Karlsruher Institut für Technologie

Bachelorarbeit

Gitterverfeinerte Lattice Boltzmann Methoden in OpenLB

Adrian Kummerländer

31. März 2019

Betreuung: Dr. Mathias J. Krause

Fakultät für Mathematik

Karlsruher Institut für Technologie

Zusammenfassung

Dank moderner paralleler Hochleistungsrechner können immer mehr praktische Strömungsprobleme mittels numerischer Simulationen gelöst werden. Ein Ansatz dazu ist die Lattice Boltzmann Methode, welche u. a. durch ihre gute Skalierbarkeit auf eben diesen Parallelrechnern zunehmend an Bedeutung gewinnt.

Trotz anhaltendem Wachstum der verfügbaren Rechenleistung können viele reale Strömungsprobleme weiterhin nur unter Einschränkungen in akzeptabler Zeit und Genauigkeit gelöst werden. Ein Ansatz, diesem Konflikt zu begegnen, ist die lokale Verfeinerung der LBM zugrunde liegenden Gitter.

OpenLB ist eine in C++ geschriebene freie Bibliothek zur Implementierung von LBM basierenden Strömungssimulationen. Aktuell bietet OpenLB noch keine Unterstützung für Gitterverfeinerung.

Ziel dieser Arbeit ist es, aus der Behebung dieser Einschränkung heraus einen Gitterverfeinerungsansatz theoretisch nachzuvollziehen und dessen Einsetzbarkeit praktisch zu evaluieren. Zu diesen Zweck werden anhand einer zweidimensionalen Lattice Boltzmann Methode verschiedene Ansätze zur Verfeinerung von Gittern diskutiert. Darauf aufbauend wird ein konkretes Verfahren detailliert ausformuliert und im Rahmen der Entwicklung eines generischen Gitterverfeinerungsframeworks umgesetzt. Der übergeordneten Fragestellung nach dem tatsächlichen Nutzen und möglichen Problemen von gitterverfeinerten Lattice Boltzmann Methoden wird durch die Evaluation von Anwendungsbeispielen Rechnung getragen. In diesem Kontext findet weiterhin eine Besprechung der formal begründeten anwendungsbezogenen Auswahl von zu verfeinernden Gebieten statt.

Inhaltsverzeichnis

1 Einführung **1**
- 1.1 Wieso Strömungen simulieren? . . . 1
- 1.2 Weshalb mit Lattice Boltzmann Methoden? . . . 2
- 1.3 Warum Gitterverfeinerung? . . . 2

2 Grundlagen **3**
- 2.1 Lattice Boltzmann Methode . . . 3
  - 2.1.1 Algorithmus . . . 7
  - 2.1.2 Chapman-Enskog Analyse . . . 7
- 2.2 Herangehensweisen an Gitterverfeinerung . . . 9
  - 2.2.1 Multi-Grid Ansatz . . . 9
  - 2.2.2 Multi-Domain Ansatz . . . 9
  - 2.2.3 Koinzidierende und versetzte Gitter . . . 10

3 Verfeinerungsmethode nach Lagrava et al. **11**
- 3.1 Übersicht . . . 11
- 3.2 Gitterdiskretisierung . . . 12
- 3.3 Komponenten der Gitterkopplung . . . 16
  - 3.3.1 Skalierung . . . 16
  - 3.3.2 Restriktion . . . 19
  - 3.3.3 Interpolation . . . 20
- 3.4 Algorithmus . . . 24

4 Implementierung in OpenLB **27**
- 4.1 Architekturübersicht . . . 27
- 4.2 Auswahl der Verfeinerungsmethode . . . 28
- 4.3 Struktur des Gitterverfeinerungsframeworks . . . 30
- 4.4 Umsetzung des Verfahrens von Lagrava et al. . . . 33

5 Evaluierung **36**
- 5.1 Rohrströmung . . . 37
  - 5.1.1 Vergleich mit der analytischen Lösung . . . 39
  - 5.1.2 Vergleich der Interpolationsverfahren . . . 40
- 5.2 Umströmter Zylinder . . . 43
  - 5.2.1 Anwendung eines formalen Kriteriums zur Gitterverfeinerung . . . 45
  - 5.2.2 Parallelisierung . . . 49
  - 5.2.3 Vergleich von Widerstands- und Auftriebskoeffizienten . . . 50

6 Fazit **57**

Abbildungsverzeichnis **59**

Literatur **61**

1 Einführung

1.1 Wieso Strömungen simulieren?

Das hoch technisierte Lebensumfeld des modernen Menschen ist ohne ein detailliertes Verständnis des Verhaltens der, durch ihn in wachsendem Maße kontrollierten, Natur undenkbar. Eine wichtige Komponente dieses Naturverständnisses ist das Wissen um das Verhalten von Flüssigkeiten und Gasen, die sich als Strömungen bewegen. Ohne dieses Verständnis führe kein Automobil, flöge kein Flugzeug und drehten sich weder ein Windrad um seine Achse noch ein Satellit um unseren Planeten.

Experimente zur Bestimmung des Verhaltens von Strömungen – z. B. in Wind- und Wasserkanälen – sind möglich, liefern naturnaheste Ergebnisse und stellen eine zentrale Komponente der Entwicklung eben genannter Errungenschaften dar. Leider sind reale Experimente im Allgemeinen nicht nur sehr aufwendig in Aufbau und Durchführung, sondern stoßen auch insbesondere bei der Betrachtung mikroskopischer Probleme – etwa im Bereich der Medizin, deren menschliches Subjekt zu einem großen Teil ebenfalls eine *Strömungsmaschine* bildet – an Grenzen von Messmethoden und Ethik.

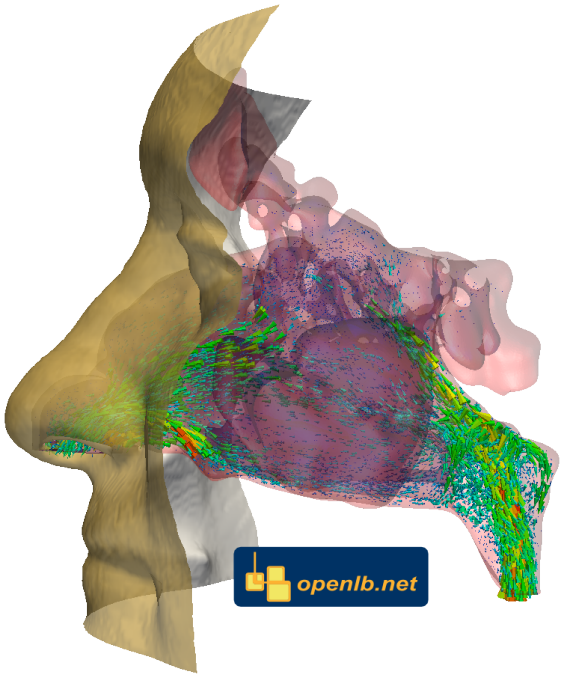


Abbildung 1: Strömung in der komplexen Geometrie der menschlichen Nase [13]

So trifft es sich, dass der Wunsch nach theoretischer Lösung komplexer und nur schwer analytisch zugänglicher Strömungsprobleme mit der Entwicklung von immer leistungsfähigeren Rechenmaschinen nicht nur einherging, sondern auch eine der Triebfedern in deren initialen Entwicklung war. Moderne numerische Verfahren zur Simulation von Strömungen versprechen eine zunehmende Reduzierung benötigter realer Experimente und sind heute gängiges Werkzeug in Forschung und Maschinenbau.

1.2 Weshalb mit Lattice Boltzmann Methoden?

Während *Finite Elemente Methoden* die wohl verbreitetsten Zugänge zur Lösung der strömungsbeschreibenden partiellen Differentialgleichungen und somit der numerischen Strömungsmechanik bilden, erfreut sich auch die Herangehensweise der *Lattice Boltzmann Methoden* (LBM) in den letzten Jahrzehnten wachsender Nutzbarkeit und Verbreitung. Im Gegensatz zu anderen Lösungsmethoden werden hier die Navier-Stokes Gleichungen nicht direkt numerisch gelöst. Lösungen ergeben sich vielmehr aus der Simulation des Fluidverhaltens auf *mesoskopischer* Ebene – d. h. aus der Betrachtung nicht aus Sicht der Kollision einzelner Fluidpartikel und nicht aus Sicht der analytischen Strömungsbeschreibung, sondern aus Sicht der Wahrscheinlichkeit, dass sich Fluidatome zu bestimmter Zeit an einem bestimmten Ort mit einer bestimmten mikroskopischen Geschwindigkeit bewegen.

Ein Vorteil dieses, auf den Arbeiten von Ludwig Eduard Boltzmann im Bereich der statistischen Physik aufbauenden, Ansatzes, ist seine Eignung für komplexe Geometrien mit verschiedensten Randbedingungen [1]. Weiterhin gewinnt in den letzten Jahren auch die sehr gute Parallelisierbarkeit von LBM in Hinblick auf einen technischen Fortschritt an Anziehungskraft, nach welchem die Leistungsfähigkeit von Großrechnern eher aus deren Parallelität als aus individueller Prozessorleistung erwächst.

1.3 Warum Gitterverfeinerung?

Die einfachsten und zugleich am weitesten verbreiteten Umsetzungen von Simulationen mit LBM basieren auf uniformen Gittern, in denen Zellen immer den gleichen Abstand zu ihren Nachbarzellen haben.

Die Genauigkeit von Lattice Boltzmann (LB) basierenden Simulationen hängt maßgeblich von der Auflösung des verwendeten Gitters ab. Bei Außerachtlassung weiterer wichtiger Faktoren wie dem verwendeten Kollisionsterm und Randkonditionen kann im Allgemeinen davon ausgegangen werden, dass eine feinere Auflösung des Gitters zu besseren Ergebnissen führt.

In praktischen Beispielen können innerhalb eines Modells große Unterschiede in der Strömungskomplexität existieren. So kann es große Gebiete eines Modells geben, die mit einem vergleichsweise groben Gitter gut simuliert werden können, während in anderen Gebieten – beispielsweise in komplexen Geometrien und an Rändern mit turbulenten Grenzschichten – ein vielfach feineres Gitter zur adäquaten Behandlung benötigt wird. In uniformen Gittern muss jedoch das gesamte Modell unabhängig der lokalen Situation mit der maximal benötigten Auflösung abgebildet werden.

Da die Anzahl der benötigten Gitterpunkte sich maßgeblich auf den Speicherbedarf und Rechenaufwand auswirkt, ist es wünschenswert, diese zu minimieren. Ein Ansatz, dies zu erreichen, ist die lokale Variation der Gitterauflösung.

2 Grundlagen

In diesem Kapitel werden wir die, dem weiteren Verlauf dieser Arbeit zugrunde liegende, Lattice Boltzmann Methode in 2D nachvollziehen [16, vgl. Kapitel 3].

2.1 Lattice Boltzmann Methode

Grundlage und Namensgeber von Simulationen mit Lattice Boltzmann Methoden ist die Boltzmann Gleichung. Sie beschreibt das Verhalten von Gasen auf mesoskopischer Ebene als Verteilungsfunktion der Masse von Partikeln in einer Raumregion mit gegebener Geschwindigkeit.

Definition 2.1 (Die Boltzmann-Gleichung) Sei $f(x, \xi, t)$ die Verteilungsfunktion der Partikelmasse zu Zeit t in Ort $x \in \mathbb{R}^2$ mit Geschwindigkeit $\xi \in \mathbb{R}^2$, ρ die Dichte und $F \in \mathbb{R}^2$ eine etwaige äußere Kraft. Die Boltzmann-Gleichung beschreibt die zeitliche Veränderung der Verteilungsfunktion anhand des totalen Differential $\Omega(f)$:

$$\Omega(f) = \left(\frac{dt}{dt}\partial_t + \frac{dx}{dt}\partial_x + \frac{d\xi}{dt}\partial_\xi\right) f = \left(\partial_t + \xi\,\partial_x + \frac{F}{\rho}\,\partial_\xi\right) f.$$

Hierbei handelt es sich um eine Advektionsgleichung wobei der Term $\partial_t f + \xi\,\partial_x f$ die Strömung der Partikelverteilung mit Geschwindigkeit ξ und $\frac{F}{\rho}\,\partial_\xi f$ einwirkende Kräfte darstellt. Der Term $\Omega(f)$ beschreibt, entsprechend als Kollisionsoperator bezeichnet, die kollisionsbedingte lokale Neuverteilung von f.

Zentrale Anforderung an den Kollisionsoperator ist die Impuls- und Masseerhaltung. Die im Folgenden betrachtete Lattice Boltzmann Methode verwendet die übliche BGK Approximation der Boltzmann-Gleichung ohne äußere Kraft von Bhatnagar, Gross und Krook (siehe *The Lattice Boltzmann Method: Principles and Practice* [16, Kap. 3.5.3]). Grundlegendes Element dieser Approximation ist der BGK Operator

$$\Omega(f) := -\frac{f - f^{\text{eq}}}{\tau}\Delta t,$$

welcher die Partikelverteilung mit Rate τ gegen eine Equilibriumsverteilung f^{eq} (vgl. 2.6) relaxiert. Ohne Beschränkung der Allgemeinheit setzen wir dabei im Folgenden $\Delta t = 1$.

Wenden wir den BGK Operator auf die Boltzmann-Gleichung ohne äußere Kräfte an, erhalten wir die BGK Approximation:

Definition 2.2 (BGK-Boltzmann-Gleichung) Sei $\tau \in \mathbb{R}_{\geq 0}$ eine Relaxationszeit und f^{eq} die von der Maxwell-Boltzmann-Verteilung gegebene Equilibriumsverteilung

$$(\partial_t + \xi \cdot \nabla_x) f = -\frac{1}{\tau}(f(x, \xi, t) - f^{\text{eq}}(x, \xi, t)).$$

Analog zur Boltzmann-Gleichung ist auch bei deren Approximation in der BGK-Boltzmann-Gleichung der beschriebene Ort $x \in \mathbb{R}^2$ im Allgemeinen frei gewählt. Da unser Ziel jedoch gerade die Diskretisierung des Simulationsgebiets in einem Gitter ist, wollen wir x einschränken:

Definition 2.3 (Ortsdiskretisierung) Sei das Simulationsgebiet $D \subseteq \mathbb{R}^2$ diskretisiert als kartesisches Gitter mit Zellabstand $\Delta x \in \mathbb{R}_+$. Dann ist die Einbettung der kartesischen Gitterdomäne $L := \mathbb{Z}^2$ in D gegeben als:

$$d : L \to D,\ l \mapsto \Delta x \cdot l.$$

Analog zur o.B.d.A. erfolgten Wahl von $\Delta t = 1$ setzen wir auch hier $\Delta x = 1$, sodass wir die Simulations- und Gitterdomäne – bei Inklusion stetiger Übergänge zwischen Elementen aus L – identisch identifizieren können. Der stetige Übergang zwischen zwei orthodonal benachbarten Gitterknoten erfolgt somit auf dem Einheitsintervall. Praktisch können wir also im Folgenden bei Annahme von $x \in L \subset D$ die explizite Ausführung der Gittereinbettung vernachlässigen.

Zu betonen ist an dieser Stelle die Wichtigkeit einer klaren Unterscheidung zwischen Simulationsgebiet und dem durch dieses zu modellierenden physikalischen System für die konkrete Interpretation des Simulationsergebnisses [16, Kap. 7].

Für die verbleibende Herleitung der LBM können wir diese Interpretation, d. h. die Unterscheidung zwischen physikalischen und Lattice-Einheiten, jedoch außer Acht lassen, da sich die modellierten physikalischen Einheiten aus der Wahl der Relaxationszeit und der Skalierung der Lattice-Momente ergeben. Diese Wahl wird hier nicht weiter eingeschränkt.

Wir bemerken nun, dass die BGK Approximation nicht nur für beliebige Orte, sondern auch für beliebige Geschwindigkeiten $\xi \in \mathbb{R}^2$ definiert ist. Da wir die LBM auf einem Computer umsetzen wollen, müssen wir die Menge der betrachteten Geschwindigkeiten auf eine endliche Menge diskretisieren.

Eine übliche Menge diskreter Geschwindigkeiten in 2D ist *D2Q9* wobei *D2* die Anzahl der Dimensionen und *Q9* die Anzahl der Geschwindigkeiten verschlüsselt.

Definition 2.4 (D2Q9 Modell)

$$\{\xi_i\}_{i=0}^{8} = \left\{ \begin{pmatrix}0\\0\end{pmatrix}, \begin{pmatrix}-1\\1\end{pmatrix}, \begin{pmatrix}-1\\0\end{pmatrix}, \begin{pmatrix}-1\\-1\end{pmatrix}, \begin{pmatrix}0\\-1\end{pmatrix}, \begin{pmatrix}1\\-1\end{pmatrix}, \begin{pmatrix}1\\0\end{pmatrix}, \begin{pmatrix}1\\1\end{pmatrix}, \begin{pmatrix}0\\1\end{pmatrix} \right\}.$$

Mithilfe einer solchen endlichen Menge diskreter Geschwindigkeiten lässt sich die BGK Approximation bezüglich der Geschwindigkeit diskretisieren:

Definition 2.5 (BGK Geschwindigkeitsdiskretisierung) Seien ξ_i Vektoren einer Menge mikroskopischer Geschwindigkeiten wie z.B. D2Q9 und $f_i(x,t) := f(x, \xi_i, t)$. Dann ist

$$(\partial_t + \xi_i \cdot \nabla_x) f_i(x,t) = -\frac{1}{\tau}(f_i(x,t) - f_i^{\mathrm{eq}}(x,t))$$

die Diskretisierung der BGK Approximation entlang der Geschwindigkeiten in diskreten Gitterknoten $x \in L$. Die Geschwindigkeiten müssen hier dank der Wahl von $\Delta x = 1$ nicht weiter skaliert werden.

Hierbei ist die diskrete Equilibriumsverteilung f_i^{eq} wie folgt definiert:

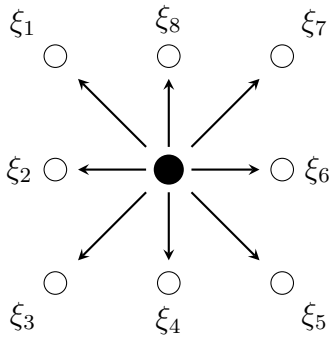


Abbildung 2: Umgebung einer Zelle in D2Q9

Definition 2.6 (Diskrete Equilibriumsverteilung) Seien $\rho \in \mathbb{R}_{\geq 0}$ die Dichte, $u \in \mathbb{R}^2$ die Gesamtgeschwindigkeit (siehe Def. 2.7), ξ_i die i-te diskrete Geschwindigkeitskomponente (siehe Def. 2.4), w_i das Gewicht (2.1) jener Komponente bzgl. des Lattice und c_s die Lattice-Schallgeschwindigkeit. Dann ist die diskrete Equilibriumsverteilung gegeben als:

$$f_i^{\text{eq}} = w_i \rho \left(1 + \frac{u \cdot \xi_i}{c_s^2} + \frac{(u \cdot \xi_i)^2}{2c_s^4} - \frac{u \cdot u}{2c_s^2}\right).$$

Die Werte von $u = u(x,t)$ und $\rho = \rho(x,t)$ in Ort x zu Zeit t ergeben sich dabei aus den *Momenten* der Verteilungsfunktion f_i:

Definition 2.7 (Momente der Verteilungsfunktion)

$$\rho(x,t) = \sum_{i=0}^{q-1} f_i(x,t)$$
$$\rho u(x,t) = \sum_{i=0}^{q-1} \xi_i f_i(x,t).$$

Für D2Q9 ergeben sich nach [16, Gl. 3.60 bzw. Tabelle 3.3] die Gewichte:

$$w_0 = \frac{4}{9},\ w_{2,4,6,8} = \frac{1}{9},\ w_{1,3,5,7} = \frac{1}{36} \tag{2.1}$$

Weiter folgt zusammen mit der Bedingung $\sum_{i=1}^{q-1} w_i (\xi_i)_a (\xi_i)_b = c_s^2 \delta_{a,b}$ aus [16, Gl. 3.60] die Schallgeschwindigkeit $c_s = \sqrt{1/3}$ des Lattice. Konditionen zur Bestimmung dieser gitterspezifischen Konstanten sind hierbei die Erhaltung von Impuls und Masse sowie die Forderung von *Rotationsisotropie*.

Zur Entwicklung einer *implementierbaren* expliziten BGK Gleichung können wir nun die Geschwindigkeitsdiskretisierung 2.5 integrieren:

$$f_i(x + \xi_i, t+1) - f_i(x,t) = \int_0^1 \Omega_i(x + \xi_i s, t+s) ds$$

Wobei $\Omega_i(x,t)$ hier die diskrete Formulierung des BGK Kollisionsoperators darstellt:

$$\Omega_i(x,t) := -\frac{1}{\tau}(f_i(x,t) - f_i^{\text{eq}}(x,t))$$

Da sich die exakte Lösung des Integrals auf der rechten Seite schwierig gestaltet, wird es in der Praxis nur approximiert. Während es dazu vielfältige Ansätze gibt, beschränken wir uns an dieser Stelle auf Anwendung der Trapezregel [3, Kap. 6]:

$$\begin{aligned} f_i(x+\xi_i, t+1) - f_i(x,t) &= \frac{1}{2}\left(\Omega_i(x,t) + \Omega_i(x+\xi_i, t+1)\right) \\ &= -\frac{1}{2\tau}\left(f_i(x+\xi_i,t+1) + f_i(x,t) - f_i^{\text{eq}}(x+\xi_i,t+1) - f_i^{\text{eq}}(x,t)\right) \end{aligned}$$

Zur expliziten Lösung dieser impliziten Gleichung benötigen wir nun nur noch eine geeignete Verschiebung von f_i und τ:

Definition 2.8 (Diskrete LBM BGK Gleichung) Seien $\overline{f_i}$ und $\overline{\tau}$ definiert:

$$\begin{aligned} \overline{f_i} &:= f_i + \frac{1}{2\tau}(f_i - f_i^{\text{eq}}) \\ \overline{\tau} &:= \tau + \frac{1}{2}. \end{aligned}$$

Setzen wir diese verschobenen Variablen in das Ergebnis der Trapezregel ein, erhalten [16, Kap. A.5 mit $\Delta t = 1$] wir die die vollständig diskretisierte LBM BGK Gleichung:

$$\overline{f_i}(x+\xi_i, t+1) = \overline{f_i}(x,t) - \frac{1}{\overline{\tau}}(\overline{f_i}(x,t) - f_i^{\text{eq}}(x,t))$$

Bemerkenswert ist an dieser Stelle, dass die Momente der Verteilungen mit $\overline{f_i}$ analog zu Definition 2.7 berechnet werden können:

$$\begin{aligned} \sum_{i=0}^{q-1} \overline{f_i} &= \sum_{i=0}^{q-1} f_i + \frac{1}{2\tau}\sum_{i=0}^{q-1}(f_i - f_i^{\text{eq}}) = \rho \\ \sum_{i=0}^{q-1} \xi_i \overline{f_i} &= \sum_{i=0}^{q-1} \xi_i f_i + \frac{1}{2\tau}\sum_{i=0}^{q-1}(f_i - f_i^{\text{eq}}) = \rho u. \end{aligned}$$

2.1.1 Algorithmus

Bei der Implementierung der diskreten LBM BGK Gleichung 2.8 auf einem Computer ist die Aufteilung in Kollisions- und Strömungsschritt üblich.

Definition 2.9 (Kollisionsschritt) Annäherung der Verteilungsfunktion an die lokal berechnete Equilibriumsverteilung entsprechend dem BGK Kollisionsoperator.

$$f_i^{\text{out}}(x,t) = f_i(x,t) - \frac{1}{\tau}(f_i(x,t) - f_i^{\text{eq}}(x,t))$$

Definition 2.10 (Strömungsschritt) Strömen der neuen Verteilungen auf die benachbarten Zellen entsprechend der jeweiligen diskreten Geschwindigkeit.

$$f_i(x+\xi_i, t+1) = f_i^{\text{out}}(x,t)$$

Bemerkenswert ist hierbei, dass der Kollisionsschritt nur lokale Informationen der jeweiligen Zelle benötigt und sich somit sehr gut zur parallelen Verarbeitung eignet.

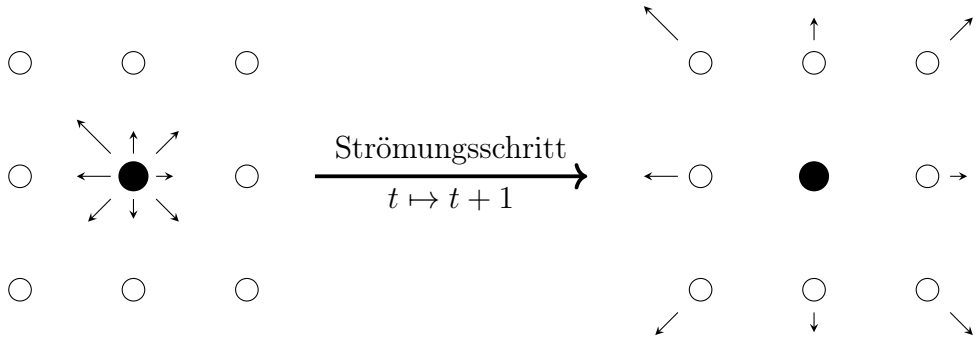


Abbildung 3: Strömung der im Kollisionsschritt relaxierten Verteilungen

2.1.2 Chapman-Enskog Analyse

Ziel der beschriebenen Lattice Boltzmann Methode ist die möglichst gute Approximation der inkompressiblen Navier-Stokes Gleichungen auf dem Simulationsgebiet.

Definition 2.11 (Inkompressible Navier-Stokes Gleichungen) Sei ρ die konstante Dichte, p der Druck und u die Geschwindigkeit zu Zeit t sowie ν die kinematische Viskosität und S der Verzerrungstensor. Die inkompressiblen Navier-Stokes Gleichungen sind dann:

$$\begin{aligned}\partial_t \rho + \nabla \cdot (\rho u) &= 0\\ \partial_t u + (u \cdot \nabla)u &= -\frac{1}{\rho}\nabla p + 2\nu \nabla \cdot (\mathrm{S}).\end{aligned}$$

Dabei sind Druck p, kinetische Viskosität ν und Verzerrungstensor S definiert als:

$$\begin{aligned}p &= c_s^2 \rho\\ \nu &= c_s^2 \tau\\ \mathrm{S} &= \frac{1}{2}(\nabla u + (\nabla u)^\top).\end{aligned} \tag{2.2}$$

Nach [16, Kap. 4.1] kann die asymptotische Äquivalenz von LBM BGK Gleichung und inkompressiblen Navier-Stokes Gleichungen mit der Entwicklung von Chapman-Enskog gezeigt werden.

Definition 2.12 (Chapman-Enskog Ansatz) Der Chapman-Enskog Ansatz besteht in der Annahme, dass die Verteilungsfunktion f_i als leicht gestörte Equilibriumsverteilung dargestellt werden kann:

$$f_i = f_i^{\mathrm{eq}} + \epsilon f_i^{(1)} + \mathcal{O}(\epsilon^2).$$

Hierbei ist $\epsilon f_i^{(1)}$ mit $\epsilon \ll 1$ der Störterm 1. Ordnung. Dieser ist gegeben als:

$$\epsilon f_i^{(1)} = \frac{w_i}{2c_s^4} \mathrm{Q}_i : \Pi^{(1)}. \tag{2.3}$$

Wobei der Geschwindigkeitstensor Q_i und das Störmoment $\Pi^{(1)}$ nach [16, Kap. 4.1.3] definiert sind als:

$$\mathrm{Q}_i = \xi_i \xi_i - c_s^2 I \tag{2.4}$$

$$\Pi^{(1)} = \sum_{i=0}^{q-1} \xi_i \xi_i \epsilon f_i^{(1)} = -2c_s^2 \rho \overline{\tau} \mathrm{S}. \tag{2.5}$$

Definition 2.13 (Nicht-Equilibriumsverteilung) Die Relaxion der Verteilungsfunktion f_i gegen die Equilibriumsverteilung f_i^{eq} impliziert eine Dekomposition in Equilibriums- und Nicht-Equilibriumsverteilung:

$$f_i := f_i^{\mathrm{eq}} + f_i^{\mathrm{neq}}.$$

Unter Vernachlässigung von Störtermen der Ordnung $\mathcal{O}(\epsilon^2)$ ergibt sich eine Näherung der Nicht-Equilibriumsverteilung:

$$f_i^{\mathrm{neq}} \cong \epsilon f_i^{(1)}$$

Diese Darstellung können wir unter Verwendung von Definition 2.12 ausführen als:

$$f_i^{\mathrm{neq}} \cong -\frac{w_i \rho \overline{\tau}}{c_s^2} \mathrm{Q}_i : \mathrm{S} \tag{2.6}$$

2.2 Herangehensweisen an Gitterverfeinerung

Grundsätzlich existieren mit der *Multi-Grid* und *Multi-Domain* Herangehenweise zwei verschiedene Ansätze für Gitterverfeinerung in LBM [17, Kap. 3.1]. Im Wesentlichen unterscheiden die Ansätze sich in den Ausmaßen der variabel aufgelösten Teilgitter des Simulationsgebiets. Weitere Unterschiede folgen dann aus dieser grundlegenden Struktur.

2.2.1 Multi-Grid Ansatz

Bei Verfahren des Multi-Grid Ansatzes [19] [22] existiert das gröbste Gitter über der gesamten Domäne. Feinere Teilgitter werden über gröberen Gittern platziert und nicht aus deren Verarbeitung ausgeschlossen. Somit existieren über der gesamten *Fläche* feinerer Gitter auch die Knoten gröberer Gitter.

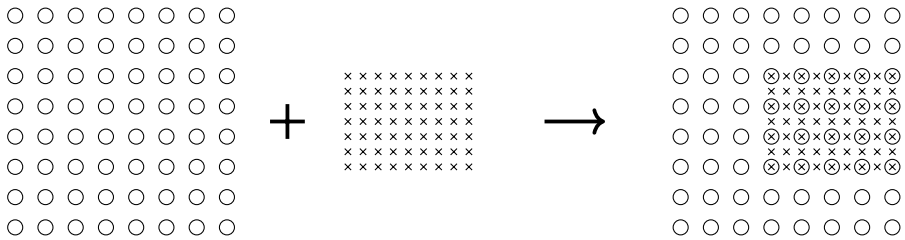


Abbildung 4: Teilgitter in der Multi-Grid Herangehensweise

Vorteil dieser Herangehensweise ist es, dass feinere Teilgitter im Simulationsverlauf ohne aufwendige Restrukturierung verschoben werden können, um beispielsweise komplexere Strömungsstrukturen zu *verfolgen*. Nachteil ist, dass das Einsparpotentiale in Speicher- und Rechenbedarf wegen mehrfacher Abdeckung von Teilen des Simulationsgebiets durch überlagerte Gitter nicht voll ausgenutzt werden können.

2.2.2 Multi-Domain Ansatz

Kern von Multi-Domain Ansätzen [5] [7] [8] [9] [10] [17, ausführlich diskutiert in Kap. 3] [20] [21] ist es, außerhalb von etwaigen verfahrensbedingten Übergangsbereichen, jede Position des Simulationsgebiets durch genau ein Teilgitter abzubilden. Konkret werden also bereits durch feinere Gitter abgedeckte Bereiche aus gröberen Teilgittern ausgespart.

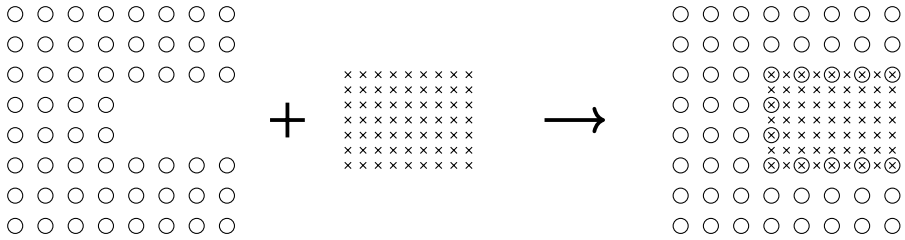


Abbildung 5: Teiligitter in der Multi-Domain Herangehensweise

Vorteil gegenüber des Multi-Grid Ansatzes ist hier der weiter reduzierte Speicherbedarf sowie erwartete Einsparungen in der benötigten Rechenzeit. *Erkauft* werden diese Vorteile durch aufwendigere Kopplung [17, Kap. 3.1] der verschiedenen Teilgitter in den Übergangsbereichen.

2.2.3 Koinzidierende und versetzte Gitter

Neben der Unterscheidung zwischen Multi-Grid und Multi-Domain Ansätzen kann auch die Positionierung des groben im Bezug auf das feine Gitter variieren. Gitterkonstellationen, in denen grobe Knoten soweit möglich mit feinen Knoten übereinstimmen – wie auch in den beiden vorherigen Abbildung zu sehen – werden als koinzidierend bezeichnet:

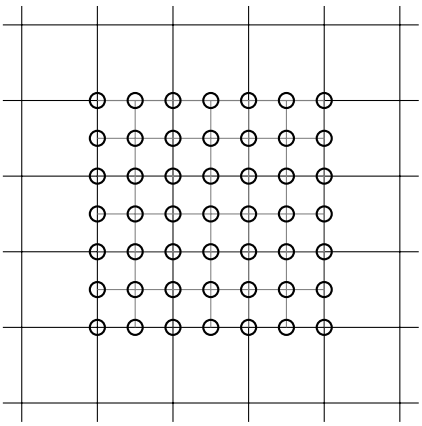


Abbildung 6: Koinzidierende Gitter

Lösen wir uns von dieser Einschränkung und positionieren das feine Gitter abseits der groben Gitterknoten, sprechen wir über zueinander versetzte – im angelsächsischen Sprachraum als *staggered* beschriebene – Gitter:

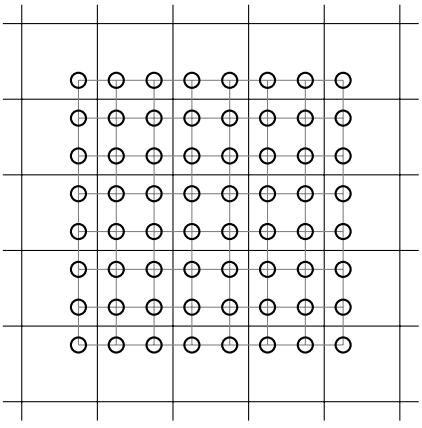


Abbildung 7: Zueinander versetzte Gitter ohne überlappende Knoten

3 Verfeinerungsmethode nach Lagrava et al.

Wie in Kapitel 4.2 noch näher begründet werden wird, bieten sich der Multi-Domain Ansatz als Grundlage für Gitterverfeinerung in OpenLB an. Passend zu dieser Wahl sowie der, im Rahmen dieser Arbeit getroffenen, Einschränkung auf zweidimensionale LBM mit BGK-Kollisionsoperator haben Lagrava et al. 2012 in *Advances in Multi-domain Lattice Boltzmann Grid Refinement* [17] ein solches Verfeinerungsverfahren entwickelt. Die Struktur dieses Verfahrens, mit potenziell austauschbaren Restriktions- und Interpolationsoperatoren im zentralen Kopplungsschritt, erscheint dabei sogleich auch als Fundament eines Multi-Domain Gitterverfeinerungsframeworks in OpenLB.

3.1 Übersicht

Das Verfahren basiert auf dem Multi-Domain Ansatz [17, Kap. 3.1] mit koinzidierenden Gittern. Es werden also die feiner aufgelösten Teilbereiche des Simulationsgebiets so aus dem gröber aufgelösten Gitter ausgeschlossen, dass sie sich nur in Übergangsbereichen überlappen.

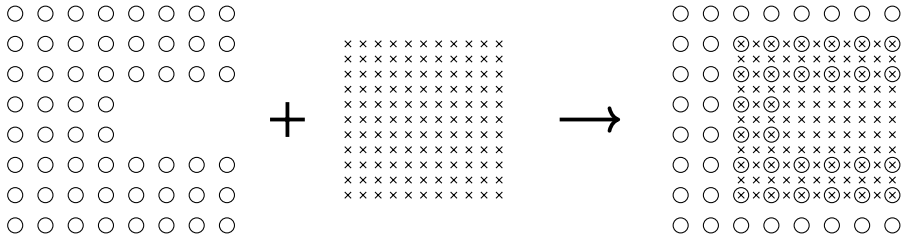


Abbildung 8: Multi-Domain Herangehensweise mit Übergangsbereich [17, vgl. Abb. 3]

In diesen Übergangsbereichen, welche eine Breite von mindestens einer groben Gitterweite haben müssen, liegt die Hauptarbeit des Verfeinerungsverfahrens.

Während der Übergang vom feinen zum groben Gitter sich im Wesentlichen auf eine skalierte und gefilterte Restriktion der Verteilungen beschränkt, gestaltet sich der Übergang vom groben zum feinen Gitter aufwendiger, da feine Knoten, für deren Position kein grober Knoten existiert, aus den übrigen Daten interpoliert werden müssen.

Entsprechend liegt der Fokus des von Lagrava et al. entwickelten Algorithmus auf der Auswahl des Interpolationsverfahrens sowie der Skalierung der physikalischen Werte zwischen den unterschiedlich aufgelösten Verteilungen. Um diese Kopplung der verschiedenen Gitterauflösungen theoretisch erfassen zu können, müssen wir zunächst die Gitter selbst konkreter definieren.

3.2 Gitterdiskretisierung

Definition 3.1 (Grobe und feine Simulationsgebiete) Sei $D \subseteq \mathbb{R}^2$ das physikalische Simulationsgebiet unabhängig der verwendeten Gitterauflösung. Diese Teilmenge von $\mathbb{R}^2$ sei dabei bereits in Hinblick auf die angestrebte kartesische Diskretisierung gewählt, d. h. als Vereinigung zusammenhängender Rechtecke.

Den durch ein grobes Gitter abzubildenden Teilbereich bezeichnen wir mit $D_g \subset D$. Weiter ist $D_f \subset D$ mit $D_f \cap D_g \neq \emptyset$ der zu verfeinernde Teilbereich. O.B.d.A. nehmen wir an, dass $D_g \cup D_f = D$ gilt, das Simulationsgebiet also in genau zwei Auflösungsstufen aufgeteilt wird.

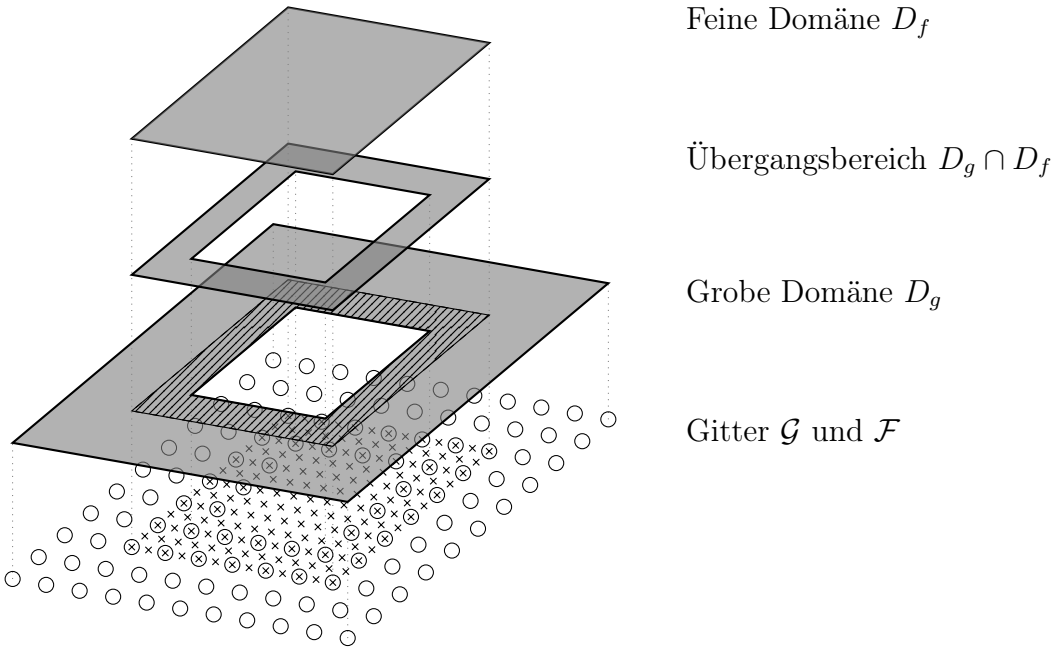


Abbildung 9: Schematische Darstellung der Simulationsgebiete der Gitter

Die Annahme von zwei Verfeinerungsstufen ist berechtigt, da mehrfach verfeinerte Gitter sich durch erneute – rekursive – Anwendung von Definition 3.1 mit $\tilde{D} := D_f$ modellieren lassen. Auch Verfeinerung eines groben Gitters in mehreren disjunkten Bereichen kann unter dieser Annahme betrachtet werden, da für die Koppelung zwischen Gittern nur Knoten in D_f relevant sind.

Definition 3.2 (Diskretisierung der Simulationsgebiete) Wir betrachten kartesische Gitter $\mathcal{G}$ und $\mathcal{F}$ als Diskretisierungen der Simulationsgebiete D_g bzw. D_f. Diese seien so gewählt, dass sie gerade die konvexen Hüllen ihrer koinzidierenden Diskretisierungsgitter beschreiben

$$\mathcal{G} \subset D_g \cap \{x \in \mathbb{R}^2 | \exists i \in \mathbb{Z}^2 : x = \delta x_g \cdot i\} \qquad \text{Gröberes Gitter}$$
$$\mathcal{F} \subset D_f \cap \{x \in \mathbb{R}^2 | \exists i \in \mathbb{Z}^2 : x = \delta x_f \cdot i\} \qquad \text{Feineres Gitter}$$

$\delta x_g = \delta x_g / 2 \in \mathbb{R}_+$ seien die Diskretisierungsauflösungen im Verhältnis $1 : 2$.

Zur Entwicklung der Gitterkopplung fordern wir, dass sich $\mathcal{G}$ und $\mathcal{F}$ um mindestens eine grobe Gitterweite δx_g überlappen – vgl. Abbildungen 8 und 11. Die Seitenlängen der konvexen Hüllen D_g und D_f sind ganzzahlige Vielfache von δx_g und δx_f. Formal sei dabei das Innere der groben Domäne D_g ohne den Übergangsbereich der Breite δx_g gegeben als:

$$D_g^\circ := \{x \in D_g | \forall y \in \mathbb{R}^2 \setminus D_g : \|x - y\| > \delta x_g\}$$

Vergleiche hierzu den unschraffierten Bereich der Darstellung von D_g in Abbildung 9. Unter der Annahme, dass D_g das feine Simulationsgebiet D_f komplett umschließt, also der komplette Rand des feinen Gitters mit dem groben gekoppelt werden soll, fordern wir dann für den Rand des feinen Gitters:

$$\partial D_f \overset{!}{\subset} \partial D_g^\circ \tag{3.1}$$

Diese Einschränkung garantiert, dass die äußersten Gitterknoten des feinen Gitters sich soweit möglich mit groben Gitterknoten schneiden und nicht etwa zwischen zwei groben *Gitterreihen* liegen.

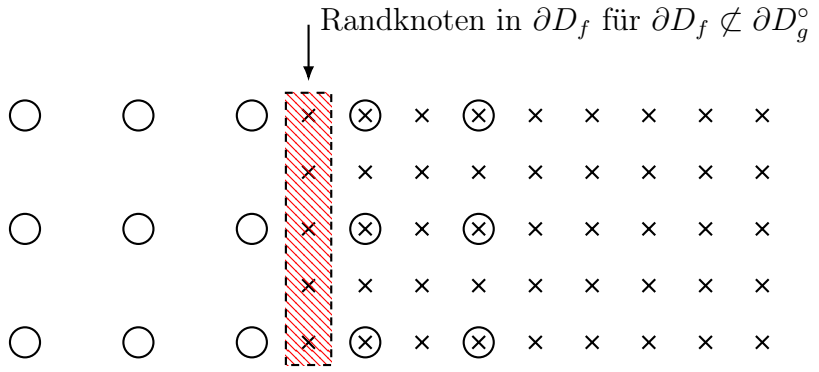


Abbildung 10: Mit Forderung (3.1) ausgeschlossener Übergangsbereich [17, vgl. Abb. 9]

Wir können die Gitterknoten der Übergangsbereiche nun detailliert klassifizieren:

Definition 3.3 (Gitterknoten der Übergangsbereiche)

$\mathcal{U}_g := \mathcal{G} \cap \mathcal{F}$	Grobe Knoten im Übergangsbereich
$\mathcal{U}_f := \mathcal{F} \cap (D_f \cap D_g)$	Feine Knoten im Übergangsbereich
$\mathcal{U}_{g\to f} := \partial D_f \cap \mathcal{U}_f$	Knoten mit Übertragung von grob nach fein
$\mathcal{U}_{f\to g} := \partial D_g \cap \mathcal{U}_g$	Knoten mit Übertragung von fein nach grob

Die Übertragungsrichtungen in $\mathcal{U}_{g\to f}$ und $\mathcal{U}_{f\to g}$ ergeben sich aus den jeweils fehlenden Verteilungsfunktionen an den Rändern der Gitter. So fehlen beispielsweise zur Kollision der groben Gitterknoten in $\mathcal{U}_{f\to g}$ Verteilungsfunktionen in Richtung des feinen Gitters, während die feinen Zellen in dieser Menge noch vollständig definiert sind, da sie im Inneren des feinen Gitters liegen.

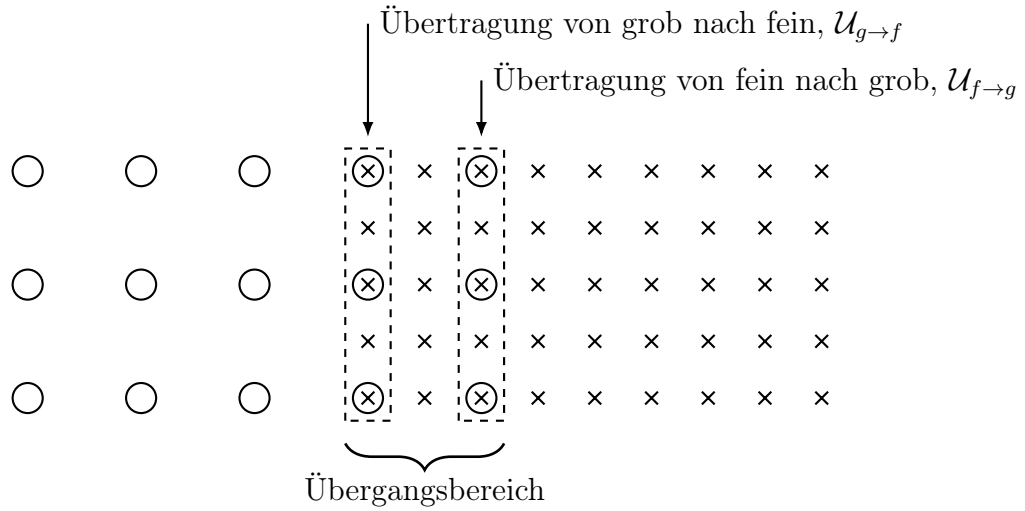


Abbildung 11: Skizze des Übergangsbereich [17, vgl. Abb. 4]

Mit diesem Argument lässt sich auch die Notwendigkeit eines Übergangsbereiches $\mathcal{U}_g \cup \mathcal{U}_f$ der Mindestbreite δx_g erklären: Gäbe es diesen nicht, so fehlten an der Grenze zwischen grobem und feinem Gitter Verteilungsfunktionen in beide Richtungen zugleich.

Anders als noch in Definition 2.3 betrachten wir die Gitter jetzt also nicht mehr unabhängig des darzustellenden physikalischen Modells, sondern unterscheiden anhand der physikalischen Auflösung δx. Während, im Kontext der LBM an sich, weiterhin für beide Gitter $\Delta x = 1$ gesetzt wird, führt die Relation von δx_g und δx_f im kommenden Kapitel 3.3.1 u.a. zu einer Relation zwischen grober und feiner Relaxationszeit.

Eine stringente Behandlung von Gitterkopplung in diesem Modell benötigt Abbildungen der physikalisch eingebetteten Knoten aus $\mathcal{G}$ und $\mathcal{F}$ in die zugehörigen *implementierenden* Gitter mit uniformer Auflösung $\Delta x = 1$. Diese stellen gerade die Definitionsmengen der groben bzw. feinen Verteilungsfunktionen dar.

Definition 3.4 (Abbildung auf implementierende Gitter) Sei $\# \in \{f, g\}$ Symbol des feinen oder groben Gitters. Dann können wir o.B.d.A. einen beliebigen physikalischen Knoten $x^{\text{phys}}_{0,\#}$ mit dem Knoten $x^{\text{impl}}_0 = 0 \in L$ identifizieren. Eine Bijektion zwischen physikalischen und implementierenden Gittern ist damit schon eindeutig definiert:

$$x^{\text{impl}}_{\#}(x^{\text{phys}}) := \frac{1}{\delta x_{\#}}(x^{\text{phys}} - x^{\text{phys}}_{0,\#})$$
$$x^{\text{phys}}_{\#}(x^{\text{impl}}) := x^{\text{phys}}_{0,\#} + \delta x_{\#} \cdot x^{\text{impl}}$$

Diese Abbildung der physikalisch eingebetteten Gitterknoten in die Definitionsmenge der Verteilungsfunktionen nehmen wir dabei zur Vereinfachung der Notation implizit an, wann immer die Verteilung in Elementen aus $\mathcal{G}$ oder $\mathcal{F}$ betrachtet wird.

Da die Einbettungen von Knoten verschiedener Auflösungen in deren Übergangsbereichen nicht disjunkt sind, benötigen wir im Weiteren gitterspezifische Bezeichnungen für die Verteilungen und deren Momente:

Definition 3.5 (Gitterspezifische Verteilungsfunktionen und Momente) Sei $\# \in \{f, g\}$ wieder Symbol des feinen oder groben Gitters. Wir bezeichnen dann mit $f_{\#,i}$ die i-te Verteilungsfunktion des entspechenden Gitters. Analog formulieren sich mit Definition 2.7 die Momente für $x \in \mathcal{G}$ respektive $x \in \mathcal{F}$:

$$\rho_{\#}(x) := \sum_{j=0}^{q-1} f_{\#,j}(x)$$

$$u_{\#}(x) := \frac{1}{\rho_{\#}(x)} \sum_{j=0}^{q-1} \xi_j f_{\#,j}(x)$$

Zusammenfassend wird die Aufgabe der im kommenden Kapitel zu erarbeitenden Skalierungs-, Restriktions- und Interpolationsschritte also *nur* darin bestehen, die jeweils fehlenden Verteilungsfunktionen möglichst gut zu rekonstruieren.

3.3 Komponenten der Gitterkopplung

3.3.1 Skalierung

Während die Skalierung räumlicher Größen durch die Festlegung des Verfahrens auf Übergänge im Verhältnis 1 : 2 definiert ist, eröffnen sich für die zeitliche Skalierung zwei Möglichkeiten: Konvektive oder diffusive Skalierung [16, Kap. 7.2.2.1]. Unterschied der beiden Ansätze ist dabei das jeweilige Verhältnis zwischen räumlicher und zeitlicher Auflösung.

Definition 3.6 (Konvektive Skalierung) Sei $\delta t > 0$ die zeitliche und $\delta x > 0$ die räumliche Diskretisierung. Dann gilt bei konvektiver Skalierung das Verhältnis:

$$\delta t \sim \delta x$$

Es besteht also eine lineare Proportionalität.

Definition 3.7 (Diffusive Skalierung) Sei $\delta t > 0$ die zeitliche und $\delta x > 0$ die räumliche Diskretisierung. Dann gilt bei diffusiver Skalierung das Verhältnis:

$$\delta t \sim \delta x^2$$

Es besteht hier also eine quadratische Proportionalität. Im Vergleich zu einer konvektiven Skalierung ist die zeitliche Auflösung somit um eine Ordnung feiner.

Es ist klar zu erkennen, dass diffusive Skalierung einen deutlich größeren numerischen Aufwand gegenüber der konvektiven Skalierung nach sich zieht. Vorteil der bei diffusiver Skalierung erhöhten Schrittanzahl pro Zeiteinheit sind kleinere Fehler. Darüber hinaus ist nach [16, S. 276] die asymptotische Konvergenz gegen die inkompressiblen Navier-Stokes Gleichungen nur bei diffusiver Skalierung gegeben.

Für die Autoren des hier erörterten Gitterverfeinerungsverfahrens überwog dabei das Argument der numerischen Effizienz, weshalb auch wir hier nun die konvektive Skalierung betrachten wollen. Die Austauschbarkeit des Skalierungsverfahrens sollte jedoch bei der Implementierung in OpenLB beachtet werden, da dieser Aspekt eine weitere prinzipiell flexible Komponente des Verfahrens darstellt.

Aus der Wahl von konvektiver Skalierung ergibt sich zunächst:

$$\frac{\delta t_g}{\delta x_g} = \frac{\delta t_f}{\delta x_f} \wedge \delta x_f = \frac{\delta x_g}{2} \implies \delta t_f = \frac{\delta t_g}{2} \tag{3.2}$$

Auf dem feinen Gitter müssen also doppelt so viele Zeitschritte wie auf dem groben Gitter durchgeführt werden. Geschwindigkeit, Druck und Dichte sind stetig im Gitterübergang. Dies gilt nicht für die kinetische Viskosität $\nu = c_s^2\tau$, was wir bei der Herleitung der feinen Relaxationszeit τ_f aus τ_g beachten müssen.

Definition 3.8 (Physikalische Reynolds-Zahl) Seien U die charakteristische Geschwindigkeit, L die charakteristische Länge und ν die kinetische Viskosität in physikalischen Einheiten. Dann ist die Reynolds-Zahl definiert als:

$$\mathrm{Re} := \frac{UL}{\nu}.$$

Definition 3.9 (Lattice Reynolds-Zahl) Sei $\# \in \{f, g\}$ Symbol des feinen oder groben Gitters. Seien $U_\# := \delta t_\# / \delta x_\# \cdot U$ die charakteristische Geschwindigkeit, $L_\# := 1/\delta x_\# \cdot L$ die charakteristische Länge und $\nu_\# := c_s^2 \tau_\#$ die kinetische Viskosität in Lattice-Einheiten. Dann ist die *Lattice* Reynolds-Zahl des feinen bzw. groben Gitters definiert als:

$$\mathrm{Re}_\# := \frac{U_\# L_\#}{\nu_\#} = \frac{\delta t_\# U L}{(\delta x_\#)^2 \nu_\#}.$$

Wir erzwingen nun mit $\mathrm{Re}_g = \mathrm{Re}_f$ die Unabhängigkeit von Reynolds-Zahl und Gitterauflösung. Diese Gleichsetzung ist sinnvoll, da die Reynolds-Zahl gerade die Vergleichbarkeit von Strömungen verschiedener Modellgrößen ermöglicht. Durch Einsetzen von Definition 3.9 erhalten wir eine Verknüpfung der Relaxationszeiten τ_f und τ_g:

$$\begin{aligned}
\mathrm{Re}_g = \mathrm{Re}_f &\iff \frac{\delta t_g U L}{(\delta x_g)^2 \nu_g} = \frac{\delta t_f U L}{(\delta x_f)^2 \nu_f} \\
&\iff \frac{\delta t_g}{(\delta x_g)^2 \nu_g} = \frac{\delta t_f}{(\delta x_f)^2 \nu_f} \\
&\iff \frac{\delta t_g}{(\delta x_g)^2 c_s^2 \tau_g} = \frac{\delta t_f}{(\delta x_f)^2 c_s^2 \tau_f} \\
&\iff \tau_f = \frac{\delta t_f (\delta x_g)^2}{(\delta x_f)^2 \delta t_g} \tau_g \\
&\iff \tau_f = 2\tau_g
\end{aligned} \tag{3.3}$$

Für die zur expliziten Lösung der diskreten LBM BGK Gleichung in Definition 2.8 verschobenen Relaxationszeiten ergibt sich somit:

$$\overline{\tau_f} = 2\overline{\tau_g} - \frac{1}{2} \tag{3.4}$$

Die Equilibriumsverteilung f_i^{eq} ergibt sich nach Definition 2.6 aus Geschwindigkeit u und Dichte ρ. Sie sind also explizit unabhängig der Gitterauflösung und, wie erwähnt, stetig im Gitterübergang. Diese Aussage gilt nicht für die Nicht-Equilibriumsverteilung f_i^{neq}, da diese nach (2.6) von dem Geschwindigkeitsgradienten ∇u abhängt.

Bezeichnen nun $f_{f,i}^{\mathrm{neq}}$ und $f_{g,i}^{\mathrm{neq}}$ die gitterspezifischen Nicht-Equilibriumanteile und S_f sowie S_g die entsprechenden Verzerrungstensoren. Zur Skalierung von $f_{f,i}^{\mathrm{neq}}$ suchen wir ein $\alpha \in \mathbb{R}$ s. d. gilt:

$$f_{f,i}^{\mathrm{neq}} = \alpha f_{g,i}^{\mathrm{neq}} \tag{3.5}$$

Mit Hilfe von (2.6) lässt sich diese Relation nach α auflösen:

$$\begin{aligned} f_{f,i}^{\mathrm{neq}} = \alpha f_{g,i}^{\mathrm{neq}} &\iff -\frac{w_i \rho \overline{\tau_f}}{c_s^2} \mathrm{Q}_i : \mathrm{S}_f = -\alpha \left(\frac{w_i \rho \overline{\tau_g}}{c_s^2} \mathrm{Q}_i : \mathrm{S}_g \right) \\ &\iff \overline{\tau_f} \mathrm{Q}_i : \mathrm{S}_f = \alpha \overline{\tau_g} \mathrm{Q}_i : \mathrm{S}_g \\ &\iff \overline{\tau_f} \delta t_f \mathrm{Q}_i : \mathrm{S} = \alpha \overline{\tau_g} \delta t_g \mathrm{Q}_i : \mathrm{S} \\ &\iff \alpha = \frac{\delta t_f}{\delta t_g} \frac{\overline{\tau_f}}{\overline{\tau_g}} \end{aligned}$$

Einsetzen der Relationen (3.2) und (3.4) reduziert den Skalierungsfaktor auf einen nur von der groben Relaxationszeit abhängigen Ausdruck:

$$\begin{aligned} \alpha &= \frac{\delta t_f}{\delta t_g} \frac{\overline{\tau_f}}{\overline{\tau_g}} \\ &= \frac{1}{2} \frac{2\overline{\tau_g} - \frac{1}{2}}{\overline{\tau_g}} \\ &= 1 - \frac{1}{4\overline{\tau_g}} \end{aligned} \tag{3.6}$$

Schließlich erhalten wir so die folgende Relation der Nicht-Equilibriumsverteilungen:

$$f_{f,i}^{\mathrm{neq}} = \left(1 - \frac{1}{4\overline{\tau_g}} \right) f_{g,i}^{\mathrm{neq}} \tag{3.7}$$

Insgesamt haben wir hiermit die Skalierung der Diskretisierungen in Raum und Zeit, der Relaxationszeit sowie der Nicht-Equilibriumsverteilung zwischen den Gittern $\mathcal{F}$ und $\mathcal{G}$ hergeleitet.

Seien $x_{f\to g} \in \mathcal{U}_{f\to g}$ und $x_{g\to f} \in \mathcal{U}_{g\to f}$ die Knoten aus dem Übergangsbereich mit Übertragung von fein nach grob bzw. von grob nach fein. Dann gelten bei Erinnerung an die implizite Knotenabbildung 3.4:

$$f_{g,i}(x_{f\to g}) = f_i^{\mathrm{eq}}(\rho(x_{f\to g}), u(x_{f\to g})) + \left(1 - \frac{1}{4\overline{\tau_g}} \right)^{-1} f_{f,i}^{\mathrm{neq}}(x_{f\to g}) \tag{3.8}$$

$$f_{f,i}(x_{g\to f}) = f_i^{\mathrm{eq}}(\rho(x_{g\to f}), u(x_{g\to f})) + \left(1 - \frac{1}{4\overline{\tau_g}} \right) f_{g,i}^{\mathrm{neq}}(x_{g\to f}) \tag{3.9}$$

Die zusammengesetzten Verteilungsfunktionen von Übergangsknoten des einen Gitters lassen sich also durch Skalierung des Nicht-Equilibriumanteils der Verteilungsfunktionen des jeweils anderen Gitters rekonstruieren. Leider reicht dies noch nicht zur vollständigen Beschreibung eines Gitterverfeinerungsverfahrens, da nicht für alle feinen Gitterknoten im Übergangsbereich passende grobe Gitterknoten existieren – vgl. dazu Abbildung 11. Auch der Übergang von fein nach grob gestaltet sich trotz passenden feinen Knoten potenziell komplizierter als eine bloße Skalierung, wie wir im nächsten Kapitel sehen werden.

3.3.2 Restriktion

Kraft seiner höheren Auflösung enthält das feine Gitter mehr Informationen als das umgebende grobe Gitter. Der Übergang von fein nach grob stellt also eine Restriktion der Verteilungsfunktionen dar.

Konkret suchen wir nach einer sinnvollen Definition der in $x_{f\to g} \in \mathcal{U}_{f\to g}$ fehlenden Verteilungsfunktionen $f_{g,i}$. Eine Solche ergibt sich aus der skalierten Dekomposition 3.8 durch Ersetzen der einfachen Nicht-Equilibriumsverteilung $f^{\text{neq}}_{f,i}(x_{f\to g})$ mit einer restringierten Variante ebendieser.

$$f_{g,i}(x_{f\to g}) = f^{\text{eq}}_i(\rho(x_{f\to g}), u(x_{f\to g})) + \alpha^{-1}\, \boldsymbol{r}(i, x_{f\to g}) \tag{3.10}$$

Wir bemerken an dieser Stelle, dass nur die Nicht-Equilibriumsverteilung durch eine Restriktionsoperation eingeschränkt wird, während der Equilibriumanteil unangetastet bleibt. Dies ist damit zu begründen, dass Dichte und Geschwindigkeit bei der von uns verwendeten konvektiven Skalierung im Gitterübergang stetig bleiben.

Die skalierte Dekomposition 3.8 lässt sich in der Schreibweise von 3.10 formulieren, wenn die Identität als Restriktionsoperation eingesetzt wird:

$$\boldsymbol{r}(i, x_{f\to g}) := f^{\text{neq}}_{f,i}(x_{f\to g})$$

Die für unser Verfahren [17, Kap. 3.3] beschriebene Restriktion ist der Mittelwert aller umliegenden gerichteten Nicht-Equilibriumanteilen:

$$\boldsymbol{r}(i, x_{f\to g}) := \frac{1}{q}\sum_{j=0}^{q-1} f^{\text{neq}}_{f,i}(x_{f\to g} + \delta x_f \xi_j) \tag{3.11}$$

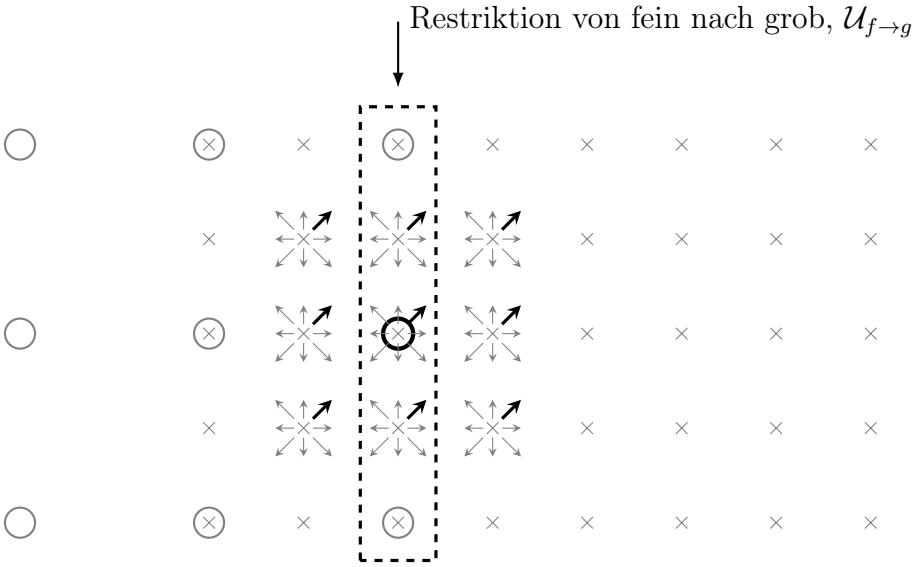


Abbildung 12: Einzugsgebiet der Restriktionsoperation $\boldsymbol{r}(7, x_{f\to g})$

3.3.3 Interpolation

Zunächst ergänzen wir die Gitterteilmengen aus Definition 3.3 um eine Unterscheidung zwischen alleinstehenden feinen Knoten und solchen für die ein, der Übertragung von grob nach fein dienlicher, grober Knoten existiert.

Definition 3.10 (Gitterknoten mit Übertragung von grob nach fein)

$\mathcal{U}^g_{g\to f} := \mathcal{U}_{g\to f} \cap \mathcal{U}_g$ Doppelte Knoten mit Übertragung von grob nach fein

$\mathcal{U}^f_{g\to f} := \mathcal{U}_{g\to f} \setminus \mathcal{U}_g$ Alleinstehende feine Knoten mit Übertragung von grob nach fein

Für $x^g_{g\to f} \in \mathcal{U}^g_{g\to f}$ gilt insbesondere $x^g_{g\to f} \in \mathcal{U}_g \cap \mathcal{U}_f$. Es existieren in diesen Gitterpunkten also vollständig definierte grobe Verteilungsfunktionen, die wir zur Bestimmung der Momente ρ und u in (3.9) heranziehen können. Entsprechend besitzen wir zur Interpolation des gesuchten Wertes geschickterweise eine Stützstelle an eben dessen Position. Die *Interpolation* von $x^g_{g\to f}$ beschränkt sich folglich auf die Skalierung (3.9):

$$f_{f,i}(x^g_{g\to f}) = f^{\mathrm{eq}}_i(\rho_g(x^g_{g\to f}), u_g(x^g_{g\to f})) + \alpha f^{\mathrm{neq}}_{g,i}(x^g_{g\to f}) \tag{3.12}$$

Für $x^f_{g\to f} \in \mathcal{U}^f_{g\to f}$ gilt insbesondere $x^f_{g\to f} \notin \mathcal{U}_g$. Es existieren in diesen Gitterpunkten also im Gegensatz zur Situation 3.12 keine groben Verteilungsfunktionen. Die fehlenden Werte zur Bestimmung der Momente sowie des Nicht-Equilibriumanteils in (3.9) müssen hier also aus den umliegenden groben Verteilungsfunktionen interpoliert werden:

$$f_{f,i}(x^f_{g\to f}) = f^{\mathrm{eq}}_i(\boldsymbol{n}_{\rho_g}(x^f_{g\to f}), \boldsymbol{n}_{u_g}(x^f_{g\to f})) + \alpha \boldsymbol{n}_{f^{\mathrm{neq}}_{g,i}}(x^f_{g\to g}) \tag{3.13}$$

Die unbekannten Werte der Moment- und Nicht-Equilibriumfunktionen werden in diesem Ausdruck durch eine Interpolationsoperation $\boldsymbol{n}$ genähert. Neben der Art der Restriktion $\boldsymbol{r}$ stellt die Wahl des Interpolationsverfahrens einen weiteren zentralen und flexiblen Bestandteil des, auf diesen Seiten nachvollzogenen, Gitterverfeinerungsverfahren dar.

Stützstellen für die Interpolation seien hier die parallel zum Gitterübergang liegenden groben Nachbarknoten des gesuchten Punktes $x^f_{g\to f}$. Wir adressieren diese, parallel zu einem Einheitsvektor v positionierten, Stützknoten mit:

$$\mathcal{N}_{x^f_{g\to f}} := \left\{ x \in \mathcal{G} \middle| x = x^f_{g\to f} + j\,\delta x_f\, v,\ j \in \mathbb{Z} \right\} \subseteq \mathcal{U}^g_{g\to f}$$

Bekannte Stützwerte von $\boldsymbol{n}_\star(x)$ befinden sich also relativ zum gesuchten Knoten $x^f_{g\to f}$ in, um ungerade Vielfache der feinen Schrittweite δx_f skalierten, Verschiebungen entlang der normierten Übergangsparallele v (vgl. Abbildungen 13 und 14). Die Einschränkung der hinzugezogenen Stützen auf Knoten, welche parallel zum Übergang liegen, wurde von Lagrava et al. so gewählt [17, Kap. 3.6], um in 2D ein eindimensionales Interpolationsproblem zu erhalten. Prinzipiell spricht nichts gegen eine Einbeziehung umfangreicherer Teilmengen der groben Knotennachbarschaft.

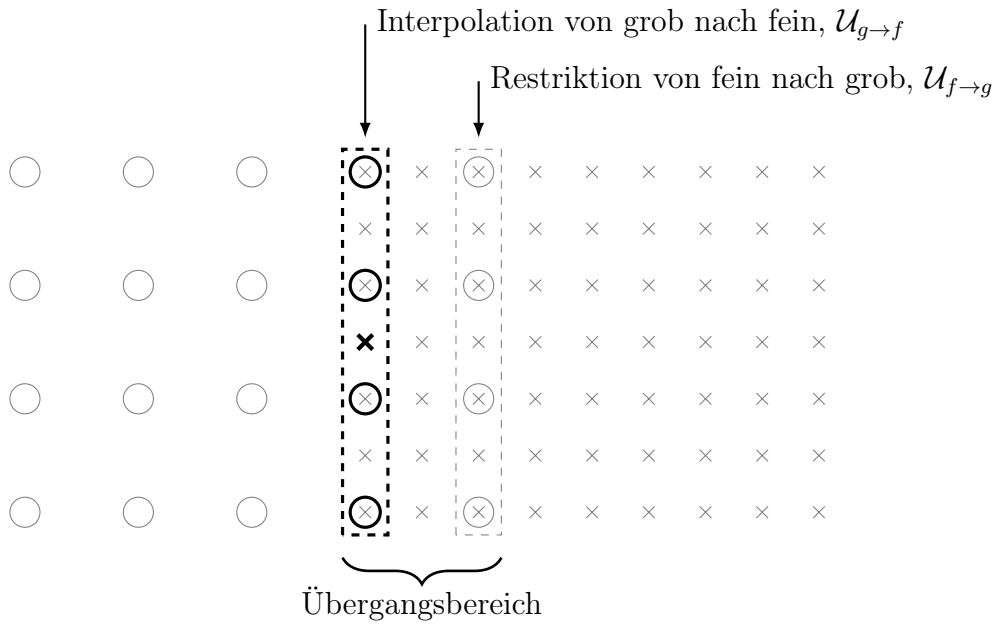


Abbildung 13: Stützstellen der Interpolation im Übergangsbereich

Um die kommenden Ausführungen auf das Wesentliche – namentlich das Verfahren unabhängig der konkret zu interpolierenden Funktion – zu konzentrieren, sei definiert:

$$\overline{\boldsymbol{n}}(h) := \boldsymbol{n}_\star(x^f_{g\to f} + h\,\delta x_f\,v) \text{ für Zielfunktion } \star \in \{\rho_g, u_g, f^{\text{neq}}_{g,i}\}$$

In dieser Formulierung suchen wir also eine möglichst gute Interpolation des Wertes in $\overline{\boldsymbol{n}}(0)$ anhand der Stützstellen $\overline{\boldsymbol{n}}(h)$ für kleine $h \in \mathbb{Z} \setminus 2\mathbb{Z}$. Ein nahe liegender Ansatz hierfür ist das arithmetische Mittel der beiden engsten Nachbarn:

$$\boldsymbol{n}_\star(x^f_{g\to f}) = \overline{\boldsymbol{n}}(0) = \frac{\overline{\boldsymbol{n}}(-1) + \overline{\boldsymbol{n}}(1)}{2} \tag{3.14}$$

Vorteil eines solch einfachen Verfahrens wäre, dass die benötigten groben Nachbarn auch an den Ecken des Übergangsbereiches existieren (vgl. Abbildung 15) und daher keine Sonderbehandlung erforderlich würde.

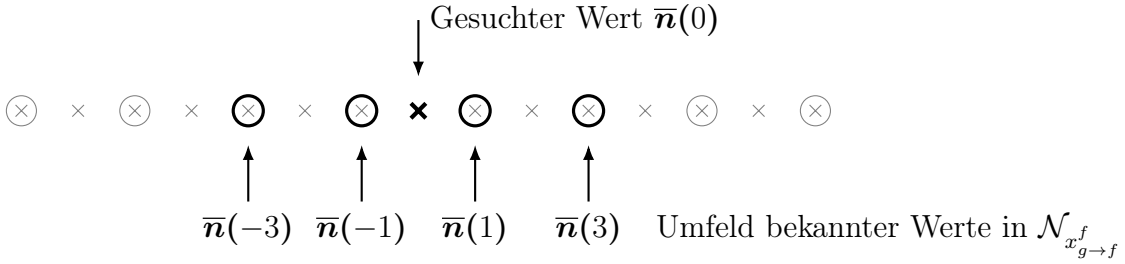


Abbildung 14: Stützstellen der Interpolation im Detail

Bessere Näherungen können unter Einsatz weiterer Stützknoten erzielt werden. Wir berechnen dazu mit dem Schema der dividierten Differenzen die Faktoren der Newtonschen Interpolationsformel [2, IV.3 (3.10)] auf den in Abbildung 14 dargestellten Stützpunkten $(-3, \overline{\boldsymbol{n}}(-3))$, $(-1, \overline{\boldsymbol{n}}(-1))$, $(1, \overline{\boldsymbol{n}}(1))$ und $(3, \overline{\boldsymbol{n}}(3))$:

$$\begin{aligned}\overline{\boldsymbol{n}}(x) :&= \overline{\boldsymbol{n}}(-3)\\ &+ \frac{1}{2}(\overline{\boldsymbol{n}}(-1) - \overline{\boldsymbol{n}}(-3))(x+3)\\ &+ \frac{1}{8}(\overline{\boldsymbol{n}}(1) - \overline{\boldsymbol{n}}(-1) + \overline{\boldsymbol{n}}(3))(x+3)(x+1)\\ &+ \frac{1}{48}(\overline{\boldsymbol{n}}(3) - 3\overline{\boldsymbol{n}}(1) + 3\overline{\boldsymbol{n}}(-1) - \overline{\boldsymbol{n}}(-3))(x+3)(x+1)(x-1)\end{aligned}$$

Ausgewertet in 0 erhalten wir folgenden Ausdruck als Näherung von $\boldsymbol{n}_\star(x^f_{g\to f})$:

$$\overline{\boldsymbol{n}}(0) = \frac{9}{16}(\overline{\boldsymbol{n}}(-1) + \overline{\boldsymbol{n}}(1)) - \frac{1}{16}(\overline{\boldsymbol{n}}(-3) + \overline{\boldsymbol{n}}(3)) \tag{3.15}$$

Hierbei handelt es sich um ein Verfahren vierter Ordnung, wie sich durch Einsetzen der Taylor-Entwicklung von $\overline{\boldsymbol{n}}$ um 0 in die Auswertung (3.15) zeigen lässt:

$$\begin{aligned}&\overline{\boldsymbol{n}}(h) = \overline{\boldsymbol{n}}(0) + \overline{\boldsymbol{n}}^{(1)}(0)h + \frac{1}{2}\overline{\boldsymbol{n}}^{(2)}(0)h^2 + \frac{1}{6}\overline{\boldsymbol{n}}^{(3)}(0)h^3 + \mathcal{O}(h^4) \qquad (3.16)\\ \Longrightarrow\ &\frac{9}{16}(\overline{\boldsymbol{n}}(-1) + \overline{\boldsymbol{n}}(1)) - \frac{1}{16}(\overline{\boldsymbol{n}}(-3) + \overline{\boldsymbol{n}}(3)) \overset{(3.16)}{=} \overline{\boldsymbol{n}}(0) + \mathcal{O}(h^4)\end{aligned}$$

In Abbildung 15 erkennen wir zwei mögliche Randfälle des Gitterübergangs, welche zu Problemen bei Nutzung des Interpolationsverfahren vierter Ordnung führen können.

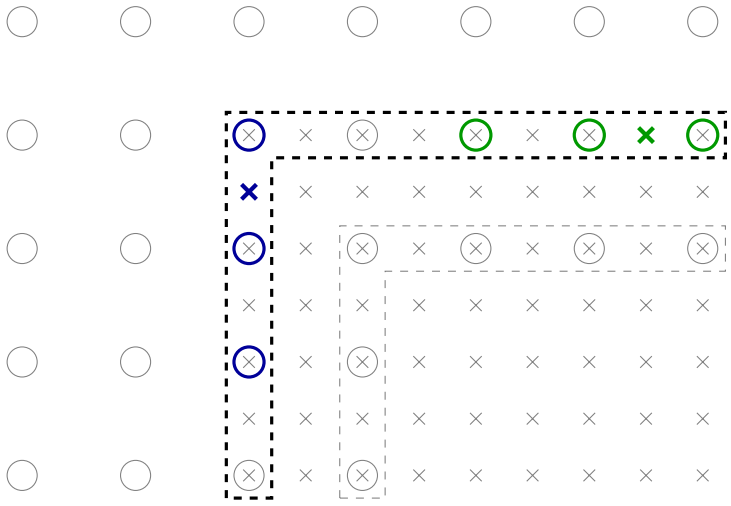


Abbildung 15: Interpolation in Ecken und Enden des feinen Gitters

So kann es an den Außengrenzen des Simulationsgebietes dazu kommen, dass nur drei der vier benötigten groben Nachbarknoten zur Verfügung stehen, wie in der grün markierten Situation dargestellt. Je nach Implementierung der Kommunikation zwischen den Gittern, kann die gleiche Einschränkung auch in Ecken des feinen Gitters – hier blau markiert – dazu kommen, dass eine Interpolation auf Grundlage von nur drei Nachbarn benötigt wird. Entsprechend erhalten wir nach Berechnung der dividierten Differenzen ein Interpolationspolynom auf drei Stützstellen:

$$\begin{aligned}\overline{\boldsymbol{n}}(x) :&= \overline{\boldsymbol{n}}(-1) \\ &+ \frac{1}{2}(\overline{\boldsymbol{n}}(1) - \overline{\boldsymbol{n}}(-1))(x+1) \\ &+ \frac{1}{8}(\overline{\boldsymbol{n}}(3) - 2\overline{\boldsymbol{n}}(1) + \overline{\boldsymbol{n}}(-1))(x+1)(x-1)\end{aligned}$$

Auch in dieser Situation ist nur der Wert in 0 zur Näherung von $\boldsymbol{n}_\star(x^f_{g\to f})$ von Interesse:

$$\overline{\boldsymbol{n}}(0) = \frac{3}{8}\overline{\boldsymbol{n}}(-1) + \frac{3}{4}\overline{\boldsymbol{n}}(1) - \frac{1}{8}\overline{\boldsymbol{n}}(3) \tag{3.17}$$

Passend zur Anzahl der Stützstellen präsentiert sich dieses Verfahren nach erneutem Einsetzen der Taylor-Entwicklung (3.16) als eine Interpolationsformel dritter Ordnung:

$$\frac{3}{8}\overline{\boldsymbol{n}}(-1) + \frac{3}{4}\overline{\boldsymbol{n}}(1) - \frac{1}{8}\overline{\boldsymbol{n}}(3) \overset{(3.16)}{=} \overline{\boldsymbol{n}}(0) + \mathcal{O}(h^3)$$

Trotz Behandlung dieses Sonderfalls werden die höheren Approximationsordnungen gegenüber (3.14) weiterhin und unumgänglich durch zusätzliche Stützstellen *erkauft*, welche bei parallelisierten LBM-Umsetzungen kommuniziert werden müssen. Es gilt also, zwischen Güte der Interpolation und Anzahl sowie Position der Stützstellen abzuwiegen.

3.4 Algorithmus

Im zurückliegenden Kapitel 3.3 haben wir, aufbauend auf der Skalierung von Verteilungsfunktionen zwischen Gitterauflösungen, die Restriktion von fein nach grob sowie die Interpolation von grob zu fein nachvollzogen. Die somit erfassten wesentlichen Grundlagen des Verfeinerungsverfahrens gilt es nun in einem *Kopplungsalgorithmus* [17, Kap. 3.5] zusammenzuführen.

Entsprechend (3.2) müssen für jeden groben Zeitschritt δt_g zwei feine Zeitschritte δt_f durchgeführt werden. Die alternierenden Kollisions- und Strömungsschritte der beiden Gitter sind also strikt gekoppelt und werden als eine Schleifeneinheit betrachtet. Als Schleifeninvariante definieren wir dabei die vollständige Bekanntheit aller Verteilungsfunktionen aller Knoten in beiden Gittern.

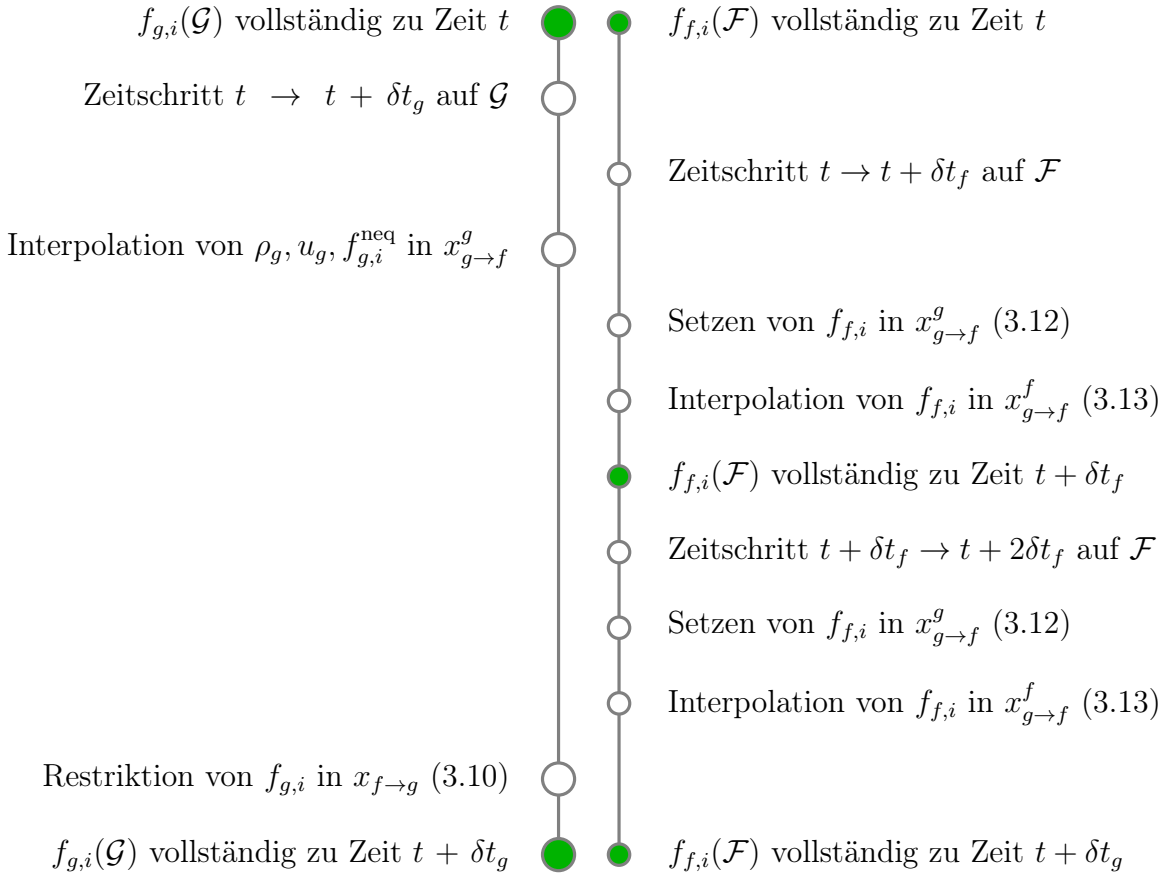


Abbildung 16: Übersicht des Verfeinerungsalgorithmus mit Invariante

Aufbauend auf dieser Invariante ergibt sich die, in Abbildung 16 dargelegte, Reihenfolge der erforderlichen Schritte direkt aus den, für die einzelnen Komponenten der Gitterkopplung benötigen, Informationen. So sind zu Beginn alle Verteilungsfunktionen vollständig bekannt, was die Ausführung eines üblichen Kollisions- und Strömungsschritts (vgl. Kapitel 2.1.1) in beiden Gittern ohne weitere Zuarbeit erlaubt. Nach diesen beiden

Schritten fehlen Verteilungsfunktionen $f_{g,i}(x_{f\to g})$ zur Wiederherstellung der Invariante des groben Gitters. Auch der benötigte zweite Simulationsschritt, um $\mathcal{F}$ auf Zeitpunkt $t+\delta t_g = t+2\delta t_f$ zu bringen, scheitert zunächst an der Unbestimmtheit von Verteilungsfunktionen $f_{f,i}(x_{g\to f})$.

$\mathcal{U}_{g\to f}$ zu Zeit $t+\delta t_f$ und $t+\delta t_g$: $\mathcal{U}_{f\to g}$ zu Zeit $t+\delta t_g$:

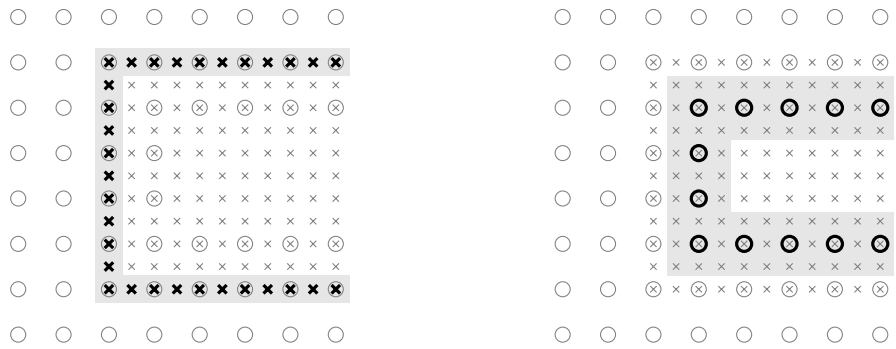


Abbildung 17: Übersicht der zu vervollständigenden Knoten

Vervollständigung von $\mathcal{F}$ zu Zeitpunkt $t+\delta t_f$: Zur Vervollständigung des feinen Gitters nach dem ersten Zeitschritt müssen die fehlenden Verteilungen aus dem groben Gitter rekonstruiert werden. Um die dazu erarbeiteten Kopplungen (3.12) und (3.13) anzuwenden, fehlen jedoch Werte der groben Stützstellen $f_{g,i}(x^g_{g\to f})$ zu Zeitpunkt $t+\delta t_f$. Diese sind zwar in den gesuchten Punkten, dank Trennung der Kopplungsrichtungen durch den Übergangsbereich, nach jedem Simulationsschritt direkt vollständig vorhanden – jedoch nur zu Zeit t und $t+\delta t_g$. Hier findet sich eine Anwendung des Interpolationsverfahrens zweiter Ordnung (3.14) zur linearen Zeitinterpolation der benötigten Werte von ρ_g, u_g und $f^{\text{neq}}_{g,i}$:

$$\star(x, t+\delta t_f) \approx \frac{\star(x,t+\delta t_g)+\star(x,t)}{2} \text{ für } \star \in \{\rho_g, u_g, f^{\text{neq}}_{g,i}\}, x \in \mathcal{G}$$

Aufbauend darauf steht der Anwendung der Kopplungsformeln (3.12) und (3.13) zur Vervollständigung von $\mathcal{F}$ nichts mehr im Wege:

$$\begin{aligned} f_{f,i}(x^g_{g\to f}, t+\delta t_f) &= f^{\text{eq}}_i(\rho_g(x^g_{g\to f}, t+\delta t_f), u_g(x^g_{g\to f}, t+\delta t_f)) + \alpha f^{\text{neq}}_{g,i}(x^g_{g\to f}, t+\delta t_f) \\ f_{f,i}(x^f_{g\to f}, t+\delta t_f) &= f^{\text{eq}}_i(\boldsymbol{n}_{\rho_g}(x^f_{g\to f}), \boldsymbol{n}_{u_g}(x^f_{g\to f})) + \alpha \boldsymbol{n}_{f^{\text{neq}}_{g,i}}(x^f_{g\to g}) \end{aligned}$$

Der Interpolationsoperator vierter Ordnung (3.15) löst sich dabei für die Zielfunktionen $\star \in \{\rho_g, u_g, f^{\text{neq}}_{g,i}\}$ und die Übergangsparallele v wie folgt auf:

$$\begin{aligned} \boldsymbol{n}_\star(x^f_{g\to f}) =& \frac{9}{16}(\star(x^f_{g\to f} - \delta x_f v, t+\delta t_f) + \star(x^f_{g\to f} + \delta x_f v, t+\delta t_f)) \\ &+ \frac{1}{16}(\star(x^f_{g\to f} - 3\delta x_f v, t+\delta t_f) + \star(x^f_{g\to f} + 3\delta x_f v, t+\delta t_f)) \end{aligned}$$

Vervollständigung von $\mathcal{F}$ zu Zeitpunkt $t + \delta t_g$: Dieser zweite Rekonstruktionsschritt auf dem feinen Gitter gestaltet sich einfacher, da die benötigten groben Verteilungen in $\mathcal{U}_{g\to f}$ zur Zeitpunkt $t+\delta t_g$ bereits durch den initialen Simulationsschritt auf dem groben Gitter bekannt sind. Entsprechend können die Kopplungsformeln (3.12) und (3.13) direkt zur Vervollständigung von $\mathcal{F}$ angewandt werden.

Vervollständigung von $\mathcal{G}$ zu Zeitpunkt $t + \delta t_g$: Nach zweimaliger Vervollständigung des feinen Gitters verbleibt zur Wiederherstellung der Schleifeninvariante der Abschluss des eingehenden Kollisions- und Strömungsschritts auf dem groben Gitter durch Restriktion der aus Richtung des feinen Gitters eingehenden Verteilungsfunktionen. Hierzu erlaubt die, durch die zuvorkommenden Schritte garantierte, Vollständigkeit des feinen Gitters zu Zeitpunkt $t + \delta t_g$, die direkte Anwendung der Kopplungsformel (3.10) mit Restriktionsoperator (3.11) auf die Knoten in $\mathcal{U}_{f\to g}$.

$$\begin{aligned} f_{g,i}(x_{f\to g}, t + \delta t_g) = f_i^{\mathrm{eq}}(\rho_f(x_{f\to g}, t + \delta t_g), u_f(x_{f\to g}, t + \delta t_g)) \\ + \frac{1}{\alpha}\frac{1}{q}\sum_{j=0}^{q-1} f_{f,i}^{\mathrm{neq}}(x_{f\to g} + \delta x_f \xi_j, t + \delta t_g) \end{aligned}$$

Zu erwähnen bleibt, dass wir aus Konsistenzgründen alle Kopplungsformeln immer auf alle – und nicht nur die fehlenden – diskreten Richtungen $i \in [q-1]$ einer betrachteten Zelle x anwenden.

Nach Durchführung der drei Vervollständigungsschritte haben wir die Invariante für $t + \delta t_g$ wieder hergestellt und können von vorne beginnen. Wir haben damit an dieser Stelle das Verfeinerungsverfahren von Lagrava et al. vollständig nachvollzogen und können mit der Implementierung in OpenLB fortfahren.

4 Implementierung in OpenLB

OpenLB [15] ist ein umfangreiches frei verfügbares C++ Framework zur Implementierung von LBM basierenden Simulationen. Schwerpunkte sind dabei eine große Flexibilität in Hinblick auf die unterstützten LB-Modelle sowie weitreichende Modularisierung zur einfachen Umsetzung neuer Anwendungen bei gleichzeitiger Eignung für hochperformante Berechnungen durch Skalierbarkeit auf parallele Großrechner.

Eine LBM Bibliothek mit idealer Unterstützung für Gitterverfeinerung würde es erlauben, dass zu lösende physikalische Problem zunächst ohne Rücksicht auf die konkrete Zusammensetzung des Gitters zu modellieren. Entstünde dann im Konflikt zwischen Rechenressourcen und angestrebter Ergebnisqualität ein Bedarf für Gitterverfeinerung, sollte diese a posteriori ohne weitere Anpassungen hinzugeschaltet werden können. In einer solchen Umgebung wäre es dann sogar denkbar, die Entscheidung über Position und Ausmaß der Verfeinerung an ein automatisches Kriterium [18] zu übertragen.

Diese Vision ist selbstredend von großem Anspruch und weit außerhalb der Grenzen dieser Arbeit. Ein erster Schritt muss jedoch die Ergänzung von OpenLB um grundlegende Unterstützung für manuelle Gitterverfeinerung sein – d. h. Position, Größe und Auflösung des zu verfeinernden Bereichs müssen durch den Nutzer vorgegeben werden können.

4.1 Architekturübersicht

OpenLB verwaltet die Diskretisierung der zu modellierenden Simulationsgebiete in einer `CuboidGeometry2D` Instanz. Diese Klasse teilt ein gegebenes physikalisches Gebiet in eine beliebige Anzahl von, durch `Cuboid2D` Instanzen dargestellten, achsenparallelen Rechtecken mit definierter Auflösung ein. Diese Aufteilung dient der Parallelisierung, da die Gitter dieser einzelnen Quader außerhalb von kommunizierenden Randbereichen vollständig nebenläufig verarbeitet werden können. Entsprechend werden diese Teilbereiche von einer Implementierung der `LoadBalancer` Schnittstelle auf die zu Verfügung stehenden Prozessoren aufgeteilt.

Auf Grundlage dieses räumlichen Umrisses wird dann das eigentliche Gitter in einer `SuperLattice2D` Instanz abhängig des gewählten Lattice Boltzmann Modells aufgebaut. Die einzelnen Gitterzellen werden dabei von `Cell` Klassen abgebildet, welche anhand des spezifischen, durch einen sogenannten Deskriptor *beschriebenen* und von `Dynamics` Instanzen durchgeführten, Kollisionsschrittes der lokalen Umsetzung des LBM Modells Sorge tragen.

Diese `Dynamics` beschreiben dabei zusammen mit zellübergreifenden Postprozessoren nicht nur das einfache Strömungsverhalten, sondern modellieren auch die zahlreichen Randkonditionen, welche die Abbildung komplexer Geometrien in LBM erst ermöglichen. Zur einfacheren Zuordnung dieser zellspezifischen Klassen verwendet OpenLB sogenannte Materialzahlen, welche von einer `SuperGeometry2D` Instanz verwaltet werden.

Parallel zu diesen Strukturen kapselt und berechnet die `UnitConverter` Klasse etwaig benötigte Konstanten zur Konvertierung zwischen physikalischen Einheiten und den,

diese modellierenden, Lattice-Einheiten sowie Relaxationszeiten und Fluidkonstanten wie die Reynolds-Nummer.

Im Allgemeinen ergibt sich aus diesen Komponenten folgende übliche Struktur von OpenLB-basierenden Anwendungen [24, Kap. 2.1]:

1. Erstellung des `UnitConverter` mit den beabsichtigten Gitterkonstanten
2. Beschreibung des Simulationsgebietes durch Konstruktion einer `CuboidGeometry2D`
3. Bereitstellung eines `LoadBalancer` zur Instantierung einer `SuperGeometry2D`
4. Definition der Materialzahlen in einer `prepareGeometry` Methode
5. Konstruktion der `SuperLattice2D` Instanz aus der `SuperGeometry2D`
6. Instantierung der benötigten `Dynamics` und etwaigen Randkonditionen
7. Bindung von `Dynamics` und Randkonditionen an die, von `SuperLattice2D` verwalteten, `Cell` Objekte anhand der Materialzahlen in einer `prepareLattice` Methode
8. Simulationsschleife zum Aufruf von `SuperLattice2D::collideAndStream`

In letzterem, die eigentliche Simulation durchführendem, Schritt, werden weiter durch kanonisch benannte Funktionen wie `getResults` und `error` die Ergebnisse zur Analyse in Dateien [25] geschrieben, Fehlernormen berechnet und Konvergenzkriterien bestimmt.

4.2 Auswahl der Verfeinerungsmethode

Ein erster Gedanke zur Integration von Gitterverfeinerung in OpenLB ist die Nutzung der bestehenden Dekomposition des Simulationsgebiets in achsenparallele Rechtecke. Insbesondere aus Sicht des Einfügens von Gitterverfeinerung in die existierende Architektur, sowie der unveränderten Weiterverwendung der `LoadBalancer` Algorithmen zur Steuerung der Parallelisierung, scheint ein solcher Ansatz sinnvoll.

Bei Variation der Auflösung einzelner Quader im Rahmen der `CuboidGeometry2D` Struktur handelte es sich zwangsweise um einen Multi-Domain Ansatz. Gingen wir diesen Weg, benötigten wir zunächst `Cuboid2D` spezifische `UnitConverter` Instanzen zur Verwaltung der auflösungsabhängigen Konstanten. Dies müsste dann im Rahmen von `prepareLattice` zur Setzung der dann ebenfalls quaderspezifischen `Dynamics` und Randkonditionen beachtet werden.

Zur Ermöglichung von Parallelisierung berücksichtigt die, der Gitterverwaltung in `SuperLattice2D` zugrunde liegende, Aufteilung der Domäne durch `CuboidGeometry2D` bereits Übergangsbereiche, deren Funktion mit zusätzlicher Auflösungskopplung in Einklang zu bringen wäre.

Weiter würde das Problemfeld eben dieser Dekomposition um die Restriktion auf Auflösungsübergänge im Verhältnis $1 : 2$ erweitert. So müsste ein guter Algorithmus zur Aufteilung des Simulationsgebietes die Anforderungen an Parallelisierung, Verteilung der

Rechenressourcen, Geometrie und Verfeinerung sinnvoll auflösen und zugleich manuelle Eingriffe erlauben. Diese zusätzliche starke Einschränkung sowie der dann bei Anpassung der Verfeinerungsstruktur unumgängliche komplette Neuaufbau der Simulation bilden ein starkes Gegenargument zu diesem ersten Gedanken.

Der tatsächlich umgesetzte Ansatz ergibt sich aus dem Verständnis von Gitterverfeinerung als Kopplung von ansonsten komplett allein stehenden Simulationen. Die Übergangsbereiche wären in diesem Modell mit Randkonditionen vergleichbar, wie sie für Ein- und Ausflüsse verwendet werden. Gitterverfeinerung könnte so weitestgehend von der bestehenden Architektur getrennt ergänzt werden, was insbesondere auch die veränderungsfreie Unterstützung existierender Anwendungen begünstigen würde.

Eine solche nebenläufige Überlagerung von Simulationen mit jeweils komplett eigenständig verwalteten Gittern gebietet sich bei erster Betrachtung als klarer Umriss eines Multi-Grid Verfahrens. Beachten wir jedoch, dass es einfach möglich ist, die überlagerten Gitterflächen durch *Nullen* der entsprechenden Materialzahlen effizient aus der Verarbeitung auszuschließen und trotzdem bei Bedarf – z.B. in Hinblick auf Verschiebung von verfeinerten Bereiche während des Simulationsverlaufs – zu reaktivieren, stellen sich auch Multi-Domain Ansätze in diesem Modell als sinnvoll implementierbar heraus. Vorteil ist hier also gerade auch, dass prinzipiell beide Ansätze zur Gitterverfeinerung umgesetzt werden können und wir nicht durch Festhalten an der existierenden Struktur auf Multi-Domain Verfahren beschränkt sind. Da die Positionierung der Gitter in diesem Ansatz komplett frei wäre, ließen sich aus Architektursicht auch nicht-koinzidierende oder sogar zueinander rotierte Verfeinerungsgitter abbilden.

Ein Vorbild für dieses Konzept zur Umsetzung von Gitterverfeinerung existiert in Form der Optimierungskomponente von OpenLB, welche ebenfalls komplette Simulationen in einem sogenannten `Solver` kapselt [14, vgl. Abb. 4.1]. Langfristig könnten mit diesem Ansatz also beide gitterübergreifenden Module in einem gemeinsamen Konzept abgebildet werden.

Nachdem nun das grobe Umfeld eines Gitterverfeinerungsframeworks feststeht, gilt es, ein geeignetes Verfahren zur Umsetzung in und Nutzung mit eben diesem Framework zu wählen. Das von Lagrava et al. in *Advances in Multi-domain Lattice Boltzmann Grid Refinement* [17] beschriebene Verfahren, welches insbesondere auf [7] und [10] einen anpassungsfähigen Multi-Domain Gitterverfeinerungsalgorithmus für BGK LBM auf koinzidierenden D2Q9 Gittern aufbaut, erscheint hier als guter Kandidat. Die anfängliche Beschränkung auf zwei Dimensionen passend zur Einschränkung dieser Arbeit sowie die Flexibilität in Hinblick auf die verwendeten Restriktions- und Interpolationsoperatoren bilden hier eine gute Grundlage für eine erste und doch ausbaufähige Umsetzung von Gitterverfeinerung in OpenLB.

4.3 Struktur des Gitterverfeinerungsframeworks

Wie im vorangehenden Kapitel erläutert, soll das Framework zur Gitterverfeinerung nicht tief in vorhandene Strukturen integriert, sondern viel mehr über diesen stehend angesiedelt werden. Eine erste Hürde zu diesem Ziel ist die, größtenteils aus zwangfreien Konventionen bestehende, Struktur von OpenLB Anwendungen. So sind zwar die einzelnen Komponenten der Simulation wie `CuboidGeometry2D` und `SuperLattice2D` vorgegeben, deren Konstruktion und Verknüpfung erfolgt jedoch größtenteils manuell.

Für sich ist diese Herangehensweise des flexiblen Zusammensetzens von Modulen durchaus erhaltenswert und bildet eine der Stärken von OpenLB. Zur übergreifenden und für den Nutzer möglichst bequemen Einbindung von Gitterverfeinerung – wir erinnern uns: Das Ziel ist es, Gitter erst im Nachhinein mit einem einzigen Funktionsaufruf zu verfeinern – muss jedoch zumindest die Konstruktion des auflösungseigenen `SuperLattice2D` mit dem dazugehörigen Umfeld aus `UnitConverter`, `LoadBalancer`, `CuboidGeometry2D` und `SuperGeometry2D` soweit wie möglich gekapselt werden.

Entsprechend besteht das Framework aus zwei Komponenten: Einer `Grid2D` Klasse, die in einem Konstruktoraufruf ein `SuperLattice2D` zusammen mit dem benötigten Umfeld instanziiert und einer `Coupler2D` Klasse zur Abbildung der Übergänge zwischen mehreren `Grid2D` Instanzen. Die Gitterklasse stellt dabei eine Methode `Grid2D::refine` bereit, die anhand einer Parametrisierung der zu verfeinernden Domäne ein neues `Grid2D` konstruiert und über entsprechende `Coupler2D` Objekte mit sich selbst verknüpft. Funktionen wie `prepareGeometry` und `prepareLattice` können in diesem Umfeld dann durch entsprechende *Getter* mit `Grid2D` zusammenarbeiten. Arbeiten diese Funktionen bereits auf Grundlage von analytischen Indikatoren, d. h. unabhängig der Auflösung, können sie sogar ohne Anpassung für alle Gitterauflösungen verwendet werden.

```
template <typename T, template<typename> class DESCRIPTOR>
RefiningGrid2D<T,DESCRIPTOR>::RefiningGrid2D(
  Grid2D<T,DESCRIPTOR>& parentGrid, Vector<T,2> origin, Vector<T,2> extend):
  Grid2D<T,DESCRIPTOR>(
    std::unique_ptr<IndicatorF2D<T>>(new IndicatorCuboid2D<T>(extend, origin)),
    2*parentGrid.getConverter().getResolution(),      // Auflösungsübergang 2 : 1
    2*parentGrid.getConverter().getLatticeRelaxationTime() - 0.5, // Siehe (3.4)
    parentGrid.getConverter().getReynoldsNumber()),                // Re_g = Re_f
  _origin(origin),
  _extend(extend),
  _parentGrid(parentGrid)
{ }
```

Listing 1: Konstruktor der verfeinernden Gitter

Die Konstruktion von `Grid2D` erfolgt anhand einer indikatorgegebenen Beschreibung des Simulationsgebiets sowie der gewünschten Auflösung zusammen mit der Relaxationszeit und der modellierenden Reynolds-Nummer:

```
Grid2D(FunctorPtr<IndicatorF2D<T>>&& domainF, int resolution, T tau, int re);
```

Während sich die Realisierung dieser Signatur als einfache Konstruktion der erläuterten OpenLB Struktur erweist, gestaltet sich die Konstruktion der vererbten `RefiningGrid2D` Klasse in Listing 1 interessanter, da hier dann Kraft der Ergebnisse von Kapitel 3.3.1 die Vorgabe des groben Gitters zusammen mit dem verfeinerungsbedürftigen Teilbereich zur Erstellung eines neuen Gitters ausreicht.

```
template <typename T, template<typename> class DESCRIPTOR>
void Grid2D<T,DESCRIPTOR>::collideAndStream()
{
  for ( auto& fineCoupler : _fineCouplers ) {
    fineCoupler->store(); // Speichern von Werten in x^g_{g→f} ∈ G zu Zeit t
  }

  this->getSuperLattice().collideAndStream(); // Zeitschritt t → t + δt_g auf G

  for ( auto& fineGrid : _fineGrids ) {
    fineGrid->collideAndStream(); // Zeitschritt t → t + δt_g/2 auf F
  }

  for ( auto& fineCoupler : _fineCouplers ) {
    fineCoupler->interpolate(); // Interpolation von Werten in x^g_{g→f} ∈ G zu Zeit t + δt_g/2
    fineCoupler->couple();      // Setzen von f_{f,i} in x_{g→f} ∈ F
  }

  for ( auto& fineGrid : _fineGrids ) {
    fineGrid->collideAndStream(); // Zeitschritt t + δt_f → t + 2δt_f auf F
  }

  for ( auto& fineCoupler : _fineCouplers ) {
    fineCoupler->store();    // Speichern von Werten in x^g_{g→f} ∈ G zu Zeit t + δt_g
    fineCoupler->couple();   // Setzen von f_{f,i} in x_{g→f} ∈ F
  }

  for ( auto& coarseCoupler : _coarseCouplers ) {
    coarseCoupler->couple(); // Setzen von f_{g,i} in x_{f→g} ∈ G
  }
}
```

Listing 2: Rekursiver Kollisions- und Strömungsschritt mit Gitterkopplung

Wie in Kapitel 3.4 dargelegt, müssen zur Gitterkopplung nach jedem Kollisions- und Strömungsschritt verschiedene Arbeiten durchführt werden. So ist die Ausführung von Kollisions- und Strömungsschritten auf dem feinen Gitter zusammen mit der jeweiligen Vor- und Nacharbeit strikt an die Nacharbeit von Kollisions- und Strömungsschritten auf dem groben Gitter gebunden.

Wurde das grobe Gitter um einen Zeitschritt weiterentwickelt, muss der Zustand des feinen Gitters ebenfalls um entsprechend zwei feine Zeitschritte evolviert werden, damit die groben Verteilungsfunktionen vervollständigt werden können. Es liegt somit nahe, die Aufrufe von `SuperLattice2D::collideAndStream` in einer `Grid2D::collideAndStream`

Methode zu kapseln, um auf diese Weise die Aufrufe von `Coupler2D` an den korrekten Stellen durchzuführen.

Konkret erhalten wir in Listing 2 bei gesammelter Verwaltung der von `Grid2D::refine` erstellten feinen Gitter und den zugehörigen Kopplern eine, der Algorithmenübersicht in Abbildung 16 nicht unähnliche, Implementierung von `Grid2D::collideAndStream`. Zu bemerken ist, dass die Konstellation aus dieser Methode zusammen mit `Grid2D::refine` durch Selbstaufruf bereits die freie Schachtelung von Verfeinerungsbereichen erlaubt.

```
// Initialisiere gröbstes Gitter mit gewünschten Fluidkonstanten
Grid2D<T,DESCRIPTOR> coarseGrid(coarseDomain, resolution, tau, Re);
// Diskretisiere gewünschte Geometrie in Materialzahlen auf dem groben Gitter
prepareGeometry(coarseGrid);

// Verfeinere ein Einheitsquadrat beginnend bei (0.5, 0.5) ∈ ℝ²
// (`RefiningGrid2D` ist eine bis auf Kopplung eigenständige Simulation)
RefiningGrid2D<T,DESCRIPTOR>& fineGrid = coarseGrid.refine({0.5, 0.5}, {1.0, 1.0});
// Diskretisiere gewünschte Geometrie in Materialzahlen auf dem feinen Gitter
prepareGeometry(fineGrid);

// Schließe den feinen Bereich aus dem groben Gitter aus
auto refinedOverlap = fineGrid.getRefinedOverlap();
coarseGrid.getSuperGeometry().reset(refinedOverlap);

// Definiere das gewünschte Fluidverhalten durch Binden von Dynamics und Rand-
// konditionen anhand der Materialzahlen (`forEachGrid` ruft `prepareLattice`
// auf allen Gittern der Verfeinerungshierarchie auf)
coarseGrid.forEachGrid(prepareLattice);

// Simulationsschleife über 100 physikalische Sekunden
for (int iT = 0; iT < coarseGrid.getConverter().getLatticeTime(100); ++iT) {
  // Führe Kollisions- und Strömungsschritt rekursiv auf allen Gittern aus,
  // Auflösungskopplung erfolgt automatisch
  coarseGrid.collideAndStream();

  // Ergebnisaufbereitung (z.B. Fehlernormberechnung, VTK-Ausgabe etc.)
  coarseGrid.forEachGrid(getResults);
}
```

Listing 3: Beispielhafte Nutzung von `Grid2D`

Wie in diesem Listing zu sehen, kann Gitterverfeinerung innerhalb des beschriebenen Frameworks schon erfreulich kompakt formuliert werden. Tatsächlich fehlt zum Etappenziel der einzeilig aktivierbaren manuell positionierten Gitterverfeinerung lediglich eine weitere Abstraktion von von `prepareGeometry` und `prepareLattice`.

Zur umfassenden Beschreibung des Gitterverfeinerungsframework fehlt uns jetzt nur noch die Definition der `Coupler2D` Objekte. Da diese die Einzelheiten des Verfeinerungsverfahrens umsetzen, werden sie im Rahmen des folgenden Kapitels zur Implementierung des Verfahrens von Lagrava et al. näher beleuchtet werden.

4.4 Umsetzung des Verfahrens von Lagrava et al.

Grundsätzlich implementiert jede Instanz von `Coupler2D` die Kopplung zweier Gitter in einer Richtung entlang einer, durch Ursprung und Ausdehnung charakterisierten, Linie innerhalb des physikalischen Simulationsgebiets. Für die Kopplung einer rechteckigen `RefiningGrid2D` Instanz werden von `Grid2D::refine` in diesem Fall acht Kopplungsobjekte erzeugt – vier Seiten mit jeweils zwei Kopplern.

```
template <typename T, template<typename> class DESCRIPTOR>
class Coupler2D {
protected:
  Grid2D<T,DESCRIPTOR>& _coarse;
  Grid2D<T,DESCRIPTOR>& _fine;

  const int  _coarseSize;
  const int  _fineSize;
  const bool _vertical;

  const Vector<T,2> _physOrigin;

  const Vector<int,3>& getFineLatticeR(int y) const;
  const Vector<int,3>& getCoarseLatticeR(int y) const;

  T getScalingFactor() const; // Skalierungsfaktor (3.6) der Nicht-Equilibriumsverteilung
  T getInvScalingFactor() const;

private:
  std::vector<Vector<int,3>> _coarseLatticeR;
  std::vector<Vector<int,3>> _fineLatticeR;

public:
  Coupler2D(Grid2D<T,DESCRIPTOR>& coarse, Grid2D<T,DESCRIPTOR>& fine,
            Vector<T,2> origin, Vector<T,2> extend);

};
```

Listing 4: Gemeinsame Struktur beider Kopplungsklassen

Die im Zuge dieser Arbeit entwickelte Version von `Coupler2D` beschränkt sich hierbei auf zu einem Einheitsvektor parallele Gitterübergänge. Sowohl für die Kopplung der mit `CuboidGeometry2D` modellierbaren Aufteilungen als auch für das umzusetzende Verfahren ist diese Einschränkung kein Hindernis, da Lagrava et al. ebenfalls nur von horizontalen bzw. vertikalen Gitterübergängen ausgehen.

Da `Grid2D` Methoden zur Diskretisierung physikalischer Koordinaten auf das Gitter bereitstellt, besteht das Fundament der beiden benötigten Kopplungsklassen größtenteils nur aus der Bestimmung aller zu setzenden Kopplungsknoten entlang der Übergangslinie. Die abgeleiteten Klassen `FineCoupler2D` und `CoarseCoupler2D` können auf die Liste dieser Knoten dann mittels `getFineLatticeR` und `getCoarseLatticeR` zugreifen und so ihre eigene Implementierung auf das Wesentliche beschränken.

```
template <typename T, template<typename> class DESCRIPTOR>
class FineCoupler2D : public Coupler2D<T,DESCRIPTOR> {
private:
  std::vector<Vector<T,1>>                _c2f_rho;
  std::vector<Vector<T,DESCRIPTOR<T>::d>> _c2f_u;
  std::vector<Vector<T,DESCRIPTOR<T>::q>> _c2f_fneq;

public:
  FineCoupler2D(Grid2D<T,DESCRIPTOR>& coarse, Grid2D<T,DESCRIPTOR>& fine,
                Vector<T,2> origin, Vector<T,2> extend);

  void store();
  void interpolate();
  void couple();
};
```

Listing 5: Struktur des Kopplers von grob nach fein

Wie in Listing 5 zu sehen, benötigt das Setzen der feinen Verteilungsfunktionen in `FineCoupler2D::couple` einen Zwischenspeicher der groben Verteilungsmomente und der groben Nicht-Equilibriumsverteilung entlang der Kopplungslinie. Dieser wird in der Methode `FineCoupler2D::store` gesetzt und dient in `FineCoupler2D::interpolate` der linearen Zeitinterpolation der groben Verteilungsfunktionen zu Zeitpunkt $t + \delta t_f$ entsprechend der Ausführungen in Kapitel 3.4.

```
for (int y=1; y < this->_coarseSize-2; ++y) {
  const auto rho  = order4interpolation(_c2f_rho,  y); // Siehe Listing 7
  const auto u    = order4interpolation(_c2f_u,    y);
  const auto fneq = order4interpolation(_c2f_fneq, y);

  const T uSqr = u*u;

  const auto finePos = this->getFineLatticeR(1+2*y);
  Cell<T,DESCRIPTOR> fineCell;
  fineLattice.get(finePos, fineCell);

  for (int iPop=0; iPop < DESCRIPTOR<T>::q; ++iPop) {
    fineCell[iPop] = lbHelpers<T,DESCRIPTOR>::equilibrium(iPop, rho[0], u.data, uSqr)
                   + this->getScalingFactor() * fneq[iPop];
  }

  fineLattice.set(finePos, fineCell);
}
```

Listing 6: Ausschnitt der Methode `FineCoupler2D::couple`

Zur Beleuchtung des Herzstücks der feinen Kopplung sehen wir in Listing 6 einen Ausschnitt der Kopplungsfunktion für Knoten $x^f_{g\to f} \in \mathcal{F}$ mit räumlicher Interpolation der benötigten groben Werte. Zusammen mit der zugehörigen Interpolationsfunktion in

Listing 7 lässt sich dabei sehen, wie sich das Verfeinerungsverfahren durch Verwenden von OpenLB Modulen wie den `lbHelpers` und der `Vector` Datenstruktur sehr nah an seiner mathematischen Formulierung umsetzen lässt.

```
template <typename T, unsigned N>
Vector<T,N> order4interpolation(const std::vector<Vector<T,N>>& data, int y)
{
  return 9./16. * (data[y] + data[y+1]) - 1./16. * (data[y-1] + data[y+2]);
}
```

Listing 7: Templatefunktion der Interpolationsformel (3.15)

Der Koppler `CoarseCoupler2D` zum Setzen der groben Verteilungsfunktionen gestaltet sich derweil einfacher, da kein Zwischenspeicher benötigt wird und nur eine Restriktionsoperation anzuwenden ist.

```
template <typename T, template<typename> class DESCRIPTOR>
void computeRestrictedFneq(const SuperLattice2D<T,DESCRIPTOR>& lattice,
                           Vector<int,3> latticeR,
                           T restrictedFneq[DESCRIPTOR<T>::q])
{
  for (int iPop=0; iPop < DESCRIPTOR<T>::q; ++iPop) {
    const auto neighbor = latticeR
                        + {0, DESCRIPTOR<T>::c[iPop][0], DESCRIPTOR<T>::c[iPop][1]};
    Cell<T,DESCRIPTOR> cell;
    lattice.get(neighbor, cell);

    T fNeq[DESCRIPTOR<T>::q] {};
    lbHelpers<T,DESCRIPTOR>::computeFneq(cell, fNeq);

    for (int jPop=0; jPop < DESCRIPTOR<T>::q; ++jPop) {
      restrictedFneq[jPop] += fNeq[jPop];
    }
  }

  for (int iPop=0; iPop < DESCRIPTOR<T>::q; ++iPop) {
    restrictedFneq[iPop] /= DESCRIPTOR<T>::q;
  }
}
```

Listing 8: Umsetzung der Restriktionsoperation (3.11)

Hiermit sind die zentralen Bestandteile der Umsetzung des Verfahrens von Lagrava et al. im Kontext des in dieser Arbeit entwickelten Gitterverfeinerungsframework für OpenLB beschrieben. Zum Abschluss verbleibt nun noch die Evaluierung der Qualität eben dieses Verfahrens anhand verschiedener Beispiele.

5 Evaluierung

Die Auswahl von Beispielen und Kriterien zur Bewertung der Qualität eines Gitterverfeinerungsverfahrens gestaltet sich zunächst unklarer, als man annehmen könnte. So existieren nur für wenige praktische Strömungssituationen analytische Lösungen der Navier-Stokes Gleichungen. Zur Evaluation können wir uns also nicht auf die optimale Situation des Vergleiches mit analytischen Lösungen beschränken, sondern müssen auch auf Vergleichsdaten aus anderen Simulationen oder realen Experimenten zurückgreifen.

Solche Referenzwerte sind – abseits offensichtlicher Gütekriterien wie der Vermeidung divergierender Simulationen – Voraussetzung für die Bewertung der Verfahrensqualität. Ohne diese ist beim Vergleich mit aus uniformen Gittern gewonnenen Lösungen nicht klar, welches Ergebnis besser ist. Entsprechend beschränken wir uns je nach Beispiel auf die Betrachtung einer Auswahl der folgenden Gütekriterien:

1. Subjektive Qualität des Strömungsbildes
2. Stetigkeit der Momente im Gitterübergang
3. Bestimmung des Fehlers zu einer analytischen Lösung
4. Anwendung eines Gitterverfeinerungskriteriums
5. Vergleich von lokal verfeinerten und uniformen Gittern mit gleicher Knotenanzahl

Das hochwertigste Kriterium ist dabei der Fehlervergleich zur analytischen Lösung bei Beschränkung der Gesamtknotenanzahl. Existiert die dazu notwendige formalisierte Lösung nicht, sind auch Vergleiche mit Referenzwerten aus realen Experimenten oder gesicherten Simulationen denkbar. Die subjektive Qualitätsbewertung hingegen erlaubt nur die Bewertung von Extremfällen wie im Strömungsbild erkennbaren Übergängen zwischen Gittern oder möglicher Divergenzvermeidung durch gezielte Verfeinerung.

5.1 Rohrströmung

Als Einstiegspunkt wollen wir zunächst die grundsätzliche Funktion des Verfeinerungsverfahrens an einem möglichst einfachen Beispiel gegen möglichst korrekte Vergleichsdaten testen. Ein solches Beispiel ist gegeben durch die laminare Strömung in einem Rohr mit kreisförmigem Querschnitt und ohne Hindernisse abseits der Wände.

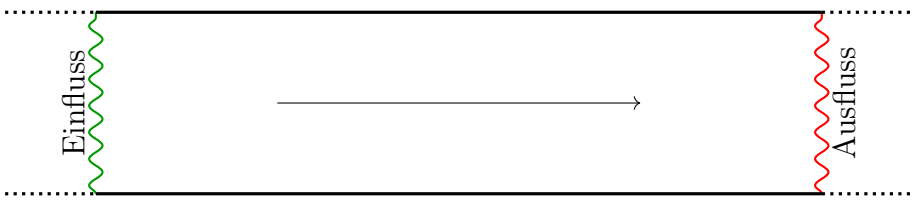


Abbildung 18: Schematische Darstellung der Rohrströmung

Diese Rohrströmung stellt nicht nur eine der einfachsten Strömungssituationen dar, sondern besitzt als Poiseuille-Fluss auch eine analytische Lösung, so dass wir ideale Vergleichsbedingungen vorfinden. Lieferte unser Verfahren in diesem Beispiel keine guten Ergebnisse, wäre nicht davon auszugehen, dass dies sich in komplexeren Situationen verbessern würde.

Definition 5.1 (Analytische Lösung des Poiseuille-Flusses) Seien $L_x, L_y \in \mathbb{R}_+$ die räumlichen Rohrdimensionen, ν die kinematische Viskosität und Δp der Druckgradient zwischen Ein- und Ausfluss. Dann ist die analytische Geschwindigkeit in x-Richtung gegeben als [4, vgl. Kap. 4]:

$$u_x(y) = \frac{1}{2\nu}\frac{\Delta p}{L_x} y(y - L_y)$$

Dies ergibt mit $u_x(L_y/2) := u_{\max}$ und $\Delta p := -p_0$ die analytische Lösung des Drucks:

$$p_0 = \frac{8 L_x \nu u_{\max}}{L_y^2}$$

Mit $u_{\max} := 1$ vereinfacht sich damit die analytische Lösung der x-Geschwindigkeit zu:

$$u_x(y) = -\frac{4}{L_y^2} y(y - L_y)$$

Das Geschwindigkeitsprofil des Poiseuille-Flusses ist also parabelförmig.

Wir wollen in einem 1×4 Meter bemessenden Rohr einen Poiseuille-Fluss simulieren. Als Auflösung einer Längeneinheit sei dabei $N = 10$ gewählt, was in der Diskretisierung durch 11×21 grobe und 21×43 feine Knoten abgebildet wird. In Abbildung 19 sehen wir das resultierende Gitter zusammen mit den zugewiesenen Materialzahlen. Wand- und Einflusszellen werden nach dieser Vorlage mit lokalen Geschwindigkeitsrandbedingungen umgesetzt. Während für den Einfluss dabei das Geschwindigkeitsprofil der analytischen Poiseuille-Lösung vorausgesetzt wird, erhält der Ausfluss eine Druckrandbedingung.

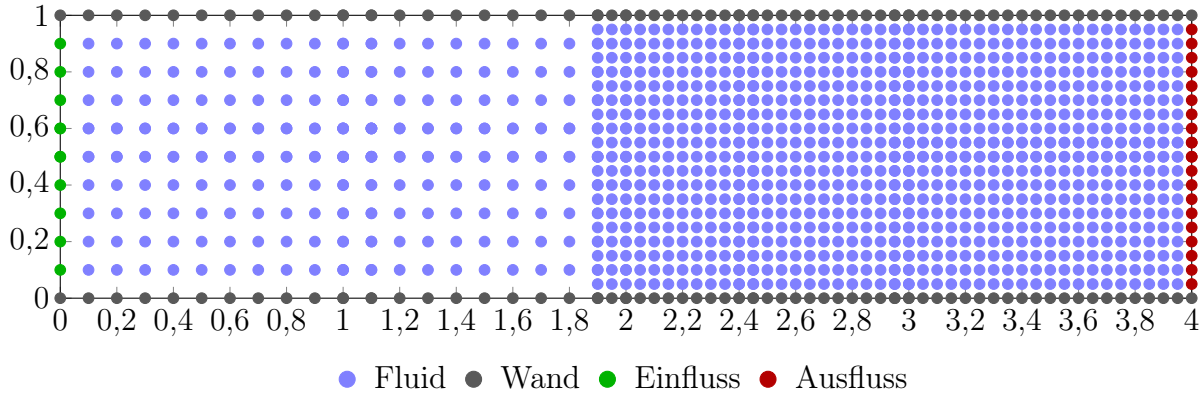


Abbildung 19: Gitterstruktur einer halbseitig verfeinerten Rohrströmung

Neben diesen knotenspezifischen Eigenschaften sei $u = 0{,}01$ die charateristische Geschwindigkeit in Lattice-Einheiten und $\mathrm{Re} = 10$ die modellierte Reynolds-Zahl. Erstellen wir unsere grobe `Grid2D` Instanz mit diesen, die Relaxationszeit τ fixierenden, Werten und führen die Simulation bis zur Konvergenz aus, erblicken wir nach geeigneter Aufbereitung der `VTK`-Ausgabe [25, Kap. 19.3] schließlich das in Abbildung 20 dargestellte Strömungsbild. Konvergenz bedeutet in diesem Kontext, dass die durchschnittliche Energie des feinen Gitters unter einen Residuumswert, hier $1 \cdot 10^{-5}$, gefallen ist.

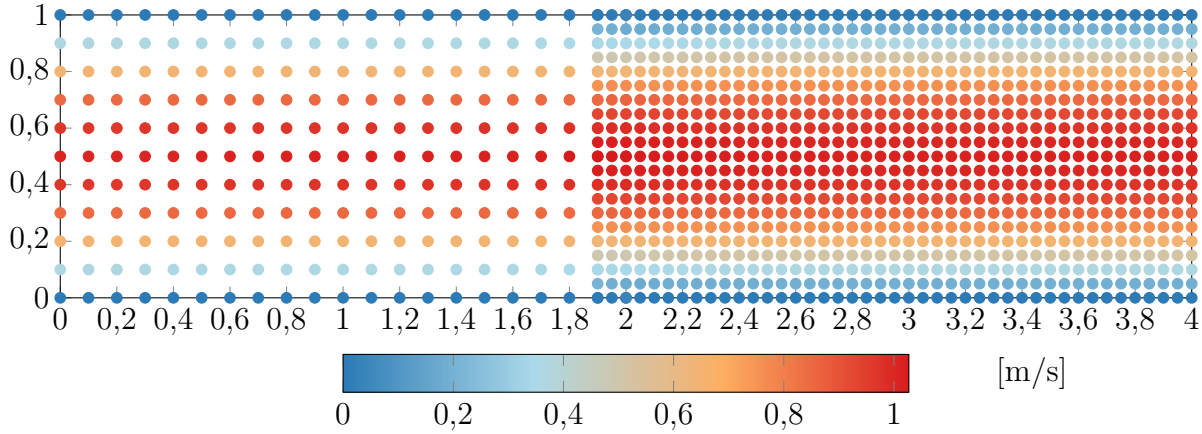
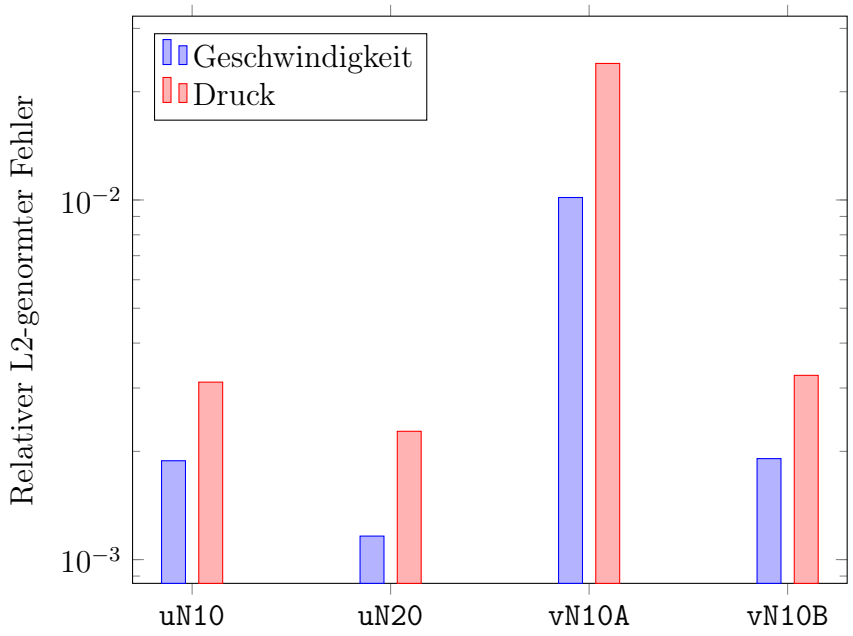


Abbildung 20: Geschwindigkeiten in x-Richtung im simulierten Poiseuille-Fluss

Bei erster Betrachtung lässt sich erkennen, dass die Strömung den Gitterübergang subjektiv ideal bestritten hat. Es treten also keine ungewöhnlichen Artefakte im Geschwindigkeitsbild auf und dieses setzt sich nach dem Übergang in, bis auf die neuen Zwischenwerte, unveränderter Weise fort. Tatsächlich ist bei Bildung einer geschlossenen Fläche durch Interpolation der Zwischenbereiche kein Gitterübergang erkennbar.

5.1.1 Vergleich mit der analytischen Lösung

Zur formalen Qualitätsbewertung ziehen wir im Folgenden die analytische Lösung von Geschwindigkeit und Druck des Poiseuille-Flusses in Definition 5.1 heran. Wir können diese in OpenLB einfach mit Hilfe des, die relative Fehlernorm $\frac{\|f_{\text{ana}} - f_{\text{sim}}\|_2}{\|f_{\text{ana}}\|_2}$ berechnenden, `SuperRelativeErrorL2Norm2D` Funktors mit der simulierten Lösung vergleichen:



	Gitterstruktur	Geschwindigkeitsfehler	Druckfehler
uN10	11×41	$1{,}88 \cdot 10^{-3}$	$3{,}12 \cdot 10^{-3}$
uN20	21×81	$1{,}16 \cdot 10^{-3}$	$2{,}27 \cdot 10^{-3}$
vN10A	11×21 | 21×43	$1{,}02 \cdot 10^{-2}$	$2{,}40 \cdot 10^{-2}$
vN10B	11×41 | 17×41	$1{,}91 \cdot 10^{-3}$	$3{,}25 \cdot 10^{-3}$

Abbildung 21: Geschwindigkeits- und Druckfehler in verschiedenen Gittern

Zunächst ist die halbseitig verfeinerte Lösung aus Abbildung 19 also um etwa eine Größenordnung schlechter als eine gleichmäßig mit $n = 20$ aufgelöste Berechnung. Auch im Vergleich mit dem uniform $n = 10$ aufgelösten Gitter erweist sich der Fehler der verfeinerten Lösung als deutlich größer – zumindest in diesem speziellen Beispiel ist Gitterverfeinerung also der Simulationsgenauigkeit messbar abträglich.

Nicht vergessen werden sollte jedoch, dass die untersuchte halbseitig verfeinerte Rohrströmung als Beispiel sehr konstruiert und nicht realitätsnah ist. Auch die noch folgenden Beobachtungen in Abbildung 23, nach welchen die lineare Interpolation zu kleineren Geschwindigkeitsfehlern führt, deutet auf eine beschränkte Aussagekraft dieses Beispiels hin. Eine Besonderheit ist in diesem Kontext auch die Existenz von Randbedingungen im Übergangsbereich. Eine etwaige Behandlung dieser wurde weder von Lagrava et al. angesprochen, noch in der dem Leser vorliegenden Arbeit näher untersucht. Tatsächlich verschwindet der *Verfeinerungsfehler* fast vollständig, wenn die Wände aus dem verfeinerten Bereich ausgespart werden.

Abschließend erscheint es beispielübergreifend intuitiv erwartbar, dass eine nicht aus dem konkreten Strömungsproblem informierte Anwendung von Gitterverfeinerung – und damit eine unbegründete Erhöhung der Simulationskomplexität gegenüber einem uniformen Gitter – einer Verbesserung der Präzision nicht zuträglich ist.

5.1.2 Vergleich der Interpolationsverfahren

Den halbseitig verfeinerten Poiseuille Simulationsaufbau können wir an dieser Stelle auch zur Nachvollziehung des, von Lagrava et al. für die Verwendung eines Verfahrens vierter Ordnung in der räumlichen Interpolation feiner Übergangsknoten ohne koinzidierende grobe Stützpunkte hervorgebrachten, Arguments verwenden.

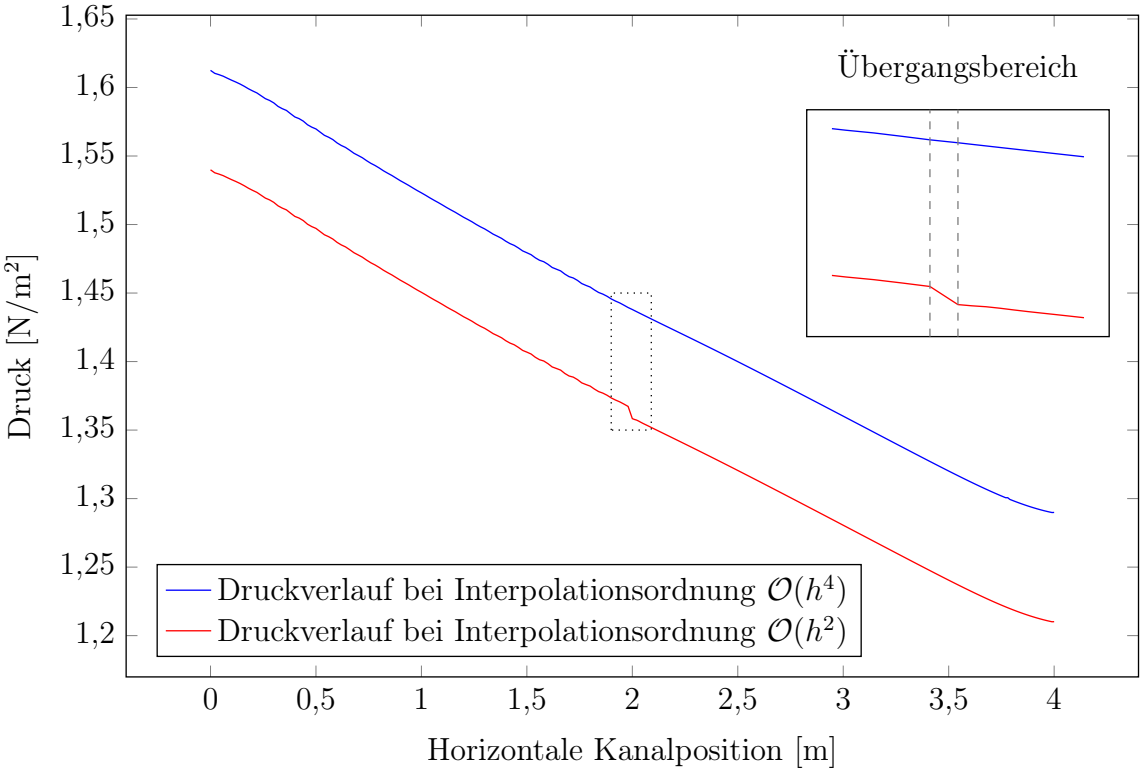


Abbildung 22: Druckverlauf bei linearer und kubischer Interpolation [17, vgl. Abb. 11]

Wir setzen dazu $N = 50$ als Auflösung der Längeneinheit, $\mathrm{Re} = 100$ als Reynolds-Zahl und eine Geschwindigkeitsrandbedingung mit Poiseuilleprofil im Ausfluss. Führen wir dann die Simulation mit dem linearen Interpolationsverfahren (3.14) aus und vergleichen den Verlauf des physikalischen Drucks auf einer vertikal zentrierten horizontalen Linie mit den, aus einem Durchlauf mit dem Verfahren vierter Ordnung (3.15) gewonnen, Daten, erhalten wir den in Abbildung 22 zu sehenden Plot.

Entsprechend der Beobachtungen in [17, Kap. 3.7] sehen auch wir bei linearer Interpolation einen prominenten Abfall des physikalischen Drucks im Übergangsbereich der Gitter. Bei kubischer Interpolation tritt dieser Fehler nicht auf, der Druckverlauf ist in diesem Fall so glatt, dass der Übergang nicht mehr zu erkennen ist.

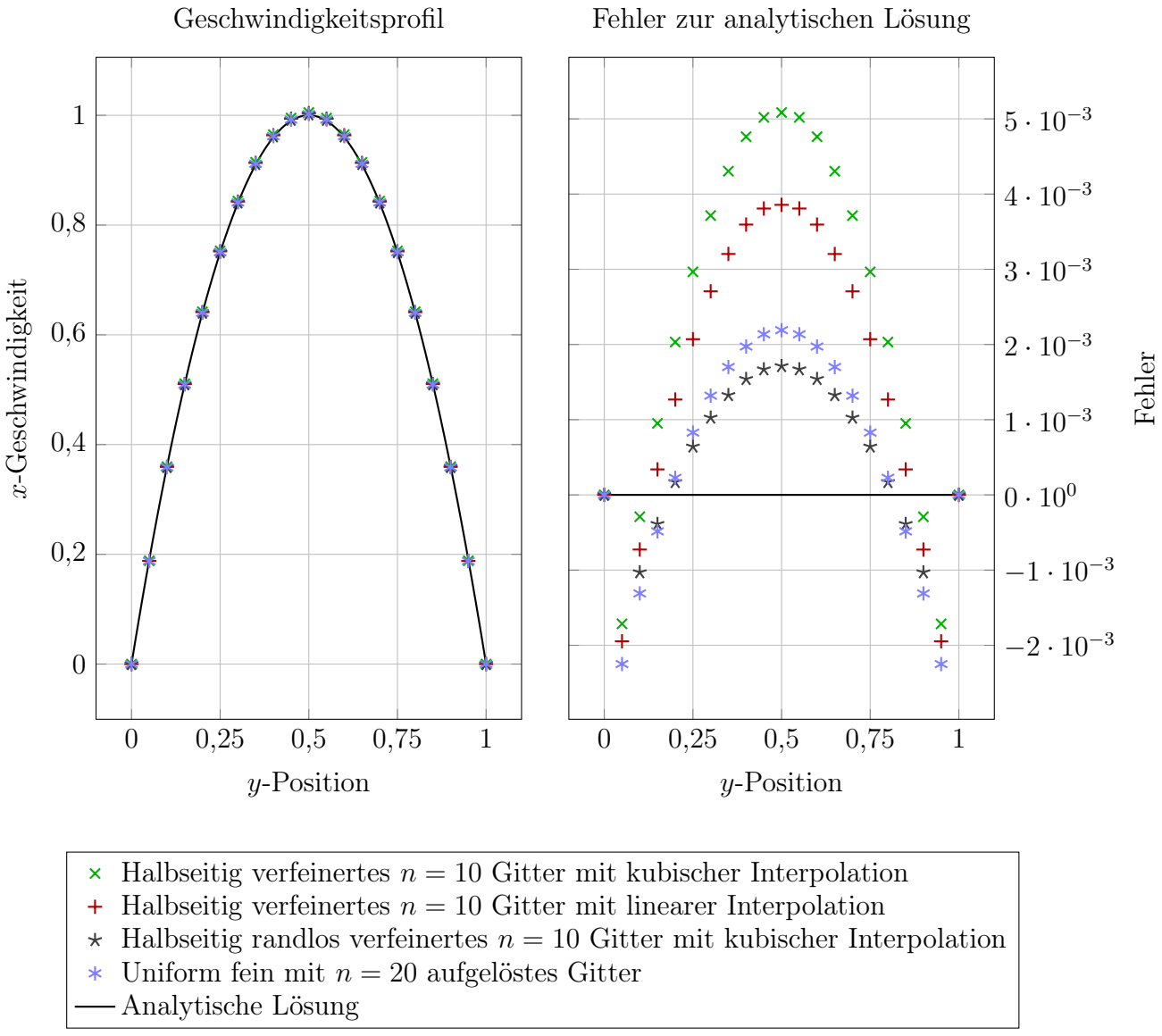


Abbildung 23: Vergleich des vertikalen Geschwindigkeitsprofil bei $x = 3$

Zum Abschluss dieses Beispiels vergleichen wir in Abbildung 23 das Geschwindigkeitsprofil entlang einer die gesamte Rohrbreite durchgeschreitenden vertikalen Linie bei $x = 3$ im halbseitig verfeinerten Fall. Wir sehen hier zunächst die subjektiv beurteilt gute Qualität des Strömungsbildes erneut bestätig, erkennen aber auch den größeren Fehler im Vergleich mit der uniform fein aufgelösten Simulation. Interessant ist hier, dass die den Druckabfall ausgleichende kubische Interpolation zugleich den Geschwindigkeitsfehler gegenüber der linearen Interpolation leicht zu erhöhen scheint.

Legen wir jedoch erneut ein randlos halbseitig verfeinertes Gitter wie beim Vergleich der L2-Normen in Abbildung 21 an, verschwindet dieses Problem und wir erhalten bei dieser Messung sogar einen leicht geringeren Fehler als im uniform fein aufgelösten Gitter.

Fassen wir die Ergebnisse der zurückliegenden ersten Analyse des Gitterverfeinerungsverfahrens unter dem Eindruck der beschränkten Aussagekraft bezogen auf die Wahl der Rohrströmung zusammen, bestätigt sich das Verfahren als grundsätzlich anwendbar. Es treten also sowohl bei subjektiver Betrachtung als auch bei Vergleich der analytischen Lösung keine extremen Fehler im Strömungsbild auf. Der beobachtete negative Einfluss von Randbedingungen im Übergangsbereich der Gitter sollte hier aber nicht unerwidert bleiben – bis die korrekte Behandlung solcher Randbedingungen nicht geklärt ist, sollten Gitterübergänge nur im offenen Fluid platziert werden, denn nur dessen Skalierung wird in Kapitel 3 und der zugrundeliegendem Arbeit [17] behandelt.

5.2 Umströmter Zylinder

Bei dem *umströmten Zylinder* handelt es sich um ein verbreitetes Strömungsbeispiel, welches entsprechend in der Menge der OpenLB Beispielanwendungen enthalten ist. Grundsätzlich ähnelt es dem Aufbau der Rohrströmung – simuliert wird die von zwei Wänden begrenzte Strömung zwischen Ein- und Ausfluss ergänzt um ein zylindrisches Hindernis im Eingangsbereich.

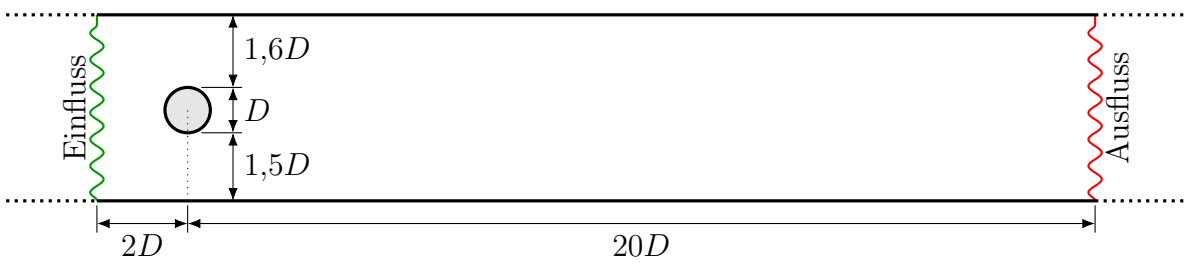


Abbildung 24: Schematische Darstellung des umströmten Zylinder [23, vgl. Abb. 1]

Während für diese Strömungssituation noch keine analytische Lösung gefunden wurde, stehen in *Recent Benchmark Computations of Laminar Flow Around a Cylinder* [23] detaillierte, hochwertige und mit verschiedenen Verfahren berechnete Vergleichsdaten zur Verfügung. Bevor wir diese jedoch zur Evaluation heranziehen, wollen wir uns vorerst der subjektiven Qualität der Ergebnisse versichern und uns nähere Gedanken zur Lokalisierung der Verfeinerunsbereiche machen.

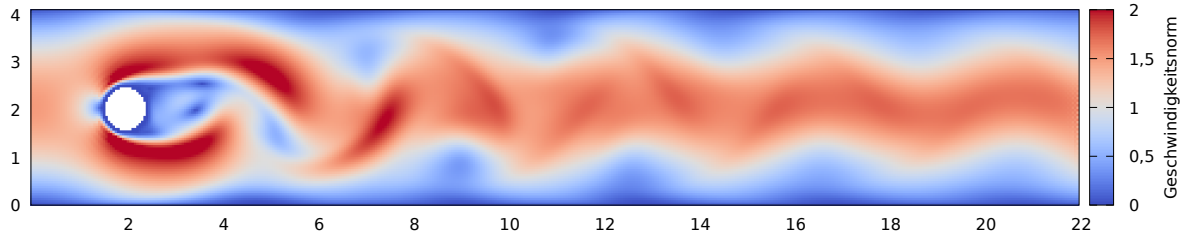


Abbildung 25: Uniform mit $N = 20$ aufgelöstes Strömungsbild zu $t = 16s$

Für die Umsetzung in OpenLB parametrisieren wir die Geometrie bezogen auf den Zylinderdurchmesser D und dimensionalisieren diesen wiederum als $D := 0{,}10\text{m}$, was zugleich der charakteristischen Länge entspreche. Auflösungsangaben beziehen sich im Folgenden also auf den Durchmesser des Zylinders in groben Gitterweiten. Hinblickend auf die Vorgaben zum instationären Testfall [23, Kapitel 2.2b] sei $\text{Re} := 100$ die Reynolds-Zahl und für den Einfluss sei ein Poiseuille-Geschwindigkeitsprofil angelegt.

Wände und Ausflüsse werden analog zur hindernisfreien Rohrströmung durch lokale Geschwindigkeits- bzw. Druckrandbedingungen konstruiert, während der Zylinder den Fluss durch *Bounce-Back* hindere. Eine Relaxationszeit $\overline{\tau}_g := 0{,}51$ des gröbsten Gitters vervollständigt unser Modell.

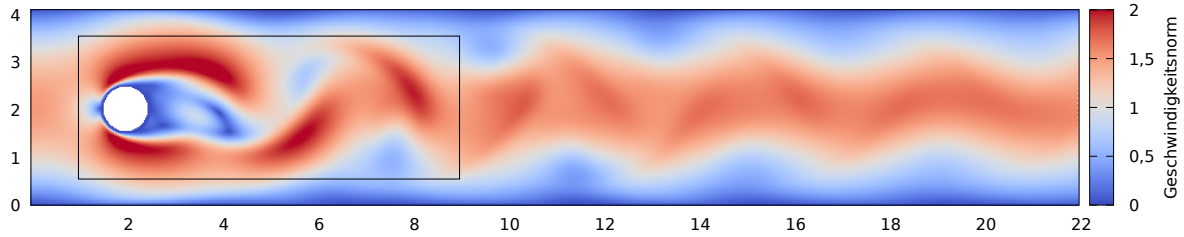


Abbildung 26: Einfach verfeinertes Strömungsbild zu $t = 16s$

Als Grundlage für den subjektiven Vergleich des Strömungsbildes simulieren wir zunächst in Abbildung 25 auf einem unverfeinert mit $N = 20$ aufgelösten Gitter. Charakteristisch ist hier direkt die Bildung einer Kármánschen Wirbelstraße zu beobachten. Vergleichen wir diese Grundsituation mit der in Abbildung 26 zu sehenden, um den Zylinder herum verfeinerten, Simulation, wirkt das Strömungsbild subjektiv gut aber unterschiedlich: Während positiv auffällt, dass der der Gitterübergang im Geschwindigkeitsbild trotz komplexerer Strömungsstruktur nicht zu erkennen ist, unterscheidet sich die Position der Wirbel trotz gleichem Zeitpunkt $t = 16s$ erkennbar.

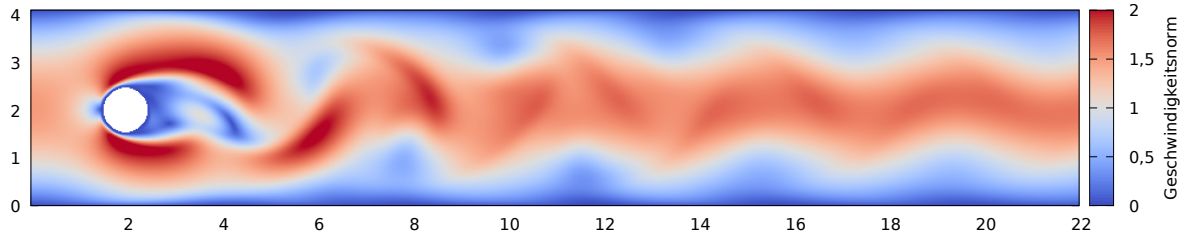


Abbildung 27: Uniform mit $N = 40$ aufgelöstes Strömungsbild zu $t = 16s$

Ergänzen wir unsere Auswahl von Geschwindigkeitsbildern jedoch um Abbildung 27, welche aus einem unverfeinerten $N = 40$ Gitter hervorgeht, ist oberflächlich gegenüber dieser kein Unterschied zur einfach verfeinerten Variante erkennbar. Beachtenswert ist dabei, dass das uniform mit $N = 40$ aufgelöste Gitter ~ 145000 Knoten enthält, während das subjektiv identische lokal verfeinerte $N = 20$ Gitter mit ~ 66000 nicht einmal halb so viele Knoten benötigt.

Noch weiter kann diese Beobachtung hier jedoch nicht bewertet werden, da nicht klar ist, welche Wirbelkonfiguration in dieser konkreten Strömungssituation korrekt ist. Aussagekräftiger für die formale Qualitätsbewertung wird der Vergleich der von Schäfer und Turek in [23] zusammengestellten Referenzwerte sein.

5.2.1 Anwendung eines formalen Kriteriums zur Gitterverfeinerung

An dieser Stelle wollen wir die Gelegenheit nutzen, uns näher mit der Wahl der Verfeinerungsbereiche auseinanderzusetzen. So scheint es auf der einen Seite intuitiv sinnvoll, einen Verfeinerungsbereich in besonders komplexen Regionen eines Simulationsgebietes zu platzieren – in diesem Fall also um den Zylinder, dessen Geometrie dann u. a. feiner aufgelöst würde. Auf der anderen Seite ist eine solch intuitive Entscheidung aber leicht fehlbar und schwer qualitativ zu beurteilen oder zu optimieren. Entsprechende Gedanken bewogen auch die Autoren unseres Verfeinerungsverfahrens, in *Automatic grid refinement criterion for lattice Boltzmann method* [18] ein auf der Knudsen-Zahl basierendes automatisches Kriterium zur Gitterverfeinerung herzuleiten.

Definition 5.2 (Knudsen-Zahl) Sei λ die mittlere freie Weglänge, L die charakteristische Länge des Systems. Die Knudsen-Zahl Kn ist definiert als:

$$\mathrm{Kn} := \frac{\lambda}{L} \quad \left(\overset{\text{[12, Kap. 1.4.2]}}{=} \frac{\mathrm{Ma}}{\mathrm{Re}} \frac{\rho^2 c_s \lambda}{\nu} \quad \overset{\text{[16, Gl. (7.22)]}}{\simeq} \frac{\mathrm{Ma}}{\mathrm{Re}} \right)$$

Diese Knudsen-Zahl ist eine dimensionslose Kennzahl der Strömungslehre. Ihr Wert beschreibt, ob ein Gas als strömungsmechanisches Kontinuum nach Navier-Stokes oder als Bewegung einzelner Teilchen betrachtet werden kann [16, S. 14]:

Kn $\ll 1$ Strömungsmechanisches Kontinuum nach Navier-Stokes

Kn $\gtrsim 1$ Betrachtung einzelner Teilchen

Definition 5.3 (Auftreten der Knudsen-Zahl in LBM) Die Knudsen-Zahl wird im LBM-Kontext nach [18, vgl. (21)] durch den Quotienten der Nicht-Equilibriums- und Equilibriumsverteilung in Richtung $i \in \{0, \ldots, q-1\}$ genähert:

$$\mathrm{Kn} \sim \frac{f_i^{\mathrm{neq}}}{f_i^{\mathrm{eq}}} \overset{!}{\ll} 1$$

Die Mittelung über alle Richtungen ergibt so die lokal in einem Knoten des Gitters genäherte Knudsen-Zahl:

$$C(x) := \frac{1}{q} \sum_{i=0}^{q-1} \left| \frac{f_i^{\mathrm{neq}}}{f_i^{\mathrm{eq}}} \right|$$

Definition 5.4 (Lokaler Verfeinerungsfaktor) Nach [18, vgl. (29)] führt die folgende Transformation der lokal genäherten Knudsen-Zahl zur Berechnung eines diskreten Verfeinerungsfaktors:

$$R(x) := \mathrm{round}\left(\log_2 \left(\frac{C(x)}{\mathrm{Kn}} \right) \right) \in \mathbb{Z}$$

Dieser Faktor beschreibt die Anzahl der empfohlenen Auflösungsverdoppelungen.

Dieses, die theoretische mit der tatsächlich simulierten Knudsen-Kennzahl des Fluids vergleichende, Kriterium liefert auf diese Weise bis auf die Ebene einzelner Zellen auflösbare Informationen zur lokalen Simulationsqualität. Beschränken wir dessen Wertebereich zur besseren Unterscheidung auf eine diskrete Menge, erhalten wir folgende Darstellung der unverfeinerten Simulation:

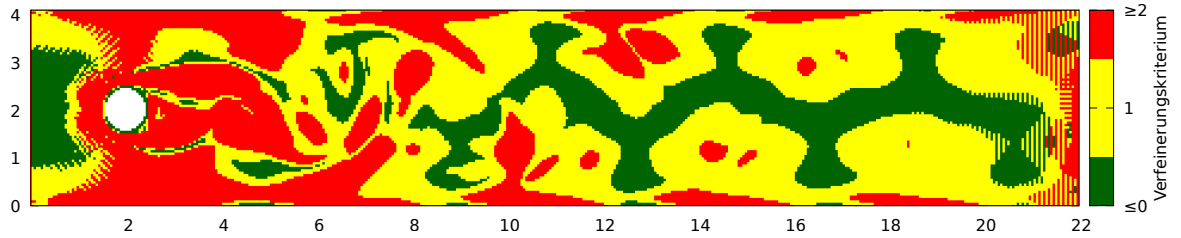


Abbildung 28: Verfeinerungskriterium in einem uniform mit $N = 20$ aufgelösten Gitter

Der lokale Vergleich der Knudsen-Zahlen eröffnet einen neuen, interessanten, Blick auf die Fluidstruktur – klar zu erkennen sind zunächst die wechselseitigen Wirbel sowie die Strömungskomplexität um den Zylinder im Eingangsbereich. Abseits dieses Hindernisses finden sich größere Bereiche mit guter Auflösung, also einem Verfeinerungsfaktor von größer oder gleich Null. Darüber hinaus weisen große Teile des Gitters mit Verfeinerungsfaktoren bis eins eine noch akzeptable Auflösung auf. Den größten Mangel an Simulationsqualität attestiert das Verfeinerungskriterium direkt um den Zylinder und in den wandnahen Wirbelbereichen.

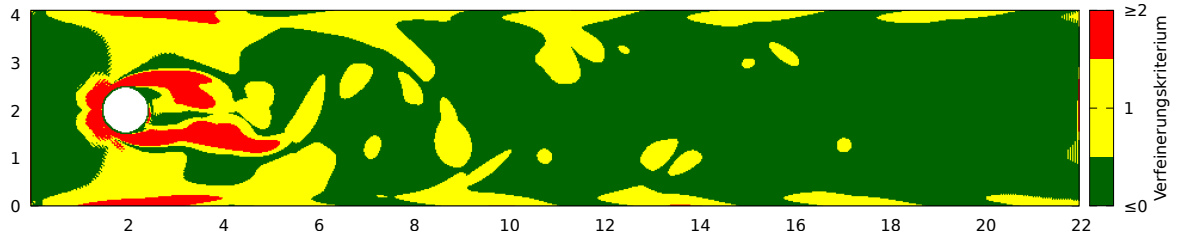


Abbildung 29: Verfeinerungskriterium in einem uniform mit $N = 40$ aufgelösten Gitter

Bei der Interpretation der Verfeinerungsfaktoren ist zu beachten, dass einzelne kleine Bereiche mit großen Faktoren keine global mangelnde Auflösung beschreiben. Das auf den Faktoren aufbauende automatische Gitterverfeinerungskriterium betrachtet dazu jeweils die durchschnittliche Qualität von a priori kartierten möglichen Verfeinerungsdomänen – nur wenn die extremalen Bereiche ausreichend aufgelöste Gebiete dominieren, ist demnach eine Verfeinerung empfohlen.

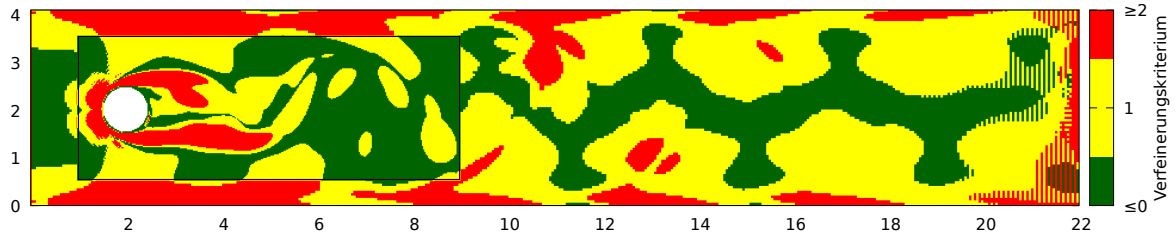


Abbildung 30: Verfeinerungskriterium in einem einfach verfeinerten $N = 20$ Gitter

Insgesamt war unter dieser formaleren Analyse unsere intuitive Wahl des in Abbildung 26 verfeinerten Bereichs akzeptabel. Dies bestätigt sich auch bei Berechnung des Knudsen-Kriterium für das einfach verfeinerte Gitter in Abbildung 30 – die angemahnten Bereiche im Umfeld des Zylinders sind hier deutlich reduziert. Gehen wir weiterhin davon aus, dass das Kriterium belastbare Aussagen zur Simulationsqualität liefert, so ist die Reduzierung des des Verfeinerungsfaktors in den verfeinerten Gebieten um genau eins ein Indiz für die Qualität des Verfeinerungsalgorithmus.

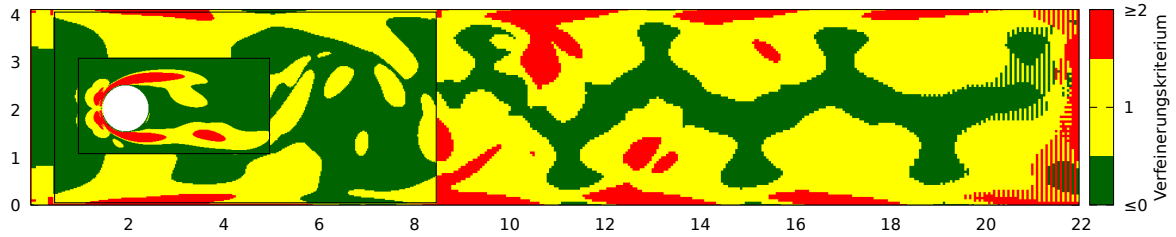


Abbildung 31: Verbesserte Verfeinerungsstruktur um den Zylinder

Typischerweise sind LBM-Simulationen des umströmten Zylinders für wirbelbildende Reynolds-Zahlen bei kleineren Auflösungen empfindlich gegenüber Divergenz im Ausflussbereich. Dies deutet sich auch bei noch ausreichender Auslösung in den Verfeinerungsfaktoren des Ausflussbereiches an. So divergiert ein uniform mit $N = 5$ aufgelöstes Gitter mit den verwendeten Parametern und Randbedingungen schon nach wenigen Schritten. Es wäre vorteilhaft, wenn sich dieses Problem unter Einsatz möglichst weniger zusätzlicher Gitterknoten beheben ließe. Und tatsächlich genügt schon die Verfeinerung eines schmalen Stücks des Ausgangsbereiches zur Stabilisierung der Simulation bei dieser niedrigen Auflösung. In Abbildung 32 sehen wir den Geschwindigkeitsplot eines solchen Gitters.

Dieses Beispiel gewinnt insbesondere an Eindruck, wenn wir die Gesamtanzahl der Gitterzellen mit den für ein stabiles uniformes Gitter benötigten Zellen vergleichen: Während das im Ausgang verfeinerte Gitter nur 3608 aktive Zellen beinhaltet, benötigt ein uniformes Gitter eine Mindestauflösung von $N = 12$, was 13500 Zellen entspricht.

Interessieren wir uns an dieser Stelle also nur für ein subjektiv korrektes und nicht-divergentes Strömungsbild, so ermöglicht uns Gitterverfeinerung ein Auskommen mit nur etwa einem Viertel der für ein unverfeinertes Gitter benötigten Zellen.

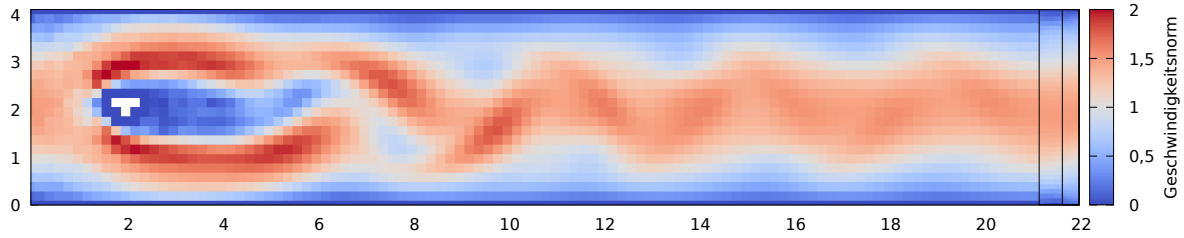
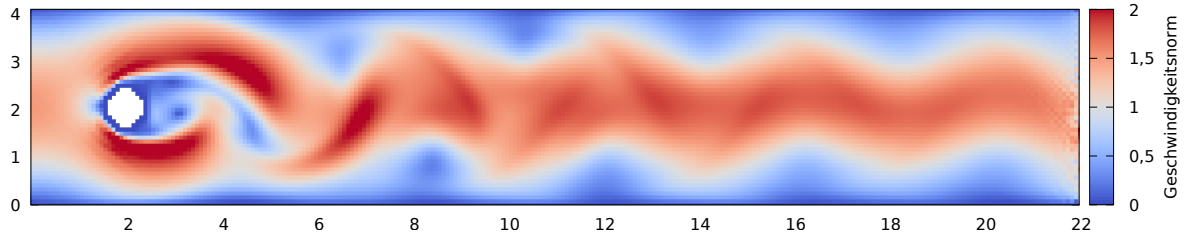


Abbildung 32: Stabile Zylinderumströmung mit $N = 5$ durch Ausgangsverfeinerung

Alternativ ist es möglich über das Festhalten der Anzahl der Freiheitsgrade, d. h. der Knotenanzahl, beschränkte Rechenressourcen besser auf die zu simulierende Strömungssituation aufzuteilen. Beispielsweise fällt auf, dass der zentrale Zylinder in Abbildung 32 nur sehr grob modelliert wird. Entsprechend wollen wir versuchen, bei Einschränkung auf die Anzahl der Gitterknoten eines uniformen Gitters mit $N = 12$, ein besseres Gitter zu erzeugen.

Uniform aufgelöstes Gitter mit 13 500 Knoten



Geeignet dreifach verfeinertes Gitter mit 13 454 Knoten

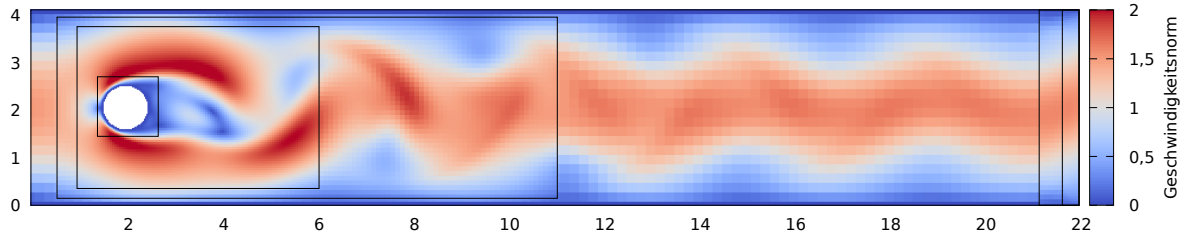


Abbildung 33: Vergleich von uniform und problembezogen verteilten Freiheitsgraden

Wir sehen das Geschwindigkeitsbild dieser Bemühungen in Abbildung 33. Die dort dargestellten Gitter beinhalten beide jeweils maximal 13500 Zellen. Der kleine Unterschied

in der Knotenanzahl ist dabei der Einschränkung auf quaderförmige Gitter geschuldet, welche eine exakte Fixierung der Knotenanzahl erschwert.

Klar zu erkennen ist die in der verfeinerten Variante deutlich bessere Diskretisierung des Zylinders durch Konzentration der verfügbaren Gitterknoten in dessen Umfeld. Auch liegt dem Ausfluss des verfeinerten Gitters die Divergenz ferner als dem Ausfluss des uniformen Gitters, an welchem sich schon Artefakte abzeichnen. Eine formalere Analyse der Qualität dieses optimierten Gitters erwartet uns in Kapitel 5.2.3.

5.2.2 Parallelisierung

Bevor wir dazu kommen, wollen wir, aufbauend auf den bezüglich der Knotenanzahl vergleichbaren Gittern in Abbildung 33, einen sehr kurzen Blick auf die Effizienz der parallelen Ausführung von verfeinerten Gittern riskieren. OpenLB implementiert hierzu mit OpenMPI [11] und OpenMP [6] zwei gängige Konzepte zur Kommunikation zwischen parallel arbeitenden Prozessen.

	Nicht parallel	OpenMPI		OpenMP	
Prozessanzahl	1	2	4	2	4
Uniform	36,46s	19,38s	10,77s	23,24s	14,32s
Verfeinert	29,90s	54,79s	73,97s	24,25s	18,38s

Tabelle 1: Vergleich der parallelen Ausführung auf einem Shared Memory System

Überraschenderweise steht es ohne die Beantwortung der Frage nach Minimierung des Kommunikationsaufwands in der Gitterkopplung noch schlechter um den Einfluss von Gitterverfeinerung auf die Ausführungszeit als erwartet. Da die einzelnen Gitter in der OpenMPI-Variante ohne Rücksicht auf deren Lokalität den vorhandenen Prozessoren zugeordnet werden, müssen die zur Kopplung notwendigen Informationen schlimmstenfalls in jedem Zeitschritt zwischen allen Prozessen kommuniziert werden. Solange das durch Verfeinerungsbeziehungen entstehende Bedürfnis nach Kommunikationsnähe von ansonsten unabhängigen Gitter von der Lastverteilung nicht beachtet wird, ist die OpenMPI-basierende Parallelverarbeitung nicht sinnvoll einsetzbar.

Bessere Ergebnisse können dagegen mit OpenMP und der Einschränkung auf Shared Memory Systeme erzielt werden. Der zusätzliche unoptimierte Kommunikationsaufwand der Gitterverfeinerung dominiert hier nicht mehr die eigentliche Simulationszeit und führt zu sinnvoller Anwendbarkeit auf Systemen mit gemeinsam genutztem Speicher.

Die Grundproblematik der effizienten Parallelisierung von verfeinerten Gittern tritt auch in den Ausführungen von Lagrava et al. [17, Kap. 4.1, letzter Abschnitt] insofern in Erscheinung, dass deren Vergleich der benötigten Rechenzeit sich auf einen Prozessor und somit Shared Memory Systemen beschränkt.

5.2.3 Vergleich von Widerstands- und Auftriebskoeffizienten

Bis hier haben wir die den Einfluss von Gitterverfeinerung auf die Zylinderumströmung entweder subjektiv, durch Vergleich der Strömungsbilder, oder grob, durch einfachen Vergleich der für eine divergenzfreie Simulation benötigten Knotenanzahl, bewertet. Wie eingangs erwähnt, existiert für die vorliegende Strömungssituation keine analytische Lösung, weshalb wir uns für eine formal belastbarere Bewertung auf vertrauenswürdige aber ebenfalls simulierte Referenzwerte stützen wollen.

Recent Benchmark Computations of Laminar Flow Around a Cylinder [23] liefert eine, dieser formalen Bewertung dienliche, Übersicht der, von 17 verschiedenen Forschungsgruppen beigetragenen und auf unterschiedliche Weisen gewonnenen, Lösungen einer klar definierten Zylinderumströmung. Diese Lösungen sind dabei anhand einer Auswahl von charakteristischen Messwerten wie dem Strömungswiderstands- und Auftriebskoeffizient sowie dem Druckunterschied zwischen Vorder- und Rückseite des Zylinders gegeben.

Definition 5.5 (Strömungswiderstands- und Auftriebskraft) Beschreite der Weg S den Rand des Zylinders und sei $n \in \mathbb{R}^2$ dessen Normale, v_t die Geschwindigkeit entlang der Tangente $t := (n_1, -n_0)$, ρ_c die charakteristische Dichte der Flüssigkeit, P der Druck sowie ν die kinematische Viskosität. Dann sind Widerstands- und Auftriebskraft des Zylinders gegeben als [23, Kap. 2.2]:

$$F_w = \int_S \left(\rho_c \nu \frac{\partial v_t}{\partial n} n_1 - P n_0 \right) dS \qquad \text{Widerstandskraft}$$

$$F_a = -\int_S \left(\rho_c \nu \frac{\partial v_t}{\partial n} n_0 + P n_1 \right) dS \qquad \text{Auftriebskraft}$$

Definition 5.6 (Strömungswiderstands- und Auftriebskoeffizient) Sei D der Zylinderdurchmesser, $\overline{U}$ die durchschnittliche Fluidgeschwindigkeit und ρ_c die charakteristische Dichte. Dann sind die Widerstands- und Auftriebskoeffizienten gegeben als [23, Kap. 2.2]:

$$c_w = \frac{2F_w}{\rho_c \overline{U}^2 D} \qquad \text{Widerstandskoeffizient}$$

$$c_a = \frac{2F_a}{\rho_c \overline{U}^2 D} \qquad \text{Auftriebskoeffizient}$$

Vergleichen wollen wir diese Koeffizienten nun im Rahmen des unstetigen Testfalls [23, 2.2b] mit Reynolds-Zahl $\mathrm{Re} = 100$, maximaler Einflussgeschwindigkeit $U = 1.5\,\mathrm{m/s}$ und charakteristischer Dichte $\rho_c = 1.0\,\mathrm{kg/m^3}$. Dieses Ziel hatten wir dabei schon zu Beginn dieses Kapitels im Blick, so dass die OpenLB-basierende Modellierung der Zylinderumströmung inklusive des optimierten Gitters in Abbildung 33 direkt verwendet werden kann. Für die Berechnung der Koeffizienten stellt OpenLB dabei in Form des `SuperLatticePhysDrag2D` Funktors bereits ein passendes *Messinstrument* bereit.

Die folgenden Abbildungen zeichnen demnach den zeitlichen Verlauf der durch das uniform mit $N = 12$ aufgelöste Gitter berechneten Charakteristiken zusammen mit den Resultaten des problembewusst verfeinerten $N = 5$ Gitters mit näherungsweise gleicher Anzahl Freiheitsgraden. Der anzustrebende formale Referenzwert ist dabei jeweils das Mittel der oberen und unteren Grenzwerte [23, Tabelle 4].

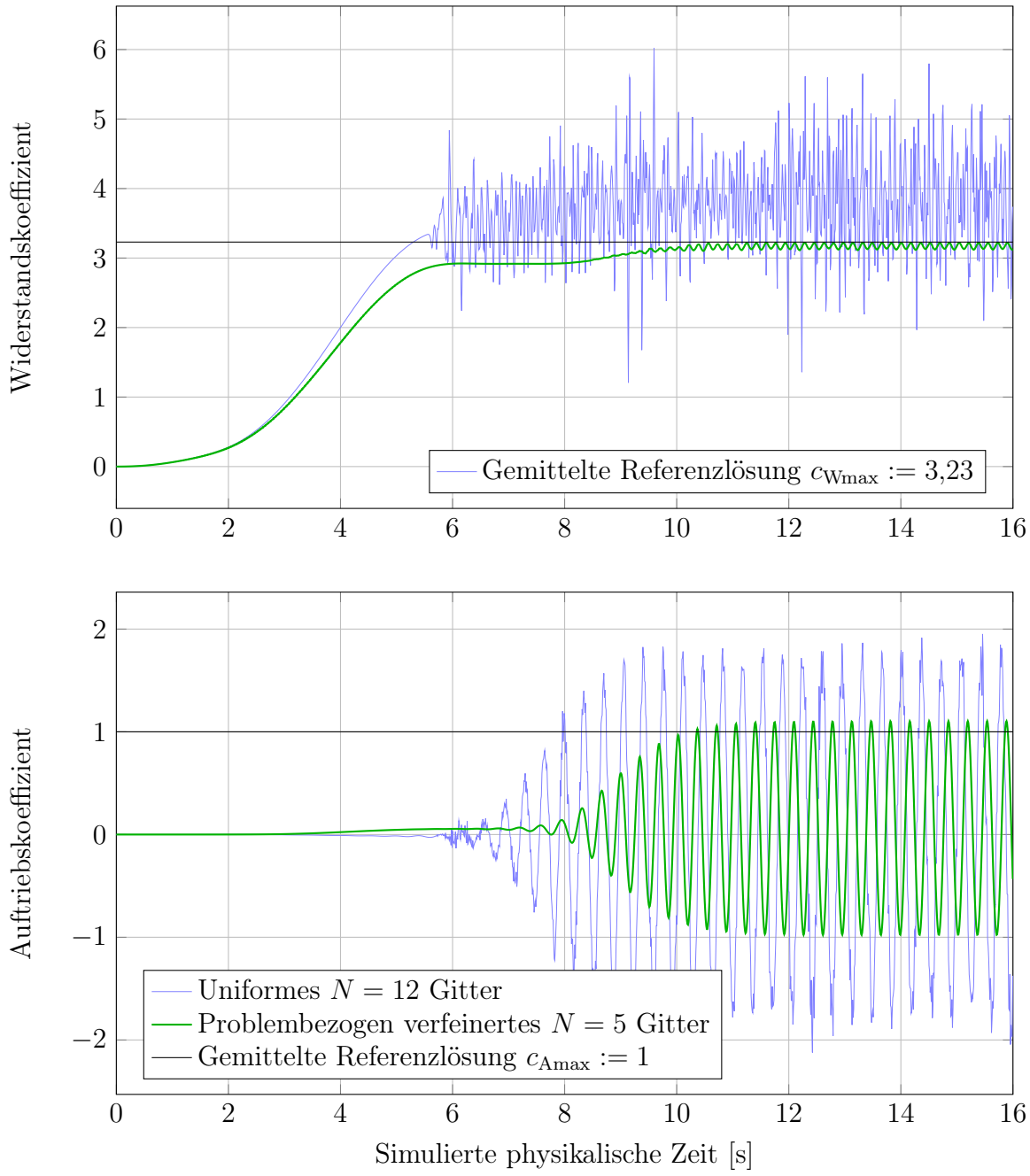


Abbildung 34: Zeitlicher Verlauf des Strömungswiderstands- und Auftriebskoeffizienten

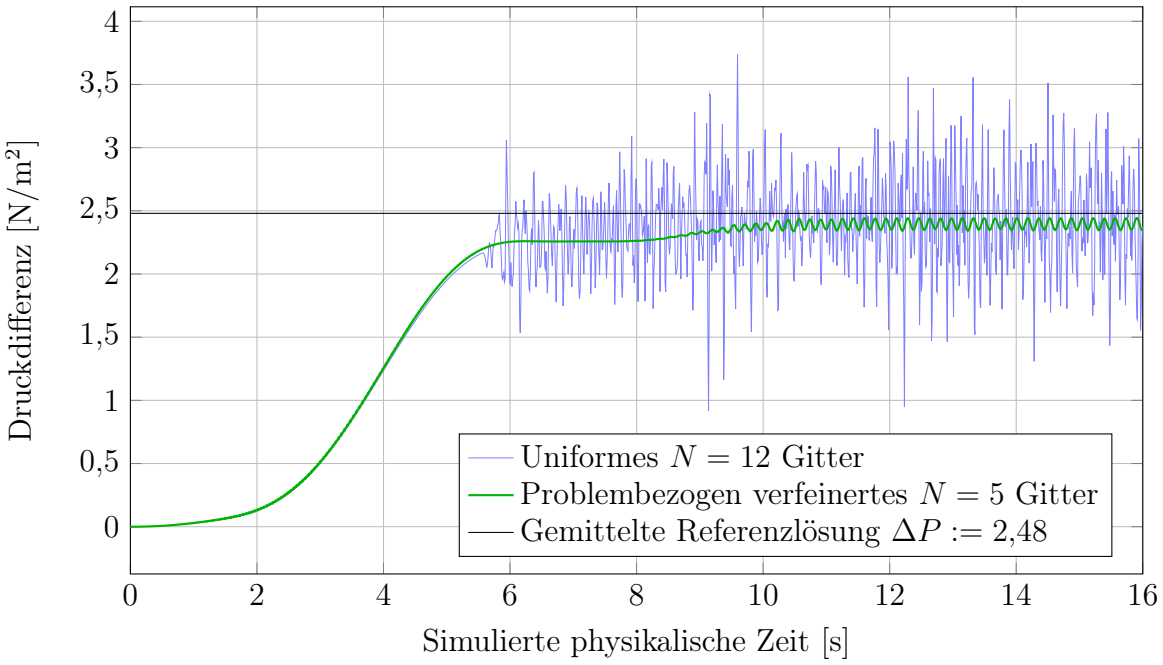


Abbildung 35: Zeitlicher Verlauf der Druckdifferenz

Klar zu erkennen ist, dass die Simulation auf dem verfeinerten Gitter über alle drei Kriterien sowohl die Referenzwerte besser trifft als auch unabhängig davon eine bessere Qualität bezogen auf Fluktuation und Konsistenz in der periodischen Strömungsphase aufweist. Da diese Ergebnisse bei näherungsweise beibehaltener Anzahl von Gitterknoten erzielt wurden – tatsächlich verwendet das verfeinerte Gitter sogar 46 Knoten weniger als das uniform aufgelöste – ist somit auch bei formaler Betrachtung der positive Nutzen von Gitterverfeinerung zur Simulation der Zylinderumströmung klar demonstriert. Wir erhalten bei leicht geringerer Anzahl von Knoten und damit näherungsweise gleichem Speicherbedarf ein deutlich besseres Simulationsergebnis.

	Uniform	Verfeinert	Referenzintervall [23]
$\widehat{c_w}$	3,2843	3,2193	[3,22, 3,24]
$\lvert\widehat{c_w} - 3{,}23\rvert$	0,0543	0,0107	[0, 0,01]
$\widehat{c_a}$	1,0705	1,1036	[0,99, 1,01]
$\lvert\widehat{c_a} - 1{,}0\rvert$	0,0705	0,1036	[0, 0,01]
ΔP	2,5793	2,4429	[2,46, 2,5]
$\lvert\Delta P - 2{,}48\rvert$	0,0993	0,0372	[0, 0,02]
$N =$	40	5 (max: 40)	
Knotenanzahl	145 314	13 454	

Tabelle 2: Vergleichswerte zweier maximal $N = 40$ aufgelöster Gitter

Das diesen Resultaten zugrunde liegende Gitter orientiert sich in der Anzahl seiner Zellen an einem uniform aufgelösten Gitter, welches knapp an der Grenze zur Divergenz im Ausflussgebiet arbeitet. Die im Vergleich dazu sehr guten Ergebnisse des optimierten Gitters sind also nicht zu überraschend. Als weiteren Vergleich betrachten wir deshalb in Tabelle 2 die entscheidenden maximalen Widerstands- und Auftriebskoeffizienten $\widehat{c_w}$ bzw. $\widehat{c_a}$ einer uniform mit $N = 40$, d. h. der maximalen lokalen Auflösung unseres problembewusst verfeinerten Gitters entsprechend, aufgelösten Simulation.

Auch im Vergleich mit diesem 145 314 Knoten umfassenden Gitter bewähren sich die, mit nur etwa einem Zehntel der Knoten gewonnenen, Ergebnisse der verfeinerten Simulation. Tatsächlich ist der Fehler des verfeinerten Gitters für Widerstandskoeffizient und Druckdifferenz sogar kleiner als der des uniformen Gitters mit ungleich mehr Knoten.

Wir haben an dieser Stelle also auch im formalen Vergleich bestätigt, dass sich Gitterverfeinerung zur besseren Verteilung beschränkter Rechenressourcen einsetzen lässt. Die bestimmten Vergleichswerte bestehen bei geeigneter Variation der lokalen Gitterweiten auch in Konkurrenz mit uniformen Gittern, die auf dem ganzen Simulationsgebiet der feinsten Gitterweite des heterogenen Gitters entsprechend aufgelöst sind. Es stellt sich daher die Frage, ob dieser klare Vorteil auch auf höher aufgelöste Gitter übertragen werden kann und sich die Ergebnisse in vergleichbarem Maße verbessern. Dazu sehen wir in Abbildungen 36 und 37 sowie zugehöriger Tabelle 3 die aerodynamischen Kennzahlen der uniformen $N = 48$ und $N = 80$ Gitter sowie eines problembezogen variierten $N = 20$ Gitters entsprechend der Struktur in Abbildung 33 ohne Ausflussverfeinerung.

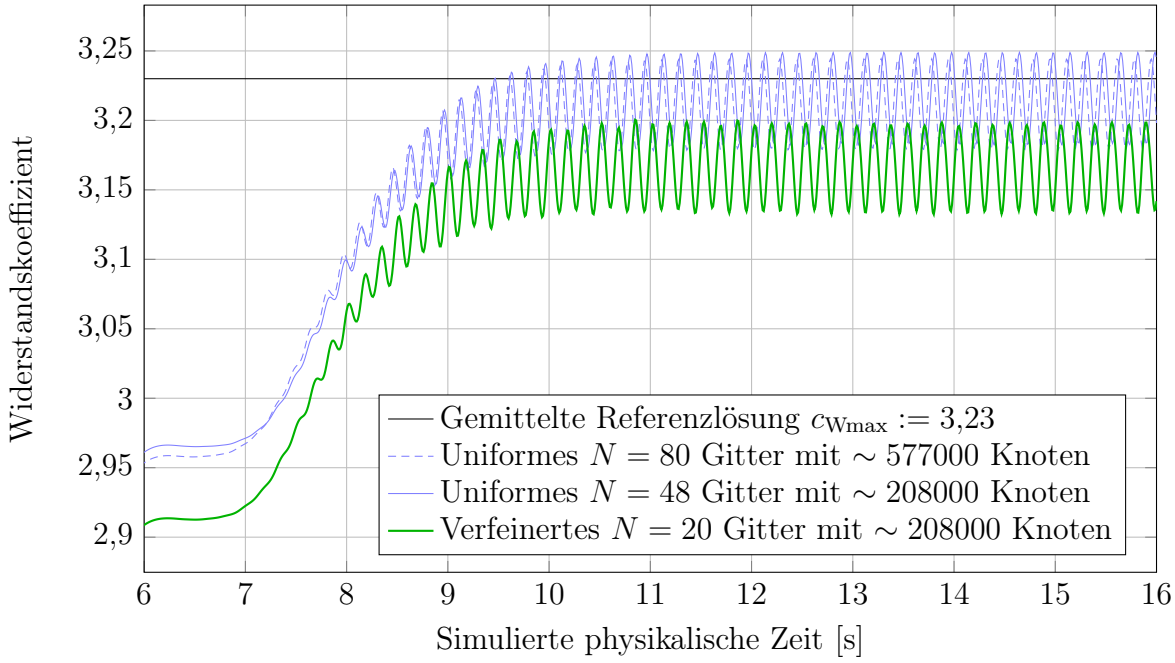


Abbildung 36: Strömungswiderstandskoeffizient in feineren Simulationen

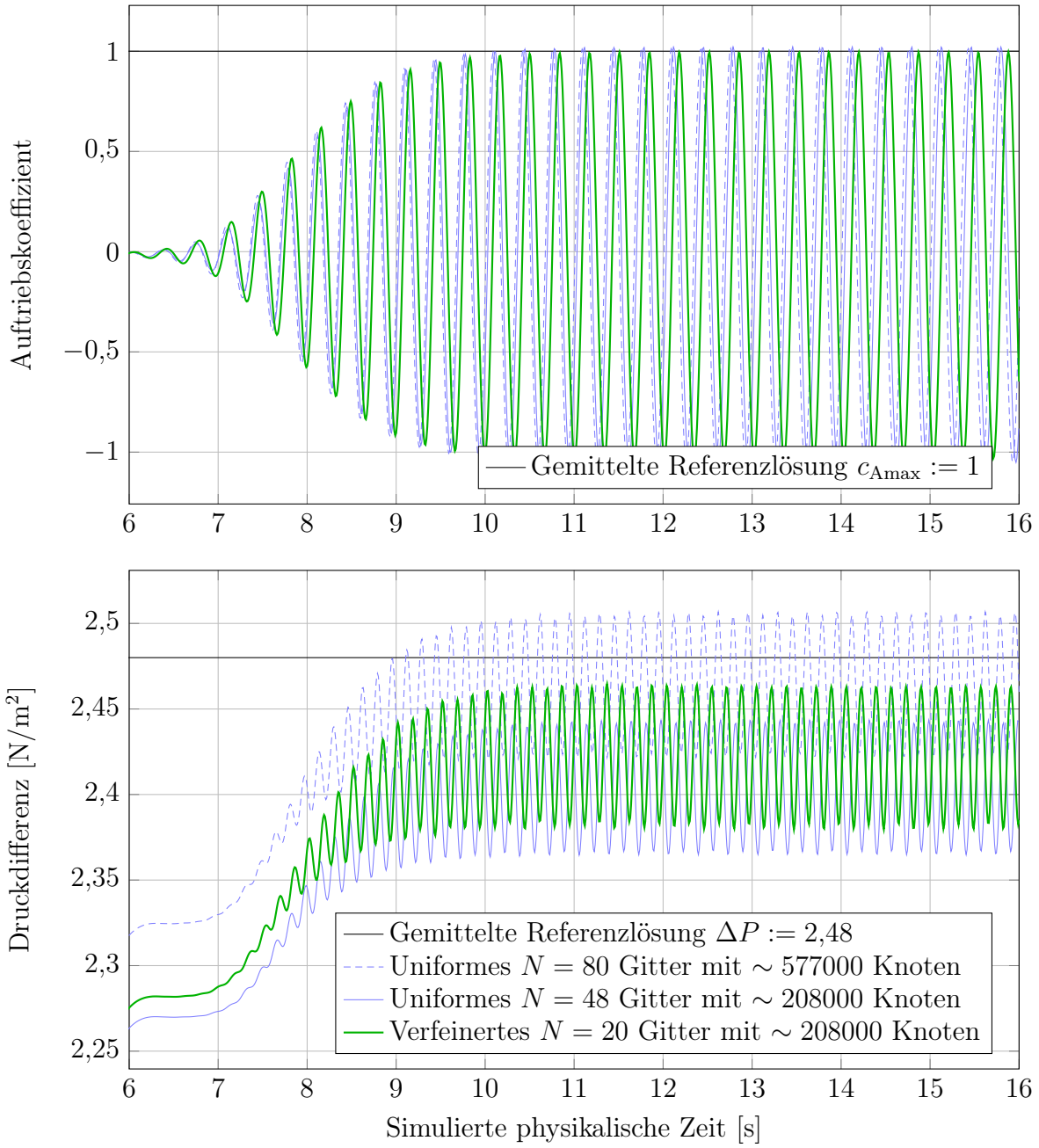


Abbildung 37: Auftriebskoeffizient und Druckdifferenz in feineren Simulationen

Das geeignet verfeinerte Gitter liefert demnach in zwei von drei Messwerten einen kleineren Fehler als ein äquivalent aufgelöstes $N = 48$ Gitter und besteht im Vergleich mit dem uniformen $N = 80$ Gitter erneut gegenüber einer signifikant mehr Knoten nutzenden Simulation. Selbst gegenüber eines mehr als zehnmal so viele Knoten aufwendenden uniformen $N = 160$ Gitter erhalten wir mit der verfeinerten Variante noch einen leicht kleineren Fehler in der Druckdifferenz.

	Uniform	Uniform	Uniform	Verfeinert	Referenzintervall [23]
$\widehat{c_w}$	3,2353	3,2444	3,2490	3,2004	[3,22, 3,24]
$\lvert\widehat{c_w} - 3{,}23\rvert$	0,0053	0,0144	0,0190	0,0296	[0, 0,01]
$\widehat{c_a}$	1,0019	1,0138	1,0212	0,9956	[0,99, 1,01]
$\lvert\widehat{c_a} - 1{,}0\rvert$	0,0019	0,0138	0,0212	0,0044	[0, 0,01]
ΔP	2,4997	2,5067	2,4435	2,4646	[2,46, 2,5]
$\lvert\Delta P - 2{,}48\rvert$	0,0197	0,0267	0,0365	0,0154	[0, 0,02]
$N =$	160	80	48	20 (max: 160)	
Knotenanzahl	2 298 014	576 758	207 862	208 031	

Tabelle 3: Kennzahlen höher aufgelöster Zylinderumströmungen mit *Bounce-Back*

Da alle vier getesteten Gitter gute Übereinstimmung zu den Referenzdaten von Schäfer und Turek aufweisen, fällt der Vorteil der Gitterverfeinerung im Allgemeinen jedoch geringer aus, als noch im Vergleich der niedrig aufgelösten Gitter aus Abbildung 33 und Tabelle 2. Darüber hinaus liegt die Knotenanzahl des betrachteten $N = 160$ Gitters weit oberhalb der maximalen referenzstiftenden Knotenzahlen [23, Tabelle 4], so dass wir hier an die Grenzen der Aussagekraft von Fehlern bezüglich dieser Werte stoßen.

Abschließend wollen wir nun noch einen besonderen verzerrenden Einfluss der, auf die Treppendiskretisierung des Zylinders angewandten, *Bounce-Back* Randbedingung durch Prüfung der Ergebnisse mit einer interpolierenden Randbedingung [16, Kap. 11.2.2.1] nach Bouzidi ausschließen. Anlass dazu liefert insbesondere die Verschlechterung des Widerstandsfehlers in unserem verfeinerten $N = 20$ Gitter gegenüber dem verfeinerten $N = 5$ Gitter sowie die starke Verbesserung des uniformen $N = 48$ Gitters verglichen mit dem $N = 40$ Gitter in Tabelle 2.

	Uniform	Uniform	Uniform	Verfeinert	Referenzintervall [23]
$\widehat{c_w}$	3,2278	3,2216	3,2109	3,2074	[3,22, 3,24]
$\lvert\widehat{c_w} - 3{,}23\rvert$	0,0023	0,0085	0,0113	0,0226	[0, 0,01]
$\widehat{c_a}$	0,9968	1,0004	0,9985	1,0093	[0,99, 1,01]
$\lvert\widehat{c_a} - 1{,}0\rvert$	0,0032	0,0004	0,0015	0,0093	[0, 0,01]
ΔP	2,5054	2,5072	2,4783	2,4804	[2,46, 2,5]
$\lvert\Delta P - 2{,}48\rvert$	0,0272	0,0254	0,0017	0,0004	[0, 0,02]
$N =$	160	80	48	20 (max: 160)	
Knotenanzahl	2 298 014	576 758	207 862	208 031	

Tabelle 4: Kennzahlen höher aufgelöster Zylinderumströmungen mit *Bouzidi*

Tatsächlich beobachten wir in Tabelle 4 mäßige Verbesserungen fast aller aus uniformen Gittern gewonnenen Kennzahlen. Ausnahmen bilden der verfeinerte Auftriebskoeffizient sowie der Druckfehler des uniformen $N = 160$ Gitters. Da dieser Druckfehler im Rahmen der betrachteten uniformen Gitter bei $N = 40$ sein Minimum annimmt und mit steigender Auflösung anwächst, deutet sich hier auch erneut die begrenzte Aussagekraft

der Referenzwerte an. Ergänzend dazu ist nur noch der Fehler in der Druckdifferenz des verfeinerten Gitters kleiner als der entsprechende Fehler der unverfeinerten Gitter – der Einfluss der Randbedingungen dominiert so im Vergleich von höher aufgelösten Gittern einen etwaigen positiven Effekt von Gitterverfeinerung.

Fassen wir die in Tabelle 3 und 4 verglichenen Ergebnisse zusammen, empfiehlt sich unter Betrachtung der durch Verfeinerung geschaffenen zusätzlichen Komplexität und Performanceproblematik (vgl. Kapitel 5.2.2) auch hier wieder, Vorsicht beim Einsatz von verfeinerten Gittern walten zu lassen. So lässt sich kein allgemeiner Genauigkeitsvorteil von verfeinerten Gittern gegenüber äquivalent aufgelösten uniformen Gittern feststellen. Ein Solcher existiert in der betrachteten Zylinderumströmung nur eindeutig, wenn ein uniformes Gitter sich nahe an der Divergenz bewegt.

Kann ein Problem mit den verfügbaren Rechenressourcen durch ein uniformes Gitter behandelt werden, sollte beim Versuch der Ergebnisverbesserung die neue Möglichkeit der lokalen Gitterverfeinerung nicht zur Vernachlässigung von ebenso wichtigen Faktoren wie der verwendeten Randbedingungen führen.

Rückblickend hat die Evaluation unseres Gitterverfeinerungsverfahrens anhand der Zylinderumströmung dessen Vorteil spendende Verwendbarkeit unter Einschränkungen demonstriert. Während bei der Betrachtung der einfachen Poiseuilleströmung Sorgen bezüglich der Ergebnisgenauigkeit verfeinerter Gitter geweckt wurden, konnten diese in der Betrachtung einer komplexeren und so einen Verfeinerungsbedarf besser begründenden Strömungssituation weitestgehend beigelegt werden. Während die aufgeworfene Frage nach der korrekten Behandlung von Randbedingungen im Übergangsbereich noch offen ist, konnten wir unter Vermeidung dieser unklaren Situation ein sinnvolles, durch ein formales Kriterium informiertes, lokal verfeinertes Gitter beschreiben.

6 Fazit

Zusammenfassend hinterlässt die Anwendung Gitterverfeinerung in Lattice Boltzmann Methoden einen vielversprechenden Eindruck mit klaren Fragestellungen für mögliche Wege des Aufbaus auf dieser Arbeit. So hat sich die theoretische Formulierung des Verfahrens von Lagrava et al. als praktisch gut umsetzbar erwiesen und die zugehörigen Ergebnisse in *Advances in Multi-domain Lattice Boltzmann Grid Refinement* [17] konnten reproduziert werden.

Die existierenden Strukturen der LBM-Bibliothek OpenLB ließen sich gut mit einer lokalen Verfeinerung des Gitters vereinbaren und der gewählte Implementierungsansatz, der Betrachtung von verfeinernden Gittern als weitestgehend eigenständige Simulationen mit besonderen Randkonditionen in den Übergangsbereichen, bewährte sich sowohl in der initialen Umsetzung des Verfahrens als auch in der Verwendung der resultierenden Schnittstelle zur Entwicklung der Evaluationsbeispiele.

Die im Rahmen dieser Beispiele betrachtete Zylinderumströmung bestätigte das Verfahren im Vergleich von verfeinerten und uniformen Simulationen als gewinnbringend einsetzbar – aerodynamische Kennzahlen mit guter Übereinstimmung zu hochwertigen Vergleichsdaten [23] konnten durch gezielte Verfeinerung mit einer deutlich reduzierten Anzahl von Gitterknoten simuliert werden. Dies ist insbesondere auch im Hinblick darauf beachtlich, dass ein mit äquivalenter Anzahl von Gitterknoten uniform aufgelöstes Gitter signifikant schlechtere und zur Divergenz neigende Ergebnisse lieferte. In diesem Kontext offenbarte sich der Einsatz von Gitterverfeinerung zur gezielten Vermeidung von Divergenz im Ausflussbereich als zusätzliches Anwendungsgebiet.

Problematischer erwies sich das Verfahren in der Anwendung auf eine einfache und analytisch lösbare Poiseuilleströmung. Im Verlauf der Untersuchung dieser grundlegenden Strömungssituation kristallisierte sich das in [17] unbeachtete Zusammenspiel von Randkonditionen und Übergangsbereichen durch Vergrößerung des Fehlers um bis zu eine Größenordnung als kritischer und weiterer theoretischer Untersuchung bedürfender Aspekt heraus. Auch bei Vermeidung von Randbedingungen in Auflösungsübergängen konnte ein negativer Einfluss von Gitterverfeinerung auf die Reproduktion der analytischen Lösung festgestellt werden. Es wurde somit klar, dass Gitterverfeinerung in dem hier untersuchten Rahmen keinesfalls unvorsichtig und in der Abwesenheit konkreter Notwendigkeiten angewendet werden sollte. Die zusätzliche Komplexität und Fehlerquelle einer lokalen Verfeinerung sollte in sinnvollen Anwendungen eben dieser durch Vorteile wie bessere Geometriediskretisierung oder Zwänge wie beschränkte Rechenressourcen aufgewogen oder geringstenfalls begründet werden.

Im Kontext der sinnvollen Anwendung von Gitterverfeinerung sowie der konkreten Strukturierung des heterogenen Gitters erwies sich das in *Automatic grid refinement criterion for lattice Boltzmann method* [18] entwickelte Gitterverfeinerungskriterium als sehr gutes Maß zur Bewertung der lokalen Simulationsqualität. Nicht nur konnten in ihrer Anzahl beschränkte Knotenfreiheitsgrade mittels dieses Kriteriums formal fundiert problembezogen umverteilt werden, sondern auch problematische und zur Divergenz neigende Bereiche der Simulation ließen sich frühzeitig erkennen und vermeiden.

Abschließend ergeben sich somit die folgenden theoretischen Fragestellungen zur weiteren Verfolgung in absteigender Dringlichkeit:

1. Wie müssen Randkonditionen in Übergangsbereichen behandelt werden?
2. Wie kann die Rechenlast zur parallelen Simulation der verfeinernden Gitter besser verteilt werden?
3. Wie verhält sich das betrachtete Verfahren bei Übertragung auf dreidimensionale Lattice Boltzmann Methoden?
4. Was gilt es bei Variation der Verfeinerung im Verlauf der Simulation zu beachten?

Bezüglich der Weiterentwicklung des nun in OpenLB existierenden Gitterverfeinerungsframework bieten sich darüber hinaus Ansatzpunkte in Form der Fragen:

1. Wie kann der Kommunikationsaufwand zur Gitterkopplung reduziert werden?
2. Welche Möglichkeiten gibt es zur weiteren Abstraktion des Simulationsaufbaus?
3. Wie lassen sich die existierenden Konzepte aus der Optimierung und Mehrphasenkopplung sinnvoll mit den neuen Möglichkeiten zur Gitterverfeinerung verbinden?
4. Kann die Verfeinerungsschnittstelle unabhängig des konkreten Verfahrens direkt auf drei Dimensionen übertragen werden?

Diese Auswahl von Möglichkeiten für anknüpfende Arbeiten beschließe an dieser Stelle unsere Betrachtung von gitterverfeinerten Lattice Boltzmann Methoden in OpenLB. Die angestrebten Ziele der theoretischen Ausarbeitung, flexiblen praktischen Umsetzung und experimentellen Evaluation wurden erreicht und Gitterverfeinerung verspricht den Funktionskatalog der freien Lattice-Boltzmann-Bibliothek OpenLB um ein nützliches und ausbaufähiges Werkzeug erweitert zu haben.

Abbildungsverzeichnis

1 Strömung in der komplexen Geometrie der menschlichen Nase [13] 1
2 Umgebung einer Zelle in D2Q9 5
3 Strömung der im Kollisionsschritt relaxierten Verteilungen 7
4 Teilgitter in der Multi-Grid Herangehensweise 9
5 Teiligitter in der Multi-Domain Herangehensweise 9
6 Koinzidierende Gitter 10
7 Zueinander versetzte Gitter ohne überlappende Knoten 10
8 Multi-Domain Herangehensweise mit Übergangsbereich [17, vgl. Abb. 3] . 11
9 Schematische Darstellung der Simulationsgebiete der Gitter 12
10 Mit Forderung (3.1) ausgeschlossener Übergangsbereich [17, vgl. Abb. 9] 13
11 Skizze des Übergangsbereich [17, vgl. Abb. 4] 14
12 Einzugsgebiet der Restriktionsoperation $\boldsymbol{r}(7, x_{f\to g})$ 19
13 Stützstellen der Interpolation im Übergangsbereich 21
14 Stützstellen der Interpolation im Detail 21
15 Interpolation in Ecken und Enden des feinen Gitters 22
16 Übersicht des Verfeinerungsalgorithmus mit Invariante 24
17 Übersicht der zu vervollständigenden Knoten 25
18 Schematische Darstellung der Rohrströmung 37
19 Gitterstruktur einer halbseitig verfeinerten Rohrströmung 38
20 Geschwindigkeiten in x-Richtung im simulierten Poiseuille-Fluss 38
21 Geschwindigkeits- und Druckfehler in verschiedenen Gittern 39
22 Druckverlauf bei linearer und kubischer Interpolation [17, vgl. Abb. 11] . 40
23 Vergleich des vertikalen Geschwindigkeitsprofil bei $x = 3$ 41
24 Schematische Darstellung des umströmten Zylinder [23, vgl. Abb. 1] . . . 43
25 Uniform mit $N = 20$ aufgelöstes Strömungsbild zu $t = 16s$ 43
26 Einfach verfeinertes Strömungsbild zu $t = 16s$ 44
27 Uniform mit $N = 40$ aufgelöstes Strömungsbild zu $t = 16s$ 44
28 Verfeinerungskriterium in einem uniform mit $N = 20$ aufgelösten Gitter . 46
29 Verfeinerungskriterium in einem uniform mit $N = 40$ aufgelösten Gitter . 46
30 Verfeinerungskriterium in einem einfach verfeinerten $N = 20$ Gitter . . . 47
31 Verbesserte Verfeinerungsstruktur um den Zylinder 47
32 Stabile Zylinderumströmung mit $N = 5$ durch Ausgangsverfeinerung . . . 48
33 Vergleich von uniform und problembezogen verteilten Freiheitsgraden . . 48
34 Zeitlicher Verlauf des Strömungswiderstands- und Auftriebskoeffizienten . 51
35 Zeitlicher Verlauf der Druckdifferenz 52
36 Strömungswiderstandskoeffizient in feineren Simulationen 53
37 Auftriebskoeffizient und Druckdifferenz in feineren Simulationen 54

Tabellenverzeichnis

1 Vergleich der parallelen Ausführung auf einem Shared Memory System . 49
2 Vergleichswerte zweier maximal $N = 40$ aufgelöster Gitter 52
3 Kennzahlen höher aufgelöster Zylinderumströmungen mit *Bounce-Back* . 55
4 Kennzahlen höher aufgelöster Zylinderumströmungen mit *Bouzidi* 55

Literatur

[1] C.K. Aidun und J.R. Clausen. “Lattice-Boltzmann Method for Complex Flows”. In: *Annual Review of Fluid Mechanics* 42 (1 Jan. 2010). DOI: 10.1146/annurev-fluid-121108-145519.

[2] H. Amann und J. Escher. *Analysis I*. Birkhäuser, 2006. ISBN: 3-7643-7755-0.

[3] H. Amann und J. Escher. *Analysis II*. Birkhäuser, 2006. ISBN: 3-7643-7105-6.

[4] Y. Bao und J. Meskas. *Lattice Boltzmann Method for Fluid Simulations*. 2011.

[5] H. Chen, O. Filippova, J. Hoch, K. Molvig, R. Shock, C. Teixeira und R. Zhang. “Grid refinement in lattice Boltzmann methods based on volumetric formulation”. In: *Physica A: Statistical Mechanics and its Applications* 362.1 (2006). Discrete Simulation of Fluid Dynamics, S. 158–167. ISSN: 0378-4371. DOI: 10.1016/j.physa.2005.09.036.

[6] L. Dagum und R. Menon. “OpenMP: An Industry-Standard API for Shared-Memory Programming”. In: *IEEE Comput. Sci. Eng.* 5.1 (Jan. 1998), S. 46–55. ISSN: 1070-9924. DOI: 10.1109/99.660313.

[7] A. Dupuis und B. Chopard. “Theory and applications of an alternative lattice Boltzmann grid refinement algorithm”. In: *Phys. Rev. E* 67 (6 Juni 2003). DOI: 10.1103/PhysRevE.67.066707.

[8] G. Eitel-Amor, M. Meinke und W. Schröder. “A lattice-Boltzmann method with hierarchically refined meshes”. In: *Computers & Fluids* 75 (2013), S. 127–139. ISSN: 0045-7930. DOI: 10.1016/j.compfluid.2013.01.013.

[9] A. Fakhari, M. Geier und T. Lee. “A mass-conserving lattice Boltzmann method with dynamic grid refinement for immiscible two-phase flows”. In: *Journal of Computational Physics* 315 (2016), S. 434–457. ISSN: 0021-9991. DOI: 10.1016/j.jcp.2016.03.058.

[10] O. Filippova und D. Hänel. “Grid Refinement for Lattice-BGK Models”. In: *Journal of Computational Physics* 147.1 (1998), S. 219–228. ISSN: 0021-9991. DOI: 10.1006/jcph.1998.6089.

[11] E. Gabriel et al. “Open MPI: Goals, Concept, and Design of a Next Generation MPI Implementation”. In: *Proceedings, 11th European PVM/MPI Users’ Group Meeting*. Budapest, Hungary, Sep. 2004, S. 97–104.

[12] D. Hänel. *Molekulare Gasdynamik*. Springer, 2004. ISBN: 978-3-540-35047-7.

[13] T. Henn, M.J. Krause und S. Zimny. *OpenLB Showcase: Respiration Nose*. URL: https://www.openlb.net/respiration-nose.

[14] M.J. Krause. “Fluid flow simulation and optimisation with Lattice Boltzmann methods on high performance computers : applications to the human respiratory system”. Karlsruhe, KIT. Diss. 2010.

[15] M.J. Krause, A. Mink, R. Trunk, F. Klemens, M.-L. Maier, M. Mohrhard, A. Claro Barreto, M. Haußmann, M. Gaedtke und J. Ross-Jones. *OpenLB Release 1.2: Open Source Lattice Boltzmann Code*. 2018. URL: `http://www.openlb.net/download`.

[16] T. Krüger, H. Kusumaatmaja, A. Kuzmin, O. Shardt, G. Silva und E.M. Viggen. *The Lattice Boltzmann Method: Principles and Practice*. Graduate Texts in Physics. Springer International Publishing, 2017. ISBN: 978-3-319-44647-9.

[17] D. Lagrava, O. Malaspinas, J. Latt und B. Chopard. "Advances in Multi-domain Lattice Boltzmann Grid Refinement". In: *Journal of Computational Physics* 231.14 (Mai 2012). ISSN: 0021-9991.

[18] D. Lagrava, O. Malaspinas, J. Latt und B. Chopard. "Automatic grid refinement criterion for lattice Boltzmann method". In: *ArXiv e-prints* (Juli 2015). arXiv: `1507.06767 [physics.flu-dyn]`.

[19] Ching-Long Lin und Yong G. Lai. "Lattice Boltzmann method on composite grids". In: *Phys. Rev. E* 62 (2 Aug. 2000), S. 2219–2225. DOI: `10.1103/PhysRevE.62.2219`.

[20] Y. Peng, C. Shu, Y. T. Chew, X. D. Niu und X. Y. Lu. "Application of Multi-block Approach in the Immersed Boundary-lattice Boltzmann Method for Viscous Fluid Flows". In: *Journal of Computational Physics* 218.2 (Nov. 2006), S. 460–478. ISSN: 0021-9991. DOI: `10.1016/j.jcp.2006.02.017`.

[21] M. Rohde, D. Kandhai, J.J. Derksen und H. Van den Akker. "A generic, mass conservative local grid refinement technique for lattice-Boltzmann schemes". In: *International Journal for Numerical Methods in Fluids* 51 (Juni 2006), S. 439–468. DOI: `10.1002/fld.1140`.

[22] J. Tölke, M. Krafczyk und E. Rank. "A Multigrid-Solver for the Discrete Boltzmann Equation". In: *Journal of Statistical Physics* 107.1 (Apr. 2002), S. 573–591. ISSN: 1572-9613. DOI: `10.1023/A:1014551813787`.

[23] S. Turek und M. Schäfer. *Recent Benchmark Computations of Laminar Flow Around a Cylinder*. 1996.

[24] *OpenLB User Guide, Associated to Release 1.2 of the Code*. 2018. URL: `https://www.openlb.net/user-guide`.

[25] *The VTK User's Guide*. Kitware, 2010. ISBN: 978-1-930-93423-8. URL: `https://vtk.org/vtk-users-guide/`.

Erklärung

Ich versichere wahrheitsgemäß, die Arbeit selbstständig verfasst, alle benutzten Hilfsmittel vollständig und genau angegeben und alles kenntlich gemacht zu haben, was aus Arbeiten anderer unverändert oder mit Abänderungen entnommen wurde, sowie die Satzung des KIT zur Sicherung guter wissenschaftlicher Praxis in der jeweils gültigen Fassung beachtet zu haben.

Karlsruhe, den 25. März 2019