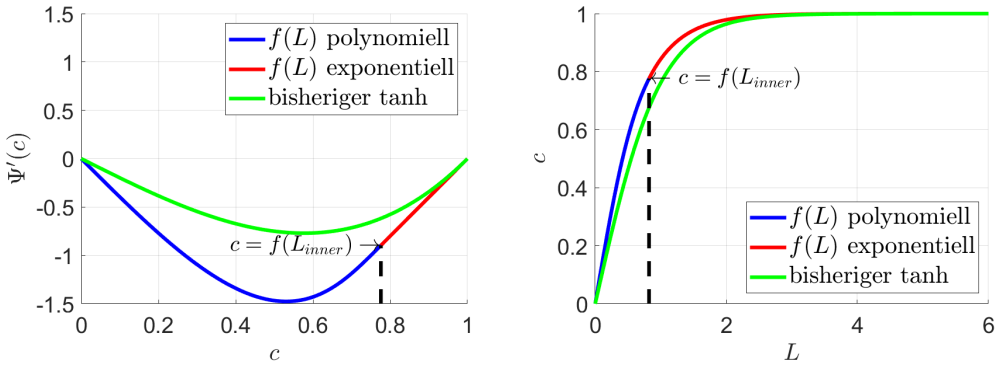


Abbildung 3.3: Vergleich der ersten Ableitungen unterschiedlicher Potentialfunktionen $\Psi(c)$ für die Verwendung in der Phase-Field-Gleichung (Gl. 3.14) (links) und resultierender Profilverlauf (rechts).

Es ist anzumerken, dass Ψ'' für die neudefinierte Formfunktion auch bei Werten $|c| > 1$ begrenzt und somit die Stabilität nach Gl.3.53 erhalten bleibt.[14] Das bisherige tanh-Profil konnte dies nicht garantieren. Betrachtet wurden an dieser Stelle lediglich positive Werte von $\Psi''(c)$, allerdings kann Gl. 3.53 theoretisch auch durch negative Werte von $\Psi''(c)$ erfüllt werden. Im polynomiellen Bereich (blau) wird $\Psi''(c)$ durch die Polynomkoeffizienten a_i bestimmt, welche maßgeblich von L_{inner} abhängen. Besonders kleine L_{inner} können dabei negative Werte für $\Psi''(c)$ erzeugen, die das eben bestimmte Maximum von α^2 betragsmäßig auch übersteigen können. Aus der linearen Stabilitäsanalyse lassen sich durch die Kombination des variablen Laplace-Operators λ_k mit dem variablen Wert von $\Psi''(c)$ jedoch keine konkreten Aussagen ableiten, welche negativen Werte von $\Psi''(c)$ noch zulässig sind. Damit das Stabilitätsgebiet des Verfahrens nicht durch unerwartete Effekte eingeschränkt wird, soll $\Psi''(c)$ daher im gesamten Definitionsbereich betragsmäßig auf α^2 beschränkt werden. Der Betrag von $\Psi''(c)$ kann in Gl. 3.53 dann sicher durch α^2 abgeschätzt werden. Hierfür wird L_{inner} so gewählt, dass der Wert von $\Psi''(c)$ nach unten durch $-\alpha^2$ begrenzt ist. Das Risiko eines instabilen Verfahrens wird auf diese Weise vermieden. Der in Abbildung 3.3 dargestellte Verlauf von $\Psi'(c)$ verdeutlicht, dass der minimale Wert von $\Psi''(c)$ bei $c = 0$ erreicht wird. Aus Gl. 3.22 und einer Linearisierung von $p^{-1}(c)$ um den Punkt $c = 0$ ergibt sich folgender Zusammenhang:

$$\Psi''(c) = \frac{d}{dc} f''(f^{-1}(c)) = p'''(p^{-1}(c)) \cdot (p^{-1})'(c) \overset{c=0}{=} 6 \cdot \frac{a_3}{a_1} \overset{!}{=} -\alpha^2 \tag{3.56}$$

Maßgebend für die Erfüllung der gestellten Forderung sind demnach die Werte a_1 und a_3. Die Abhängigkeit der Polynomkoeffizienten a_i von L_{inner} (siehe Anhang A, Gl. A.2) lässt sich jedoch nicht praktikabel explizit angeben, weshalb L_{inner} zur Erfüllung von Gl. 3.56 iterativ durch ein Newton-Verfahrens bestimmt wird. Der Parameter $\beta_{opt.}$ passt sich entsprechend an (siehe Anhang A, Gl. A.4). Theoretisch kann L_{inner} auch kleiner gewählt werden. Dies würde kleinere Werte für $\beta_{opt.}$ herbeiführen

[14] Für Werte $|c| > 1$ kann es mehrere Gründe geben. Im Allgemeinem strebt der Ansatz der Energieminimierung aus der Phase-Field-Gleichung nur nach einer Minimierung der globalen Gesamtenergie F (Gl. 2.29). Dies kann durchaus eine lokale Übersättigung von c beinhalten. Darüber hinaus kann die Anwendung von Diskretisierungsschemata, die nicht dem TVD-Ansatz folgen (total variation diminisching), ebenfalls zu einem unphysikalischen Überschießen von c führen.

und somit die gewünschte Annäherung von $f(L) = \text{sign}(L) \cdot (1 - \exp(\alpha \cdot |L| + \beta_{opt.}))$ an die Grenzwerte ± 1 beschleunigen. Dem steht jedoch gegenüber, dass hierdurch die höheren Ableitungen von $f(L)$ stark ansteigen würden. Neben der Gefahr eines numerisch instabilen Verfahrens bedingt dies auch einen größeren zeitlichen Diskretisierungsfehler, da die Abbruchfehler der verwendeten Runge-Kutta-Verfahren direkt mit den Werten der höheren Ableitungen von $f(L)$ skalieren. Der Anspruch eines kleinen zeitlichen Diskretisierungsfehler steht demnach im Widerspruch zu einer möglichst raschen Annäherung von $f(L)$ an den Grenzwert in der Bulkphase. Die Festsetzung von L_{inner} durch die Erfüllung von Gl. 3.56 dient also nicht nur einer vorsichtigen Sicherstellung der numerischen Stabilität, sondern stellt auch einen Kompromiss zwischen Genauigkeit und Geschwindigkeit des Verfahrens dar. Optimierungspotential ist demnach noch vorhanden, wird jedoch eher gering eingeschätzt und ist zudem vom jeweiligen Anwendungsfall abhängig.

Da die Stabilitätsanalyse nach von Neumann stets nur ein notwendiges Kriterium liefert, wird zunächst die Gültigkeit von Gl. 3.53 verifiziert. Gegenstand der Untersuchung ist ein liegender Tropfen mit einem Radius von 0,6 mm. Aus Symmetriegründen muss lediglich ein Viertel betrachtet werden, weshalb ein zweidimensionales, kartesisches Rechengitter mit den Abmessungen von 0,8 mm gewählt wird (siehe Abbildung 3.4). Eine Kopplung mit der Kräftebilanz erfolgt an dieser Stelle

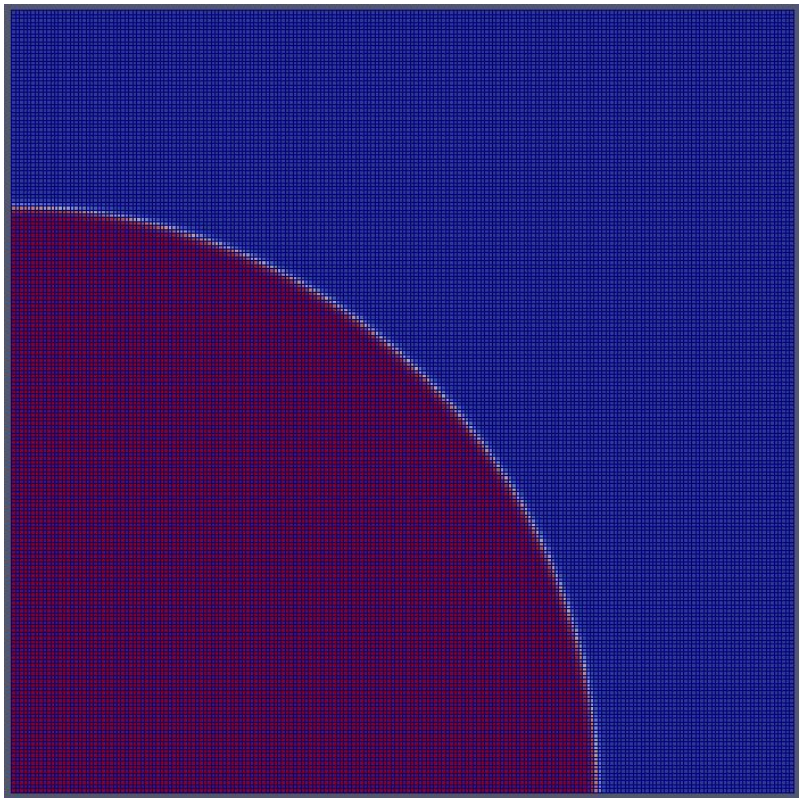


Abbildung 3.4: Betrachteter Vierteltropfen (rot) mit Radius 0,6 mm innerhalb der kontinuierlichen Phase (blau) auf einem kartesischem Rechengitter mit den Abmessungen von 0,8 mm bei einer Gitterweite von $4 \cdot 10^{-6}$ m.

noch nicht und es wird lediglich der Kompressionsterm nach Gl. 2.32 berechnet und auf numerische Stabilität geprüft. Die getesteten Parametersätze ergeben sich aus den Permutationen der in Tabelle 3.2 gegebenen Parameterwerte. Zu jeder Permua-

Tabelle 3.2: Permutationsparameter zur Verifizierung von Gl. 3.53.

$\Delta x = \Delta y$	α	Δt	$\%D_{max.}$	L_{outer}	ε
$4{,}0 \cdot 10^{-6}\,\mathrm{m}$	$-1,75$	$2 \cdot 10^{-6}\,\mathrm{s}$	$99,5\%$	6	$1.5 \cdot \Delta x$
$3{,}2 \cdot 10^{-6}\,\mathrm{m}$	$-2,00$	$1 \cdot 10^{-6}\,\mathrm{s}$	$100,5\%$		
$2{,}0 \cdot 10^{-6}\,\mathrm{m}$	$-2,25$	$5 \cdot 10^{-7}\,\mathrm{s}$			

tation aus Δx, α und Δt wird die maximal zulässige Kompressionskraft $D_{max.}$ nach Gl. 3.53 berechnet und um 0.5% gesenkt, was ein stabiles Verfahren ergeben sollte, oder um 0.5% angehoben, was eine anwachsende Instabilität erzeugen sollte. Die Werte für L_{outer} und ε werden zunächst willkürlich gesetzt, nicht variiert, und erst im späteren Verlauf genauer betrachtet. Hieraus ergeben sich $n = 3 \cdot 3 \cdot 3 \cdot 2 \cdot 1 \cdot 1 = 54$ betrachtete Parametervariationen. Als Indikator für numerische Stabilität wird die Differenz $|\Phi_{max} - \Phi_{min}|$ im gesamtem Rechengebiet betrachtet und über den Zeitbereich 0,1 s bis 0,12 s gemittelt. Ein stetiges Anwachsen der Potentialdifferenz signalisiert ein instabiles Verfahren, während ein stetiges Abklingen der Potentialdifferenz auf ein stabiles Verfahren hindeutet. Tatsächlich liegen die berechneten Mittelwerte für $D = 0,995 \cdot D_{max}$ zwischen 1,1 $\frac{1}{\mathrm{m}}$ und 7,8 $\frac{1}{\mathrm{m}}$, während die Nutzung von $D = 1,005 \cdot D_{max}$ auf gemittelte Potentialdifferenzen von $2{,}5 \cdot 10^{6}\,\frac{1}{\mathrm{m}}$ bis $1{,}3 \cdot 10^{7}\,\frac{1}{\mathrm{m}}$ führt.[15] Eine Konvergenz der Potentialdifferenz gegen den Wert 0 kann allerdings auch bei Einhaltung des Stabilitätskriteriums nicht erwartet werden, da nahe des Grenzbereiches ($L \approx L_{outer}$) eine Unstetigkeit in der Berechnung des Potential Φ entsteht. Für $|c| > f(L_{outer})$ werden die für Φ notwendigen räumlichen Ableitungen über c selbst gebildet, während für $|c| < f(L_{outer})$ alle räumlichen Ableitungen und der Wert von Ψ über L gebildet werden.

Um Rechenzeit einzusparen und den Modellfehler einer verschmierten Grenzfläche so gering wie möglich zu halten, ist ein möglichst geringer Wert von ε wünschenswert. Nach Gl. 3.53 wäre ein beliebig kleiner Wert von ε möglich, was unrealistisch ist und auf ein weiteres limitierendes Kriterium hindeutet. Eine stetige Reduzierung von ε unter Einhaltung von Gl. 3.53 durch Anpassung von D, aber sonst konstanten Bedingungen, deutet in Modellrechnungen darauf hin, dass Gl. 3.57 ebenfalls erfüllt sein muss, um ein stabiles Gesamtverfahren zu erhalten.

$$\left| \frac{\alpha \cdot \Delta x}{\varepsilon} \right| \leq 1,6 \tag{3.57}$$

Mögliche Ursache dieser Limitierung kann die starke Nichtlinearität des verwendeten Modells sein. Eine mathematische Begründung dieser Limitierung konnte im Rahmen dieser Arbeit nicht ermittelt werden. Stattdessen soll das ergänzende Kriterium mit Hilfe einer umfassenden Variationsstudie am bereits beschriebenen Vierteltropfen (siehe Abbildung 3.4) im relevanten Parameterraum verifiziert werden. Als Variationsparameter werden hierfür die Gitterweite Δx, die relative Grenzschichtdicke

[15] Die Einheiten eines chemischen Potentials sind physikalisch eigentlich J/m^3. Die Definition nach Gl. 3.25 erzwingt jedoch intern in OpenFOAM die Einheit $1/m$. Die Kompressionsstärke wird demnach zum Erhalt der Einheiten innerhalb von OpenFOAM in m^3/s gegeben. Da die Phase-Field-Gleichung jedoch lediglich als ebenfalls unphysikalischer Kompressionsterm auf einer nicht realen Pseudozeitskala genutzt wird, kann über diese Unstimmigkeit hinweggesehen werden.

$\varepsilon/\Delta x$ und der Exponentialparameter α gewählt. Zu jeder möglichen Permutation aus Δx und $\varepsilon/\Delta x$ wird der nach Gl. 3.57 noch zulässige Wert von α berechnet und anschließend um einen bestimmten Betrag $\delta\alpha$ variiert. Eine Übersicht der betrachteten Permutationsparameter gibt Tabelle 3.3. Das Kriterium zur Stabilätsbewertung ist

Tabelle 3.3: Permutationsparameter zur Verifizierung von Gl. 3.57.

$\Delta x = \Delta y$	$\varepsilon/\Delta x$	$\delta\alpha$	$\%D_{max.}$	L_{outer}	Δt
$4{,}0 \cdot 10^{-6}$ m	$1,00$	$-0,4$	$99,5\%$	10	$1 \cdot 10^{-6}$ s
$3{,}2 \cdot 10^{-6}$ m	$1,05$	$-0,3$			
$2{,}0 \cdot 10^{-6}$ m	$1,10$	$-0,2$			
	$1,15$	$-0,1$			
	$1,20$	$0,0$			
	$1,25$	$0,1$			
		$0,2$			
		$0,3$			
		$0,4$			

erneut die maximale Potentialdifferenz, welche nun über den Zeitraum 0,2 s bis 0,3 s gemittelt und verglichen wird. Um eine eindeutigere Abgrenzung zwischen stabilen und instabilen Parameterpermutationen zu erzielen, wird für diese Fallbetrachtung $L_{outer} = 10$ gewählt, da die angesprochene Unstetigkeit bei der Berechnung von Φ mit steigendem Werten von L_{outer} kleiner und irgendwann vernachlässigbar gering wird. Für jeden der 162 betrachteten Parametersätze wird das Verhältnis $\alpha \cdot \Delta x/\varepsilon$ gebildet und mit der zugehörigen Potentialdifferenz identifiziert. Die grafische Auftragung in Abbildung 3.5 macht durch den roten Balken bei $\alpha \cdot \Delta x/\varepsilon = 1,6$ deutlich, dass ein stabiles Verhalten nur unter gleichzeitiger Einhaltung von Gl. 3.53 und Gl. 3.57 erreicht werden kann. Der stabile Anwendungsbereich des Kompressionsterms

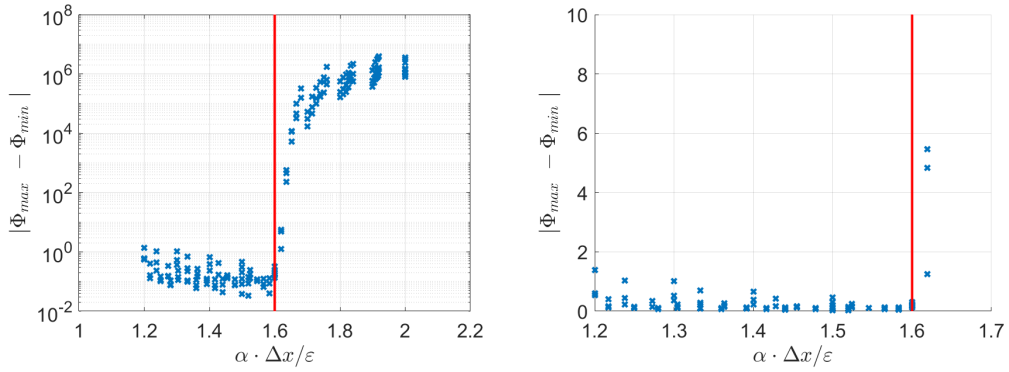


Abbildung 3.5: Über den Zeitraum 0,2 s bis 0,3 s gemittelte Potentialdifferenz $|\Phi_{max} - \Phi_{min}|$ der nach Tabelle 3.3 betrachteten Parameterpermutationen in Abhängigkeit des Verhältnisses $\alpha \cdot \Delta x/\varepsilon$ als logarithmisch skalierte globale Ansicht (links) und in linear dargestellter Detailansicht (rechts) mit Stabilitätsgrenze nach Gl. 3.57 als roter Balken.

ist damit hinreichend genau charakterisiert. Im Folgenden wird mit D_{max} stets eine Kompressionsstärke bezeichnet welche durch sinnvolles Abrunden des nach Gl. 3.53

zulässigen Maximalwertes gerade noch ein stabiles Verhalten ermöglicht. Hiermit kann nun eine Optimierung der Parameter L_{outer} und α hinsichtlich einer möglichst geringen parasitären Restströmung U_{Fehl} erfolgen. Hierzu werden zwei Ursachen für parasitäre Strömungen getrennt voneinander betrachtet. Auf der einen Seite können parasitäre Strömungen durch eine stark fehlerbehaftete Berechnung des Drucksprungs Δp_{cap} hervorgerufen werden. Auf der anderen Seite kann der Kompressionsterm durch numerische Diskretisierungsfehler und einen daraus resultierendem ungenauen Wert für das chemische Potential Φ die Phasengrenzfläche in ihrer Form in einen Zustand versetzen, welcher nicht der analytischen Lösung der Kompressionsgleichung (Gl. 2.32) entspricht. Sind die angestrebten Gleichgewichtszustände von berechnetem Druckfeld und genutztem Kompressionsterm voneinander verschieden, so lässt sich aus den Ergebnissen von Francois et al. ([84]) ableiten, dass Kompressionsterm und Kräftebilanz kontinuierlich gegeneinander arbeiten werden. Dies führt auch am liegenden und unbewegten Tropfen bereits zu parasitären Strömungen, die mit der Zeit nicht abklingen werden. Im Umkehrschluss bedeutet eine vernachlässigbar kleine parasitäre Strömung am liegenden Tropfen jedoch nicht, dass eine gute Übereinstimmung beider Gleichgewichtszustände mit der exakten Lösung vorliegt, sondern es deutet lediglich auf eine gute Übereinstimmung beider Zustände untereinander hin. Beispielhaft verdeutlicht Abbildung 2.4 diesen Sachverhalt. Wird sowohl durch das berechnete Druckfeld als auch durch den Kompressionsterm jeweils Zustand a) oder von beiden jeweils Zustand b) angestrebt, so ist die Stärke der parasitären Strömung am liegenden Tropfen eher gering, doch eine Übereinstimmung mit der korrekten analytischen Lösung muss nicht gegeben sein. Aus diesem Grund wird eine entkoppelte Betrachtungsweise angestrebt. Zunächst wird hierzu die Implementierung der Berechnung des Drucksprungs Δp_{cap} betrachtet, indem die analytische Lösung des Kompressionsterms (Gl. 2.32) für einen liegenden und unbewegten Tropfen mit Radius R im gesamten Rechengebiet vordefiniert wird (siehe Gl. 3.58).

$$c = f(L) \text{ nach Gl. 3.21 mit } L = \frac{R - |\vec{x}|}{\varepsilon} \tag{3.58}$$

Die untersuchten Parameterpermutationen der Phasengrenzfläche sind in Tabelle 3.4 dargestellt. Der Parameter L_{outer} trägt hierbei nicht zum Ergebnis bei, da die analytische Lösung unabhängig von L_{outer} ist. An diesem Tropfen wird das Druck-

Tabelle 3.4: Permutationsparameter zur Charakterisierung des numerischen Fehlers in der Drucksprungberechnung am analytisch vordefinierten Tropfen.

$\Delta x = \Delta y$	α	ε
$4{,}0 \cdot 10^{-6}\,\mathrm{m}$	$1,75$	$(\Delta x \cdot \alpha)/1,6$
$3{,}2 \cdot 10^{-6}\,\mathrm{m}$	$2,00$	
$2{,}0 \cdot 10^{-6}\,\mathrm{m}$	$2,25$	
	$2,50$	

feld und die daraus resultierende parasitäre Strömung ohne konvektiven Transport der Grenzfläche berechnet. Eine genaue Beschreibung der Berechnung von Δp_{cap} erfolgt in Abschnitt 3.8. Verwendet werden wasserähnliche Stoffdaten für beide Phasen ($\rho = 1000\,\mathrm{kg/m^3}$, $\eta = 1 \cdot 10^{-3}\,\mathrm{Pa \cdot s}$), ein Tropfenradius von $R = 0{,}6\,\mathrm{mm}$ und eine

Grenzflächenspannung von $\sigma = 0{,}02\,\mathrm{N/m}$. Als Maß für die Stärke der parasitären Strömung wird der Anteil de Geschwindigkeitsfeldes ermittelt, der in Normalenrichtung durch die Grenzfläche tritt. Besitzt L zwischen zwei benachbarten Zellen einen Vorzeichenwechsel, wird dazwischen ein linearer Verlauf von L angenommen und die Stelle $L = 0$ mit der Position der Grenzfläche identifiziert. An diese Stelle werden daraufhin Geschwindigkeits- und der zur Berechnung des Normalenvektors nötige Gradient von L interpoliert und zur Bildung von U_{fehl} in Gl. 3.59 genutzt.

$$U_{fehl} = \left|\vec{U}_{L=0} * \vec{n}_{L=0}\right| \tag{3.59}$$

Diese Definition wird gewählt, da nur Anteile in Normalenrichtung auch die Position der Grenzfläche beeinflussen und diese Definition darüber hinaus den Vergleich von Tropfen vereinfacht, die entweder durch das Gitter transportierten werden oder stationär im Rechengitter liegen. Verglichen werden im Folgenden stets die über das gesamte Rechengebiet ermittelten Maximalwertwerte von U_{fehl}. Da c und L analytisch vordefiniert wurden, entsteht diese Restströmung einzig und allein durch numerische Fehler in der Berechnung von $\vec{\nabla}L$, $\vec{n}$, κ und Δp_{cap}. Die Konfiguration des Rechengitters entspricht dem in Abbildung 3.4 dargestellten Fall. Für alle betrachteten Fälle liegt die parasitäre Strömung für $\Delta x = 4{,}0 \cdot 10^{-6}\,\mathrm{m}$ bei $U_{fehl} = 4{,}58 \cdot 10^{-5}\,\mathrm{m/s}$, für $\Delta x = 3{,}2 \cdot 10^{-6}\,\mathrm{m}$ bei $U_{Fehl} = 2{,}96 \cdot 10^{-5}\,\mathrm{m/s}$ und für $\Delta x = 2{,}0 \cdot 10^{-6}\,\mathrm{m}$ bei $U_{Fehl} = 1{,}16 \cdot 10^{-5}\,\mathrm{m/s}$. Die parasitäre Strömung nimmt demnach mit zweiter Ordnung mit der Gitterweite ab, ist aber unabhängig von α und ε. Alle Werte für U_{Fehl} liegen in einem akzeptablen Rahmen. Im zweiten Schritt wird nun der numerische Fehler in der Berechnung des Kompressionsterm näher charakterisiert. Hierzu wird erneut ein Vierteltropfen analytisch vordefiniert und durch den Kompressionsterm für 1,0 s der Gleichgewichtszustand von c bei annähernd maximal zulässiger Kompressionsstärke ($D = 99,5\%\ D_{max}$) angestrebt. Die hierbei permutierten Parameter sind in Tabelle 3.5 aufgeführt. Es handelt sich dabei um eine Erweiterung von Tabelle 3.4, in der nun auch der Einfluss von L_{outer} untersucht wird. Im Zeitraum von

Tabelle 3.5: Permutationsparameter zur Charakterisierung des numerischen Fehlers im Kompressionsterm.

$\Delta x = \Delta y$	α	ε	L_{outer}	Δt
$4{,}0 \cdot 10^{-6}\,\mathrm{m}$	$-1,75$	$(\Delta x \cdot \alpha)/1,6$	5	$1 \cdot 10^{-6}\,\mathrm{s}$
$3{,}2 \cdot 10^{-6}\,\mathrm{m}$	$-2,00$		6	
$2{,}0 \cdot 10^{-6}\,\mathrm{m}$	$-2,25$		7	
	$-2,50$		8	
			9	

1,0 s bis 1,1 s wird dann zusätzlich zur Kompressionsgleichung die Kräftebilanz gelöst und das Druckfeld mit resultierendem Geschwindigkeitsfeld berechnet. Erneut wird die Phasengrenzfläche nicht konvektiv transportiert, um eine unabhängige Charakterisierung der Diskretisierungsfehler im Kompressionsterm zu vereinfachen. Es wird darauf hingewiesen, dass im Gegensatz zur vorangegangenen Betrachtung mit vordefiniertem L nun der Wert von L in jedem Zeitschritt iterativ über die Volumenintegration (siehe Abschnitt 3.4) aus dem vorliegendem c-Feld berechnet wird

und sowohl zur Berechnung des Durckfeldes, als auch zur Berechnung des aktuellen Potentials Φ genutzt wird. Die resultierenden Strömungen U_{Fehl} erstrecken sich dabei über mehrere Größenordnungen. Eine Analyse wird dabei speziell hinsichtlich des Parameters L_{outer} durchgeführt. Wie bereits angesprochen bestimmt L_{outer} den Übergangspunkt, ab dem alle numerischen Diskretisierungen über c und nicht mehr über das annähernd lineare L erfolgen. Die Diskretisierungsfehler bei der Verwendung von L werden dabei durchweg konstant bleiben, während die Berechnung von beispielsweise Δc einen immer geringeren numerischen Fehler aufweisen wird, je näher sich c betragsmäßig dem Wert 1 annähert und dadurch selbst einen annähernd linearen Verlauf annimmt. Als geeignetes Kriterium zur Korrelation von U_{Fehl} erweist sich daher der abgeschätzte Diskretisierungsfehler der Mittelung durch Volumenintegration nach der Finite-Volumen-Methode (vgl. Gl. 2.11). Dieser ist im Allgemeinen proportional zur zweiten Ableitung von c, die sich unter Vernachlässigung des Einflusses von κ über die Ableitung von L formulieren lässt (Gl. 3.60).

$$|\bar{c} - c_{center}| \propto \exp\left(\alpha \cdot L_{outer} + \beta\right) \cdot \left(|\vec{\nabla} L| \cdot \alpha \cdot \Delta x\right)^2 \tag{3.60}$$

Dabei lässt sich $|\vec{\nabla} L|$ mit der Größenordnung $1/\varepsilon$ abschätzen, da dies dem analytischem Gleichgewichtszustand des Kompressionsterms entspricht. Nach der Definition von ε aus Tabelle 3.5, lässt sich daraufhin der gesamte Term $\left(|\vec{\nabla} L| \cdot \alpha \cdot \Delta x\right)$ mit dem Zahlenwert $1,6$ ersetzen. Die grafische Auftragung dieser Fehlerabschätzung mit der beobachteten parasitären Restströmung ist in Abbildung 3.6 gegeben. Es ist ein abnehmender Trend der parasitären Strömung mit dem abgeschätzten

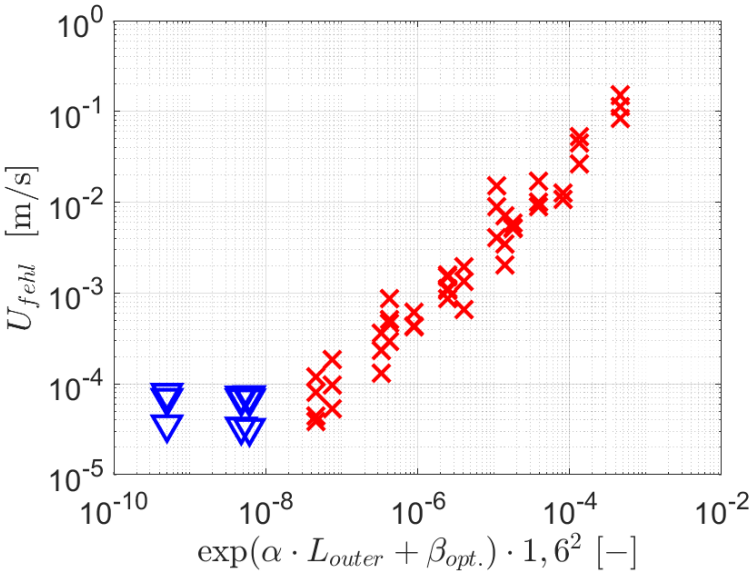


Abbildung 3.6: Korrelation der Fehlerabschätzung bezüglich der Volumenmittelung mit parasitären Strömungen am kompressionsdominierten Tropfen ohne konvektiven Transport der Grenzfläche (rote Kreuze) mit abweichenden unteren Plateauwerten (blaue Dreiecke).

Fehler der Volumenintegration nach Mittelpunktsregel zu erkennen (rote Kreuze), bis bei ungefähr $4 \cdot 10^{-8}$ des Schätzers ein unterer Grenzwert erreicht wird, ab dem die parasitäre Restströmung nicht weiter gesenkt werden kann (blaue Dreiecke). Es wird vermutet, dass ab diesem Punkt andere numerische Fehler dominieren, wie z.B.

die Genauigkeit der Gradientenberechnung oder die maximale Genauigkeit des Algorithmus zur Volumenintegration (siehe Abschnitt 3.4). Dies lässt den Rückschluss zu, dass Kombinationen von hohen L_{outer}-Werten und stark negativen α-Werten den numerischen Fehler bei der Berechnung von Φ hinreichend gut eindämmen können, eine weitere Steigerung von L_{outer} oder α aber keine Verbesserung einbringt und lediglich die notwendige Rechenzeit anwachsen lässt. Daher wird in allen weiteren Untersuchungen der Wert von $4 \cdot 10^{-8}$ als sinnvoller Grenzwert bei der Bestimmung von L_{outer} zu gegebenem α genutzt. Ein Vergleich der Größenordnung der verbleibenden parasitären Restströmung unter diesen Bedingungen mit den Ergebnissen am analytisch vordefiniertem Tropfen zeigt nur eine sehr geringe Steigerung von U_{fehl}. So liegen die Geschwindigkeitswerte nach Erreichen des unteren Plateaus nun in einer Größenordnung von $3{,}26 \cdot 10^{-5}\,\mathrm{m/s}$ bis $1{,}19 \cdot 10^{-4}\,\mathrm{m/s}$, während sie am analytisch vordefinierten Tropfen zwischen $1{,}16 \cdot 10^{-5}\,\mathrm{m/s}$ und $4{,}58 \cdot 10^{-5}\,\mathrm{m/s}$ liegen. Diese Steigerung wird im Vergleich zu den in Abbildung 3.6 auftretenden Größenordnungen als vernachlässigbar klein angesehen. Es wird daher geschlussfolgert, dass die angestrebten Gleichgewichtszustände des berechneten Druckfeldes und des implementierten Kompressionstermes hinreichend genau untereinander übereinstimmen und zudem auch eine ausreichende Ähnlichkeit mit der analytischen Lösung besitzen (vgl. Abbildung 2.4).

Aus den bisherigen Ergebnissen lässt sich nun begründet ableiten, dass eine unabhängige Wahl von α und ε zwar möglich, aber nicht sinnvoll ist. In Gl. 3.21 zeigt sich, dass eine gleichzeitige Verdopplung von α und ε zu genau dem gleichen Verlauf von $f(L)$ führt, da L invers mit ε skaliert. Statt der festen Werte von α und ε steht demnach eher das Verhältnis α/ε im Vordergrund. Nach oben ist dieses Verhältnis durch Gl. 3.57 beschränkt. Kleine Werte von α/ε sind zwar möglich, aber wegen des dann sehr langsamen Übergangs von $c = -1$ zu $c = +1$ nicht sinnvoll. Das Verhalten der numerischen Methode wird daher lediglich durch das Verhältnis α/ε bestimmt. Im Rahmen dieser Arbeit soll eine möglichst dünne Grenzfläche realisiert werden. Im weiteren Verlauf wird das Verhältnis α/ε daher stets nach Gl. 3.57 so groß wie möglich gewählt. Willkürlich wird hierzu $\alpha = -2$ gesetzt, was einen Wert von $\varepsilon = 1,25 \cdot \Delta x$ bedingt. Diese Festlegung schließt die Optimierung der für $f(L)$ zu wählenden Parameter ab. Tabelle 3.6 gibt den abschließenden Überblick des in diesem Abschnitt abgeleitetem Parametersatzes, der stets Verwendung findet, sofern nicht andere Parameter explizit erwähnt werden.[16]

Tabelle 3.6: Übersicht des im Folgenden verwendeten Parametersatzes zur Definition von $f(L)$ nach Gl. 3.21.

α	ε	L_{outer}	L_{inner}	$\beta_{opt.}$
-2	$1,25 \cdot \Delta x$	9	$0,8214743858088364$	$0,1469702052449879$

[16]Die exakte Bestimmung von L_{inner} und $\beta_{opt.}$ ist numerisch schwierig. Eine sehr genaue Bestimmung ist nur über das Rechnen mit mehr als 16 Nachkommastellen z.B. in MATLAB 2018a, möglich. Leichte Abweichungen im Bereich von 10^{-8} zu den hier angegebenen Werten, die z.B. durch Berechnungen in anderen Programmen wie MS Excel entstehen können, führen nach Sichtung der in dieser Arbeit durchgeführten Simulationen jedoch zu keiner signifikanten Abweichung im Verhalten der numerischen Methode.

3.6 Verhalten der numerischen Methode

Im folgenden Schritt werden nun die Restströmungen betrachtet, wenn die Phasengrenzfläche tatsächlich konvektiv transportiert wird. Für die genaue Implementierung und die Beschreibung der Kopplung aller Gleichungen der Methode wird auf Abschnitt 3.8 verwiesen. Stattdessen wird dargestellt, welchen Einfluss unter Anderem die Kompressionsstärke D und die effektive Transportgeschwindigkeit U_0 auf die Größe der parasitären Restströmungen U_{fehl} haben.

Grundsätzlich stehen bei der Wahl der Kompressionsstärke D verschiedene Interessen miteinander in Konflikt. Auf der einen Seite sollen die parasitären Restströme effektiv unterdrückt werden, da sie unphysikalisch sind und das Ergebnis verfälschen können. Hier kann das Hinzufügen eines Kompressionsterms effektiv sein. Der Kompressionsterms soll jedoch die notwendige Rechenzeit kaum beeinflussen. Dies ist gerade dann zu beachten, wenn durch die Limitierung von D durch Gl. 3.53 mehrere Kompressionsschritte notwendig werden, um eine ausreichende Glättung der Grenzfläche zu erzielen. Um den Einfluss von D zu charakterisieren wird daher an der gleichen Konfiguration aus Abbildung 3.4, dieses Mal jedoch bei gleichzeitigem konvektiven Transport der Grenzfläche, die parasitäre Strömung U_{fehl} am liegenden Tropfen betrachtet. Im Gegensatz zu den Parametern aus Tabelle 3.6 wird hierbei nun $\varepsilon = 2,5 \cdot \Delta x$ gesetzt. Dabei werden auf den Gitterweiten $\Delta x = 4{,}0 \cdot 10^{-6}\,\mathrm{m}$, $\Delta x = 3{,}2 \cdot 10^{-6}\,\mathrm{m}$ und $\Delta x = 2{,}0 \cdot 10^{-6}\,\mathrm{m}$ jeweils Kompressionsstärken im Bereich von $0,005 \cdot D_{max}$ bis $32 \cdot D_{max}$ mit D_{max} nach Gl. 3.53 untersucht.[17] Alle Simulationen erreichen nach spätestens 0,5 s ihren stationären Zustand, welcher sich durch das Erreichen eines konstanten unteren Plateauwertes für U_{fehl} auszeichnet. Die verbleibende Restströmung U_{fehl} ist bei verschiedenen Gitterweiten Δx in Abhängigkeit der genutzten Kompressionsstärke D_{eff} in Abbildung 3.7 dargestellt. Es ist zu beobachten, dass mit ansteigenden Werten von D auch die Restströmungen ansteigen. Dabei führt jedoch ein immer stärkeres Erhöhen von D zu einer immer schwächeren Steigerung von U_{fehl}. Dies wird darauf zurückgeführt, dass der Kompressionsterm und die Krümmungsberechnung in der Druckgleichung durch numerische Diskretisierungsfehler leicht unterschiedliche Gleichgewichtszustände anstreben. Die Transportmechanismen der Konvektion durch U_{fehl} und der Kompression durch D arbeiten dabei gegeneinander, wobei die Stärke der Kompression durch D gesteuert wird. Je höher D ist, desto näher liegt der stationäre Zustand der Grenzfläche am angestrebten Gleichgewichtszustand der Kompressionsgleichung. Die Druckgleichung verzeichnet dadurch eine immer größere Diskrepanz, was zu einer Steigerung der Restströmung führt. Die Diskrepanz beider Gleichgewichtszustände sinkt dabei mit zweiter Ordnung bei einer Verfeinerung des Rechengitters, was durch Abbildung 3.7 bestätigt wird. Zur Einordnung der Ergebnisse dienen die beobachteten Restströmungen am analytisch vordefinierten Tropfen ohne Kompression und ohne Advektion. Hier liegen die Restströmungen bei den verschiedenen Gitterweiten jeweils bei $1{,}16 \cdot 10^{-5}\,\mathrm{m/s}$, $2{,}96 \cdot 10^{-5}\,\mathrm{m/s}$ und $4{,}58 \cdot 10^{-5}\,\mathrm{m/s}$. Der Gleichgewichtszustand der Kompression wird demnach durch gleichzeitigen konvektiven Transport bei Weitem nicht erreicht. So liegen die in Abbildung 3.7 dargestellten Restströmungen um einige Größenordnungen unterhalb der betrachteten Grenzfälle ohne konvektiven Transport. Die Konvektion dominiert demnach die makroskopische Position der

[17] Werte von $D > D_{max}$ werden dabei durch mehrfache Anwendung des Kompressionsschrittes mit $D < D_{max}$ realisiert. Siehe hierzu Abschnitt 3.8.

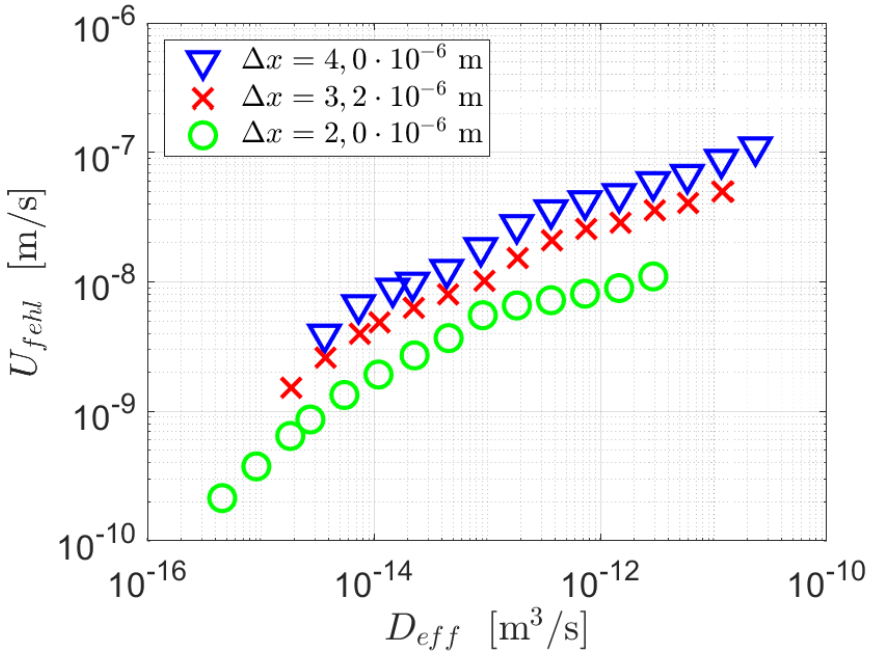


Abbildung 3.7: Parasitäre Strömung im stationären Zustand bei verschiedenen effektiven Gesamtkompressionsstärken D mit konvektivem Transport der Grenzfläche bei verschiedenen Gitterauflösungen Δx.

Grenzfläche, während die Kompression lokal die Form der Grenzfläche dominiert und die gewünschte Glättungseigenschaft einbringt. Zur Abgrenzung bei immer kleiner werdenden Werten von D dient die Relaxation des Tropfens ohne Kompression bei $D = 0$. Hier werden Restströmungen von $5{,}52 \cdot 10^{-12}$ m/s, $2{,}55 \cdot 10^{-14}$ m/s und $6{,}83 \cdot 10^{-13}$ m/s erreicht. Diese Werte entsprechen nicht der maximal möglichen Maschinengenauigkeit, allerdings werden die auftretenden Gleichungssysteme für p und U auch nur bis auf ein Restresiduum von 10^{-14} und 10^{-8} gelöst und die maximalen Iterationszahlen der Druckgleichung zudem auf 20 limitiert. Ein Vergleich dieser Werte mit dem bisherigen Standardsolver für Mehrphasenströmungen interFoam der Version OpenFOAM 6 erfolgt in Abschnitt 3.9.

3.6.1 Parallelisierung und Optimierung des numerischen Codes

Die bisherigen Ergebnisse wurden stets mit einem seriell ausgeführtem numerischen Code berechnet. Um eine kürzere Rechenzeit zu erzielen, wurde eine Parallelisierung des Berechnungsmodells implementiert. Die Vorgehensweise und die Validierung der Parallelisierung werden in diesen Abschnitt dargestellt. Darüber hinaus werden zur weiteren Beschleunigung des Codes zusätzliche Annahmen eingeführt.

Für die Parallelisierung von OpenFOAM-Code existieren verschiedene Ansätze. Zunächst kann vor der Ausführung der Berechnung das Rechengitter in mehrere Untergitter zerteilt werden. Jeder Prozessor übernimmt dann das Lösen der zugehörigen Gleichungssystem für das ihm zugewiesene Untergitter. Ein steter Abgleich der Randwerte an den Berührstellen der Untergitter sorgt für eine rasche Annäherung des Gesamtsystems an die Lösung einer serieller Ausführung. Beide Lösungen

unterscheiden sich zwar, jedoch höchstens in der Größenordnung des Iterationsfehlers, der auch beim Lösen der Gleichungssystem im seriellen Fall akzeptiert worden wäre. Der Zeitgewinn entsteht durch die massive Verkleinerung der Gleichungssysteme für jeden einzelnen Prozessor. Zu der eigentlichen Rechenzeit kommt allerdings der Aufwand für den Abgleich der Randbedingungen an den Schnittstellen der Rechengitter hinzu (parallel overhead), weshalb es durch die immer weiter ansteigende Zeit für die Kommunikation zwischen den Prozessoren untereinander eine optimale Zahl an Prozessoren gibt. Zur optimalen Nutzung dieser Parallelisierungsart, sollten alle Prozessoren ungefähr die gleiche Zeit für eine Iteration benötigen, da die Ausführung des Gesamtcodes auf die Fertigstellung des langsamsten Prozessors warten muss. Daher bestimmt die anfängliche Aufteilung des gesamten Rechengitters auf die jeweiligen Untergitter maßgeblich die Effizienz dieser Parallelisierungsmethode. Diese Art der Parallelisierung ist in OpenFOAM bereits implementiert und kann über die Befehle *decomposePar* und *mpirun* genutzt werden.

Eine weitere Art der Parallelisierung kann durch die Verwendung der openMP-Bibliothek erzeugt werden. Hierdurch können Codeelemente, wie z.B. for-Schleifen, in ihrer Ausführung auf mehrere Prozessoren verteilt werden. Voraussetzung ist dabei, dass die einzelnen Wiederholungen der for-Schleife unabhängig voneinander sind, also in beliebiger Reihenfolge ausgeführt werden können. Die durchzuführenden Iterationen verteilen sich dann auf die verschiedenen Prozessoren und werden gleichzeitig ausgeführt. Diese Zuweisung kann fix oder dynamisch erfolgen, wodurch Leerlauf und Wartestellung einzelner Prozessoren weitgehend vermieden wird. Da die Durchführung der parallelisierten Rechenschritte unabhängig voneinander sind, wird durch diese Art der Parallelisierung sogar das exakt gleiche Endergebnis erhalten, wie bei serieller Ausführung des Codes. Das Lösen der Gleichungssysteme für $\vec{U}$ und p erfolgt dann jedoch durch einen einzelnen Prozessor.

Ein elementarer und zugleich geschwindigkeitsbestimmender Schritt der in dieser Arbeit vorgestellten numerischen Methode ist die Rekonstruktion von L aus gegebenem $\bar{c}$ nach dem in Abschnitt 3.4 beschriebenem Algorithmus. Hierfür kommen prinzipiell beide Arten der Parallelisierung in Frage. Bei der fixen Aufteilung des Rechengitters auf verschiedene Prozessoren hat jedoch automatisch der Prozessor die meiste Arbeitslast, dessen Untergitter auch die meisten Zellen mit $|L| < 9$ enthält. Alle anderen Prozessoren stehen in Wartestellung bis die Rekonstruktion von L auch auf dem letzten Prozessor vollständig abgeschlossen ist. Da die Phasengrenzfläche während der Simulation durch das Rechengitter bewegt wird, kann dieser Überbelastung eines einzelnen Prozessors nicht durch eine geschickte Aufteilung des Rechengitters entgegengewirkt werden. Die Ursache für die Ungleichverteilung der Arbeitslast wandert während der Simulation von einem Untergitter in das nächste, wodurch die Überbelastung eines einzelnen Prozessors sich zwar verschiebt, aber weiterhin bestehen bleibt. Aus diesem Grund wird primär die Parallelisierung über openMP bevorzugt. Es existieren erste Ansätze, innerhalb eines Untergitters, mehrere Prozessoren über openMP an den enthaltenen for-Schleifen arbeiten zu lassen und demnach beide Parallelisierungsmethoden miteinander zu kombinieren [103]. Hierfür sind jedoch sehr tiefgehende Programmierarbeiten erforderlich und die Übertragung von überschüssiger Arbeitslast auf andere Untergitter und dessen Prozessoren wurde in der Literatur soweit nicht gefunden.

Die parallelisierte Ausführung von for-Schleifen lässt sich im numerischen Code dabei an verschiedenen Stellen implementieren. Die in dieser Arbeit realisierten

Parallelisierungen beziehen sich auf:

- Auswertung des Volumenintegrals zur iterativen Bestimmung von L aus $\bar{c}$.
- Auswertung der Flächenintegrale auf den Zellwänden beim konvektiven Transport von $\bar{c}$.
- Identifikation der direkten Grenzflächenzellen mit zugehöriger Berechnung des entstehenden Drucksprungs über die Grenzfläche zwischen den Zellen.
- Entscheidung zwischen der Berechnung räumlicher Ableitungen über L oder direkt über $\bar{c}$ für Δc, $\vec{\nabla} c$ und $\vec{n}$. (Siehe Gl. 3.5 und Gl. 3.6)
- Berechnung des chemischen Potentials Φ in jeder Zelle.

Zur weiteren Beschleunigung der Rechnungen werden bei der Rekonstruktion von L aus $\bar{c}$ zwei weitere Vereinfachungen getroffen. Hierzu wird erneut Abbildung 3.6 aus Abschnitt 3.5 herangezogen. Es wurde geschlussfolgert, dass eine iterative Verbesserung der über $L = f^{-1}(\bar{c})$ bestimmten Schätzung nur dann sinnvoll ist, wenn der Fehlerschätzter nach Gl. 3.60 größer ist als $4 \cdot 10^{-8}$. Bisher wurde dieser Grenzwert lediglich dazu genutzt, über die Annahme von $|\vec{\nabla} L| \approx 1/\varepsilon$ einen sinnvollen Wert für L_{outer} zu bestimmen. In den Randgebieten der Phasengrenzfläche mit $|c| \approx 1$ können trotz Nutzung des Kompressionsterms jedoch auch Gradienten von L auftreten deren Betrag kleiner als $1/\varepsilon$ ist. Anstatt die Zellen für eine Nachiteration von L lediglich nach dem Wert L_{outer} zu bestimmen, wird nun zusätzlich der Fehlerschätzer nach Gl. 3.60 herangezogen. Bei Unterschreiten der Schranke von $4 \cdot 10^{-8}$ erfolgt trotz $|L| < L_{outer}$ keine Nachiteration. Hierdurch sinkt die Zahl der Zellen, in denen die Volumenintegration durchgeführt werden muss. Weiterhin wird die maximale Änderung von L pro Iteration nach Gl. 3.46 auf maximal $0,05$ begrenzt. Dies verbessert das Konvergenzverhalten in Randgebieten der Grenzfläche mit $|c| \approx 1$, da hier oft sehr kleine Werte im Nenner der Iterationsformel entstehen, was eine sehr große Änderung von L während eines Iterationsschrittes bewirken würde.

Um die erfolgreiche Parallelisierung des Codes nachzuweisen, wurde für 0,01 s die Relaxation eines quadratisch initialisierten Tropfens im Bereich zweier rechtwinkliger Wände simuliert. Der initialisierte Tropfen hatte dabei eine Kantenlänge von 0,55 mm und strebt mit der Wand einen Kontaktwinkel von 120° an. Die Implementierung der hierfür erforderlichen Randbedingung für L, $\bar{c}$ und $\vec{n}$ wird in Abschnitt 3.7 beschrieben. Das gesamte Rechengebiet erstreckt sich auf 0,8 mm x 0,8 mm und wird mit einer Gitterweite von $\Delta x = 4 \cdot 10^{-6}$ m in 40000 Zellen unterteilt. Die verwendete Zeitschrittweite beträgt $\Delta t = 1 \cdot 10^{-6}$ s. Zur Beschleunigung des Relaxationsvorgangs wird für die Geschwindigkeit an allen Wänden eine SLIP-Bedingung vorgegeben. Die Initialisierung zum Zeitpunkt $t = 0$ s und der Relaxationszustand bei $t = 0{,}01$ s sind in Abbildung 3.8 dargestellt. Verwendet wurden erneut wasserähnliche Stoffdaten ($\rho = 1000\,\mathrm{kg/m^3}$, $\eta = 1 \cdot 10^{-3}\,\mathrm{Pa \cdot s}$) für beide Phasen und eine Grenzflächenspannung σ von 0,02 N/m, während stets 24 Kompressionsschritte mit $D = D_{max}$ pro Zeitschritt durchgeführt wurden. Die Rechnungen sind dabei mit jeweils unterschiedlichen Prozessorzahlen zwischen 1 und 20 durchgeführt worden. Um auch die notwendigen Simulationszeiten vergleichen zu können, sind alle Rechnungen insgesamt vier Mal durchgeführt worden. Die gebildeten Mittelwerte der Rechenzeiten belegen in Abbildung 3.9 deutlich den Vorteil der Nutzung mehrerer Prozessoren. Mit zunehmender Prozessorzahl wird der erzielbare Nutzen durch

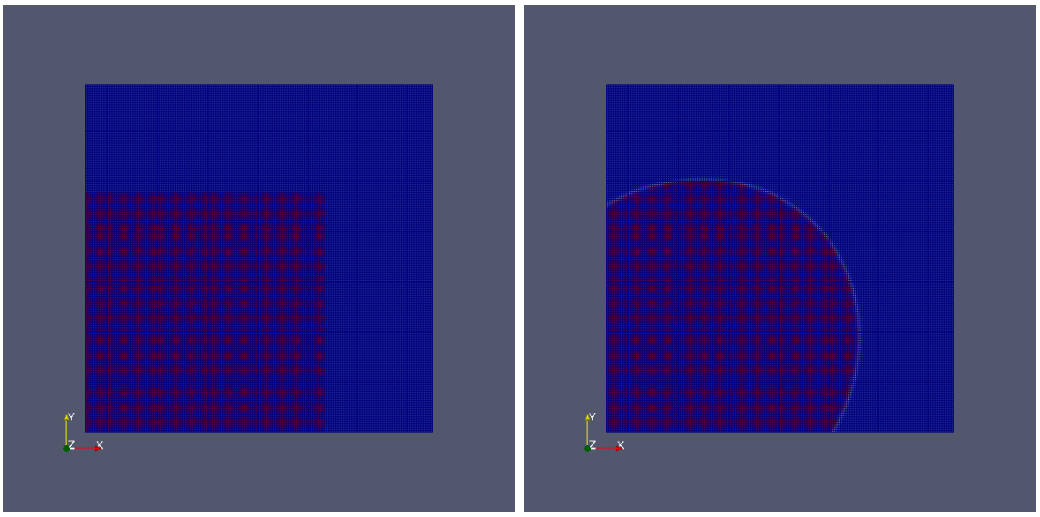


Abbildung 3.8: Relaxation eines quadratisch initialisierten Tropfens (links) zwischen zwei Wänden mit dem erreichten Vergleichszustand bei $t = 0{,}01\,\mathrm{s}$ (rechts) zur Überprüfung der erfolgreichen Codeparallelisierung.

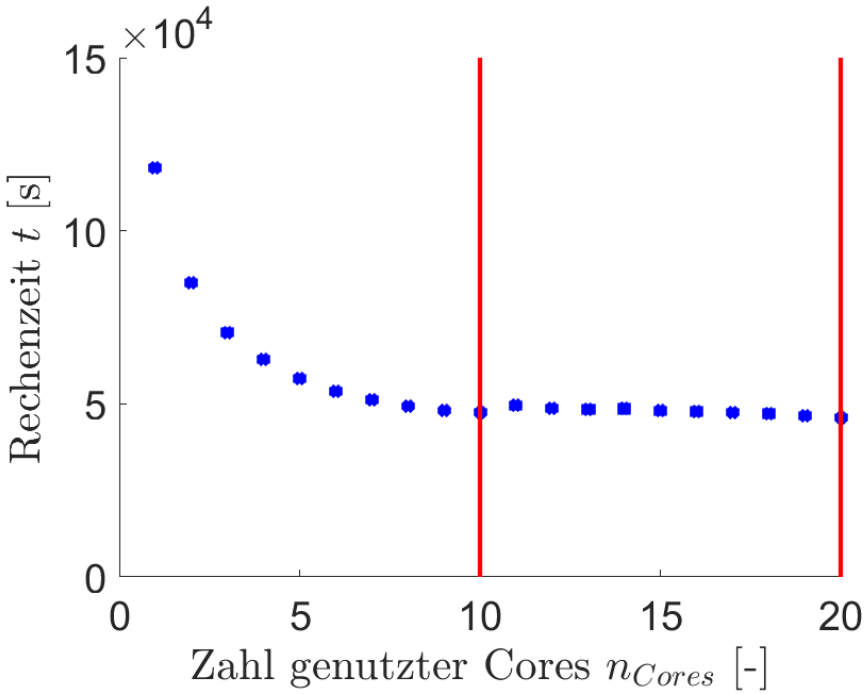


Abbildung 3.9: Darstellung der Zeitersparnis in Abhängigkeit der genutzten Prozessoren durch Nutzung einer Parallelisierung durch die Bibliothek openMP am Beispiel des relaxierenden quadratischen Tropfens. Fehlerindikatoren sind eingezeichnet jedoch zu klein, um in gewählter Darstellung erkennbar zu sein. Sprungstellen für das Hinzufügen weiterer CPUs sind als rote Balken dargestellt.

den steigenden Kommunikationsaufwand zwischen den Prozessoren immer geringer. Darüber hinaus enthält der Code etliche Stellen, welche durch die verwendete Parallelisierungsmethode nicht beschleunigt werden können. Weiterhin ist zu erkennen, dass ab einer Prozessorzahl von zehn die notwendige Rechenzeit sogar teilweise wieder ansteigt. Dies wird auf die Architektur des verwendeten Rechenclusters LiDO3

der Technischen Universität Dortmund zurückgeführt. Auf den genutzten Rechenknoten existieren jeweils zwei CPUs bestehend aus jeweils zehn Prozessorkernen. Das Hinzufügen eines elften Prozessorkerns erfordert somit zusätzliche Kommunikation zwischen den einzelnen CPUs und nicht nur innerhalb einer CPU. Diese Sprungstellen werden durch die roten senkrechten Balken visualisiert. Eine Prozessorkernzahl von 10 wird daher als optimal angesehen.

Eine erfolgreiche Parallelisierung über die openMP-Bibliothek sollte dabei stets zu den exakt gleichen numerischen Ergebnissen führen, wie die serielle Ausführung. Alle 32 numerisch berechneten Felder (p, $\vec{U}$, $\bar{c}$, L, etc.) der durchgeführten Simulationen wurden daher per MATLAB-Skript miteinander verglichen. Eine Übereinstimmung bis auf die letzte Zahl und den letzten Buchstaben ist in allen 32 Fällen untereinander gegeben, was die Parallelisierung validiert. Für weiterführende Untersuchungen besteht somit die Möglichkeit, bei exakt gleichen numerischen Ergebnissen, die Rechenzeit erheblich zu verkürzen.

3.6.2 Weiterführende Untersuchungen

Durch die erfolgreiche Parallelisierung steht ein Werkzeug für die Durchführung breiter Parameterstudien zur Verfügung. Von besonderem Interesse ist nun das Verhalten einer Grenzfläche, welche konvektiv durch das Rechengitter transportiert wird, statt stationär an einem Ort des Gitters zu verweilen. Hierzu wird in einer Kapillare mit einem Radius $R_{cap} = 0{,}8\,\mathrm{mm}$ ein Tropfen mit einem Radius $R_{drop} = 0{,}7\,\mathrm{mm}$ initialisiert und mit der Transportgeschwindigkeiten U_0 eine Strecke von $1{,}6\,\mathrm{mm}$ transportiert. Es gelten erneut $\rho_1 = \rho_2 = 1000\,\mathrm{kg/m^3}$, $\eta_1 = \eta_2 = 1 \cdot 10^{-3}\,\mathrm{Pa \cdot s}$ und $\sigma = 0{,}02\,\mathrm{N/m}$. Angesetzt wird hierbei ein kastenförmiges Strömungsprofil $\vec{U} = U_0 \cdot \vec{e}_x$ am Einlasse mit Haftbedingung an der ebenfalls mit U_0 bewegenden Wand. Hierdurch lässt sich besonders einfach die Größe der parasitären Strömung als Abweichung vom sonst einheitlich vorherrschendem U_0 bestimmen, indem vom an die Stelle $L = 0$ interpolierten Wert der Geschwindigkeitsvektor $\vec{U}_{L=0}$ der Sollanteil $\vec{e}_x \cdot U_0$ subtrahiert wird. In einem parabolischen Geschwindigkeitsprofil wäre der eigentliche Sollwert der Geschwindigkeit beim Tropfentransport nicht bekannt und U_{fehl} könnte daher nicht ermittelt werden.

Im stationären Zustand bezüglich U_{fehl}, der in vielen Fällen bereits weit vor dem Erreichen der Transportstrecke von $1{,}6\,\mathrm{mm}$ vorliegt, wird aus den letzten Zeitschritten durch Mittelwertbildung ein repräsentativer Wert für U_{fehl} gebildet. Abbildung 3.10 veranschaulicht dies. Die beobachtete Restströmung liegt am dargestellten Beispiel bei $2{,}07 \cdot 10^{-5}\,\mathrm{m/s}$. Demnach beträgt die Größe der parasitären Restströmung verglichen mit der dominierenden Hauptströmung U_0 lediglich etwa $0,4\%$. In wie weit D und U_0 die Größe U_{fehl} beeinflussen, wird mit Hilfe der Permutationsparameter aus Tabelle 3.7 in insgesamt 75 Testkonfigurationen untersucht. Die auftretenden parasitären Restströmungen liegen dabei in einem Bereich von $1{,}48 \cdot 10^{-5}\,\mathrm{m/s}$ bis $2{,}13 \cdot 10^{-3}\,\mathrm{m/s}$. Es ist zu beobachten, dass die Größe der parasitären Strömungen dabei mit steigender Transportgeschwindigkeit U_0 zunimmt, während sie mit einer Gitterverfeinerung und einer höheren Kompressionsstärke abnimmt. Zur quantitativen Beschreibung wurde eine Korrelation nach Gl. 3.61 an den erhaltenen Datensatz angepasst. Alle Werte sind ungeachtet der Parameterwerte a_1 bis a_4 in SI-Einheiten einzusetzen und zu verstehen.

$$U_{fehl,Korr}\left[\frac{m}{s}\right] = a_1 \cdot {U_0}^{a_2} \cdot \left(n_{Comp} \cdot D_{max}\right)^{a_3} \cdot \Delta x^{a_4} \tag{3.61}$$

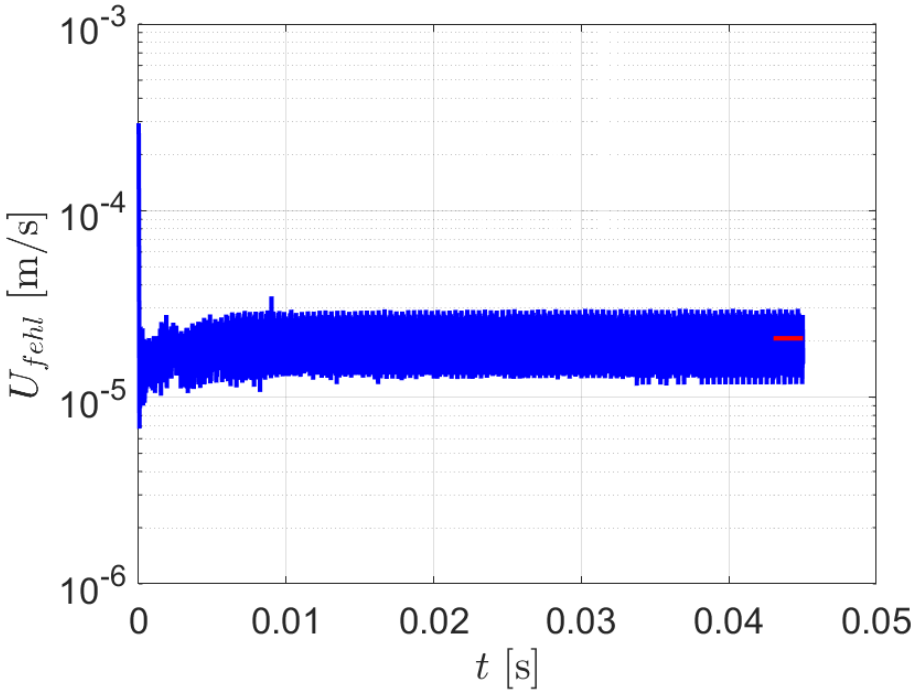


Abbildung 3.10: Verlauf der parasitären Strömung über die Zeit am konvektiv transportiertem Tropfen für $U_0 = 0{,}005\,\mathrm{m/s}$, $D = 16 \cdot D_{max}$ und $\Delta x = 2{,}0 \cdot 10^{-6}\,\mathrm{m}$ (blau) mit zugehörigem Mittelwert (rot) der letzten 2000 Zeitschritte.

Tabelle 3.7: Erster Satz an Permutationsparametern zur Charakterisierung der parasitären Strömungen am konvektiv transportiertem Tropfen.

$\Delta x = \Delta y$	$n_{Comp} = \frac{D}{D_{max}}$	U_0
$4{,}0 \cdot 10^{-6}\,\mathrm{m}$	1	$0{,}005\,\mathrm{m/s}$
$3{,}2 \cdot 10^{-6}\,\mathrm{m}$	2	$0{,}01\,\mathrm{m/s}$
$2{,}0 \cdot 10^{-6}\,\mathrm{m}$	4	$0{,}02\,\mathrm{m/s}$
	8	$0{,}04\,\mathrm{m/s}$
	16	$0{,}08\,\mathrm{m/s}$

Die Parameter selbst werden durch Minimierung von relativen Fehlerquadrate nach Gl. 3.62 bestimmt.

$$Q_{res} = \sum_i \left(\frac{U_{fehl,Sim,i} - U_{fehl,Korr,i}}{U_{fehl,Sim,i}} \right)^2 \tag{3.62}$$

Die Wahl von relativen Fehlerquadraten liegt darin begründet, dass die zu beschreibenden Daten sich über mehrere Größenordnungen erstrecken, eine Abweichung zwischen Korrelation und Simulation in der Größe von $1{,}0 \cdot 10^{-4}\,\mathrm{m/s}$ bezogen auf einen Wert von $1{,}48 \cdot 10^{-5}\,\mathrm{m/s}$ jedoch wesentlich schwerwiegender ist als im Vergleich zum oberen Grenzwert von $2{,}13 \cdot 10^{-3}\,\mathrm{m/s}$. Die Normierung der Fehlerquadrate mit dem zutreffendem Geschwindigkeitswert $U_{fehl,Sim,i}$ im Nenner führt die entsprechende Gewichtung ein. Abbildung 3.11 verdeutlicht die gute Übereinstimmung zwischen Korrelation und Simulationsergebnis für alle 75 betrachteten Fälle. Mit einem Rest-

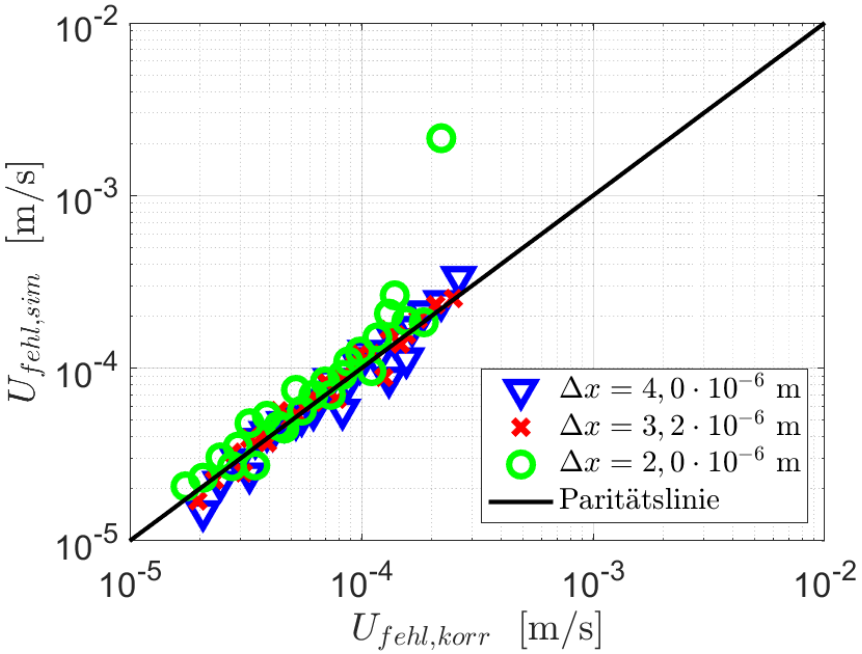


Abbildung 3.11: Quantitativer Vergleich zwischen Korrelation und Simulation für den konvektiv transportierten Tropfen bei verschiedenen Gitterweiten Δx.

residuum von $Q_{res} = 2,66$ beträgt die relative Abweichung zwischen Korrelation und den 75 Simulation durch die in Tabelle 3.8 dargestellten Konstanten demnach $\sqrt{Q_{res}/75} \cdot 100\% \approx 18,9\%$. Die Konstanten werden dabei wie folgt interpretiert:

Tabelle 3.8: Angepasste Konstanten a_i zur Minimierung der Fehlerquadrate nach Gl. 3.62.

a_1 , Konstante	a_2 , Exponent U_0	a_3 , Exponent D_{eff}	a_4 , Exponent Δx
$0,047963511$	$0,676081045$	$-0,320128131$	$0,983695548$

Die parasitären Strömungen nehmen erster Ordnung mit der Gitterweite Δx ab, was einen Wert von $a_4 = 1$ bedeuten würde. Dieses Verhalten entspricht nach den Überlegungen aus Abschnitt 3.2 und 3.4 den Erwartungen. In Anlehnung an normale Diffusionsprozesse, welche durch die zweite Ableitung von c und eine mittlere Eindringtiefe von $\bar{x} = \sqrt[2]{D \cdot t}$ charakterisiert sind, erscheint für die Modellierung des Kompressionsterms eine Beschreibung über die vierte Wurzel der Kompressionsstärke $\sqrt[4]{D}$ sinnvoll. Dies ist darin begründet, dass effektiv die vierte Ableitung von c über das chemische Potential Φ die Glättung bestimmt. Der entsprechende Koeffizient betrüge somit $a_3 = -0,25$. Für das Verhalten bezüglich der Transportgeschwindigkeit U_0 konnte keine tiefere Begründung identifiziert werden. Da Diskretisierungsfehler beim Transport einer Grenzfläche mit dem Fluss durch die Zellwand multipliziert werden, wäre eine lineare Zunahme von U_{fehl} mit U_0 zu erwarten gewesen. Stattdessen erscheint eine fast perfekte Abhängigkeit mit $a_2 = 2/3$. Die letzte Konstante a_1 beinhaltet alle weiteren nicht variierten Einflüsse, wie Dichte, Viskosität, Radius des Tropfens, relative Grenzflächendicke und Grenzflächenspan-

nung. Wird der entstandene Datensatz nicht durch die in Tabelle 3.8 dargestellten Konstanten, sondern durch die hier diskutierten anscheinend zu Grunde liegenden Mechanismen beschrieben, ergibt sich Gl. 3.63. Alle Werte sind erneut unabhängig der Parameter a_i in SI-Einheiten einzusetzen und zu verstehen.

$$U_{fehl,Korr}\left[\frac{m}{s}\right] = 0,370538556 \cdot \frac{{U_0}^{\frac{2}{3}} \cdot \Delta x}{\sqrt[4]{n \cdot D_{max}}} \tag{3.63}$$

Die relative Abweichung der Korrelation steigt durch diese Änderungen von $18,9\%$ marginal auf $21,4\%$. Zudem wurde gerade bei hohen Werten von U_{fehl} beobachtet, dass kein monoton glatter Verlauf der parasitären Strömungen über die Zeit zu beobachten ist, wie nach Abbildung 3.10. Würden diese Datenpunkte nicht berücksichtigt ergäben sich wesentlich kleinere relative Abweichungen.

Zur Untersuchung weiterer Einflussfaktoren wurde in einer zusätzlichen Parameterstudie die relative Grenzschichtdicke $\varepsilon/\Delta x$ variiert. Die Permutationsparameter sind in Tabelle 3.9 dargestellt. Wie in Abschnitt 3.2 und 3.4 beschrieben wurde,

Tabelle 3.9: Zweiter Satz an Permutationsparametern zur Charakterisierung der parasitären Strömungen am konvektiv transportiertem Tropfen.

$\Delta x = \Delta y$	$n_{Comp} = \frac{D}{D_{max}}$	U_0	ε
$4{,}0 \cdot 10^{-6}\,\mathrm{m}$	4	$0{,}04\,\mathrm{m/s}$	$1,25 \cdot \Delta x$
$3{,}2 \cdot 10^{-6}\,\mathrm{m}$	8	$0{,}08\,\mathrm{m/s}$	$1,65 \cdot \Delta x$
$2{,}0 \cdot 10^{-6}\,\mathrm{m}$	16		$2,05 \cdot \Delta x$
			$2,45 \cdot \Delta x$

sinken Diskretisierungsfehler nicht nur mit einer Verkleinerung von Δx, sondern auch mit einer Vergrößerung von ε. Da die relative Grenzschichtdicke bisher stets $\varepsilon/\Delta x = 1,25$ betrug, wird daraus abgeleitet, dass die Korrelation zu Gl. 3.64 zu erweitern ist.

$$U_{fehl,Korr}\left[\frac{m}{s}\right] = 0,370538556 \cdot \frac{{U_0}^{\frac{2}{3}} \cdot \Delta x}{\sqrt[4]{n \cdot D}} \cdot \left(\frac{1,25 \cdot \Delta x}{\varepsilon}\right)^{a_5} \tag{3.64}$$

Hierdurch werden unabhängig von a_5 alle bisher beschriebenen 75 Rechnungen unverändert wiedergegeben. Die Bestimmung von a_5 durch erneute Minimierung der relativen Fehlerquadrate führt auf einen Wert von $a_5 = 5,945$. Die Übereinstimmung mit den Simulationsergebnissen ist jedoch nicht in dem Maße überzeugend, wie in der vorangegangenen Parameterstudie. Die Resultate sind in Abbildung 3.12 dargestellt. In grün sind dabei die 18 Wiederholungssimulationen dargestellt, bei denen $\varepsilon = 1,25 \cdot \Delta x$ auf unterschiedlichen Gitterweiten Δx gilt. Die blauen Datenpunkte fügen sich gut der neu angepassten Korrelation mit $a_5 = 5,945$ und besitzen eine größere relative Grenzschichtdicken als $\varepsilon = 1,25 \cdot \Delta x$ auf ebenfalls unterschiedlichen Gitterweiten Δx. Eine Ausnahme hiervon bilden die violetten Datenpunkte. Sie zeigen eine schlechte Übereinstimmung und scheinen bezüglich der Restströmungen einen Wert von $U_{fehl} = 1{,}10 \cdot 10^{-5}\,\mathrm{m/s}$ nicht unterschreiten zu können. Allen diesen Datenpunkten ist gemein, dass sie mit besonders hohen relativen Grenzschichtdicken von $\varepsilon = 2,05 \cdot \Delta x$ und $\varepsilon = 2,45 \cdot \Delta x$ auf dem besonders feinen Gitter mit

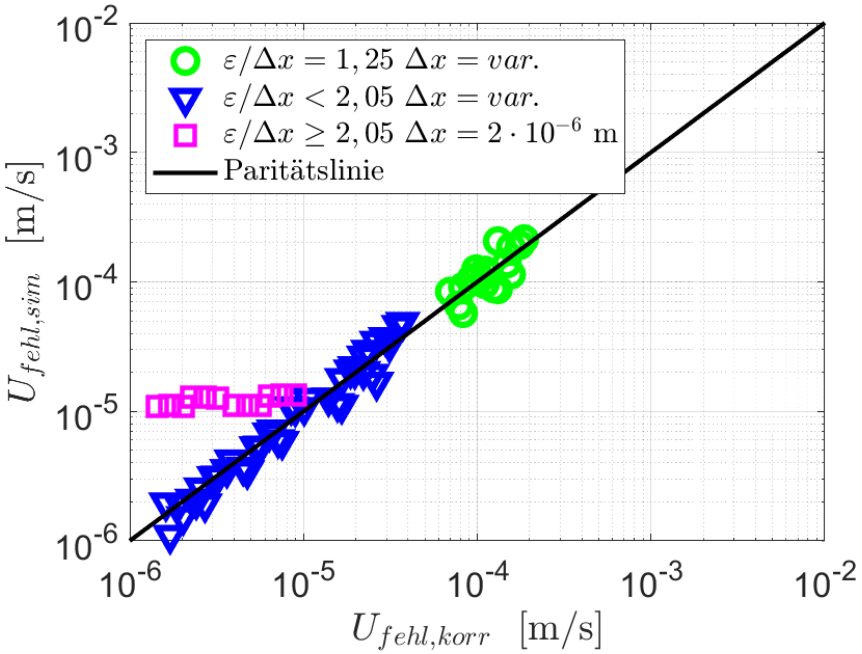


Abbildung 3.12: Quantitativer Vergleich zwischen Korrelation und Simulation für den konvektiv transportierten Tropfen bei verschiedenen relativen Grenzschichtdicken $\varepsilon/\Delta x$.

$\Delta x = 2 \cdot 10^{-6}$ m entstanden sind. Ein kausaler Zusammenhang zwischen diesen Beobachtungen konnte jedoch nicht gefunden werden. Trotz der nur unzureichenden quantitativen Beschreibung ist jedoch offensichtlich, dass ε einen essentiellen Einfluss auf U_{fehl} besitzt. Dennoch kann ε kaum effektiv gesteigert werden, da zur Beschreibung der dünnen Wandfilmregionen in einer Pfropfenströmung die Phasengrenzfläche ebenfalls sehr dünn aufgelöst werden muss.

Zur Charakterisierung des zeitlichen Diskretisierungsfehlers wurden weiterhin ausgewählte Fälle mit reduzierter Zeitschrittweite erneut berechnet. Die untersuchten Parametervariationen sind in Tabelle 3.10 dargestellt. Aus den 54 Fällen lassen

Tabelle 3.10: Dritter Satz an Permutationsparametern zur Charakterisierung der parasitären Strömungen am konvektiv transportiertem Tropfen.

$\Delta x = \Delta y$	$n_{Comp} = \frac{D}{D_{max}}$	U_0	Δt
$4{,}0 \cdot 10^{-6}$ m	16	0,08 m/s	$1{,}0 \cdot 10^{-6}$ s
$3{,}2 \cdot 10^{-6}$ m	8	0,04 m/s	$5{,}0 \cdot 10^{-7}$ s
$2{,}0 \cdot 10^{-6}$ m		0,02 m/s	$2{,}5 \cdot 10^{-7}$ s

sich insgesamt 16 Gruppierungen bilden, in denen innerhalb einer Gruppe jeweils alle Parameter bis auf die Zeitschrittweite Δt identisch sind. Demnach sollte innerhalb einer Gruppierung laut Gl. 3.64 ein einheitliches U_{fehl} vorliegen. Die beobachteten Abweichungen innerhalb jeder Gruppierung betragen höchsten $18,1\%$ und liegen im Mittel bei $4,9\%$. Abweichungen dieser Größe liegen im Rahmen der relativen Genauigkeit der Korrelation und werden daher nicht als signifikant eingestuft. Ebenso

ist nicht klar erkennbar, dass kleinere Zeitschrittweiten konsequent ein kleineres U_{fehl} bewirken. Zeitliche Diskretisierungsfehler durch das Runge-Kutta-Verfahren oder den verwendeten PISO-Algorithmus werden daher als vernachlässigbar klein erachtet.

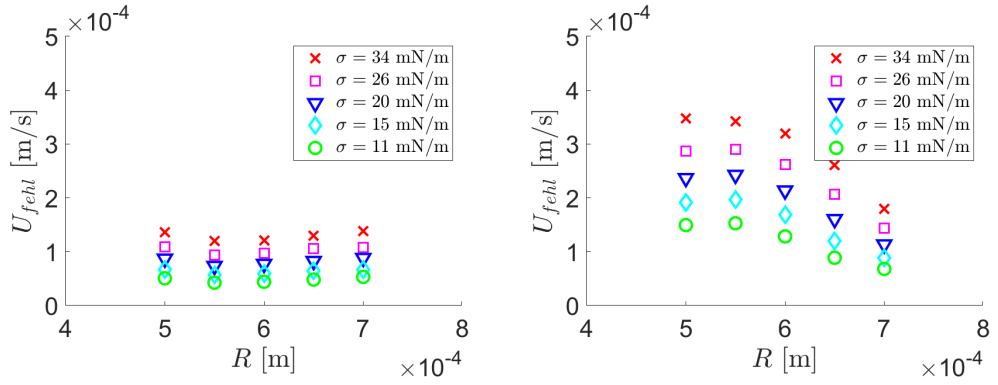
Zuletzt soll noch die Abhängigkeit der parasitären Strömungen von der vorliegendem Grenzflächenspannung σ und dem Tropfenradius R durch eine Parametervariation nach Tabelle 3.11 untersucht werden. Die entsprechenden Verläufe sind

Tabelle 3.11: Vierter Satz an Permutationsparametern zur Charakterisierung der parasitären Strömungen am konvektiv transportiertem Tropfen.

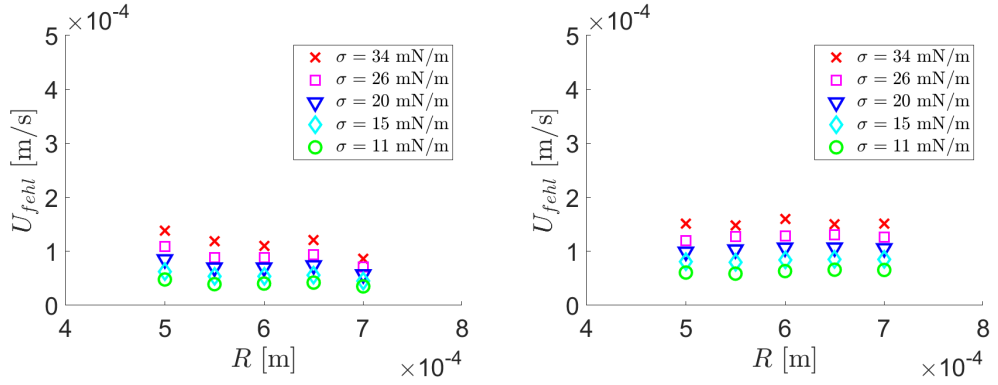
$\Delta x = \Delta y$	$n_{Comp} = \frac{D}{D_{max}}$	U_0	σ	R
$4{,}0 \cdot 10^{-6}\,\mathrm{m}$	16	0,08 m/s	34 mN/m	0,70 mm
	8	0,04 m/s	26 mN/m	0,65 mm
			20 mN/m	0,60 mm
			15 mN/m	0,55 mm
			11 mN/m	0,50 mm

für verschiedene Grenzflächenspannungen σ und Tropfenradien R in Abbildung 3.13 dargestellt. Die Abhängigkeit bezüglich der Grenzflächenspannung σ lässt sich ausgehend von den Referenzfällen mit $\sigma_0 = 0{,}02\,\mathrm{N/m}$ sehr gut durch einen Potenzansatz mit $(\sigma/\sigma_0)^{0{,}85}$ beschreiben. In Bezug auf den Tropfenradius lässt sich in der Abhängigkeit jedoch kein eindeutiger Trend erkennen. Auffällig ist, dass bei niedriger Transportgeschwindigkeit U_0 kaum eine Abhängigkeit bei variierenden Radien zu erkennen ist, während bei hoher Transportgeschwindigkeit und niedrigerer Kompressionskraft ein starker Anstieg zu niedrigeren Tropfenradien hin zu verzeichnen ist. Die Interpretation gestaltet sich schwierig, da sich verschiedene Effekte überlagern. Auf der einen Seite steigt der Kapillardruck mit sinkendem Tropfenradius, was einen Anstieg der parasitären Strömungen U_{fehl} verursacht. Dieses Verhalten wurde bereits bezüglich der Grenzflächenspannung σ beobachtet. Zusätzlich steigen mit sinkendem Tropfenradius die höheren Ableitungen von L an. Die Annahme eines linearen Verlaufs von L innerhalb jeder Zelle verliert dadurch zunehmend an Gültigkeit, was ebenfalls einen Anstieg von U_{fehl} bewirkt. Aus den dargestellten Mechanismen lässt sich jedoch keine quantitative Beschreibung ableiten, die eine Abhängigkeit im Falle von hoher Transportgeschwindigkeit und kleinerer Kompressionskraft beschreiben kann.

Nachdem der Modellfehler der numerischen Methode hinreichend charakterisiert wurde, soll nun die Frage nach dem zulässigen Iterationsfehler untersucht werden. Eine zu große Iterationszahl und eine zu strenge Toleranz beim Lösen der Gleichungssysteme kostet lediglich Rechenzeit ohne einen Zugewinn an Genauigkeit zu liefern. Zunächst wird untersucht, wie viele Iterationen bei der Rekonstruktion von L aus $\bar{c}$ notwendig sind. Hierfür werden die 54 Permutationen nach Tabelle 3.12 auf ihre parasitäre Restströmung U_{Fehl} untersucht. Die Variation der Iterationszahl n_{Iter} hatte bei sonst konstanten Randbedingungen keinerlei Einfluss auf die Größe der parasitäre Restströmung U_{Fehl}. Ein Absenken der Iterationszahl auf Werte $n_{Iter} < 5$ wird wegen der Implementierung der Randbedingung für den Kontaktwinkel (siehe Abschnitt 3.7) jedoch nicht in Erwägung gezogen. Abschließend wurde an zwei



(a) $n_{Comp} = 16$ und $U_0 = 0{,}08\,\mathrm{m/s}$ (b) $n_{Comp} = 8$ und $U_0 = 0{,}08\,\mathrm{m/s}$



(c) $n_{Comp} = 16$ und $U_0 = 0{,}04\,\mathrm{m/s}$ (d) $n_{Comp} = 8$ und $U_0 = 0{,}04\,\mathrm{m/s}$

Abbildung 3.13: Abhängigkeit der parasitären Strömungen U_{fehl} von der Grenzflächenspannung σ und dem Tropfenradius R bei verschiedenen Kompressionsstärken und Transportgeschwindigkeiten.

Tabelle 3.12: Fünfter Satz an Permutationsparametern zur Bestimmung der minimal notwendigen Iterationszahl n_{Iter} bei der Rekonstruktion von L aus $\bar{c}$.

$\Delta x = \Delta y$	$n = \frac{D}{D_{max}}$	U_0	n_{Iter}
$4{,}0 \cdot 10^{-6}\,\mathrm{m}$	16	$0{,}08\,\mathrm{m/s}$	5
	8	$0{,}04\,\mathrm{m/s}$	6
	4	$0{,}02\,\mathrm{m/s}$	7
			8
			9
			10

weiteren Fallbeispielen mit besonders hohem und besonders niedrigem U_{Fehl} identifiziert, dass die Gleichungssysteme für $\vec{U}$ lediglich auf 10^{-6} genau gelöst werden müssen, wo bisher eine Toleranz von 10^{-10} gesetzt worden ist. Die Druckgleichung konnte von einem zunächst gefordertem Residuum von 10^{-14} dahin geführt werden, dass in jeder Iterationsschleife des PISO-Algorithmus das Residuum lediglich um 3 Größenordnungen sinken muss. Alle drei Maßnahmen konnten eine erhebliche Beschleunigung der Simulationen herbeiführen.

3.7 Implementierung der Randbedingung für den Kontaktwinkel

In diesem Abschnitt soll detailliert erläutert werden, auf welche Weise der vorgegebene Kontaktwinkel θ in den Randbedingungen für c, L und $\vec{n}$ mathematisch modelliert und in OpenFOAM 2.1.1 implementiert wird. Jacqmin ([59], [60])) gibt hierfür in einer seiner Arbeiten beispielhaft die Realisierung eines Kontaktwinkels von 45° für sein verwendetes Profil des tangens hyperbolicus an. [60] Kernpunkt ist dabei die Annahme, dass die Grenzfläche direkt an der Wand im thermodynamischen Gleichgewicht ist und jegliche Potentialdifferenzen Φ in Normalenrichtung zur Wand verschwinden.[18] Daher wird angenommen, dass die Grenzfläche direkt an der Wand ihren Gleichgewichtszustand annimmt und durch $f(L)$ beschrieben werden kann. Es ist daher sinnvoll zunächst die Randbedingung für L zu formulieren, da $\vec{\nabla}L$ nach Gl. 3.5 zur Formulierung von $\vec{\nabla}c$ benötigt wird. Im Gleichgewichtszustand sollte $\vec{\nabla}L$ stets den Betrag von $1/\varepsilon$ besitzen und in Normalenrichtung der Grenzfläche zeigen. Der Winkel zwischen den Normalenvektoren von Grenzfläche und Wand wird durch die Randbedingung vorgegeben. Hieraus ergibt sich die in Gl. 3.65 gegebene Beziehung, welche in Abbildung 3.14 visuell verdeutlicht wird.

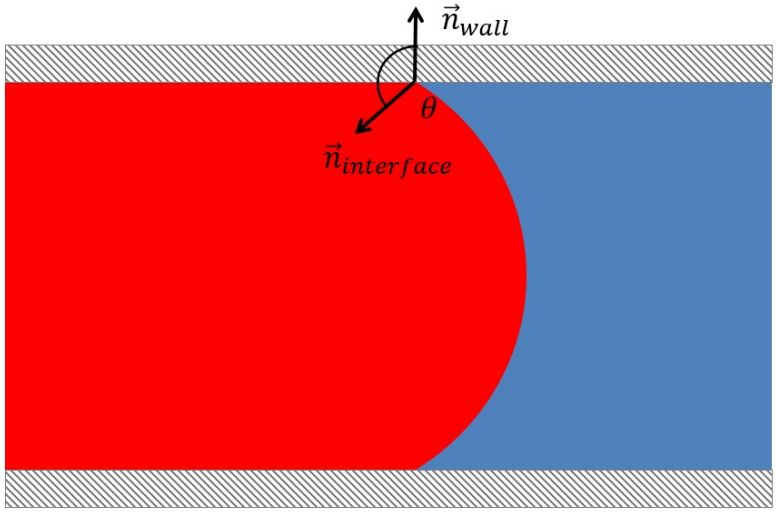


Abbildung 3.14: Veranschaulichung der Definition des Kontaktwinkels θ unter Verwendung der Normalenvektoren $\vec{n}_{interface}$ und $\vec{n}_{wall}$.

$$\varepsilon \cdot \vec{\nabla}L * \vec{n}_{wall} = \cos\theta \tag{3.65}$$

Das Skalarprodukt aus $\vec{\nabla}L$ und $\vec{n}_{wall}$ beschreibt dabei direkt den Normalengradienten $\partial L/\partial x_n$, der in OpenFOAM in der Randbedingung implementiert wird. Ausgehend von Gl. 3.65 und Gl. 3.5 kann nun die Randbedingung für c durch Gl. 3.66 formuliert werden.

$$\frac{\partial c}{\partial x_n} = \vec{\nabla}c * \vec{n}_{wall} = \frac{\partial f(L)}{\partial L} \cdot \vec{\nabla}L * \vec{n}_{wall} = \frac{\partial f(L)}{\partial L} \cdot \frac{\cos\theta}{\varepsilon} \tag{3.66}$$

[18] Zur Absicherung der Massenerhaltung ist es sogar zwingend erforderlich, dass in Normalenrichtung zur Wand keinerlei chemische Potentialdifferenz Φ vorliegt. Würde eine Potentialdifferenz existieren, so würde Material durch die Wand hinein- oder hinausdiffundieren.

Die Nutzung der Ableitung von $f(L)$ in der Randbedingung für c erzwingt dabei die Berechnung von L, was nur für Werte mit $|c| < 1$ möglich ist. Gerade in Bereichen mit $|c| \approx 1$, ist jedoch eine explizite Inverse von $f(L)$ bekannt (Gl. 3.23). Hier lässt sich die Randbedingung für c daher umformulieren und über Gl. 3.67 darstellen.

$$\frac{\partial c}{\partial x_n} = -\exp(\alpha \cdot |L| + \beta) \cdot \frac{\cos\theta}{\varepsilon} = (|c| - 1) \cdot \alpha \cdot \frac{\cos\theta}{\varepsilon} \tag{3.67}$$

Im Gegensatz zur Darstellung über L lässt sich diese Gleichung analytisch fortsetzen und auch dann anwenden, wenn $|c| > 1$ gilt. Prinzipiell ist in Gl. 3.67 der Wert an der Wand c_{wall} zu verwenden, der jedoch noch unbekannt ist. Den Wert der Zelle c_{cell} zu verwenden führt einen Fehler erster Ordnung in Δx ein, vereinfacht jedoch wesentlich die Implementierung. Zudem ist der Fehler allgemein klein, da Gl. 3.67 stets nur in Zellen mit $|c| \approx 1$ Anwendung findet und der Normalengradient dadurch bereits verschwindend klein ist.[19] Aus Normalengradient und Zellwert c_{cell} kann der Wandwert c_{wall} berechnet werden, welcher aus numerischen Gründen anschließend noch auf $|c| \leq 1$ begrenzt wird.

Für die Berechnung der Krümmung κ wird der Normalenvektor $\vec{n}$ an der Wand benötigt. Hierfür wird zunächst über $\vec{\nabla}c$ bzw. $\vec{\nabla}L$ im anliegenden Zellzentrum der dortige Normalenvektor $\vec{n}_{interface,cell}$ gebildet. Es wird angenommen, dass dieser Normalenvektor grundsätzlich korrekt orientiert ist und lediglich im Winkel korrigiert werden muss, den er mit der Wand einschließt. Hierfür wird ein Orthonormalsystem auf der Wand definiert mit $\vec{e}_x = \vec{n}_{wall}$, während $\vec{e}_y$ nach dem Gram-Schmidtschen Verfahren ([65]) aus $\vec{n}_{interface,cell}$ entsteht. Im Bezug auf Abbildung 3.14 gilt demnach Gl. 3.68.

$$\vec{n}_{interface,wall} = \cos\theta \cdot \vec{e}_x + \sin\theta \cdot \vec{e}_y \tag{3.68}$$

3.7.1 Auswirkungen einer verschmierten Grenzflächen auf den lokalen Kontaktwinkel

Eine direkte Implementierung dieser Vorgehensweise durch Vorgabe eines konstanten Kontaktwinkels θ an der gesamten Wand führt jedoch zu erheblichen Ungleichgewichten zwischen Kompressionsterm und berechnetem Druckfeld, wodurch erneut beträchtliche parasitäre Strömungen entstehen. Veranschaulicht wird dies auf einem wedge-Gitter mit den Abmessungen von 0,8 mm und 1,2 mm bei einem Öffnungswinkel von 0,14°. Erneut werden analytisch Phasengrenzflächen mit bestimmten Kontaktwinkeln an der Wand vordefiniert. Betrachtet werden die Kontaktwinkel 40°, 60°, 80°, 100° , 120° und 140°. Die Grenzflächen wurden für 1,0 s durch den Kompressionsterm in ihre Gleichgewichtszustände versetzt, wobei zunächst $n_{Iter} = 10$ Iterationen zur Bestimmung von L aus c verwendet werden. Abbildung 3.15 verdeutlicht das Vorgehen. Bei Kontaktwinkeln zwischen 60° und 120° ist kein auffälliges Verhalten zu verzeichnen. Für die Kontaktwinkel 40° und 140° erzeugt der Kompressionsterm hingegen wellenförmige Deformationen in der Grenzfläche, die definitiv unphysikalisch sind. Abbildung 3.16 zeigt dieses Phänomen. Darauf folgend wurde 1,0 s bei weiterhin aktivem Kompressionsterm die Kräftebilanz gelöst und das resultierende Geschwindigkeitsfeld berechnet, jedoch kein konvektiver Transport der Grenzfläche vorgenommen. Die auftretenden Störströmungen entstehen daher erneut einzig

[19] In allen anderen Gebieten ($|c| \ll 1$) werden die räumlichen Diskretisierungen von c stattdessen über L berechnet (siehe Gl. 3.5 und Gl. 3.6)

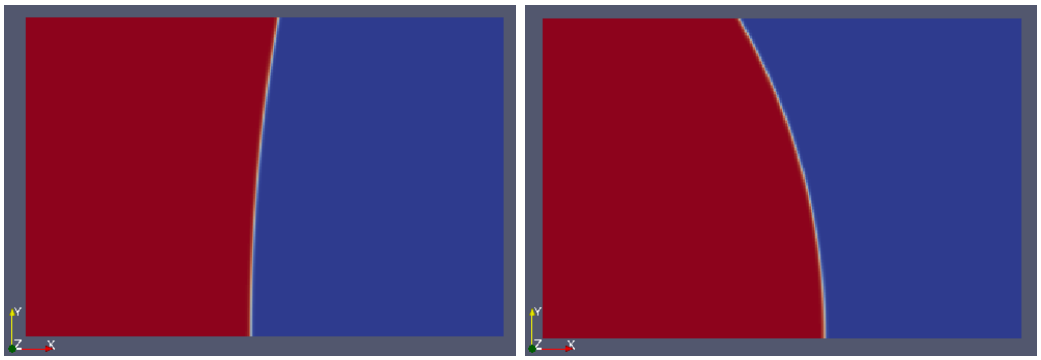


Abbildung 3.15: Meniskus mit vorgeschriebenem Kontaktwinkel von 80° (links) und 120° (rechts) nach 1,0 s unter Einfluss des Kompressionsterms.

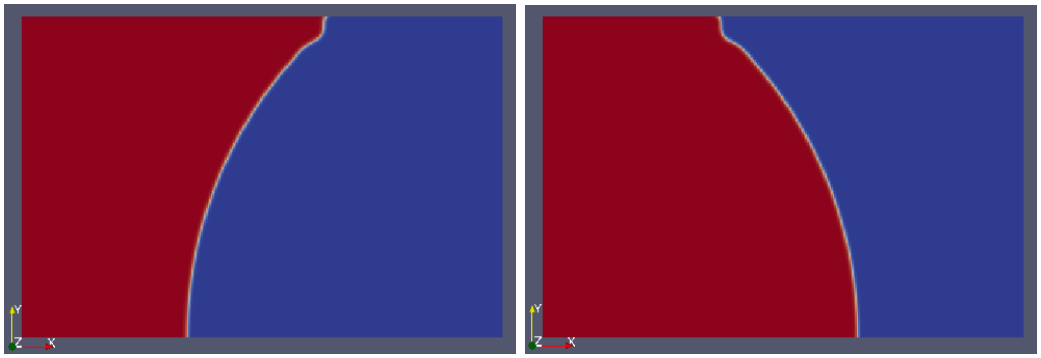


Abbildung 3.16: Meniskus mit vorgeschriebenem Kontaktwinkel von 40° (links) und 140° (rechts) nach 1,0 s unter Einfluss des Kompressionsterms.

durch ein Ungleichgewicht zwischen den angestrebten Gleichgewichtszuständen von Kompressionsterm und Druckfeld. Im Falle von 40° und 140° führt dies sogar zu einem Versagen des Modells. Ursache für dieses Ungleichgewicht ist der Modellfehler einer verschmierten Phasengrenzfläche. Eine verschmierte Phasengrenzfläche besitzt im dreidimensionalen Raum keinen linienförmigen Kontakt mit der Wand. Stattdessen kontaktiert die Grenzfläche, numerisch durch $|\vec{\nabla} c| \ggg 0$ charakterisiert, die Wand in einem ganzen Bereich. Laut Benetzungstheorie liegt genau an der Kontaktlinie der vorgegebene Kontaktwinkel θ_0 vor. Dies entspricht lediglich den Werten $c = 0$ bzw. $L = 0$. Für andere Werte von c und L kann aus der Benetzungstheorie keine Aussage über θ abgeleitet werden. Aus Abbildung 3.17 wird ersichtlich, dass die Annahme eines konstanten Kontaktwinkels nicht korrekt ist, da sich θ_1 und θ_2 deutlich voneinander unterscheiden. Dieses Phänomen resultiert einzig und allein aus der endlichen Dicke der Grenzfläche und würde mit einer Verfeinerung von ε verschwinden. Dies würde allerdings auch ein deutlich feineres Rechengitter und eine deutlich längere Rechenzeit mit sich bringen. Stattdessen wird versucht, den variierenden Kontaktwinkel zu modellieren und in der Randbedingung für c, L und $\vec{n}_{interface,wall}$ zu berücksichtigen. Für die Modellierung wird an der Kontaktstelle $L = 0$ ein lokales Koordinatensystem definiert, in dem nun $\vec{e}_y = \vec{n}_{wall}$ gilt und $\vec{e}_x$ durch Orthonormalisierung aus dem anliegenden $\vec{\nabla} L$ entsteht. Hiervon ausgehend wird angenommen, dass L sich in Wandnähe annähernd wie ein Ellipsoid, Paraboloid oder Hyperboloid verhält. Aus dieser Darstellung von L in Wandnähe resultieren zwei Hauptkrümmungsrichtungen mit jeweils zugehörigen Krümmungsradien R_i und

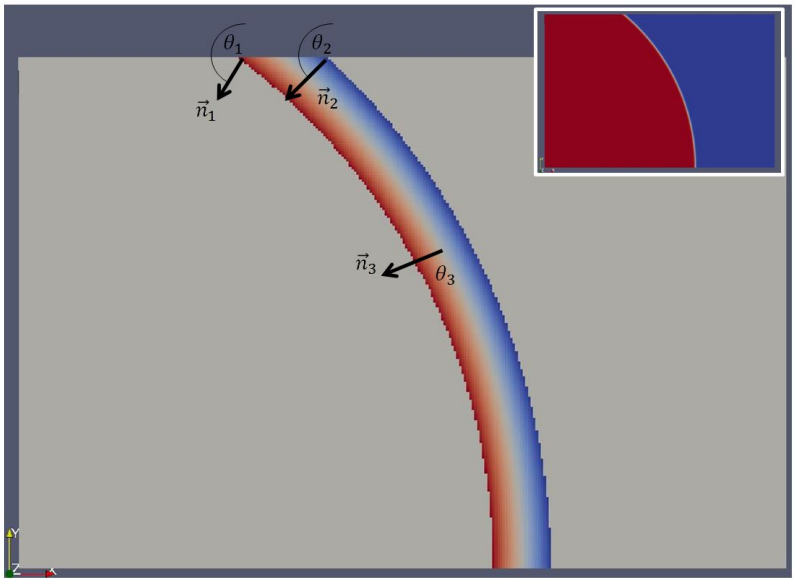


Abbildung 3.17: Darstellung unterschiedlicher Normalenvektoren einer verschmierten Grenzfläche der Dicke ε am Beispiel von L. Darstellung des zugehörigen c-Feldes im oberen rechten Bildausschnitt.

jeweiliger Hauptkrümmungsebene. Ist die hierbei betrachtete Umgebung hinreichend klein, kann davon ausgegangen werden, dass der Normalenvektor von einer der beiden Hauptkrümmungsebenen der Einheitsvektor $\vec{e}_z$ ist. Hieraus lässt sich ableiten, dass sich L in Wandnähe durch Gl. 3.69 beschreiben lässt.

$$L = a \cdot \frac{R - |\vec{x} - \vec{x}_0|}{\varepsilon} \tag{3.69}$$

R beschreibt dabei den zugehörigen Krümmungsradius und a gibt die Orientierung der Grenzfläche wieder ($a = \pm 1$). Der Vektor $\vec{x}_0$ beschreibt den Ursprungsort der Krümmung, welcher den bisherigen Überlegungen zur Folge ein Vielfaches von $\vec{n}_{interface,wall}$ sein muss und somit keine Komponente in z-Richtung enthalten kann. Aus der Definition des Koordinatenursprungs direkt in der Berührstelle $L = 0$ folgt direkt $R = \sqrt{{x_0}^2 + {y_0}^2}$. Durch Gl. 3.69 kann nun das Verhalten des Kontaktwinkels θ entlang der Wand analysiert werden. Die Werte von a und $\vec{x}_0$ werden dabei vorläufig als bekannt angesehen. Zunächst wird aus dem Verlauf von L (Gl. 3.69) $\vec{\nabla} L$ und daraus der Normalenvektor $\vec{n}_{interface}$ auf die Grenzfläche gebildet (Gl. 3.70).

$$\vec{n}_{interface} = -a \cdot \frac{1}{\sqrt{{x_0}^2 + {y_0}^2}} \cdot (\vec{x} - \vec{x}_0) \tag{3.70}$$

Der Kontaktwinkel θ steht mit den Normalenvektoren über Gl. 3.71 in Beziehung.

$$\cos\theta = \vec{n}_{interface} * \vec{n}_{wall} = -\frac{a \cdot (y - y_0)}{\sqrt{(x - x_0)^2 + (y - y_0)^2}} \tag{3.71}$$

Entlang der Wand (x-Richtung) können $\cos\theta$ und L linearisiert werden (Gl. 3.72 und Gl. 3.73) und ineinander eingesetzt werden, um den Verlauf von $\cos\theta$ zu erhalten, der nun unabhängig vom lokalen Koordinatensystem ist (Gl. 3.74).

$$L(x, y = 0) = \frac{1}{\varepsilon} \cdot \frac{a \cdot x_0}{\sqrt{{x_0}^2 + {y_0}^2}} \cdot x = \frac{\sin\theta_0}{\varepsilon} \cdot x \tag{3.72}$$

$$\cos\theta(x, y=0) = \cos\theta_0 + a \cdot \frac{x_0 \cdot y_0}{\sqrt{{x_0}^2 + {y_0}^2}^3} \cdot x \tag{3.73}$$

$$\cos\theta(x, y=0) = \cos\theta_0 + a \cdot \frac{\cos\theta_0 \cdot \sin\theta_0}{\sqrt{{x_0}^2 + {y_0}^2}} \cdot x = \cos\theta_0 \cdot \left(1 + a \cdot \frac{L \cdot \varepsilon}{R}\right) \tag{3.74}$$

Der lokale Kontaktwinkel θ lässt sich demnach an jeder Stelle entlang der Wand (x-Richtung) aus dem lokalen Wert von L und dem Kontaktwinkel θ_0 an der Berührstelle $L = 0$ berechnen. Unbekannt sind dabei noch die Werte a und R. Um diese zu gewinnen, wird die Ableitung von $\cos\theta$ in y-Richtung, also normal zur Wand, betrachtet, welche direkt an der Wand ($y = 0$) auf der einen Seite auf analytische Weise gewonnen werden kann (Gl. 3.75) und auf der anderen Seite durch eine Rückwärtsdifferenz in y-Richtung gebildet werden kann.

$$\left.\frac{\partial \cos\theta}{\partial y}\right|_{y=0} = -a \cdot \frac{(x - x_0)^2}{\sqrt{(x - x_0)^2 + (y_0)^2}^3} = -a \cdot \frac{\sin^2\theta}{\sqrt{(x - x_0)^2 + (y_0)^2}} \approx -\sin^2\theta \cdot \frac{a}{R} \tag{3.75}$$

Für diese Rückwärtsdifferenz wird an der Wand direkt der Kosinuswert des lokal vorliegenden Winkels θ verwendet. In der anliegenden Zelle wird der Kosinuswert aus dem dort numerisch berechneten Normalenvektor $\vec{n}_{interface,cell}$ und dem Normalenvektor der Wand $\vec{n}_{wall}$ gewonnen (Gl. 3.71). Ist aus der Rückwärtsdifferenz numerisch die Ableitung $\partial(\cos\theta)/\partial y$ gebildet, kann hierdurch der Term a/R berechnet und in Gl. 3.74 genutzt werden. Es fällt auf, dass hierbei ein Zirkelschluss entsteht, denn für die Berechnung des lokalen Wertes von θ wird der Wert a/R benötigt, welcher jedoch nur bei bekanntem Wert von θ über Gl. 3.75 zugänglich ist. Hinzu kommt die notwendige Kenntnis des lokalen L-Wertes an der Wand, welcher ebenfalls von θ abhängig ist. L wird direkt an der Wand sowohl in Gl. 3.74 als auch zur numerischen Berechnung von $\vec{n}_{interface,cell}$ über $\vec{\nabla}L$ benötigt. Der lokale Wert von θ ist daher nur iterativ zugänglich. Um eine Konvergenz der Gesamtiteration zur Bestimmung von L in der Wandzelle aus c zu gewährleisten, wird in den ersten drei Iterationen der Volumenintegration grundsätzlich $\theta = \theta_0$ verwendet. Ab $n_{Iter} > 3$ werden ausgehend von L, $\vec{\nabla}L$, $\vec{n}_{interface,wall}$ die Werte a/R und der nach Gl. 3.74 extrapolierte Wert von θ berechnet. Ziel der Iteration ist es, in jeder Wandzelle den bisherigen Wert von θ mit dem nach Gl. 3.74 extrapolierten Wert von θ in Übereinstimmung zu bringen. Hierfür wird das Newton-Verfahren verwendet. Die notwendige Ableitung der Zielfunktion wird durch Auslenkung von θ um einen kleinen Winkel $\mathrm{d}\alpha$ numerisch bestimmt. Zur Verbesserung der Konvergenz des Verfahrens werden die maximal möglichen Werte von a/R dabei limitiert. Sinnvoll haben sich hierbei $-1/(10 \cdot \Delta x)$ und $1/(10 \cdot \Delta x)$ als Begrenzungen für a/R erwiesen. Für Phasengrenzflächen, die eine noch stärkere Krümmung aufweisen, wird θ dadurch zwar unterschätzt, jedoch immer noch genauer dargestellt, als würde über die gesamte Wand ein konstanter Kontaktwinkel angenommen werden. Um den Kompressionsterm dabei numerisch stabil zu halten, dürfen die Werte von a/R sich in den Wandzellen jedoch nicht voneinander unterscheiden. Variable Werte von a/R in jeder Zelle führen zu Oszillationen im chemischen Potential, deren genaue Ursachen im Rahmen dieser Arbeit nicht näher untersucht worden sind. Anstatt in jeder Zelle einen eigenen Wert a/R zu bestimmen werden zunächst die Paare an Wandzellen identifiziert, zwischen denen L einen Vorzeichenwechsel ($L_1 \cdot L_2 < 0$) besitzt. Für die Wandflächen dieser Zellpaare wird jeweils a/R nach beschriebenem iterativen Verfahren bestimmt. Alle anderen Zellflächen adaptieren die so bestimmten Werte für a/R, indem sie nach

einem Extrapolationsalgorithmus ihr zugehöriges Zellpaar identifizieren und dessen Wert für a/R übernehmen. Dazu wird der aktuell lokal vorliegenden Wert von θ (aus vorherigen Iterationen bestimmt oder θ_0 zur Initialisierung) genutzt und hierüber $\vec{\nabla}L$ in der anliegenden Zelle berechnet. Da L sich annähernd linear verhält, lässt sich über L und $\vec{\nabla}L$ in einer Wandzelle entlang der Wand sehr genau die Stelle $L = 0$ extrapolieren. Aus der Zelle, die nach dieser Extrapolation die Stelle $L = 0$ beinhaltet, wird der Wert a/R übernommen und über Gl. 3.74 der lokale Wert von θ berechnet. Liegt der extrapolierte Punkt $L = 0$ nicht innerhalb von einem der genannten Zellpaare, wird von diesem Extrapolationspunkt der gleiche Algorithmus erneut durchgeführt. Wird daraufhin immer noch kein Zellpaar getroffen, wird für die Startzelle eine Wert $a/R = 0$ vorgeschrieben.[20] Dies trifft im Allgemeinen nur auf Zellen zu, die sehr weit von der Berührstelle $L = 0$ entfernt sind und daher auch mit nicht korrigiertem Kontaktwinkel θ kaum einen Einfluss auf Kompressionsterm und Drucksprung direkt an der Grenzfläche besitzen.

Aus Abbildung 3.18 wird der Einfluss eines variablen Kontaktwinkels auf die Berechnung der Krümmung κ in den wandnahen Zellen deutlich. Ohne Annahme

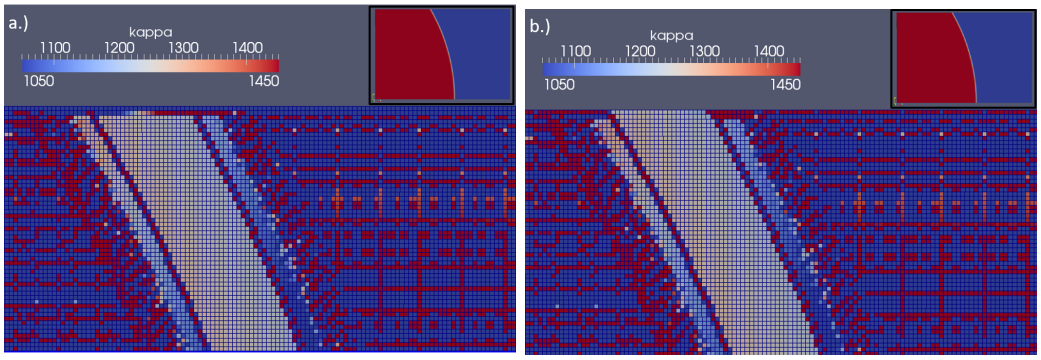


Abbildung 3.18: Numerisch berechnete Krümmung κ unter Annahme eines konstanten Kontaktwinkels (a) und eines variablen Kontaktwinkels entlang der Wand (b) bei 120° Sollwert an der Stelle $L = 0$. Der resultierender Sollwert der Krümmung κ beträgt 1250 1/m.

eines variablen Kontaktwinkels werden ungenaue Krümmungswerte berechnet, welche sich auf die Berechnung des Druckfeldes und den Kompressionsterm auswirken. Lediglich unter Annahme eines variablen Kontaktwinkels entlang der Wand werden auch in den wandnahen Zellen die korrekten Werte in der Größenordnung von 1250 1/m berechnet. Für den direkten Vergleich sind die in Abbildung 3.15 und 3.16 dargestellten Rechnungen daher erneut unter Nutzung eines variablen Kontaktwinkels durchgeführt worden. Das Phänomen der durch den Kompressionsterm eingebrachten Deformationen (vgl. Abbildung 3.16) tritt hierbei nicht mehr auf. Im Bezug auf die parasitären Strömungen hat sich gezeigt, dass sich die Paare an Kontaktwinkeln, die sich jeweils zu 180° ergänzen, in jedem Zeitschritt nur um maximal $0,8\%$ unterscheiden, weshalb nur 40°, 60° und 80° dargestellt werden. Die Entwicklungen der parasitären Strömungen im Zeitraum $1\,\mathrm{s} < t < 2\,\mathrm{s}$ in Abbildung 3.19 zeigen, dass die Rechnungen noch nicht komplett auskonvergiert sind, aber dennoch einen guten Vergleich untereinander zulassen. Zu beachten ist die logarithmische Skalierung. Lediglich unter Annahme eines variablen Kontaktwinkels

[20] Ein Wert von $a/R = 0$ bewirkt effektiv, dass $\theta = \theta_0$ gesetzt wird. In diesem seltenem Fall fällt das Modell daher lediglich auf den bisherigen Stand der Technik zurück.

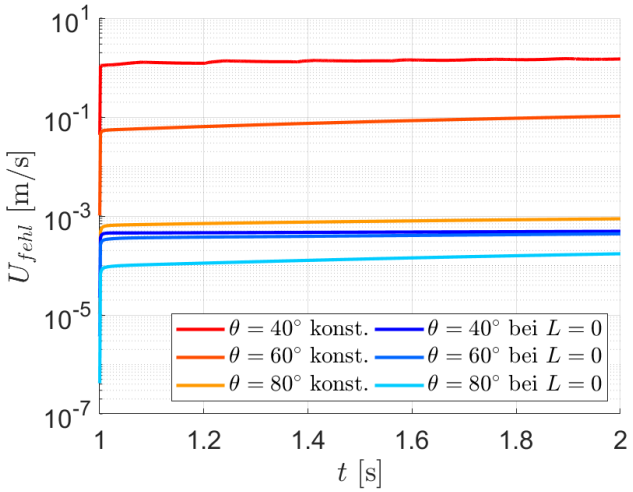


Abbildung 3.19: Verlauf der parasitären Strömungen U_{fehl} am liegenden Tropfen ohne konvektivem Transport der Grenzfläche bei verschiedenen Kontaktwinkeln und unterschiedlichen Ansätzen zu dessen Implementierung.

liegen die parasitären Strömungen in einer akzeptablen Größenordnung, da in Mikrokanälen selten Geschwindigkeiten größer als 0,1 m/s erreicht werden. Abschließend wurde in allen Konfigurationen ab einer Zeit von 2,0 s die Phasengrenzfläche konvektiv transportiert und die parasitäre Strömung U_{fehl} ermittelt. Hierbei wurden jeweils 4 Kompressionsschritte pro Zeitschritt mit $D = D_{max}$ angesetzt. Die Ergebnisse werden durch Tabelle 3.13 verdeutlicht. Wird ein konstanter Kontaktwinkel

Tabelle 3.13: Vergleich der auftretenden parasitäre Strömungen am liegenden Tropfen bei verschiedenen vorgegebenen Kontaktwinkeln und Nutzung eines konstanten Kontaktwinkels an der Wand (oben) und eines variablen Kontaktwinkels entlang der Wand (unten) mit Zuschalten des konvektiven Grenzflächentransportes ab $t = 2\,\mathrm{s}$.

$\theta = \text{const}$	$U_{fehl}(t = 2s)$	$U_{fehl}(t > 2.0s)$	$U_{fehl}(t > 2.0s, n_{Iter} = 5)$
40°	$1,50 \cdot 10^{0}$ m/s	$3{,}31 \cdot 10^{-3}$ m/s	$3{,}37 \cdot 10^{-3}$ m/s
60° / 120°	$1{,}05 \cdot 10^{-1}$ m/s	$2{,}10 \cdot 10^{-3}$ m/s	$2{,}11 \cdot 10^{-3}$ m/s
80° / 100°	$8{,}97 \cdot 10^{-4}$ m/s	$1{,}36 \cdot 10^{-5}$ m/s	$1{,}36 \cdot 10^{-5}$ m/s
140°	$1,50 \cdot 10^{0}$ m/s	$5{,}23 \cdot 10^{-4}$ m/s	$3{,}28 \cdot 10^{-3}$ m/s

$\theta = \text{var}$	$U_{fehl}(t = 2s)$	$U_{fehl}(t > 2.0s)$	$U_{fehl}(t > 2.0s, n_{Iter} = 5)$
40° / 140°	$4{,}91 \cdot 10^{-4}$ m/s	$6{,}12 \cdot 10^{-6}$ m/s	$9{,}86 \cdot 10^{-5}$ m/s
60° / 120°	$4{,}36 \cdot 10^{-4}$ m/s	$1{,}83 \cdot 10^{-6}$ m/s	$8{,}41 \cdot 10^{-5}$ m/s
80° / 100°	$1{,}72 \cdot 10^{-4}$ m/s	$8{,}00 \cdot 10^{-8}$ m/s	$7{,}06 \cdot 10^{-7}$ m/s

angesetzt, so liegt die parasitäre Strömung im Vergleich zur Anwendung eines variablen Kontaktwinkels deutlich höher und weicht sogar vom bisher symmetrischen Verhalten zwischen den Kontaktwinkeln 40° und 140° ab. Zuletzt wurde der Einfluss der Iterationszahl n_{Iter} untersucht. Es wird beobachtet, dass die parasitären Strömungen mit einem Absenken der Iterationszahl n_{Iter} stetig ansteigen, während beim

Transport eines Tropfens ohne Wandkontakt in freier Strömung keine Abhängigkeit von n_{Iter} beobachtet wurde (siehe Unterabschnitt 3.6.2). Die Bestimmung des variablen Kontaktwinkels entlang der Wand bedarf demnach eine gewisse Mindestzahl an Iterationen. Ein Unterschreiten von $n_{Iter} = 5$ wird wegen des starken Anstiegs von U_{fehl} daher nicht in Betracht gezogen.

3.8 Darstellung und Validierung der Gesamtmethode

In diesem Abschnitt soll die resultierende Gesamtmethode inklusive der Kopplung mit der Kräftebilanz dargestellt und an literaturbekannten experimentellen Daten validiert werden. Darüber hinaus erfolgt eine Diskussion, in welchem Maße der eingeführte Kompressionsterm die physikalische Genauigkeit des Models beeinträchtigt.

Zu Beginn jedes Zeitschrittes wird der konvektive Transport der Indikatorfunktion $\bar{c}$ mit einem expliziten Runge-Kutta-Verfahren in der Zeit nach Gl. 3.76 durchgeführt.

$$\bar{c}^{*,n+1} = \bar{c}^n + (k_1 + 2 \cdot k_2 + 2 \cdot k_3 + k_4)/6 \tag{3.76}$$

mit

$$\begin{aligned} k_1 &= -\Delta t \cdot \vec{\nabla} * (\vec{u} \cdot \bar{c}^n) \\ k_2 &= -\Delta t \cdot \vec{\nabla} * (\vec{u} \cdot (\bar{c}^n + k_1/2\ )) \\ k_3 &= -\Delta t \cdot \vec{\nabla} * (\vec{u} \cdot (\bar{c}^n + k_2/2)) \\ k_4 &= -\Delta t \cdot \vec{\nabla} * (\vec{u} \cdot (\bar{c}^n + k_3\ )) \end{aligned}$$

Die Auswertungen $\vec{\nabla} * (\vec{u} \cdot \bar{c}_i)$ sind dabei stets nach der Finite-Volumen-Methode zu verstehen. Die notwendigen Flächenintegrale werden nach dem in Abschnitt 3.4 beschriebenem Verfahren berechnet. Hierzu ist stets die Rekonstruktion von L aus dem aktuell zu verwendenden $\bar{c}_i$ notwendig. Die Geschwindigkeiten werden innerhalb des Zeitschrittes als konstant angesehen und einmalig auf die Zellwände interpoliert. Im Anschluss an den konvektiven Transport wird der Kompressionsterm auf $\bar{c^*}^{n+1}$ angewendet, um numerische Fehler zu glätten, die trotz Verwendung eines Runge-Kutta-Verfahrens hoher Ordnung und genauer Berechnung der Oberflächenintegrale beim Transport entstanden sind. Hierzu wird Gl. 3.77 gelöst.

$$\bar{c^{**}}^{n+1} = \bar{c^*}^{n+1} + \Delta t \cdot D \cdot \Delta\Phi(\bar{c^*}^{n+1}) \tag{3.77}$$

Diese Gleichung kann nur explizit gelöst werden, was Ursache für Stabilitätsbedingung nach Gl. 3.53 ist und den Kompressionkoeffizienten D in seinem Maximalwert limitiert. Um dennoch eine ausreichend hohe Kompressionsstärke wählen zu können, wird Gl. 3.77 bei Bedarf mehrfach hintereinander gelöst und jeweils das gerade erhaltene $\bar{c}^{**,n+1}$ zur erneuten Berechnung von Φ verwendet. Da der Kompressionsprozess nur auf einer Pseudozeitskala stattfindet, entspricht laut Modellvorstellung beispielhaft das zehnfache Lösen mit kleinem Wert D_1 dem einmaligem Lösen mit dem zehnfachen Wert $D_2 = 10 \cdot D_1$, das jedoch aus Stabilitätsgründen nicht möglich ist. Jede Berechnung von Φ erfordert dabei eine Rekonstruktion von L.

Nach Abschluss des Kompressionsschrittes wird mit dem finalen $\bar{c^{**}}^{n+1}$ erneut L rekonstruiert, um die Krümmung κ für die Berechnung des Kapillardrucks in der Kräftebilanz zu erhalten. Hierzu wird zwischen zwei Zellzentren ein linearer Verlauf von L angenommen. Der Ort $L = 0$ repräsentiert laut Modellvorstellung die Position der Phasengrenzfläche. An dieser Koordinate wird mittles linearer Interpolation die Krümmung der Grenzfläche aus den anliegenden Zellzentren berechnet. Der Drucksprung zwischen den beiden Zellzentren ergibt sich dann nach Gl. 3.78.

$$\Delta p_{cap} = \kappa_{interpol.} \cdot \sigma \tag{3.78}$$

Ziel ist nach Francois et al. ([84]) die exakt gleiche Behandlung von Kapillardruck und hydrodynamischem Druck. Letzterer wird nach der Finite-Volumen-Methode jeweils auf der Zellwand als Gradient in Normalenrichtung benötigt. Dementsprechend wird auch der Kapillardruck als Normalengradient auf der Zellwand nach Gl. 3.79 formuliert. Dabei ist exakt die gleiche Diskretisierung zu verwenden, die auch für den hydrodynamischen Druck genutzt wird und demnach auch Korrekturen für nicht orthogonale Zellwände berücksichtigen muss (in Gl. 3.79 nicht enthalten)

$$\vec{n}_{face} * \vec{\nabla} p_{cap} = \mathrm{snGrad}(p_{cap}) = \kappa_{interpol.} \cdot \sigma \cdot \frac{1}{\Delta x} \tag{3.79}$$

Zu beachten ist an dieser Stelle, dass die lokale Krümmung κ der Grenzfläche nicht auf die Zellwand interpoliert wird, sondern an die nach $L = 0$ berechnete Position der Grenzfläche. Die Kopplung aus Kräftebilanz und Kontinuitätsgleichung erfolgt nach dem von Issa dargestellten PISO-Algorithmus. [104] Hierbei wird zunächst durch die Kräftebilanz eine neue Geschwindigkeit $\vec{u}$ berechnet. Der hierfür benötigte vektorielle Druckgradient $\vec{\nabla} p$ wird aus den negativen Normalengradienten auf den Zellwänden rekonstruiert, nachdem dort nach Gl. 3.79 der Kapillardruck hinzuaddiert wurde. Die Stoffwerte berechnen sich nach Gl. 3.80 und Gl. 3.81.

$$\rho = \rho_1 + (\rho_2 - \rho_1) \cdot \frac{1 + \min(1, \max(-1, \bar{c}))}{2} \tag{3.80}$$

$$\eta = \eta_1 + (\eta_2 - \eta_1) \cdot \frac{1 + \min(1, \max(-1, \bar{c}))}{2} \tag{3.81}$$

Die Begrenzung von $\bar{c}$ auf Werte zwischen -1 und 1 schließt dabei die Entstehung von negativen Werte für Dichte und Viskosität grundsätzlich aus. Die neue Geschwindigkeit $\vec{u}$ erzeugt dabei Flüsse auf den Zellwände, welche nach Addition der Flüsse durch Kapillarkräfte divergenzfrei sein müssen. Um dies zu gewährleisten wird eine Laplace-Gleichung für den neuen Druck gelöst und der jeweilige Fluss auf jeder Zellwand korrigiert. Werden Kräftebilanz und die aus der Massenerhaltung hergeleitete Druckgleichung iterativ im Wechsel gelöst, so sinkt der Iterationsfehler mit jedem Durchlauf nach $\mathcal{O}(\Delta t)$.[21] Wie viele Iterationsschleifen genutzt werden sollten hängt dabei von der Größe des erwarteten zeitlichen Diskretisierungsfehlers ab. [104] Große Teile dieses Kopplungsverfahren konnten aus dem numerischen Code von interFoam adaptiert werden. [105] Für tatsächlich konvektiv transportierte Grenzflächen werden in dieser Arbeit stets zwei Korrekturschritte zwischen Kräftebilanz und Kontinuitätsgleichung durchgeführt, was pro Zeitschritt dem Lösen von

[21] Nach der Berechnung von $\vec{u}$ ist zwar die Kräftebilanz, aber allgemein nicht die Kontinuität erfüllt. Erst die Berechnung eines neuen Drucks mit Korrektur der Flüsse garantiert die Massenerhaltung. Durch den veränderten Druck ist nun jedoch die Kräftebilanz wieder verletzt.

drei Laplace-Gleichungen für den Druck entspricht. Die Kräftebilanz wird nach dem PBiCG-Algorithmus mindestens auf ein Restresiduum von 10^{-6} gelöst, während das Residuum jeder zu lösenden Druckgleichung mit Hilfe eines GAMG-Algorithmus um mindestens 3 Größenordnungen gesenkt wird. Teil der Validierung ist zunächst die Betrachtung des Kompressionsterms. Die Diskussion von unterschiedlichen Gleichgewichtszuständen in Abschnitt 3.6 wirft die Frage auf, in welchem Maße der Kompressionsterm eine unphysikalische Bewegung der Phasengrenzfläche hervorruft. Da die ursprüngliche Formulierung des Phase-Field-Modells auf der Energieminimierung des Gesamtsystems beruht, ist z.B. ein Quader im ursprünglichen Phase-Field-Modell allein durch den Diffusionsprozess bereits dazu angetrieben, die Kugelform anzunehmen.[22] Auch das Phänomen von schrumpfenden Tropfen und der stattfindenden Übersättigung im Tropfeninneren ($\bar{c} > 1$) ist literaturbekannt. [98] Das Schrumpfen eines Tropfens um ΔR wird begünstigt, da kleinere Tropfen weniger ernegieenthaltende Oberfläche besitzen. Limitiert wird das Schrumpfen um ΔR lediglich durch den Energieaufwand zur Übersättigung/Untersättigung der Bulkphasen um Δc. Der hierfür notwendige Energieaufwand steigt jedoch nur langsam mit Δc an, sodass durch das Schrumpfen bis zu einem gewissen Punkt dennoch Gesamtenergie freigesetzt wird. Dem Auftreten beider Phänomenen steht gegenüber, dass in dieser Arbeit nicht die ursprüngliche Phase-Field-Gleichung verwendet wird, sondern ein modifizierter Laplace-Operator, der lediglich die Krümmung der Grenzfläche in Normalenrichtung berücksichtigt (siehe Gl. 3.18). Dies unterstreicht die Notwendigkeit einer gesonderten Betrachtung, welche an der gleichen Tropfenrelaxation durchgeführt wird, die in Abschnitt 3.6.1 bereits zur Validierung der Codeparallelisierung diente. Die Simulation des Relaxationsverhaltens wird einmal nach der hier beschriebenen Methode und einmal lediglich durch den Diffusionsprozess angetrieben. Die Unterschiede der Grenzflächenpositionen nach 0,01 s sind in Abbildung 3.20 dargestellt. Es ist zu erkennen, dass der Diffusionsprozess des Kom-

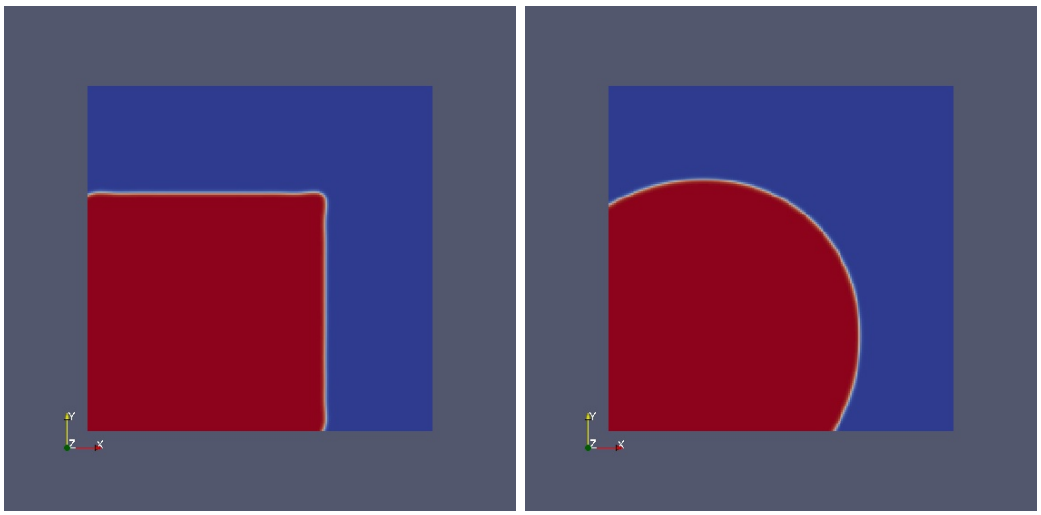


Abbildung 3.20: Vergleich der Grenzflächenposition nach 0,01 s durch reine Kompression (links) und durch Kompression und gleichzeitige Advektion (rechts).

[22] Ein Quader besitzt im Vergleich zur Kugel mehr energieenthaltende Oberfläche, sodass die Gesamtsystemenergie erst bei Erreichen der Kugelform ein lokales Minimum annimmt.

Abbildung 3.21: Zeitliche Entwicklung der Grenzflächenposition zu verschiedenen Zeitpunkten durch reine Kompression (oben) und durch Kompression und gleichzeitige Advektion (unten).

pressionsterms die geforderten Kontaktwinkel an den anliegenden Wänden bildet und den scharfen Knick des initialisierten Quadrates nach 0,01 s leicht verrundet. Die Advektion der Grenzfläche durch das entstehende Geschwindigkeitsfeld hingegen bringt den Tropfen bereits nach kurzer Zeit in seine angestrebte Kugelform. Die in Abbildung 3.21 dargestellte zeitliche Entwicklung der Grenzflächenposition verdeutlicht dies noch einmal. Bereits nach einem Hundertstel der Gesamtsimulationszeit ($t = 0{,}0001\,\mathrm{s}$) ruft der advektive Transport gut sichtbare Wellen in der Tropfenkontur hervor, während der Kompressionsterm lediglich die im Gleichgewichtszustand durch ε definierte Dicke der Grenzfläche einstellt. Die Übersättigung im Tropfeninneren ist mit $1,0000309$ als Maximalwert im Tropfeninneren und $-1,0000971$ als Minimalwert im Tropfenaußenraum vernachlässigbar klein und wird eher auf Diskretisierungsfehler zurückgeführt, da der Effekt der Übersättigung im Inneren auch eine globale Untersättigung mit $\bar{c} > -1$ im Außenbereich herbeiführen würde. Dies wurde jedoch nicht beobachtet. Der in Abbildung 3.21 durchgeführte Vergleich der Grenzflächenpositionen ist soweit rein qualitativ, zeigt jedoch bereits gut, dass die Position der Phasengrenzfläche fast ausschließlich durch die Advektion angetrieben wird und der Kompressionsterm lediglich die lokale Form der Grenzfläche beeinflusst.

Zur quantitativen Validierung der Genauigkeit der Gesamtmethode werden simulativ bestimmte Druckverluste und Wandfilmdicken einer flüssig-flüssig Pfropfenströmung mit experimentell bestimmten Werten aus der Literatur verglichen. Zum Vergleich der Wandfilmdicken wird die empirische Korrelation nach Eain et al. herangezogen, welche in Gl. 3.82 dargestellt ist. [106]

$$\frac{h}{R} = \frac{1,34 \cdot Ca^{2/3}}{1 + 1,34 \cdot (1,6 \cdot Ca)^{2/3}} \tag{3.82}$$

Die Kapillar-Zahl wird hierbei mit der Pfropfengeschwindigkeit u_{slug} und der Viskosität der kontinuierlichen Phase gebildet. Die Validierung der Druckverluste erfolgt mit der Korrelation von Jovanovic et al. nach Gl. 3.83. Beide Autoren geben einer relative Genauigkeit von ≈10% an.

$$\frac{\Delta p}{L} = 8 \cdot u_{slug} \cdot \left(\frac{\eta_{disp} \cdot \alpha}{(R-h)^2} + \frac{\eta_{cont} \cdot (1-\alpha)}{R^2} \right) + \frac{C}{l_u} \cdot (3 \cdot Ca)^{2/3} \cdot \frac{\sigma}{2 \cdot R} \tag{3.83}$$

Die hierin enthaltenen geometrischen Parameter werden durch Abbildung 3.22 erläutert. Der Parameter C beschreibt den Einfluss der Pfropfenkappen auf den Druckverlust und variiert bei Jovanovic et al. in Abhängigkeit des Innendurchmessers, sodass für einen Innendurchmesser von $248\,\mu$m ein Wert von $7,16$ zu verwenden ist, während die Druckverluste in Kapillaren mit $498\,\mu$m Innendurchmesser durch $C = 3,48$ besser beschrieben werden. [107] Die Wandfilmdicke h zur Berechnung des Druckverlustes ist durch Messungen, Korrelationen oder aus der Simulation heraus zu bestimmen. In allen Messungen ist die wässrige Phase die kontinuierliche Phase. Zur Validierung werden in einem Kapillarauschnitt von $500\,\mu$m Länge bei einem Kapillarradius von $125\,\mu$m die Experimente mit Toluol und einem Phasenverhältnis von 1 im Geschwindigkeitsbereich von $0{,}07\,\mathrm{m/s}$ bis $0{,}12\,\mathrm{m/s}$ nachgestellt. Die in der Simulation verwendeten Stoffdaten sind in Tabelle 3.14 aufgelistet. Gewählt wird eine Gitterweite von $\Delta x = 0{,}625\,\mu$m, was einer Auflösung von 200 Zellen in radialer Richtung entspricht, und aus Stabilitätsgründen eine Zeitschrittweite von

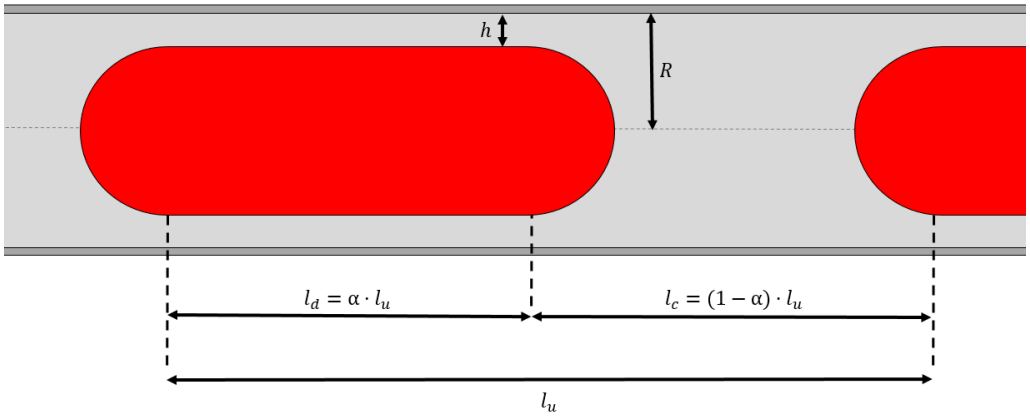


Abbildung 3.22: Definition der relevanten geometrischen Parameter einer Pfropfenströmung. Nachgestellt aus [107].

Tabelle 3.14: Stoffwerte des betrachteten Stoffsystems zur Simulation des Druckverlustes. Entnommen aus [107].

Komponente	ρ	η	σ
VE-Wasser	$998{,}2\,\mathrm{kg/m^3}$	$1{,}0\,\mathrm{mPa \cdot s}$	$0{,}0371\,\mathrm{N/m}$
Toluol	$866{,}7\,\mathrm{kg/m^3}$	$0{,}59\,\mathrm{mPa \cdot s}$	-

$\Delta t = 7 \cdot 10^{-8}\,\mathrm{s}$. Pro Zeitschritt werden 4 Kompressionsschritte der Grenzfläche mit $D = D_{max}$ durchgeführt. Gewählt wird ein mitbewegter Beobachter, wodurch die äußere Kapillarwand sich mit der Geschwindigkeit u_{slug} zur Seite bewegt. Zwischen Ein- und Auslass des Rechengebietes werden zyklische Randbedingungen angelegt, während der Druck hingegen mit einem Sprung beaufschlagt wird. Je größer die Auslenkung des Pfropfens aus der Gebietsmitte ist, desto größer wählt ein programmierter PD-Regler den angelegten Drucksprung, bis der Pfropfen schließlich im mitbewegten System eine stationäre Position erreicht (P-Anteil). Der D-Anteil wurde hinzugefügt, um Oszillationen zu dämpfen. Beide Regelkonstanten wurden iterativ bestimmt. Die Gitterunabhängigkeit der Lösung wird am Beispiel einer Pfropfengeschwindigkeit von $0{,}09\,\mathrm{m/s}$ untersucht. Hierfür wird die stationäre Lösung des Rechengitters mit $\Delta x = 0{,}625\,\mu\mathrm{m}$ auf verfeinerte Gitter mit $\Delta x = 0{,}5\,\mu\mathrm{m}$ und $\Delta x = 0{,}3125\,\mu\mathrm{m}$ interpoliert und erneut bis in den stationären Zustand gelöst. Der Parameter ε wird dabei so angepasst, dass die relative Grenzflächendicke $\Delta x/\varepsilon$ konstant bleibt. Über die direkte Abbildung der Experimente von Jovanovic et al. ([107]) hinaus werden weitere fiktive Konstellationen an Betriebspunkten in einer Kapillare von $1000\,\mu\mathrm{m}$ Länge bei gleichem Radius von $125\,\mu\mathrm{m}$ durch Permutation der Parameter in Tabelle 3.15 berechnet. Zuletzt werden noch Fälle betrachtet, in denen die Viskosität der dispersen Phase von $0{,}001\,\mathrm{Pa \cdot s}$ bis $0{,}015\,\mathrm{Pa \cdot s}$ variiert wird, während die restlichen Parameter mit $u_{slug} = 0{,}11\,\mathrm{m/s}$, $\rho_{cont} = \rho_{disp} = 1000\,\mathrm{kg/m^3}$, $\eta_{cont} = 0{,}001\,\mathrm{Pa \cdot s}$ und $\sigma = 0{,}035\,\mathrm{N/m}$ konstant bleiben. Für den Abgleich der bestimmten Druckverluste sind geometrische Daten nach Abbildung 3.22 aus der Pfropfenkontur zu extrahieren. Die Pfropfenkontur wird jeweils mit der Stelle $L = 0$ zwischen zwei Zellen identifiziert. Die Wandfilmdicke h und der Pfropfenradius R_{slug} variieren jedoch über die Pfropfenlänge. Als repräsentativ wird die Wandfilmdicke

Tabelle 3.15: Permutationsparameter zur Validierung der numerischen Methode gegen die empirische Korrelation nach Jovanovic et al. ([107]) in einer Kapillare von 1000 μm Länge und 125 μm Radius.

u_{slug}		$\rho_{cont} = \rho_{disp}$	$\eta_{cont} = \eta_{disp}$	σ
0,06 m/s	0,10 m/s	1000 kg/m^3	0,001 Pa · s	0,015 N/m
0,07 m/s	0,11 m/s		0,002 Pa · s	0,025 N/m
0,08 m/s	0,12 m/s		0,003 Pa · s	0,035 N/m
0,09 m/s				

auf exakt halber Länge zwischen vorderer und hinterer Pfropfenkappe angesehen. Die Länge l_d des angenommenen Zylinderkörpers zwischen den Kugelkappen zur korrekten Anwendung von Gl. 3.83 ergibt sich zu $l_d = L_{slug} - 2 \cdot R_{slug}$. Es wurde beobachtet, dass bei der gewählten Auflösung von $\Delta x = 0{,}625\,\mu$m in allen Simulationen mit einer Kapillar-Zahl Ca, gebildet mit der Wandgeschwindigkeit und der Viskosität der kontinuierlichen Phase, von Ca $\leq 1,8 \cdot 10^{-3}$ der Pfropfen Wandkontakt ausbildet. Dieses Verhalten ist unphysikalisch, da experimentell bereits Wandfilme bei kleineren Kapillar-Zahlen nachgewiesen wurden. Dies stellt die erste Gültigkeitseinschränkung des numerischen Modells dar. Für alle anderen Fälle sind die Paritätsplot der bestimmten Druckverluste und Wandfilmdicken in Abbildung 3.23 dargestellt. Abgesehen von den Simulationen mit $Ca \leq 1,8 \cdot 10^{-3}$ weichen die si-

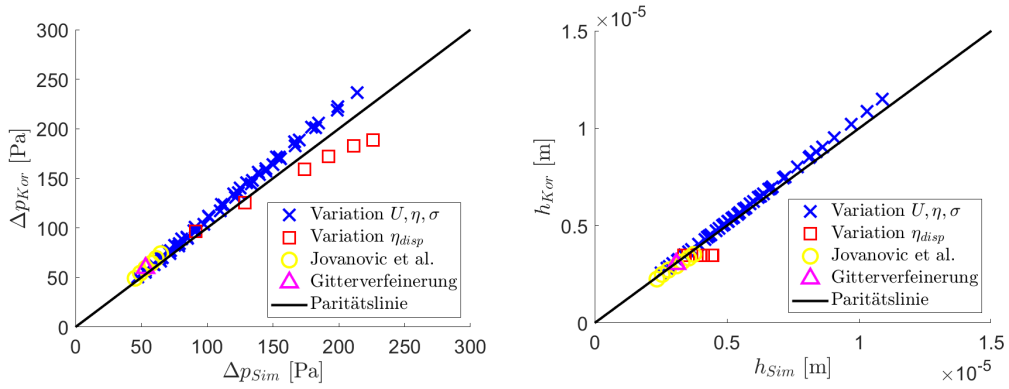


Abbildung 3.23: Paritätsplot der simulativ bestimmten Druckverluste (links) und Wandfilmdicken (rechts) mit den empirischen Korrelation nach Jovanovic et al. ([107]) und Eain et al. ([106]): fiktive Konstellationen nach Tabelle 3.15 (blaue Kreuze), weiterführende Variation des Viskositätsverhältnisses (rote Quadrate), direkte Nachbildung der Experimente von Jovanovic et al. (([107]), gelbe Kreise), Untersuchungen zur Gitterunabhängigkeit (violette Dreiecke).

mulativ bestimmten Wandfilmdicken für die Konstellationen nach Tabelle 3.15, die nachgestellten Fälle nach Jovanovic et al. ([107]) und die Untersuchungen zur Gitterunabhängigkeit im Mittel um 3,6% und maximal um 6,8% von der Korrelation ab. Bei den Druckverlusten betragen die Abweichungen in den genannten Fällen im Mittel 8,4% und maximal 12,6%. Zudem stimmt für die Druckverluste die qualitative Richtung der Abweichung in soweit, dass für Geschwindigkeiten größer als 0,06 m/s die Korrelation tendenziell einen zu großen Druckverlust voraussagt (vgl.

Abbildung 12 in [107]). Die Übereinstimmung mit den eigentlich durchgeführten Experimenten ist daher sogar besser als mit der Korrelation. Die Variation von η_{disp} bei sonst konstanten Parametern zeigt hingegen deutlich größere Abweichungen. Die Wandfilmdicke sollte unberührt von η_{disp} konstant bleiben, wächst jedoch mit zunehmender Viskosität an. Auch die Druckverluste, deren Berechnung von den Wandfilmdicken abhängt, zeigen markante Abweichungen von der Vorhersage. Der Grund hierfür wird in der begrenzten räumlichen Auflösung gesehen. Betrachtet man die Zahl an Zellen, welche den Wandfilmbereich auflösen, so sind dies während der Variation von η_{disp} lediglich fünf bis sieben Zellen. Im Vergleich dazu nimmt der Übergangsbereich, innerhalb dessen die Viskosität den Übergang von η_{disp} auf η_{cont} vollzieht, einen Bereich in der Größenordnung von ε und damit von ein bis zwei Zellen ein. Innerhalb des Wandfilmbereichs liegt somit nicht strikt die Viskosität η_{cont} vor, sondern zu gewissen Teilen auch eine abweichende Mischviskosität. Dies führt zur Ausbildung eines dickeren Wandfilms. Der Vergleich von Wandfilmdicken und Druckverlusten validiert somit die numerische Methode bei der gewählten Auflösung von 200 Zellen in radialer Richtung für Kapillar-Zahlen $Ca > 1,8 \cdot 10^{-3}$ und Viskositätsverhältnisse nahe 1. Kleinere Kapillar-Zahlen und stark unterschiedliche Viskositäten erfordern hingegen feinere Rechengitter.

3.9 Vergleich der Methode mit dem Standardsolver interFoam

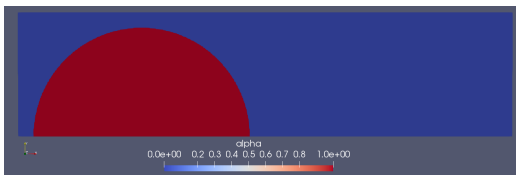
Die in diesem Kapitel entwickelte numerische Methode soll zur Evaluierung der Performance mit dem Standardsolver von OpenFOAM 6 für Mehrphasenströmungen interFoam verglichen werden. Hierbei wird der Faktor der parasitären Strömungen als primäres Vergleichskriterium gesetzt. Mit Hilfe von interFoam wird das Standardbeispiel berechnet, an dem in Abschnitt 3.6 auch die parasitären Strömungen der entwickelten Methode charakterisiert worden sind.[23] Die Parameter des Solvers interFoam werden dabei an die von OpenFOAM selbst empfohlenen Einstellungen für die Tutorialfälle *capillaryRise* und *damBreak* angelehnt und können in Anhang D eingesehen werden. Variiert werden die Transportgeschwindigkeit U_0 und die Kompressionsstärke des Solvers über den Wert c_α durch Parametervariation nach Tabelle 3.16, wobei $c_\alpha = 1$ die von OpenFOAM empfohlene Einstellung ist. Abbildung 3.24

Tabelle 3.16: Satz an Permutationsparametern zur Charakterisierung der parasitären Strömungen am konvektiv transportiertem Tropfen im Solver interFoam.

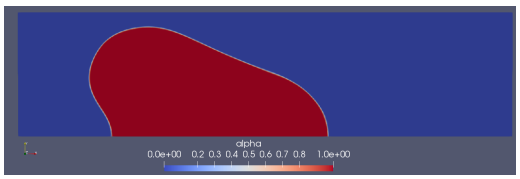
U_0		c_α	
0,000 m/s	0,020 m/s	0, 00	0, 75
0,005 m/s	0,040 m/s	0, 25	1, 00
0,010 m/s	0,080 m/s	0, 50	1, 25

[23] Das Standardbeispiel besteht aus dem Transport einer Tropfens mit 0,7 mm Radius in einer Kapillare mit 1,6 mm Durchmesser über eine Distanz von 1,6 mm mit jeweils variabler Transportgeschwindigkeit U_0 bei einer Gitterweite $\Delta x = 4 \cdot 10^{-6}$ m. Die Stoffdaten beider Phasen sind identisch und betragen $\rho = 1000\,\mathrm{kg/m^3}$, $\eta = 1\,\mathrm{mPa \cdot s}$ und $\sigma = 20\,\mathrm{mN/m}$

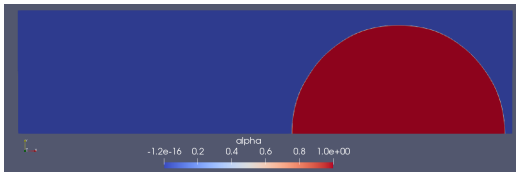
stellt die typischerweise beobachteten Fälle dar, die in den 36 Simulationen beobachtet wurden. Unterabbildung 3.24a zeigt die Ausgangsposition der Simulation.



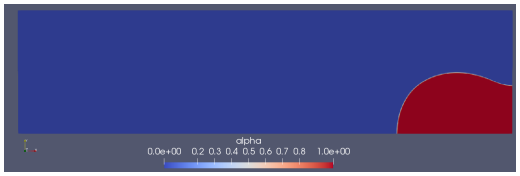
(a) Ausgangsposition jeder einzelnen Simulation



(b) Bewegter Tropfen trotz $U_0 = 0\,\mathrm{m/s}$ mit $c_\alpha = 0,25$ bei $t = 0{,}32\,\mathrm{s}$



(c) Normaler konvektiver Transport mit $U_0 = 0{,}005\,\mathrm{m/s}$ und $c_\alpha = 1,25$ zu der Zeit $t = 0{,}32\,\mathrm{s}$



(d) Gestörter konvektiver Transport mit $U_0 = 0{,}02\,\mathrm{m/s}$ und $c_\alpha = 0,5$ zu der Zeit $t = 0{,}08\,\mathrm{s}$

Abbildung 3.24: Beobachtete Fälle beim Transport des Tropfen durch eine angelegte Geschwindigkeit U_0 bei verschiedenen Kompressionsstärken c_α durch den Solver interFoam.

Unterabbildung 3.24b stellt einen Fall dar, in dem sich der Tropfen trotz keinerlei anliegender Transportgeschwindigkeit U_0 bewegt hat. Lediglich bei keiner anliegenden Transportgeschwindkeit $U_0 = 0\,\mathrm{m/s}$ und gleichzeitig verschwindender Kompressionsstärke $c_\alpha = 0$ bleibt der Tropfen in Position und die parasitären Strömungen klingen kontinuierlich ab. Unterabbildung 3.24c stellt beispielhaft für $U_0 = 0{,}005\,\mathrm{m/s}$ einen gelungenen Fall dar, in dem der Tropfen durch die anliegende Transportgeschwindigkeit innerhalb von $0{,}32\,\mathrm{s}$ die geforderten $1{,}6\,\mathrm{mm}$ weit transportiert wird. Dieser Fall tritt unabhängig von U_0 erst bei der höchsten betrachteten Kompressionsstärke von $c_\alpha = 1,25$ zuverlässig ein. Unterabbildung 3.24d stellt hingegen einen Fall mit $c_\alpha = 0,5$ dar, in dem der Tropfen durch die Transportgeschwindigkeit U_0 bewegt, aber durch die auftretenden parasitären Strömungen über das Ziel von $1{,}6\,\mathrm{mm}$ hinaus aus dem Rechengebiet heraustransportiert wird.

Zur Quantifizierung der parasitären Strömungen wird die gleiche Interpretation von U_{fehl} wie in Abschnitt 3.6 angesetzt (vgl. Gl. 3.59). Die Position der Grenzfläche wird hierfür an der Stelle $\alpha = 0,5$ definiert, da α im Bereich zwischen 0 und 1 variiert. An diese Stelle werden Geschwindigkeitsvektor $\vec{U}$ und der aus $\vec{\nabla}\alpha$ gebildete Normalenvektor $\vec{n}$ der anliegenden Zellen interpoliert. Bereinigt um den Sollanteil der Geschwindigkeit, welcher $\vec{e}_x \cdot U_0$ beträgt, wird der Anteil in direkter Richtung durch die Grenzfläche hindurch $\left|(\vec{U}_{\alpha=0,5} - \vec{e}_x \cdot U_0) * \vec{n}_{\alpha=0,5}\right|$ als charakteristische Größe der Störströmung U_{fehl} angesehen. Die Implementierung zur Bestimmung von U_{fehl} in interFoam ist ebenfalls in Anhand D einzusehen. Im Folgenden werden die beobachteten parasitären Strömungen in interFoam mit den in Unterabschnitt 3.6.2 erarbeiteten Ergebnissen verglichen. Hierzu werden sowohl die durchgeführten Simulationen selbst, als auch die daran angepasste Korrelation nach Gl. 3.64 für unterschiedliche Anzahl an Kompressionsschritten herangezogen. Der Vergleich ist in Abbildung 3.25 dargestellt. Lediglich für $c_\alpha = 1,25$ wurde zuverlässig ein stabiler Transport des Tropfens beobachtet. Für $c_\alpha = 1,00$ lagen bereits Störungen des Tropfentransports

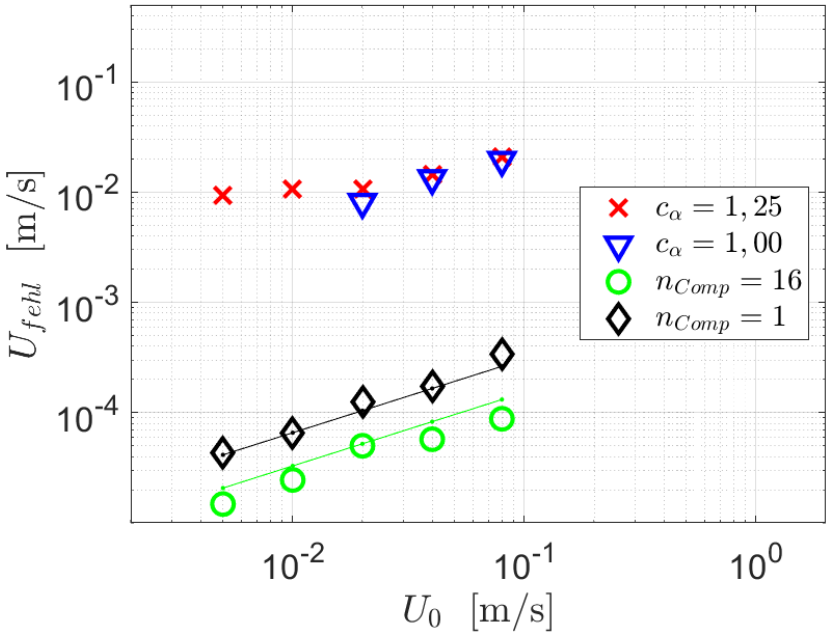


Abbildung 3.25: Vergleich der parasitären Strömungen am transportierten Tropfen bei der Nutzung von interFoam unter Variation von c_α (rote Kreuze/blaue Dreiecke) und der in dieser Arbeit entwickelten Methode mit unterschiedlich vielen Kompressionsschritten n_{Comp} (schwarze Rauten/grüne Kreise).

ähnlich zu Unterabbildung 3.24d vor, welche nicht eingetragen wurden. Die parasitären Strömungen von interFoam liegen in den verglichenen Fällen im Bereich von 0,008 m/s−0,021 m/s und besitzen damit die gleiche Größenordnung wie die angelegten Transportgeschwindigkeiten U_0 oder übersteigen diese für $U_0 = 0{,}005\,\mathrm{m/s}$ sogar. Eine große Abweichung von der zu erwartenden Position des Tropfens ist dadurch nicht verwunderlich. Dem gegenüber erreichen die in der Simulation verzeichneten Fehlströmungen unter Nutzung der neuen Methodik abhängig von der betrachteten Transportgeschwindigkeit U_0 maximal 0,30% der Hauptströmung für 16 Kompressionsschritte und maximal 0,87% bei einem Kompressionsschritt pro Zeitschritt. Im Vergleich zu interFoam wird demnach für $n_{Comp} = 1$ eine Verbesserung um einen Faktor zwischen 57 und 215 erreicht, für $n_{Comp} = 16$ sogar um einen Faktor zwischen 162 und 627. Durch die stark zurückgedrängten parasitären Strömungen, ist eine fehlerhaften Auslenkung der Grenzfläche mit der verbesserten Methode nun nicht mehr zu erwarten, was für eine akkurate und zuverlässige Simulation der Phasentrennung als unerlässlich angesehen wird. Die verbesserte numerische Methode liefert somit den notwendigen Grundstein für die in Abschnitt 4.1 durchgeführten Simulationen der Phasentrennung.

Kapitel 4

Untersuchungen zu Anwendungsgebieten und Problembereichen

Ziel der vorliegenden Arbeit ist die Charakterisierung eines konkreten Anwendungsgebietes für die Pfropfenströmung als Extraktionswerkzeug. Hierzu wurde in Kapitel 2 eine ausführliche Übersicht zum Stand der Technik erarbeitet, aus der als größtes Problem der zu hohe apparative Aufwand hervorgeht. Darüber hinaus wurde herausgearbeitet, dass einige Apparate zur Phasentrennung bereits gut, aber noch nicht vollständig analysiert sind. Beide Problematiken und deren Kopplung werden in diesem Kapitel näher untersucht. Zur Etablierung der Pfropfenströmung als Extraktionswerkzeug werden dabei zwei mögliche Szenarien betrachtet.

- Szenario 1: Zum Einen kann die Pfropfenströmung durch Beseitigung ihrer bisherigen Nachteile für herkömmliche Extraktionsaufgaben an Attraktivität gewinnen. Hierzu sind massive Parallelisierung zur Steigerung des Durchsatzes und extreme Kostenreduktion für die Bereitstellung eines einzelnen Extraktionsstrangs notwendig. Dieser Ansatz wird im Folgenden als erstes Szenario bezeichnet.

- Szenario 2: Zum Anderen kann die Pfropfenströmung sich bei Extraktionsausgaben beweisen, bei denen bisherige Apparate noch an ihre Grenzen stoßen. Trotz gängiger Schwierigkeiten in der Anwendung wäre die mikroverfahrenstechnische Pfropfenströmung dann trotzdem das Mittel der Wahl. Im Weiteren wird diese Herangehensweise als das zweite mögliche Szenario bezeichnet.

Die vorgestellten Untersuchungen teilen sich daher in zwei Teilgebiete, welche beide Szenarien näher untersuchen. Zunächst wird bezüglich des ersten Szenarios in Abschnitt 4.1 an einer Beispielgeometrie numerisch untersucht, welche Faktoren die bisherigen Phasentrenner trotz Einhaltung der Kräftebilanz nach Gl. 2.6 versagen lassen. Hierzu wird die in Kapitel 3 entwickelte numerische Methode genutzt, welche den notwendigen Grundstein für die akkurate Modellierung liefert. Auf den Ergebnissen aufbauend wird in Abschnitt 4.2 ein Phasentrenner untersucht, der die zuvor identifizierten Probleme zu umgehen versucht. Dieser soll ohne den Einsatz von Membranen eine stabile Trennleistung bereitstellen. Sofern bei der Phasentrennung auf den Einsatz von Membranseparatoren verzichtet werden kann, reduziert dies den zu überwindenden Druckverlust der Pumpen. Dies ermöglicht den Einsatz

von Förderkonzepten, die nur einen geringen Förderdruck aufbauen können, jedoch sehr kostengünstig und einfach anzuwenden sind. Die Grundzüge eines solchen Förderkonzeptes durch Magnetfelder und eine ferrofluidische Pfropfenphase, welches gleichzeitig auch als Sensorik zur Überwachung der Strömungsverhältnisse in den Kapillaren fungiert, und dadurch weitere Kosten einspart, wird in Abschnitt 4.3 vorgestellt. Die vereinfachte Parallelisierung und verbesserte Phasentrennung sollen die Pfropfenströmung dabei konkurrenzfähig zu bisherigen Extraktionsapparaten bei gewöhnlichen Extraktionsaufgaben machen. Genannte Abschnitte summieren die Untersuchungen zum ersten Szenario.

Parallel wird die Pfropfenströmung bezüglich des zweiten Szenarios in Anwendungsgebieten näher charakterisiert, die durch bisherige Extraktionsapparate nicht oder nur unzureichend erschlossen werden können. Die durchgeführte Literaturanalyse zeigte dabei, dass bisherige Extraktionsapparate bei kleinen Dichtedifferenzen, großen Grenzflächenspannungen und stark erhöhter Viskosität an ihre Grenzen stoßen. Die kleine Dichtedifferenz erschwert die gravitative Phasentrennung, während die große Grenzflächenspannung zur Bildung sehr großer und stabiler Tropfen innerhalb von Kolonnen führt, wodurch die Austauschfläche a_{eff} sinkt. Eine erhöhte Viskosität behindert hingegen den gravitativen Gegenstrombetrieb und führt zu einem frühen Fluten der Kolonne. In Abschnitt 4.4 wird daher der Stofftransport in einer Pfropfenströmung bei Anwendung eines hochviskosen Stoffsystems mit geringer Dichtedifferenz untersucht. Auch für diesen Anwendungsfall stehen einer industriellen Umsetzung noch die hohen apparativen Kosten gegenüber. Hierzu wird in Abschnitt 4.5 ein Konzept zur Parallelisierung mehrerer Extraktionsstränge bei reduzierter Pumpenzahl präsentiert. Der weiterhin gute Stofftransport und die erleichterte Parallelisierung machen den Einsatz der Pfropfenströmung für ungewöhnliche Extraktionsaufgaben attraktiver. Dies summiert die Untersuchungen zum zweiten Szenario.

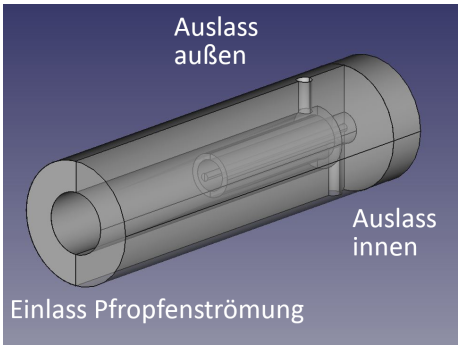
4.1 Numerische Untersuchung der Phasentrennung

Für eine erfolgreiche Etablierung der Pfropfenströmung als Werkzeug in der Mikroextraktion ist eine stabile und schnelle Trennung der Phasen nach Abschluss der Extraktion notwendig. Ist die Dichtedifferenz für eine gravitative Phasentrennung zu gering, bietet sich in der Mikroverfahrenstechnik eine Trennung über die Grenzflächenspannung und Benetzbarkeit an Oberflächen an. Wie aus Paragraph 2.2.2.2 hervorgeht, existieren bereits Apparate, die eine robuste Trennleistung bereitstellen können. Diese arbeiten jedoch mit feinen Membranen oder Porenkämmen, um einen großen Kapillardruck bereitstellen zu können. Solche Maßnahmen bringen gleichzeitig hohe Druckverluste mit sich, wodurch die Pumpen mehr Leistung benötigen oder manche Förderkonzepte erst gar nicht anwendbar sind. Die in Kapitel 3 erarbeitete Methode wird daher genutzt, um den Trennvorgang in einem Phasentrenner nach Trennprinzip A (vgl. Abbildung 2.3) nachzubilden und so die experimentell beobachteten Abweichungen von der Modellvorstellung (Gl. 2.6) zu erklären.

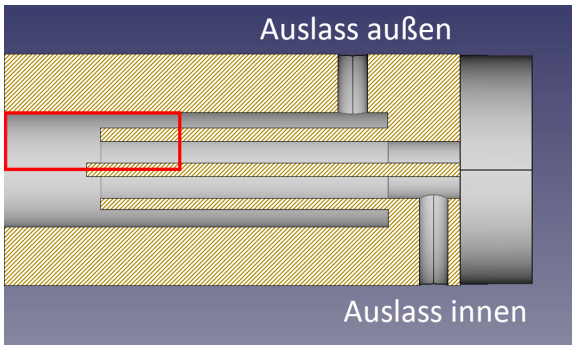
Das bisherige Modell sagt eine stabile Phasentrennung voraus, wenn eine Phase beim Eintritt in den falschen Auslass einen höheren Kapillardruck $\Delta p_{cap,i}$ zu überwinden hat, als den aktuellen hydrodynamische Druckverlust $\Delta p_{hydr,i}$ im eigentlich vorgesehenen Auslass.

$$\Delta p_{cap,i} \geq \Delta p_{hydr,i} \tag{4.1}$$

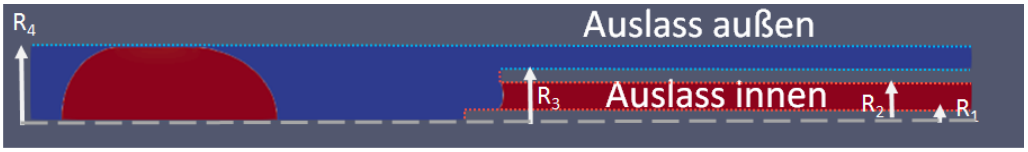
Erfüllen beide Fluide diese Ungleichung, sollte an beiden Auslässen jeweils nur reines Fluid vorliegen. Zur numerischen Untersuchung dieser Modellvorstellung wird ein fiktiver Phasentrenner verwendet, dessen Design auf Ideen von Sauf basiert ([108]) und damit die Umkehrung des koaxialen Pfropfenerzeugers darstellt, den Arsenjuk et al. bereits numerisch untersuchten ([24]). Die Wahl dieser Geometrie ermöglicht durch die Rotationssymmetrie numerisch eine pseudo-2D-Modellierung auf einem wedge-Gitter und ist gleichzeitig repräsentativ für alle Phasentrenner des Typs A nach Abbildung 2.3. Die komplette Geometrie mit Ausschnitt des simulierten Fluidgebiets ist in Abbildung 4.1 dargestellt. Genaue Angaben zu den gesetzten Abmessungen sind in Anhang E enthalten. Der Radius der Kapillare wird zu $R_4 = 0{,}8\,\text{mm}$



(a) 3D-Ansicht des fiktiven Phasentrenners.



(b) Schnittansicht mit schematisch in rot angedeutetem Simulationsgebiet.



(c) In OpenFOAM abgebildeter Ausschnitt der Geometrie mit einlaufendem Pfropfen (rot) und umgebender kontinuierlicher Phase (blau). Bevorzugt benetzende Phase an jeder Wand durch Strichlinien gleicher Farbe angedeutet.

Abbildung 4.1: Dreidimensionale Komplettansicht (a) und zweidimensionale Schnittansicht (b) des simulierten Phasentrennungs mit Visualisierung des simulierten Fluidgebietes (c).

gewählt. In Abschnitt 3.8 wurde erarbeitet, dass eine räumliche Auflösung von 200 Zellen in radialer Richtung ausreichend ist, um Pfropfenströmungen mit einem Viskositätsverhältnis nahe 1 und einer Kapillar-Zahl $Ca > 1,8 \cdot 10^{-3}$ abbilden zu können. Dies entspricht einer Gitterweite von $\Delta x = 4 \cdot 10^{-6}\,\text{m}$. In der Mitte des inneren Auslasses wurde ein Koaleszenzdorn platziert, auf den die ankommenden Pfropfen zunächst auflaufen. Durch den benetzenden Effekt am Dorn bauen die Pfropfen bereits früh Kontakt mit der restlichen dispersen Phase auf. Darüber hinaus erhöht dies den Kapillardruck, welchen die disperse Phase der einlaufenden Strömung entgegen setzen kann. Der eigentliche Auslauf der Fluide in die freie Umgebung ist im Simulationsgebiet nicht enthalten. Das Fluidgebiet wird allerdings als ausreichend angesehen, um die grundsätzlichen Effekte bei der Phasentrennung verfolgen zu können. Bei der Deutung der Ergebnisse ist jedoch auf die eigentlich größere Länge der angeschlossenen Kapillaren zu achten. Gestrichelte Linien deuten in Unterabbildung 4.1c die an jeder Wand bevorzugt benetzende Phase an. Im äußeren Auslass weist die kontinuierliche Phase einen Kontaktwinkel von 40° mit der Wand auf. Im in-

neren Auslass benetzt stattdessen die disperse Phase das Wandmaterial. Auch hier wird ein Kontaktwinkel von 40° gesetzt. Diese Zahlenwerte sind fiktiv, aber erlauben dennoch eine Studie des allgemeinen Trennverhaltens. Die Stoffwerte der simulierten Fluide sind in Tabelle 4.1 aufgeführt. [1] Um die Gültigkeit des bisher geltenden

Tabelle 4.1: Den Simulationen zu Grunde liegende Stoffdaten für kontinuierliche Phase (blau) und disperse Phase (rot), orientiert an Styrol, Wasser und Luft ([109], [110]).

Fluidphase	Dichte ρ	Viskosität η	Grenzflächenspannung σ
kontinuierlich	$901{,}6\,\mathrm{kg/m^3}$	$0{,}695\,\mathrm{mPa \cdot s}$	$32{,}0\,\mathrm{mN/m}$
dispers	$996{,}87\,\mathrm{kg/m^3}$	$0{,}974\,\mathrm{mPa \cdot s}$	

Vorhersagemodells zu prüfen, müssen Kapillardruck $\Delta p_{cap,i}$ und hydrodynamischer Druck $\Delta p_{hydr,i}$ quantifiziert werden. Die postulierten Gleichungen hierfür sind in Gl. 4.2 und Gl. 4.3 dargestellt.

$$\Delta p_{cap,i} = \frac{\sigma}{R_{eff,i}} \cdot \cos(\theta) \tag{4.2}$$

$$\begin{aligned} \Delta p_{hydr,i} &= \frac{\rho_i}{2} \cdot (\bar{u})^2 \cdot \left(\lambda \cdot \frac{L}{D} + \sum \zeta \right) \\ &= \frac{\rho_i}{2} \cdot (\bar{u})^2 \cdot \left(\frac{A_i}{\mathrm{Re}} + B_i \right) \end{aligned} \tag{4.3}$$

Dabei gilt $R_{eff} = (R_4 - R_3)/2 = 0{,}12\,\mathrm{mm}$ für die disperse Phase, welche einen Meniskus im äußeren Auslass ausbildet, und $R_{eff} = (R_2 - R_1)/2 = 0{,}15\,\mathrm{mm}$ für einen Meniskus kontinuierlicher Phasen, welcher in den inneren Auslass einzudringen versucht. Gl. 4.3 resultiert aus dem Ansatz eines verlustbehafteten Stromfadens. [111] Der für laminare geradlinige Rohrströmungen typischerweise geltende Term $\lambda = 64/\mathrm{Re}$ mit $\mathrm{Re} = 2 \cdot R_4 \cdot \bar{u} \cdot \rho_i/\eta_i$ führt auf die Annahme einer Modellierbarkeit über A_i und B_i. Beide Werte werden jeweils für den inneren und äußeren Auslass durch einen Parameterfit bestimmt. Die Parametervariationen zur Verifikation von Gl. 4.2 und Erstellung der Datengrundlage für den Parameterfit von Gl. 4.3 sind in Tabelle 4.2 aufgeführt. Für die Bestimmung von $\Delta p_{cap,i}$ werden beide Auslässe verschlossen. Es erfolgt keinerlei Einströmung. In einem Auslass wird ein Meniskus platziert, welcher einen Kapillardruck ausbildet. Die Druckdifferenz zwischen beiden Auslässen wird dann als $\Delta p_{cap,i}$ angesehen. Für die Bestimmung von $\Delta p_{hydr,i}$ wird jeweils ein Auslass verschlossen und der andere mit Ausströmung in die freie Umgebung geöffnet. Fluid wird durch den Einlass eingeleitet und erfährt beim Verlassen durch den einzigen Auslass einen Druckverlust, während das Fluid im verschlossenen Auslass praktisch still steht. Die Geschwindigkeit $\bar{u}$ wird dabei als parabolisches Geschwindigkeitsprofil der Form $u(r) = 2 \cdot \bar{u} \cdot \left(1 - (r/R_4)^2\right)$ am Einlass aufgegeben. Als zu modellierende Druckdifferenz wird wieder der Unterschied zwischen den mittleren Druckniveaus beider Auslassöffnungen angesehen.

[1] Stoffdaten angelehnt an Styrol und Wasser. Keine exakte Übereinstimmung, da Interpolationsfehler einfließen und zudem ursprünglich die Systeme Luft/Styrol ($\sigma = 32{,}0\,\mathrm{mN/m}$) und Wasser/Styrol ($\sigma = 35{,}48\,\mathrm{mN/m}$) bei dennoch konstanter Grenzflächenspannung verglichen werden sollten.

Tabelle 4.2: Permutationsparameter zur Validierung von Gl. 4.2 (links) und zur Erstellung der Datengrundlage für den Parameterfit (A_i, B_i) von Gl. 4.3 (rechts).

θ	σ	$\bar{u}$		ρ_i	η_i
40 °	8,0 mN/m	0,010 m/s	0,015 m/s	600 kg/m^3	0,4 mPa · s
50 °	20,0 mN/m	0,020 m/s	0,025 m/s	800 kg/m^3	0,7 mPa · s
60 °	32,0 mN/m	0,030 m/s	0,035 m/s	1000 kg/m^3	1,0 mPa · s
70 °	44,0 mN/m	0,040 m/s	0,045 m/s		
		0,050 m/s	0,055 m/s		
		0,060 m/s			

Abbildung 4.2 und 4.3 zeigen die gute Übereinstimmung der postulierten Korrelationen mit den Simulationsergebnissen. Die Modellierung von $\Delta p_{cap,i}$ wird of-

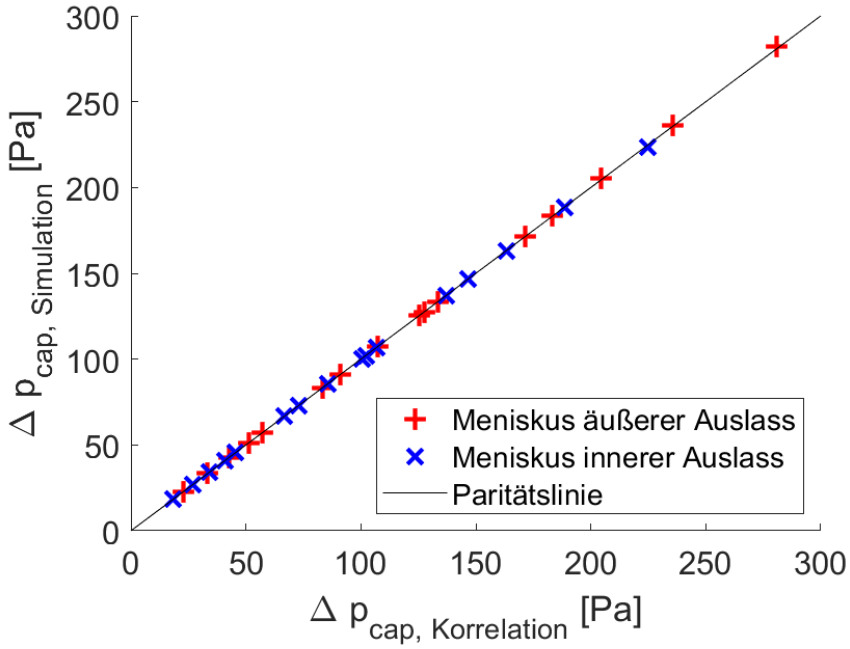


Abbildung 4.2: Paritätsplot zur Beschreibung von Δp_{cap} nach Gl. 4.2.

fensichtlich nur wenig dadurch beeinflusst, dass eine durch Rotationssymmetrie hervorgerufene Krümmung der Grenzfläche in die zweite Raumrichtung bei der Beschreibung durch Gl. 4.2 vernachlässigt wurde. Die Korrelation weicht im Mittel um $0,16\%$ und maximal um $0,47\%$ von den Simulationsergebnissen ab. Bei der Modellierung von Δp_{hydr} ergeben sich für eine Durchströmung des äußeren Auslasses Konstanten von $A = 6966,28$ und $B = 7,41$. Ein Durchfluss durch den inneren Auslass benötigt Konstanten von $A = 9471,33$ und $B = 30,11$ zur Beschreibung. Die Korrelation für Δp_{hydr} weicht von den Simulationsergebnissen dabei im Mittel um $0,04\%$ und maximal um $0,20\%$ ab. Die Reynolds-Zahl variiert im betrachteten Parameterbereich zwischen $9,6$ und 240. Trägheitsdruckverluste sind daher bereits nicht mehr zu vernachlässigen. Dies zeigt sich auch darin, dass die jeweiligen Werte von B, welche im Stromfaden unter Anderen die Umlenkungdruckverluste durch $\sum \zeta$ beschreiben, nicht verschwinden. Eine derartige Berücksichtigung wurde in bisherigen Literaturstellen noch nicht vorgefunden, was bereits die ersten Abweichungen

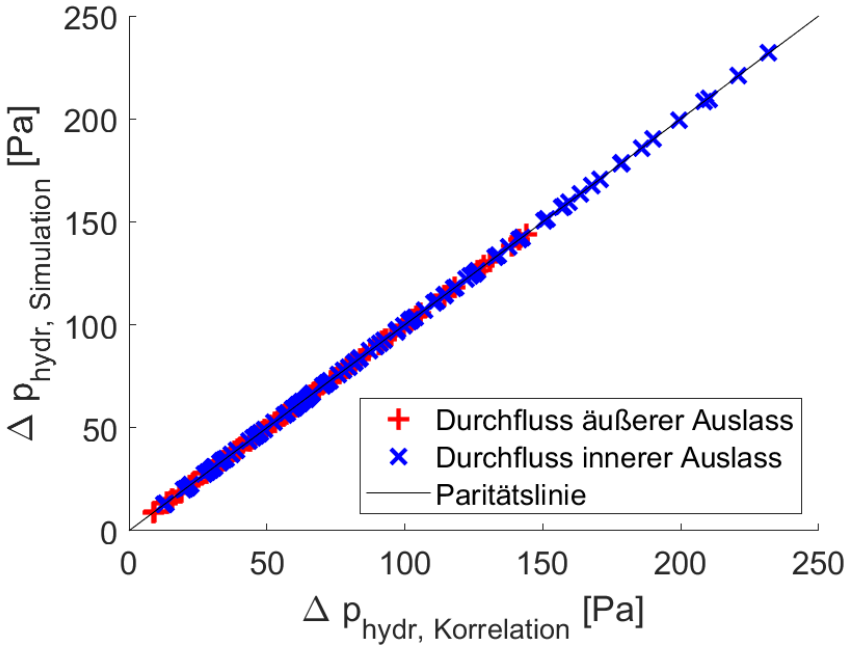


Abbildung 4.3: Paritätsplot zur Beschreibung von Δp_{hydr} nach Gl. 4.3.

von theoretischen Modellvorhersagen und experimentellen Befunden erklärt. Werden nur viskose Druckverluste berücksichtigt, so reicht nach Vorhersage ein kleiner Kapillardruck aus, der in Realität jedoch keine vollständige Trennung garantieren kann und daher zu Verunreinigungen in den Auslässen führt.

Für die eigentliche Simulation der Phasentrennung wird zunächst die Strömung der kontinuierlichen Phase bei verschlossenem innerem Auslass initialisiert. Nach einer fallspezifischen Einlaufzeit wird der innere Auslass geöffnet und mit disperser Phase gefüllt. Im ersten Zeitschritt wird der Pfropfen kurz hinter der Einlassöffnung als Zylinderkörper mit aufgesetzten Kugelkappen initialisiert. Die Länge des Zylinderkörpers wird äquivalent zum Kapillardurchmesser $R_4 = 0{,}8\,\mathrm{mm}$ gewählt. Um direkt eine Strömung mit Wandfilm zu initialisieren, werden die Radien des Zylinderkörpers und der Kugelkappen leicht kleiner zu $R_{slug} = 0{,}79\,\mathrm{mm}$ gesetzt. Im Bereich $0{,}45\,\mathrm{m/s} \leq \bar{u} \leq 0{,}60\,\mathrm{m/s}$ als Einlassgeschwindigkeit wird die Phasentrennung mit Stoffdaten nach Tabelle 4.1 simuliert. Laut Modellvorhersage ist eine saubere Trennung bis einschließlich $\bar{u} = 0{,}55\,\mathrm{m/s}$ zu erwarten. Abbildung 4.4 visualisiert den simulierten zeitlichen Verlauf. Zunächst zeigen die Ergebnisse, dass die Trennung nicht sauber verläuft, obwohl der Kapillardruck Δp_{cap} im äußeren Auslass größer ist als der hydrodynamische Druckverlust Δp_{hydr}. Dies legt nahe, dass in der Modellvorstellung noch nicht alle zu berücksichtigen Effekte enthalten sind. Da die Korrelationen für Δp_{cap} und Δp_{hydr} mit der gleichen numerischen Methode und identischem Rechengitter erstellt wurden, wie die Simulation der Trennung selbst, deutet dies nicht auf einen räumlichen Diskretisierungs- oder Implementierungsfehler hin, sondern auf einen methodischen Unterschied. Während die Simulationen zur Verifizierung von Gl. 4.2 und Gl. 4.3 in einen stationären Endzustand mündeten, ist die Trennung selbst stets transient. Aus den Ergebnissen der Variationsstudie nach Tabelle 3.10 geht jedoch hervor, dass zeitliche Diskretisierungfehler bis zu einer Courant-Zahl von $0,04$ durch das verwendete Runge-Kutta-Verfahren vernachlässigbar klein

Abbildung 4.4: Zeitlicher Verlauf der simulierten unsauberen Phasentrennung für die Werte $\bar{u} = 0{,}55\,\mathrm{m/s}$ und $\sigma = 32\,\mathrm{mN/m}$ in den ersten 150 ms.

sind. Die Trennungen selber wurden mit einer adaptiven Zeitschrittweitensteuerung durchgeführt, die aus Stabilitätsgründen der Kompression (Gl. 3.53) die maximale Zeitschrittweite auf $1 \cdot 10^{-6}\,\mathrm{s}$ limitiert und bei besonders schnellen Bewegungen der Grenzfläche die Courant-Zahl auf $0,15$ begrenzt. Von einer Beeinflussung der Ergebnisse durch zeitliche Diskretisierungsfehler wird daher ebenfalls nicht ausgegangen. Dementsprechend sind Modellierungsfehler zu betrachten, die ausschließlich in transienten Fällen auftreten. In der Praxis können sowohl der dynamische Kontaktwinkel θ_{dyn} als auch Beschleunigungsdruckverluste in den Auslässen die Gültigkeit der Ungleichung 4.1 beeinflussen. Die Simulationen wurden hingegen mit einem statischen Kontaktwinkel durchgeführt. Dennoch wurde eine unsaubere Trennung beobachtet, weshalb den Beschleunigungsdruckverlusten in der Praxis mindestens ein Teileffekt zugeschrieben werden muss. Daher werden die Beschleunigungsdruckverluste in der Formulierung des Stromfadens genauer betrachtet, welcher Grundlage der Korrelation 4.3 ist. Die allgemeine Form ist in Gl. 4.4 dargestellt. [111]

$$p_1 + \frac{\rho}{2} \cdot {u_1}^2 = p_2 + \frac{\rho}{2} \cdot {u_2}^2 + \frac{\rho}{2} \cdot {u_{ref}}^2 \cdot \left(\lambda \cdot \frac{L}{D} + \sum \zeta\right) + \int_1^2 \rho \cdot \frac{\partial u}{\partial t} \mathrm{d}s \qquad (4.4)$$

Die Beschleunigungsverluste treten dadurch auf, dass bei Phasentrennern nach dem Trennprinzip A (vgl. Abbildung 2.3) die Pfropfen den weiteren Abfluss der kontinuierlichen Phase blockieren. Trifft ein Pfropfen auf den Auslass der dispersen Phase, so muss die Geschwindigkeit im Auslass der kontinuierlichen Phase abrupt zum Stehen kommen. Das Fluid im dispersen Auslass muss hingegen beschleunigt werden. Ist der Pfropfen komplett abgeflossen, kehrt sich die Situation um und das Segment kontinuierlicher Phase folgt. Die durch den Pfropfen hervorgerufene Geschwindigkeit im Auslass der dispersen Phase muss nun schnellstmöglich abgebaut werden, während das Fluid im Auslass der kontinuierlichen Phase beschleunigt werden muss. Dieser Effekt wird umso stärker, je weniger Zeit für die Beschleunigung zur Verfügung steht und je größer die notwendige Abflussgeschwindigkeit ist. Beide Effekte skalieren demnach mit der Einlassgeschwindigkeit $\bar{u}$, weshalb zur genaue-

ren Untersuchung in weiteren Simulationen $\bar{u}$ variiert und iterativ die notwendige Grenzflächenspannung σ bestimmt wird, welche gerade noch ein saubere Trennung garantieren kann. Der auftretende instationäre Druckverlust Δp_{inst} ist dabei nur auf ein Intervall bestimmbar, welches durch die kleinste Grenzflächenspannung mit sauberer Trennung und die größte Grenzflächenspannung mit unsauberer Trennung bestimmt wird. Abbildung 4.5 zeigt den tatsächlich notwendigen Kapillardruck Δp_{cap} als Mittelwert der bestimmten Intervallgrenzen im Vergleich mit dem zu erwartetem Wert nach Gl. 4.3. Für $\bar{u} = 0{,}04\,\mathrm{m/s}$ konnte kein Versagen der Trennung

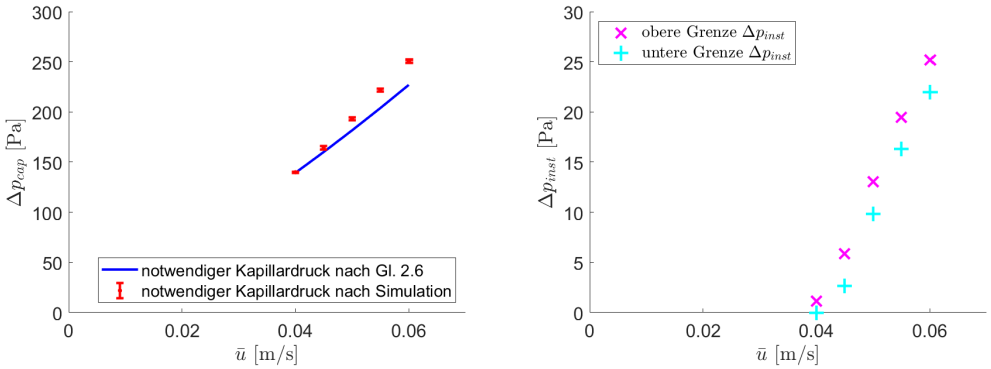


Abbildung 4.5: Vergleich aus notwendigem und erwartetem Kapillardruck für eine saubere Phasentrennung mit Intervallgrenzen als Fehlerbalken (links). Resultierender zusätzlicher Druckverlust durch instationäre Effekte (rechts).

unter gleichzeitiger Einhaltung von Gl. 4.1 beobachtet werden. Die durchgeführten Simulationen beschreiben lediglich das Eintreten eines einzelnen Pfropfens in den Trennbereich. Das Nachfolgen weiterer Pfropfen kann eine saubere Trennung jedoch sowohl begünstigen, wie auch erschweren. Folgt sehr rasch auf den ersten Pfropfen ein zweiter, so kann es je nach Gestaltung des Trennraums zu einer Koaleszenz der Pfropfen kommen. Dies ermöglicht einen kontinuierlichen Abfluss der dispersen Phase ohne Geschwindigkeitsänderungen im zugehörigen Auslass. Dies würde den Betrag von Δp_{inst} senken. Anhand von Abbildung 4.6 zeigen die Simulationen jedoch auch, dass ein nachfolgender Pfropfen die Anforderungen an den Kapillardruck weiter steigern kann. Die disperse Phase im inneren Auslass ist erst nach $t = 0{,}11\,\mathrm{s}$ wieder an ihre ursprüngliche Position zurückgekehrt. Tritt zuvor im Zeitfenster zwischen $t = 0{,}09\,\mathrm{s}$ und $t = 0{,}11\,\mathrm{s}$ ein zweiter Pfropfen in den Trennbereich ein, so kann dies zu Einschlüssen an kontinuierlicher Phase im inneren Auslass führen. Das starke Zurückweichen der dispersen Phase kann nur durch einen höheren Kapillardruck ausgeglichen werden. Instationäre Effekte treten demnach in beiden Auslässen auf und kommen nicht nur im äußeren Auslass zum Tragen. Dies bestätigt auf theoretischer Ebene die Beobachtung von Gaakeer et al., die einen ungefähr 30% höheren Kapillardruck fordern musste, um eine sauber Trennung garantieren zu können. [34] Ist die Trennung erst einmal nicht sauber erfolgt, und befinden sich Verunreinigungen in den Auslässen, steigert dies den hydrodynamischen Druckverlust weiter. Eine Rückkehr in den Zustand einer sauberen Trennung ist dann sehr unwahrscheinlich, was die saubere Phasentrennung, je nach Sicherheitsaufschlag auf den Kapillardruck, zu einem metastabilen Betriebszustand macht.

Mit Hilfe der erarbeiteten numerischen Methode konnte in einer fiktiven Trenngeometrie gezeigt werden, dass neben den viskosen auch trägheitsbestimmte und

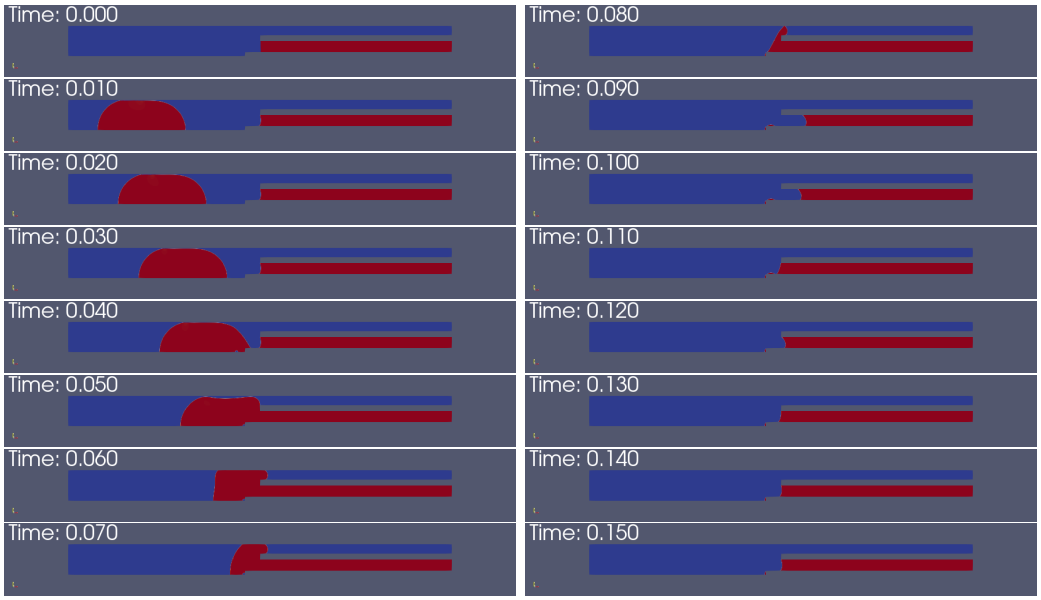


Abbildung 4.6: Zeitlicher Verlauf der simulierten sauberen Phasentrennung für die Werte $\bar{u} = 0{,}55\,\mathrm{m/s}$ und $\sigma = 35\,\mathrm{mN/m}$ in den ersten 150 ms.

instationäre Druckverluste bei der Auslegung des notwendigen Kapillardrucks in Phasentrennern des Trennprinzips A (vgl. Abbildung 2.3) berücksichtigt werden müssen. Die Simulationen bilden damit das in Experimenten beobachtete Verhalten gut ab, dass aktuelle Vorhersagemodelle nur für kleine Volumenströme gut anwendbar sind, bei denen Trägheitseffekte vernachlässigbar klein sind. Die Aussagen der Simulationen sind vor dem Hintergrund zu bewerten, dass nur eine begrenzte Länge des Auslassbereichs im Simulationsgebiet enthalten ist. Die instationären Terme des modellierten Stromfadens zeigen allerdings, dass viskose und trägheitsbestimmte Druckverluste gleichermaßen mit der Länge der Auslassleitungen skalieren und die Aussagekraft der Simulationen in Bezug auf die Praxis daher unberührt bleibt. Eine genaue Modellierung des notwendigen Aufschlags wurde nicht erzielt. Praxisrelevant ist hingegen die Annahme einer Rotationssymmetrie. Die untersuchte Trennergeometrie wird in der dargestellten Weise nur schwer zu fertigen sein und würde durch Toleranzen keine perfekte Symmetrie aufweisen. Dies macht den wirkenden Kapillardruck ebenfalls unsymmetrisch. Das Auftreten instationärer Effekt im rotationssymmetrischen Fall lässt jedoch klar darauf schließen, dass auch im unsymmetrischen Fall ein nennenswerter Einfluss zu beobachten sein wird. Dreidimensionale Berechnungen der Phasentrennung zur Stützung dieser Schlussfolgerung sind durch die notwendige Rechenleistung bedingt in absehbarer Zeit jedoch nicht zu realisieren.

Die gewonnenen Erkenntnisse ermöglichen damit nun die Konstruktion und Auswahl von Trennapparaten, welche besser durch das einfache Vorhersagemodel nach Gl. 2.6 zu beschreiben sind. In der Praxis können die beobachteten Effekte minimiert werden, wenn die Segmente kontinuierlicher Phase, wie auch die Pfropfen selbst, im Trennraum die Gelegenheit zur Koaleszenz erhalten und somit eine konstante Abflussgeschwindigkeit in den Auslässen ermöglicht wird. Dies vermeidet unnötige Beschleunigungen der Fluide. Porenkämme mit seitlichem Abzug der kontinuierlichen Phase ermöglichen dies konstruktiv. Auch die Umwandlung der Pfropfenströmung in eine Parallelströmung eröffnet die Möglichkeit der vorzeitigen Pfropfenkoaleszenz und meidet dabei gleichzeitig kleine Durchtrittsöffnungen mit hohem hydrodyna-

mischen Druckverlusten. Die Phasentrennung durch Umwandlung in eine Parallelströmung wird auf den gewonnen Erkenntnissen aufbauend daher in Abschnitt 4.2 genauer untersucht.

4.2 Phasentrennung durch Umwandlung in Parallelströmung

Aus Abschnitt 4.1 geht hervor, dass die Phasentrennung bei hohen Geschwindigkeiten nicht nur durch viskose und Grenzflächenkräfte, sondern auch durch Trägheitskräfte beeinflusst wird. Die saubere Phasentrennung in Apparaten des Typs A (vgl. Abbildung 2.3) kann dann ohne übermäßigen Kapillardruck nicht erzielt werden. Höhere Kapillardrücke erbringen z.B. Porenkämme und Membranseparatoren, benötigen hierzu jedoch kleine Spalte und Poren. Dies ist besonders bei viskoseren Medien unerwünscht und erzeugt unnötige Druckverluste. Der Einsatz von Suspensionen entfällt wegen der Verblockungsgefahr ebenfalls weitgehend. Um das stete Beschleunigen und Abbremsen der Fluide in den Auslässen der Phasentrenner zu vermeiden, sollte ein Apparat zur Phasentrennung verwendet werden, der den Pfropfen vor Eintritt in die peripheren Leitungen eine Koaleszenz ermöglicht. Gleiches ist für die Fluidbereiche an kontinuierlicher Phase zwischen den Pfropfen anzustreben. Als vielversprechendes Konzept, welches auf die unerwünschten kleinen Poren und Spalte verzichtet, wird hierfür die Umwandlung der Pfropfenströmung in eine Parallelströmung in Betracht gezogen. Beide Phasen können daraufhin als kontinuierliche Ströme abgezogen werden, was die instationären Trägheitsdruckverluste auf den Raum der Trennkammer begrenzt. Der Verzicht auf die Anwendung von Membranen für die Phasentrennung vermeidet dabei auf der einen Seite die Gefahr von Verstopfungen bei der Nutzung von Suspensionen und reduziert auf der anderen Seite den zu überwindenden Druckverlust während der Phasentrennung. Dies macht neuartige Förderkonzepte attraktiver, die zwar kostengünstig sind, jedoch nur einen geringen Förderdruck aufbauen können (siehe Abschnitt 4.3). Ein im Rahmen dieser Arbeit konstruierter Phasentrenner zur Aufreinigung einer Wasser-Heptan Pfropfenströmung wird hierzu in Unterabschnitt 4.2.1 untersucht. Zur Aufklärung der dabei herausgearbeiteten Fragen wird in Unterabschnitt 4.2.2 die allgemeine Stabilität von Parallelströmungen in oberflächenmodifizierten Rechteckkanälen betrachtet.

4.2.1 Auftrennung einer Wasser-Heptan Pfropfenströmung

Ziel dieses Abschnittes ist die Konstruktion und experimentelle Vermessung eines Phasentrenners für die stabile Aufreinigung einer Wasser-Heptan Pfropfenströmung. Im Besonderen sollen aus den Eigenschaften des Stoffsystems und dem Volumenstromverhältnis Φ passende Kriterien zur Konstruktion des Trennapparates hergeleitet und verifiziert werden. Zhao et al. zeigten experimentell bereits, dass Oberflächenmodifikationen der verwendeten Materialien eine Parallelströmung begünstigen können. [112] Alternativ können direkt Materialien mit stark unterschiedlichen Benetzungseingenschaften gewählt werden. Für die Aufreinigung einer Wasser-Heptan Pfropfenströmung wird daher ein Phasentrenner angefertigt, der als hydrophiles Material Aluminium und als hydrophobes Material PTFE verwendet. Das Prinzip und die Realisierung sind in Abbildung 4.7 dargestellt. Die Abmessungen der Kanäle be-

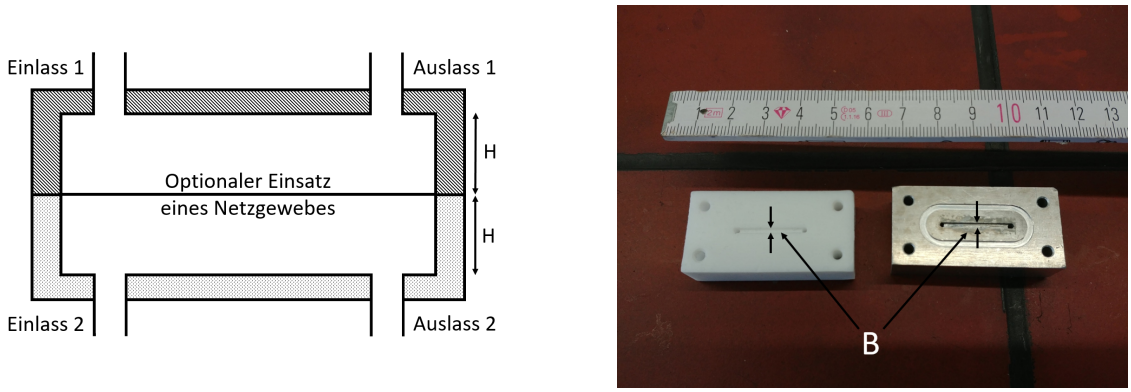


Abbildung 4.7: Prinzipskizze (links) und Darstellung des gefertigten Phasentrenners (rechts) für die Aufreinigung einer Wasser-Heptan Pfropfenströmung mit angedeuteter Tiefe H und Breite B der eingefrästen Kanäle.

tragen für beide Teilhälften jeweils 1,1 mm für die Tiefe H und 1,6 mm für die Breite B des Kanals. Bei der Verschraubung der Hälften kann im Zwischenraum zusätzlich ein Netzgewebe eingesetzt werden, das die Trennwirkung bei Bedarf verbessern soll. Neben dem Einfluss des verwendeten Netzgewebes kann durch die Konstruktion zudem untersucht werden, durch welchen Einlass die Pfropfenströmung idealerweise zugeführt werden sollte. Der andere Einlass wird jeweils verschlossen. Die Benetzungseigenschaften bedingen, dass sich die wässrige Phase bevorzugt in der Aluminiumseite des Kanals ansammelt, während die Organik bevorzugt eine Ansammlung in der Kanalhälfte aus PTFE bildet. Allgemein wird durch die Geometrie und die Wahl des Stoffsystems der optimale Betriebspunkt des Phasentrenners definiert, den es zu berechnen gilt. Zunächst wird dieser bestimmt, wenn im Apparat ein Siebgewebe eingesetzt ist. Allgemein anerkannt ist die Annahme, dass in einer eingelaufenen Parallelströmung in beiden Phasen der gleiche Druckverlust $\frac{\partial p}{\partial x_i}$ in Strömungsrichtung vorherrscht. Zur Bestimmung der Strömungsfelder in beiden Kanalhälften kann aus der Navier-Stokes-Gleichung unter Annahme einer eingelaufenen stationären Strömung in z-Richtung die Poisson-Gleichung (Gl. 4.5) für die Geschwindigkeitskomponente u_z abgeleitet werden.

$$\frac{1}{\eta} \cdot \frac{\partial p}{\partial z} = \Delta u_z \tag{4.5}$$

Wird das Koordinatensystem symmetrisch in die Mitte des betrachteten Rechteckgebietes der Breite B und Höhe H gelegt, so ist unter Setzen von Haftbedingungen an den Rändern ($u_z(x, y = \pm H/2) = 0$ und $u_z(x = \pm B/2, y) = 0$) und Verwendung der Konstanten α_k und C_k aus Gl. 4.7 und Gl. 4.8 eine partikuläre Lösung durch Gl. 4.6 gegeben.

$$U_{z,part.} = \frac{1}{2 \cdot \eta} \cdot \frac{\partial p}{\partial z} \cdot \left(x^2 - \frac{B^2}{4} - \sum_{k=0}^{\infty} \left(\alpha_k \cdot \cos\left(C_k \cdot x\right) \cdot \cosh\left(C_k \cdot y\right) \right) \right) \tag{4.6}$$

$$\alpha_k = \frac{-8 \cdot (-1)^k}{\cosh\left(C_k \cdot \frac{H}{2}\right) \cdot B \cdot {C_k}^3} \tag{4.7}$$

$$C_k = \frac{\pi \cdot (1 + 2 \cdot k)}{B} \tag{4.8}$$

Da am Siebgewebe nach Ausbildung der Parallelströmung auch Haftbedingung vorliegt, kann der Gesamtkanal in zwei unabhängige Untergebiete aufgespalten werden. Ist die Phasentrennung erfolgreich, so strömt Wasser nur in der Kanalhälfte aus Aluminium und Organik stets auf der Seite aus PTFE. Durch die gleichen geometrischen Abmessungen der Kanalhälften unterscheiden sich die Strömungsprofile dann lediglich im Wert der Viskosität η. Die Volumenströme, welche jeweils durch die obere und untere Kanalhälfte strömen, stehen bei identischem Druckverlust $\frac{\partial p}{\partial z}$ beider Phasen also im Verhältnis $\eta_{aq.}/\eta_{org.}$ zueinander, was das erste Betriebsfenster definiert. Die Geometrie des Apparates bestimmt demnach das mögliche Betriebsfenster, beziehungsweise ist der Trennapparat nach dem Stoffsystem und dem angestrebten Phasenverhältnis zu konstruieren.

Ist im Apparat kein Siebgewebe eingebracht, ist die Berechnung des gekoppelten Strömungsprofils im Kanal komplexer. Grundlegend ist der Druckverlust einer Phase von dem Volumenstrom $\dot{V}_i$, der Viskosität η_i und der durchströmten Querschnittsfläche abhängig. Bei eingesetztem Siebgewebe definieren die Tiefe H und Breite B (vgl. Abbildung 4.7) eindeutig den Volumenanteil der jeweiligen Phasen. Ohne Netzgewebe kann die Phasengrenzfläche jedoch durch ihre Form und Position die von jeder Phase durchströmte Querschnittsfläche und damit auch die Druckverluste beeinflussen. Idealerweise ist die hydrophile Kanalhälfte komplett mit Wasser gefüllt, während die hydrophobe Kanalhälfte ausschließlich Organik enthält. Die Grenzfläche ist demnach eben und trifft dort auf die Kanalwände, wo auch die Benetzungseigenschaften wechseln, also auf der Höhe $y = 0$. Der Kapillardruck hat dann bei Wölbung einer Phase in die gegenüberliegende Kanalhälfte einen stabilisierenden Effekt, da der Kapillardruck eine Rückstellkraft erzeugt, welche die Grenzfläche wieder in die ursprüngliche Position drücken sollte. Bei der Berechnung des Strömungsprofils kann an der Berührstelle der Fluide die Geschwindigkeit jedoch nicht zu 0 gesetzt werden, wie es am Siebgewebe der Fall ist. Stattdessen müssen das Strömungsprofil der wässrigen Phase $\vec{u}_1$ und der organischen Phase $\vec{u}_2$ an der Grenzfläche Gl. 4.9 und Gl. 4.10 erfüllen, wobei $\vec{n}$ den Normalenvektor der Grenzfläche bezeichnet.

$$\vec{u}_1 = \vec{u}_2 \tag{4.9}$$

$$\eta_1 \cdot \frac{\partial \vec{u}_1}{\partial \vec{n}} = \eta_2 \cdot \frac{\partial \vec{u}_2}{\partial \vec{n}} \tag{4.10}$$

Hierfür sind der partikulären Lösung (Gl. 4.6) homogene Lösungsterme hinzuzufügen, um die Randbedingung an der Grenzfläche erfüllen zu können. Die notwendigen Terme sind für das obere Teilgebiet ($y > 0$) durch Gl. 4.11 und für das untere Teilgebiet ($y < 0$) durch Gl. 4.12 gegeben.

$$u_{z,hom,1} = \sum_{k=0}^{\infty} \left(\beta_k \cdot \cos\left(C_k \cdot x\right) \cdot \sinh\left(C_k \cdot \left(\frac{H}{2} - y\right)\right)\right) \tag{4.11}$$

$$u_{z,hom,2} = \sum_{k=0}^{\infty} \left(\gamma_k \cdot \cos\left(C_k \cdot x\right) \cdot \sinh\left(C_k \cdot \left(\frac{H}{2} + y\right)\right)\right) \tag{4.12}$$

Zur Handhabung der Lösung werden bei allen unendlichen Reihen nur die Terme bis $k = 10$ verwendet. Zunächst wird der diskutierte optimale Betriebspunkt mit ebener Grenzfläche und komplett einphasig gefüllten Kanalhälften angesetzt. Die Symmetrie der beiden Lösungen bzgl. der Variable x wird genutzt, indem für 11 äquidistant verteilte Werte $x \in [0; B/2]$ die Abgleichbedingungen aus 4.9 und 4.10

formuliert werden, was auf ein Gleichungssystem zur Bestimmung der insgesamt 22 Werte für β_k und γ_k führt. Zwischen den gewählten Stützstellen werden die geforderten Abgleichbedingungen nicht analytisch erfüllt, die Abweichungen sind jedoch klein, wie Abbildung 4.14 zeigt, und werden umso geringer, je mehr weitere Terme mit $k > 10$ und je mehr Stützstellen in x-Richtung hinzugenommen werden. Die so berechneten Strömungsprofile bei vorgegebenen Druckverlust $\frac{\partial p}{\partial z}$ bestimmen nun den in jeder Kanalhälfte vorliegenden Volumenstrom $\dot{V}_i$ und dadurch das anzustrebende Phasenverhältnis $\Phi = \dot{V}_{org.}/\dot{V}_{aq.}$ der eintretenden Pfropfenströmung am optimalen Betriebspunkt. Für die Experimente ohne Netz wird ein Phasenverhältnis von $\Phi \approx 2/1$ festgesetzt, während die Experimente mit eingesetztem Netzgewebe jeweils bei einem Phasenverhältnis von $\Phi \approx 7/3$ durchgeführt werden. Dies wird als ausreichende Näherung an die berechneten optimalen Verhältnisse angesehen, die wegen Temperatureinflüssen auf die Viskosität und vor allem durch Fertigungstoleranzen bzgl. der Geometrie bedingt auch nicht als fehlerfrei angesehen werden können.

Zur Überprüfung der hergeleiteten Konstruktionskriterien werden die berechneten Volumenströme mit Hilfe von Spritzenpumpen einem Y-Mischer zugeführt, woraufhin die erzeugte Pfropfenströmung durch einen der beiden Einlässe (vgl. Abbildung 4.7) in den Trennapparat eintritt. Die austretenden Ströme werden jeweils durch ein einstellbares Nadelventil gedrosselt und aufgefangen, um abschließend in einem Messzylinder den Volumenanteil der erwünschten Phase zu bestimmen, welcher als Maß für die Trennleistung verwendet wird. Die Nadelventile dienen lediglich dem Ausgleich von unterschiedlichen Druckverlusten in den Austrittskapillaren von 0,8 mm Innendurchmesser hinter dem Trennraum und werden händisch und iterativ auf eine möglichst saubere Trennleistung eingestellt. In einer ersten Versuchsreihe wird bei einem Gesamtvolumenstrom von 1 ml/min die Einlassseite der Pfropfenströmung (PTFE-Seite/Aluminiumseite) mit dem verwendetem Netzmaterial (Stahlnetz 25 µm Maschenweite/FEP-Netz 70 µm Maschenweite/kein Netz) permutiert. Eine zweite Versuchsreiche wird mit einem Gesamtvolumenstrom von 4 ml/min durchgeführt. In beiden Versuchsreihen wird jeweils ein Gesamtvolumen von 24 ml gefördert. Die Reproduzierbarkeit wird stets durch einen Wiederholungsversuch überprüft. Die erzielten Reinheiten der Phasen sind in Abbildung 4.8 dargestellt. Gemein ist allen

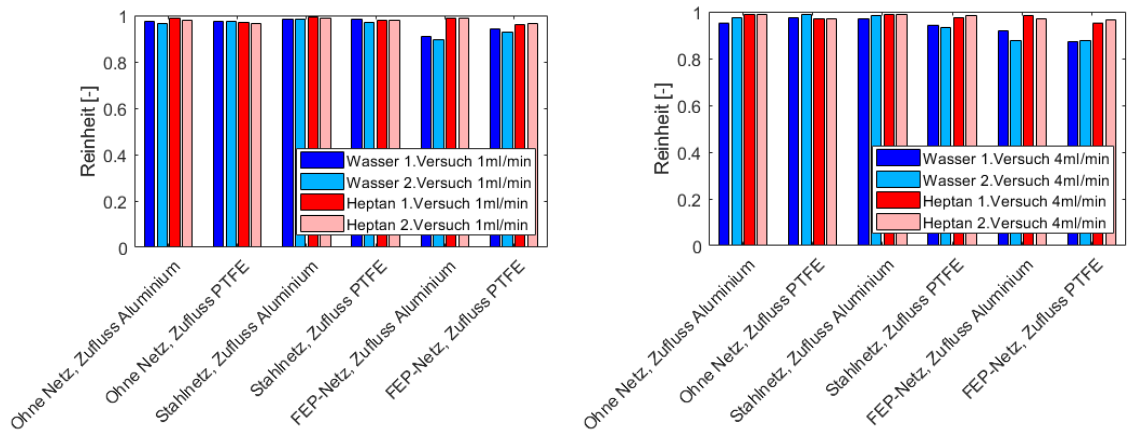


Abbildung 4.8: Erzielte Reinheiten von wässriger und organischer Phase bei einem Gesamtvolumenstrom von 1 ml/min (links) und 4 ml/min (rechts) unter Permutation von verwendeter Einlassseite und Netzmaterial.

Versuchen, dass in keiner Konfiguration beide Phasen komplett rein erhalten werden konnten. Die erzielten Reinheiten unterscheiden sich nicht signifikant. Das FEP-Netz erzielt jedoch tendenziell schlechtere Ergebnisse. Für den Zufluss der Strömung sollte präferiert die Aluminiumseite genutzt werden, obwohl auch hier der Unterschied zu den anderen Konfigurationen nicht sonderlich groß ist. Im Idealfall sollte für eine erfolgreiche Trennung kein Netzmaterial notwendig sein.

Genauer untersucht und weiterhin verglichen werden daher die beste Konfiguration ohne Netzmaterial (ohne Netz, Zuflussseite Aluminium) mit der besten Konfiguration mit Netzmaterial (Stahlnetz, Zuflussseite Aluminium). Die Ergebnisse dieser Versuchsreihe sind in Abbildung 4.9 dargestellt. Jeder Versuch wird durch zwei Wie-

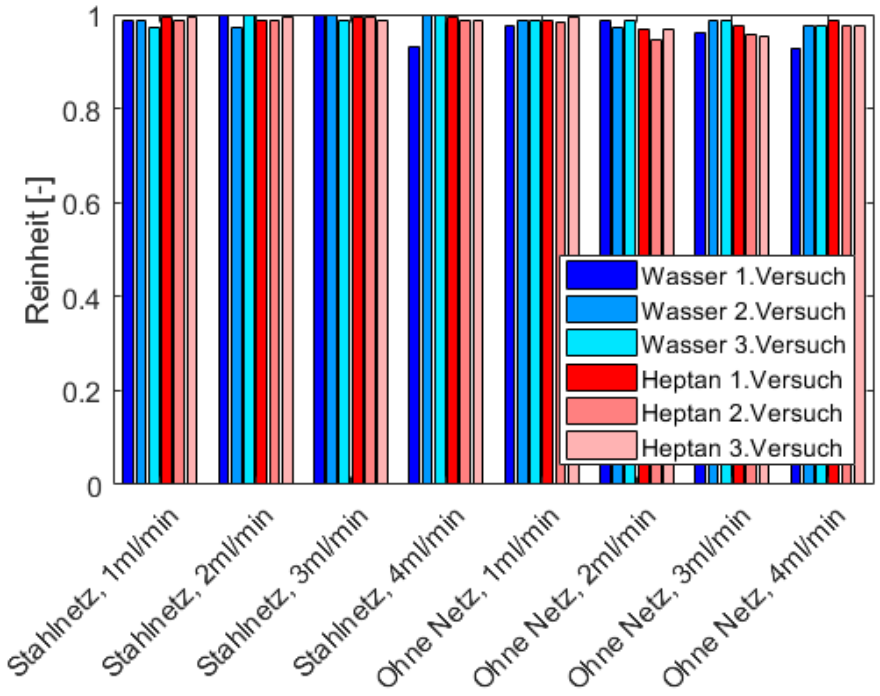
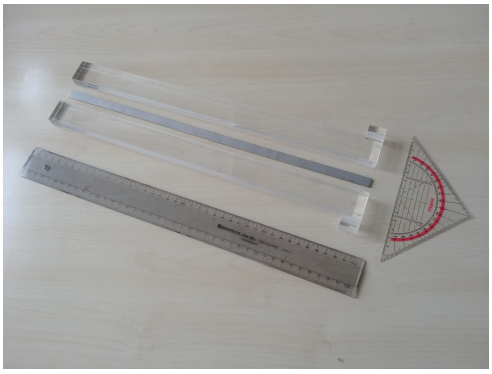


Abbildung 4.9: Erzielte Reinheiten von wässriger und organischer Phase bei einem Gesamtvolumenstrom von 1 ml/min bis 4 ml/min beim Zufluss auf der Aluminiumseite mit und ohne eingesetztem Stahlnetz.

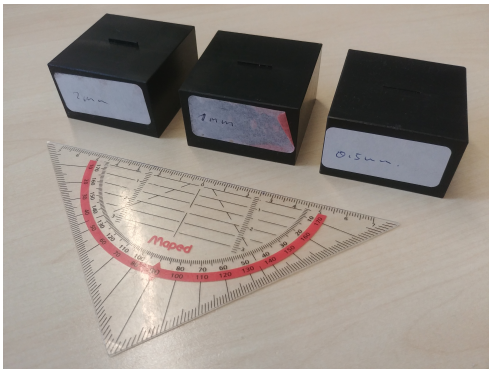
derholungsversuche in seiner Reproduzierbarkeit überprüft. Tendenziell erreicht die Konfiguration mit Netzgewebe wieder bessere Reinheiten, doch erneut kann trotz angepasster Phasenverhältnisse und optimalem Betriebsfenster in keinem Versuch eine vollständige Reinheit beider Phasen erreicht werden. Dies wirft die Frage auf, warum der Kapillardruck nicht ausreichend ist, um die Reinheit beider Phasen zu garantieren. Auffällig in den Experimenten ist, dass es große Zeiträume gibt, in denen die Trennung einwandfrei funktioniert, dann jedoch immer wieder einzelne Verunreinigungen in den Auslasskapillaren entstehen. Neben experimentellen Fehlerquellen, wie z.B. nicht konstant fördernden Pumpen, nicht optimal eingestellten Nadelventilen oder zu großen Fertigungstoleranzen der gefrästen Kanäle, können jedoch auch andere Gründe für die Verunreinigungen verantwortlich sein. Da der Phasentrenner optisch nicht einsehbar ist, wird im folgenden Abschnitt 4.2.2 in einem visuell zugänglichen Kanal die prinzipielle Stabilität einer Parallelströmung in Rechteckkanälen mit modifizierten Oberflächen untersucht.

4.2.2 Stabilität einer Parallelströmung in modifizierten Rechteckkanälen

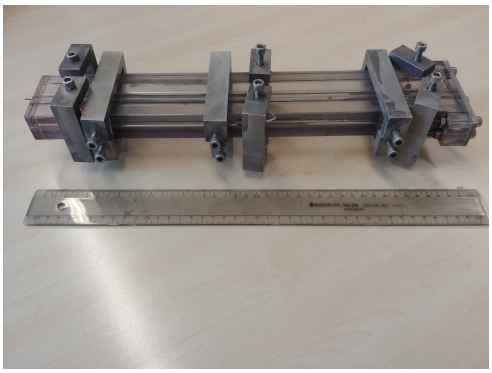
Für die Erklärung der in Unterabschnitt 4.2.1 beobachteten Verunreinigungen wird die allgemein Stabilität einer Parallelströmung in Rechteckkanälen in Frage gestellt. Die in der Literatur vorzufindenden Beschreibungen von Parallelströmungen beziehen sich stets auf sehr flache bzw. schmale Kanäle mit einer Spaltbreite von maximal 400 µm ([39], [38], [32], [112]), wohingegen der untersuchte Phasentrenner zur Reduzierung des auftretenden Druckverlustes Abmessungen von 1,6 mm · 2,2 mm besitzt. Zur Beurteilung der Stabilität einer Parallelströmung wird ein Rechteckkanal konstruiert, in dem die Strömung optisch erfasst werden kann, der aber zudem in seiner Geometrie variabel ist, um den Einfluss der verschiedenen Abmessungen studieren zu können. Die gefertigten Bauteile, sowie der zusammengesetzte Gesamtkanal sind in Abbildung 4.10 dargestellt. Die in Ausschnitt 4.10a abgebildeten Bauteile dienen



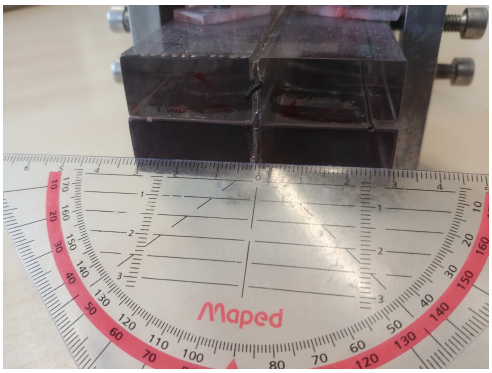
(a) Metallplatte zur Definition der Kanalbreite für eine Kanalhälfte und Seitenwände aus PC bzw. PMMA.



(b) Blöcke mit Noppen definierter Höhe zur einfachen Montage von jeweils einer Kanalhälfte.



(c) Zusammengesetzter Gesamtkanal aus je zwei Kanalhälften.



(d) Auslassöffnung des Kanals in die freie Umgebung.

Abbildung 4.10: Konstruierte Bauteile und Montagehilfen zur Erstellung eines transparenten Rechteckkanals variabler Geometrie.

der Konstruktion jeweils einer Kanalhälfte. Um die Montage zu vereinfachen werden Noppenblöcke verwendet, die in Ausschnitt 4.10b dargestellt sind. Die Noppen selber bilden effektiv das Negativ der zu konstruierenden Kanalgeometrie, wenn die Seitenwände des Kanals seitlich und die Metallplatte von oben an die Noppen angelegt werden. Durch Noppen verschiedener Höhe und Metallplatten unterschiedlicher Dicke können so reproduzierbar beliebige Kanalgeometrien definiert werden. Im Anschluss

fixieren Spannklemmen die Bauteile. Weitere Klemmen fügen den Gesamtkanal aus oberer und unterer Hälfte zusammen (Ausschnitt 4.10c). Um für die Stabilität der Strömung einen unterstützenden Effekt durch den Kapillardruck zu erzielen, sollen die Wände einer Kanalhälfte hydrophil sein, und die Wände der andere Seite aus einem hydrophoben Material bestehen. Für die Fertigung wird auf PC (Makrolon ©, Bayer) und PMMA als Material zurückgegriffen, welche durchsichtig und in ihrer normalen Form beide hydrophob sind. In einem Verfahren nach Jankowski et al. kann auf der Oberfläche von PC jedoch chemisch ein verzweigtes Iminpolymer verankert werden, an das in einem zweiten Schritt ein polymeres Maleinsäureanhydrid koppelt. [113] Die anschließende Hydrolyse der verbliebenen Anhydridgruppen durch Natronlauge erzeugt eine Vielzahl polarer Carbonsäuregruppen an der Oberfläche, welche die Benetzungseigenschaften bis in den hydrophilen Bereich verschieben. Die optischen Eigenschaften des PCs bleiben dabei weitgehend erhalten, während sich die chemische Beständigkeit sogar verbessert, da die Iminschicht durch ihre kovalenten Bindungen zu den Polymerketten des PCs wie eine zusätzliche Vernetzung wirkt. [114] [2] Die kovalenten Bindungen machen diese Oberflächenmodifikation zudem beständiger als andere Verfahren wie z.B. die Bestrahlung mit UV-Licht oder die Behandlung mit Sauerstoffplasma. Tabelle 4.3 gibt eine Übersicht zu den untersuchten Kanalgeometrien. In jeder Geometrie kommen sowohl das Stoffsystem

Tabelle 4.3: Untersuchte Geometrievarianten des Rechteckkanals.

B	H_{aq}	H_{org}
1,0 mm	1,0 mm	1,0 mm
1,0 mm	1,0 mm	1,5 mm
1,0 mm	1,0 mm	2,0 mm
1,0 mm	1,5 mm	1,0 mm
1,0 mm	1,5 mm	1,5 mm
1,0 mm	2,0 mm	1,0 mm

B	H_{aq}	H_{org}
1,5 mm	1,5 mm	1,5 mm
1,5 mm	1,5 mm	2,0 mm
1,5 mm	2,0 mm	1,5 mm
2,0 mm	2,0 mm	2,0 mm

Heptan-Wasser als auch Hexadecan-Wasser zum Einsatz. Dabei werden unter Variation der Volumenströme von Wasser und Organik in einem Bereich von jeweils 1,0 ml/min bis 5,0 ml/min insgesamt 25 Datenpunkte pro Geometrie und Stoffsystem erfasst. Obwohl die Schwerkraft in Mikroapparaten eine untergeordnete Rolle spielen sollte, wird die schwerere wässrige Phase stets in der unteren Kanalhälfte geführt. Die Strömungsverhältnisse werden während der Experimente zur späteren Auswertung aus einer oberen und einer unteren Perspektive gefilmt.

Bei der Auswertung der Videoaufnahmen sind dabei Reflexionen an den Berührstellen der transparenten Bauteile zu berücksichtigen, wie Abbildung 4.11 veranschaulicht. Die gleichzeitige Auswertung einer oberen und unteren Ansicht ermöglicht jedoch gezielt die Identifikation der im Kanal vorliegenden Strömungsform. Unterschieden werden dabei drei verschiedene Strömungsbilder, die beispielhaft an einem Kanal mit $B = 1{,}0\,\text{mm}$, $H_{aq} = 2{,}0\,\text{mm}$ und $H_{org} = 1{,}0\,\text{mm}$ für das Stoffsystem Heptan-Wasser in der jeweiligen oberen und unteren Ansicht in Abbildung 4.12 dargestellt sind. Eine instabile Strömung wird daran identifiziert, dass es zu Durch-

[2] PMMA ist durch dieses Verfahren prinzipiell ebenfalls modifizierbar. Je nach Hersteller und Lieferant wurden jedoch starke Unterschiede in der Reaktivität des ersten Schrittes und damit dem grundsätzlichen Erfolg des Verfahrens festgestellt.

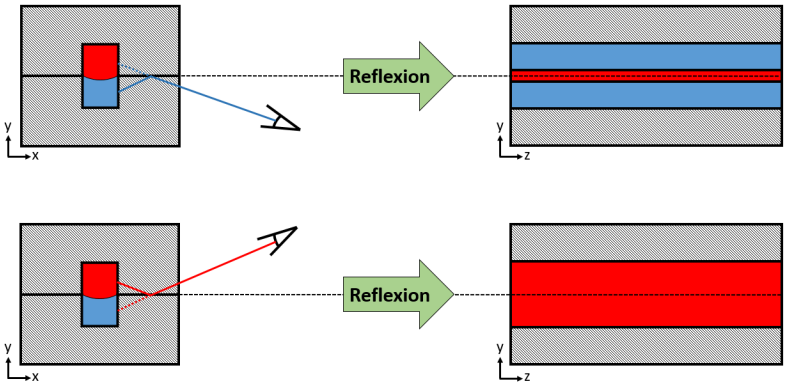
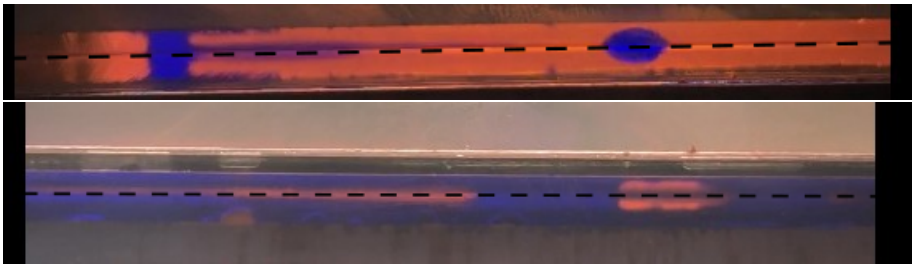
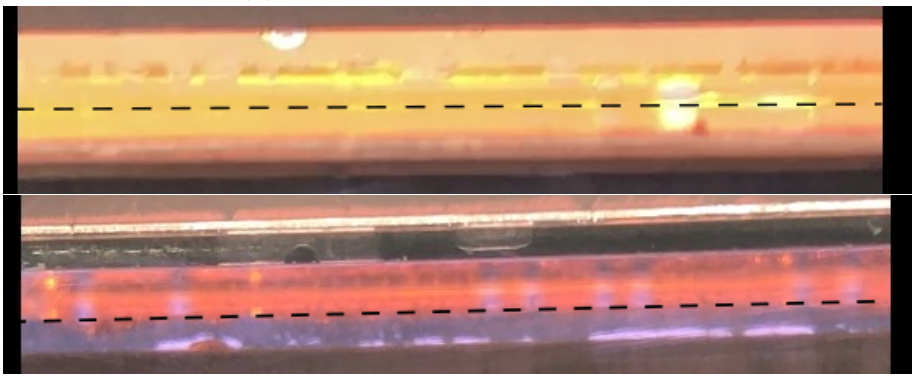


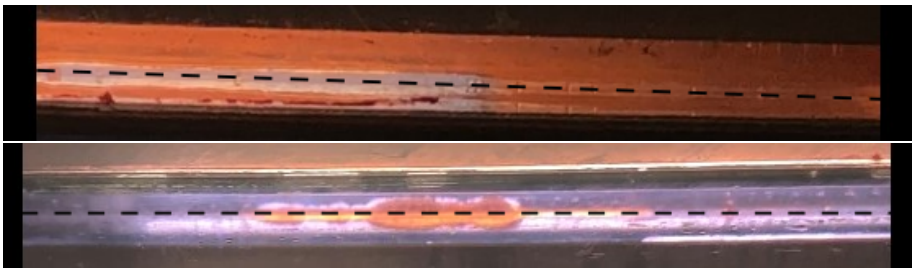
Abbildung 4.11: Veranschaulichung der auftretenden Lichtrefelxionen an den Berührstellen der transparenten Kanalwände mit vorliegendem Strömungsbild (links) und beobachtetem Strömungsbild (rechts).



(a) Instabiles Verhalten im Kanal.



(b) Tropfeneinschluss von Wasser in Organik.



(c) Ausgebildete stabile Parallelströmung.

Abbildung 4.12: Übersicht der unterschiedenen Strömungsformen im Rechteckkanal mit eingezeichneter Reflexionslinie (vgl. Abbildung 4.11) in der jeweiligen Ansicht von oben (obere Darstellung) und unten (untere Darstellung).

schlägen einer Phase bis zur Decke oder dem Boden der gegenüberliegenden Kanalhälfte kommt. So ist in Unterabbildung 4.12a beispielhaft in der oberen Ansicht das Durchschlagen von Wasser zu erkennen. Eine stabile Parallelströmung zeichnet sich hingegen dadurch aus, dass in der oberen Ansicht stets rote Organik an der Decke anliegt und in der unteren Ansicht stets blaues Wasser den Kanalboden benetzt (Unterabbildung 4.12c). Die scheinbaren Einschlüsse in der Kanalmitte sind durch die Reflexionen und das perfekt symmetrische Verhalten eindeutig auf Wölbungen der Grenzfläche in die gegenüberliegende Kanalhälfte zurückzuführen. Neben der stabilen Parallelströmung wurde jedoch auch eine Parallelströmung mit Tropfeneinschluss beobachtet. Hier liegen am Boden des Kanals die blaue wässrige Phase und an der Decke des Kanals die rote Organik an. In der unteren Ansicht von Unterabbildung 4.12b sind allerdings blaue Wasserbläschen zu beobachten, welche sich noch einmal durch eine dünne rote Organikschicht von der darunter fließenden wässrigen Kernphase abgrenzen. Dies kann nicht durch symmetrische Reflexionsphänomene einer gewölbten Parallelströmung erklärt werden.

Neben der reinen empirischen Aufnahme und Auswertung der Strömungsbilder soll auch eine Vorhersage möglich sein, unter welchen Bedingungen eine stabile Parallelströmung zu erwarten ist. Hierzu wird erneut als notwendige Bedingung angesehen, dass in beiden Phasen der gleiche Druckverlust $\frac{\partial p}{\partial z}$ vorliegt. Der postulierte optimale Betriebszustand setzt dabei jedoch voraus, dass beide Phasen jeweils nur den Querschnitt ihrer eigenen Kanalhälfte durchströmen. Abbildung 4.13 zeigt allerdings, dass sich die Phasengrenzfläche auch in die jeweils gegenüberliegende Kanalhälfte hineinwölben kann. Wird vom optimalen Betriebspunkt aus z.B.

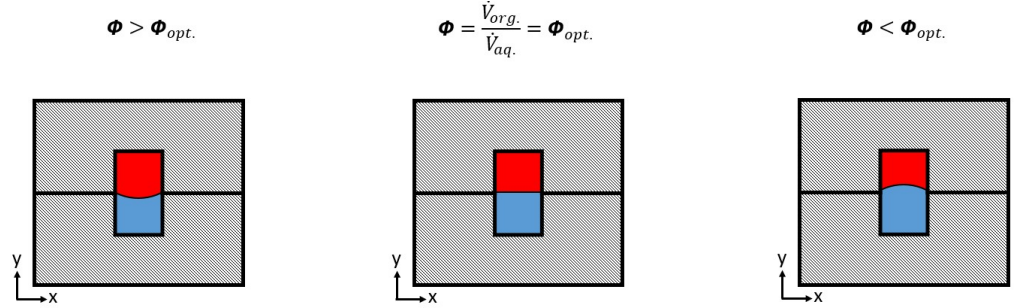


Abbildung 4.13: Wölbung der Grenzfläche in die gegenüberliegnde Kanalhälfte bei Abweichung vom optimalen Betriebspunkt durch zu hohen Durchfluss an Organik (links) oder Wasser (rechts).

der Volumenstrom an Organik erhöht, würde dies einen erhöhten Druckverlust in der oberen Kanalhälfte erzeugen. Um dies auszugleichen wölbt sich die Grenzfläche nach unten. Der Organik steht dann mehr Platz zur Verfügung, was den oberen Druckverlust absenkt, das Wasser wiederum einengt und so den unteren Druckverlust anhebt, bis beide wieder den gleichen Wert erreichen. Wie stark sich die Grenzfläche dabei maximal in die andere Kanalhälfte wölben kann, wird durch die Kontaktwinkel der Phasen an den Kanalwänden bestimmt. Die verwendeten Stoffdaten und gemessenen Kontaktwinkel sind in Anhang F.1 aufgeführt. Wäre eine stärkere Wölbung notwendig, als der Kontaktwinkel erlauben würde, oder müsste die Kontaktlinie sich von der Berührstelle der beiden Kanalhälften lösen, so trifft dieses einfache Modell keine Aussage über die Stabilität einer Parallelströmung. Die Vorhersage ist sehr konservativ, jedoch durch die analytische Berechnung der Strö-

mungsfelder nach Gl. 4.6 bis Gl. 4.12 bei bekannter Position der Grenzfläche einfach zu treffen. Als Grenzflächenposition wird eine zylinderförmige Wölbung mit maximal möglichem Kontaktwinkel an den Kanalwänden in die jeweils gegenüberliegende Kanalhälfte angenommen. Hieraus ergibt sich für jede Geometrie und jedes Stoffsystem ein charakteristisches maximales und minimales Volumenstromverhältnis Φ, bei dessen Einhaltung eine stabile Parallelströmung vorliegen sollte. Die mathematische Korrektheit der analytischen Gleichungen mit ausgelenkter Phasengrenzfläche wird durch zweidimensionale Berechnungen in OpenFOAM validiert, bei denen ein an die vordefinierte Grenzfläche angepasstes Rechengitter verwendet und kein Transport der Grenzfläche vorgenommen wird. Ein hierfür verwendetes Rechengitter ist beispielhaft in Anhang F.2 einzusehen. Die berechneten Volumenstromverhältnisse Φ des analytischen Modells weichen im Mittel um 2,0% und maximal um 7,9% von den numerischen Ergebnissen aus OpenFOAM ab. Ein Vergleich der so berechneten Strömungsprofile ist in Abbildung 4.14 enthalten. Die Prüfung der empirischen

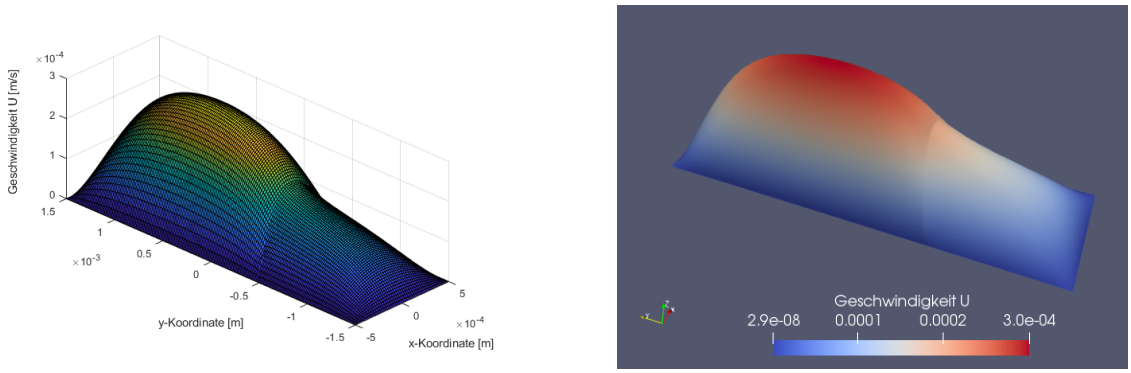


Abbildung 4.14: Berechnete Strömungsprofile über den Kanalquerschnitt durch analytische Lösung mit Reihenabbruch ab $k > 10$ nach Gl. 4.6 bis Gl. 4.12 (links) und numerische Lösung durch OpenFOAM (rechts) in einem Kanal mit $B = 1{,}0\,\text{mm}$, $H_{aq} = 1{,}0\,\text{mm}$ und $H_{org} = 2{,}0\,\text{mm}$ für das Stoffsystem Heptan-Wasser bei maximal nach oben ausgelenkter Grenzfläche.

Korrektheit des Vorhersagemodells über Φ erfolgt durch die Experimente. Für beide Stoffsysteme sind die experimentellen Ergebnisse der Videoauswertungen bei einer Kanalgeometrie von $B = 2{,}0\,\text{mm}$, $H_{aq} = 2{,}0\,\text{mm}$ und $H_{org} = 2{,}0\,\text{mm}$ in Abbildung 4.15 veranschaulicht. Die Stabilitätsdiagramme aller weiteren durchgeführten Versuchsreihen sind in Anhang F.3 zur Ansicht aufgeführt. Stabile Parallelströmungen sind durch einen grünen Kreis gekennzeichnet, während instabile Strömungen durch ein rotes Kreuz dargestellt sind. Parallelströmungen, die jedoch Tropfenbildung aufweisen, sind durch gelbe Quadrate markiert. Die Modellvorhersage ist durch gestrichelte Linien dargestellt, die das maximale und minimale Volumenstromverhältnis Φ repräsentieren. Im Bereich zwischen den Linien wird demnach eine stabile Parallelströmung erwartet. Beim Vergleich aller durchgeführten Versuchsreichen können dabei folgende Beobachtungen gemacht werden:

- Speziell in den Kanälen kleiner Breite B treten häufig Instabilitäten auf. Es scheint so, als würde die eintretende Pfropfenströmung nicht gebrochen werden können, obwohl gerade ein enger Kanal früh einen Kontakt zwischen disperser Phase und Wandmaterial erzwingen sollte. Das Modell sagt in diesen Bereichen Stabilität voraus, die jedoch nicht vorliegt. Die breiteren Kanäle mit $B \geq$

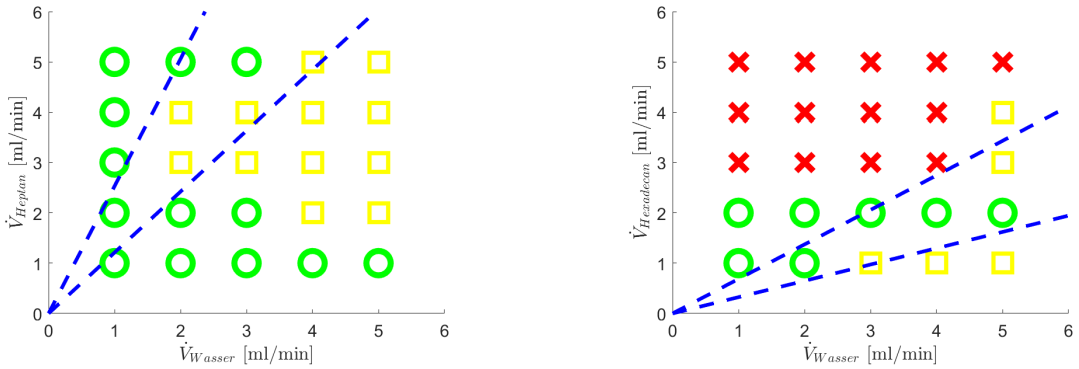


Abbildung 4.15: Vergleich der Strömungsbilder in einem Kanal mit $B = 2{,}0\,\mathrm{mm}$, $H_{aq} = 2{,}0\,\mathrm{mm}$ und $H_{org} = 2{,}0\,\mathrm{mm}$ bei Verwendung von Heptan-Wasser (links) und Hexadecan-Wasser (rechts): Stabile Parallelströmung (grüne Kreise), Parallelströmung mit Tropfenbildung (gelbe Quadrate), instabile Strömung (rote Kreuze).

1,5 mm weisen hingegen stets ein ausgeprägtes Stabilitätsgebiet auf, das auch gut vorhergesagt wird.

- Sofern ein Stabilitätsgebiet auftritt, wird dies durch das Vorhersagemodell in seiner Größe oft unterschätzt. Dies wird darauf zurückgeführt, dass stabilisierende Effekte durch den Kapillardruck einer gewölbten Grenzfläche nicht berücksichtigt wurden.

- Tropfenbildung an der freien Grenzfläche tritt ohne erkennbares Muster auf. Es bilden sich sowohl Organiktropfen im Wasser als auch umgekehrt.

Da die Dichteunterschiede beider Stoffsysteme sehr ähnlich sind, wird an einer festgesetzten Kanalgeometrie von $B = 1{,}0\,\mathrm{mm}$, $H_{aq} = 2{,}0\,\mathrm{mm}$ und $H_{org} = 1{,}0\,\mathrm{mm}$ in drei weiteren Versuchsreihen dem Wasser Kaliumiodid zugesetzt, welches in beträchtlichen Mengen löslich ist und somit die Dichte des Wassers auf bis zu $2300\,\mathrm{kg/m^3}$ steigern kann. Eine größere Dichtedifferenz sollte das Stabilitätsgebiet vergrößern, die Modellvorhersage jedoch wenig beeinflussen, da lediglich die Viskosität der beiden Fluide in die Berechnung des Stabilitätsgebietes eingeht. Der erwartete stabilisierende Trend ist jedoch nicht zu beobachten. Bis auf die Kombination von Hexadecan mit Wasser bei einer Dichte von $\rho = 1400\,\mathrm{kg/m^3}$ verhalten sich alle Stabilitätsgebiete unauffällig.

Die durchgeführten Versuchsreihen und gemachten Beobachtungen ermöglichen nun eine Deutung der in Unterabschnitt 4.2.1 beobachteten Verunreinigungen (Abbildung 4.9): Während eine Parallelströmung durchaus die angestrebte Strömungsform im Kanal sein kann, ist Tropfenbildung in der jeweils anderen Phase weiterhin möglich. Dieses Phänomen ist aus der Literatur soweit nicht ersichtlich, doch sind die dort dokumentierten Anwendungen in Mikrokanälen auch stets mit Spaltbreiten von maximal $400\,\mu\mathrm{m}$ durchgeführt worden ([39], [38], [32], [112]). Gelangen die gebildeten Tropfen bei anschließender Auftrennung der Ströme in den falschen Auslass, so erzeugen sie dort zusätzliche Grenzflächen. Die Anwesenheit einer zweiten Phase beeinflusst den hydrodynamischen Druckverlust in der Kapillare. Verdeutlicht wird der Effekt an einem Beispiel. Für beide Phasen werden wasserähnliche Stoffdaten und eine Grenzflächenspannung von $\sigma = 35\,\mathrm{mN/m}$ angenommen. Dann

erzeugt ein Pfropfen disperser Phase in einer Auslasskapillare von $d_i = 0{,}8\,\text{mm}$ bei einem Volumenstrom von $\dot{V} = 2\,\text{ml/min}$ nach Jovanovic et al. (siehe Gl. 3.83, [107]) bereits einen zusätzlichen Druckverlust in der Größenordnung von 10 Pa. Damit die Trennung aufrecht erhalten werden kann, muss durch Wölbung der Fluidgrenzfläche im Spalt der Trennkammer ein Kapillardruck entstehen, der diesen zusätzlichen Druckverlust kompensieren kann. Wird eine Spaltbreite von $B = 1{,}6\,\text{mm}$ und ein Kontaktwinkel von 45 ° angenommen, so entsteht ein Kapillardruck von ca. 15 Pa. Die ähnlichen Größenordnungen zeigen, dass eine nicht perfekte Ausbalancierung der hydrodynamischen Druckverluste schnell dazu führen kann, dass auch in den anderen Fluidausgang die falsche Phase eintritt. Die Phasentrennung kann demnach zwar über einen gewissen Zeitraum hinweg sauber betrieben werden, eine saubere Phasentrennung wird sich jedoch nur sehr unwahrscheinlich wieder einstellen, sobald die ersten Verunreinigungen aufgetreten sind. Dieses metastabile Verhalten wurde bereits numerisch in Abschnitt 4.1 beobachtet. Das Auftreten der beobachteten Tropfenbildung an der Fluidgrenzfläche macht die Phasentrennung einer Pfropfenströmung durch Umwandlung zu einer Parallelströmung in breiten Kanälen daher zu einem unattraktiven Konzept. Breitere Kanäle werden jedoch für eine Reduzierung der zu überwindenden Druckverluste und eine Steigerung des Durchsatzes für notwendig erachtet, da Parallelisierungskonzepte bisher nur eine begrenzte Zahl an Kapillaren gleichzeitig betreiben können (siehe Unterabschnitt 2.2.5). In besonders engen Kanälen kann die Parallelströmung hingegen durchaus effektiv eingesetzt werden, wie die Literatur dokumentiert.

Zur Aufreinigung einer Pfropfenströmung durch Umwandlung in eine Parallelströmung sind daher weiterhin enge Kanäle und Spalte notwendig, in diesem Fall jedoch nicht, um instationäre Druckverluste auszugleichen (vgl. Abschnitt 4.1), sondern um das identifizierte metastabile Betriebsfenster in ein echt stabiles Betriebsfenster zu überführen. Leichte Verunreinigungen in den Auslässen müssen akzeptiert werden und die dadurch zeitweise erhöhten hydrodynamischen Druckverlust erzwingen zum Ausgleich einen höheren Kapillardruck im Spalt. Dies stellt erhöhte Anforderungen an das Förderkonzept in den Kapillaren, da in engen Kanälen auch höhere Druckverluste zu erwarten sind. Der Verzicht auf Porenkämme und Membranseparatoren ermöglicht jedoch weiterhin den Einsatz von Suspensionen. Ein mögliches Förderkonzept, welches auf Suspensionen und externen Magnetfelder basiert, wird im folgenden Abschnitt 4.3 erarbeitet und experimentell untersucht.

4.3 Förder- und Sensorkonzept für Pfropfenströmungen durch Magnetfelder

Damit die Mikroverfahrenstechnik im Vergleich zum Einsatz bisheriger Extraktionsapparate konkurrenzfähig wird, sind Parallelisierungsstrategien notwendig, die das Betreiben vieler Extraktionsstränge gleichzeitig ermöglichen. Der Einsatz von jeweils $n + 1$ Pumpen für jeden n-stufigen Extraktionsstrang stellt allerdings eine zu große Investitionssumme dar. Das Betreiben aller Extraktionseinheiten einer Stufe mit nur jeweils einer Pumpe führt hingegen zu Ungleichverteilungen in der Flüssigkeitsbelastung, da Fertigungstoleranzen zu ungleichen Druckverlusten pro Strang führen. Bisherige Strategien diese Unterschiede auszugleichen sind in Unterabschnitt 2.2.5 zusammengefasst. In diesem Abschnitt soll ein weiteres Konzept untersucht werden,

welches mit Magnetfeldern die Pfropfenströmung erfasst und gleichzeitig beeinflussen kann. Da laut Antweiler et al. ([25]) pro Strang stets mindestens ein Sensor und ein Aktuator notwendig sind, bietet die Kombination beider Funktionalitäten großes Potenzial zur Kostenreduktion. Voraussetzung ist der Einsatz eines Ferrofluids in der Pfropfenströmung. Da der Wertstoff bei einer Extraktion häufig sowieso noch aus dem Extrakt zurückgewonnen werden muss, steht dem Einsatz eines Ferrofluids an dieser Stelle wenig im Weg. Die Bauteile dürfen jedoch nicht zu Verstopfung neigen, was gerade den Einsatz von Membranen zur Phasentrennung hinfällig macht.

Magentismus wurde mit Hilfe von Ferrofluiden in der Mikroverfahrenstechnik bereits vielfach zu Förderzwecken eingesetzt. Kurtoglu et al. montierten auf drehenden Walzen Permanentmagnete und platzierte diese neben einer Kapillare. [115] Die Rotation simulierte einen einzelnen Magneten, der sich entlang der Kapillare bewegt und so einen Pfropfen nach vorne zog. Bruno und Ciocanel untersuchten durch Simulationen die Bewegung eines Ferrofluids innerhalb einer Kapillare durch elektromagnetische Spulen, realisierten dies im Experiment jedoch auch nur durch einen bewegten Permanentmagneten. [116] Hatch et al. hingegen konstruierten einen Ringspalt, in dem durch einen äußeren rotierenden Magneten ein Pfropfen aus Ferrofluid im Kreis gezogen wurde. [117] Durch einen stationären Magneten wird zwischen Zu- und Ablauf in den Ringspalt eine weitere Ansammlung Ferrofluid an Ort und Stelle gehalten. Stetige Koaleszenz und Ablösung von Pfropfen aus dieser Ansammlung erzeugten im Kreiskanal eine Förderwirkung der umliegenden kontinuierlichen Phase. Der Ansatz von Lung-Ming et al. basiert auf einem ähnlichen Prinzip. [118] Der Einsatz eines einzelnen Pfropfens Ferrofluid ist ebenfalls möglich. Liu et al. ließen diesen innerhalb einer Pumpkammer durch einen externen Magneten eine Kreisbewegung ausführen. [119] Die geschickte Platzierung von gekrümmten Leitwänden in der Kammer sorgte für eine Verformung des Pfropfen, der bei jeder Umdrehung neues Fluid ansaugte und altes Fluid aus der Kammer ausstieß. Rotierende externe Bauteile werden in großer Stückzahl jedoch wartungsintensiv und teuer. Starre Bauteile sind hingegen weniger fehleranfällig. Die Kombination von Elektromagnetismus und Ferrofluiden als flexible Membran in Membranpumpen wird soweit bereits genutzt. [120] Weitere Anwendungen von Magnetfeldern in Kombination mit Ferrofluiden finden sich in der Übersichtsarbeit von Yang. [121]

Das Funktionsprinzip des hier untersuchten Ansatzes basiert auf dem gleichzeitigem Einsatz von Gleich- und Wechselstrom in elektromagnetischen Spulen. Für Wechselstrom wirkt eine Spule wie ein Blindwiderstand, da sich das Magnetfeld der Spule ständig auf und wieder abbauen muss. Wie groß der Blindwiderstand einer Spule ist, variiert dabei unter anderem mit dem Material innerhalb der Spule. Wird die Spule um eine Kapillare gewickelt, ändert sich demnach die Induktivität L in Abhängigkeit des Fluids, welches sich gerade in der Spule befindet, wie Abbildung 4.16 veranschaulicht. Wird die Induktivität kontinuierlich gemessen, kann hierüber die Strömung vermessen und die Spule als Sensor genutzt werden. Gleichstrom erzeugt in einer Spule hingegen ein konstantes magnetisches Feld, welches Ferrofluide anzieht. Dies kann gezielt dazu genutzt werden, um Pfropfen in die Spule hineinzuziehen (linke Spule). Bei vollständigem Eintritt des Pfropfens kann der Gleichstrom abgeschaltet werden und der Pfropfen kann die Spule, der vorliegenden Strömung folgend, ungehindert verlassen (rechte Spule). Effektiv wird eine unterstützende Förderwirkung auf die Strömung erzeugt. Eine alternative Betriebsweise lässt den Pfropfen ungehindert in die Spule eintreten und erzeugt beim Austritt aus

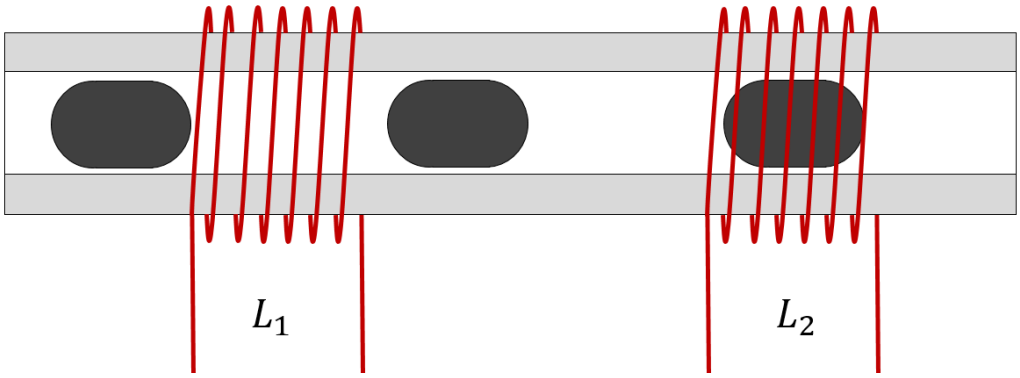


Abbildung 4.16: Prinzipskizze des ferromagnetischen Förderprinzips: Spule ohne ferrofluidischem Pfropfen (links) und Spule mit ferrofluidischem Pfropfen (rechts) mit messbarem Unterschied zwischen den Induktivitäten L_1 und L_2.

der Spule ein konstantes Magnetfeld, welches den Pfropfen bremst und somit einen Druckverlust in der Strömung erzeugt. Die korrekten Schaltpunkte für die anzulegende Gleichspannung können über die Messung der Spuleninduktivität mit Hilfe der hochfrequenten Wechselspannung bestimmt werden. Die Kombination mehrerer Spulen mit bekanntem Abstand zueinander erlaubt dabei die Unterscheidung zwischen langen Pfropfen mit hoher Geschwindigkeit und kurzen Pfropfen mit langsamer Geschwindigkeit in der Kapillare. Dadurch agiert dieses Konzept gleichzeitig als Sensor und Aktuator. Es weist zudem die Flexibilität auf, sowohl eine Förderwirkung wie auch einen Druckverlust erzeugen zu können. Die Förder- und Bremswirkung muss dabei nicht groß ausfallen, sondern lediglich die Gleichverteilung der Fluidströme unter den parallelisierten Strängen garantieren können, wenn jede Fluidphasen insgesamt mit nur einer Pumpe gefördert wird. Grundlage hierfür ist zunächst die korrekte Detektion der Pfropfen und ihrer Position relativ zur Spule durch eine angelegte Wechselspannung $\tilde{U}_0$. Die direkte Messung von Induktivitäten ist allgemein schwierig, wohingegen auch kleinste Spannungen noch sehr genau gemessen werden können. Daher wird auf eine Maxwell-Wien-Brücke zurückgegriffen. Eine typische Maxwell-Wien-Brücke mit den vier Impedanzen Z_i ist in Abbildung 4.17 dargestellt. Die gemessene Brückenspannung $\tilde{U}_B$ ist dabei nach Gl. 4.13 von den Werten der Impedanzen Z_i abhängig.

$$\tilde{U}_B = \tilde{U}_0 \cdot \frac{Z_1 \cdot Z_4 - Z_2 \cdot Z_3}{(Z_1 + Z_2) \cdot (Z_3 + Z_4)} \tag{4.13}$$

mit

$$Z_1 = R_1 + i \cdot \omega \cdot L \qquad Z_2 = R_2$$

$$Z_3 = R_3 \qquad Z_4 = \frac{R_4}{1 + i \cdot \omega \cdot C \cdot R_4}$$

Eine Brückenschaltung wird abgeglichen genannt, wenn unabhängig von der Amplitude und Frequenz der Erregerspannung $\tilde{U}_0$ keine Brückenspannung $\tilde{U}_B$ gemessen werden kann. Dies ist der Fall, falls $R_4 = R_2 \cdot R_3 / R_1$ und $L = R_2 \cdot R_3 \cdot C$ gilt. Ermöglicht wird der exakte Abgleich der Brücke durch die Nutzung von Drehpotentiometern für die Widerstände R_2 und R_4. Ändert sich in einer abgeglichenen Brücke die

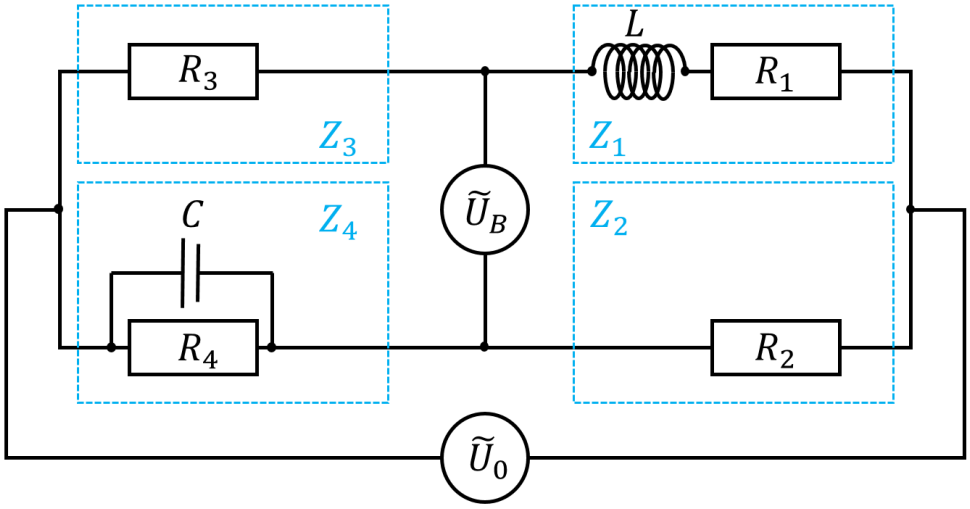


Abbildung 4.17: Schaltbild einer Maxwell-Wien-Brücke mit den vier Impedanzen Z_i, der Erregerspannung $\tilde{U}_0$ und der gemessenen Brückenspannung $\tilde{U}_B$.

Induktivität L im Laufe einer Messung, z.B. durch die Anwesenheit eines Ferrofluids im Spulenkern, so ist die Brücke dadurch nicht mehr abgeglichen und es kann eine resultierende Brückenspannung $\tilde{U}_B$ gemessen werden. Ist der Pfropfen vollständig aus der Spule ausgetreten und erreicht die Induktivität L ihren Ausgangswert, verschwindet die Brückenspannung wieder. Zur optimalen Auslegung der Brücke wird die zu erwartende Brückenspannung $\tilde{U}_B$ bei Auslenkung der Induktivität von ihrem Ausgangswert L_0 auf L_{Fe} nach Gl. 4.14 betrachtet.

$$\tilde{U}_{B,L_0 \to L_{Fe}} = \tilde{U}_0 \cdot \frac{R_3 \cdot i \cdot \omega \cdot (L_{Fe} - L_0)}{(R_1 + R_3 \cdot i \cdot \omega \cdot L_{Fe}) \cdot (R_1 + R_3 \cdot i \cdot \omega \cdot L_0)} \tag{4.14}$$

Die Widerstände R_1 und R_3 sind allgemein klein zu wählen. Dominiert wird die zu messende Spannung von der Differenz der Induktivitäten $L_{Fe} - L_0$, welche maßgeblich von der Beschaffenheit das Ferrofluids abhängt. Die Herstellung des Ferrofluids erfolgt nach Vorgaben von Knebel et. al ([122]) und ist in Anhang G.1 erläutert. Um die entstehende Brückenspannung $\tilde{U}_0$ gut von anderem Messrauschen unterscheiden zu können wird das Messsignal fouriertransformiert und gezielt auf den Ausschlag bei der angelegten Erregerfrequenz geachtet. Die verwendete LabJack-Messkarte erlaubt hierfür eine Messfrequenz von knapp unter 50 kHz. Für die Brückenschaltung wurden verschiedene Spulengeometrien in Betracht gezogen. Die besten experimentellen Ergebnisse lieferte eine Schaltung, deren Parameter in Tabelle 4.4 zusammengefasst sind. Die verschiedenen Induktivitäten wurden vor dem Einbau der Spule

Tabelle 4.4: Parameter der verwendeten Brückenschaltung.

Parameter	Wert	Parameter	Wert	Kernmaterial der Spule
$R_1 = R_3$	$5\,\Omega$	L_0	$1{,}184 \cdot 10^{-4}\,\mathrm{H}$	Luft
$R_2 = R_4$	$789\,\Omega$	$L_{Fe,1}$	$1{,}196 \cdot 10^{-4}\,\mathrm{H}$	Ferrofluid in Pipette
C	$3 \cdot 10^{-8}\,\mathrm{F}$	$L_{Fe,2}$	$2{,}284 \cdot 10^{-4}\,\mathrm{H}$	Schraubendreher

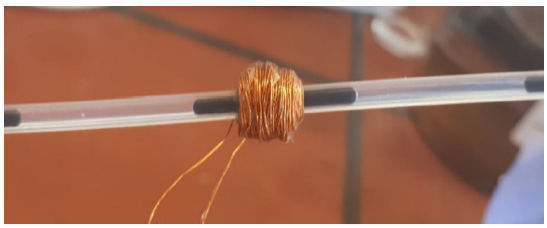
mit Hilfe eines sensiblen LCR-Meters vermessen, das sich jedoch nur für stationäre Messungen eignet und hier lediglich zu Validierungszwecken diente. Nach Abgleich der Brücke auf eine Erregerspannung von 5 V mit einer Frequenz von 20 kHz wurde durch Einführen eines Schraubendrehers eine Brückenspannung von 0,9 V gemessen, während Gl. 4.14 einen Wert von 0,78 V ergibt. Das Einführen des in einer Pipette aufgezogenen Ferrofluids erzeugte einen Messwert von 0,016 V bei einer erwarteten Spannung von 0,012 V. Auszüge der Messsoftware zu diesen Messungen sind in Anhang G.2 enthalten. Die Übereinstimmungen sind hinreichend gut und validieren den experimentellen Aufbau.

Die tatsächliche Detektion einer Pfropfenströmung erfolgt mit Cyclohexan als kontinuierliche Phase in einer FEP-Kapillare mit 1,6 mm Innen- und 3,2 mm Außendurchmesser. Für eine zeitaufgelöste Überwachung der Strömung müssen die 49000 Messpunkte pro Sekunde auf n einzelne Fouriertransformationen aufgeteilt werden. Je größer n gewählt wird, desto häufiger steht pro Sekunde die aktuelle Brückenspannung $\tilde{U}_B$ zur Bestimmung der Pfropfenposition zur Verfügung, das Messrauschen wird allerdings auch weniger effektiv unterdrückt je kleiner die Datenpakete pro Fouriertransformation gewählt werden. Ein Vergleich von 10 und 100 Fouriertransformationen pro Sekunde ist in Anhang G.2 enthalten. Zur eindeutigen Detektion werden unter gegebenen Umständen 4900 Datenpunkte pro Transformation für notwendig erachtet. Abbildung 4.18 zeigt, dass hierdurch lange Pfropfen sehr gut detektiert werden können, wohingegen kurze Pfropfen keine Unterscheidung zwischen disperser und kontinuierlicher Phase zulassen. Dies wird eindeutig auf die geometrischen Abmessungen der Spule zurückgeführt. Die erfolgreiche Detektion der langen Pfropfen verifiziert jedoch die prinzipielle Anwendbarkeit des Konzeptes. Eine nachfolgende zweite Fouriertransformationen auf die in Unterabbildung 4.18b dargestellten Daten könnte die Frequenz zugänglich machen, mit der die Pfropfen der Strömung aufeinander folgen, und so das gezielte Zuschalten der Gleichspannung zu Förderzwecken optimal steuerbar machen.

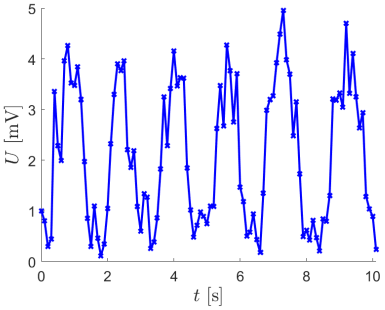
Wie die Stofftransportkorrelationen der Literatur darlegen (siehe Unterabschnitt 2.2.3) und auch eigene Ergebnisse bestätigen (siehe Abschnitt 4.4) besitzen lange Pfropfen jedoch einen verminderten Stofftransport gegenüber kurzen Pfropfen. Sollte die Detektion kürzerer Pfropfen nicht möglich sein, erscheint das vorgestellte Förderkonzept wenig erfolgsversprechend. Hierzu sind Modifikationen am experimentellen Aufbau notwendig, die im Rahmen dieser Arbeit nicht weiter untersucht werden konnten. Auf theoretischer Ebene können jedoch die dabei zu überwindenden Hindernisse diskutiert werden. Zunächst ist die Spule genauer zu betrachten. Für die erfolgreiche Detektion erscheint es zwingend notwendig, dass die Spule eine kleinere Länge als der durchlaufende Pfropfen besitzt. Dies limitiert die Spulenlänge l bei Nutzung von Kapillaren mit 1,6 mm Innendurchmesser auf etwa 3 mm. Für eine gute Detektierbarkeit von $\tilde{U}_B$ sollte allerdings eine hohe Grundinduktivität L_0 angestrebt werden, was nach Gl. 4.15 wiederum große Abmessungen der Spule voraussetzt.

$$L_0 \approx \mu_0 \cdot \mu_r \cdot A \cdot N^2/l \tag{4.15}$$

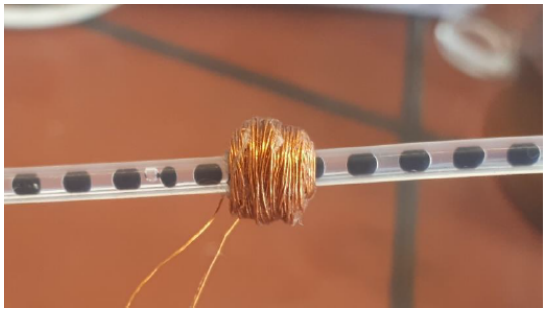
Viele Windungen N erhöhen die Induktivität, steigern allerdings auch das eingenommene Volumen der Spule. Dünnerer Draht erlaubt dabei eine Vielzahl an Windungen, steigert jedoch auch den elektrischen Widerstand R_1 durch die begrenzte Querschnittsfläche für den Stromfluss. Bei sehr dünnem Kupferdraht von 0,1 mm Durchmesser könnte unter optimalen Bedingungen eine vierlagige Spule mit



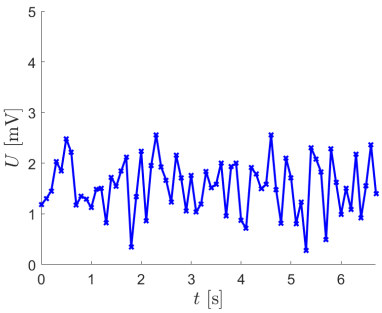
(a) Strömung aus Ferrofluid und Cyclohexan mit einer Pfropfenlänge von ≈20 mm.



(b) Ermittelte Brückenspannung $\tilde{U}_B$ für lange Pfropfen.



(c) Strömung aus Ferrofluid und Cyclohexan mit einer Pfropfenlänge von ≈3 mm.



(d) Ermittelte Brückenspannung $\tilde{U}_B$ für kurze Pfropfen.

Abbildung 4.18: Detektion von langen und kurzen Pfropfen durch die entwickelte Messsoftware bei 10 Fouriertransformationen zu jeweils 4900 Datenpunkten pro Sekunde.

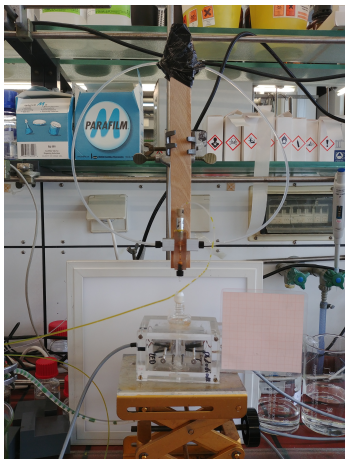
$l = 3\,\text{mm}$ insgesamt 120 Windungen und nach Gl. 4.15 eine Grundinduktivität von $7{,}6 \cdot 10^{-5}\,\text{H}$ erreichen, besäße aber bereits einen Widerstand von etwa $3\,\Omega$. In dieser Arbeit konnten lediglich $l = 7\,\text{mm}$, $N = 214$ und $L_0 = 6{,}6 \cdot 10^{-5}\,\text{H}$ mit $R_1 = 2{,}6\,\Omega$ realisiert werden. Neben der Spulengeometrie trägt auch die Erregerspannung durch ihre maximale Amplitude und ihre Frequenz zur Detektierbarkeit bei. Hier ist jedoch die Kopplung von L_0, R_1 und ω nach Gl. 4.14 zu berücksichtigen, da die Fouriertransformation im Optimalfall den Betrag $|\tilde{U}_B|$ zurückgibt und eine Variation der Parameter immer Amplitude und Phase von $\tilde{U}_B$ gleichzeitig beeinflusst. Zuletzt soll durch Zuschalten einer Gleichspannung im richtigen Moment ein möglichst starkes Magnetfeld in der Spule erzeugt werden, um durchfließende Pfropfen effektiv beeinflussen zu können. Das erzeugte Magnetfeld einer Spule ist allgemein proportional zur Windungszahl N und der durchfließenden Stromstärke I, was die Optimierung der Spulengeometrie weiter erschwert. Eine weitere Verbesserung der Detektierbarkeit kann durch eine stärkere Beladung des Ferrofluids mit Magnetit erreicht werden. Die maximale Beladung des Ferrofluids ist jedoch von der konkreten Anwendung abhängig, da die anschließende Rückgewinnung des Wertstoffes nach der Extraktion auf den konkreten Wertstoff abgestimmt sein muss. Zuletzt verbessert eine höhere Messfrequenz die Detektierbarkeit. Um das entwickelte Konzept jedoch günstig zu halten, ist Spezialelektronik mit besonders hohen Messfrequenzen wenig attraktiv. Eine signifikant große Steigerung über die bereits genutzten 50 kHz hinaus wird daher nicht erwartet.

Das Konzept ist damit nachweislich in der Lage, Pfropfen mit einer Länge von 20 mm erfolgreich zu detektieren. Die Analyse des Konzeptes begründet jedoch, dass mit der aktuellen Implementierung auch Pfropfenlängen bis hinunter zu 7 mm detektierbar sein müssten. Diese Länge wird für sinnvolle Anwendungen jedoch noch als zu groß erachtet, da lange Pfropfen nachweislich einen geringeren Stofftransport aufwiesen als kurze (siehe Unterabschnitt 2.2.3, Abschnitt 4.4). Die Detektierbarkeit ist weiter zu verbessern, wofür die Problemgebiete hinreichend spezifiziert wurden. Die Förderwirkung wurde daher noch nicht untersucht, da diese bei Variation der Spulengeometrie sowieso erneut zu evaluieren ist. Wie weit die Pfropfenlänge der Strömung zu Gunsten der Detektierbarkeit sinnvoll gesteigert werden sollte, ohne dadurch den Stofftransport in der Strömung zu stark abzusenken, stellt ein Optimierungsproblem dar, welches nur fallspezifisch gelöst werden kann.

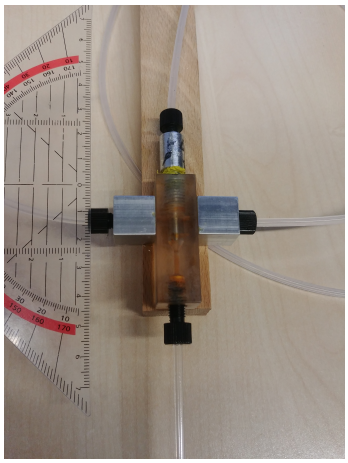
4.4 Stofftransport in hochviskosen Pfropfenströmungen

Als potentielles Anwendungsgebiet für die Pfropfenströmung wurde der Einsatz zur Extraktion aus hochviskosen Medien heraus identifiziert. Während in Extraktionskolonnen bei viskosen kontinuierlichen Phasen der Gegenstrombetrieb behindert wird und bei viskosen dispersen Phasen die interne Zirkulation der Tropfen zum Erliegen kommt, werden bei der Pfropfenströmung stets beide Phasen von ausgeprägten Zirkulationswirbeln durchzogen. [21] [44] Zur Beurteilung der Eignung für hochviskose Stoffsysteme wird daher in diesem Abschnitt der Einfluss der Viskosität auf den Stofftransport untersucht.

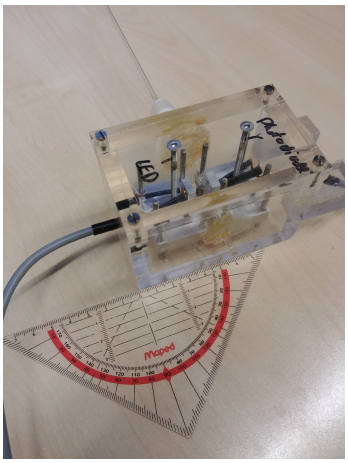
Für die Experimente wird als Demonstrationsbeispiel ein Stoffsystem aus Wasser/Aceton/Silikonöl gewählt. Das ausgewählte Silikonöl Polydimethylsiloxan hat eine sehr ähnliche Dichte wie Wasser und ist durch die variablen Länge des Polymerrückgrates in unterschiedlichen Viskositäten bei sonst gleich bleibenden Stoffeigenschaften erhältlich. Die Messung der Acetonkonzentration erfolgt online durch ein hierfür konstruiertes UV-Spektroskop. Verwendet werden die Silikonöle M20, M50 und M100, welche mit ihrem Zahlenwert direkt die Nennviskosität η laut Hersteller in $mPa \cdot s$ angeben. Der verwendete Pfropfenerzeuger wurde nach dem Vorbild von Arsenjuk et al. ([24]) in den mechanischen Werkstätten gefertigt und ermöglicht eine Einstellung der Pfropfenlänge, ohne dabei andere Prozessparameter verändern zu müssen. Die Verbindungsstücke zwischen Pfropfenerzeuger und Spektroskop bestehen aus Quarzglaskapillaren, um die notwendige Durchlässigkeit für das verwendete UV-Licht zu gewährleisten. Die Länge der Verbindungsstücke kann variiert werden, um auf diese Weise Extraktionsstrecken im Bereich von 10,5 cm bis 50,0 cm zu erzeugen und so die Extraktionsdauer zu beeinflussen. Da Quarzglas selber sowohl für Silikonöl als auch für Wasser ein gute Benetzbarkeit aufweist, wurden die Quarzglaskapillaren von innen mit Hilfe von Octadecyltrichlorsilan hydrophobisiert ([112], [49]). Erst durch diese Oberflächenbehandlung konnte eine stabile Pfropfenströmung erzeugt werden. Abbildung 4.19 zeigt den experimentellen Aufbau und die verwendeten Elemente. Die Detektion von Aceton erfolgt durch eine UV-LED und eine passende UV-Photodiode der Firma Roithner bei 280 nm. Das verwendete Silikonöl weist in diesem Wellenlängenbereich keine nennenswerte Absorption auf. Zum Schutz vor der emittierten Strahlung und zur Abschirmung der Photodiode vor UV-



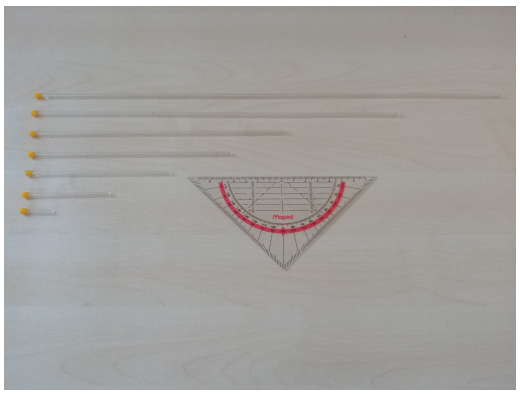
(a) Gesamtansicht des experimentellen Aufbaus.



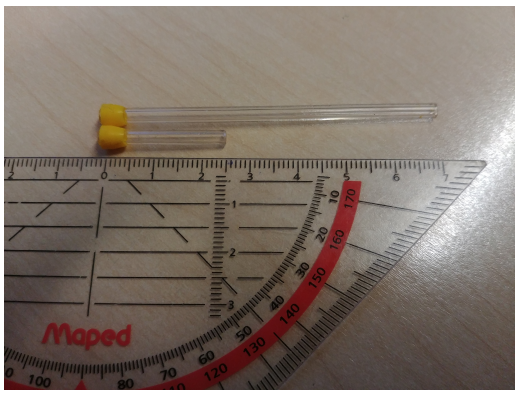
(b) Verwendeter Pfropfenerzeuger.



(c) Ausgebautes Spektroskop.



(d) Abstandsstücke zur Variation der Extraktionsdauer.



(e) Kürzeste Abstandsstücke in Detailansicht.

Abbildung 4.19: Experimentelles Set-Up zur Vermessung des Stofftransportes.

Licht der Umgebung wurde das Gehäuse des Spektroskops aus 1 cm dickem PMMA gefertigt, welches für photographische Aufnahmen mit sichtbarem Licht weiterhin durchlässig ist. Um eine reproduzierbare Online-Messung der Acetonkonzentration gewährleisten zu können, mussten im Vorfeld besondere Maßnahmen getroffen werden, die im Folgenden erläutert werden sollen. Da der Strahlengang durch die gekrümmte Glaskapillare hindurch nicht geradlinig verläuft, ist auf eine symmetrische Anordnung von LED und Photodiode innerhalb des Spektroskops zu achten (siehe Abbildung 4.20 links). Hierfür sind UV-LED und Photodiode auf Metallhalterungen montiert worden, welche durch Stellschrauben in ihrer Höhe verstellbar sind. Vor der erstmaligen Kalibrierung sind die Halterungen so eingestellt worden, dass eine händische Auslenkung der leicht biegsamen Glaskapillare innerhalb des Spektroskops in z-Richtung eine von der Auslenkungsrichtung unabhängige identische Veränderungen der gemessenen Signalstärke an der Photodiode hervorruft. Eine identische Signalveränderungen bei unabhängiger Auslenkungsrichtung kann nur dann entstehen, falls die nicht ausgelenkte Glaskapillare symmetrisch im Strahlengang liegt. Darüber hinaus strahlt die LED das UV-Licht in einem gewissen Öffnungswinkel aus. Hierdurch wird nicht nur die zu vermessende kontinuierliche Phase (Silikonöl) durchstrahlt, sondern es entstehen ebenfalls Lichtreflexionen an den Kappen der Pfropfen, welche das Messsignal zusätzlich stören (siehe Abbildung

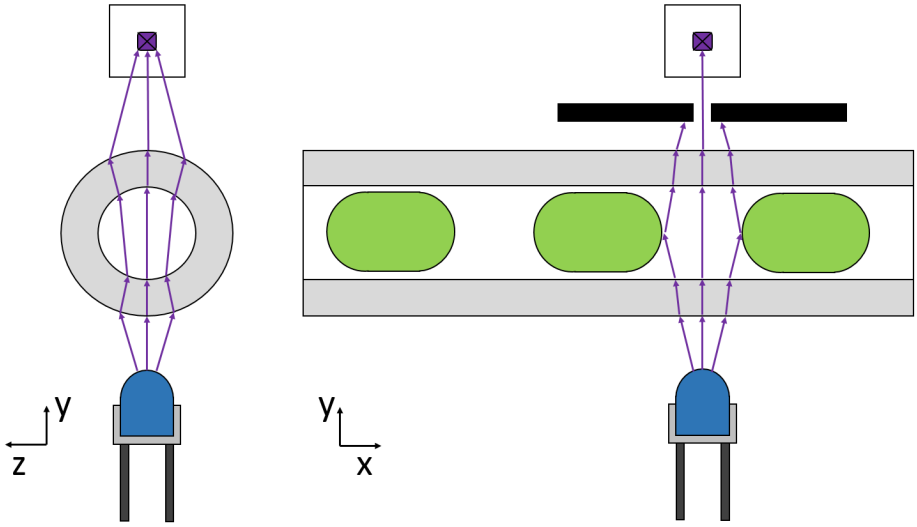


Abbildung 4.20: Schematische Darstellung des Strahlengangs für die Durchleuchtung einer gekrümmten Glaskapillare mit UV-Licht.

4.20 rechts). Um diese Reflexionen weitgehend auszublenden wurde eine Blende vor den Eingang der Photodiode montiert, die eine Spaltbreite von ca. 1 mm besitzt. Für eine bessere Unterscheidung zwischen kontinuierlicher und disperser Phase im Messsignal wurde der wässrigen Pfropfenphase Natrium-Fluoreszein zugesetzt, welches das UV-Licht nahezu vollständig absorbiert. Durch diese Maßnahmen entstehen im aufgezeichneten Messsignal klar definierte Plateauwerte. Der untere Plateauwert kann durch die Absorption des UV-Lichts im Wasser mit dem Durchtritt eines Pfropfens identifiziert werden, während der obere Plateauwert eindeutig dem Durchtritt der kontinuierlichen Phase zugeordnet werden kann. Das Messsignal wird mit LabVIEW aufgezeichnet und durch ein hierfür entwickeltes MATLAB-Skript ausgewertet. Abbildung 4.21 visualisiert ein typisches Messsignal mit Identifikation der charakteristischen Werte und den vorgenommenen Auswertungsschritten. Trotz der installierten Blende konnten nicht alle Lichtreflexionen unterdrückt werden. Diese rufen im Messsignal ein Überschießen der gemessenen Spannung über den oberen Plateauwert hinaus hervor. Nach automatisierter Identifikation der steilen Anstiegsflanke (cyan) wird das obere Plateau daher um einen für jede Messung individuell bestimmbaren Anteil links und rechts verkürzt (typischerweise 1/3), um sicherzustellen, dass keine Verfälschung vorliegt. Die verbliebenen Datenpunkte (rot) werden zu einem repräsentativem Plateauwert (grün) mit zugehöriger Standardabweichung σ gemittelt. Das Signal zeigt neben dem periodischem Wechsel der Plateauwerte auch eine Oberschwingung mit einer Frequenz im Bereich von ca. 50 Hz. Auch die Rücksprache mit den elektrischen Werkstätten und der Umstieg auf eine Gleichspannungsquelle zur Energieversorgung konnten diese Oberschwingung nicht vollständig beseitigen, was die Qualität des Messsignals bei besonders kurzen Pfropfenlängen und gleichzeitig hohen Durchtrittsgeschwindigkeiten beeinträchtigt. Die Auswirkung der Oberschwingung auf die bestimmten Werte für $k_l \cdot a$ wird daher durch Gauss'sche Fehlerfortpflanzung bestimmt. Als Fehler des jeweiligen Voltsignals wird die einfache Standardabweichung verwendet. Für die Umrechnung der gemessenen Spannungssignale in Konzentrationen wird das UV-Spektroskop für jede Sorte an verwendetem Silikonöl mit sechs Lösungen bekannter Konzentration im Bereich von 0 mol/L

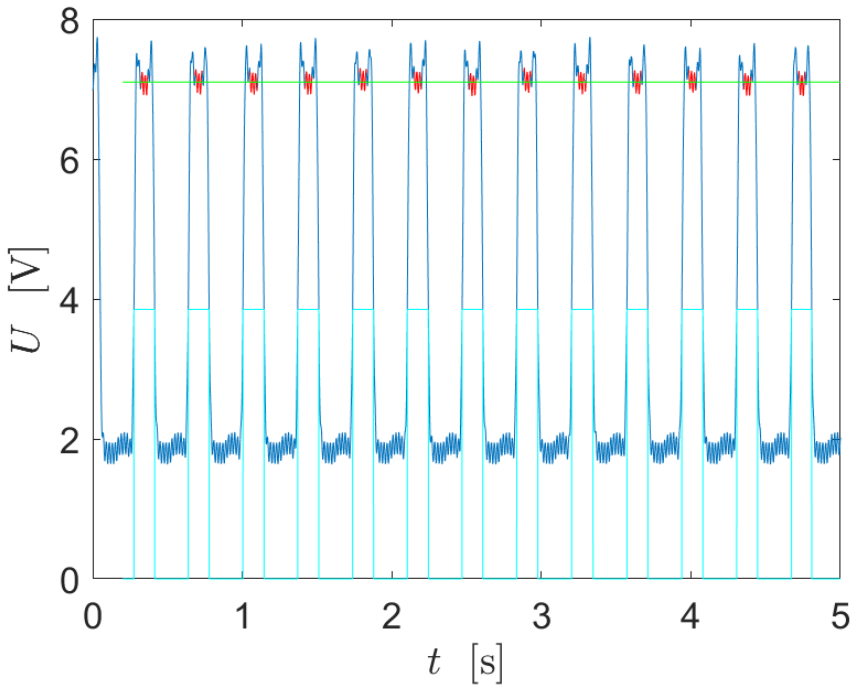


Abbildung 4.21: Beispielhafte Auswertung der aufgezeichneten Signale durch MATLAB: Rohdaten (blau), identifizierte kontinuierliche Phase (cyan), auszuwertende Datenpunkte (rot), reduzierter oberer Plateauwert (grün).

- 0,2 mol/L kalibriert. Die zugehörigen Kalibrierkurven sind in Anhang H beigefügt. Neben den Spannungssignalen der Photodiode sind die Pfropfen innerhalb des Spektroskops photographiert worden. Durch den bekannte Außendurchmesser der Kapillare von 3,2 mm ist ein Referenzmaßstab verfügbar, über den die Pfropfenlänge zugänglich ist. Abbildung 4.22 veranschaulicht die aufgenommenen Bilder und die nachfolgenden Auswertungsschritte durch MATLAB. Das Bild wird zunächst gedreht, sodass die Kapillare senkrecht im Bild steht. Daraufhin folgt eine Entzerrung, sodass die Kapillare am oberen und unteren Bildrand die gleiche Dicke aufweist. Ein ausgewählter Ausschnitt des Bildes wird daraufhin aufgehellt und in ein ausgefülltes binäres Bild umgewandelt, aus dem die Pfropfenlänge bestimmt werden kann.

Insgesamt wurden 19 Versuchsreihen durchgeführt. Die ersten 9 Versuchsreihen variieren bei konstantem Gesamtvolumenstrom von 2 ml/min die Pfropfenlänge (3 mm, 4 mm, 5 mm) in Kombination mit dem verwendetem Silikonöl (M20, M50, M100). Weitere 10 Versuchsreihen variieren für M100 bei einer Pfropfenlänge von 3 mm und 5 mm den Gesamtvolumenstrom im Bereich von 1 ml/min - 5 ml/min. Das Phasenverhältnis betrug stets 1 : 1. Innerhalb einer Versuchsreihe wird für jede der 7 möglichen Längen an Extraktionsstrecken (10,5 cm - 50,0 cm) die Konzentration der kontinuierlichen Phase bestimmt und über Gl. 4.16 in einen Extraktionsgrad E umgerechnet.

$$E = \frac{c_{org,0} - c_{org}}{c_{org,0} - c_{org,GGW}} \tag{4.16}$$

Die hierfür benötigten Gleichgewichtskonzentrationen sind in Tabelle 4.5 aufgeführt. Die Bestimmung erfolgte durch Schüttelversuche und Vermessung der jeweiligen Konzentration im UV-Spektroskop. Eine Anpassung von Gl. 4.17 an die bestimm-

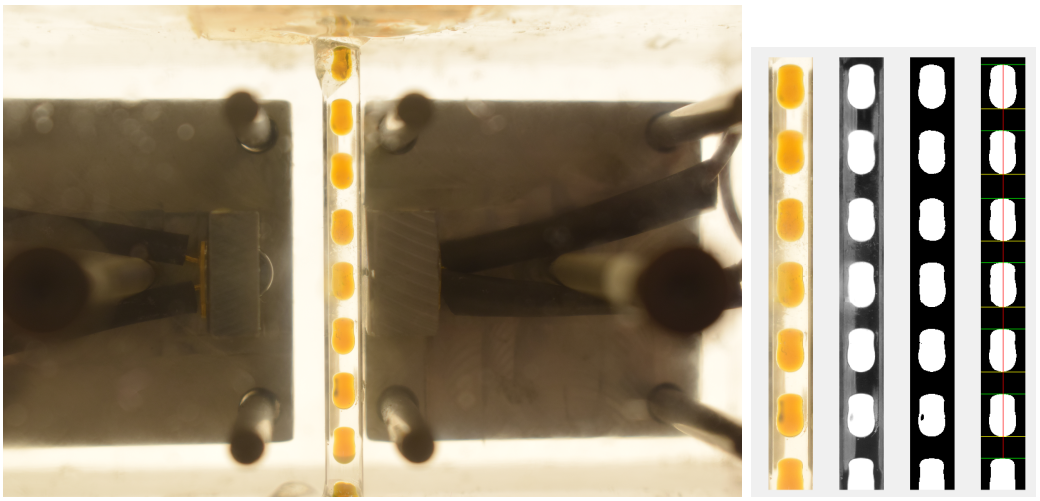


Abbildung 4.22: Auswertung der Bildaufnahmen zur Bestimmung der Pfropfenlänge.

Tabelle 4.5: Vermessene Gleichgewichtskonzentrationen von Aceton für verschiedenen Silikonöle und Wasser.

	M20	M50	M100
$c_{aq,0}[mol/L]$	0	0	0
$c_{org,0}[mol/L]$	0,2	0,2	0,2
$c_{aq,GGW}[mol/L]$	0,170 ± 0,003	0,164 ± 0,003	0,177 ± 0,004
$c_{org,GGW}[mol/L]$	0,030 ± 0,003	0,036 ± 0,003	0,023 ± 0,004

ten Extraktionsgrade E_i durch Minimierung von gewichteten Fehlerquadraten ermöglicht eine akkurate Bestimmung des Stofftransportkoeffizienten $k_l \cdot a$. Die Zeit t errechnet sich aus dem vorliegendem Volumenstrom und der Geometrie der Extraktionsstrecke.

$$E = 1 - \exp\left(-k_l \cdot a \cdot t\right) \tag{4.17}$$

Damit nach diesem Ansatz verfahren werden kann, müssen alle Extraktionsgrade E_i einer Versuchsreihe zu annähernd identischen Bedingungen entstanden sein. Besonders bezüglich der spezifischen Austauschfläche a sollten keine Schwankungen vorliegen. Als Maß hierfür wird die Pfropfenlänge L betrachtet. Die spezifische Austauschfläche a ist besonders bei kurzen Pfropfen zwar nicht direkt proportional zur Pfropfenlänge, doch kann bei sonst gleichen Versuchsbedingungen aus identischen Pfropfenlängen L auch auf die Gleichheit der spezifischen Austauschflächen a geschlossen werden. Der verwendete Pfropfenerzeuger besitzt bereits eine sehr gute Reproduzierbarkeit, doch weisen trotzdem nicht alle erzeugten Pfropfenströmungen die gleiche Pfropfenlänge L auf. Es werden daher Datenpunkte verworfen, bis die verbleibenden Datenpunkte einer Versuchsreihe bezüglich ihrer Pfropfenlänge innerhalb eines ±10%-Intervalls um ihren Mittelwert herum liegen. Hierdurch soll sichergestellt werden, dass auch die spezifischen Austauschflächen a aller zur Auswertung weiterhin verwendeten Datenpunkte möglichst ähnlich sind. Unter der Pfropfenlänge wird dabei der Abstand zwischen dem vordersten Punkt der Pfropfenkappe und dem hintersten Punkt des Pfropfenhecks verstanden. Neben den photographisch bestimmten Pfropfenlängen werden zudem die Längen der Pfropfen aus dem Voltsi-

gnalen mit einbezogen (cyanfarbene Plateaus; siehe Abbildung 4.21). Die hierdurch bestimmten Werte für $k_l \cdot a$ sind in Abbildung 4.23 und 4.24 dargestellt. Die 19

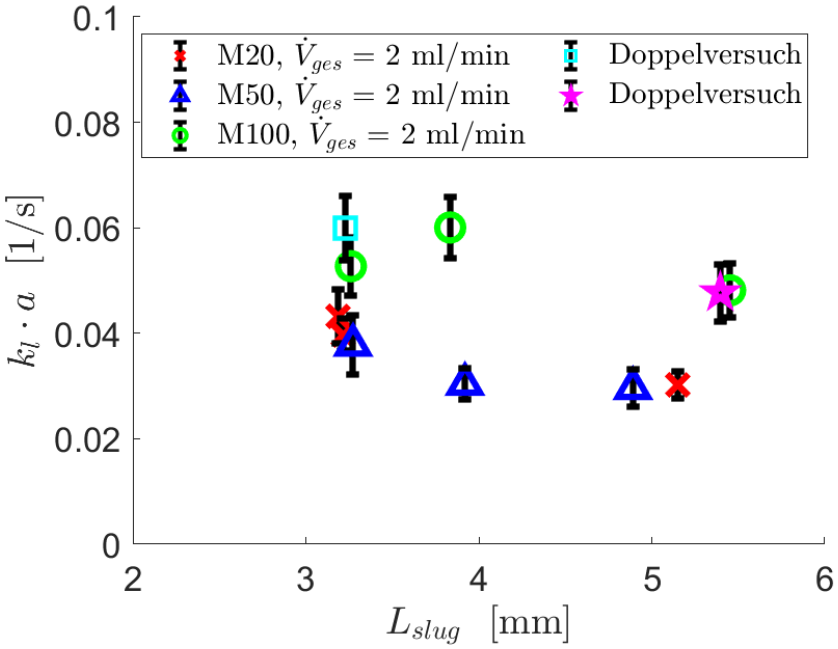


Abbildung 4.23: Stofftransportkoeffizienten $k_l \cdot a$ bei Verwendung von verschiedenen Silikonölen und Pfropfenlängen mit einem Gesamtvolumenstrom von $\dot{V}_{ges} = 2\,\mathrm{ml/min}$. Überschneidungen mit den Versuchsreichen aus Abbildung 4.24 sind zur Beurteilung der Reproduzierbarkeit im oberen Diagramm mit enthalten.

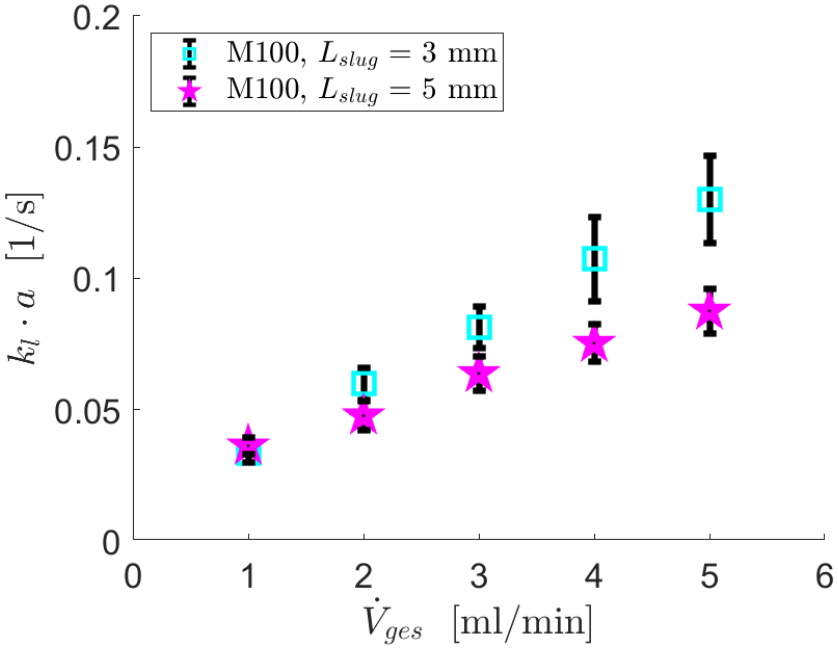


Abbildung 4.24: Stofftransportkoeffizienten $k_l \cdot a$ bei Verwendung des Silikonöls M100 bei verschiedenen Pfropfenlängen und Gesamtvolumenströmen.

durchgeführten Versuchsreihen weisen bereits 2 Überschneidungen auf. So ist für eine Pfropfenlänge von 3 mm und 5 mm bei einem Volumenstrom von jeweils 2 ml/min der Stofftransport doppelt bestimmt worden. Diese Datenpunkte sind ebenfalls in Abbildung 4.23 eingetragen und deuten auf eine gute Reproduzierbarkeit hin. Ebenso scheint bei der Vermessung des Öls M20 mit einer beabsichtigten Pfropfenlänge von 4 mm der Pfropfenerzeuger falsch eingestellt worden zu sein. Die aufgenommenen Bilddaten weisen erneut eine Pfropfenlänge von 3 mm auf, was den Stofftransport bei 4 mm Pfropfenlänge ungeklärt lässt, jedoch die Genauigkeit der Messung bei 3 mm Pfropfenlänge und damit ebenfalls die allgemeine Reproduzierbarkeit bestätigt. Trotz der verhältnismäßig großen Fehlerindikatoren lässt sich erkennen, dass die bestimmten Stofftransportkoeffizienten zu größeren Pfropfenlängen hin abfallen. Die einzige Ausnahme hierzu bildet der Datenpunkt bei 4 mm Pfropfenlänge und Verwendung von M100. Dieser Trend deckt sich mit den in der Literatur dokumentierten Stofftransportmessungen für niedrigviskose Systeme (siehe Unterabschnitt 2.2.3). Darüber hinaus zeigt sich kein signifikanter Rückgang des Strofftransportes bei zunehmender Viskosität. Den gängigen Modellierungen zur Folge ist der Stofftransport mit dem Diffusionskoeffizienten D verknüpft. Nach dem Filmmodell sinkt k_l linear mit D, während nach der Penetrationstheorie zumindest eine Abhängigkeit mit $\sqrt{D}$ gegeben ist. Der Diffusionskoeffizient in den unterschiedlichen Silikonölen ist nicht vermessen worden, jedoch kann über die Stokes-Einstein-Gleichung (Gl. 4.18, [123]) näherungsweise abgeleitet werden, dass der Diffusionskoeffizient D umgekehrt proportional zur Viskosität η ist.

$$D = \frac{k_b \cdot T}{6 \cdot \Pi \cdot \eta \cdot R_0} \tag{4.18}$$

Der erwartete Rückgang des Stofftransports wird jedoch nicht beobachtet. Eine mögliche Interpretation der Beobachtung ist eine zunehmende Beteiligung des Wandfilms am Stofftransport. Durch die erhöhte Viskosität steigt die Wandfilmdicke h der Pfropfenströmung. Je höher die Viskosität der kontinuierlichen Phase ist, desto mehr wird das Strömungsprofil im Wandfilm abflachen und der Pfropfen zunehmend über einen nahezu stagnierenden Wandfilm abrollen. Die im Wandfilm gespeicherte Stoffmenge an Aceton wird während des Abrollens diffusiv in den Pfropfen hineinextrahiert und nimmt kontinuierlich ab. Je größer die Wandfilmdicke h ist, desto größer ist die Speicherkapazität des Wandfilmbereichs für Aceton. Gleichzeitig steigt mit der Wandfilmdicke h jedoch auch die Strecke an, die ein Acetonmolekül zurücklegen muss, um von der Kapillarwand in den Wasserpfropfen zu gelangen. Trotz dieser gegenläufigen Effekte ist eine erhöhte Wandfilmdicke zuträglich für den Gesamtstofftransport. So wiesen Kashid et al. experimentell durch Zugabe von Tensiden eine Zunahme des Stofftransportes bei sonst konstanten Stoffwerten nach. [50] Die Tenside senkten die Grenzflächenspannung und riefen durch die gesenkte Kapillar-Zahl Ca einen dickeren Wandfilm hervor. Weiterhin zeigte Heckmann simulativ, dass eine Zunahme der Kapillar-Zahl Ca, wie sie z.B. durch einen Anstieg der Viskosität η hervorgerufen wird, den Stofftransport begünstigt. [44] Diese Beobachtungen decken sich mit den in dieser Arbeit durchgeführten Experimenten. Demnach findet eine teilweise Verarmung des Wandfilmbereichs statt. Die Einbringung von neuem Aceton in den Wandfilmbereich erfolgt erst durch Austausch mit der kontinuierlichen Phase, sobald der Pfropfen komplett vorübergezogen ist. Kaske visualisierte diesen Mischvorgang durch Fluoreszenzmessungen. [23] Diese Argumentation erklärt

zudem die Abnahme des Stofftransportes hin zu größeren Pfropfenlängen. Je ausgedehnter der Pfropfen ist, desto länger wird dem Wandfilm Aceton entzogen bis eine erneute Verwirbelung mit der kontinuierlichen Phase stattfindet. Die treibende Konzentrationsdifferenz im Wandfilm fällt ab und der Gesamtstofftransport, welcher stets auf die makroskopischen mittleren Konzentrationen bezogen ist, sinkt dadurch.

Bezüglich der Strömungsgeschwindigkeit zeigt sich ein Anstieg des Stofftransportes hin zu höheren Strömungsgeschwindigkeiten, was sich ebenfalls mit den dokumentierten Literaturdaten für niedrigviskose Systeme deckt. Allgemein werden Werte für $k_l \cdot a$ erreicht, die in der gleichen Größenordnung liegen, wie Scheiff et al. sie bei ihren Experimenten mit ionischen Flüssigkeiten erzielten, jedoch arbeiteten Scheiff et al. mit kleineren Innendurchmessern und setzte die viskosere Flüssigkeit als disperse Phase ein. [19] Eine Variation des Innendurchmessers der Kapillare könnte weitere Aufschlüsse über das Verhalten des Systems geben, da die Speicherkapazität des Wandfilms mit dem Quadrat des Durchmessers skaliert, der nötige Diffusionsweg hingegen nur linear ansteigt.[3] Diese Variation wurde im Rahmen dieser Arbeit jedoch nicht durchgeführt.

Für die Bewertung der Ergebnisse sind neben der bereits erwähnten Oberschwingung im Messsignal weitere Fehlerquellen zu beachten, welche nicht durch die Fehlerindikatoren berücksichtigt werden konnten. Zu nennen ist hier zum einen die Mittelung des Konzentrationssignals über den gesamten Kanalquerschnitt. Während bei der Kalibrierung des UV-Spektroskops die Kapillare einphasig mit der Kalibrierlösung durchströmt worden ist, herrschte an jedem Punkt der Kapillare die gleiche Konzentration c. Die Fluoreszenzmessungen von Kaske ([23]) legen jedoch nahe, dass während des Extraktionsvorgangs ein Konzentrationsprofil in der kontinuierlichen Phase herrscht, welches durch dieses Messverfahren nicht erfasst werden kann. Zum anderen ist die Viskosität des Öls im vermessenen Konzentrationsbereich an Aceton nicht konstant. So nimmt beispielsweise die Viskosität des Silikonöls M100 durch Zugabe von 0,2 mol/L um etwa 15% ab. Im Vergleich zu den Viskositätsänderungen durch den Polymerisationsgrad der Öle (M20, M50, M100) ist diese Änderung hingegen klein. Eine Ermüdung der UV-LED über die Dauer der Versuchsreihen hinweg und eine Verarmung der Stammlösung durch Ausdampfen des Acetons konnten als Fehlerquelle hingegen ausgeschlossen werden, da nach Abschluss der Versuchsreihen die Stammlösung jedes Öls erneut vermessen wurde und eine gute Übereinstimmung mit der Kalibrierkurve vorlag (vgl. Anhang H). Trotz der erwähnten Fehlerquellen zeigen die Ergebnisse, dass die Pfropfenströmung in besonderem Maße für die Extraktion aus viskosen Medien heraus geeignet ist, sofern die viskose Phase als kontinuierliche Phase gewählt wird. Während sowohl eine hohe Grenzflächenspannung als auch eine erhöhte Viskosität die Effektivität herkömmlicher Extraktionskolonnen einschränken, bleibt durch den gegenläufigen Effekt beider Parameter im Bezug auf die Kapillar-Zahl Ca die Pfropfenströmung ein effektives Extraktionswerkzeug.

[3] Häufig wird die mittlere Eindringtiefe mit $\bar{x} = 2 \cdot \sqrt{D \cdot t}$ abgeschätzt.

4.5 Reduzierung des apparativen Aufwands durch die Kreuz-Gegenstrom-Verschaltung

Für eine erfolgreiche Anwendung der Pfropfenströmung im industriellen Maßstab sind Maßnahmen zur Steigerung der Produktionsmenge erforderlich. Eine einfache Vervielfältigung der Apparate ist wegen der vergleichsweise teuren Pumpen und Regelkomponenten nicht wirtschaftlich. Da eine Vergrößerung der Abmessungen die vorteilhaften Eigenschaften der Mikroapparate zerstört und die Durchflussmengen pro Kapillare begrenzt sind, werden Parallelisierungsstrategien eingesetzt, wie sie in Unterabschnitt 2.2.5 beschrieben wurden. Die bisherigen Methoden lassen dabei verschiedene Ansätze erkennen. Gemein ist jeweils, dass alle parallelisierten Kapillaren mit nur zwei Pumpen betrieben werden. Hierbei bedingen die Fertigungstoleranzen in den einzelnen Bauteilen unterschiedlich hohe Druckverluste pro Extraktionsstrang. Dies führt zu einer Ungleichverteilung der in den Kapillaren fließenden Volumenströme. Der Versuch, dies auszugleichen, kann durch sehr genaue Fertigungstoleranzen der Kapillaren und der Apparate selbst (siehe [51], [52]), individuelle Nachjustierung der Druckverluste pro Kapillare (siehe [56]) oder durch einen großen vorgeschalteten Druckverlust mit geringer Fertigungstoleranz geschehen, welcher durch seine Größe die nachfolgenden Schwankungen im Druckverlust marginal werden lässt (siehe [53], [54], [55]). In diesem Abschnitt soll eine neue Parallelisierungsstrategie vorgestellt werden, welche die erforderliche Pumpenzahl nicht auf zwei reduziert, dafür jedoch große Freiheiten bzgl. der Stabilität der Strömung und der geforderten Fertigunstoleranzen der Bauteile erzeugt. Es garantiert im Gegensatz zu bisherigen Ansätzen zudem exakt die Gleichverteilung der Volumenströme auf die einzelnen Stränge. In Unterabschnitt 4.5.1 wird das Konzept der neuen Verschaltungsstrategie erläutert. Unterabschnitt 4.5.2 beschreibt die experimentelle Umsetzung zur Plausibilitätsbeurteilung des Konzeptes, während Unterabschnitt 4.5.3 das Konzept bewertet, auf Anwendbarkeit prüft und weitere Untersuchungsaspekte fokussiert.

4.5.1 Konzept der Kreuz-Gegenstrom-Verschaltung

In einer Pfropfenströmung kann eine Extraktion nur im Gleichstrombetrieb durchgeführt werden. Durch die ausgezeichneten Stofftransporteigenschaften wird innerhalb einer Extraktionskapillare allerdings schnell das thermodynamische Gleichgewicht erreicht. Dieses Verhalten ähnelt den Mixer-Settler-Anlagen (siehe Unterabschnitt 2.1.1), weshalb die Kombination aus Pfropfenerzeuger, Extraktionskapillare und Phasentrenner oft als eine Mikro-Mixer-Settler-Einheit betrachtet wird. Die austretenden Ströme einer Einheit können jedoch mit anderen Extraktionseinheiten im Gegenstrom verschaltet werden. Da innerhalb jeder Einheit Druckverluste auftreten, kann nur einer der beiden Ströme ohne Einsatz einer weiteren Pumpe in die nachfolgende Einheit eingespeist werden. Der andere Strom muss zur Überwindung des sich einstellenden Druckgefälles entlang eines Strangs mit Hilfe von Pumpen jeweils den davorliegenden Stufe zugeführt werden. Abbildung 4.25 verdeutlicht den Sachverhalt. Insgesamt werden für den Betrieb von n Stufen in einem Strang demnach $n + 1$ Pumpen benötigt. Daraus ergibt sich für den parallelen Betrieb von m Strängen ein Bedarf von $m \cdot (n + 1)$ Pumpen. Da die Extraktionsstränge lediglich Kopien voneinander sind, werden die jeweiligen Einheiten der Stränge sich untereinander im Bezug auf Konzentration und Geschwindigkeit nicht unterscheiden. Für

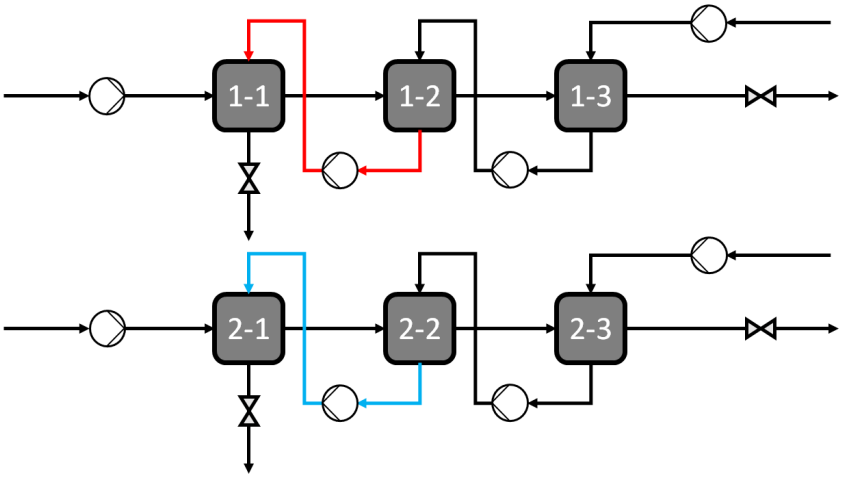


Abbildung 4.25: Parallelisierung von zwei Extraktionssträngen zu jeweils drei Mixer-Settler-Einheiten nach normalem Numbering-Up-Konzept.

eine erfolgreiche Gegenstromfahrweise ist es daher nicht zwingend notwendig, dass der in Abbildung 4.25 rot markierte Strom aus Einheit 1-2 zwingend in die Einheit 1-1 eingespeist wird. Er kann auch im zweiten Strang in die erste Einheit 2-1 geführt werden. Im Gegenzug wird der blaue Strom aus Einheit 2-2 nicht der Einheit 2-1 sondern dem oberen Strang in Einheit 1-1 zugeführt. Diese Kreuzvermischung der Extraktionsstränge ist dann vorteilhaft, wenn jeder Extraktionsstrang auf einem anderem Druckniveau betrieben wird. Für flüssig-flüssig-Systeme entsteht dieser Freiheitsgrad, da ihr Verhalten unabhängig vom Absolutdruck ist. Der rote Strom kann dadurch ein Druckniveau besitzen, das ausreichend ist, um ohne zusätzliche Pumpe in die Einheit 2-1 eingespeist zu werden. Im Gegenzug dafür muss von der Pumpe des blau markierten Strangs eine höhere Druckdifferenz überwunden werden als bisher. Effektiv kann durch diese Verschaltung jedoch eine Pumpe eingespart werden. Diese Herangehensweise kann auch auf die anderen Ströme angewendet werden, wodurch eine Verschaltung nach Abbildung 4.26 entsteht. Die bisher beispielhaft genannten Ströme sind zur leichteren Identifikation erneut in rot und blau eingefärbt. Allgemein werden für diese Betriebsweise nicht mehr $m \cdot (n + 1)$ Pumpen benö-

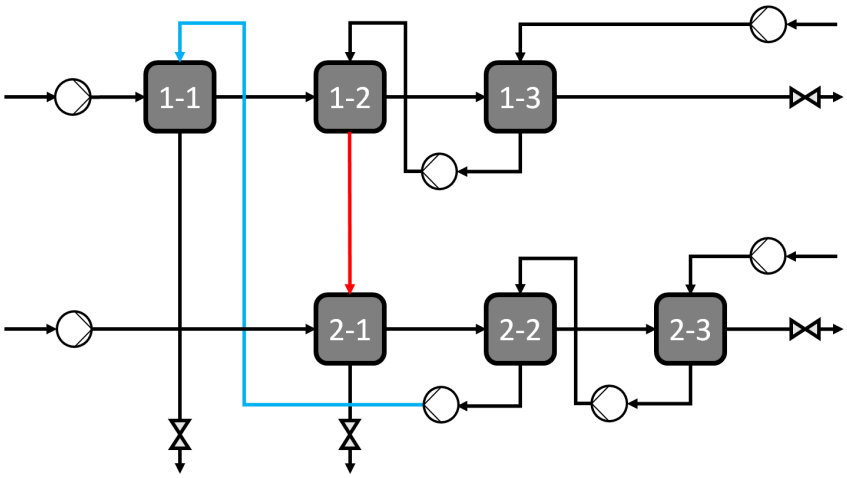


Abbildung 4.26: Beispielhafte Verschaltung zweier Extraktionsstränge durch Überkreuzen der Ströme zur Einsparung einer notwendigen Pumpe.

tigt, sondern lediglich $2 \cdot m + n - 1$, was für die Parallelisierung vieler Stränge eine deutliche Einsparung bzgl. der benötigten Pumpen bedeutet. Voraussetzung für die Anwendung dieses Verschaltungskonzeptes ist eine stabile und saubere Phasentrennung innerhalb jeder einzelnen Einheit. Fertigungstoleranzen der Bauteile erzeugen weiterhin ungleiche Druckverluste in den Zuleitungen und den Einheiten der Stränge selbst. Diese Unregelmäßigkeiten müssen durch einen ausreichend hohen Kapillardurck der Phasentrennung aufgefangen werden. Den Ergebnissen aus Abschnitt 4.1 zur Folge bieten lediglich Membranphasentrenner durch ihren außerordentlich hohen Kapillardruck die geforderte sehr stabile Phasentrennung.

4.5.2 Experimentelle Umsetzung des Konzeptes

Zur Demonstration der grundsätzlichen Realisierbarkeit des Konzeptes wurde für die experimentelle Umsetzung beispielhaft das Stoffsystem n-Decan/Wasser verwendet. Es bietet eine hohe Grenzflächenspannung von 52,0 mN/m, was ein typisches Anwendungsfeld der Pfropfenströmung darstellt, und weist bei 25 °C zudem ähnlichen Viskositäten beider Fluide von 0,838 mPa · s für Wasser und 0,890 mPa · s für n-Decan auf. [109] [110] Dies vereinfachte den experimentellen Aufbau, da alle Zuleitungen gleich ausgeführt werden konnten. In den hier dargestellten Experimenten wurden FEP-Kapillaren mit einem Innendurchmesser von 1,6 mm verwendet. Für Stoffsysteme unterschiedlicher Viskosität sollten die Innendurchmesser und Längen der Verbindungsleitungen so gewählt werden, dass alle Leitungen untereinander einen ähnlichen Druckverlust aufweisen. Dies reduziert die notwendige Druckdifferenz, welche die Phasentrenner durch ihren Kapillardruck ausgleichen müssen. Eine Extraktion wurde nicht durchgeführt, da zunächst nur das hydrodynamische Konzept der Gesamtverschaltung validiert werden sollte. Hierfür waren in einem ersten Schritt Phasentrenner zu konstruieren, die eine außerordentlich stabile Phasentrennung garantieren können. Hier wurde auf Membranphasentrenner zurückgegriffen, da diese durch die engen Poren einen besonders hohen Kapillardruck liefern können. Es werden Gehäuse aus PMMA gefertigt, in denen sowohl am Kopf eine hydrophobe als auch am Boden eine hydrophile Membran eingesetzt werden können. Der allgemein Aufbau ist in Abbildung 4.27 dargestellt. Als hydrophobe Membranen

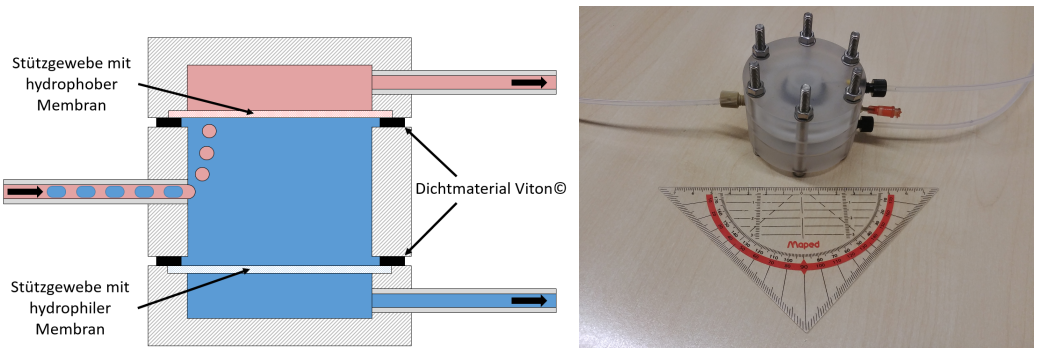


Abbildung 4.27: Schnittansicht des Phasentrenners (links) und gefertigtes Exemplar aus PMMA (rechts) mit allen Zu- und Ableitungen und Größenvergleich.

wurden PTFE und PC getestet, wobei PTFE den kleineren Kontaktwinkel mit n-Decan aufwies. Als hydrophile Membran wurden Nitrozellulose und modifiziertes Polyester untersucht. Im Gegensatz zum Polyester zeigte die Nitrozellulose zumin-

dest brauchbare Kontaktwinkeleigenschaften im Kontakt mit reinem Wasser. Im direkten Einsatz konnte sie die Permeation von n-Decan jedoch nicht verhindern. Der Betrieb der Phasentrenner musste daher im vollständig wassergefüllten Zustand erfolgen. Die Membranen wurden auf einem Stützgewebe platziert und mit Dichtungen aus Viton © im Gehäuse arretiert. Die Permeation von Wasser und Luft durch das PTFE wurde nach der ersten Benetzung mit n-Decan durch den Kapillardruck verhindert, welcher mit Hilfe des beobachteten Kontaktwinkels und des angegebenen Porendurchmessers von 5 µm auf 0,3 bar geschätzt wird. Für die Inbetriebnahme der Verschaltung war daher der Einbau seitlicher Entlüftungsstutzen notwendig. Die Funktionsweise eines einzelnen Phasentrenners wurde durch Zufuhr einer Pfropfenströmung mit einem Gesamtvolumenstrom von 4 ml/min bei einem Phasenverhältnis von 1:1 überprüft. Zur besseren Visualisierung erhielten die Organik durch Sudanrot und das Wasser durch blaue Tinte eine entsprechende Färbung. Als Pfropfenerzeuger wurden einfache T-Stücke aus PTFE verwendet, welche jeweils in eine wenige cm lange Zweiphasenstrecke mündeten. Die einwandfreie Trennung der eingehenden Strömung validiert die Funktionsweise eines einzelnen Phasentrenners. Daraufhin wurde zunächst eine Verschaltung aus 2 Strängen mit jeweils 2 Stufen nach Abbildung 4.28 erprobt. Besonderes Augenmerk erhielt dabei der Phasentrenner in Stufe

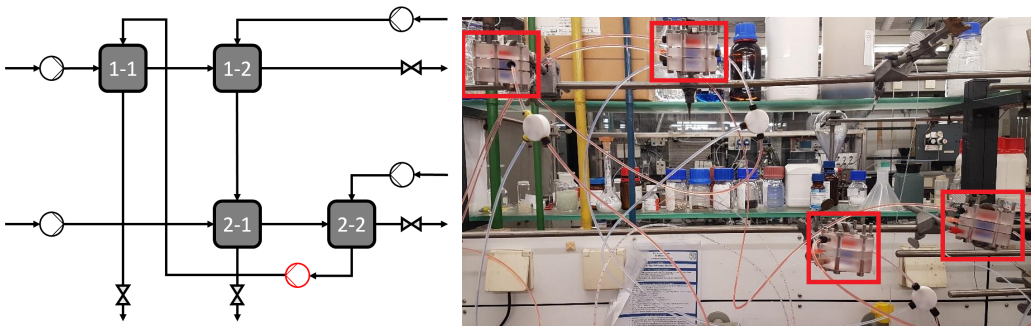


Abbildung 4.28: Verschaltungsplan (links) und experimenteller Aufbau einer 2x2 Kreuz-Gegenstrom-Verschaltung mit hervorgehobenen Phasentrennern (rechts).

2-1, dessen Pfropfenströmung nicht durch zwei Pumpen, sondern durch eine Pumpe und den austretenden Strom des höher liegenden Strangs erzeugt wurde. Soweit möglich wurden Spritzenpumpen verwendet, da kontinuierlich arbeitende Pumpen im Volumenstrom schwanken, falls sich der Gegendruck durch Nachjustierung der Nadelventile verändert. Eine Onlinemessung der Fördermenge ist nicht möglich gewesen, weshalb der Volumenstrom nicht nachgeregelt sondern lediglich durch ein Vorkalibrierung ungefähr eingestellt werden konnte. Diese Unregelmäßigkeit sollte zumindest in der grundsätzlichen Validierung des Konzeptes ausgeschlossen werden. Da während der Untersuchungen keine Extraktion stattfand, konnte die rot markierte kontinuierlich fördernden Pumpe in Abbildung 4.28 ebenfalls durch eine Spritzenpumpe ersetzt werden, sofern die jeweilige Zufuhr der Pumpe stattdessen über ein Nadelventil auf Umgebungsdruck entspannt wird. Nach iterativer Justierung der verwendeten Nadelventile zur Erbringung des notwendigen Gegendrucks im oberen Strang und zum Ausgleich ungleicher Druckverluste in den Kapillaren konnte die Gesamtschaltung für 20 Minuten bei einem Gesamtvolumenstrom von 4 ml/min mit 2 ml/min je Phase bzw. für 30 Minuten bei einem Gesamtvolumenstrom von 2 ml/min mit je 1 ml/min pro Phase betrieben werden. Bedingt durch das begrenzte Volumen der in den Pumpen eingesetzten Spritzen von ca. 60 ml

und die notwendige Einlaufzeit der Anlage konnte keine längeren Betriebszeit realisiert werden. Limitierend war demnach keineswegs ein Wasserdurchbruch an den PTFE-Membranen oder die Hydrodynamik der Pfropfenströmung. Aufbauend auf diesem Ergebnis wurde eine 3x3 Verschaltung nach Abbildung 4.29 realisiert. In

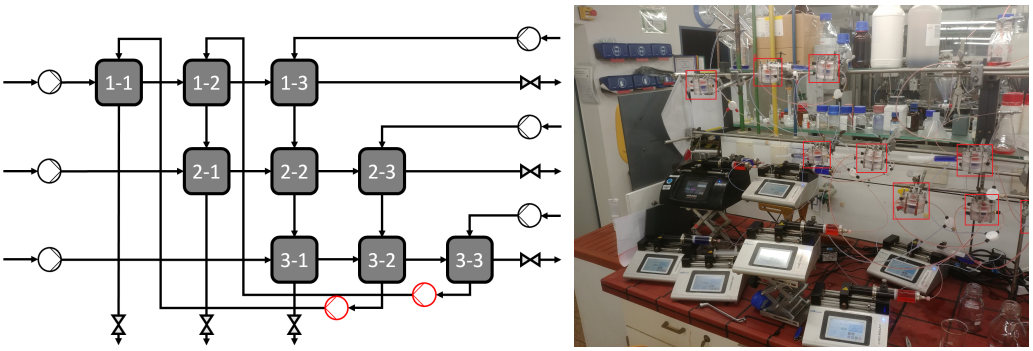


Abbildung 4.29: Verschaltungsplan (links) und experimenteller Aufbau einer 3x3 Kreuz-Gegenstrom-Verschaltung mit hervorgehobenen Phasentrennern (rechts).

dieser Gesamtverschaltung existierten somit bereits Mixer-Settler-Einheiten, welche durch keinerlei Pumpe beschickt wurden, sondern sich lediglich aus anderen Stufen mit jeweils höherem Druckniveau speisten. Erneut wurden die rot markierten Rückführungspumpen durch einen gedrosselten Auslass in die Umgebung und eine stabil fördernde Spritzenpumpe substituiert. Die Justierung der Nadelventile gestaltete sich für die 3x3-Verschaltung bereits außerordentlich komplex. Als Indikator dienten lediglich die langsam steigenden oder fallenden Wasserfüllstände in den Trennkammern der Mixer-Settler-Einheiten. Die Anlage konnte nur nach vielfachen Versuchen und Nachjustierungen in einen Betriebszustand ohne offensichtlichen Phasendurchbruch an den Membranen gebracht werden, in dem sie dann bei einem Gesamtvolumenstrom von 1 ml/min bei 0,5 ml/min je Phase betrieben wurde. Über einen Zeitraum von 20 Minuten wurden keinerlei weiteren Nachjustierungen vorgenommen und der Flüssigkeitsstand in den Phasentrennern der Anlage beobachtet. In vier von neun Trennkammern traten dabei merkliche Füllstandveränderungen oder Durchbrüche von Wasser auf. Im obersten Strang sank der Wasserspiegel in der Trennkammer der ersten Stufe stark ab, während in der letzten Stufe ein leichter Wasserdurchbruch zu verzeichnen war. Die mittlere Stufe arbeitete einwandfrei weiter. Abbildung 4.30 verdeutlicht diesen Sachverhalt. Auch im zweiten Strang wurden in zwei Stufen leichte Wasserdurchbrüche beobachtet. Im dritten Strang arbeiteten alle Trennkammern über die gesamte Betriebszeit hinweg ideal. Nach weiteren 5 Minuten waren die Spritzenvolumina erschöpft. Die beschriebenen Zustände sind soweit erhalten geblieben. Die korrekte Betriebsweise des kompletten unteren Stranges deutet dabei an, dass die Schwierigkeiten bei der korrekten Einstellung der Nadelventile vornehmlich in Stufen auftreten, welche in der Hydrodynamik in den oberen Strängen liegen und mit ihren Austritten an eine Vielzahl nachfolgender Stufen gekoppelt sind. Die 3x3 Verschaltung weist demnach noch Schwierigkeiten in der Umsetzung auf, demonstriert aber die grundlegende Plausibilität des Konzeptes. Die Experimente legen nahe, dass lediglich die Nadelventile korrekt eingestellt werden müssen, wofür gegenüber der händischen Nachiteration eine konkrete Vorgehensstrategie genutzt werden sollte.

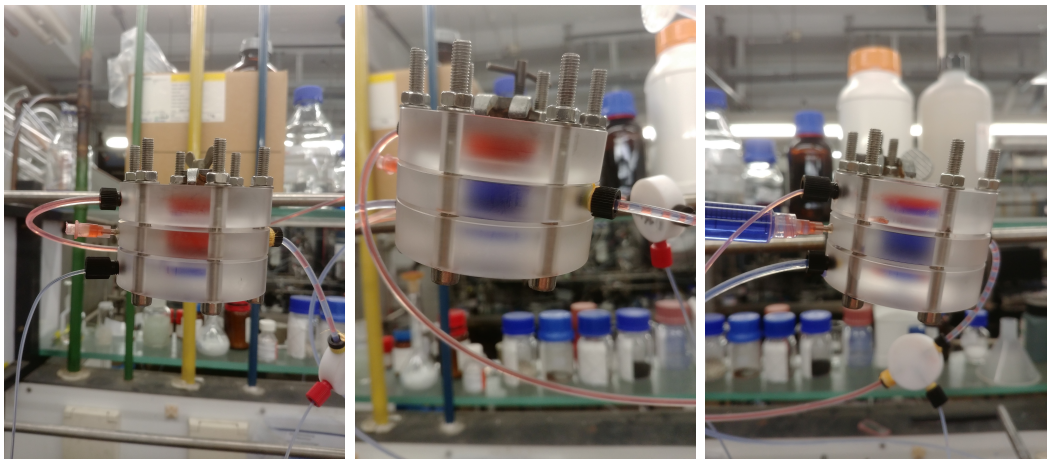


Abbildung 4.30: Trenneinheiten des obersten Strangs mit sinkendem Wasserspiegel in der ersten Stufe 1-1 (links), funktionierender Trennung in Stufe 1-2 (mitte) und leichtem Wasserdurchbruch an der PTFE-Membran in Stufe 1-3 (rechts).

4.5.3 Fehlerbetrachtung und weiterführende Analyse

Eine mögliche Ursache für das Versagen des Trennverhaltens einiger Stufen kann im nichtlinearen Zusammenhang zwischen Ventilstellung und erzeugtem Druckverlust der Nadelventile liegen. Beim Schließen ist der Anstieg des Druckverlustes nicht linear abhängig von der Änderung im Drehwinkel, sondern umso größer, je weiter das Ventil bereits geschlossen ist. Dies erschwerte stark das Vorgehen während der Feinjustierung der Anlage. Hier sollten in nachfolgenden Untersuchungen Kapillaren mit kleinen Innendurchmessern und definierter Länge verwendet werden, die zusätzlich durch die Höhenlage ihrer Austrittsöffnung eine gezieltere Einstellung des Druckverlustes erlauben. Auch der Einsatz thermorheologischer Ventile kann der Feinjustierung dienen. [25] [57] Ein erweitertes Screening zusätzlicher hydrophiler Membranmaterialien kann zudem am Boden der Trennkammern eine Durchtrittsbarriere für die Organik erzeugen, sodass ein sinkender Wasserspiegel nicht zwangsläufig zu einer versagenden Trennwirkung im Dauerbetrieb führen muss. Weiterhin kann der Porendurchmesser von 3 µm bzw. 5 µm weiter herabgesetzt werden, um den Kapillardruck zu steigern, welcher für das Aufrechterhalten der Hydrodynamik essentiell wichtig ist. Dabei sollte für alle Membranmaterialien auf eine enge Verteilung der Porengrößen geachtet werden, da die größte Pore der Membran den kleinsten Kapillardruck bereit stellt und somit die maximale Druckdifferenz definiert bevor der erste Durchbruch auftritt. Für eine Realisierung im Dauerbetrieb sind die Spritzenpumpen durch kontinuierliche Pumpen zu ersetzen, die pulsationsfrei und in der Fördermenge regel- und nicht nur steuerbar sind. Eine Differenzdruckmessung an den Auslässen der Trennkammern würde zudem das Anfahren der Anlage stark vereinfachen, konnte jedoch wegen des großen apparativen Aufwandes durch die notwendige neunfache Ausführung in dieser Arbeit nicht realisiert werden. In fortführenden Untersuchungen sollte über die 3x3-Verschaltung hinaus auch das Hinzufügen eines vierten Stranges untersucht werden.

Auf theoretischer Seite können durch das vorgestellte Konzept beliebig viele Stränge parallelisiert werden. Die Praxis zeigte jedoch bereits eine hohe Komplexität bei der Einstellung der Anlage und dem Ansteuern des möglichen Betriebsfensters für 3x3-Verschaltungen. Das mögliche Betriebsfenster der Anlage wird dabei

umso größer je höher die erzeugten Kapillardrücke in der Phasentrennung sind. Begrenzt wird das Betriebsfenster dadurch, dass sich unterschiedliche Druckverluste in den Leitungen, bedingt durch Fertigungstoleranzen, und fehlerhaft eingestellte Nadelventile aufsummieren aber weiterhin durch den Kapillardruck einer einzelnen Membran kompensiert werden müssen. Je mehr Stränge und Stufen verschaltet werden, desto höher sind demnach die Anforderungen an den aufzubringenden Kapillardruck und bzw. oder an die Genauigkeit der Voreinstellung der Anlage. Dabei muss auch berücksichtigt werden, dass während einer ablaufenden Extraktion durch den Übergang des Solvatstoffes die Stoffdaten, wie Dichte und Viskosität, aber auch die absoluten Volumenströme von organischer und wässriger Phase von Stufe zu Stufe variieren werden, was den Druckverlust der Stufe beeinflusst und ebenfalls durch den Kapillardruck an der Membran kompensiert werden muss. Eine genaue Aussage zur maximalen Anzahl der parallelisierbaren Stränge ist daher vom Stoffsystem abhängig und nicht allgemein möglich.

Allgemeiner Nachteil des Konzeptes ist die integrierte Abhängigkeit der Stufen untereinander. Fällt eine Einheit aus, so muss die gesamte Verschaltung und nicht nur ein einzelner Strang stillgelegt werden. Am Beispiel von $m = 10$ Strängen zu jeweils $n = 4$ Stufen kann auf der einen Seite eine Gesamtverschaltung mit einem Bedarf von $2 \cdot 10 + 4 - 1 = 23$ Pumpen errichtet werden, oder zur Erhöhung der Ausfallsicherheit und Steigerung der Robustheit in zwei Verschaltungen mit jeweils $m = 5$ Strängen und einem Gesamtbedarf von $2 \cdot (2 \cdot 5 + 4 - 1) = 26$ Pumpen investiert werden. Im Vergleich dazu bietet ein herkömmliches Numbering-Up die bestmögliche Ausfallsicherheit, besitzt allerdings einen Bedarf von $10 \cdot (4 + 1) = 50$ Pumpen. Obwohl etliche Pumpen in der Kreuz-Gegenstrom-Verschaltung jeweils eine größere Druckdifferenz überwinden müssen, werden absolut gesehen weniger Pumpen betrieben. Dies hat ebenfalls Auswirkungen auf die Betriebskosten. Da weniger Pumpen betrieben werden, ist wenn überhaupt nur ein moderater Anstieg der Betriebskosten zu erwarten.

Trotz der aufgetretenen Schwierigkeiten ist die entwickelte Kreuz-Gegenstrom-Verschaltung durch die Reduktion der notwendigen Pumpen von $m \cdot (n + 1)$ auf $2 \cdot m + n - 1$ ein aussichtsreicher Kandidat für die Parallelisierung von Extraktionsaufgaben in Mikro-Mixer-Settler-Anlagen. Das vorgestellte Verschaltungskonzept schafft somit eine zusätzliche Optimierungsmöglichkeiten zwischen den Parametern Fertigungstoleranz der Bauteile, Gesamtinvestitionskosten und Ausfallsicherheit der Anlage. Allgemein sollte vor der Nutzung dieses Verschaltungskonzeptes jedoch jede einzelne Mixer-Settler-Einheit auf ihre maximale Belastbarkeit untersucht werden, da sich trotz innovativer Parallelisierung der Betrieb weniger stark belasteter Extraktionseinheiten einfacher gestaltet als die Parallelisierung vieler Einheiten.

4.6 Resümee der durchgeführten Untersuchungen

Die zu Beginn dieses Kapitels angesprochenen möglichen Szenarien zur Etablierung der Pfropfenströmung als Extraktionswerkzeug wurden durch die Experimente und Simulationen ausgearbeitet und näher charakterisiert.

Bezüglich des ersten Szenarios wurde versucht, die bisherigen Nachteile im Vergleich zu herkömmlichen Extraktionskolonnen zu beseitigen. Hierfür wurden die schwere Parallelisierungbarkeit und die nicht stabile Phasentrennung bei hohen Volumenströmen, bzw. die für eine stabile Phasentrennung vergleichsweise hohen zu

überwindenden Druckverluste identifiziert. Die Phasentrennung wurde daraufhin in Abschnitt 4.1 mit der in Kapitel 3 erarbeiteten Methode numerisch untersucht. Dabei wurden Beschleunigungsdruckverluste in den folgenden Leitungen als ausschlaggebend für die abweichenden Vorhersagen der bisherigen Modelle identifiziert. Zur Begrenzung der Beschleunigungseffekte wurde in Abschnitt 4.2 die Umwandlung der Pfropfenströmung in eine Parallelströmung zur einfachen Phasentrennung bei gleichzeitig geringem hydrodynamischen Druckverlust untersucht. Es wurde ein Vorhersagemodel entwickelt, welches in Abhängigkeit der Volumenströme und Stoffdaten der eintretenden Pfropfenströmung die Stabilität der Parallelströmung beschreiben kann, jedoch keine Aussage über die beobachtete Tropfenbildung trifft. Die auftretenden Tropfen können zu Verunreinigungen während der Phasentrennung führen und erfordern dadurch wiederum hohe Kapillardrücke zur Absicherung. Dies zieht kleine Spalte und Poren mit sich und bringt dadurch wieder hohe Druckverluste ein. Die Vermeidung kleiner Spalte und Poren diente neben der Reduzierung des Druckverlustes der Erschließung von Suspensionen als Pfropfenphase. Die Erarbeitung eines neuen Aktuatorkonzeptes für die gezieltere Regelung von parallelisierten Mikrokapillaren mit ferrofluidischen Pfropfen wurde in Abschnitt 4.3 begonnen. Die Kombination von überwachendem Sensor und regelndem Aktuator in einem Bauteil verspricht eine starke Kostensenkung, sobald die Detektion der Pfropfen durch verbesserte Messelektronik oder optimierte Spulenparameter auch bei Längen im Bereich von 3 mm möglich ist. Trotz der gewonnen Erkenntnisse in den einzelnen Bereichen bleibt die Pfropfenströmung in der Anwendung gegenüber herkömmlichen Extraktionskolonnen unattraktiv für die Bewältigung gewöhnlicher Extraktionsaufgaben.[4]

Dem ersten Szenario steht die Identifikation alternativer Anwendungsgebiete als zweites Szenario gegenüber. Die bisherigen Schwierigkeiten makroskopischer Apparate zur Extraktion bei kleinen Dichtedifferenzen, hohen Viskositäten oder großer Grenzflächenspannung stellen für die Pfropfenströmung kein Hindernis dar. Am Beispielsystem Wasser/Aceton/Silikonöl wurde in Abschnitt 4.4 bei den Nennviskositäten $20\,\mathrm{mPa \cdot s}$, $50\,\mathrm{mPa \cdot s}$ und $100\,\mathrm{mPa \cdot s}$ nachgewiesen, dass der Stofftransport trotz gesteigerter Viskosität mit niedrigviskosen Systemen der Literatur vergleichbar ist und damit weit über den Transportraten makroskopischer Apparate liegt. Die Anwendbarkeit ist weiterhin durch den maximalen Durchsatz limitiert. Zur Überwindung wurde in Abschnitt 4.5 ein Parallelisierungskonzept für die Reduktion der Zahl an mindestens notwendigen Pumpen hergeleitet und experimentell untersucht. Die Zahl an Pumpen zur Parallelisierung von m Extraktionssträngen mit jeweils n gegenstromartig verschalteten Mixer-Settler-Einheiten kann hierdurch von $m \cdot (n+1)$ auf bis zu $2 \cdot m + n - 1$ gesenkt werden. Die erarbeiteten Ergebnisse machen die Pfropfenströmung damit im Vergleich zu Extraktionskolonnen bei der Extraktion aus viskosen Medien heraus oder bei verschwindender Dichtedifferenz attraktiver, wodurch ein konkretes Anwendungsfeld für die Pfropfenströmung als Extraktionswerkzeug identifiziert und näher untersucht wurde.

[4]Unter einer gewöhnlichen Extraktionsaufgabe wird nach Pfennig eine Extraktion ohne herausragende Besonderheiten in den Stoffwerten verstanden: $\Delta\rho > 50\,\mathrm{kg/m^3}$; $\eta \simeq 1\,\mathrm{mPa \cdot s}$; $\sigma > 5\,\mathrm{mN/m}$; $n_{th} < 10$ [3]

Kapitel 5

Fazit und Ausblick

In dieser Arbeit wurde die flüssig-flüssig Pfropfenströmung als Extraktionswerkzeug auf ihre möglichen Anwendungsgebiete hin untersucht. Um als Extraktionswerkzeug attraktiv eingesetzt zu werden, muss sie sich gegenüber bisherigen Lösungen und Apparaten für Extraktionsaufgaben als vorteilhaft erweisen. Hierfür wurden zwei Ansätze unterschieden und parallel untersucht.

Der erste Ansatz fokussierte dabei, wie die Pfropfenströmung ihre bisherigen Nachteile als Anwendung der Mikroverfahrenstechnik überwinden kann, um dadurch zu bisherigen Apparaten konkurrenzfähig zu werden. Als größter Nachteil wurde hier eine fehlende Numbering-Up-Strategie identifiziert, welche zudem kosteneffizient anzuwenden ist. Hierauf begründet wurde ein alternatives Sensor- und Aktuatorkonzept für ferrofluidische Pfropfenströmungen durch Magnetfelder vorgestellt. Die eingesetzten Spulen können dabei durch Überwachung ihrer Induktivität die Strömung bezüglich Geschwindigkeit und/oder Pfropenlänge vermessen und die Strömung gleichzeitig durch Erzeugung von Magnetfeldern beeinflussen. Die Kombination von Sensor und Aktuator in einem Bauteil, der magnetischen Spule, verspricht eine signifikante Kostenreduktion. Die Detektierbarkeit langer Pfropfen wurde hierfür als erster Schritt nachgewiesen. Voraussetzung für die Anwendbarkeit dieses Konzeptes ist jedoch die Möglichkeit zum Einsatz von Ferrofluiden und eine stabile, druckverlustarme Phasentrennung der Pfropfenströmung. An einer Beispielgeometrie wurden daher repräsentativ für eine bestimmte Bauart von Phasentrennern, die allgemein besonders niedrige Druckverluste aufweist, die Vorgänge während der Phasentrennung numerisch nachverfolgt. Ziel der numerischen Untersuchung war die Identifikation von Modellfehlern bei der Vorhersage des stabilen Betriebsfensters für die Phasentrennung. Hierfür wurde auf Basis von bisherigen numerischen Arbeiten und Modellen ein hybrider Ansatz zur Vereinigung der Vorteile von Phase-Field-Methode, THINC-Methode, dem Levelset-Ansatz und der Volume of Fluid-Methode kondensiert und auf das Minimieren parasitärer Strömungen an freien Grenzflächen hin optimiert. Das numerische Modell wurde anhand von Druckverlusten und Wandfilmdicken aus der Literatur validiert. Im Vergleich zum Standardsolver von OpenFOAM 6, interFoam, konnten die parasitären Strömungen um zwei bis annähernd drei Größenordnungen reduziert werden. Dieser Aspekt bildete den Hauptteil der vorliegenden Arbeit. Die numerische Betrachtung der Phasentrennung konnte dabei instationäre Effekte durch stetige Beschleunigungs- und Abbremsvorgänge im Trennraum als Ursache für das Versagen der Phasentrennung bei hohen Druchsätzen nachweisen, die in bisherigen Modellierungen des möglichen Betriebsfensters

noch nicht berücksichtigt wurden. Die Simulationen erweitern damit das Verständnis der ablaufenden Prozesse im Phasentrenner. Als Alternative zu der numerisch untersuchten Bauform wurde experimentell die Umwandlung der Pfropfenströmung in eine Parallelströmung untersucht, da hierdurch die instationären Effekte auf den Trennraum begrenzt werden können und sich nicht zusätzlich auf die peripheren Anschlussleitungen auswirken. Die prinzipielle Stabilität von Parallelströmungen in oberflächenmodifizierten Rechteckkanälen großer Breite wurde über ein Vorhersagemodell beschrieben, welches jedoch nicht das Phänomen der Tropfenbildung an der Grenzfläche abbildet. Durch die Tropfenbildung in Kombination mit den niedrigen Kapillardrücken wird die Umwandlung in eine Parallelströmung in breiten Kanälen als unzureichendes Trennkonzept eingestuft. Abschließend konnten für die Verfolgung des ersten Ansatzes die bisherigen Nachteile bei der Anwendung der Pfropfenströmung als Extraktionswerkzeug nur in geringem Maße verringert werden.

Daraus ergab sich der zweite, parallel verfolgte Ansatz dieser Arbeit, nach dem die Pfropfenströmung Anwendungsgebiete erschließen kann, welche bisherigen Extraktionsapparaten gar nicht oder nur schwer zugänglich sind. Eine Literaturrecherche ergab dabei Potential in der Bewältigung von Extraktionsaufgaben mit geringer Dichtedifferenz der Fluide, großer Grenzflächenspannung und gesteigerter Viskosität. Alle Faktoren behindern den reibungslosen Gegenstrombetrieb von Füllkörper- und Packungskolonnen. Hieraus motiviert, wurde am Stoffsystem Wasser/Aceton/-Silikonöl beispielhaft die Stofftransportleistung der Pfropfenströmung untersucht. Die experimentell nachgewiesenen Stoffübergangskoeffizienten liegen mit 0,03 1/s bis 0,13 1/s trotz verhältnismäßig großem Innendurchmesser der Kapillaren von 1,6 mm und der erhöhten Viskosität der kontinuierlichen Phase in ähnlicher Größenordnung wie bisherige Literaturbefunde für niedrigviskose Stoffsysteme. Der positive Effekt beim Einsatz des viskosen Fluids als kontinuierliche Phase wurde in Kombination mit bisherigen Untersuchungen der Literatur auf eine verbesserte Nutzung des Wandfilmbereichs durch geringere Verarmung an Solvatstoff zurückgeführt. Da für den Einsatz als Extraktionswerkzeug weiterhin eine effektive Parallelisierungsstrategie fehlt, wurde ein alternatives Verschaltungskonzept entwickelt und untersucht, die Kreuz-Gegenstrom-Verschaltung. Das Konzept reduziert die notwendige Zahl an kostspieligen Pumpen für die Realisierung von m Extraktionssträngen zu jeweils n Stufen von den bisher notwendigen $m \cdot (n+1)$ schließlich auf bis zu $2 \cdot m + n - 1$. Das Konzept garantiert dabei im Vergleich zu anderen Strategien exakt die Gleichverteilung der Volumenströme auf die einzelnen m Stränge, welche durch die Pumpen vordefiniert werden. Diese Gleichverteilung ist durch den Verteilungskoeffizienten zwischen den Phasen bei Extraktionen von besonderer Wichtigkeit. Erste experimentelle Untersuchungen bestätigen die prinzipielle Umsetzbarkeit. Größere Verschaltungen benötigen jedoch noch weitere Optimierungsschritte. Demnach wird die Pfropfenströmung als Extraktionswerkzeug nicht für den breiten Einsatz in bisherigen Extraktionsvorgängen, sondern in der Bewältigung spezieller Trennaufgaben gesehen, welche aufgrund des erschwerten Gegenstrombetriebs in makroskopischen Kolonnen nur schwer realisiert werden können.

Für die verbesserte Anwendbarkeit der Pfropfenströmung bei herkömmlichen Trennaufgaben sollte in zukünftigen Arbeiten die Detektion von kurzen ferrofluidischen Pfropfen weiterentwickelt werden. Es ist zudem eine allgemeine Beschreibung der Förderwirkung in Abhängigkeit der noch zu modifizierenden Spulengeometrie notwendig. Mit Hilfe der entwickelten numerischen Methode steht dabei weiterfüh-

rend ein Werkzeug zur Verfügung, welches besonders arm an parasitären Strömungen ist und dadurch sehr genau die Bewegung der Grenzfläche in Mikrokanälen abbilden kann. Hiermit besteht die Möglichkeit zur Untersuchung alternativer Trennapparate. Zur weiteren Beschleunigung der Simulationen kommen neben der bereits genutzten openMP-Bibliothek zur Parallelisierung von Schleifen noch die Implementierung von zerlegten Rechengittern und die Kombination beider Ansätze in Betracht.

Als Extraktionswerkzeug für hochviskose Medien kann die Datengrundlage durch Vermessung des Stofftransportes bei weiteren Viskositäten, Pfropfenlängen und Kapillardurchmessern erweitert werden. Der Modellvorstellung nach wird es eine Viskosität geben, ab der ein reduzierter Diffusionskoeffizient in der viskosen kontinuierlichen Phase die vergrößerte Speicherkapazität des Wandfilms überwiegen wird und der Gesamtstofftransport dadurch sinkt. Diese Grenzen gilt es näher zu charakterisieren. Zur Parallelisierung steht durch die vorliegende Arbeit dabei das Konzept der Kreuz-Gegenstromverschaltung zur Verfügung. Für dieses Konzept sollte eine konkrete Strategie zur Einstellung der Drosselventile entwickelt werden, die den Aufbau und die Inbetriebnahme von größeren Verschaltungen erleichtert. Eine Verbesserung der Phasentrenner kann zudem das stabile Betriebsfenster der Gesamtverschaltung vergrößern. Hierfür sollte am Boden der Trennkammern ebenfalls eine Membran platziert werden, welche Wasser weitgehend ungehindert permeieren lässt, der Permeation von Organik jedoch einen großen Kapillardruck entgegensetzt. Ein solches Membranmaterial wurde bisher nicht gefunden.

Kapitel 6

Symbol- und Abkürzungsverzeichnis

Zur deutlichen Kenntlichmachung, an welchen Stellen in den dargestellten Gleichungen mit Vektoren und Matrizen zu rechnen ist, werden Vektoren stets mit einem Vektorpfeil bezeichnet und Multiplikationen zweier Matrizen oder Vektoren (z.B. im Skalarprodukt) mit den Symbol $*$ dargestellt. Dies soll Missverständnisse vermeiden ($\vec{u}$ als Geschwindigkeitsvektor gegenüber u als Geschwindigkeitskomponente in x-Richtung, bzw. Durchmesser d und Abstandvektor $\vec{d}$).

lateinisches Symbol	mögliche Bedeutungen des Symbols	Einheit
a	Pulsationshub	m
	Orientierung Krümmungsradius	$-$
	spezifische Austauschfläche	$\frac{m^2}{m^3}$
a_i	Polynomkoeffizienten in $f(L)$	$-$
	Korrelationskoeffizienten für U_{Fehl}	$-$
A	Fläche	m^2
B	kombinierte spezifische Belastung der Kolonne	$\frac{m^3}{m^2 \cdot h}$
	Kanalbreite	m
C	Kapazität des Kondensators	F, $\frac{A^2 \cdot s^4}{kg \cdot m^2}$
C_1, C_2	Konstanten in Abschätzung für Δt_{cap}	$-$
C_k	Konstante in DGL-Lösung	$\frac{1}{m}$
C_m	Konstante in Fehlerabschätzung	$-$
c_α	Kompressionsstärke im Solver interFoam	$-$
c	Konzentration einer Komponente in Lösung	$\frac{mol}{m^3}$
	Volumenanteil disperse Phase in Rechenzelle	$-$
d	Exponent in Literaturstelle nach Olsson	$-$
	Absolutglied eines Polynoms	$-$
	Durchmesser	m
D	Diffusionskoeffizient	$\frac{m^2}{s}$
	Kompressionskoeffizient	$\frac{m^3}{s}$
e	Fehlerschätzer	$-$
E	Extraktionsgrad	$-$
$\vec{e}_i$	Einheitsvektor in Raumrichtung i	$-$
F	freie Energie	J

	Füllgrad Zelle nach Volume-of-Fluid Methode	$-$
$F_i(L)$	aus $\vec{g}_i$ abgespaltener Funktionsterm	m^i
f	freie Energiedichte konstruierte Formfunktion der Grenzfläche Pulsationsfrequenz	$\frac{J}{m^3}$ $-$ $\frac{1}{s}$
$\vec{g}$	Erdbeschleunigung	$\frac{m}{s^2}$
$\vec{g}_i$	Vektorpotential zu $f(L)$ bzw. $F_{i-1}(L)$	m^i
H	Kanaltiefe	m
i	imaginäre Einheit $\sqrt{-1}$	$-$
I	Stromstärke	A
$\vec{k}$	Kompressionsfluss nach Olsson Wellenzahlvektor	$\frac{m}{s}$ $\frac{1}{m}$
K_1, K_2	Geometriekonstante in Flutpunktkorrelation	$\sqrt{\frac{m}{s}}$
L	psuedo-Levelsetfunktion Induktivität	$-$ H, $\frac{kg \cdot m^2}{A^2 \cdot s^2}$
l	Länge	m
$\vec{n}$	Normalenvektor der Länge 1	$-$
n, N	Anzahl z.B. theoretischer Trennstufen	$-$
$P(x,y,z)$	mehrdimensionales Polynom	$-$
p	Druck	$\frac{kg}{m \cdot s^2}$
$p(L)$	polynomieller Anteil von $f(L)$	$-$
Q	Residuum Korrelationsanpassung	$-$
R	Radius	m
t	Zeit	s
$\vec{u}$, $\vec{U}$	Geschwindigkeit	$\frac{m}{s}$
U	Spannung	V, $\frac{kg \cdot m^2}{A \cdot s^3}$
V	Volumen	m^3
x,y,z	Raumkoordinaten	m
Z	Impedanz	Ω, $\frac{kg \cdot m^2}{A^2 \cdot s^3}$

griechisches Symbol	mögliche Bedeutungen des Symbols	Einheit
α	Dispersphasenanteil im Solver interFoam	$-$
α, β	Parameter in Literaturstellen nach Jacqmin [59] [60] Parameter in konstruierter Formfunktion $f(L)$	$-$ $-$
α, β, γ	Komponentenzerlegung von $\vec{\nabla} L$	$\frac{1}{m}$
α_k, β_k, γ_k	Koeffizienten für Randbedingung	m^2, $\frac{m}{s}$, $\frac{m}{s}$
δ	Inkrement der nachgestellten Menge Delta-Distribution/Dirac'sche Deltafunktion	$var.$ $\frac{1}{m}$
Δ	Differenz zweier Werte differentieller Laplace-Operator	$var.$ $\frac{1}{m^2}$
ε	numerische Dicke der Grenzfläche	m, (m^2 in [58])
Φ	(pseudo-)chemisches Potential Levelsetfunktion nach Literaturstelle Osher et al. [77] Volumenstromverhältnis	$\frac{1}{m}$ $-$ $-$
Ψ	Potentialfunktion in Phase-Field-Methode	$-$
$\vec{\nabla}$	differentieller Nabla-Operator	$\frac{1}{m}$

ω	Geschwindigkeit in Flutpunktkorrelation	$\frac{m}{s}$
ρ	Dichte	$\frac{kg}{m^3}$
κ	Krümmung der Grenzfläche	$\frac{1}{m}$
λ	transformierter Laplace-Operator	$\frac{1}{m^2}$
σ	Grenzflächenspannung	$\frac{kg}{s^2}$
η	dynamische Viskosität	$\frac{kg}{m \cdot s}$
τ	Pseudozeitskala	s
ξ	Stelle im Intervall bei Fehler-abschätzung in Talyorformel	*var.*

Abkürzung	Bedeutung
CLS-VOF	coupled levelset volume of fluid
CPU	central processing unit
FEP	Perfluorethylpropylen
GAMG	gneralized geometric algebraic multigrid
DGL	Differentialgleichung
disp.	disperse Phase in Mehrphasensystemen
Gl.	Gleichung
kont.	kontinuierliche Phase in Mehrphasensystemen
PBiCG	Preconditioned bi-conjugate gradient
PC	Polycarbonat
PMMA	Polymethylmethacrylat
PTFE	Polytetrafluorethylen
THINC	tangent of hyperbola for interface capturing
var.	variabel, je nach Anwendungsfall verschieden
TVD	total variation dimishing
VOF	volume of fluid

Dimensionslose Kennzahl	Bezeichnung	Definition	Beschriebenes Verhältnis
Ca	Kapillar-Zahl	$u \cdot \eta / \sigma$	Viskose Kraft zu Grenzflächenkraft
Co	Courant-Zahl	$u \cdot \Delta t / \Delta x$	Konvektiver Informationstransport zu Gitterweite
Pe	Peclet-Zahl	$u \cdot L / D$	Konvektiver zu diffusiver Stofftransport
Re	Reynolds-Zahl	$u \cdot \rho \cdot L / \eta$	Trägheitskraft zu viskoser Kraft
We	Weber-Zahl	$u^2 \cdot \rho \cdot L / \sigma$	Trägheitskraft zu Grenzflächenkraft

Literatur

[1] M. N. Kashid und L. Kiwi-Minsker. "Microstructured Reactors for Multiphase Reactions: State of the Art". In: *Industrial & Engineering Chemistry Research* 48.14 (Juli 2009), S. 6465–6485. DOI: 10.1021/ie8017912.

[2] H. A. M. Wilhelm R.A. Vauck. *Grundoperationen chemischer Verfahrenstechnik*. DVG Deutshcer Verlag für Grundstoffindustrie, 2000.

[3] A. Pfennig, T. Pilhofer und J. Schröter. *Fluidverfahrenstechnik Band 2 (Kapitel 10, Flüssig-Flüssig-Extraktion)*. Hrsg. von R. Goedecke. Wiley VCH Verlag GmbH, 21. Sep. 2011. ISBN: 3527332707. URL: https://www.ebook.de/de/product/16502943/fluidverfahrenstechnik.html.

[4] A. Mersmann, M. Kind und J. Stichlmair. *Thermische Verfahrenstechnik*. Springer-Verlag GmbH, 20. Juni 2005. ISBN: 3540236481. URL: https://www.ebook.de/de/product/5305677/alfons_mersmann_matthias_kind_johann_stichlmair_thermische_verfahrenstechnik.html.

[5] S. Soboll und N. Kockmann. "Hydrodynamics and Mass Transfer in a Lab-Scale Stirred-Pulsed Extraction Column". In: *Chemical Engineering & Technology* 41.9 (Juli 2018), S. 1847–1856. DOI: 10.1002/ceat.201800283.

[6] F. Scheiff, M. Mendorf, D. Agar, N. Reis und M. Mackley. "The separation of immiscible liquid slugs within plastic microchannels using a metallic hydrophilic sidestream". In: *Lab on a Chip* 11.6 (2011), S. 1022. DOI: 10.1039/c0lc00442a.

[7] D. Harvie, M. Davidson und M. Rudman. "An analysis of parasitic current generation in Volume of Fluid simulations". In: *Applied Mathematical Modelling* 30.10 (Okt. 2006), S. 1056–1066. DOI: 10.1016/j.apm.2005.08.015.

[8] A. Schönbucher. *Thermische Verfahrenstechnik*. Springer-Verlag, 2002. ISBN: 3-540-42005-3.

[9] J. Maćkowiak. *Fluiddynamik von Füllkörpern und Packungen*. Springer Berlin Heidelberg, 2003. DOI: 10.1007/978-3-642-55575-6.

[10] J. S. Watson, L. E. McNeese, J. Day und P. A. Carroad. "Flooding rates and holdup in packed liquid-liquid extraction columns". In: *AIChE Journal* 21.6 (Nov. 1975), S. 1080–1086. DOI: 10.1002/aic.690210606.

[11] R. Houlihan und J. Landau. "Packed extraction columns: A review of correlations for maximum throughput". In: *The Canadian Journal of Chemical Engineering* 52.6 (Dez. 1974), S. 758–766. DOI: 10.1002/cjce.5450520609.

[12] M. Schmalenberg, T. A. Frede, C. Mathias und N. Kockmann. “Efficient Shortcut Method for Determining the Process Window in Stirred-Pulsed Extraction Columns”. In: *Chemie Ingenieur Technik* 93.3 (Jan. 2021), S. 466–472. DOI: 10.1002/cite.202000066.

[13] E. Y. Kenig, Y. Su, A. Lautenschleger, P. Chasanis und M. Grünewald. “Micro-separation of fluid systems: A state-of-the-art review”. In: *Separation and Purification Technology* 120 (Dez. 2013), S. 245–264. DOI: 10.1016/j.seppur.2013.09.028.

[14] J. Bobers, J. Grühn, S. Höving, T. Pyka und N. Kockmann. “Two-Phase Flow in a Coiled Flow Inverter: Process Development from Batch to Continuous Flow”. In: *Organic Process Research & Development* 24.10 (Juni 2020), S. 2094–2104. DOI: 10.1021/acs.oprd.0c00152.

[15] P. Sobieszuk, J. Aubin und R. Pohorecki. “Hydrodynamics and Mass Transfer in Gas-Liquid Flows in Microreactors”. In: *Chemical Engineering & Technology* 35.8 (Juli 2012), S. 1346–1358. DOI: 10.1002/ceat.201100643.

[16] M. Kashid und L. Kiwi-Minsker. “Quantitative prediction of flow patterns in liquid–liquid flow in micro-capillaries”. In: *Chemical Engineering and Processing: Process Intensification* 50.10 (Okt. 2011), S. 972–978. DOI: 10.1016/j.cep.2011.07.003.

[17] A. Holbach und N. Kockmann. “Counter-current arrangement of microfluidic liquid-liquid droplet flow contactors”. In: *Green Processing and Synthesis* 2.2 (Jan. 2013). DOI: 10.1515/gps-2013-0006.

[18] K. G. Biswas, G. Das, S. Ray und J. K. Basu. “Mass transfer characteristics of liquid–liquid flow in small diameter conduits”. In: *Chemical Engineering Science* 122 (Jan. 2015), S. 652–661. DOI: 10.1016/j.ces.2014.07.029.

[19] F. Scheiff, A. Holbach und D. W. Agar. “Slug Flow of Ionic Liquids in Capillary Microcontactors: Fluid Dynamic Intensification for Solvent Extraction”. In: *Chemical Engineering & Technology* 36.6 (Mai 2013), S. 975–984. DOI: 10.1002/ceat.201200600.

[20] I. Dittmar und P. Ehrhard. “Numerical study of a liquid/liquid slug flow in a micro-reactor”. In: *PAMM* 12.1 (Dez. 2012), S. 511–512. DOI: 10.1002/pamm.201210244.

[21] I. Dittmar. “Numerische Untersuchung der flüssig, flüssig Pfropfenströmung in einem Mikrokapillarreaktor”. Diss. TU Dortmund, 2015.

[22] F. Kaske, S. Dick, S. A. Pajoohi und D. W. Agar. “The influence of operating conditions on the mass transfer performance of a micro capillary contactor with liquid–liquid slug flow”. In: *Chemical Engineering and Processing: Process Intensification* 108 (Okt. 2016), S. 10–16. DOI: 10.1016/j.cep.2016.06.010.

[23] F. Kaske. “Experimentelle und theoretische Untersuchungen zum Flüssig-Flüssig-Stofftransport in Mikrokapillaren”. Diss. TU Dortmund, 2019.

[24] L. Arsenjuk, M. Asshoff, J. Kleinheider und D. W. Agar. “A device for continuous and flexible adjustment of liquid-liquid slug size in micro-channels”. In: *Journal of Flow Chemistry* 10.2 (Feb. 2020), S. 409–422. DOI: 10.1007/s41981-019-00064-7.

[25] N. Antweiler, S. Gatberg, J. Franzke und D. W. Agar. "Neue kosteneffektive Mess- und Regeltechnik für das Numbering-up von reaktiven Pfropfenströmungen in Mikrokanälen". In: *Chemie Ingenieur Technik* 87.9 (Juni 2015), S. 1221–1229. DOI: 10.1002/cite.201500032.

[26] M. N. Kashid, Y. M. Harshe und D. W. Agar. "Liquid-Liquid Slug Flow in a Capillary: An Alternative to Suspended Drop or Film Contactors". In: *Industrial & Engineering Chemistry Research* 46.25 (Dez. 2007), S. 8420–8430. DOI: 10.1021/ie070077x.

[27] Susanti, J. G. M. Winkelman, B. Schuur, H. J. Heeres und J. Yue. "Lactic Acid Extraction and Mass Transfer Characteristics in Slug Flow Capillary Microreactors". In: *Industrial & Engineering Chemistry Research* 55.16 (Apr. 2016), S. 4691–4702. DOI: 10.1021/acs.iecr.5b04917.

[28] D. Peroni, W. van Egmond, W. T. Kok und H.-G. Janssen. "Advancing liquid/liquid extraction through a novel microfluidic device: Theory, instrumentation and applications in gas chromatography". In: *Journal of Chromatography A* 1226 (Feb. 2012), S. 77–86. DOI: 10.1016/j.chroma.2011.08.001.

[29] A. J. Ufer. "Untersuchungen zu neuartigen Reaktionsführungen von flüssig-flüssig Pfropfenströmungen im Kapillarmikrokontaktor". Diss. Technische Universität Dortmund, 2015.

[30] M. Bussmann. "Entwicklung eines Mikrosystemmoduls aus Kunststofffolien zur Phasentrennung". Masterarbeit. Technische Universität Dortmund, Fakultät für Bio- und Chemieingenieruwesen, Arbeitsgruppe Apparatedesign, 2015.

[31] O. K. Castell, C. J. Allender und D. A. Barrow. "Liquid–liquid phase separation: characterisation of a novel device capable of separating particle carrying multiphase flows". In: *Lab Chip* 9.3 (2009), S. 388–396. DOI: 10.1039/b806946h.

[32] D. M. Fries, T. Voitl und P. R. von Rohr. "Liquid Extraction of Vanillin in Rectangular Microreactors". In: *Chemical Engineering & Technology* 31.8 (Aug. 2008), S. 1182–1187. DOI: 10.1002/ceat.200800169.

[33] A. Günther, M. Jhunjhunwala, M. Thalmann, M. A. Schmidt und K. F. Jensen. "Micromixing of Miscible Liquids in Segmented Gas-Liquid Flow". In: *Langmuir* 21.4 (Feb. 2005), S. 1547–1555. DOI: 10.1021/la0482406.

[34] W. Gaakeer, M. de Croon, J. van der Schaaf und J. Schouten. "Liquid–liquid slug flow separation in a slit shaped micro device". In: *Chemical Engineering Journal* 207-208 (Okt. 2012), S. 440–444. DOI: 10.1016/j.cej.2012.06.148.

[35] A. Adamo, P. L. Heider, N. Weeranoppanant und K. F. Jensen. "Membrane-Based, Liquid–Liquid Separator with Integrated Pressure Control". In: *Industrial & Engineering Chemistry Research* 52.31 (Juli 2013), S. 10802–10808. DOI: 10.1021/ie401180t.

[36] J. G. Kralj, H. R. Sahoo und K. F. Jensen. "Integrated continuous microfluidic liquid–liquid extraction". In: *Lab Chip* 7.2 (2007), S. 256–263. DOI: 10.1039/b610888a.

[37] S. K. Kurt. "Design and Characterization of Tunular Equipment for Process Intensification". Diss. TU Dortmund University, 2017.

[38] A. Aota, K. Mawatari, S. Takahashi, T. Matsumoto, K. Kanda, R. Anraku, A. Hibara, M. Tokeshi und T. Kitamori. “Phase separation of gas–liquid and liquid–liquid microflows in microchips”. In: *Microchimica Acta* 164.3-4 (Nov. 2008), S. 249–255. DOI: 10.1007/s00604-008-0085-3.

[39] E. Kolehmainen und I. Turunen. “Micro-scale liquid–liquid separation in a plate-type coalescer”. In: *Chemical Engineering and Processing: Process Intensification* 46.9 (Sep. 2007), S. 834–839. DOI: 10.1016/j.cep.2007.05.027.

[40] F. Scheiff. “Kombinierte Fluiddynamik, Stofftransporte und Reaktion der Suspensionskatalyse bei der Flüssig/flüssig-Pfropfenströmung in Mikrokanälen”. Diss. Technische Universität Dortmund, 2014.

[41] D. Tsaoulidis und P. Angeli. “Effect of channel size on mass transfer during liquid–liquid plug flow in small scale extractors”. In: *Chemical Engineering Journal* 262 (Feb. 2015), S. 785–793. DOI: 10.1016/j.cej.2014.10.012.

[42] J. R. Burns und C. Ramshaw. “The intensification of rapid reactions in multiphase systems using slug flow in capillaries”. In: *Lab on a Chip* 1.1 (2001), S. 10. DOI: 10.1039/b102818a.

[43] A.-L. Dessimoz, L. Cavin, A. Renken und L. Kiwi-Minsker. “Liquid–liquid two-phase flow patterns and mass transfer characteristics in rectangular glass microreactors”. In: *Chemical Engineering Science* 63.16 (Aug. 2008), S. 4035–4044. DOI: 10.1016/j.ces.2008.05.005.

[44] C. Heckmann. “Spatially-resolved mass transport in a liquid/liquid slug-flow micro-capillary reactor”. Diss. TU Dortmund University, 2019.

[45] C. Heckmann, S. K. Kurt, P. Ehrhard und N. Kockmann. “Stofftransport in einer Flüssig/Flüssig-Pfropfenströmung im Mikrokapillarreaktor”. In: *Chemie Ingenieur Technik* 89.12 (Nov. 2017), S. 1642–1649. DOI: 10.1002/cite.201700030.

[46] N. Assmann, A. Ładosz und P. R. von Rohr. “Continuous Micro Liquid-Liquid Extraction”. In: *Chemical Engineering & Technology* 36.6 (Apr. 2013), S. 921–936. DOI: 10.1002/ceat.201200557.

[47] H. S. Santana, J. L. Silva, B. Aghel und J. Ortega-Casanova. “Review on microfluidic device applications for fluids separation and water treatment processes”. In: *SN Applied Sciences* 2.3 (Feb. 2020). DOI: 10.1007/s42452-020-2176-7.

[48] T. Xie, Y. Ma und C. Xu. “Passive continuous-flow microextraction/stripping system with high throughput”. In: *Chemical Engineering Science* 223 (Sep. 2020), S. 115745. DOI: 10.1016/j.ces.2020.115745.

[49] A. Aota, M. Nonaka, A. Hibara und T. Kitamori. “Countercurrent Laminar Microflow for Highly Efficient Solvent Extraction”. In: *Angewandte Chemie International Edition* 46.6 (Jan. 2007), S. 878–880. DOI: 10.1002/anie.200600122.

[50] M. Kashid, A. Gupta, A. Renken und L. Kiwi-Minsker. “Numbering-up and mass transfer studies of liquid–liquid two-phase microstructured reactors”. In: *Chemical Engineering Journal* 158.2 (Apr. 2010), S. 233–240. DOI: 10.1016/j.cej.2010.01.020.

[51] R. Schenk, V. Hessel, C. Hofmann, H. Löwe und F. Schönfeld. "Novel Liquid-Flow Splitting Unit Specifically Made for Numbering-Up of Liquid/Liquid Chemical Microprocessing". In: *Chemical Engineering & Technology* 26.12 (Dez. 2003), S. 1271–1280. DOI: 10.1002/ceat.200301867.

[52] R. Schenk, V. Hessel, C. Hofmann, J. Kiss, H. Löwe und A. Ziogas. "Numbering-up of micro devices: a first liquid-flow splitting unit". In: *Chemical Engineering Journal* 101.1-3 (Aug. 2004), S. 421–429. DOI: 10.1016/j.cej.2003.11.034.

[53] M. Al-Rawashdeh, X. Nijhuis, E. V. Rebrov, V. Hessel und J. C. Schouten. "Design methodology for barrier-based two phase flow distributor". In: *AIChE Journal* 58.11 (Feb. 2012), S. 3482–3493. DOI: 10.1002/aic.13750.

[54] M. Al-Rawashdeh, F. Yu, T. Nijhuis, E. Rebrov, V. Hessel und J. Schouten. "Numbered-up gas–liquid micro/milli channels reactor with modular flow distributor". In: *Chemical Engineering Journal* 207-208 (Okt. 2012), S. 645–655. DOI: 10.1016/j.cej.2012.07.028.

[55] M. Al-Rawashdeh, L. Fluitsma, T. Nijhuis, E. Rebrov, V. Hessel und J. Schouten. "Design criteria for a barrier-based gas–liquid flow distributor for parallel microchannels". In: *Chemical Engineering Journal* 181-182 (Feb. 2012), S. 549–556. DOI: 10.1016/j.cej.2011.11.086.

[56] M. Mendorf, H. Nachtrodt, A. Mescher, A. Ghaini und D. W. Agar. "Design and Control Techniques for the Numbering-up of Capillary Microreactors with Uniform Multiphase Flow Distribution". In: *Industrial & Engineering Chemistry Research* 49.21 (Nov. 2010), S. 10908–10916. DOI: 10.1021/ie100473d.

[57] L. Arsenjuk, N. von Vietinghoff, A. W. Gladius und D. W. Agar. "Actively homogenizing fluid distribution and slug length of liquid-liquid segmented flow in parallelized microchannels". In: *Chemical Engineering and Processing - Process Intensification* 156 (Okt. 2020), S. 108061. DOI: 10.1016/j.cep.2020.108061.

[58] E. Olsson und G. Kreiss. "A conservative level set method for two phase flow". In: *Journal of Computational Physics* 210.1 (Nov. 2005), S. 225–246. DOI: 10.1016/j.jcp.2005.04.007.

[59] D. Jacqmin. "Calculation of Two-Phase Navier–Stokes Flows Using Phase-Field Modeling". In: *Journal of Computational Physics* 155.1 (Okt. 1999), S. 96–127. DOI: 10.1006/jcph.1999.6332.

[60] D. Jacqmin. "Contact-line dynamics of a diffuse fluid interface". In: *Journal of Fluid Mechanics* 402 (Jan. 2000), S. 57–88. DOI: 10.1017/s0022112099006874.

[61] F. Xiao, Y. Honma und T. Kono. "A simple algebraic interface capturing scheme using hyperbolic tangent function". In: *International Journal for Numerical Methods in Fluids* 48.9 (2005), S. 1023–1040. DOI: 10.1002/fld.975.

[62] B. Xie, S. Ii und F. Xiao. "An efficient and accurate algebraic interface capturing method for unstructured grids in 2 and 3 dimensions: The THINC method with quadratic surface representation". In: *International Journal for Numerical Methods in Fluids* 76.12 (Okt. 2014), S. 1025–1042. DOI: 10.1002/fld.3968.

[63] B. Xie und F. Xiao. “Toward efficient and accurate interface capturing on arbitrary hybrid unstructured grids: The THINC method with quadratic surface representation and Gaussian quadrature”. In: *Journal of Computational Physics* 349 (Nov. 2017), S. 415–440. DOI: 10.1016/j.jcp.2017.08.028.

[64] M. P. Joel H. Ferziger. *Numerische Strömungsmechanik*. Springer Berlin Heidelberg, 2008. DOI: 10.1007/978-3-540-68228-8.

[65] I. N. Bronstein. *Taschenbuch der Mathematik*. Deutsch Harri GmbH, 2008. ISBN: 978-3-8171-2007-9.

[66] A. Ralston. “Runge-Kutta methods with minimum error bounds”. In: *Mathematics of Computation* 16.80 (1962), S. 431–431. DOI: 10.1090/s0025-5718-1962-0150954-0.

[67] H. Oertel, Hrsg. *Prandtl - Führer durch die Strömungslehre*. Springer Fachmedien Wiesbaden, 2016. DOI: 10.1007/978-3-658-08933-7.

[68] J. U. Brackbill, D. B. Kothe und C. Zemach. “A continuum method for modeling surface tension”. In: *Journal of Computational Physics* 100.2 (Juni 1992), S. 335–354. DOI: 10.1016/0021-9991(92)90240-y.

[69] J. Glimm, J. Grove, B. Lindquist, O. A. McBryan und G. Tryggvason. “The Bifurcation of Tracked Scalar Waves”. In: *SIAM Journal on Scientific and Statistical Computing* 9.1 (Jan. 1988), S. 61–79. DOI: 10.1137/0909006.

[70] S. O. Unverdi und G. Tryggvason. “A front-tracking method for viscous, incompressible, multi-fluid flows”. In: *Journal of Computational Physics* 100.1 (Mai 1992), S. 25–37. DOI: 10.1016/0021-9991(92)90307-k.

[71] S. Popinet und S. Zaleski. “A front-tracking algorithm for accurate representation of surface tension”. In: *International Journal for Numerical Methods in Fluids* 30.6 (Juli 1999), S. 775–793. DOI: 10.1002/(sici)1097-0363(19990730)30:6<775::aid-fld864>3.0.co;2-#.

[72] S. Muzaferija und M. Peri´c. “Computation of free-surface flows using the fintie-volume method and moving grids”. In: *Numerical Heat Transfer, Part B: Fundamentals* 32.4 (Dez. 1997), S. 369–384. DOI: 10.1080/10407799708915014.

[73] Ž. Tuković und H. Jasak. “A moving mesh finite volume interface tracking method for surface tension dominated interfacial fluid flow”. In: *Computers & Fluids* 55 (Feb. 2012), S. 70–84. DOI: 10.1016/j.compfluid.2011.11.003.

[74] F. H. Harlow und J. E. Welch. “Numerical Calculation of Time-Dependent Viscous Incompressible Flow of Fluid with Free Surface”. In: *Physics of Fluids* 8.12 (1965), S. 2182. DOI: 10.1063/1.1761178.

[75] C. Hirt und B. Nichols. “Volume of fluid (VOF) method for the dynamics of free boundaries”. In: *Journal of Computational Physics* 39.1 (Jan. 1981), S. 201–225. DOI: 10.1016/0021-9991(81)90145-5.

[76] M. Sussman, P. Smereka und S. Osher. “A Level Set Approach for Computing Solutions to Incompressible Two-Phase Flow”. In: *Journal of Computational Physics* 114.1 (Sep. 1994), S. 146–159. DOI: 10.1006/jcph.1994.1155.

[77] S. Osher und J. A. Sethian. “Fronts propagating with curvature-dependent speed: Algorithms based on Hamilton-Jacobi formulations”. In: *Journal of Computational Physics* 79.1 (Nov. 1988), S. 12–49. DOI: 10.1016/0021-9991(88)90002-2.

[78] A.-K. Tornberg und B. Engquist. “A finite element based level-set method for multiphase flow applications”. In: *Computing and Visualization in Science* 3.1-2 (Mai 2000), S. 93–101. DOI: 10.1007/s007910050056.

[79] H. Zhang, L. Zheng, V. Prasad und T. Hou. “A curvilinear level set formulation for highly deformable free surface problems with application to solidification”. In: *Numerical Heat Transfer, Part B: Fundamentals* 34.1 (Juli 1998), S. 1–30. DOI: 10.1080/10407799808915045.

[80] M. Shams, A. Q. Raeini, M. J. Blunt und B. Bijeljic. “A numerical model of two-phase flow at the micro-scale using the volume-of-fluid method”. In: *Journal of Computational Physics* 357 (März 2018), S. 159–182. DOI: 10.1016/j.jcp.2017.12.027.

[81] H. Hemida. *OpenFOAM tutorial: Free surface tutorial using interFoam and rasInterFoam*. Apr. 2008. URL: http://www.tfd.chalmers.se/~hani/kurser/OS_CFD_2007/HassanHemida/Hassan_Hemida_VOF.pdf,%20zuletzt%20abgerufen%2008.08.2019.

[82] B. Lafaurie, C. Nardone, R. Scardovelli, S. Zaleski und G. Zanetti. “Modelling Merging and Fragmentation in Multiphase Flows with SURFER”. In: *Journal of Computational Physics* 113.1 (Juli 1994), S. 134–147. DOI: 10.1006/jcph.1994.1123.

[83] D. Jamet, D. Torres und J. Brackbill. “On the Theory and Computation of Surface Tension: The Elimination of Parasitic Currents through Energy Conservation in the Second-Gradient Method”. In: *Journal of Computational Physics* 182.1 (Okt. 2002), S. 262–276. DOI: 10.1006/jcph.2002.7165.

[84] M. M. Francois, S. J. Cummins, E. D. Dendy, D. B. Kothe, J. M. Sicilian und M. W. Williams. “A balanced-force algorithm for continuous and sharp interfacial surface tension models within a volume tracking framework”. In: *Journal of Computational Physics* 213.1 (März 2006), S. 141–173. DOI: 10.1016/j.jcp.2005.08.004.

[85] M. Aboukhedr, A. Georgoulas, M. Marengo, M. Gavaises und K. Vogiatzaki. “Simulation of micro-flow dynamics at low capillary numbers using adaptive interface compression”. In: *Computers & Fluids* 165 (März 2018), S. 13–32. DOI: 10.1016/j.compfluid.2018.01.009.

[86] C. Galusinski und P. Vigneaux. “On stability condition for bifluid flows with surface tension: Application to microfluidics”. In: *Journal of Computational Physics* 227.12 (Juni 2008), S. 6140–6164. DOI: 10.1016/j.jcp.2008.02.023.

[87] M. Sussman und M. Ohta. “A Stable and Efficient Method for Treating Surface Tension in Incompressible Two-Phase Flow”. In: *SIAM Journal on Scientific Computing* 31.4 (Jan. 2009), S. 2447–2471. DOI: 10.1137/080732122.

[88] H. Ganapathy, E. Al-Hajri und M. M. Ohadi. “Phase field modeling of Taylor flow in mini/microchannels, Part I: Bubble formation mechanisms and phase field parameters”. In: *Chemical Engineering Science* 94 (Mai 2013), S. 138–149. DOI: 10.1016/j.ces.2013.01.049.

[89] R. Nochetto, M. Paolini und C. Verdi. “A Dynamic Mesh Algorithm for Curvature Dependent Evolving Interfaces”. In: *Journal of Computational Physics* 123.2 (Feb. 1996), S. 296–310. DOI: 10.1006/jcph.1996.0025.

[90] Q. He und N. Kasagi. “Phase-Field simulation of small capillary-number two-phase flow in a microtube”. In: *Fluid Dynamics Research* 40.7-8 (Juli 2008), S. 497–509. DOI: 10.1016/j.fluiddyn.2008.01.002.

[91] H. Ganapathy, E. Al-Hajri und M. M. Ohadi. “Phase field modeling of Taylor flow in mini/microchannels, Part II: Hydrodynamics of Taylor flow”. In: *Chemical Engineering Science* 94 (Mai 2013), S. 156–165. DOI: 10.1016/j.ces.2013.01.048.

[92] M. Coquerelle und S. Glockner. “A fourth-order accurate curvature computation in a level set framework for two-phase flows subjected to surface tension forces”. In: *Journal of Computational Physics* 305 (Jan. 2016), S. 838–876. DOI: 10.1016/j.jcp.2015.11.014.

[93] H. Zhao, P. Ming, W. Zhang und J. Chen. “A direct time-integral THINC scheme for sharp interfaces”. In: *Journal of Computational Physics* 393 (Sep. 2019), S. 139–161. DOI: 10.1016/j.jcp.2019.05.011.

[94] L. Qian und Y. Wei. “A coupled THINC/QQ and LS framework for simulating incompressible free-surface flows with surface tension”. In: *Computers & Fluids* 187 (Juni 2019), S. 12–26. DOI: 10.1016/j.compfluid.2019.05.003.

[95] M. M. Kamra und C. Hu. “Implementation of Unstructured Multi-dimensional THINC for Practical Multi-Phase Flow Simulations”. In: *Evergreen* 4.1 (März 2017), S. 52–57. DOI: 10.5109/1808453.

[96] Y. Li und C. Yu. “Research on dam-break flow induced front wave impacting a vertical wall based on the CLSVOF and level set methods”. In: *Ocean Engineering* 178 (Apr. 2019), S. 442–462. DOI: 10.1016/j.oceaneng.2019.02.064.

[97] Y. Li, Y. Ma, R. Deng, D. Jiang und Z. Hu. “Research on dam-break induced tsunami bore acting on the triangular breakwater based on high order 3D CLSVOF-THINC/WLIC-IBM approaching”. In: *Ocean Engineering* 182 (Juni 2019), S. 645–659. DOI: 10.1016/j.oceaneng.2019.03.067.

[98] P. Yue, C. Zhou und J. J. Feng. “Spontaneous shrinkage of drops and mass conservation in phase-field simulations”. In: *Journal of Computational Physics* 223.1 (Apr. 2007), S. 1–9. DOI: 10.1016/j.jcp.2006.11.020.

[99] L. Qian, Y. Wei und F. Xiao. “Coupled THINC and level set method: A conservative interface capturing scheme with high-order surface representations”. In: *Journal of Computational Physics* 373 (Nov. 2018), S. 284–303. DOI: 10.1016/j.jcp.2018.06.074.

[100] L. Qian und Y. Wei. “Improved THINC/SW scheme for computing incompressible two-phase flows”. In: *International Journal for Numerical Methods in Fluids* 89.6 (Okt. 2018), S. 216–234. DOI: 10.1002/fld.4690.

[101] J. Assion. *Quadraturformeln und ihre Konvergenz*. Bachelorarbeit, Fakultät Mathematik, Universität Bielefeld. 2010.

[102] B. P. Vollmayr-Lee und A. D. Rutenberg. "Fast and accurate coarsening simulation with an unconditionally stable time step". In: *Physical Review E* 68.6 (Dez. 2003). DOI: 10.1103/physreve.68.066703.

[103] Y. Liu. "Hybrid Parallel Computation of OpenFOAM Solver on Multi-Core Cluster Systems". Masterarbeit. KTH Royal Institute of Technology, 2011.

[104] R. I. Issa. "Solution of the implicitly discretised fluid flow equations by operator-splitting". In: *Journal of Computational Physics* 62.1 (Jan. 1986), S. 40–65. DOI: 10.1016/0021-9991(86)90099-9.

[105] O. O. website. *www.openfoam.org*.

[106] M. M. G. Eain, V. Egan und J. Punch. "Film thickness measurements in liquid–liquid slug flow regimes". In: *International Journal of Heat and Fluid Flow* 44 (Dez. 2013), S. 515–523. DOI: 10.1016/j.ijheatfluidflow.2013.08.009.

[107] J. Jovanović, W. Zhou, E. V. Rebrov, T. Nijhuis, V. Hessel und J. C. Schouten. "Liquid–liquid slug flow: Hydrodynamics and pressure drop". In: *Chemical Engineering Science* 66.1 (Jan. 2011), S. 42–54. DOI: 10.1016/j.ces.2010.09.040.

[108] M. Sauf. *Experimentelle Untersuchung verschiedener Mikro-Phasentrenner für Flüssig-Flüssig-Pfropfenströmungen*. Bachelorarbeit, TU Dortmund, Lehrstuhl CVT. 2014.

[109] W. M. Hayner, Hrsg. *CRC Handbook of Chemistry and Physics, 97th Edition (CRC Handbook of Chemistry & Physics)*. CRC Press, 2016. ISBN: 978-1498754286. URL: https://www.amazon.com/CRC-Handbook-Chemistry-Physics-97th/dp/1498754287?SubscriptionId=AKIAIOBINVZYXZQZ2U3A&tag=chimbori05-20&linkCode=xm2&camp=2025&creative=165953&creativeASIN=1498754287.

[110] A. H. Demond und A. S. Lindner. "Estimation of interfacial tension between organic liquids and water". In: *Environmental Science & Technology* 27.12 (Nov. 1993), S. 2318–2331. DOI: 10.1021/es00048a004.

[111] K. B. Jürgen Zierep. *Grunzüge der Strömungslehre*. 7. Auflage. Teubner, 2008. ISBN: 978-3-8351-0231-6.

[112] D. J. B. Bin Zhao Jeffrey S. Moore. "Surface-Directed Liquid Flow Inside Microchannels". In: *Science* 291.5506 (Feb. 2001), S. 1023–1026. DOI: 10.1126/science.291.5506.1023.

[113] P. Jankowski und P. Garstecki. "Stable hydrophilic surface of polycarbonate". In: *Sensors and Actuators B: Chemical* 226 (Apr. 2016), S. 151–155. DOI: 10.1016/j.snb.2015.11.100.

[114] P. Jankowski, D. Ogończyk, W. Lisowski und P. Garstecki. "Polyethyleneimine coating renders polycarbonate resistant to organic solvents". In: *Lab on a Chip* 12.14 (2012), S. 2580. DOI: 10.1039/c2lc21297h.

[115] E. Kurtoğlu, A. Bilgin, M. Şeşen, B. Mısırlıoğlu, M. Yıldız, H. F. Y. Acar und A. Koşar. “Ferrofluid actuation with varying magnetic fields for micropumping applications”. In: *Microfluidics and Nanofluidics* 13.4 (Juni 2012), S. 683–694. DOI: 10.1007/s10404-012-1008-5.

[116] N. M. Bruno und C. Ciocanel. “Magnetic fluid driven flow in a capillary channel”. In: *Active and Passive Smart Structures and Integrated Systems 2010*. Hrsg. von M. N. Ghasemi-Nejhad. SPIE, März 2010. DOI: 10.1117/12.858564.

[117] A. Hatch, A. Kamholz, G. Holman, P. Yager und K. Bohringer. “A ferrofluidic magnetic micropump”. In: *Journal of Microelectromechanical Systems* 10.2 (Juni 2001), S. 215–221. DOI: 10.1109/84.925748.

[118] F. Lung-Ming, F. Wei-Ching, H. Ting-Fu und L. Chia-Yen. “A Magnetic Micropump Based on Ferrofluidic Actuation”. In: *International Journal of Automation and Smart Technology* 4.2 (Juni 2014), S. 77–82. DOI: 10.5875/ausmt.v4i2.311.

[119] B. Liu, Z. Zhang, J. Yang, J. Yang und D. Li. “A rotary ferrofluidic vane micropump with C shape baffle”. In: *Sensors and Actuators B: Chemical* 263 (Juni 2018), S. 452–458. DOI: 10.1016/j.snb.2018.02.113.

[120] C. Yamahata, F. Lacharme und M. A. Gijs. “Glass valveless micropump using electromagnetic actuation”. In: *Microelectronic Engineering* 78-79 (März 2005), S. 132–137. DOI: 10.1016/j.mee.2004.12.018.

[121] R.-J. Yang, H.-H. Hou, Y.-N. Wang und L.-M. Fu. “Micro-magnetofluidics in microfluidic systems: A review”. In: *Sensors and Actuators B: Chemical* 224 (März 2016), S. 1–15. DOI: 10.1016/j.snb.2015.10.053.

[122] C. M. S. Knebel M. Dietiker. *Ferrofluide Experimentieranleitung*. Website. Aug. 2015. URL: http://www.swissnanocube.ch/uploads/tx_rfnanoteachbox/Ferrofluid_Lehrer_031210_01.pdf.

[123] A. Einstein. “Über die von der molekularkinetischen Theorie der Wärme geforderte Bewegung von in ruhenden Flüssigkeiten suspendierten Teilchen”. In: *Annalen der Physik* 322.8 (1905), S. 549–560. DOI: 10.1002/andp.19053220806.

Abbildungsverzeichnis

Tabellenverzeichnis

Anhang A

Zusatzinformationen zur Entwicklung der numerischen Methode

A.1 Interpolationsfehler bei Verwendung von Polynomen

Am Beispiel einer einfachen Interpolation durch Polynome wird für $c = \tanh\left(\frac{x}{\varepsilon}\right)$ das Abfallen des Interpolationsfehlers für verschiedene Ordnungen m demonstriert. Die Gitterweite liegt konstant bei $\Delta x = 1$, während die Dicke der Grenzfläche durch verschiedene Werte von ε verändert wurde. Berechnet wurde jeweils eine Interpolation auf den zugehörigen Punkt $x/\varepsilon = 2,5$ bei symmetrischer Verteilung der Stützstellen um den Punkt von Interesse. Das exakte Ergebnis lautet demnach $c_{exakt} = \tanh(2,5) \approx 0,98661429815143$. Ein starker Abfall des Inteprolationsfehlers

Tabelle A.1: Interpolation von c auf zugehörigen Wert von $x/\varepsilon = 2,5$ bei Verwendung von Polynomen verschiedener Ordnung m.

m	$\varepsilon = 1$	$\lvert c_{int.} - c_{exakt}\rvert$	$\varepsilon = 2$	$\lvert c_{int.} - c_{exakt}\rvert$
1	0,979541167	0,007073131	0,98494292	0,001671378
3	0,991926097	0,005311799	0,986912461	0,000298163
5	0,98765231	0,001038012	0,986582347	3,19513E-05

mit steigender Polynomordnung m ist nur dann zu beobachten, wenn das Verhältnis $\Delta x/\varepsilon$ Werte kleiner als 1 annimmt.

A.2 Mathematische Formulierung der Stetigkeitsbedingungen

Um eine ausreichende stetige Differnezierbarkeit von $f(L)$ an den Übergangsstellen L_{inner} zu erreichen, wird eine Stetigkeit des Funktionswertes und der ersten fünf Ableitungen gefordert. Die Polynomkoeffizienten garantieren in Abhängigkeit von

α, β und L_{inner} eine Stetigkeit bis zur vierten Ableitung. Im Anschluss wird β so bestimmt, dass auch die fünfte Ableitung stetig ist. Daraus ergeben sich folgende Gleichungen zur Bestimmung der Polynomkoeffizienten a_i:

$$p(L) = a_1 \cdot L + a_3 \cdot L^3 + a_5 \cdot L^5 + a_7 \cdot L^7 + a_9 \cdot L^9 \tag{A.1}$$

$$\underbrace{\overbrace{\begin{pmatrix} L & L^3 & L^5 & L^7 & L^9 \\ 1 & 3 \cdot L^2 & 5 \cdot L^4 & 7 \cdot L^6 & 9 \cdot L^8 \\ 0 & 6 \cdot L & 20 \cdot L^3 & 42 \cdot L^5 & 72 \cdot L^7 \\ 0 & 6 & 60 \cdot L^2 & 210 \cdot L^4 & 504 \cdot L^6 \\ 0 & 0 & 120 \cdot L & 840 \cdot L^3 & 3024 \cdot L^5 \end{pmatrix}}^{A} * \begin{pmatrix} a_1 \\ a_3 \\ a_5 \\ a_7 \\ a_9 \end{pmatrix}}_{\begin{pmatrix} p(L) \\ p'(L) \\ p''(L) \\ p'''(L) \\ p^{(IV)}(L) \end{pmatrix}} = \underbrace{\begin{pmatrix} 1 - \exp(\alpha \cdot L + \beta) \\ -\alpha \cdot \exp(\alpha \cdot L + \beta) \\ -\alpha^2 \cdot \exp(\alpha \cdot L + \beta) \\ -\alpha^3 \cdot \exp(\alpha \cdot L + \beta) \\ -\alpha^4 \cdot \exp(\alpha \cdot L + \beta) \end{pmatrix}}_{\begin{pmatrix} f(L) \\ f'(L) \\ f''(L) \\ f'''(L) \\ f^{(IV)}(L) \end{pmatrix}} \tag{A.2}$$

Zu erfüllen ist Gl. A.2 an der Stelle $L = L_{inner}$. Die linke Seite der Gleichung (A) kann invertiert werden, um die Koeffizienten a_i in Abhängigkeit von α, β und L_{inner} zu bestimmen. Die Bestimmung der Inverse A^{-1} erfolgt in OpenFOAM über die Klasse SVD (Singular Value Decomposition) und ist lediglich auf $\approx 10^{-8}$ genau. Über die Fixpunktiteration $A_{n+1}^{-1} = (2 \cdot I_{5,5} - A_n^{-1} * A) * A_n^{-1}$ wird diese Genauigkeit weiter verbessert, bevor A^{-1} verwendet wird. Daraufhin kann die Übereinstimmung der fünften Ableitung durch β nach Gl. A.3 gefordert werden.

$$\begin{aligned} p^{(V)}(L) &= 120 \cdot a_5 + 2520 \cdot a_7 \cdot L^2 + 15120 \cdot a_9 \cdot L^4 \\ &= \vec{v} * \begin{pmatrix} a_5 \\ a_7 \\ a_9 \end{pmatrix} \\ &= \vec{v} * \begin{pmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} * A^{-1} * \begin{pmatrix} 1 - \exp(\alpha \cdot L + \beta) \\ \alpha \cdot \exp(\alpha \cdot L + \beta) \\ \alpha^2 \cdot \exp(\alpha \cdot L + \beta) \\ \alpha^3 \cdot \exp(\alpha \cdot L + \beta) \\ \alpha^4 \cdot \exp(\alpha \cdot L + \beta) \end{pmatrix} \\ &= \vec{v} * I_{3,5} * A^{-1} * \left[\begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} - \exp(\alpha \cdot L + \beta) \cdot \begin{pmatrix} \alpha^0 \\ \alpha^1 \\ \alpha^2 \\ \alpha^3 \\ \alpha^4 \end{pmatrix} \right] \\ &\stackrel{!}{=} f^{(V)}(L) = -\alpha^5 \cdot \exp(\alpha \cdot L + \beta) \end{aligned} \tag{A.3}$$

Das optimale $\beta_{opt.}$ lässt sich dann aus Gl. A.4 isolieren und in Abhängigkeit von α und L_{inner} berechnen.

$$\vec{v} * I_{3,5} * A^{-1} * \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} = \exp(\alpha \cdot L + \beta) \cdot \left(\vec{v} * I_{3,5} * A^{-1} * \begin{pmatrix} \alpha^0 \\ \alpha^1 \\ \alpha^2 \\ \alpha^3 \\ \alpha^4 \end{pmatrix} - \alpha^5 \right) \tag{A.4}$$

Anhang B

Implementierung der Methode in OpenFOAM

Für das in Kapitel 3 entwickelte numerische Model sind die in OpenFOAM 2.1.1[1] implementierten Komponenten nachfolgend dargestellt.[2] OpenFOAM selbst, worauf der hier dargestellte Code basiert, steht unter der GNU General Public Licence.[3] Jegliche daraus entstehende Software steht daher ebenfalls unter der entsprechenden GNU General Public Licence.

Programmcode B.1: freeSurface.C

```
1  /*
     ------------------------------------------------------------------------
2    =========                 |
3    \\      /  F ield         | OpenFOAM: The Open Source CFD
       Toolbox
4      \\    /   O peration     |
5       \\  /    A nd           | Copyright (C) 2011 OpenFOAM
       Foundation
6        \\/     M anipulation  |
7  ------------------------------------------------------------------------

8  License
9      This file is part of OpenFOAM.
10
11     OpenFOAM is free software: you can redistribute it and/
       or modify it
12     under the terms of the GNU General Public License as
       published by
13     the Free Software Foundation, either version 3 of the
       License, or
14     (at your option) any later version.
15
```

[1] Download über https://openfoam.org/download/archive/
[2] Sonderzeichen zur Darstellung ersetzt
[3] siehe https://openfoam.org/licence/

```
16      OpenFOAM is distributed in the hope that it will be
        useful, but WITHOUT
17      ANY WARRANTY; without even the implied warranty of
        MERCHANTABILITY or
18      FITNESS FOR A PARTICULAR PURPOSE. See the GNU General
        Public License
19      for more details.
20
21      You should have received a copy of the GNU General
        Public License
22      along with OpenFOAM. If not, see <http://www.gnu.org/
        licenses/>.
23
24  Application
25      freeSurface
26
27  Description
28      Transient solver for incompressible, laminar flow of 2
        Newtonian fluids (VOF).
29      Only for equal densities and equal viscosities.
30
31  \*---------------------------------------------------------------------------
        */
32
33  #include <omp.h>
34  #include "fvCFD.H"
35  #include "dynamicFvMesh.H"                                  //
        Change 1
36  #include "volPointInterpolation.H"
37  #include "RectangularMatrix.H"
38  #include "SVD.H"
39  #include "different_functions.H"
40  #include "phaseFieldExpLFvPatchScalarField.H"
41  #include "phaseFieldExpCFvPatchScalarField.H"
42  #include "nHatFvPatchVectorField.H"
43  #include "meshSearch.H"
44
45
46
47
48
49  // * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
        * * * * * * * * * //
50
51  int main(int argc, char *argv[])
52  {
53      #include "setRootCase.H"
54      #include "createTime.H"
```

```
55      #include "createDynamicFvMesh.H"                          //
    Change 2
56      #include "createFields.H"
57      #include "initContinuityErrs.H"
58      #include "readTimeControls.H"
59      #include "CourantNo.H"
60      #include "setInitialDeltaT.H"
61
62
63      // * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
        * * * * * * * * * //
64
65
66
67      Info<< "\nStarting time loop\n" << endl;
68
69      //lduMatrix::debug = 0;
70
71      #include "readPISOControls.H"
72
73
74
75
76          Info<< "Time = " << runTime.timeName() << nl << endl
    ;
77
78          #include "resetReferences.H"
79          const double nGetLfromC = pisoDict.lookupOrDefault<
    double>("nGetLfromC", 0);
80          const double nDiffusionLoop = pisoDict.
    lookupOrDefault<double>("nDiffusionLoop", 0);
81          const double GetLfromC_accuracy = pisoDict.
    lookupOrDefault<double>("GetLfromC_accuracy", 1.0);
82          const double MaxInverseR = pisoDict.lookupOrDefault<
    double>("MaxInverseR", 0);
83          cout << "MaxInverseR = " << MaxInverseR << nl;
84          cout << "nGetLfromC = " << nGetLfromC << "    with
    GetLfromC_accuracy = " << GetLfromC_accuracy << nl;
85          const dictionary& expParameterDict = mesh.
    solutionDict().subDict("expParameter");
86          double alpha_profile(expParameterDict.
    lookupOrDefault<double>("alpha",0));
87          double beta_profile(expParameterDict.lookupOrDefault
    <double>("beta",0));
88          double L0_inner_profile(expParameterDict.
    lookupOrDefault<double>("L0_inner",0));
89          double L0_outer_profile(expParameterDict.
    lookupOrDefault<double>("L0_outer",0));
```

```
90          const double Kp = pisoDict.lookupOrDefault<double>("
    Kp", 0);
91          const double Td = pisoDict.lookupOrDefault<double>("
    Td", 0);
92
93
94          profile CfromL(alpha_profile, beta_profile,
    L0_inner_profile, L0_outer_profile);
                //Erschafft Objekt der Klasse profile (
    Constructor mit Argumenten)
95          meshSearch searchEngine(mesh);
96
97
98
99          if(runTime.time().value() < 0.000000001)
100         {
101             levelset = -999999999;
102             cout << "insert slug" << nl;
103
104             forAll(cells, cellI)
105             {
106
107                 if(cellCentre[cellI].component(0) < mid1.
    value().component(0))
108                 {
109                     L_ref[cellI] = (R_slug.value() - mag(
    cellCentre[cellI] - mid1.value()))/epsilon.value();
110                     C_volaverage_ref[cellI] = CfromL.IntL(0,
     L_ref[cellI]);
111                 }
112                 else
113                 {
114                     if(cellCentre[cellI].component(0) < mid2
    .value().component(0))
115                     {
116                         L_ref[cellI] = (R_slug.value() -
    cellCentre[cellI].component(1))/epsilon.value();
117                         C_volaverage_ref[cellI] = CfromL.
    IntL(0, L_ref[cellI]);
118                     }
119                     else
120                     {
121                         if(cellCentre[cellI].component(0) <
    mid2.value().component(0) + 2.0*R_slug.value())
122                         {
123                             L_ref[cellI] = (R_slug.value() -
     mag(cellCentre[cellI] - mid2.value()))/epsilon.value();
124                             C_volaverage_ref[cellI] = CfromL
```

```
    .IntL(0, L_ref[cellI]);
125                         }
126                     }
127                 }
128             }
129
130             #include "pureLevelsetLists.H"
131             #include "directInterface.H"
132             levelset.correctBoundaryConditions();
133             gradL = fvc::grad(levelset);
134             #include "harmoniceBC.H"
135             levelset.correctBoundaryConditions();
136             gradL = fvc::grad(levelset);
137             #include "harmoniceBC.H"
138             levelset.correctBoundaryConditions();
139             gradL = fvc::grad(levelset);
140             #include "harmoniceBC.H"
141             levelset.correctBoundaryConditions();
142             gradL = fvc::grad(levelset);
143             #include "harmoniceBC.H"
144             levelset.correctBoundaryConditions();
145             gradL = fvc::grad(levelset);
146             #include "harmoniceBC.H"
147             levelset.correctBoundaryConditions();
148             gradL = fvc::grad(levelset);
149             #include "harmoniceBC.H"
150             levelset.correctBoundaryConditions();
151             gradL = fvc::grad(levelset);
152             #include "harmoniceBC.H"
153             levelset.correctBoundaryConditions();
154             gradL = fvc::grad(levelset);
155             #include "harmoniceBC.H"
156             levelset.correctBoundaryConditions();
157             gradL = fvc::grad(levelset);
158             #include "harmoniceBC.H"
159             lapL = fvc::laplacian(levelset);
160             levelset.correctBoundaryConditions();
161             gradL = fvc::grad(levelset);
162
163             cout << "finished␣setup␣L.␣start␣
    Volumeintegration." << nl;
164
165
166             for(int cellI = 0; cellI < cells.size(); cellI
    ++)
167             {
168                 if(pureLevelset[cellI])
169                 {
```

```
170                     if(!n_face_BoolList[cellI])
171                     {
172                         #include "updateNormalVectors.H"
            //Vorsicht! "updateNormalVectors.H" noch nicht parallel aufrufbar!
173                     }
174                     #include "initiateC_volaverage_parallel.H"
175                 }
176             }
177         }
178
179
180
181
182
183     cout << "Start" << nl;
184
185     U.correctBoundaryConditions();
186     phi = fvc::interpolate(U) & mesh.Sf();
187
188     levelset.correctBoundaryConditions();
189
190     while (runTime.loop())
191     {
192
193         #include "CourantNo.H"
194         if (adjustTimeStep)
195         {
196             scalar maxDeltaTFact = maxCo/(CoNum + SMALL);
197             scalar deltaTFact = min(min(maxDeltaTFact, 1.0 + 0.1*maxDeltaTFact), 1.2);
198             runTime.setDeltaT(min(deltaTFact*runTime.deltaTValue(), maxDeltaT));
199         }
200         Info<< "Zeit = " << runTime.timeName() << "    current dt = " << runTime.deltaTValue() << endl;
201         dimensionedScalar dt(runTime.deltaT());
202         Info<< "Zwischenzeit: " << runTime.elapsedClockTime() << " s" << nl;
203
204
205         //Runge-Kutta-Verfahren vierter Ordnung setzten voraus, dass die fuenfte Ableitung existiert und stetig ist. mit a1 a3 a5 a7 a9 und beta stimmen bei L_outer p, p', p'', p''', p'''' und p''''' ueberein.
206         C_volaverage.correctBoundaryConditions();
207         C_volaverage_old = C_volaverage;
```

```
208         C_face = fvc::interpolate(C_volaverage);
209         #include "getLfromC.H"
210         #include "newGaussFaceIntegrate_parallel.H"
211         volScalarField k1( -dt*fvc::div(phi*(C_face) ));
212 
213         C_volaverage = C_volaverage_old + k1/2;
214         C_volaverage.correctBoundaryConditions();
215         C_face = fvc::interpolate(C_volaverage);
216         #include "getLfromC.H"
217         #include "newGaussFaceIntegrate_parallel.H"
218         volScalarField k2( -dt*fvc::div(phi*(C_face) ));
219 
220         C_volaverage = C_volaverage_old + k2/2;
221         C_volaverage.correctBoundaryConditions();
222         C_face = fvc::interpolate(C_volaverage);
223         #include "getLfromC.H"
224         #include "newGaussFaceIntegrate_parallel.H"
225         volScalarField k3( -dt*fvc::div(phi*(C_face) ));
226 
227         C_volaverage = C_volaverage_old + k3;
228         C_volaverage.correctBoundaryConditions();
229         C_face = fvc::interpolate(C_volaverage);
230         #include "getLfromC.H"
231         #include "newGaussFaceIntegrate_parallel.H"
232         volScalarField k4( -dt*fvc::div(phi*(C_face) ));
233 
234         C_volaverage = C_volaverage_old + (k1 +2*k2 +2*k3 +
    k4)/6;
235         correction.internalField() = (k1 +2*k2 +2*k3 +k4)/6;
236 
237         Info<< "Zwischenzeit: " << runTime.elapsedClockTime
    () << " s" << nl;
238 
239         cout << nl << "start compression" << nl;
240         for (int k=1; k <= nDiffusionLoop; k++)
241         {
242             #include "getLfromC.H"
243             #include "calculateDiffusion_explicit.H"
244             cout << "compressions step " << k << nl;
245         }
246 
247         Info<< "Zwischenzeit: " << runTime.elapsedClockTime
    () << " s" << nl;
248 
249         cout << "Max(ChemPot): " << max(ChemPot.
    internalField()) << nl;
250         cout << "Min(ChemPot): " << min(ChemPot.
    internalField()) << nl;
```

```
251         cout << "Max(ChemPot) - Min(ChemPot) = " << (max(
    ChemPot.internalField()) - min(ChemPot.internalField()))
    << nl;
252
253         #include "pureLevelsetLists.H"
254         #include "getLfromC.H"
255         #include "calculateFluxes.H"
256         #include "PISO_Loop_pred.H"
257
258         #include "continuityErrs.H"
259         mesh.update();
260
261
262         runTime.write();
263
264         Info<< "ExecutionTime = " << runTime.elapsedCpuTime
    () << " s"
265             << "  ClockTime = " << runTime.elapsedClockTime
    () << " s"
266             << nl << endl;
267         cout << nl;
268
269
270     }            //Ende while (runTime.loop())
271
272     Info<< "End\n" << endl;
273
274     return 0;
275 }
276
277
278 //
    ************************************************************************
     //
```

Programmcode B.2: PISO_Loop_pred.H

```
1 //cout << "start PISO" << nl;
2 // --- PISO loop
3 U.correctBoundaryConditions();
4
5
6 rho = rho1 + (rho2-rho1) * (1.0 + min(1.0,max(-1.0,
    C_volaverage)))/2;
7 nu = nu1 + (nu2-nu1) * (1.0 + min(1.0,max(-1.0,C_volaverage)
    ))/2;
8
9 fvVectorMatrix UEqn( fvm::ddt(rho, U) + fvm::div(fvc::
    interpolate(rho)*phi, U) - fvm::laplacian(nu, U) );
```

```
10 solve(UEqn == fvc::reconstruct( (sigma*pressureJump - fvc::
       snGrad(p))*mesh.magSf()) );
11
12
13 for (int corr=0; corr<nCorr; corr++)
14 {
15
16     volScalarField rAU(1.0/UEqn.A());
17     surfaceScalarField rAUface(fvc::interpolate(rAU));
18
19     U = rAU*(UEqn.H());          //vorlaeufiges U, um phi zu
       berechnen.
20     surfaceScalarField phiU( (fvc::interpolate(U) & mesh.Sf
       ()));// + fvc::ddtPhiCorr(rAU, rho, U, phi));
21     phi = phiU + rAUface*pressureJump*sigma*mesh.magSf();
22
23     for (int nonOrth=0; nonOrth<=nNonOrthCorr; nonOrth++)
24     {
25         fvScalarMatrix pEqn
26         (
27             fvm::laplacian(rAUface, p) == fvc::div(phi)
28         );
29
30         pEqn.setReference(pRefCell, pRefValue);
31         pEqn.solve();
32
33         if (nonOrth == nNonOrthCorr)
34         {
35             phi -= rAUface*fvc::snGrad(p)*mesh.magSf();
36         }
37     }   //Ende for nNonOrthCorr
38
39
40     U += rAU*fvc::reconstruct( (phi-phiU)/rAUface  );
41     U.correctBoundaryConditions();
42
43 }       //Ende PISO-Loop
44
45 //cout << "end PISO" << nl;
46 cout << "Max(p) = " << max(p.internalField()) << "  Min(p) =
       " << min(p.internalField()) << nl;
47 cout << "Max(mag(U)) = " << max(mag(U.internalField())) <<
       nl;
```

Programmcode B.3: calculateC_test2_parallel.H

```
1     scalar local_C_test(0.0);
2     scalar error_edge_anal(0.0);
3     scalar error_edge_num(0.0);
```

```
4       scalar error_face_num(0.0);
5       scalar error_face_anal(0.0);
6       scalar error_cell_num(0.0);
7       scalar error_cell_anal(0.0);
8
9       scalar local_volume(volume[cellI]);
10
11      scalar face_Zwischenergebnis(0.0);
12      scalar local_L_value(L_ref[cellI]);
13      vector local_gradL_value(gradL_ref[cellI]);
14      scalar df1dL1_local(CfromL.IntL(-1.0, local_L_value));
15      scalar df2dL2_local(mag(  CfromL.IntL(-2.0, max( CfromL.
    L_df2dL2_max ,mag(local_L_value)))));                    //
    Liege mit dem L-Wert ausserhalb von max(df2dL2)
16      Ableitung_ref[cellI] = df1dL1_local;
17      //Man koennte die Ableitung d(C_test)/dL exakter
    ausrechnen, aber die quadr. Konvergenzgeschw. des Newton-
    Verfahren, erreicht man eh nicht,
18      //weil Lx, Ly und Lz als konstant angesehen werden, was
    sie ja nicht sind.
19
20
21
22              if(     mag(df2dL2_local*(local_gradL_value &
     local_gradL_value)*epsilon.value()*epsilon.value()) <
    0.000000015625)
23              //Wenn der Gradient sehr klein ist, kann ich
     direkt die numeirsche Integration anwenden.
24              //Ausserdem: caseStudy3 hat gezeigt, dass
    eine Volumenintegration kein weiteres Abfallen von U_Fehl
     bringt, wenn L_outer groesser 9 gewaehlt wird.
25              //Beziehungsweise, wenn die Abschaetzung der
     zweiten Ableitung von c (d2c/dx2  = exp(alpha*L + beta)
    *(alpha*magGradL)2) multipliziert mit der Gitterweite2
    kleiner als 4e-8 ist.
26              //Jetzt gilt nach dem Stabkrit, dass alpha*
    dx/eps <=1,6 gilt und demnach nutze ich exp(alpha*L +
    beta)*(1,6*magGradL*eps)2 < 4e-8
27              {
28                  error_cell_anal = 10000000000000000;
29                  //Fehler der (semi-)analytischen Int.
    mag hoeher liegen. Hier wird angenommen, dass 10^17
    ausreichend ist, damit schlussendlich die num. cell-Int.
    verwendet wird.
30              }
31              else
32              {
33
```

```
34                      const labelList ListeDerPunkte(
    cellPoints[cellI]);  //Fuer jeden Punkt schon einmal L-
    Wert und F3-Wert berechnen
35                      for( int pointI = 0; pointI <
    ListeDerPunkte.size(); pointI++)
36                      {
37                          label point_labelI(ListeDerPunkte[
    pointI]);
38                          L_point[point_labelI] = (
    local_gradL_value & (points[point_labelI] - cellCentre[
    cellI])) + local_L_value;
39                          F3_point[point_labelI] = CfromL.IntL
    (3.0, L_point[point_labelI]);
40                          //Die F3-Werte berechne ich schon
    einmal auf Vorrat. Es sollte wesenltich seltener
    vorkommen, dass ein Wert nicht gebraucht wird,
41                          //weil eigenltich stets mag(gradL) =
     1/eps herrschen sollte. Dann wird F3 auf einem der faces
     immer gebraucht.
42                      }
43
44                      const labelList ListeDerEdges(
    ZelleBestehtAus[cellI]);      //Fuer jedes Edge der Zelle
     das zugehoerige gamma berechnen
45                      for( int edgeI = 0; edgeI <
    ListeDerEdges.size(); edgeI++)
46                      {
47                          label edge_labelI(ListeDerEdges[
    edgeI]);
48                          scalar local_gamma(local_gradL_value
     & n_line_VectorList[edge_labelI]);
49                          if(local_gamma == 0)    //Wenn der
    Line-Gradient echt gleich 0 ist, kann ich direkt die
    numeirsche Integration anwenden.
50                          {
51                              error_edge_anal =
    10000000000000000;
52                              //Fehler der analytischen Int.
    mag hoeher liegen. Hier wird angenommen, dass 10^17
    ausreichend ist, damit schlussendlich die num. Int.
    verwendet wird.
53                          }
54                          else
55                          {
56                              error_edge_anal = eps*CfromL.
    F3_max/mag(local_gamma);
57                          }
58                          error_edge_num = (eps*length_edge[
```

```
    edge_labelI] + mag(df2dL2_local/4320.0 *pow(local_gamma
    ,4) *length_edge_pow5[edge_labelI]));
59                      //Fehler des numerischen Verfahrens
    durch Gauss + Auswertungsfehler, da IntL nur auf eps
    genau berechnet wird und dann noch mit der Edgelaenge
    multipliziert wird.
60
61
62
63
64                      if(error_edge_anal < error_edge_num)
65                      {       //analytischer Fall
66                          IntegratedEdge[edge_labelI] = (
    F3_point[edges[edge_labelI].end()] - F3_point[edges[
    edge_labelI].start()])/local_gamma;
67                          error_estimate_edge[edge_labelI]
     = error_edge_anal;
68                      }
69                      else
70                      {       //numerischer Fall
71                          if(mag(local_gamma) < eps)
          //weitere Beschleunigung, da fuer gamma kleiner
    eps auch die Mittelpunktsregel ausreichend ist  und man
    eine FKT-Auswertung einspart
72                          {
73                              scalar L_mid( (L_point[edges
    [edge_labelI].start()] + L_point[edges[edge_labelI].end()
    ])/2);
74                              IntegratedEdge[edge_labelI]
    = (CfromL.IntL(2.0, L_mid)*length_edge[edge_labelI]);

75                              error_estimate_edge[
    edge_labelI] = eps*length_edge[edge_labelI];
76                              //Fehler des numerischen
    Verfahrens ~0 + Auswertungsfehler, da IntL nur auf eps
    genau berechnet wird und dann noch mit der Edgelaenge
    multipliziert wird.
77
78                          }
79                          else
80                          {
81                              scalar L_unten(L_point[edges
    [edge_labelI].start()]*gauss_weight1 + L_point[edges[
    edge_labelI].end()]*gauss_weight2);
82                              scalar L_oben(L_point[edges[
    edge_labelI].start()]*gauss_weight2 + L_point[edges[
    edge_labelI].end()]*gauss_weight1);
83
```

```
84                                      IntegratedEdge[edge_labelI]
    = (CfromL.IntL(2.0, L_unten) + CfromL.IntL(2.0, L_oben))*
    length_edge[edge_labelI]/2;
85                                      error_estimate_edge[
    edge_labelI] = error_edge_num;
86                                      edgeGauss_ref[cellI] = 1;
87                                  }
88                              } // end if(error_edge_anal <
    error_edge_num)
89                          } // end for( int edgeI = 0; edgeI <
    ListeDerEdges.size(); edgeI++)
90
91
92                          const labelList ListeDerFaces(cells[
    cellI]);
93                          List<vector>& n_face_VectorSubList =
    n_face_VectorList[cellI];
94                          for(int faceI = 0; faceI < ListeDerFaces
    .size(); faceI++)
95                          {
96                              label face_labelI(ListeDerFaces[
    faceI]);
97                              scalar local_alpha(local_gradL_value
     & n_face_VectorSubList[faceI]);
98                              scalar magGradLface( (
    local_gradL_value & local_gradL_value) - local_alpha*
    local_alpha);       //Vorsicht: MagGradLface enthaelt
    mag(gradL)2
99
100                             if(magGradLface == 0) //Wenn der
    Face-Gradient echt gleich 0 ist, kann ich direkt die
    numeirsche Integration anwenden.
101                             {
102                                 error_face_anal =
    10000000000000000;
103                                 //Fehler der "(semi-)
    analytischen" Int. mag hoeher liegen. Hier wird
    angenommen, dass 10^17 ausreichend ist, damit
    schlussendlich die num. face-Int. verwendet wird.
104                             }
105                             else
106                             {
107                                 face_Zwischenergebnis = 0;
108                                 const labelList ListeDerEdges2(
    FaceBestehtAus[face_labelI]);
109                                 List<vector>
    n_edge_VectorSubList(n_edge_VectorList[face_labelI]);
110
```

```
111                             for( int edgeI = 0; edgeI <
    ListeDerEdges2.size(); edgeI++)
112                             {
113                                 label edge_labelI(
    ListeDerEdges2[edgeI]);
114                                 scalar local_beta(
    local_gradL_value & n_edge_VectorSubList[edgeI]);
115                                 scalar preFactor(local_beta/
    magGradLface);                //Oben wurde ausgeschlossen,
    dass magGradLface = 0 ist
116
117                                 face_Zwischenergebnis +=
    IntegratedEdge[edge_labelI]*preFactor;
118                                 error_face_anal = max(
    error_face_anal, max(mag(error_estimate_edge[edge_labelI
    ]*preFactor), mag(eps*face_Zwischenergebnis)));
119                             }    //end for( int edgeI = 0;
    edgeI < ListeDerEdges2.size(); edgeI++)
120                         }    //end if(magGradLface == 0)
121
122
123                         scalar magSf(mag(Sf_ref[face_labelI
    ]));
124                         error_face_num = (  magSf*(magSf*
    df1dL1_local/24*magGradLface  + eps));
125                         // Formel: A/6*df/dL_max*(Lx2 + Ly2)
    *A/4, wobei das letzte A/4 nur ein verstecktes *dx2 ist
    und dabei ein Quadrat als Flaeche angenommen wurde.
126                         // Fehler des numerischen Verfahrens
     + Auswertungsfehler, da IntL nur auf eps genau berechnet
     wird und dann noch mit magSf multipliziert wird.
127
128                         scalar preFactor(local_alpha / (
    local_gradL_value & local_gradL_value));        //Ganz
    oben wurde ausgeschlossen, dass (local_gradL_value &
    local_gradL_value) = 0 ist
129
130                         if(error_face_anal < error_face_num)
131                         {       //analytischer Fall
132                             local_C_test += preFactor*
    face_Zwischenergebnis;
133                             error_cell_anal = max(
    error_cell_anal, max(mag(error_face_anal*preFactor), mag(
    eps*local_C_test)));
134                         }
135                         else
136                         {       //numerischer Fall
137                             local_C_test += magSf*preFactor
```

```
    *(CfromL.IntL(1.0, local_L_value + (local_gradL_value & (
    faceCentre[face_labelI] - cellCentre[cellI]))));
138                             error_cell_anal = max(
    error_cell_anal, max(mag(error_face_num*preFactor), mag(
    eps*local_C_test)));
139                             faceMidPoint_ref[cellI] = 1;
140                         }
141                     }    //end for(int faceI = 0; faceI <
    ListeDerFaces.size(); faceI++)
142
143
144                 } //end if(    mag(df2dL2_local*(
    local_gradL_value & local_gradL_value)*epsilon.value()*
    epsilon.value()) < 0.000000015625)
145
146
147                 error_cell_num = ( Foam::pow(local_volume,
    5.0/3.0)/24.0 *(local_gradL_value & local_gradL_value)*
    df2dL2_local + eps*local_volume) ;
148
149
150                 //Hier kommen die einzigen Unterschiede zu
    initiateC_volaverage.H
151                 if(error_cell_anal > error_cell_num)
152                 {       //numerischer Fall
153
154                     local_C_test = (CfromL.IntL(0.0,
    local_L_value) - C_volaverage_ref[cellI]);
155                     error_estimate_cell_ref[cellI] =
    error_cell_num/local_volume;
156                     cellMidPoint_ref[cellI] = 1;
157                     Ly_ref[cellI] = 0;
158                     #pragma omp critical
159                     {
160                         pureLevelset[cellI] = false;
                      //Aus irgendeinem Grund muss das hier
    critical sein. Ich verstehe es nicht.
161                     }
162                 }
163                 else
164                 {       //analytischer Fall
165
166                     local_C_test = (local_C_test/
    local_volume - C_volaverage_ref[cellI]);
167                     error_estimate_cell_ref[cellI] =
    error_cell_anal/local_volume;
168                 }
169                 C_test_ref[cellI] = local_C_test;
```

```
170
171
```

Programmcode B.4: calculateCurvature.H

```
1 #include "harmoniceBC.H"
2
3 levelset.correctBoundaryConditions();
4 gradL = fvc::grad(levelset);
5 C_volaverage.correctBoundaryConditions();
6 gradC = fvc::grad(C_volaverage);
7
8 #pragma omp parallel for schedule(static)
9 for(int cellI = 0; cellI < cells.size(); cellI++)
10 {
11     if(  pureLevelset[cellI] )
12     {
13         nHat_center_ref[cellI] = gradL_ref[cellI] / (mag(
    gradL_ref[cellI]) + numStab1.value());
14     }
15     else
16     {
17         nHat_center_ref[cellI] = gradC_ref[cellI] / (mag(
    gradC_ref[cellI]) + numStab1.value());
18     }
19 }
20
21 nHat_center.correctBoundaryConditions();
22
23 nHat_face = fvc::interpolate(nHat_center);
24 kappa = -fvc::div( nHat_face & mesh.Sf() );
```

Programmcode B.5: calculateDiffusion_explicit.H

```
1 #include "calculateCurvature.H"
2
3 lapL = fvc::laplacian(levelset);
4 lapC = fvc::laplacian(C_volaverage) + kappa*mag(gradC);
5
6 kontrolle = 0;
7
8 #pragma omp parallel for schedule(guided)
9 for(int cellI = 0; cellI < cells.size(); cellI++)
10 {
11
12     if (mag(C_volaverage_ref[cellI]) < CfromL.p0_fWert_outer
    )
13     {
14         if(pureLevelset[cellI])
15         {
```

```
16              lapC_ref[cellI] =    Ly_ref[cellI]*((gradL_ref[
    cellI] & gradL_ref[cellI])*CfromL.IntL(-2,L_ref[cellI])
    + (lapL_ref[cellI] + kappa_ref[cellI]*mag(gradL_ref[cellI
    ]))*CfromL.IntL(-1,L_ref[cellI])) + (1-Ly_ref[cellI])*
    lapC_ref[cellI];
17              gradC_ref[cellI] = gradL_ref[cellI]*CfromL.IntL
    (-1,L_ref[cellI]);
18          }
19          if (mag(C_volaverage_ref[cellI]) < CfromL.
    p0_fWert_inner)
20          {
21              kontrolle_ref[cellI] = 2;
22          }
23          else
24          {
25              kontrolle_ref[cellI] = 1;
26          }
27          ChemPot_ref[cellI] = CfromL.IntL(-2,L_ref[cellI])/
    epsilon.value() - lapC_ref[cellI]*epsilon.value();

28      }
29      else
30      {
31          if (mag(C_volaverage_ref[cellI]) < 1)
32          {
33              ChemPot_ref[cellI] = CfromL.ChemPot_outer(
    C_volaverage_ref[cellI])/epsilon.value()  - lapC_ref[
    cellI]*epsilon.value();
34          }
35          else                                //Ausserhalb des
    eigenltichen Bereiches
36          {
37              ChemPot_ref[cellI] = sign((C_volaverage_ref[
    cellI]))*(CfromL.Psi_erste*(mag(C_volaverage_ref[cellI])
    -1) + CfromL.Psi_zweite*pow(mag(C_volaverage_ref[cellI])
    -1,2))/epsilon.value()  - lapC_ref[cellI]*epsilon.value()
    ;
38          }
39      }
40 }
41 ChemPot.correctBoundaryConditions();
42
43
44
45 //cout << "Max(ChemPot): " << max(ChemPot.internalField())
    << nl;
46 //cout << "Min(ChemPot): " << min(ChemPot.internalField())
    << nl;
```

```
47 //cout << "Max(ChemPot) - Min(ChemPot) = " << (max(ChemPot.
       internalField()) - min(ChemPot.internalField())) << nl;
48 lapChemPot = fvc::laplacian(D,ChemPot);
49
50 C_volaverage += fvc::laplacian(D,ChemPot)*dt;
```

Programmcode B.6: calculateFluxes.H

```
1
2 #include "calculateCurvature.H"
3
4 U_Fehl *= 0;
5 pressureJump *= 0;
6
7 #pragma omp parallel for schedule(guided)
8 for(int faceI = 0; faceI < owner.size(); faceI++)
9 {
10     if ( (L_ref[owner[faceI]]*L_ref[neighbour[faceI]]) < 0)
11     {
12         scalar e(kappa_ref[owner[faceI]]);
13         scalar f(kappa_ref[neighbour[faceI]]-e);
               //Lineare Interpolation von kappa auf genau die
       Stelle, an der L = 0 ist.
14
15         scalar a(L_ref[owner[faceI]]);
16         scalar b(L_ref[neighbour[faceI]]-a);
17
18         scalar x_hat(-a/b);
19
20         pressureJump_ref[faceI] = (e + f*x_hat)*
       deltaCoeffs_ref[faceI]*sign(L_ref[neighbour[faceI]]);
21
22         //vector n_interface((gradL_ref[owner[faceI]] + (
       gradL_ref[neighbour[faceI]]-gradL_ref[owner[faceI]])*
       x_hat));
23         //n_interface /= (mag(n_interface) + numStab1.value
       ());
24         //vector U_interface((U_ref[owner[faceI]] + (U_ref[
       neighbour[faceI]]-U_ref[owner[faceI]])*x_hat));
25         //U_interface -= U0.value()*e_x;
26         //U_Fehl_ref[owner[faceI]] = max(        mag(
       U_interface & n_interface), U_Fehl_ref[owner[faceI]] );
27         //U_Fehl_ref[neighbour[faceI]] = max(U_Fehl_ref[
       owner[faceI]], U_Fehl_ref[neighbour[faceI]]);
28     }
29 }
30
31 cout << "max(mag(U_Fehl)) = " << max(mag(U_Fehl.
       internalField())) << " max(mag(U)) = " << max(mag(U.
```

```
    internalField())) << nl;
```

Programmcode B.7: createFields.H

```
1     Info<< "Reading transportProperties\n" << endl;
2
3     IOdictionary transportProperties
4     (
5         IOobject
6         (
7             "transportProperties",
8             runTime.constant(),
9             mesh,
10            IOobject::MUST_READ_IF_MODIFIED,
11            IOobject::NO_WRITE
12        )
13    );
14    dimensionedScalar sigma
15    (
16        transportProperties.lookup("sigma")
17    );
18    dimensionedScalar U0
19    (
20        transportProperties.lookup("U0")
21    );
22    dimensionedScalar epsilon
23    (
24        transportProperties.lookup("epsilon")
25    );
26    dimensionedScalar D
27    (
28        transportProperties.lookup("D")
29    );
30    dimensionedScalar R_slug
31    (
32        transportProperties.lookup("R_slug")
33    );
34    dimensionedVector mid1
35    (
36        transportProperties.lookup("mid1")
37    );
38    dimensionedVector mid2
39    (
40        transportProperties.lookup("mid2")
41    );
42    dimensionedScalar numStab1
43    (
44        transportProperties.lookup("numStab1")
45    );
```

```
46      dimensionedScalar numStab2
47      (
48          transportProperties.lookup("numStab2")
49      );
50      dimensionedScalar rho1
51      (
52          transportProperties.lookup("rho1")
53      );
54      dimensionedScalar rho2
55      (
56          transportProperties.lookup("rho2")
57      );
58      dimensionedScalar nu1
59      (
60          transportProperties.lookup("nu1")
61      );
62      dimensionedScalar nu2
63      (
64          transportProperties.lookup("nu2")
65      );
66
67
68
69
70      Info<< "Reading␣field␣levelset\n" << endl;
71      volScalarField levelset
72      (
73          IOobject
74          (
75              "levelset",
76              runTime.timeName(),
77              mesh,
78              IOobject::MUST_READ,
79              IOobject::AUTO_WRITE
80          ),
81          mesh
82      );
83
84      volScalarField cellMidPoint
85      (
86          IOobject
87          (
88              "cellMidPoint",
89              runTime.timeName(),
90              mesh,
91              IOobject::MUST_READ,
92              IOobject::AUTO_WRITE
93          ),
```

```
94          mesh
95      );
96      volScalarField faceMidPoint
97      (
98          IOobject
99          (
100             "faceMidPoint",
101             runTime.timeName(),
102             mesh,
103             IOobject::MUST_READ,
104             IOobject::AUTO_WRITE
105         ),
106         mesh
107     );
108     volScalarField edgeGauss
109     (
110         IOobject
111         (
112             "edgeGauss",
113             runTime.timeName(),
114             mesh,
115             IOobject::MUST_READ,
116             IOobject::AUTO_WRITE
117         ),
118         mesh
119     );
120
121     volScalarField error_estimate_cell
122     (
123         IOobject
124         (
125             "error_estimate_cell",
126             runTime.timeName(),
127             mesh,
128             IOobject::MUST_READ,
129             IOobject::AUTO_WRITE
130         ),
131         mesh
132     );
133
134
135     Info<< "Reading field C_volaverage\n" << endl;
136     volScalarField C_volaverage
137     (
138         IOobject
139         (
140             "C_volaverage",
141             runTime.timeName(),
```

```
142             mesh,
143             IOobject::MUST_READ,
144             IOobject::AUTO_WRITE
145         ),
146         mesh
147     );
148
149     Info<< "Reading field correction\n" << endl;
150     volScalarField correction
151     (
152         IOobject
153         (
154             "correction",
155             runTime.timeName(),
156             mesh,
157             IOobject::MUST_READ,
158             IOobject::AUTO_WRITE
159         ),
160         mesh
161     );
162
163     Info<< "Reading field kappa\n" << endl;
164     volScalarField kappa
165     (
166         IOobject
167         (
168             "kappa",
169             runTime.timeName(),
170             mesh,
171             IOobject::MUST_READ,
172             IOobject::AUTO_WRITE
173         ),
174         mesh
175     );
176
177
178     Info<< "Reading field inverseR\n" << endl;
179     volScalarField inverseR
180     (
181         IOobject
182         (
183             "inverseR",
184             runTime.timeName(),
185             mesh,
186             IOobject::MUST_READ,
187             IOobject::AUTO_WRITE
188         ),
189         mesh
```

```
190     );
191
192
193     volScalarField kontrolle
194     (
195         IOobject
196         (
197             "kontrolle",
198             runTime.timeName(),
199             mesh,
200             IOobject::MUST_READ,
201             IOobject::AUTO_WRITE
202         ),
203         mesh
204     );
205
206     Info<< "Reading field ChemPot\n" << endl;
207     volScalarField ChemPot
208     (
209         IOobject
210         (
211             "ChemPot",
212             runTime.timeName(),
213             mesh,
214             IOobject::MUST_READ,
215             IOobject::AUTO_WRITE
216         ),
217         mesh
218     );
219
220     Info<< "Reading field lapChemPot\n" << endl;
221     volScalarField lapChemPot
222     (
223         IOobject
224         (
225             "lapChemPot",
226             runTime.timeName(),
227             mesh,
228             IOobject::MUST_READ,
229             IOobject::AUTO_WRITE
230         ),
231         mesh
232     );
233     Info<< "Reading field lapC\n" << endl;
234     volScalarField lapC
235     (
236         IOobject
237         (
```

```
238             "lapC",
239             runTime.timeName(),
240             mesh,
241             IOobject::MUST_READ,
242             IOobject::AUTO_WRITE
243         ),
244         mesh
245     );
246     Info<< "Reading field lapL\n" << endl;
247     volScalarField lapL
248     (
249         IOobject
250         (
251             "lapL",
252             runTime.timeName(),
253             mesh,
254             IOobject::MUST_READ,
255             IOobject::AUTO_WRITE
256         ),
257         mesh
258     );
259
260
261     Info<< "Reading field C_test\n" << endl;
262     volScalarField C_test
263     (
264         IOobject
265         (
266             "C_test",
267             runTime.timeName(),
268             mesh,
269             IOobject::MUST_READ,
270             IOobject::AUTO_WRITE
271         ),
272         mesh
273     );
274     Info<< "Reading field Ableitung\n" << endl;
275     volScalarField Ableitung
276     (
277         IOobject
278         (
279             "Ableitung",
280             runTime.timeName(),
281             mesh,
282             IOobject::MUST_READ,
283             IOobject::AUTO_WRITE
284         ),
285         mesh
```

```
286     );
287
288
289     Info<< "Reading␣field␣p\n" << endl;
290     volScalarField p
291     (
292         IOobject
293         (
294             "p",
295             runTime.timeName(),
296             mesh,
297             IOobject::MUST_READ,
298             IOobject::AUTO_WRITE
299         ),
300         mesh
301     );
302
303     Info<< "Reading␣field␣visualDirectInterFace\n" << endl;
304     volScalarField visualDirectInterFace
305     (
306         IOobject
307         (
308             "visualDirectInterFace",
309             runTime.timeName(),
310             mesh,
311             IOobject::MUST_READ,
312             IOobject::AUTO_WRITE
313         ),
314         mesh
315     );
316
317     Info<< "Reading␣field␣Ly\n" << endl;
318     volScalarField Ly
319     (
320         IOobject
321         (
322             "Ly",
323             runTime.timeName(),
324             mesh,
325             IOobject::MUST_READ,
326             IOobject::AUTO_WRITE
327         ),
328         mesh
329     );
330
331
332     Info<< "Reading␣field␣U\n" << endl;
333     volVectorField U
```

```
334     (
335         IOobject
336         (
337             "U",
338             runTime.timeName(),
339             mesh,
340             IOobject::MUST_READ,
341             IOobject::AUTO_WRITE
342         ),
343         mesh
344     );
345 
346     Info<< "Reading field U_Fehl\n" << endl;
347     volScalarField U_Fehl
348     (
349         IOobject
350         (
351             "U_Fehl",
352             runTime.timeName(),
353             mesh,
354             IOobject::MUST_READ,
355             IOobject::AUTO_WRITE
356         ),
357         mesh
358     );
359 
360     Info<< "Reading field rho\n" << endl;
361     volScalarField rho
362     (
363         IOobject
364         (
365             "rho",
366             runTime.timeName(),
367             mesh,
368             IOobject::MUST_READ,
369             IOobject::AUTO_WRITE
370         ),
371         mesh
372     );
373 
374     Info<< "Reading field nu\n" << endl;
375     volScalarField nu
376     (
377         IOobject
378         (
379             "nu",
380             runTime.timeName(),
381             mesh,
```

```
382             IOobject::MUST_READ,
383             IOobject::AUTO_WRITE
384         ),
385         mesh
386     );
387
388
389     Info<< "Reading field nHat_center\n" << endl;
390     volVectorField nHat_center
391     (
392         IOobject
393         (
394             "nHat_center",
395             runTime.timeName(),
396             mesh,
397             IOobject::MUST_READ,
398             IOobject::AUTO_WRITE
399         ),
400         mesh
401     );
402
403     Info<< "Reading field gradC\n" << endl;
404     volVectorField gradC
405     (
406         IOobject
407         (
408             "gradC",
409             runTime.timeName(),
410             mesh,
411             IOobject::MUST_READ,
412             IOobject::AUTO_WRITE
413         ),
414         mesh
415     );
416     Info<< "Reading field gradL\n" << endl;
417     volVectorField gradL
418     (
419         IOobject
420         (
421             "gradL",
422             runTime.timeName(),
423             mesh,
424             IOobject::MUST_READ,
425             IOobject::AUTO_WRITE
426         ),
427         mesh
428     );
429
```

```
430
431     Info<< "Reading/calculating face flux field phi\n" <<
        endl;
432
433     surfaceScalarField phi
434     (
435         IOobject
436         (
437             "phi",
438             runTime.timeName(),
439             mesh,
440             IOobject::MUST_READ,
441             IOobject::AUTO_WRITE
442         ),
443         mesh
444     );
445
446     surfaceScalarField C_face
447     (
448         IOobject
449         (
450             "C_face",
451             runTime.timeName(),
452             mesh,
453             IOobject::MUST_READ,
454             IOobject::AUTO_WRITE
455         ),
456         mesh
457     );
458
459     surfaceScalarField L_face
460     (
461         IOobject
462         (
463             "L_face",
464             runTime.timeName(),
465             mesh,
466             IOobject::MUST_READ,
467             IOobject::AUTO_WRITE
468         ),
469         (linearInterpolate(levelset))
470     );
471
472     surfaceScalarField int_face
473     (
474         IOobject
475         (
476             "int_face",
```

```
477             runTime.timeName(),
478             mesh,
479             IOobject::MUST_READ,
480             IOobject::AUTO_WRITE
481         ),
482         (linearInterpolate(levelset))
483     );
484
485     surfaceVectorField gradL_face
486     (
487        IOobject
488         (
489             "gradL_face",
490             runTime.timeName(),
491             mesh,
492             IOobject::MUST_READ,
493             IOobject::AUTO_WRITE
494         ),
495         (linearInterpolate(gradL))
496     );
497
498     Info<< "Reading field nHat_face\n" << endl;
499     surfaceVectorField nHat_face
500     (
501         IOobject
502         (
503             "nHat_face",
504             runTime.timeName(),
505             mesh,
506             IOobject::MUST_READ,
507             IOobject::AUTO_WRITE
508         ),
509         mesh
510     );
511
512
513
514
515     label pRefCell = 0;
516     scalar pRefValue = 0.0;
517     //setRefCell(p, mesh.solutionDict().subDict("PISO"),
    pRefCell, pRefValue);
```

Programmcode B.8: different_functions.H

```
1 //Matrixmultiplikation
2 RectangularMatrix<scalar> MatrixMult(RectangularMatrix<
    scalar> A, RectangularMatrix<scalar> B)
3 {
```

```
4       RectangularMatrix<scalar> C(A.n(), B.m(),0);
5       for (int i = 0; i < A.n(); i++)
6       {
7           for (int j = 0; j < B.m(); j++)
8           {
9               for (int k = 0; k < A.m(); k++)
10              {
11                  C[i][j] += A[i][k]*B[k][j];
12              }
13          }
14      }
15      return C;
16  }
17
18  class profile
19  {
20      public:
21
22          double alpha;
23          double beta;
24
25          double L0_inner;
26          double L0_outer;
27          double p0_fWert_inner;
28          double p0_erste_inner;
29          double p0_zweite_inner;
30          double p0_dritte_inner;
31          double p0_vierte_inner;
32          double p0_fWert_outer;
33
34          double a1;
35          double a3;
36          double a5;
37          double a7;
38          double a9;
39
40          double F3_max;
41          double F2_max;
42          double df1dL1_max;
43          double df2dL2_max;
44          double df3dL3_max;
45          double L_df2dL2_max;
46          double L_df3dL3_max;
47
48          double Psi_erste;
49          double Psi_zweite;
50
51          double IntegrationConst1;                    //Neccessary
```

```
     to make the transition of Int1(L) at L_inner
52         double IntegrationConst2;                //Neccessary
     to make the transition of Int2(L) at L_inner
53         double IntegrationConst3;                //Neccessary
     to make the transition of Int3(L) at L_inner
54
55         double minimalDerivative;
56
57
58         profile(double input1, double input2, double input3,
     double input4)     //Constructor defines all constants
    used in the profile
59         {
60             //Vordefinieren von Konstanten, die nachher bei
    der numeirschen Invertierung des C-Profils benoetigt
    werden:
61             // c = a1*L + a3*L3 + .... + a9*L9
62
63
64
65
66             alpha = input1;                          //Dadurch
    ist definiert, dass c = 1-exp( ++++ alpha*L + beta)
67             beta = input2;
68             L0_inner = input3;          //Uebergangsstelle
    des inneren Polynoms auf die exp-Fkt. Iterative
    Berechnung des opt L_inner in Excel-Tabelle exp-Profil3
69             L0_outer = input4;                       //
    Uebergangsstelle, ab der L nicht mehr berechnet wird
70
71             //Berechnung des optimalen beta in Excel-Tabelle
     exp-Profil3, damit Psi''(c) sowohl im Bereich c=0, als
    auch im Bereich c=1 ausgereizt wird und
    Stabilitaetskriterium bestmoeglich genutzt wird.
72
73
74             minimalDerivative = -alpha*std::exp(alpha*
    L0_inner+beta);          //haendisch ausrechnen, damit
    im Newton-Verfahren niemals durch 0 geteilt wird.
75
76             Psi_erste = alpha*alpha;
77             Psi_zweite = 0.0;
78
79             p0_fWert_inner = 1-std::exp(alpha*L0_inner+beta)
    ;                  //Funktionswert an der
    Uebergangsstelle L0_inner
80             p0_erste_inner = (p0_fWert_inner-1)*alpha;
                                //erste Ableitung an der
```

```
    Uebergangsstelle L0_inner
81          p0_zweite_inner = p0_erste_inner*alpha;
                              //zweite Ableitung an der
    Uebergangsstelle L0_inner
82          p0_dritte_inner = p0_zweite_inner*alpha;
                              //zweite Ableitung an der
    Uebergangsstelle L0_inner
83          p0_vierte_inner = p0_dritte_inner*alpha;
                              //zweite Ableitung an der
    Uebergangsstelle L0_inner
84          p0_fWert_outer = 1-std::exp(alpha*L0_outer+beta)
    ;
85
86          RectangularMatrix<scalar> A(5,5,0);
87          A[0][0] = pow(L0_inner,1);
88          A[0][1] = pow(L0_inner,3);
89          A[0][2] = pow(L0_inner,5);
90          A[0][3] = pow(L0_inner,7);
91          A[0][4] = pow(L0_inner,9);
92
93          A[1][0] = 1.0;
94          A[1][1] = 3*pow(L0_inner,2);
95          A[1][2] = 5*pow(L0_inner,4);
96          A[1][3] = 7*pow(L0_inner,6);
97          A[1][4] = 9*pow(L0_inner,8);
98
99          A[2][0] = 0.0;
100         A[2][1] = 3*2*pow(L0_inner,1);
101         A[2][2] = 5*4*pow(L0_inner,3);
102         A[2][3] = 7*6*pow(L0_inner,5);
103         A[2][4] = 9*8*pow(L0_inner,7);
104
105         A[3][0] = 0.0;
106         A[3][1] = 3*2;
107         A[3][2] = 5*4*3*pow(L0_inner,2);
108         A[3][3] = 7*6*5*pow(L0_inner,4);
109         A[3][4] = 9*8*7*pow(L0_inner,6);
110
111         A[4][0] = 0.0;
112         A[4][1] = 0.0;
113         A[4][2] = 5*4*3*2*pow(L0_inner,1);
114         A[4][3] = 7*6*5*4*pow(L0_inner,3);
115         A[4][4] = 9*8*7*6*pow(L0_inner,5);
116
117         RectangularMatrix<scalar> B(5,1,0);
118         B[0][0] = p0_fWert_inner;
119         B[1][0] = p0_erste_inner;
120         B[2][0] = p0_zweite_inner;
```

```
121             B[3][0] = p0_dritte_inner;
122             B[4][0] = p0_vierte_inner;
123 
124             RectangularMatrix<scalar> I5(5,5,0);
125             I5[0][0] = 1.0;
126             I5[1][1] = 1.0;
127             I5[2][2] = 1.0;
128             I5[3][3] = 1.0;
129             I5[4][4] = 1.0;
130 
131             SVD Inverse(A);
132             RectangularMatrix<scalar> ZW(Inverse.VSinvUt());
133             RectangularMatrix<scalar> konv(MatrixMult(ZW,A))
    ;
134 
135             ZW = MatrixMult((2*I5 - konv),ZW);
136             konv = MatrixMult(ZW,A);
137             ZW = MatrixMult((2*I5 - konv),ZW);
138             konv = MatrixMult(ZW,A);
139             ZW = MatrixMult((2*I5 - konv),ZW);
140             konv = MatrixMult(ZW,A);
141             ZW = MatrixMult((2*I5 - konv),ZW);
142             konv = MatrixMult(ZW,A);
143             ZW = MatrixMult((2*I5 - konv),ZW);
144             konv = MatrixMult(ZW,A);
145 
146             int zeile(0);
147             int spalte(0);
148             scalar maxMatrix(0.0);
149 
150             for(zeile = 0; zeile < 4.5; zeile++)
151             {
152                 for(spalte = 0; spalte < 4.5; spalte++)
153                 {
154                     maxMatrix = max( maxMatrix, std::abs(I5[
    zeile][spalte] - konv[zeile][spalte]));
155                 }
156             }
157 
158             B = MatrixMult(ZW,B);
159             a1 = B[0][0];
160             a3 = B[1][0];
161             a5 = B[2][0];
162             a7 = B[3][0];
163             a9 = B[4][0];
164 
165             IntegrationConst1 = L0_inner*L0_inner*(a1/2 +
    L0_inner*L0_inner*(a3/4 + L0_inner*L0_inner*(a5/6 +
```

```
    L0_inner*L0_inner*(a7/8 + L0_inner*L0_inner*a9/10)))) - (
    L0_inner-std::exp(alpha*L0_inner+beta)/alpha);
166             IntegrationConst2 = L0_inner*L0_inner*L0_inner*(
    a1/2/3 + L0_inner*L0_inner*(a3/4/5 + L0_inner*L0_inner*(
    a5/6/7 + L0_inner*L0_inner*(a7/8/9 + L0_inner*L0_inner*a9
    /10/11)))) - (L0_inner*(IntegrationConst1+L0_inner/2)-std
    ::exp(alpha*L0_inner+beta)/alpha/alpha);
167             IntegrationConst3 = L0_inner*L0_inner*L0_inner*
    L0_inner*(a1/2/3/4 + L0_inner*L0_inner*(a3/4/5/6 +
    L0_inner*L0_inner*(a5/6/7/8 + L0_inner*L0_inner*(a7
    /8/9/10 + L0_inner*L0_inner*a9/10/11/12)))) - (L0_inner*(
    IntegrationConst2+L0_inner*(IntegrationConst1/2+L0_inner
    /6))-std::exp(alpha*L0_inner+beta)/alpha/alpha/alpha);
168
169             F3_max = ((IntegrationConst3 + L0_outer*(
    IntegrationConst2 + L0_outer*(IntegrationConst1/2 +
    L0_outer/6)) - std::exp(alpha*L0_outer+beta)/(alpha*alpha
    *alpha)));
170             F2_max = ((IntegrationConst2 + L0_outer*(
    IntegrationConst1 + L0_outer/2) - std::exp(alpha*L0_outer
    +beta)/(alpha*alpha)));
171             df1dL1_max = a1;
172             L_df2dL2_max = L0_inner/2;  //Analytische
    Berechnung des Maximums von df2/dL2 wuerde Cardanische
    Formel mit a3 bis a9 erfordern. Hier ein wenig Newton-
    Methode
173             L_df2dL2_max -= (6*a3 +L_df2dL2_max*L_df2dL2_max
    *(60*a5 + L_df2dL2_max*L_df2dL2_max*(210*a7 + 504*a9*
    L_df2dL2_max*L_df2dL2_max)))/(L_df2dL2_max*(120*a5 +
    L_df2dL2_max*L_df2dL2_max*(840*a7 + 3024*a9*L_df2dL2_max*
    L_df2dL2_max)));
174             L_df2dL2_max -= (6*a3 +L_df2dL2_max*L_df2dL2_max
    *(60*a5 + L_df2dL2_max*L_df2dL2_max*(210*a7 + 504*a9*
    L_df2dL2_max*L_df2dL2_max)))/(L_df2dL2_max*(120*a5 +
    L_df2dL2_max*L_df2dL2_max*(840*a7 + 3024*a9*L_df2dL2_max*
    L_df2dL2_max)));
175             L_df2dL2_max -= (6*a3 +L_df2dL2_max*L_df2dL2_max
    *(60*a5 + L_df2dL2_max*L_df2dL2_max*(210*a7 + 504*a9*
    L_df2dL2_max*L_df2dL2_max)))/(L_df2dL2_max*(120*a5 +
    L_df2dL2_max*L_df2dL2_max*(840*a7 + 3024*a9*L_df2dL2_max*
    L_df2dL2_max)));
176             L_df2dL2_max -= (6*a3 +L_df2dL2_max*L_df2dL2_max
    *(60*a5 + L_df2dL2_max*L_df2dL2_max*(210*a7 + 504*a9*
    L_df2dL2_max*L_df2dL2_max)))/(L_df2dL2_max*(120*a5 +
    L_df2dL2_max*L_df2dL2_max*(840*a7 + 3024*a9*L_df2dL2_max*
    L_df2dL2_max)));
177             L_df2dL2_max -= (6*a3 +L_df2dL2_max*L_df2dL2_max
    *(60*a5 + L_df2dL2_max*L_df2dL2_max*(210*a7 + 504*a9*
```

```
    L_df2dL2_max*L_df2dL2_max)))/(L_df2dL2_max*(120*a5 +
    L_df2dL2_max*L_df2dL2_max*(840*a7 + 3024*a9*L_df2dL2_max*
    L_df2dL2_max)));
178             L_df2dL2_max -= (6*a3 +L_df2dL2_max*L_df2dL2_max
    *(60*a5 + L_df2dL2_max*L_df2dL2_max*(210*a7 + 504*a9*
    L_df2dL2_max*L_df2dL2_max)))/(L_df2dL2_max*(120*a5 +
    L_df2dL2_max*L_df2dL2_max*(840*a7 + 3024*a9*L_df2dL2_max*
    L_df2dL2_max)));
179             L_df2dL2_max -= (6*a3 +L_df2dL2_max*L_df2dL2_max
    *(60*a5 + L_df2dL2_max*L_df2dL2_max*(210*a7 + 504*a9*
    L_df2dL2_max*L_df2dL2_max)))/(L_df2dL2_max*(120*a5 +
    L_df2dL2_max*L_df2dL2_max*(840*a7 + 3024*a9*L_df2dL2_max*
    L_df2dL2_max)));
180             df2dL2_max = mag(L_df2dL2_max*(6*a3 +
    L_df2dL2_max*L_df2dL2_max*(20*a5 + L_df2dL2_max*
    L_df2dL2_max*(42*a7 + L_df2dL2_max*L_df2dL2_max*72*a9))))
    ;
181
182             L_df3dL3_max = (L0_inner + L_df2dL2_max)/2;  //
    Analytische Berechnung des Maximums von df3/dL3 wuerde
    Cardanische Formel mit a3 bis a9 erfordern. Hier ein
    wenig Newton-Methode
183             L_df3dL3_max -= (L_df3dL3_max*(120*a5 +
    L_df3dL3_max*L_df3dL3_max*(840*a7 + 3024*a9*L_df3dL3_max*
    L_df3dL3_max)))/(120*a5 + L_df3dL3_max*L_df3dL3_max
    *(2520*a7 + 15120*a9*L_df3dL3_max*L_df3dL3_max));
184             L_df3dL3_max -= (L_df3dL3_max*(120*a5 +
    L_df3dL3_max*L_df3dL3_max*(840*a7 + 3024*a9*L_df3dL3_max*
    L_df3dL3_max)))/(120*a5 + L_df3dL3_max*L_df3dL3_max
    *(2520*a7 + 15120*a9*L_df3dL3_max*L_df3dL3_max));
185             L_df3dL3_max -= (L_df3dL3_max*(120*a5 +
    L_df3dL3_max*L_df3dL3_max*(840*a7 + 3024*a9*L_df3dL3_max*
    L_df3dL3_max)))/(120*a5 + L_df3dL3_max*L_df3dL3_max
    *(2520*a7 + 15120*a9*L_df3dL3_max*L_df3dL3_max));
186             L_df3dL3_max -= (L_df3dL3_max*(120*a5 +
    L_df3dL3_max*L_df3dL3_max*(840*a7 + 3024*a9*L_df3dL3_max*
    L_df3dL3_max)))/(120*a5 + L_df3dL3_max*L_df3dL3_max
    *(2520*a7 + 15120*a9*L_df3dL3_max*L_df3dL3_max));
187             L_df3dL3_max -= (L_df3dL3_max*(120*a5 +
    L_df3dL3_max*L_df3dL3_max*(840*a7 + 3024*a9*L_df3dL3_max*
    L_df3dL3_max)))/(120*a5 + L_df3dL3_max*L_df3dL3_max
    *(2520*a7 + 15120*a9*L_df3dL3_max*L_df3dL3_max));
188             L_df3dL3_max -= (L_df3dL3_max*(120*a5 +
    L_df3dL3_max*L_df3dL3_max*(840*a7 + 3024*a9*L_df3dL3_max*
    L_df3dL3_max)))/(120*a5 + L_df3dL3_max*L_df3dL3_max
    *(2520*a7 + 15120*a9*L_df3dL3_max*L_df3dL3_max));
189             L_df3dL3_max -= (L_df3dL3_max*(120*a5 +
    L_df3dL3_max*L_df3dL3_max*(840*a7 + 3024*a9*L_df3dL3_max*
```

```
    L_df3dL3_max)))/(120*a5 + L_df3dL3_max*L_df3dL3_max
    *(2520*a7 + 15120*a9*L_df3dL3_max*L_df3dL3_max));
190             L_df3dL3_max -= (L_df3dL3_max*(120*a5 +
    L_df3dL3_max*L_df3dL3_max*(840*a7 + 3024*a9*L_df3dL3_max*
    L_df3dL3_max)))/(120*a5 + L_df3dL3_max*L_df3dL3_max
    *(2520*a7 + 15120*a9*L_df3dL3_max*L_df3dL3_max));
191             df3dL3_max = mag(6*a3 + L_df3dL3_max*
    L_df3dL3_max*(60*a5 + L_df3dL3_max*L_df3dL3_max*(210*a7 +
     L_df3dL3_max*L_df3dL3_max*504*a9)));  // Der Wert 6*a3
    mag groesser sein, aber fuer den Uebergang in
    df3dL3_local brauche ich auch diesen Wert.
192
193
194
195
196             cout << "p0_fWert_inner: " << p0_fWert_inner <<
    nl;
197             cout << "p0_erste_inner: " << p0_erste_inner <<
    nl;
198             cout << "p0_zweite_inner: " << p0_zweite_inner
    << nl;
199             cout << "p0_dritte_inner: " << p0_dritte_inner
    << nl;
200             cout << "p0_vierte_inner: " << p0_vierte_inner
    << nl;
201
202             cout << "p0_fWert_outer: " << p0_fWert_outer <<
    nl;
203
204             cout << "Genauigkeit Inverse = " << maxMatrix <<
    nl;
205             cout << "a1: " << a1 << nl;
206             cout << "a3: " << a3 << nl;
207             cout << "a5: " << a5 << nl;
208             cout << "a7: " << a7 << nl;
209             cout << "a9: " << a9 << nl;
210
211             cout << "IntegrationConst1: " <<
    IntegrationConst1 << nl;
212             cout << "IntegrationConst2: " <<
    IntegrationConst2 << nl;
213             cout << "IntegrationConst3: " <<
    IntegrationConst3 << nl;
214
215             cout << "F3_max: " << F3_max << nl;
216             cout << "df1dL1_max: " << df1dL1_max << nl;
217             cout << "df2dL2_max: " << df2dL2_max << nl;
218             cout << "L_df2dL2_max: " << L_df2dL2_max << nl;
```

```
219             cout << "df3dL3_max: " << df3dL3_max << nl;
220             cout << "L_df3dL3_max: " << L_df3dL3_max << nl;
221
222
223             cout << "minimalDerivative: " <<
    minimalDerivative << nl;
224
225         }
226         double ChemPot_outer(double value)
227         {
228             return ( (value-Foam::sign(value))*alpha*alpha )
    ;
229         }
230         double Inverse(double C_value)
231         {
232             if(mag(C_value) < p0_fWert_inner)
233             {
234                 scalar L_value(0.0);
235                 scalar residuum(10.0);
236                 do
237                 {
238                     residuum = IntL(0,L_value)-C_value;
239                     L_value -= (residuum/IntL(-1,L_value));
240                 } while (mag(residuum) > 0.00000000000001);
241                 return (L_value);
242             }
243             else
244             {
245                 return ((Foam::log(1-mag(C_value))-beta)/
    alpha*Foam::sign(C_value));
246             }
247         }
248         double IntL(int nIntegration, double value)
249         {
250             scalar L_square (value*value);
251             switch (nIntegration)
252             {
253                 case -3:
254                     if(L_square < L0_inner*L0_inner)
255                     {
256                         return (6*a3 + L_square*(60*a5 +
    L_square*(210*a7 + L_square*504*a9)));
257                     }
258                     else
259                     {
260                         return (-alpha*alpha*alpha*std::exp(
    alpha*mag(value) + beta));
261                     }
```

```
262
263                 case -2:
264                     if(L_square < L0_inner*L0_inner)
265                     {
266                         return (value*(6*a3 + L_square*(20*
    a5 + L_square*(42*a7 + L_square*72*a9))));
267                     }
268                     else
269                     {
270                         return (-alpha*alpha*sign(value)*std
    ::exp(alpha*mag(value) + beta));
271                     }
272
273                 case -1:
274                     if(L_square < L0_inner*L0_inner)
275                     {
276                         return (a1 + L_square*(3*a3 +
    L_square*(5*a5 + L_square*(7*a7 + L_square*9*a9))));
277                     }
278                     else
279                     {
280                         return (-alpha*std::exp(alpha*mag(
    value) + beta));
281                     }
282
283                 case 0:
284                     if(L_square < L0_inner*L0_inner)
285                     {
286                         return (value*(a1 + L_square*(a3 +
    L_square*(a5 + L_square*(a7 + L_square*a9)))));
287                     }
288                     else
289                     {
290                         return(sign(value)*(1-std::exp(alpha
    *mag(value) + beta)));
291                     }
292
293                 case 1:
294                     if(L_square < L0_inner*L0_inner)
295                     {
296                         return (L_square*(a1/2 + L_square*(
    a3/4 + L_square*(a5/6 + L_square*(a7/8 + L_square*a9/10))
    )));
297                     }
298                     else
299                     {
300                         return (mag(value)-std::exp(alpha*
    mag(value)+beta)/alpha+IntegrationConst1);
```

```
301                     }
302
303                 case 2:
304                     if(L_square < L0_inner*L0_inner)
305                     {
306                         return (value*L_square*(a1/6 +
    L_square*(a3/20 + L_square*(a5/42 + L_square*(a7/72 +
    L_square*a9/110)))));
307                     }
308                     else
309                     {
310                         return (sign(value)*(
    IntegrationConst2 + mag(value)*(IntegrationConst1 + mag(
    value)/2) - std::exp(alpha*mag(value)+beta)/(alpha*alpha)
      ));
311                     }
312
313                 case 3:
314                     if(L_square < L0_inner*L0_inner)
315                     {
316                         return (L_square*L_square*(a1/24 +
    L_square*(a3/120 + L_square*(a5/336 + L_square*(a7/720 +
    L_square*a9/1320)))));
317                     }
318                     else
319                     {
320                         return ((IntegrationConst3 + mag(
    value)*(IntegrationConst2 + mag(value)*(IntegrationConst1
    /2 + mag(value)/6)) - std::exp(alpha*mag(value)+beta)/(
    alpha*alpha*alpha)));
321                     }
322
323                 default:
324                     cout << "nIntegration = " <<
    nIntegration << "     value = " << value << nl;
325                     FatalErrorIn("no macth for nIntegration.
    Select -1,0,1,2, or 3 as nIntegration") << abort(
    FatalError);
326                     return 0;
327             }
328         }
329 };
```

Programmcode B.9: directInterface.H

```
1 directInterface = false;
2 visualDirectInterFace = 0;
3
4
```

```
5 //#pragma omp parallel for schedule(guided)      //Aus
      irgendeinem Grund fuehrt diese Parallelisierung zu
      abweichenden Ergebnissen zwischen seriell und parallel
6 for(int faceI = 0; faceI < owner.size(); faceI++)
7 {
8     if ((L_ref[owner[faceI]]*L_ref[neighbour[faceI]]) < 0.0)
9     {
10        directInterface[owner[faceI]] = true;
11        visualDirectInterFace_ref[owner[faceI]] = 1;
12        directInterface[neighbour[faceI]] = true;
13        visualDirectInterFace_ref[neighbour[faceI]] = 1;
14    }
15 }
```

Programmcode B.10: getLfromC.H

```
1 //benuetigt ein vorheriges pureLevelsset.
2 #include "pureLevelsetLists.H"
3
4 #pragma omp parallel for schedule(guided)
5 for(int cellI = 0; cellI < cells.size(); cellI++)
6 {
7     if(mag(C_volaverage_ref[cellI]) <= CfromL.p0_fWert_outer
      )           //Wie viele Iterationen von Newton noetig
      sind, muss ich noch genauer bestimmen
8     {
9         L_ref[cellI] = CfromL.Inverse(C_volaverage_ref[cellI
      ]);
10    }
11    else
12    {
13        L_ref[cellI] = 0;
14    }
15 }
16
17 #include "directInterface.H"
18
19 forAll(levelset.boundaryField(), patchI)
20 {
21    if(levelset.boundaryField()[patchI].type() == "
      phaseFieldExpL")
22    {
23        phaseFieldExpLFvPatchScalarField& L_patch_ref =
      dynamic_cast<phaseFieldExpLFvPatchScalarField&>(levelset.
      boundaryField()[patchI]);
24        L_patch_ref.alpha() = L_patch_ref.theta();
25    }
26 }
27
```

```
28
29 i = 0;
30 j = 0;
31
32 C_test = 0;
33 cellMidPoint = 0;
34 faceMidPoint = 0;
35 edgeGauss = 0;
36 IntegratedEdge = 0;
37 error_estimate_edge = 0;
38
39
40
41
42
43 do
44 {
45     levelset.correctBoundaryConditions();
46     gradL = fvc::grad(levelset);
47
48     if(i > 3.0)
49     {
50         #include "harmoniceBC.H"
51         //Erst beim dritten die Korrektur der alpha-Werte
    zuschalten. So funktioniert es auf jeden Fall. Anders
    eventuell auch (nicht getestet)
52     }
53
54
55     for(int cellI = 0; cellI < cells.size(); cellI++)
56     {
57         if(pureLevelset[cellI])
58         {
59             if(!n_face_BoolList[cellI])
60             {
61                 #include "updateNormalVectors.H"
62             }
63         }
64     }
65     #pragma omp parallel for schedule(guided) firstprivate(
    L_point, F3_point, IntegratedEdge, error_estimate_edge)
66     for(int cellI = 0; cellI < cells.size(); cellI++)
67     {
68         if(pureLevelset[cellI])
69         {
70             #include "calculateC_test2_parallel.H"

71             scalar change_in_L (local_C_test/max(mag(
```

```
    df1dL1_local), 0.000000001));
72              L_ref[cellI] -= max(  min (change_in_L, 0.05)
    ,-0.05);                //L soll sich pro Iteration
    hoechstens um 0.1 aendern duerfen
73              /*if(mag(L_ref[cellI]) > CfromL.L0_outer)
74              {
75                  L_ref[cellI] = Foam::max(Foam::min(L_ref[
    cellI], CfromL.L0_outer),-CfromL.L0_outer);
76                  #pragma omp critical
77                  {
78                      pureLevelset[cellI] = false;
                    //In "calculate_C_test2_parallel" musste
    das auf jeden Fall critical sein. Zur Sicherheit hier
    dann lieber auch.
79                  }
80                  Ly_ref[cellI] = 0;
81                  C_test_ref[cellI] = 0;
82              }*/
83          }
84      }
85
86      if(max(mag(C_test_ref)) < GetLfromC_accuracy)
87      {
88          i = nGetLfromC;
89      }
90      else
91      {
92          i++;
93          j++;
94      }
95
96  } while (i < nGetLfromC);
97
98  cout << "Max(|C_test|) = " << max(mag(C_test_ref)) << "
    in " << j << " Iterations" << nl;
99  if (max(mag(C_test_ref)) > 1.0)
100 {
101     forAll(cells, cellI)
102     {
103         if (mag(C_test_ref[cellI]) > 1.0)
104         {
105             cout << "WARNING! cellI = " << cellI << "
    C_test_ref[cellI] = " << C_test_ref[cellI] << nl;
106         }
107     }
108 }
```

Programmcode B.11: harmoniceBC.H

```
1  inverseR *= 0;
2
3  forAll(levelset.boundaryField(), patchI)
4  {
5      if(levelset.boundaryField()[patchI].type() == "
       phaseFieldExpL")
6      {
7          phaseFieldExpLFvPatchScalarField& L_patch_ref =
       dynamic_cast<phaseFieldExpLFvPatchScalarField&>(levelset.
       boundaryField()[patchI]);
8          nHatFvPatchVectorField& nHat_patch_ref =
       dynamic_cast<nHatFvPatchVectorField&>(nHat_center.
       boundaryField()[patchI]);
9          const labelUList& faceCells = L_patch_ref.patch().
       faceCells();
10         scalarField& alpha_L_ref = L_patch_ref.alpha();
11         scalarField& alpha_nHat_ref = nHat_patch_ref.alpha()
       ;
12         const vectorField nf(L_patch_ref.patch().nf());
13         const vectorField& localSf_ref(L_patch_ref.patch().
       Sf());
14         //const vectorField& delta_ref = L_patch_ref.patch()
       .delta();   //quote from fvPtach.H:  Return cell-centre
       to face-centre vector. Das geht aber aus irgendeinem
       Grund nicht. delta[Index = 0] ist stets falsch.
15         const scalarField& deltaCoeff_ref = L_patch_ref.
       patch().deltaCoeffs();
16
17
18         //#pragma omp parallel for schedule(guided)     //
       Aus irgendeinem Grund faehrt diese Parallelisierung zu
       abweichenden Ergebnissen zwischen seriell und parallel
19         for(int faceI = 0; faceI < faceCells.size(); faceI
       ++)
20         {
21             if((pureLevelset[faceCells[faceI]] &&
       directInterface[faceCells[faceI]]))
22                 //Koordinatensystem wird so gewaehlt, dass
       nf des Patches unser Einheitsnormalenvektor e_y ist
23                 //Dadurch zeigt delta effektiv in positive y
       -Richtung.
24                 // => d(cos(theta))/dy = (Winkel_patch -
       Winkel_zelle)/(y_patch - y_zelle)
25             {
26                 scalar angle((nf[faceI] & gradL_ref[
       faceCells[faceI]])/(mag(gradL_ref[faceCells[faceI]]) +
       numStab1.value()));  //Winkel in der anliegenden Zelle
27                 scalar local_inverseR(    -(Foam::cos(
```

```
    alpha_L_ref[faceI]/180.0*Foam::constant::mathematical::pi
    )-angle)*deltaCoeff_ref[faceI]/pow(Foam::sin(alpha_L_ref[
    faceI]/180.0*Foam::constant::mathematical::pi),2));
                    //    dcos(theta)/dy =     - x2/sqrt(x2+y2)
    3    =    -sin(theta)2/R
28                  //Angenommen wird dabei L = ?????
29
30                  local_inverseR = max(-MaxInverseR,   min(
    MaxInverseR, local_inverseR));
31                  //Damit es konvergieren kann, muss ich
    beschraenken, wie gross inverseR sein kann.
32                  //Kontaktwinkel mit inverseR > MaxInverseR
    werden dann "Falsch" abgebildet, bzw mit den Problemen,
    die ein konstanter Kontaktwinkel am gesamten Patch haette
33
34                  inverseR_ref[faceCells[faceI]] =
    local_inverseR;
35                  scalar extrapolierterWinkel1(Foam::cos(
    L_patch_ref.theta()/180.0*Foam::constant::mathematical::
    pi)*(1.0 + L_patch_ref.epsilon()*local_inverseR*
    L_patch_ref[faceI]) - Foam::cos(alpha_L_ref[faceI]/180.0*
    Foam::constant::mathematical::pi));
36
37                  scalar dL(1/deltaCoeff_ref[faceI]/
    L_patch_ref.epsilon()*(Foam::cos((alpha_L_ref[faceI]+
    dalpha)/180*Foam::constant::mathematical::pi) - Foam::cos
    (alpha_L_ref[faceI]/180*Foam::constant::mathematical::pi)
    ));
38                  vector newGradL(gradL_ref[faceCells[faceI]]
    + dL*localSf_ref[faceI]/volume[faceCells[faceI]]);  //
    Nimmt an, dass gradL per Gaussdiskretisierung berechnet
    wird.
39                  angle = (nf[faceI] & newGradL)/(mag(newGradL
    ) + numStab1.value());
40                  local_inverseR = (    (angle - Foam::cos((
    alpha_L_ref[faceI]+dalpha)/180.0*Foam::constant::
    mathematical::pi))*deltaCoeff_ref[faceI]/pow(Foam::sin((
    alpha_L_ref[faceI]+dalpha)/180.0*Foam::constant::
    mathematical::pi),2));
41                  local_inverseR = max(-MaxInverseR,   min(
    MaxInverseR, local_inverseR));
42                  scalar extrapolierterWinkel2((Foam::cos(
    L_patch_ref.theta()/180.0*Foam::constant::mathematical::
    pi)*(1.0 + L_patch_ref.epsilon()*local_inverseR*(
    L_patch_ref[faceI]+dL)) - Foam::cos((alpha_L_ref[faceI]+
    dalpha)/180.0*Foam::constant::mathematical::pi)));
43
44                  alpha_L_ref[faceI] -= (extrapolierterWinkel1
```

```
    / (extrapolierterWinkel2 - extrapolierterWinkel1) *
    dalpha);
45                  alpha_nHat_ref[faceI] = alpha_L_ref[faceI];

46              }
47          }
48
49
50          //#pragma omp parallel for schedule(guided)     //
    Aus irgendeinem Grund fuehrt diese Parallelisierung zu
    abweichenden Ergebnissen zwischen seriell und parallel
51          for(int faceI = 0; faceI < faceCells.size(); faceI
    ++)
52          {
53                  //Aus Stabilitaetsgrurnden kann nicht jede
    Zelle ihr eigenes inverseR haben. Dann divergiert der
    Diffusionsterm.
54                  //Muss mir aus zugehoeriger Beruehrstelle an
     der Wand das inverseR von dort holen.
55              if(pureLevelset[faceCells[faceI]])
56              {
57                  if(!directInterface[faceCells[faceI]])
58                  {
59                      vector ny(gradL_ref[faceCells[faceI]] -
    nf[faceI]*(nf[faceI] & gradL_ref[faceCells[faceI]]));
60                      ny /= (mag(ny) + numStab1.value());
61                      vector prob_L0_point(cellCentre[
    faceCells[faceI]] - ny*L_ref[faceCells[faceI]]*
    L_patch_ref.epsilon()/Foam::sin(alpha_L_ref[faceI]/180.0*
    Foam::constant::mathematical::pi));
62                      const label foundCell(searchEngine.
    findCell(prob_L0_point, faceCells[faceI]));
63                      if(foundCell < 0.0)
64                      {
65                          inverseR_ref[faceCells[faceI]] =
    0.0;
66                          //wenn der vermutete L0-Punkt
    ausserhalb des Meshes liegt, nehme ich einen unendlichen
    Kruemmungsradius an, was effektiv bedeutet, dass alpha =
    theta wird.
67                      }
68                      else    //Demnach liegt foundCell
    innerhalb des Meshes
69                      {
70                          if(directInterface[foundCell])  //
    Zelle getroffen? Super! Dann den inverseR-Wert
    uebernehmen.
71                          {
```

```
72                              inverseR_ref[faceCells[faceI]] =
    inverseR_ref[foundCell];
73                          }
74                          else    //Zelle nicht getroffen.
    Noch ein Versuch, den richtigen Wert zu finden.
75                          {
76                              ny = (    gradL_ref[foundCell] -
    nf[faceI]*(nf[faceI] & gradL_ref[foundCell]));
          //Hier nehme ich weiterhin nf[faceI], aber ich
    arbeite eh ohne gekruemmte Patches
77                              ny /= (mag(ny) + numStab1.value
    ());
78                              prob_L0_point = cellCentre[
    foundCell] - ny*L_ref[foundCell]*L_patch_ref.epsilon()/
    Foam::sin(L_patch_ref.theta()/180.0*Foam::constant::
    mathematical::pi);
79                              const label foundCell2(
    searchEngine.findCell(prob_L0_point, faceCells[faceI]));
80                              if(foundCell2 < 0.0)    //Liegt
    der zweite Versuch, im Mesh?
81                              {
82                                  inverseR_ref[faceCells[faceI
    ]] = 0.0;
83                                  //wenn der vermutete L0-
    Punkt ausserhalb des Meshes liegt, nehme ich fuer einen
    unendlichen Kruemmungsradius an, was effektiv bedeutet,
    dass alpha = theta wird.
84                              }
85                              else
86                              {
87                                  inverseR_ref[faceCells[faceI
    ]] = inverseR_ref[foundCell2];
88                              }
89                          }
90                      }
91
92                      scalar inner_of_Acos(max(-0.999, min
    (0.999, Foam::cos(L_patch_ref.theta()/180.0*Foam::
    constant::mathematical::pi)*(1.0 + L_patch_ref.epsilon()*
    inverseR_ref[faceCells[faceI]]*L_patch_ref[faceI]))));

93                      //Begrenzund des Argumentes des cos
    noetig. Extrapolation des Kontaktwinkels durch
    Linearisierung kann in bestimmten Bereichen
    unphysiklaische Ergebnisse liefern.
94
95                      alpha_L_ref[faceI] = Foam::acos(
    inner_of_Acos)/Foam::constant::mathematical::pi*180.0;
```

```
96                      alpha_nHat_ref[faceI] = alpha_L_ref[
    faceI];
97                  }     // end if(!directInterface[faceCells[
    faceI]])
98              }
99              else //also !(pureLevelset[faceCells[faceI]])
100             {
101                 alpha_L_ref[faceI] = L_patch_ref.theta();
                      //Finde ich keine Beruehrstelle, nehme ich
     das normale theta.
102                 alpha_nHat_ref[faceI] = alpha_L_ref[faceI];
                    //Finde ich keine Beruehrstelle, nehme ich
    das normale theta.
103             }     //end if(pureLevelset[faceCells[faceI]])
104         }
105     }
106 }
```

Programmcode B.12: initiateC_volaverage_parallel.H

```
1     scalar local_C_test(0.0);
2     scalar error_edge_anal(0.0);
3     scalar error_edge_num(0.0);
4     scalar error_face_num(0.0);
5     scalar error_face_anal(0.0);
6     scalar error_cell_num(0.0);
7     scalar error_cell_anal(0.0);
8
9     scalar local_volume(volume[cellI]);
10
11    scalar face_Zwischenergebnis(0.0);
12    scalar local_L_value(L_ref[cellI]);
13    vector local_gradL_value(gradL_ref[cellI]);
14    scalar df1dL1_local(CfromL.IntL(-1.0, local_L_value));
15    scalar df2dL2_local(mag(  CfromL.IntL(-2.0, max( CfromL.
    L_df2dL2_max ,mag(local_L_value)))));                  //
    Liege mit dem L-Wert ausserhalb von max(df2dL2)
16    Ableitung_ref[cellI] = df1dL1_local;
17    //Man koennte die Ableitung d(C_test)/dL exakter
    ausrechnen, aber die quadr. Konvergenzgeschw. des Newton-
    Verfahren, erreicht man eh nicht,
18    //weil Lx, Ly und Lz als konstant angesehen werden, was
    sie ja nicht sind.
19
20
21                  if( (local_gradL_value & local_gradL_value)
    == 0)     //Einer der wenigen Unterschiede zu
    calculate_C_test2_parallel
22                  {
```

```
23                      error_cell_anal = 10000000000000000;
24                      //Fehler der "(semi-)analytischen" Int. mag hoeher liegen. Hier wird angenommen, dass 10^17 ausreichend ist, damit schlussendlich die num. cell-Int. verwendet wird.
25                  }
26                  else
27                  {
28
29                      const labelList ListeDerPunkte(cellPoints[cellI]);  //Fuer jeden Punkt schon einmal L-Wert und F3-Wert berechnen
30                      for( int pointI = 0; pointI < ListeDerPunkte.size(); pointI++)
31                      {
32                          label point_labelI(ListeDerPunkte[pointI]);
33                          L_point[point_labelI] = (local_gradL_value & (points[point_labelI] - cellCentre[cellI])) + local_L_value;
34                          F3_point[point_labelI] = CfromL.IntL(3.0, L_point[point_labelI]);
35                          //Die F3-Werte berechne ich schon einmal auf Vorrat. Es sollte wesenltich seltener vorkommen, dass ein Wert nicht gebraucht wird,
36                          //weil eigenltich stets mag(gradL) = 1/eps herrschen sollte. Dann wird F3 auf einem der faces immer gebraucht.
37                      }
38
39                      const labelList ListeDerEdges(ZelleBestehtAus[cellI]);      //Fuer jedes Edge der Zelle das zugehoerige gamma berechnen
40                      for( int edgeI = 0; edgeI < ListeDerEdges.size(); edgeI++)
41                      {
42                          label edge_labelI(ListeDerEdges[edgeI]);
43                          vector local_n_line(n_line_VectorList[edge_labelI]);
44                          scalar local_gamma(local_gradL_value & n_line_VectorList[edge_labelI]);
45                          if(local_gamma == 0)    //Wenn der Line-Gradient echt gleich 0 ist, kann ich direkt die numeirsche Integration anwenden.
46                          {
47                              error_edge_anal = 10000000000000000;
```

```
48                              //Fehler der analytischen Int. mag hoeher liegen. Hier wird angenommen, dass 10^17 ausreichend ist, damit schlussendlich die num. Int. verwendet wird.
49                          }
50                          else
51                          {
52                              error_edge_anal = eps*CfromL.F3_max/mag(local_gamma);
53                          }
54                          error_edge_num = (eps*length_edge[edge_labelI] + mag(df2dL2_local/4320.0 *pow(local_gamma,4) *length_edge_pow5[edge_labelI]));
55                          //Fehler des numerischen Verfahrens durch Gauss + Auswertungsfehler, da IntL nur auf eps genau berechnet wird und dann noch mit der Edgelaenge multipliziert wird.
56
57                          if(error_edge_anal < error_edge_num)
58                          {       //analytischer Fall
59                              IntegratedEdge[edge_labelI] = (F3_point[edges[edge_labelI].end()] - F3_point[edges[edge_labelI].start()])/local_gamma;
60                              error_estimate_edge[edge_labelI] = error_edge_anal;
61                          }
62                          else
63                          {       //numerischer Fall
64                              if(mag(local_gamma) < eps)
        //weitere Beschleunigung, da fuer gamma kleiner eps auch die Mittelpunktsregel ausreichend ist  und man eine FKT-Auswertung einspart
65                              {
66                                  scalar L_mid( (L_point[edges[edge_labelI].start()] + L_point[edges[edge_labelI].end()])/2);
67                                  IntegratedEdge[edge_labelI] = (CfromL.IntL(2.0, L_mid)*length_edge[edge_labelI]);

68                                  error_estimate_edge[edge_labelI] = eps*length_edge[edge_labelI];
69                                  //Fehler des numerischen Verfahrens ~0 + Auswertungsfehler, da IntL nur auf eps genau berechnet wird und dann noch mit der Edgelaenge multipliziert wird.
70
71                              }
72                              else
```

```
73                                  {
74                                      scalar L_unten(L_point[edges
    [edge_labelI].start()]*gauss_weight1 + L_point[edges[
    edge_labelI].end()]*gauss_weight2);
75                                      scalar L_oben(L_point[edges[
    edge_labelI].start()]*gauss_weight2 + L_point[edges[
    edge_labelI].end()]*gauss_weight1);
76
77                                      IntegratedEdge[edge_labelI]
    = (CfromL.IntL(2.0, L_unten) + CfromL.IntL(2.0, L_oben))*
    length_edge[edge_labelI]/2;
78                                      error_estimate_edge[
    edge_labelI] = error_edge_num;
79                                      edgeGauss_ref[cellI] = 1;
80                                  }
81                              } // end if(error_edge_anal <
    error_edge_num)
82                          } // end for( int edgeI = 0; edgeI <
    ListeDerEdges.size(); edgeI++)
83
84                          const labelList ListeDerFaces(cells[
    cellI]);
85                          List<vector>& n_face_VectorSubList =
    n_face_VectorList[cellI];
86                          for(int faceI = 0; faceI < ListeDerFaces
    .size(); faceI++)
87                          {
88                              label face_labelI(ListeDerFaces[
    faceI]);
89                              scalar local_alpha(local_gradL_value
     & n_face_VectorSubList[faceI]);
90                              scalar magGradLface( (
    local_gradL_value & local_gradL_value) - local_alpha*
    local_alpha);       //Vorsicht: MagGradLface enthaelt
    mag(gradL)2
91
92                              if(magGradLface == 0) //Wenn der
    Face-Gradient echt gleich 0 ist, kann ich direkt die
    numeirsche Integration anwenden.
93                              {
94                                  error_face_anal =
    10000000000000000;
95                                  //Fehler der "(semi-)
    analytischen" Int. mag hoeher liegen. Hier wird
    angenommen, dass 10^17 ausreichend ist, damit
    schlussendlich die num. face-Int. verwendet wird.
96                              }
97                              else
```

```
98                          {
99                              face_Zwischenergebnis = 0;
100                             const labelList ListeDerEdges2(
    FaceBestehtAus[face_labelI]);
101                             List<vector>
    n_edge_VectorSubList(n_edge_VectorList[face_labelI]);
102
103                             for( int edgeI = 0; edgeI <
    ListeDerEdges2.size(); edgeI++)
104                             {
105                                 label edge_labelI(
    ListeDerEdges2[edgeI]);
106                                 scalar local_beta(
    local_gradL_value & n_edge_VectorSubList[edgeI]);
107                                 scalar preFactor(local_beta/
    magGradLface);                  //Oben wurde ausgeschlossen,
    dass magGradLface = 0 ist
108
109                                 face_Zwischenergebnis +=
    IntegratedEdge[edge_labelI]*preFactor;
110                                 error_face_anal = max(
    error_face_anal, max(mag(error_estimate_edge[edge_labelI
    ]*preFactor), mag(eps*face_Zwischenergebnis)));
111                             }     //end for( int edgeI = 0;
    edgeI < ListeDerEdges2.size(); edgeI++)
112                         }     //end if(magGradLface == 0)
113
114
115                         scalar magSf(mag(Sf_ref[face_labelI
    ]));
116                         error_face_num = (   magSf*(magSf*
    df1dL1_local/24*magGradLface  + eps));
117                         // Formel: A/6*df/dL_max*(Lx2 + Ly2)
    *A/4, wobei das letzte A/4 nur ein verstecktes *dx2 ist
    und dabei ein Quadrat als Flaeche angenommen wurde.
118                         // Fehler des numerischen Verfahrens
     + Auswertungsfehler, da IntL nur auf eps genau berechnet
     wird und dann noch mit magSf multipliziert wird.
119
120                         scalar preFactor(local_alpha / (
    local_gradL_value & local_gradL_value));          //Ganz
    oben wurde ausgeschlossen, dass (local_gradL_value &
    local_gradL_value) = 0 ist
121
122                         if(error_face_anal < error_face_num)
123                         {         //analytischer Fall
124                             local_C_test += preFactor*
    face_Zwischenergebnis;
```

```
125                             error_cell_anal = max(
    error_cell_anal, max(mag(error_face_anal*preFactor), mag(
    eps*local_C_test)));
126                         }
127                         else
128                         {       //numerischer Fall
129                             local_C_test += magSf*preFactor
    *(CfromL.IntL(1.0, local_L_value + (local_gradL_value & (
    faceCentre[face_labelI] - cellCentre[cellI]))));
130                             error_cell_anal = max(
    error_cell_anal, max(mag(error_face_num*preFactor), mag(
    eps*local_C_test)));
131                             faceMidPoint_ref[cellI] = 1;
132                         }
133                     }    //end for(int faceI = 0; faceI <
    ListeDerFaces.size(); faceI++)
134
135
136                 } //end if(    mag(df2dL2_local*(
    local_gradL_value & local_gradL_value)*epsilon.value()*
    epsilon.value()) < 0.000000015625)
137
138
139                 error_cell_num = ( Foam::pow(local_volume,
    5.0/3.0)/24.0 *(local_gradL_value & local_gradL_value)*
    df2dL2_local + eps*local_volume) ;
140
141
142                 //Hier kommen die einzigen Unterschiede zu
    initiateC_volaverage.H
143                 if(error_cell_anal > error_cell_num)
144                 {       //numerischer Fall
145                     local_C_test = (CfromL.IntL(0.0,
    local_L_value));   //Einer der wenigen Unterschiede zu
    calculate_C_test2_parallel
146                     error_estimate_cell_ref[cellI] =
    error_cell_num/local_volume;
147                     cellMidPoint_ref[cellI] = 1;
148                     Ly_ref[cellI] = 0;
149                     #pragma omp critical
150                     {
151                         pureLevelset[cellI] = false;
                      //Aus irgendeinem Grund muss das hier
    critical sein. Ich verstehe es nicht.
152                     }
153                 }
154                 else
155                 {       //analytischer Fall
```

```
156                     local_C_test = (local_C_test/
    local_volume);          //Einer der wenigen Unterschiede
    zu calculate_C_test2_parallel
157                     error_estimate_cell_ref[cellI] =
    error_cell_anal/local_volume;
158                 }
159                 C_volaverage_ref[cellI] = local_C_test;
                       //Einer der wenigen Unterschiede zu
    calculate_C_test2_parallel
160
161
```

Programmcode B.13: newGaussFaceIntegrate_parallel.H

```
 1
 2 #include "pureLevelsetLists.H"
 3 #include "harmoniceBC.H"
 4 levelset.correctBoundaryConditions();
 5 C_volaverage.correctBoundaryConditions();
 6 gradL = fvc::grad(levelset);
 7 gradL_face = fvc::interpolate(gradL);
 8 C_face = fvc::interpolate(C_volaverage);
 9 L_face = fvc::interpolate(levelset);
10 int_face *=0;
11
12
13 #pragma omp parallel for schedule(guided) firstprivate(
    L_point, F2_point)
14 for( int faceI = 0; faceI < owner.size(); faceI++)
15 {
16     label local_owner(owner[faceI]);
17     label local_neighbour(neighbour[faceI]);
18
19     if((pureLevelset[local_owner] && pureLevelset[
    local_neighbour]))
20     {
21         vector local_Sf(Sf_ref[faceI]);
22         scalar magSf(mag(local_Sf));
23         vector n_face(local_Sf/magSf);
24
25         vector local_GradL_face(gradL_face_ref[faceI] -
    n_face*(n_face & gradL_face_ref[faceI]));
26         scalar magGradLface(local_GradL_face &
    local_GradL_face);
27         scalar local_L_value(L_face_ref[faceI]);
28
29         scalar error_edge_anal(0.0);
30         scalar error_edge_num(0.0);
31         scalar error_face_num(0.0);
```

```
32          scalar error_face_anal(0.0);
33          scalar face_Zwischenergebnis(0.0);
34
35          scalar df2dL2_local(mag(  CfromL.IntL(-2.0, max(
        CfromL.L_df2dL2_max ,mag(local_L_value)))));          //
        Liege mit dem L-Wert ausserhalb von max(df2dL2)
36          scalar df3dL3_local(mag(  CfromL.IntL(-3.0, max(
        CfromL.L_df3dL3_max ,mag(local_L_value)))));          //
        Hier mache ich einen kleinen Fehler. Siehe Verlauf von f
        '''(L). Max bei L=0
37
38
39          if(magGradLface == 0)    //Wenn der face-Gradient = 0
         ist, kann ich direkt die numerische Integration anwenden
        .
40          {
41              error_face_anal = 10000000000000000;
42              //Fehler der analytischen Int. mag hoeher liegen
        . Hier wird angenommen, dass 10^17 ausreichend ist, damit
         schlussendlich die num. Int. verwendet wird.
43          }
44          else
45          {
46
47              List<vector> n_edge_VectorSubList(
        n_edge_VectorList[faceI]);
48              const labelList ListeDerEdges(FaceBestehtAus[
        faceI]);
49              const labelList ListeDerPunkte(faces[faceI]);
50
51              //Wenn AdvectionCorrection betrieben wird, wird
        vorher einmal getLfromC durchgefuehrt. Innerhalb von
        getLfromC wird pureLevelset geprueft und in jeder Zelle,
        die gradL besitzt, volumenintegriert. => Alle
        Stuetzvektoren sind vorhanden, wenn beide Nachbarn
        pureLevelset sind.
52
53              for( int pointI = 0; pointI < ListeDerPunkte.
        size(); pointI++)
54              {
55                  label point_labelI(ListeDerPunkte[pointI]);
56                  L_point[point_labelI] = (local_GradL_face &
        (points[point_labelI] - faceCentre[faceI])) +
        local_L_value;
57                  F2_point[point_labelI] = CfromL.IntL(2.0,
        L_point[point_labelI]);
58                  //Die F2-Werte berechne ich schon einmal auf
         Vorrat. Es sollte wesenltich seltener vorkommen, dass
```

```
    ein Wert nicht gebraucht wird,
59                  //weil eigenltich stets mag(gradL) = 1/eps
    herrschen sollte. Dann wird F2 auf einem der edges immer
    gebraucht.
60              }
61
62              for( int edgeI = 0; edgeI < ListeDerEdges.size()
    ; edgeI++)
63              {
64                  label edge_labelI(ListeDerEdges[edgeI]);
65                  scalar local_gamma(local_GradL_face &
    n_line_VectorList[edge_labelI]);
66                  scalar local_beta(local_GradL_face &
    n_edge_VectorSubList[edgeI]);
67
68                  if(local_gamma == 0) //Wenn der Line-
    Gradient echt gleich 0 ist, kann ich direkt die
    numeirsche Integration anwenden.
69                  {
70                      error_edge_anal = 10000000000000000;
71                      //Fehler der analytischen Int. mag
    hoeher liegen. Hier wird angenommen, dass 10^17
    ausreichend ist, damit schlussendlich die num. Int.
    verwendet wird.
72                  }
73                  else
74                  {
75                      error_edge_anal = (eps*CfromL.F2_max/mag
    (local_gamma));
76                  }
77                  error_edge_num = (eps*length_edge[
    edge_labelI] + mag(df3dL3_local/4320.0 *pow(local_gamma
    ,4) *length_edge_pow5[edge_labelI])); //Gauss mit 2
    Stuetzstellen
78
79                  scalar preFactor(local_beta/magGradLface);
80                  if(error_edge_anal < error_edge_num)
81                  {    //analytischer Fall
82                      face_Zwischenergebnis += (F2_point[edges
    [edge_labelI].end()] - F2_point[edges[edge_labelI].start
    ()])/local_gamma*preFactor;
83                      error_face_anal = max(error_face_anal,
    max(mag(error_edge_anal*preFactor), mag(eps*
    face_Zwischenergebnis)));
84                  }
85                  else
86                  {    //numerischer Fall
87                      if(mag(local_gamma) < eps)  //weitere
```

```
    Beschleunigung, da fuer gamma kleiner eps auch die
    Mittelpunktsregel ausreichend ist  und man eine FKT-
    Auswertung einspart
88                  {
89                      scalar L_mid((L_point[edges[
    edge_labelI].start()] + L_point[edges[edge_labelI].end()
    ])/2);
90
91                      face_Zwischenergebnis += (CfromL.
    IntL(1.0, L_mid))*preFactor*length_edge[edge_labelI];
92                      error_face_anal = max(
    error_face_anal, max(mag(eps*length_edge[edge_labelI]*
    preFactor), mag(eps*face_Zwischenergebnis)));
93                  }
94                  else
95                  {
96                      scalar L_unten(L_point[edges[
    edge_labelI].start()]*gauss_weight1 + L_point[edges[
    edge_labelI].end()]*gauss_weight2);
97                      scalar L_oben(L_point[edges[
    edge_labelI].start()]*gauss_weight2 + L_point[edges[
    edge_labelI].end()]*gauss_weight1);
98
99                      face_Zwischenergebnis += (CfromL.
    IntL(1.0, L_unten) + CfromL.IntL(1.0, L_oben))*preFactor*
    length_edge[edge_labelI]/2;
100                     error_face_anal = max(
    error_face_anal, max(mag(error_edge_num*preFactor), mag(
    eps*face_Zwischenergebnis)));
101                 }
102             } //end if(error_edge_anal < error_edge_num)
103         }
104     }    //end if((gradL_face_ref[faceI] &
    gradL_face_ref[faceI]) == 0)
105
106
107
108     error_face_num = (  magSf*(magSf/24 *df2dL2_local *
    magGradLface + eps));
109     // Formel: A/6* max(d2f/dL2) * (Lx2 + Ly2)*A/4,
    wobei das letzte A/4 nur ein verstecktes *dx2 ist und
    dabei ein Quadrat als Flaeche angenommen wurde.
110     // Fehler des numerischen Verfahrens +
    Auswertungsfehler, da IntL nur auf eps genau berechnet
    wird und dann noch mit magSf multipliziert wird.
111
112     if(error_face_anal > error_face_num)
113     {       //numerischer Fall
```

```
114             int_face_ref[faceI] = CfromL.IntL(0,
    local_L_value);
115             //error_estimate_face[faceI] = error_face_num/
    magSf;                  //error_estimate_face[faceI] wird
    nachher nicht weiter verwendet und auch nicht als File
    niedergeschrieben
116         }
117         else
118         {
119             int_face_ref[faceI] = face_Zwischenergebnis/
    magSf;
120             //error_estimate_face[faceI] = error_face_anal/
    magSf;                //error_estimate_face[faceI] wird
    nachher nicht weiter verwendet und auch nicht als File
    niedergeschrieben
121         }
122         C_face_ref[faceI] = int_face_ref[faceI];
123
124     }     //end if((pureLevelset[owner[faceI]] &&
    pureLevelset[neighbour[faceI]]))
125 }     //end for( int faceI = 0; faceI < owner.size(); faceI
    ++)
```

Programmcode B.14: pureLevelsetLists.H

```
1  pureLevelset = true;
2  Ly=1;
3
4  //#pragma omp parallel for schedule(guided)      //Aus
       irgendeinem Grund fuehrt diese Parallelisierung zu
       abweichenden Ergebnissen zwischen seriell und parallel
5  for(int faceI = 0; faceI < owner.size(); faceI++)
6  {
7      if (mag(C_volaverage_ref[owner[faceI]]) > CfromL.
       p0_fWert_outer)
8      {
9          pureLevelset[owner[faceI]] = false;
10         pureLevelset[neighbour[faceI]] = false;
11         Ly_ref[owner[faceI]] = 0;
12         Ly_ref[neighbour[faceI]] = 0;
13     }
14     if (mag(C_volaverage_ref[neighbour[faceI]]) > CfromL.
       p0_fWert_outer)
15     {
16         pureLevelset[owner[faceI]] = false;
17         pureLevelset[neighbour[faceI]] = false;
18         Ly_ref[owner[faceI]] = 0;
19         Ly_ref[neighbour[faceI]] = 0;
20     }
```

```
21 }
```

Programmcode B.15: resetReferences.H

```
1  //Read in fvSolution from PISO dictionary
2  const cellList& cells = mesh.cells();
3  const faceList& faces = mesh.faces();
4  const edgeList& edges = mesh.edges();
5  const pointField& points = mesh.points();
6  const labelListList& FaceBestehtAus = mesh.faceEdges();
7  const labelListList& ZelleBestehtAus = mesh.cellEdges();

8  const labelListList& cellPoints = mesh.cellPoints();
9
10
11 //faceEdges: Dieses face besteht aus folgenden Edges
12 //edgeFaces: Dieses Edge wird von folgenden Faces benutzt
13 //pointCells: Dieser Punkt wird von folgenden Zellen benutzt
14 //cellPoints: Diese Zelle besteht aus folgenden Punkten
15
16 //Ein face an sich: Liste an Punkten, die dieses Face bilden
17 //Eine cell an sich: Liste an faces, die diese Zelle bilden
18
19
20 const labelList& owner = mesh.owner();
21 const labelList& neighbour = mesh.neighbour();
22 const scalarField& deltaCoeffs_ref = mesh.deltaCoeffs();
23 const surfaceVectorField& Sf_ref = mesh.Sf();
24 const vectorField& cellCentre = mesh.C();
25 const vectorField& faceCentre = mesh.Cf();
26 const scalarField& volume = mesh.V();
27
28 scalarField& L_ref = levelset.internalField();
29 levelset.correctBoundaryConditions();
30 L_face = fvc::interpolate(levelset);
31 scalarField& L_face_ref = L_face.internalField();
32 scalarField& kappa_ref = kappa.internalField();
33 scalarField& inverseR_ref = inverseR.internalField();
34 scalarField& kontrolle_ref = kontrolle.internalField();
35 scalarField& Ableitung_ref = Ableitung.internalField();
36 scalarField& int_face_ref = int_face.internalField();
37 scalarField& C_face_ref = C_face.internalField();
38 vectorField& gradL_ref = gradL.internalField();
39 vectorField& gradC_ref = gradC.internalField();
40 vectorField& gradL_face_ref = gradL_face.internalField();
41 vectorField& nHat_center_ref = nHat_center.internalField();
42 vectorField& U_ref = U.internalField();
43 scalarField& U_Fehl_ref = U_Fehl.internalField();
44 scalarField& error_estimate_cell_ref = error_estimate_cell.
```

```
    internalField();
45
46
47 //Deklaration
48 // C_mid der Konzentrationswert am Zellmittelpunkt. f(L) =
    C_mid
49 // C_volaverage zellgemittelter Konzetrationswert dieser
    Zelle. Ausserdehalb von pureLevelset wird C_volaverage =
    C_mid angenommen, was einen Fehler beinhaltet, der aber
    klein ist.
50 // C_test Wir zur Iteration von L benoetigt. Es ist ein
    zweites Konzentrationsfeld noetig, damit L iteriert
    werden kann, bis C_test = C_volaverage
51
52 scalarField& C_volaverage_ref = C_volaverage.internalField()
    ;
53 scalarField& C_test_ref = C_test.internalField();
54 scalarField& Ly_ref = Ly.internalField();
55 scalarField& visualDirectInterFace_ref =
    visualDirectInterFace.internalField();
56 scalarField& ChemPot_ref = ChemPot.internalField();
57 scalarField& lapC_ref = lapC.internalField();
58 scalarField& lapL_ref = lapL.internalField();
59 //scalarField& lapChemPot_ref = lapChemPot.internalField();
60 surfaceScalarField pressureJump(fvc::interpolate(kappa*kappa
    ));
61 scalarField& pressureJump_ref = pressureJump.internalField()
    ;
62
63
64 PackedBoolList directInterface(mesh.nCells(), false);
65 PackedBoolList pureLevelset(mesh.nCells(), false);
66 Ly = 0;
67
68 List< List<vector> > n_face_VectorList(mesh.cells().size());
69 List< List<vector> > n_edge_VectorList(mesh.faces().size());
70 Field<vector> n_line_VectorList(mesh.edges().size(), vector
    (0.0, 0.0, 0.0));
71
72 PackedBoolList n_face_BoolList(mesh.nCells(), false);   //
    Sind fuer eine Zelle alle Normalenvektoren auf faces,
    edges und lines berechnet, ist dieser Wert true
73 PackedBoolList n_edge_BoolList(mesh.faces().size(), false);
        //Sind fuer ein face alle n_line und alle n_edges
    berechnet, ist das hier true.
74 PackedBoolList n_line_BoolList(mesh.edges().size(), false);
        //Ist fuer dieses Edge n_line berechnet, wird dieser
     Wert true.
```

```
75 scalarField length_edge(mesh.edges().size(), 0.0);
            //Hierin speichere ich bei der Berechnung von
   n_line direkt die (Laenge der Kante) ab
76 scalarField length_edge_pow5(mesh.edges().size(), 0.0);
            //Hierin speichere ich bei der Berechnung von
   n_line direkt die (Laenge der Kante)^5 ab
77 scalarField L_point(mesh.points().size(), 0.0);
            //Hierin speichere ich die L_werte der jeweiligen
    Punkte einer Zelle
78 scalarField F3_point(mesh.points().size(), 0.0);
                    //Hierin speichere ich die f(L,3)-Werte
   der jeweiligen Punkte einer Zelle
79 scalarField F2_point(mesh.points().size(), 0.0);
                    //Hierin speichere ich die f(L,2)-Werte
   der jeweiligen Punkte eines Faces
80
81 scalarField& cellMidPoint_ref = cellMidPoint.internalField()
   ;
82 scalarField& faceMidPoint_ref = faceMidPoint.internalField()
   ;
83 scalarField& edgeGauss_ref = edgeGauss.internalField();
84
85 Field<scalar> IntegratedEdge(mesh.edges().size(), 0.0);
                    //Wird private uebergeben
86 scalarField error_estimate_edge(mesh.edges().size(),0.0);
          //es als surfaceScalarField zu initialisieren mcht
    Probleme mit empty-Patches, da diese keinen Wert haben
   koennen.
87
            //Wird beides auch private uebergeben.
88
89 volScalarField C_volaverage_old(C_volaverage);
90
91 dimensionedScalar dt(runTime.deltaT());
92 scalar eps(pow(10.0,-16));
93 scalar gauss_weight1( (3.0 + Foam::sqrt(3.0))/6.0  );   //
   L_untererStuetzpunkt = L1*weight1 + L2*weight2
94 scalar gauss_weight2( (3.0 - Foam::sqrt(3.0))/6.0  );   //
   L_obererStuetzpunkt = L1*weight2 + L2*weight1
95 vector e_x(1.0, 0, 0);
96 scalar dalpha(0.00001);
97 int i(0);
98 int j(0);
99
100
101
102
```

Programmcode B.16: updateNormalVectors.H

```
1  const labelList& ListeDerFaces = cells[cellI];
                    // eine Objekt der Klasse cell ist eine
   Liste an faces
2  List<vector>& n_face_VectorSubList = n_face_VectorList[cellI
   ];
3
4  for (int faceI = 0; faceI < ListeDerFaces.size(); faceI++)
5  {
6      vector n_face(Sf_ref[ListeDerFaces[faceI]]/mag(Sf_ref[
   ListeDerFaces[faceI]]));
7      n_face_VectorSubList.append(n_face*Foam::sign(n_face & (
   faceCentre[ListeDerFaces[faceI]]-cellCentre[cellI])));
8      if(!n_edge_BoolList[ListeDerFaces[faceI]])
9      {
10         List<vector>& n_edge_VectorSubList =
   n_edge_VectorList[ListeDerFaces[faceI]];
11         const labelList& ListeDerEdges = FaceBestehtAus[
   ListeDerFaces[faceI]];
12         for (int edgeI = 0; edgeI < ListeDerEdges.size();
   edgeI++)
13         {
14             label labelI(ListeDerEdges[edgeI]);
15             const edge& DieseKante = edges[labelI];
16             if(!n_line_BoolList[labelI])
17             {
18                 vector n_line(DieseKante.vec(points));
19                 length_edge[labelI] = mag(n_line);
20                 length_edge_pow5[labelI] = pow(length_edge[
   labelI],5);
21                 n_line /= length_edge[labelI];
22                 n_line_VectorList[labelI] = n_line;
23                 n_line_BoolList[labelI] = true;
24             }
25             vector n_edge(n_face ^ n_line_VectorList[labelI
   ]);
26             n_edge_VectorSubList.append(n_edge*Foam::sign(
   n_edge & (DieseKante.centre(points)-faceCentre[
   ListeDerFaces[faceI]])));
27         }
28         n_edge_BoolList[ListeDerFaces[faceI]] = true;
29     }
30 }
31 n_face_BoolList[cellI] = true;
```

Programmcode B.17: nHatFvPatchVectorField.H

```
1  /*---------------------------------------------------------------------------
2   =========                 |
3   \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
4    \\    /   O peration     |
5     \\  /    A nd           | Copyright (C) 2011 OpenFOAM Foundation
6      \\/     M anipulation  |
7  -------------------------------------------------------------------------------
8  License
9      This file is part of OpenFOAM.
10
11     OpenFOAM is free software: you can redistribute it and/or modify it
12     under the terms of the GNU General Public License as published by
13     the Free Software Foundation, either version 3 of the License, or
14     (at your option) any later version.
15
16     OpenFOAM is distributed in the hope that it will be useful, but WITHOUT
17     ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
18     FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
19     for more details.
20
21     You should have received a copy of the GNU General Public License
22     along with OpenFOAM.  If not, see <http://www.gnu.org/licenses/>.
23
24 Class
25     Foam::gradLFvPatchVectorField
26
27 Description
28     Foam::gradLFvPatchVectorField
29
30 SourceFiles
31     gradLFvPatchVectorField.C
32
33 \*---------------------------------------------------------------------------*/
```

```
#ifndef gradLFvPatchVectorField_H
#define gradLFvPatchVectorField_H

#include "fvPatchFields.H"
#include "fixedValueFvPatchFields.H"

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
    * * * * * * * * //

namespace Foam
{

/*
   ----------------------------------------------------------------------

                    Class nHatFvPatch Declaration
\*----------------------------------------------------------------------
   */

class nHatFvPatchVectorField
:
    public fixedValueFvPatchVectorField
{
    // Private data

        scalar theta_ ;          //Angabe als Gradzahl!!!
   NICHT im Bogenmass. Sollwert des Kontaktwinkels bei L =
   0;

        scalarField alpha_ ;     //Tatsaechlicher
   Kontaktwinkel unter Einbeziehung der Kruemmung. Auch hier
    erfolgt die Angabe als Gradzahl!


public:

    //- Runtime type information
    TypeName("nHat");


    // Constructors

        //- Construct from patch and internal field
        nHatFvPatchVectorField
        (
            const fvPatch&,
            const DimensionedField<vector, volMesh>&
```

```
74          );
75
76          //- Construct from patch, internal field and
    dictionary
77          nHatFvPatchVectorField
78          (
79              const fvPatch&,
80              const DimensionedField<vector, volMesh>&,
81              const dictionary&
82          );
83
84          //- Construct by mapping given
    nHatFvPatchVectorField
85          //  onto a new patch
86          nHatFvPatchVectorField
87          (
88              const nHatFvPatchVectorField&,
89              const fvPatch&,
90              const DimensionedField<vector, volMesh>&,
91              const fvPatchFieldMapper&
92          );
93
94          //- Construct as copy
95          nHatFvPatchVectorField
96          (
97              const nHatFvPatchVectorField&
98          );
99
100         //- Construct and return a clone
101         virtual tmp<fvPatchVectorField> clone() const
102         {
103             return tmp<fvPatchVectorField>
104             (
105                 new nHatFvPatchVectorField(*this)
106             );
107         }
108
109         //- Construct as copy setting internal field
    reference
110         nHatFvPatchVectorField
111         (
112             const nHatFvPatchVectorField&,
113             const DimensionedField<vector, volMesh>&
114         );
115
116         //- Construct and return a clone setting internal
    field reference
117         virtual tmp<fvPatchVectorField> clone
```

```
118         (
119             const DimensionedField<vector, volMesh>& iF
120         ) const
121         {
122             return tmp<fvPatchVectorField>
123             (
124                 new nHatFvPatchVectorField(*this, iF)
125             );
126         }
127
128
129
130     // Member functions
131
132         // Access
133
134             scalar theta() const
135             {
136                 return theta_;
137             }
138
139             scalar& theta()
140             {
141                 return theta_;
142             }
143
144             const scalarField& alpha() const
145             {
146                 return alpha_;
147             }
148
149             //- Return reference to the ref value to allow
    adjustment
150             scalarField& alpha()
151             {
152                 return alpha_;
153             }
154
155
156         //- Update the coefficients associated with the
    patch field
157         virtual void updateCoeffs();
158
159         //- Write
160         virtual void write(Ostream&) const;
161
162
163 };
```

```


// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

} // End namespace Foam

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

#endif

// ************************************************************************* //
```

Programmcode B.18: nHatFvPatchVectorField.C

```
/*---------------------------------------------------------------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     |
    \\  /    A nd           | Copyright (C) 2011 OpenFOAM Foundation
     \\/     M anipulation  |
-------------------------------------------------------------------------------
License
    This file is part of OpenFOAM.

    OpenFOAM is free software: you can redistribute it and/or modify it
    under the terms of the GNU General Public License as published by
    the Free Software Foundation, either version 3 of the License, or
    (at your option) any later version.

    OpenFOAM is distributed in the hope that it will be useful, but WITHOUT
    ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
    FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
    for more details.

```

```
21      You should have received a copy of the GNU General
        Public License
22      along with OpenFOAM.  If not, see <http://www.gnu.org/
        licenses/>.
23
24 \*---------------------------------------------------------------------------
        */
25
26 #include "nHatFvPatchVectorField.H"
27 #include "addToRunTimeSelectionTable.H"
28 #include "volFields.H"
29 #include "surfaceFields.H"
30 #include "fvcMeshPhi.H"
31
32
33 // * * * * * * * * * * * * * * * * * * Constructors  * * * * * *
       * * * * * * * * * * //
34
35 Foam::nHatFvPatchVectorField::
36 nHatFvPatchVectorField
37 (
38     const fvPatch& p,
39     const DimensionedField<vector, volMesh>& iF
40 )
41 :
42     fixedValueFvPatchVectorField(p, iF),
43     theta_(0.0),
44     alpha_(p.size())
45 {}
46
47
48 Foam::nHatFvPatchVectorField::
49 nHatFvPatchVectorField
50 (
51     const nHatFvPatchVectorField& ptf,
52     const fvPatch& p,
53     const DimensionedField<vector, volMesh>& iF,
54     const fvPatchFieldMapper& mapper
55 )
56 :
57     fixedValueFvPatchVectorField(ptf, p, iF, mapper),
58     theta_(ptf.theta_),
59     alpha_(ptf.alpha_)
60 {}
61
62
63 Foam::nHatFvPatchVectorField::
64 nHatFvPatchVectorField
```

```
65 (
66     const fvPatch& p,
67     const DimensionedField<vector, volMesh>& iF,
68     const dictionary& dict
69 )
70 :
71     fixedValueFvPatchVectorField(p, iF),
72     theta_(readScalar(dict.lookup("theta")))
73 {
74     fvPatchVectorField::operator=(vectorField("value", dict,
       p.size()));
75
76     if (dict.found("alpha"))
77     {
78         alpha_ = scalarField("alpha", dict, p.size());
79     }
80     else
81     {
82         alpha_ = scalarField(p.size(), readScalar(dict.
       lookup("theta")));
83         cout << "alpha-entry is missing. Assuming uniform
       alpha-value for initialization" << nl;
84     }
85 }
86
87
88 Foam::nHatFvPatchVectorField::
89 nHatFvPatchVectorField
90 (
91     const nHatFvPatchVectorField& mwvpvf
92 )
93 :
94     fixedValueFvPatchVectorField(mwvpvf),
95     theta_(mwvpvf.theta_),
96     alpha_(mwvpvf.alpha_)
97 {}
98
99
100 Foam::nHatFvPatchVectorField::
101 nHatFvPatchVectorField
102 (
103     const nHatFvPatchVectorField& mwvpvf,
104     const DimensionedField<vector, volMesh>& iF
105 )
106 :
107     fixedValueFvPatchVectorField(mwvpvf, iF),
108     theta_(mwvpvf.theta_),
109     alpha_(mwvpvf.alpha_)
```

```
110 {}
111
112
113 // * * * * * * * * * * * * * * * * * * Member Functions  * * * * *
        * * * * * * * * * //
114
115 void Foam::nHatFvPatchVectorField::updateCoeffs()        //
    routine aus interFoam uebernommen und angepasst
116 {
117     if (updated())
118     {
119         return;
120     }
121
122
123     //Ansatz: Im Prinzip muss der Normalenvektor der PG nur
    auf das Patch kopiert werden und um einen bestimmten
    Winkel gedreht werden, sodass der Kontaktwinkel nun
    stimmt.
124     //Jetzt zur Mathematik dahinter: Vektor auf dem Patch
    soll in die gleiche Richtung zeigen, wie der
    Normalenvektor in der faceCell
125     // => erster Einheitsvektor des Koordinatensystems ist
    der faceNormalenVektor des Patches n_face.
126     //    zweiter Einheitsvektor wird der Richtung nach
    durch den Normalenvektor auf die PG in der faceCell
    vorgegeben. Hieraus den Anteil von n_face
    herausprojezieren und neu normieren
127     //    Dieser zweite EinheitsvektorVektor liegt nun
    tangential auf dem face des Patches auf, n_y
128     //    Dritter Einheitsvektor des Koordinatensystems
    ueber Kreuzprodukt berechenbar, aber gar nicht notwendig.
129     //    Konstruiere aus dem ersten und zweiten
    Einheitsvektor einen Normalenvektor, der im gewuenschten
    Winkel alpha_[i] auf das face steht.
130
131     //n_corrected = cos(alpha_[i])*n_face + sin(alpha_[i])*
    n_y
132
133     // (n_corrected & n_face) = cos(theta_soll)       ->
    Check
134     // |n_corrected|^2 =  (n_corrected & n_corrected) = cos(
    theta_soll)*cos(theta_soll)*(n_face & n_face) + 2*cos(
    theta_soll)*sin(theta_soll)*(n_face & n_y) + sin(
    theta_soll)*sin(theta_soll)*(n_y & n_y)
135     //                                                = cos(
    theta_soll)*cos(theta_soll) * 1 + sin(theta_soll)*sin(
    theta_soll)* 1 = 1
```

```
136     //                                                  ->
        Check
137
138     const vectorField n_face(this->patch().nf());
139     vectorField n_y(patchInternalField()    - (
        patchInternalField() & n_face) * n_face    );
140     n_y = n_y/( mag(n_y) +
        0.000000000000000000000000000000001);
141     vectorField::operator=( Foam::sin(alpha_/180*Foam::
        constant::mathematical::pi)*n_y + Foam::cos(alpha_/180*
        Foam::constant::mathematical::pi)*n_face );
142
143
144     fixedValueFvPatchVectorField::updateCoeffs();
145 }
146
147
148 void Foam::nHatFvPatchVectorField::write(Ostream& os) const
149 {
150     fvPatchVectorField::write(os);
151     os.writeKeyword("theta") << theta_ << token::
        END_STATEMENT << nl;
152     writeEntry("value", os);
153     alpha_.writeEntry("alpha", os);
154 }
155
156
157 // * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
        * * * * * * * * * //
158
159 namespace Foam
160 {
161     makePatchTypeField
162     (
163         fvPatchVectorField,
164         nHatFvPatchVectorField
165     );
166 }
167
168 //
        ************************************************************************
        //
```

Programmcode B.19: phaseFieldExpCFvPatchScalarField.H

```
1 /*
    ---------------------------------------------------------------------------

2   =========                 /
```

```
3   \\      /  F ield          | OpenFOAM: The Open Source CFD
    Toolbox
4    \\    /   O peration      |
5     \\  /    A nd            | Copyright (C) 2011 OpenFOAM
    Foundation
6      \\/     M anipulation   |
7 -------------------------------------------------------------------------------

8 License
9     This file is part of OpenFOAM.
10
11    OpenFOAM is free software: you can redistribute it and/
    or modify it
12    under the terms of the GNU General Public License as
    published by
13    the Free Software Foundation, either version 3 of the
    License, or
14    (at your option) any later version.
15
16    OpenFOAM is distributed in the hope that it will be
    useful, but WITHOUT
17    ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or
18    FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General
    Public License
19    for more details.
20
21    You should have received a copy of the GNU General
    Public License
22    along with OpenFOAM.  If not, see <http://www.gnu.org/
    licenses/>.
23
24 Class
25    Foam::phaseFieldExpCFvPatchScalarField
26
27 Description
28    Foam::phaseFieldExpCFvPatchScalarField
29
30 SourceFiles
31    phaseFieldExpCFvPatchScalarField.C
32
33 \*-------------------------------------------------------------------------------
    */
34
35 #ifndef phaseFieldExpCFvPatchScalarField_H
36 #define phaseFieldExpCFvPatchScalarField_H
37
38 #include "fixedValueFvPatchFields.H"
```

```

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

namespace Foam
{

/*---------------------------------------------------------------------------*\
                                Class phaseFieldExpCFvPatch Declaration
\*---------------------------------------------------------------------------*/

class phaseFieldExpCFvPatchScalarField
:
    public fixedValueFvPatchScalarField
{
    // Private data

        scalar theta_;  //Angabe als Gradzahl!!! NICHT im Bogenmass

        scalar epsilon_;

        scalar alpha_;


public:

    //- Runtime type information
    TypeName("phaseFieldExpC");


    // Constructors

        //- Construct from patch and internal field
        phaseFieldExpCFvPatchScalarField
        (
            const fvPatch&,
            const DimensionedField<scalar, volMesh>&
        );

        //- Construct from patch, internal field and dictionary
        phaseFieldExpCFvPatchScalarField
        (
```

```
80              const fvPatch&,
81              const DimensionedField<scalar, volMesh>&,
82              const dictionary&
83          );
84
85          //- Construct by mapping given
    phaseFieldExpCFvPatchScalarField
86          //  onto a new patch
87          phaseFieldExpCFvPatchScalarField
88          (
89              const phaseFieldExpCFvPatchScalarField&,
90              const fvPatch&,
91              const DimensionedField<scalar, volMesh>&,
92              const fvPatchFieldMapper&
93          );
94
95          //- Construct as copy
96          phaseFieldExpCFvPatchScalarField
97          (
98              const phaseFieldExpCFvPatchScalarField&
99          );
100
101         //- Construct and return a clone
102         virtual tmp<fvPatchScalarField> clone() const
103         {
104             return tmp<fvPatchScalarField>
105             (
106                 new phaseFieldExpCFvPatchScalarField(*this)
107             );
108         }
109
110         //- Construct as copy setting internal field
    reference
111         phaseFieldExpCFvPatchScalarField
112         (
113             const phaseFieldExpCFvPatchScalarField&,
114             const DimensionedField<scalar, volMesh>&
115         );
116
117         //- Construct and return a clone setting internal
    field reference
118         virtual tmp<fvPatchScalarField> clone
119         (
120             const DimensionedField<scalar, volMesh>& iF
121         ) const
122         {
123             return tmp<fvPatchScalarField>
124             (
```

```
125                 new phaseFieldExpCFvPatchScalarField(*this,
    iF)
126             );
127         }
128
129
130     // Member functions
131
132         // Access
133
134             scalar theta() const
135             {
136                 return theta_;
137             }
138
139             scalar& theta()
140             {
141                 return theta_;
142             }
143
144             scalar epsilon() const
145             {
146                 return epsilon_;
147             }
148
149             scalar& epsilon()
150             {
151                 return epsilon_;
152             }
153
154             scalar alpha() const
155             {
156                 return alpha_;
157             }
158
159             scalar& alpha()
160             {
161                 return alpha_;
162             }
163         // Mapping functions
164
165             //- Map (and resize as needed) from self given a
    mapping object
166             virtual void autoMap
167             (
168                 const fvPatchFieldMapper&
169             );
170
```

```
171             //- Reverse map the given fvPatchField onto this
    fvPatchField
172             virtual void rmap
173             (
174                 const fvPatchScalarField&,
175                 const labelList&
176             );
177
178
179         // Evaluation functions
180
181             //- Update the coefficients associated with the
    patch field
182             //  using the given patch total pressure and
    velocity fields
183             virtual void updateCoeffs
184             (
185                 const scalarField& Cp
186             );
187
188             //- Update the coefficients associated with the
    patch field
189             virtual void updateCoeffs();
190
191
192         //- Write
193         virtual void write(Ostream&) const;
194 };
195
196
197 // * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
    * * * * * * * * * * //
198
199 } // End namespace Foam
200
201 // * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
    * * * * * * * * * * //
202
203 #endif
204
205 //
    ************************************************************************
    //
```

Programmcode B.20: phaseFieldExpCFvPatchScalarField.C

```
1 /*
    ------------------------------------------------------------------------
```

```
2   =========                 |
3   \\      /  F ield         | OpenFOAM: The Open Source CFD
    Toolbox
4    \\    /   O peration     |
5     \\  /    A nd           | Copyright (C) 2011 OpenFOAM
    Foundation
6      \\/     M anipulation  |
7 -------------------------------------------------------------------------------

8 License
9     This file is part of OpenFOAM.
10
11    OpenFOAM is free software: you can redistribute it and/
    or modify it
12    under the terms of the GNU General Public License as
    published by
13    the Free Software Foundation, either version 3 of the
    License, or
14    (at your option) any later version.
15
16    OpenFOAM is distributed in the hope that it will be
    useful, but WITHOUT
17    ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or
18    FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General
    Public License
19    for more details.
20
21    You should have received a copy of the GNU General
    Public License
22    along with OpenFOAM.  If not, see <http://www.gnu.org/
    licenses/>.
23
24 \*---------------------------------------------------------------------------
    */
25
26 #include "phaseFieldExpCFvPatchScalarField.H"
27 #include "addToRunTimeSelectionTable.H"
28 #include "fvPatchFieldMapper.H"
29 #include "volFields.H"
30 #include "surfaceFields.H"
31
32
33 // * * * * * * * * * * * * * * * * * * Constructors  * * * * * *
    * * * * * * * * * * //
34
35 Foam::phaseFieldExpCFvPatchScalarField::
    phaseFieldExpCFvPatchScalarField
```

```
36 (
37     const fvPatch& p,
38     const DimensionedField<scalar, volMesh>& iF
39 )
40 :
41     fixedValueFvPatchScalarField(p, iF),
42     theta_(0.0)
43 {
44     const fvMesh& mesh_ref(patch().boundaryMesh().mesh());
45     IOdictionary transportProperties
46     (
47         IOobject
48         (
49             "transportProperties",
50             mesh_ref.time().constant(),
51             mesh_ref,
52             IOobject::MUST_READ_IF_MODIFIED,
53             IOobject::NO_WRITE
54         )
55     );
56     dimensionedScalar epsilon(transportProperties.lookup("
    epsilon"));
57     epsilon_ = epsilon.value();
58
59     const dictionary& expParameterDict = mesh_ref.
    solutionDict().subDict("expParameter");
60     alpha_ = expParameterDict.lookupOrDefault<double>("alpha
    ", 0.0);
61 }
62
63
64
65 Foam::phaseFieldExpCFvPatchScalarField::
    phaseFieldExpCFvPatchScalarField
66 (
67     const fvPatch& p,
68     const DimensionedField<scalar, volMesh>& iF,
69     const dictionary& dict
70 )
71 :
72     fixedValueFvPatchScalarField(p, iF),
73     theta_(readScalar(dict.lookup("theta")))
74 {
75     if (dict.found("value"))
76     {
77         fvPatchField<scalar>::operator=
78         (
79             scalarField("value", dict, p.size())
```

```
80          );
81      }
82      else
83      {
84          fvPatchField<scalar>::operator=(scalarField(p.size()
    , 0));
85          cout << "value-entry is missing. Assuming uniform 0
    as value-entry" << nl;
86      }
87
88      const fvMesh& mesh_ref(patch().boundaryMesh().mesh());
89      IOdictionary transportProperties
90      (
91          IOobject
92          (
93              "transportProperties",
94              mesh_ref.time().constant(),
95              mesh_ref,
96              IOobject::MUST_READ_IF_MODIFIED,
97              IOobject::NO_WRITE
98          )
99      );
100     dimensionedScalar epsilon(transportProperties.lookup("
    epsilon"));
101     epsilon_ = epsilon.value();
102
103     const dictionary& expParameterDict = mesh_ref.
    solutionDict().subDict("expParameter");
104     alpha_ = expParameterDict.lookupOrDefault<double>("alpha
    ", 0.0);
105 }
106
107
108 Foam::phaseFieldExpCFvPatchScalarField::
    phaseFieldExpCFvPatchScalarField
109 (
110     const phaseFieldExpCFvPatchScalarField& ptf,
111     const fvPatch& p,
112     const DimensionedField<scalar, volMesh>& iF,
113     const fvPatchFieldMapper& mapper
114 )
115 :
116     fixedValueFvPatchScalarField(ptf, p, iF, mapper),
117     theta_(ptf.theta_),
118     epsilon_(ptf.epsilon_),
119     alpha_(ptf.alpha_)
120 {}
121
```

```
122
123 Foam::phaseFieldExpCFvPatchScalarField::
        phaseFieldExpCFvPatchScalarField
124 (
125     const phaseFieldExpCFvPatchScalarField& tppsf
126 )
127 :
128     fixedValueFvPatchScalarField(tppsf),
129     theta_(tppsf.theta_),
130     epsilon_(tppsf.epsilon_),
131     alpha_(tppsf.alpha_)
132 {}
133
134
135 Foam::phaseFieldExpCFvPatchScalarField::
        phaseFieldExpCFvPatchScalarField
136 (
137     const phaseFieldExpCFvPatchScalarField& tppsf,
138     const DimensionedField<scalar, volMesh>& iF
139 )
140 :
141     fixedValueFvPatchScalarField(tppsf, iF),
142     theta_(tppsf.theta_),
143     epsilon_(tppsf.epsilon_),
144     alpha_(tppsf.alpha_)
145 {}
146
147
148 // * * * * * * * * * * * * * * * * * * Member Functions  * * * * *
        * * * * * * * * * * //
149
150 void Foam::phaseFieldExpCFvPatchScalarField::autoMap
151 (
152     const fvPatchFieldMapper& m
153 )
154 {
155     fixedValueFvPatchScalarField::autoMap(m);
156 }
157
158
159 void Foam::phaseFieldExpCFvPatchScalarField::rmap
160 (
161     const fvPatchScalarField& ptf,
162     const labelList& addr
163 )
164 {
165     fixedValueFvPatchScalarField::rmap(ptf, addr);
166
```

```
167     const phaseFieldExpCFvPatchScalarField& tiptf =
168         refCast<const phaseFieldExpCFvPatchScalarField>(ptf)
    ;
169 }
170
171
172 void Foam::phaseFieldExpCFvPatchScalarField::updateCoeffs
173 (
174     const scalarField& Lp
175 )
176 {
177     if (updated())
178     {
179         return;
180     }
181
182 //Wert_patchField = patchinternalField - gradient/deltaCoeff
183
184         operator==(
185 min(
186     1.0,
187     max(
188         -1.0,
189          patchInternalField() + alpha_*(mag(
    patchInternalField()) -1.0)*Foam::cos(Foam::constant::
    mathematical::pi*(theta_/180.0))/epsilon_/(patch().
    deltaCoeffs())
190         )
191     )
192 );
193
194
195     fixedValueFvPatchScalarField::updateCoeffs();
196 }
197
198
199 void Foam::phaseFieldExpCFvPatchScalarField::updateCoeffs()
200 {
201     if (updated())
202     {
203         return;
204     }
205
206 //Wert_patchField = patchinternalField - gradient/deltaCoeff
207
208     operator==(
209 min(
210     1.0,
```

```
211     max(
212         -1.0,
213          patchInternalField() + alpha_*(mag(
    patchInternalField()) -1.0)*Foam::cos(Foam::constant::
    mathematical::pi*(theta_/180.0))/epsilon_/(patch().
    deltaCoeffs())
214         )
215     )
216 );
217
218     fixedValueFvPatchScalarField::updateCoeffs();
219 }
220
221
222 void Foam::phaseFieldExpCFvPatchScalarField::write(Ostream&
    os) const
223 {
224     fvPatchScalarField::write(os);
225     os.writeKeyword("theta") << theta_ << token::
    END_STATEMENT << nl;
226     writeEntry("value", os);
227 }
228
229
230 // * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
    * * * * * * * * * //
231
232 namespace Foam
233 {
234     makePatchTypeField
235     (
236         fvPatchScalarField,
237         phaseFieldExpCFvPatchScalarField
238     );
239 }
240
241 //
    ************************************************************************
    //
```

Programmcode B.21: phaseFieldExpLFvPatchScalarField.H

```
1 /*
    ------------------------------------------------------------------------


2   =========                 |
3   \\      /  F ield         | OpenFOAM: The Open Source CFD
    Toolbox
4    \\    /   O peration     |
```

```
5       \\  /    A nd           | Copyright (C) 2011 OpenFOAM
    Foundation
6        \\/     M anipulation  |
7 -------------------------------------------------------------------------------

8 License
9     This file is part of OpenFOAM.
10
11    OpenFOAM is free software: you can redistribute it and/
    or modify it
12    under the terms of the GNU General Public License as
    published by
13    the Free Software Foundation, either version 3 of the
    License, or
14    (at your option) any later version.
15
16    OpenFOAM is distributed in the hope that it will be
    useful, but WITHOUT
17    ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or
18    FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General
    Public License
19    for more details.
20
21    You should have received a copy of the GNU General
    Public License
22    along with OpenFOAM.  If not, see <http://www.gnu.org/
    licenses/>.
23
24 Class
25    Foam::phaseFieldExpLFvPatchScalarField
26
27 Description
28    Foam::phaseFieldExpLFvPatchScalarField
29
30 SourceFiles
31    phaseFieldExpLFvPatchScalarField.C
32
33 \*---------------------------------------------------------------------------
    */
34
35 #ifndef phaseFieldExpLFvPatchScalarField_H
36 #define phaseFieldExpLFvPatchScalarField_H
37
38 #include "fixedValueFvPatchFields.H"
39
40 // * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
    * * * * * * * * * * //
```

```
41
42 namespace Foam
43 {
44
45 /*
       ---------------------------------------------------------------------------

46                                 Class phaseFieldExpLFvPatch
       Declaration
47 \*---------------------------------------------------------------------------
       */
48
49 class phaseFieldExpLFvPatchScalarField
50 :
51     public fixedValueFvPatchScalarField
52 {
53     // Private data
54
55         scalar theta_;           //Angabe als Gradzahl!!!
       NICHT im Bogenmass. Sollwert des Kontaktwinkels bei L =
       0;
56
57         scalar epsilon_;
58
59         scalarField alpha_;     //Tatsaechlicher
       Kontaktwinkel unter Einbeziehung der Kruemmung. Auch hier
        erfolgt die Angabe als Gradzahl!
60
61
62 public:
63
64     //- Runtime type information
65     TypeName("phaseFieldExpL");
66
67
68     // Constructors
69
70         //- Construct from patch and internal field
71         phaseFieldExpLFvPatchScalarField
72         (
73             const fvPatch&,
74             const DimensionedField<scalar, volMesh>&
75         );
76
77         //- Construct from patch, internal field and
       dictionary
78         phaseFieldExpLFvPatchScalarField
79         (
```

```
80              const fvPatch&,
81              const DimensionedField<scalar, volMesh>&,
82              const dictionary&
83          );
84
85          //- Construct by mapping given
    phaseFieldExpLFvPatchScalarField
86          //  onto a new patch
87          phaseFieldExpLFvPatchScalarField
88          (
89              const phaseFieldExpLFvPatchScalarField&,
90              const fvPatch&,
91              const DimensionedField<scalar, volMesh>&,
92              const fvPatchFieldMapper&
93          );
94
95          //- Construct as copy
96          phaseFieldExpLFvPatchScalarField
97          (
98              const phaseFieldExpLFvPatchScalarField&
99          );
100
101         //- Construct and return a clone
102         virtual tmp<fvPatchScalarField> clone() const
103         {
104             return tmp<fvPatchScalarField>
105             (
106                 new phaseFieldExpLFvPatchScalarField(*this)
107             );
108         }
109
110         //- Construct as copy setting internal field
    reference
111         phaseFieldExpLFvPatchScalarField
112         (
113             const phaseFieldExpLFvPatchScalarField&,
114             const DimensionedField<scalar, volMesh>&
115         );
116
117         //- Construct and return a clone setting internal
    field reference
118         virtual tmp<fvPatchScalarField> clone
119         (
120             const DimensionedField<scalar, volMesh>& iF
121         ) const
122         {
123             return tmp<fvPatchScalarField>
124             (
```

```
125                 new phaseFieldExpLFvPatchScalarField(*this,
    iF)
126             );
127         }
128
129
130     // Member functions
131
132         // Access
133
134             scalar theta() const
135             {
136                 return theta_;
137             }
138
139             scalar& theta()
140             {
141                 return theta_;
142             }
143
144             scalar epsilon() const
145             {
146                 return epsilon_;
147             }
148
149             scalar& epsilon()
150             {
151                 return epsilon_;
152             }
153
154             const scalarField& alpha() const
155             {
156                 return alpha_;
157             }
158
159             //- Return reference to the ref value to allow
    adjustment
160             scalarField& alpha()
161             {
162                 return alpha_;
163             }
164
165         // Mapping functions
166
167             //- Map (and resize as needed) from self given a
     mapping object
168             virtual void autoMap
169             (
```

```
170                 const fvPatchFieldMapper&
171             );
172
173             //- Reverse map the given fvPatchField onto this
    fvPatchField
174             virtual void rmap
175             (
176                 const fvPatchScalarField&,
177                 const labelList&
178             );
179
180
181         // Evaluation functions
182
183             //- Update the coefficients associated with the
    patch field
184             //  using the given patch total pressure and
    velocity fields
185             virtual void updateCoeffs
186             (
187                 const scalarField& Cp
188             );
189
190             //- Update the coefficients associated with the
    patch field
191             virtual void updateCoeffs();
192
193
194         //- Write
195         virtual void write(Ostream&) const;
196 };
197
198
199 // * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
    * * * * * * * * * //
200
201 } // End namespace Foam
202
203 // * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
    * * * * * * * * * //
204
205 #endif
206
207 //
    ************************************************************************
    //
```

Programmcode B.22: phaseFieldExpLFvPatchScalarField.C

```
1  /*---------------------------------------------------------------------------
2  =========                 |
3  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
4   \\    /   O peration     |
5    \\  /    A nd           | Copyright (C) 2011 OpenFOAM Foundation
6     \\/     M anipulation  |
7  -------------------------------------------------------------------------------
8  License
9      This file is part of OpenFOAM.
10
11     OpenFOAM is free software: you can redistribute it and/or modify it
12     under the terms of the GNU General Public License as published by
13     the Free Software Foundation, either version 3 of the License, or
14     (at your option) any later version.
15
16     OpenFOAM is distributed in the hope that it will be useful, but WITHOUT
17     ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
18     FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
19     for more details.
20
21     You should have received a copy of the GNU General Public License
22     along with OpenFOAM.  If not, see <http://www.gnu.org/licenses/>.
23
24 \*---------------------------------------------------------------------------*/
25
26 #include "phaseFieldExpLFvPatchScalarField.H"
27 #include "addToRunTimeSelectionTable.H"
28 #include "fvPatchFieldMapper.H"
29 #include "volFields.H"
30 #include "surfaceFields.H"
31
32
33 // * * * * * * * * * * * * * * * * * * Constructors  * * * * * * * * * * * * * * * * * //
```

```
34
35 Foam::phaseFieldExpLFvPatchScalarField::
       phaseFieldExpLFvPatchScalarField
36 (
37     const fvPatch& p,
38     const DimensionedField<scalar, volMesh>& iF
39 )
40 :
41     fixedValueFvPatchScalarField(p, iF),
42     theta_(0.0),
43     alpha_(p.size())
44 {
45     const fvMesh& mesh_ref(patch().boundaryMesh().mesh());
46     IOdictionary transportProperties
47     (
48         IOobject
49         (
50             "transportProperties",
51             mesh_ref.time().constant(),
52             mesh_ref,
53             IOobject::MUST_READ_IF_MODIFIED,
54             IOobject::NO_WRITE
55         )
56     );
57     dimensionedScalar epsilon(transportProperties.lookup("
       epsilon"));
58     epsilon_ = epsilon.value();
59     cout << "erster␣Constructor" << nl;
60 }
61
62
63 Foam::phaseFieldExpLFvPatchScalarField::
       phaseFieldExpLFvPatchScalarField
64 (
65     const fvPatch& p,
66     const DimensionedField<scalar, volMesh>& iF,
67     const dictionary& dict
68 )
69 :
70     fixedValueFvPatchScalarField(p, iF),
71     theta_(readScalar(dict.lookup("theta")))
72 {
73     if (dict.found("value"))
74     {
75         fvPatchField<scalar>::operator=
76         (
77             scalarField("value", dict, p.size())
78         );
```

```
79      }
80      else
81      {
82          fvPatchField<scalar>::operator=(scalarField(p.size()
        , 0));
83          cout << "value-entry is missing. Assuming uniform 0
        as value-entry" << nl;
84      }
85
86      if (dict.found("alpha"))
87      {
88          alpha_ = scalarField("alpha", dict, p.size());
89      }
90      else
91      {
92          alpha_ = scalarField(p.size(), readScalar(dict.
        lookup("theta")));
93          cout << "alpha-entry is missing. Assuming uniform
        alpha-value for initialization" << nl;
94      }
95
96      const fvMesh& mesh_ref(patch().boundaryMesh().mesh());
97      IOdictionary transportProperties
98      (
99          IOobject
100         (
101             "transportProperties",
102             mesh_ref.time().constant(),
103             mesh_ref,
104             IOobject::MUST_READ_IF_MODIFIED,
105             IOobject::NO_WRITE
106         )
107     );
108     dimensionedScalar epsilon(transportProperties.lookup("
        epsilon"));
109     epsilon_ = epsilon.value();
110 }
111
112
113 Foam::phaseFieldExpLFvPatchScalarField::
        phaseFieldExpLFvPatchScalarField
114 (
115     const phaseFieldExpLFvPatchScalarField& ptf,
116     const fvPatch& p,
117     const DimensionedField<scalar, volMesh>& iF,
118     const fvPatchFieldMapper& mapper
119 )
120 :
```

```
121     fixedValueFvPatchScalarField(ptf, p, iF, mapper),
122     theta_(ptf.theta_),
123     epsilon_(ptf.epsilon_),
124     alpha_(ptf.alpha_)
125 {
126 }
127
128
129 Foam::phaseFieldExpLFvPatchScalarField::
    phaseFieldExpLFvPatchScalarField
130 (
131     const phaseFieldExpLFvPatchScalarField& tppsf
132 )
133 :
134     fixedValueFvPatchScalarField(tppsf),
135     theta_(tppsf.theta_),
136     epsilon_(tppsf.epsilon_),
137     alpha_(tppsf.alpha_)
138 {
139 }
140
141
142 Foam::phaseFieldExpLFvPatchScalarField::
    phaseFieldExpLFvPatchScalarField
143 (
144     const phaseFieldExpLFvPatchScalarField& tppsf,
145     const DimensionedField<scalar, volMesh>& iF
146 )
147 :
148     fixedValueFvPatchScalarField(tppsf, iF),
149     theta_(tppsf.theta_),
150     epsilon_(tppsf.epsilon_),
151     alpha_(tppsf.alpha_)
152 {
153 }
154
155
156 // * * * * * * * * * * * * * * * * * * Member Functions  * * * * *
    * * * * * * * * * * //
157
158 void Foam::phaseFieldExpLFvPatchScalarField::autoMap
159 (
160     const fvPatchFieldMapper& m
161 )
162 {
163     fixedValueFvPatchScalarField::autoMap(m);
164 }
165
```

```
166
167 void Foam::phaseFieldExpLFvPatchScalarField::rmap
168 (
169     const fvPatchScalarField& ptf,
170     const labelList& addr
171 )
172 {
173     fixedValueFvPatchScalarField::rmap(ptf, addr);
174
175     const phaseFieldExpLFvPatchScalarField& tiptf =
176         refCast<const phaseFieldExpLFvPatchScalarField>(ptf)
    ;
177 }
178
179
180 void Foam::phaseFieldExpLFvPatchScalarField::updateCoeffs
181 (
182     const scalarField& Lp
183 )
184 {
185     if (updated())
186     {
187         return;
188     }
189
190 //Wert_patchField = patchinternalField - gradient/deltaCoeff
191     //operator==(patchInternalField() - Foam::cos(Foam::
    constant::mathematical::pi*(1 - theta_/180))/epsilon_/(
    patch().deltaCoeffs()));
192
193     /*const fvMesh& mesh_ref(patch().boundaryMesh().mesh());
194     IOdictionary transportProperties
195     (
196         IOobject
197         (
198             "transportProperties",
199             mesh_ref.time().constant(),
200             mesh_ref,
201             IOobject::MUST_READ_IF_MODIFIED,
202             IOobject::NO_WRITE
203         )
204     );
205     dimensionedScalar R(transportProperties.lookup("R"));
206     dimensionedVector mid(transportProperties.lookup("mid"))
    ;
207     scalarField newtheta(     Foam::atan( patch().Cf().
    component(1)/(patch().Cf().component(0)-mid.value().
    component(0)) )    *180.0/Foam::constant::mathematical::pi
```

```
    +90.0   );
208     newtheta = theta_;*/
209     operator==(patchInternalField() + Foam::cos(Foam::
    constant::mathematical::pi*(alpha_/180))/epsilon_/(patch
    ().deltaCoeffs()));
210
211
212     fixedValueFvPatchScalarField::updateCoeffs();
213 }
214
215
216 void Foam::phaseFieldExpLFvPatchScalarField::updateCoeffs()
217 {
218     if (updated())
219     {
220         return;
221     }
222
223 //Wert_patchField = patchinternalField - gradient/deltaCoeff
224     //operator==(patchInternalField() - Foam::cos(Foam::
    constant::mathematical::pi*(1 - theta_/180))/epsilon_/(
    patch().deltaCoeffs()));
225
226     /*const fvMesh& mesh_ref(patch().boundaryMesh().mesh());
227     IOdictionary transportProperties
228     (
229         IOobject
230         (
231             "transportProperties",
232             mesh_ref.time().constant(),
233             mesh_ref,
234             IOobject::MUST_READ_IF_MODIFIED,
235             IOobject::NO_WRITE
236         )
237     );
238     dimensionedScalar R(transportProperties.lookup("R"));
239     dimensionedVector mid(transportProperties.lookup("mid"))
    ;
240     scalarField newtheta(     Foam::atan( patch().Cf().
    component(1)/(patch().Cf().component(0)-mid.value().
    component(0)) )    *180.0/Foam::constant::mathematical::pi
    +90.0   );
241     newtheta = theta_;*/
242
243     operator==(   patchInternalField() + Foam::cos(Foam::
    constant::mathematical::pi*(alpha_/180))/epsilon_/(patch
    ().deltaCoeffs()));
244
```

```
245
246     fixedValueFvPatchScalarField::updateCoeffs();
247 }
248
249
250 void Foam::phaseFieldExpLFvPatchScalarField::write(Ostream&
        os) const
251 {
252     fvPatchScalarField::write(os);
253     os.writeKeyword("theta") << theta_ << token::
        END_STATEMENT << nl;
254     writeEntry("value", os);
255     alpha_.writeEntry("alpha", os);
256 }
257
258
259 // * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
        * * * * * * * * * //
260
261 namespace Foam
262 {
263     makePatchTypeField
264     (
265         fvPatchScalarField,
266         phaseFieldExpLFvPatchScalarField
267     );
268 }
269
270 //
    ************************************************************************
    //
```

Anhang C

Validierung der Volumenintegration in MATLAB

C.1 Volumenintegration auf nicht würfelförmigen Zellgeometrien

Zur Validierung des Algorithmus und Sicherstellung der korrekten Anwendung aller Fehlerschätzer ist die Methode innerhalb der Software MATLAB 2018a implementiert worden, welche die Möglichkeit zur Verfügung stellt, mit mehr als 16 gültigen Stellen zu rechnen. Der hierfür verwendete Code ist in Abschnitt C.2 aufgeführt. Zu gegebenen Werten von L und $\vec{\nabla}L$ ist durch den vorgestellten Algorithmus der zugehörige Wert von $\bar{c}$ berechenbar. Verglichen werden nun die Ergebnisse der vollständig analytischen Integration, welche mit 150 Nachkommastellen Rechengenauigkeit durchgeführt wird, mit den Ergebnissen der Integration unter normaler Gleitkommaarithmetik und Verwendung numerischen Hilfsverfahren bei kleinen Gradienten. Eine Differenz der beiden Integrationswerte resultiert daher ausschließlich aus der begrenzte Anzahl gespeicherter Stellen oder der Anwendung numerischer Integrationsverfahren für einzelne Kanten, Flächen oder die Gesamtzelle. Im Idealfall sollte die Differenz der beiden Werte verschwindend gering sein. Beispielhaft wird dieser Vergleich an einem Würfel, einem Parallelepiped und einem Prisma als Zellgeometrie vorgenommen. Das Prisma ist dabei der typische Anwendungsfall in wedge-Gittern, die bei der Simulation von rotationssymmetrischen Problemen Anwendung finden. Die Referenzgeometrien haben jeweils ein Volumen von 1 und sind in Abbildung C.1 dargestellt. Für den Vergleich wird für L ein Parameterraum von 0 bis 9 verwendet,

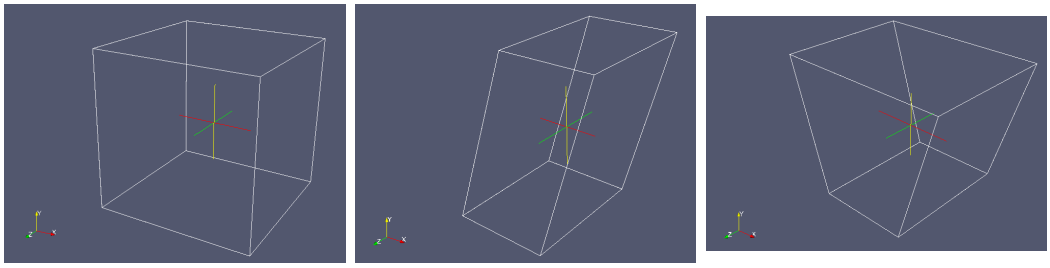


Abbildung C.1: Verwendete Referenzgeometrien zur Validierung der Implementierung des Algorithmus in MATLAB: Würfel (links), Parallelepiped (mitte) und Prisma (rechts).

wobei bis $2,5$ jeweils in Schritten von $0,25$ inkrementiert wird. Zwischen 3 und 9 wird der Raum nur in Schritten von $0,5$ unterteilt. Als Parametersatz für $f(L)$ wird $\alpha = 2$, $\beta = 0,1469702052449879$ und $L_{inner} = 0,8214743858088364$ gewählt. Der Grund für die Wahl dieser Parameter und die Begrenzung von L auf 9 wurde in Abschnitt 3.5 erläutert. Die Anteile von $\vec{\nabla}L$ werden in allen drei Raumdimensionen unabhängig voneinander logarithmisch in Zehnerpotenzen zwischen 10^{-16} und 10^{0} variiert. In allen 117912 Fällen beträgt der Unterschied zwischen der vollständigen analytischen Volumenintegration mit 150 gültigen Stellen und der Anwendung von numerischen Hilsverfahren bei zu kleinen Gradienten unter Gleitkommaarithmetik für die Würfelgeometrie dabei höchstens $4,30 \cdot 10^{-8}$. Der Parallelepiped und das Prisma liegen mit einem maximalen Unterschied von $2,28 \cdot 10^{-8}$ und $7,47 \cdot 10^{-8}$ in der gleichen Größenordnung wie der Würfel. Im Anschluss wurde zudem untersucht, ob der Algorithmus bei einer Skalierung der Geometrie weiterhin zuverlässig arbeitet, da Zellen im Rechengitter keineswegs eine Größe von 1 besitzen. Die Geometrien wurden in allen Raumrichtungen um den Faktor $4 \cdot 10^{-6}$ verkleinert, während alle Gradienten mit dem inversen Wert skaliert wurden. Die maximalen Unterschiede für Würfel, Parallelepiped und Prisma liegen mit $1,44 \cdot 10^{-8}$, $1,78 \cdot 10^{-8}$ und $1,44 \cdot 10^{-8}$ ebenfalls in einer ähnlichen Größenordnung. Der Algorithmus und die Implementierung erscheinen damit validiert. Ebenso erweisen sich damit die Ungenauigkeiten bei der Anwendung von Fehlerschätzern für Würfel und Quadrate auf anderweitig geformte Volumina und Flächen als vernachlässigbar klein.

C.2 Sourcecode der verwendeten m-files

Nachfolgend sind beispielhaft für die verkleinerte Primsengeometrie (siehe Abbildung C.1) die m-files aufgeführt, die zur Validierung des Algorithmus der Volumenintegration auf beliebigen Zellgeometrien genutzt wurden.[1]

Programmcode C.1: calculateKonstanten

```
clear
alpha = vpa(-2.0);


digits(100);
[L_inner, beta] = calculate_parameter(alpha);

L = L_inner;
M = vpa(zeros(5,5));
M(1,:) = [L L^3 L^5 L^7 L^9];
M(2,:) = [1 3*L^2 5*L^4 7*L^6 9*L^8];
M(3,:) = [0 3*2*L 5*4*L^3 7*6*L^5 9*8*L^7];
M(4,:) = [0 3*2 5*4*3*L^2 7*6*5*L^4 9*8*7*L^6];
M(5,:) = [0 0 5*4*3*2*L 7*6*5*4*L^3 9*8*7*6*L^5];

M_inv = inv(M);

```

[1] Sonderzeichen zur Darstellung ersetzt

```
18 D = M_inv(3:5,:);
19 D(1,:) = D(1,:)*5*4*3*2*1;
20 D(2,:) = D(2,:)*7*6*5*4*3*L^2;
21 D(3,:) = D(3,:)*9*8*7*6*5*L^4;
22
23 D(:,2) = D(:,2)*alpha;
24 D(:,3) = D(:,3)*alpha^2;
25 D(:,4) = D(:,4)*alpha^3;
26 D(:,5) = D(:,5)*alpha^4;
27
28 b = vpa(zeros(5,1));
29 b(1) = 1-exp(alpha*L+beta);
30 b(2) = -exp(alpha*L+beta)*alpha;
31 b(3) = -exp(alpha*L+beta)*alpha^2;
32 b(4) = -exp(alpha*L+beta)*alpha^3;
33 b(5) = -exp(alpha*L+beta)*alpha^4;
34
35 koeff = M_inv*b;
36 a1 = koeff(1);
37 a3 = koeff(2);
38 a5 = koeff(3);
39 a7 = koeff(4);
40 a9 = koeff(5);
41
42 L0_inner = L_inner;
43 C1 = L0_inner*L0_inner*(a1/2 + L0_inner*L0_inner*(a3/4 +
       L0_inner*L0_inner*(a5/6 + L0_inner*L0_inner*(a7/8 +
       L0_inner*L0_inner*a9/10)))) - (L0_inner-exp(alpha*
       L0_inner+beta)/alpha);
44 C2 = L0_inner*L0_inner*L0_inner*(a1/2/3 + L0_inner*L0_inner
       *(a3/4/5 + L0_inner*L0_inner*(a5/6/7 + L0_inner*L0_inner
       *(a7/8/9 + L0_inner*L0_inner*a9/10/11)))) - (L0_inner*(C1
       +L0_inner/2)-exp(alpha*L0_inner+beta)/alpha/alpha);
45 C3 = L0_inner*L0_inner*L0_inner*L0_inner*(a1/2/3/4 +
       L0_inner*L0_inner*(a3/4/5/6 + L0_inner*L0_inner*(a5/6/7/8
        + L0_inner*L0_inner*(a7/8/9/10 + L0_inner*L0_inner*a9
       /10/11/12)))) - (L0_inner*(C2+L0_inner*(C1/2+L0_inner/6))
       -exp(alpha*L0_inner+beta)/alpha/alpha/alpha);
46
47
48 save('konstanten.mat','a1','a3','a5','a7','a9','C1','C2','C3
       ','L_inner','beta','alpha');
49 clear
50
51
52
53
54
```

```
55
56 function [L_inner, beta] = calculate_parameter(alpha_num)
57
58 L_inner_out = zeros(size(alpha_num));
59 beta_out = zeros(size(alpha_num));
60
61 for i = 1:length(L_inner_out)
62
63     L_inner = vpa(1);
64
65     digits(100)
66     dL = vpa(0.00000000000000001);
67
68     error = alpha_num(i)^2 + inner_function(L_inner,
    alpha_num(i));
69
70     while (abs(error) > 0.0000000000000000001)
71         %disp(alpha_num(i))
72         %disp(L_inner)
73         AbleitungError = (inner_function(L_inner + dL,
    alpha_num(i))-inner_function(L_inner-dL, alpha_num(i)))
    /2/dL;
74         L_inner = L_inner - error/AbleitungError;
75         error = alpha_num(i)^2 + inner_function(L_inner,
    alpha_num(i));
76     end
77
78
79     beta = final_function(L_inner, alpha_num(i));
80     beta_out(i) = double(beta);
81     L_inner_out(i) = double(L_inner);
82
83 end
84
85 end
86
87 function dfdL0 = inner_function(L_inner, alpha_num)
88
89 digits(100);
90 L = vpa(L_inner);
91 alpha = vpa(alpha_num);
92
93 M = vpa(zeros(5,5));
94 M(1,:) = [L L^3 L^5 L^7 L^9];
95 M(2,:) = [1 3*L^2 5*L^4 7*L^6 9*L^8];
96 M(3,:) = [0 3*2*L 5*4*L^3 7*6*L^5 9*8*L^7];
97 M(4,:) = [0 3*2 5*4*3*L^2 7*6*5*L^4 9*8*7*L^6];
98 M(5,:) = [0 0 5*4*3*2*L 7*6*5*4*L^3 9*8*7*6*L^5];
```

```
99
100 M_inv = inv(M);
101
102 D = M_inv(3:5,:);
103 D(1,:) = D(1,:)*5*4*3*2*1;
104 D(2,:) = D(2,:)*7*6*5*4*3*L^2;
105 D(3,:) = D(3,:)*9*8*7*6*5*L^4;
106
107 D(:,2) = D(:,2)*alpha;
108 D(:,3) = D(:,3)*alpha^2;
109 D(:,4) = D(:,4)*alpha^3;
110 D(:,5) = D(:,5)*alpha^4;
111
112 beta = log(sum(D(:,1))/(sum(sum(D))-alpha^5))-alpha*L;
113
114 b = vpa(zeros(5,1));
115 b(1) = 1-exp(alpha*L+beta);
116 b(2) = -exp(alpha*L+beta)*alpha;
117 b(3) = -exp(alpha*L+beta)*alpha^2;
118 b(4) = -exp(alpha*L+beta)*alpha^3;
119 b(5) = -exp(alpha*L+beta)*alpha^4;
120
121 koeff = M_inv*b;
122
123 dfdL0 = 3*2*koeff(2)/koeff(1);
124
125 end
126
127 function beta = final_function(L_inner, alpha_num)
128
129 digits(100);
130 L = vpa(L_inner);
131 alpha = vpa(alpha_num);
132
133 M = vpa(zeros(5,5));
134 M(1,:) = [L L^3 L^5 L^7 L^9];
135 M(2,:) = [1 3*L^2 5*L^4 7*L^6 9*L^8];
136 M(3,:) = [0 3*2*L 5*4*L^3 7*6*L^5 9*8*L^7];
137 M(4,:) = [0 3*2 5*4*3*L^2 7*6*5*L^4 9*8*7*L^6];
138 M(5,:) = [0 0 5*4*3*2*L 7*6*5*4*L^3 9*8*7*6*L^5];
139
140 M_inv = inv(M);
141
142 D = M_inv(3:5,:);
143 D(1,:) = D(1,:)*5*4*3*2*1;
144 D(2,:) = D(2,:)*7*6*5*4*3*L^2;
145 D(3,:) = D(3,:)*9*8*7*6*5*L^4;
146
```

```
147 D(:,2) = D(:,2)*alpha;
148 D(:,3) = D(:,3)*alpha^2;
149 D(:,4) = D(:,4)*alpha^3;
150 D(:,5) = D(:,5)*alpha^4;
151
152 beta = log(sum(D(:,1))/(sum(sum(D))-alpha^5))-alpha*L;
153
154 end
```

Programmcode C.2: validateAlgorithm

```
 1 digits(150)
 2 test = vpa(1);
 3
 4 clear
 5
 6 L0_values = [ [0:0.25:2.5]  [3:0.5:9]];
 7 Lx_values = 10.^[-16:1:0]*250000;
 8 Ly_values = 10.^[-16:1:0]*250000;
 9 Lz_values = 10.^[-16:1:0]*250000;
10
11
12
13 ergebnis = zeros(5,length(L0_values),length(Lx_values),
      length(Ly_values),length(Lz_values));
14
15 Fehler = zeros(length(L0_values),length(Lx_values),length(
      Ly_values),length(Lz_values));
16 Schaetzer = Fehler;
17 Cell_MidPoint = Fehler*0;
18 Face_MidPoint = Fehler*0;
19 Edge_Gauss = Fehler*0;
20
21 tic
22 parfor i = 1:length(L0_values)
23     disp(i)
24     ergebnis = Zwischenfunktion(L0_values(i),Lx_values,
      Ly_values,Lz_values);
25     Fehler(i,:,:,:) = ergebnis(1,:,:,:);
26     Schaetzer(i,:,:,:) = ergebnis(2,:,:,:);
27     Cell_MidPoint(i,:,:,:) = ergebnis(3,:,:,:);
28     Face_MidPoint(i,:,:,:) = ergebnis(4,:,:,:);
29     Edge_Gauss(i,:,:,:) = ergebnis(5,:,:,:);
30 end
31 toc
32 save('fehlerdaten.mat');
33 disp(max(max(max(max(Fehler)))));
34 disp(max(max(max(max(Schaetzer)))));
```

Programmcode C.3: Zwischenfunktion

```
function ergebnis = Zwischenfunktion(L0, Lx_values,
    Ly_values, Lz_values)
ergebnis = zeros(5,length(Lx_values),length(Ly_values),
    length(Lz_values));



    for j = 1:length(Lx_values)
        for k = 1:length(Ly_values)
            for l = 1:length(Lz_values)
                results = Gesamtintegration_Zelle2(L0,
    Lx_values(j), Ly_values(k), Lz_values(l));
                ergebnis(1,j,k,l) = results(1);
                ergebnis(2,j,k,l) = results(2);
                ergebnis(3,j,k,l) = results(3);
                ergebnis(4,j,k,l) = results(4);
                ergebnis(5,j,k,l) = results(5);
            end
        end
    end
end
```

Programmcode C.4: Gesamtintegration_Zelle2

```
function results = Gesamtintegration_Zelle2(L0,Lx,Ly,Lz)%%
    Setup
    digits(150);
    syms symDelta symLx symLy symLz symL0


    %Die Zelle besteht aus folgenden Punkten
    cellPoints = [1; 2; 3; 4; 5; 6; 7; 8];
    %Die Zelle besteht aus folgenden Faces
    cellFaces = [1; 2; 3; 4; 5; 6];
    %Die Zelle besteht aus folgenden Edges
    cellEdges = [1; 2; 3; 4; 5; 6; 7; 8; 9; 10; 11; 12];
    %Das Face besteht aus folgenden Punkten
    facePoints = [1 2 3 4; 5 6 7 8; 1 2 6 5; 4 3 7 8;1 5 8
    4; 2 6 7 3];
    %Das Face besteht aus folgenden Edges
    faceEdges = [1 2 3 4;5 6 7 8; 1 11 5 12; 3 10 7 9;4 12 8
    9;2 11 6 10];
    %Das Edge besteht aus folgenden Punkten
    edgePoints = [1 2; 2 3; 3 4; 4 1; 5 6; 6 7; 7 8; 8 5;4
    8; 3 7; 2 6; 1 5];

    symDelta = vpa([-0.7 -1.1 -1; 0.7 -1.1 -1; 1.3  0.9 -1;
    -1.3  0.9 -1; -0.7 -1.1  1; 0.7 -1.1  1; 1.3  0.9  1;
```

```
    -1.3   0.9   1])/2/250000;
20   symVolume = vpa((4*10^(-6))^3);
21
22
23   symL0 = vpa(L0);
24   symLx = vpa(Lx);
25   symLy = vpa(Ly);
26   symLz = vpa(Lz);
27   symGradL = [symLx; symLy; symLz];
28
29   delta = double(symDelta);
30   Volume = double(symVolume);
31   L0 = double(symL0);
32   Lx = double(symLx);
33   Ly = double(symLy);
34   Lz = double(symLz);
35   gradL = [Lx; Ly; Lz];
36
37   %Diese Werte wurden zum Einsparen von Rechenzeit direkt
     aus den
38   %Konsolenausgaben von Fallstudie 6 herauskopiert. Es
     sind also exakt
39   %die Werte, mit denen OF auch gerechnet hat. Werte
     haengen von der Wahl
40   %des Profils (alpha; L_inner; beta) ab.
41   L_df2dL2_max = 0.44923834201914203401;
42   F3_max = 101.3324842274769253;
43
44   eps = 10^-16;
45   gauss_weight1 = ( (3.0 + sqrt(3.0))/6.0  );
46   gauss_weight2 = ( (3.0 - sqrt(3.0))/6.0  );
47
48
49   ergebnis = 0;
50   fehlerschaetzer = 0;
51   midPoint_cell = false;
52   midPoint_face = false;
53   gauss_edge = false;
54
55   %Das hier ist effektiv das Mesh. Jede Zeile von
     facePoitnts enthaelt die
56   %Punkte, die dieses Face bilden.
57
58
59   IntegratedEdge = zeros(size(cellEdges,1),1);
60   error_estimate_edge = zeros(size(cellEdges,1),1);
61   error_estimate_face = zeros(size(cellFaces,1),1);
62   error_estimate_cell = 0;
```

```
63      ergebnis = 0;
64      midPoint_cell = false;
65      midPoint_face = false;
66      gauss_edge = false;
67
68
69      %calculate n_face, faceMid and magSf
70      n_face = zeros(size(cellPoints,1),3);
71      magSf = zeros(size(cellFaces,1),1);
72      faceCentre = zeros(size(cellPoints,1),3);
73      for faceI = 1:size(cellFaces,1)
74          koordinaten = zeros(size(facePoints(faceI,:),2),3);
75          for i = 1:size(facePoints(faceI,:),2)
76              koordinaten(i,:) = delta(facePoints(faceI,i),:);
77          end
78
79          vector = cross(koordinaten(2,:)-koordinaten(1,:),
    koordinaten(3,:)-koordinaten(1,:)); %Trifft Annahme, dass
     face eine ebene Flaeche ist
80          vector = vector*sign(vector*koordinaten(1,:)'); %
    Trifft Annahme, dass Ursprung des Koordinatensystems
    innerhalb der Zelle liegt.
81          vector = vector/sqrt(vector*vector');
82          n_face(faceI,:) = vector;
83          [faceCentre(faceI,:), magSf(faceI)] = calcFaceMid(
    koordinaten);
84      end
85      %calculate n_line
86      n_line = zeros(size(cellEdges,1),3);
87      for edgeI = 1:size(cellEdges,1)
88          vector = delta(edgePoints(edgeI,2),:) - delta(
    edgePoints(edgeI,1),:);
89          vector = vector/sqrt(vector*vector');
90          n_line(edgeI,:) = vector;
91      end
92      %calculate n_edge
93      n_edge = cell(size(cellFaces,1),1);
94      for faceI = 1:size(cellFaces,1);
95          submatrix = zeros(size(faceEdges(faceI,:),2),3);
96          for edgeI = 1:size(faceEdges(faceI,:),2)
97              vector = cross(n_face(faceI,:),n_line(faceEdges(
    faceI,edgeI),:));
98              edgeMid = (delta(edgePoints(faceEdges(faceI,
    edgeI),1),:) + delta(edgePoints(faceEdges(faceI,edgeI),2)
    ,:))/2;
99              vector = vector*sign( (edgeMid - faceCentre(
    faceI,:))*vector');
100             vector = vector/sqrt(vector*vector');
```

```
101             submatrix(edgeI,:) = vector;
102         end
103         n_edge{faceI} = submatrix;
104     end
105
106
107     if((gradL*gradL')==0)
108         error_estimate_cell = 10000000000000000;
109     else
110         df1dL1_local = f(L0,-1);
111         df2dL2_local = abs(f(max(abs(L0),L_df2dL2_max),-2));
112         error_estimate_cell = 0;
113
114         L_values = L0 + delta*gradL;
115         F3_values = abs(arrayfun(@f,L_values,3*ones(size(
    L_values))));
116
117         %Loope ueber alle edges und berechne IntegratedEdge
    und
118         %error_estimate_edge
119         for edgeI = 1:size(cellEdges,1)
120             gamma = n_line(edgeI,:)*gradL;
121             if (gamma == 0)
122                 error_edge_anal = 10000000000000000;
123             else
124                 error_edge_anal = eps*F3_max/abs(gamma);
125             end
126             laenge_edge = sqrt(  (delta(edgePoints(edgeI,1)
    ,:) - delta(edgePoints(edgeI,2),:))  *(delta(edgePoints(
    edgeI,1),:) - delta(edgePoints(edgeI,2),:))'  );
127             error_edge_num = eps*laenge_edge + abs(
    df2dL2_local/4320*gamma^4)*laenge_edge^5;
128
129             if (error_edge_anal < error_edge_num)
130                 %analytische Integration des Edges
131                 IntegratedEdge(edgeI) = (  F3_values(
    edgePoints(edgeI,2)) - F3_values(edgePoints(edgeI,1)))/
    gamma;
132                 error_estimate_edge(edgeI) = error_edge_anal
    ;
133             else
134                 %numerische Integration des Edges
135                 if (abs(gamma) < eps)
136                     L_mid = (  L_values(edgePoints(edgeI,2))
     + L_values(edgePoints(edgeI,1)))/2;
137                     IntegratedEdge(edgeI) = laenge_edge*f(
    L_mid,2);
138                     error_estimate_edge(edgeI) = eps*
```

```
    laenge_edge;
139                 else
140                     L_unten = gauss_weight1*L_values(
    edgePoints(edgeI,2)) + gauss_weight2*L_values(edgePoints(
    edgeI,1));
141                     L_oben = gauss_weight2*L_values(
    edgePoints(edgeI,2)) + gauss_weight1*L_values(edgePoints(
    edgeI,1));
142                     IntegratedEdge(edgeI) = laenge_edge*(f(
    L_unten,2) + f(L_oben,2))/2;
143                     error_estimate_edge(edgeI) =
    error_edge_num;
144                     gauss_edge = true;
145                 end
146             end
147         end
148
149         %Loope ueber alle faces und berechne die
150         for faceI = 1:size(cellFaces,1)
151             alpha = n_face(faceI,:)*gradL;
152             magGradLface = ( gradL'*gradL - alpha^2);
153
154             if(magGradLface == 0)
155                 error_estimate_face(faceI) =
    10000000000000000;
156             else
157                 face_Zwischenergebnis = 0;
158                 error_estimate_face(faceI) = 0;
159                 submatrix = n_edge{faceI,1};
160                 for edgeI = 1:size(faceEdges(faceI,:),2)
161                     beta = submatrix(edgeI,:)*gradL;
162                     preFactor = beta/magGradLface;
163                     face_Zwischenergebnis =
    face_Zwischenergebnis + preFactor*IntegratedEdge(
    faceEdges(faceI,edgeI));
164                     error_estimate_face(faceI) = max(
    error_estimate_face(faceI), max(abs(error_estimate_edge(
    faceEdges(faceI,edgeI))*preFactor), abs(eps*
    face_Zwischenergebnis)));
165                 end
166             end
167             error_face_num = magSf(faceI)*(eps + magSf(faceI
    )*df1dL1_local/24*magGradLface);
168             preFactor = alpha/(gradL'*gradL);
169             if (error_estimate_face(faceI) < error_face_num)
170                 ergebnis = ergebnis + preFactor*
    face_Zwischenergebnis;
171                 error_estimate_cell = max(
```

```
    error_estimate_cell, max(abs(error_estimate_face(faceI)*
    preFactor), abs(eps*ergebnis)));
172             else
173                 ergebnis = ergebnis + magSf(faceI)*preFactor
    *f(L0 + faceCentre(faceI,:)*gradL,1);
174                             error_estimate_cell = max(
    error_estimate_cell, max(abs(error_face_num*preFactor),
    abs(eps*ergebnis)));
175                 midPoint_face = true;
176             end
177         end
178         error_cell_num = ( Volume^(5/3)/24.0 *(gradL' *
    gradL)*df2dL2_local + eps*Volume);
179
180         if(error_estimate_cell > error_cell_num)
181                         %ganze Zelle numerisch
182                     ergebnis = f(L0,0)*Volume;
183                     error_estimate_cell = error_cell_num;
184                     midPoint_cell = true;
185         end
186     end
187
188
189     %%
190
191     ergebnis3 = 0;
192
193     symL_values = symL0 + symDelta*symGradL;
194     symF3_values = abs(arrayfun(@f,symL_values,3*ones(size(
    symL_values))));
195
196     %Loope ueber alle edges und berechne IntegratedEdge und
197     %error_estimate_edge
198     for edgeI = 1:size(cellEdges,1)
199         gamma = n_line(edgeI,:)*symGradL;
200         symIntegratedEdge(edgeI,1) = (  symF3_values(
    edgePoints(edgeI,2)) - symF3_values(edgePoints(edgeI,1)))
    /gamma;
201     end
202
203     %Loope ueber alle faces und berechne die
204     for faceI = 1:size(cellFaces,1)
205         alpha = n_face(faceI,:)*symGradL;
206         symMagGradLface = ( symGradL'*symGradL - alpha^2);
207
208         sym_face_Zwischenergebnis = vpa(0);
209         submatrix = n_edge{faceI,1};
210         for edgeI = 1:size(faceEdges(faceI,:),2)
```

```
211             beta = submatrix(edgeI,:)*symGradL;
212             preFactor = beta/symMagGradLface;
213             sym_face_Zwischenergebnis =
    sym_face_Zwischenergebnis + preFactor*symIntegratedEdge(
    faceEdges(faceI,edgeI));
214         end
215
216         preFactor = alpha/(symGradL'*symGradL);
217         ergebnis3 = ergebnis3 + preFactor*
    sym_face_Zwischenergebnis;
218     end
219
220     disp(double(ergebnis/Volume));
221     disp(double(ergebnis3/Volume));
222
223     results = [ abs(double((ergebnis - ergebnis3)/Volume))
    abs(error_estimate_cell/Volume) midPoint_cell
    midPoint_face gauss_edge];
```

Programmcode C.5: f

```
1  function ergebnis = f(L,order)
2
3  %Wie in Fallstudie 6 vom Cluster
4  %beta = 0.14697020524498 79;          //Absolutglied im
       Exponenten
5  %L0_inner = 0.8214743858088364;       //Wechsel von Polynom
       auf Exp
6  %L0_outer = 9;                        //Begrenzung der
       Volumenintegration
7
8
9  %Diese Werte stammen direkt aus fvSolution aus Fallstudie 6.
10 L_inner = 0.8214743858088364;
11 alpha = -2.0;
12 beta = 0.1469702052449879;
13
14
15 if (isnumeric(L))
16
17
18     if (abs(L) > L_inner)
19         %Diese Werte habe ich aus den Konsolenausgaben von
       Fallstudie 6 herauskopiert; es sind also exakt die Wert,
       mit denen OF intern gerechnet hat.
20         C1 = -0.55822222657321263206;
21         C2 =  0.28722709817673897081;
22         C3 = -0.14455948210376459717;
23         switch order
```

```
24            case 3
25                 ergebnis = ((C3 + abs(L)*(C2 + abs(L)*(C1/2
     + abs(L)/6)) - exp(alpha*abs(L)+beta)/(alpha*alpha*alpha
     )));
26            case 2
27                ergebnis = sign(L)*(     C2 + abs(L)*(C1 +
     abs(L)/2)  - exp(alpha*abs(L) + beta)/(alpha*alpha)   );
28            case 1
29                ergebnis = abs(L)-exp(alpha*abs(L) + beta)/
     alpha + C1;
30            case 0
31                ergebnis = (1-exp(alpha*abs(L) + beta))*sign
     (L);
32            case -1
33                ergebnis = -alpha*exp(alpha*abs(L) + beta);
34            case -2
35                ergebnis = -alpha*alpha*sign(L)*exp(alpha*
     abs(L) + beta);
36            case -3
37                ergebnis = -alpha*alpha*alpha*exp(alpha*abs(
     L) + beta);
38        end
39     else
40
41        %Diese Werte habe ich aus den Konsolenausgaben von
     Fallstudie 6 herauskopiert; es sind also exakt die Wert,
     mit denen OF intern gerechnet hat.
42        a1 =  1.3378668144719525479;
43        a3 = -0.89191120964796499759;
44        a5 =  0.62605419216357993673;
45        a7 = -0.29063495726888988191;
46        a9 =  0.061911006025051826263;
47
48
49        L_square = L*L;
50        switch order
51            case 3
52                    ergebnis = (L_square*L_square*(a1/24 +
     L_square*(a3/120 + L_square*(a5/336 + L_square*(a7/720 +
     L_square*a9/1320)))));
53            case 2
54                ergebnis = L*L_square*(a1/6 + L_square*(a3
     /20 + L_square*(a5/42 + L_square*(a7/72 + L_square*a9
     /110))));
55            case 1
56                ergebnis = L_square*(a1/2 + L_square*(a3/4 +
      L_square*(a5/6 + L_square*(a7/8 + L_square*a9/10))));
57            case 0
```

```
58                  ergebnis = L*(a1 + L_square*(a3 + L_square*(
    a5 + L_square*(a7 + L_square*a9))));
59              case -1
60                  ergebnis = a1 + L_square*(3*a3 + L_square
    *(5*a5 + L_square*(7*a7 + L_square*9*a9)));
61              case -2
62                  ergebnis = L*(6*a3 + L_square*(20*a5 +
    L_square*(42*a7 + L_square*72*a9)));
63              case -3
64                  ergebnis = 6*a3 + L_square*(60*a5 + L_square
    *(210*a7 + L_square*504*a9));
65          end
66      end
67  else
68      load('konstanten.mat');
69      if (abs(L) > L_inner)
70          switch order
71              case 3
72                   ergebnis = ((C3 + abs(L)*(C2 + abs(L)*(C1/2
     + abs(L)/6)) - exp(alpha*abs(L)+beta)/(alpha*alpha*alpha
    )));
73              case 2
74                  ergebnis = sign(L)*(     C2 + abs(L)*(C1 +
    abs(L)/2)  - exp(alpha*abs(L) + beta)/(alpha*alpha)   );
75              case 1
76                  ergebnis = abs(L)-exp(alpha*abs(L) + beta)/
    alpha + C1;
77              case 0
78                  ergebnis = (1-exp(alpha*abs(L) + beta))*sign
    (L);
79              case -1
80                  ergebnis = -alpha*exp(alpha*abs(L) + beta);
81              case -2
82                  ergebnis = -alpha*alpha*sign(L)*exp(alpha*
    abs(L) + beta);
83              case -3
84                  ergebnis = -alpha*alpha*alpha*exp(alpha*abs(
    L) + beta);
85          end
86      else
87          L_square = L*L;
88          switch order
89              case 3
90                      ergebnis = (L_square*L_square*(a1/24 +
    L_square*(a3/120 + L_square*(a5/336 + L_square*(a7/720 +
    L_square*a9/1320)))));
91              case 2
92                  ergebnis = L*L_square*(a1/6 + L_square*(a3
```

```
    /20 + L_square*(a5/42 + L_square*(a7/72 + L_square*a9
    /110))));
93              case 1
94                  ergebnis = L_square*(a1/2 + L_square*(a3/4 +
     L_square*(a5/6 + L_square*(a7/8 + L_square*a9/10))));
95              case 0
96                  ergebnis = L*(a1 + L_square*(a3 + L_square*(
    a5 + L_square*(a7 + L_square*a9))));
97              case -1
98                  ergebnis = a1 + L_square*(3*a3 + L_square
    *(5*a5 + L_square*(7*a7 + L_square*9*a9)));
99              case -2
100                 ergebnis = L*(6*a3 + L_square*(20*a5 +
    L_square*(42*a7 + L_square*72*a9)));
101             case -3
102                 ergebnis = 6*a3 + L_square*(60*a5 + L_square
    *(210*a7 + L_square*504*a9));
103         end
104     end
105 end
```

Programmcode C.6: calcFaceMid

```
1  function [faceMid, A] = calcFaceMid(koordinaten)
2
3  sumN = [0 0 0];
4  sumA = 0;
5  sumAc = [0 0 0];
6  faceMid = sumN;
7
8  for i = 1:length(koordinaten)
9      faceMid = faceMid + koordinaten(i,:);
10 end
11 faceMid = faceMid / length(koordinaten);
12
13 for i = 1:length(koordinaten)
14     centerTriangle = (koordinaten(i,:) + koordinaten(1+mod(i
    , length(koordinaten)),:) + faceMid)/3;
15     areaVector = cross( (koordinaten(i,:) - faceMid) , (
    koordinaten(1+ mod(i, length(koordinaten)),:) - faceMid)
       )/2;
16     sumA = sumA + sqrt(areaVector*areaVector');
17     sumAc = sumAc + sqrt(areaVector*areaVector')*
    centerTriangle;
18 end
19
20 faceMid = sumAc/sumA;
21 A = sumA;
22
```

```
23
24 end
```

Anhang D

Parameter des Solvers interFoam

Der Solver interFoam stellt für die Simulation von kapillarkraftdominierten Mehrphasenströmungen nach dem volume-tracking-Verfahren den Standardsolver von OpenFOAM dar. In Abschnitt 3.9 wurde im Solver interFoam von OpenFOAM 6[1] zum besseren Vergleich die Berechnung von U_{fehl} nachimplementiert. Der resultierende Code ist nachfolgend aufgeführt.[2] OpenFOAM selbst, worauf der hier dargestellte Code basiert, steht unter der GNU General Public Licence.[3] Jegliche daraus entstehende Software steht daher ebenfalls unter der entsprechenden GNU General Public Licence. Die Einstellungen des Solvers orientieren sich an den empfohlenen Einstellungen für die in OpenFOAM 6 enthaltenen Tutorialfälle *capillaryRise* und *damBreak* und sind ebenfalls dargestellt.

Programmcode D.1: ownInterFoam.C

```
1  /*
   ---------------------------------------------------------------------------
2    =========                 |
3    \\      /  F ield         | OpenFOAM: The Open Source CFD
     Toolbox
4     \\    /   O peration     | Website:  https://openfoam.org
5      \\  /    A nd           | Copyright (C) 2011-2018
     OpenFOAM Foundation
6       \\/     M anipulation  |
7  ---------------------------------------------------------------------------

8  License
9      This file is part of OpenFOAM.
10
11     OpenFOAM is free software: you can redistribute it and/
     or modify it
12     under the terms of the GNU General Public License as
     published by
13     the Free Software Foundation, either version 3 of the
     License, or
```

[1] Download über https://openfoam.org/download/archive/
[2] Sonderzeichen zur Darstellung ersetzt
[3] siehe https://openfoam.org/licence/

```
14      (at your option) any later version.
15
16      OpenFOAM is distributed in the hope that it will be
        useful, but WITHOUT
17      ANY WARRANTY; without even the implied warranty of
        MERCHANTABILITY or
18      FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General
        Public License
19      for more details.
20
21      You should have received a copy of the GNU General
        Public License
22      along with OpenFOAM.  If not, see <http://www.gnu.org/
        licenses/>.
23
24  Application
25      interFoam
26
27  Description
28      Solver for 2 incompressible, isothermal immiscible
        fluids using a VOF
29      (volume of fluid) phase-fraction based interface
        capturing approach,
30      with optional mesh motion and mesh topology changes
        including adaptive
31      re-meshing.
32
33  \*---------------------------------------------------------------------------
        */
34
35  #include "fvCFD.H"
36  #include "dynamicFvMesh.H"
37  #include "CMULES.H"
38  #include "EulerDdtScheme.H"
39  #include "localEulerDdtScheme.H"
40  #include "CrankNicolsonDdtScheme.H"
41  #include "subCycle.H"
42  #include "immiscibleIncompressibleTwoPhaseMixture.H"
43  #include "turbulentTransportModel.H"
44  #include "pimpleControl.H"
45  #include "fvOptions.H"
46  #include "CorrectPhi.H"
47  #include "fvcSmooth.H"
48
49  // * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
        * * * * * * * * * * //
50
51  int main(int argc, char *argv[])
```

```
52 {
53     #include "postProcess.H"
54
55     #include "setRootCaseLists.H"
56     #include "createTime.H"
57     #include "createDynamicFvMesh.H"
58     #include "initContinuityErrs.H"
59     #include "createDyMControls.H"
60     #include "createFields.H"
61     #include "createAlphaFluxes.H"
62     #include "initCorrectPhi.H"
63     #include "createUfIfPresent.H"
64
65     turbulence->validate();
66
67     if (!LTS)
68     {
69         #include "CourantNo.H"
70         #include "setInitialDeltaT.H"
71     }
72
73     // * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
     * * * * * * * * * //
74     Info<< "\nStarting time loop\n" << endl;
75
76     dimensionedScalar numStab("numStab",dimensionSet( 0, 1,
    0, 0, 0, 0, 0),1e-16);
77     scalar U0(pimple.dict().lookupOrDefault<scalar>("U0"
    ,123456789));
78     vector e_x(1, 0, 0);
79     Info << "U0 = " << U0 << nl;
80
81
82     while (runTime.run())
83     {
84         #include "readDyMControls.H"
85
86         if (LTS)
87         {
88             #include "setRDeltaT.H"
89         }
90         else
91         {
92             #include "CourantNo.H"
93             #include "alphaCourantNo.H"
94             #include "setDeltaT.H"
95         }
96
```

```
97          runTime++;
98
99          Info<< "Time = " << runTime.timeName() << nl << endl
    ;
100
101         // --- Pressure-velocity PIMPLE corrector loop
102         while (pimple.loop())
103         {
104             if (pimple.firstIter() ||
    moveMeshOuterCorrectors)
105             {
106                 mesh.update();
107
108                 if (mesh.changing())
109                 {
110                     // Do not apply previous time-step mesh
    compression flux
111                     // if the mesh topology changed
112                     if (mesh.topoChanging())
113                     {
114                         talphaPhi1Corr0.clear();
115                     }
116
117                     gh = (g & mesh.C()) - ghRef;
118                     ghf = (g & mesh.Cf()) - ghRef;
119
120                     MRF.update();
121
122                     if (correctPhi)
123                     {
124                         // Calculate absolute flux
125                         // from the mapped surface velocity
126                         phi = mesh.Sf() & Uf();
127
128                         #include "correctPhi.H"
129
130                         // Make the flux relative to the
    mesh motion
131                         fvc::makeRelative(phi, U);
132
133                         mixture.correct();
134                     }
135
136                     if (checkMeshCourantNo)
137                     {
138                         #include "meshCourantNo.H"
139                     }
140                 }
```

```
141             }
142
143             #include "alphaControls.H"
144             #include "alphaEqnSubCycle.H"
145
146             mixture.correct();
147
148             #include "UEqn.H"
149
150             // --- Pressure corrector loop
151             while (pimple.correct())
152             {
153                 #include "pEqn.H"
154             }
155
156             if (pimple.turbCorr())
157             {
158                 turbulence->correct();
159             }
160         }
161
162         UFehl*=0;
163         gradAlpha = fvc::grad(alpha1);
164
165         forAll(mesh.owner(), faceI)
166         {
167             label ownerCellI(mesh.owner()[faceI]);
168             label neighbourCellI(mesh.neighbour()[faceI]);
169
170             scalar alphaOnwer(alpha1[ownerCellI]);
171             scalar alphaNeighbour(alpha1[neighbourCellI]);

172
173             if((alphaOnwer-0.5)*(alphaNeighbour-0.5) < 0)
174             {
175                 vector U1(U[ownerCellI]);
176                 vector U2(U[neighbourCellI]);
177
178                 //In calculateFLuxes habe ich gradL
    interpoliert und nicht nHat. Das ist das gleiche, wenn
    gradL1 und gradL2 den gleichen Betrag haben.
179                 //Fuer gradL wird das wohl ganz gut erfuellt
     gewesen sein. Fuer gradAlpha auf keinen Fall. Daher hier
     erst nHat berechnen und den interpolieren.
180
181                 vector n1(gradAlpha[ownerCellI]/(mag(
    gradAlpha[ownerCellI]) + 1e-16));
182                 vector n2(gradAlpha[neighbourCellI]/(mag(
```

```
    gradAlpha[neighbourCellI]) + 1e-16));
183
184                 scalar xHat( (0.5 - alphaOnwer)  / (
    alphaNeighbour - alphaOnwer ));
185
186                 vector nMix(n1 + xHat*(n2 - n1));
187                 nMix = nMix/mag(nMix);
188                 vector UMix(U1 + xHat*(U2 - U1) - U0*e_x);
189
190                 scalar localUFehl(mag(nMix & UMix));
191
192                 UFehl[ownerCellI] = max(UFehl[ownerCellI],
    localUFehl);
193                 UFehl[neighbourCellI] = max(UFehl[
    neighbourCellI], localUFehl);
194             }
195         }
196
197         forAll(alpha1.boundaryField(), patchI)
198         {
199             const fvPatchScalarField& patchField_ref(alpha1.
    boundaryField()[patchI]);
200             if(patchField_ref.coupled())
201             {
202                 const labelList faceCells(patchField_ref.
    patch().faceCells());
203                 scalarField alphaNeighbourField(alpha1.
    boundaryField()[patchI].patchNeighbourField());
204                 vectorField gradAlphaNeighbourField(
    gradAlpha.boundaryField()[patchI].patchNeighbourField());
205                 vectorField UNeighbourField(U.boundaryField
    ()[patchI].patchNeighbourField());
206
207                 forAll(faceCells, faceI)
208                 {
209                     label ownerCellI(faceCells[faceI]);
210
211                     scalar alphaOnwer(alpha1[ownerCellI]);

212                     scalar alphaNeighbour(
    alphaNeighbourField[faceI]);
213
214                     if((alphaOnwer-0.5)*(alphaNeighbour-0.5)
    < 0)
215                     {
216                         vector U1(U[ownerCellI]);
217                         vector U2(UNeighbourField[faceI]);
218
```

```
219                         //In calculateFLuxes habe ich gradL interpoliert und nicht nHat. Das ist das gleiche, wenn gradL1 und gradL2 den gleichen Betrag haben.
220                         //Fuer gradL wird das wohl ganz gut erfuellt gewesen sein. Fuer gradAlpha auf keinen Fall. Daher hier erst nHat berechnen und den interpolieren.
221
222                         vector n1(gradAlpha[ownerCellI]/(mag(gradAlpha[ownerCellI]) + 1e-16));
223                         vector n2(gradAlphaNeighbourField[faceI]/(mag(gradAlphaNeighbourField[faceI]) + 1e-16));
224
225                         scalar xHat( (0.5 - alphaOnwer) / ( alphaNeighbour - alphaOnwer ));
226
227                         vector nMix(n1 + xHat*(n2 - n1));
228                         nMix = nMix/mag(nMix);
229                         vector UMix(U1 + xHat*(U2 - U1) - U0*e_x);
230
231                         scalar localUFehl(mag(nMix & UMix));
232
233                         UFehl[ownerCellI] = max(UFehl[ownerCellI], localUFehl);
234                     }   //end if
235                 }       //end forAll(faceCells)
236             }           //end if(coupled)
237         }               //end forAll (patchI)
238
239
240         Info << "max(mag(U)) = " << max(mag(U)) << nl;
241
242         runTime.write();
243
244         Info<< "ExecutionTime = " << runTime.elapsedCpuTime() << " s"
245             << "  ClockTime = " << runTime.elapsedClockTime() << " s"
246             << nl << endl;
247     }
248
249     Info<< "End\n" << endl;
250
251     return 0;
252 }
253
254
255 //
```

```
    ********************************************************************************
    //
```

Programmcode D.2: controlDict

```
1  /*--------------------------------*- C++
   -*----------------------------------*\
2  | =========                 |
                  |
3  | \\      /  F ield         | OpenFOAM: The Open Source CFD
   Toolbox           |
4  |  \\    /   O peration     | Version:  2.1.1
                  |
5  |   \\  /    A nd           | Web:      www.OpenFOAM.org
                  |
6  |    \\/     M anipulation  |
                  |
7  \*---------------------------------------------------------------------------
   */
8  FoamFile
9  {
10     version     2.0;
11     format      ascii;
12     class       dictionary;
13     location    "system";
14     object      controlDict;
15 }
16 // * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
   * * * * * * * * * //
17 //libs ("PIDvelocity.so");
18
19 application          interFoam;
20
21 startFrom            latestTime;
22
23 startTime            0;
24
25 stopAt               endTime;
26
27 endTime              0.16; //Stopzeit
28
29 deltaT               1e-06;     //Zeitschrittweite
30
31 writeControl         timeStep;
32
33 writeInterval        1000;
34
35 purgeWrite           2;
36
```

```
writeFormat             ascii;

writePrecision          10;

writeCompression        off;

timeFormat              general;

timePrecision           12;

runTimeModifiable       true;

maxCo                   0.2;

maxAlphaCo              0.2;

maxDeltaT               1e-6;

functions
{
    volFieldValue1
{
    type            volFieldValue;
    libs            ("libfieldFunctionObjects.so");

    // Mandatory entries (runtime modifiable)
    fields          (UFehl);
    operation       max;
    regionType      all;

    writeFields     false;


    executeControl  timeStep;
    executeInterval 1;
    writeControl    timeStep;
    writeInterval   1;
}



}


//
    ************************************************************************
    //
```

Programmcode D.3: fvSchemes

```
1  /*--------------------------------*- C++
       -*----------------------------------*\
2    =========                 |
3    \\      /  F ield         | OpenFOAM: The Open Source CFD
       Toolbox
4     \\    /   O peration     | Website:  https://openfoam.org
5      \\  /    A nd           | Version:  6
6       \\/     M anipulation  |
7  \*---------------------------------------------------------------------------
       */
8  FoamFile
9  {
10     version     2.0;
11     format      ascii;
12     class       dictionary;
13     location    "system";
14     object      fvSchemes;
15 }
16 // * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
       * * * * * * * * * //
17
18 ddtSchemes
19 {
20     default Euler;
21 }
22
23 gradSchemes
24 {
25     default         Gauss linear;
26 }
27
28 divSchemes
29 {
30     div(rhoPhi,U)  Gauss upwind;
31     div(phi,alpha)  Gauss vanLeer;
32     div(phirb,alpha) Gauss linear;
33     div(((rho*nuEff)*dev2(T(grad(U))))) Gauss linear;
34 }
35
36 laplacianSchemes
37 {
38     default         Gauss linear corrected;
39 }
40
41 interpolationSchemes
42 {
43     default         linear;
```

```
44 }
45
46 snGradSchemes
47 {
48     default           corrected;
49 }
50
51
52 //
    ************************************************************************
    //
```

Programmcode D.4: fvSolution

```
1  /*--------------------------------*- C++
   -*----------------------------------*\
2  =========                 |
3  \\      /  F ield         | OpenFOAM: The Open Source CFD
   Toolbox
4   \\    /   O peration     | Website:  https://openfoam.org
5    \\  /    A nd           | Version:  6
6     \\/     M anipulation  |
7  \*---------------------------------------------------------------------------
   */
8  FoamFile
9  {
10     version     2.0;
11     format      ascii;
12     class       dictionary;
13     location    "system";
14     object      fvSolution;
15 }
16 // * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
   * * * * * * * * * //
17
18 solvers
19 {
20     "alpha.water.*"
21     {
22         nAlphaCorr      1;
23         nAlphaSubCycles 2;
24         cAlpha      0;
25         MULESCorr       yes;
26         nLimiterIter    5;
27
28         solver          smoothSolver;
29         smoother        symGaussSeidel;
30         tolerance       1e-8;
31         relTol          0;
```

```
32
33      }
34
35      "pcorr.*"
36      {
37          solver          PCG;
38          preconditioner  DIC;
39          tolerance       1e-10;
40          relTol          0;
41      }
42
43      p_rgh
44      {
45          solver          GAMG;           // very efficient
    multigrid solver
46          tolerance       1e-7;          // solver finishes if
     either absolute
47          relTol          0.01;             // tolerance is
    reached or the relative
48                                         // tolerance here
49          minIter         1;             // a minimum number
    of iterations
50          maxIter         20;             // limitation of
    iterions number
51          smoother        DIC;           // setting for GAMG
52          nPreSweeps      2;             // 1 for p, set to 0
     for all other!
53          nPostSweeps     2;             // 2 is fine
54          nFinestSweeps   2;             // 2 is fine
55          scaleCorrection true;          // true is fine
56          directSolveCoarsestLevel false; // false is fine
57          cacheAgglomeration off;        // on is fine; set
    to off, if dynamic
58                                         // mesh refinement
    is used!
59          nCellsInCoarsestLevel 1000;    // 500 is fine,
60                                         // otherwise sqrt(
    number of cells)
61          agglomerator    faceAreaPair;  // faceAreaPair is
    fine
62          mergeLevels     1;             // 1 is fine

63      }
64
65      p_rghFinal
66      {
67          $p_rgh;
68          relTol          0;
```

```
69      }
70
71      U
72      {
73          solver          PBiCG;
74          preconditioner  DILU;
75          tolerance       1e-6;
76          relTol          0.01;
77      }
78
79      UFinal
80      {
81          $U;
82          relTol          0;
83      }
84  }
85
86  PIMPLE
87  {
88      U0      0.01;
89      momentumPredictor yes;
90      nCorrectors       3;
91      nNonOrthogonalCorrectors 0;
92  }
93
94
95  //
        ************************************************************************
        //
```

Programmcode D.5: setFieldsDict

```
1  /*--------------------------------*- C++
       -*----------------------------------*\
2  | =========                 |
                               |
3  | \\      /  F ield         | OpenFOAM: The Open Source CFD
       Toolbox                 |
4  |  \\    /   O peration     | Version:  2.1.1
                               |
5  |   \\  /    A nd           | Web:      www.OpenFOAM.org
                               |
6  |    \\/     M anipulation  |
                               |
7  \*---------------------------------------------------------------------------
       */
8  FoamFile
9  {
10     version     2.0;
```

```
    format      ascii;
    class       dictionary;
    location    "system";
    object      setFieldsDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
    * * * * * * * * * //

defaultFieldValues
(
    volScalarFieldValue alpha.water  0
    volScalarFieldValue p_rgh  0
    volVectorFieldValue U (0 0 0)
);

regions
(

    sphereToCell
    {
        centre  (0 0 0);
        radius  0.0007;
        fieldValues
        (
            volScalarFieldValue alpha.water 1
        );
    }

);


//
   ***************************************************************************
    //
```

Anhang E

Details zur numerischen Simulation der Phasentrennung

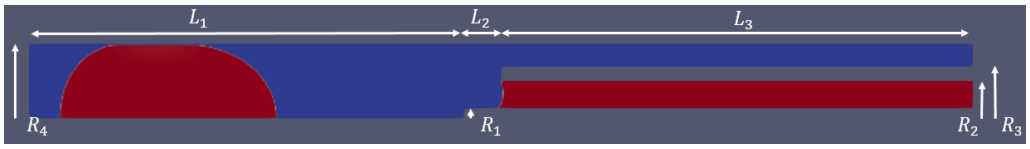


Abbildung E.1: Abmessungen des Simulationsgebietes.

Die Bestimmung von R_4 auf 0,8 mm und die radiale Auflösung von 200 Zellen legen die Gitterweite auf $4 \cdot 10^{-6}$ m fest. Für die Erzeugung eines möglichst hochwertigen Rechengitter mit einer minimalen Verzerrung der Gitterzellen wurden die Radien R_2, R_3 und R_4, sowie die Längen L_1, L_2 und L_3 daher als Vielfaches von $4 \cdot 10^{-6}$ m gewählt. Die Darstellung eines rotationssymmetrischen Problems über ei-

Tabelle E.1: Geometrische Abmessungen des in Abbildung E.1 dargestellten Fluidgebietes.

Parameter	Wert		Parameter	Wert
R_1	0,1 mm		L_1	4,8 mm
R_2	0,4 mm		L_2	0,4 mm
R_3	0,56 mm		L_3	5,2 mm
R_4	0,8 mm			

ne pseudo-2D-Modellierung erfordert die Nutzung eines wedge-Gitter, bei dem die Gitterweite in der dritten Raumrichtung direkt an der Rotationsachse verschwindet und dann linear mit dem Radius anwächst. Bei maximalem Abstand zur Rotationsachse von 0,8 mm wird in der Tiefe ebenfalls eine Gitterweite von $4 \cdot 10^{-6}$ m gesetzt. Dies entspricht einem Öffnungswinkel des wedge-Gitters von $0,58°$.

Anhang F

Untersuchungen zu Strömungsbildern im Rechteckkanal

F.1 Materialdaten und Kontakwinkel

Tabelle F.1: Verwendete Stoffdaten zur Berechnung der Strömungsprofile im Rechteckkanal.

Stoff	Dichte ρ	Viskosität η
Heptan	679,5 kg/m^3 [109]	0,387 mPa · s [109]
Hexadecan	770,1 kg/m^3 [109]	3,03 mPa · s [109]
Wasser	997,0 kg/m^3 [109]	0,890 mPa · s [109]
Wasser + KI	1400 kg/m^3	1,06 mPa · s
Wasser + KI	2000 kg/m^3	1,33 mPa · s
Wasser + KI	2300 kg/m^3	1,57 mPa · s

Tabelle F.2: Gemessene Kontaktwinkel zwischen Phasengrenzfläche und Wandfläche der verwendeten Stoffsysteme auf PMMA und hydrophilisiertem PC.

Stoffsystem	Polymethylmethacrylat	hydrophilisiertes Polycarbonat
Heptan-Wasser	64,8 °	58,3 °
Hexadecan-Wasser	63,6 °	59,5 °

F.2 Numerische Simulationen zum Abgleich

Zur gegenseitigen Validierung von analytisch berechnetem Volumenstromverhältnis Φ und der in OpenFOAM implementierten numerischen Methode wurden Simulationen auf einem an die Phasengrenzfläche angepassten Rechengitter durchgeführt. Ein solches Rechengitter ist in Abbildung F.1 dargestellt.

Abbildung F.1: An die Phasengrenzfläche angepasstes Gitter zur gegenseitigen Verifikation von analytischem und numerisch berechnetem Volumenstromverhältnis Φ.

F.3 Strömungsbilder in untersuchen Kanalgeometrien

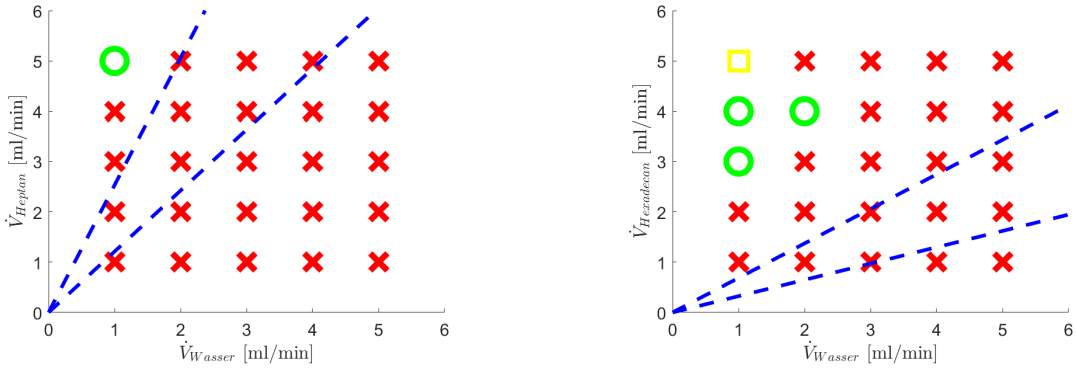


Abbildung F.2: Vergleich der Strömungsbilder in einem Kanal mit $B = 1{,}0\,\text{mm}$, $H_{aq} = 1{,}0\,\text{mm}$ und $H_{org} = 1{,}0\,\text{mm}$ bei Verwendung von Heptan-Wasser (links) und Hexadecan-Wasser (rechts): Stabile Parallelströmung (grüne Kreise), Parallelströmung mit Tropfenbildung (gelbe Quadrate), instabile Strömung (rote Kreuze), berechnetes Stabilitätgebiet zwischen gestrichelten Linien.

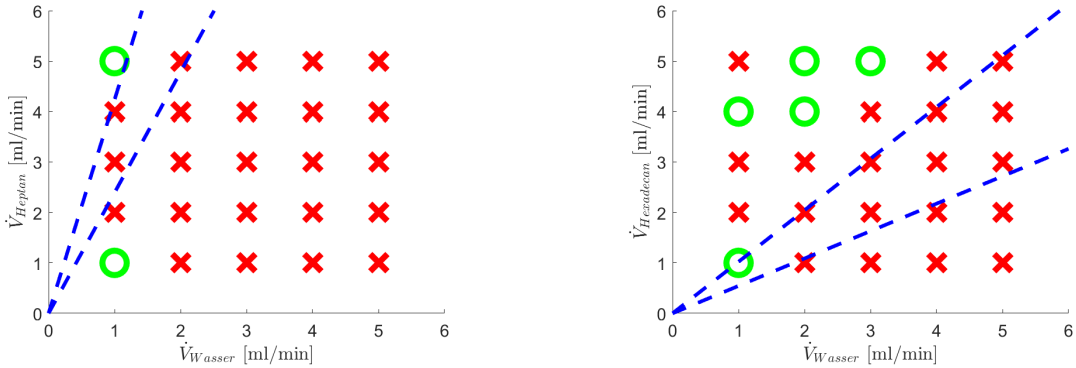


Abbildung F.3: Vergleich der Strömungsbilder in einem Kanal mit $B = 1{,}0\,\text{mm}$, $H_{aq} = 1{,}0\,\text{mm}$ und $H_{org} = 1{,}5\,\text{mm}$ bei Verwendung von Heptan-Wasser (links) und Hexadecan-Wasser (rechts): Stabile Parallelströmung (grüne Kreise), Parallelströmung mit Tropfenbildung (gelbe Quadrate), instabile Strömung (rote Kreuze), berechnetes Stabilitätgebiet zwischen gestrichelten Linien.

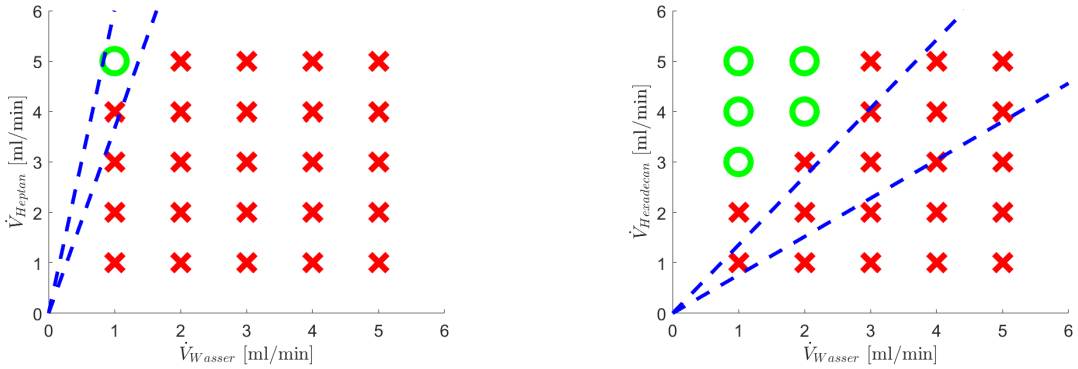


Abbildung F.4: Vergleich der Strömungsbilder in einem Kanal mit $B = 1{,}0\,\text{mm}$, $H_{aq} = 1{,}0\,\text{mm}$ und $H_{org} = 2{,}0\,\text{mm}$ bei Verwendung von Heptan-Wasser (links) und Hexadecan-Wasser (rechts): Stabile Parallelströmung (grüne Kreise), Parallelströmung mit Tropfenbildung (gelbe Quadrate), instabile Strömung (rote Kreuze), berechnetes Stabilitätgebiet zwischen gestrichelten Linien.

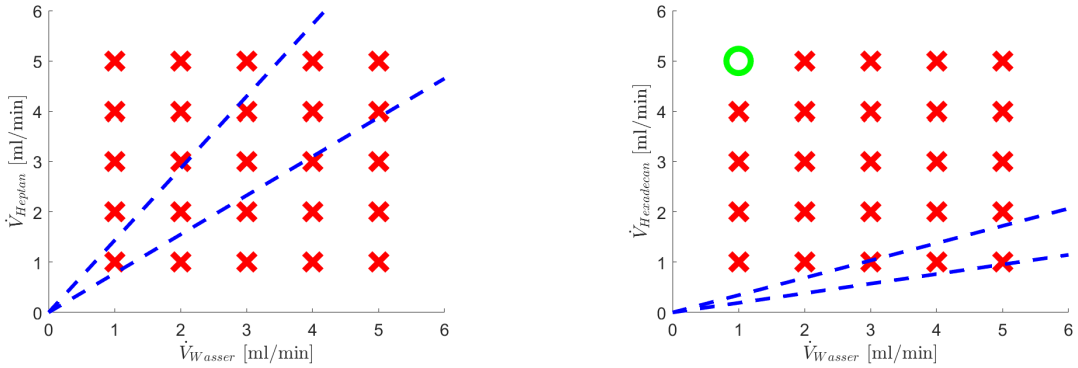


Abbildung F.5: Vergleich der Strömungsbilder in einem Kanal mit $B = 1{,}0\,\text{mm}$, $H_{aq} = 1{,}5\,\text{mm}$ und $H_{org} = 1{,}0\,\text{mm}$ bei Verwendung von Heptan-Wasser (links) und Hexadecan-Wasser (rechts): Stabile Parallelströmung (grüne Kreise), Parallelströmung mit Tropfenbildung (gelbe Quadrate), instabile Strömung (rote Kreuze), berechnetes Stabilitätgebiet zwischen gestrichelten Linien.

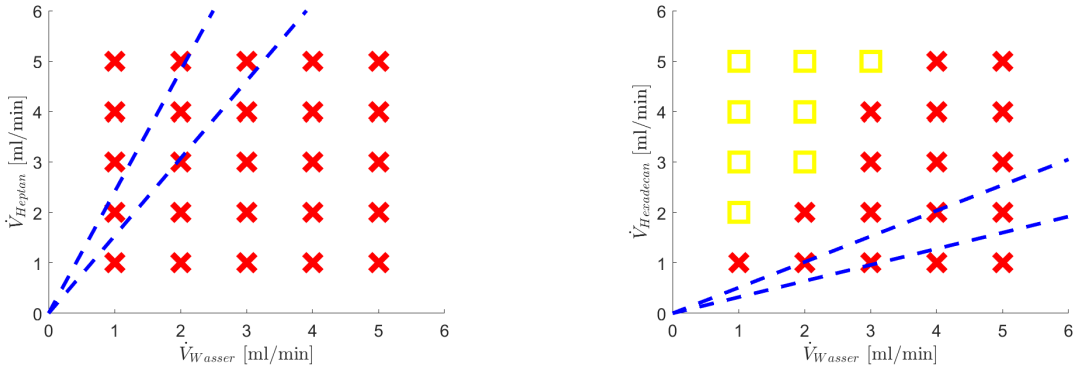


Abbildung F.6: Vergleich der Strömungsbilder in einem Kanal mit $B = 1{,}0\,\text{mm}$, $H_{aq} = 1{,}5\,\text{mm}$ und $H_{org} = 1{,}5\,\text{mm}$ bei Verwendung von Heptan-Wasser (links) und Hexadecan-Wasser (rechts): Stabile Parallelströmung (grüne Kreise), Parallelströmung mit Tropfenbildung (gelbe Quadrate), instabile Strömung (rote Kreuze), berechnetes Stabilitätgebiet zwischen gestrichelten Linien.

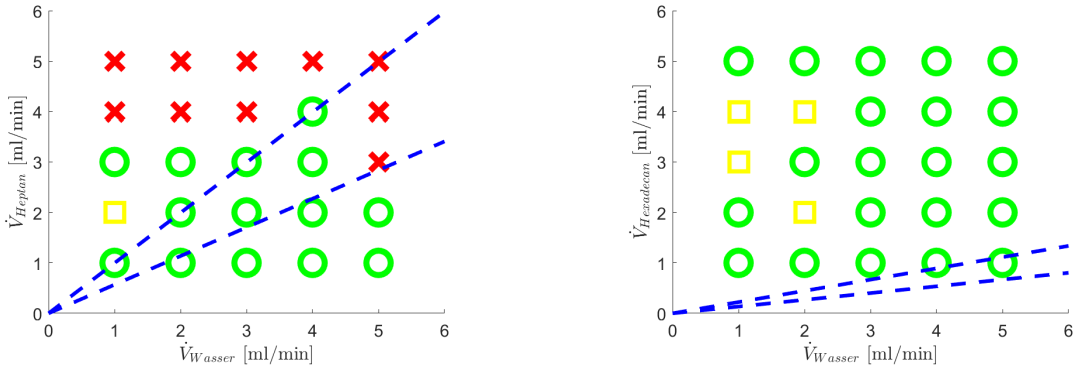


Abbildung F.7: Vergleich der Strömungsbilder in einem Kanal mit $B = 1{,}0\,\mathrm{mm}$, $H_{aq} = 2{,}0\,\mathrm{mm}$ und $H_{org} = 1{,}0\,\mathrm{mm}$ bei Verwendung von Heptan-Wasser (links) und Hexadecan-Wasser (rechts): Stabile Parallelströmung (grüne Kreise), Parallelströmung mit Tropfenbildung (gelbe Quadrate), instabile Strömung (rote Kreuze), berechnetes Stabilitätgebiet zwischen gestrichelten Linien.

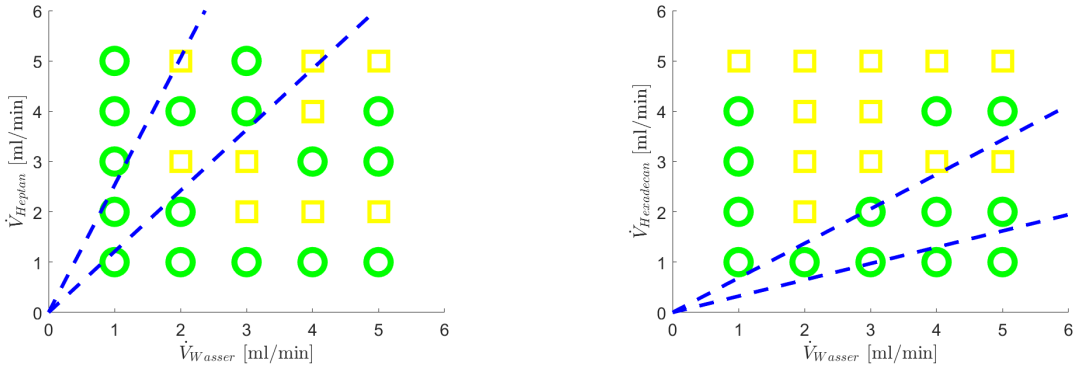


Abbildung F.8: Vergleich der Strömungsbilder in einem Kanal mit $B = 1{,}5\,\mathrm{mm}$, $H_{aq} = 1{,}5\,\mathrm{mm}$ und $H_{org} = 1{,}5\,\mathrm{mm}$ bei Verwendung von Heptan-Wasser (links) und Hexadecan-Wasser (rechts): Stabile Parallelströmung (grüne Kreise), Parallelströmung mit Tropfenbildung (gelbe Quadrate), instabile Strömung (rote Kreuze), berechnetes Stabilitätgebiet zwischen gestrichelten Linien.

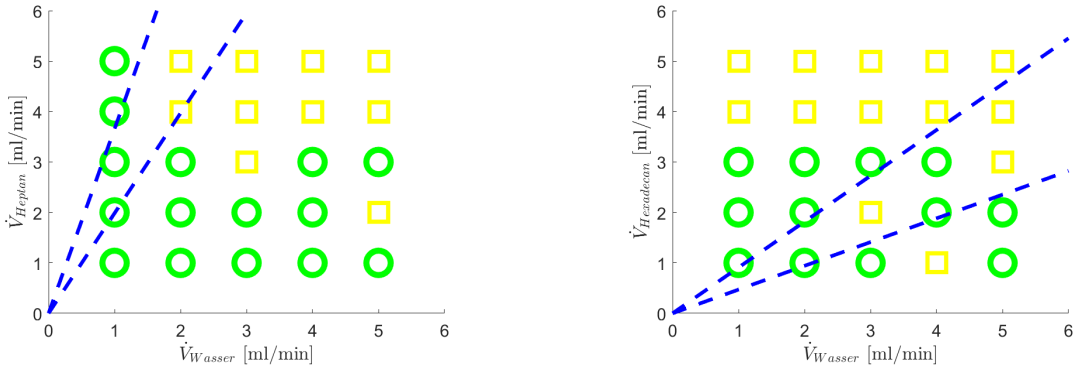


Abbildung F.9: Vergleich der Strömungsbilder in einem Kanal mit $B = 1{,}5\,\mathrm{mm}$, $H_{aq} = 1{,}5\,\mathrm{mm}$ und $H_{org} = 2{,}0\,\mathrm{mm}$ bei Verwendung von Heptan-Wasser (links) und Hexadecan-Wasser (rechts): Stabile Parallelströmung (grüne Kreise), Parallelströmung mit Tropfenbildung (gelbe Quadrate), instabile Strömung (rote Kreuze), berechnetes Stabilitätgebiet zwischen gestrichelten Linien.

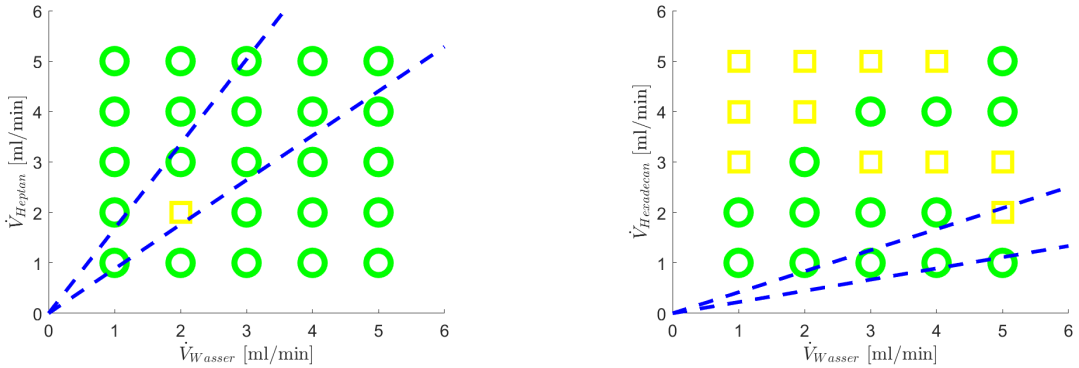


Abbildung F.10: Vergleich der Strömungsbilder in einem Kanal mit $B = 1{,}5\,\mathrm{mm}$, $H_{aq} = 2{,}0\,\mathrm{mm}$ und $H_{org} = 1{,}5\,\mathrm{mm}$ bei Verwendung von Heptan-Wasser (links) und Hexadecan-Wasser (rechts): Stabile Parallelströmung (grüne Kreise), Parallelströmung mit Tropfenbildung (gelbe Quadrate), instabile Strömung (rote Kreuze), berechnetes Stabilitätgebiet zwischen gestrichelten Linien.

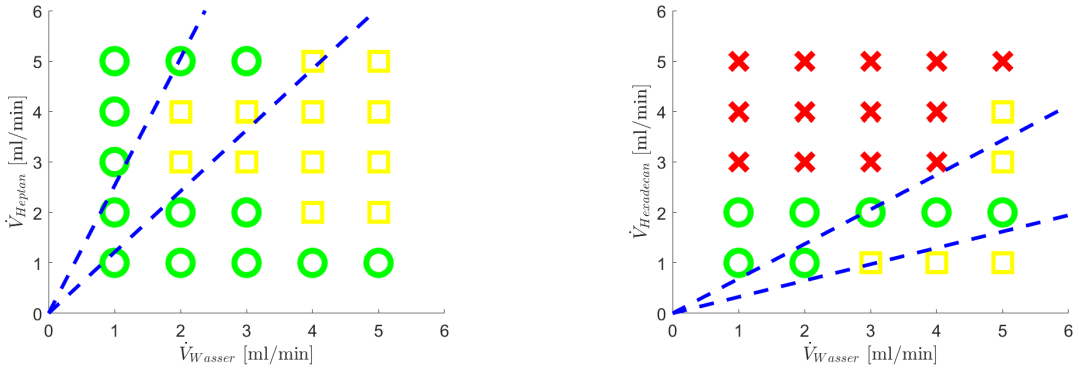


Abbildung F.11: Vergleich der Strömungsbilder in einem Kanal mit $B = 2{,}0\,\mathrm{mm}$, $H_{aq} = 2{,}0\,\mathrm{mm}$ und $H_{org} = 2{,}0\,\mathrm{mm}$ bei Verwendung von Heptan-Wasser (links) und Hexadecan-Wasser (rechts): Stabile Parallelströmung (grüne Kreise), Parallelströmung mit Tropfenbildung (gelbe Quadrate), instabile Strömung (rote Kreuze), berechnetes Stabilitätgebiet zwischen gestrichelten Linien.

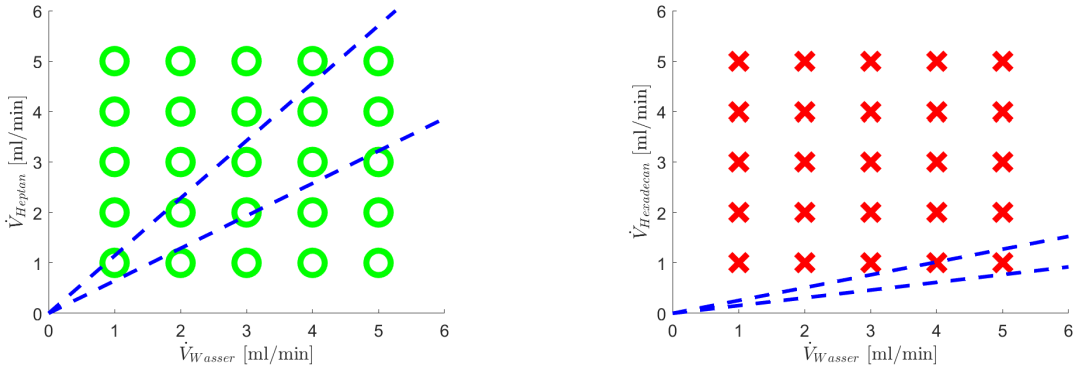


Abbildung F.12: Vergleich der Strömungsbilder in einem Kanal mit $B = 1{,}0\,\mathrm{mm}$, $H_{aq} = 2{,}0\,\mathrm{mm}$ und $H_{org} = 1{,}0\,\mathrm{mm}$ bei Verwendung von Heptan-Wasser + Kaliumiodid (links) und Hexadecan-Wasser + Kaliumiodid (rechts) bei einer Dichte von $\rho = 1400\,\mathrm{kg/m^3}$ und einer Viskosität von $\eta = 1{,}06\,\mathrm{mPa \cdot s}$ des Wassers: Stabile Parallelströmung (grüne Kreise), Parallelströmung mit Tropfenbildung (gelbe Quadrate), instabile Strömung (rote Kreuze), berechnetes Stabilitätgebiet zwischen gestrichelten Linien.

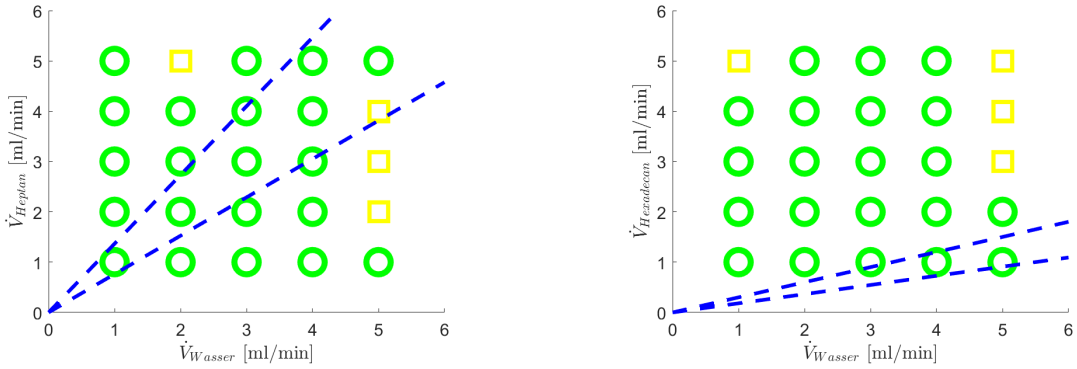


Abbildung F.13: Vergleich der Strömungsbilder in einem Kanal mit $B = 1{,}0\,\mathrm{mm}$, $H_{aq} = 2{,}0\,\mathrm{mm}$ und $H_{org} = 1{,}0\,\mathrm{mm}$ bei Verwendung von Heptan-Wasser + Kaliumiodid (links) und Hexadecan-Wasser + Kaliumiodid (rechts) bei einer Dichte von $\rho = 2000\,\mathrm{kg/m^3}$ und einer Viskosität von $\eta = 1{,}33\,\mathrm{mPa \cdot s}$ des Wassers: Stabile Parallelströmung (grüne Kreise), Parallelströmung mit Tropfenbildung (gelbe Quadrate), instabile Strömung (rote Kreuze), berechnetes Stabilitätgebiet zwischen gestrichelten Linien.

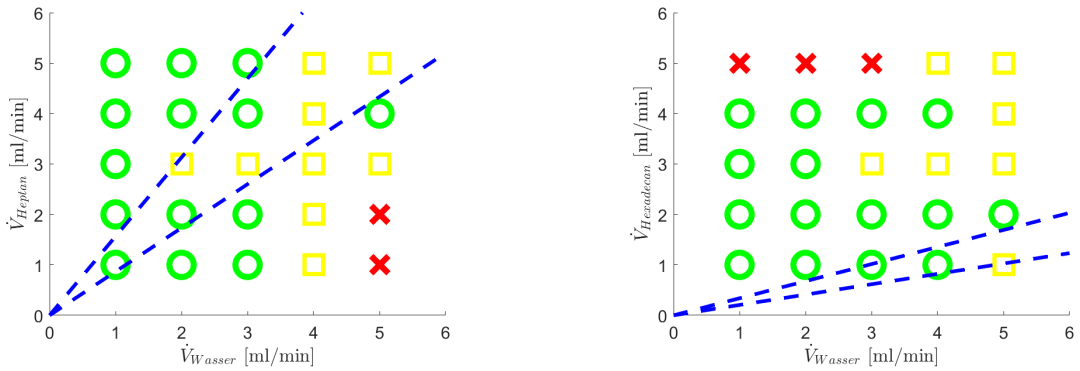


Abbildung F.14: Vergleich der Strömungsbilder in einem Kanal mit $B = 1{,}0\,\mathrm{mm}$, $H_{aq} = 2{,}0\,\mathrm{mm}$ und $H_{org} = 1{,}0\,\mathrm{mm}$ bei Verwendung von Heptan-Wasser + Kaliumiodid (links) und Hexadecan-Wasser + Kaliumiodid (rechts) bei einer Dichte von $\rho = 2300\,\mathrm{kg/m^3}$ und einer Viskosität von $\eta = 1{,}57\,\mathrm{mPa \cdot s}$ des Wassers: Stabile Parallelströmung (grüne Kreise), Parallelströmung mit Tropfenbildung (gelbe Quadrate), instabile Strömung (rote Kreuze), berechnetes Stabilitätgebiet zwischen gestrichelten Linien.

Anhang G

Auswertungen zur magnetischen Pfropfendetektion

G.1 Herstellung des Ferrofluids

Die Herstellung des Ferrofluids erfolgt nach Vorgaben von Knebel et. al ([122]) mit Hilfe von salzsaurem Eisenchlorid und Ammoniak nach Gl. G.1.

$$2FeCl_3 + FeCl_2 + 8NH_3 + 4H_2O \rightarrow Fe_3O_4 + 8NH_4Cl \tag{G.1}$$

Langsames Eintropfen der Ammoniaklösung erzeugte unter ständigem Rühren Nanopartikel aus Magnetit, welche abschließend durch Zugabe von Tetramethylammoniumhydroxid an ihrer Agglomeration gehindert wurden und so in Schwebe blieben. Überschüssiges Wasser wurde abdekantiert. Die erhaltenen 45 ml Ferrofluid können nach Stöchiometrie maximal 5 Gew.-% Magnetit enthalten. Durch die Verluste beim Abdekantieren, die trotz Zuhilfenahme eines starken Rückhaltemagneten auftraten, wird jedoch lediglich von 3-4 Gew.-% ausgegangen. Trotz angepasster Bauteile in der Brückenschaltung wird daher lediglich eine kleine Brückenspannung $\tilde{U}_0$ erwartet.

G.2 Detektion durch Messsoftware

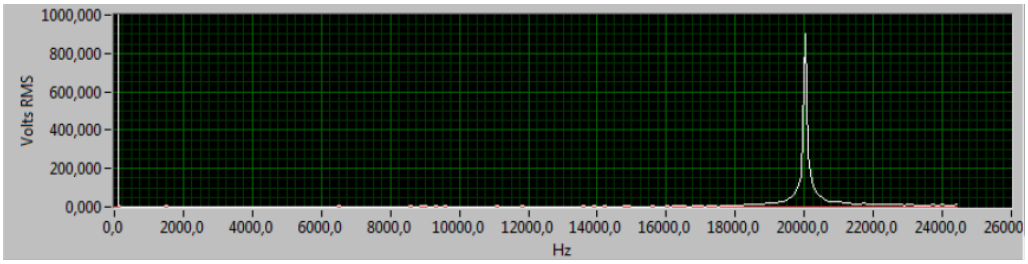


Abbildung G.1: Gemessene Brückenspannung $\tilde{U}_B$ von 0,9 V bei Anwesenheit eines Schraubendrehers im Spulenkern.

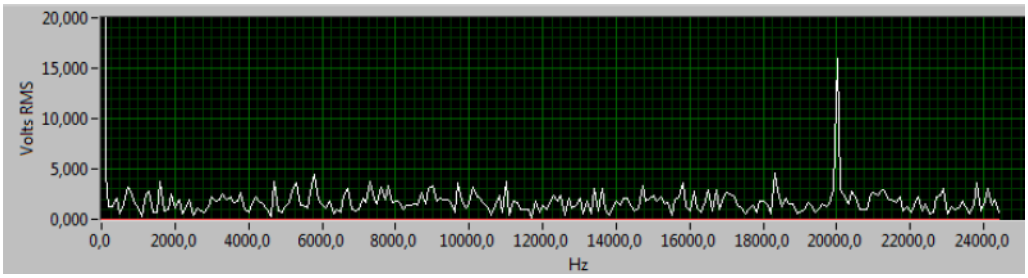


Abbildung G.2: Gemessene Brückenspannung $\tilde{U}_B$ von 0,016 V bei Anwesenheit von Ferrofluid innerhalb einer Pipette im Spulenkern.

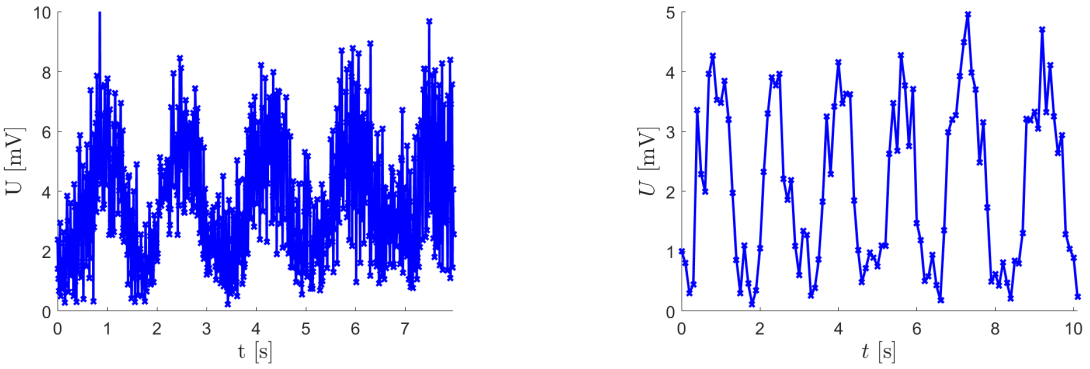


Abbildung G.3: Vergleich der Detektion einer ferrofluidischen Pfropfenströmung bei 100 Fouriertransformation mit jeweils 490 Datenpunkte pro Sekunde (links) und 10 Fouriertransformationen mit je 4900 Datenpunkten pro Sekunde (rechts).

Anhang H

Kalibriermessungen des UV-Spektroskops

Tabelle H.1: Polynomkoeffizienten der Kalibrierkurve $y = a + b \cdot x + c \cdot x^2$.

	M20	M50	M100
a	10,03 ± 0,05	9,81 ± 0,05	9,30 ± 0,07
b	-38,46 ± 1,26	-35,34 ± 1,30	-32,90 ± 1,55
c	61,09 ± 5,97	49,05 ± 6,24	47,84 ± 7,04

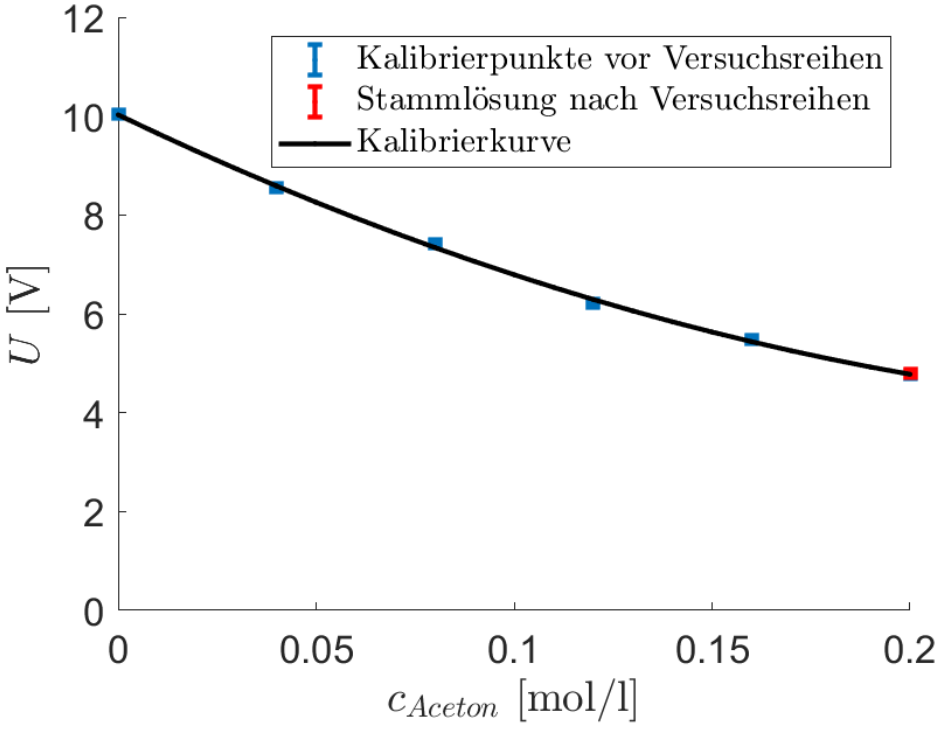


Abbildung H.1: Bestimmung der Kalibrierkurve für das Silikonöl M20.

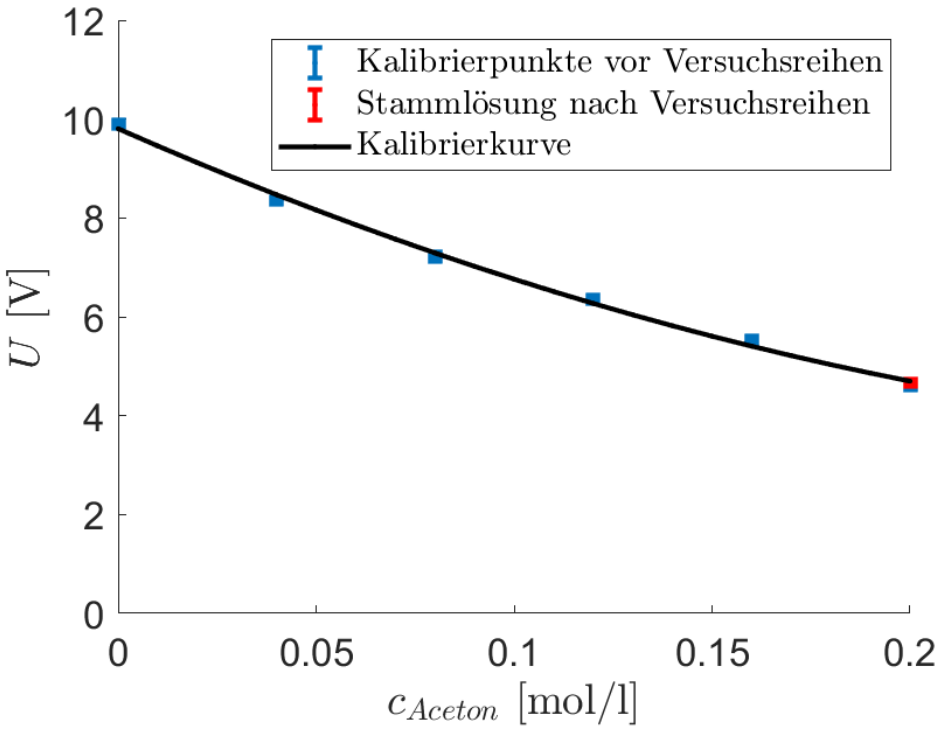


Abbildung H.2: Bestimmung der Kalibrierkurve für das Silikonöl M50.

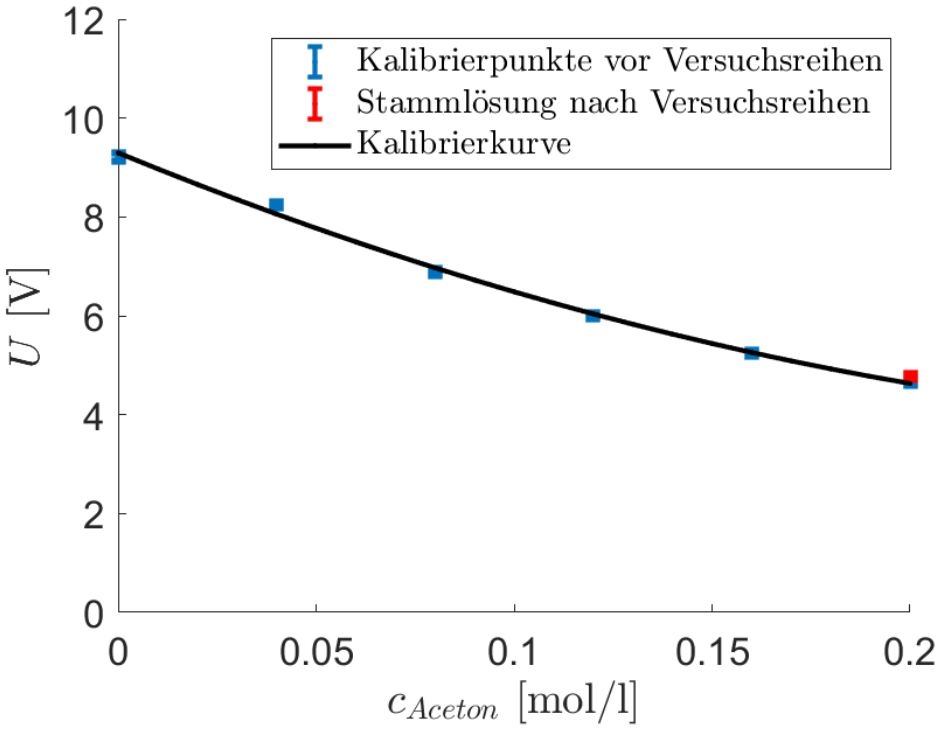


Abbildung H.3: Bestimmung der Kalibrierkurve für das Silikonöl M100.