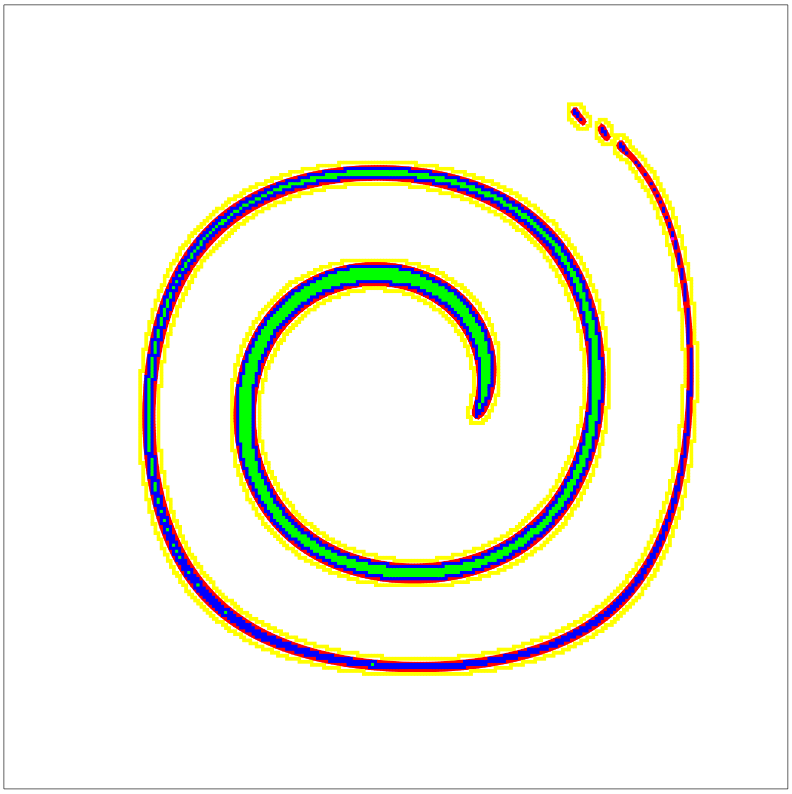


Volume-of-Fluid-Methoden zur Frontenverfolgung

Jan Köhnlein

Diplomarbeit am Institut für Angewandte Mathematik
Fachbereich Mathematik
Universität Hamburg

Eingereicht am 21. Mai 1999
korrigierte Version

Inhaltsverzeichnis

1. Einleitung

1.1 Was ist Frontenverfolgung ?

In vielen fluidmechanischen Anwendungen ist es erforderlich, die Grenze zwischen zwei unterschiedlichen Materialien für die numerische Berechnung zu rekonstruieren oder zu verfolgen. Eine solche Grenzlinie nennen wir *Front*. Ein paar typische Beispiele für die Anwendung numerischer Frontenverfolgung sind die Modellierung von fluidausstoßenden Geräten, wie zum Beispiel Tintenstrahldruckern, von Schmelzvorgängen, von Halbleiterplatinenätzung, die Simulation dünner Flammenfronten in Verbrennungsmodellen, Strömungssimulation mehrerer reagierender und nichtreagierender Fluide, zum Beispiel Öl und Wasser, oder die Behandlung von Strömungen mit freier Oberfläche, wie unter anderem im Schiffbau benötigt wird.
Da auf den von der Front getrennten Bereichen verschiedene physikalische Bedingungen modelliert werden müssen, ist es wichtig, zuverlässige Verfahren zu entwickeln, die die Front möglichst genau detektieren, verfolgen und bewegen können.
Die meisten hierfür verwendeten Methoden sind die Frontenverfolgungsmethoden (englisch *front tracking methods*): Eine Anfangsfront wird durch ein gegebenes Geschwindigkeitsfeld in der Zeit fortbewegt. Eine andere Technik wäre zum Beispiel die Frontenerkennung (englisch *front detection*), bei der die Front als Unstetigkeit aus gegebenen Daten herausgearbeitet wird.
Die vorliegende Arbeit beschäftigt sich mit Frontenverfolgungsalgorithmen. Eine kurze Darstellung üblicher Methoden findet sich in Kapitel 2. Der Schwerpunkt liegt aber auf den *volume-of-fluid-Methoden*, deren Theorie ausführlich in Kapitel 3 dargestellt werden soll, gefolgt von der Auswertung einiger numerischer Testbeispiele.

1.2 Formalia

Es werden im folgenden einige Notationskonventionen benutzt, die das Verständnis erleichtern sollen. Es bezeichnet $\mathbb{R}$ die reellen und $\mathbb{R}^+$ alle positiven von null verschiedenen reellen Zahlen. Zweidimensionale Vektoren werden immer fett gedruckt. Die Komponenten eines solchen Vektors $\mathbf{n}$ bezeichnen wir mit n_x und n_y. Punkte in Raum und Zeit bezeichnen wir mit Großbuchstaben, die letzte Komponente beschreibt dabei den Zeitwert. Alle Vektoren sind Spaltenvektoren. Für Skalarprodukte verwenden wir die Notation $A^T B$. Flächenintegrale werden immer als Doppelintegrale geschrieben. Das Integral

$$\iint\limits_{ABED} f \, dx \, dy$$

bezeichnet das Flächenintegral von f über das Polygon mit den Ecken A, B, E, und D. Volumenintegrale werden als Dreifachintegrale notiert. Das Integral

$$\iiint\limits_{ABCH} f\,dx\,dy\,dt$$

bezeichnet das Volumenintegral von f über das Polyeder mit den Ecken A, B, C, und H. Tiefgestellte Indizes bezeichnen Gitterkoordinaten, hochgestellte Zeitschichten.

Eine Schwierigkeit beim Erstellen dieser Arbeit war, daß fast alle Fachbegriffe aus dem Englischen stammen. Da sich deren Verwendung in der Fluidmechanik auch im Deutschen durchgesetzt hat, wurden viele davon in *kursiver Schrift* beibehalten. Im Anhang findet sich eine Liste dieser englischen Termini mit deutschen Übersetzungen.

2. Frontenverfolgungsalgorithmen

Grundsätzlich ist ein Frontenverfolgungsalgorithmus ein Algorithmus, der im $\mathbb{R}^n$ die Bewegung einer Mannigfaltigkeit der Dimension $n-1$, der Front, in einem gegebenen Geschwindigkeitsfeld berechnet.

In diesem Kapitel werden einige populäre Ansätze zur Verfolgung von Fronten beschrieben. Da das eigentliche Thema jedoch *volume-of-fluid*-Methoden sind, ist dieses Kapitel bewußt kurz gehalten. Für einen Vergleich der aus den verschiedenen Ansätzen resultierenden Algorithmen siehe zum Beispiel die Arbeiten von Rider und Kothe ([29] und [18]).

Da die meisten Anwendungen der Frontenverfolgungsalgorithmen strömungsmechanischer Natur sind, wird im folgenden immer von einem Fluid und von Luft die Rede sein. Dabei gehen wir davon aus, daß sich das Fluid innerhalb des von der Front berandeten Gebiets befindet und die Luft außerhalb. Natürlich werden dadurch andere Anwendungen nicht ausgeschlossen.

2.1 Grundbegriffe

2.1.1 Charakteristische Fluidfunktion

Wir werden im folgenden die charakteristische Funktion f des Fluids benötigen:

$$\begin{aligned} &f : D \longrightarrow \{0, 1\} \\ &f(x,y) = \begin{cases} 1, & \text{wenn sich an der Stelle } (x,y) \text{ Flüssigkeit befindet} \\ 0, & \text{sonst.} \end{cases} \end{aligned} \tag{2.1}$$

Man beachte, daß f eine Sprungfunktion ist. Die Front ist genau die Untermannigfaltigkeit, entlang derer f unstetig ist.

2.1.2 Advektionsgleichung

Auf unserem Gitter sei ein Geschwindigkeitsfeld $\mathbf{U}(x,y) = (u(x,y), v(x,y))^T$ gegeben, das zum Beispiel aus einer vorangegangenen Strömungsberechnung resultiert.

Die charakteristische Flüssigkeitsfunktion f aus Gleichung (2.1) wird passiv in diesem Geschwindigkeitsfeld bewegt und genügt damit der Advektionsgleichung

$$\partial_t f + u\,\partial_x f + v\,\partial_y f = 0, \tag{2.2}$$

wobei $\partial_t f$ die Ableitung im schwachen Sinne von f nach der Zeit t und $\partial_x f$ bzw. $\partial_y f$ entsprechende Ortsableitungen bezeichnen. Im Falle einer inkompressiblen Strömung ist das Geschwindigkeitsfeld divergenzfrei, das heißt, es gilt:

$$\operatorname{div} \mathbf{U} = \partial_x u + \partial_y v = 0. \tag{2.3}$$

Addiert man nun die Gleichung $(\partial_x u + \partial_y v)f = 0$ auf beiden Seiten von (2.2), erhält man im inkompressiblen Fall eine hyperbolische Erhaltungsgleichung für f, nämlich

$$\partial_t f + \partial_x(uf) + \partial_y(vf) = 0. \tag{2.4}$$

Hyperbolische Erhaltungsgleichungen spielen eine wichtige Rolle in der Strömungsmechanik, und so kann man sich für die Lösung der Gleichung (2.4) auf einen großen Fundus von Algorithmen stützen. Die Gleichung (2.4) bedeutet, daß sich die Änderung des Fluidvolumens in einem beliebigen zeitunabhängigen Kompaktum Ω mit stückweise glattem Rand mit dem Fluß über den Rand $\delta\Omega$ balanciert, denn es gilt

$$\begin{aligned}\frac{d}{dt}\int\limits_{\Omega} f\, d\Omega &= \int\limits_{\Omega} \frac{\partial f}{\partial t}\, d\Omega \\ &\overset{(2.4)}{=} -\int\limits_{\Omega} \operatorname{div}(f\mathbf{U})\, d\Omega \\ &= -\int\limits_{\delta\Omega} \mathbf{n}^T(f\mathbf{U})\, dS,\end{aligned}$$

wobei im letzten Schritt der Gaußsche Integralsatz angewendet wurde und $\mathbf{n}$ für die äußere Einheitsnormale des Randes $\delta\Omega$ steht.

Für kompressible Strömungen erhält man

$$\partial_t f + \partial_x(uf) + \partial_y(vf) = (\partial_x u + \partial_y v)f. \tag{2.5}$$

In diesem Fall erfüllt das Geschwindigkeitsfeld $\mathbf{U}$ die Bedingung (2.3) nicht, und das Gesamtvolumen wird nicht erhalten. In jedem Falle wird jedoch die Gesamtmasse der Flüssigkeit erhalten.

2.2 *Shock-Capturing*-Verfahren

Der erste Ansatz für ein Frontenverfolgungsverfahren ist, die Gleichung (2.5) bzw. (2.4) mit hochauflösenden *shock-capturing*-Verfahren zu behandeln. Diese Methoden wird man sowieso für andere Teile der Strömungsberechnungen benötigen, und meist erfordern sie nur geringe Modifikationen, um die Advektion der charakteristischen Fluidfunktion zu bewältigen. Typische Vertreter dieser Algorithmen sind *Godunov-Methoden* höherer Ordnung, wie zum Beispiel PPM [12].

Die größte Schwierigkeit bei diesem Ansatz ist, die numerische Diffusion möglichst gering zu halten, das heißt, daß die Front mit der Zeit möglichst wenig „verschmieren“ soll.

2.3 *Level-Set*-Methoden

Level-set-Methoden zur Frontenverfolgung wurden zum ersten Mal von Osher und Sethian in [21] vorgestellt. Wir halten uns hier an die Beschreibung von Sethian in [30].

Gegeben sei eine $(n-1)$ dimensionale Mannigfaltigkeit $\Gamma(t=0)$, die Ausgangsfront. Wir suchen nach einer Beschreibung der zeitlichen Änderung von $\Gamma(t)$. Wir gehen davon aus, daß sich Γ mit einer Geschwindigkeit F in Normalenrichtung fortbewegt. F kann dabei von Eigenschaften der Kurve, wie zum Beispiel der Krümmung oder der Normalenrichtung, abhängen. Wir benutzen eine *Eulersche Formulierung*, das heißt, wir beobachten die Flüsse auf einem festen Gitter, anstatt Teilchenbahnen zu verfolgen (*Lagrange-Formulierung*).

Die Hauptidee ist nun, daß diese Kurve die Nullniveaumenge einer Funktion $\phi : \mathbb{R}^{n+1} \to \mathbb{R}$ darstellt. Wir definieren ϕ als

$$\phi(\mathbf{x}, t = 0) = \pm d,$$

wobei d der kürzeste Abstand des Punktes $\mathbf{x}$ zu $\Gamma(t = 0)$ ist, und das Vorzeichen unterscheidet, auf welcher Seite von $\Gamma(t = 0)$ der Punkt $\mathbf{x}$ liegt.[1] Wir haben damit also eine Startfunktion $\phi(\mathbf{x}, t = 0) : \mathbb{R}^n \to \mathbb{R}$ mit der Eigenschaft

$$\Gamma(t = 0) = \{x \mid \phi(\mathbf{x}, t = 0) = 0\}.$$

Ziel ist es, eine Gleichung für die zeitliche Entwicklung von $\phi(\mathbf{x}, t)$ aufzustellen, in die die Bewegung von $\Gamma(t)$ als die der Niveaumenge $\phi(\mathbf{x}, t) = 0$ eingebettet ist. Hierzu sei $\mathbf{x}(t)$ der Weg, den ein Punkt auf der sich ausbreitenden Front verfolgt, das heißt, $\mathbf{x}(t = 0)$ ist ein Punkt auf der Ausgangsfront $\Gamma(t = 0)$ und $\mathbf{n}^T\mathbf{x}' = F(\mathbf{x}(\mathbf{t}))$ ist die Geschwindigkeit in der Normalenrichtung $\mathbf{n}$ mit $\mathbf{x}' = d\mathbf{x}/dt$. Die Forderung, daß die Nullniveaumenge der sich entwickelnden Funktion ϕ immer der Front $\Gamma(t)$ entspricht, bedeutet dann

$$\phi(\mathbf{x}(t), t) = 0.$$

Mit der Kettenregel folgt

$$\partial_t\phi + \nabla\phi(\mathbf{x}(t), t)^T\mathbf{x}'(t) = 0.$$

Da F die Geschwindigkeit in Richtung der äußeren Normalen beschreibt, gilt $\mathbf{n}^T\mathbf{x}' = F$ mit $\mathbf{n} = \nabla\phi/|\nabla\phi|$, und wir erhalten als gewünschte Evolutionsgleichung für ϕ

$$\partial_t\phi + F|\nabla\phi| = 0 \quad \text{mit gegebenem} \quad \phi(\mathbf{x}, t = 0). \tag{2.6}$$

Es sei an dieser Stelle noch angemerkt, daß Gleichung 2.6 für bestimmte Geschwindigkeitsfunktionen F die Form einer *Standard-Hamilton-Jacobi-Gleichung* annimmt.

Die Hauptvorteile der *level-set*-Methode sind:

- Die Funktion $\phi(\mathbf{x}, t)$ bleibt für alle Zeiten eine Funktion, während die Nullniveaumenge $\Gamma(t)$ topologische Änderungen durch das Ablösen oder das Verschmelzen mehrerer Bereiche erfahren kann.

- Weil ϕ eine Funktion bleibt, kann man numerische Methoden zur Lösung von (2.6) entwickeln, indem man das Rechengebiet diskretisiert und Approximationen durch finite Differenzen für die zeitlichen und räumlichen Ableitungen benutzt. Man kann zum Beispiel einfach ein kartesisches Gitter mit Maschenweite h über das Rechengebiet legen und Vorwärtsdifferenzen für die Zeitableitung sowie eine passende diskrete Version des Gradienten einsetzen. Solche expliziten finiten Differenzenansätze sind möglich und werden ausführlich in dem Buch von Sethian [30] beschrieben. Wichtig ist, daß die Differenzenoperatoren so wenig numerische Diffusion wie möglich produzieren, damit die Front nicht verschmiert, und daß sie die korrekte *Entropielösung* liefern.

[1] Für einen geschlossenen Weg $\Gamma(t = 0)$ kann man zum Beispiel mittels der Umlaufszahl von $\Gamma(t = 0)$ bezüglich x entscheiden, ob x von $\Gamma(t = 0)$ umlaufen wird oder nicht.

- Geometrische Eigenschaften der Front, wie die äußere Normale oder die Krümmung, können leicht aus den Werten von ϕ berechnet werden. Für einen Punkt auf der Front gilt beispielsweise

  $$\mathbf{n} = \frac{\nabla\phi}{|\nabla\phi|},$$

  und im zweidimensionalen Fall berechnet sich die Krümmung aus der Divergenz des Einheitsnormalenvektors

  $$\kappa = \nabla \cdot \frac{\nabla\phi}{|\nabla\phi|} = \frac{(\partial_x)^2\phi \cdot (\partial_y\phi)^2 - 2 \cdot \partial_x\phi \cdot \partial_y\phi \cdot \partial_x\partial_y\phi + (\partial_y)^2\phi \cdot (\partial_x\phi)^2}{\left((\partial_x\phi)^2 + (\partial_y\phi)^2\right)^{3/2}}.$$

  Hat man ein Problem zu lösen, in dem die Geschwindigkeitsfunktion F von solchen Größen abhängt, sind die *level-set*-Methoden wohl kaum zu schlagen. Im Buch von Sethian werden eine Reihe von Anwendungen auf den Fall $F = F(\kappa)$ zurückgeführt, das heißt, F ist eine Funktion der Krümmung.

- Die Verallgemeinerung auf drei Raumdimensionen ist direkt und einfach. Man muß in der Hauptsache eine passende Gradientenapproximation finden.

Das Verfahren wird jedoch alle enttäuschen, die an einer Parametrisierung der Front interessiert sind: Es wird ausdrücklich vermieden, die Front in irgendeiner Form zu rekonstruieren. Stattdessen muß man sie in diesem Falle mit Hilfe eines Höhenliniendetektors aus der Funktion ϕ extrahieren. Außerdem sind *level-set*-Methoden in der Regel nicht massenerhaltend.

2.4 Partikelmethoden

Generell läßt sich sagen, daß Partikelmethoden von einer *Lagrange-Darstellung* der Strömung ausgehen. Es werden die Teilchenbahnen einer Anzahl von Partikeln, den *marker*-Partikeln, in der Strömung berechnet und dann aus der Lage der Partikel wieder auf die Lage der Front geschlossen.

Man unterscheidet zwischen direkten und indirekten Methoden: Die direkten Methoden verfolgen eine Kette von *marker*-Partikeln, die auf der Front liegen. Es gibt eine Reihe von Nachteilen der direkten Methoden. Zum Beispiel ist eine Änderung der Topologie der Front nur schwer simulierbar, weil man an einer beliebigen Stelle nur über die lokalen Informationen eines Teilchens verfügt. Es können auch Probleme wie sogenannte *Schwalbenschwänze* (englisch *swallow tails*) auftreten, weil Teilchen von der Front ins Innere des Fluids wandern. Außerdem muß man für die numerische Stabilität die *marker*-Partikel eventuell nach einigen Zeitschritten neu verteilen. Eine ausführliche Beschreibung der Probleme dieser direkten Methoden findet sich in Sethians Buch [30].

Die indirekten Methoden weisen den einzelnen Partikeln Werte physikalischer Größen wie zum Beispiel Masse zu, und schließen dann auf die Lage der Front mittels Interpolation der Größen auf das Gitter.

Die Geschichte der Partikelmethoden reicht bis in die späten fünfziger Jahre zurück, und es gibt daher eine große Anzahl verschiedener Implementierungen, zum Beispiel die *particle-in-cell*-Methode (PIC), die *marker-and-cell*-Methode (MAC) und die *fluid-implicit-particle*-Methode (FLIP).

Eine einfache Darstellung einer Partikelmethode findet sich in der Arbeit [28] von Rider und Kothe, die folgenden Algorithmus vorschlagen:

Man nehme sich eine große Anzahl von Partikeln und bestimme, ob sie innerhalb oder außerhalb des Fluids liegen. Dann bewege man diese in jedem Zeitschritt mit einem *predictor-corrector*-Verfahren zweiter Ordnung vermöge

$$\mathbf{x}^{n+\frac{1}{2}} = \mathbf{x}^n + \frac{\Delta t}{2}\mathbf{u}^n(\mathbf{x}^n)$$

und

$$\mathbf{x}^{n+1} = \mathbf{x}^n + \Delta t\, \mathbf{u}^{n+\frac{1}{2}}\left(\mathbf{x}^{n+\frac{1}{2}}\right).$$

Nach einem solchen Advektionsschritt bestimmt man nun zum Beispiel die Dichte $\rho_{i,j}$ des Fluids in der Zelle (i, j) durch

$$\rho_{i,j} = \frac{\sum_k w_{i,j}^k \rho^k}{\sum_k w_{i,j}^k},$$

wobei ρ^k die Dichte des k-ten Partikels bezeichnet und die $w_{i,j}^k$ passende Gewichte für das k-te Partikel. Kothe und Rider schlagen verschiedene Gewichtungen vor, wie die nächster-Nachbar-Gewichtung

$$w_{i,j}^k = \begin{cases} 1 & \text{wenn} \quad 0 \leq \dfrac{x^k - x_{i,j}}{\Delta x}, \dfrac{y^k - y_{i,j}}{\Delta y} \leq \dfrac{1}{2} \\ 0 & \text{sonst,} \end{cases}$$

oder die bilineare Interpolation

$$w_{i,j}^k = \begin{cases} \left(1 - \dfrac{x^k - x_{i,j}}{\Delta x}\right)\left(1 - \dfrac{y^k - y_{i,j}}{\Delta y}\right) & \text{für} \quad 0 \leq \dfrac{x^k - x_{i,j}}{\Delta x}, \dfrac{y^k - y_{i,j}}{\Delta y} \leq 1 \\ 0 & \text{sonst,} \end{cases}$$

wobei $(x^k, y^k)^T$ die Position des k-ten Partikels beschreibt, $(x_{i,j}, y_{i,j})^T$ der Zellmittelpunkt der Zelle (i, j) ist und Δx und Δy die Breite bzw. Höhe der Zelle bezeichnet. Man kann mit dieser Methode auch kompressible Strömungen beschreiben, indem man den Partikeln unterschiedliche Dichten verleiht. Ebenso können Partikel Masse oder Volumen tragen.

Vorausgesetzt, man benutzt eine ausreichende Anzahl von Partikeln, und vorausgesetzt, daß die Divergenz des Geschwindigkeitsfeldes nicht zu groß wird, bieten die Partikelmethoden hervorragende Lösungen: Die numerische Diffusion der Front ist vernachlässigbar, und obwohl man keine Massenerhaltungsmechanismen eingebaut hat, sind die Verfahren in hohem Grade massenkonservativ. Der größte Nachteil ist, daß für zuverlässige Ergebnisse eine Unmenge von Partikeln benötigt wird, was enorme Rechenzeiten fordert. In der Genauigkeit schneiden die Partikelmethoden im Vergleich zu anderen Methoden hervorragend ab. Siehe hierzu [29].

3. *Volume-of-Fluid*-Methoden

3.1 Grundideen

Schon der Name *volume-of-fluid* (englisch für Fluidvolumen, im folgenden oft VOF abgekürzt) suggeriert, um was für einen Ansatz es sich hierbei handelt: Während Partikelmethoden Teilchenbahnen verfolgen und *level-set*-Methoden Niveaulinien berechnen, sind VOF-Verfahren volumenbasiert.

Man beginnt mit einer Aufteilung des Rechengebiets in Zellen. In jeder Zelle wird die *Füllung* berechnet, das Verhältnis von Flüssigkeitsvolumen zum Gesamtvolumen[1] der Zelle. Die Füllung kann nur Werte zwischen null (leere Zelle) und eins (Zelle ist voll) annehmen. Die ursprünglichen Informationen über die Lage der Front werden dann fallen gelassen. Stattdessen wird die Veränderung der Füllungen in der Zeit betrachtet. Die exakte Lage der Front ist zu keinem Zeitpunkt mehr bekannt. Sie wird vielmehr in jedem Zeitschritt neu aus den Füllungen rekonstruiert. Die so rekonstruierte Front ist in der Regel nicht eindeutig, das heißt, eine gegebene Füllungsverteilung garantiert keine eindeutige Topologie der Front. Diese richtet sich vielmehr nach den Grundannahmen über die Geometrie der Front und nach den Fähigkeiten der Rekonstruktionsmethode. Die für die Advektion der Füllungen notwendigen Flüsse über die Kanten der Zellen werden aus der rekonstruierten Front berechnet.

Ein Zeitschritt in einem *volume-of-fluid*-Verfahren besteht demnach aus zwei hintereinander ausgeführten Schritten:

1. Rekonstruiere die Front in allen in Frage kommenden Zellen aus den Füllungswerten der Zellen in einer gewissen Umgebung (*Rekonstruktionsschritt*).

2. Berechne die Volumenflüsse über die Kanten der Zellen und aktualisiere die Füllungen für den nächsten Zeitschritt (*Advektionsschritt*).

Grob gesagt, kann man VOF-Methoden an den Annahmen über die Frontgeometrie, am Rekonstruktionsalgorithmus und am Advektionsalgorithmus unterscheiden. Einige dieser Methoden sollen in dieser Arbeit verglichen werden.

Da die Geschichte der *volume-of-fluid*-Methoden bis in die Mitte der siebziger Jahre zurückreicht, sind die verschiedenen Algorithmen recht unterschiedlich im Ansatz, in der Rechenzeit und in der Genauigkeit der Ergebnisse. Es ist daher Vorsicht angebracht, wenn in der Literatur von *volume-of-fluid*-Methoden die Rede ist. In vielen kommerziellen wie nicht kommerziellen Programmpaketen sind ältere Versionen der VOF-Methoden in Gebrauch, deren relativ armselige Resultate den Ansatz als solchen ein wenig in Verruf gebracht haben. Modernere Varianten

[1] im zweidimensionalen Fall sind diese Volumen natürlich Flächen. Im folgenden wird der Begriff Volumen auch immer anstelle von Fläche benutzt, wenn Verwechslungen auszuschließen sind.

sind leider noch recht wenig verbreitet, und daher findet man auch in vielen neueren Veröffentlichungen Beschreibungen von Mängeln, die längst Teil der Geschichte sind. Zum Glück gibt es schon einige Arbeiten, die hier Aufklärungsarbeit leisten, zum Beispiel von Kothe und Rider [17] oder Pilliod [24].
Ein paar typische positive Eigenschaften haben jedoch alle VOF-Methoden gemeinsam:

- Änderungen in der Topologie des Flüssigkeitsgebietes wie das Abreißen oder Verschmelzen verschiedener Bereiche stellen grundsätzlich kein Problem dar. Im Gegenteil: In derartigen Fällen schneiden die VOF-Algorithmen sehr gut gegenüber den anderen Algorithmen ab.

- Alle VOF-Verfahren sind auf natürliche Weise massenerhaltend. Dies ist ein großer Vorteil zum Beispiel gegenüber den *level-set*-Methoden, bei denen oft Massenverluste in Kauf zu nehmen sind.

- Die Front bleibt zu allen Zeiten scharf. Die auftretende numerische Diffusion wirkt hauptsächlich transversal zur Front (Ecken werden „gebügelt“).

- Alle Berechnungen sind lokal, das heißt auf Daten einer kleinen Umgebung beschränkt. Die Verfahren eignen sich dadurch hervorragend zur Parallelisierung und Vektorisierung. Da eine Front eine Mannigfaltigkeit ist, deren Dimension um eins kleiner ist als die des Rechengebiets, ist der Rechenaufwand bei einem Gitter der Größe N^d typischerweise $O(N^{d-1})$ und kann daher gegenüber den Strömungsberechnungen auf dem gesamten Gitter angesichts der heutigen Rechnerkapazitäten vernachlässigt werden.

Einige Nachteile sind bei den VOF-Methoden jedoch auch in Kauf zu nehmen:

- Während andere Algorithmen teilweise nur unter großem Aufwand ein Abreißen von Bereichen der Flüssigkeit ermöglichen, leiden alle VOF-Algorithmen bei zu grober Diskretisierung des Gebietes unter *numerischer Oberflächenspannung*: In zu dünnen Bereichen der Flüssigkeit neigt die numerische Lösung zu unphysikalischer „Tröpfchenbildung“. Dies läßt sich nur durch Verfeinerung des Gitters beheben.

- In „schwierigen“ Geschwindigkeitsfeldern treten manchmal Fehler in den Flüssen auf, die zur Über- oder Unterfüllung einzelner Zellen führen. Man spricht hierbei von *numerischer Verwirbelung* (englisch *numerical vorticity*).

- Bei allen VOF-Methoden wird die Front stückweise mit linearen Funktionen approximiert. Dabei wird an den Zellübergängen oft eine Unstetigkeit der rekonstruierten Frontlinie in Kauf genommen. Bei einer hohen Auflösung fällt das nur an Stellen großer Krümmung und an den Ecken der Front auf. Es gibt außerdem Ungenauigkeiten bei der Berechnung lokaler Eigenschaften wie zum Beispiel der Krümmung der Frontkurve.

- Viele Vertreter der VOF-Algorithmen haben Schwierigkeiten mit Geschwindigkeitsfeldern, die eine große Wirbelstärke (englisch *vorticity*) aufweisen. Das ist allerdings kein auf VOF-Verfahren beschränktes Problem.

Folgende Nachteile sind typisch für ältere VOF-Methoden:

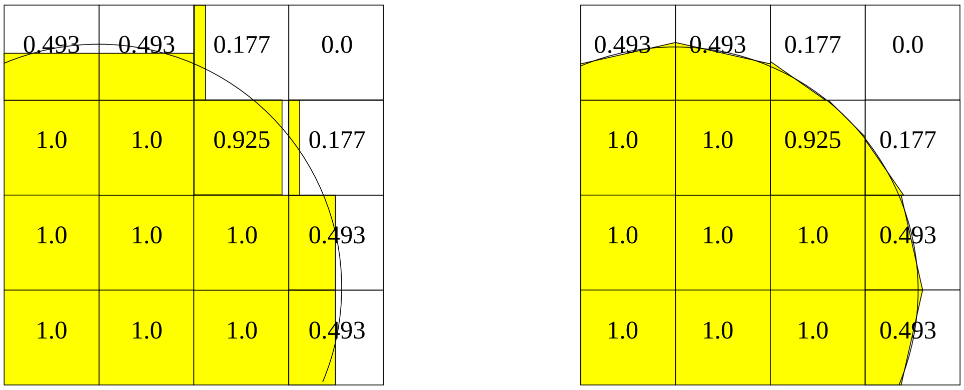


Abbildung 1: Rekonstruktion eines Kreisbogens (durchgezogene Linie), links mit einem SLIC-, rechts mit einem PLIC-Verfahren. Die Zahlen geben die Füllungen der Zellen an.

- Es werden größere Stücke aus der Flüssigkeit herausgerissen, die einsam im Raum treiben, oder es werden künstliche Blasen eingeschlossen[2]. Beides sind sehr auffällige und unphysikalische Phänomene.
- Um diese Effekte zu verhindern, werden Modifikationen angebracht, die die Massenerhaltung des Verfahrens schädigen.
- Durch den Einsatz von *operator splitting* in Rekonstruktion und Advektion ist das Verfahren stark abhängig von der Reihenfolge der Bearbeitung der verschiedenen Koordinatenrichtungen.
- Das Verfahren ist nicht in der Lage, stückweise lineare Bereiche der wahren Front exakt zu rekonstruieren.
- Die rekonstruierte Front erscheint extrem zackig.

Die Devise kann also nur heißen: Finger weg von den alten Algorithmen. Doch nun zu den Einzelheiten.

Die Frontenrekonstruktionsverfahren lassen sich in zwei Klassen unterteilen:

- Die *simple-line-interface-calculation*-Methoden (kurz SLIC-Methoden) rekonstruieren die Front stückweise mit Geradenstücken[3], die jeweils parallel zu einer der Koordinatenachsen ausgerichtet sind. Wir werden das im folgenden auch als *stückweise konstante* Rekonstruktion bezeichnen.
- Die *piecewise-linear-interface-calculation*-Methoden (kurz PLIC-Methoden) rekonstruieren die Front *stückweise linear*, also mit Geradenstücken beliebiger Ausrichtung.

Zur Veranschaulichung der unterschiedlichen Rekonstruktionen siehe Abbildung 1. Mit der eben gegebenen Beschreibung sind die SLIC-Methoden eine Teilmenge der PLIC-Methoden. Da jedoch noch unterschiedliche Voraussetzungen an die Lage der Geradenstücke gestellt werden, liefert ein PLIC-Verfahren nicht unbedingt immer bessere Ergebnisse als jedes SLIC-Verfahren.

[2]In der englischsprachigen Literatur finden sich hierfür die bildreichen Bezeichnungen *flotsam* (Treibgut) und *jetsam* (etwa: über Bord geworfene Ladung).

[3]im dreidimensionalen Fall mit ebenen Flächenstücken

Eines haben jedoch beide Typen gemeinsam: Die Rekonstruktion der Front ist ein lokales Problem, das zunächst auf die Berechnung einer Approximation der Frontnormalen in der betrachteten Zelle führt. Aus der Normalen und der Füllung läßt sich dann die Lage des Frontliniensegments in der Zelle berechnen.

3.2 Eine kurze Geschichte der VOF-Methoden

Die Geschichte der *volume-of-fluid*-Methoden beginnt in der Mitte der siebziger Jahre. Der erste Algorithmus geht auf R. deBar [13] zurück. Es ist bemerkenswert, daß deBars Algorithmus bereits ein PLIC-Algorithmus ist, also die Front stückweise linear rekonstruiert. An jeder Kante, durch die der Volumenfluß berechnet werden soll, wird die Frontnormale durch eindimensionale finite Differenzen der Füllungen berechnet. Diese Rekonstruktion ist eine sogenannte *operator-splitting*-Methode, da die Berechnung der Normalen auf eindimensionalen Füllungsdifferenzen in der aktuellen Advektionsrichtung beruht.

Wenig später veröffentlichten Noh und Woodward ihren SLIC-Algorithmus [20]. Gegenüber dem Algorithmus von deBar ist die Wahl stückweise konstanter Geradenstücke als Frontapproximation eigentlich ein Rückschritt. Angesichts der damals verfügbaren Rechnerkapazitäten war es wahrscheinlich jedoch gerade diese Vereinfachung, die diesen Algorithmus so populär machte. Die Berechnung der Normalen (horizontal oder vertikal) beruht auch hier auf *operator splitting*. Dieser Algorithmus hat schon die Möglichkeit, Strömungen mit mehr als zwei verschiedenen Materialien zu berechnen, ein Problem, das die PLIC-Algorithmen erst später bewältigen konnten. Die Behandlung solcher *Mehrphasenflüsse* ist insofern schwierig, da sich sich verschiedene Frontsegmente in einer Zelle schneiden können, was dazu führt, daß bestimmte Bereiche mehr als einem Material zugeordnet werden. Diese unphysikalische Durchdringung wird in SLIC-Methoden meistens durch die sogenannte „Zwiebelhautbehandlung“ (englisch *onion skin treatment*) gelöst, in der die Materialien in einer bestimmten, meist ganz am Anfang festgelegten Reihenfolge abgearbeitet werden. Eine modernere Implementierung wird zum Beispiel in der Arbeit [16] beschrieben.

Anfang der achtziger Jahre folgte dann der Algorithmus von Hirt und Nichols [14], die als erste den Namen *volume-of-fluid* benutzten. Dieser Algorithmus fand, an einen Löser für die *Navier-Stokes-Gleichung* gekoppelt, unter dem Namen SOLA-VOF und später als NASA-VOF2D und NASA-VOF3D weite Verbreitung. Die Front wird hier stückweise linear durch mehrdimensionale Füllungsdifferenzen rekonstruiert. Allerdings wird diese Frontapproximation nur zur Berücksichtigung der Oberflächenrandbedingungen benutzt. Der Advektionsalgorithmus beruht immer noch auf *operator splitting* und rekonstruiert die Front implizit treppenförmig. Eine genauere Beschreibung hierfür wird in Abschnitt 3.4.2.1 gegeben.

Verbesserungen und Erweiterungen auf dreidimensionale Probleme folgten von Chorin [10] und von Barr und Ashurst [8]. In jeder dieser Varianten wird die Front für den Advektionsschritt weiterhin treppenförmig rekonstruiert. Alle diese Methoden produzieren wegen dieser für die meisten Probleme zu ungenauen Approximation *flotsam* und *jetsam*.

In der FLAIR-Methode von Ashgriz und Poo [7] wird die Frontlinie mit einem Polygonzug rekonstruiert. Es ist dadurch die einzige Methode, bei der eine stetige Front erzeugt wird. Dies geht allerdings auf Kosten der Volumenerhaltungseigenschaft, da nicht gleichzeitig die Bedingung, daß das von der Front abgetrennte Gebiet die richtige Größe hat und die Stetigkeitsbedingung der Front erfüllt werden können.

Eine echte Verbesserung der VOF-Methoden stellte die Arbeit von Youngs [31] dar. Jedes Frontliniensegment wird durch seine Normale und einen Abstandsparameter beschrieben. Die Normale wird aus einer Gradientenapproximation durch finite Differenzen der Füllungen der Zelle und ihrer Nachbarn berechnet (in 2D entspricht das einem 3×3-Block, $3 \times 3 \times 3$ in 3D). Der Abstandsparameter wird so gewählt, daß das in der Zelle unter der Frontlinie liegende Volumen der Füllung entspricht. Youngs formulierte seinen Algorithmus in zwei und drei Dimensionen auf orthogonalen Gittern.

Seitdem sind eine Reihe von Erweiterungen und Verbesserungen dieses Algorithmus' erschienen. Einige widmeten sich der Verallgemeinerung der Methode auf nichtorthogonale Gitter (Johnson [6] für 2D, Kothe et al. [15] für 3D) oder koppelten sie einen adaptiven Gitterverfeinerungsalgorithmus (Puckett und Saltzman [27]). Schließlich entwickelten Pilliod [24] und Puckett [26], indem sie eine „optimale“ Normale berechnen, ein Verfahren zweiter Ordnung in dem Sinne, daß Geraden in einem räumlich und zeitlich konstanten Geschwindigkeitsfeld exakt rekonstruiert und transportiert werden.

Trotz der offensichtlichen Überlegenheit der moderneren (post Youngs) PLIC-Algorithmen sind wegen ihrer einfachen Implementierung noch eine Menge SLIC-Varianten im Umlauf. Dies ist auch darauf zurückzuführen, daß es noch kaum Veröffentlichungen über die neueren Varianten gibt. Doch gibt es schon eine Menge Anwendungen dieser Methoden, hauptsächlich mit inkompressiblen Strömungen. Hochburgen der VOF-Forschung sind die Los Alamos National Laboratories (Los Alamos, USA), die University of California (Davis, USA) und die Université Pierre et Marie Curie (Paris VI, Frankreich).

3.3 Frontenrekonstruktionsmethoden

In diesem Abschnitt wollen wir uns mit verschiedenen Algorithmen zur Rekonstruktion der Front beschäftigen. In Abschnitt 3.3.1 werden einige grundlegende Begriffe definiert, der Abschnitt 3.3.2 behandelt kurz, wie man aus der errechneten Normalen die genaue Lage der Frontlinie bestimmt, und der Abschnitt 3.3.3 beschäftigt sich mit verschiedenen Möglichkeiten der Normalenberechnung.

3.3.1 Füllungsfunktion

Wir beginnen mit der Diskretisierung des Rechengebiets D, indem wir es in rechteckige Zellen[4] $Z_{i,j}$ der Seitenlängen Δx und Δy aufteilen. Wir werden uns in dieser Arbeit ausschließlich auf den zweidimensionalen Fall mit äquidistantem Rechtecksgitter beziehen.

Jeder Zelle $Z_{i,j}$ wollen wir einen Wert $f_{i,j}$ zuordnen:

$$f_{i,j} \Delta x \Delta y = \text{ Volumen des Fluids in } Z_{i,j}.$$

$f_{i,j}$ nennen wir die Füllung[5] der Zelle $Z_{i,j}$. Die Tatsache, daß eine Zelle ein Stück der gesuchten Front enthält, ist äquivalent zu $0 < f_{i,j} < 1$. $f_{i,j}$ ist eine Diskretisierung der charkteristischen Funktion f des Fluids. Es gilt

$$f_{i,j} \Delta x \Delta y = \iint_{Z_{i,j}} f(x,y)\, dx\, dy. \tag{3.1}$$

Die Füllung ist also das Zellmittel der charakteristischen Fluidfunktion f.

[4] auch *Kontrollvolumina* oder *Gitterboxen* genannt

[5] In der Literatur finden sich auch Namen wie Farbfunktion (englisch *color-function*), Volumenanteil (englisch *volume-fraction*) oder VOF-Funktion.

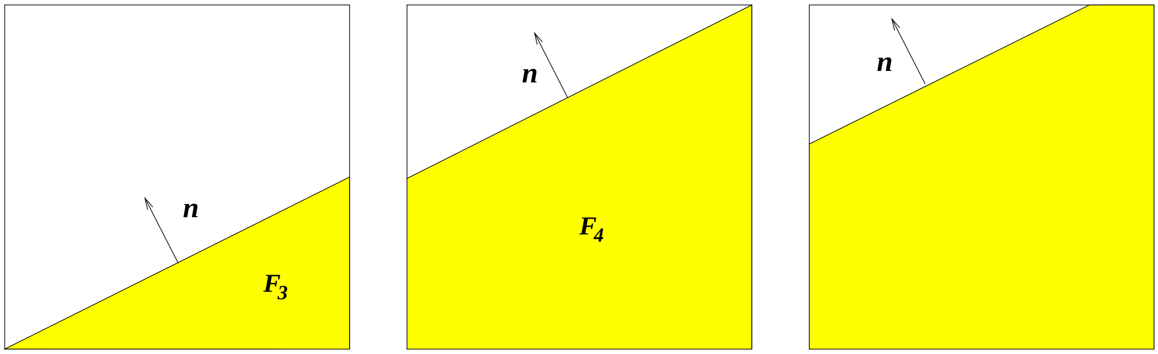


Abbildung 2: Berechnung des Parameters d des Frontliniensegments: Das entstehende Polygon hat drei Ecken, wenn $0 < f_{i,j} \leq F_3$, vier für $F_3 < f_{i,j} \leq F_4$ und fünf, wenn $F_4 < f_{i,j} < 1$ ist.

3.3.2 Frontberechnungen

Alle hier vorgestellten VOF-Verfahren rekonstruieren die Front so, daß die durch die Frontrekonstruktion erzeugte Füllung einer Zelle mit der aktuellen Füllung der Zelle übereinstimmt. Das wird erreicht, indem man das Frontsegment in Richtung der Normalen $\mathbf{n}$ verschiebt. Siehe zur Veranschaulichung Abbildung 2: Alle drei abgebildeten Frontapproximationen haben dieselbe Normale, erzeugen jedoch unterschiedlich große Füllungen der Zelle.
Diese Anpassung ist unbedingt erforderlich, wenn man einen volumenerhaltenden Algorithmus entwerfen will. Einige ältere VOF-Varianten wie zum Beispiel der Algorithmus von deBar [13] berücksichtigen das nicht, sondern zentrieren die rekonstruierte Frontlinie in jeder Frontzelle.
Zur Repräsentation des Frontliniensegments verwenden wir zwei Darstellungen: Die eine ist einfach die Darstellung

$$g := \{(x,y)^T \mid y = mx + b\}, \quad \text{mit } m, b \in \mathbb{R}$$

der Geraden. In dieser Darstellung gibt es Probleme mit vertikalen Fronten $x = const$[6]. Für die Theorie ist diese Darstellung allerdings oft ein bißchen handlicher. Etwas unanschaulicher, aber dafür für alle Geraden gültig ist die auch in der Implementierung verwendete Normalendarstellung:

$$g := \{(x,y)^T \mid \mathbf{n}^T \begin{pmatrix} x \\ y \end{pmatrix} = d\},$$

mit der Normalen $\mathbf{n} \in \mathbb{R}^2$ und einem Abstandsparameter $d \in \mathbb{R}$. Dem Verschieben des Frontsegments zur Anpassung der erzeugten Füllung entspricht dann die Berechnung der Parameter b bzw. d. Dabei gilt es, entsprechend dem Füllungswert der Zelle, die drei Fälle zu unterscheiden, ob das entstehende Polygon drei, vier oder fünf Ecken hat (Abbildung 2).
Die Normalendarstellung ist zum Rechnen etwas geschickter. Das unterhalb der Front entstehende Polygon ist der Schnitt der Zelle mit der Halbebene

$$H = \{\mathbf{x} \in \mathbb{R}^2 \mid \mathbf{n}^T \mathbf{x} < d\}. \tag{3.2}$$

Die Berechnung dieses Polygons nennt man *clipping* der Zelle mit der Halbebene H. Durch die Gleichung (3.2) hat man ein einfaches Kriterium, ob eine Ecke der Zelle innerhalb oder außerhalb

[6] Da diese Darstellung nur für die Theorie benutzt wird, seien Geradensteigungen $m = \infty$ zugelassen.

der Front liegt. Man muß dann nur noch die Schnittpunkte der Frontgeraden mit den Kanten, die eine Ecke außerhalb von H haben, berechnen.
Das Volumen eines solchen Polygons berechnet man am besten durch Aufteilung in Dreiecke ABC, für die man eine explizite Volumenformel

$$\begin{aligned} V(ABC) &= \frac{1}{2}||(B-A) \times (C-A)|| \\ &= \frac{1}{2}\left|(B_x - A_x) \cdot (C_y - A_y) - (B_y - A_y) \cdot (C_x - A_x)\right| \end{aligned}$$

aus dem $\mathbb{R}^3$ Vektorprodukt herleitet. Da ein VOF-Programm sehr häufig solche Polygonvolumina berechnen muß, lohnt es sich, die Aufspaltung in Dreiecke explizit, statt, wie es programmiertechnisch schöner wäre, rekursiv zu implementieren.
Bei all diesen Berechnungen zeigt sich wie an vielen anderen Stellen die eher geometrische als arithmetische Natur der VOF-Algorithmen.

3.3.3 Rekonstruktionsverfahren

Die beiden entscheidenden Unterschiede zwischen den verschiedenen Frontrekonstruktionsverfahren liegen zum einen im Ansatz der Rekonstruktion, nämlich stückweise konstant oder stückweise linear, und zum anderen in der Berechnung der Normalen eines Frontliniensegments.
Die hier beschriebenen Verfahren stellen nur eine kleine Auswahl aus vielen VOF-Methoden dar. Auf nicht volumenerhaltende Verfahren soll vollständig verzichtet werden. Bis auf das SLIC-Verfahren in Abschnitt 3.3.4 gehen alle Verfahren in dieser Arbeit von einer stückweise linearen Frontrekonstruktion aus. Der Schwerpunkt liegt also auf den modernen Methoden.

Es wird außerdem untersucht, ob die Methoden in der Lage sind, lineare Fronten exakt zu reproduzieren. Pilliod formuliert in [24] daraus einen Ordnungsbegriff: Ein Rekonstruktionsverfahren *zweiter Ordnung* reproduziert jede Gerade beliebiger Steigung exakt. Ein Verfahren *erster Ordnung* ist dann eines, das vertikale und horizontale Frontlinien exakt rekonstruiert. Wir werden prüfen, ob dieser Ordnungsbegriff sinnvoll ist, das heißt, ob ein Verfahren zweiter Ordnung wirklich bessere Ergebnisse liefert als eines erster Ordnung.

3.3.4 SLIC-Methode (*Simple-Line-Interface-Calculation*)

Die hier dargestellte Methode ist zwar nicht die ausgefeilteste SLIC-Variante, zeigt aber die wesentlichen Probleme bei der stückweise konstanten Rekonstruktion.

Der Algorithmus ist streng eindimensional. Die Verallgemeinerung auf mehrdimensionale Probleme beruht auf *operator splitting*. Die Front wird für jede Advektionsrichtung neu rekonstruiert. Die Lage der Front wird aus den Füllungen eines in der aktuellen Advektionsrichtung ausgerichteten 3×1-Blocks von Zellen mit der betrachteten Frontzelle in der Mitte bestimmt. Da die beiden Nachbarzellen jeweils voll, leer oder Frontzellen sein können, gibt es $3^2 = 9$ verschiedene Fälle, die in der Abbildung 3 dargestellt sind. Man beachte hierbei, daß die Abbildung keine Beispiele zeigt, sondern nur, wie die Front in der mittleren Zelle abhängig von den Nachbarfüllungen in der Advektionsrichtung ausgerichtet wird. Die Lage der Nachbarfüllungen wird, sofern es sich wieder um Frontzellen handelt, ebenfalls nach dem in Abbildung 3 skizzierten Schema bestimmt. Da auch der Advektionsschritt eindimensional ist, ist es ebenfalls unerheblich, ob ein in der Advektionsrichtung ausgerichtetes Frontsegment „liegt“ oder „hängt“ (Abbildung 3e). Das Frontsegment wird aber in jedem Falle noch so in Normalenrichtung verschoben, daß die

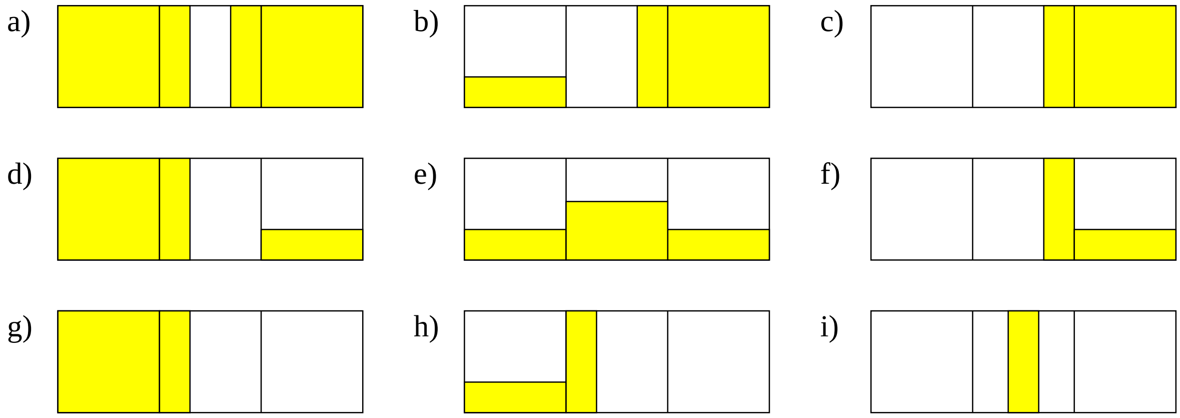


Abbildung 3: Beim SLIC-Algorithmus hängt die Ausrichtung der Front von den Füllungen der in Advektionsrichtung liegenden Nachbarzellen ab.

Volumina des ausgefüllten Bereichs mit der Füllung der Zelle übereinstimmen (siehe Abschnitt 3.3.2).

In diesem Verfahren sind in zwei Fällen auch zwei Frontlinien pro Zelle möglich (Abbildung 3a bis i).

Wegen der speziellen Ausrichtung der Frontnormalen (waagerecht oder senkrecht) ist der Algorithmus nicht in der Lage, Geraden exakt zu rekonstruieren, die nicht parallel zu einer der Koordinatenachsen stehen. Dadurch ist er höchstens von erster Ordnung.

3.3.5 Vereinfachte SLIC-Methode

Der Original-SLIC-Algorithmus aus dem letzten Abschnitt sieht vor, daß eine Zelle auch mehrere Frontliniensegmente enthalten kann. Das hätte eine Änderung der Datenstrukturen in der Implementierung notwendig gemacht. Stattdessen enthält das Programm einen vereinfachten SLIC-Algorithmus.

Es werden in jeder Zelle zwei Füllungsdifferenzen errechnet:

$$dx = f_{i+1,j} - f_{i-1,j},$$
$$dy = f_{i,j+1} - f_{i,j-1}.$$

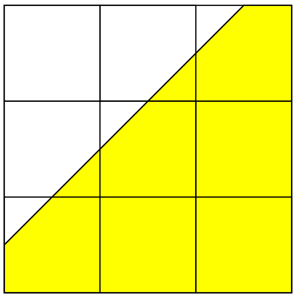
Das rekonstruierte Frontgeradenstück ist waagerecht, wenn $|dx| \leq |dy|$, und senkrecht, wenn $|dx| > |dy|$. Anhand des Vorzeichens von dy bzw. dx wird dann ermittelt, ob das Fluid oberhalb (rechts) oder unterhalb (links) von der Front liegt. Dann wird das Frontgeradenstück so noch verschoben, daß $f_{i,j}$ dem Fluidvolumen angepaßt ist.

Dieser Algorithmus kann natürlich aus demselben Grund wie der Original-SLIC-Algorithmus nicht von zweiter Ordnung sein.

3.3.6 Hirts und Nichols' Methode

Diese Methode wurde von Hirt und Nichols [14] für die Implementierung im SOLA-VOF-Code entwickelt. Sie basiert auf mehrdimensionalen finiten Differenzen der Füllungen und verzichtet somit auf *operator splitting*. Im weiteren Gegensatz zur SLIC-Methode wird die Front außerdem stückweise linear statt stückweise konstant rekonstruiert.

a)



b)

0	1/8	7/8
1/8	7/8	1
7/8	1	1

Abbildung 4: Der Algorithmus von Hirt und Nichols ist nicht in der Lage, die Gerade in a aus den Füllungen in b zu rekonstruieren, und ist deshalb nur von erster Ordnung.

Wir gehen von einem äquidistanten kartesischen Gitter mit Maschenweiten Δx und Δy aus. Wir nehmen an, die Front lasse sich in der Zelle $Z_{i,j}$ durch eine Gerade als Funktion $Y(x)$ oder $X(y)$ beschreiben. Eine gute Approximation an eine solche Gerade liefert

$$Y_i = Y(x_i) = \Delta y \cdot (f_{i,j-1} + f_{i,j} + f_{i,j+1}),$$

wobei der Ursprung $(0,0)$ an der unteren linken Ecke des 3×3-Zellenblocks mit Zelle $Z_{i,j}$ im Zentrum gesetzt wird. Die Steigung einer solchen Geraden läßt sich berechnen als

$$\frac{dY}{dx}(x_i) = \frac{Y_{i+1} - Y_{i-1}}{2\Delta x}.$$

Das entspricht der Differenz des Fluidvolumens der linken und rechten Spalte im 3×3-Block, dividiert durch den Abstand der Spalten.
Parametrisiert man die Gerade als $X(y)$, erhält man als Steigung

$$\frac{dX}{dy}(y_j) = \frac{X_{j+1} - X_{j-1}}{2\Delta y}$$

mit

$$X_j = X(y_j) = \Delta x \cdot (f_{i-1,j} + f_{i,j} + f_{i+1,j}),$$

was der zentralen Differenz der Zeilensumme der Füllungen entspricht. Hirt und Nichols wählen nun die Parametrisierung mit der betragsmäßig kleineren Geradensteigung. Anhand des Vorzeichens der anderen Steigung erkennt man, auf welcher Seite der Geraden das Fluid liegt. Nun kann man die Gerade so verschieben, daß das Volumen des durch die Gerade in der Zelle $Z_{i,j}$ berandeten Fluids mit der Füllung übereinstimmt.
Zum Beweis, daß dieser Algorithmus nicht in der Lage ist, beliebige Geraden exakt zu rekonstruieren, betrachte man das Gegenbeispiel in Abbildung 4: Mit $\Delta x = \Delta y$ hat die Gerade die Steigung eins, wenn man sie als $Y(x)$ parametrisiert, und die Steigung minus eins für $X(y)$. Für die Approximationen gilt jedoch

$$\frac{dY}{dx}(x_i) = \frac{Y_{j+1} - Y_{j-1}}{2} = \frac{1}{2}\left(\frac{7}{8} + 1 + 1 - \frac{1}{8} - \frac{7}{8}\right) = \frac{15}{16} \neq 1$$

und

$$\frac{dX}{dy}(y_j) = \frac{X_{i+1} - X_{i-1}}{2} = \frac{1}{2}\left(\frac{1}{8} + \frac{7}{8} - \frac{7}{8} - 1 - 1\right) = -\frac{15}{16} \neq -1.$$

Damit ist diese Rekonstruktionsmethode höchstens von erster Ordnung.
In Abschnitt 3.3.10 über den ELVIRA-Algorithmus werden wir sehen, daß man, wenn man außer den zentralen Zeilen- und Spaltendifferenzen auch noch Vorwärts- und Rückwärtsdifferenzen hinzunimmt, einen Algorithmus zweiter Ordnung erhält.

Es bleibt festzuhalten, daß dieser Algorithmus einer der schnellsten ist und, verglichen mit den anderen Rekonstruktionsmethoden erster Ordnung, sehr gute Ergebnisse liefert. Leider verwendeten Hirt und Nichols diese Methode nur, um Oberflächenrandbedingungen formulieren zu können. In dem zugehörigen *donor-acceptor*-Advektionsalgorithmus rekonstruieren sie die Front leider implizit treppenförmig in jeder Zelle. Siehe hierzu den Abschnitt 3.4.2.1.

3.3.7 *Center-of-Mass*-Algorithmus

Diese Methode geht auf Saltzman [27] zurück. Angenommen, unsere Flüssigkeit hat die Dichte eins und der leere Raum die Dichte null. Für jede Frontzelle wird im umgebenden 3×3-Zellenblock aus den Füllungen und der Geometrie der Zellen der Massenschwerpunkt des Fluids berechnet. Die Normale ist dann der normierte Vektor, der von diesem Punkt zum Mittelpunkt der betrachteten Zelle zeigt.

Wir wollen sehen, ob dieser Algorithmus eine Gerade $y = mx + b$ genau reproduzieren kann. Dazu betrachten wir die „exakte“ Version des Verfahrens. Das soll heißen, daß wir davon ausgehen, die exakte Front zu kennen, und daß wir daher auch die Lage des Schwerpunkts genau berechnen können. Wir zentrieren unser Koordinatensystem in der Mitte der betrachteten Frontzelle und untersuchen den Fall $b = 0$. Sei h die Breite der Zellen. Dann gilt für den Massenschwerpunkt $(\hat{x}, \hat{y})$

$$\hat{x} = \frac{\int_{-1.5h}^{1.5h}\int_{-1.5h}^{mx} x\,dy\,dx}{\int_{-1.5h}^{1.5h}\int_{-1.5h}^{mx} dy\,dx} = \frac{\int_{-1.5h}^{1.5h} mx^2 + 1.5hx\,dx}{\int_{-1.5h}^{1.5h} mx + 1.5h\,dx} = \frac{\frac{2}{3}m(1.5h)^3}{2(1.5h)^2} = \frac{mh}{2},$$

und

$$\hat{y} = \frac{\int_{-1.5h}^{1.5h}\int_{-1.5h}^{mx} y\,dy\,dx}{\int_{-1.5h}^{1.5h}\int_{-1.5h}^{mx} dy\,dx} = \frac{\int_{-1.5h}^{1.5h}(mx)^2 - (1.5h)^2\,dx}{9h^2} = \frac{h(1.5mh)^2 + 2(1.5h)^3}{9h^2}$$
$$= \frac{m^2h - 3h}{4}.$$

Für die Normale $\hat{\mathbf{n}}$ des rekonstruierten Geradenstücks gilt dann

$$\hat{\mathbf{n}} = const \cdot \begin{pmatrix}\hat{x}\\ \hat{y}\end{pmatrix} = const \cdot \begin{pmatrix}\frac{mh}{2}\\ \frac{m^2h-3h}{4}\end{pmatrix} = const \cdot \begin{pmatrix}2m\\ m^2-3\end{pmatrix}. \tag{3.3}$$

Die Ausgangsgerade hatte aber die Steigung m und damit eine Normale $\mathbf{n} = const(1, m)^T$.
Da das Verfahren nicht einmal in seiner exakten Form in der Lage ist, diese Ausgangsgerade zu rekonstruieren, wird auch eine numerische Approximation höchstens eine Genauigkeit erster

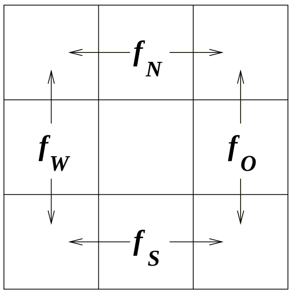


Abbildung 5: Berechnung der Differenzen für Parker's und Youngs' Algorithmus

Ordnung bieten. Weil in (3.3) die Richtung der Normalen nicht einmal mehr von h abhängt, ist auch keine Verbesserung zu erwarten, wenn wir die Ortsdiskretisierung verfeinern.
In der vorliegenden Implementierung wurde der Massenschwerpunkt als mit den Füllungen $f_{i,j}$ gewichtetes Mittel der benachbarten Zellmittelpunkte $x_{i,j}$ berechnet:

$$\hat{\mathbf{x}}_{i,j} = \sum_{k,l \in \{-1,0,1\}} f_{i+k,j+l}\, \mathbf{x}_{i+k,j+l},$$

und die Normale ist dann

$$\mathbf{n}_{i,j} = \frac{\mathbf{x}_{i,j} - \hat{\mathbf{x}}}{||\mathbf{x}_{i,j} - \hat{\mathbf{x}}||}.$$

3.3.8 Parkers und Youngs' Methode

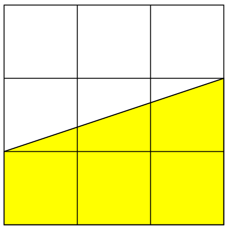
Diese Methode wurde von Parker und Youngs [22] entwickelt. Sie basiert auf der Berechnung einer Approximation des Gradienten der Fluidfunktion f. Da f eine Sprungfunktion ist, existiert dieser streng genommen nur im distributiven Sinne. Die Approximation wird aus finiten Differenzen der Füllungswerte berechnet und stellt somit eher den Gradienten einer Funktion dar, die die Füllungswerte in den Zellmittelpunkten interpoliert, als den der charakteristischen Fluidfunktion f. Der Gradient einer Funktion an einem Punkt zeigt bekanntlich immer in Richtung des stärksten Anstiegs der Funktion an dieser Stelle. Negiert und normiert stellt er daher eine gute Näherung an die Frontnormale dar.
Man berechnet ∇f wie folgt:

$$\frac{\partial f}{\partial x} = \frac{f_O - f_W}{2},$$
$$\frac{\partial f}{\partial y} = \frac{f_N - f_S}{2},$$

wobei f_N, f_O, f_S und f_W in der Nachbarzelle der jeweiligen Himmelsrichtung zentrierte Differenzen (siehe Abbildung 5) sind, nämlich

$$f_N = \frac{1}{2+\alpha}(f_{i-1,j+1} + \alpha f_{i,j+1} + f_{i+1,j+1}),$$
$$f_O = \frac{1}{2+\alpha}(f_{i+1,j-1} + \alpha f_{i+1,j} + f_{i+1,j+1}),$$
$$f_S = \frac{1}{2+\alpha}(f_{i-1,j-1} + \alpha f_{i,j-1} + f_{i+1,j-1})$$

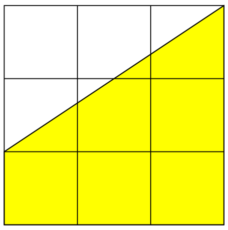
a)



b)

0	0	0
1/6	1/2	5/6
1	1	1

c)



d)

0	1/12	2/3
1/3	11/12	1
1	1	1

Abbildung 6: Zwei Beispielgeraden und die zugehörigen Zellfüllungen

und

$$f_W = \frac{1}{2+\alpha}(f_{i-1,j-1} + \alpha f_{i-1,j} + f_{i-1,j+1}).$$

Hierbei ist α ein freier Modellparameter. Parker und Youngs berichten, daß in numerischen Tests $\alpha = 2$ die besten Ergebnisse liefert.

Jetzt untersuchen wir, wie gut dieser Algorithmus Geraden rekonstruieren kann. Wir betrachten dazu einen 3×3-Block von Zellen der Breite h, dessen linke untere Ecke im Ursprung liegt. Die Ausgangsgerade sei $y = \frac{1}{3}x + h$. Abbildung 6a zeigt die Gerade, Abbildung 6b die daraus resultierenden Füllungen der einzelnen Zellen. Die Werte von f_N, f_O, f_S und f_W berechnen sich nun aus

$$\begin{aligned} f_N &= 0, \\ f_O &= \frac{1}{2+\alpha}(\frac{5}{6}\alpha + 1), \\ f_S &= \frac{1}{2+\alpha}(\alpha + 2) = 1 \end{aligned}$$

und

$$f_W = \frac{1}{2+\alpha}(\frac{1}{6}\alpha + 1),$$

und daraus ergibt sich

$$\frac{\partial f}{\partial x} = \frac{(f_O - f_W)}{2} = \frac{\alpha}{3(\alpha+2)}$$

und

$$\frac{\partial f}{\partial y} = \frac{(f_N - f_S)}{2} = -\frac{1}{2}.$$

Die Steigung $\hat{m}$ der rekonstruierten Geraden ist

$$\hat{m} = \frac{-\partial f/\partial x}{\partial f/\partial y} = \frac{2\alpha}{3(\alpha+2)}.$$

Setzt man diese mit der Steigung der ursprünglichen Geraden gleich, erhält man für α die Bedingung

$$\frac{2\alpha}{3(\alpha+1)} = \frac{1}{3} \quad \Longleftrightarrow \quad \alpha = 2.$$

Diese Gerade wird genau dann exakt konstruiert, wenn $\alpha = 2$ gewählt wird.
Setzen wir also $\alpha = 2$ und starten wir mit einer neuen Geraden $y = \frac{2}{3}x + h$, so erhalten wir

$$\begin{aligned} f_N &= \frac{1}{4}(0 + \frac{2}{12} + \frac{2}{3}) = \frac{5}{24}, \\ f_O &= \frac{1}{4}(1 + 2 + \frac{2}{3}) = \frac{11}{12}, \\ f_S &= \frac{1}{4}(1 + 2 + 1) = 1 \end{aligned}$$

und

$$f_W = \frac{1}{4}(1 + \frac{2}{3} + 0) = \frac{5}{12},$$

und für die Komponenten des Gradienten folgt

$$\frac{\partial f}{\partial x} = \frac{(f_O - f_W)}{2} = \frac{1}{4} \quad \text{und} \quad \frac{\partial f}{\partial y} = \frac{(f_N - f_S)}{2} = -\frac{19}{48},$$

und in diesem Falle ist die Steigung der rekonstruierten Geraden

$$\hat{m} = \frac{12}{19} \neq \frac{2}{3} = m.$$

Also rekonstruiert der Algorithmus von Parker und Youngs bei fester Wahl des Parameters α nicht alle Geraden exakt, und ist damit wie der *center-of-mass*-Algorithmus höchstens von erster Ordnung. Auch hier ist übrigens die berechnete Normale nicht mehr abhängig von der Maschenweite h des Gitters, so daß eine Verfeinerung ebenfalls nichts an der Genauigkeit ändern würde.

3.3.9 LVIRA (*Least Squares Volume-of-Fluid Interface Reconstruction Algorithm*)

Dieser Algorithmus wurde von Puckett [26] entwickelt.

Nehmen wir uns wieder einen 3×3-Zellenblock vor. Es sei $g(\tilde{\mathbf{n}})$ die Gerade mit der Normalen $\tilde{\mathbf{n}}$, die in die Richtung von $\tilde{\mathbf{n}}$ so verschoben wurde, daß die Fläche unter der Geraden in der mittleren Zelle $Z_{i,j}$ mit der Füllung übereinstimmt (siehe Abschnitt 3.3.2). Die so konstruierte Gerade erzeugt neue Füllungswerte $\tilde{f}_{k,l}(\tilde{\mathbf{n}})$ im Zellenblock $k = i-1, 0, i+1$ und $l = j-1, 0, j+1$. Durch die beschriebene Verschiebung ist

$$\tilde{f}_{i,j}(\tilde{\mathbf{n}}) = f_{i,j}. \tag{3.4}$$

Man kann die euklidische Norm der Differenz zwischen diesen neuen Füllungen $\tilde{f}_{k,l}(\tilde{\mathbf{n}})$ und den Originalfüllungen $f_{k,l}$ berechnen als

$$E_2(\tilde{\mathbf{n}}) = \left(\sum_{k,l=-1}^{1} \left(f_{i+k,j+l} - \tilde{f}_{i+k,j+l}(\tilde{\mathbf{n}}) \right)^2 \right)^{1/2}. \tag{3.5}$$

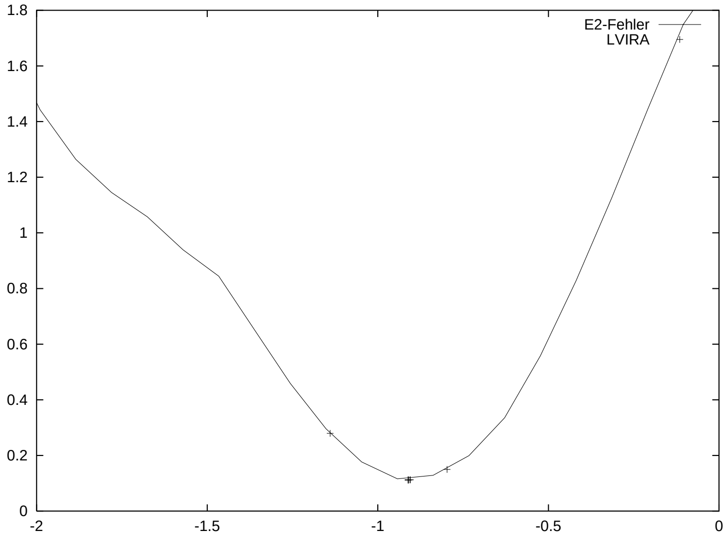


Abbildung 7: Beispiel für die Abhängigkeit des E_2-Fehlers (y-Achse) vom Argument der Normalen (x-Achse). Die Kreuze geben alle von LVIRA geprüften Normalen wieder. Man sieht, daß die Startnäherung gut ist und der Minimierungsalgorithmus relativ schnell konvergiert.

Der LVIRA-Algorithmus minimiert nun diese Abweichung E_2 in (3.5) durch Drehung der Normalen $\tilde{\mathbf{n}}$.

Die Wahl der Norm in (3.5) ist willkürlich. Man könnte stattdessen ebenso gut die Maximumnorm

$$E_\infty(\tilde{\mathbf{n}}) = \max_{k,l=-1,0,1} |f_{i+k,j+l} - \tilde{f}_{i+k,j+l}(\tilde{\mathbf{n}})|$$

oder zum Beispiel die 1-Norm

$$E_1(\tilde{\mathbf{n}}) = \sum_{k,l=-1}^{1} |f_{i+k,j+l} - \tilde{f}_{i+k,j+l}(\tilde{\mathbf{n}})|$$

verwenden. Wichtig ist nur, daß man die Bedingung (3.4) für jede Gerade durch Verschiebung entlang der Normalen garantiert.

Da die Normale immer die euklidische Länge eins hat, ist es sinnvoll, als Minimierungsparameter nicht die Normale $\tilde{\mathbf{n}}$ selbst zu benutzen, sondern deren Argument (den Winkel in der Polarkoordinatendarstellung). Damit wird deutlicher, daß wir in der Minimierungsaufgabe nur einen freien Parameter haben.

Nun zeigen wir, wie es um die exakte Reproduktion von geraden Frontstücken steht. Die Füllungen $\tilde{f}_{k,l}$ stimmen genau dann mit den Originalfüllungen überein, wenn die Gerade $g(\tilde{\mathbf{n}})$ mit der Ausgangsgeraden übereinstimmt. In jeder der betrachteten Normen ist der Fehler genau dann null. Das heißt, daß der LVIRA-Algorithmus mit jeder Norm die Ausgangsgerade exakt reproduziert, vorausgesetzt, daß der Minimierungsalgorithmus das korrekte Minimum findet.

Bei der hier untersuchten Variante des LVIRA-Algorithmus wird zur Minimierung der Algorithmus von Brent benutzt, der beispielsweise in [25] zu finden ist. Dieser iterative Algorithmus legt durch drei Startwerte eine Parabel, deren Minimum dann im nächsten Schritt einen der äußeren Startwerte ersetzt. Kann keine Parabel interpoliert werden, wird der neue Wert durch den Goldenen Schnitt zwischen den beiden äußeren Werten ermittelt.

Eine Startnäherung für die Normale berechnen wir mit dem Algorithmus von Hirt und Nichols (siehe Abschnitt 3.3.6).

Um die für Brents Algorithmus notwendigen anderen beiden Startwerte zu erhalten, konstruiert man „vorsichtig“ ein kleines Intervall um die Startnäherung, bis die Füllungsabweichungen an den Intervallrändern größer sind als die der Startnäherung.[7] Dann iteriert man, bis eine vorgegebene Abbruchgenauigkeit erreicht wird.

Abbildung 7 zeigt den typischen Verlauf der Abhängigkeitskurve des Fehlers E_2 vom Argument der Normalen. Wenn die lokalen Minima der Kurve globale Minima sind, konvergiert der Algorithmus von Brent gegen eine richtige Lösung. In einigen Fällen können jedoch mehrere globale Minima auftreten. Siehe hierzu den Abschnitt 3.3.10.1. Existieren lokale Minima, die nicht auch globale Minima sind, kann man nur hoffen, daß das Startintervall ausschließlich die richtige globale Minimalstelle umschließt. Aus demselben Grund ist man auch auf eine gute Startnäherung angewiesen.

Die Bezeichnung *least squares* für den LVIRA-Algorithmus ist etwas irreführend. Sie kommt daher, daß die Methode im Original den E_2-Fehler (3.5) minimiert. Man sollte das nicht mit der *least-squares*-Methode der Optimierung (deutsch: Methode der kleinsten Fehlerquadrate) verwechseln. Die Abbildung

$$F : F(\tilde{\mathbf{n}}, f_{i,j}) \mapsto (\tilde{f}_{k,l})_{(k=i-1,i,i+1,\, l=j-1,j,j+1)},$$

die einer gegebenen Normalen $\tilde{\mathbf{n}}$ und der Füllung der zentralen Zelle die entsprechenden, durch die eingepaßte Gerade erzeugten Füllungen zuordnet, ist nichtlinear. Daher kann man das Minimierungsproblem nicht wie beim gleichnamigen Optimierungsalgorithmus auf die Lösung eines linearen Gleichungssystems zurückführen.

Die LVIRA-Methode ist, wie wir oben beschrieben haben, in der Lage, Geraden exakt zu rekonstruieren, und das unabhängig von der Feinheit der Ortsdiskretisierung. Dadurch ist sie nach unserem Ordnungsbegriff von zweiter Ordnung. Leider ist sie durch den iterativen Minimierungsansatz die langsamste hier untersuchte Methode. Hat man die heuristischen Probleme, wie die Berechnung des Startintervalls für den Brent-Algorithmus aus der Startnäherung für die Normale, stabil gelöst, liefert die Methode sehr gute Ergebnisse. Ein weiterer nicht zu unterschätzender Vorteil ist, daß LVIRA direkt auf unstrukturierte Ortsgitter verallgemeinert werden kann. Man muß sich dann nur noch über eventuelle Gewichtungen der E_2-Fehler in den Nachbarzellen Gedanken machen.

3.3.10 ELVIRA (*Efficient Least Squares Volume-of-Fluid Interface Reconstruction Algorithm*)

Der Name läßt es schon vermuten: Hier wurde der eben vorgestellte LVIRA-Algorithmus effizienter gemacht und das bedeutet schlicht und einfach *schneller*. ELVIRAs Erfinder heißt Pilliod, und die wesentlichen Charakteristika dieses Algorithmus' finden sich in [24] und [23]. Leider waren beide nur als Preprints erhältlich und enthielten eine Reihe von Fehlern und Unschlüssigkeiten. Ich hoffe, diese alle aufgespürt und beseitigt zu haben. Jedenfalls läuft der von mir implementierte Algorithmus stabil.

[7] In der Praxis ist das oft schwierig, da man a priori nicht weiß, wie groß das Intervall sein muß.

Im Prinzip wird hier, wie bei LVIRA, der E_2-Fehler der verlängerten Frontsegmentapproximation in den Nachbarzellen minimiert. Anstatt jedoch einen langsamen iterativen Minimierungsalgorithmus zu benutzen, wählt ELVIRA das optimale Geradenstück aus sechs Kandidaten.

Es ist zur Betrachtung zweckmäßig, die Geradenapproximationen in der Form $y = m_x x + b$ bzw. $x = m_y y + b$ darzustellen. Die ersten drei Kandidaten sind Geraden mit den Steigungen m_x^r, m_x^z und m_x^v, die als

$$\begin{aligned} m_x^r &= \sum_{k=-1}^{1} f_{i,j+k} - f_{i-1,j+k}, \\ m_x^z &= \frac{1}{2} \sum_{k=-1}^{1} f_{i+1,j+k} - f_{i-1,j+k} \qquad \text{und} \\ m_x^v &= \sum_{k=-1}^{1} f_{i+1,j+k} - f_{i,j+k} \end{aligned} \tag{3.6}$$

definiert sind. Das entspricht Rückwärts- (m_x^r) und Vorwärtsdifferenzen (m_x^v) bzw. zentralen Differenzen (m_x^z) der Spaltensummen der Füllungen. Die restlichen drei Kandidaten sind

$$\begin{aligned} m_y^r &= \sum_{k=-1}^{1} f_{i+k,j} - f_{i+k,j-1}, \\ m_y^z &= \frac{1}{2} \sum_{k=-1}^{1} f_{i+k,j+1} - f_{i+k,j-1} \qquad \text{und} \\ m_y^v &= \sum_{k=-1}^{1} f_{i+k,j+1} - f_{i+k,j}, \end{aligned} \tag{3.7}$$

also Vorwärts-, Rückwärts- und zentrale Differenzen der Zeilensummen der Füllungen. Man beachte, daß die m_y Geraden als $x = m_y y + b$ darstellen, also Geraden in einem um 90° gedrehten Koordinatensystem.

Man beachte die Analogie zum Algorithmus von Hirt und Nichols: Zu den zentralen Differenzen sind noch einseitige Differenzen hinzugenommen worden.

Natürlich werden auch bei ELVIRA alle Geradenstücke so verschoben, daß die Füllung der zentralen Zelle mit der durch die Gerade erzeugten Füllung übereinstimmt.

Die Steigungen in (3.7) sind so gewählt, daß beliebige Geraden exakt reproduziert werden. Wir wollen die Frontapproximation immer in der mittleren Zelle berechnen. Daher muß die Frontlinie diese immer schneiden.

Durch Drehung des Zellblocks können wir immer erreichen, daß die Ausgangsgerade $y = mx + b$ den linken Rand und entweder den oberen oder den rechten Rand des Zellblocks schneidet. Wir untersuchen die in Abbildung 8 dargestellten Fälle.

Im in Abbildung 8a dargestellten Fall schneidet die Ausgangsgerade zwei gegenüberliegende Kanten des 3×3-Zellblocks. Sei A_1 die Summe der Füllungen der Zellen der ersten Spalte. Das entspricht dem Volumen des in der linken Spalte gefüllten Trapezes, multipliziert mit h^2, dem Faktor zur Umrechnung von Volumen auf Füllungen. A_2 sei das Pendant der rechten Spalte. Es gilt

$$A_1 = \frac{1}{h^2} \cdot h \cdot \frac{b + b + mh}{2} = \frac{2b + mh}{2h}$$

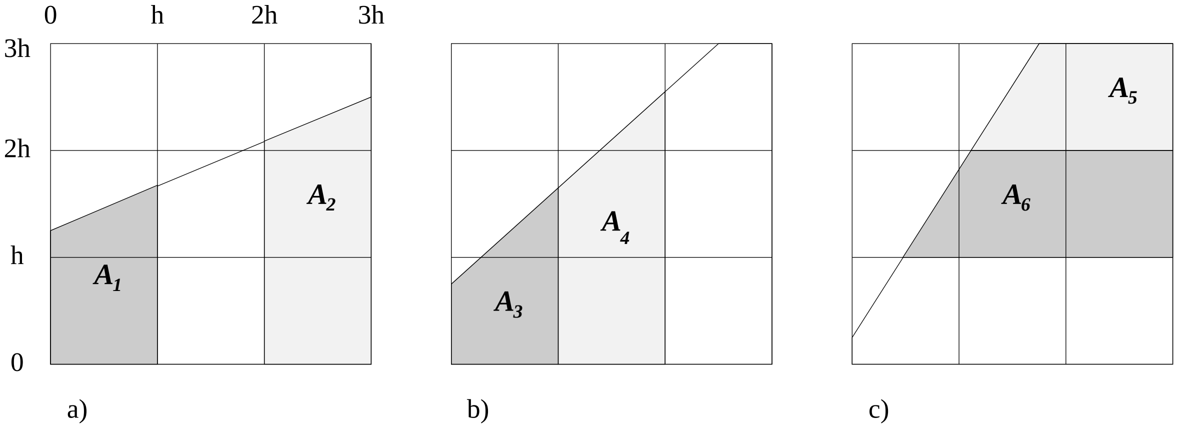


Abbildung 8: Drei mögliche Frontgeradenverläufe durch einen 3×3-Zellenblock.

und

$$A_2 = \frac{1}{h^2} \cdot h \cdot \frac{b + 2mh + b + 3mh}{2} = \frac{2b + 5mh}{2h}.$$

Die zentrale Differenz in x-Richtung aus (3.6) liefert

$$m_x^z = \frac{A_2 - A_1}{2} = \frac{2b + 5mh}{4h} - \frac{2b + mh}{4h} = m.$$

Die Ausgangsgerade und das rekonstruierte Geradenstück mit Steigung m_x^z haben also dieselbe Steigung. Da sie laut Voraussetzung auch dieselbe Füllung in der Zelle $Z_{i,j}$ erzeugen, sind sie identisch. ELVIRA wird eine solche Ausgangsgerade exakt reproduzieren, da für diese Gerade der E_2-Fehler null sein wird. Man mache sich klar, daß in diesem Fall alle drei durch die Steigungen in den Gleichungen (3.6) beschriebenen Geraden identisch sind.

In Abbildung 8b betrachten wir die Füllungssummen A_3 und A_4 der ersten und zweiten Spalte. Die durch Rückwärtsdifferenz berechnete Geradensteigung m_x^r ist dann

$$m_x^r = A_4 - A_3 = \frac{h}{h^2} \left(\frac{b + mh + b + 2mh}{2} - \frac{b + b + mh}{2} \right) = \frac{1}{h} \frac{2mh}{2} = m.$$

Hier liefert also die Rückwärtsdifferenz der Füllungssummen in den Spalten die richtige Geradensteigung.

Als letztes werfen wir einen Blick auf Abbildung 8c. Wir parametrisieren die Ausgangsgerade diesmal als $x = \tilde{m}y + \tilde{b}$ und erhalten als Vorwärtsdifferenz der Zeilensummen

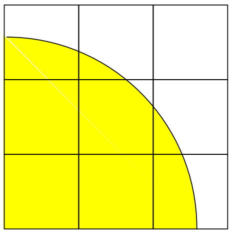
$$m_y^v = A5 - A6 = \frac{1}{h} \left(\frac{\tilde{b} + 2\tilde{m}h + \tilde{b} + 3\tilde{m}h}{2} - \frac{\tilde{b} + \tilde{m}h + \tilde{b} + 2\tilde{m}h}{2} \right) = \tilde{m}.$$

In diesem Falle liefert also die Vorwärtsdifferenz der Zeilen die richtige Lösung. Alle restlichen Fälle lassen sich durch Rotation und damit gegebenenfalls verbundener Vertauschung der Rollen von x und y oder durch Spiegelung aus diesen Fällen herleiten.

Der ELVIRA-Algorithmus ist also von zweiter Ordnung.

Wenn man wie hier nur lineare Fronten (Geraden) rekonstruiert, scheinen die zentralen Differenzen der Zeilen- und Spaltensummen überflüssig. Es hat sich jedoch gezeigt, daß die zentralen Differenzen in vielen Fällen nichtlinearer Fronten die geringsten E_2-Fehler aufweisen.

a)



b)

0.493	0.118	0.000
1.000	0.925	0.118
1.000	1.000	0.493

Abbildung 9: Symmetrie in den Zellfüllungen bereitet dem ELVIRA-Algorithmus Probleme.

3.3.10.1 Mehrere Minima der E_2-Fehlerkurve

Auf ein Problem der Algorithmen LVIRA und ELVIRA bei nichtlinearen Fronten sollte noch anhand eines Beispiels hingewiesen werden: In Abbildung 9a ist ein Kreisbogen dargestellt, der die Füllungen 9b erzeugt. Für die mittlere Zelle erwarten wir ein Geradenstück mit der Normalen $-45°$, da die Füllungen in 9b symmetrisch sind: $f_{i,j} = f_{j,i}$ für alle Zellen im Block. Sehen wir uns nun in Abbildung 10 die E_2-Fehlerkurve in Abhängigkeit vom Argument der Normalen an. Man sieht deutlich die beiden Minima des E_2-Fehlers, die übrigens sowohl von ELVIRA als auch von LVIRA gefunden werden. Der erwarteten Normalen entspricht die lokale Maximalstelle bei $-\frac{\pi}{2}$ genau zwischen den beiden globalen Minimalstellen.

Wir schließen daraus, daß die beiden Algorithmen LVIRA und ELVIRA in einigen Fällen ein Minimum des E_2-Fehlers finden, das aber, insbesondere bei mehreren Minima, nicht der erwarteten Normalen entspricht. Das ist ein erster Hinweis dafür, daß der von Pilliod postulierte Ordnungsbegriff nicht ganz das leistet, was er verspricht.

Dieses Problem tritt besonders deutlich zutage, wenn die erwartete Geradensteigung $m = \pm 1$ sein soll. Um dieses Problem zu umgehen, kann in beiden Algorithmen am Anfang ein Test auf eine Symmetrie wie in unserem Beispiel durchgeführt werden. Bei LVIRA realisiert man das, indem eine $\pm 45°$ Normale als Lösung zurückgegeben wird, wenn beide Startnäherungen aus dem Algorithmus von Hirt und Nichols annähernd dieselbe Geradensteigung liefern. Bei ELVIRA entspricht diesem Vorgehen eine Abfrage, ob die Beträge von m_x^z und m_y^z eng beieinander liegen, wobei gegebenenfalls die entsprechende $\pm 45°$-Normale als Lösung zurückgegeben wird. Diese

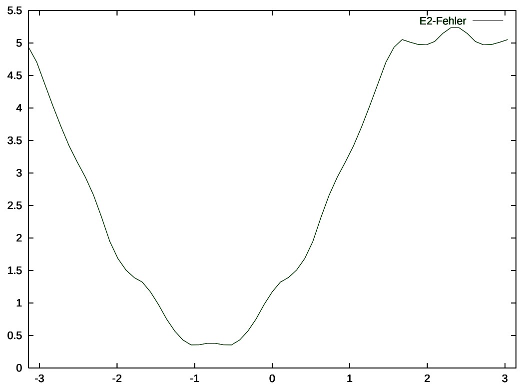


Abbildung 10: E_2-Fehlerkurve bei erwartetem Normalenargument $\theta = -\frac{\pi}{2}$.

Modifikationen retten zwar unseren Beispielfall, jedoch ist nicht allgemein geklärt, was bei multiplen Minima geschehen soll. Ein Trost ist, daß Tests gezeigt haben, daß mehrfache Minima hauptsächlich in solchen Symmetriefällen auftreten.

3.4 VOF-Advektionsalgorithmen

In diesem Abschnitt wollen wir auf verschiedene Advektionsalgorithmen der VOF-Methoden eingehen. Hat man mit einer der oben beschriebenen Methoden die Frontlinie rekonstruiert, interessiert man sich dafür, wie sich diese in einem Geschwindigkeitsfeld verändert.

In dieser Arbeit werden wir uns auf den Fall inkompressibler Strömungen beschränken. Auf der Basis von *volume-of-fluid*-Techniken lassen sich jedoch auch relativ einfach massenerhaltende Algorithmen zur Lösung kompressibler Probleme formulieren. Siehe hierzu zum Beispiel die Arbeit von Miller und Puckett [19].

Für Advektionsalgorithmen verwenden wir den Ordnungsbegriff von Pilliod [24]: Ein Advektionsverfahren ist zweiter Ordnung, wenn es in einem konstanten Geschwindigkeitsfeld zusammen mit einem Rekonstruktionsverfahren zweiter Ordnung Geraden so transportiert, daß diese Geraden bleiben.

In Abschnitt 3.4.1 stellen wir einige allgemeine Überlegungen zur numerischen Behandlung der Advektionsgleichung an. Danach beschäftigen wir uns in Abschnitt 3.4.2 mit Advektionsalgorithmen, die auf sogenanntes *operator splitting* zurückgreifen, und in Abschnitt 3.4.3 mit *unsplit*-Advektionsalgorithmen.

3.4.1 Diskretisierung der Erhaltungsgleichung

Betrachten wir zum Zeitpunkt $t = t^n$ die Zelle $Z_{i,j}$, die die Ecken (x_i, y_j), (x_{i+1}, y_j), (x_i, y_{j+1}) und (x_{i+1}, y_{j+1}) besitzt. Hochgestellte Indizes sollen die diskreten Zeitpunkte kennzeichnen. Wir suchen ein Verfahren, um die Füllungen zum nächsten Zeitpunkt $t^{n+1} = t^n + \Delta t$ zu berechnen. Dazu integrieren wir beide Seiten der Advektionsgleichung (2.4) über die betrachtete Zelle und über das Intervall des Zeitschritts und erhalten

$$\int_{t_n}^{t_n+\Delta t} \int_{y_j}^{y_{j+1}} \int_{x_i}^{x_{i+1}} \partial_t f + \partial_x(uf) + \partial_y(vf)\, dx\, dy\, dt = 0. \tag{3.8}$$

Mit der Definition

$$\mathbf{F}(x,y,t) := (f(x,y,t),\ u(x,y) \cdot f(x,y,t),\ v(x,y) \cdot f(x,y,t))^T$$

entspricht der Integrand in (3.8) der Divergenz von $\mathbf{F}$. Wir können daher den Gaußschen Integralsatz anwenden und erhalten Integrale über die Randflächen des betrachteten Integrationsquaders. Da die Flächennormalen jeweils senkrecht auf zwei Koordinatenachsen stehen, vereinfacht sich (3.8) mit der Definition (3.1) zu

$$f_{i,j}^{n+1} = f_{i,j}^n - \frac{F_{i+1,j}^n - F_{i,j}^n + G_{i,j+1}^n - G_{i,j}^n}{\Delta x \Delta y}, \tag{3.9}$$

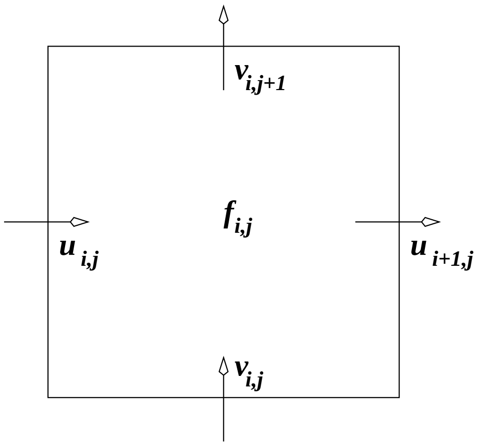


Abbildung 11: Die *staggered-grid*-Anordnung der physikalischen Größen auf dem Rechengitter. Die Pfeile zeigen die jeweilige positive Richtung der entsprechenden Geschwindigkeitskomponente an.

wobei

$$\begin{aligned} F^n_{i+1,j} &= \int_{t_n}^{t_n+\Delta t} \int_{y_j}^{y_{j+1}} (uf)(x_{i+1}, y, t)\, dy\, dt, \\ F^n_{i,j} &= \int_{t_n}^{t_n+\Delta t} \int_{y_j}^{y_{j+1}} (uf)(x_i, y, t)\, dy\, dt, \\ G^n_{i,j+1} &= \int_{t_n}^{t_n+\Delta t} \int_{x_i}^{x_{i+1}} (vf)(x, y_{j+1}, t)\, dx\, dt \qquad \text{und} \\ G^n_{i,j} &= \int_{t_n}^{t_n+\Delta t} \int_{x_i}^{x_{i+1}} (vf)(x, y_j, t)\, dx\, dt \end{aligned} \tag{3.10}$$

die *Volumenflüsse* in einem Zeitschritt Δt über die entsprechende Kante der Zelle, in x-Richtung als F und in y-Richtung als G, bezeichnen. Für die Kanten am Rand des Rechengebiets setzen wir den Volumenfluß gleich null (*homogene von-Neumann-Randbedingungen*).

Man sagt, die Formel (3.9) zur Aktualisierung der Füllung $f_{i,j}$ hat *Erhaltungsform*. Berechnet man nämlich alle neuen Zellfüllungen nach dieser Vorschrift, bleibt die Summe der Füllungen im gesamten Gebiet erhalten, da für je zwei benachbarte Zellen der Volumenfluß über die gemeinsame Kante einmal addiert und einmal subtrahiert wird.

Da die Geschwindigkeiten u und v nur noch an den Kanten benötigt werden, empfiehlt sich die verschobene Anordnung (englisch *staggered-grid*) der physikalischen Größen auf unserem Gitter: Die Werte der Geschwindigkeit u in x-Richtung werden an den Kantenmittelpunkten der senkrechten Kanten angegeben, während die Werte von v auf den waagerechten Kantenmittelpunkten angegeben werden (siehe hierzu Abbildung 11). An manchen Stellen werden wir eine Diskretisierung für die partiellen Ableitungen der Geschwindigkeitskomponenten benötigen. Wir verwenden dann zentrale Differenzen, also

$$(\partial_x u)_{i,j} = \frac{u_{i+1,j} - u_{i,j}}{\Delta x} \quad \text{und} \quad (\partial_y v)_{i,j} = \frac{v_{i+1,j} - v_{i,j}}{\Delta y}. \tag{3.11}$$

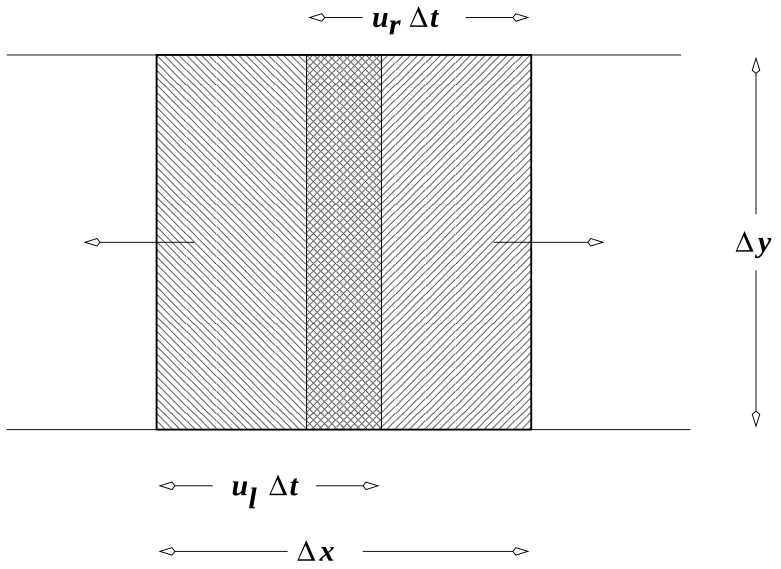


Abbildung 12: Die CFL-Bedingung regelt, daß in einem Zeitschritt nicht mehr Volumen die Zelle verlassen kann, als sie enthält.

Wir werden auch annehmen, daß sich die Inkompressibilitätsbedingung (2.3) auf die Diskretisierung überträgt, das heißt, es gilt

$$\frac{u_{i+1,j} - u_{i,j}}{\Delta x} + \frac{v_{i,j+1} - v_{i,j}}{\Delta y} \approx 0 \tag{3.12}$$

auf dem gesamten Gitter.

3.4.1.1 CFL-Bedingung und Frontband

Unser Advektionsverfahren ist explizit in der Zeit, weil wir zur Berechnung der Füllungen auf der neuen Zeitschicht ausschließlich Daten der alten Zeitschicht benutzen werden. Man muß daher eine CFL-Bedingung[8] stellen, um sicherzustellen, daß das numerische Abhbängigkeitsgebiet das physikalische einschließt.

Was bedeutet das in unserem Fall? Betrachten wir den waagerechten Fluß aus einer Zelle in Abbildung 12. Das von unserem diskreten Verfahren maximal in einem Zeitschritt über die linke Kante transportierte Volumen entspricht der Fläche $u_l \Delta t \Delta y$, also der linken schraffierten Fläche, das auf der rechten Seite $u_r \Delta t \Delta y$ der rechten schraffierten Fläche. Schlimmstenfalls wechselt u in der Zelle das Vorzeichen. Wählt man Δt zu groß, könnte es sein, daß sich die beiden Bereiche wie in der Abbildung überschneiden. Im Zeitschritt Δt würde der Überschneidungsbereich sowohl über die rechte wie auch über die linke Kante die Zelle verlassen, das heißt, die Zelle würde mehr Flüssigkeit abgeben als sie enthält. Um das zu vermeiden, wählt man Δt so klein, daß

$$\Delta t \leq \frac{\Delta x}{2 \max\limits_{i,j} |u_{i,j}|} \quad \text{und} \quad \Delta t \leq \frac{\Delta x}{2 \max\limits_{i,j} |v_{i,j}|} \tag{3.13}$$

[8] nach Courant, Friedrichs und Levy

oder

$$\Delta t \leq \frac{\Delta x}{\max\limits_{i,j}(u_{i+1,j} - u_{i,j})} \quad \text{und} \quad \Delta t \leq \frac{\Delta x}{\max\limits_{i,j}(v_{i,j+1} - v_{i,j})} \tag{3.14}$$

gilt. Die Bedingung in (3.13) liefert für gewöhnlich kleinere Zeitschritte als die in (3.14). Die CFL-Zahl sowie die Methode der Zeitschrittberechnung werden im Programmteil model.c definiert. Die Berechnung des Zeitschritts findet im Programmteil cfl.c statt.

Aus der obigen Argumentation folgt außerdem, daß ein Fluidpartikel sich bei erfüllter CFL-Bedingung in einem Zeitschritt maximal eine Zelle weit fortbewegen kann. Diese Tatsache kann man sich zunutze machen: Nimmt man etwas Abstand zur Frontlinie, so findet man nur volle oder leere Zellen vor (denn sonst wären sie ja Frontzellen), die von Zellen gleicher Füllung umgeben sind. Das bedeutet aber, daß in solchen Bereichen $\partial_x f = \partial_y f = 0$ ist. Mit der Erhaltungsgleichung (2.4) folgt dort

$$\partial_t f = -\left(\partial_x \left(uf\right) + \partial_y \left(vf\right)\right) = -\left(\left(\partial_x u\right) f + \left(\partial_y v\right) f\right) = -(\partial_x u + \partial_y v) f = 0$$

wegen der Divergenzfreiheit des Geschwindigkeitsfeldes im inkompressiblen Fall. Da sich die charakteristische Fluidfunktion in solchen Bereichen nicht ändert, ändert sich dort auch die Füllung nicht. Insgesamt haben wir damit gezeigt, daß sich im hier behandelten inkompressiblen Fall alle Änderungen der Füllungen nur in unmittelbarer Nachbarschaft der Front abspielen. Wir müssen also tatsächlich in einem Zeitschritt alle Berechnungen nur auf den Frontzellen und deren Nachbarn durchführen[9]. Diese Tatsache reduziert die Komplexität des gesamten Frontenrekonstruktionsvorgangs in 2D von $O(n^2)$ auf $O(n)$, wenn n die Anzahl der Diskretisierungspunkte in einer Richtung ist, weil die Front eine Mannigfaltigkeit der Dimension eins ist. Die Detektion des daraus resultierenden *Frontbandes* erfolgt im Programmteil frontutil.c.

Man beachte, daß diese Reduktion des Rechenaufwands nur bei Geschwindigkeitsfeldern inkompressibler Strömungen möglich ist.

3.4.1.2 Behandlung von Füllungsfehlern

Egal, wie sauber man einen Advektionsalgorithmus implementiert hat: Durch Rundungsfehler kann es immer vorkommen, daß Zellen, die eigentlich voll bzw. leer sein sollten, nur fast voll bzw. fast leer sind. Das führt zu einem Verschmieren der Front auf einer Breite mehrerer Zellen, was in einem Frontenverfolgungsalgorithmus ein bestimmt ungewollter Effekt ist. Man sucht daher nach Strategien, an diesen Stellen die Front „schärfer“ zu machen. Solche Probleme kennt man zum Beispiel aus der Bildverarbeitung, wenn Konturen aus Bildern herausgearbeitet werden sollen. Das entspricht oft einer in der Zeit rückwärtsgerichteten Diffusion. Mathematisch ist diese Aufgabe nicht korrekt gestellt, und die Berechnung erweist sich als äußerst instabil. Gewöhnlicherweise führt dieser Ansatz zu starken Oszillationen in der Lösung. In unserem Fall sind die Füllungswerte jedoch beschränkt, $0 \leq f_{i,j} \leq 1$, so daß diese Oszillationen nicht unbegrenzt wachsen können.

Wir haben unser Advektionsschema in Gleichung (3.9) auf Erhaltungsform gebracht. Somit ist die zu erwartende Gesamtvolumenänderung selbst bei fehlerhaften Flußberechnungen in der Größenordnung der Maschinengenauigkeit, da man den Fluß über eine Kante jeweils bei der

[9] Die Flußberechnungen muß man natürlich noch an allen Kanten, die an Frontnachbarzellen grenzen, ausführen.

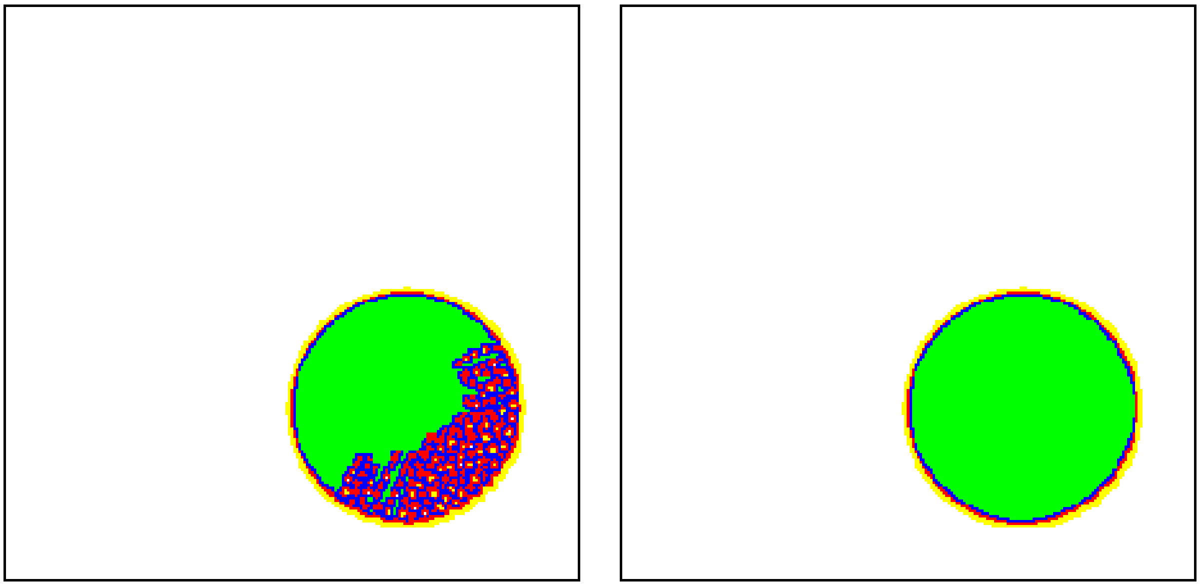


Abbildung 13: Durch die Wahl eines zu geringen Wertes für $\epsilon_{cleanup}$ kommt es zu Blasenbildung in der Front (links). Im rechten Bild ist ein größerer Wert gewählt worden.

Füllung einer angrenzenden Zelle addiert und bei der anderen Zelle subtrahiert. Es kann allerdings passieren, daß durch Fehler in den Flüssen negative Füllungen oder Füllungen, die größer als eins sind, auftreten. Dieses Phänomen ist besonders bei Geschwindigkeitsfeldern mit großen lokalen Schwankungen in Bereichen des Vorzeichenwechsels der Geschwindigkeiten zu bemerken (siehe die Abschnitte 3.5.6 und 3.5.7).

Auf jeden Fall sollte man, um die Stabilität des Verfahrens zu verbessern, nach jedem Advektionsschritt einen Säuberungsschritt über alle Füllungen durchführen. Üblicherweise setzt man zu einem festen $\epsilon_{cleanup} > 0$

$$\begin{aligned} f_{i,j} &\leftarrow 0 \quad \text{wenn} \quad f_{i,j} < \epsilon_{cleanup}, \\ f_{i,j} &\leftarrow 1 \quad \text{wenn} \quad f_{i,j} > 1 - \epsilon_{cleanup}. \end{aligned} \tag{3.15}$$

Bei der Wahl von $\epsilon_{cleanup}$ sollte man äußerst behutsam vorgehen: Ein zu großer Wert schädigt ernsthaft die Volumenerhaltungseigenschaft des Algorithmus', während ein zu kleiner Wert zu wenig von den Füllungsfehlern eliminiert, so daß die Front verschmiert. Eine unvorsichtige Wahl von $\epsilon_{cleanup}$ führt zu schlechten Ergebnissen mit überraschenden Bildern, wie sie in Abbildung 13 zu sehen sind. Die Implementierung sieht eine Adaption vor: Gilt in einer Zelle $f_{i,j} < -\epsilon_{cleanup}$ oder $f_{i,j} > 1 + \epsilon_{cleanup}$, setzt man $\epsilon_{cleanup} \leftarrow -f_{i,j}$ bzw. $f_{i,j} \leftarrow f_{i,j} - 1$ und führt den Säuberungsschritt auf dieser Zeitschicht erneut durch. Dieses Vorgehen gegen Füllungsfehler nennen wir im Folgenden *einfache Bereinigung*.

Werden die Verletzungen der Füllungsbeschränkung zu groß, muß man sich eine bessere Strategie ausdenken, weil man sonst ernsthaft die Volumenerhaltung verschlechtert. Man nennt solche Fehler auch *numerische Kompressibilität* (englisch *numerical compressibility*), da sie einem kompressiblen Term in der Erhaltungsgleichung (2.4) entsprechen. Man kann sich das so vorstellen, daß durch Kompression von einem kompressiblen Fluid mehr in eine Zelle gepreßt werden darf als paßt.

Kothe und Rider schlagen zur Beseitigung dieser Fehler eine Umverteilung der Füllungsüberschüsse bzw. -defizite auf benachbarte Frontzellen vor. Dieser heuristische Rettungsversuch fordert, wenn man ihn ernsthaft durchführen will, einen kompletten Optimierungsalgorithmus, der nach jedem Advektionsschritt die Fehler so umverteilt, daß möglichst alle Zellen gültige Füllungswerte besitzen. Selbst diese Methode garantiert jedoch nicht immer die Wiederherstellung der Füllungsbeschränkung.

Diese Art der Fehlerbereinigung nennen wir *Umverteilungsbereinigung*. In der Implementierung wird versucht, die auftretenden Fehler auf benachbarte Frontzellen zu verteilen, wobei allerdings eine feste Bearbeitungsreihenfolge eingehalten wird. Dadurch kann es zu Einschlüssen solcher Fehler in Zellen kommen, die nur noch von vollen Zellen umgeben sind, auf die nichts mehr umverteilt werden kann. Um trotzdem die Stabilität der restlichen Berechnungen zu gewährleisten, wird dann der Restfehler einfach abgeschnitten, was sich in einem sprunghaften Anstieg des Volumenfehlers bemerkbar macht. Dies ist der einzige Punkt, an dem ein *volume-of-fluid*-Algorithmus ernsthaft Massenfehler erzeugt.

Die Implementierung sieht noch eine dritte Möglichkeit der Fehlerbereinigung vor, nämlich einfach gar nichts zu tun. Man beobachtet dann fast immer Instabilitäten (Verschmieren der Front oder gar Abbruch der Rechnung) bei fast allen Algorithmen.

3.4.2 *Operator-Split*-Advektionsmethoden

Das Hauptmerkmal von *operator-split*-Methoden ist, daß sie die mehrdimensionale Lösung durch sukzessive Behandlung eindimensionaler Probleme ermitteln. Die Hauptprobleme bei diesem Ansatz bestehen zum einen in der Willkür der Reihenfolge der Behandlung der eindimensionalen Probleme und zum anderen in der Verstärkung von Gittereffekten.

In unserem zweidimensionalen Fall lassen sich alle *operator-split*-Verfahren als Zweischrittverfahren schreiben:

$$f_{i,j}^{n+\frac{1}{2}} = f_{i,j}^{n} - \frac{F_{i+1,j} - Fi,j}{\Delta x \Delta y} \tag{3.16}$$

und

$$f_{i,j}^{n+1} = f_{i,j}^{n+\frac{1}{2}} - \frac{G_{i+1,j} - Gi,j}{\Delta x \Delta y}, \tag{3.17}$$

wobei man erst mittels Gleichung (3.16) alle Werte $f_{i,j}^{n+\frac{1}{2}}$ als Zwischenwerte berechnet, und dann aus Gleichung (3.17) die endgültigen Werte $f_{i,j}^{n+1}$ auf der neuen Zeitschicht erhält. Die meisten Verfahren wechseln in jedem Zeitschritt die Reihenfolge der Behandlung von x- und y-Richtung und erhalten so eine höhere Konvergenzordnung.

3.4.2.1 *Donor-Acceptor*-Methode

Diese Methode wird von vielen SLIC-Methoden benutzt. Die hier vorliegende Beschreibung geht auf Hirt und Nichols [14] zurück. Wir wollen die Methode exemplarisch an einer senkrechten Kante anwenden. Waagerechte Kanten werden in einem zweiten Advektionsschritt analog behandelt.

Zuerst wird anhand des Vorzeichens von u im Kantenmittelpunkt ermittelt, in welche Richtung die Flüssigkeit strömt. Die stromabwärts liegende Zelle ist dann die Empfänger- (englisch *acceptor*) und die stromaufwärts liegende die Spenderzelle (englisch *donor*). Das ist in Abbildung 14a dargestellt. In unserem Falle sei f_D die Füllung der Spender- und f_A die der Empfängerzelle.

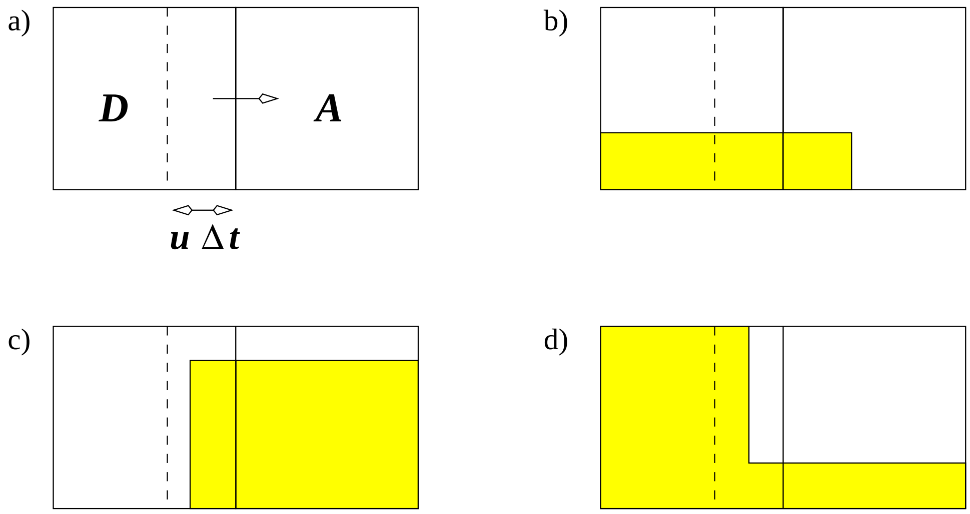


Abbildung 14: Die treppenförmige Rekonstruktion der Front beim Spender-Empfänger-Verfahren

Der *donor-acceptor*-Algorithmus rekonstruiert die Front implizit treppenförmig. Das heißt, daß Frontlinien benachbarter Zellen einander an der gemeinsamen Kante berühren. Das soll in den Abbildungungen 14b bis d veranschaulicht werden.
In Fall 14b berechnet sich der Volumenfluß $F_{i,j}$ über die Kante dann als

$$F_{i,j} = u_{i,j}\Delta t \Delta y \cdot f_D.$$

Im Falle 14c würde diese Art der Berechnung einen zu großen Fluß liefern. Die Spenderzelle kann nicht mehr Flüssigkeit abgeben als sie enthält. Der Fluß wird hier berechnet als

$$F_{i,j} = \Delta x \Delta y \cdot f_D.$$

Ähnlich würde im Fall 14d ein zu großer Fluß berechnet. Stattdessen gilt hier die Formel

$$F_{i,j} = u_{i,j}\Delta t \Delta y - (1 - f_D)\Delta x \Delta y.$$

Alle diese Fallunterscheidungen werden in den Formeln

$$F_{i,j} = \min(f_{AD}|u_{i,j}|\Delta t \Delta y + CF, f_D \Delta x \Delta y) \tag{3.18}$$

und

$$CF = \max\left((1 - f_{AD})\,|u_{i,j}|\Delta t \Delta y - (1 - f_D)\,\Delta x \Delta y, 0\right) \tag{3.19}$$

zusammengefaßt. Kurz gesagt, sorgt das Minimum in (3.18) dafür, daß nicht mehr Fluid aus der Spenderzelle strömen kann als sie enthält, während das Maximum in (3.19) dafür sorgt, daß eventuell etwas mehr Fluid transportiert wird, falls nicht genügend Luft in der Spenderzelle enthalten ist.

In den obigen Gleichungen (3.18) und (3.19) bezeichnet f_{AD} entweder die Füllung der Spender- oder der Empfängerzelle. Die Wahl von $f_{AD} = f_A$ führt zu einem Aufstauen des Fluids in Advektionsrichtung, das heißt, daß sich die Spenderzelle erst füllt, bevor sie Fluid abgeben kann, während $f_{AD} = f_D$ die Leerung der Spenderzelle möglich macht. Die Entscheidung, welche Füllung nun als f_{AD} gewählt wird, fällt in Frontzellen aufgrund der Normalen der Front: Liegt die berechnete Flußrichtung eher normal als tangential zur Front, wird $f_{AD} = f_A$ gesetzt, anderenfalls $f_{AD} = f_D$. Außerdem wird grundsätzlich $f_{AD} = f_A$ verwendet, wenn die Empfängerzelle leer ist, damit sich die Spenderzelle füllt, bevor sie Fluid an die leere Nachbarzelle abgibt. Sind Spender- und Empfängerzelle voll, so ist die Wahl unerheblich, und leere Zellen erzeugen keine Flüsse.

Hirt und Nichols rechtfertigen diese Fallunterscheidung für f_{AD} damit, daß dadurch ein Anschwellen von Oberflächenwellen verhindert wird. Man stelle sich ein horizontales Frontstück vor, auf dem eine kleine Welle in die positive x-Richtung läuft. Würde man immer die Empfängerzelle zur Bestimmung der Flüsse benutzen, würde diese Welle immer stärker anschwellen. Tatsächlich ist die reine Empfängermethode instabil, weil sie negative Diffusion in f erzeugt (siehe [14]). Durch die Minimum- und Maximumtests wachsen solche Unstetigkeiten jedoch nicht unbegrenzt. Im Falle der Advektion normal zur Front ist ein Aufstauen genau der erwünschte Effekt, da dadurch ein Verschmieren der Front verhindert wird. Die reine Spender-Methode wäre in diesem Fall zu diffusiv und würde die Front in jedem Zeitschritt weiter über mehrere Zellen verteilen.

Hat man alle Flüsse in eine Richtung berechnet, führt man gemäß (3.16) eine Aktualisierung der Zellfüllungen durch. Dann berechnet man die Flüsse in die andere Koordinatenrichtung, wozu man die Front erneut rekonstruieren muß. Danach benutzt man (3.17) für eine erneute Aktualisierung, und fertig ist der Advektionsschritt.

Wie im Abschnitt 3.4.1.1 beschrieben, muß man, wie bei allen expliziten Advektionsalgorithmen, die CFL-Bedingung für die Größe des Zeitschritts beachten.

Das ganze Verfahren wirkt etwas willkürlich. Insbesondere die zuletzt genannten Fallunterscheidungen verwirren zunächst. Hirt und Nichols verwendeten zur Frontenrekonstruktion stückweise lineare Frontsegmente, doch die dort gewonnene Information geht hier nur bei der Fallunterscheidung für f_{AD} ein, und dort würde eine stückweise konstante Rekonstruktion dieselben Flüsse liefern. Wesentlich ist eben, daß dieser Advektionsalgorithmus die Front implizit stückweise treppenförmig rekonstruiert. Bei den Testrechnungen produzierte dieser Algorithmus außerdem *flotsam* und *jetsam*.

3.4.2.2 *Fractional-Step*-Methode

Die etwas unphysikalische und rein auf Stabilität ausgerichtete Flußberechnung in Abschnitt 3.4.2.1 liefert bei schwierigeren Problemen sehr schlechte Lösungen. Wir wollen nun eine eher geometrisch-anschauliche Ermittlung der Flüsse betrachten, in der auch die Information aus der Frontrekonstruktionsphase voll ausgenutzt wird. Eine solche Methode ist die *fractional-step*-Methode, die von Youngs [22] auf *volume-of-fluid*-Algorithmen übertragen wurde.

In Abbildung 15 sehen wir die Frontrekonstruktion in einer Zelle, an deren rechter Kante der Fluß in x-Richtung berechnet werden soll. Das numerische Abhängigkeitsgebiet dieses Flusses ist das markierte Rechteck $ABED$ mit den Seitenlängen $u\Delta t$ und Δy. Die Idee ist nun, daß alles, was sich an Flüssigkeit in diesem Rechteck befindet, im Zeitschritt Δt über die Kante fließen wird. Der Volumenfluß entspricht der dunkel eingefärbten Fläche, also dem Integral

$$F_{i,j} = \int_{ABED} f \, dx \, dy$$

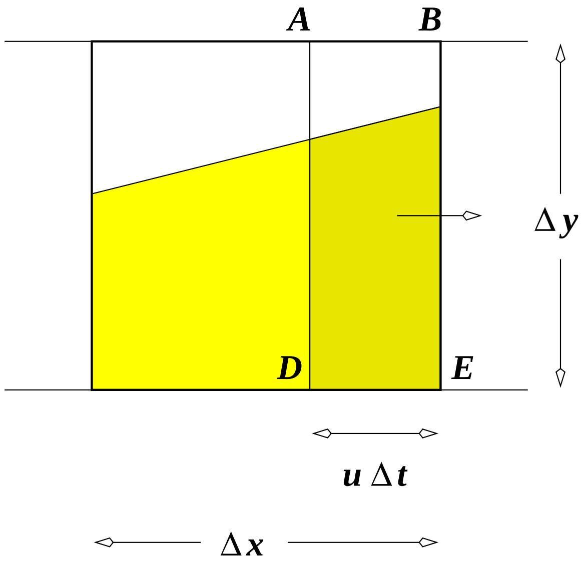


Abbildung 15: Beim *fractional-step*-Verfahren berechnet man den Fluß durch eine Kante aus dem Volumen des Fluids in dem Rechteck der Seitenlängen $u\Delta t$ und Δy.

der charakteristischen Fluidfunktion f über dem Rechteck $ABED$.

Zur Berechnung des Integrals muß man also dieses Rechteck am Frontliniensegment abschneiden und den Flächeninhalt des resultierenden Polygons berechnen. Siehe hierzu auch den Abschnitt 3.3.2.

Wir berechnen auf diese Weise zunächst alle Flüsse $F_{i,j}$ in x-Richtung und bilden daraus die Zwischenwerte der Füllungen mittels (3.16). Dann müssen wir die Flüsse $G_{i,j}$ in y-Richtung ermitteln, wozu die Front aus den Füllungszwischenwerten neu zu rekonstruieren ist, und benutzen (3.17), um die Füllungswerte $f_{i,j}^{n+1}$ auf der neuen Zeitschicht zu erhalten. Indem man in jedem Zeitschritt die Reihenfolge von x-Advektion und y-Advektion vertauscht, erhält man ein Verfahren zweiter Ordnung.

Wegen der Anwendung von *operator splitting* muß man in jedem Zeitschritt die Front zweimal rekonstruieren. Es sind auch verstärkt Gittereffekte zu erwarten. Außerdem funktioniert die Methode bei Geschwindigkeitsfeldern mit großer Wirbelstärke nicht mehr. Für einfache Geschwindigkeitsfelder sind die Ergebnisse jedoch sehr gut.

3.4.3 *Unsplit*-Advektionsmethoden

Für Fälle, in denen *operator-split*-Methoden versagen, sucht man nach ungespaltenen Methoden (englisch *unsplit methods*). Statt mit Zwischenergebnissen zu arbeiten, will man alle Advektionsrichtungen in einem Schritt bearbeiten und aus den Flüssen F und G die neuen Füllungen ermitteln als

$$f_{i,j}^{n+1} = f_{i,j}^{n} - \frac{F_{i+1,j}^{n} - F_{i,j}^{n} + G_{i,j+1}^{n} - G_{i,j}^{n}}{\Delta x \Delta y}. \tag{3.20}$$

Die Grundidee ist ähnlich wie bei der *fractional-step*-Methode, nämlich daß man zunächst das Gebiet ermittelt, dessen Flüssigkeit im Zeitschritt durch die untersuchte Kante transportiert wird, und dann das Volumen dieser Flüssigkeit als Volumenfluß berechnet.

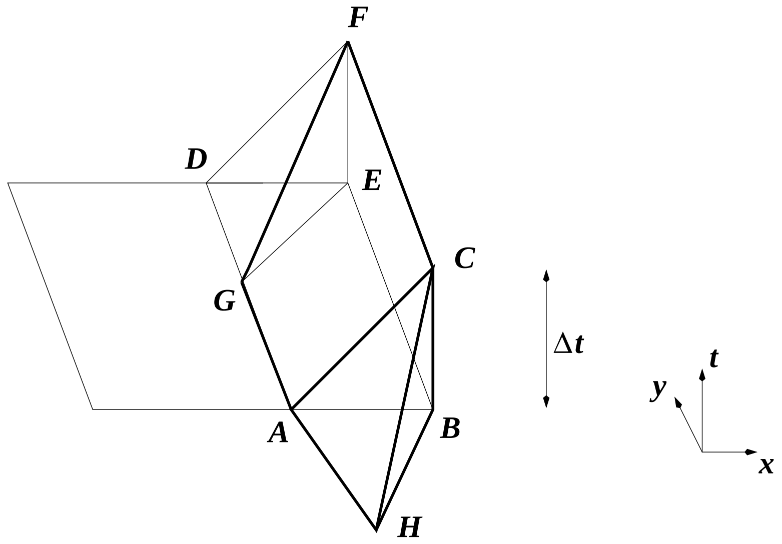


Abbildung 16: Das charakteristische Polyeder zur Kante $BCFE$

Wir wollen den Volumenfluß $F_{i+1,j}$ über die rechte Kante der Zelle $Z_{i,j}$ im Zeitintervall $[t^n, t^n + \Delta t]$ berechnen. Abbildung 16 soll zur Veranschaulichung der Ermittlung des *charakteristischen Polyeders* dienen. Nach der Definition des Flusses in (3.10) entspricht dieser dem Integral von uf über das Rechteck $BCFE$, wobei

$$C = B + (0, 0, \Delta t)^T \quad \text{und} \quad F = E + (0, 0, \Delta t)^T$$

sind[10]. Wir gehen davon aus, daß u und v entlang jeder einzelnen Kante konstant sind, das heißt, wir rekonstruieren die Geschwindigkeiten auf den Kanten konstant aus den diskreten Werten. Wir erhalten die Punkte A und H, indem wir die Charakteristiken, die den Punkt C schneiden, in der Zeit zurückverfolgen. Es ist

$$A = C - \Delta t \begin{pmatrix} u_{i+1,j} \\ 0 \\ 1 \end{pmatrix}$$

und

$$H = \begin{cases} C - \Delta t \begin{pmatrix} u_{i+1,j-1} \\ v_{i,j} \\ 1 \end{pmatrix} & \text{für } v_{i,j} > 0, \\ C - \Delta t \begin{pmatrix} u_{i+1,j} \\ v_{i,j} \\ 1 \end{pmatrix} & \text{für } v_{i,j} \leq 0. \end{cases} \tag{3.21}$$

[10] In Abbildung 16 sieht es wegen einer optischen Täuschung leider nicht so aus, als würde C (F) genau über B (E) stehen.

Ebenso erhalten wir die Punkte D und G durch Rückverfolgung der Charakteristiken durch F als

$$D = F - \Delta t \begin{pmatrix} u_{i+1,j} \\ 0 \\ 1 \end{pmatrix}$$

und

$$G = \begin{cases} F - \Delta t \begin{pmatrix} u_{i+1,j} \\ v_{i,j+1} \\ 1 \end{pmatrix} & \text{für } v_{i,j+1} > 0, \\ F - \Delta t \begin{pmatrix} u_{i+1,j+1} \\ v_{i,j+1} \\ 1 \end{pmatrix} & \text{für } v_{i,j+1} \leq 0. \end{cases}$$

Das charakteristische Polyeder ist das numerische Abhängigkeitsgebiet des Volumenflusses durch die Kante $BCFE$.

3.4.3.1 *Corner-Transport-Upwind*-Methode

Motiviert durch die Abbildung 16, approximieren wir den Volumenfluß $F_{i+1,j}$ als

$$F_{i+1,j} \approx \iint\limits_{ABED} f\,dx\,dy \;-\; \operatorname{sign}(v_{i,j}) \cdot \iint\limits_{ABH} f\,dx\,dy \;+\; \operatorname{sign}(v_{i,j+1}) \cdot \iint\limits_{DEG} f\,dx\,dy,$$

wobei die Flächenintegrale, die jeweils dem Volumen des im Integrationsbereich enthaltenen Fluids entsprechen, wieder durch *clipping* der entsprechenden Bereiche mit der rekonstruierten Front und anschließender Volumenberechnung des Fluidpolygons (siehe Abschnitt 3.3.2) ausgewertet werden.

Man berechnet auf diese Weise also die Volumenflüsse F in x-Richtung und G in y-Richtung und aktualisiert mittels (3.20) die Zellfüllungen.

Dieser Algorithmus wird von Colella [11] *corner-transport-upwind*-Schema genannt (kurz CTU). Er ist nur erster Ordnung.

3.4.3.2 Pilliods Methode

Wie kann man den CTU-Algorithmus verbessern? Die folgende Methode wurde von Bell, Dawson und Shubin [9] für skalare hyperbolische Erhaltungsgleichungen entwickelt und von Pilliod [24] auf *volume-of-fluid*-Methoden übertragen.

Wir wollen wieder die rechte Kante einer Zelle betrachten. Für unsere charakteristische Flüssigkeitsfunktion gilt die Erhaltungsgleichung (2.4). Wir linearisieren diese in x-Richtung und erhalten so unter der Annahme, daß die Geschwindigkeitskomponenten auf den Kanten konstant sind

$$\partial_t f + u_{i+1,j} \partial_x f + (\partial_x u)_{i,j} f + \partial_y (v f) = 0. \tag{3.22}$$

Wir integrieren nun die Gleichung (3.22) über das Polyeder $ABCDEF$

$$\iiint_{ABCDEF} \partial_t f + u_{i+1,j}\partial_x f + (\partial_x u)_{i,j} f + \partial_y(vf)\, dx\, dy\, dt = 0.$$

In dieser Form integrieren wir jeden Summanden für sich und benutzen dazu den Gaußschen Integralsatz. Es ergibt sich

$$\begin{aligned}0 &= \iiint_{ABCDEF} \partial_t f + u_{i+1,j}\partial_x f + \partial_y(vf) + (\partial_x u)_{i,j} f\, dx\, dy\, dt \\ &= -\iint_{ABED} f\, dx\, dy + u_{i+1,j}\iint_{BCEF} f\, dy\, dt - \iint_{ABC} vf\, dx\, dt + \iint_{DEF} vf\, dx\, dt + \\ &\quad + \iiint_{ABCDEF} (\partial_x u)_{i,j} f\, dx\, dy\, dt.\end{aligned}$$

Aufgrund der Normalen der Seitenflächen des charakteristischen Polyeders sind viele Randintegrale null. Der zweite Summand entspricht dem Volumenfluß $F_{i+1,j}$ über die betrachtete Kante, weil wir vorausgesetzt haben, daß die Geschwindigkeitskomponenten entlang der Kanten konstant sind. Wir formen nach $F_{i+1,j}$ um, und erhalten

$$F_{i+1,j} = \iint_{ABED} f\, dx\, dy + \iint_{ABC} vf\, dx\, dt - \iint_{DEF} vf\, dx\, dt - \iiint_{ABCDEF} f(\partial_x u)_{i,j}\, dx\, dy\, dt. \tag{3.23}$$

Das erste Integral läßt sich wie beim CTU-Algorithmus berechnen, indem wir mit einem Frontenrekonstruktionsalgorithmus die Front in der Zelle rekonstruieren, das Rechteck $ABED$ entlang der Frontlinie abschneiden und das Volumen des Fluidpolygons berechnen. Den so erhaltenen Wert I_{ABED} des Integrals benötigen wir auch für eine Approximation des Volumenintegrals über das Prisma $ABCDEF$: Wir nähern einfach

$$\begin{aligned}\iiint_{ABCDEF} f(\partial_x u)_i, j\ dx\ dy\ dt &= (\partial_x u)_{i,j} \cdot \mathrm{Vol}(ABCDEF) \cdot I_{ABED}/\mathrm{Vol}(ABED) \\ &= \frac{u_{i+1,j} - u_{i,j}}{\Delta x} \cdot u_{i+1,j}\Delta y \frac{(\Delta t)^2}{2} \cdot I_{ABED}/(u_{i+1,j}\Delta t \Delta y).\end{aligned}$$

Die Integrale über die Dreiecke ABC und DEF müssen noch weiter umgeformt werden. Dabei müssen wir berücksichtigen, daß die Tetraeder $DEFG$ und $ABCH$, je nach dem Vorzeichen von v auf der entsprechenden Kante, auf der betrachteten Zelle selbst oder in der oberen bzw. unteren Nachbarzelle stehen. Für den Fall, der in der Abbildung 16 dargestellt ist, erhalten wir eine Bestimmungsgleichung für das Integral über DEF, indem wir die Gleichung (3.22) auch noch in y-Richtung linearisieren

$$\partial_t f + u_{i,j}\partial_x f + v_{i,j+1}\partial_y f + (\partial_x u)_{i,j} f + (\partial_y v)_{i,j} f = 0$$

und über das Tetraeder $DEFG$ integrieren

$$\iiint_{DEFG} \partial_t f + u_{i,j}\partial_x f + v_{i,j+1}\partial_y f + (\partial_x u)_{i,j} f + (\partial_y v)_{i,j} f\, dx\, dy\, dt = 0.$$

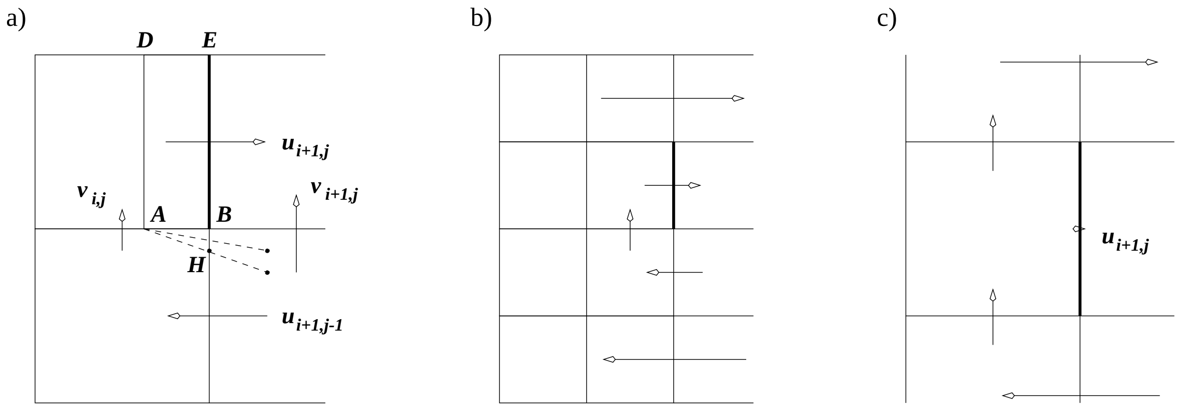


Abbildung 17: Probleme der Positionsbestimmung des Punktes H bei bestimmten Konstellationen der Vorzeichen des Geschwindigkeitsfeldes

Ebenfalls mit dem Gaußschen Integralsatz und mit der Annahme, daß v entlang der oberen Kante konstant ist, erhalten wir

$$\iint\limits_{DEF} vf\,dx\,dt = \iint\limits_{DEF} v_{i,j+1} f\,dx\,dt = \iint\limits_{DEG} f\,dx\,dy - \iiint\limits_{DEFG} \left((\partial_x u)_{i,j} + (\partial_y v)_{i,j+1} \right) f\,dx\,dy\,dt.$$

Die Berechnung des Integrals über das Dreieck DEG erfolgt wie beim CTU-Algorithmus. Für das Volumenintegral über das Tetraeder $DEFG$ rufen wir uns ins Gedächtnis, daß wir nur inkompressible Strömungen betrachten: Da wir in Gleichung (3.12) auch für die Diskretisierung Divergenzfreiheit gefordert haben, ist der Integrand und damit das Integral null. Pilliod übersieht das in seiner Arbeit [24] und approximiert auch noch dieses Integral.

Analog verfahren wir mit dem Integral über ABC. Wir linearisieren allerdings in der unteren Nachbarzelle und integrieren über das Tetraeder $ABCH$. Wir erhalten

$$\iint\limits_{ABC} vf\,dx\,dt = \iint\limits_{ABC} v_{i,j}\,dx\,dt = -\iint\limits_{ABH} f\,dx\,dy + \iiint\limits_{ABCH} \left((\partial_x u)_{i,j+1} + (\partial_y v)_{i,j} \right) f\,dx\,dy\,dt.$$

Für das Integral über das Dreieck ABH muß man natürlich die Front in der unteren Nachbarzelle rekonstruieren.

So weit so gut, wir haben also für jedes Integral in (3.23) eine passende Approximation gefunden. Für andere Vorzeichen von u und v muß man entsprechend die Vorzeichen der Integralausdrücke und die Indizes der Geschwindigkeiten und deren Ableitungen anpassen. Hat man alle Flüsse berechnet, benutzt man (3.20) für die Aktualisierung der Füllungen.
Schwierigkeiten treten bei gewissen Vorzeichenverteilungen der Geschwindigkeiten in der Nachbarschaft einer Zelle auf. In Abbildung 17a soll der Fluß durch die fett gezeichnete Kante berechnet werden. Wir können die Lage des Punktes H nicht mehr eindeutig bestimmen: Seine x-Komponente berechnet sich nach unserer Konstruktion als $H_x = B_x - \Delta t u_{i+1,j-1}$. Damit läge er in der rechten unteren Zelle. Für die y-Komponente müßten wir uns jetzt zwischen $H_y = B_y - \Delta t v_{i,j}$ und $H_y = B_y - \Delta t v_{i+1,j}$ entscheiden. Wenn $v_{i,j}$ und $v_{i+1,j}$ auch noch unterschiedliche Vorzeichen besitzen, wissen wir nicht einmal mehr, in welcher Zelle H liegt. Pilliod

löst das Dilemma, indem er den Punkt H bestimmt als

$$H = C - \Delta t \begin{pmatrix} \min(u_{i+1,j-1}, 0) \\ v_{i,j} \\ 1 \end{pmatrix}.$$

Damit ist sichergestellt, daß der Punkt H immer in derselben Zellenspalte liegt wie die betrachtete Zelle.

Das Problem ist, daß er hierdurch einen Fehler im Fluß durch die betrachtete Kante produziert, wenn die Front in den Zellen $Z_{i,j-1}$ oder $Z_{i+1,j-1}$ verläuft. Infolgedessen kommt es dann zu Verletzungen der Zellfüllungsbeschränkung. Siehe hierzu auch den Abschnitt 3.4.1.2.

Eine Verminderung dieser Fehler wäre durch eine Gitterverfeinerung an solchen Stellen zu erreichen. In Abbildung 17b ist eine solche skizziert. Durch die Verfeinerung wird der Fehler, den man durch Pilliods Trick macht, verkleinert.

Weiß man vorher, wo das Geschwindigkeitsfeld das Vorzeichen wechselt, sollte man das Gitter wenn möglich so wählen, wie in Abbildung 17c angedeutet ist. Da die Geschwindigkeit $u_{i+1,j}$ null ist, fließt gar nichts durch die fett gezeichnete Kante, und die Berechnung der Lage von H entfällt. Tatsächlich wird durch die Wahl eines solchen Gitters die Genauigkeit der Ergebnisse meßbar verbessert. Siehe dazu den Abschnitt 3.5.8.1.

3.5 Testrechnungen

Es werden im folgenden die Ergebnisse aus Testrechnungen mit verschiedenen Rekonstruktions- und Advektionsalgorithmen sowie mit verschiedenen Gitterdimensionen vorgestellt. In der Analyse werden nur die auffälligsten Resultate betrachtet. Eine komplette Liste der Ergebnisse findet man im Anhang.

Die Geschwindigkeitsfelder werden am einfachsten durch eine Stromfunktion $\Psi(x, y)$ beschrieben. Die Komponenten des Geschwindigkeitsfeldes berechnet man dann als

$$u(x, y) = \partial_y \Psi(x, y) \quad \text{und} \quad v(x, y) = -\partial_x \Psi(x, y).$$

Aus dieser Definition folgt sofort, daß das erzeugte Geschwindigkeitsfeld divergenzfrei ist, wenn Ψ zweimal stetig partiell differenzierbar ist, denn es gilt dann

$$\partial_x u + \partial_y v = \partial_x \partial_y \Psi - \partial_y \partial_x \Psi = 0.$$

An den Rändern werden die Flüsse immer auf null gesetzt, was zur Folge hat, daß Fluidbereiche direkt am Rand Kompressionseffekten ausgesetzt sind, weil die Randzellen nicht mehr divergenzfrei sind. Alles in allem wurde darauf geachtet, daß das Fluid den Rand in der vorgegebenen Zeitspanne nicht erreichen kann. In einigen Fällen, insbesondere beim Abriß kleiner Fluidtröpfchen konnte das jedoch nicht immer verhindert werden. Daher zeigen die Bilder manchmal merkwürdige Effekte an den Rändern.

Die gemessenen Größen waren

E_1-Fehler: Nur berechnet, falls man die analytische Lösung des Problems kennt: Dann ist es die Summe der Beträge der Differenzen aus den Füllungen der Rechnung und denen der analytischen Lösung. Dieser Wert wird noch durch das Gesamtvolumen geteilt, um einen relativen Fehler zu erhalten.

E_1-Fehlerkontraktion: Der Quotient des E_1-Fehlers auf einem Gitter und dem auf dem nächst gröberen Gitter. Ein heuristisches Maß für die Konvergenzordnung eines Verfahrens.

Gesamter relativer Volumenverlust: Das Endvolumen des gesamten Fluids minus dem Anfangsvolumen, geteilt durch das Anfangsvolumen. Ebenfalls eine relative Fehlergröße, die allerdings nur den Anfangs- und Endzustand berücksichtigt.

Durchschnittlicher relativer Volumenverlust: Das Mittel des Volumenverlustes in jedem Zeitschritt. Dadurch, daß alle Zeitschritte berücksichtigt werden, ein genaueres Maß für die Volumenerhaltungseigenschaft eines Verfahrens.

Füllungstoleranz: Der endgültige Wert von $\epsilon_{cleanup}$, wenn die Füllungstoleranz adaptiv eingestellt wurde (siehe Abschnitt 3.4.1.2). Ein heuristisches Maß für die Diffusivität eines Verfahrens. Ist er groß, so muß die Front stärker mittels Bereinigung der Füllungsfehler geschärft werden.

Getestet wurde auf einem Pentium PC mit dem Betriebssystem Linux sowie unter Solaris auf SUN-Workstations. Kompiliert wurde mit dem GNU-C-Compiler gcc und der libc C-Bibliothek. Die Maschinengenauigkeit auf dem PC ist etwas besser als auf den SUN-Workstations. Der Grund dafür ist wohl, daß die Fließkommaeinheit im Prozessor intern mit mehr Stellen rechnet und mehrere Schritte intern durchgeführt werden, bevor das Ergebnis auf die erwünschte double-Genauigkeit umgerechnet wird. Auf dem PC erhält man als Maschinengenauigkeit

$$\epsilon_{machine} = 5.4210 \cdot 10^{-20} \quad \text{bzw.} \quad \sqrt{\epsilon_{machine}} = 2.3283 \cdot 10^{-10},$$

während auf den SUN-Workstations

$$\epsilon_{machine} = 1.1102 \cdot 10^{-16} \quad \text{bzw.} \quad \sqrt{\epsilon_{machine}} = 1.0537 \cdot 10^{-08}$$

errechnet wird.

Es wurden Rechnungen auf Gittern mit 32, 64, 128 und 256 Zellen in jeder Koordinatenrichtung durchgeführt. In der niedrigsten Auflösung ist allerdings bei einigen Problemen nicht viel zu erwarten, weil die Größe der entstehenden Strukturen deutlich unter der Maschenweite liegt. Daher wurde auch bei den letzten beiden Problemen auf niedrige Auflösungen verzichtet. Die Tests zeigten, daß die CFL-Zahl möglichst groß gewählt werden sollte. Für alle Rechnungen wurde sie daher auf 0.99 gesetzt.

Der *donor-acceptor*-Advektionsalgorithmus rekonstruiert die Front selbst implizit treppenförmig (siehe Abschnitt 3.4.2.1). Aus dem Rekonstruktionsschritt wird nur die Information übernommen, ob die Front eher senkrecht oder tangential zur Advektionsrichtung steht. Er wurde daher nicht mit allen Rekonstruktionsmethoden getestet, sondern nur mit dem SLIC-Algorithmus aus Abschnitt 3.3.5, um eine komplett stückweise treppenförmige bzw. konstante Methode zu demonstrieren, mit der Methode von Hirt und Nichols (siehe Abschnitt 3.3.6), weil in deren Arbeit [14] genau diese Kombination beschrieben wird, und mit ELVIRA (siehe Abschnitt 3.3.10), um beispielhaft die Wechselwirkung mit einem Rekonstruktionsalgorithmus zweiter Ordnung zu untersuchen. Des weiteren wurde aufgrund der schlechten Ergebnisse auf die schwierigeren Tests verzichtet.

Es hat sich gezeigt, daß die *donor-acceptor*-Methode die Front von selbst relativ „scharf“ hält. Falls doch Verletzungen der Füllungsbeschränkung auftreten, sind diese, verglichen mit den anderen Advektionsmethoden, sehr groß. Aus diesen Gründen wurde auf eine Adaption der Füllungstoleranz verzichtet und diese stattdessen auf $\epsilon_{cleanup} = \sqrt{\epsilon_{machine}}$ gesetzt.

Alle Tests mit adaptiver Füllungstoleranz starten mit $\epsilon_{cleanup} = \epsilon_{machine}^{3/4}$ und passen diese bis zu einem Maximalwert von $\sqrt{\epsilon_{machine}}$ an.

In den erzeugten Bildern wurden Zellen verschiedenen Typs mit verschiedenen Farben gekennzeichnet: Das Fluid in den Frontzellen ist rot, ansonsten sind diese weiß. Volle Zellen innerhalb des Frontbandes (siehe Abschnitt 3.4.1.1) sind blau, leere gelb. Volle Zellen außerhalb des Frontbands sind grün, leere bleiben weiß. Ich hoffe, man kann diese auch in der Schwarz-Weiß-Version gut unterscheiden.

Für die Analyse werden die folgenden Abkürzungen benutzt: SLIC für den Rekonstruktionsalgorithmus aus Abschnitt 3.3.5, H&N für den von Hirt und Nichols (siehe Abschnitt 3.3.6), COM für den *center-of-mass*-Algorithmus (siehe Abschnitt 3.3.7), P&Y für Parker und Youngs' Methode (siehe Abschnitt 3.3.8), DONOR für die *donor-acceptor*-Advektionsmethode (siehe Abschnitt 3.4.2.1), SPLIT für die *fractional-step*-Methode (siehe Abschnitt 3.4.2.2) und UNSPLIT für den Advektionsalgorithmus von Pilliod (siehe Abschnitt 3.4.3).

3.5.1 Rechtecktransport

Das Rechengebiet ist das Quadrat $[-2.5, 2.5] \times [-2.5, 2.5]$. Ein Rechteck mit den Abmessungen 2×1.5 wird im Punkt $(-1, -1)$ zentriert und in einem konstanten linearen Geschwindigkeitsfeld mit Stromfunktion

$$\Psi(x, y) = x - y$$

transportiert. Nach zwei Zeiteinheiten müßte sich der Mittelpunkt des Rechtecks am Punkt $(1, 1)$ befinden. Es wird die einfache Füllungsbereinigung aus Gleichung (3.15) verwendet. Die Ergebnisse sind im Anhang auf den Seiten 61ff. zu finden.

Das Beispiel ist denkbar simpel: Bis auf eine leichte Abrundung der Ecken sollte kein Algorithmus damit ernsthafte Schwierigkeiten haben. Dem ist generell auch so: Alle Advektionsalgorithmen liefern gute Werte für die E_1-Fehler, die relativ eng beieinander liegen. Diese sind bei DONOR aufgrund der impliziten Rekonstruktion und der sehr einfachen Geometrie des Rechtecks überall gleich. Die besten Werte bei 256×256 Zellen liefert COM-SPLIT. Der hervorragende Wert von SLIC-UNSPLIT bei der 32×32 Auflösung ist erst nach zwei Gitterverfeinerungen deutlich verbesserbar.

Ein Ausreißer ist bei SLIC-SPLIT mit 128×128 Zellen zu beobachten: Etwa zum Zeitpunkt $t = 1.29$ tritt eine negative Füllung von etwa -0.5 auf. Nach Bereinigung spaltet sich am rechten unteren Rand ein Tröpfchen ab, das dem Rechteck von nun an vorauseilt. Dieser Effekt setzt noch früher ein, wenn man die CFL-Zahl verkleinert. Für $CFL = 1$ verschwindet dieser Fehler, aber die Rechnung wird durch Rundungsfehler schnell instabil.

Die Volumenveränderungen sind ansonsten, wie zu erwarten war, verschwindend gering, was genau der zentrale Vorteil der VOF-Methoden gegenüber anderen Frontenverfolgungsalgorithmen ist. Beide Volumenänderungswerte sind bei LVIRA immer etwas schlechter als bei den übrigen Frontrekonstruktionen. Grund dafür ist die von LVIRA geforderte größere Füllungstoleranz. In diesem Sinne ist LVIRA etwas dissipativer, weil die Front stärker „nachgeschärft“ werden muß als bei den anderen Rekonstruktionsverfahren.

Wegen der speziellen Geometrie des an den Koordinatenachsen ausgerichteten Rechtecks liefert SLIC hervorragende Werte. Der Vorteil der Rekonstruktionsmethoden zweiter Ordnung kommt aus demselben Grunde nicht so deutlich zum Vorschein. Entscheidend ist bei diesem Testbeispiel, insbesondere bei den gröberen Auflösungen, vielmehr die Behandlung der Ecken des Rechtecks, die auch von SLIC sehr gut wiedergegeben werden.

3.5.2 Kreistransport

In demselben Rechengebiet und Geschwindigkeitsfeld wie beim Rechtecktransportproblem wird statt des Rechtecks ein ebenfalls in $(-1,-1)$ zentrierter Kreis mit Radius eins bewegt, so daß sich sein Mittelpunkt nach zwei Zeiteinheiten an der Stelle $(1,1)$ befinden müßte. Füllungsfehler werden einfach bereinigt. Die Resultate sind im Anhang auf den Seiten 69ff. abgebildet.

Für DONOR sind die Bilder aussagekräftiger als die Tabellen: Bei den niedrigen Auflösungen ist der Kreis schwer als solcher zu erkennen, je höher man geht, um so stärker wandert die Front in das innere des Kreises (dunkle Flächen). Dieser Effekt ist in Kombination mit ELVIRA besonders ausgeprägt. Außerdem tritt *flotsam* auf. Bei der feinsten Auflösung steigt auch die Volumenveränderung auffällig an.

SLIC-SPLIT und SLIC-UNSPLIT erzeugen eine Figur, die irgendwo zwischen einem Quadrat und einem Kreis liegt: je gröber die Auflösung desto quadratischer. Der Grund hierfür ist natürlich die armselige Frontnormalenapproximation von SLIC. Mit COM-SPLIT und COM-UNSPLIT zeigt der Kreis ähnliche Ausbeulungstendenzen. Bei SLIC-SPLIT bekommt der Kreis bei einer Auflösung von 256×256 Zellen Luftblasen.

Die übrigen Bilder unterscheiden sich optisch kaum.

SPLIT und UNSPLIT teilen auch die Rekonstruktionsalgorithmen deutlich in zwei Klassen: Bei H&N, LVIRA und ELVIRA sind die Fehlerkontraktionen deutlich besser als bei den übrigen Rekonstruktionsalgorithmen. Das rechtfertigt zum einen den Ordnungsbegriff der Frontrekonstruktionsverfahren. Es deutet sich aber auch an, daß H&N ähnlich gut rekonstruiert wie LVIRA und ELVIRA, denn die niedrigsten E_1-Fehler treten bei H&N auf, und nur in der höchsten Auflösung sind LVIRA-SPLIT und ELVIRA-SPLIT besser.

3.5.3 Rechteckrotation

Im Quadrat $[-1,1] \times [-1,1]$ wird ein in $(0.5, 0.75)$ zentriertes Rechteck mit den Abmessungen 0.2×0.3 in einem Rotationsfeld mit Stromfunktion

$$\Psi(x,y) = xy$$

gedreht. Nach 2π Zeiteinheiten sollte es sich im Ursprungszustand befinden. Mit diesem Test soll sich herausstellen, wie gut ein Algorithmus lineare Frontstücke verschiedener Steigungen rekonstruieren kann. Auf den Seiten 77ff. sind die Ergebnisse abgebildet.

Wir widmen uns zunächst denen Resultaten von DONOR. Die Bilder zeigen bei jeder Auflösung *flotsam*. Je feiner das Gitter, um so mehr nimmt das Endprodukt die Form des gewünschten Rechtecks an. SLIC-DONOR und H&N-DONOR unterscheiden sich dabei nur unwesentlich, während ELVIRA-DONOR immer diffusiver wird, je höher man auflöst. Bei 256×256 Zellen existieren mit ELVIRA-DONOR beinahe keine vollen Zellen mehr. Dadurch steigt auch die Rechenzeit an. Die Ergebnisse sind alles in allem höchstens mit einem Konturplot brauchbar. Die Volumenänderungen haben deutlich zugenommen.

Nun zu den beiden anderen Advektionsalgorithmen. Mit SLIC-SPLIT sehen die Ergebnisse schlecht aus, da sich besonders bei den niedrigen Auflösungen die Front über das gesamte Gebiet verteilt. Mit SLIC-UNSPLIT ist dieser Effekt etwas weniger stark, doch selbst bei höchster Auflösung kann man nur ahnen, daß das Ergebnis ein Rechteck darstellen soll.

Die Ergebnisse sind bei niedrigen Auflösungen noch sehr rund, was sich bei feineren Gittern allerdings deutlich verbessert. COM liefert immer etwas „verbeultere“ Lösungen als die übrigen

Rekonstruktionsalgorithmen. Das spiegelt sich auch in den E_1-Fehlern wider. H&N, P&Y, LVIRA und ELVIRA unterscheiden sich im E_1-Fehler nur marginal. LVIRA fordert wieder eine etwas höhere Füllungstoleranz, woraus auch etwas größere Volumenänderungen resultieren.

Der zu erwartende Vorteil von ELVIRA und LVIRA, Geraden exakt zu reproduzieren, ist nicht zu sehen. Das liegt daran, daß die entstehenden Geradenstücke zu kurz sind und somit der Einfluß der schwierig zu reproduzierenden Ecken überwiegt.

3.5.4 Kreisrotation

Alles ist wie beim Rechteckrotationsproblem, nur daß statt des Rechtecks ein im Ursprung zentrierter Kreis mit Radius 0.9 gedreht wird. Die Ergebnisse befinden sich auf den Seiten 85ff. im Anhang.

Dies ist das letzte Mal, daß wir die Ergebnisse des DONOR-Algorithmus betrachten wollen. Die Bilder bei niedriger Auflösung zeigen, daß dieser Algorithmus mit dem Problem überfordert ist. Erst bei hoher Auflösung entsteht wieder ein Kreis, allerdings immer umgeben von *flotsam* und mit stark diffundierter Front (besonders mit ELVIRA). Die Tabellen zeigen annähernd dieselben schlechten Werte für alle Rekonstruktionsalgorithmen bei Verwendung mit DONOR.

Es zeigt sich auch wieder die schlechte Zusammenarbeit von SLIC und der UNSPLIT-Advektion: Beim Übergang von 128×128 auf 256×256 Zellen nimmt der E_1-Fehler sogar zu. In den Bildern sieht man deutlich, daß sich das Fluid in winzigen Tröpfchen über das gesamte Rechengebiet verteilt. Mit UNSPLIT sind die Werte leicht besser.

Überraschend sind dagegen die Bilder der restlichen Rekonstruktionsalgorithmen mit SPLIT und UNSPLIT. Schon bei der niedrigsten Auflösung ist der Kreis deutlich erhalten geblieben. Nur COM verformt den Kreis bei niedriger Auflösung ein bißchen stärker als H&N, P&Y, LVIRA und ELVIRA. In der E_1-Fehlerkontraktion bestätigt sich das: Bei H&N, LVIRA und ELVIRA scharen sich die Werte um 0.25, während die von COM über 0.5 liegen und die von P&Y bis etwas über diesen Wert ansteigen. Mit SPLIT sind die Fehler etwas niedriger als mit UNSPLIT. LVIRA zeigt die üblichen erhöhten Werte in Füllungstoleranz und Volumenveränderungen. In der höchsten Auflösung übertrifft ELVIRA den ansonsten leicht besseren H&N-Algorithmus.

3.5.5 Lochkreisrotation

Der Kreis aus dem Kreisrotationsproblem bekommt zusätzlich eine rechteckige Aussparung. Dieses auf Zalesak [32] unter dem Namen *notched circle* zurückgehende Testbeispiel ist ein Standardproblem für Frontenverfolgungsalgorithmen, weil sowohl Ecken als auch lange lineare und gekrümmte Frontstücke auftauchen. Die Ergebnisse dieses Testbeispiels sind auf den Seiten 93ff. aufgeführt.

SLIC wird erwartungsgemäß nicht mit diesem Testbeispiel fertig. Die Ergebnisse von SLIC-SPLIT sind wie üblich noch schlechter als die von SLIC-UNSPLIT. Eine weitere Interpretation der Ergebnisse lohnt sich angesichts dieser Bilder nicht.

Oszillationen in den beiden langen Geradenstücken sind bei allen Algorithmen in der 32×32-Zellen-Auflösung noch vorhanden, begrenzen sich aber immer stärker auf die Ecken, je mehr man verfeinert. Schon bei 64×64 Zellen sieht man lange Geradenstücke.

COM schneidet mit beiden Advektionsalgorithmen etwas schlechter ab als die übrigen Rekonstruktionsmethoden. In den Bildern sieht man, daß die Front hier stärker oszilliert. Das äußert sich auch in den E_1-Fehlern. Die übrigen Rekonstruktionsalgorithmen liefern untereinander sehr ähnliche Ergebnisse, wobei die Fehler in der Reihenfolge P&Y, LVIRA, ELVIRA und H&N leicht abnehmen.

Der UNSPLIT-Algorithmus liefert überall niedrigere Fehlerwerte als SPLIT. Was die Volumenveränderungen angeht, zeigt LVIRA die gewohnte Abweichung von den anderen untereinander beinahe gleichen Algorithmen.

3.5.6 Spirale

Bisher haben wir eher einfache Geschwindigkeitsfelder behandelt. Die Länge der Front blieb zum Beispiel immer gleich, und das Fluid wurde keinen großen Deformationen ausgesetzt. Dies soll sich jetzt ändern. Die Resultate lassen sich auf den Seiten 99ff. begutachten.

Kothe und Rider [29] präsentieren folgenden Testfall: Im Einheitsquadrat $[0,1] \times [0,1]$ wird ein Kreis mit Zentrum $(0.50, 0.75)$ und Radius 0.15 drei Zeiteinheiten lang in einem Geschwindigkeitsfeld mit der Strömungsfunktion

$$\Psi(x,y) = \frac{1}{\pi} \sin^2(\pi x) \sin^2(\pi y)$$

bewegt. Dadurch wird der Kreis gedehnt und in einer Spiralbewegung in die Mitte gezogen. Für dieses Geschwindigkeitsfeld reicht eine einfache Bereinigung der Füllungsfehler nicht aus. Wir verwenden daher die in Abschnitt 3.4.1.2 beschriebene Methode der Umverteilung von Füllungsfehlern auf benachbarte Frontzellen.

Es liegt hierzu leider keine exakte Lösung vor, so daß die Berechnung von E_1-Fehlern nicht möglich ist. Kothe und Rider verglichen ihre Lösungen mit der einer „exakten“ Partikelmethode, das heißt, daß sie anstelle interpolierter Werte die exakten analytischen Geschwindigkeiten für die einzelnen Partikel in der Rechnung benutzten. Um unsere Ergebnisse interpretieren zu können, sollte man wissen, daß die exakte Lösung eine glatte Spirale ist, die nirgendwo abreißt.

Zuerst zum SPLIT-Algorithmus: SLIC liefert immer schlechte Ergebnisse. Die übrigen Rekonstruktionsmethoden unterscheiden sich hauptsächlich dadurch, wie oft die Front reißt beziehungsweise wie stark sie noch zusammenhängt. Die Front oszilliert immer stärker bei feinerer Diskretisierung, und durch diese Oszillationen reißt sie noch öfter. Am besten hält LVIRA die Front zusammen, dann ELVIRA, H&N, P&Y und am schlechtesten COM. Alle zeigen große Veränderungen der Gesamtmasse.

Die Oszillationen entstehen, weil eine auf *operator splitting* basierende Methode hier nicht angemessen ist. Das Geschwindigkeitsfeld ist zu unruhig, als daß man die Advektionsschritte in den Koordinatenrichtungen einzeln und hintereinander ausführen könnte.

Gestützt wird diese Aussage von den Ergebnissen mit UNSPLIT-Advektion: Die Front reißt hier zwar auch, jedoch vermindert sich dieser Effekt bei Gitterverfeinerung. Oszillationen wie bei SPLIT treten nicht auf. Auch hier hält LVIRA die Front am besten zusammen. Bei 256×256 Zellen tritt allerdings etwas *flotsam* auf, was sich aber durch eine geschicktere Wahl der Füllungstoleranz optimieren läßt. Ohne *flotsam* aber dafür auch weniger zusammenhängend sind die Ergebnisse bei ELVIRA, H&N, P&Y und COM, in der Reihenfolge des besten Zusammenhalts.

Das Phänomen des Abreißens einzelner Fluidtröpfchen nennen Kothe und Rider *numerische Oberflächenspannung* (englisch *numerical surface tension*), weil so etwas normalerweise nur unter Einfluß von Oberflächenspannung passiert. Es ist einer der Hauptnachteile der VOF-Methoden. Numerische Oberflächenspannung tritt fast immer auf, wenn ein Fluidbereich dünner wird als eine Zellbreite. Eine Strategie zur Vermeidung dieses Effektes ist adaptive Gitterverfeinerung. Siehe hierzu den Abschnitt 3.5.8.2.

LVIRA hat den anderen Rekonstruktionsmethoden voraus, daß es sich nicht allein auf eine Normalenapproximation aus finiten Füllungsdifferenzen verläßt, sondern iterativ eine optimale Normale sucht. Damit ist LVIRA etwas flexibler als die anderen Methoden. Der Preis dafür ist offenbar ein größerer Fehler im Gesamtvolumen.

3.5.7 Deformation

Frei nach Kothe und Rider [29] wird der Kreis mit Radius 0.15 aus dem Spiralproblem diesmal zentriert im Punkt $(0.5, 0.5)$ in $[0,1] \times [0,1]$ der durch

$$\Psi(x,y) = \frac{1}{4\pi} \sin\left(4\pi\left(x+\frac{1}{2}\right)\right) \cos\left(4\pi\left(y+\frac{1}{4}\right)\right)$$

erzeugten Strömung ausgesetzt[11]. Diese hat vier Knoten, in die das Fluid spiralförmig hineingezogen wird. Der Kreis wird dabei in vier Richtungen gleichzeitig gezerrt. Wir wollen mit diesem Beispiel testen, wie gut die Algorithmen ein Geschwindigkeitsfeld bewältigen, das die Front in verschiedene Richtungen zerrt. Was dabei herauskommt, kann man auf den Seiten 103ff. sehen.

SPLIT und SLIC zeigen dieselben Effekte wie beim Spiralproblem. Wir widmen uns daher direkt den Ergebnissen mit UNSPLIT-Advektion:

Bei 64×64 Zellen bekommt man noch keine Vorstellung, wie die exakte Lösung aussehen könnte. Vergleicht man diese mit den Abbildungen höherer Auflösung, sieht man, daß hier wieder numerische Oberflächenspannung aufgetreten ist. Leider ist diese aber bei 256×256 Zellen immer noch deutlich sichtbar. Erstaunlich ist, daß diesmal H&N die feinste, am wenigsten gerissene Front — man beachte die senkrechte Linie in der Mitte — und den geringsten Volumenfehler bietet. LVIRA sieht nicht mehr so überzeugend aus wie noch beim Spiralproblem, gleiches gilt für ELVIRA. P&Y und COM haben dünne Fluidbereiche längst aufgegeben.

3.5.8 Gitterwahl

3.5.8.1 Gitterverschiebung

Anhand des letzten Beispiels wollen wir noch untersuchen, wie die Wahl des Gitters das Ergebnis beeinflußt. Bei einer Auflösung von 128×128 Zellen verschieben wir das Gitter um $1/256$, also eine halbe Zellenbreite. Entlang einer Geraden $y = 0.5$ ist die Geschwindigkeit in y-Richtung null. Durch die Verschiebung des Gitters liegt diese Gerade nun genau auf einer Gitterlinie. Das führt dazu, daß die Geschwindigkeit in y-Richtung für alle an diese Linie grenzenden Zellen an der entsprechenden Kante verschwindet. Im Abschnitt 3.4.3.2 haben wir bereits skizziert, daß eine solche Gitterwahl für den UNSPLIT-Algorithmus günstig ist.

Tatsächlich verbessert sich die Volumenerhaltung drastisch (der Gesamtverlust fällt zum Beispiel auf $2.55 \cdot 10^{-16}$ und der durchschnittliche Verlust auf $9.47 \cdot 10^{-17}$). Allerdings tritt die numerische Oberflächenspannung nun an anderen Stellen intensiver als vorher auf. In Abbildung 18 sieht man, daß die querliegenden Fluidbereiche oben und unten verschwinden, während die linken und rechten äußeren Randbereiche in der rechten Skizze stärker zusammenhängen.

Es sei an dieser Stelle noch einmal angemerkt, daß die in allen Beispielen auftretenden Gesamtvolumenänderungen (maximal $5 \cdot 10^{-7}$) für die meisten fluidmechanischen Berechnungen noch

[11] Im Originalbeispiel wird die Front über den oberen Rand hinweggezerrt und erscheint aufgrund des periodischen Rechengebiets am unteren Rand wieder. Da in unserem Programm das Rechengebiet nicht-periodisch ist, wurde alles um $\frac{1}{4}$ in y-Richtung verschoben, wodurch kein Rand berührt wird.

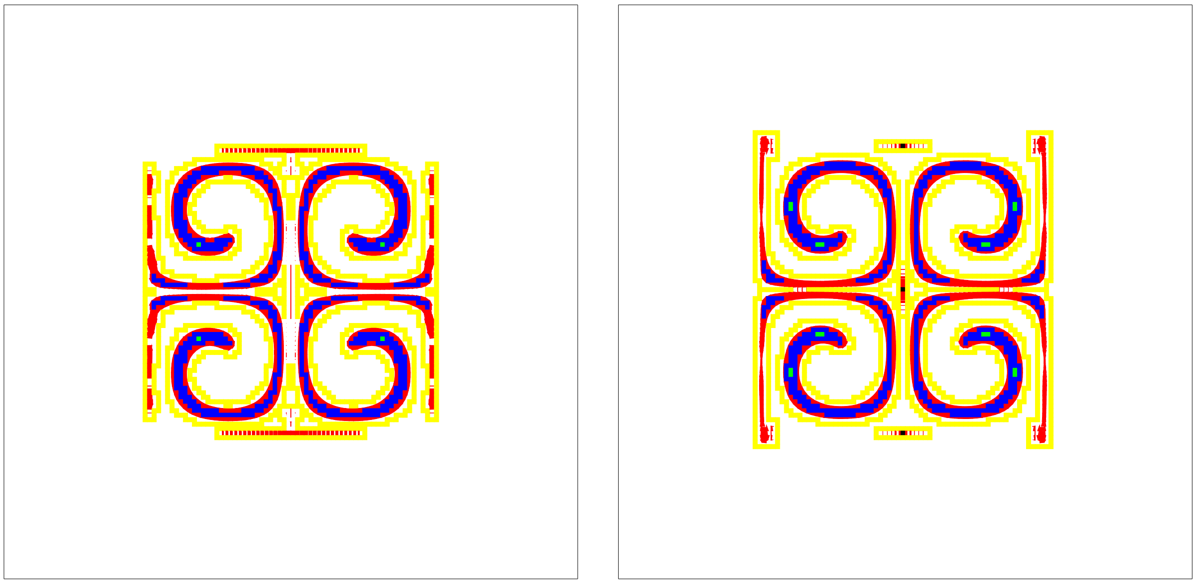


Abbildung 18: Das Deformationsproblem mit UNSPLIT-H&N Algorithmus: Links wurde das normale Gitter verwendet, rechts ein um eine halbe Zellbreite in y-Richtung verschobenes Gitter.

vollständig zufriedenstellend sein dürften, eine abgerissene Front oder Fluidtröpfchen an Stellen, wo sie nicht sein dürften, jedoch weniger.

Es ist daher fraglich, welches Ergebnis nun das bessere ist. Im Normalfall kennt man das Geschwindigkeitsfeld außerdem nicht a priori, so daß man gar nicht erst in die Verlegenheit kommt, hier eine Entscheidung zu fällen.

3.5.8.2 Adaptive Gitterverfeinerung

Anders sieht der Fall aus, wenn man mit adaptiver Gitterverfeinerung arbeitet. Ergebnisse über die Verwendung von *volume-of-fluid*-Methoden zusammen mit adaptiver Gitterverfeinerung finden sich zum Beispiel in [19] und [27]. An Stellen, an denen ein grobes Gitter zu schlechte Ergebnisse liefert, wird es lokal verfeinert. Wie erkennt man, welche Zellen verfeinert werden sollen? Nach Auswertung der Testbeispiele kommen wir zu folgenden Indikatoren für die Gitterverfeinerung:

- Frontzellen, bei denen die Normalenrichtung stark von der Normalenrichtung der benachbarten Frontzellen abweicht. Auf diese Weise werden zum Beispiel Ecken höher aufgelöst.

- Frontzellen, bei denen sich das Vorzeichen der Geschwindigkeit in x- oder y-Richtung ändert. Hier sollte man versuchen, die neue Kante möglichst so zu legen, daß die entsprechende Geschwindigkeitskomponente verschwindet. Dadurch wird der UNSPLIT-Advektionsalgorithmus stabiler.

3.5.9 Rechenzeit

In einer vollständigen Analyse sollte auch ein Rechenzeitvergleich nicht fehlen. Als Testbeispiel wurde das Rechtecktransportproblem aus Abschnitt 3.5.1 gewählt, weil hier alle Algorithmen stabil laufen. Die Dimension des Rechengitters wurde auf 256×256 Zellen gesetzt. Bei der verwendeten CFL-Zahl von 0.99 ergibt das 207 Zeitschritte. Die Zeitangaben in der folgenden Tabelle stammen von Rechnungen auf einem Pentium PC mit 133MHz und wurden komplett im Arbeitsspeicher also ohne Speicherauslagerung auf die Festplatte durchgeführt. Zur Zeitmessung diente das UNIX-Dienstprogramm time.

min : sec	DONOR	SPLIT	UNSPLIT
SLIC	1:20	1:22	1:10
H&N	1:22	1:26	1:11
COM	1:32	1:36	1:16
P&Y	1:23	1:26	1:11
LVIRA	3:56	5:27	3:10
ELVIRA	2:32	2:38	1:47

Es fällt auf, daß DONOR und SPLIT fast überall gleich schnell sind, UNSPLIT aber etwas schneller. Das liegt daran, daß die Front bei einem *operator-split*-Algorithmus pro Zeitschritt zweimal rekonstruiert werden muß, nämlich für jeden der zwei Advektionsschritte einmal. Bei UNSPLIT genügt einmalige Rekonstruktion, was dieser Methode trotz der aufwendigeren Berechnung der Kantenflüsse einen deutlichen Rechenzeitvorteil verschafft.

LVIRA fällt in der Rechenzeittabelle sehr unangenehm auf. Das Hauptproblem liegt darin, daß LVIRA zu viele Parameter hat, von denen die Ergebnisse empfindlich abhängen. Kritisch sind vor allem Abbruchgenauigkeit und die maximale Iterationsanzahl für Brents Minimierungsalgorithmus. Außerdem ist die Konstruktion des Suchintervals für das E_2-Minimum eine heikle Angelegenheit. Die hier verwendeten Parameterwerte sind so gewählt, daß man ohne Modifikation alle beschriebenen Testbeispiele mit annehmbaren Ergebnissen mit sehr wenigen Warnungen durchrechnen kann. Natürlich kann man versuchen, die Parameter dem jeweiligen Problem anzupassen. In Kombination mit einem echten Strömungslöser wird man jedoch kaum sehr viel Zeit auf die Feineinstellung der Frontrekonstruktionsparameter verwenden wollen. Da als Startnormale das Ergebnis des H&N-Algorithmus verwendet wird, wird LVIRA immer langsamer als dieser bleiben.

ELVIRA ist auch etwas langsamer als die Algorithmen, die formal erste Ordnung besitzen. Mit DONOR benötigt man hier etwa doppelt so viel Rechenzeit, mit UNSPLIT immer noch etwa das 1.4-fache.

Die anderen Rekonstruktionsalgorithmen liegen in der Rechenzeit so nah beieinander, daß sich eine Unterscheidung kaum lohnt. COM ist nur ein bißchen langsamer.

Diese Tabelle zeigt nur, wie sich die Rechenzeiten bei stabiler Rechnung verhalten. Ist das nicht der Fall, wie zum Beispiel bei SLIC-SPLIT im Rechteckrotationsproblem oder fast allen anderen Rechnungen mit DONOR, steigt die Rechenzeit drastisch mit der Anzahl der wegen Rechenfehlern hinzukommenden Frontzellen an.

Im übrigen bleibt noch einmal anzumerken, daß die Rechenzeiten für die Frontenverfolgung in einer kompletten Strömungsberechnung nur einen sehr kleinen Bruchteil ausmachen, weil der Frontberechnungsaufwand nur $O(n)$ ist (siehe Abschnitt 3.4.1.1).

3.6 Zusammenfassung der Testergebnisse

Nach Auswertung aller Testläufe kommen wir zu folgenden Ergebnissen über die Rekonstruktionsalgorithmen:

SLIC: Dieser Algorithmus sollte nicht verwendet werden. Die stückweise konstante Rekonstruktion der Front ist zu ungenau und führt leicht zu Abtrennungen von Fluidteilchen oder Lufteinschlüssen. Besonders schlecht arbeitet SLIC mit dem *fractional-step*-Advektionsalgorithmus. Für die Verwendung mit dem *donor-acceptor*-Algorithmus liefert SLIC die benötigte minimale Information über die Lage der Front und wäre dann aufgrund von Rechenzeitüberlegungen allen anderen Rekonstruktionsalgorithmen vorzuziehen. Der getestete SLIC-Algorithmus stellt allerdings nur eine Vereinfachung des Originalalgorithmus' von Noh und Woodward [20] dar. In der Literatur finden sich über diesen aber ähnliche Ergebnisse.

Center of Mass: Ursprünglich ist dieser Algorithmus für dreidimensionale Probleme entworfen worden, wo er sehr schnell relativ gute Ergebnisse produziert. Unübersehbar war allerdings, daß er zu stärkeren Oszillationen in der Frontlinie neigt als die übrigen Algorithmen.

Parker und Youngs: Ein durchweg zufriedenstellender Algorithmus, der mit allen Advektionsalgorithmen gute Ergebnisse liefert. Leider rekonstruiert er etwas ungenauer als der genauso schnelle Algorithmus von Hirt und Nichols, was stets zu etwas größeren E_1-Fehlern der Lösungen führt.

Hirt und Nichols: Ein unerwartet guter Frontenrekonstruktionsalgorithmus. Formal zwar nur von erster Ordnung, erzeugt er bei den niedrigeren Auflösungen geringere E_1- und Volumenfehler als die Algorithmen der zweiten Ordnung. Außerdem ist er sehr schnell. Leider produziert der Algorithmus aber bei dem Spiralproblem zuviel numerische Oberflächenspannung. Die E_1-Fehlerkontraktion ist ähnlich gut wie bei den Verfahren zweiter Ordnung, was uns zu dem Schluß bringt, daß dieser Algorithmus zwar formal erster, aber numerisch zweiter Ordnung ist.

LVIRA: Dieser Algorithmus ist leider zu langsam und beinhaltet zu viele empfindliche Parameter. Meistens liefert er schlechtere E_1-Fehler als der Algorithmus von Hirt und Nichols, dessen Normalenapproximation als Startnäherung benutzt wird. LVIRA tendiert in der getesteten Form etwas mehr zu Gesamtvolumenänderungen als die übrigen Verfahren. Beim Spiralproblem wurde die Front allerdings am besten zusammengehalten. Wer diesen Algorithmus benutzen will, sollte nach günstigeren, an das jeweilige Problem angepassten Parameterwerten suchen. LVIRA ist allerdings als einzige Methode ohne zusätzliche Modifikationen auf unstrukturierten Gittern anwendbar.

ELVIRA: Was Pilliod in [24] von ELVIRA verspricht, ist leider nicht ganz eingetroffen. Außer bei der höchsten Auflösung steht ELVIRA immer hinter dem Algorithmus von Hirt und Nichols zurück. Die formale zweite Ordnung äußert sich zwar in guten E_1-Fehlerkontraktionsraten, aber auch in einer manchmal beinahe doppelt so langen Rechenzeit. Außerdem ist der Umgang mit mehreren den lokalen E_2-Fehler minimierenden Normalenkandidaten sehr willkürlich. Trotzdem ist, wer genügend Rechnerkapazitäten zur Verfügung hat und lieber einen Algorithmus zweiter Ordnung benutzt, mit ELVIRA gut beraten.

Der von Pilliod propagierte Ordnungsbegriff, der einem Verfahren zweite Ordnung zuschreibt, wenn es in der Lage ist, lineare Fronten exakt zu rekonstruieren, hat sich nur bedingt als Garantie für niedrige E_1-Fehler herausgestellt. Die Verfahren von Hirt und Nichols oder von Parker und Youngs lieferten zumindest bei den getesteten Beispielen ähnlich gute Ergebnisse wie ELVIRA und LVIRA, für die man außerdem noch deutlich längere Rechenzeiten in Kauf nehmen muß. Außerdem produzierte LVIRA in einigen Beispielen *flotsam* und ELVIRA zeigte in Verwendung mit DONOR stärkere Dispersivität als die anderen Rekonstruktionsmethoden. Beide Eigenschaften sind bei Frontrekonstruktionsmethoden unerwünscht.

Eine Empfehlung würde ich für den Hirt-Nichols-Algorithmus oder ELVIRA aussprechen.

Allgemein muß aber gesagt werden, daß die Qualität der Testergebnisse viel stärker von der verwendeten Advektionsmethode abhängt als vom Rekonstruktionsalgorithmus. Daher jetzt zu diesen:

donor-acceptor-Methode: Die Ergebnisse sind sehr schlecht. Der Algorithmus bewältigt nur die einfachen Testbeispiele und selbst diese nur mit Schwierigkeiten. Die Front wird zwar in Verbindung mit SLIC oder dem Hirt-Nichols-Algorithmus von selbst ohne Bereinigungsschritte scharf gehalten, doch produziert die *donor-acceptor*-Methode schnell *flotsam*-Tröpfchen und läßt sich durch Ecken in der Front leicht aus dem Konzept bringen. Trotz der weiten Verbreitung sollte von diesem Algorithmus Abstand genommen werden.

fractional-step-Methode: Sehr schöne Leistung bei den einfachen Geschwindigkeitsfeldern täuschen nicht darüber hinweg, daß die Front bei schwierigeren Strömungen bis hin zum Aufreißen oszilliert, und das um so stärker, je mehr man das Gitter verfeinert. In diesen Beispielen ist die *fractional-step*-Methode also instabil. Einen solchen Algorithmus wird man auch durch adaptive Gitterverfeinerung nur verschlechtern. Die Methode ist außerdem als *operator-split*-Methode wegen der zweifachen Frontrekonstruktion pro Zeitschritt langsam. Wer nicht genau weiß, daß die *fractional-step*-Methode das Geschwindigkeitsfeld bewältigen kann, sollte sie meiden, um von eventuellen Überraschungen verschont zu bleiben.

Pilliods Methode: Diese beste hier behandelte Advektionsmethode ist wegen des *unsplit*-Ansatzes schnell und bewältigt als einzige alle Testbeispiele zufriedenstellend. Die Programmierung ist allerdings etwas komplizierter als die der *operator-split*-Algorithmen. Größere Volumenfehler sind zu erwarten, wenn eine Geschwindigkeitskomponente in einer Zelle das Vorzeichen wechselt und dazu noch die Differenz der Beträge der Geschwindigkeit an den gegenüberliegenden Kanten zu groß wird. Gitteradaption ist dann angeraten.

Man sollte auf jeden Fall bei komplizierteren Geschwindigkeitsfeldern, wie zum Beispiel Strömungen durch poröse Medien, von der Verwendung von *operator-split*-Methoden vollständig absehen.

Weiß man, daß das Fluid in der Berechnung dünne Stellen aufweisen wird, ist die Verwendung adaptiver Gitterverfeinerung angeraten, damit die von der VOF-Methode erzeugte numerische Oberflächenspannung gering gehalten wird.

4. Schlußbemerkungen

4.1 Zusammenfassung

Es wurden einige Methoden zur Frontrekonstruktion mit ihren speziellen Vor- und Nachteilen dargestellt. Der Schwerpunkt lag dabei bei Anwendungen auf zweidimensionale inkompressible Strömungen in der Fluidmechanik. Im Speziellen wurde die *volume-of-fluid*-Methode auf kartesischen Gittern untersucht. Nach einer Aufgliederung in Rekonstruktions- und Advektionsalgorithmen wurden viele Varianten der Teilschritte dargestellt und in numerischen Experimenten miteinander verglichen.

Es wurde gezeigt, daß die Qualität der Ergebnisse stärker von der verwendeten Advektionsmethode als von der Rekonstruktionsmethode abhängt. Des weiteren wurde die Überlegenheit stückweise linear rekonstruierender Methoden gegenüber den stückweise konstant bzw. treppenförmig rekonstruierenden Algorithmen gezeigt.

Der von Pilliod eingeführte Ordnungsbegriff für Frontrekonstruktionsverfahren hat sich nicht unbedingt als angemessen erwiesen, da der Rekonstruktionsalgorithmus von Hirt und Nichols nach dieser Klassifizierung nur von erster Ordnung ist und dennoch oft bessere Ergebnisse liefert als die Rekonstruktionsmethoden der formalen zweiten Ordnung.

Die moderneren Rekonstruktionsalgorithmen stellten in vielen Fällen nur einen Mehraufwand an Rechenzeit dar, wenngleich der LVIRA-Algorithmus beim Spiralproblem die geringste numerische Oberflächenspannung aufwies.

Unter den Advektionsverfahren dominieren die *fractional-step*-Methode und der Algorithmus von Pilliod. Der *donor-acceptor*-Algorithmus gehört trotz immer noch weiter Verbreitung in die Zeit der Lochkarten und Mainframes und sollte nicht mehr benutzt werden. Für kompliziertere Fälle sollte auf jeden Fall der Algorithmus von Pilliod gewählt werden, da er nicht die üblichen Probleme der auf *operator splitting* beruhenden Algorithmen aufweist.

Es wurde außerdem gezeigt, daß VOF-Algorithmen bei geschickter Gitterwahl und korrekt durchgeführter Füllungsfehlerbereinigung das Gesamtvolumen erhalten. Anhand von Beispielen, in denen das Fluid in sehr feine Bereiche gedehnt wurde, wurde das Phänomen der numerischen Oberflächenspannung illustriert, das ein typisches bei VOF-Verfahren auftretendes Phänomen ist.

4.2 Danksagung

An dieser Stelle möchte ich noch einigen Personen im besonderen danken, deren Unterstützung eine große Erleichterung bei der Erstellung dieser Arbeit war:

Mein Dank gilt zunächst meiner Freundin Elisabeth, die viel von dem Streß auffangen mußte, den ich während der Programmierung der unterschiedlichen Verfahren und beim Verfassen des

Textes zu bewältigen hatte, und die außerdem den Text Korrektur gelesen hat.

Danke auch an Thomas Sonar für die Motivation und Unterstützung beim Verfassen der Arbeit.

Ebenfalls möchte ich mich bei Frank Schimmel und Michael Breuss für das fachkundige Korrekturlesen des Textes und für Tips zum Umgang mit LaTeX bedanken.

Zu guter Letzt noch ein Dankeschön an meine Mutter, die den Text noch einmal abschließend auf Rechtschreibfehler durchsucht hat.

Literaturverzeichnis

[1] *convert*. http://www.wizards.dupont.com/cristy/ImageMagick.html. Version 3.7.3.

[2] *mpeg*. ftp://havefun.stanford.edu/pub/mpeg. Version 1.2.

[3] *ppm2cyuv*. ftp://havefun.stanford.edu/pub/cv.

[4] *transfig*. ftp://ftp.x.org/contrib/applications/drawing_tools/transfig. Release 3.2.

[5] *xfig*. ftp://ftp.x.org/contrib/applications/drawing_tools/xfig. Release 3.2.

[6] F. L. Addessio, D. E. Carroll, J. K. Dukowicz, F. H. Harlow, J. N. Johnson, B. A. Kashiwa, M. E. Maltrud und H. M. Ruppel, *CAVEAT: A Computer Code for Fluid Dynamics Problems with Large Distortion and Internal Slip*, tech. rep., Los Alamos National Laboratory, http://lib-www.lanl.gov/la-pubs/00175068.pdf, 1992.

[7] N. Ashgriz und J. Y. Poo, *FLAIR: Flux Line-Segment Model for Advection and Interface Reconstruction*, Journal of Computational Physics, 93 (1991), pp. 449–468.

[8] P. K. Barr und W. T. Ashurst, *An Interface Scheme for Turbulent Flame Propagation*, tech. rep., Sandia National Laboratories, 1984.

[9] J. B. Bell, C. N. Dawson und G. R. Shubin, *An Unsplit, High Order Godunov Method for Scalar Conservation Laws in Multiple Dimensions*, Journal of Computational Physics, 74 (1988), pp. 1–24.

[10] A. J. Chorin, *Flame Advection and Propagation Algorithms*, Journal of Computational Physics, 35 (1980), pp. 1–9.

[11] P. Colella, *Multidimensional Upwind Methods for Hyperbolic Conservation Laws*, Journal of Computational Physics, 87 (1990), pp. 171–200.

[12] P. Colella und P. Woodward, *The Piecewise Parabolic Method (PPM) for Gas-Dynamical Simulations*, Journal of Computational Physics, 54 (1984), pp. 174–201.

[13] R. DeBar, *Fundamentals of the KRANKEN Code*, tech. rep., Lawrence Livermore National Laboratory, 1974.

[14] C. W. Hirt und B. D. Nichols, *Volume-of-Fluid (VOF) Method for the Dynamics of Free Boundaries*, Journal of Computational Physics, 39 (1981), pp. 201–225.

[15] D. Kothe, C. Zemach und T. Chaumeazeau, *Three-Dimensional Volume Tracking of Fluid Interfaces on General Hexahedral Meshes*, in Vorbereitung.

[16] D. B. Kothe, J. R. Baumgardner, S. T. Bennion, J. H. Cerutti, B. J. Daly, K. S. Holian, E. M. Kober, S. J. Mosso, J. W. Painter, R. D. Smith und M. D. Torrey, *PAGOSA: A Massively-Parallel, Multi-Material Hydro-Dynamics Model for Three-Dimensional High-Speed Flow and High-Rate Deformation*, Tech. Rep. LA-UR-92-4306, Los Alamos National Laboratory, http://lib-www.lanl.gov/la-pubs/00374823.pdf, 1992.

[17] D. B. Kothe und W. B. Rider, *Comments on Modeling Interfacial Flows with Volume-of-Fluid Methods*, tech. rep., Los Alamos National Laboratory, 1995.

[18] D. B. Kothe und W. J. Rider, *Practical Considerations for Interface Tracking Methods*, tech. rep., Los Alamos National Laboratory, http://www-xdiv.lanl.gov/XHM/personnel/wjr/Web_papers/pubs.html, 1995.

[19] G. H. Miller und E. G. Puckett, *A High-Order Godunov Method for Multiple Condensed Phases*, eingereicht beim Journal of Computational Physics, May 1996.

[20] W. Noh und P. Woodward, *Lecture Notes in Physics*, Springer Verlag, 1976, ch. SLIC (Simple Line Interface Method), pp. 330–340.

[21] S. Osher und J. A. Sethian, *Fronts Propagating with Curvature-Dependent Speed: Algorithms Based on Hamilton-Jacobi Formulation*, Journal of Computational Physics, 79 (1988), pp. 12–49.

[22] B. J. Parker und D. L. Youngs, *Two and Three Dimensional Eulerian Simulation of Fluid Flow with Material Interfaces*, Tech. Rep. 01/92, UK Atomic Weapons Establishment, Aldermaston, Berkshire, Feb 1992.

[23] J. E. Pilliod, *A Second-Order Unsplit Method for Modeling Flames in Two-Dimensional Compressible Flows*, PhD thesis, 1996.

[24] J. E. Pilliod und E. G. Puckett, *Second-Order Accurate Volume-of-Fluid Algorithms for Tracking Material Interfaces*, eingereicht beim Journal of Computational Physics.

[25] W. Press, B. Flannery, S. Teukolsky und W. Vetterling, *Numerical Recipes in C*, Cambridge University Press, 1988.

[26] E. G. Puckett, *A Volume-of-Fluid Interface Tracking Algorithm with Applications to Computing Shock Wave Refraction*, in Proceedings of the 4th International Symposium on Computational Fluid Dynamics, 1991, pp. 933–938.

[27] E. G. Puckett und J. S. Saltzman, *A 3D Adaptive Mesh Refinement Algorithm for Multimaterial Gas Dynamics*, Physica D, 60 (1992), pp. 84–93.

[28] W. J. Rider und D. B. Kothe, *A Marker Particle Method for Interface Tracking*, tech. rep., Los Alamos National Laboratories, http://www-xdiv.lanl.gov/XHM/personnel/wjr/Web_papers/pubs.html, May 1995.

[29] W. J. Rider und D. B. Kothe, *Stretching and Tearing Interface Tracking Methods*, Tech. Rep. AIAA-95-1717, Los Alamos National Laboratory, http://www-xdiv.lanl.gov/XHM/personnel/wjr/Web_papers/pubs.html, 1995.

[30] J. A. Sethian, *Level Set Methods*, Cambridge Monographs on Applied and Computational Mathematics, Cambridge University Press, 1996.

[31] D. L. Youngs, *Time-Dependent Multi-Material Flow with Large Fluid Distortion*, K. W. Morton and M. J. Baines, 1982, pp. 273–285.

[32] S. Zalesak, *Fully Multidimensional Flux-Corrected Transport Algorithm for Fluids*, Journal of Computational Physics, 31 (1979), pp. 235–263.

Anhang

A.1 Implementierung

Zur numerischen Auswertung wurden die meisten beschriebenen Algorithmen im Programm vof implementiert. Das vof ist in der Programmiersprache ANSI-C geschrieben, um Portabilität zu gewährleisten. Es gliedert sich in folgende Module:

advection.c: Der *fractional-step*-Advektionsalgorithmus (siehe Abschnitt 3.4.2.2)

advectionfields.c: Einige Beispiele für Geschwindigkeitsfelder

centerofmass.c: Der *center-of-mass*-Rekonstruktionsalgorithmus (siehe Abschnitt 3.3.7)

cfl.c: Berechnung der Zeitschrittweite aus der CFL-Zahl und der Zellenbreite (siehe Abschnitt 3.4.1.1)

cleanup.c: Bereinigung von Füllungsfehlern (siehe Abschnitt 3.4.1.2)

donoracceptor.c: Der *donor-acceptor*-Advektionsalgorithmus (siehe Abschnitt 3.4.2.1)

elvira.c: Der *ELVIRA*-Rekonstruktionsalgorithmus (siehe Abschnitt 3.3.10)

frontutil.c: Funktionen zur Verwaltung der Frontsegmente

geometry.c: Funktionen für die geometrischen Berechnungen wie zum Beispiel Flächeninhalte, Schnittpunkte von Geraden und dem Abstandsparameter einer Frontgeraden (siehe Abschnitt 3.3.2)

hirtnichols.c: Der Rekonstruktionsalgorithmus von Hirt und Nichols (siehe Abschnitt 3.3.6)

initialfronts.c: Beispiele für Ausgangsfronten: Rechteck, Kreis und gelochter Kreis

initialization.c: Initialisierungsroutinen

io_functions.c: Einfache Ein- und Ausgabefunktionen in ASCII-files

linear_algebra.c: Funktionen für (zweidimensionale) Vektorrechnungen

lowlevel.c: Minimum-, Maximum- und Quadratfunktion

lvira.c: Der *LVIRA*-Rekonstruktionsalgorithmus (siehe Abschnitt 3.3.9)

machine.c: Bestimmung der Maschinengenauigkeit $\epsilon_{machine}$ und deren Wurzel $\sqrt{\epsilon_{machine}}$

model.c: Hier wird festgelegt, mit welchem Algorithmus und welchen Parametern welche Aufgabe gelöst werden soll.

parkerandyoungs.c: Der Rekonstruktionsalgorithmus von Parker und Youngs (siehe Abschnitt 3.3.8)

slic.c: Ein *SLIC*-Rekonstruktionsalgorithmus (siehe Abschnitt 3.3.4)

types.h: Dimensionsdefinition, Typdefinitionen und Deklaration einiger Macros und globaler Variablen

unsplit_advection.c: Der Advektionsalgorithmus von Bell, Dawson und Shubin (siehe Abschnitt 3.4.3.2)

vof.c: Hauptprogramm und einige Steuerungsfunktionen

volumeerror.c: E_1- und E_2-Fehlerberechnungen für Nachbarzellen und global

xfig_output.c: Funktionen zur Ausgabe der Ergebnisse im xfig-Format

Außerdem befinden sich im Unterverzeichnis models die Modelldateien für einige Testbeispiele. Um sie zu verwenden, muß man die Datei model.c durch die gewünschte Datei ersetzen und neu kompilieren.

Die graphische Ausgabe erfolgt im xfig-Format, das mit dem Programm xfig [5] angesehen und nachbearbeitet und mit dem transfig-Paket [4] in diverse Standardgraphikformate konvertiert werden kann. Für die Herstellung von mpeg-Filmen aus Einzelbildern wurde das Programm mpeg [2] benötigt. Die Programme convert [1] oder ppm2cyuv [3] lieferten die hierzu notwendigen Bilder im YUV-Format, wobei ppm2cyuv leider kein *antialiasing* unterstützt, dafür aber um ein Vielfaches schneller arbeitet. Zur Steuerung der Filmenherstellung wurden die shell-Skripte makemovie.convert und makemovie.ppm2cyuv entworfen. Alle Programme sind freie Software.

A.2 Testproblem Rechtecktransport

A.2.1 Rechtecktransport, *donor-acceptor*-Advektion

E_1-Fehler

Dim.	SLIC	H&N	ELVIRA
32	1.2473e-01	1.2901e-01	1.2927e-01
64	5.8512e-02	5.9640e-02	5.9990e-02
128	3.1892e-02	3.2893e-02	3.2843e-02
256	1.8038e-02	1.8871e-02	1.8325e-02

E_1-Fehlerrate

Dim.	SLIC	H&N	ELVIRA
64/32	0.4691	0.4623	0.4641
128/64	0.5451	0.5515	0.5475
256/128	0.5656	0.5737	0.5580

Gesamter relativer Volumenverlust

Dim.	SLIC	H&N	ELVIRA
32	-1.5689e-10	-1.2663e-10	-2.9035e-08
64	-1.2207e-09	-1.3672e-09	-5.5825e-08
128	-2.0143e-09	-1.9903e-09	-9.7834e-08
256	-3.7268e-09	-3.4952e-09	-1.8863e-07

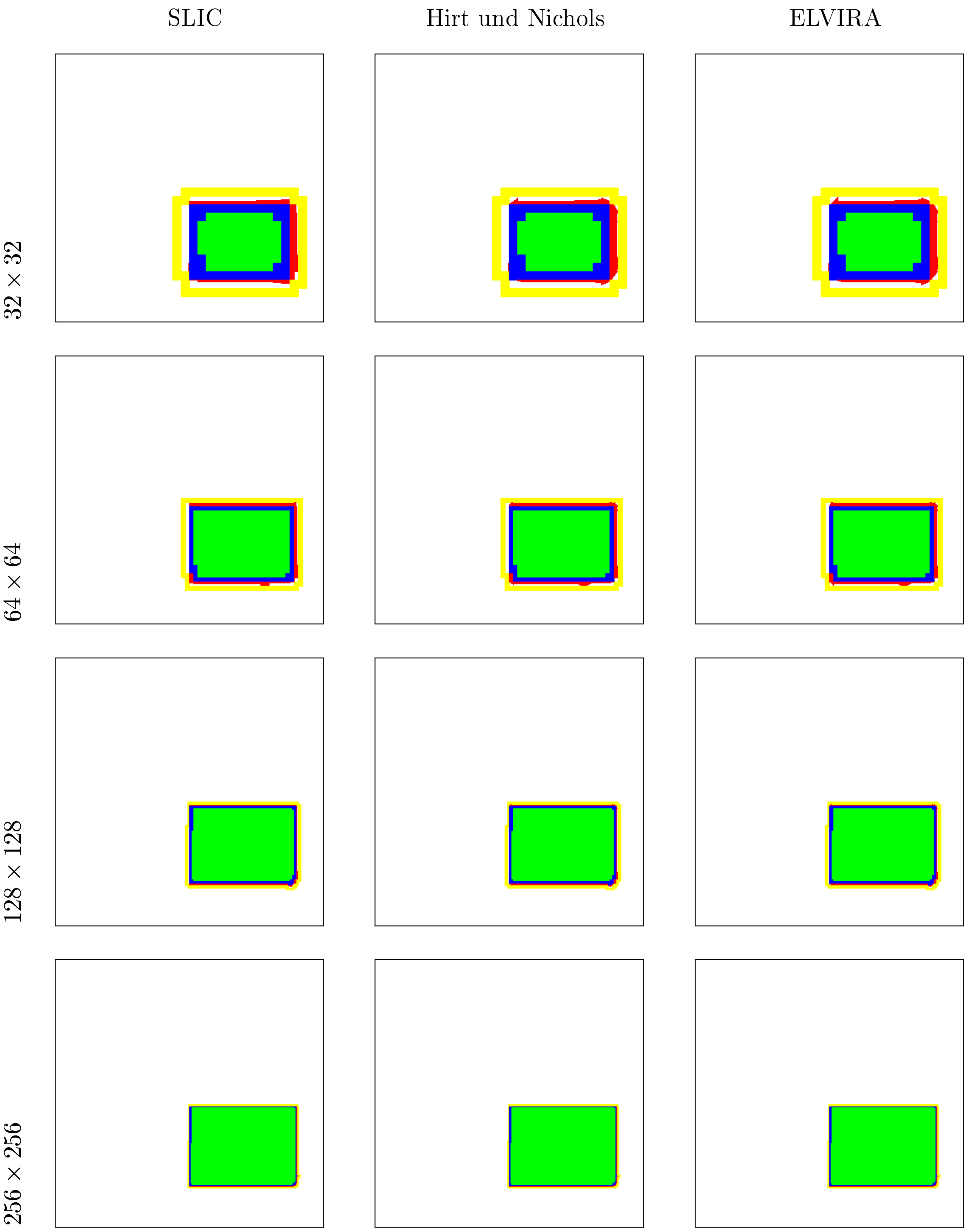
Durchschnittlicher relativer Volumenverlust

Dim.	SLIC	H&N	ELVIRA
32	3.0512e-11	4.1644e-11	1.3563e-08
64	5.4984e-10	6.0483e-10	2.6403e-08
128	9.1177e-10	8.8491e-10	4.4625e-08
256	1.4610e-09	1.3821e-09	8.0203e-08

Füllungstoleranz

Dim.	SLIC	H&N	ELVIRA
32	2.3283e-10	2.3283e-10	1.0537e-08
64	2.3283e-10	2.3283e-10	1.0537e-08
128	2.3283e-10	2.3283e-10	1.0537e-08
256	2.3283e-10	2.3283e-10	1.0537e-08

Rechtecktransport, *donor-acceptor*-Advektion



A.2.2 Rechtecktransport, *fractional-step*-Advektion

E_1-Fehler

Dim.	SLIC	H&N	COM	P&Y	LVIRA	ELVIRA
32	1.5991e-02	4.3326e-02	3.1066e-02	3.8040e-02	3.5106e-02	4.4031e-02
64	4.9357e-03	1.5589e-02	9.5915e-03	1.2791e-02	1.2510e-02	1.6338e-02
128	4.2469e-03	5.6086e-03	2.9781e-03	4.8204e-03	6.0234e-03	5.9469e-03
256	1.9712e-03	1.9946e-03	9.2335e-04	1.5320e-03	1.9362e-03	2.1235e-03

E_1-Fehlerrate

Dim.	SLIC	H&N	COM	P&Y	LVIRA	ELVIRA
64/32	0.3087	0.3598	0.3087	0.3363	0.3563	0.3711
128/64	0.8604	0.3598	0.3105	0.3769	0.4815	0.3640
256/128	0.4642	0.3556	0.3100	0.3178	0.3214	0.3571

Gesamter relativer Volumenverlust

Dim.	SLIC	H&N	COM	P&Y	LVIRA	ELVIRA
32	3.9641e-17	5.5283e-17	1.0102e-16	1.5286e-16	-5.8140e-14	1.5286e-16
64	-4.3425e-16	2.6258e-17	1.5811e-17	-1.1282e-16	2.0546e-14	1.6116e-16
128	-2.5177e-04	-1.7573e-16	-4.5175e-18	-2.7235e-16	-2.6705e-15	-2.5213e-16
256	4.6236e-15	4.6717e-16	3.0087e-16	4.6553e-16	3.8964e-14	6.2788e-16

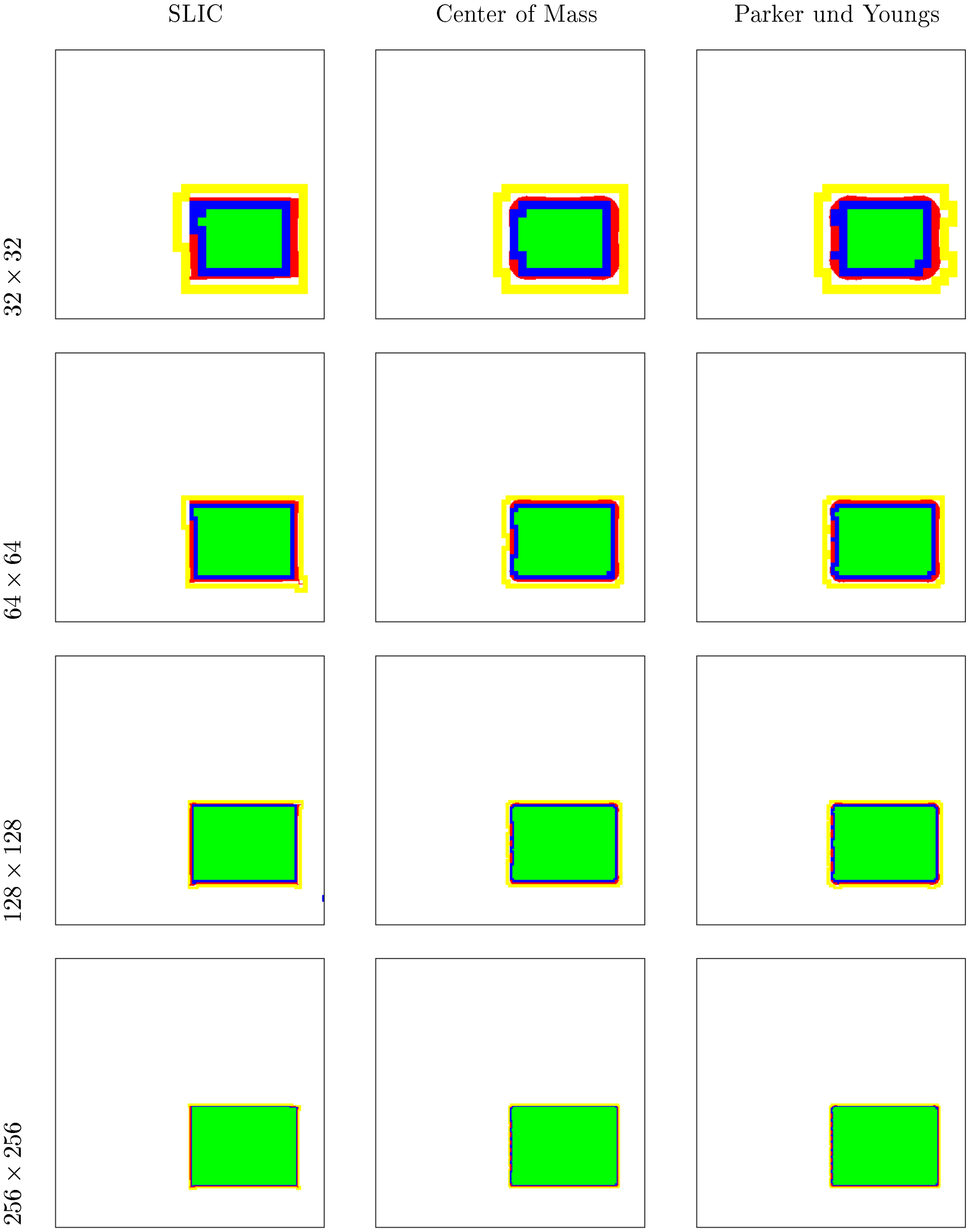
Durchschnittlicher relativer Volumenverlust

Dim.	SLIC	H&N	COM	P&Y	LVIRA	ELVIRA
32	2.3594e-16	2.4445e-16	1.8628e-16	2.4662e-16	3.5809e-14	2.3823e-16
64	9.2233e-16	6.6223e-16	6.0590e-16	6.4694e-16	2.8029e-14	5.9375e-16
128	9.1993e-05	7.5372e-16	6.2632e-16	7.6512e-16	1.1455e-14	8.1604e-16
256	3.0407e-07	3.0393e-15	2.9397e-15	3.0029e-15	1.6519e-14	3.1669e-15

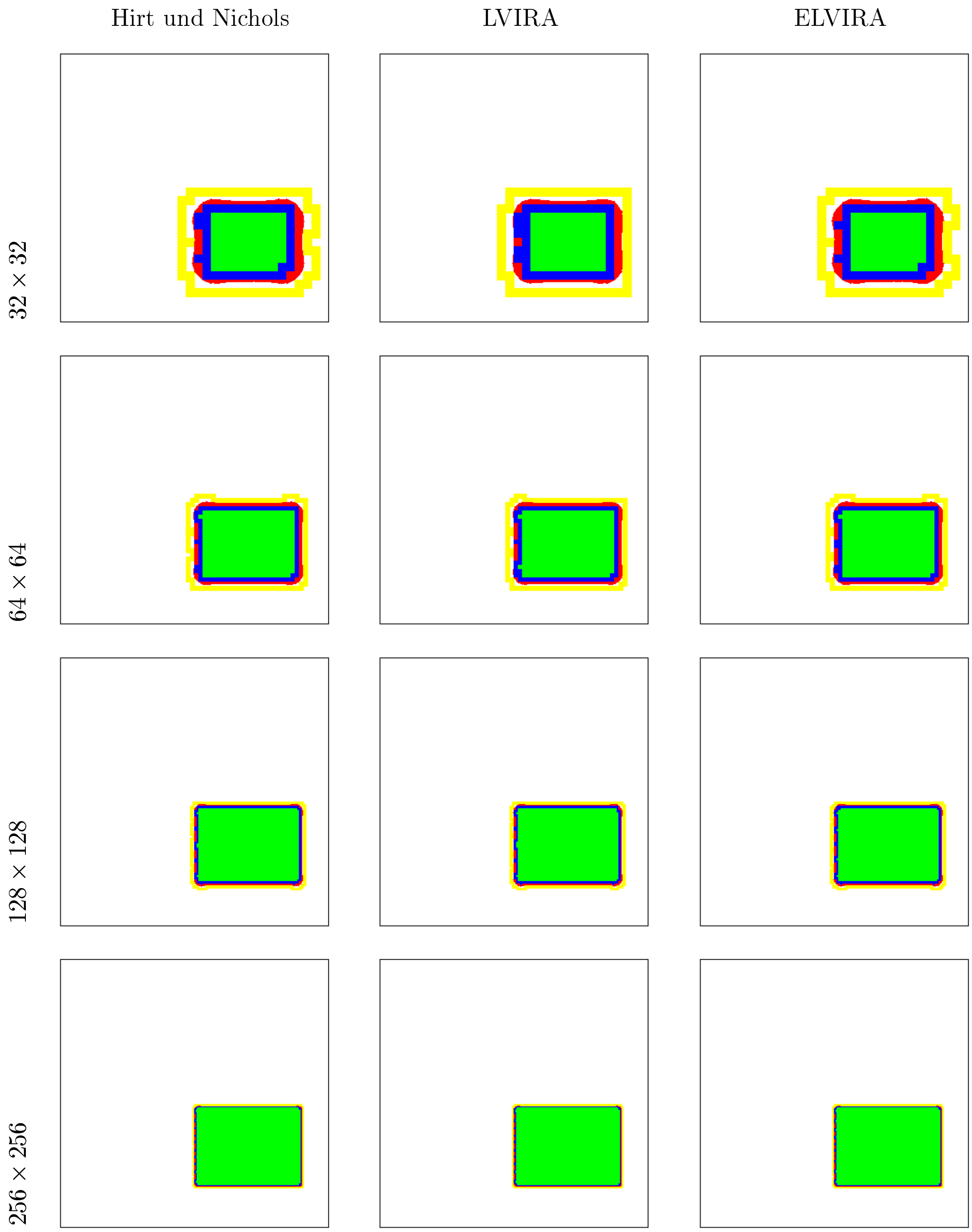
Füllungstoleranz

Dim.	SLIC	H&N	COM	P&Y	LVIRA	ELVIRA
32	3.5527e-15	3.5527e-15	3.5527e-15	3.5527e-15	6.5375e-13	3.5527e-15
64	3.5527e-15	5.3291e-15	5.1070e-15	5.7732e-15	1.5875e-12	5.3291e-15
128	5.6621e-15	1.3101e-14	9.3259e-15	1.4433e-14	3.1659e-12	1.0880e-14
256	1.8652e-14	1.9362e-14	1.7786e-14	1.6530e-14	6.4088e-12	2.0051e-14

Rechtecktransport, *fractional-step*-Advektion



Rechtecktransport, *fractional-step*-Advektion



A.2.3 Rechtecktransport, Pilliods Advektion

E_1-Fehler

Dim.	SLIC	H&N	COM	P&Y	LVIRA	ELVIRA
32	7.5918e-03	4.3464e-02	3.1834e-02	3.7477e-02	2.8862e-02	4.5084e-02
64	7.5628e-03	1.5583e-02	9.5291e-03	1.2898e-02	1.0361e-02	1.5987e-02
128	3.5523e-03	5.2553e-03	3.0784e-03	4.2157e-03	4.1170e-03	5.2616e-03
256	2.0925e-03	1.9412e-03	9.3607e-04	1.4940e-03	1.8489e-03	2.0773e-03

E_1-Fehlerrate

Dim.	SLIC	H&N	COM	P&Y	LVIRA	ELVIRA
64/32	0.9962	0.3585	0.2993	0.3442	0.3590	0.3546
128/64	0.4697	0.3372	0.3231	0.3268	0.3974	0.3291
256/128	0.5891	0.3694	0.3041	0.3544	0.4491	0.3948

Gesamter relativer Volumenverlust

Dim.	SLIC	H&N	COM	P&Y	LVIRA	ELVIRA
32	4.0104e-16	1.7872e-16	1.7449e-16	1.5885e-16	-2.4040e-14	3.2944e-16
64	-2.8844e-16	-1.1762e-16	-2.6314e-17	-2.2520e-16	4.4094e-14	-1.0627e-16
128	-6.6859e-16	-2.6902e-16	1.2310e-16	-2.7003e-16	2.8850e-14	-1.4620e-16
256	-7.2568e-16	2.8878e-16	6.7717e-16	3.7077e-16	-9.1971e-14	6.6594e-16

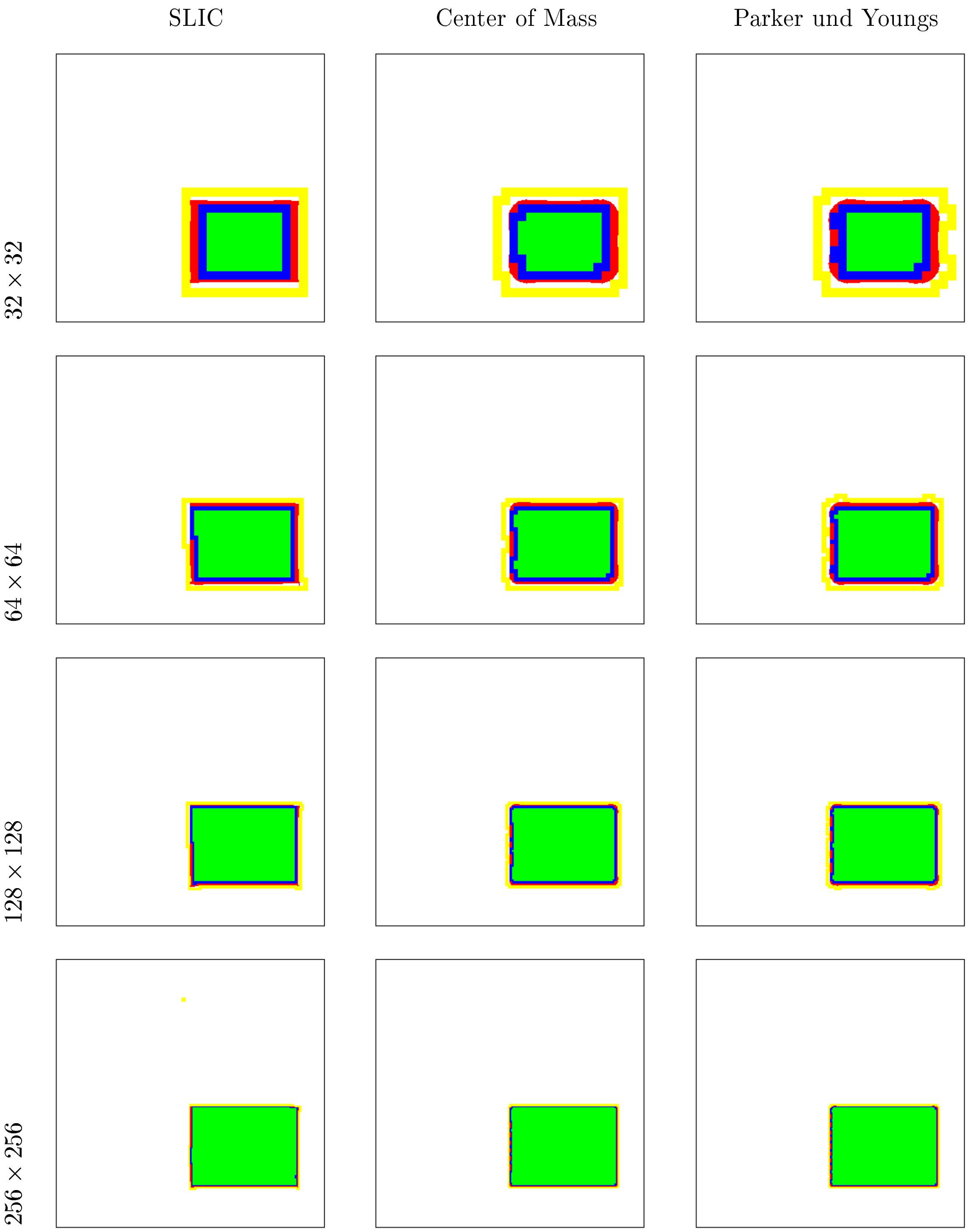
Durchschnittlicher relativer Volumenverlust

Dim.	SLIC	H&N	COM	P&Y	LVIRA	ELVIRA
32	3.5682e-16	2.1351e-16	1.7789e-16	1.8357e-16	8.8566e-15	2.6952e-16
64	6.0996e-16	4.7663e-16	4.0912e-16	4.7643e-16	4.0733e-14	4.4928e-16
128	9.4171e-16	5.3929e-16	3.2366e-16	5.5563e-16	2.2421e-14	4.7066e-16
256	5.7132e-16	1.3246e-15	1.7134e-15	1.3599e-15	3.8385e-14	1.5660e-15

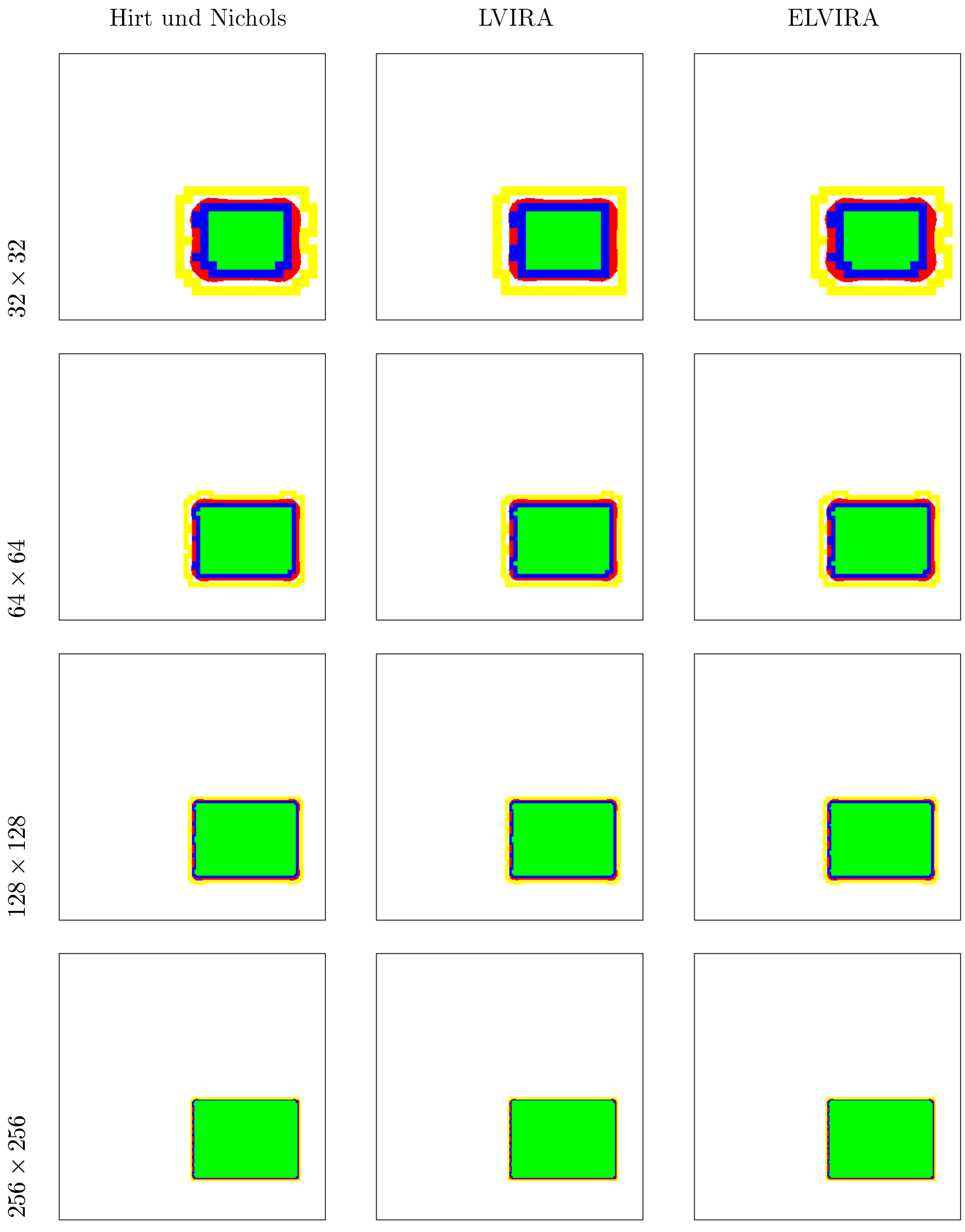
Füllungstoleranz

Dim.	SLIC	H&N	COM	P&Y	LVIRA	ELVIRA
32	3.5527e-15	3.5527e-15	3.5527e-15	3.5527e-15	6.3923e-13	3.5527e-15
64	3.5527e-15	7.3275e-15	6.2172e-15	5.8065e-15	1.2577e-12	8.8818e-15
128	4.2166e-15	1.5543e-14	1.1546e-14	1.3545e-14	2.8781e-12	1.2212e-14
256	5.7305e-15	2.4647e-14	2.0428e-14	1.8266e-14	6.6803e-12	2.2871e-14

Rechtecktransport, Pilliods Advektion



Rechtecktransport, Pilliods-Advektion



A.3 Testproblem Kreistransport

A.3.1 Kreistransport, *donor-acceptor*-Advektion

E_1-Fehler

Dim.	SLIC	H&N	COM	P&Y	LVIRA	ELVIRA
32	4.1984e-02	4.1940e-02	4.1940e-02	4.1940e-02	4.1940e-02	4.1984e-02
64	2.6117e-02	2.5628e-02	2.5628e-02	2.5628e-02	2.5628e-02	2.5656e-02
128	1.2338e-02	1.2301e-02	1.2301e-02	1.2253e-02	1.2326e-02	1.2674e-02
256	7.2441e-03	7.1824e-03	7.1824e-03	7.3819e-03	7.2425e-03	7.3456e-03

E_1-Fehlerrate

Dim.	SLIC	H&N	COM	P&Y	LVIRA	ELVIRA
64/32	0.6221	0.6111	0.6111	0.6111	0.6111	0.6111
128/64	0.4724	0.4800	0.4800	0.4781	0.4810	0.4940
256/128	0.5871	0.5839	0.5839	0.6025	0.5876	0.5796

Gesamter relativer Volumenverlust

Dim.	SLIC	H&N	COM	P&Y	LVIRA	ELVIRA
32	-4.0227e-17	-4.1845e-17	-4.1845e-17	-4.1845e-17	-4.1845e-17	-4.0227e-17
64	-1.8334e-17	-2.0599e-17	-2.0599e-17	-2.0599e-17	-2.0599e-17	-1.3299e-11
128	-6.9022e-18	-7.7650e-18	-7.7650e-18	-7.3336e-18	-2.9573e-11	-4.2490e-11
256	-5.1166e-11	-4.7296e-11	-4.7012e-11	-3.3280e-11	-2.5139e-10	-2.6818e-10

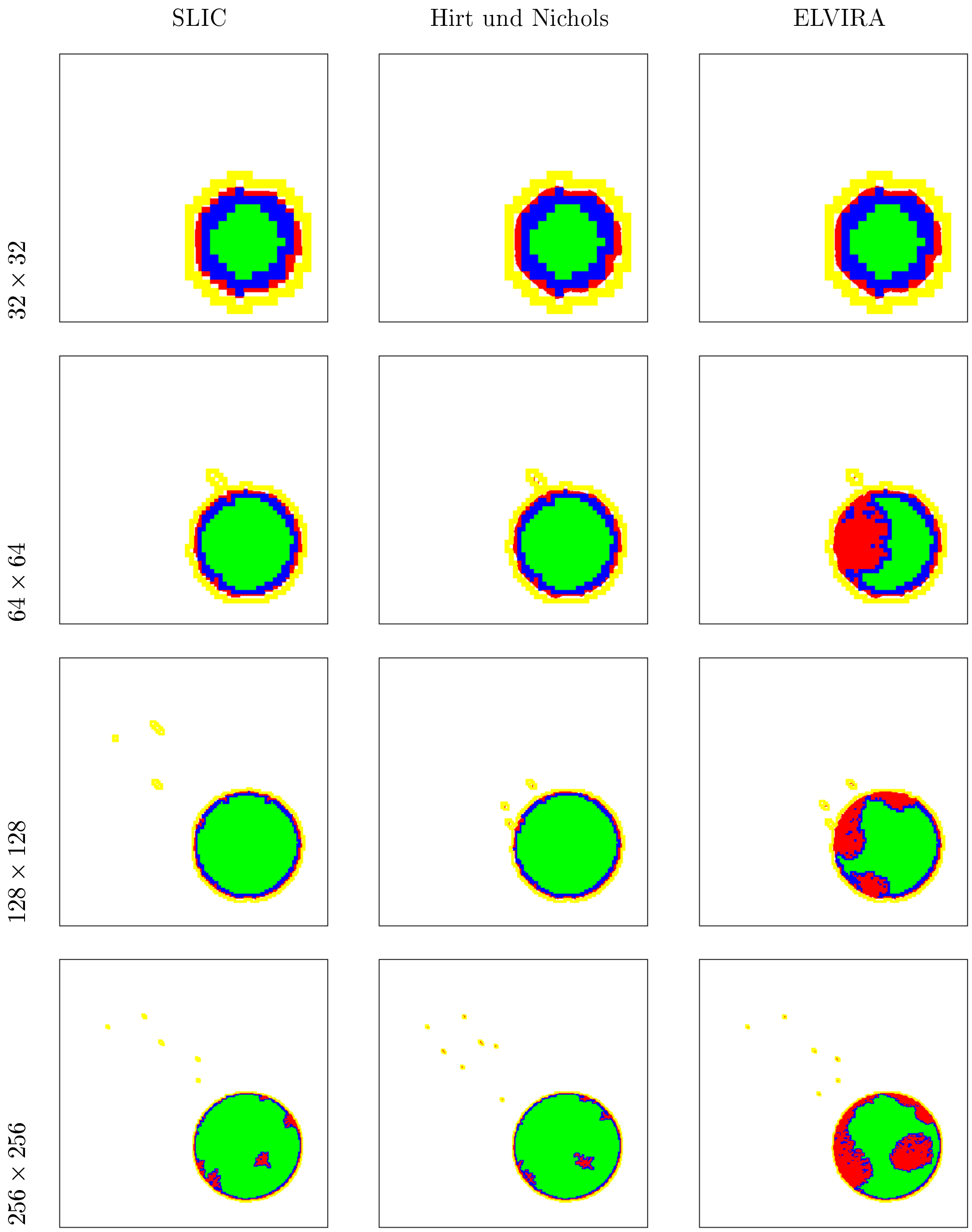
Durchschnittlicher relativer Volumenverlust

Dim.	SLIC	H&N	COM	P&Y	LVIRA	ELVIRA
32	1.8376e-17	1.8525e-17	1.8525e-17	1.8525e-17	1.8525e-17	1.8376e-17
64	1.8782e-17	1.9261e-17	1.9261e-17	1.9261e-17	1.9261e-17	1.4115e-12
128	2.8345e-17	2.8467e-17	2.8467e-17	2.8592e-17	5.9329e-12	1.0245e-11
256	1.8479e-11	1.6712e-11	1.6931e-11	1.2966e-11	9.2186e-11	9.0288e-11

Füllungstoleranz

Dim.	SLIC	H&N	COM	P&Y	LVIRA	ELVIRA
32	2.3283e-10	2.3283e-10	2.3283e-10	2.3283e-10	2.3283e-10	2.3283e-10
64	2.3283e-10	2.3283e-10	2.3283e-10	2.3283e-10	2.3283e-10	2.3283e-10
128	2.3283e-10	2.3283e-10	2.3283e-10	2.3283e-10	2.3283e-10	2.3283e-10
256	2.3283e-10	2.3283e-10	2.3283e-10	2.3283e-10	2.3283e-10	2.3283e-10

Kreistransport, *donor-acceptor*-Advektion



A.3.2 Kreistransport, *fractional-step*-Advektion

E_1-Fehler

Dim.	SLIC	H&N	COM	P&Y	LVIRA	ELVIRA
32	1.2272e-01	4.9864e-03	3.9240e-02	5.2924e-03	8.1717e-03	7.3919e-03
64	7.2453e-02	9.2564e-04	2.1938e-02	4.3387e-03	1.7830e-03	1.5249e-03
128	4.4097e-02	1.5684e-04	1.1431e-02	2.5927e-03	3.4544e-04	2.3257e-04
256	2.5407e-02	1.0752e-04	6.6116e-03	1.2979e-03	9.9029e-05	8.7223e-05

E_1-Fehlerrate

Dim.	SLIC	H&N	COM	P&Y	LVIRA	ELVIRA
64/32	0.5904	0.1856	0.5591	0.8198	0.2182	0.2063
128/64	0.6086	0.1694	0.5211	0.5976	0.1937	0.1525
256/128	0.5762	0.6855	0.5784	0.5006	0.2867	0.3750

Gesamter relativer Volumenverlust

Dim.	SLIC	H&N	COM	P&Y	LVIRA	ELVIRA
32	8.7680e-17	1.9413e-17	8.0346e-17	7.5278e-17	-1.8794e-14	2.0459e-16
64	-4.2708e-17	-1.1162e-16	-1.6921e-16	-2.5021e-17	-2.9022e-14	-4.2815e-17
128	-9.6200e-17	-2.6315e-17	-1.2521e-16	5.4247e-17	-1.7519e-14	-1.5961e-17
256	3.0053e-04	6.5787e-17	3.3293e-16	1.6005e-16	2.4823e-14	2.2691e-16

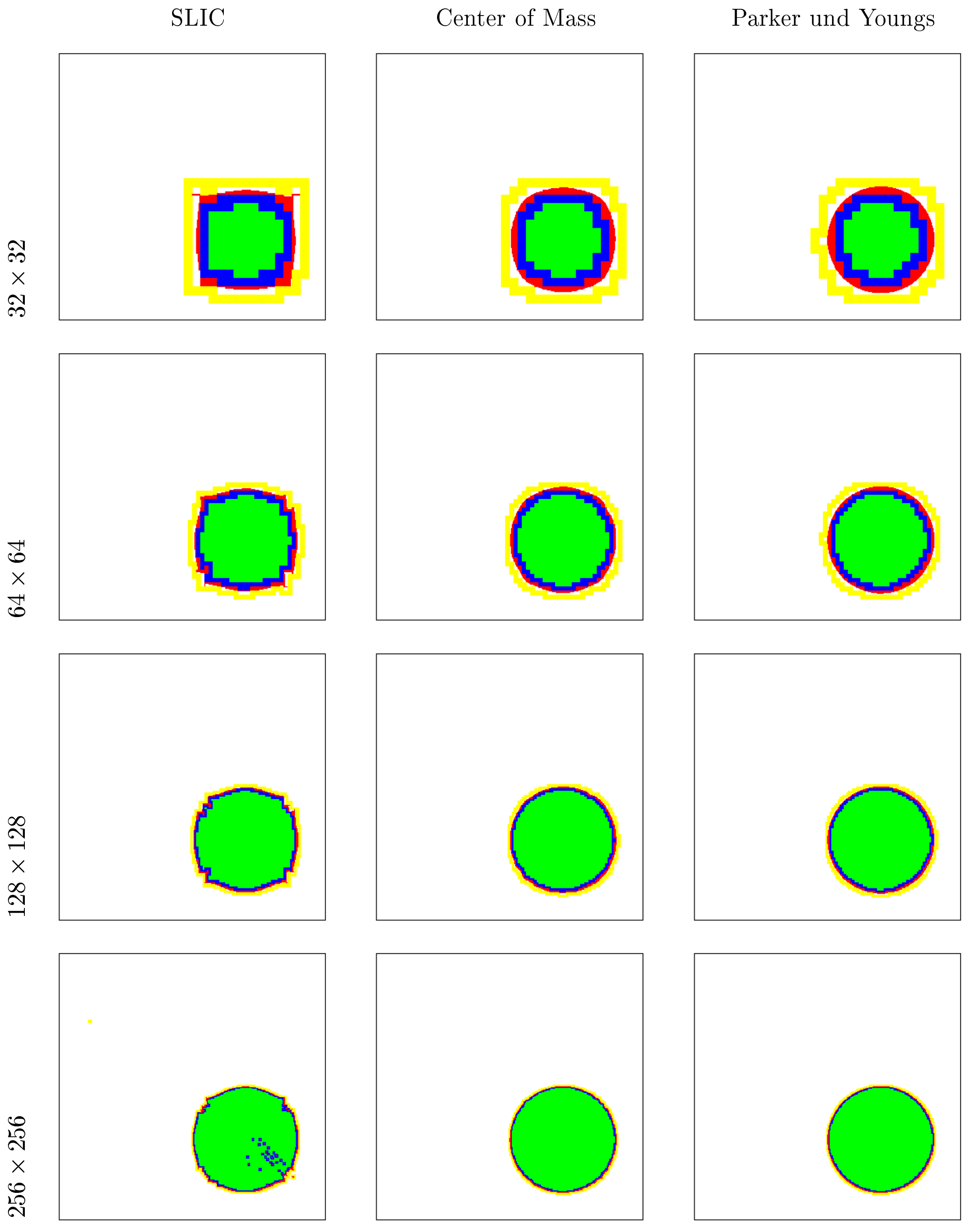
Durchschnittlicher relativer Volumenverlust

Dim.	SLIC	H&N	COM	P&Y	LVIRA	ELVIRA
32	1.9061e-16	1.9081e-16	1.8760e-16	1.9495e-16	1.5592e-14	3.0042e-16
64	6.6553e-16	6.9013e-16	7.3791e-16	6.6423e-16	1.8375e-14	6.2848e-16
128	7.1927e-16	6.6517e-16	7.1903e-16	6.1653e-16	1.1257e-14	6.4215e-16
256	1.1092e-04	2.8247e-15	3.0970e-15	2.8900e-15	4.0811e-14	2.9028e-15

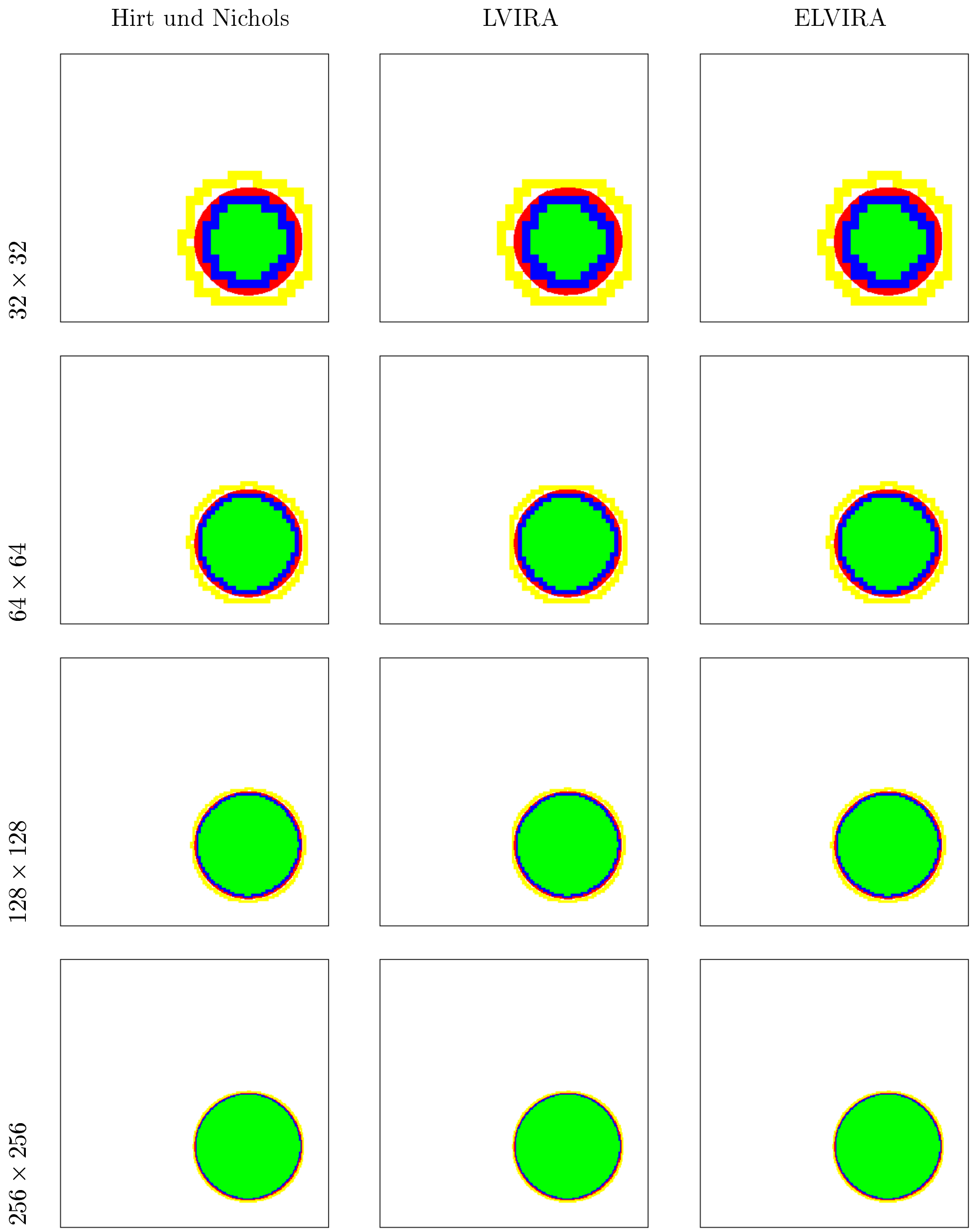
Füllungstoleranz

Dim.	SLIC	H&N	COM	P&Y	LVIRA	ELVIRA
32	3.5527e-15	3.5527e-15	3.5527e-15	3.5527e-15	5.0271e-13	3.5527e-15
64	3.5527e-15	3.9968e-15	5.9952e-15	5.3291e-15	1.1130e-12	4.2188e-15
128	3.9968e-15	1.0880e-14	1.3323e-14	1.2657e-14	1.8844e-12	1.0164e-14
256	5.6843e-15	2.2649e-14	2.5163e-14	2.7861e-14	4.4618e-12	2.5313e-14

Kreistransport, *fractional-step*-Advektion



Kreistransport, *fractional-step*-Advektion



A.3.3 Kreistransport, Pilliods Advektion

E_1-Fehler

Dim.	SLIC	H&N	COM	P&Y	LVIRA	ELVIRA
32	1.2265e-01	4.9401e-03	3.2621e-02	5.0486e-03	9.5064e-03	7.9033e-03
64	8.2403e-02	1.0670e-03	1.4449e-02	3.1037e-03	2.3099e-03	1.8443e-03
128	5.0898e-02	1.7915e-04	9.9987e-03	2.2727e-03	4.0129e-04	2.8759e-04
256	2.9699e-02	1.2647e-04	5.0068e-03	1.2176e-03	1.4084e-04	1.3264e-04

E_1-Fehlerrate

Dim.	SLIC	H&N	COM	P&Y	LVIRA	ELVIRA
64/32	0.6719	0.2160	0.4429	0.6148	0.2430	0.2334
128/64	0.6177	0.1679	0.6920	0.7323	0.1737	0.1559
256/128	0.5835	0.7059	0.5007	0.5358	0.3510	0.4612

Gesamter relativer Volumenverlust

Dim.	SLIC	H&N	COM	P&Y	LVIRA	ELVIRA
32	2.8580e-17	1.3330e-16	9.4367e-17	-8.6278e-19	-5.2533e-16	2.6638e-17
64	-8.8327e-17	-1.1000e-17	-1.3352e-16	-1.5746e-17	-6.4455e-15	-8.2719e-17
128	9.6308e-17	-5.2522e-17	-1.6177e-17	3.1707e-17	1.5116e-14	5.5002e-18
256	2.2540e-17	2.8224e-16	2.0103e-16	8.2396e-17	1.9286e-15	-2.4266e-17

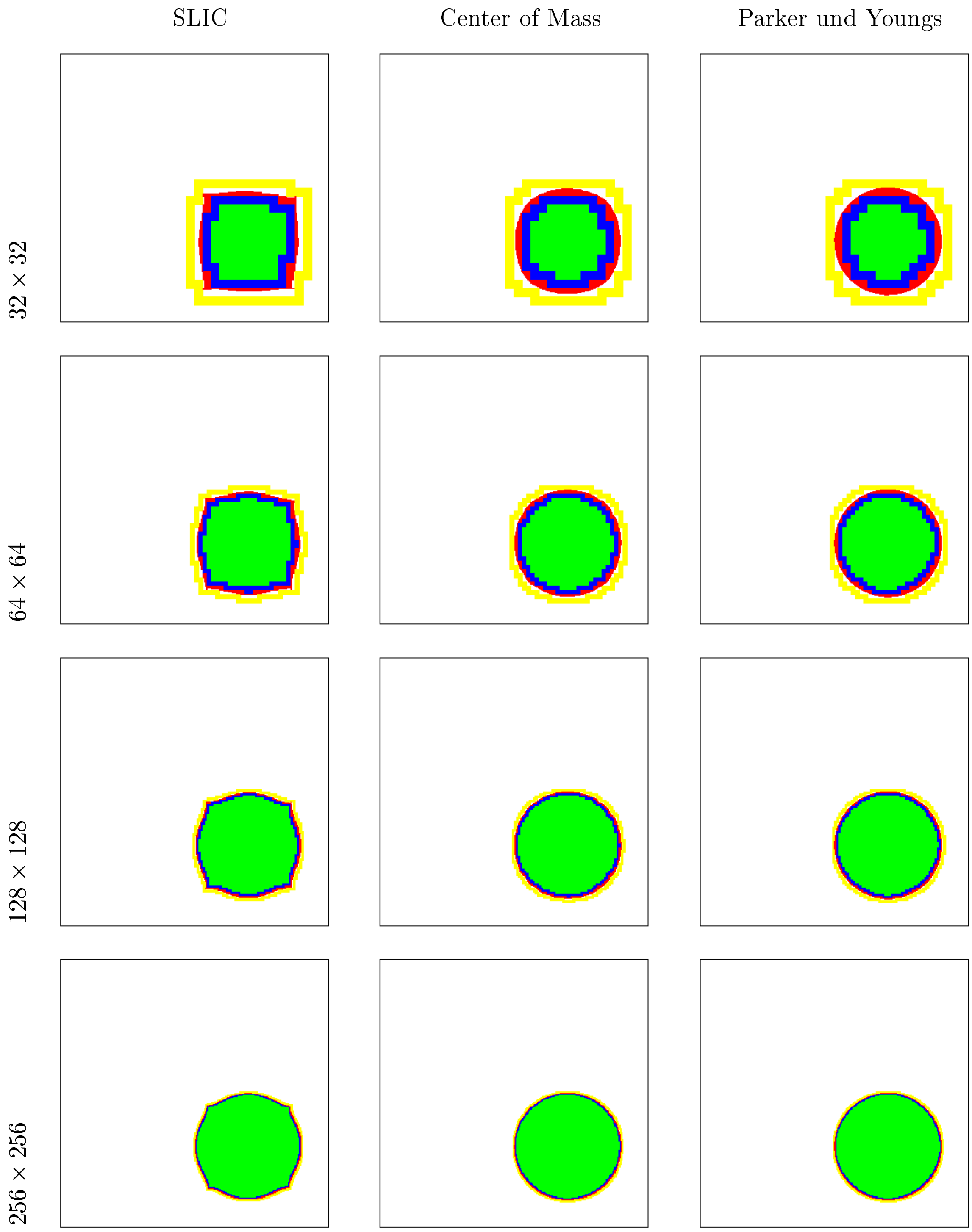
Durchschnittlicher relativer Volumenverlust

Dim.	SLIC	H&N	COM	P&Y	LVIRA	ELVIRA
32	9.0683e-17	1.1555e-16	1.6401e-16	1.0528e-16	1.9842e-15	9.4806e-17
64	3.1632e-16	2.5342e-16	3.4164e-16	2.0108e-16	4.8799e-15	2.7165e-16
128	1.9316e-16	2.5719e-16	2.3306e-16	2.1733e-16	3.5288e-15	2.2345e-16
256	3.0081e-16	1.0779e-15	1.0873e-15	1.0093e-15	1.8694e-15	1.0063e-15

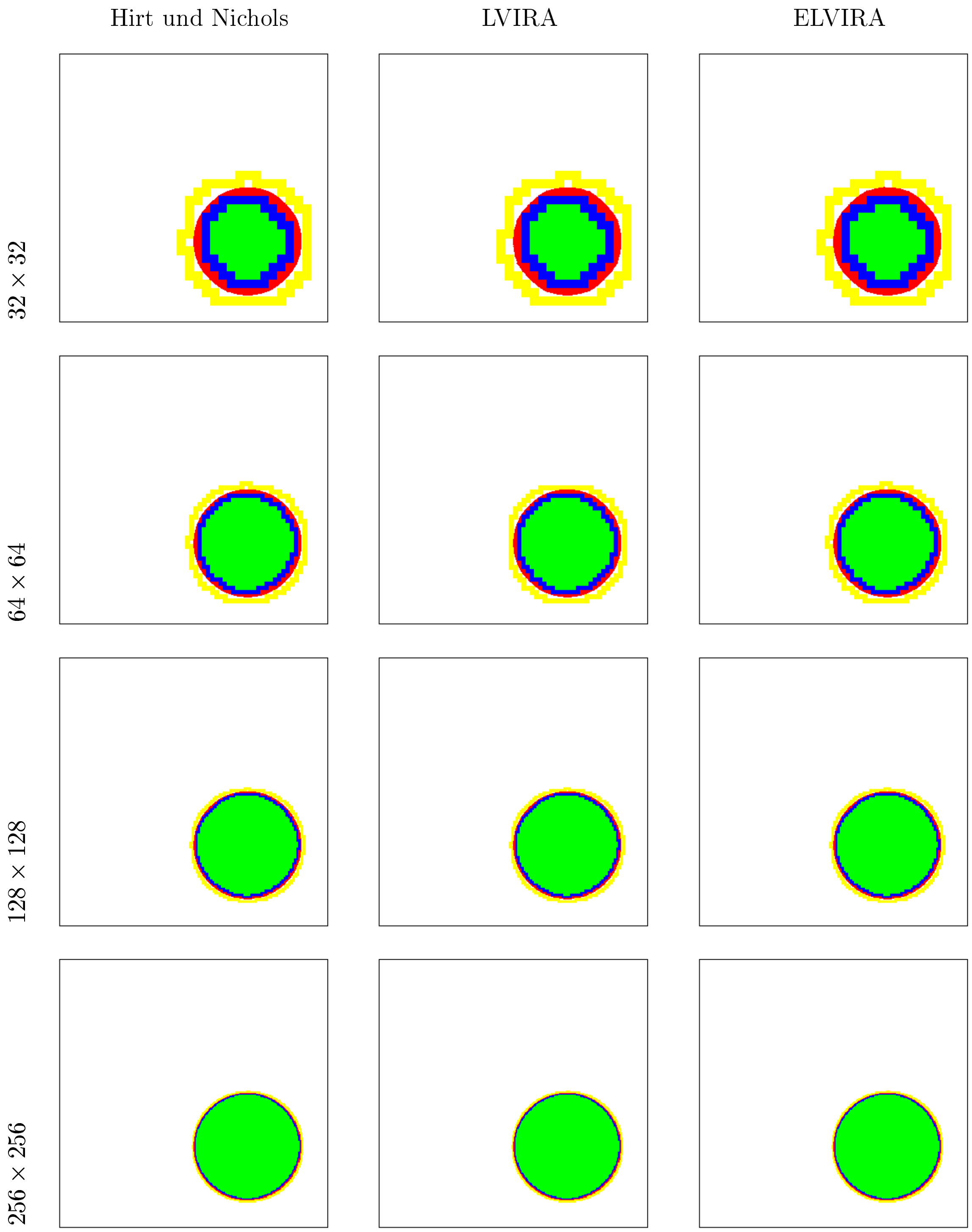
Füllungstoleranz

Dim.	SLIC	H&N	COM	P&Y	LVIRA	ELVIRA
32	3.5527e-15	3.5527e-15	3.5527e-15	3.5527e-15	3.9611e-13	3.5527e-15
64	3.5527e-15	9.1904e-15	9.6412e-15	7.5495e-15	8.7451e-13	7.8448e-15
128	4.2188e-15	1.5267e-14	1.5321e-14	1.5479e-14	1.8028e-12	1.7319e-14
256	5.7127e-15	2.7756e-14	3.0420e-14	2.4072e-14	3.7756e-12	2.9130e-14

Kreistransport, Pilliods Advektion



Kreistransport, Pilliods Advektion



A.4 Testproblem Rechteckrotation

A.4.1 Rechteckrotation, *donor-acceptor*-Advektion

E_1-Fehler

Dim.	SLIC	H&N	ELVIRA
32	2.2328e-01	2.1445e-01	2.1689e-01
64	9.2107e-02	9.1615e-02	9.4031e-02
128	5.6003e-02	7.0272e-02	7.1227e-02
256	3.1209e-02	3.5120e-02	3.5884e-02

E_1-Fehlerrate

Dim.	SLIC	H&N	ELVIRA
64/32	0.4125	0.4272	0.4335
128/64	0.6080	0.7670	0.7575
256/128	0.5573	0.4998	0.5038

Gesamter relativer Volumenverlust

Dim.	SLIC	H&N	ELVIRA
32	-1.3876e-06	-9.5401e-07	-2.7880e-06
64	-4.4693e-06	-4.4362e-06	-4.6898e-06
128	-9.6398e-06	-8.3532e-06	-8.7308e-06
256	-4.6906e-09	-4.4073e-09	-1.9177e-09

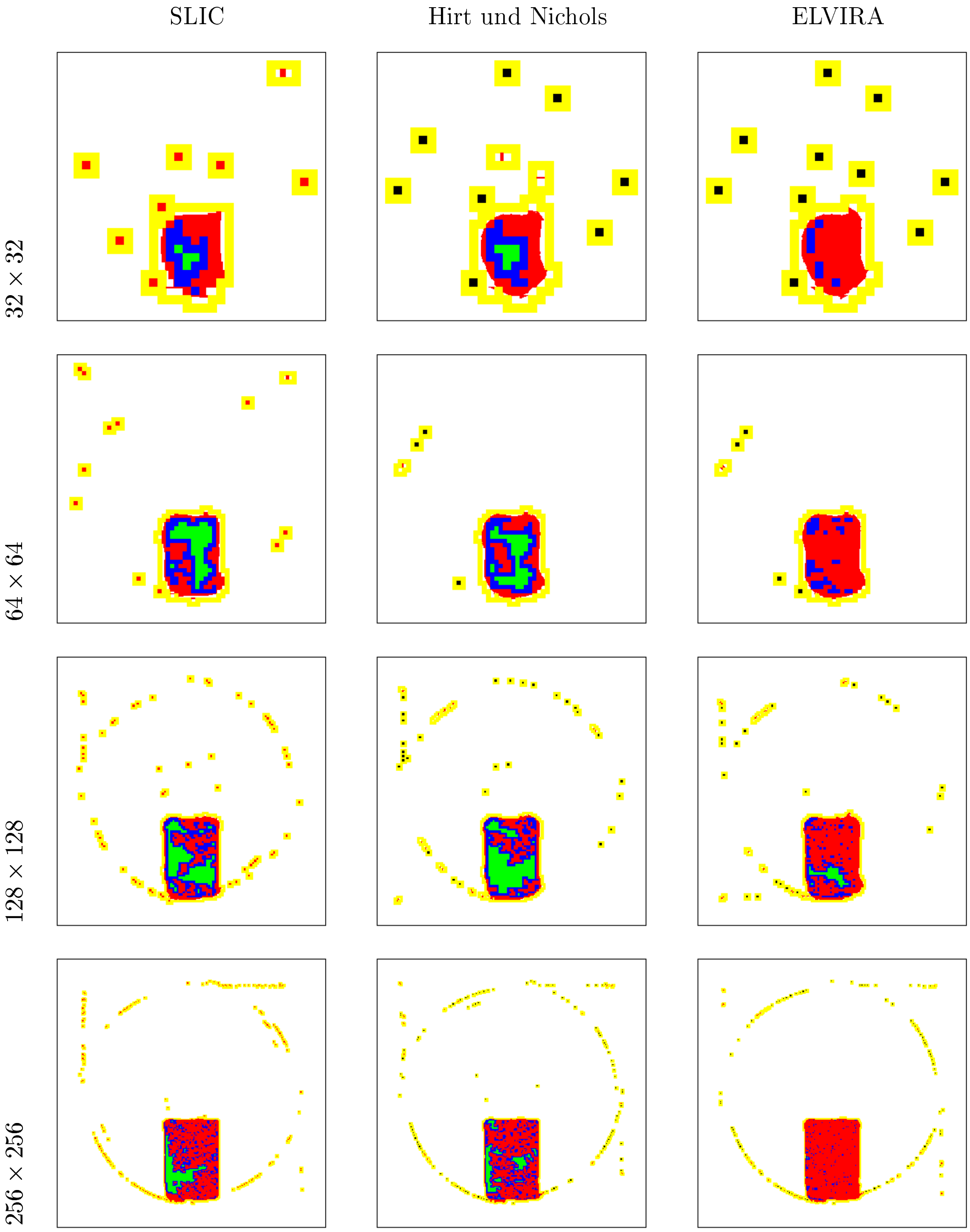
Durchschnittlicher relativer Volumenverlust

Dim.	SLIC	H&N	ELVIRA
32	6.3726e-07	4.4555e-07	1.2962e-06
64	2.0743e-06	2.0745e-06	2.1909e-06
128	4.2327e-06	3.6166e-06	4.0218e-06
256	2.0805e-09	1.9343e-09	1.0514e-09

Füllungstoleranz

Dim.	SLIC	H&N	ELVIRA
32	6.9849e-07	6.9849e-07	6.9849e-07
64	6.9849e-07	6.9849e-07	6.9849e-07
128	6.9849e-07	6.9849e-07	6.9849e-07
256	2.3283e-10	2.3283e-10	2.3283e-10

Rechteckrotation, *donor-acceptor*-Advektion



A.4.2 Rechteckrotation, *fractional-step*-Advektion

E_1-Fehler

Dim.	SLIC	H&N	COM	P&Y	LVIRA	ELVIRA
32	3.6348e-01	1.3019e-01	1.3630e-01	1.2741e-01	1.2891e-01	1.2971e-01
64	3.6659e-01	5.3984e-02	6.4145e-02	5.3306e-02	5.5032e-02	5.4798e-02
128	3.4005e-01	2.0274e-02	2.7092e-02	1.9984e-02	2.0557e-02	2.0497e-02
256	3.8321e-01	7.4082e-03	1.2986e-02	7.3754e-03	7.6451e-03	7.5629e-03

E_1-Fehlerrate

Dim.	SLIC	H&N	COM	P&Y	LVIRA	ELVIRA
64/32	1.0086	0.4147	0.4706	0.4184	0.4269	0.4225
128/64	0.9276	0.3756	0.4224	0.3749	0.3735	0.3740
256/128	1.1269	0.3654	0.4793	0.3691	0.3719	0.3690

Gesamter relativer Volumenverlust

Dim.	SLIC	H&N	COM	P&Y	LVIRA	ELVIRA
32	-1.6881e-01	1.8619e-14	1.8851e-14	1.9198e-14	4.2475e-11	1.9776e-14
64	-2.3415e-01	-3.7007e-15	-3.9320e-15	-3.7007e-15	6.1933e-11	-3.1225e-15
128	-2.6316e-01	4.5103e-15	4.2790e-15	3.4694e-15	2.3269e-11	2.8912e-15
256	-3.3867e-01	-3.9320e-15	-4.1633e-15	-4.1633e-15	2.1996e-13	-4.7259e-15

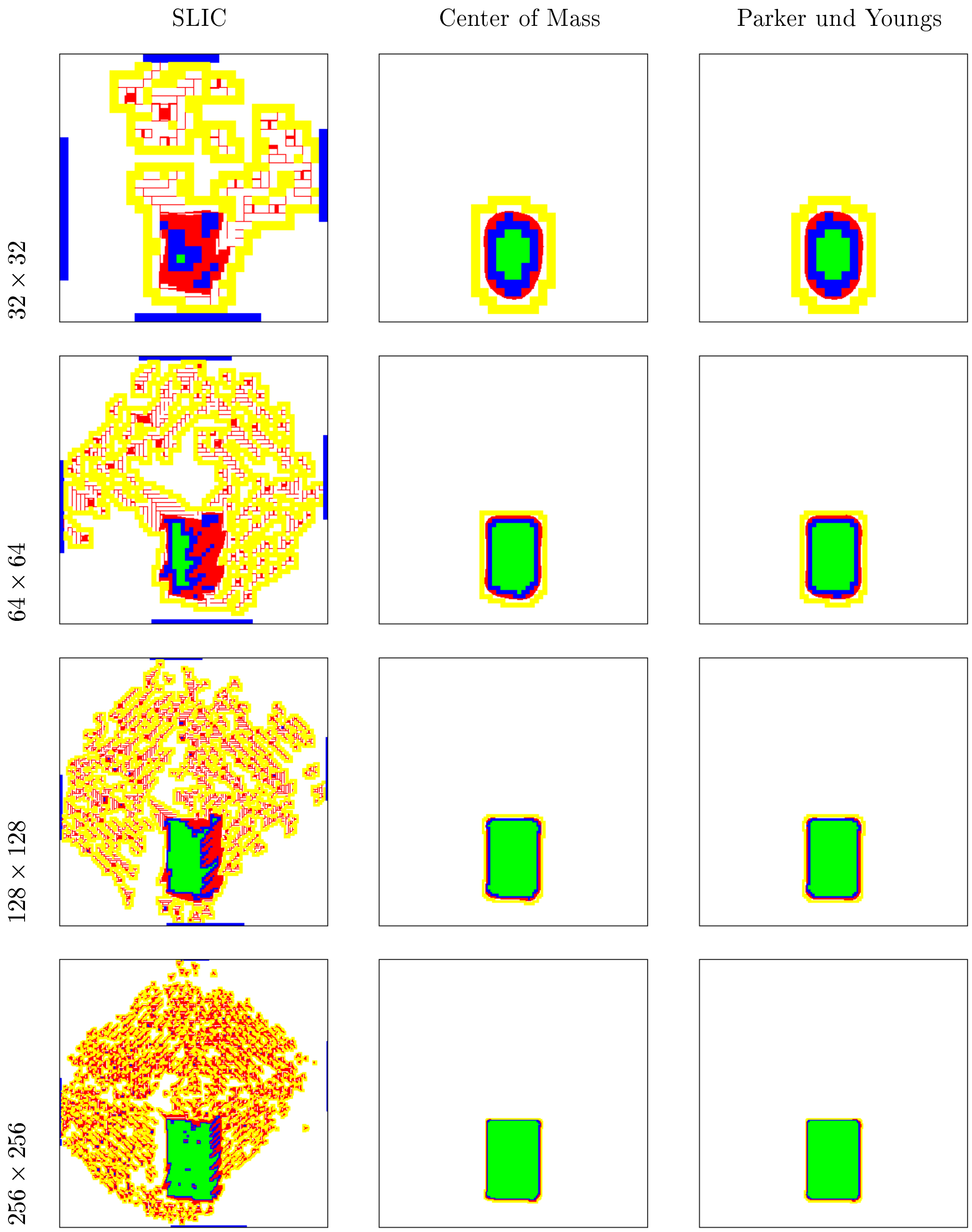
Durchschnittlicher relativer Volumenverlust

Dim.	SLIC	H&N	COM	P&Y	LVIRA	ELVIRA
32	5.3038e-02	8.4648e-15	8.5242e-15	8.3054e-15	1.2941e-11	8.7393e-15
64	7.7391e-02	1.6283e-15	1.8650e-15	1.5458e-15	3.4674e-11	1.4426e-15
128	7.5929e-02	1.7685e-15	1.6484e-15	1.7311e-15	1.5065e-11	2.0584e-15
256	9.7094e-02	1.2233e-15	1.6626e-15	1.2158e-15	2.3924e-13	1.3107e-15

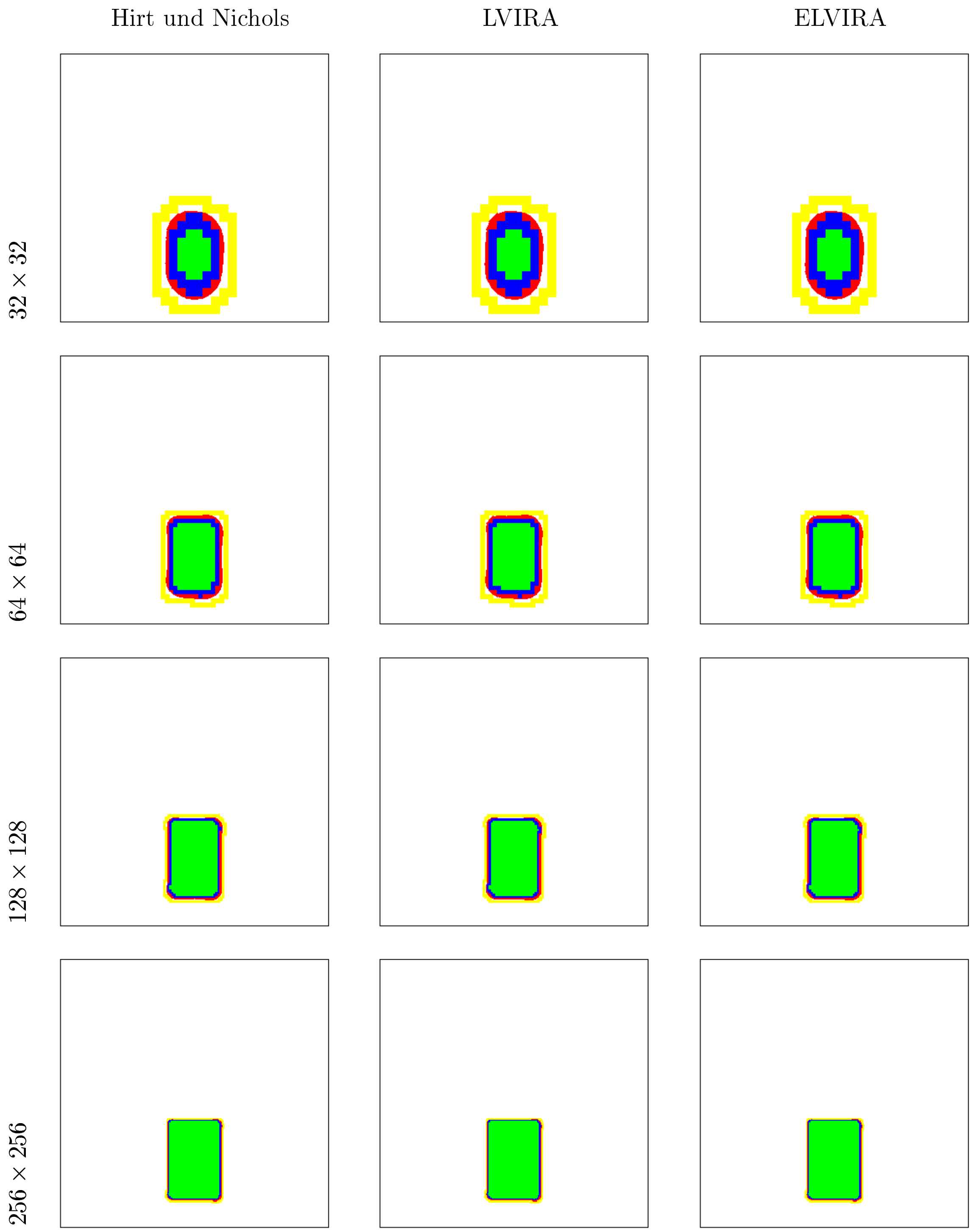
Füllungstoleranz

Dim.	SLIC	H&N	COM	P&Y	LVIRA	ELVIRA
32	1.0816e-12	1.0816e-12	1.0816e-12	1.0816e-12	5.2109e-10	1.0816e-12
64	1.0816e-12	1.0816e-12	1.0816e-12	1.0816e-12	1.1517e-09	1.0816e-12
128	1.0816e-12	1.0816e-12	1.0816e-12	1.0816e-12	2.4846e-09	1.0816e-12
256	1.0816e-12	1.0816e-12	1.0816e-12	1.0816e-12	1.6980e-11	4.7740e-14

Rechteckrotation, *fractional-step*-Advektion



Rechteckrotation, *fractional-step*-Advektion



A.4.3 Rechteckrotation, Pilliods Advektion

E_1-Fehler

Dim.	SLIC	H&N	COM	P&Y	LVIRA	ELVIRA
32	2.8640e-01	1.2772e-01	1.3125e-01	1.2493e-01	1.2690e-01	1.2723e-01
64	1.5287e-01	5.1309e-02	5.9937e-02	5.0747e-02	5.1790e-02	5.1598e-02
128	9.7611e-02	1.9484e-02	2.4529e-02	1.9087e-02	1.9744e-02	1.9664e-02
256	6.4730e-02	7.1205e-03	1.1840e-02	7.0161e-03	7.2981e-03	7.2529e-03

E_1-Fehlerrate

Dim.	SLIC	H&N	COM	P&Y	LVIRA	ELVIRA
64/32	0.5338	0.4017	0.4567	0.4062	0.4081	0.4055
128/64	0.6385	0.3797	0.4092	0.3761	0.3812	0.3811
256/128	0.6631	0.3655	0.4827	0.3676	0.3696	0.3688

Gesamter relativer Volumenverlust

Dim.	SLIC	H&N	COM	P&Y	LVIRA	ELVIRA
32	-8.4775e-03	-1.1294e-19	-6.8672e-16	-1.7500e-16	-4.5749e-14	2.1786e-16
64	-5.8305e-03	-4.7434e-18	-7.6628e-17	-2.0865e-16	-1.5469e-14	-4.9241e-16
128	-3.3063e-03	-6.0072e-16	1.1808e-16	-2.3248e-16	-9.6306e-14	5.7937e-17
256	-1.2097e-03	-2.5264e-16	-5.1133e-16	-8.4590e-17	-7.9314e-14	6.9965e-17

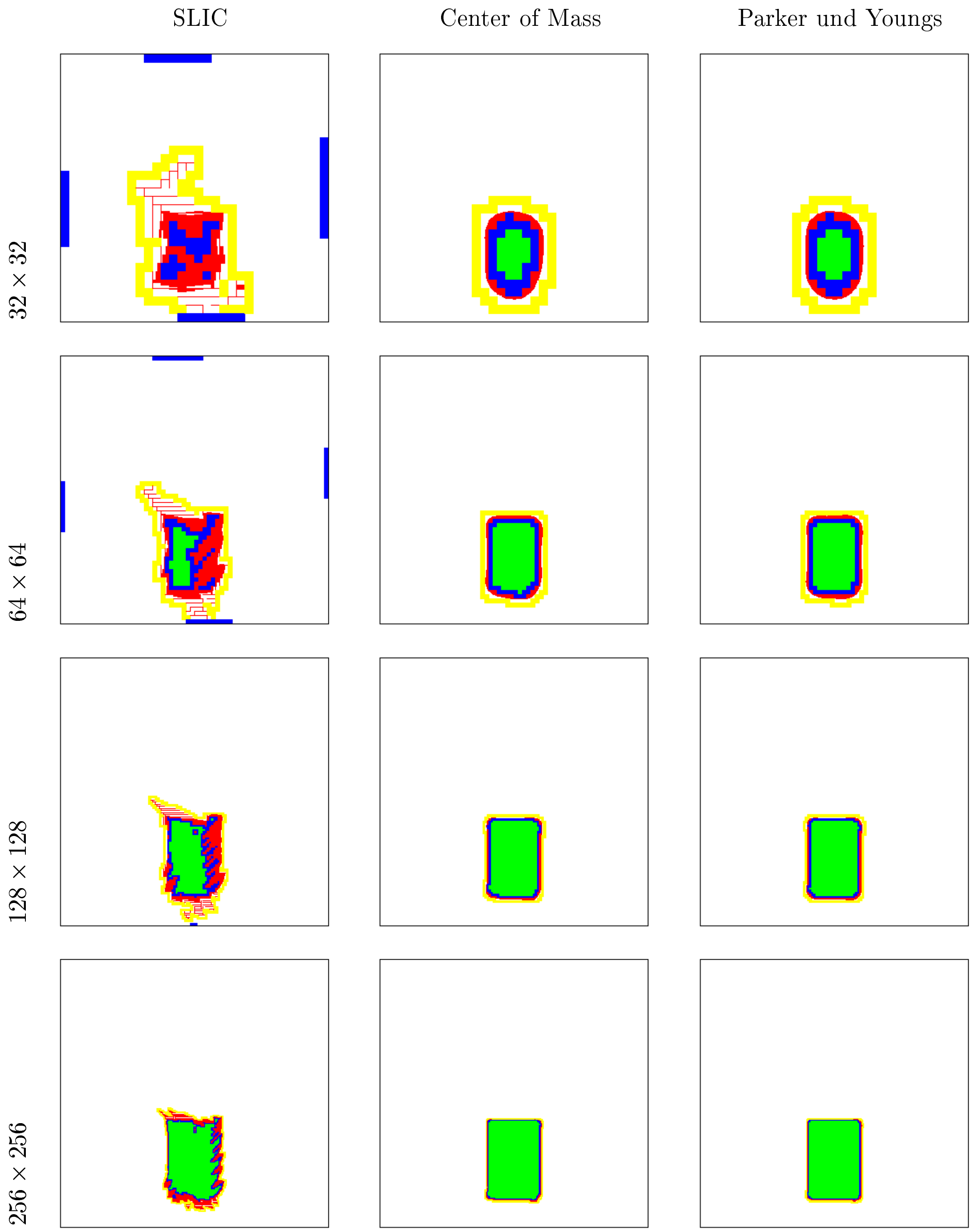
Durchschnittlicher relativer Volumenverlust

Dim.	SLIC	H&N	COM	P&Y	LVIRA	ELVIRA
32	1.0540e-03	1.6721e-16	2.6864e-16	2.7866e-16	4.1385e-14	2.7963e-16
64	2.6262e-03	1.0501e-16	1.8326e-16	1.9095e-16	9.6628e-14	1.7867e-16
128	1.2370e-03	2.9834e-16	2.9321e-16	1.8131e-16	1.0497e-13	1.0167e-16
256	5.6100e-04	9.6285e-17	8.8560e-17	1.7500e-16	1.0566e-13	1.1457e-16

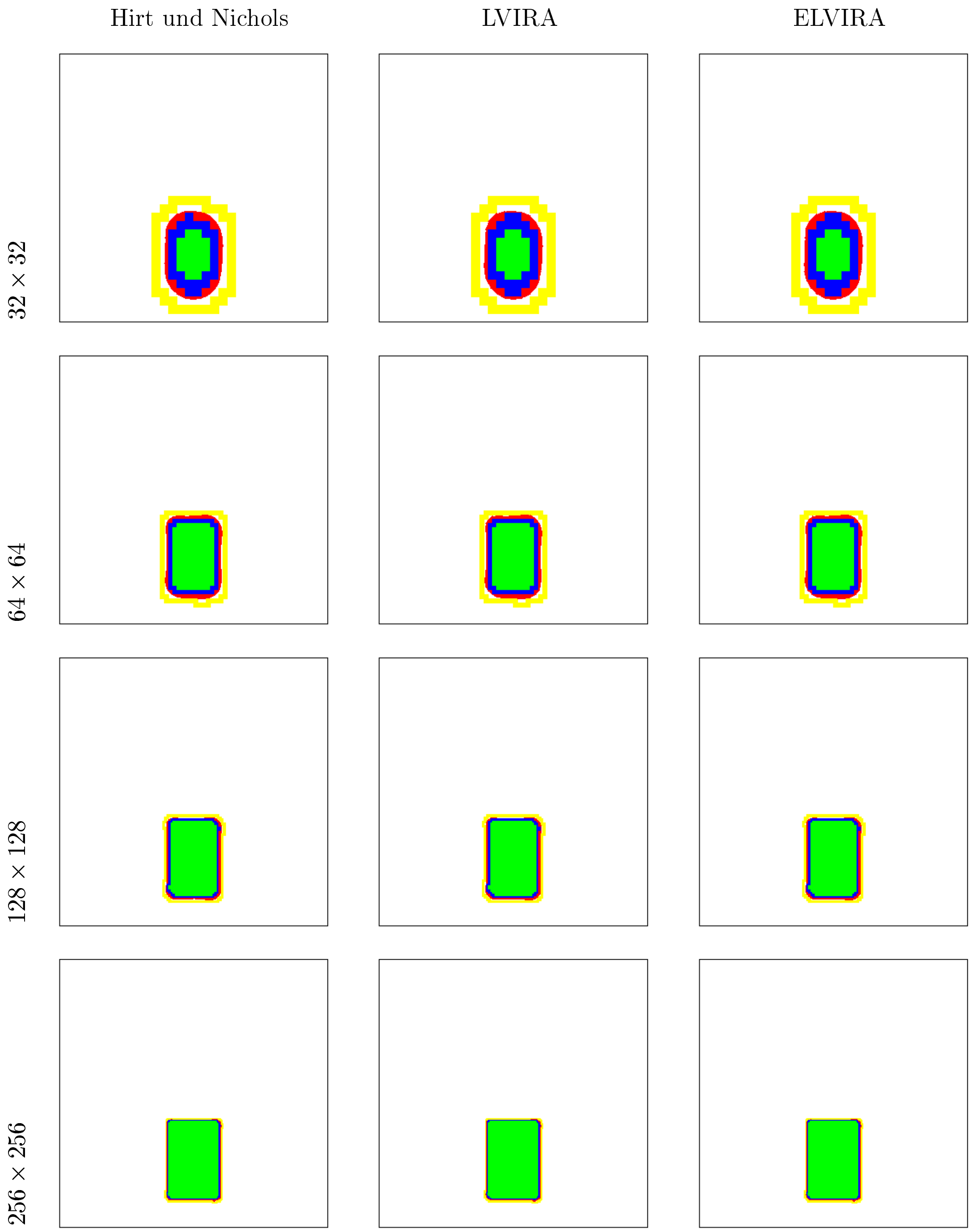
Füllungstoleranz

Dim.	SLIC	H&N	COM	P&Y	LVIRA	ELVIRA
32	3.5527e-15	3.7748e-15	4.2188e-15	3.7748e-15	1.5816e-12	3.5527e-15
64	6.1843e-15	9.1676e-15	9.9920e-15	1.0880e-14	3.0518e-12	1.0065e-14
128	1.2285e-14	2.7311e-14	1.9901e-14	2.0650e-14	8.6814e-12	2.0650e-14
256	2.8866e-14	4.7073e-14	4.2633e-14	5.1830e-14	1.7818e-11	4.5075e-14

Rechteckrotation, Pilliods Advektion



Rechteckrotation, Pilliods Advektion



A.5 Testproblem Kreisrotation

A.5.1 Kreisrotation, *donor-acceptor*-Advektion

E_1-Fehler

Dim.	SLIC	H&N	ELVIRA
32	1.2473e-01	1.2901e-01	1.2927e-01
64	5.8512e-02	5.9640e-02	5.9990e-02
128	3.1892e-02	3.2893e-02	3.2843e-02
256	1.8038e-02	1.8871e-02	1.8325e-02

E_1-Fehlerrate

Dim.	SLIC	H&N	ELVIRA
64/32	0.4691	0.4623	0.4641
128/64	0.5451	0.5515	0.5475
256/128	0.5656	0.5737	0.5580

Gesamter relativer Volumenverlust

Dim.	SLIC	H&N	ELVIRA
32	-1.5689e-10	-1.2663e-10	-2.9035e-08
64	-1.2207e-09	-1.3672e-09	-5.5825e-08
128	-2.0143e-09	-1.9903e-09	-9.7834e-08
256	-3.7268e-09	-3.4952e-09	-1.8863e-07

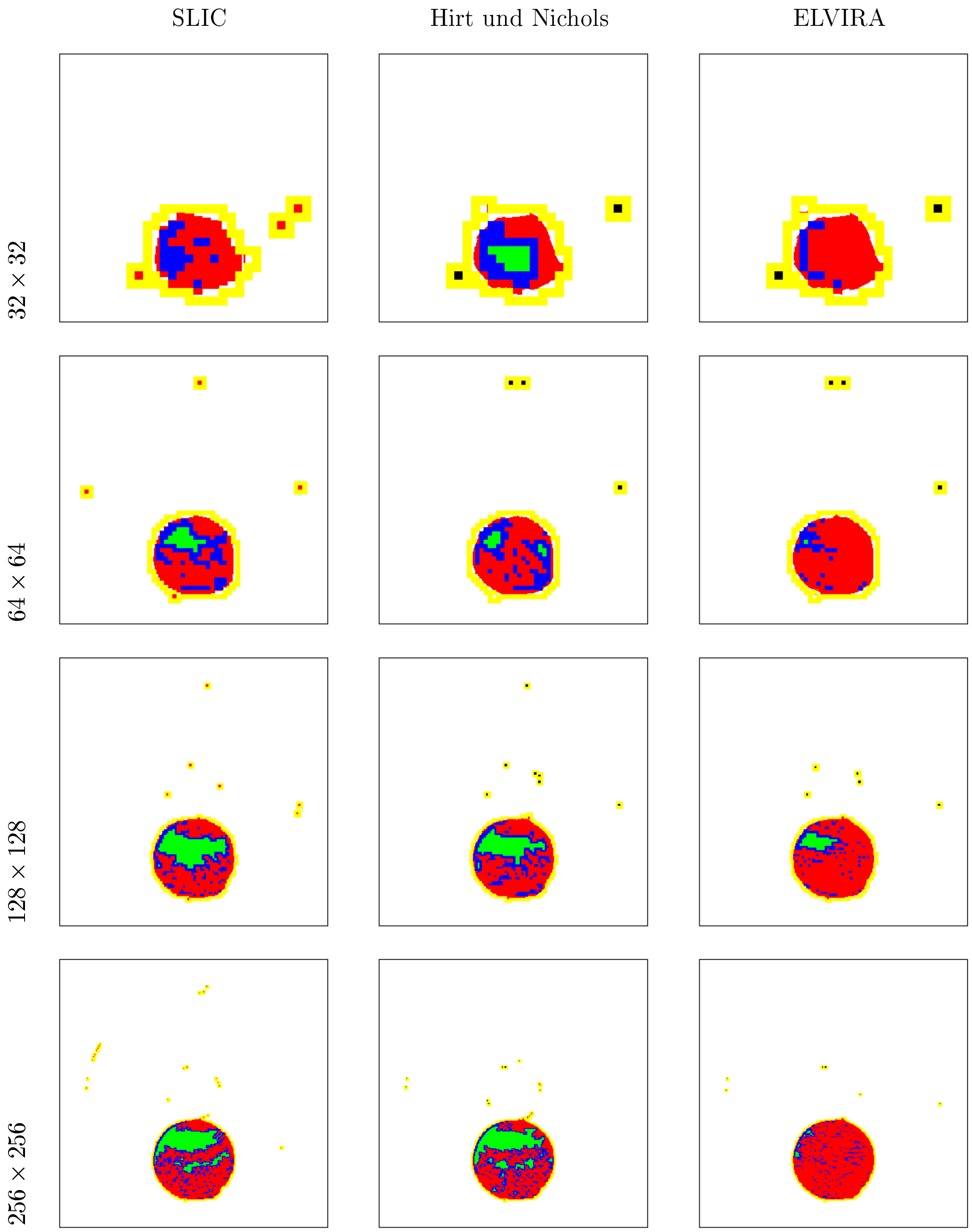
Durchschnittlicher relativer Volumenverlust

Dim.	SLIC	H&N	ELVIRA
32	3.0512e-11	4.1644e-11	1.3563e-08
64	5.4984e-10	6.0483e-10	2.6403e-08
128	9.1177e-10	8.8491e-10	4.4625e-08
256	1.4610e-09	1.3821e-09	8.0203e-08

Füllungstoleranz

Dim.	SLIC	H&N	ELVIRA
32	2.3283e-10	2.3283e-10	1.0537e-08
64	2.3283e-10	2.3283e-10	1.0537e-08
128	2.3283e-10	2.3283e-10	1.0537e-08
256	2.3283e-10	2.3283e-10	1.0537e-08

Kreisrotation, *donor-acceptor*-Advektion



A.5.2 Kreisrotation, *fractional-step*-Advektion

E_1-Fehler

Dim.	SLIC	H&N	COM	P&Y	LVIRA	ELVIRA
32	3.1629e-01	1.7665e-02	3.2135e-02	1.6642e-02	2.2966e-02	2.1100e-02
64	2.8413e-01	4.4874e-03	2.1768e-02	5.6598e-03	5.0678e-03	4.7590e-03
128	2.6215e-01	1.1078e-03	1.3075e-02	2.7634e-03	1.2915e-03	1.1497e-03
256	2.9992e-01	2.8612e-04	6.7218e-03	1.4851e-03	3.2374e-04	2.7408e-04

E_1-Fehlerrate

Dim.	SLIC	H&N	COM	P&Y	LVIRA	ELVIRA
64/32	0.8983	0.2540	0.6774	0.3401	0.2207	0.2255
128/64	0.9226	0.2469	0.6007	0.4883	0.2548	0.2416
256/128	1.1441	0.2583	0.5141	0.5374	0.2507	0.2384

Gesamter relativer Volumenverlust

Dim.	SLIC	H&N	COM	P&Y	LVIRA	ELVIRA
32	-1.0469e-01	1.8252e-14	1.8369e-14	1.8358e-14	2.7388e-14	1.8225e-14
64	-1.7905e-01	-2.3417e-15	-2.4601e-15	-1.6542e-15	-8.4276e-14	-2.1761e-15
128	-2.0549e-01	7.3038e-15	6.5184e-15	6.7013e-15	-7.4766e-14	6.6096e-15
256	-2.7058e-01	-2.6834e-15	-2.8930e-15	-2.4781e-15	-2.7030e-14	-1.9573e-15

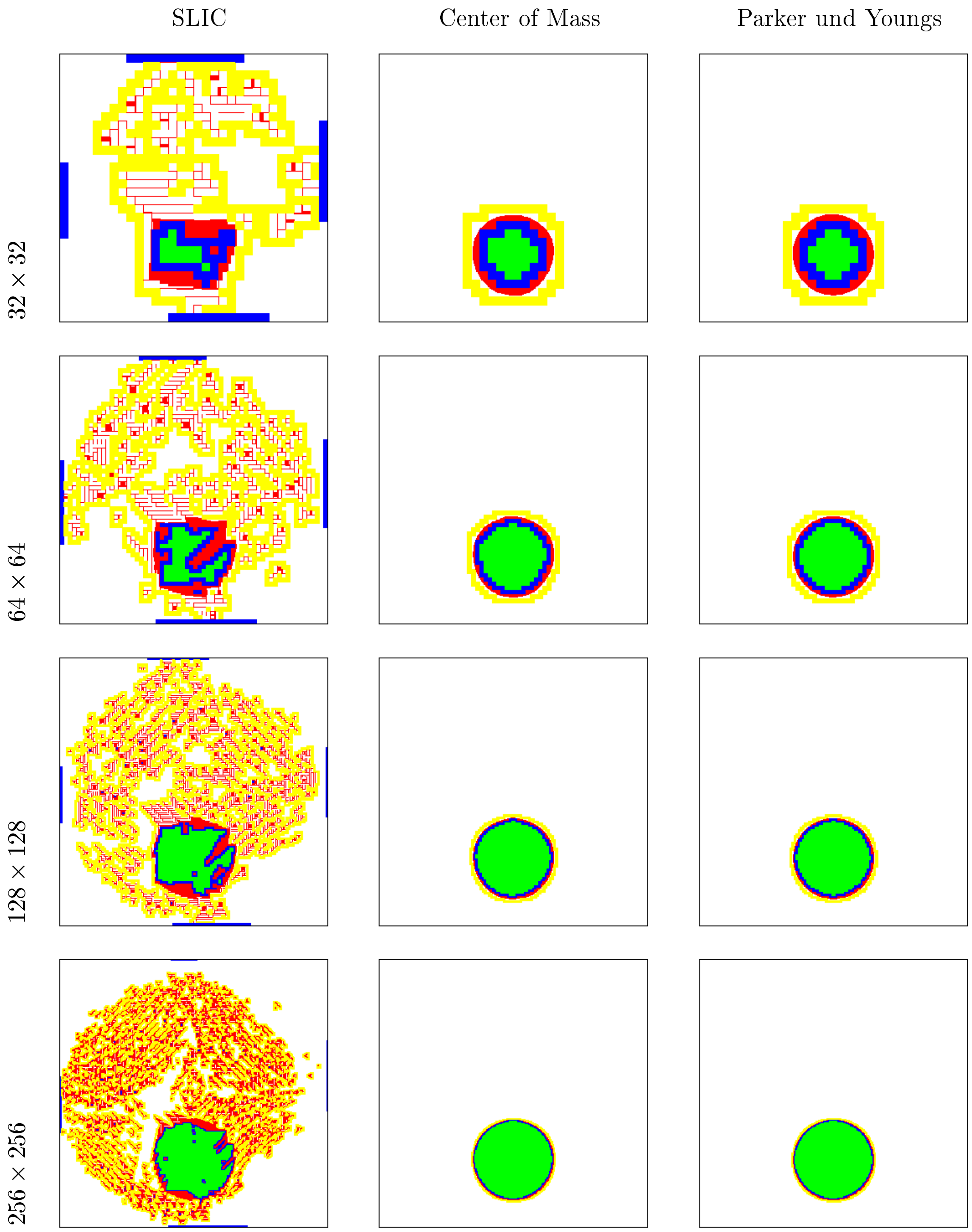
Durchschnittlicher relativer Volumenverlust

Dim.	SLIC	H&N	COM	P&Y	LVIRA	ELVIRA
32	2.3980e-02	8.1315e-15	8.5183e-15	8.1815e-15	3.9013e-14	8.1089e-15
64	5.2965e-02	9.5891e-16	1.2187e-15	5.7143e-16	5.0146e-14	8.7698e-16
128	6.1070e-02	2.1041e-15	2.0427e-15	1.9659e-15	2.7507e-14	2.0961e-15
256	7.4295e-02	8.4862e-16	9.9608e-16	8.7121e-16	4.5256e-14	6.2834e-16

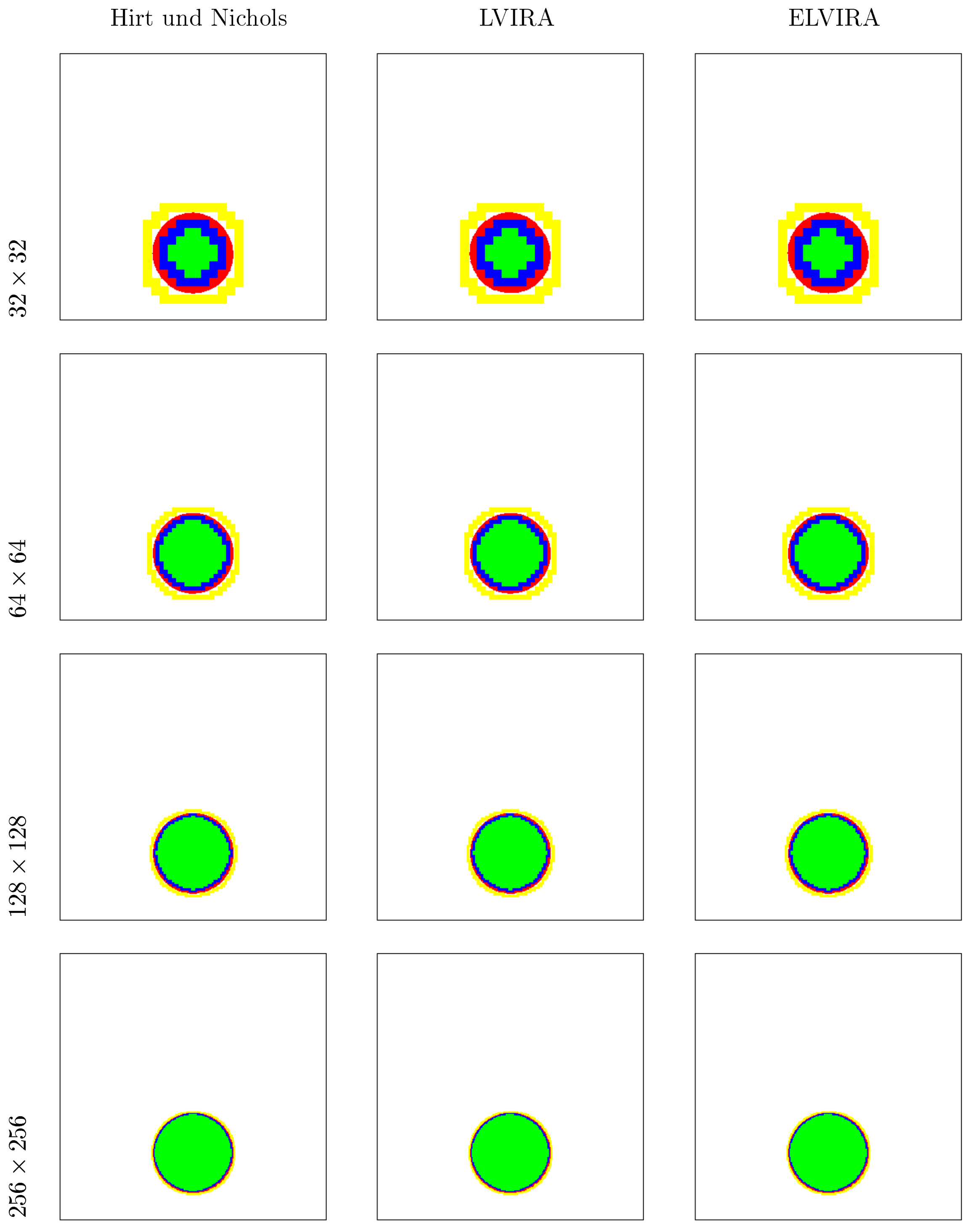
Füllungstoleranz

Dim.	SLIC	H&N	COM	P&Y	LVIRA	ELVIRA
32	3.5527e-15	3.5527e-15	4.5519e-15	3.5527e-15	1.7140e-12	3.5527e-15
64	7.1054e-15	8.4377e-15	1.1546e-14	9.8255e-15	3.6222e-12	9.9920e-15
128	1.4211e-14	2.3315e-14	2.6201e-14	2.4869e-14	7.9531e-12	2.1982e-14
256	2.7978e-14	5.1514e-14	5.6177e-14	5.5955e-14	1.4988e-11	5.2403e-14

Kreisrotation, *fractional-step*-Advektion



Kreisrotation, *fractional-step*-Advektion



A.5.3 Kreisrotation, Pilliods Advektion

E_1-Fehler

Dim.	SLIC	H&N	COM	P&Y	LVIRA	ELVIRA
32	2.4990e-01	2.0440e-02	3.0035e-02	2.0025e-02	2.3904e-02	2.2472e-02
64	1.2806e-01	5.1252e-03	1.9448e-02	5.5939e-03	5.5864e-03	5.2282e-03
128	7.5283e-02	1.2692e-03	1.1874e-02	2.5769e-03	1.3975e-03	1.2888e-03
256	4.4943e-02	3.2948e-04	6.0765e-03	1.3589e-03	3.6027e-04	3.2037e-04

E_1-Fehlerrate

Dim.	SLIC	H&N	COM	P&Y	LVIRA	ELVIRA
64/32	0.5124	0.2507	0.6475	0.2793	0.2337	0.2327
128/64	0.5879	0.2476	0.6106	0.4607	0.2502	0.2465
256/128	0.5970	0.2596	0.5117	0.5273	0.2578	0.2486

Gesamter relativer Volumenverlust

Dim.	SLIC	H&N	COM	P&Y	LVIRA	ELVIRA
32	-7.1960e-03	2.6353e-16	9.1167e-17	3.9458e-16	1.2295e-14	-2.2845e-16
64	-5.0047e-03	-1.5396e-16	7.5925e-17	-1.1139e-16	1.8444e-13	1.7706e-16
128	-5.1702e-03	-4.9370e-17	1.5952e-16	-6.3079e-17	2.0223e-13	-1.7831e-16
256	-2.2197e-03	1.8885e-16	-1.8684e-16	1.1398e-16	1.9699e-13	3.4454e-16

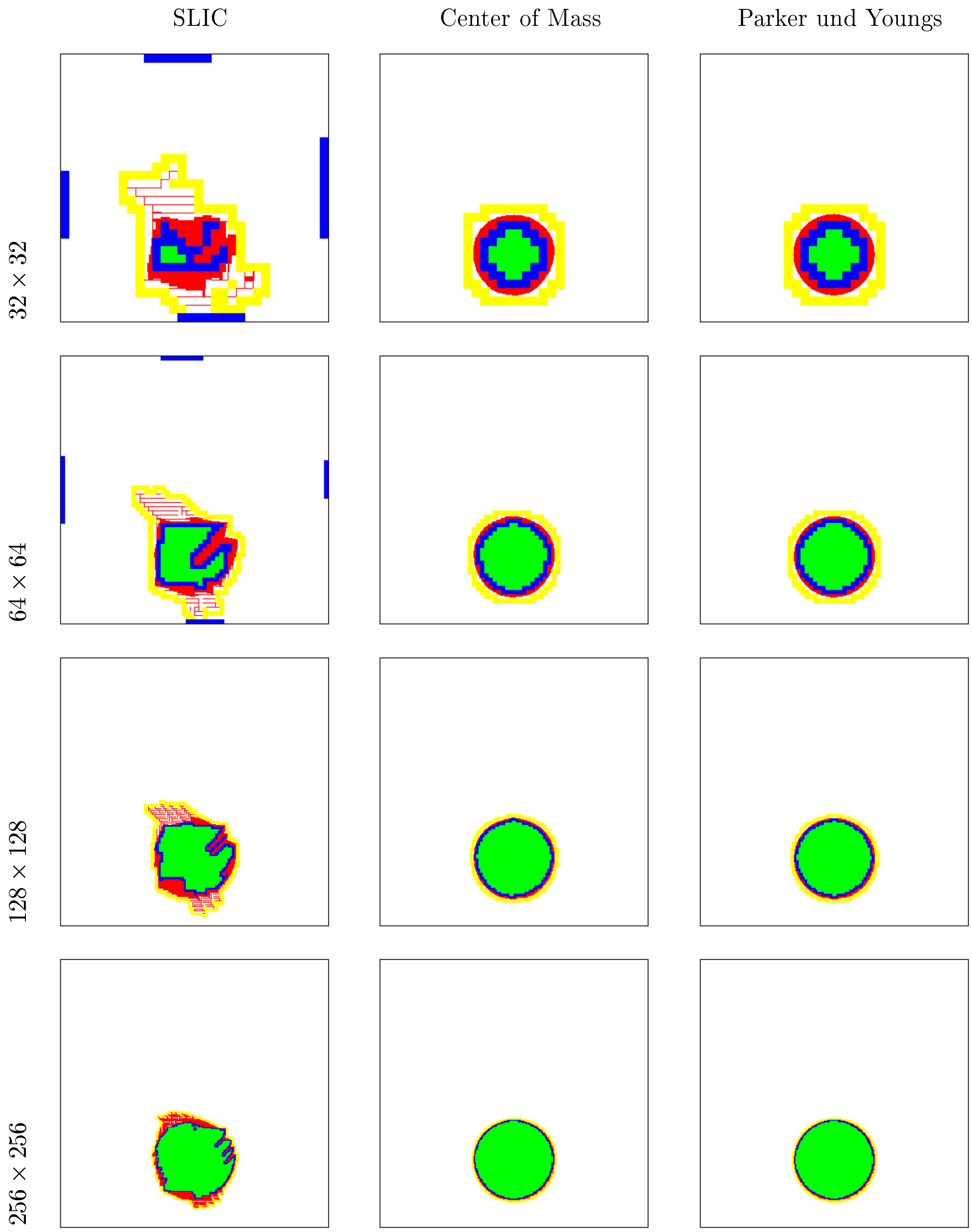
Durchschnittlicher relativer Volumenverlust

Dim.	SLIC	H&N	COM	P&Y	LVIRA	ELVIRA
32	7.7794e-04	1.9934e-16	2.0818e-16	3.8557e-16	3.2108e-14	1.7605e-16
64	3.0461e-03	1.8591e-16	4.5063e-17	1.0333e-16	3.6639e-14	8.9890e-17
128	2.2414e-03	5.3283e-17	9.3652e-17	6.8011e-17	4.6871e-14	1.1204e-16
256	1.0993e-03	9.8926e-17	2.1648e-16	1.1618e-16	5.6376e-14	1.8643e-16

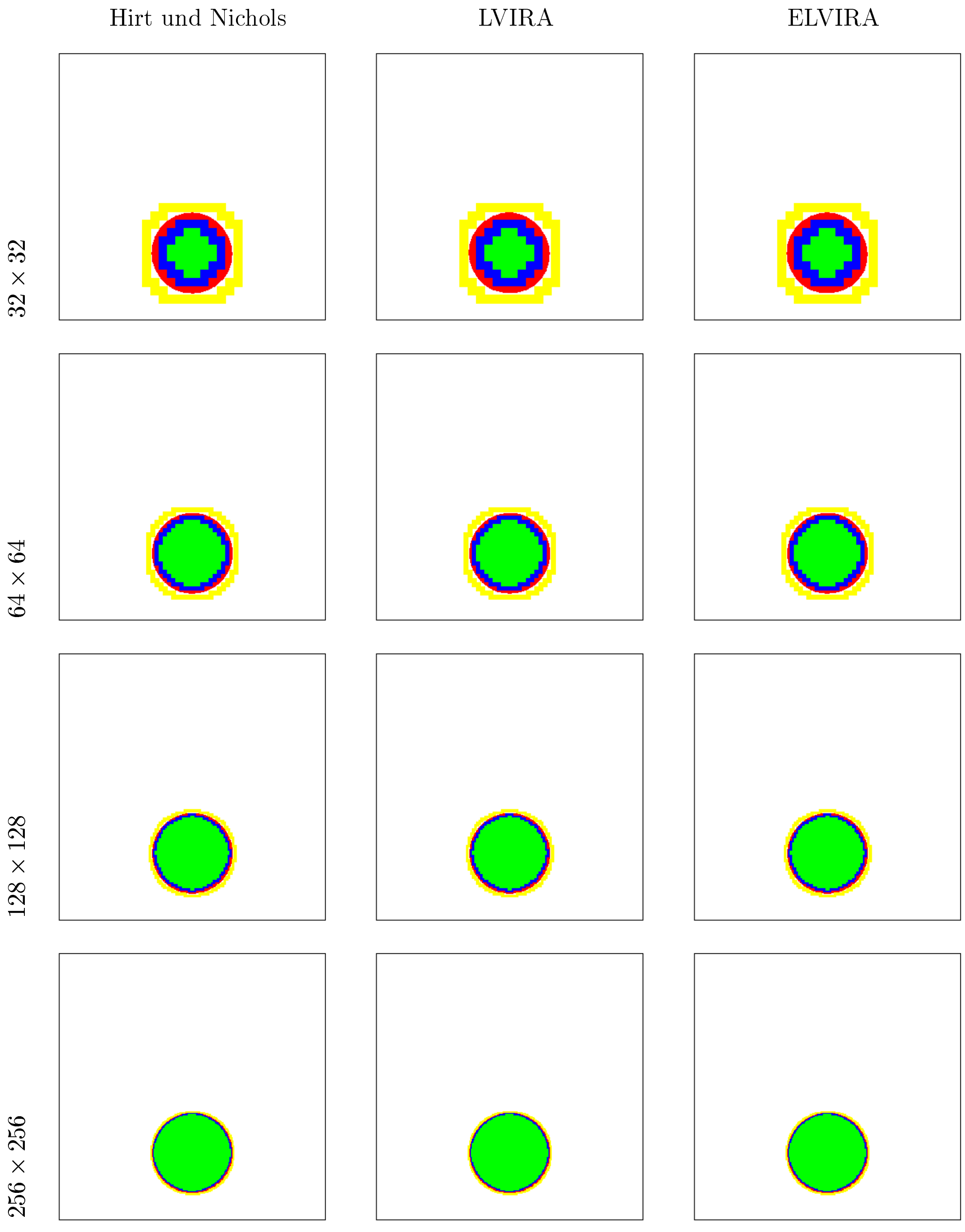
Füllungstoleranz

Dim.	SLIC	H&N	COM	P&Y	LVIRA	ELVIRA
32	3.5527e-15	3.5527e-15	4.6629e-15	3.5527e-15	1.5599e-12	3.5527e-15
64	5.9952e-15	1.0214e-14	9.4690e-15	8.4377e-15	3.1169e-12	8.9648e-15
128	1.3810e-14	1.9206e-14	2.2523e-14	2.0872e-14	6.5703e-12	2.2649e-14
256	2.7828e-14	4.5741e-14	4.7632e-14	4.5843e-14	1.4331e-11	5.2847e-14

Kreisrotation, Pilliods Advektion



Kreisrotation, Pilliods Advektion



A.6 Testproblem Lochkreisrotation

A.6.1 Lochkreisrotation, *fractional-step*-Advektion

E_1-Fehler

Dim.	SLIC	H&N	COM	P&Y	LVIRA	ELVIRA
32	9.7209e-02	1.8017e-02	2.2721e-02	1.8342e-02	1.8461e-02	1.8389e-02
64	6.1191e-02	8.3692e-03	1.2132e-02	8.7552e-03	8.5360e-03	8.4392e-03
128	3.7851e-02	3.1846e-03	5.9281e-03	3.4866e-03	3.2739e-03	3.1991e-03
256	3.3092e-02	1.2204e-03	3.0073e-03	1.4081e-03	1.2593e-03	1.2354e-03

E_1-Fehlerrate

Dim.	SLIC	H&N	COM	P&Y	LVIRA	ELVIRA
64/32	0.6295	0.4645	0.5340	0.4773	0.4624	0.4589
128/64	0.6186	0.3805	0.4886	0.3982	0.3835	0.3791
256/128	0.8743	0.3832	0.5073	0.4039	0.3846	0.3862

Gesamter relativer Volumenverlust

Dim.	SLIC	H&N	COM	P&Y	LVIRA	ELVIRA
32	8.7975e-04	-2.4230e-15	-3.0066e-15	-2.5162e-15	-7.6905e-15	-2.5037e-15
64	7.1492e-03	-5.4652e-15	-5.9756e-15	-5.4896e-15	2.4641e-15	-5.4187e-15
128	6.6216e-03	-6.4350e-17	3.4917e-16	-7.8110e-17	1.5182e-14	-7.5420e-17
256	1.0002e-02	-6.3336e-15	-5.3578e-15	-5.8888e-15	4.3678e-15	-6.3353e-15

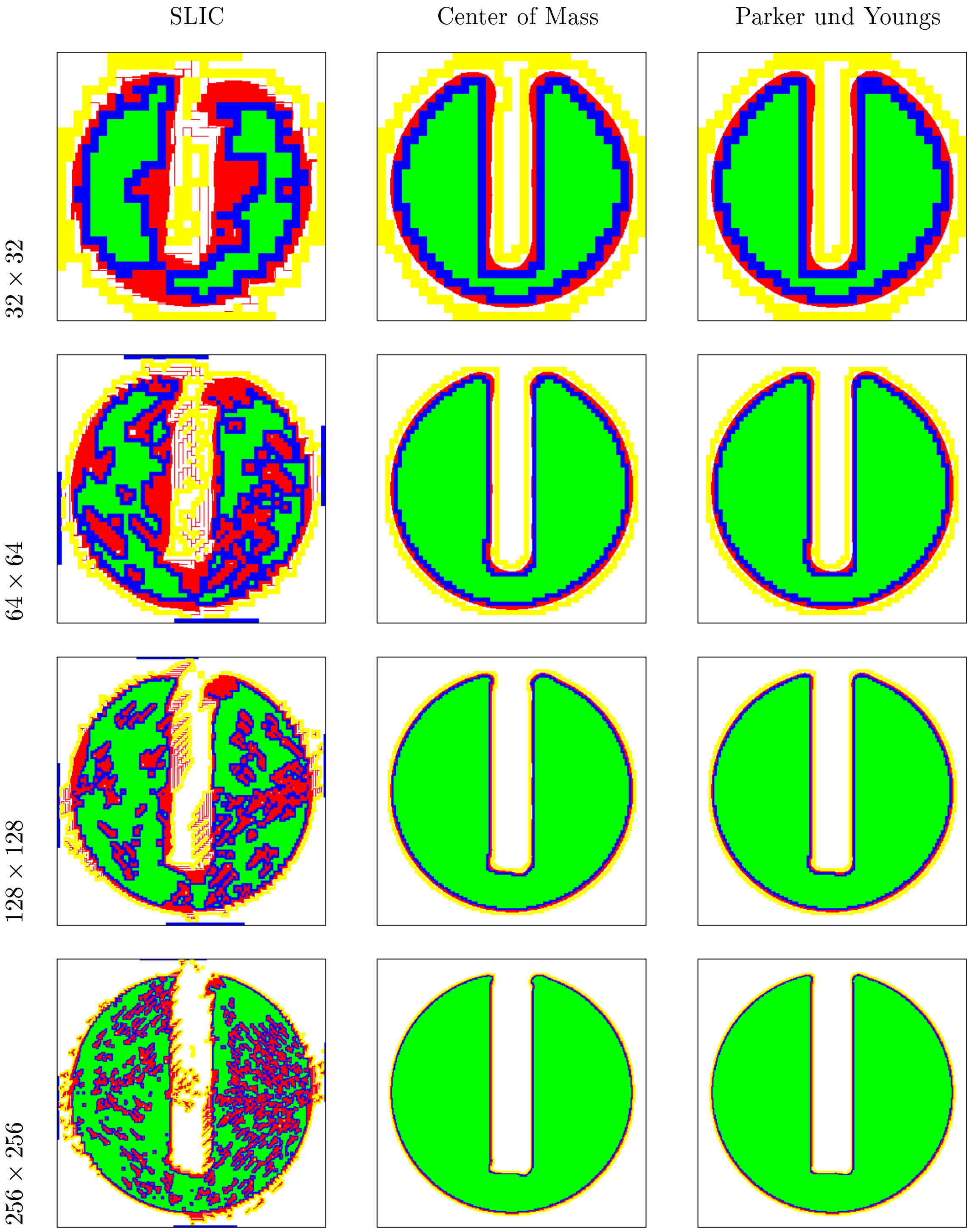
Durchschnittlicher relativer Volumenverlust

Dim.	SLIC	H&N	COM	P&Y	LVIRA	ELVIRA
32	6.2933e-04	1.2218e-15	1.5383e-15	1.2742e-15	8.3605e-15	1.2595e-15
64	2.7899e-03	2.6373e-15	2.9169e-15	2.6537e-15	1.0507e-14	2.6127e-15
128	2.0231e-03	4.9256e-17	1.6843e-16	6.4687e-17	7.4990e-15	5.3555e-17
256	2.5165e-03	3.0878e-15	2.5878e-15	2.8642e-15	3.1215e-15	3.0975e-15

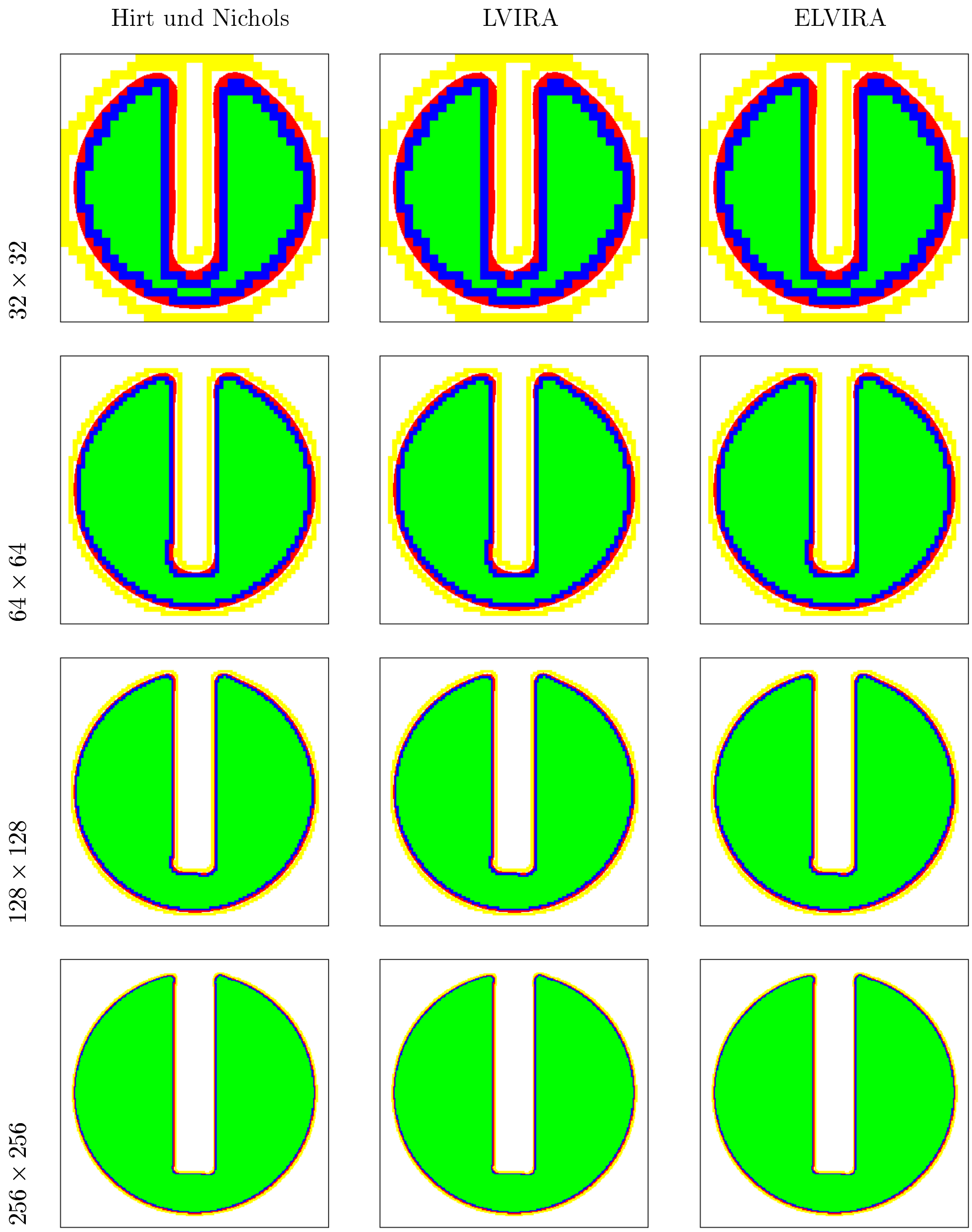
Füllungstoleranz

Dim.	SLIC	H&N	COM	P&Y	LVIRA	ELVIRA
32	3.5527e-15	3.5527e-15	3.5527e-15	3.5527e-15	8.1551e-13	3.5527e-15
64	3.5527e-15	3.5527e-15	5.1070e-15	4.2813e-15	1.8293e-12	4.8850e-15
128	4.9738e-13	1.0880e-14	9.1038e-15	9.7700e-15	3.8467e-12	8.4377e-15
256	7.5033e-12	2.4425e-14	2.3981e-14	2.3315e-14	7.7805e-12	1.9984e-14

Lochkreisrotation, *fractional-step*-Advektion



Lochkreisrotation, *fractional-step*-Advektion



A.6.2 Lochkreisrotation, Pilliods Advektion

E_1-Fehler

Dim.	SLIC	H&N	COM	P&Y	LVIRA	ELVIRA
32	1.0522e-01	1.7661e-02	2.1370e-02	1.7587e-02	1.8086e-02	1.7920e-02
64	5.4992e-02	8.1110e-03	1.1422e-02	8.4141e-03	8.2197e-03	8.0812e-03
128	3.5109e-02	3.0298e-03	5.4850e-03	3.2942e-03	3.1234e-03	3.0565e-03
256	1.9302e-02	1.1616e-03	2.8008e-03	1.3343e-03	1.2035e-03	1.1711e-03

E_1-Fehlerrate

Dim.	SLIC	H&N	COM	P&Y	LVIRA	ELVIRA
64/32	0.5226	0.4593	0.5345	0.4784	0.4545	0.4510
128/64	0.6384	0.3735	0.4802	0.3915	0.3800	0.3782
256/128	0.5498	0.3834	0.5106	0.4050	0.3853	0.3832

Gesamter relativer Volumenverlust

Dim.	SLIC	H&N	COM	P&Y	LVIRA	ELVIRA
32	3.0903e-16	3.0861e-16	1.5446e-16	2.9620e-16	-3.1593e-14	2.3754e-16
64	-2.9802e-05	1.4832e-15	2.1188e-15	1.4832e-15	-3.8259e-12	2.1188e-15
128	7.7340e-05	-3.3901e-15	-3.3901e-15	-2.5426e-15	1.8425e-12	-3.3901e-15
256	-8.2024e-05	4.2055e-16	1.5384e-16	3.4410e-16	-2.4469e-14	4.2717e-16

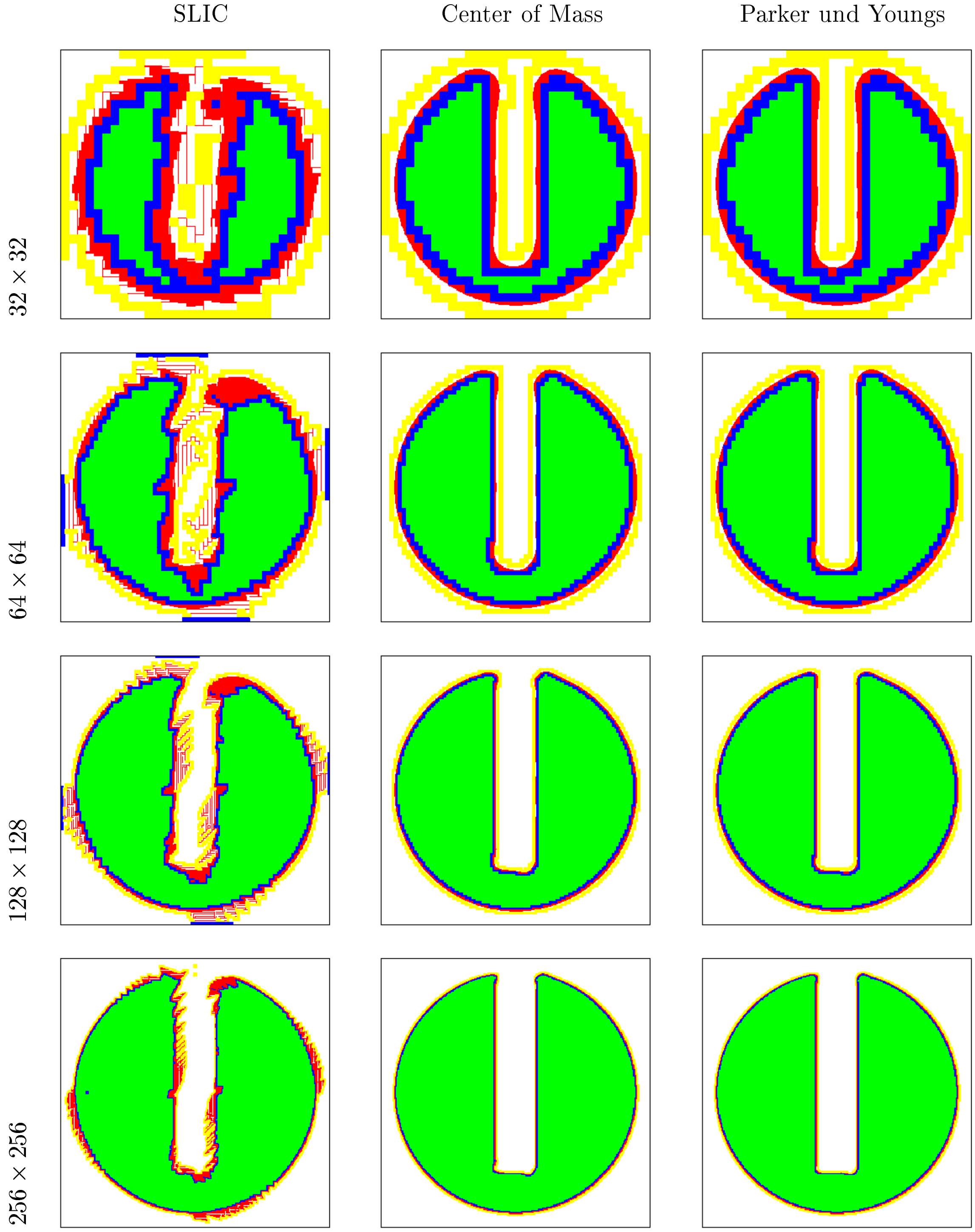
Durchschnittlicher relativer Volumenverlust

Dim.	SLIC	H&N	COM	P&Y	LVIRA	ELVIRA
32	1.4576e-16	1.4299e-16	5.7626e-17	1.2728e-16	2.0792e-14	1.2054e-16
64	5.6800e-05	1.9742e-15	2.0215e-15	2.0107e-15	4.0186e-12	1.9990e-15
128	2.9130e-05	3.5590e-15	3.5457e-15	3.5793e-15	1.7949e-12	3.5895e-15
256	4.9114e-05	2.6760e-16	1.2924e-16	2.1955e-16	1.2305e-14	2.6518e-16

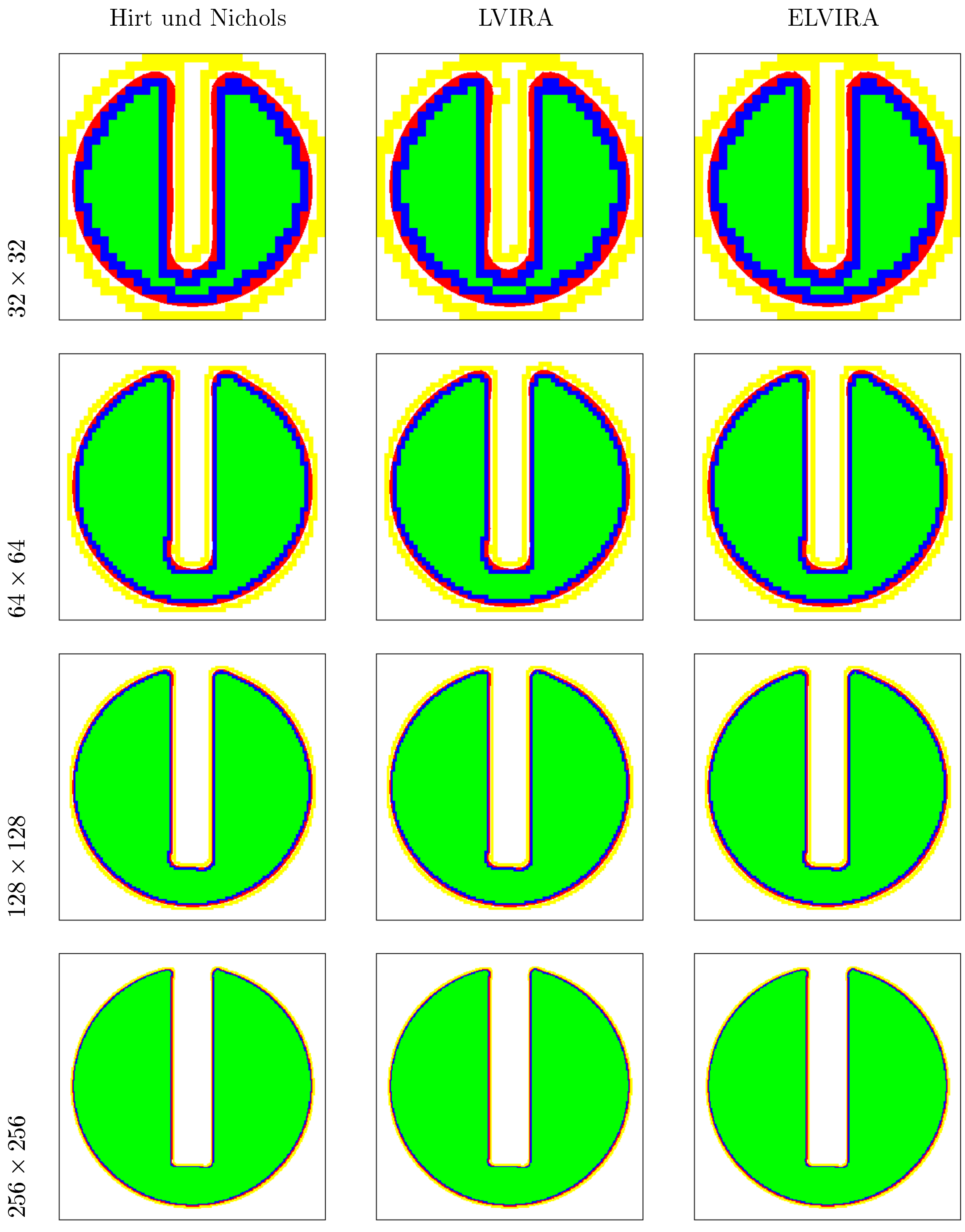
Füllungstoleranz

Dim.	SLIC	H&N	COM	P&Y	LVIRA	ELVIRA
32	3.5527e-15	3.5527e-15	3.5527e-15	3.5527e-15	8.3494e-13	3.5527e-15
64	1.0816e-12	1.0816e-12	1.0816e-12	1.0816e-12	5.5597e-10	1.0816e-12
128	1.0816e-12	1.0816e-12	1.0816e-12	1.0816e-12	1.1679e-09	1.0816e-12
256	1.4022e-14	1.9762e-14	2.2204e-14	2.2649e-14	7.8217e-12	2.0890e-14

Lochkreisrotation, Pilliods Advektion



Lochkreisrotation, Pilliods Advektion

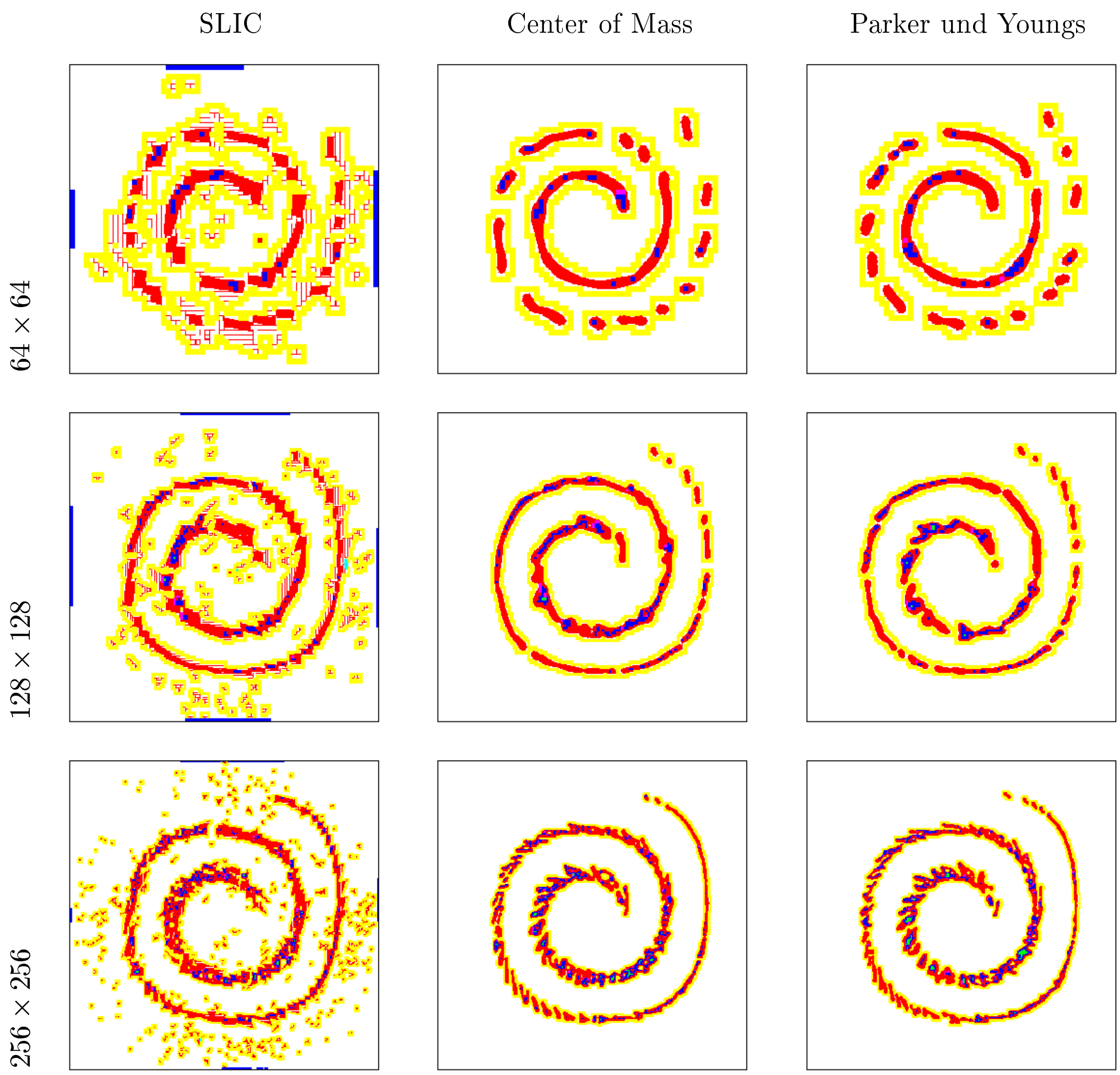


A.7 Testproblem Spirale

A.7.1 Spirale, *fractional-step*-Advektion

Gesamter relativer Volumenverlust

Dim.	SLIC	H&N	COM	P&Y	LVIRA	ELVIRA
64	9.7343e-03	1.1876e-02	2.6421e-02	2.3222e-02	1.0923e-02	1.5870e-02
128	1.7514e-02	3.0931e-02	3.4108e-02	5.4538e-02	3.2570e-02	2.7184e-02
256	-6.2422e-03	2.0536e-02	1.1054e-02	3.1038e-02	4.3703e-02	5.1785e-03



Spirale, *fractional-step*-Advektion

Durchschnittlicher relativer Volumenverlust

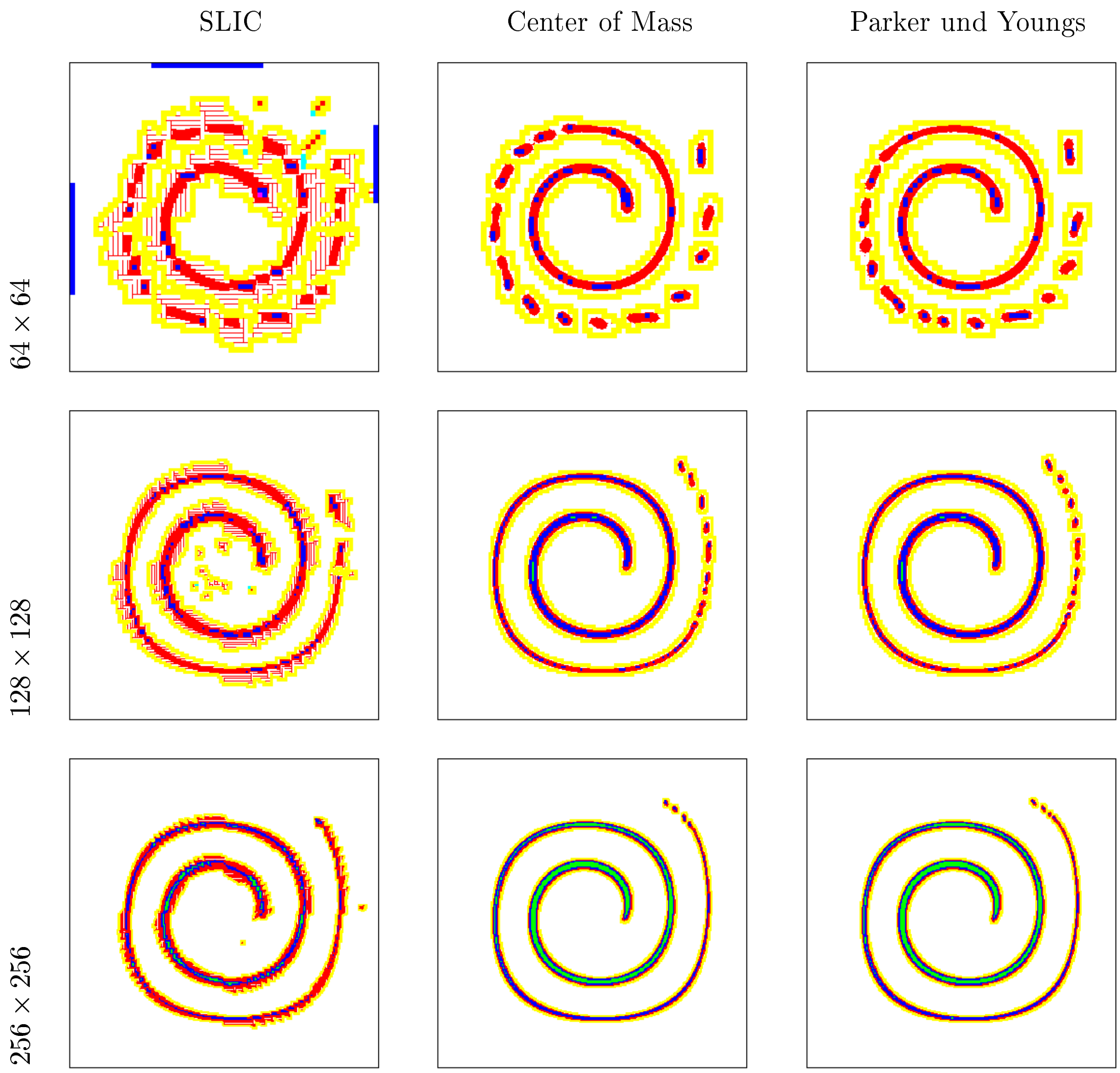
Dim.	SLIC	H&N	COM	P&Y	LVIRA	ELVIRA
64	1.3775e-02	1.5026e-02	2.5559e-02	2.3512e-02	1.0847e-02	1.6049e-02
128	3.4743e-02	4.7638e-02	5.3340e-02	6.2779e-02	4.4820e-02	4.6778e-02
256	3.6038e-02	6.1602e-02	5.8458e-02	6.4724e-02	6.3900e-02	5.8520e-02

Füllungstoleranz fixiert auf 2.3283e-10.

A.7.2 Spirale, Pilliods Advektion

Gesamter relativer Volumenverlust

Dim.	SLIC	H&N	COM	P&Y	LVIRA	ELVIRA
64	-9.7137e-04	-1.2914e-08	-2.7782e-08	-2.1295e-08	-3.0936e-07	3.2693e-13
128	-5.8708e-04	2.4350e-17	3.6333e-16	-4.7932e-19	-5.8057e-09	-1.1341e-16
256	-1.7017e-04	-4.4980e-16	-3.3754e-16	-4.1308e-16	-4.2861e-16	-4.4146e-16

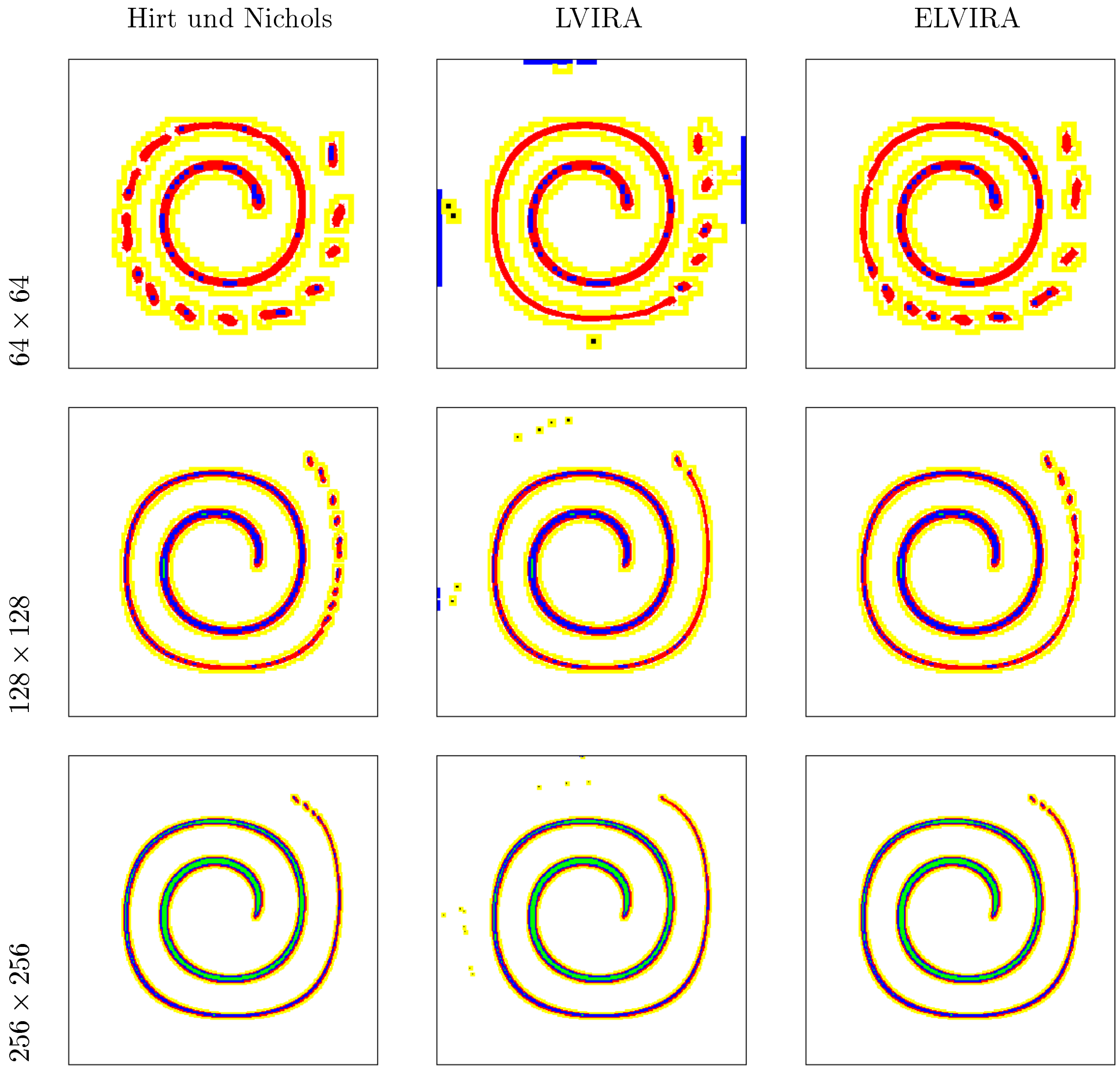


Spirale, Pilliods Advektion

Durchschnittlicher relativer Volumenverlust

Dim.	SLIC	H&N	COM	P&Y	LVIRA	ELVIRA
64	1.2571e-04	1.3316e-09	2.2558e-08	1.9978e-08	2.8073e-07	3.0654e-13
128	1.6817e-04	1.1386e-16	4.3410e-16	1.1005e-16	5.4690e-09	1.0855e-16
256	6.2199e-05	1.8916e-16	1.6366e-16	1.7811e-16	1.6680e-16	1.8818e-16

Füllungstoleranz fixiert auf 2.3283e-10.

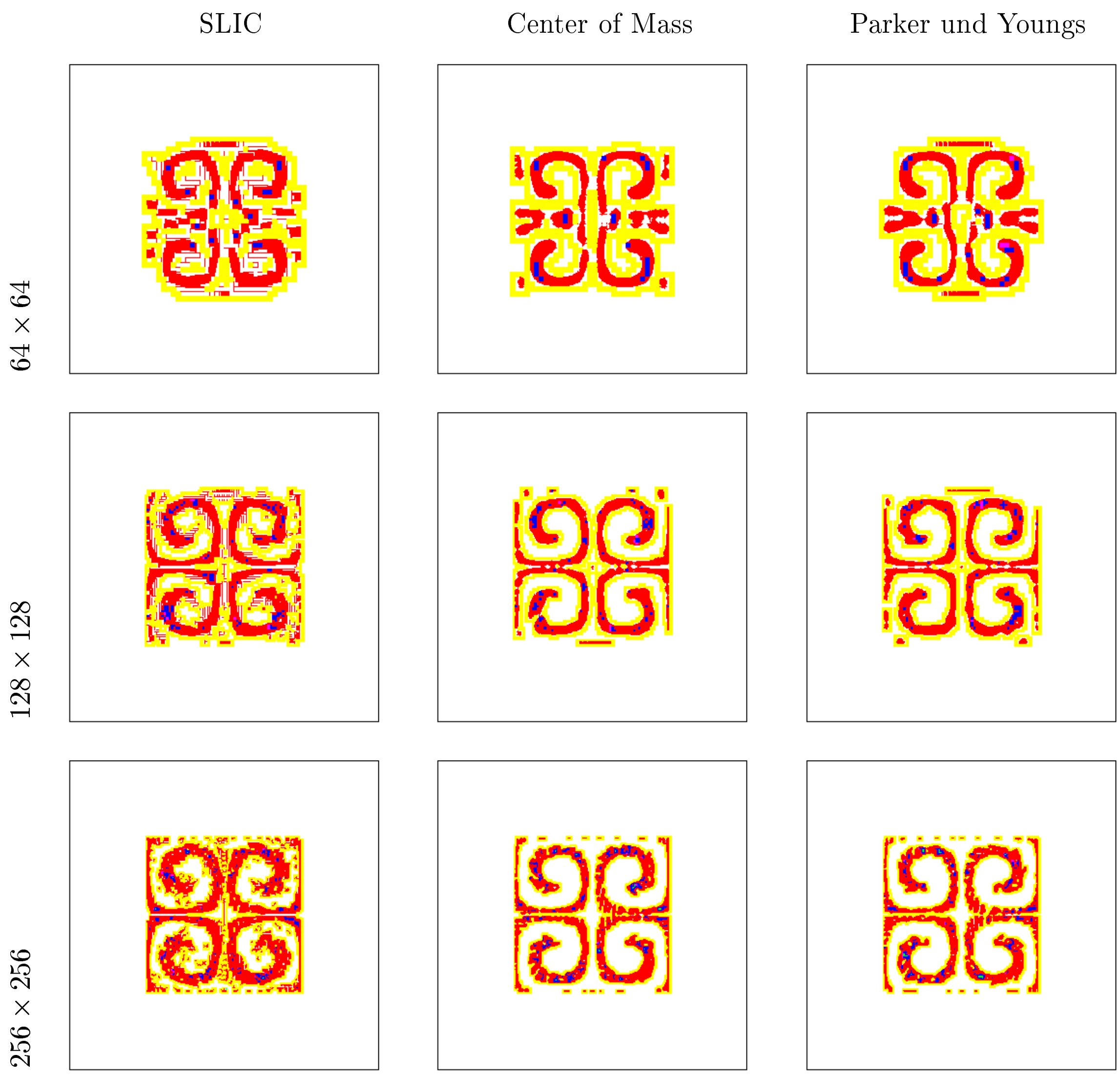


A.8 Testproblem Deformation

A.8.1 Deformation, *fractional-step*-Advektion

Gesamter relativer Volumenverlust

Dim.	SLIC	H&N	COM	P&Y	LVIRA	ELVIRA
64	2.6881e-02	2.7196e-02	9.9790e-03	5.3734e-03	-2.4239e-02	1.7445e-02
128	-2.0108e-02	-1.5670e-02	-2.7973e-03	-1.3532e-02	-7.4092e-02	-1.1896e-02
256	-5.7078e-02	8.8020e-03	8.1629e-03	-2.3234e-03	-4.1913e-02	3.4501e-03



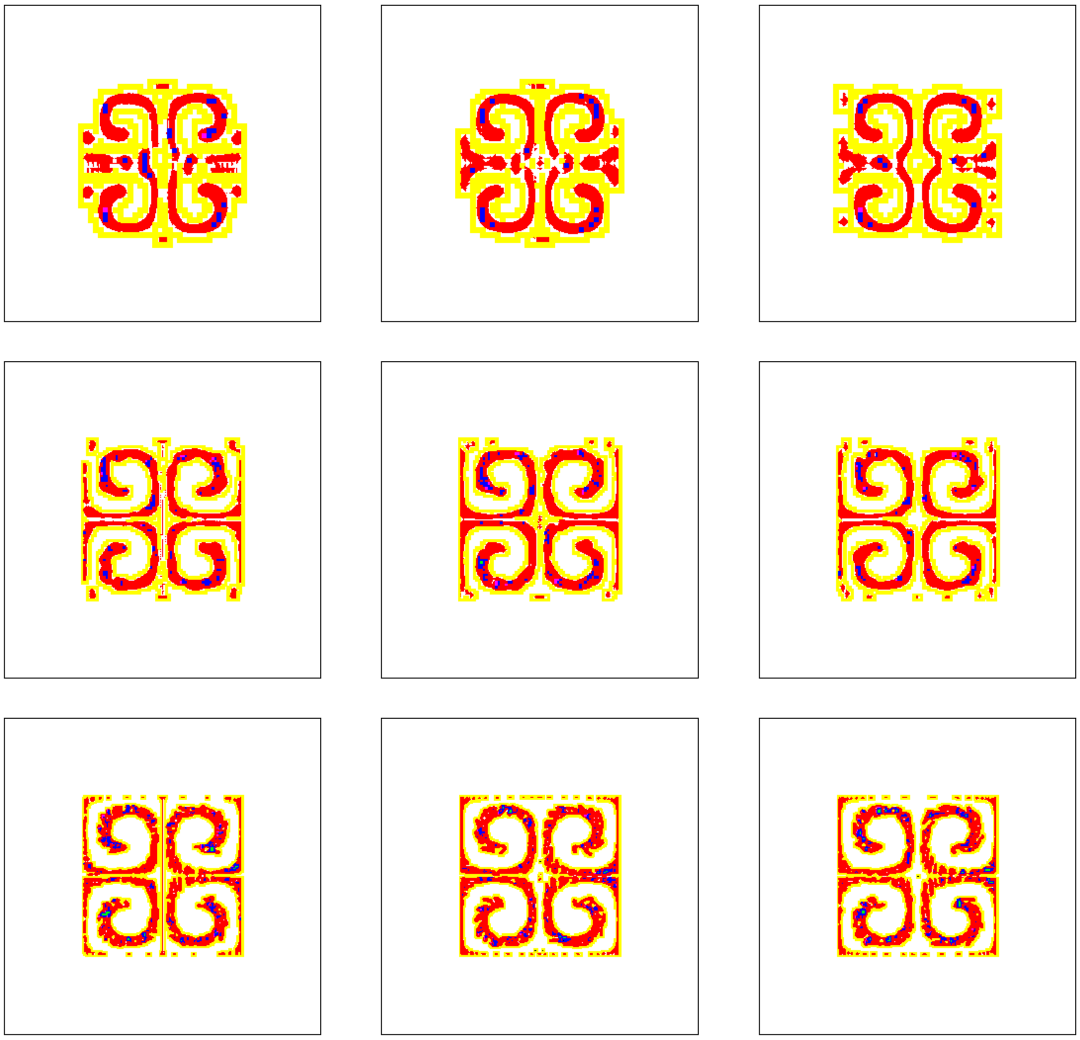
Deformation, *fractional-step*-Advektion

Durchschnittlicher relativer Volumenverlust

Dim.	SLIC	H&N	COM	P&Y	LVIRA	ELVIRA
64	2.2388e-02	2.1669e-02	8.7934e-03	6.0356e-03	1.9767e-02	1.5853e-02
128	8.9839e-03	8.0792e-03	9.1896e-03	7.9457e-03	4.0975e-02	6.3201e-03
256	2.3444e-02	2.1277e-02	2.0734e-02	1.5457e-02	1.6988e-02	1.7960e-02

Füllungstoleranz fixiert auf 2.3283e-10

Hirt und Nichols LVIRA ELVIRA



64×64

128×128

256×256

A.8.2 Deformation, Pilliods Advektion

Gesamter relativer Volumenverlust

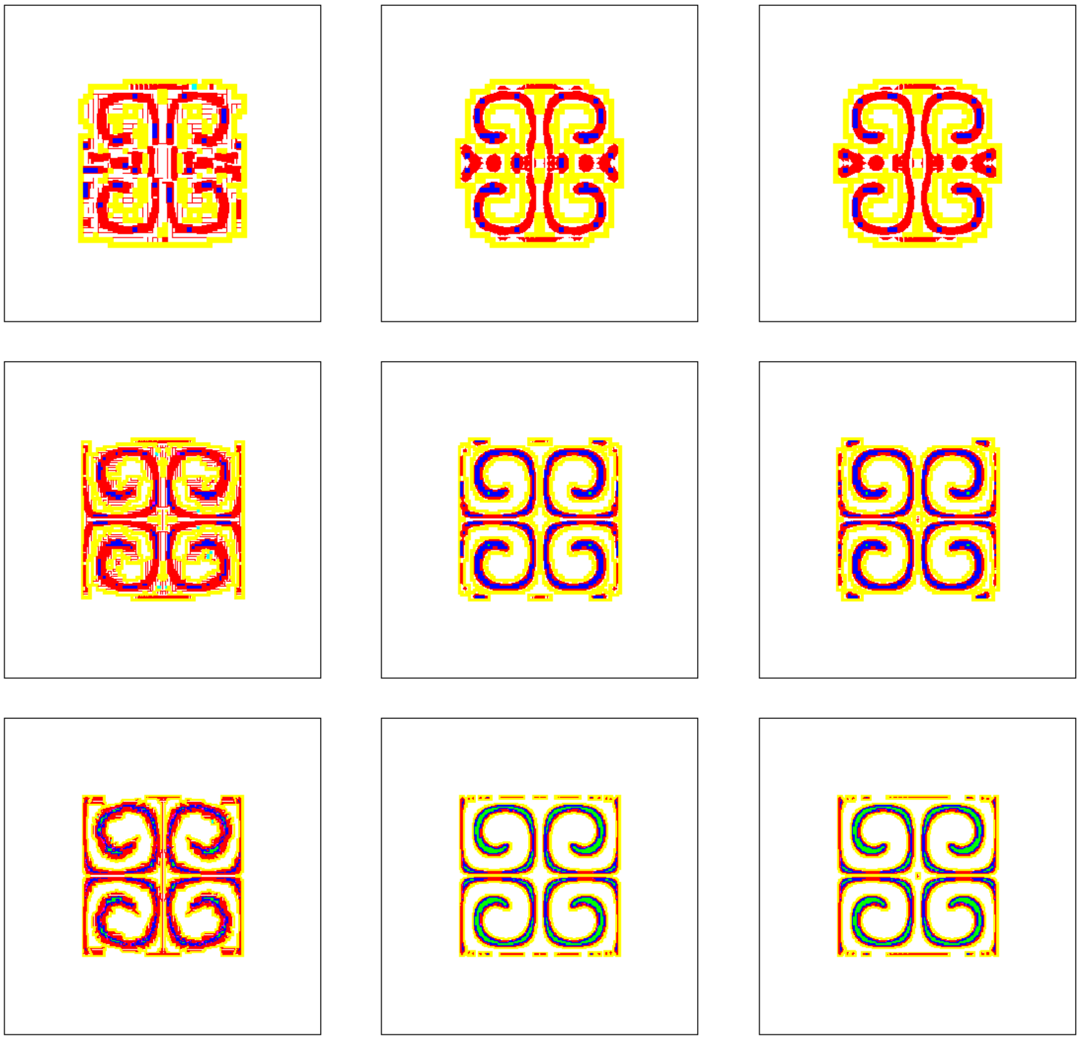
Dim.	SLIC	H&N	COM	P&Y	LVIRA	ELVIRA
64	-4.6382e-04	-4.6981e-06	-4.5889e-06	-3.9418e-06	-1.2354e-05	-6.9387e-06
128	-2.0479e-04	-5.4415e-07	-9.4990e-07	-7.2753e-07	-1.0759e-06	-9.4093e-07
256	-8.7496e-05	4.5351e-14	-8.8950e-09	-4.9799e-13	-9.4909e-09	9.6915e-11

SLIC Center of Mass Parker und Youngs

64×64

128×128

256×256



Deformation, Pilliods Advektion

Durchschnittlicher relativer Volumenverlust

Dim.	SLIC	H&N	COM	P&Y	LVIRA	ELVIRA
64	2.8037e-04	2.2018e-06	2.2429e-06	2.5259e-06	5.2199e-06	3.4138e-06
128	4.1735e-05	5.2419e-07	6.1894e-07	6.1515e-07	7.0620e-07	5.9649e-07
256	1.9168e-05	1.5079e-14	4.6707e-09	2.6298e-13	3.4294e-09	2.5518e-11

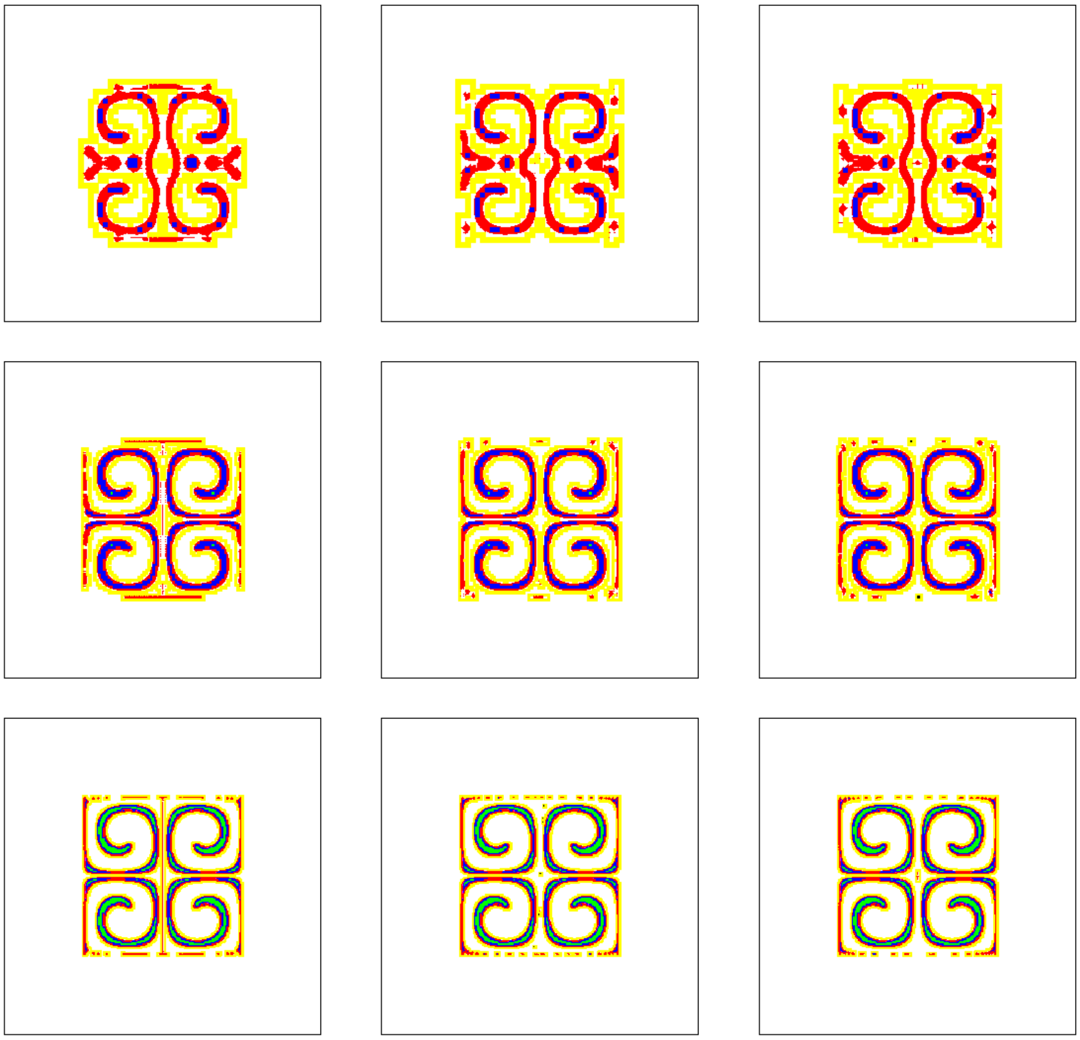
Füllungstoleranz fixiert auf 2.3283e-10.

Hirt und Nichols LVIRA ELVIRA

64 × 64

128 × 128

256 × 256



A.9 Englische Fachbegriffe

Englisch	Deutsch
antialiasing	Glättung eines Bildes, so daß statt scharfer Kanten sanftere Übergänge entstehen.
clipping	Berechnung der Schnittmenge zweier Polygone mittels sukzessiver Bearbeitung der Kanten und eventueller Schnittpunktberechnung.
center of mass method	Ein auf Puckett und Saltzman zurückgehendes Rekonstruktionsverfahren, das die Normale aus dem Massenschwerpunkt des Fluids berechnet (siehe Abschnitt 3.3.7).
color-function	Siehe *volume–of–fluid–function*
corner transport upwind scheme	Aufwind-Eckentransport-Schema. Ein auf Colella [11] zurückgehender mehrdimensionaler Advektionsalgorithmus (siehe Abschnitt 3.4.3.1).
donor-acceptor method	Spender–Empfänger–Methode. Ein eindimensionaler Advektionsalgorithmus, der die Front implizit treppenförmig rekonstruiert (siehe Abschnitt 3.4.2.1).
flotsam	Treibgut. Ein Fluidtröpfchen, das sich aufgrund von Fehlern bei der Frontenverfolgung von einem größeren Fluidbereich abreißt.
fractional-step method	Advektionsmethode, die auf *operator splitting* beruht (siehe Abschnitt 3.4.2.2).
jetsam	„Über Bord geworfene Ladung“. Ähnlich wie *flotsam* ein typischer Fehler bei VOF-Methoden
level set method	Niveaumengenmethode. Methode zur Frontenverfolgung, bei der statt der Front selbst eine Funktion verfolgt wird, in die die Front als Nullniveaulinie eingebettet ist (siehe Abschnitt 2.3).
marker particle	Ein Markierungsteilchen, dessen Bewegung in einer Partikelmethode verfolgt wird (siehe Abschnitt 2.4).
numerical vorticity	Spezieller Fehler in den Füllungen, durch den die Beschränkung der Füllungsfunktion verletzt wird.
numerical surface tension	Numerische Oberflächenspannung. Die typische Eigenschaft von VOF–Methoden, zu dünne Fluidbereiche in Tröpfchen zu zerteilen.
onion-skin treatment	Zwiebelschalenbehandlung. Eine Methode zur Behandlung von Mehrphasenflüssen, bei der die Phasen nacheinander bearbeitet werden (siehe Abschnitt 3.2).
operator splitting	Operatorspaltung. Mehrdimensionale Probleme werden durch sukzessive Behandlung eindimensionaler Probleme gelöst (siehe Abschnitt 3.4.2).
predictor-corrector method	Lösungsmethode für gewöhnliche Differentialgleichungen
shock-capturing method	Advektionsmethode, die speziell auf Stöße reagiert

Englisch	Deutsch
staggered grid	Anordnung der diskreten physikalischen Größen auf dem Rechengitter, bei der die horizontalen Geschwindigkeitskomponenten auf den Mittelpunkten der senkrechten Kanten und die vertikalen auf den Mittelpunkten der waagerechten Kanten abgenommen werden (siehe Abschnitt 3.4.1).
unsplit advection scheme	Ein echt mehrdimensionales Advektionsschema, also eines, daß kein *operator splitting* benutzt (siehe Abschnitt 3.4.3).
volume fraction	Wert der *volume-of-fluid-function* in einer Zelle.
volume of fluid	Fluidvolumen.
volume-of-fluid-function	Füllungsfunktion. Eine Funktion, die jeder Zelle der Diskretisierung das Verhältnis zwischen Fluid- und Gesamtvolumen der Zelle zuweist (siehe Abschnitt 3.3.1).
vorticity	Wirbelstärke, die Divergenz des Geschwindigkeitsfeldes. Bei inkompressiblen Strömungen verschwindet sie überall.

A.10 Erklärung

Ich versichere hiermit, daß ich die vorliegende Diplomarbeit selbständig verfaßt habe und keine außer den angegebenen Quellen und Hilfsmitteln benutzt habe.

Hamburg, den Eingereicht am 21. Mai 1999
korrigierte Version