Masterarbeit

zur Erlangung des akademischen Grades

Master of Science (M.Sc.)

Soft Objects in Newtonian and Non-Newtonian Fluids - a Computational Study of Bubbles and Capsules in Flow

---

vorgelegt von

Fabian Häusl
Matrikelnummer: 1540550

Abgabedatum: 9. Dezember 2021

Erstprüfer: Prof. Dr. Stephan Gekle
Zweitprüfer: Prof. Dr. Michael Wilczek

Biofluid Simulation and Modeling
Theoretische Physik VI
Fakultät für Mathematik, Physik und Informatik
Universität Bayreuth

UNIVERSITÄT
BAYREUTH

Zusammenfassung

Die meisten biologischen Flüssigkeiten sind viskoelastisch, was komplexe Phänomene wie scherabhängige Viskositäten oder nichtverschwindende Speichermodule mit sich bringt. Weiche Objekte, die in diesen Flüssigkeiten eingebettet sind, verursachen komplizierte Strömungsgeometrien und Spannungsverteilungen, bei denen Computermodelle helfen können, die zugrunde liegenden Wechselwirkungen zu verstehen. Zu diesem Zweck wird ein hybrider Algorithmus vorgestellt, der die Lattice-Boltzmann-Methode (LBM) mit dem Finite-Volumen (FV) Schema kombiniert, wobei die Erstere zur Lösung der hydrodynamischen Gleichungen und das Letztere zur Modellierung der Polymerdynamik verwendet wird. Um die verschiedenen viskoelastischen Eigenschaften eines Polymerfluids gesondert betrachten zu können, wird die Kinetik der gelösten Polymere wahlweise durch zwei verschiedene konstitutive Gleichungen erfasst. Diese beschreiben viskoelastische Flüssigkeiten, die lineare Federkräfte (Oldroyd-B) oder endlich dehnbare nichtlineare elastische Federn mit Peterlin-Schluss (FENE-P) beinhalten. Indem der Algorithmus durch sorgfältige Anpassung mit verschiedenen LBM Randbedingungstypen kompatibel gemacht wird, kann das korrekte rheologische Verhalten bei mehreren Validierungen mit hoher Genauigkeit reproduziert werden, einschließlich oszillierender Scherung, Einsetzen der Poiseuille-Strömung, Elongationsströmung und Beendigung des linearen Scherflusses. Die Kopplung dieses Algorithmus mit der Immersed Boundary-Methode (IBM) ermöglicht die Modellierung mitschwimmender Kapseln, die sowohl bei Newtonscher als auch bei Oldroyd-B-Scherströmung gut mit Literaturdaten übereinstimmen. Auf dieser Grundlage werden in dieser Arbeit numerische Simulationen von Kapseln in FENE-P-Fluiden vorgestellt, bei denen die Einsatzmöglichkeiten der IB-LBM noch nie quantifiziert wurden. Hierbei wird der zu erwartende Abfall der dimensionslosen Rotationsfrequenz von Kapseln bei Weissenberg-Zahl $\mathrm{Wi} \approx 1$ reproduziert, die zeitabhängige Kapsel-Deformation unter Scherung untersucht und eine Vorhersage zum Verhalten von Zellen unter Bioprinting-Bedingungen gemacht. Die Komplexität wird gesteigert, indem das Modell auf Kapseln mit unterschiedlichen inneren und äußeren Fluideigenschaften erweitert wird, wobei sowohl Spannungserhaltung als auch eine hohe Advektionsgenauigkeit an der Kapselgrenzschicht angestrebt werden.

Im Falle von Blasen, die in Newtonschen Flüssigkeiten suspendiert sind, wird keine biologische Grenzschicht zur Trennung der beiden Phasen benötigt. Stattdessen spielt nun die Oberflächenspannung eine dominante Rolle, die mit Hilfe eines Volume-of-Fluid (VoF) Ansatzes in die LBM integriert wird, wodurch das Problem einer aus Flüssigkeit und Gas bestehenden Zweiphasenströmung auf ein Einphasenfluid mit freier Oberfläche reduziert wird. Die Druckänderung aufgrund der (De-)Kompression des Blasenvolumens wird unter Einbeziehung der idealen Gasgleichung approximiert. Insbesondere wenn mehrere Blasen vorhanden sind, wird das Tracking der topologischen Veränderungen der freien Oberfläche unter Berücksichtigung der Verschmelzung (Merge) oder Teilung (Split) von Blasen zu einer schwierigen Aufgabe. In dieser Arbeit wird ein auf dem Hoshen-Kopelman-Algorithmus basierender Algorithmus zur effizienten Verwaltung von Merge/Split-Prozessen vorgestellt, der die Rechenlast gegenüber rein CPU-basierten Ansätzen erheblich reduziert, indem er diese Prozesse über GPU-kompatible Kriterien triggert. Die Genauigkeit des Algorithmus wird anhand der Rayleigh-Plesset-Gleichung überprüft. Darüber hinaus wird die Fähigkeit des Algorithmus, die korrekte Form und Geschwindigkeit von Blasen beim

Aufsteigen zur Atmosphären-Oberfläche zu reproduzieren, über einen weiten Bereich von Morton- und Bond-Zahlen untersucht. Zusammen mit der Simulation einer platzenden Blase wird das Modell bezüglich seiner möglichen Anwendung in der Mikroplastikforschung betrachtet, wo Blasen vermutlich eine wichtige Rolle beim Austausch von Partikeln an der Luft-Wasser-Grenzfläche spielen.

Abstract

Most biological fluids are viscoelastic, giving rise to complex phenomena like shear-dependent viscosities or non-zero storage moduli. Soft objects suspended in these fluids cause complicated flow geometries and stress distributions, where computational models can help to understand the underlying interactions. To this end, a hybrid algorithm combining the lattice Boltzmann method (LBM) with the finite volume (FV) scheme is proposed, where the former is used to solve the hydrodynamic equations while the latter models the polymer dynamics. To be able to separate the different viscoelasticity properties within a polymeric liquid, the kinetics of the dissolved polymers is captured by two different constitutive equations, modeling viscoelastic fluids consisting of linear dumbbells (Oldroyd-B) or finitely extensible non-linear elastic dumbbells with Peterlin closure (FENE-P). By carefully adapting the algorithm for compatibility with various types of LBM boundary conditions, the correct rheological behavior in several validation setups is reproduced with high accuracy, including oscillatory shear, onset of poiseuille flow, elongational flow and cessation of steady shear. Coupling this algorithm to the immersed boundary method (IBM) allows for the modeling of immersed capsules, which are shown to match well with literature data both in Newtonian and Oldroyd-B shear flow. With this basis, this thesis proposes numerical simulations of capsules in FENE-P fluids, where the capabilities of the IB-LBM were never quantified before. Here, the well-known drop of the dimensionless rotational frequency of capsules at Weissenberg number $\mathrm{Wi} \approx 1$ is reproduced, the time-dependent capsule deformation under shear is explored and a prediction on the behavior of cells under bioprinting conditions is made. Increasing the complexity, the model is extended to capsules with differing interior and exterior fluid properties, whilst at the same time accounting for both conservation of stress and high advection accuracy at the capsule boundary.

In the case of bubbles suspended in Newtonian fluids, no biological boundary is needed to separate the two phases. Instead, surface tension now plays a dominant role, which is integrated into the LBM using a Volume-of-Fluid (VoF) approach, reducing the problem of a liquid-gas two-phase flow to a single-phase fluid with a free interface. The change in pressure due to (de-)compression of the bubble volume is approximated by incorporation of the ideal gas equation. Especially in the presence of multiple bubbles, tracking the topological changes of the free surface while accounting for merging or splitting of bubbles becomes a difficult task. This thesis presents an algorithm based on the Hoshen-Kopelman algorithm to efficiently manage merge/split processes, significantly reducing the computational load compared to purely CPU-based approaches by triggering these processes via GPU-compatible criteria. The accuracy of the algorithm is benchmarked using the Rayleigh-Plesset equation. Furthermore, its capability of reproducing the correct shape and speed of bubbles rising to the atmospheric surface is investigated over a long range of Morton and Bond numbers. Together with the simulation of a bursting bubble, this tests the model for its application in microplastics research, where bubbles are believed to play an important role in the exchange of particles at the air-water interface.

Contents

1 Introduction

Tissue engineering aims to combine bioactive materials with cells to compose three-dimensional structures that can be used in medicine. Applications range from the regeneration of injured tissues to the replacement of failing organs. A young and evolving technique within this field is bioprinting, where cell-laden hydrogels termed "bio-inks" are used in specifically designed printers that enable the fabrication of complex biohybrid structures with high shape fidelity [1, 2]. The potential of this approach has recently been demonstrated by printing cellularized human hearts with major blood vessels [3]. The fact that the hearts had to be down-scaled to a diameter of 14 mm highlights the existing trade-off between print speed and cell viability. Print speed is limited, since for high fluid velocities a large shear stress is generated in the hydrogel, which can cause damage to the cells. This is especially true for the most common and affordable bioprinting technique, the extrusion-based printing, which will be examined in this thesis from a theoretical perspective. In contrast to droplet-based and laser-assisted methods, extrusion-based methods employ a continuous dispersion. Moreover, the latter allows the use of bio-inks with high viscosities in the range from 10 Pa s to 10^4 Pa s and high cell densities. However, with nozzle diameters typically being greater than 100 µm only a relatively coarse resolution can be achieved with this method.

In extrusion-based printing, an important viscoelastic characteristic of favorable bio-inks is the shear-thinning property. It allows for a reduction of physical stresses during printing, without compromising the mechanical integrity of the manufactured structure [4 – 6]. Much research is done in order to adapt the rheological properties like shear-dependent viscosity, storage and loss modulus by testing different polymer classes, altering polymer concentration and molecular weight or by applying (pre-)crosslinking techniques. However, it is still poorly understood how a change in the rheological properties of the bio-inks affects the overall system of printing nozzle, hydrogel and cells. This is because quantities such as the time-dependent deformation of cells or the stress of the surrounding hydrogel are difficult to measure experimentally. By employing an HPC approach, this thesis aims to make these quantities accessible in a numerical manner.

To provide the necessary theoretical background, the basic equations of Newtonian fluids are summarized in chapter 3. Furthermore, in chapter 4 two popular polymer models, namely the Oldroyd-B and the FENE-P model, are introduced. In the methods part of this thesis the lattice Boltzmann method (LBM) for the simulation of Newtonian fluids and the immersed boundary method (IBM) for the modeling of soft deformable objects are explained. With these foundations, finally, in chapter 10 a numerical method for the simulation of cell-like capsules dissolved in viscoelastic fluids is developed, implemented, validated and brought to first applications.

Another microfluidics problem that can be tackled using the LBM is the transport of microplastics at the water-air interface. Due to the plastic pollution of the marine environment, a high load of small-sized plastic debris with sizes in the order of 10 – 1000 µm are found in the ocean. Due to the interplay of their hydrophobicity and low mass density they have been found to accumulate below the ocean surface. Accounting only for the three most-littered plastics, a recent

study estimated the combined mass of microplastics suspended in the top 200 m of the Atlantic Ocean to be about 10 – 20 Million Tonnes [7]. The sea surface microlayer right at the top of the ocean undergoes a particular enrichment of plastic debris, with bubble scavenging proposed as a possible explanation of the transport mechanism [8]. The observation of airborne microplastic particles in the remote marine atmosphere indicates that another transport mechanism exists, leading from the sea surface microlayer into the atmosphere [9]. As a possible mediator of this transport, again bubbles are considered, since they eject a number of small droplets into the atmosphere during bursting.

In order to investigate these two processes theoretically, a bubble model is developed in this thesis. To this end, the computational models of a recent study are applied, which investigated raindrops as transport vehicles for the transition of microplastic particles from ocean water to the atmosphere, using the LBM extended by a Volume-of-Fluid (VoF) approach and the IBM [10]. Chapter 6 provides an overview of the VoF method. In chapter 9, a bubble model is introduced with a focus on the necessary detection of bubble merge/split processes, where the developed method makes heavy use of the Hoshen-Kopelman (HK) algorithm. In order to assess the accuracy of the model with regard to the investigation of the transport processes described above, the subsequent validations and tests of the model focus mainly on the correct behavior during the ascending of the bubbles to the atmospheric interface and the burst process.

Both microfluidics topics are approached by extending the capabilities of the computational fluid dynamics (CFD) software FluidX3D [11]. To this end, algorithms known from literature are implemented from scratch and new problem-specific algorithms are developed. Chapter 8 briefly describes what was considered in the design of these algorithms in order to make them compatible with high performance computing (HPC) prerequisites and to be able to execute them in parallel on GPUs. Furthermore, for validation purposes some setups are compared against simulations conducted with the LBM implementation of the open source software package ESPResSo [12].

2 Nomenclature and notation

2.1 Declaration of symbols

The following tables list the frequently used symbols for physical quantities, lattice Boltzmann quantities and dimensionless numbers occurring in this thesis.

Physical quantities:

Symbol	Physical unit	Meaning
ρ	$\mathrm{kg\,m^{-3}}$	Fluid mass density
η, η_s & η_p	$\mathrm{Pa\,s}$	*Total*, *solvent* & *polymer* dynamic shear viscosity
ν	$\mathrm{m^2\,s^{-1}}$	Total kinematic shear viscosity
η_E	$\mathrm{Pa\,s}$	Extensional viscosity
p & p_B	Pa	*Fluid* & *bubble* pressure
σ	$\mathrm{kg\,s^{-2}}$	Surface tension
$\underline{\sigma}$, $\underline{\sigma}_s$ & $\underline{\tau}_p$	Pa	*Total*, *solvent* & *polymer* stress tensor
τ_{ij}	Pa	A component of stress tensor $\underline{\tau}_p$ (suppressing the subscript)
$\underline{c}$	-	Conformation tensor
$\underline{D}$	$\mathrm{s^{-1}}$	Deformation rate tensor
$\boldsymbol{F}$	$\mathrm{N\,m^{-3}}$	Body force
$\boldsymbol{\mathcal{F}}$	N	Force
θ	K	Temperature
n	$\mathrm{m^{-3}}$	Number density (e.g. of dumbbells)
k_B	$\mathrm{m^2\,kg\,s^{-2}\,K^{-1}}$	Boltzmann constant
R_sp	$\mathrm{m^2\,s^{-2}\,K^{-1}}$	Specific gas constant
g	$\mathrm{m\,s^{-2}}$	Gravitational acceleration
G, G' & G''	Pa	*Elastic*, *storage* & *loss* modulus
κ_S & $\kappa_{S,1}$, $\kappa_{S,2}$	$\mathrm{N\,m^{-1}}$	Shear modulus of *neo-Hookean* model & shear moduli of *Skalak* model
λ_p	s	Polymer relaxation time
$\dot{\gamma}$	$\mathrm{s^{-1}}$	Shear rate
$\dot{\epsilon}$	$\mathrm{s^{-1}}$	Elongation rate
β & $\hat{\beta}$	-	*Polymer* & *Newtonian* fraction of total viscosity
b	-	Finite extensibility parameter

Lattice Boltzmann quantities:

Symbol	Meaning
f_i	Particle distribution function or population (pre-collision)
f_i^*	Population (post-collision)
$f_{\bar{i}}$	Population with $\boldsymbol{c}_{\bar{i}} = -\boldsymbol{c}_i$
f_i^{eq}	Equilibrium population
Ω_i	Collision operator
c_s	Lattice speed of sound
w_i	Velocity weight
$\boldsymbol{c}_i$	Discretized grid velocity
q & d	*Number of discrete velocities* contained in a set & its *dimensionality*
Δt	Discrete time step
Δx	Lattice spacing
τ_{r} & $\tau_{\mathrm{r},+}$, $\tau_{\mathrm{r},-}$	One *SRT* relaxation time & two *TRT* relaxation times
$\underline{S}$	Matrix containing MRT relaxation times
$\underline{M}$	MRT matrix transforming from population space to moment space
φ	Fill level
F_F, F_I & F_G	Node types: *fluid*, *interface* & *gas*
I	Field storing bubble IDs
$\rho_{\mathrm{SI}} = 1000\,\mathrm{kg\,m^{-3}}$ vs. $\rho = 1.0$	When both SI and lattice quantities occur in the same context, SI quantities are marked by a subscript and have units, while lattice quantities are unitless.

Dimensionless numbers:

Symbol	Meaning
Kn	Knudsen number
Re	Reynolds number
Ca	Capillary number
De	Deborah number
Wi	Weissenberg number
Bo	Bond number (also called Eötvös number)
Mo	Morton number
t^*	Dimensionless time

2.2 Acronyms

The following abbreviations for important equations and methods will be used.

Acronym	written out
LB	lattice Boltzmann
LBM	lattice Boltzmann method
NS	Navier-Stokes
NSE	Navier-Stokes equation
BGK	Bhatnagar-Gross-Krook
FENE-P	finitely extensible nonlinear elastic model with Peterlin closure
UCM	upper-convected Maxwell
FD	finite difference
FV	finite volume
FVM	finite volume method
FE	finite element
HK	Hoshen-Kopelman
SRT	single relaxation time
TRT	two relaxation time
MRT	multi relaxation time
IBM	immersed boundary method
IB-LBM	immersed boundary lattice Boltzmann method
BIM	boundary integral method
RBC	red blood cell
CPU	central processing unit
GPU	graphics processing unit
PCIe	peripheral component interconnect express
VoF	Volume-of-Fluid
HPAM	hydrolyzed Poly-Acrylamide
POx	Poly(2-oxazoline)
CTU	corner transport upwind
COM	center of mass

2.3 Mathematical notation and conventions

The following table lists the conventions and mathematical notation used in this thesis.

Notation	Explanation
$\boldsymbol{a}$	A column vector
$\boldsymbol{a}^{\mathrm{T}}$	Transpose of a vector
$\hat{\boldsymbol{n}}$	Normal vector
$\hat{\boldsymbol{e}}_{\boldsymbol{i}}$	i-th unit vector
a_α	Subscript for vector components (or tensor components i.g.)
$a_\alpha b_\alpha$	Einstein summation convention applies unless stated otherwise
$\underline{A}$	A matrix (or tensor i.g.)
$\underline{\mathrm{I}}$	Identity matrix
$\boldsymbol{a} \cdot \boldsymbol{a}$	Scalar product
$\boldsymbol{aa}$	Outer product
$\delta_{\alpha\beta}$	Kronecker delta
∇	Nabla operator
Δ	Laplace operator
$\overset{\nabla}{\underline{A}}$	Upper convected derivative
$\mathrm{Tr}\underline{A}$	Trace of matrix $\underline{A}$
$\langle\rangle_\psi$	Expectation value using probability density ψ
$\Re(a+ib)$	Real part of imaginary number

3 Newtonian fluids - two mathematical descriptions

Before aiming to describe Newtonian fluids in terms of mathematics, one should be aware of the different scales on which such a description can happen. Concerning length scales there is the size of the fluid atoms and molecules l_a, the mean free path (distance travelled between two successive collisions) l_mfp and the typical scale l_g for gradients of macroscopic quantities (density, temperature etc.). The typical hierarchy of these length scales is $l_\mathrm{a} \ll l_\mathrm{mfp} \ll l_\mathrm{g}$. These scales correspond to a microscopic (i.e. molecular), mesoscopic or macroscopic (continuum picture) description of fluids. "Mesoscopic" here refers to a description in-between microscopic and macroscopic, where distributions (i.e. representative collections) of molecules rather than the molecules itself are tracked. Closely related to this hierarchy of length scales is the hierarchy of time scales. There is the collision time $t_\mathrm{a} \propto l_\mathrm{a}/u_\theta$ where $u_\theta = (k_\mathrm{B}\theta/m)^{1/2}$ is the average thermal velocity of molecules of size m and temperature θ. A slower time scale is given by the time between two successive collisions $t_\mathrm{mfp} \propto l_\mathrm{mfp}/u_\theta$, which is also the time scale at which the system relaxes to local equilibrium. Finally, in the macroscopic description the shorter of the two time scales $t_\mathrm{adv} \propto l/u$ (advective dynamics) and $t_\mathrm{diff} \propto l^2/\nu$ (diffusive dynamics) plays the more important role. Here l and u are a typical length and velocity of the considered flow problem. ν is the kinematic viscosity which is related to the dynamic shear viscosity by $\eta = \rho\nu$, where ρ is the density of the fluid. The ratio of diffusive to advective dynamics is denoted in the *Reynolds number*

$$\mathrm{Re} = \frac{t_\mathrm{diff}}{t_\mathrm{adv}} = \frac{ul}{\nu}. \tag{3.1}$$

An other useful dimensionless number that can capture the relative importance of scales of the considered problem is the *Knudsen number*

$$\mathrm{Kn} = \frac{l_\mathrm{mfp}}{l}. \tag{3.2}$$

It is obvious now, that the microscopic description acts on time and length scales that are not able to capture the problem size of the problems mentioned in chapter 1. The Navier-Stokes equation (NSE) and its related equations are a macroscopic description of Newtonian fluids and will be described in the next section. Kinetic theory, on the other hand, is a mesoscopic fluid description on which the LBM is based. Its main equations and its relation to the NSE will be given in chapter 3.2.

3.1 Continuum theory

The equations describing an ideal fluid can be found when considering "fluid elements", which are small compared to system size, but large compared to an individual molecule as needed for the continuum description. By demanding, that the mass of an arbitrary fluid element can only change by mass flux across the elements surface, the *continuity equation*

$$\frac{\partial\rho}{\partial t} + \nabla\cdot(\rho\boldsymbol{u}) = 0 \tag{3.3}$$

is found. Similarly, the change of momentum of a fluid element can be considered via the momentum density $\boldsymbol{j} = \rho\boldsymbol{u}$. This change can only be due to (i) flow of momentum into or out of its surface, (ii) differences in pressure p and (iii) external body forces $\boldsymbol{F}_{\text{ext}}$. This directly corresponds to the last three terms of the *Euler equation*

$$\frac{\partial(\rho\boldsymbol{u})}{\partial t} + \nabla\cdot(\rho\boldsymbol{u}\boldsymbol{u}) = -\nabla p + \boldsymbol{F}_{\text{ext}}. \tag{3.4}$$

Here, $\boldsymbol{u}\boldsymbol{u}$ denotes the outer product. This is a special case of the *Cauchy momentum equation*

$$\frac{\partial(\rho\boldsymbol{u})}{\partial t} + \nabla\cdot\Pi = \boldsymbol{F}_{\text{ext}}, \tag{3.5}$$

where $\Pi_{\alpha\beta} = \rho u_\alpha u_\beta - \sigma_{\alpha\beta}$ denotes the *momentum flux density tensor*. The term $\sigma_{\alpha\beta}$ is called *stress tensor*. The Euler equation yields an isotropic stress $\sigma_{\alpha\beta} = -p\delta_{\alpha\beta}$, where $\delta_{\alpha\beta}$ denotes the Kronecker delta.

In real fluids, internal friction in form of viscosity is present. This effect is captured by an additional term of the stress tensor, the *viscous stress tensor* $\underline{\sigma}_s$. The viscous stress tensor can be separated into a traceless shear stress and a normal stress:

$$\sigma_{s,\alpha\beta} = \eta_s\left(\frac{\partial u_\alpha}{\partial x_\beta} + \frac{\partial u_\beta}{\partial x_\alpha} - \frac{2}{3}\delta_{\alpha\beta}\frac{\partial u_\gamma}{\partial x_\gamma}\right) + \eta_B\delta_{\alpha\beta}\frac{\partial u_\gamma}{\partial x_\gamma}. \tag{3.6}$$

The coefficients η_s and η_B are shear viscosity and bulk viscosity respectively. They are usually assumed to be isotropic and uniform, but don't have to be in general. Incorporating the viscous stress tensor into eq. (3.5) leads to the *Navier-Stokes equation*

$$\frac{\partial(\rho\boldsymbol{u})}{\partial t} + \nabla\cdot(\rho\boldsymbol{u}\boldsymbol{u}) = -\nabla p + \nabla\cdot\underline{\sigma}_s + \boldsymbol{F}_{\text{ext}}. \tag{3.7}$$

The NSE can be simplified by assuming an incompressible fluid with $\rho = \text{const}$. The continuity equation (3.3) then reduces to

$$\nabla\cdot\boldsymbol{u} = 0. \tag{3.8}$$

The NSE (3.7) simplifies to the *incompressible NSE*

$$\rho\left(\frac{\partial\boldsymbol{u}}{\partial t} + \boldsymbol{u}\cdot\nabla\boldsymbol{u}\right) = -\nabla p + \eta_s\Delta\boldsymbol{u} + \boldsymbol{F}_{\text{ext}}, \tag{3.9}$$

where $\Delta = \nabla\cdot\nabla$ denotes the Laplace operator and obviously $\eta_s\Delta\boldsymbol{u} = \nabla\cdot\underline{\sigma}_s$ holds. A more general form of the incompressible NSE for future reference is

$$\rho\left(\frac{\partial\boldsymbol{u}}{\partial t} + \boldsymbol{u}\cdot\nabla\boldsymbol{u}\right) = -\nabla p + \nabla\cdot\underline{\sigma} + \boldsymbol{F}_{\text{ext}}, \tag{3.10}$$

where $\underline{\sigma}$ denotes an arbitrary stress tensor and $\underline{\sigma} = \underline{\sigma}_s$ leads back to eq. (3.9). In the case of an incompressible fluid, eqs. (3.8) and (3.9) form a closed system of equations, i.e. there are four equations for the four unknowns u_x, u_y, u_z and p. For a general fluid ρ is an additional unknown, therefore one has to introduce an equation of state and additional assumptions. One of the most simple ways to do so would be to assume constant temperature $\theta = \theta_0 = \text{const}$ and to take the *ideal gas law*

$$p = \rho R_{\text{sp}}\theta, \tag{3.11}$$

where R_{sp} is the specific gas constant. This isothermal equation of state results in a linear relationship between pressure and density. The *speed of sound* c_s is i.g. given by the relation

$$c_s^2 = \left(\frac{\partial p}{\partial \rho}\right)_s, \tag{3.12}$$

taking the derivative at constant entropy s. In this isothermal case it results in $c_s = \sqrt{R_{\mathrm{sp}}\theta_0}$.

3.2 Kinetic theory

Kinetic theory considers distributions of particles in a gas, a quantity evolving on timescales proportional to the mean collision time t_{mfp}. While in principle this description is more general than the NSE, here only the most common case of a dilute monoatomic gas is considered. The fundamental variable in kinetic theory is the particle distribution function $f(\boldsymbol{x}, \boldsymbol{\xi}, t)$, which represents the density of particles with velocity $\boldsymbol{\xi}$ at position $\boldsymbol{x}$ and time t. It therefore has the units $\mathrm{kgs}^3/\mathrm{m}^6$. The distribution function f is connected to macroscopic variables by its moments. Here a general notation for the moments of a function g is used:

$$\mathcal{M}_0(g) = \int g \,\mathrm{d}^3\xi, \qquad \mathcal{M}_\alpha(g) = \int \xi_\alpha g \,\mathrm{d}^3\xi, \qquad \mathcal{M}_{\alpha\beta}(g) = \int \xi_\alpha \xi_\beta g \,\mathrm{d}^3\xi. \tag{3.13}$$

The quantities *mass density*, *momentum density* and *total energy density* are found as the first three moments of f, respectively:

$$\rho(\boldsymbol{x}, t) = \mathcal{M}_0(f), \qquad \rho(\boldsymbol{x}, t) u_\alpha(\boldsymbol{x}, t) = \mathcal{M}_\alpha(f), \qquad \rho(\boldsymbol{x}, t) E(\boldsymbol{x}, t) = \mathcal{M}_{\alpha\alpha}(f). \tag{3.14}$$

A gas left alone sufficiently long will even out the distribution of particle velocities around the mean velocity u and finally reach an equilibrium distribution, namely the *Maxwell-Boltzmann distribution* for an ideal monoatomic gas

$$f^{\mathrm{eq}}(\boldsymbol{x}, |\boldsymbol{v}|, t) = \rho \left(\frac{1}{2\pi R_{\mathrm{sp}}\theta}\right)^{3/2} e^{-|\boldsymbol{v}|^2/(2R_{\mathrm{sp}}\theta)}. \tag{3.15}$$

For the above equation, the definition $\boldsymbol{v}(\boldsymbol{x}, t) = \boldsymbol{\xi}(\boldsymbol{x}, t) - \boldsymbol{u}(\boldsymbol{x}, t)$ is introduced. The time evolution of f can be found by looking at its total derivative with respect to time:

$$\frac{\mathrm{d}f}{\mathrm{d}t} = \left(\frac{\partial f}{\partial t}\right)\frac{\mathrm{d}t}{\mathrm{d}t} + \left(\frac{\partial f}{\partial x_\beta}\right)\frac{\mathrm{d}x_\beta}{\mathrm{d}t} + \left(\frac{\partial f}{\partial \xi_\beta}\right)\frac{\mathrm{d}\xi_\beta}{\mathrm{d}t}. \tag{3.16}$$

Applying Newton's second law ($\rho\frac{\mathrm{d}\xi_\beta}{\mathrm{d}t} = F_\beta$ with $[F_\beta] = \mathrm{N\,m^{-3}}$) and using the notation $\mathrm{d}f/\mathrm{d}t = \Omega(f)$, the *Boltzmann equation* is obtained:

$$\frac{\partial f}{\partial t} + \xi_\beta \frac{\partial f}{\partial x_\beta} + \frac{F_\beta}{\rho}\frac{\partial f}{\partial \xi_\beta} = \Omega(f). \tag{3.17}$$

The source term on the right hand side represents the local redistribution of f due to collisions and is therefore called *collision operator*. A useful collision operator for monoatomic gases has to conserve the quantities mass, momentum and translational energy. This can be formulated as constraints towards the first three moments of the collision operator:

$$\mathcal{M}_0(\Omega(f)) = 0, \qquad \mathcal{M}_\alpha(\Omega(f)) = 0, \qquad \mathcal{M}_{\alpha\alpha}(\Omega(f)) = 0. \tag{3.18}$$

One possible collision operator that is also used in the LBM is the very simple Bhatnagar-Gross-Krook (BGK) collision operator

$$\Omega(f) = -\frac{1}{\tau_\mathrm{r}}(f - f^\mathrm{eq}), \tag{3.19}$$

where τ_r is the relaxation time.

The macroscopic equations of fluid mechanics can be found directly from the Boltzmann eq. (3.17). This is done by looking at its first three moments (i.e. again multiplying it with functions of $\boldsymbol{\xi}$ and integrating over velocity space) and using the equalities from eq. (3.14) and (3.18). The first moment of the Boltzmann equation reproduces the continuity equation (3.3). The second moment reproduces the Cauchy momentum equation (3.5) with the resulting stress tensor

$$\sigma_{\alpha\beta} = -\int v_\alpha v_\beta f \,\mathrm{d}^3\xi. \tag{3.20}$$

This equation is not closed, since f is unknown. Finally, the *total energy equation*

$$\frac{\partial(\rho E)}{\partial t} + \frac{\partial(\rho u_\beta E)}{\partial x_\beta} = \frac{\partial(u_\alpha \sigma_{\alpha\beta})}{\partial x_\beta} + F_\beta u_\beta - \frac{\partial q_\beta}{\partial x_\beta} \tag{3.21}$$

is found from the trace of the third moment. Here the *heat flux* is defined as

$$q_\beta = \frac{1}{2}\int v_\alpha v_\alpha v_\beta f \,\mathrm{d}^3\xi, \tag{3.22}$$

so again this equation is not closed. The pendant in continuum theory would be the Navier-Stokes-Fourier equations, when taking the total energy equation (3.21) with the heat flux

$$\boldsymbol{q} = -\kappa\nabla\theta, \tag{3.23}$$

(κ being the fluid's thermal diffusivity) together with the continuity equation and the NSE. In general, to approximate eq. (3.20) and (3.22), an explicit approximation of the distribution function f has to be found.

The simplest possible approximation is $f \approx f^\mathrm{eq}$, which leads the second moment of the Boltzmann equation back to the Euler momentum equation, and the trace of the third moment to the *Euler energy equation*, i.e. a simplified version of eq. (3.21). If one also want to consider non-equilibrium, i.e. $f - f^\mathrm{eq} \neq 0$, an analysis based on the perturbation expansion can be performed, which is called *Chapman-Enskog analysis* and is an established method of connecting the kinetic and continuum pictures. Restricting the expansion to first order, i.e. $f \approx f^\mathrm{eq} + \epsilon f^{(1)}$ with smallness parameter ϵ, and explicitly finding $f^{(1)}$ from the macroscopic derivatives of the equilibrium distributions f^eq, the full Navier-Stokes-Fourier model with its viscous stress and heat conduction is recovered. The resulting transport coefficients are:

$$\eta_s = p\tau_\mathrm{r}, \qquad \eta_B = 0, \qquad \kappa = \frac{5}{2}R_\mathrm{sp}p\tau_\mathrm{r}. \tag{3.24}$$

The shown expansion is valid for small Knudsen number Kn. For high Kn, a higher order expansion has to be considered.

4 Polymer models

In this chapter, two polymer models, namely the Oldroyd-B and the FENE-P model, will be described together with their basic properties. For some flow geometries, analytical solutions of polymer stress or fluid velocity will be given. This serves as an preparation for the validation of the implementation of these models in chapter 10.4 and 10.6. In chapter 10.1 it will be made clear, why these models are relevant for bioprinting.

4.1 Oldroyd-B model

4.1.1 From the microscopic picture to a constitutive equation

The Oldroyd-B model can be derived from microscopical principles [1]. This connection using the *elastic dumbbell model* was first proposed by [14], but here the description of [13, ch. 2.2] is followed (it is also found in [15, ch. 7.5]). In this model the complex chains of polymers are simply described by elastic dumbbells (cf. fig. 4.1) diluted in incompressible Newtonian fluid with viscosity η_s. Each dumbbell consists of two beads of mass m connected by a Hookean spring (zero mass). The vectors $\boldsymbol{r}_1(t)$ and $\boldsymbol{r}_2(t)$ being the positions of the beads at time t and $\boldsymbol{q} = \boldsymbol{r}_1 - \boldsymbol{r}_2$ denoting the elongation vector, the equation of motion for both beads is described by Newton's fist law

$$m\frac{\mathrm{d}^2\boldsymbol{x}_i}{\mathrm{d}t^2} = \boldsymbol{\mathcal{F}}_i^s - \boldsymbol{\mathcal{F}}_i^d + \boldsymbol{B}_i, \;\; i \in \{1,2\}, \tag{4.1}$$

where on the right-hand side three forces play a role. $\boldsymbol{\mathcal{F}}_i^s$ is the force of the spring acting on the i-th bead and is given by the Hookean law

$$\boldsymbol{\mathcal{F}}_1^s = -\boldsymbol{\mathcal{F}}_2^s = K\boldsymbol{q}, \tag{4.2}$$

where K is the spring constant. $\boldsymbol{\mathcal{F}}_i^d$ is the drag force acting on the i-th bead. Since the beads are modeled as spheres of radius a, it can be described by Stokes law

$$\boldsymbol{\mathcal{F}}_i^d = \zeta\left(\frac{\mathrm{d}\boldsymbol{r}_i}{\mathrm{d}t} - \boldsymbol{u}(\boldsymbol{r}_i)\right), \tag{4.3}$$

where $\boldsymbol{u}(\boldsymbol{r}_i)$ is the fluid velocity and $\zeta = 6\pi\eta_s a$. $\boldsymbol{B}_i$ is the Brownian force caused by molecules of the fluid acting on the i-th bead and can be written in the form

$$\boldsymbol{B}_i\mathrm{d}t = \sqrt{2k_B\theta\zeta}\mathrm{d}\boldsymbol{W}_i, \tag{4.4}$$

where $\boldsymbol{W}_i$ is the three-dimensional Wiener process, k_B is the Boltzmann constant and θ the temperature.

[1] An alternative derivation of the Oldroyd-B model uses the theory of simple fluids with fading memory [13, ch. 2.2].

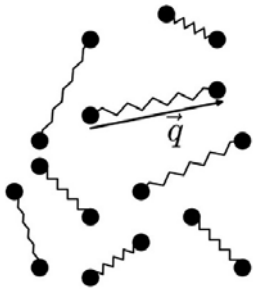


Fig. 4.1: Illustration of dumbbells dissolved in a Newtonian fluid. Image taken from [16].

The actual derivation of the Oldroyd-B model will be skipped, instead only some additional assumptions are outlined. First of all, the left hand side of eq. (4.1) can be assumed to be zero, since m is generally very small and inertia effects are much smaller than viscous and elastic effects. Secondly, thermodynamic equilibrium is assumed to make use of the equipartition theorem and find $\frac{1}{2}K\left\langle|\boldsymbol{q}|^2\right\rangle_\psi = \frac{3}{2}k_\mathrm{B}\theta$, where $\langle\cdot\rangle_\psi$ denotes the expectation value as defined in eq. (4.6). Furthermore, beads are assumed to not interact with each other and can be treated as an ideal gas. As a last step, the view is changed to the macroscopic picture, where the quantity of interest is the conformation tensor $\underline{c}$ defined as

$$\underline{c} = \frac{K}{k_\mathrm{B}\theta}\left\langle \boldsymbol{q}\boldsymbol{q}\right\rangle_\psi, \tag{4.5}$$

where

$$\left\langle x(t)\right\rangle_\psi = \int_{\mathbb{R}} x(\boldsymbol{q},t)\psi(\boldsymbol{q},t)\,\mathrm{d}\boldsymbol{q} \tag{4.6}$$

is the expectation value using the dumbbell probability density $\psi(\boldsymbol{q},t)$. The conformation tensor is positive definite. Putting this together, a joint Fokker-Planck equation for $\psi(\boldsymbol{q},t)$ can be found. By integrating it over $\boldsymbol{q}$, the Oldroyd-B model is obtained:

$$\lambda_p \overset{\nabla}{\underline{c}} + \underline{c} = \underline{\mathrm{I}}, \tag{4.7a}$$
$$\underline{\sigma} = \underline{\sigma}_s + \underline{\tau}_p, \tag{4.7b}$$
$$\underline{\tau}_p = G(\underline{c} - \underline{\mathrm{I}}), \tag{4.7c}$$

where $\underline{\mathrm{I}}$ denotes the identity matrix and the total stress tensor $\underline{\sigma}$ from eq. (4.7b) additionally has to solve the incompressible NSE (3.10). The *upper convected derivative* is defined as

$$\overset{\nabla}{\underline{c}} = \frac{\mathrm{d}\underline{c}}{\mathrm{d}t} - ((\nabla\boldsymbol{u})^\mathrm{T}\cdot\underline{c} + \underline{c}\cdot(\nabla\boldsymbol{u})). \tag{4.8}$$

Note that in the equation above the *total (material) derivative*

$$\frac{\mathrm{d}\underline{c}}{\mathrm{d}t} = \frac{\partial\underline{c}}{\partial t} + \boldsymbol{u}\cdot\nabla\underline{c} \tag{4.9}$$

is being used. The connection to the microscopic model is given by the relaxation time $\lambda_p = \frac{\zeta}{4K}$ and the elastic modulus $G = nk_\mathrm{B}\theta$ with n being the number of dumbbells per unit volume.

4.1.2 Alternative formulations

While the *conformation tensor formulation* via eq. (4.7) has the advantage of being strongly connected to the microscopic model, in literature numerous other formulations can be found. Here the connection to two other often used formulations is presented following the overview provided by [17] [2]. For both, only the total stress tensor $\underline{\sigma}$ is given. To obtain the full model, the total stress tensor again has to enter the incompressible NSE (3.10).

In polymer science it is customary to write the stress tensor as the sum of a viscoelastic component (originating from the polymers) and a Newtonian component (the solvent contribution). Doing so, in the *polymer formulation* the constitutive equation is changed to:

$$\underline{\sigma} = \underline{\tau}_p + 2\eta_s \underline{D}, \tag{4.10}$$

$$\underline{\tau}_p = -\lambda_p \overset{\nabla}{\underline{\tau}}_p + 2\eta_p \underline{D}, \tag{4.11}$$

where λ_p and η_p are polymer relaxation time and viscosity respectively. The deformation rate tensor $\underline{D}$ is given by

$$D_{ij} = \frac{1}{2}(\partial_i u_j + \partial_j u_i), \tag{4.12}$$

and it holds $\underline{\sigma}_s = 2\eta_s \underline{D}$. The justification for calling η_p a viscosity can be found when investigating steady shear flow (see chapter 4.3.2.1). The *polymer formulation* presented here will be the one used for implementation purposes in chapter 10.3. It is related to the conformation tensor formulation via $G = \eta_p/\lambda_p$, as shown in eq. (A2).

In the *original formulation* of the Oldroyd-B model as first proposed in [18] the constitutive equation for $\underline{\sigma}$ obeys

$$\underline{\sigma} + \lambda_p \overset{\nabla}{\underline{\sigma}} = 2\eta(\underline{D} + \lambda_\mathrm{r} \overset{\nabla}{\underline{D}}). \tag{4.13}$$

Here, $\eta = \eta_s + \eta_p$ is the total viscosity and λ_r is named *retardation time*. This original formulation is equivalent to the polymer one, if one takes (proof see eq. (A1))

$$\eta_p = (1 - \hat{\beta})\eta, \qquad \eta_s = \hat{\beta}\eta, \tag{4.14}$$

with $\hat{\beta} = \lambda_\mathrm{r}/\lambda_p = \eta_s/\eta$. The Newtonian fraction of total viscosity $\hat{\beta}$ is not to be confused with the polymer fraction $\beta = \eta_s/\eta = 1 - \hat{\beta}$ used in later chapters.

For $\eta_s = 0$ (or equally $\lambda_\mathrm{r} = 0$) the Oldroyd-B model is reduced to the *upper convected Maxwell* (UCM) model (see chapter 7.5.5 in [15]). It can be shown that the UCM model is solved by (see [15], chapter 7.6)

$$\underline{c}(t) = \frac{G}{\lambda_p} \int_{-\infty}^{t} \exp^{(s-t)/\lambda_p} \underline{C}_t(s)^{-1} \mathrm{d}s, \tag{4.15}$$

with $\underline{C}_t$ being the right relative Cauchy–Green tensor. This integral version of the UCM model is called the *Lodge rubber-like liquid* model, a model with fading memory meant for concentrated polymer solutions and melts.

[2] [17] collects even two more Oldroyd-B formulations not mentioned here.

4.2 FENE-P model

4.2.1 Microscopic model and stress constitutive formulation

The FENE-P model has it origins in the finitely extensible nonlinear elastic (FENE) dumbbell model. This is a dumbbell model analogous to chapter 4.1.1 and obeys a similar equation of motion as in eq. (4.1). Again the drag force and the effect of Brownian dynamics act on the beads, but the spring now obeys the following force law [19, eq. 1]:

$$\boldsymbol{\mathcal{F}}_1^s = -\boldsymbol{\mathcal{F}}_2^s = \frac{K\boldsymbol{q}}{1-(q^2/q_0^2)}, \tag{4.16}$$

where $\boldsymbol{q}$ is again the vector connecting the beads, and K and q_0 are constants. The parameter K again has the meaning of a spring constant and q_0 is the maximum extensibility of the dumbbells [20, eq. 10]. However, with this force law it is not possible to solve for the configurational distribution function $\psi(\boldsymbol{q},t)$ in the diffusion equation (Fokker-Planck equation) of dumbbell kinetic theory [19, eq. 2]. Therefore the force law of the microscopic model is changed to [19, eq. 6]:

$$\boldsymbol{\mathcal{F}}_1^s = -\boldsymbol{\mathcal{F}}_2^s = \frac{K\boldsymbol{q}}{1-\langle q^2/q_0^2\rangle_\psi}, \tag{4.17}$$

with $\langle\cdot\rangle_\psi$ from eq. (4.6). This idea is based on Peterlin and therefore the model is called FENE-P. By inserting $\boldsymbol{\mathcal{F}}_1^s$ into the Kramers form, taking the trace and eliminating the appearing term $\langle\boldsymbol{qq}\rangle_\psi$ with the help of the Giesekus form, the original formulation of the constitutive equation for the stress tensor of the FENE-P model was obtained [19, eq. 4-10] [21, eq. 13-14] for $\epsilon = 0$:

$$Z\underline{\tau}_p + \lambda_p\overset{\nabla}{\underline{\tau}}_p - \lambda_p(\underline{\tau}_p + (1-\epsilon b)nk_\mathrm{B}\theta\underline{\mathrm{I}})\frac{\mathrm{d}\ln Z}{\mathrm{d}t} = 2(1-\epsilon b)nk_\mathrm{B}\theta\lambda_p\underline{D} \tag{4.18}$$

where ∇ and $\frac{\mathrm{d}}{\mathrm{d}t}$ are the upper convected and the total derivative, respectively. $\underline{D}$ is the deformation rate tensor and it holds

$$Z = Z(\underline{\tau}_p) = 1 + \frac{3}{b}\left((1-\epsilon b) + \frac{\mathrm{Tr}(\underline{\tau}_p)}{3nk_\mathrm{B}\theta}\right). \tag{4.19}$$

Here Tr is the trace, and b is the *finite extensibility parameter* (typically values range from 10 to 1000). Because Z depends on $\mathrm{Tr}(\underline{\tau}_p)$, the constitutive equation is nonlinear in stress. When b goes to infinity, Z becomes unity and the Oldroyd-B limit is obtained, because eq. (4.11) is recovered. Later, it has been shown in [22, eq. 13.5-49 - 13.5-56] that for $\epsilon = 2/(b(b+2))$ a better approximation for eq. (4.16) can be obtained, which is actually exact in equilibrium. Note that my definition of the stress tensor deviates from the one in the original formulation [19, 22] by a factor -1. Equation (4.18) is conservative ([21, eq. 15]. The model can describe the effect of shear-thinning, as will be shown in chapter 4.3.3.1. Furthermore, next to $\lambda_p = \zeta/(4K)$, a second time constant $\lambda_Q = \zeta q_0^2/(12k_\mathrm{B}\theta)$ for rigid dumbbells can be defined and it holds (see [20, p. 20]):

$$b = 3\lambda_Q/\lambda_p = Kq_0^2/(k_\mathrm{B}\theta). \tag{4.20}$$

Some more connections to the microscopic model for the case $\epsilon = 0$ are give by:

$$\eta_p/\lambda_p = nk_\mathrm{B}\theta, \tag{4.21}$$

$$b + 3 = 3\frac{q_0^2}{q_\mathrm{eq}^2}, \tag{4.22}$$

where q_eq is the average dumbbell length in equilibrium.

4.2.2 Alternative formulation

An alternative formulation [21, eq. 20] uses the parameter L, which is the maximum extensibility of a dumbbell scaled with its equilibrium value, $L^2 = 3q_0^2/q_{\text{eq}}^2$. The relation to the former model is obtained via

$$\hat{b} = L^2 - 3 \tag{4.23}$$

where

$$\hat{b} = \begin{cases} b, & \text{if } \epsilon = 0, \\ b+2, & \text{if } \epsilon = 2/(b(b+2)). \end{cases} \tag{4.24}$$

A similar formulation is given in [23], who implement the FENE-P model for the case $\epsilon = 0$ using the LBM. The paper contains many good validation setups, some of which are also used in this thesis in chapter 10.6. However, the model formulation seems to contain an error, that appears in several parts of the paper and will be briefly discussed here. Their constitutive equation is formulated via the conformation tensor $\underline{c}$ and the "(dimensionless) maximum possible extension" L. However, they don't give the concrete relations $L \leftrightarrow b$ und $\underline{c} \leftrightarrow \underline{\tau}_p$. Looking at the analytic solutions they give in their chapters 3.1 (steady shear flow), 3.2 (steady elongational flow) and 3.4 (small amplitude oscillatory shearing) using a so called *polymer feedback stress tensor* $\underline{\sigma}_P$ and comparing them to ours from chapter 4.3.2.1, 4.3.2.2 and 4.3.2.3 one finds that

$$\underline{\sigma}_P = \frac{\eta_p}{\lambda_p}\underline{\tau}_p + \underline{\mathrm{I}}, \qquad\qquad L^2 - 3 = b \tag{4.25}$$

must hold. However, inconsistencies then arise with their equations (22)-(24), which are supposed to be a direct mapping into the original formulation of Bird et al. According to their mapping

$$Z = \frac{L^2 - 3 + \mathrm{Tr}(\underline{\sigma}_P)}{L^2}, \tag{4.26}$$

holds, while in the FENE-P model formulation of this thesis

$$Z = 1 + \frac{3}{b}\left(1 + \frac{\mathrm{Tr}(\underline{\tau}_p)}{3}\frac{\lambda_p}{\eta_p}\right) \tag{4.27}$$

holds. Both cannot be transformed into each other via eq. (4.25). Moreover, it is noticeable that in eq. (4.27) $Z = 1$ holds for $\mathrm{Tr}(\underline{\tau}_p) = -3\frac{\eta_p}{\lambda_p}$, which would correspond to $\mathrm{Tr}(\underline{\sigma}_P) = 0$ when taking the definition from eq. (4.25). But contradictory to this, $Z = 1$ in eq. (4.26) is only satisfied for $\mathrm{Tr}(\underline{\sigma}_P) = 3$. However, if instead of eq. (4.26) a slightly modified Z is used, i.e.

$$Z = \frac{L^2 - 3 + \mathrm{Tr}(\underline{\sigma}_P)}{L^2 - 3}, \tag{4.28}$$

both discrepancies can be solved. This would be a direct error in their algorithm[3] and would have to be noticeable especially for small L^2.

[3] To be exact, their FENE-P potential f from page 179 would be erroneous and would have to be adjusted to match with eq. (4.28).

4.3 Behavior in simple flows

4.3.1 Flow geometries and definition of basic quantities

The following simple flows will be used to analyse the properties of the Oldroyd-B and FENE-P model:

- Homogeneous shear flow:

$$\boldsymbol{u} = (\dot{\gamma} y, 0, 0)^{\mathrm{T}} \text{ with } \dot{\gamma} = \text{const.} \tag{4.29}$$

- Oscillatory shear flow:

$$\boldsymbol{u} = (\dot{\gamma} y, 0, 0)^{\mathrm{T}} \text{ with } \dot{\gamma} = \dot{\gamma}_0 \Re(e^{i\omega t}). \tag{4.30}$$

- Homogeneous uniaxial elongational flow:

$$\nabla \boldsymbol{u} = \mathrm{diag}(\dot{\epsilon},\ -\dot{\epsilon}/2,\ -\dot{\epsilon}/2) \text{ with } \dot{\epsilon} = \text{const.} \tag{4.31}$$

Here, $\dot{\gamma} = \frac{\partial u_x}{\partial y}$ denotes the shear rate and ω is an angular frequency. The parameter $\dot{\epsilon}$ denotes the elongation rate and is not related with the constant ϵ of the FENE-P model.

Using these flow geometries, the basic fluid properties are derived by solving for the stress tensor $\underline{\sigma}$. Firstly, the shear viscosity η and the first and second normal stress differences, N_1 and N_2 respectively, are defined via the stress tensor $\underline{\sigma}$ observed in homogeneous shear flow at steady state:

$$\eta = \sigma_{12}/\dot{\gamma}, \qquad N_1 = \sigma_{11} - \sigma_{22}, \qquad N_2 = \sigma_{22} - \sigma_{33}. \tag{4.32}$$

Secondly, the elastic moduli G' and G'' (also called storage and loss modulus respectively) result from a fluid subject to oscillatory shear flow with small amplitude and read

$$\eta^* = \eta' - i\eta'' = \sigma_{12}/\dot{\gamma}_0, \qquad G' = \omega\eta'', \qquad G' = \omega\eta', \tag{4.33}$$

where η^* is the complex total viscosity. Finally, the elongational viscosity η_E is obtained in homogeneous uniaxial elongational flow at steady state as

$$\eta_E = \frac{\sigma_{11} - \sigma_{22}}{\dot{\epsilon}}. \tag{4.34}$$

4.3.2 Oldroyd-B behavior in simple flows

Here the description of [23], chapter 3, and [15], chapter 7.6.1, is followed.

4.3.2.1 Steady shear flow

Considering Eq. (4.11) in 2D under the effect of a homogeneous shear flow, the equations written out in components become

$$\begin{pmatrix} \tau_{11} & \tau_{12} \\ \tau_{21} & \tau_{22} \end{pmatrix} + \lambda_p \begin{pmatrix} \dot{\tau}_{11} & \dot{\tau}_{12} \\ \dot{\tau}_{21} & \dot{\tau}_{22} \end{pmatrix} - \lambda_p \dot{\gamma} \begin{pmatrix} 2\tau_{12} & \tau_{22} \\ \tau_{22} & 0 \end{pmatrix} - \eta_p \begin{pmatrix} 0 & \dot{\gamma} \\ \dot{\gamma} & 0 \end{pmatrix} = 0. \tag{4.35}$$

If the stress components start from zero initial states, then $\tau_{22} = 0$ is valid for all time (as do all additional components for the case of 3D). Since $\underline{\tau}_p$ is symmetric, only two non-trivial components remain:

$$\begin{aligned} \tau_{11} + \lambda_p(\dot{\tau}_{11} - 2\dot{\gamma}\tau_{12}) &= 0 \\ \tau_{12} + \lambda_p(\dot{\tau}_{12} - \dot{\gamma}\tau_{22}) &= \eta_p\dot{\gamma} \end{aligned} \tag{4.36}$$

At steady state (i.e. $\dot{\tau}_{11} = 0 = \dot{\tau}_{12}$), $\tau_{12} = \eta_p\dot{\gamma}$ and $\tau_{11} = 2\eta_p\lambda_p\dot{\gamma}^2$ is found. Therefore the viscometric functions are

$$\eta = \sigma_{12}/\dot{\gamma} = \eta_p + \eta_s, \qquad N_1 = \sigma_{11} - \sigma_{22} = 2\eta_p\lambda_p\dot{\gamma}^2, \qquad N_2 = \sigma_{22} - \sigma_{33} = 0. \tag{4.37}$$

In this linear elastic dumbbell model the polymer shear stress τ_{12} is directly proportional to the shear strain $\dot{\gamma}$. This is only realistic for small shear strains. Furthermore, dilute polymer solutions usually show some degree of shear-thinning. To observe this, one would have to develop a more realistic force law for the chain, as it is the case for the FENE-P model. Looking at the shear and normal stress of the Oldroyd-B model motivates a new dimensionless number used in the context of viscoelastic flows. The Weissenberg number $\mathrm{Wi} = \dot{\gamma}\lambda_p$ relates the elastic forces to the viscous forces present in the polymers:

$$\mathrm{Wi} = \frac{\text{elastic forces}}{\text{viscous forces}} = \frac{\tau_{11} - \tau_{22}}{2\tau_{12}} = \frac{2\eta_p\lambda_p\dot{\gamma}^2}{2\eta_p\dot{\gamma}} = \dot{\gamma}\lambda_p. \tag{4.38}$$

4.3.2.2 Small amplitude oscillatory shearing

In an oscillatory flow, looking at eq. (4.36) the Ansatz

$$\tau_{12} = \Re(\alpha_1 e^{i\omega t}), \qquad \tau_{11} = \Re(\alpha_2 e^{2i\omega t}) \tag{4.39}$$

is an obvious choice. Substituting this into eq. (4.36) yields

$$\alpha_1 = \frac{\eta_p\dot{\gamma}_0}{1 + i\lambda_p\omega}, \qquad \alpha_2 = \frac{2\eta_p\lambda_p\dot{\gamma}_0^2 i\omega}{(1 + i\lambda_p\omega)(1 + 2i\lambda_p\omega)}. \tag{4.40}$$

Consequently, the dynamic properties of Oldroyd-B are

$$\eta^* = \eta_s + \frac{\eta_p}{1 + i\lambda_p\omega} \tag{4.41}$$

$$G' = \omega\eta'' = \frac{G\lambda_p^2\omega^2}{1 + \lambda_p^2\omega^2}, \qquad G'' = \omega\eta' = \eta_s\omega + \frac{G\lambda_p\omega}{1 + \lambda_p^2\omega^2}. \tag{4.42}$$

The result for the storage and loss moduli can be brought to a nondimensional form only dependent on the Deborah number $\mathrm{De} = \omega\lambda_p$ by describing G'' without the viscosity component η_s and dividing by G (see [24], p. 140), which results in

$$G' = \frac{\mathrm{De}^2}{1 + \mathrm{De}^2}, \qquad G'' = \frac{\mathrm{De}}{1 + \mathrm{De}^2} \tag{4.43}$$

4.3.2.3 Elongational flow

In an uniaxial elongational flow eq. (4.11) becomes

$$\underline{\tau}_p + \lambda_p \underline{\dot{\tau}}_p - \lambda_p \frac{\dot{\epsilon}}{2} \begin{pmatrix} 4\tau_{11} & \tau_{12} & \tau_{13} \\ \tau_{21} & -2\tau_{22} & -2\tau_{23} \\ \tau_{31} & -2\tau_{32} & -2\tau_{33} \end{pmatrix} = \eta_p \dot{\epsilon} \begin{pmatrix} 2 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{pmatrix} \tag{4.44}$$

Assuming again zero initial states ($\underline{\tau}_p = \underline{0}$) the only three terms that not remain zero are (note that $\tau_{22} = \tau_{33}$)

$$\tau_{11} + \lambda_p(\dot{\tau}_{11} - 2\dot{\epsilon}\tau_{11}) = 2\eta_p\dot{\epsilon} \tag{4.45}$$

$$\tau_{22} + \lambda_p(\dot{\tau}_{22} + \dot{\epsilon}\tau_{22}) = -\eta_p\dot{\epsilon}. \tag{4.46}$$

For $\dot{\epsilon} = \text{const}$ the solution is

$$\tau_{11} = \frac{2\eta_p\dot{\epsilon}}{1 - 2\lambda_p\dot{\epsilon}}(1 - e^{-(1-2\lambda_p\dot{\epsilon})t/\lambda_p}), \tag{4.47}$$

$$\tau_{22} = \tau_{33} = -\frac{\eta_p\dot{\epsilon}}{1 + \lambda_p\dot{\epsilon}}(1 - e^{-(1+\lambda_p\dot{\epsilon})t/\lambda_p}). \tag{4.48}$$

At steady state ($t \to \infty$) for $-1 < \lambda_p\dot{\epsilon} < 1/2$ the extensional viscosity becomes

$$\eta_E - 3\eta_s = \frac{\tau_{11} - \tau_{22}}{\dot{\epsilon}} = \frac{3\eta_p}{(1 - 2\lambda_p\dot{\epsilon})(1 + \lambda_p\dot{\epsilon})} \tag{4.49}$$

The Trouton ratio η_E/η is bigger than the Newtonian value of 3 for $\dot{\epsilon} \neq 0$ and becomes unbounded when $\lambda_p\dot{\epsilon}$ approaches -1 or $1/2$. Therefore, if either $\lambda_p\dot{\epsilon} \geq 1/2$ or $\lambda_p\dot{\epsilon} \leq -1$ at least one component of the stress grows unboundedly. This is due to the dumbbell model allowing the linear spring to grow without bound in a strong flow. Constraining the dumbbell to a maximum allowable length would fix this (e.g. FENE dumbbell, Phan-Thien/Tanner model).

4.3.3 FENE-P behavior in simple flows

The derivation of the rheological quantities is taken from [19, ch. 3] and [25, ch. 13] for $\epsilon = 0$ and $\epsilon = 2/(b(b+2))$, respectively.

4.3.3.1 Steady shear flow

For a homogeneous shear flow the following polymer viscosity can be derived:

$$\hat{\eta}_p = S3nk_\mathrm{B}\theta(1 - \epsilon b)/\dot{\gamma}, \tag{4.50}$$

with

$$p = \frac{b}{54(1-\epsilon)} + \frac{1}{18}, \qquad q = \frac{b\lambda_p\dot{\gamma}}{108(1-\epsilon b)}, \qquad S = 2p^{1/2}\sinh\left(\frac{1}{3}\mathrm{arcsinh}\Big(qp^{-3/2}\Big)\right). \tag{4.51}$$

Note that $\hat{\eta}_p$ depends on shear rate and should not be confused with the model parameter $\eta_p = nk_\mathrm{B}\theta\lambda_p$. Two limits can be observed:

$$\dot{\gamma} = 0: \quad \hat{\eta}_p = nk_\mathrm{B}\theta\lambda_p\frac{b}{\hat{b}+3}, \tag{4.52}$$

$$\dot{\gamma} \to \infty: \quad \hat{\eta}_p = nk_\mathrm{B}\theta\lambda_p\left(\frac{\sqrt{b/2}}{\lambda_p\dot{\gamma}}\frac{b}{\hat{b}}\right)^{2/3} \propto \dot{\gamma}^{-2/3}, \tag{4.53}$$

where the two cases of ϵ are given by the definition of $\hat{b}$ in eq. (4.24). For high shear rates $\dot{\gamma}$, shear-thinning is often described by the *power law*

$$\hat{\eta}_p = m\dot{\gamma}^{n-1}. \tag{4.54}$$

The *consistency parameter* m has no real physical meaning, as is already implied by its somewhat artificial units of $\mathrm{Pa\,s}^n$. For the FENE-P model this yields a power law coefficient of $n = 1/3$. The first normal stress difference is found to be:

$$N_{1,p} = \tau_{xx} - \tau_{yy} = 18nk_\mathrm{B}\theta S^2. \tag{4.55}$$

4.3.3.2 Small amplitude oscillatory shearing

The moduli G' and G'' for an FENE-P oscillatory shear flow can be obtained by the expansion of the covariance matrix in the velocity gradient around the equilibrium state. It is therefore only valid for small shear strain. Here only the case $\epsilon = 0$ is presented [19, eq. 43] [20, eq.50-51], which is very similar to the Oldroyd-B solution:

$$G' = \frac{G\lambda_p^2\omega^2}{(\frac{b+3}{b})^2 + \lambda_p^2\omega^2}, \qquad G'' = \eta_s\omega + \frac{(\frac{b+3}{b})G\lambda_p\omega}{(\frac{b+3}{b})^2 + \lambda_p^2\omega^2}. \tag{4.56}$$

with the same definition $G = \eta_p/\lambda_p$ as for Oldroyd-B.

4.3.3.3 Elongational flow

The behavior in elongational flow is also interesting, where an analytical solution exists for the normal stress $N_1 = \tau_{xx} - \tau_{yy}$, and thus the elongational viscosity $\eta_E = N_1/\dot{\epsilon}$ can be inferred. In the derivation, for the FENE-P constant ϵ only the case $\epsilon = 0$ will be considered.

For elongational flows ($u_x = \dot{\epsilon}x$, $u_y = -\dot{\epsilon}y/2$, $u_z = -\dot{\epsilon}z/2$) the constitutive equation reduces to (cf. [19]):

$$\begin{aligned} &Z\begin{pmatrix}\tau_{xx}&0&0\\0&\tau_{yy}&0\\0&0&\tau_{zz}\end{pmatrix} + \lambda_p\left(\frac{\mathrm{d}}{\mathrm{d}t}\begin{pmatrix}\tau_{xx}&0&0\\0&\tau_{yy}&0\\0&0&\tau_{zz}\end{pmatrix} + \begin{pmatrix}-2\tau_{xx}&0&0\\0&\tau_{yy}&0\\0&0&\tau_{zz}\end{pmatrix}\dot{\epsilon}\right)\\ &-\lambda_p\begin{pmatrix}\tau_{xx}+nk_\mathrm{B}\theta&0&0\\0&\tau_{yy}+nk_\mathrm{B}\theta&0\\0&0&\tau_{zz}+nk_\mathrm{B}\theta\end{pmatrix}\frac{\mathrm{d}\ln Z}{\mathrm{d}t} = nk_\mathrm{B}\theta\lambda_p\dot{\epsilon}\begin{pmatrix}2&0&0\\1&-1&0\\0&0&-1\end{pmatrix} \end{aligned} \tag{4.57}$$

With the definition of the dimensionless elongational rate $\Lambda_e = \dot{\epsilon}\lambda_p$ this simplifies in the stationary case to

$$Z\begin{pmatrix}\tau_{xx}&0&0\\0&\tau_{yy}&0\\0&0&\tau_{zz}\end{pmatrix} + \Lambda_e\begin{pmatrix}-2(\tau_{xx}+nk_\mathrm{B}\theta)&0&0\\0&\tau_{yy}+nk_\mathrm{B}\theta&0\\0&0&\tau_{zz}+nk_\mathrm{B}\theta\end{pmatrix} = 0. \tag{4.58}$$

This results in two independent equations:

$$ZT - 2D\Lambda_e = 0, \tag{4.59}$$

$$ZD - (D+T)\Lambda_e = \Lambda_e, \tag{4.60}$$

where the definitions $T = \mathrm{Tr}(\underline{\tau}_p)/(3nk_\mathrm{B}\theta)$, $D = (\tau_{xx} - \tau_{yy})/(3nk_\mathrm{B}\theta)$ and $B = 3/b$ apply and furthermore $Z = Z(T) = 1 + B(1 + T)$ holds (cf. eq. (4.19)). After elimination of T, this can be transformed into a quadratic equation in Λ_e (cf. [19]). Its solution reads:

$$P_1 = 2BD, \tag{4.61a}$$

$$P_2 = -5B^2D^2 - (B-1)BD + B, \tag{4.61b}$$

$$P_3 = 2B^3D^3 + (B+1)(B^2D^2 - BD), \tag{4.61c}$$

$$\Lambda_e = \frac{-P_2 \pm \sqrt{P_2^2 - 4P_1P_3}}{2P_1}. \tag{4.61d}$$

For $BD = 1$ the root in the above equation becomes zero, whereby $\Lambda_e = 1$ holds. When analyzing the limits for large and small D, one of the two solutions can be excluded as physically not reasonable (small D: otherwise negative viscosity; large D: otherwise no continuous derivative). It follows that for $D \leq 1/B$ the plus solution is valid, for $D \geq 1/B$ the minus solution. The valid limits are:

$$\Lambda_e \to 0: \quad \Lambda_e = (B+1)D, \tag{4.62}$$

$$\Lambda_e \to \infty: \quad \Lambda_e = BD/2. \tag{4.63}$$

Translating this back to the old nomenclature, solving for η_E and including the case $\epsilon = 2/(b(b{+}2))$, the limits now read [19, eq. 38] [22, eq. 13.5-63]:

$$\dot{\epsilon} = 0: \quad \eta_E - 3\eta_s = 3G\lambda_p \frac{b}{\hat{b}+3}, \tag{4.64}$$

$$\dot{\epsilon} \to \infty: \quad \eta_E - 3\eta_s \approx 2G\lambda_p b. \tag{4.65}$$

5 Lattice Boltzmann method

Finding a computational solution of the incompressible NSE can be targeted in multiple ways. The conventional class of Navier-Stokes solvers tries to discretize eq. (3.9) itself. The most general of these methods are the *finite difference* (FD) method, the *finite volume* (FV) method (which will be used in chapter 10 to solve the constitutive equations of viscoelastic fluids) or the *finite element* (FE) method. Another class of solvers are the particle-based solvers, which model particle interactions on the mesoscale and by this reproduce the incompressible NSE. The lattice Boltzmann method (LBM) belongs to the latter class and is a discretized version of the Boltzmann equation (3.17).

5.1 Basic algorithm

In the LBM, discretization takes place with respect to time, space and velocity. In other words, the LBM works on a regular 3D grid called *lattice* (representing space) with grid spacing Δx, a discrete time step Δt and a finite velocity set $\{\boldsymbol{c}_i\}$. Let's first look at the discretization with respect to velocity space. In order to reproduce the correct macroscopic behavior, the first three moments of the Boltzmann equation corresponding to mass, momentum and energy conservation should be captured as accurate as possible. Therefore, two steps in the discretization process of the velocity are important: firstly, f^{eq} from eq. (3.15) is approximated by the first three terms of its Hermite series expansion, whose coefficients are directly related to the conserved quantities density, momentum and energy. The coefficients can be computed analytically since f^{eq} itself has the same form as the generating function of the Hermite polynomials. Secondly, the moment integrals over the approximated form of f^{eq} can be written as a sum over a small number of discrete points, the so-called *abscissae*, via the Gauss-Hermite quadrature rule. In this process, the possible velocity sets $\{\boldsymbol{c}_i\}$ and their corresponding weights w_i are derived. The final form of the discrete equilibrium distribution reads

$$f_i^{\mathrm{eq}}(\rho, \boldsymbol{u}) = w_i \rho \left(1 + \frac{c_{i\alpha} u_\alpha}{c_s^2} + \frac{u_\alpha u_\beta (c_{i\alpha} c_{i\beta} - c_s^2 \delta_{\alpha\beta})}{2 c_s^4} \right), \tag{5.1}$$

where $c_s = 1/\sqrt{3}$ is the speed of sound. The same techniques can also be applied to the particle distribution function f. By doing so, the macroscopic moments (fluid density ρ and momentum $\rho\boldsymbol{u}$) are found as discretized moments:

$$\rho(\boldsymbol{r}, t) = \sum_i f_i(\boldsymbol{r}, t), \qquad \rho(\boldsymbol{r}, t)\boldsymbol{u}(\boldsymbol{r}, t) = \sum_i \boldsymbol{c}_i f_i(\boldsymbol{r}, t). \tag{5.2}$$

At this point of discretization, the conservation laws of the first three moments are still fulfilled exactly. It is only because of the discretization with respect to time and space that the LBM is a second order solver. In the algorithm, eq. (5.1) and (5.2) are only evaluated between discrete time steps Δt and at the grid positions separated by Δx. With $\boldsymbol{r}$ being a grid point, the full

discretized version of the Boltzmann equation (3.17) in the force-free case reads

$$f_i(\boldsymbol{r} + \boldsymbol{c}_i\Delta t, t + \Delta t) = f_i(\boldsymbol{r}, t) + \Omega_i(\boldsymbol{r}, t) \tag{5.3}$$

and is called the *lattice Boltzmann equation*.

Here, the evolution of $f_i(\boldsymbol{r}, t)$ is described by two terms, that correspond to the two main parts of the basic LBM algorithm called *streaming* and *collision*: The term on the left hand side expresses that populations $f_i(\boldsymbol{r}, t)$ move with velocity $\boldsymbol{c}_i$ to the neighboring point $\boldsymbol{r} + \boldsymbol{c}_i\Delta t$ at the next time step $t + \Delta t$ (streaming). At the same time, particles are redistributed due to their local interaction (collision), which is modeled by the collision operator Ω_i. From now on, the populations right before streaming will be referred to as f_i^*, so for normal streaming it always holds $f_i(\boldsymbol{r} + \boldsymbol{c}_i\Delta t, t + \Delta t) = f_i^*(\boldsymbol{r}, t)$.

Let's first take a closer look at possible velocity sets. To describe them, the established notation DdQq is used, where d denotes the number of spatial dimensions and q is the number of discrete velocities. While a larger q provides higher resolution and is therefore more accurate in general, memory consumption as well as the number of computational operations rise linear with q. In 3D simulations, D3Q19 is an often used standard as it represents a payoff between accuracy and efficiency. But throughout this thesis, also other sets like D2Q9, D3Q7, D3Q13, D3Q15 and D3Q27 will be used in different contexts (HK, FVM, advection tests). The sets available in FluidX3D are depicted in fig. 5.1. It should be noted, however, that D3Q13 is the minimal velocity set in 3D to simulate the NSE (D2Q7 is minimal in 2D) [26, ch. A.4]. Note also, that all sets have the *rest velocity* with zero magnitude at index $i = 0$ and that opposite velocity vectors have consecutive indices, which has advantages in terms of the implementation. The population with opposite velocity vector to f_i will be referred to as $f_{\bar{i}}$. In tab. 5.1 the velocity components as well as the weights w_i (needed for several computations below) for D3Q19 are given as an example. The values of the other sets as used in FluidX3D are documented in [11, ch. 3.2]. In tab. 5.1 the concept of velocity sets is extended by D2Q5. This set is not suitable for Navier-Stokes, but used to define a possible neighborhood for the Hoshen-Kopelman algorithm in chapter 9.1.2.2.

Set	Velocities $\boldsymbol{c}_i$	Number	Length $\lvert\boldsymbol{c}_i\rvert$	Weight w_i
D2Q5	$(0,0)$	1	0	-
	$(\pm1,0)$,$(0,\pm1)$	4	1	-
D3Q19	$(0,0,0)$	1	0	1/3
	$(\pm1,0,0)$,$(0,\pm1,0)$,$(0,0,\pm1)$	6	1	1/18
	$(\pm1,\pm1,0)$,$(0,\pm1,\pm1)$,$(\pm1,0,\pm1)$	12	$\sqrt{2}$	1/36

Tab. 5.1: Properties of two selected velocity sets. D3Q19 is a popular set suitable for Navier-Stokes simulations. D2Q5 defines a possible neighborhood for the Hoshen-Kopelman algorithm in chapter 9.1.2.2.

There exist many different possible collision operators. The software FluidX3D provides three of them, all of which can be used for Navier-Stokes simulations. The simplest among them is the BGK operator. It is obtained by exchanging f and f^{eq} in eq. (3.19) by its discretized versions f_i and f_i^{eq}, respectively. Since τ_{r} is the only time constant present here, it is also called *single relaxation time* (SRT) operator. The next operator in ascending accuracy is the *two relaxation time* (TRT) operator. It relaxes symmetric and antisymmetric linear combinations of f_i with two independent time constants, $\tau_{\mathrm{r},+}$ and $\tau_{\mathrm{r},-}$, respectively. They are related by the

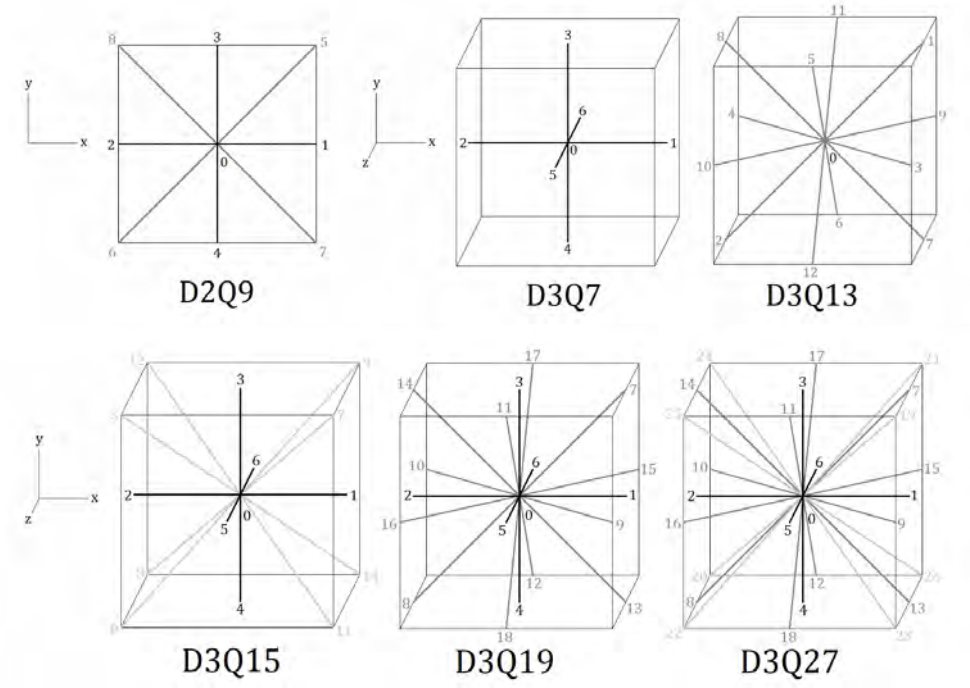


Fig. 5.1: The velocity sets available in FluidX3D. Image taken from [11].

"magic parameter" Λ_{TRT} [11]:

$$\tau_{\mathrm{r},+} = \tau_{\mathrm{r}}, \qquad \tau_{\mathrm{r},-} = \left(\frac{\Lambda_{\mathrm{TRT}}}{\tau_{\mathrm{r}}/\Delta t - 1/2} + \frac{1}{2}\right)\Delta t. \tag{5.4}$$

The symmetric relaxation time $\tau_{\mathrm{r},+}$ directly determines the fluid viscosity, while $\tau_{\mathrm{r},-}$ can be used to tune the accuracy and stability of the LBM solver. Throughout this thesis, $\Lambda_{\mathrm{TRT}} = 3/16$ will be used whenever TRT is chosen. This causes the boundary wall implemented via the bounce-back rule (cf. chapter 5.2) in planar poiseuille flow simulations to be exactly located between horizontal walls and fluid nodes [26, ch. 10.7.2]. The last option is the *multi relaxation time* (MRT) operator

$$\Omega_i = (\underline{M}^{-1}\,\underline{S}\,\underline{M}(f(\boldsymbol{r},t) - f^{\mathrm{eq}}(\boldsymbol{r},t)))_i. \tag{5.5}$$

Here, $\underline{M}$ is a $q \times q$ matrix that transforms from population space into moment space. The most important moments after transformation correspond to density ρ, energy density e, energy density squared ϵ, momentum density j_α, heat flux q_α and momentum flux $p_{\alpha\beta}$, with $\alpha, \beta \in \{x, y, z\}$. The remaining moments are mostly non-physical higher-order polynomials depending on the choice of the velocity set. $\underline{S}$ is a diagonal matrix of same size as $\underline{M}$ containing individual relaxation times $\tau_{\mathrm{r},i}$ for each moment. The relaxation times of the conserved quantities ρ and j_α are infinite, while the relaxation times of e and $p_{\alpha\beta}$ are determined by the kinematic shear viscosity. For the non-physical moments the corresponding relaxation rate $\Delta t/\tau_{\mathrm{r},i}$ is normally set to 1 (instant relaxation). The remaining relaxation times for ϵ and q_α can be used for tuning, which in principle allows for high accuracy. However, since tuning is not as easy as in the TRT case, throughout this thesis they will be set to 1 as well. Concerning computational cost, a naive implementation of MRT it is much more expensive than the former two collision operators. That is because it involves several matrix products of $q \times q$ matrices, q again denoting the number of discrete velocities of the used set. But if $\underline{S}$ and $\underline{M}$ are constants throughout the simulations, $\underline{M}^{-1}\,\underline{S}\,\underline{M}$ can be precomputed,

so only a single matrix-vector product remains. The software FluidX3D uses this optimization. Here, the MRT operator is available for all velocity sets except D3Q7 and D3Q27. The explicit expressions for $\underline{M}$ used in the software for each available velocity set are provided in [11, ch. 3.3.3].

The force-free case described above can be extended to the case including external forces. The form of the force density $\boldsymbol{F}_{\text{ext}}$ itself depends of the underlying physics and is not given by the LBM. The interaction of $\boldsymbol{F}_{\text{ext}}$ with the fluid in FluidX3D is implemented via the Guo forcing scheme. For this scheme the volume force enters the algorithm at two points: Firstly, as an additional term in the computation of the fluid velocity in eq. (5.2), which changes to

$$\boldsymbol{u}(\boldsymbol{r},t) = \frac{1}{\rho}\sum_{i=0}^{q-1}\boldsymbol{c}_i f_i(\boldsymbol{r},t) + \frac{1}{2\rho}\boldsymbol{F}_{\text{ext}}(\boldsymbol{r},t)\Delta t. \tag{5.6}$$

Missing out this so-called *half-force correction* would lead back to a first-order rather than second-order space-time accuracy [26, ch. 6.2]. Secondly, an additional source term S_i enters the LB equation (5.3), which now reads

$$f_i(\boldsymbol{r}+\boldsymbol{c}_i\Delta t, t+\Delta t) = f_i(\boldsymbol{r},t) + (\Omega_i(\boldsymbol{r},t) + S_i)\Delta t. \tag{5.7}$$

The concrete form of S_i depends on the choice of the collision operator Ω_i. For SRT, TRT and MRT its form as used in FluidX3D is documented in [11, ch. 3.5].

Finally, it should be mentioned, that all LB simulations are non-dimensional. When both SI and non-dimensional quantities occur in the same context, SI quantities will be marked by a subscript and have units, e.g. $\rho_{\text{SI}} = 1000\,\text{kg}\,\text{m}^{-3}$. After the conversion of a SI quantity into a non-dimensional quantity, the non-dimensional value is often referred to as being given in *lattice units*. In fluid mechanics, three independent conversion factors are needed in order to convert a system from SI units into lattice units. Given that the quantities ρ, L and u denote a typical density, length and velocity in the system, the ratios ρ/ρ_{SI}, L/L_{SI} and u/u_{SI} are sufficient to convert all other quantities like time, viscosity, surface tension etc. During this chapter, the grid spacing Δx as well as the time spacing Δt in lattice units was assumed to equal one. This is a common choice and will always remain so throughout this thesis.

5.2 Boundaries

In this thesis, different boundary conditions are used in connection with LB. In the simplest case, *periodic boundary* conditions are applied. Fluid leaving the simulation domain on one side immediately re-enters it at the opposite side. E.g. in a domain with side length s_x in x-direction the grid point at coordinate $(s_x - 0.5, y, z)$ is reached from grid point $(0.5, y, z)$ with the velocity vector $\boldsymbol{c}_i = (-1, 0, 0)$ in a single time step. Furthermore, there are velocity boundary conditions in form of *moving no-slip bounce-back boundaries*. Here it holds

$$f_i(\boldsymbol{r}_w + \boldsymbol{c_i}\Delta t, t+\Delta t) = f_{\bar{i}}^*(\boldsymbol{r}_w + \boldsymbol{c_i}\Delta t, t) + \frac{6\rho w_i \Delta t^2}{\Delta x^2}\boldsymbol{c}_i \cdot \boldsymbol{u}_w, \tag{5.8}$$

where $\boldsymbol{r}_w$ is a boundary node with velocity $\boldsymbol{u}_w$ and $\boldsymbol{r}_w + \boldsymbol{c}_i\Delta t$ is a fluid node [27]. This corresponds to a boundary wall moving with velocity $\boldsymbol{u}_w$. It is important that the actual position of the boundary is approximately between the fluid nodes and the boundary nodes, not on the boundary nodes themselves. More precisely, assuming the boundary position at the boundary node itself

would introduce a first-order error, while assuming the wall location exactly between the two nodes makes the method formally second-order accurate. Furthermore, a distinction is made between half-way and full-way bounce-back methods, where in the case of FluidX3D the half-way method is implemented [26, ch. 5.3.3.2]. The case of resting walls, i.e. non-moving no-slip bounce-back boundary conditions, is obtained for $\boldsymbol{u}_w = 0$. For this case, the populations are simply reflected at the boundary nodes in the streaming step.

Finally, there is the *equilibrium boundary*. A corresponding equilibrium node at position $\boldsymbol{r}_e$ has a fixed density ρ_e and velocity $\boldsymbol{u}_e$. Instead of using eq. (5.7), the populations are set to their equilibrium value, i.e. $f_i(\boldsymbol{r}_e, t) \equiv f_i^{\mathrm{eq}}(\boldsymbol{r}_e, t)$. Thus, in the streaming step

$$f_i(\boldsymbol{r}_w + \boldsymbol{c_i}\Delta t, t + \Delta t) = f_i^{\mathrm{eq}}(\boldsymbol{u}_e, \rho_e) \tag{5.9}$$

holds, where $\boldsymbol{r}_e + \boldsymbol{c}_i\Delta t$ is a fluid node. All incoming populations arriving at $\boldsymbol{r}_e$ during streaming are simply discarded, which makes this method non-reflecting. Furthermore, according to [26] it is necessary to use $\tau_{\mathrm{r}}/\Delta t = 1$, i.e. $\nu = 1/6$, in order to still have a second order solver. For all other cases, the method is first order [28], so moving bounce-back boundaries are preferable to equilibrium boundaries if possible. Another difference is that for equilibrium boundaries the given values for density and velocity apply to the nodes themselves, not to positions in between nodes as it is the case for bounce-back boundaries.

In the previous implementation of FluidX3D (cf. [11, ch. 3.4]), $\boldsymbol{u}_w$, $\boldsymbol{u}_e$ and ρ_e could only be specified location-dependent. Their constant values were read from the fields $\boldsymbol{u}$ and ρ at the corresponding grid positions, where the term *field* denotes a buffer storing a value for each lattice node in the computational domain. For some setups in this work, however, additional time dependence was necessary, so the boundary methods were extended to this end. This is done as follows: Two additional fields $\boldsymbol{u}_0$ and ρ_0 are introduced, in which location-dependent amplitudes are stored before the simulation starts. Furthermore, functions $f_u(\boldsymbol{u}_0, \rho_0, t)$ and $f_\rho(\boldsymbol{u}_0, \rho_0, t)$ can be specified [4]. Now, before each actual simulation step, simply set $\boldsymbol{u} = f_u(\boldsymbol{u}_0, \rho_0, t)$ and $\rho = f_\rho(\boldsymbol{u}_0, \rho_0, t)$ at the nodes marked as boundary. This method is simple and robust, but consumes a lot of memory.

[4] The functions must be specified in the form of valid OpenCL code, which is injected at the appropriate location before compiling the GPU code.

6 Volume of Fluid method

The LBM can be used for the simulation of free surface flows by combining it with a Volume-of-Fluid (VoF) approach for interface tracking [11, 29, 30]. The fluid interface position is tracked by a newly introduced field storing a *fill level* φ for every lattice node. Additionally, every lattice node can assume one of the three states *fluid*, *interface* or *gas*, marked by the flags F_F, F_I and F_G, respectively. Depending on the state of the node, the fill level is obtained by

$$\varphi(\boldsymbol{r},t) = \begin{cases} 1, & \text{if } \boldsymbol{r} \text{ is fluid,} \\ \texttt{min}\left(\texttt{max}\left(0, \frac{m(\boldsymbol{r},t)}{\rho(\boldsymbol{r},t)}\right), 1\right), & \text{if } \boldsymbol{r} \text{ is interface,} \\ 0, & \text{if } \boldsymbol{r} \text{ is gas.} \end{cases} \tag{6.1}$$

where m is an additional field storing the fluid mass. The fluid mass is a conserved quantity. The LBM populations f_i can directly be used to advect m:

$$m(\boldsymbol{r},t+\Delta t) = m(\boldsymbol{r},t) + \sum_{i=1}^{q-1} k(\boldsymbol{r},i,t) \cdot [f_{\bar{i}}^*(\boldsymbol{r}+\boldsymbol{c}_i\Delta t,t) - f_i^*(\boldsymbol{r},t))]. \tag{6.2}$$

The coefficient $k(\boldsymbol{r},i,t)$ is always 1 if $\boldsymbol{r}$ is a fluid node. If $\boldsymbol{r}$ is an interface node, it holds

$$k(\boldsymbol{r},i,t) = \begin{cases} 0, & \text{if } \boldsymbol{r}+\boldsymbol{c}_i \text{ is gas,} \\ \frac{1}{2}[\varphi(\boldsymbol{r}+\boldsymbol{c}_i,t) + \varphi(\boldsymbol{r},t)], & \text{if } \boldsymbol{r}+\boldsymbol{c}_i \text{ is interface,} \\ 1, & \text{if } \boldsymbol{r}+\boldsymbol{c}_i \text{ is fluid,} \end{cases} \tag{6.3}$$

which means there is no mass flux to gas cells.

Gas nodes are excluded from all calculations and do not have valid LBM populations. Therefore, interface nodes cannot stream populations from gas nodes. Instead, the free surface boundary conditions for an interface node at positions $\boldsymbol{r}$ and a gas node at position $\boldsymbol{r}+\boldsymbol{c}_i$ reads

$$f_{\bar{i}}(\boldsymbol{r},t+\Delta t) = f_i^{\text{eq}}(\rho_b,\boldsymbol{u}_b) + f_{\bar{i}}^{\text{eq}}(\rho_b,\boldsymbol{u}_b) - f_i^*(\boldsymbol{r},t). \tag{6.4}$$

Eq. (6.4) ensures mass and momentum conservation for interface nodes and approximates a free surface boundary condition with first order spatial accuracy [31]. The velocity $\boldsymbol{u}_b$ at the interface is approximated as the local fluid velocity $\boldsymbol{u}(\boldsymbol{r},t)$. The density ρ_b at the interface is defined as

$$\rho_b(\boldsymbol{r},t) = \frac{1}{c_s^2} p_b(\boldsymbol{r},t) = \frac{1}{c_s^2}\left(p_0 - \Delta p(\boldsymbol{r},t)\right), \tag{6.5}$$

where $p_0 \equiv \frac{1}{3}\rho_0$ is the ambient pressure and $\Delta p = 2\sigma\kappa$ is the Young-Laplace pressure consisting of the surface tension parameter σ and the local mean curvature κ. The quality of the free surface simulations strongly depends on the accuracy of the approximation of κ. It usually has to be extracted from the fill levels φ, and a variety of approaches to do so exist [11, ch. 7][30]. In FluidX3D, the exact positions of the interface of an F_I-node and all its F_I-neighbors are

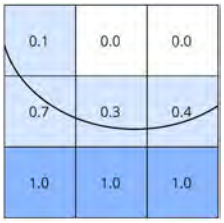


Fig. 6.1: 2D illustration of the Volume-of-Fluid method. An interface (light blue nodes) with thickness of exactly one lattice node divides the gas phase (white nodes) from the fluid phase (dark blue nodes). From the fill levels $\varphi \in [0, 1]$ (example values given at node centers) the position and mean curvature of the actual sharp interface (black curved line) is reconstructed by PLIC and least-squares fitting of an paraboloid. Image taken from [11].

computed via piecewise linear interface construction (PLIC). Then the local curvature is obtained by fitting a paraboloid curve to these positions using the least-squares method. Finally, the mean curvature is calculated analytically from the fitting parameters.

After computing the mass flux and reconstructing missing populations of interface nodes according to eq. (6.2) and (6.4), the flags F_F, F_I and F_G marking the node states have to be updated. However, the whole algorithm only works if the interface is ensured to be exactly one lattice node thick, i.e. if every F_G-node is separated from every F_F-node by exactly one F_I-node. Updating the flags directly from the fill levels φ of the next iteration might violate this condition. Instead, a flag handling procedure as described below is used. To avoid race conditions, it has to make use of additional kernels and newly introduced flags. F_{IF} marks a planned change from $F_I \to F_F$ without already conducting the change. The definition of F_{IG} and F_{GI} work analogously [5]. The flag handling procedure originally developed in [11, ch. 6.3] is described in detail here, as it will be extended to simulate bubbles in chapter 9.1.2.

Kernel 1: In the end of kernel `stream_collide`, for F_I-nodes the flags F_{IG} and F_{IF} are set according to

$$m(\boldsymbol{r}, t) \begin{cases} > (1+\epsilon)\rho(\boldsymbol{r}, t), & \text{set } F_{IF}, \\ < (0-\epsilon)\rho(\boldsymbol{r}, t), & \text{set } F_{IG}, \end{cases} \tag{6.6}$$

where $\epsilon \ll 1$ is a threshold to avoid flickering of flags between consecutive simulation steps. Furthermore, F_I-nodes with no neighboring F_G-nodes are marked as F_{IF}, while F_I-nodes with no neighboring F_F-nodes are marked as F_{IG}.

Kernel 2: The kernel `surface_1` looks for F_{IF}-nodes next to F_{IG}-nodes. The conflict is resolved by clearing the F_{IG}-flags, so these interface nodes are not marked for a change to gas nodes any more. Furthermore, F_G-nodes next to F_{IF}-nodes are marked as F_{GI}.

Kernel 3: The kernel `surface_2` computes the previously undefined populations f_i for nodes marked as F_{GI} via $f_i := f_i^{\text{eq}}(\rho_{\text{avg}}, \boldsymbol{u}_{\text{avg}})$, where ρ_{avg} and $\boldsymbol{u}_{\text{avg}}$ are obtained by averaging over the fluid and interface neighbors. Moreover, if a F_{IG}-node has a neighboring F_F-node or F_{IF}-node, the flags of these nodes are cleared and instead the F_I-flag is set, turning these nodes into interface. Moreover, the F_{IG}-flag is applied, i.e., for F_{IG}-nodes the F_I and the F_{IG} flags are cleared, and they are converted to gas nodes by setting the F_G-flag.

[5] The flag F_{GI} does actually not exist in the implementation. To save memory capacities, it is represented by the flag combination (F_{IG} `AND` F_G), which does not occur otherwise.

Kernel 4: The kernel `surface_3` applies the F_{IG}-flag, i.e., for F_{IG}-nodes the F_I and the F_{IG} flags are cleared, and they are converted to gas nodes by setting the F_G-flag. Analogously, the F_{IF}-flag is applied. By now, all node types for the next iteration have been set. The fluid mass m is restricted to

$$m \begin{cases} = \rho, & \text{fluid node,} \\ \in]-\epsilon, \rho+\epsilon[, & \text{interface node,} \\ = 0, & \text{gas node.} \end{cases} \tag{6.7}$$

To ensure mass conservation, any mass added/removed during this restriction is stored as the excess mass m_{ex} for every node in an newly introduced field. Then, the fill levels for the next iteration are set according to eq. (6.1). At the very beginning of the kernel `stream_collide`, the excess mass m_{ex} will be distributed to all neighboring F_F-nodes and F_I-nodes and added to their fluid mass m. To be able to do this in parallel, the distribution is already prepared here by dividing m_{ex} by the total number of fluid and interface neighbors[6].

[6] A neighbor might not have already applied the flag changes due to parallelization. Still their future state can already be read out from the flags F_{IG}, F_{IF} and F_{GI} marking the state change.

7 Immersed boundary method

In chapter 5.2 boundary methods are described, that work directly on the populations of the LBM. As a consequence, boundaries can only be defined on a regular grid. Non-planar boundaries can only be approximated by marking discrete lattice nodes, therefore the flow solution near such boundaries contains numerical errors caused by the staircase effect. As an additional restriction, the boundary position cannot change over time. In order to simulate curved, deformable, moving objects, the *immersed boundary method* (IBM) is used instead.

7.1 The general method

In the IBM, two coordinate systems are used. The fluid properties are defined on an Eulerian grid, which is regular and stationary (non-moving). In our case, the LBM provides this grid. For the boundaries, a Lagrangian mesh is introduced, which in general is non-stationary and unstructured. The following explanation will be restricted to the case of a 2D massless boundary immersed in 3D space. The IBM boundary positions will be referred to as $\boldsymbol{x}^{(i)}(t)$ and they will be called particles, while for the LBM grid positions still the notation $\boldsymbol{r}$ applies. The interaction between Eulerian grid and Lagrangian mesh is bi-directional [32]: The particle i at position $\boldsymbol{x}^{(i)}(t)$ is advected with fluid velocity

$$\dot{\boldsymbol{x}}^{(i)}(t) = \sum_{\boldsymbol{r}\in\text{grid}} \boldsymbol{u}(\boldsymbol{r}, t+1)\phi_{\text{IBM}}(\boldsymbol{x}^{(i)} - \boldsymbol{r}), \tag{7.1}$$

where ϕ_{IBM} is an appropriate interpolation stencil. The forces $\boldsymbol{\mathcal{F}}^{(i)}(\boldsymbol{x}^{(i)}(t), t)$ acting on particle i are spread to the ambient fluid according to Newton's law "actio = reactio", which is done by

$$\boldsymbol{\mathcal{F}}_{\text{IBM}}(\boldsymbol{r}, t) = \sum_{\boldsymbol{x_i}\in\text{mesh}} \boldsymbol{\mathcal{F}}^{(i)}(t)\phi_{\text{IBM}}(\boldsymbol{x}^{(i)} - \boldsymbol{r}). \tag{7.2}$$

Finally, $\boldsymbol{\mathcal{F}}_{\text{IBM}}(\boldsymbol{r}, t)$ can be translated into a force density and included into the LBM via the Guo forcing scheme described in chapter 5. Note that the particles only interact with each other over the forces they exert on the fluid. Also note that there is no direct boundary condition for the LB populations. They can cross the boundaries of the IBM without restriction, but the macroscopic fluid behaves as if there was a boundary. As a consequence, a closed 2D IBM boundary is filled with fluid as well.

The interpolation stencil ϕ_{IBM} is normally short-ranged with finite cut-off length. In the implementation of FluidX3D, trilinear interpolation is used [11, ch. 3.8]. Thus, only a cube of eight lattice nodes around $\boldsymbol{x}^{(i)}(t)$ interacts with particle i as visualized in fig. 7.1. This reduces the required memory read/write operations to a minimum.

As further described in chapter 8, for algorithms running in parallel it is aimed that during a single kernel call each memory address is guaranteed to be written by a single thread only.

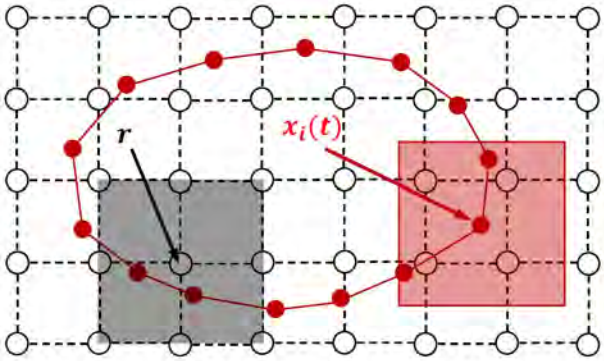


Fig. 7.1: 2D illustration of the IBM. The Eulerian grid (black) is fixed and regular, while the Lagrangian mesh (red) is non-stationary and unstructured. During velocity interpolation, all lattice nodes within the red square region around $\boldsymbol{x}^{(i)}$ are used to determine the velocity of particle i. During force spreading, all particles within the black square region contribute to the force acting on node at position $\boldsymbol{r}$.

The force spreading according to eq. (7.2) is the only exception in FluidX3D: multiple particles might add volume forces to a single lattice node at the same time. Possible race-conditions are therefore bypassed with `atomic_add` operations [11, ch. 3.8]. Besides performance drawbacks, these operations have the disadvantage that simulations become non-deterministic, since the order of terms during addition is random. But in total there is still a large performance gain (about a factor of 20) compared to outsourcing the IBM to the CPU.

7.2 IBM for deformable objects - the capsule model

7.2.1 Modeling of fluid-particle interaction

Up to now, the general two-way interaction between particles and fluid has been described. By concretely modeling the particle forces $\boldsymbol{\mathcal{F}}^{(i)}$, a big variety of applications are possible. In a simple case, the IBM particles have no coupling forces among each other. E.g. in microplastics research, each microplastics particle can be modeled as a single IBM particle being exhibited to buoyancy and a hard potential permitting them to cross the air-water interface [10]. If one introduces coupling forces, rigid walls with non-trivial shapes (i.e. not made up of staircases) and rigid or quasi-rigid objects can be modeled [33, ch. 4]. However, the IBM reveals its strength especially in the context of deformable objects.

For deformable objects, the present deformation state causes elastic forces. The force on particle i is found by looking at deformation energy E_D and using the principle of virtual work

$$\boldsymbol{\mathcal{F}}^{(i)}(\boldsymbol{x}^{(i)}) = -\frac{\partial E_D}{\partial \boldsymbol{x}^{(i)}}. \tag{7.3}$$

In order to apply the algorithms as outlined below, in addition to the current positions of the particles that together make up a capsule, the connections between the particles must also be known. This is realized by the particles representing the vertices of triangles. The initial states used in this work are shown in fig. 7.2, where the edges represent the connections between particles. They all discretize a sphere using the Loop subdivision surface scheme [34]. Starting from an icosahedron, all triangles are subdivided recursively, where the number of recursive steps determines the number of resulting triangles.

(a) (b) (c) (d) (e)

Fig. 7.2: Generation of the capsule mesh visualized. Starting from an icosahedron (a), in (b) to (e) the triangles are recursively split increasing resolution with each step. The number of triangles n_Δ of the meshes (d) and (e) equals 1280 and 5120, respectively.

7.2.2 Stretching via neo-Hookean or Skalak law

The deformation energy of capsules arises due to shear elasticity, area dilatation and bending forces. The former two effects are often modeled by empirical laws. For capsules, the neo-Hookean law is widely used (e.g. [35]), whose in-plane energy density due to stretching ϵ_S can be written as

$$\epsilon_S = \frac{\kappa_S}{6}\left(I_1 + \frac{1}{I_2+1} - 1\right). \tag{7.4}$$

The model parameter κ_S is the shear modulus for the in-plane tensions. I_1 and I_2 are the strain invariants, which are obtained by the principle in-plane stretch ratios λ_1 and λ_2 via $I_1 = \lambda_1^2 + \lambda_2^2 - 2$ and $I_2 = \lambda_1^2\lambda_2^2 - 1$. In the context of RBCs, on the other hand, the Skalak law is very prominent [36]:

$$\epsilon_S = \frac{\kappa_{S,1}}{4}\left(\frac{1}{2}I_1^2 + I_1 - I_2\right) + \frac{\kappa_{S,2}}{8}I_2^2. \tag{7.5}$$

For I_1 and I_2 the same definitions as for the neo-Hookean law hold, the shear moduli $\kappa_{S,1}$ and $\kappa_{S,2}$ are again model parameters. Independent of the chosen law, by surface integration

$$E_S = \oint_{S_0} \epsilon_S \, \mathrm{d}S_0 \tag{7.6}$$

the total energy is obtained, where S_0 denotes the surface of the *reference* state. For all simulations in this thesis, the reference state is defined by the initial state. Finally, the actual force $\boldsymbol{\mathcal{F}}_S^{(i)}(\boldsymbol{x}^{(i)})$ on node i is found as an analytical expression by using eq. (7.3) together with the FE method. Details are elaborated in [35, 37].

The bending energy, on the other hand, is calculated via the local mean curvature of the membrane surface. This can be implemented approximating the Laplace-Beltrami operator as described in the work of Gompper and Kroll [35, 38]. Since the bending forces won't be included in the modeling of the capsules (i.e. the bending modulus κ_B will be zero for all times), an in-depth description will be skipped.

7.3 Volume tracking

As mentioned above, in the standard IB-LBM the fluid inside the deformable objects has the same parameters (e.g. viscosity, density) as outside. This is not always desirable. E.g. in the simulation of RBCs moving through blood vessels, the fluid inside the cell consists of a hemoglobin

solution, while outside it is surrounded by blood plasma. Also in the context of this thesis, in chapter 10.10.3 a Newtonian capsule embedded in a viscoelastic fluid will be simulated, where interior and exterior fluids differ. Fluid properties are very easy to change locally in LBM: the parameter set used for the simulation can be switched for each node individually using a flag field. Then, only two things are important: On the one hand, conservation laws might be violated at the interface between the two parameter sets. This problem will be discussed in chapter 10.8.3.1 in the context of the conservation of polymer stress. Another difficulty is how to obtain the flag field from the IBM data (particle positions and triangle information), which will be discussed here.

One approach would be to use the ray-casting method. In this method, for each lattice node a ray in an arbitrary direction is generated, leading from the node itself to the edge of the simulation domain. The number of membrane crossings is counted using the triangle mesh. This method is very costly to do in every time step[7] and furthermore not compatible with periodic boundary conditions. Therefore it is only used during initialization and instead an implementation of the algorithms described in [39] is used, which will be referred to as the *InOut* algorithm. Assuming that the Lagrangian mesh does not move further than one lattice node during one time step, only the lattice nodes next to the capsule surface might have changed their inside/outside status and are the only ones to be considered. Their inside/outside status can be checked via the distance vector $\boldsymbol{d}$ from the node mid point to the capsule surface and the normal vector $\hat{\boldsymbol{n}}$ of the closest capsule vertex. This can simply be done via the scalar product, i.e.

$$\boldsymbol{d} \cdot \hat{\boldsymbol{n}} = \begin{cases} < 0, & \text{inside,} \\ \geq 0, & \text{outside.} \end{cases} \tag{7.7}$$

This criterion is not sufficient in rare cases, where the angle between adjacent triangles is small. Therefore an additional correction step is applied: Assuming the boundary between inside/outside nodes to be locally smooth, strange flag geometries (e.g. a node marked as inside enclosed by outside nodes) should not occur. As described in [39], such geometries can be detected by counting the inside/outside flags of one nodes 26 neighbors. If the count exceeds a certain threshold, the inside/outside flag has to be inverted. This strategy is successfully used in ESPResSo. Since this strategy is not compatible with GPU parallelization due to possible race conditions, in FluidX3D a different correction method is used. In the implementation of Moritz Lehmann, each lattice node is assigned a cube-shaped volume with side length Δx. A distance vector $\boldsymbol{d}_j$ to the capsule surface is calculated for all corner points $j = 1, ..., 8$ of the cube around the respective lattice node. If one of the corner points is believed outside the boundary via eq. (7.7), the whole node is marked as outside.

[7] For N lattice nodes and T triangles the algorithm scales with $\mathcal{O}(NT)$.

8 GPU parallelization

As seen in chapter 5, the LB algorithm is highly parallel. Regarding performance, this allows for orders of magnitude gains when using massively parallel hardware, especially in the case of *graphics processing units* (GPUs). Since these are mainly used in gaming industry, where computational speed is more important than computational accuracy, they are especially optimized for the use of `float32` (hereafter: `float`). Fortunately, this accuracy is sufficient for most applications of the LBM, which is why all computations occurring on the GPU in this thesis could be performed with `float` accuracy. The only exception is the bioprinting simulation from chapter 10.10.3, which was found to only be working when using `float64` (hereafter: `double`). The parallel code of FluidX3D was programmed in the OpenCL [40] language. This language has the advantage over CUDA, for example, that it is designed vendor independently for *Nvidia*, *AMD*, *Intel* and others.

The CPU side of FluidX3D is programmed in C++. Here, non-parallel computations can take place; furthermore, OpenCL kernels are enqueued in the OpenCL queue. Within a kernel execution, no dynamic memory can be allocated; all buffers must be fixed at the time of enqueuement. For each kernel, an integer index space must be defined. It specifies how many work-items are available for a kernel and further assigns a unique ID to each work-item. For a typical LBM kernel, the index space is chosen to be as large as the total number of lattice nodes, so that the calculations for each lattice point are performed by a separate work-item. On the hardware side, each work-item is processed by a single GPU thread. The order in which the entire index space is processed is random. On GPUs, the distinction between global and private memory is particularly relevant. Reading/writing is associated with high latency for global memory, but is the only way to store results between kernel calls. In fact, the pure LBM algorithm without extensions is typically in the memory limit because of its low arithmetic intensity, meaning it can only be accelerated by hardware with faster memory bandwidth. After caching data from global memory, computations are done in private memory, where memory latency is negligible. Every work-item can only read/write its own private memory.

In addition to avoiding unnecessary write/read operations to global memory, the following was taken into account when implementing GPU code:

- *Race conditions*: if the same memory address is written by two work-items during a single kernel execution, the last written value is kept, while all previous values are overwritten. Since the order of the work-items is random, this leads to non-deterministic behavior. For the same reason, a memory address must not be read and written simultaneously during a kernel execution. Again, it is not clear whether the read takes place before or after the write. To prevent this, double-buffers may be utilized, or read and write operations must be implemented in two different kernels.

- *Branching*: `if-else` branches should be avoided where possible. Work-items are grouped into work-groups (size: 32 work-items or more, typically 256) that cannot execute commands independently. For example, if only one part of the work-group executes an `if` statement,

the other part must wait and cannot already execute the `else` statement. This of course directly affects performance.

- *PCIe data transfer*: data between CPU and GPU is transferred via the PCIe bus. The transfer comes with comparably slow bandwidth and high latency. Data transfer in every time step can significantly reduce overall performance. Where this cannot be avoided, special buffers should be used. OpenCL offers the flag `CL_MEM_USE_HOST_PTR` which can speed up the PCIe transfer multiple times. In this thesis, this optimization has been successfully used in the context of simulating bubbles. In order to use `CL_MEM_USE_HOST_PTR`, however, a corresponding CPU buffer must be permanently assigned to the GPU buffer.

- *multi-GPU compatibility*: FluidX3D has the ability to run a simulation on multiple GPUs simultaneously [41]. This widens the memory limitations of a single GPU and also increases performance. For an extension to be compatible with multi-GPU, it must consist of local operations only, just like the pure LBM. In other words, a node may only change the values of its 27 surrounding neighbors within a time step.

With these restrictions, an OpenCL kernel has a "natural" length. It should accommodate as many computational operations as possible before having to write to global memory again. However, a new kernel must be used at the latest for global synchronization between discrete time steps and to avoid race conditions.

In tab. 8.1 one can find an overview of the extent to which the extensions used/implemented in this work meet the above guidelines. For the simulation of capsules using the IBM with volume-conservation switched on as well as for the simulation of bubbles, volume calculations of the respective object have to be performed. Since these can only be done non-locally, PCIe data transfer must take place in every time step and multi-GPU compatibility is no longer given. Moreover, the IBM implementation uses `atomic_add` operations for force spreading. Since the order of these parallel additions to the same memory location is random, for floating-point variables this leads to non-determinism due to rounding. Accordingly, two simulations with exactly the same initial and boundary conditions lead to different results at bit level. Other optimizations that are successfully used in FluidX3D but are not directly relevant in this thesis are documented in [11].

extension	deterministic	PCIe Data Transfer	multi-GPU compatible	comment
pure LBM	yes	no	yes	-
Oldroyd-B / FENE-P	yes	no	yes	-
IBM	no	yes	no	`atomic_add` operations, volume computation
VoF	yes	no	yes	-
Bubble	yes	yes	no	volume computation

Tab. 8.1: Compatibility of several extensions of FluidX3D with the GPU programming guidelines mentioned in the text.

9 Bubbles

9.1 Implementation

The bubbles will be modeled based on the assumption that the dynamics of the gas phase can be neglected, i.e., the problem is reduced to a single phase flow with a free boundary. The free boundary is implemented via the VoF method outlined in chapter 6. The bubble dynamics are introduced using the ideal gas law

$$p_B(t)V_B(t) = p_{B,0}V_{B,0} = \text{const}, \tag{9.1}$$

where p_B and V_B denote bubble pressure and volume, respectively. $p_{B,0} \equiv p_0 = \rho_0 c_s^2$ is the ambient pressure, $V_{B,0}$ is the volume of the bubble at ambient pressure. The pressure p_B enters the VoF method, by changing eq. (6.5) to

$$\rho_b(\boldsymbol{r}, t) = \frac{1}{c_s^2}\left(p_B(V_B) - \Delta p(\boldsymbol{r}, t)\right). \tag{9.2}$$

Two methods how to obtain $p_B(t)$ for each bubble $1 \leq B \leq N_B$ were implemented and will be discussed in the next two chapters.

9.1.1 Implementation idea 1: implicit pressure equilibration

For the *implicit pressure equilibration* method a new field ϵ is introduced. It is initialized with $\epsilon(\boldsymbol{r}) = (1 - \varphi(\boldsymbol{r}))\, p_B(\boldsymbol{r})$ and thus stores a value proportional to the kinetic energy of the ideal gas contained in the respective lattice node. The field ϵ is ensured to be zero for fluid nodes F_F throughout the whole algorithm. The pressure $p_B(\boldsymbol{r}, t)$ can be restored locally by

$$p_B(\boldsymbol{r}, t) = \frac{\sum_{i=0}^{q-1} \epsilon(\boldsymbol{r} + \boldsymbol{c}_i \Delta t, t)}{\sum_{i=0}^{q-1}(1 - \varphi(\boldsymbol{r} + \boldsymbol{c}_i \Delta t, t))}, \tag{9.3}$$

i.e., by taking the mean including the own node and all neighbor nodes. To reflect correct bubble dynamics, the field ϵ must satisfy two conditions. First, $\sum_{\boldsymbol{r} \in B} \epsilon(\boldsymbol{r}) = p_{B,0}V_{B,0}$ must hold for a bubble B, otherwise conservation of energy would be violated. Second, $p_B(\boldsymbol{r}, t)$ must assume approximately the same value for all $\boldsymbol{r} \in B$, since there should be no pressure differences within a bubble.

These conditions are implemented by the kernel `equilibration`, which is called after the kernel `surface_3` and before the kernel `stream_collide` of the next iteration. Here, for an interface or gas node at position $\boldsymbol{r}$ it holds:

$$\epsilon(\boldsymbol{r}, t + \Delta t) = \epsilon(\boldsymbol{r}, t) + \sum_{i \in \Gamma(\boldsymbol{r}, t)} \alpha_i(\boldsymbol{r}, t)\, \Delta \hat{p}_i(\boldsymbol{r}, t), \tag{9.4}$$

where $\Gamma(\boldsymbol{r}, t)$ denotes the set of neighbors of the node at position $\boldsymbol{r}$ that are labeled as gas or interface at time t. Furthermore, the definition

$$\Delta\hat{p}_i(\boldsymbol{r}) = \frac{\epsilon(\boldsymbol{r} + \boldsymbol{c}_i\Delta t)}{\hat{\varphi}(\boldsymbol{r} + \boldsymbol{c}_i\Delta t)} - \frac{\epsilon(\boldsymbol{r})}{\hat{\varphi}(\boldsymbol{r})} \tag{9.5}$$

applies. For the last equation, the definition $\hat{\varphi} = 1 - \varphi$ is introduced. The factor $\alpha_i(\boldsymbol{r}, t)$ controls how much a possible pressure difference between two neighboring nodes is equilibrated. Since

$$\Delta\hat{p}_i(\boldsymbol{r}) = -\Delta\hat{p}_{\bar{i}}(\boldsymbol{r} + \boldsymbol{c}_i\Delta t) \tag{9.6}$$

holds,

$$\alpha_i(\boldsymbol{r}) = \alpha_{\bar{i}}(\boldsymbol{r} + \boldsymbol{c}_i\Delta t) \tag{9.7}$$

must be demanded in order to guarantee conservation of energy. Suppose each node has only one neighbor, where all quantities of the node at position $\boldsymbol{r}$ are denoted by subscript a, all quantities of the neighbor at position $\boldsymbol{r} + \boldsymbol{c}_i\Delta t$ by subscript b. Moreover, let be $\alpha_i(\boldsymbol{r}) = \alpha$ here. Quantities marked with * depend on time step $t + \Delta t$, all other quantities depend on time step t. Then the pressure difference is equalized in one step if

$$\frac{\epsilon_a^*}{\hat{\varphi}_a} = \frac{\epsilon_b^*}{\hat{\varphi}_b} \tag{9.8}$$

$$\overset{(9.4),(9.6),(9.7)}{\iff} (\alpha\Delta\hat{p}_a + \epsilon_a)\hat{\varphi}_b = (-\alpha\Delta\hat{p}_a + \epsilon_b)\hat{\varphi}_a \tag{9.9}$$

$$\iff \alpha = \frac{\epsilon_a\hat{\varphi}_b - \epsilon_b\hat{\varphi}_a}{\Delta\hat{p}_a(\hat{\varphi}_a + \hat{\varphi}_b)} = \frac{\hat{\varphi}_a\hat{\varphi}_b}{\hat{\varphi}_a + \hat{\varphi}_b}. \tag{9.10}$$

But since $q - 1$ neighbors exist i.g., $\alpha_i(\boldsymbol{r})$ is chosen to be

$$\alpha_i(\boldsymbol{r}) = \frac{1}{q-1}\frac{\hat{\varphi}(\boldsymbol{r})\hat{\varphi}(\boldsymbol{r} + \boldsymbol{c}_i\Delta t)}{\hat{\varphi}(\boldsymbol{r}) + \hat{\varphi}(\boldsymbol{r} + \boldsymbol{c}_i\Delta t)}. \tag{9.11}$$

To improve the equilibration of the pressure, the kernel `equilibration` is called not only once, but N_{eq} times in a row within a simulation step. Nodes that change from interface to fluid in kernel `surface_3` will have their remaining ϵ distributed evenly between all neighboring interface nodes during the first call only. The same is done with interface nodes for which $\hat{\varphi} \ll 1$ applies. These are then excluded from the kernel `equilibration` for the rest of the $N_{eq} - 1$ calls. The reason for this is that for nodes with $\hat{\varphi} \ll 1$ equation (9.5) becomes too inaccurate.

With the method just described, bubbles inside a fluid rising to the atmosphere could be simulated. However, $N_{\text{eq}} > 100$ had to be chosen, which has a significant performance impact. The exact choice of N_{eq} strongly depends on how fast the interface moves during the simulation. If N_{eq} is chosen too small, an artificial slowdown of the interface is observable. In principle, the merging and splitting of bubbles can also be simulated. In this case the bubbles do not have to be tracked explicitly, a possible pressure difference due to the merging of two bubbles is compensated implicitly via the kernel `equilibration`. However, N_{eq} would have to be chosen many times larger, since at the moment of merging a large pressure gradient has to be compensated over a large number of lattice nodes. If one wanted to adjust N_{eq} dynamically in such situations, one would have to track the bubbles explicitly and the advantage of the implicitness of this method would be lost. For these reasons, the algorithm is discarded.

9.1.2 Implementation idea 2: bubble tracking

In the *bubble tracking* method, each bubble is assigned its own ID (implemented by consecutive positive integer values). A newly introduced field I stores these IDs and by this assigns each lattice node to its bubble. Since fluid nodes have no bubble membership, they are marked with the ID -1. The pressure within a bubble can thus be easily calculated by means of

$$p_B(t) = \frac{V_{B,0}}{V_B(t)}\, p_{B,0}, \tag{9.12}$$

where the current volume is computed by

$$V_B(t) = \sum_{\boldsymbol{r}\in B} (1 - \varphi(\boldsymbol{r}, t)). \tag{9.13}$$

The volume update is therefore performed in each time step via the fill levels φ. Since eq. (9.13) cannot be fully parallelized, the calculation is done on the CPU. So in each time step memory transfer of the fields φ and I is necessary.

9.1.2.1 Algorithm overview

An overview of the program flow is given in fig. 9.1. First, the ID field I is initialized using the Hoshen-Kopelman algorithm. For each bubble, the reference volume $V_{B,0}$ is set. The reference pressure is independent of the particular bubble $p_B, 0 \equiv p_0 = 1/3\rho_0$. For most applications, $V_{B,0}$ is chosen such that $p_B(t = 0)$ equals the sum of the mean surrounding fluid pressure and the Laplace pressure caused by surface tension. Then the call to `stream_collide` is made, the only change from the normal VoF method being that eq. (9.2) instead of eq. (6.5) is used as the pressure at the interface. The current $p_B(t)$ can be read locally using the bubble ID. An unchanged call to `surface_1` is made. Remember from chapter 6 that in `surface_2` the neighbors of F_{IG} nodes are considered. If these are F_F nodes, they are converted to F_I nodes. Since they are now no longer fluid nodes, they need a valid ID and are therefore initialized with the ID of the neighboring F_{IG} node. Thus, it is exploited that neighboring nodes are part of the same bubble. After the execution of kernel `surface_2` the new node types (fluid, gas or interface) are already fixed, but the information which node type is converted to which is still available. This is exploited in the kernel `bubble_1` to check whether a merge or split of bubbles has occurred due to the node changes. If a F_{IF} node causes a split, its ID is marked with -3. If two adjacent interface nodes with different IDs are found, a merge must have occurred and their IDs are marked with -2. Then kernel `surface_3` is called. The only change to the normal VoF method from chapter 6 is here that at a conversion $F_{IF} \to F_F$, the ID of the respective node is set to -1 (normal fluid node) if it has not already been marked as -3 (fluid node causing a split) during kernel `bubble_1`. The updated fields φ, I and the flag field must be transferred to the CPU. It is also reported to the CPU if a split/merge was triggered. The rest of the algorithm is done on the CPU and is divided into three parts. If a merge was triggered, it is executed. Then the recalculation of the volume $V_B(t)$ and the pressure $p_B(t)$ takes place. If a split was triggered, it is also executed. In every time step, the newly calculated pressure is transferred to the GPU. If a merge or split was present, additionally the updated field I has to be transferred and the triggers have to be reset.

9.1.2.2 Bubble labeling: Hoshen-Kopelman

Now it must be clarified by which algorithm the bubble is labeled with IDs. Labeling is not only relevant during initialization, but also when the bubble IDs have to be reassigned due to

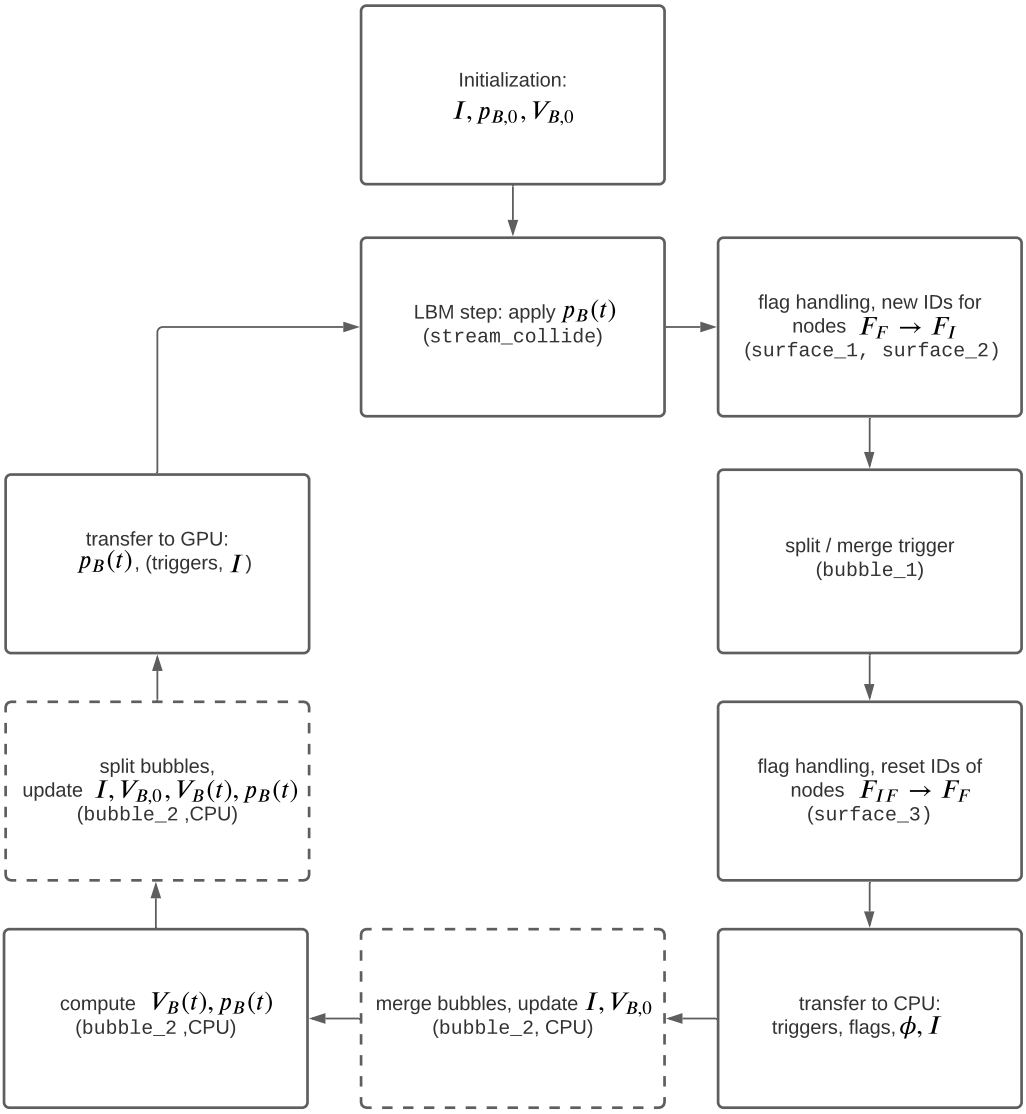


Fig. 9.1: Program flow of FluidX3D when using the bubble extension. The events marked with dashed frames are not executed in every time step, but have to be triggered.

bubble merges and splits. Therefore labeling should be done as performant as possible. [42] also model bubbles with the LBM and the VoF method. For the ID labeling they use the *flood fill* algorithm [43]. When the flood fill algorithm was implemented during this thesis, it proved to be poorly performing due to its recursive structure. Instead, the *Hoshen-Kopelman* (HK) algorithm is used, which was originally described in [44]. It is a special case of the class of *Union-Find* algorithms, which is a method for maintaining a collection of disjoint sets. An element x defines its membership to a set by its set leader, which is determined by the `find(`x`)` operation. Two sets A, B can be united by a `union(`A`,`B`)` operation, where the elements of both sets are assigned to a new common leader. Working on trees and using *union by rank* with *path compression*, a `union` operation needs $\mathcal{O}(1)$. The amortized time of the `find` operation, on the other hand, is $\mathcal{O}(\alpha(n))$, where n denotes the total number of elements and $\alpha(n)$ denotes the inverse Ackermann function[8]. $\alpha(n)$ is a limiting case of the iterated logarithm, grows extremely slowly, and can be assumed constant for all conceivable input quantities, since $\alpha^{-1}(5)$ is already larger than the number of atoms in the universe [45].

The implementation of the Hoshen-Kopelman algorithm is based on [46]. It uses a *naive union* in combination with *path compression*, so no extra rank field is needed. First, the ID field I is initialized using the flag field, where

$$I(\boldsymbol{r}) = \begin{cases} -2, & \boldsymbol{r} \text{ is } F_I \text{ or } F_G \text{ node,} \\ -1, & \boldsymbol{r} \text{ is } F_F \text{ node.} \end{cases} \tag{9.14}$$

The rest of the algorithm can be well described in pseudocode for the case D2Q5. The algorithm 1 consists of a raster scan of the LB lattice of size $s_x \times s_y$. Each time an interface or gas node (marked by ID -2) is encountered, a check is done including all neighbors who have already been scanned. A union is done between all those neighbors, excluding fluid nodes (marked by ID -1). Finally, the leader of the set obtained after union is assigned to the current node. If, on the other hand, the current node has no valid neighbors, it is assigned the next free ID. Algorithm 1 only yields *intermediate* IDs. Afterwards, the lattice must be iterated over one more time to assign each node its *final* ID (marked by the leader of the intermediate ID, who can be read out of T). While doing so, the IDs are reassigned such that they contiguously range from 0 to the number of IDs needed.

The `find` operation works with a tree T represented by an array of size N_T and initialized with $T[n] = n$ for all $0 \leq n \leq N_T - 1$. N_T must be at least as great as the number of *intermediate* IDs N_{inter} that occur during the Hoshen-Kopelman algorithm. N_{inter} is typically much larger than the number of *final* IDs N_{final}. The `find` operation with built-in path compression is described in algorithm 2. A `union` operation between two elements x and y can then be simply realized by $I[\texttt{find}(x)] = \texttt{find}(y)$.

In order to apply the algorithm just shown for bubbles, two things still have to be done. First, the algorithm must be extended to other DdQq neighborhoods. In this thesis, this was done for the sets D2Q5, D2Q9, D3Q7, D3Q13, D3Q15, D3Q19, and D3Q27, where $(q-1)/2$ neighbors must always be examined when searching for neighbor IDs, and more than two neighbors at once must be united for a `union` operation in general. Furthermore, the HK algorithm must be extended to periodic boundary conditions. To this end, after the `for`-loop over x in algorithm 1 (i.e., after line 24) one more time the node with $x = 0$ performs a `union` operation between

[8] More precisely, $\alpha(n) \equiv A(n,n)^{-1}$ holds, where $A(m,n)$ is the two-argument Ackermann-Péter function.

Algorithm 1 Hoshen-Kopelman for D2Q5

Input: initialized ID field I and tree T

Output: I containing intermediate IDs and T leading to final IDs

```
1:  next_ID ← 0;
2:  for y ← 0 to s_y − 1 do
3:      for x ← 0 to s_x − 1 do
4:          if I[x, y] = −2 then
5:              left ← −1; below ← −1;
6:              if x ≠ 0 then    // x = 0 has no left neighbor
7:                  left ← I[x − 1, y];
8:              end if
9:              if y ≠ 0 then    // y = 0 has no below neighbor
10:                 below ← I[x, y − 1];
11:             end if
12:             if (left = −1) and (below = −1) then    // left and below are fluid
13:                 I[x, y] ← next_ID;
14:                 next_ID ← next_ID + 1
15:             else if (left ≠ −1) and (below = −1) then    // only left has valid ID
16:                 I[x, y] ← find(left, T);
17:             else if (left = −1) and (below ≠ −1) then    // only below has valid ID
18:                 I[x, y] ← find(below, T);
19:             else    // both neighbors have valid ID
20:                 union(left, below, T);    // link cluster of left and below
21:                 I[x, y] ← find(left, T);
22:             end if
23:         end if
24:     end for
25: end for
```

Algorithm 2 find

Input: ID x and tree T

Output: y (leader of x) and the compressed tree T

```
1: y ← x;
2: while T[y] ≠ y do
3:     y ← T[y];
4: end while
5: while T(x) ≠ x do
6:     z ← T[y];
7:     T[x] ← y;
8:     x ← z;
9: end while
```

itself and its $(q-1)/2$ relevant neighbors. At the end of the `for`-loop over y and z (in case of 3D) the procedure is analogous. An example for the entire algorithm using D2Q5 is shown in fig. 9.2. The HK algorithm implemented during this thesis, as described so far, was already successfully brought to application in [10] for the tasks of counting and measuring simulated droplets.

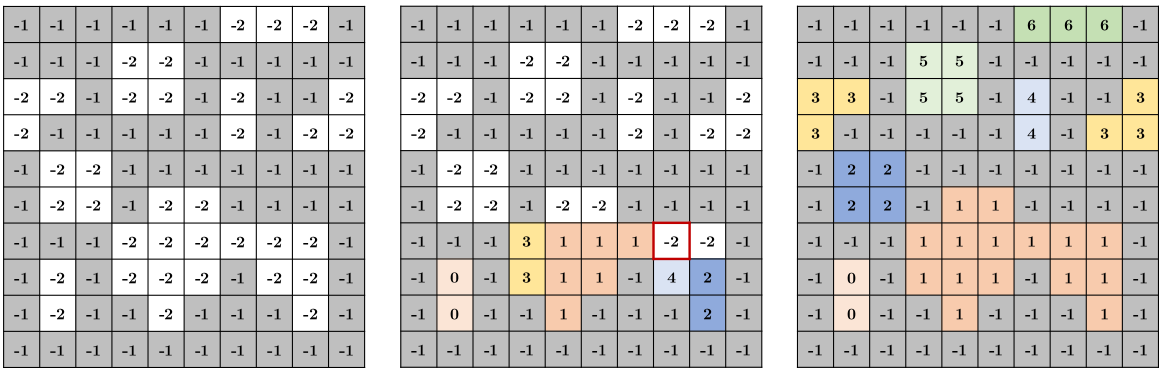


Fig. 9.2: Illustration of the HK algorithm used with a D2Q5 neighborhood and periodic boundary conditions. In (a) the ID field I is shown after initialization via eq. (9.14). In (b) a step of the algorithm is illustrated. Before processing the node framed in red, T has the entries $[0|1|2|1|2|5|6|7|...|N_T-1]$. After processing, the ID 1 is assigned to the node and the entries of T change to $[0|1|1|1|1|5|6|7|...|N_T-1]$. In (c) the final field I is shown after reassigning the bubble IDs in order to have a contiguous ID range from 0 to N_{final}.

9.1.2.3 Split/merge detection

With the help of the HK algorithm, a naive implementation of merging and splitting of bubbles can already be worked out. In each time step, the HK algorithm is executed on the entire field I. Then, based on the coordinates of all F_G and F_I nodes, a mapping between old and new IDs is established. If one old ID is mapped to two new ones, the associated bubble must have split in the current step. If, on the other hand, two old IDs are mapped to one new ID, two bubbles must have merged. The same principle can be applied to a simultaneous merge/split of more than two bubbles. Finally, the reference volumes $V_{B,0}$ must be adjusted. Let β be a set of bubbles to merge. Then the reference volume of the new bubble B_{new} is calculated from $V_{B_{\text{new}},0} = \sum_{B_{\text{old}}\in\beta} V_{B_{\text{old}},0}$. Given a split of one bubble B_{old} into multiple bubbles $B_{\text{new}} \in \beta$, on the other hand, $V_{B_{\text{new}},0} = V_{B_{\text{old}}} p_{B_{\text{old}}}/p_0$ is set. In many time steps, neither a split nor a merge is present. Nevertheless, the naive implementation must run the HK algorithm on the entire field I. This motivates the implementation of triggers, which already determine locally on the GPU whether a merge/split can be present at all and thus prevent an unnecessary execution of the HK algorithm.

9.1.2.4 Optimization: split/merge on GPU

The idea of the merge trigger is very simple. During the kernel `bubble_1` all future F_I and F_G nodes test if a neighboring F_I or F_G node with a different ID than their own exists. If this is the case, a merge of bubbles must have occurred. The merge trigger is activated and the corresponding nodes are marked with -2. By means of these markings the ID mapping

between old and new iteration can take place on the CPU without having to use the HK algorithm.

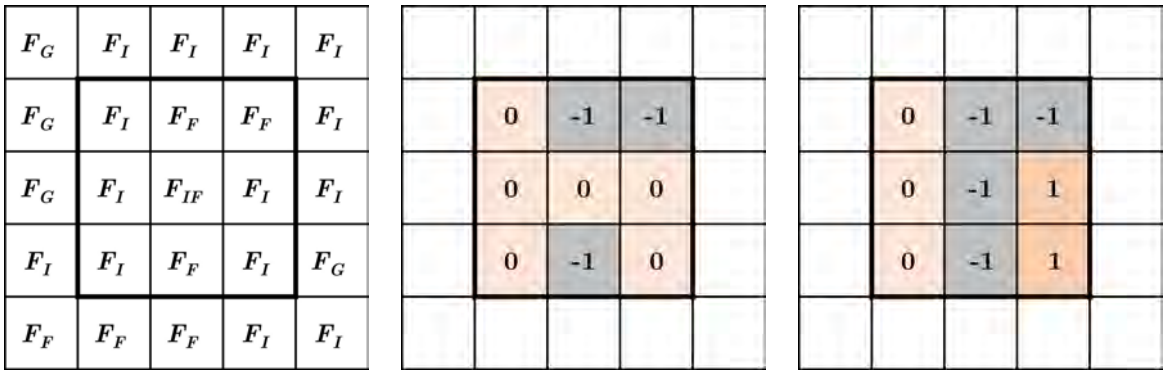
The split trigger is much more difficult to implement. [42] look for a grid configuration where in the neighborhood of a fluid node two interface sections of the same bubble have anti-parallel surface normals. This criterion assumes that the surface normals are present in memory, which is not yet the case in the VoF module of FluidX3D. Furthermore, it is not quite clear up to which angle the surface normals should be considered anti-parallel: on the one hand, one wants to minimize the number of false positives, i.e., the falsely triggered splits. These cause unnecessary CPU work. On the other hand, there must be no false negatives, since these are equivalent to an error in the algorithm and can no longer be corrected in later time steps. All in all, it is difficult to show from which minimal angle this method does not yield false negatives for all conceivable grid configurations. Therefore, another method was developed in this thesis. For this method, all nodes which have the flag F_{IF} are considered. Only this node type can cause a split. In a cubic region (square in 2D) around this node the HK algorithm is executed locally several times. The size of the region can be chosen depending on the application. A cube with side length $3\Delta x$ is still compatible with the present multi-GPU implementation, a cube with side length $5\Delta x$ reduces the number of false positives somewhat more at the expense of the runtime of the trigger itself. The boundaries of the local cube are not treated periodically, since it represents only a small part of the entire simulation domain.

Two criteria are tested on this cubic region. The first criterion determines via the local HK the number of bubbles n_F and n_I that are counted, if the central F_{IF} node is replaced by an F_F or F_I node, respectively. All nodes except the central node are interpreted as their future type (e.g. a F_{IG} node is interpreted as F_G node). If $n_F > n_I$ holds, the conversion $F_{IF} \rightarrow F_F$ happening in `surface_3` will locally lead to a split[9], so the trigger is activated and the central F_{IF} node's ID is marked with -3. In grid constellations with several F_{IF} nodes next to each other, a split can occur which is not covered by the first criterion. The second criterion therefore checks whether the central node has at least one neighboring F_{IF} node. If yes, first the HK algorithm assigns separate IDs to all disjoint clusters of F_{IF} nodes within the test region. Second, it is checked within the cluster containing the central node, if F_{IF} nodes exist that are not neighbor to any *native bubble* (a bubble to which the central F_{IF} node is neighbor to) [10]. If any such node is neighbor to a *foreign bubble* (a bubble to which the central F_{IF} node is not neighbor to) or is adjacent to the border of the test region, the split trigger is activated and the central node is marked with -3. Example configurations for both criteria in a D2Q9 neighborhood are shown in fig. 9.3 and 9.4.

If a split is triggered, the HK algorithm must be executed on the CPU. However, it is sufficient to reassign IDs only to the bubbles that are adjacent to a node marked with -3 and are thus involved in a potential split. Since a possible bubble merge is already done (cf. fig. 9.1), the ID -2 is free again to initialize the nodes of these bubbles for the HK algorithm with -2.

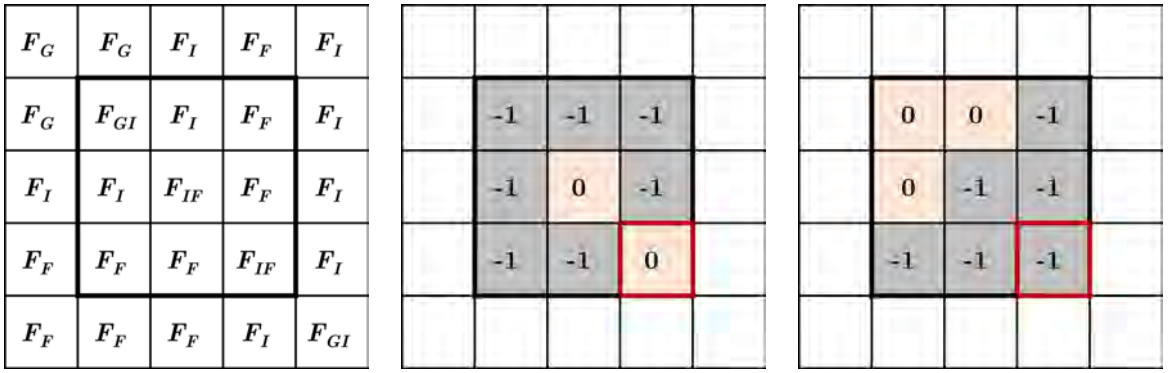
In chapter 9.2.1 it will be validated, that the presented split and merge triggers don't lead to false negatives. Furthermore, the fraction of false positives trigger events will be evaluated.

[9] But at another location the bubble could still be connected, which would lead to a false positive.
[10] Again, when determining bubble membership all nodes are interpreted as their future type.



(a) (b) (c)

Fig. 9.3: First split trigger criterion for a D2Q9 neighborhood and a test region of 3×3 grid nodes. (a) Flag field after kernel `surface_2`. (b) and (c) are the results of the HK algorithm with the central F_{IF} node being exchanged by F_I and F_F, respectively. With this constellation, the split trigger criterion is fulfilled, since $2 = n_F > n_I = 1$. This case is a false positive, because the two bubbles of (c) are actually connected, which can already be seen when looking at the 5×5 region.



(a) (b) (c)

Fig. 9.4: Second split trigger criterion for a D2Q9 neighborhood and a test region of 3×3 grid nodes. (a) Flag field after kernel `surface_2`. (b) The HK algorithm marks cluster of F_{IF} nodes. The non-center F_{IF} node (marked in red) and the central node are within the same cluster. (c) The HK algorithm marks bubbles. The node marked in red is adjacent to the border of the test region and is not neighbor to a *native bubble*. Thus, the split trigger criterion is fulfilled. Doing the same in a 5×5 region would also trigger the split, this time because the node marked in red is neighbor to a *foreign bubble*.

9.1.2.5 Further optimizations

The PCIe transfer of a GPU buffer can be optimized by assigning the `CL_MEM_USE_HOST_PTR` OpenCL flag to it and allocating a fixed corresponding CPU buffer. This is done for the buffers φ, I, flags, p_B and triggers, since a transfer of these buffers happens in every time step. Furthermore, all heavy computations on the CPU[11] except the HK algorithm are parallelized via the OpenMP API, carefully ensuring that the parallelization doesn't break determinism. With these two optimizations and when using 16 CPU cores for OpenMP, the performance of the trigger validation setup of chapter 9.2.1 measured in *mega lattice updates per second* (MLUPs) was investigated. The performance with bubble extension enabled is about half the performance with VoF extension only. This is remarkable, since using the VoF extension only does not involve any PCIe transfer or CPU computation. To further increase the performance or to prepare a multi-GPU implementation of the bubble extension, it would be possible to also parallelize the HK algorithm itself by dividing the computational domain into several sub-domains [47], which is left as an open task.

9.2 Validation

9.2.1 Triggers

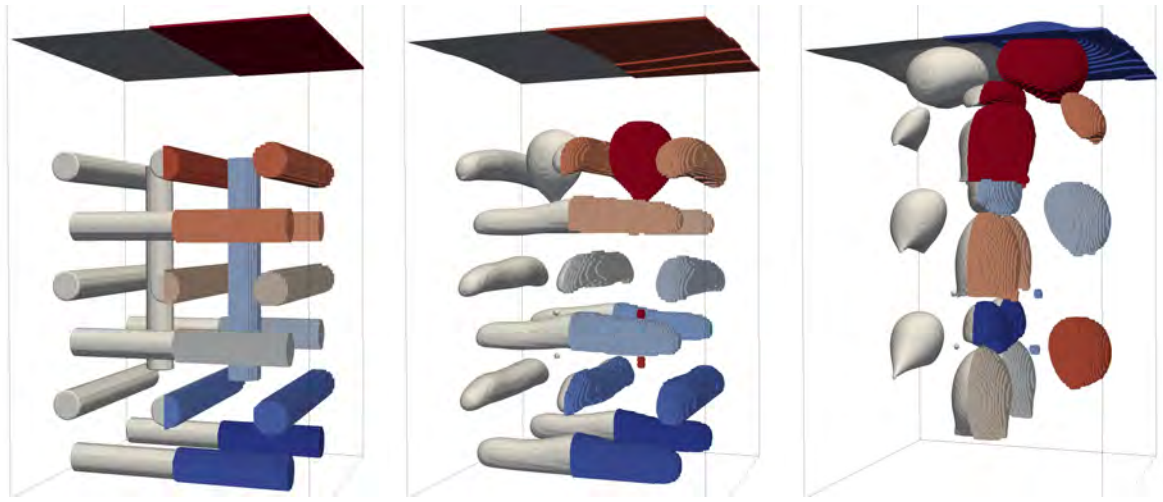
The trigger validation setup consists of several cylinder shaped bubbles pointing in different directions and an open atmosphere above the fluid phase. See fig. 9.5a for a visualization. The atmosphere in this and every future simulation is treated separately from the other bubbles: its pressure in lattice units has a constant value of $p_0 = \rho_0 c_s^2$ independent of eventual volume changes. Not doing so results in a strange wiggling of the atmosphere-fluid interface. As an external homogeneous volume force pointing in downward z-direction, i.e. gravity $F_g \hat{\boldsymbol{e}}_z$, is switched on. The density in lattice units is initialized according to the hydrostatic pressure

$$\rho(z) = \rho_0 + \frac{1}{c_s^2}(z - H/2)F_g, \tag{9.15}$$

where H is the total height of the liquid column. It holds $\rho(H/2) = \rho_0 = 1$. This initialization is necessary, since the LBM is weakly compressible. If the hydrostatic pressure is not regarded, pressure/density waves occur at the beginning of the simulation and disturbance of the bubbles is clearly visible.

After the simulation start, the bubbles rise towards the atmospheric surface and perform many merges as well as splits on their way. Looking at the number of split trigger events in tab. 9.1, one observes that using a test region of $5 \times 5 \times 5$ lattice nodes reduces the amount of false positive split trigger events by a third compared to a region of $3 \times 3 \times 3$ lattice nodes. But also the latter region size has a reasonable fraction of false positive to total split trigger events. To test, whether false negatives occur, one can proceed as follows: At every time step, an independent HK algorithm is run on the whole simulation domain, using the flag field to initialize a second ID field separate from I. The number of counted bubbles has to match with the number determined by the algorithm from chapter 9.1.2.4 at all times. This was successfully tested for a region of $3 \times 3 \times 3$ and $5 \times 5 \times 5$ lattice nodes, running 20000 time steps.

[11] E.g. computation of bubble volume, reassignment of IDs.



(a) $t = 0\Delta t$ (b) $t = 3000\Delta t$ (c) $t = 10000\Delta t$

Fig. 9.5: Trigger validation setup for different time steps. One half of the simulation domain shows the IDs of interface nodes color coded, the other half shows the position of the actual surface. Due to the limited color space, separate bubbles might have very similar color, their IDs are different though.

$t/\Delta t$	number of bubbles	region size	total split triggers	false positive split triggers
3000	21	$3 \times 3 \times 3$	102	65
		$5 \times 5 \times 5$	95	58
10000	22	$3 \times 3 \times 3$	641	478
		$5 \times 5 \times 5$	551	388
20000	17	$3 \times 3 \times 3$	4104	3088
		$5 \times 5 \times 5$	3116	2100

Tab. 9.1: Number of total and false positive split trigger events occurring during the trigger validation setup.

9.2.2 Rayleigh-Plesset

After the triggers have been evaluated, the next step is to validate the dynamics of the bubble interface. Here eq. (9.2) plays a central role. In the simplified form with constant $p_B \equiv 1/3\rho_0$, i.e. in the form of eq. (6.5), it has already been validated in [11]. For this purpose, the Plateau-Rayleigh Instability was used; furthermore, a simulated droplet impact was successfully made to match experimental data. To validate the bubble extension on top of this, the Rayleigh-Plesset equation [48] is used, which is an ordinary differential equation governing the dynamics of a spherical bubble in an infinitely extending incompressible fluid and reads

$$R\frac{\mathrm{d}^2R}{\mathrm{d}t^2} + \frac{3}{2}\left(\frac{\mathrm{d}R}{\mathrm{d}t}\right)^2 + \frac{4\nu}{R}\frac{\mathrm{d}R}{\mathrm{d}t} + \frac{2\sigma}{\rho R} + \frac{\Delta P(t)}{\rho} = 0. \tag{9.16}$$

Here, R is the radius of the bubble and ρ, σ and ν denote density, surface tension and kinematic viscosity of the surrounding fluid, respectively. Furthermore $\Delta P(t) = p_\infty(t) - p_B(t)$, where $p_B(t)$ is the pressure within the bubble and $p_\infty(t)$ is the external pressure infinitely far from the bubble.

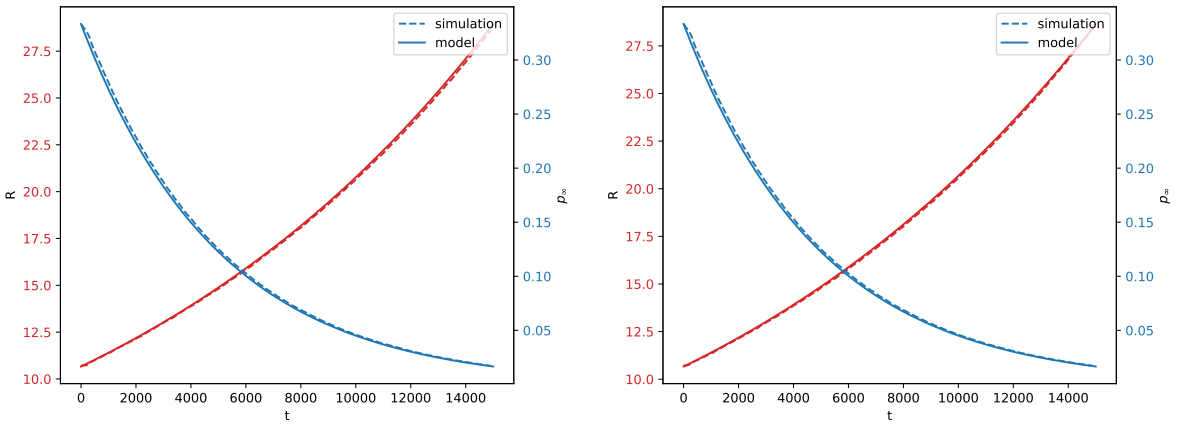
This differential equation is solved via an implicit Runge-Kutta method of the Radau IIA family of order 5 using SciPy [49, Sec. IV.8.]. Consistent to the LBM implementation, the bubble pressure p_B occurring in eq. (9.16) is computed via the ideal gas law. Besides viscosity ν and surface tension σ, fluid density ρ is also handled as a constant, which is only approximately the case for LBM due to weak compressibility.

For the LBM simulation, a cubic domain with side length $L = 256\Delta x$ is used. Nodes that lie outside a sphere with radius $R_S = 127\Delta x$ and center at $\boldsymbol{m}_S = (L/2, L/2, L/2)^\mathrm{T}$ are used as equilibrium nodes. Inside the sphere there is fluid, only in the center of the domain there is a bubble of radius $R(t=0) = L/24$. Since pressure and density are related in LBM, a density of $\rho_e(t) = p_\infty(t)/c_s^2$ is applied to the equilibrium boundaries.

First, cavitating bubble growth is simulated by setting $p_\infty(t) = p_{\infty,0}\exp(-\alpha t)$. Here, the decay rate in lattice units is chosen as $\alpha = 0.0002$. In preparation for chapter 9.2.5, the parameters given in tab. 9.2 are tested. The conversion to lattice units is given by the ratios ρ_SI/ρ, $R_\mathrm{SI}/R(t=0)$ and u_SI/u, where $\rho = 1$ holds and all other parameters are included in tab. 9.2. In all cases, 15000 simulation steps are performed. Throughout the parameter space, the simulation remains stable thanks to the thoroughly chosen tuning factor u_SI/u. The agreement between theory and simulation is very good, examples are given in fig. 9.6. A comparable simulation can be found in [50, fig. 8], who study bubble dynamics using a 2D two-phase LBM solver. At time t_2 with $R(t_2)/R(0) = 2$, their simulation result is already more than 25% above the Rayleigh-Plesset solution. The authors explain this deviation by the fact that eq. (9.16) is only valid for an infinite domain. But the real reason is probably that in their setup they apply the equilibrium boundaries to the surface of a cubic simulation domain, instead of using spherical geometry. Applying them with spherical geometry in the present thesis results in the very high accuracy shown in fig. 9.6. In the same figure it is also noticeable that the simulation results are shifted a little bit backwards in time compared to the theoretical Rayleigh-Plesset solution. This is due to the fact that for LBM the pressure p_∞ does not propagate instantaneously. Instead, the equilibrium boundaries slowly adjust the density and thus the pressure in the entire domain. Therefore, for the simulation data shown in the graph, p_∞ is computed as the mean pressure at the bubble interface (not at the equilibrium boundaries).

number	R_{SI} [m]	u_{SI}/u [m/s]	η_{SI} [Pa s]	σ_{SI} [N/m]	ρ_{SI} [kg/m^3]
1	5.0×10^{-3}	64	8.36×10^{-4}	7.2×10^{-2}	1000
2	2.5×10^{-3}	32	8.36×10^{-4}	7.2×10^{-2}	1000
3	1.5×10^{-3}	32	8.36×10^{-4}	7.2×10^{-2}	1000
4	1.0×10^{-2}	8	1.0×10^{-3}	1.0×10^{-3}	100
5	$\sqrt{10} \times 10^{-3}$	8	1.0×10^{-3}	1.0×10^{-3}	100
6	$\sqrt{10} \times 10^{-3}$	8	$\sqrt{10} \times 10^{-3}$	1.0×10^{-3}	100
7	1.0×10^{-2}	2.5	1.0×10^{-2}	1.0×10^{-3}	100
8	1.0×10^{-3}	2.5	1.0×10^{-2}	1.0×10^{-3}	100
9	1.0×10^{-2}	2.5	$\sqrt{10} \times 10^{-2}$	1.0×10^{-3}	100
10	1.0×10^{-2}	2.5	1.0×10^{-1}	1.0×10^{-3}	100
11	6.3×10^{-5}	6	8.36×10^{-4}	7.2×10^{-2}	1000

Tab. 9.2: SI-values and the conversion factor u_{SI}/u used for the cavitating bubble growth simulation and later in chapter 9.2.5.

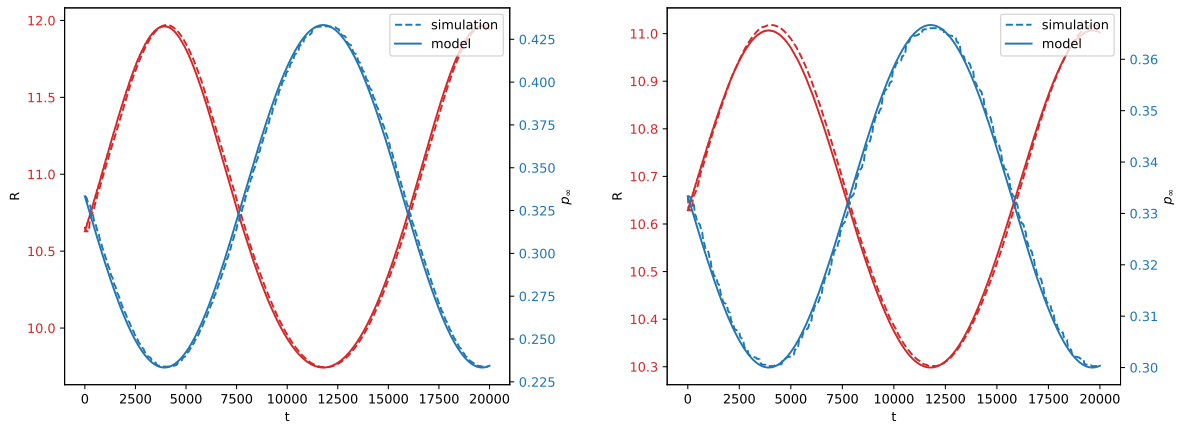


(a) Number 4

(b) Number 10

Fig. 9.6: Rayleigh-Plesset validation for exponential decay of p_∞. Solid lines show the solution of the Rayleigh-Plesset equation itself, dashed lines are simulation results. All results are shown in lattice units. The parameters used are listed in tab. 9.2.

To have a measure of the accuracy of the simulation, the pressure at infinity is now changed to $p_\infty = 1 - A \sin(\omega t)$, where in lattice units $\omega = 4 \times 10^{-4}$ is chosen. In fig. 9.7, the results of theory and simulation differ noticeably only from $A = 0.1$, where noise appears in the pressure near the bubble interface. Overall, sub-lattice accuracy is achieved.



(a) $A = 0.3$ (b) $A = 0.1$

Fig. 9.7: Rayleigh-Plesset validation for oscillating p_∞. All results are shown in lattice units. Parameter set 4 from tab. 9.2 is used. The interface moves with accuracy far below lattice resolution.

9.2.3 Cubic initialization

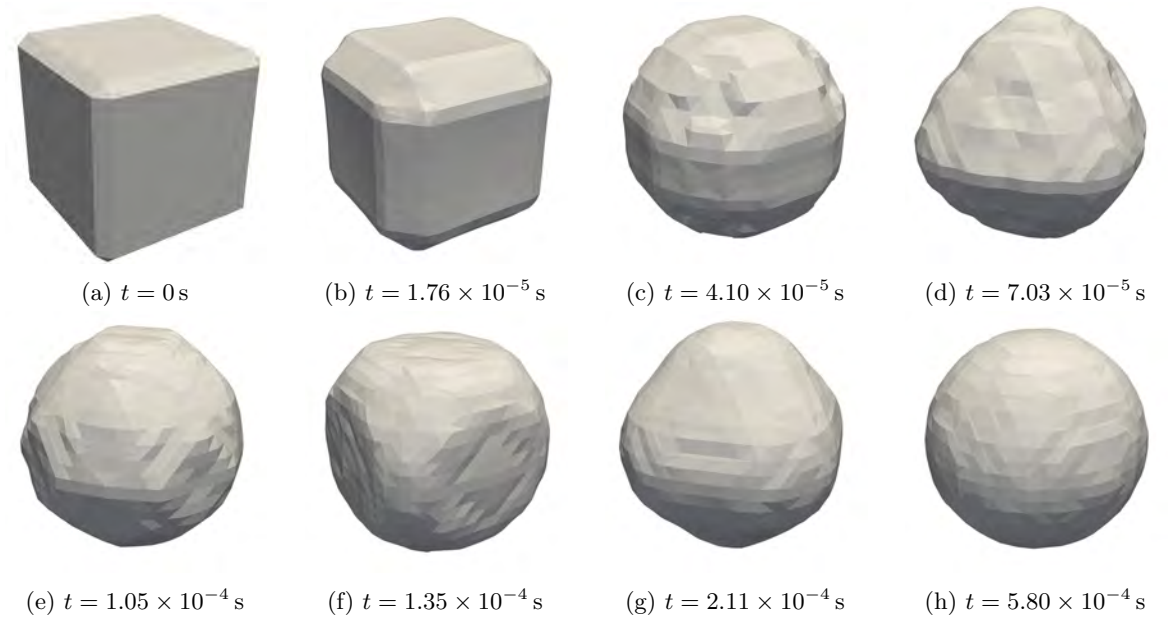
In the previous chapter, the interface was considered in pure spherical geometry. Now, the bubble is initialized in cubic shape instead, and periodic boundary conditions apply in the simulation domain. Since there are no external forces, the shape of the bubble should change to a sphere in order to minimize its surface energy. This is now done for all parameter sets from tab. 9.2, where the missing conversions ρ_{SI}/ρ and R_{SI}/R are given by $\rho = 1$ and $R(t = \infty) = 10\Delta x$. Overall, a cubic simulation domain with side length $L = 256\Delta x$ is used.

For parameter set 11 from tab. 9.2, the bubble oscillates several times between the shapes of a cube and an octahedron due to the high surface tension, but transitions to a resting sphere for long times (cf. fig. 9.8). For other parameter sets the dynamics are different, e.g. the transition from cube to sphere for parameter set 9 happens creeping due to the high viscosity. For all parameter sets, however, the bubble shape converges to a sphere.

9.2.4 Rising bubble: shape

The shape of a bubble in a liquid pool rising due to gravity is mainly determined by two dimensionless numbers. The Bond Number Bo describes the ratio of gravitational to capillary forces. The Morton Number Mo relates the viscous force to the force due to surface tension. They are defined as

$$\mathrm{Bo} = \frac{\Delta\rho g R_e^2}{\sigma}, \qquad \mathrm{Mo} = \frac{g\eta^4\Delta\rho}{\rho^2\sigma^3}, \tag{9.17}$$



(a) $t = 0\,\mathrm{s}$ (b) $t = 1.76 \times 10^{-5}\,\mathrm{s}$ (c) $t = 4.10 \times 10^{-5}\,\mathrm{s}$ (d) $t = 7.03 \times 10^{-5}\,\mathrm{s}$

(e) $t = 1.05 \times 10^{-4}\,\mathrm{s}$ (f) $t = 1.35 \times 10^{-4}\,\mathrm{s}$ (g) $t = 2.11 \times 10^{-4}\,\mathrm{s}$ (h) $t = 5.80 \times 10^{-4}\,\mathrm{s}$

Fig. 9.8: Bubble dynamics after cubic initialization. The parameter set 11 from tab. 9.2 is used. The bubble shape oscillates between cube and octahedron, the first two oscillations are shown in (a)-(g). Finally, the bubble converges to a sphere (h).

where R_e denotes the radius of the equivalent sphere of a bubble of arbitrary shape and g is the gravitational acceleration. Furthermore, $\Delta\rho$ is the density difference between fluid and gas phase, where in this context $\Delta\rho = \rho$ holds. [51] have summarized the experimentally occurring bubble shapes by means of these two dimensionless numbers in a shape regime map, which is depicted in fig. 9.9. With their LBM extended by a Cahn-Hilliard diffuse interface approach, [52] were able to successfully reproduce correct shapes for some combinations of $\mathrm{Bo} \in [1, 1000]$ and $\mathrm{Mo} \in [10^{-5}, 10^{4}]$.

In this thesis, the shape regime map is investigated for the parameters listed in table 9.3. For the setup a domain of size $L_x \times L_y \times L_z = L \times L \times 1.6L$ with $L = 256\Delta x$ is used. The gravitational acceleration acts in negative z-direction. Above height $H_a = 0.8L_z$ the atmosphere begins, below which the fluid at rest is initialized with hydrostatic pressure. An initially quiescent and spherical bubble with radius $R = L/6$ and center at $(L_x/2, L_y/2, R + 5\Delta x)^{\mathrm{T}}$ rises for $t > 0$ under the action of gravity, often assuming a stable shape, which is depicted in fig. 9.10.

The parameter sets 1, 2 and 3 represent small air bubbles dissolved in a fluid with water-like parameters. The boundary between wobbling and spherical shape is very well visible. Bubbles with $R < 10^{-3}\,\mathrm{m}$ "get stuck", i.e. do not rise during the simulation, and are therefore classified as faulty. This is also true for parameter set 11. The shape of parameter set 4 arises because the interface of the bubble bottom pushes the slower interface of the bubble top ahead of it, which is considered unphysical. The shape shown does not remain stable until the end of the simulation, but breaks up into several small bubbles. Similar phenomena were observed also for smaller Mo, so the whole spherical cap regime could not be reproduced. Parameter set 5 forms an annular bubble whose circumference grows until it finally decays into several bubbles. Parameter sets 7 to 10, on the other hand, are shape stable and agree well with the shape regime map. If the radius of parameter set 8 is reduced and thus Bo, the phenomenon of "stuck drops" appears

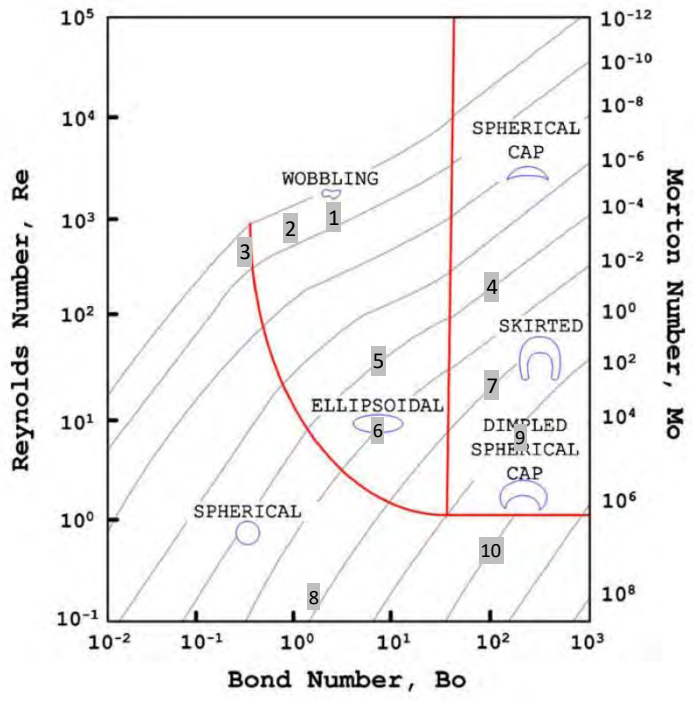
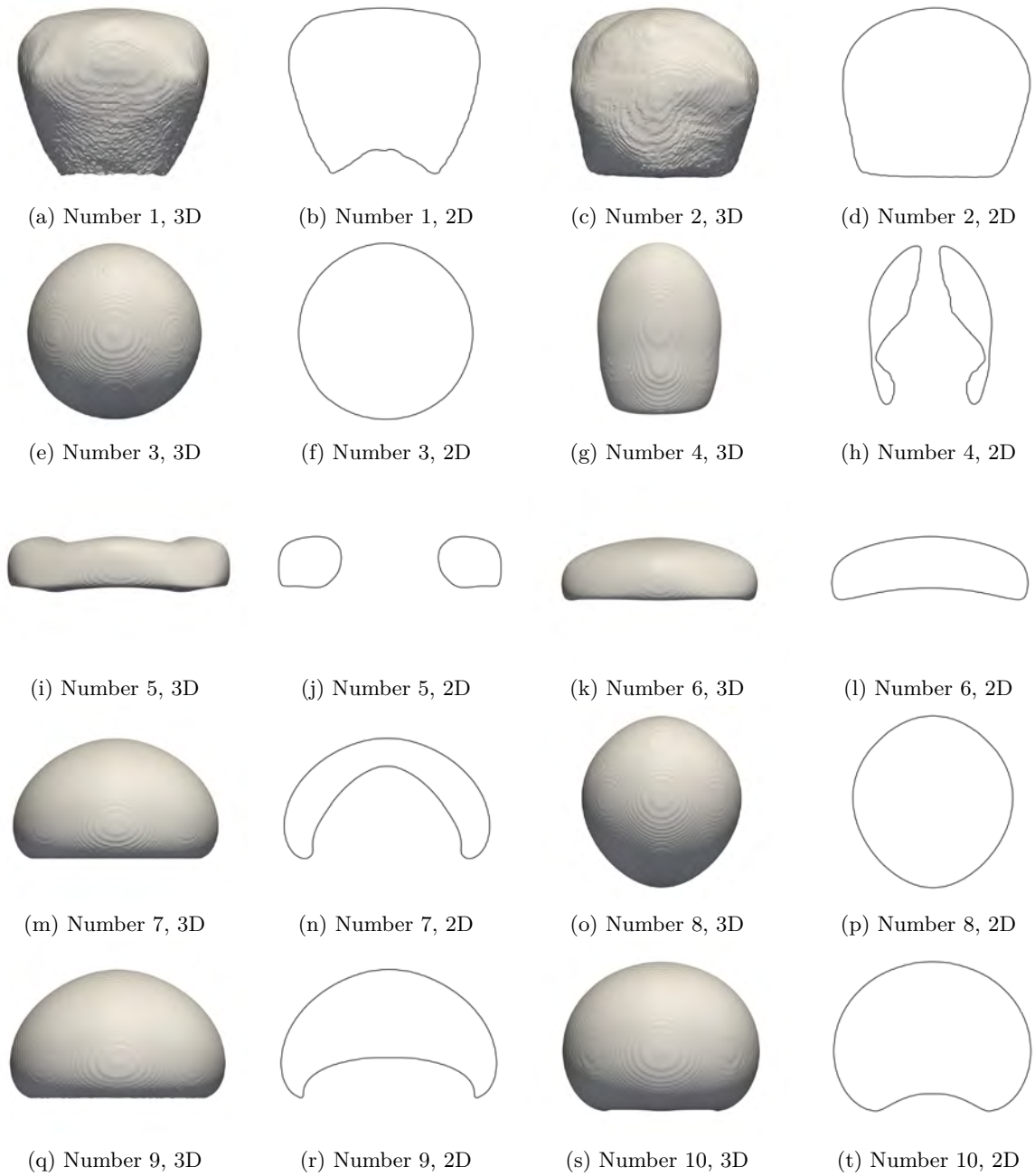


Fig. 9.9: Shape regime map, taken from [52, fig. 1]. The numbers in gray refer to the numbers from tab. 9.3.

number	Mo	Bo	g_{SI} [m/s^2]
1	1.31×10^{-11}	3.47	10
2	1.31×10^{-11}	0.868	10
3	1.31×10^{-11}	0.313	10
4	10^{-4}	100	10
5	10^{-4}	10	10
6	10^{-2}	10	10
7	1	100	10
8	1	1	10
9	200	200	20
10	10^4	100	10

Tab. 9.3: Parameters used for validation via shape of rising bubble. Additionally, all parameters from tab. 9.2 are used.



(a) Number 1, 3D (b) Number 1, 2D (c) Number 2, 3D (d) Number 2, 2D

(e) Number 3, 3D (f) Number 3, 2D (g) Number 4, 3D (h) Number 4, 2D

(i) Number 5, 3D (j) Number 5, 2D (k) Number 6, 3D (l) Number 6, 2D

(m) Number 7, 3D (n) Number 7, 2D (o) Number 8, 3D (p) Number 8, 2D

(q) Number 9, 3D (r) Number 9, 2D (s) Number 10, 3D (t) Number 10, 2D

Fig. 9.10: Shapes of rising bubble for the parameters from tab. 9.3. Alternating, the 3D interface and the 2D intersection at $x = L_x/2$ is shown. The shapes match quite good with the shape regime map fig. 9.9.

again. To be able to simulate the parameter sets 3 and 8, the conversion factor u_{SI}/u has to be adjusted very carefully: if it is chosen too small, the whole simulation becomes unstable due to the high surface tension, but if it is too large, the bubble does not rise.

The phenomenon of stuck drops in simulations with $\mathrm{Bo} \lesssim 1$ was further investigated. These occur in a region where capillary forces dominate gravitational force. For testing, the bubble extension was turned off and the initial conditions were changed to simulate an initially resting droplet falling into a liquid pool under the action of gravity. This droplet also "gets stuck", i.e. does not fall, for the same parameter sets as for the rising bubble simulations. In principle, however, this parameter range should be accessible with the methods used here: For the VoF method, no limitation to this parameter range is known in the literature. [53] combine the LBM with the VoF method and simulate bubbles using the ideal gas law. In [53, fig. 7.4], they simulate a bubble with $R = 0.6\,\mathrm{mm}$ and do not mention any principal lower bound. Suspecting a numerical error due to round off errors, the VoF algorithm of FluidX3D was changed from `float` to `double` by Moritz Lehmann for testing purposes, without showing any improvement. Solving this problem remains as an open task.

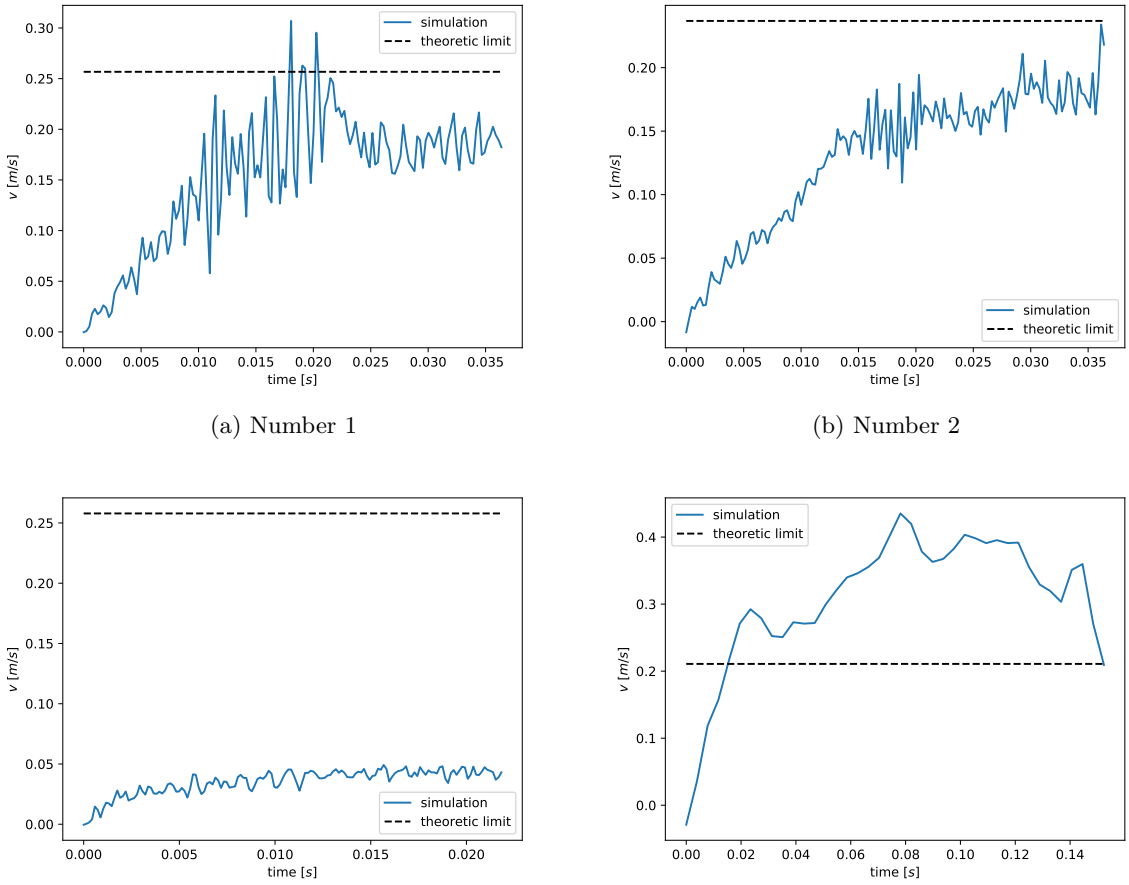
9.2.5 Rising bubble: speed

In addition to the bubble shape, the terminal rising velocity v_t of bubbles can provide information on how well the simulation can reproduce reality. Depending on which of the three regimes marked in red in fig. 9.9 is entered, other forces play a role. In the spherical regime (lower left region), surface tension and viscous forces dominate over inertial forces and low Reynolds numbers are typical. In the ellipsoidal regime (middle region), inertial forces are no longer negligible, but still surface tension plays an important role. Finally, the spherical cap regime (upper right region) is mostly governed by inertial force. For the spherical, the ellipsoidal and the spherical cap regime thus different approximate terminal velocities $v_{t,\mathrm{sph}}$, $v_{t,\mathrm{ell}}$ and $v_{t,\mathrm{cap}}$ apply, respectively. Using the equivalent diameter $d_e = R_e/2$, they are described by [52, 54–57]

$$v_{t,\mathrm{sph}} = \frac{g d_e^2 \rho}{12\eta}, \qquad v_{t,\mathrm{ell}} = \sqrt{\frac{2.14\sigma}{\rho d_e} + 0.505 g d_e}, \qquad v_{t,\mathrm{cap}} = \frac{2}{3}\sqrt{\frac{g d_e}{2}}. \tag{9.18}$$

For all parameter sets from tab. 9.3 a simulation is performed, where every 100 steps the z position of the center of the bubble $z_M(t)$ is calculated. The simulation is terminated if one surface point of the bubble is less than $2R$ nodes away from the atmosphere, but at the latest after 15000 steps. Using central differences, the rising velocity $v_z(t)$ is determined from $z_M(t)$. This velocity is shown in fig. 9.11 and fig. 9.12 together with the terminal velocity v_t valid for the respective region. To give the bubble more time to converge to a constant velocity as it rises, some parameters from chapter are adjusted: it now holds $H_a = 0.9L_z$, $L_z = 3.0L$, and $R = L/8$. This also reduces the self-interaction of the bubble via its periodic image and via pressure waves reflected at the atmospheric interface. However, due to memory limitations a further increase of L/R is only possible by a degradation of the resolution, which will not be pursued further here. Nevertheless, some important tendencies can already be read from the available data. For all simulations in the spherical regime or close to it (number 3, 8 and 10) the simulated velocity $v_z(t)$ is significantly smaller than the theoretical terminal rising velocity v_t. A connection to the phenomenon of stuck drops, which also occurred in this regime, is likely. In the ellipsoidal regime the correct order of magnitude of the rising velocity is reached, but it is systematically

underestimated in the simulation. Taking the ratio of simulated over theoretical terminal rising velocity and averaging over the simulations number 1, 2, 5 and 6 gives $v_z/v_t \approx 0.73$. The self interaction via the atmospheric interface might still cause this slowdown. In the spherical cap regime, however, the terminal rising velocities are well predicted, only simulation number 4 is an exception due to its unphysical interface dynamics as discussed in the preceding chapter.



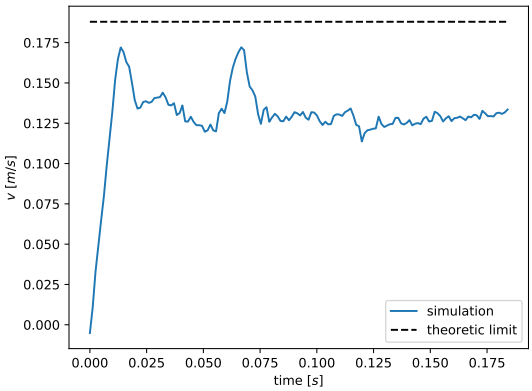
(a) Number 1

(b) Number 2

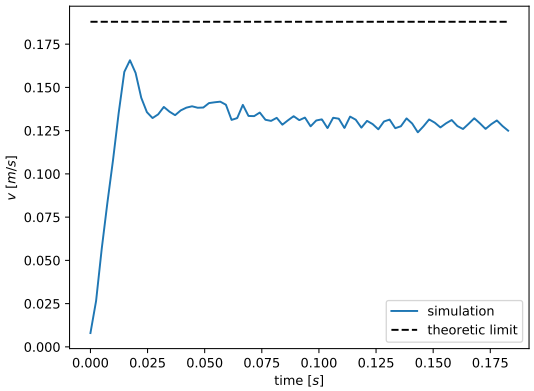
(c) Number 3

(d) Number 4

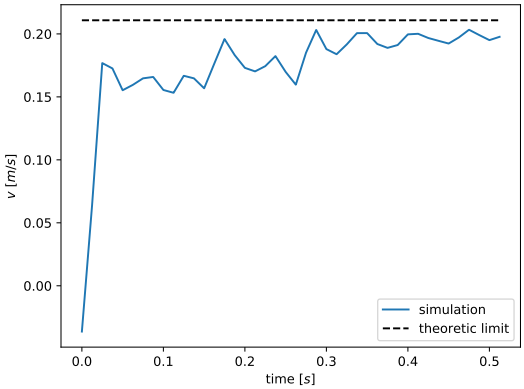
Fig. 9.11: Rising velocity of an initially spherical bubble at rest. The setup numbers 1-4 refer to tab. 9.3. As the rising velocity eventually reaches a steady-state, is should match the theoretical terminal velocity from eq. (9.18).



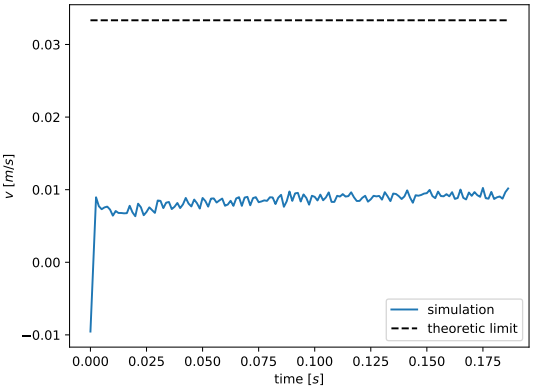
(a) Number 5



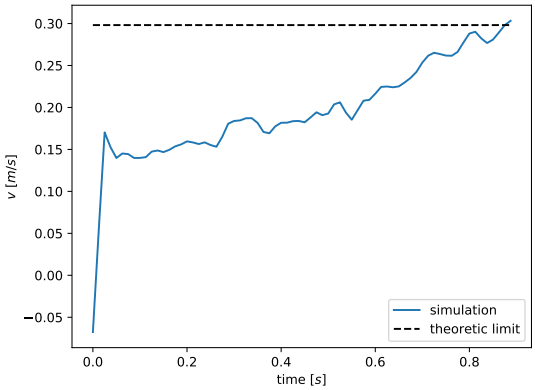
(b) Number 6



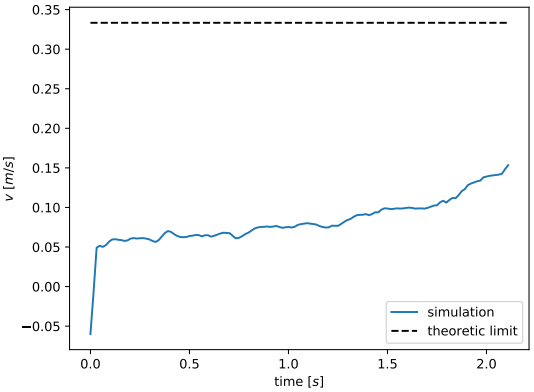
(c) Number 7



(d) Number 8



(e) Number 9

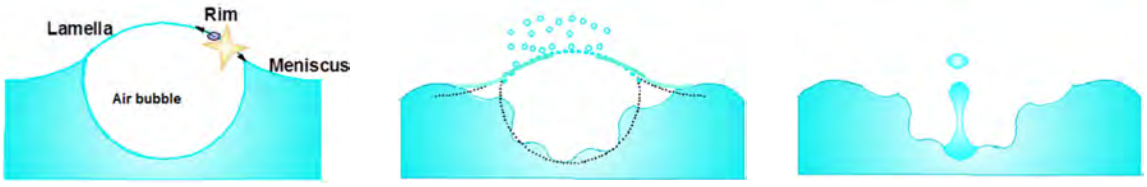


(f) Number 10

Fig. 9.12: Like fig. 9.11, but for setup numbers 5-10.

9.3 Limitations - bursting bubble

When a bubble approaches the liquid-atmosphere free interface, a very thin fluid film forms between the bubble and the atmosphere, which is called *lamella*. The actual burst is triggered by a rupture of the lamella. Two main processes cause droplets to be ejected into the air: Firstly, the lamella gets unstable and bursts into droplets. Secondly, as the bubble cavity collapses, a water jet is ejected into the air and breaks into droplets. This process is visualized in fig. 9.13. It is of special interest for microplastics research, since these droplets could play a major role at the exchange of microplastics particles at the air-water interface.



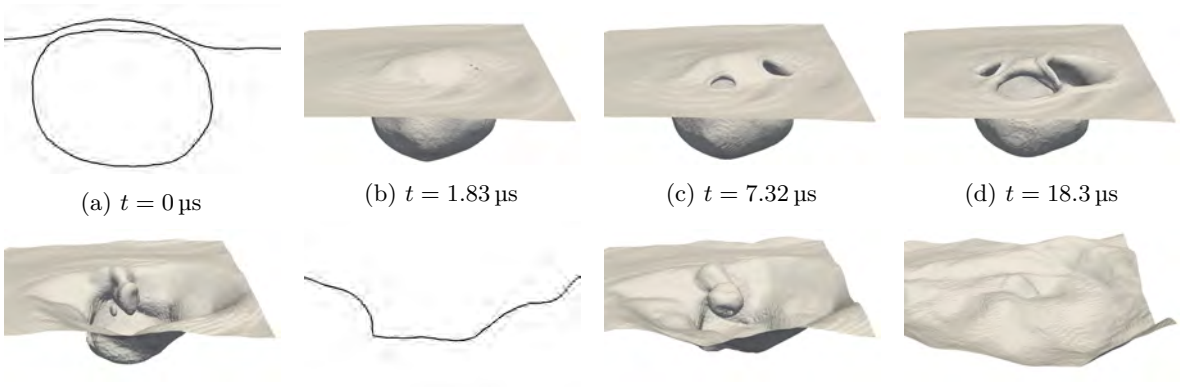
(a) (b) (c)

Fig. 9.13: Schematic of a bursting bubble. (a) When the lamella ruptures, it retreats towards the meniscus and a rim forms at its edge. (b) The rim becomes unstable, tiny droplets are ejected into the air. As the bubble's pressure is released, oscillations at the atmosphere-water interface are triggered. (c) The bubble cavity collapses, which forms a water jet. This jet becomes unstable and breaks into droplets. Images taken from [58].

With the current bubble implementation, a physically correct simulation of a bursting bubble is not possible. This is mainly due to the time and length scales involved. The typical equivalent diameter of the bubbles in question is in the range $d_e = 10^{-4} - 10^{-2}\,\mathrm{m}$. Popular instability models require a film thickness less than 10 nm [58]. [59] estimate the film mean thickness prior to the rupture to be between 0.3 and 0.9 µm. Furthermore, they determine the films rim velocity to be in the order of 10 m/s. In the current VoF implementation, a film of minimal thickness consists of three lattice nodes (two interface nodes separated by one fluid node). For a bubble of size $d_e = 2\,\mathrm{mm}$ in a cubic simulation domain with side length $L = 5d_e$, in order to resolve a critical thickness of 0.1µm a total of $(3 \times \frac{10\,\mathrm{mm}}{0.1\,\mu\mathrm{m}})^3 = (3 \times 10^5)^3 = 2.7 \times 10^{16}$ lattice nodes would be needed. For a single AMD Radeon VII GPU with 16 GB of internal memory, only about 4.5×10^8 lattice nodes fit into memory when using the VoF with `float` accuracy. Since this memory discrepancy cannot be overcome by a multi-GPU approach, instead the VoF would need to be changed. In particular, the limitation that every interface node must be neighbor to both a fluid and a gas node must be lifted in such a way, that interface nodes of arbitrary fill level between two regions of gas may be simulated. This would mean, that a slice through a fluid film of minimal thickness consists of the nodes $(F_G|F_I|F_G)$ instead of the nodes $(F_G|F_I|F_F|F_I|F_G)$. Thus, whole fluid films could be simulated with sub-grid resolution. In order to achieve this, the VoF algorithm including the curvature calculation and the HK algorithm would need to be adapted, which is beyond the scope of this work.

To illustrate the current limitations of the bubble extension, fig. 9.14 shows a bubble burst for parameter set 2 from tab. 9.3, with only u_{SI}/u changed to 32 m/s. The size of the domain is $L \times L \times L_z$, where it holds $L_z = 2L$, $L = 3d_e$ and $L = 256\Delta x$. Thus, the length of a lattice node in SI-units is $\frac{15\,\mathrm{mm}}{256} \approx 0.18\,\mathrm{mm}$, so the expected thickness of the lamella can by far not be resolved. Furthermore, $\rho = 1$ is chosen in lattice units. At time $t < 0\,\mathrm{s}$ the bubble is spherically initialized and rises to the atmospheric interface, at time $t = 0\,\mathrm{s}$ the simulation is one simulation

step away from the lamella rupturing. At frame 9.14d, the largest of the holes in the lamella has a radius of approximately 1 mm. Thus, the rim velocity is estimated to be $1\,\mathrm{mm}/18\,\mu\mathrm{s} \approx 56\,\mathrm{m/s}$, which is about the right order of magnitude. During the entire simulation, only a single droplet detaches from the lamella. No liquid jet forms. One reason for this could be that for $t > 30\,\mu\mathrm{s}$ the cavity interacts with itself via the periodic boundary conditions. The choice of a larger simulation domain could remedy this.



(a) $t = 0\,\mu\mathrm{s}$ (b) $t = 1.83\,\mu\mathrm{s}$ (c) $t = 7.32\,\mu\mathrm{s}$ (d) $t = 18.3\,\mu\mathrm{s}$

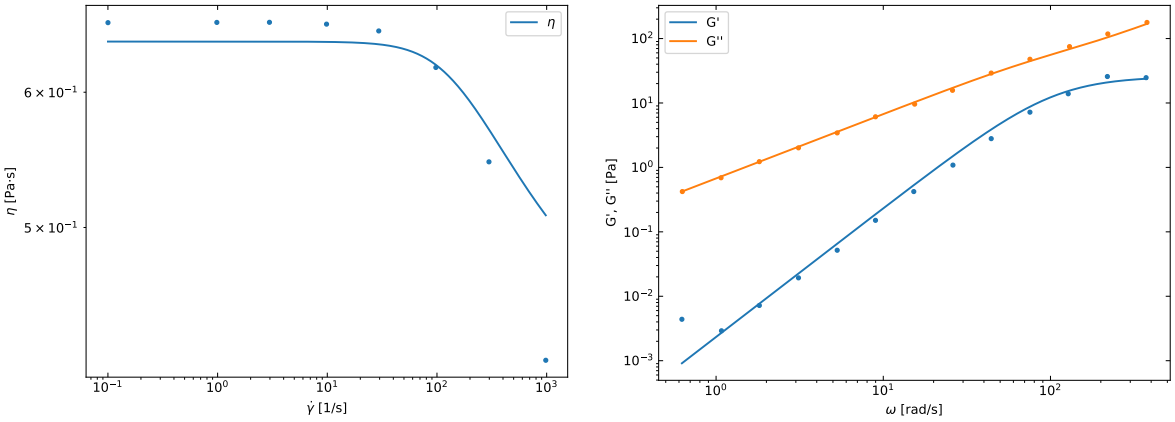
(e) $t = 29.3\,\mu\mathrm{s}$ (f) $t = 62.3\,\mu\mathrm{s}$ (g) $t = 62.3\,\mu\mathrm{s}$ (h) $t = 102.5\,\mu\mathrm{s}$

Fig. 9.14: Bursting of a bubble of size $d_e = 5\,\mathrm{mm}$ using water parameters (parameter set 2 from tab. 9.3). 2D slices as well as 3D side views are shown. Directly before rupture of the interface, the lamella has a critical thickness of more than 0.18 mm due to the restricted resolution. The rim velocity is about 56 m/s. No liquid jet forms after breakdown of the cavity.

10 Viscoelasticity

10.1 Fits to rheological data

In the following chapters, the implementation of the Oldroyd-B and FENE-P models will be described and validated. But first, the question of how well these models can describe real data must be answered. [60] work with dilute polymer solutions. They are prepared from powders of polyacrylamide and dissolved in distilled water (HPAM), resulting in a concentration of 11% w/w (weight percentage concentrations of the solvent in the solution). They explain their rheological data with the Oldroyd-B model. Since the dilute polymers have only a weak shear-thinning behavior, this yields a good description. In fig. 10.1 the rheological functions of η, G' and G'' are simultaneously fitted to the data taken from their fig. 3. The FENE-P model is chosen for the fit, i.e., eq. (4.51) and (4.56) are used with $\epsilon = 0$ and the free parameters G, λ_p, b and η_s. It gets obvious that the shear-thinning behavior and the elastic properties of HPAM 11% can be captured well within a single model.

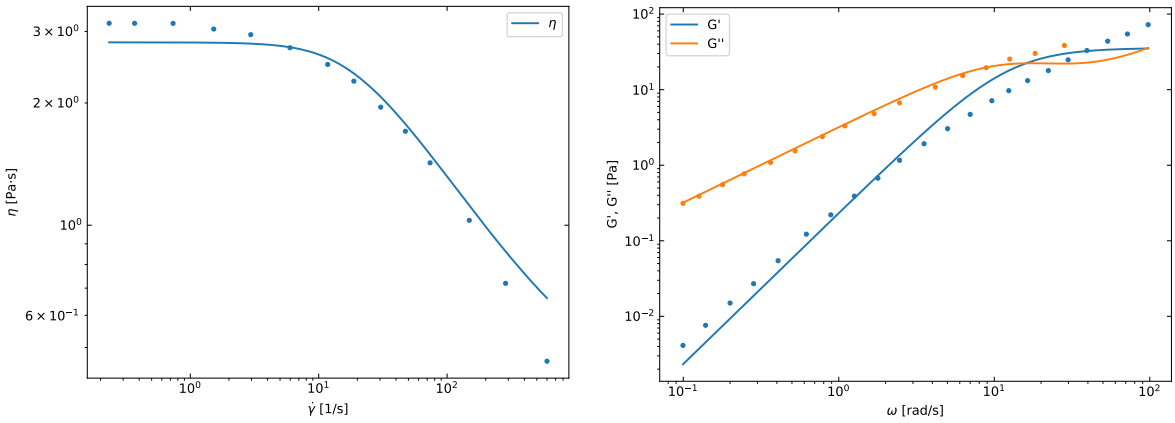


(a) (b)

Fig. 10.1: Fit of FENE-P simultaneously to shear-thinning and moduli data from [60, fig. 3], HPAM 11%. The parameters determined by the fit are listed in tab. 10.1.

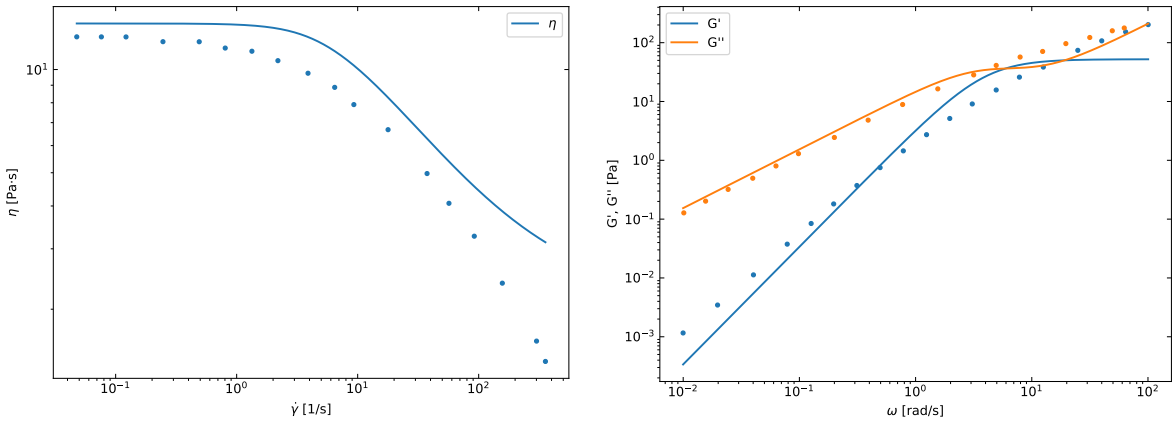
Next, literature is taken into account that provides data of polymer classes also used for bioprinting. [61] studies the rheological properties of sodium alginate in solution. Alginates, salts of alginic acid, are linear long-chain polysaccarides. With their high biocompatibility and non-toxicity, they are widely used as biomaterials in applications such as tissue engineering. For bioprinting, polymer solutions with rather high concentrations are of interest, so in fig. 10.2 and 10.3 alignates with 2.1 g/dL and 3.0 g/dL are shown, respectively. The data is taken from [61, fig. 2 & 4]. Again the rheological functions η, G' and G'' of the FENE-P model are simultaneously fit to the data. The elastic behavior is captured well, only for high ω the model starts to deviate more from the data. Concerning the shear-thinning data, the FENE-P model can describe the

alginate with 2.1 g/dL better than the ones with 3.0 g/dL. The data set with 1.0 g/dL yields an even better description (not shown), so all in all it is found that for increasing concentration it becomes more and more difficult to describe the experimental data with the FENE-P model.



(a) (b)

Fig. 10.2: Fit of FENE-P simultaneously to shear-thinning and moduli data from [61], fig. 2 und 4, 2.1 g/dL. The parameters determined by the fit are listed in tab. 10.1.



(a) (b)

Fig. 10.3: Like fig. 10.2, but for a polymer concentration of 3.0 g/dL.

Finally, the rheobase of the *Sonderforschungsbereich TRR 225 Biofabrication* [62] is inspected and two representative polymers are selected, namely alginate 4 g/dL and Poly(2-oxazoline)s (POx) 25% w/w [12]. POx are accessible via living cationic ring-opening of 2-oxazolines. They are known for their biocompatibility, high modulation of solubility and chemical functionality and are also used in other contexts, e.g. drug delivery [63]. Shear-thinning and elasticity data of both alginate and POx is shown in fig. 10.4b and fig. 10.5, where the fits were made with the same procedure as above. In fig. 10.4b, the gray region has been excluded from the fit. In the rheobase, the zero shear viscosity of POx ($\approx 10^4$) is generally orders of magnitude greater than

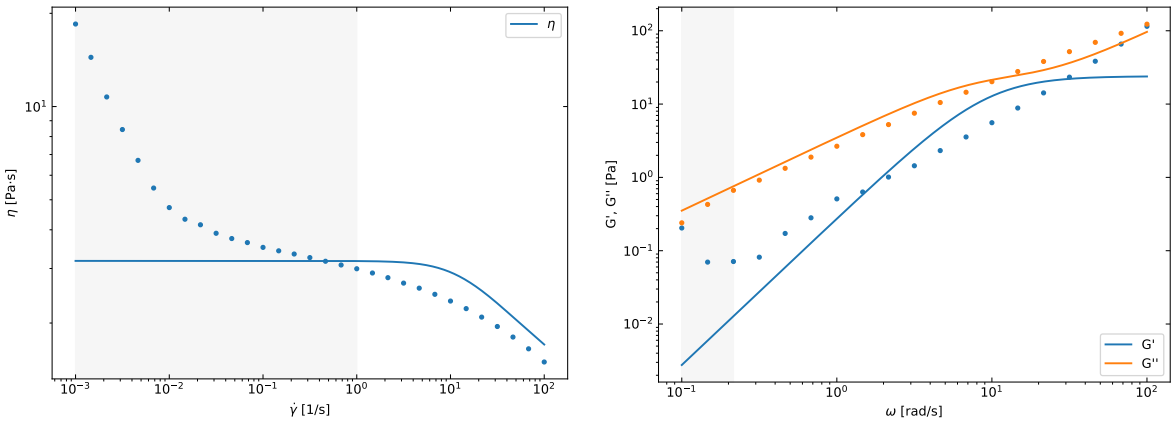
[12]The exact names as occurring in the rheobase are alginate_PH176_4wv_1.2 and POx_25ww_2.

the one of alginates ($\approx 10^1$).

Concerning the shear-thinning, in the data set alginates often exhibit multiple "modes" (see the flanks with different slopes in fig. 10.4a). Since the unextended FENE-P model consists of only one mode (it only has one polymer relaxation time), only one of the slopes can be captured by the fit. However, the slope shows good compatibility with the fixed power-law with exponent 1/3 of the FENE-P model - the isolated fit to the viscosity data only has very good agreement (not shown). For POx, although there is only one "mode", even an isolated fit to the viscosity data could not capture the shear-thinning behavior. The Carreau-Yasuda model

$$\eta = \frac{\eta_0}{(1 + (\dot{\gamma}/\dot{\gamma}_0)^\alpha)} \tag{10.1}$$

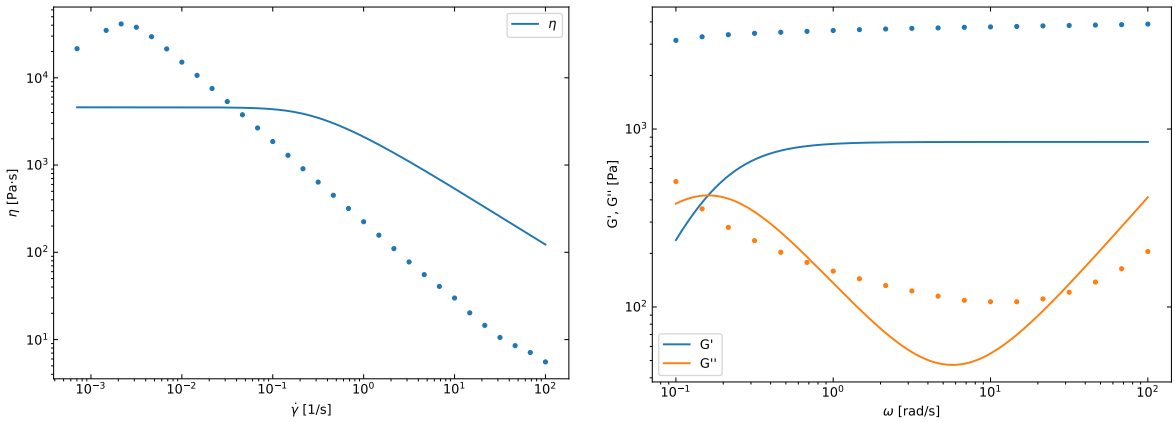
is used to describe shear-thinning fluids [64]. When this model is fitted to the shear-thinning data with α, η_0 and $\dot{\gamma}_0$ as free parameters, the power law corresponding to POx can be determined. Obviously, in the limit of large shear rates ($\dot{\gamma} \gg \dot{\gamma}_0$) it holds $\alpha \to 1 - n$, with the power law coefficient n from eq. (4.54). While this yields a value close to $n = 1/3$ for the previous datasets (e.g., for alginate 3.0 g/dL from [61]: $n = 0.28 \pm 0.01$) and thus is well compatible with FENE-P, for POx it yields $n = 0.085 \pm 0.018$. This explains why the fit to the viscosity data is poor, even if they are fitted isolated from the elasticity data. Furthermore, the FENE-P model can capture the elastic properties of alginates much better than the one of POx. The former has a strong frequency dependence, while for the latter G' is nearly constant over the whole frequency domain.



(a) (b)

Fig. 10.4: Fit of FENE-P simultaneously to shear-thinning and moduli data of alginate 4 g/dL from the rheobase [62]. The parameters determined by the fit are listed in tab. 10.1.

In summary, in the context of bioprinting, the FENE-P model is preferable to the Oldroyd-B model because it can describe the shear-thinning property in addition to the viscoelastic properties of the polymers used. The FENE-P model thus contains the two most important properties of the polymers to be modeled, without losing its strong relation to microscopic theories. Nevertheless, the FENE-P model can only represent the variety of polymer types used in bioprinting to a limited extent. However, it provides a good starting point from which the description could be further refined. For example, it would be possible to extend the single-mode FENE-P implementation of this work to a multi-mode FENE-P [65] implementation using a spectrum of N relaxation



(a) (b)

Fig. 10.5: Like fig. 10.4, but for POx 25% w/w.

solution	G [Pa]	λ_p [s]	b	η_s [Pa s]	η_p [Pa s]
HPAM11%	26 ± 6	0.015 ± 0.017	5 ± 16	0.44 ± 0.06	0.39 ± 0.44
Alginate 2.1 g/dL	36 ± 4	0.12 ± 0.03	6 ± 4	0.39 ± 0.16	4.3 ± 1.2
Alginate 3.0 g/dL	52 ± 9	0.33 ± 0.08	10 ± 10	2.1 ± 0.5	17 ± 5
Alginate 4 g/dL	24 ± 5	0.14 ± 0.05	10 ± 14	0.9 ± 0.2	3.3 ± 1.4
POx 25% w/w	850 ± 260	8 ± 5	10 ± 17	4.1 ± 2.8	6900 ± 5000

Tab. 10.1: Parameters obtained when fitting the FENE-P model to data from fig. 10.1 - 10.5.

times. Looking at the number of slopes present in the shear-dependent viscosity measurements of the rheobase, $N = 2$ would already be sufficient to describe most of the data. In order to widen the limitation of the fixed power law with exponent 1/3, the spring law of FENE-P could be adjusted phenomenologically. However, since in this thesis only the most important properties of bioprinting polymers are to be captured and in order to be able to make use of the analytical solutions to numerous flow geometries during the validation, this work is limited to the unextended versions of Oldroyd-B and FENE-P.

10.2 Algorithm overview

In this section, an overview of the simulation program will be given, the individual parts of which will be explained in more detail in the following chapters. The most complex case of the simulation of a Newtonian capsule in viscoelastic flow is shown in form of a flow chart in fig. 10.6. The names of the GPU kernels as they appear in FluidX3D are given in parentheses. After initializing the fields (step 0) according to the desired initial and boundary conditions, the kernel `stream_collide` (step 1) computes the values of ρ, u and f_i of the respective next iteration as described in chapter 5. The external force caused by viscoelasticity is also included. Afterwards, the stress tensor $\underline{\tau}_p$ is updated according to the constitutive equation (step 2, see chapters 10.3 and 10.5). These two steps are already sufficient to simulate purely viscoelastic fluids. In step 4 the capsule in the fluid is taken into account (see chapters 7 and 10.7). Finally, steps 3, 5, 6 and 7 are necessary to make the interior fluid of the capsule Newtonian (see chapters 7.3 and 10.8.3).

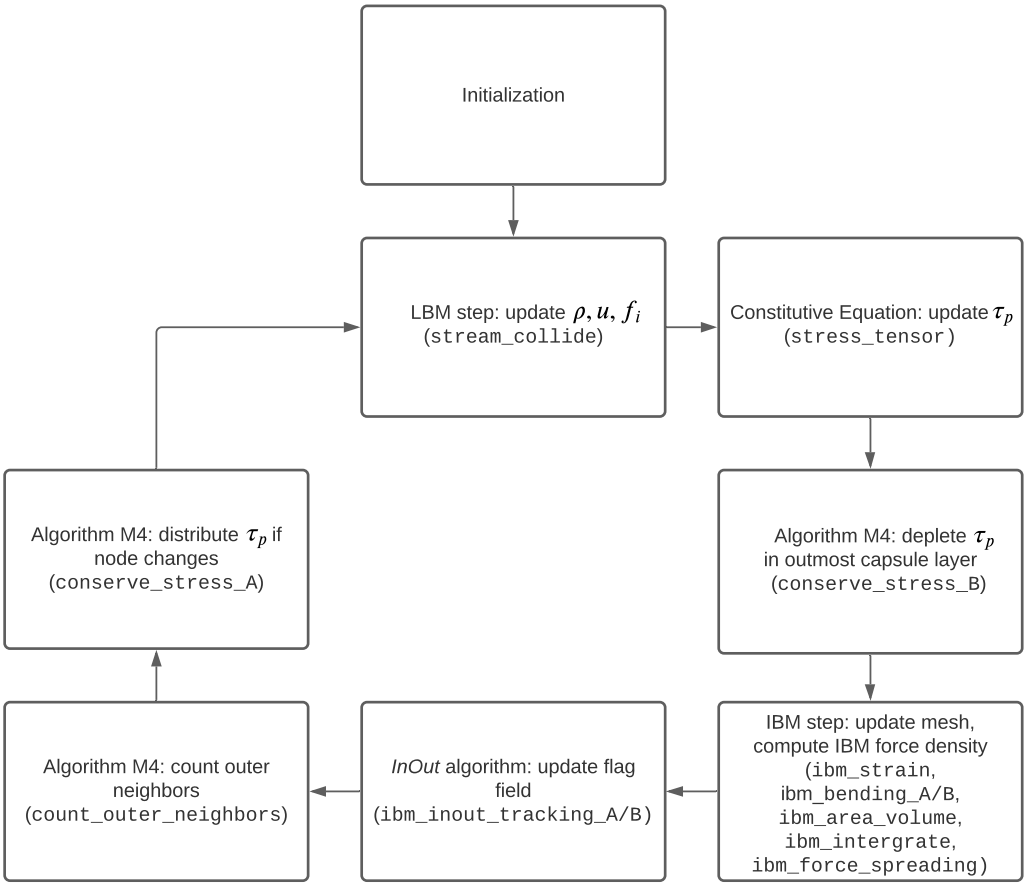


Fig. 10.6: Overview of the program cycle when simulating a Newtonian capsule in viscoelastic flow.

10.3 Implementation idea Oldroyd-B

Different approaches are conceivable in the numerical treatment of the Oldroyd-B model. [16] and [66] solve the time evolution of the dumbbell probability density function $\psi(\boldsymbol{q}, t)$ and recover the macroscopic polymer stress at each lattice node. But it is also possible to directly use the constitutive equation, which is preferred in this thesis. In the polymer formulation of Oldroyd-B according to eq. (4.10) and (4.11) the total stress $\underline{\sigma}$ is split into a Newtonian part $\underline{\sigma}_s$ and a polymer part $\underline{\tau}_p$. Since in the associated constitutive equation only the latter part appears, this allows the whole flow problem to be solved by two coupled solvers. $\underline{\sigma}_s$ can be solved by the standard LBM, while the FVM is used to update $\underline{\tau}_p$ via the constitutive equation. The coupling LBM $\rightarrow$ FVM happens via the velocity field. For the reverse direction LBM $\leftarrow$ FVM two options come up. On the one hand, the polymer stress could be added directly to the Newtonian stress in moment space. This has however the disadvantage that the implementation is compatible with MRT only, since it is the only collision operator where the moment space is directly available during transformation. Furthermore, the matrix $\underline{M}^{-1}\underline{S}\,\underline{M}$ from eq. (5.5) can no longer be precomputed (since it is no longer constant during simulation), which comes with performance drawbacks. The second option is to let the polymer stress enter as an external force (see below). For this thesis the latter option is chosen, even though according to [23, 67] the former option should bring advantages in terms of stability.

Next, the constitutive equation must be solved. For this task the FVM is not the only possibility. E.g. [68] couple their D3Q19-LBM-NS solver with a D3Q7-LBM solver for viscoelasticity (the LBM is not restricted to the NSE in general). But since they need a separate set of distribution functions for each component of the stress tensor, their memory consumption is at least 7 times larger than that of the FVM implementation of this thesis. [23] use an explicit second-order central finite-difference scheme in space and a second-order Adams-Bashforth method for temporal evolution. Both alternatives introduce additional diffusive terms to increase stability. In this thesis, a solution without diffusive terms using the FVM analogous to [69] is presented. However, to increase stability, later an alternative advection algorithm will be used (see below) that intrinsically incorporates artificial diffusion.

Reformulation of the basic equations: First one can make use of the fact that

$$u_k(\boldsymbol{r},t)\frac{\partial}{\partial r_k}\tau_{ij}(\boldsymbol{r},t) = \frac{\partial}{\partial r_k}\left(u_k(\boldsymbol{r},t)\tau_{ij}(\boldsymbol{r},t)\right) - \tau_{ij}(\boldsymbol{r},t)\underbrace{\frac{\partial}{\partial r_k}u_k(\boldsymbol{r},t)}_{=0 \text{ per eq. (3.8)}} =$$

$$= \frac{\partial}{\partial r_k}\left(u_k(\boldsymbol{r},t)\tau_{ij}(\boldsymbol{r},t)\right) \equiv \frac{\partial}{\partial r_k}J_{ijk}(\boldsymbol{r},t). \tag{10.2}$$

With this definition of the flux term $J_{ijk}(\boldsymbol{r},t)$, the constitutive equation (4.11) is brought to the form of a conservation law:

$$\frac{\partial}{\partial t}\tau_{ij}(\boldsymbol{r},t) = -\frac{\partial}{\partial r_k}J_{ijk}(\boldsymbol{r},t) + S_{ij}(\boldsymbol{r},t), \tag{10.3}$$

where all remaining terms are interpreted as source terms (the $(\boldsymbol{r},t)$ dependence is suppressed here):

$$S_{ij} = \tau_{ik}\frac{\partial}{\partial r_k}u_j + \tau_{kj}\frac{\partial}{\partial r_k}u_i + \frac{\eta_p}{\lambda_p}\left(\frac{\partial}{\partial r_i}u_j + \frac{\partial}{\partial r_j}u_i\right) + \frac{1}{\lambda_p}\tau_{ij}. \tag{10.4}$$

Finally, for coupling to LBM the effect of polymer stress must be incorporated into the external body force:

$$F^p_{\text{ext},i}(\boldsymbol{r},t) = \frac{\partial}{\partial r_k}\tau_{ki}(\boldsymbol{r},t). \tag{10.5}$$

Discretization of the basic equations: Next comes the discretization of eq. (10.3). To this end, it is first averaged over one cell with volume $V = \Delta x^d$ (d being the dimensionality of space) and surface unit normal $\hat{\boldsymbol{n}}$:

$$\frac{\partial}{\partial t}\overline{\tau}_{ij}(\boldsymbol{r},t) = -\frac{1}{V}\int_V \frac{\partial}{\partial r_k}J_{ijk}(\boldsymbol{r},t)\,\mathrm{d}V + \overline{S}_{ij}(\boldsymbol{r},t) = -\frac{1}{V}\int_{\partial V} J_{ijk}(\boldsymbol{r},t)\hat{n}_k\,\mathrm{d}S + \overline{S}_{ij}(\boldsymbol{r},t), \tag{10.6}$$

where the overbar marks the volume average and in the last step Gauss's divergence theorem was applied. Then the same grid spacing Δx and the same time step Δt as for LBM is chosen, so that $\underline{\overline{\tau}}$ and $\underline{\overline{S}}$ correspond to the values at the grid positions. Furthermore, $\hat{\boldsymbol{n}}$ can be defined using a velocity set $\{\boldsymbol{c}_i\}$, which does not necessarily have to match the set used in the coupled LBM solver.

$$\hat{\boldsymbol{n}}_{(i)} = \frac{1}{|\boldsymbol{c}_i|A_0}\boldsymbol{c}_i \quad \text{with} \quad A_0 = \frac{1}{2d}\sum_{l=1}^{q-1}|\boldsymbol{c}_l|, \tag{10.7}$$

where in this equation no Einstein summation applies. We share the observation of [69] that D3Q7 is sufficient for most cases. Using an arbitrary velocity set $\{\boldsymbol{c}_i\}$, the following discretized form of the constitutive equation is obtained:

$$\overline{\tau}_{ij}(\boldsymbol{r},t+\Delta t) \approx -\frac{1}{\Delta x}J_{ijk}(\boldsymbol{r}+\boldsymbol{c}_l\Delta t/2,t)\hat{n}_{(l)k} + \overline{S}_{ij}(\boldsymbol{r},t) + \overline{\tau}_{ij}(\boldsymbol{r},t), \tag{10.8}$$

where $l \in \{1,\ldots,q-1\}$ and $i,j,k \in \{1,\ldots,d\}$. To compute the term $\frac{\partial}{\partial r_i}u_k(\boldsymbol{r},t)$ appearing in $\underline{\overline{S}}(\boldsymbol{r},t)$, its first-order FV discretization is used. This is again done by averaging over a volume, this time using not a cell defined by an arbitrary velocity set, but a cube-shaped cell only, which means a restriction to the D3Q7 neighborhood. This results in the approximation (derivation cf. [69, eq. 32-33])

$$\frac{\partial}{\partial r_i}u_k(\boldsymbol{r},t) \approx \left(u_k(\boldsymbol{r}+\hat{\boldsymbol{e}}_i/2,t) - u_k(\boldsymbol{r}-\hat{\boldsymbol{e}}_i/2,t)\right), \tag{10.9}$$

where $\hat{\boldsymbol{e}}_i$ denotes the unit vector. Values of quantities $Q \in \{\boldsymbol{u}, \underline{\tau}_p\}$ at the cell surface, as they appear in eq. (10.8) and (10.9), are obtained via interpolation:

$$Q(\boldsymbol{r}+\boldsymbol{c}_i\Delta t/2,t) \approx \frac{1}{2}(Q(\boldsymbol{r},t) + Q(\boldsymbol{r}+\boldsymbol{c}_i\Delta t,t)). \tag{10.10}$$

Note that inserting eq. (10.10) into eq. (10.9) yields an expression identical to the first-order FD scheme. Analogous to above, the averaged value for the force after applying Gauss's divergence theorem is obtained as

$$\overline{F}^p_{\text{ext},j}(\boldsymbol{r},t) = \frac{1}{V}\int_{\partial V}\tau_{ij}(\boldsymbol{r},t)\hat{n}_i\,\mathrm{d}S \approx -\frac{1}{\Delta x}\tau_{ij}(\boldsymbol{r}+\boldsymbol{c}_l\Delta t/2,t)\hat{n}_{(l)i}. \tag{10.11}$$

For the remainder of this thesis, the overbar notation for the mean value at a grid position will be dropped again.

Optimized memory format: To save memory as well as computational operations, the symmetry of the stress tensor is exploited. Instead of the full d^2 entries, an upper triangular matrix with $d(d+1)/2$ entries is used. The matrix itself is stored in a 1D array, so the mapping from the full matrix to the 1D array can be specified defining the index function $f_{\text{map}}(i,j)$ as follows:

$$f_{\text{map}}(i,j) = \begin{cases} id-(i-1)i/2+j-i, & i \leq j \\ jd-(j-1)j/2+i-j, & \text{else.} \end{cases} \tag{10.12}$$

Alternative advection: This thesis will refer to the computation of the flux term via the first term of eq. (10.8) as the **simple** scheme. An alternative advection scheme called *corner transport upwind* (**CTU**) scheme will be used in several simulations throughout this thesis. It will be further described in chapter 10.8.1.

Boundaries: In order to handle the two possible boundary conditions from chapter 5.2, namely the *moving no-slip bounce-back boundary* (also referred to as moving "wall") and the *equilibrium boundary*, the FVM has to be adapted accordingly. No stress should be transported over wall boundaries, which is achieved by simply setting $\underline{J}(\boldsymbol{r}_w + \boldsymbol{c}_i\Delta t/2) \equiv 0$, where $\boldsymbol{r}_w$ is a wall node and $\boldsymbol{r}_w + \boldsymbol{c}_i\Delta t$ is a fluid node. Also, note that

$$\boldsymbol{u}(\boldsymbol{r}_w + \hat{\boldsymbol{e}}_i/2, t) \equiv \boldsymbol{u}_w \tag{10.13}$$

holds (cf. chapter 5.2), so the velocity between boundary node and fluid node is given by $\boldsymbol{u}_w$ and not by interpolation via eq. (10.10). To obtain a value for $\underline{\tau}_p(\boldsymbol{r}_w)$ from eq. (10.11), constant interpolation is used: $\underline{\tau}_p(\boldsymbol{r}_w) \equiv \underline{\tau}_p(\boldsymbol{r}_w + \boldsymbol{c}_i\Delta t)$.

Also for equilibrium boundaries at position $\boldsymbol{r}_e$ one gets $\underline{\tau}_p(\boldsymbol{r}_e)$ by constant extrapolation. Since the velocity boundary condition of equilibrium boundaries might not be parallel to the wall, here $\underline{J}(\boldsymbol{r}_e + \boldsymbol{c}_i\Delta t/2) \neq 0$ in general. Since the main focus of this work is not on the use of equilibrium boundaries for advection problems (it will only used once for a validation in chapter 10.6.3), I use constant extrapolation of velocity for simplicity. Together with the constant extrapolation of stress this e.g. results in $\underline{J}(\boldsymbol{r}_e + \boldsymbol{c}_i\Delta t/2) \equiv \underline{J}(\boldsymbol{r}_e + \boldsymbol{c}_i\Delta t)$ in case of the **simple** scheme. How to implement better elaborated equilibrium boundary conditions for advection diffusion problems can be found in [26, ch. 8.5].

Two programs: A fully parallelized version designed for GPUs including all the details mentioned above was implemented in FluidX3D. For testing purposes, a reduced version was implemented in ESPResSo, running CPU parallel via the *Message Passing Interface* MPI. It does not make use of the optimized memory format, has only the **simple** advection scheme available and is not compatible with equilibrium boundaries.

10.4 Validation of Oldroyd-B

10.4.1 Validation via rheometer

The rheometer setup consists of a cuboid domain filled with initially quiescent fluid, bounded by walls at $z = 0$ and $z = H$ and periodic along the other two directions. The lattice nodes used in stream-flow direction (here x-direction) and traverse-flow direction (here y-direction) are denoted as L_s and L_t, respectively. This notation will be used for all shear flow setups throughout the

remainder of this thesis. The wall at $z = 0$ has velocity $u_x(t) = -\omega \sin(\omega t + \Theta)\gamma_0/2$, the wall at $z = H$ moves exactly opposite with $u_x(t) = \omega \sin(\omega t + \Theta)\gamma_0/2$, the other velocity components are zero in each case. This is realized by moving bounce-back boundary conditions. The stress components $\sigma_{s,xz}$ and τ_{xz} are measured for different heights $z = h + 1/2$, $0 \leq h \leq H - 1$ (each value at a specific height is averaged over the whole corresponding fluid layer). It holds

$$\sigma_{xz}(t) = G'\gamma(t) + \frac{G''}{\omega}\dot{\gamma}(t), \tag{10.14}$$

By means of a fit

$$-G' \cos(\omega t + \Theta)\gamma_0/H + G'' \sin(\omega t + \Theta)\gamma_0/H + c, \tag{10.15}$$

with G', G'' and c as the only free parameters the moduli can be determined. The fit is started only after half of the total calculated oscillations (wait for transient to be finished). No dependence of z for G', G'' is to be expected, which must be checked in the following simulations in each case. For the final determination of the moduli, the average value over all fluid layers is taken.

To quantify the accuracy of the results, two error definitions are useful. One is the error

$$s_r(Q_{\text{theo}}, Q_{\text{sim}}) = \frac{|Q_{\text{theo}} - Q_{\text{sim}}|}{|Q_{\text{theo}}|}, \tag{10.16}$$

which is the relative deviation between the analytical solution Q_{theo} and the numerical result Q_{sim}. On the other hand the error

$$s_1(Q) = \sqrt{\text{Var}(Q) + \text{Mean}(s_{Q,\text{fit}})^2}. \tag{10.17}$$

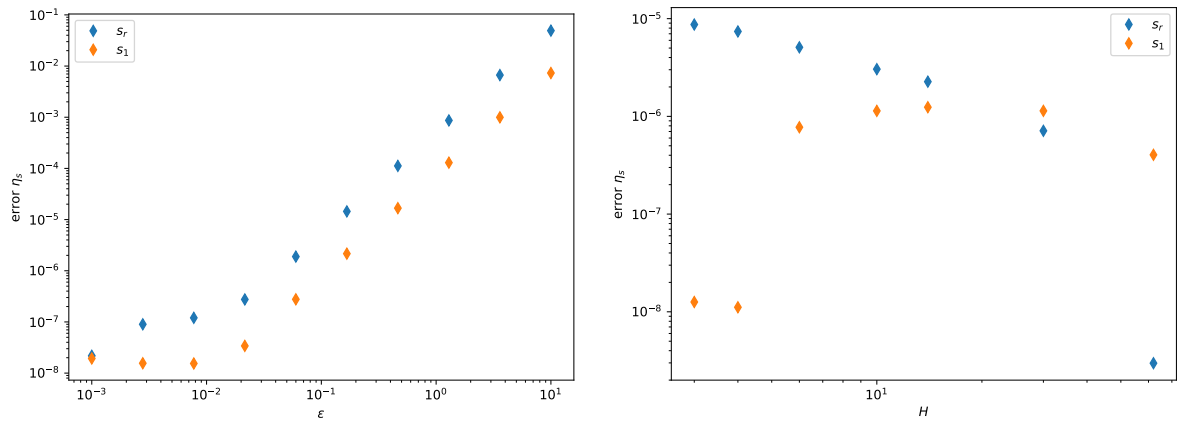
The term $\text{Var}(Q)$ denotes the variance of the means of the fluid layers. The error $s_{Q,\text{fit}}$ is the standard deviation of Q obtained when fitting via eq. (10.15). For all the following simulations, D3Q19 and the TRT operator with $\Lambda_{\text{TRT}} = 3/16$ are used.

10.4.1.1 Pure viscous fluid

A simple method to check this setup is to calculate the viscosity of a pure viscous fluid using $\eta = \eta_s = G''/\omega$. This Newtonian case is also used to tune the accuracy of the simulation via the free parameters. The procedure is as follows: First, $\rho = 1$ and $\nu = 1/6$ are chosen (all parameters in lattice units), which is an often chosen standard for LBM. Furthermore, $\gamma_0 = 0.1$ and $\Theta = 0$ are chosen. Then a reasonable value for ω must be determined. As [23] correctly notes, care must be taken to ensure that a steady shear flow can form during each oscillation. The time scale required for this is $\tau_{\nu_s} = \frac{H^2}{\nu_s}$, as can be seen by looking at eq. (10.18). So $\tau_{\nu_s}\omega \ll 1$ must hold, otherwise one would still be in the transient regime. Therefore, $H = 6$ and $\omega = \epsilon/\tau_{\nu_s}$ is chosen, where the smallness parameter ϵ is varied. Furthermore, $L_s = 32$ and $L_t = 1$ are set. Ten oscillations are calculated, of which the last five are used for the fit. In fig. 10.7a, $\tau_{\nu_s}\omega < 0.1$ is evident as a sufficient condition. Moreover, it is nice to see how the exponential turn-on behavior of the steady shear flow is reflected as a straight line in the log-log plot of the error. Overall, the error is dominated by the transient regime, since $s_r > s_1$ holds.

Next, the channel height H is varied with fixed $\epsilon = 0.1$. In this process, L_s must also be adjusted (see tab. 10.2), since for optimization reasons the total number of lattice nodes must be

divisible by 256 without remainder. For increasing channel resolution, a small decrease of error s_r is visible in fig. 10.7b. The error s_1 is rather constant over a long range of H, the parameter selection strategy is therefore considered successful.



(a) (b)

Fig. 10.7: Measurement of the error $s_r(\eta_{s,\text{theo}}, \eta_{s,\text{sim}})$ and $s_1(\eta_{s,\text{sim}})$ via eq. (10.16) and eq. (10.17) for a pure viscous fluid ($\eta_s \equiv \eta$) using the rheometer setup. In (a), the smallness parameter ϵ is varied while keeping $H = 6$. In (b), channel height H is varied while keeping $\epsilon = 0.1$.

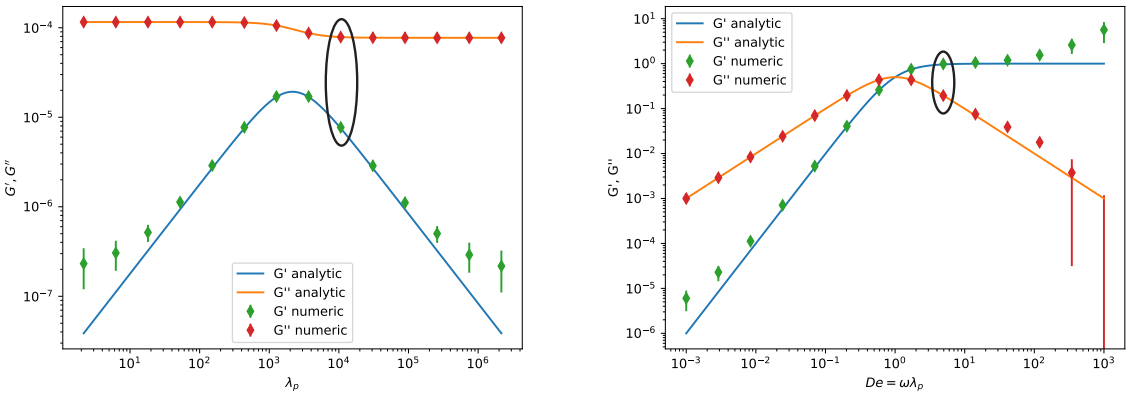
H	3	4	6	10	14	30	62
L	256	128	32	64	16	8	4

Tab. 10.2: Since the product $L_s L_t (H + 2)$ must be a multiple of 256 and $L_t = 1$ holds, L_s is chosen suitable for H in each case.

10.4.1.2 Viscoelastic fluid

Now the same setup is used for a viscoelastic fluid. Again the last five of ten total oscillations are used for the fit. In addition, $H = 6$ and for the moment $\epsilon = 0.1$ are chosen. In fig. 10.8a one can compare the analytical solution according to eq. (4.41) with the simulation results, where for the fitting of G' and G'' to eq. (10.15) the whole tensor $\underline{\sigma}$ consisting of both the pure viscous fraction $\underline{\sigma}_s = 2\eta_s \underline{D}$ and the viscoelastic fraction $\underline{\tau}_p$ according to eq. (4.10) was used. One could also just use $\underline{\tau}_p$, since the contribution of $\underline{\sigma}_s$ to G'' is known to be $\omega\eta_s$, while the contribution to G' should be zero. However, this would reduce the reliability of the validation. Significant deviations from the analytical solution are observed for G' for large and small De, respectively. In fig. 10.8b, the moduli are shown in the non-dimensional form according to eq. (4.43). Here, in addition to the deviation in G', a deviation in G'' is evident. All deviations can be explained by s_1. Another simulation with $\epsilon = 0.001$ can improve the result significantly and is shown in fig. 10.8b. The reason why $\epsilon = 0.1$ is not sufficient is presumably that in the viscoelastic case further time constants than τ_{ν_s} play a role. Minor deviations now exist only for large De in fig. 10.8b. A possible explanation is that a smaller ϵ leads to a more accurate shear flow, but at the same time $\lambda_p \propto 1/\epsilon$ and $\lambda_p \propto \text{De}$ hold. If the implemented Oldroyd-B model is no longer valid for large λ_p, this is especially noticeable for large De and even worsens for a smaller choice

of ϵ. For De $= 1000$, actually $\lambda_p \approx 2 \times 10^8$ holds, which is far from the $\lambda_p \approx 10^3$ usually used. Attempts were made to reduce λ_p by rescaling ν, but this only worsened the result.



(a) (b)

Fig. 10.8: Rheometer setup with $\epsilon = 0.1$. The points show the simulation results, while the solid lines show the analytical solution given by eq. (4.41). (a) shows G' and G'' in lattice units, (b) shows the results in non-dimensional form in accordance with eq. (4.43). The error bars indicate the error s_1 computed via eq. (10.17). In fig. 10.9 it is shown in more detail, how the value pair encircled in black is determined.

In all simulation results shown, **simple** with a D3Q7 neighborhood was sufficient for advection. Other combinations of {**simple**, **CTU**} x {D3Q7, D3Q19, D3Q27} were tested successfully as well, but bring no apparent increase in accuracy.

10.4.2 Validation via inception of poiseuille flow

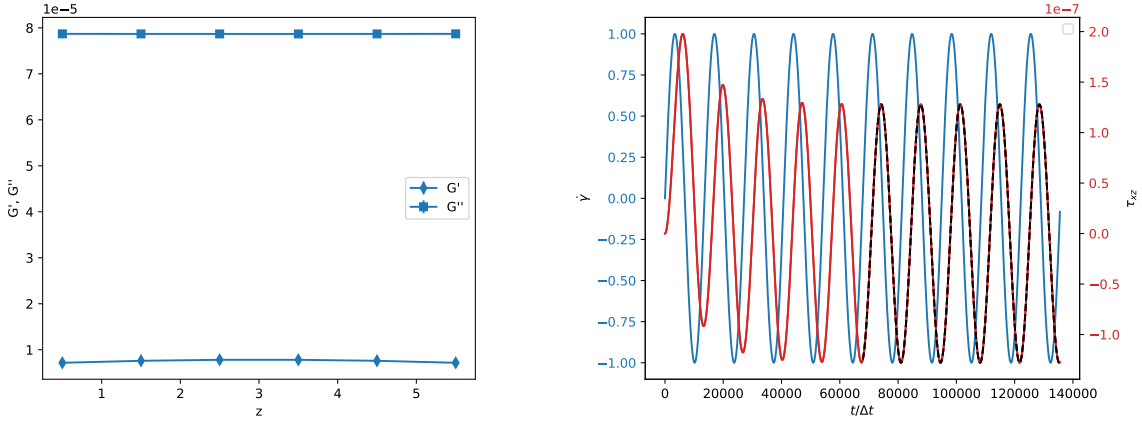
To additionally check the dynamics of the system, a validation procedure analogous to [69, fig. 2] is followed. A planar poiseuille channel is considered, which has non-moving no-slip bounce-back boundaries with $\boldsymbol{u}((x,0,z)^{\mathrm{T}},t) = \boldsymbol{u}((x,H,z)^{\mathrm{T}},t) = 0$ and is otherwise periodic. The size of the domain is L_s in stream-flow direction (x-direction) and L_t in traverse-flow direction (z-direction). The flow is driven by a homogeneous force parallel to the channel walls, $\boldsymbol{\mathcal{F}} = \mathcal{F}_x \hat{\boldsymbol{e}}_x$, which leads to a parabolic steady state flow profile both in the Newtonian and the Oldroyd-B case. After instantaneously incepting the force in a resting Newtonian fluid, the steady-state flow is approached in a monotonous manner and is described by [70, eq. 65]

$$\frac{u}{u_0}(y_1,t_1) = -4y_1(y_1-1) - 32\sum_{n=1}^{\infty}\frac{\sin(Ny_1)}{N^3}\exp(-N^2 t_1), \tag{10.18}$$

where

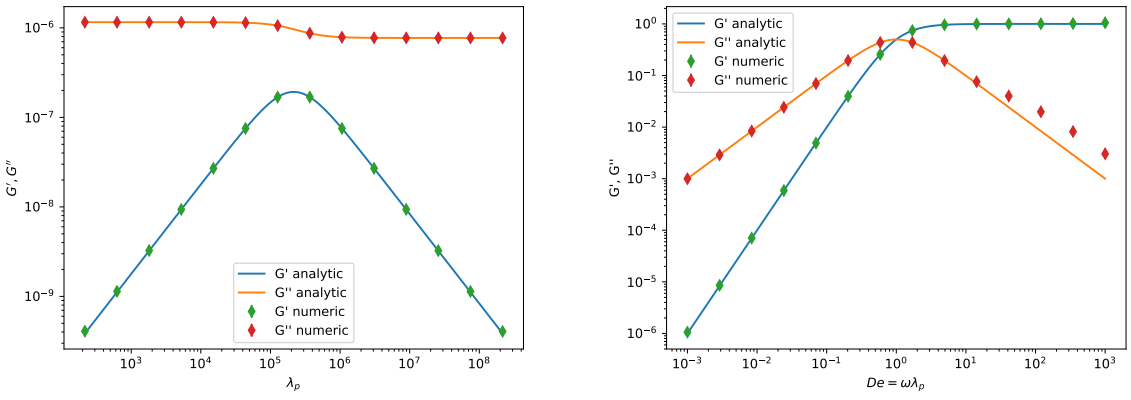
$$t_1 = \frac{\eta t}{\rho H^2}, \qquad y_1 = \frac{y}{H}, \qquad u_0 = \frac{\mathcal{F}_x H}{\eta}, \qquad N = (2n+1)\frac{\pi}{2}. \tag{10.19}$$

The flow velocity of a viscoelastic fluid, however, can overshoot its steady-state value and then decay to it on a time scale given by λ_p. The analytical expression is provided by [70, eq. 64] for the *liquid B'* model, which has shown to be equivalent to Oldroyd-B [69]. The time dependent



(a) (b)

Fig. 10.9: The encircled value pair of fig. 10.8 is determined as the mean value of the liquid layers between 0 and H, where the corresponding variance is ideally small (a). In (b), the shear rate (normalized to the interval [-1,1]) is shown in blue. The component τ_{xz} in lattice units is shown in red red, the means of all individual fluid layers are plotted on top of each other. After 5 oscillations the fit according to eq. (10.15) starts (black dashed line, again the fits of all layers are plotted on top of each other). For the calculation of G' and G'' in reality σ_{xz} is used, but here the phase difference to $\dot{\gamma}$ would no longer be visible to the naked eye.



(a) (b)

Fig. 10.10: Like fig. 10.8, but for $\epsilon = 0.001$.

solution reads

$$\frac{u}{u_0}(y_1, t_1) = -4y_1(y_1 - 1) - 32 \sum_{n=1}^{\infty} \frac{\sin(Ny_1)}{N^3} \exp\left(-\frac{\alpha_N t_1}{2S_1}\right) G_N(t_1), \tag{10.20}$$

where

$$G_N(t_1) = \left(\cosh\left(\frac{\beta_N t_1}{2S_1}\right) + \frac{(1 + N^2(S_2 - 2S_1))}{\beta_N} \sinh\left(\frac{\beta_N t_1}{2S_1}\right)\right). \tag{10.21}$$

Here the additional definitions

$$\alpha_N = 1 + S_2 N^2, \quad \beta_N = ((1 + S_2 N^2)^2 - 4S_1 N^2)^{(1/2)}, \quad S_1 = \frac{\eta\lambda_1}{\rho H^2}, \quad S_2 = \frac{\eta\lambda_2}{\rho H^2} \tag{10.22}$$

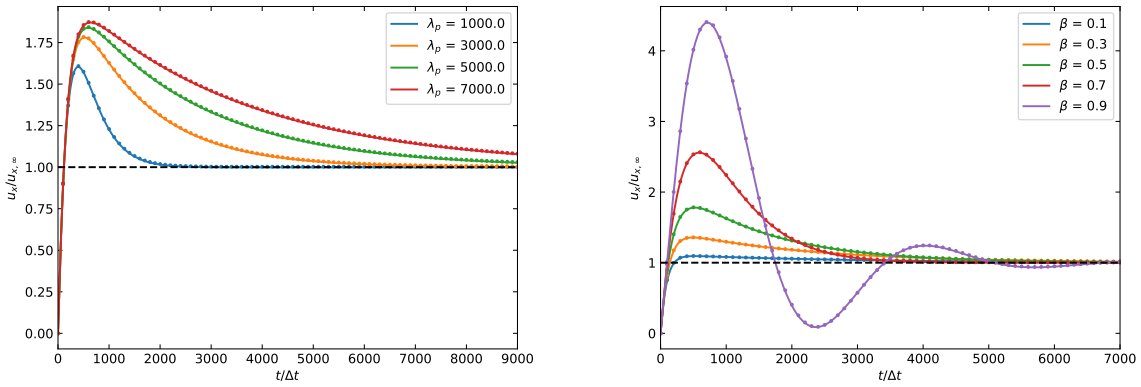
are used and it holds $\lambda_1 = \lambda_p$ and $\lambda_2 = (1 - \beta)\lambda_p$, where β, as always, denotes the polymer fraction of total viscosity.

To evaluate the above solution, the infinite sum above has to be truncated after m terms, where $m = 10$ was found to be sufficient. Furthermore, special care has to be taken when evaluating $G_N(t_1)$ from eq. (10.21), since cosh and sinh do overflow for higher n. To prevent the overflow, one has to rewrite the term $\exp\left(-\frac{\alpha_N t1}{2S_1}\right) G_N(t1)$ in a form, where besides polynomials only terms $\propto \exp\left(-\frac{\alpha_N t_1}{2S_1} \pm \frac{\beta_N t_1}{2S_1}\right)$ appear.

For the simulation, channel dimensions of $L_s \times H \times L_t = 32\Delta x \times 28\Delta x \times 4\Delta x$ are chosen. Furthermore, $\rho = 1$ and $\eta = 1$ are fixed. The body force $F_x = 2u_0\rho\nu_s/R^2$ is chosen such that $u_0 = 10^{-3}$ results. For comparison with the analytical solution, only the velocity in the center of the channel is looked at. In fig. 10.11a, $\lambda_p \in \{1000, 3000, 5000, 7000, 9000\}$ is varied and $\beta = 0.5$ is fixed. Both the magnitude of the overshoot and the characteristic decay time scale with λ_p. In fig. 10.11b, $\beta \in \{0.1, 0.3, 0.5, 0.7, 0.9\}$ is varied and $\lambda_p = 3000$ is fixed. The magnitude of the overshoot increases with β. For large β the flow decays to steady-state velocity in an oscillatory fashion.

Both figures show very good agreement with the analytical solution. Indeed, they are much better than in [69], who has to fit the analytical solution using H as a free parameter in order to improve the solution and still doesn't reach a comparable accuracy. The author speculates that the constant extrapolation of stress to the wall nodes might be a reason for the deviations, but in this thesis the same extrapolation is used. Still the reason can be found in the treatment of the boundaries in the FV solver of the Oldroyd-B constitutive equation. As the author confirmed, [69] does not use special treatment of velocity interpolation at boundaries like it is done in this thesis by accounting for eq. (10.13). This is equivalent to assuming wrong boundary positions, which according to chapter 5.2 reduces the accuracy of the fluid solver to first order.

The results of fig. 10.11 were generated in FluidX3D with TRT and D3Q19 concerning the LBM, and **simple** and D3Q7 concerning the advection in the FVM. All other combinations of {**simple**, **CTU**} x {D3Q7, D3Q19, D3Q27} for the FVM were tested successfully as well. The implementation in ESPResSo yields similarly good results.



(a) (b)

Fig. 10.11: Time dependent center velocity in a planar poiseuille channel after inception of volume force. The analytical solution (solid lines) and the numerical results (dots) match very good. In (a) $\beta = 0.5$ was fixed, while in (b) $\lambda_p = 3000$ was chosen.

10.5 Implementation idea FENE-P

First, it should be noted that it is possible to reformulate the FENE-P model in such a way that only one time derivative occurs in the constitutive equation. Then $\underline{\tau}_p(t + \Delta t) \approx \underline{\tau}_p(t) + \frac{\partial}{\partial t}\underline{\tau}_p(t)$ could simply be set as was done for the Oldroyd-B model in chapter 10.3. This strategy is e.g. used in [23]. However, such a reformulation has the disadvantage that it is no longer possible to resort to the preliminary work of the Oldroyd-B implementation. Furthermore, [23] have an inconsistency in their implementation (cf. chapter 4.2.2). Instead, this thesis implements the FENE-P model in a very simple way: First, the constitutive equation for $\epsilon = 0$ is solved for $\frac{\partial}{\partial t}\underline{\tau}_p$, i.e.

$$\frac{\partial}{\partial t}\underline{\tau}_p = 2\frac{\eta_p}{\lambda_p}\underline{D} + ((\nabla \boldsymbol{v})^{\mathrm{T}} \cdot \underline{\tau}_p + \underline{\tau}_p \cdot (\nabla \boldsymbol{v})) - \boldsymbol{v} \cdot \nabla \underline{\tau}_p - \frac{1}{\lambda_p} Z \underline{\tau}_p + \frac{1}{\lambda_p}\frac{\mathrm{d} \ln Z}{\mathrm{d}t}(\underline{\tau}_p + \frac{\eta_p}{\lambda_p}\underline{\mathrm{I}}). \quad (10.23)$$

The treatment of the first three terms on the right hand side is exactly the same as for the Oldroyd-B model. For term four, Z must be known, which can be easily calculated if $\underline{\tau}_p$ is known. The last term contains a total time derivative $\frac{\mathrm{d} \ln Z}{\mathrm{d}t} = \frac{\partial \ln Z}{\partial t} + \boldsymbol{v} \cdot \nabla \ln Z$. The partial time derivative occurring here is implemented as a backward difference. To do this, Z must be stored from the previous iteration. The term $\boldsymbol{v} \cdot \nabla \ln Z$ is an advection term and is treated analogously to the term $\boldsymbol{v} \cdot \nabla \underline{\tau}_p$.

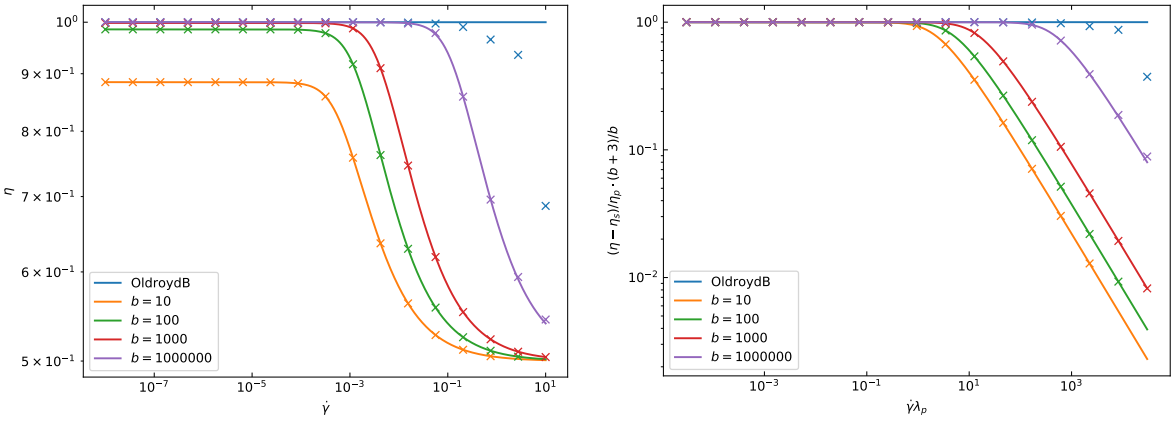
Specifically, this is done as follows: In the FENE-P implementation, a new field must be introduced, in order to be able to perform a backward difference of Z using the previous time step. If instead of Z the value $Z - Z_0$ is stored with $Z_0 = Z(\underline{0}) = 1 + 3/b$ (cf. eq. (4.19)), the field can be treated analogously to the stress tensor $\underline{\tau}_p$ with respect to all advection algorithms: e.g. for the algorithm M3 and M4 from chapter 10.8.3.1 it applies $\underline{\tau}_p = \underline{0}$ within the capsules, where now also the new field storing $Z - Z_0$ takes the value zero. To further reduce special treatments, the new field is implemented as an additional component of the field that manages the stress tensor. Doing so, the changes remain minimal, only the number of different components `def_symm_dimXdim_size` of the symmetric stress tensor and the number of components related to all advection processes `def_stress_advection_size` must be carefully distinguished.

10.6 Validation of FENE-P

10.6.1 Validation via steady shear flow

The analytical solution for the shear-dependent viscosity of the FENE-P fluid is known from eq. (4.51). With this, the shear-thinning behavior of the present implementation is now to be checked by means of a simple setup. Just as in [23], a domain of size $L_s \times L_t \times H = 2\Delta x \times 2\Delta x \times 62\Delta x$ is used, applying periodic boundary conditions in stream-flow and traverse-flow direction and having walls with bounce-back rule at $z = 0$ and $z = H$. They move in opposite directions to each other with velocity $\boldsymbol{u}_w(z = 0) = (\dot{\gamma}H/2, 0, 0)^{\mathrm{T}} = -\boldsymbol{u}_w(z = H)$ and thus generate a shear flow with constant steady state shear rate $\dot{\gamma}$. Furthermore, D3Q19 and MRT are chosen for the LBM, while D3Q7 and **simple** are sufficient for the advection regarding the FVM. Fixed viscosity fraction $\beta = 0.5$, total viscosity $\eta = 1$ and relaxation time $\lambda_p = 3000$ are used. The shear rate $10^{-8} \leq \dot{\gamma} \leq 10^{1}$ is varied such that equal spacing results in a logarithmic plot. Several $b \in \{10, 100, 1000, 1000000\}$ are tested for the FENE-P model, the Oldroyd-B model is tested as well. All mentioned parameters are rescaled depending on the value of $\dot{\gamma}$ to reduce simulation time, leading from one set of lattice parameters to an other without changing spacial resolution. A fixed number of simulation steps $t_f = 10^7 \Delta t$ is chosen for all simulations.

The simulation results are shown in fig. 10.12 and yield very good agreement with the analytical solution for $\dot{\gamma}\lambda_p \leq 10^3$. At the same time, one can also observe well how the FENE-P model for $b \to \infty$ transitions into the Oldroyd-B model, i.e. has constant viscosity. For $\dot{\gamma}\lambda_p > 10^3$ the simulation takes a very long time to get into a steady-state. Since nevertheless a constant number of simulation steps was simulated independent of $\dot{\gamma}$, this can explain the deviations from the analytical solution. By using a smaller domain, i.e. $H = 2$, the transition to steady-state is accelerated and now the simulation also shows very good agreement with the analytical solution for $\dot{\gamma}\lambda_p > 10^3$ (cf. fig. 10.13). In contrast to [23, fig. 1] no deviations between theory and simulation for low dimensionless shear rates $\dot{\gamma}\lambda_p$ are observed. Moreover, the simulation data provided in this thesis extends two orders of magnitude further towards high dimensionless shear rates.



(a) (b)

Fig. 10.12: Validation of shear-thinning behavior of FENE-P using a domain of size $L_s \times L_t \times H = 2\Delta x \times 2\Delta x \times 62\Delta x$. The solid lines show the theory curves, the crosses mark the simulation results. (a) shows the total viscosity, (b) the dimensionless viscosity due to polymers.

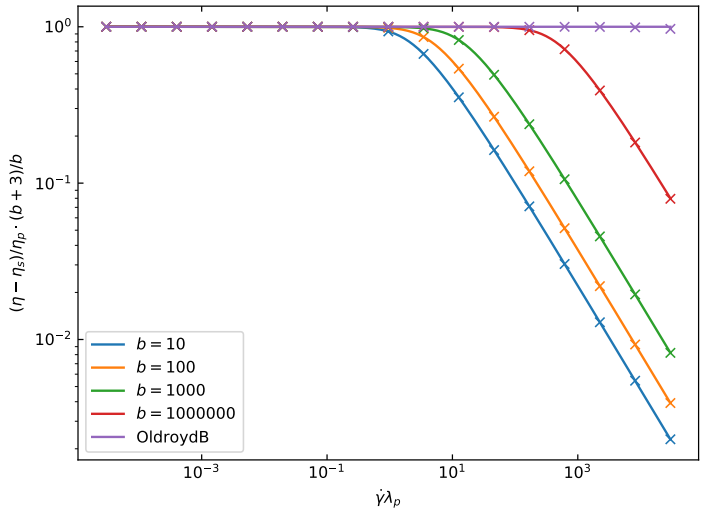
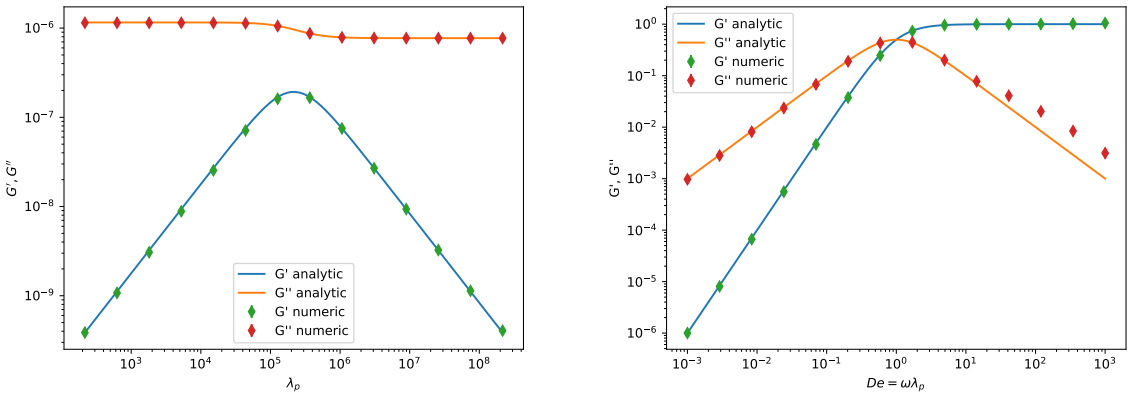


Fig. 10.13: Like fig. 10.12b, but for $H = 2$.

10.6.2 Validation via rheometer

For the oscillating shear flow, the solution for G' and G'' for small amplitudes is very similar to the Oldroyd-B case. Therefore, the same parameters as in fig. 10.10 and additionally $b = 100$ were chosen. In fig. 10.14 very good agreement between theory and simulation can be observed, with the problems for large De as already mentioned in chapter 10.4.1.2.



(a) (b)

Fig. 10.14: Like fig. 10.10, but for the FENE-P model and with the additional parameter $b = 100$.

10.6.3 Validation via elongational flow

So far, the FENE-P model has only been investigated in different variants of shear flow. The behavior in elongational flow is also interesting, where for validation the analytical solution according to eq. (4.61) can be used.

[23] use this validation successfully. Unfortunately, they do not specify how they generate an elongational flow with the LBM. They only state that their simulation domain is periodic in all directions. Accordingly, they cannot use equilibrium boundaries. They might give a location-dependent volume force that produces the desired velocity field, but it's not clear how this force field would look like. Or perhaps they just specify a location-dependent, time-independent velocity field throughout the domain without actually computing the velocities by the LBM.

For the validation setup of this thesis, at first equilibrium boundaries were used on which the elongation velocity profile according to eq. (4.31) is given. The expectation is that the same velocity profile will then form throughout the whole domain for $t \to \infty$. It was taken into account that, in contrast to moving bounce-back boundaries, the desired velocity is defined at the center of the node, not in-between nodes. Furthermore, $\nu = 1/6$ was chosen in order to still have a second-order solver (cf. chapter 5.2).

Unfortunately, the validation via equilibrium boundaries is not very successful. This is probably mainly due to the fact that the target velocity field does not evolve. After convergence of the simulation, the total error s_2 of the flow profile $\boldsymbol{u}_{\text{sim}}$ obtained by simulation is calculated via the L_2 norm [26]:

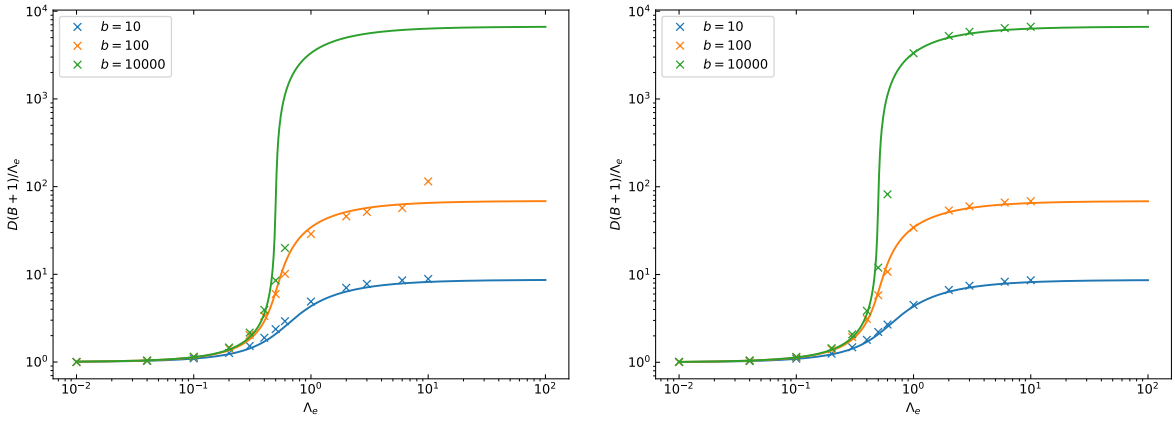
$$s_2(\boldsymbol{u}_{\text{sim}}) = \sqrt{\frac{\sum_{\boldsymbol{r}\in\text{domain}} |\boldsymbol{u}_{\text{sim}}(\boldsymbol{r}) - \boldsymbol{u}_{\text{theo}}(\boldsymbol{r})|^2}{\sum_{\boldsymbol{r}\in\text{domain}} |\boldsymbol{u}_{\text{theo}}(\boldsymbol{r})|^2}}, \tag{10.24}$$

where $\boldsymbol{u}_{\text{theo}}(\boldsymbol{r})$ denotes the theoretical elongational flow profile. The L_2 error shown in fig. 10.16 is proportional to the deviation of simulation and theory from fig. 10.15a. Furthermore, it is easy to see that the algorithm quickly reaches its stability limits (cf. the missing simulation data in fig. 10.15a versus fig. 10.15b). In a second approach, the LBM is switched off and instead the target flow profile is prescribed by a time-independent velocity field everywhere in the domain. The equilibrium boundaries are still present in order to correctly handle the advection of polymer stress at the boundaries. With this approach, there are only small deviations from the analytical solution (cf. fig. 10.15b). However, there is still a slight offset compared to the analytical solution. This could possibly be further improved if the simulation domain were made completely periodic. Then the equilibrium boundaries wouldn't be needed anymore, and the presumable boundary effect due to their naive implementation would disappear. However, this will not be done here. For the first approach with equilibrium boundaries only, a cube-shaped domain with side length $L_d = 29\Delta x$ was used and D was determined as the mean value of a cube-shaped region of side length $L_m = 8$ around the center of the domain, i.e. as far away as possible from the domain boundaries. For the second approach where the velocity field was prescribed everywhere, $L_d = 61\Delta x$ and an evaluation region of size $L_m = L_d/2$ was chosen to improve accuracy. Each simulation was run until the convergence of D.

For the plots from fig. 10.15 the theoretical solution was computed as follows: Equation (4.61) was numerically inverted in the desired range of values to obtain $D(\Lambda_e, b)$. Then $D(B+1)/\Lambda_e$ was plotted along Λ_e, i.e., the dimensionless elongation viscosity scaled by its limit for small Λ_e.

10.6.4 Validation via cessation of steady shear flow

Up to now, only steady-state solutions have been used for validation. To test the model dynamics, instead of looking at the inception of volume force in a planar poiseuille flow like in chapter 10.4.2 for Oldroyd-B, the cessation of steady shear flow will be investigated. For shear flows



(a) (b)

Fig. 10.15: Validation via elongational flow using equilibrium boundaries. The simulation must match the numerically inverted analytical solution (4.61). In (a), only the boundaries are given, which results in a flow deviating from the elongational one (see fig. 10.16). In (b), additionally the required velocity field is prescribed everywhere in the fluid, which improves stability as well as accuracy significantly.

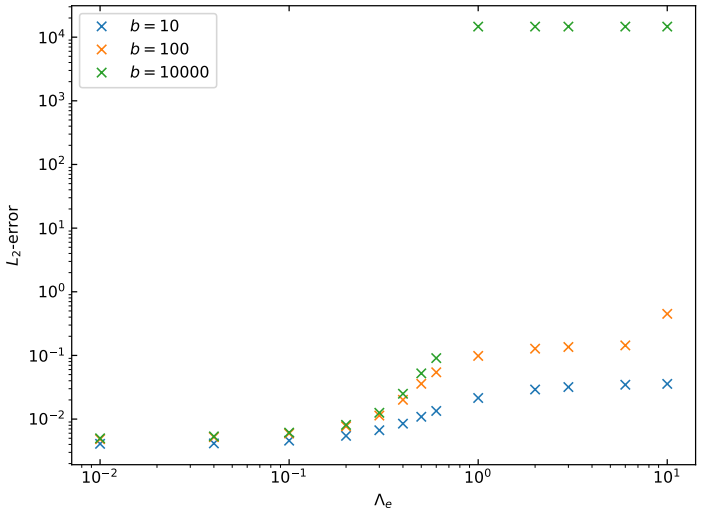


Fig. 10.16: L_2-error of the velocity fields corresponding to the simulations from fig. 10.15a. The L_2-error $s_2(\boldsymbol{u}_{\text{sim}})$ is computed via eq. (10.24).

($u_x = \dot{\gamma}(t)y$, $u_y = u_z = 0$) the FENE-P constitutive equation [19, eq. 12] results in

$$Z\begin{pmatrix}\tau_{xx} & \tau_{xy} & 0\\ \tau_{yx} & \tau_{yy} & 0\\ 0 & 0 & \tau_{zz}\end{pmatrix} + \lambda_p\left(\frac{\mathrm{d}}{\mathrm{d}t}\begin{pmatrix}\tau_{xx} & \tau_{xy} & 0\\ \tau_{yx} & \tau_{yy} & 0\\ 0 & 0 & \tau_{zz}\end{pmatrix} - \begin{pmatrix}2\tau_{yx} & \tau_{yy} & 0\\ \tau_{yy} & 0 & 0\\ 0 & 0 & 0\end{pmatrix}\dot{\gamma}\right)$$
$$-\lambda_p\begin{pmatrix}\tau_{xx}+nk_\mathrm{B}\theta & \tau_{xy} & 0\\ \tau_{yx} & \tau_{yy}+nk_\mathrm{B}\theta & 0\\ 0 & 0 & \tau_{zz}+nk_\mathrm{B}\theta\end{pmatrix}\frac{\mathrm{d}\ln Z}{\mathrm{d}t} = nk_\mathrm{B}\theta\lambda_p\dot{\gamma}\begin{pmatrix}0 & 1 & 0\\ 1 & 0 & 0\\ 0 & 0 & 0\end{pmatrix} \tag{10.25}$$

After switching off the shear flow, this simplifies further due to $\dot{\gamma} = 0$. For $T = \mathrm{Tr}(\underline{\tau}_p)/(3nk_\mathrm{B}\theta)$ it therefore holds

$$ZT + \lambda_p\frac{\mathrm{d}T}{\mathrm{d}t} - \lambda_p(T+1)\frac{\mathrm{d}\ln Z}{\mathrm{d}t} = 0. \tag{10.26}$$

With the definition $B = 3/b$ and because of $Z = Z(T) = 1 + B(T+1)$ and $\frac{\mathrm{d}\ln Z}{\mathrm{d}t} = \frac{1}{Z}\frac{\mathrm{d}Z}{\mathrm{d}t} = \frac{B}{Z}\frac{\mathrm{d}T}{\mathrm{d}t}$ this simplifies to

$$ZT + \lambda_p\frac{\mathrm{d}T}{\mathrm{d}t}\left(1 - (T+1)\frac{B}{Z}\right) = 0. \tag{10.27}$$

Finally, using $(T+1)B/Z = 1 - 1/Z$ this results in the following differential equation:

$$Z^2T + \lambda_p\frac{\mathrm{d}T}{\mathrm{d}t} = 0. \tag{10.28}$$

In the Oldroyd-B limit ($Z = 1$), a simple exponential decay $T(t) = T_0e^{(-(t-t_0)/\lambda_p)}$ results after switching off the shear flow at time $t = t_0$. The solution of the general case with $Z = Z(T)$ cannot be given by means of elementary functions. However, one can specify a function for $S = \tau_{xy}/(3nk_\mathrm{B}\theta)$, where $S = S(T, T_0, S_0, B)$. The differential equation for S can be extracted from eq. (10.25) and reads as follows:

$$ZS + \lambda_p\frac{\mathrm{d}S}{\mathrm{d}t} - \lambda_pS\frac{\mathrm{d}\ln Z}{\mathrm{d}t} = 0. \tag{10.29}$$

Substituting Z in the first term by the relation as given in eq. (10.28), calculating $\frac{\mathrm{d}\ln Z}{\mathrm{d}t}$ explicitly as above and exploiting $\frac{1}{S}\frac{\mathrm{d}S}{\mathrm{d}t} = \frac{\mathrm{d}\ln S}{\mathrm{d}t}$, eq. (10.29) is transferred into

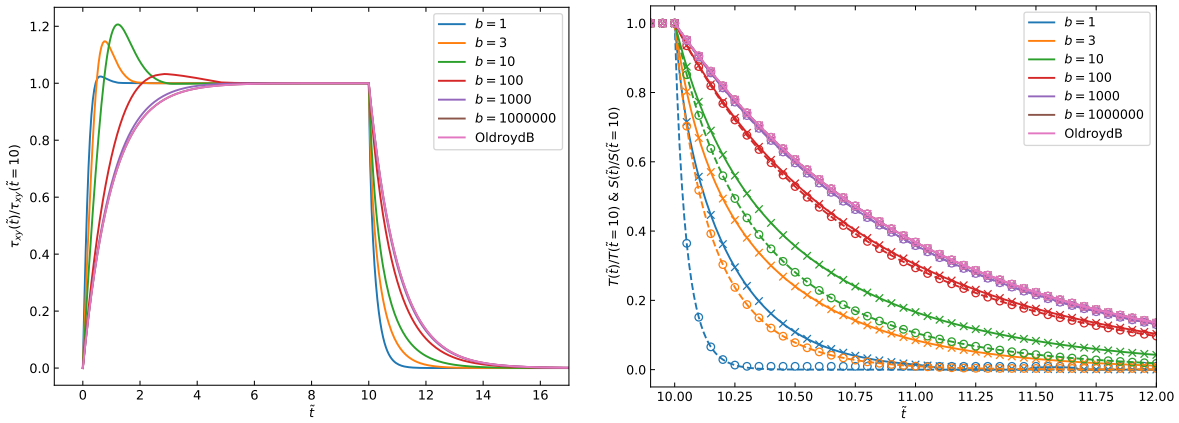
$$\frac{\mathrm{d}\ln S}{\mathrm{d}\tilde{t}} = \frac{\mathrm{d}T}{\mathrm{d}\tilde{t}}\frac{1}{TZ}(1+BT), \tag{10.30}$$

where $\tilde{t} = t/\lambda_p$. Equation (10.30) is integrated and then rearranged to finally obtain

$$\frac{S}{S_0} = \left(\frac{1+B(1+T)}{1+B(1+T_0)}\right)^{B/(B+1)}\left(\frac{T}{T_0}\right)^{1/(B+1)}. \tag{10.31}$$

For the purpose of validation, a geometry analogous to chapter 10.6.1 is chosen, i.e., a domain of size $L_s \times L_t \times H = 2\Delta x \times 2\Delta x \times 62\Delta x$ with the same boundary conditions as in the steady shear flow setup. However, at $\tilde{t} = 10$ the wall velocity $\boldsymbol{u}_w$ is instantaneously set to zero. Slightly different simulation parameters than in [23] are chosen here to demonstrate a more interesting switch-on behavior: in fig. 10.17a the normalized stress component τ_{xy} of the simulation is shown; when the shear flow is switched on, overshooting occurs to some extent, reaching its maximum amplitude at intermediate values $b \approx 10$. Now eq. (10.28) is numerically solved for

each parametric set separately to obtain $\mathrm{Tr}(\underline{\tau}_p)_{\mathrm{num}}$ and plotted against the simulation results. In the same plot, $S = S(\mathrm{Tr}(\underline{\tau}_p)_{\mathrm{num}}/(3nk_{\mathrm{B}}\theta))$ is also solved via eq. (10.31) and compared with the simulation outcome. In fig. 10.17b very good agreement between theory and simulation is obtained. As expected, the normalized curves for $S(t)$ and $T(t)$ differ only for small b. For large b, on the other hand, the transition to the Oldroyd-B limit can be observed.



(a) (b)

Fig. 10.17: Validation via cessation of steady shear flow. The wall velocity is switched off at time $\tilde{t} = 10$. In (a), normalized simulation data for τ_{xy} is shown. In (b), the numerical expectation for S (solid) and T (dashed) is compared with the simulation data (S crosses, T circles).

10.7 Validation of capsule in Newtonian fluid

The IBM was transferred from ESPResSo to FluidX3D by Moritz Lehmann, but not validated yet concerning the capsule model. The ESPResSo version itself has been validated extensively in [35, ch. 4.4] together with their *boundary integral method* (BIM) implementation against various other methods (projection methods combined with IBM or BIM). The validation was carried out for a capsule of radius R in shear flow for various ratios $\hat{\kappa}_B = \frac{\kappa_B}{R^2\kappa_S}$ between shear and bending resistance (including the case $\hat{\kappa}_B = 0$) and in a range $0.005 \leq \mathrm{Ca} \leq 0.2$, where the capillary number is defined as $\mathrm{Ca} = \frac{\eta\dot{\gamma}R}{\kappa_S}$.

This ESPResSo implementation will now be used to validate the FluidX3D implementation. An initially spherical capsule of radius R placed in a resting fluid is considered. Moving bounce-back walls at $z = 0$ and $z = H$ instantaneously start moving against each other in the x-plane with $|u_w| = \dot{\gamma}H/2$ at time $t^* = 0$, which lead to a linear shear flow with shear rate $\dot{\gamma}$. This is in contrast to the validations in [35], where the shear flow is already fully developed at $t^* = 0$. The initial sphere is taken as the reference state for the in-plane tensions. For the capsule, the neo-Hookean model is used. The Reynolds number $\mathrm{Re} = \frac{R|u_w|}{\nu}$ is much smaller than one. Capsules with two different resolutions are considered: the capsule with $R = 6\Delta x$ is resolved by $n_\Delta = 1280$ triangles, the one with $R = 13.5\Delta x$ has $n_\Delta = 5120$. A simulation domain of size $L_s \times L_t \times H = 10R \times 5R \times 15R$ is chosen. It is important that the capsule is placed with its center of mass exactly in between the two walls. Misplacing it by only half a lattice node from the channel center causes the capsule to drift off during the simulation. Furthermore, D3Q19 and MRT are chosen for the LBM simulation. Using TRT makes an observable difference, as is

documented in fig. A2.

As a benchmark, three quantities will be investigated. For not too large shear rates the shape of the capsule becomes approximately an ellipsoid. This can be described by the Taylor deformation parameter $D = \frac{a-c}{a+c}$, where a and c denote the major and minor semi axes, respectively. The ellipsoid is inclined towards flow direction, which is measured by the inclination angle ϕ_{incl} between the flow direction (x-axis) and the major axis of the ellipsoid. Furthermore, the membrane rotates around the capsules centroid, which is called "tank treading" motion. An illustration of the considered quantities is given in fig. 10.18. From the capsule mesh the inertia tensor is computed by [71, eq. 3.9]. The diameters of the semi-axes are calculated by its eigenvalues, their orientation by its respective eigenvectors [33, 35]. To compute the rotation angle $\phi_{\text{rot}}(t)$ of the capsule, the capsule at $t^* = 0$ is taken as a reference. All points making up the capsule are first projected in the plane of rotation. Then the vector $\boldsymbol{v}_i$ from the capsule mid towards each point is computed. The angle between $\boldsymbol{v}_i$ and the reference vector $\boldsymbol{v}_{0,i}$ is computed via

$$\text{atan2}(|v_i \times v_{0,i}|, |v_i \cdot v_{0,i}|) \tag{10.32}$$

with the `atan2` definition as in C++. Taking the mean over all points results in smooth curves for ϕ_{rot}. The reference capsule is always a sphere, but the capsule deforms later in shear flow, which is reflected in ϕ_{rot} not fully covering the range $[0, \pi]$. The dimensionless frequency $\omega/\dot{\gamma}$ is extracted from ϕ_{rot} by averaging over the distances of its maxima and minima.

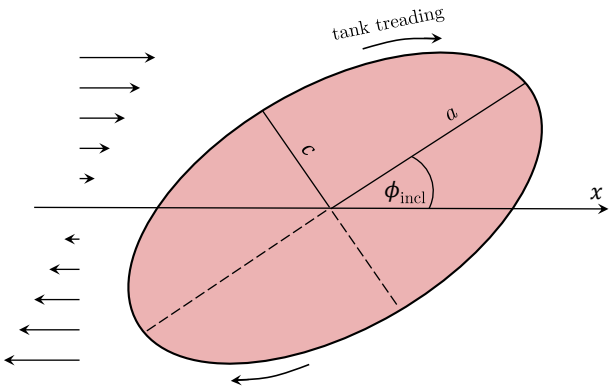
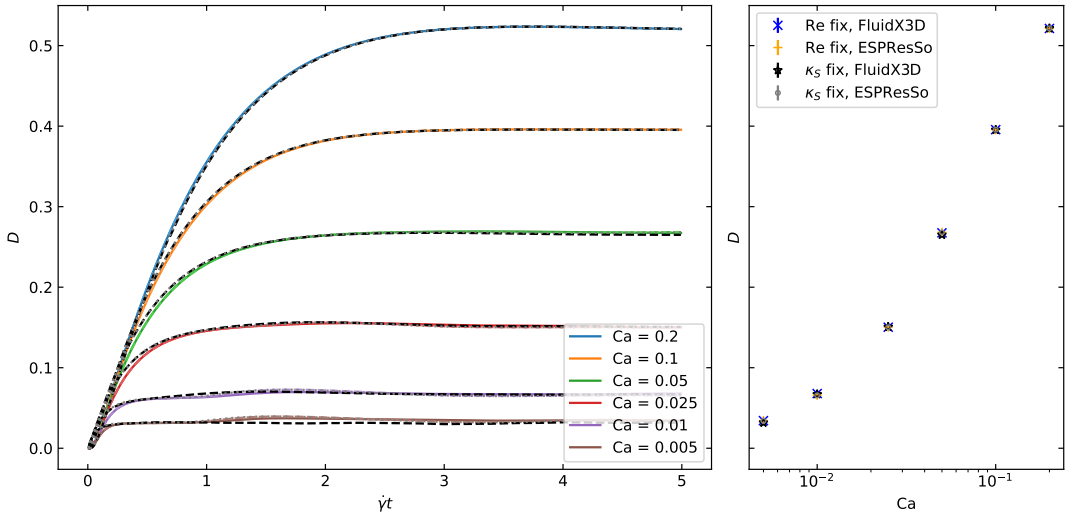


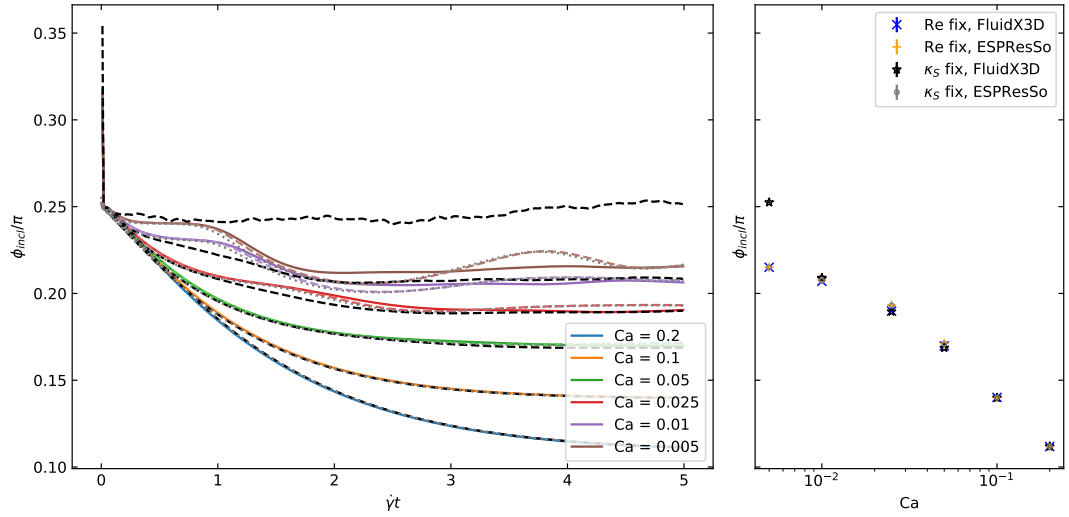
Fig. 10.18: 2D illustration of the capsule in linear shear flow. The initially spherical capsule deforms to an ellipsoid. The Taylor deformation parameter D is calculated via the major and minor semi axes a and c, respectively. ϕ_{incl} denotes the inclination towards streaming direction. The membrane performs a "tank treading" motion around the capsule's centroid.

First, Ca is varied by holding κ_S fixed and changing $\dot{\gamma}$. This yields very good agreement of D between ESPResSo and FluidX3D. After testing the parameter space of $\dot{\gamma}$, the validation strategy is switched by holding $\dot{\gamma}$ fixed and varying κ_S. This has the effect that by fixing $\dot{\gamma}$, Re is also fixed. More importantly, it now holds $\dot{\gamma}\Delta t = \text{const}$, so the number of simulation steps is independent of Ca. $\text{Re} = 0.05$ is chosen, which results in $\dot{\gamma}\Delta t \approx 5.9 \times 10^{-5}$. A comparison of the two strategies for FluidX3D and ESPResSo is done in fig. 10.19. By looking at the Taylor deformation D one can see that both strategies yield the same steady-state values, but result in slightly different behavior at inception of shear (since the evolution to steady state scales with Re). Furthermore, in ESPResSo both strategies show small oscillations of D for low Ca, which

can also be seen in the BIM simulations of [35, fig. 12]. In FluidX3D these oscillations only show up for fixed Re, and are still poorer resolved than in ESPResSo. The good match of both strategies for ESPResSo / the mismatch in FluidX3D for low Ca can even better be obtained for the inclination angle. Finally, in FluidX3D for Ca = 0.005 the fixed κ_S strategy seems to fail concerning the inclination angle.



(a) Taylor deformation D



(b) Inclination angle ϕ_{incl}

Fig. 10.19: Capsule of $R = 13.5\Delta x$ in Newtonian shear flow. Comparison of the two strategies fixed Re and fixed κ_S for FluidX3D and ESPResSo. On the left in (a) and (b), the lines have the following meanings: fixed Re + FluidX3D (solid colored), fixed Re + ESPResSo (dashed colored), fixed κ_S + FluidX3D (dashed black), fixed κ_S + ESPResSo (dotted gray). The steady state values averaged starting from $t^* = 4.5$ are shown on the right.

For all future simulations in shear flow the strategy of fixed Re is applied. Next, the capsule of radius $R = 13.5\Delta x$ (high resolution) is compared with a capsule of $R = 6\Delta x$ (low resolution) (cf. fig. 10.20). For both softwares, the small oscillations of D for small Ca disappear completely for

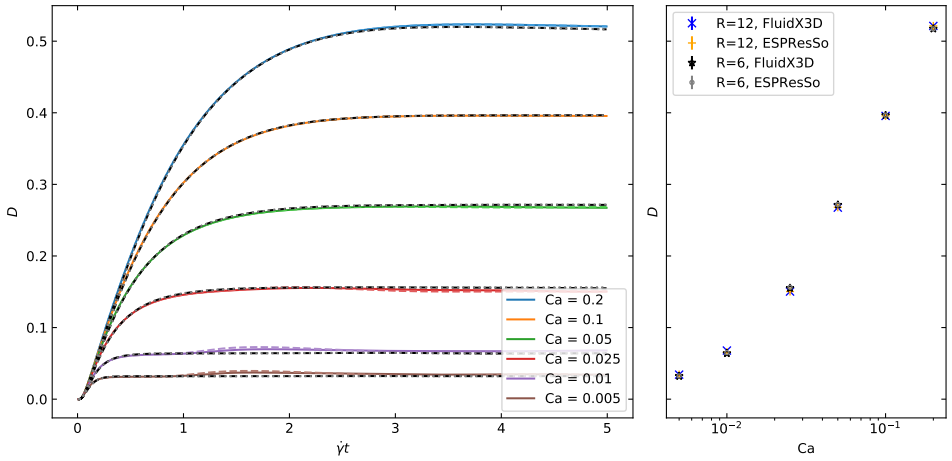
$R = 6\Delta x$, but the steady state values of D remain nearly the same as in the case $R = 13.5\Delta x$. The inclination angle ϕ_{incl} is shown for a much longer time now. It depends on resolution much stronger than D, but still the effect of decreasing ϕ_{incl} for increasing Ca is captured well. Furthermore, for low Ca one can see periodic patterns in ϕ_{incl}. They are much more pronounced in ESPResSo than in FluidX3D, and have a bigger amplitude for higher resolution. Finally, the rotation frequency has a weak dependency on resolution, but this dependency is nearly equal for ESPResSo and FluidX3D. Overall, the differences caused by resolution are much more severe than the differences between the two softwares. The differences between the softwares might be due to floating-accuracy: while ESPResSo uses `double` for the IBM, FluidX3D normally uses `float` (except in chapter 10.10.3).

The periodic patterns observed in fig. 10.20 are investigated in more detail. The frequency of the pattern seems to be proportional to the frequency of the rotation of the capsule: smaller Ca is associated with a higher frequency. Looking at the videos of the capsule meshes, the patterns are caused by tiny wrinkles on the capsule surface. In ESPResSo they have a stable shape and rotate along with the capsule mesh itself, while in FluidX3D they appear and disappear seemingly at randomly. For further comparison, a BIM simulation was provided by Katharina Gräßel. Here, differently than in the LBM the capsule is in infinite shear flow. The results for $\text{Ca} \in \{0.005, 0.025, 0.05\}$ are shown in fig. A3. One can see that with respect to the amplitude of the pattern, BIM is somewhere between ESPResSo and FluidX3D. Secondly, especially for $\text{Ca} = 0.025$ one can see how the periodicity matches with the one from ESPResSo. Katharina Gräßel did some further investigations showing that the concrete pattern strongly depends on the initial/reference shape of the capsule[13]: slightly rotating the capsule before starting the simulations results in different patterns. Because already so many discrepancies exist between BIM and ESPResSo simulations and also within ESPResSo simulations, not being able to match the periodic patterns between FluidX3D and ESPResSo doesn't seem to be a problem.

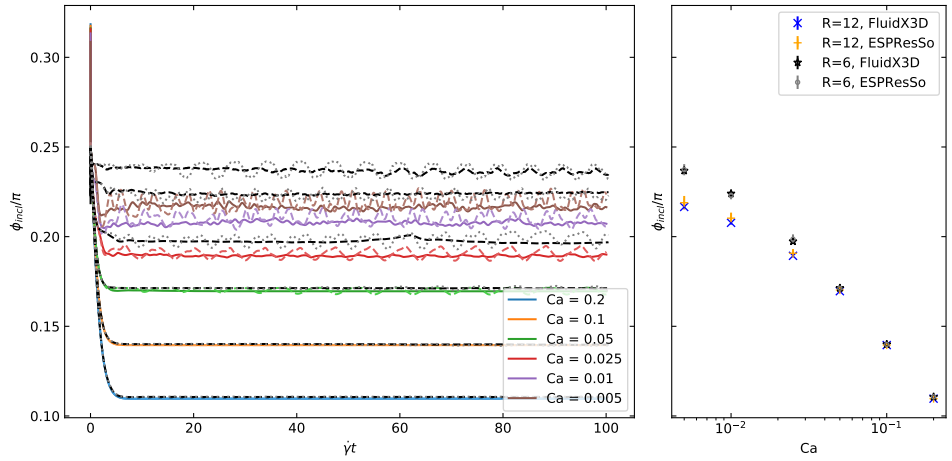
As a next step, the implementation of the *InOut* algorithm in FluidX3D shall be validated. This is done by using the algorithm to implement a viscosity contrast $\Lambda = \eta_{\text{in}}/\eta_{\text{out}}$ between the interior and exterior fluid of the capsule. The ESPResSo version was validated in [39] by looking at the center of mass displacement of red blood cells (RBCs) in a rectangular channel and comparing it to the results of BIM solutions. Here, a more dynamic validation is chosen, where the Taylor deformation D of a capsule with $\Lambda = 5$ is investigated. This validation was first done by [72, fig. 5] (IBM + thin-shell model), then confirmed in [73, fig. 11] (again IBM + thin-shell model) and [74, fig. 7] (immersed-finite-element method + FVM based incompressible fluid solver). They all use the neo-Hookean constitutive equation for the capsule and set the bending energy to zero.

[73] use 7776 Catmull-Clark subdivision elements for the mesh representation of the capsule (generated via the Catmull-Clark subdivision scheme). Aiming for a similar resolution, at first a refinement with $n_\Delta = 5120$ triangles is used in this thesis. The domain should be chosen large enough so that boundary effects become negligible. A size of $L_s \times L_t \times H = 10R \times 5R \times 10R$ is chosen, which is very close to the cube of side length $10R$ [73] use. A smaller domain in traverse-flow direction should not affect the simulation much. One can see in fig. 10.21a that the *InOut* implementation of FluidX3D can qualitatively reproduce the increasing overshoot of the deformation parameter for increasing Ca. However, the absolute deviation from the data

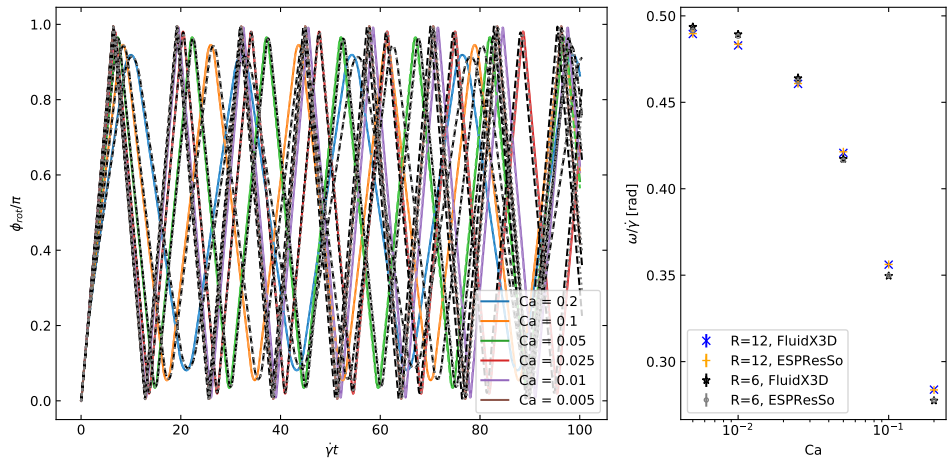
[13] Unpublished results.



(a) Taylor deformation



(b) Inclination



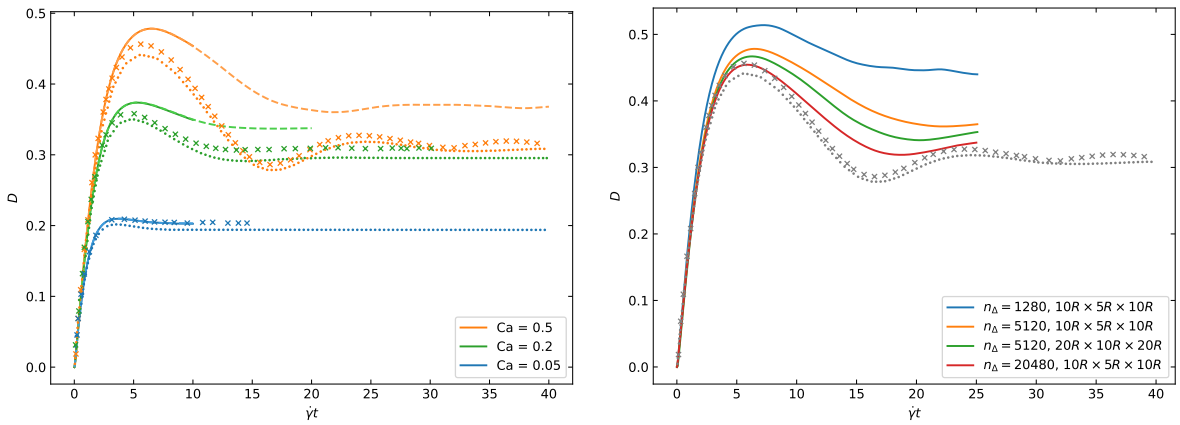
(c) Rotation

Fig. 10.20: Capsule in Newtonian shear flow with Re = 0.05. Comparison of two resolutions $R = 13.5\Delta x$ and $R = 6\Delta x$ for FluidX3D and ESPResSo. On the left in (a), (b) and (c) the lines have the following meaning: $R = 13.5\Delta x$ + FluidX3D (solid colored), $R = 13.5\Delta x$ + ESPResSo (dashed colored), $R = 6\Delta x$ + FluidX3D (dashed black), $R = 6\Delta x$ + ESPResSo (dotted gray). For (a) and (b), the steady state values averaged from $t^* = 4.5$ are shown on the right. For (c), on the right the rotation frequency was determined by reading the maxima and minima of ϕ_{rot}.

from [73] increases with increasing Ca - not only for the overshoot, but also for the steady state $\dot{\gamma}t \to \infty$. However, a comparison with the ESPResSo implementation (dashed) shows that ESPResSo and FluidX3D match perfectly. A bug in the new, parallelized *InOut* implementation in FluidX3D is therefore unlikely. Nevertheless, it was visually checked whether the LBM nodes were correctly marked as inside/outside. Additionally, a BIM simulation with $n_\Delta = 5120$ is used (performed by Katharina Gräßel). The outcome agrees well with the results from [73]. Two reasons for the deviation of the two LBM simulations are most likely: first, the domain could have been chosen too small, so that boundary effects play a role. Secondly, the discretization could be not fine enough. Since only discrete LBM nodes are marked as inside/outside during simulation, e.g. the staircase effect could have too great an influence. To confirm this theory, in fig. 10.21b simulations at $\mathrm{Ca} = 0.5$ for different domain sizes and capsule mesh refinements have been conducted. As the number of triangles n_Δ grows, the grid refinement of the LBM also increases, as listed in tab. 10.3. Fig. 10.21b suggests, that for even finer resolution and bigger domain size, the LBM solution converges into the true solution. However, such a resolution is no longer practically computable. Instead, VoF methods could be used to circumvent the staircase effect, e.g. similar to the work of [27] in the context of the simulation of charged particles. However, this is beyond the scope of this work.

n_Δ	1280	5120	20480
$R/\Delta x$	6	12	25

Tab. 10.3: LBM grid resolution used for the different capsule mesh refinement shown in fig. 10.21.



(a) (b)

Fig. 10.21: Time evolution of the Taylor deformation parameter for an initially spherical capsule in shear flow with $\Lambda = 5$ and $\hat{\kappa}_B = 0$. The solid lines are the results obtained by the FluidX3D implementation and the dashed lines are obtained by ESPResSo. The dots indicate the results found by BIM simulations and the crosses indicate the data taken from [73, fig. 11]. In (a), the resolution $n_\Delta = 5120$ in a domain of $10R \times 5R \times 10R$ was used both for FluidX3D and ESPResSo. In (b), all simulations were taken at $\mathrm{Ca} = 0.5$.

10.8 Advection

10.8.1 The CTU advection scheme

In chapter 10.3 the treatment of the flux term $\frac{\partial}{\partial r_k} J_{ijk}(\boldsymbol{r}, t)$ via the FVM was shown. However, especially in the context of high stress contrasts, a checkerboard instability can arise, as visualized in the next chapter. Therefore, an alternative advection scheme called the *corner-transport upwind* (CTU) scheme is presented. It's basic description can be found in [75], the concrete implementation is taken from [69] as used in the software *pystencils* [76].

In the CTU scheme, the grid is assumed to be made up from cubic cells (squares in 2D). The flux $J_Q^{(\text{out})}(\boldsymbol{r}, t)$ of a quantity $Q(\boldsymbol{r}, t)$ out of the cell at position $\boldsymbol{r}$ and time t is computed by virtually displacing the cell by the velocity $\boldsymbol{u}(\boldsymbol{r}, t)$ and computing the overlap with the neighboring cubes (cf. fig. 10.22). The full scheme is therefore defined on a D3Q27 neighborhood (D2Q9 in 2D). To check if a cell at position $\boldsymbol{r}$ displaced by $\boldsymbol{u}(\boldsymbol{r}, t)$ overlaps with a neighboring cell at position $\boldsymbol{r} + \boldsymbol{c}_i \Delta t$, it must hold $\alpha(\boldsymbol{r}, \boldsymbol{c}_i, t) = 1$, where:

$$\alpha(\boldsymbol{r}, \boldsymbol{c}_i, t) \begin{cases} 1, & (u_k(\boldsymbol{r}, t) c_{ik} > 0) \vee (c_{ik} = 0) \text{ for all } k \in \{1, \ldots, d\}, \\ 0, & \text{else.} \end{cases} \tag{10.33}$$

In the condition above no Einstein summation applies. Furthermore, the actual overlap volume $V^{(\text{out})}$ is computed via

$$V^{(\text{out})}(\boldsymbol{r}, \boldsymbol{c}_i, t) = \alpha(\boldsymbol{r}, \boldsymbol{c}_i, t) \prod_{k=1}^{d} l_k(\boldsymbol{r}, \boldsymbol{c}_i, t), \quad \text{where} \quad l_k(\boldsymbol{r}, \boldsymbol{c}_i, t) = \begin{cases} \Delta x - u_k(\boldsymbol{r}, t)\Delta t, & \text{if } c_{ik} = 0 \\ u_k(\boldsymbol{r}, t)\Delta t, & \text{else.} \end{cases} \tag{10.34}$$

Finally, $J_Q^{(\text{out})}(\boldsymbol{r}, t)$ is obtained via

$$J_Q^{(\text{out})}(\boldsymbol{r}, t) = \sum_{i=1}^{q-1} J_{Q,i}^{(\text{out})}(\boldsymbol{r}, t), \quad \text{with} \quad J_{Q,i}^{(\text{out})}(\boldsymbol{r}, t) = V^{(\text{out})}(\boldsymbol{r}, \boldsymbol{c}_i, t) Q(\boldsymbol{r}, t). \tag{10.35}$$

To ensure conservation, the incoming flux $J_Q^{(\text{in})}(\boldsymbol{r}, t)$ is defined by the overlap of the cell with the neighboring cells displaced by their respective velocities

$$J_Q^{(\text{in})}(\boldsymbol{r}, t) = \sum_{i=1}^{q-1} J_{Q,i}^{(\text{in})}(\boldsymbol{r}, t), \tag{10.36}$$

where $J_{Q,i}^{(\text{in})}(\boldsymbol{r}, t) = J_{Q,i}^{(\text{out})}(\boldsymbol{r} - \boldsymbol{c}_i \Delta t, t)$. The total flux $-J_Q(\boldsymbol{r}, t) = -J_Q^{(\text{out})}(\boldsymbol{r}, t) + J_Q^{(\text{in})}(\boldsymbol{r}, t)$ finally replaces for $Q(\boldsymbol{r}, t) \equiv \tau_{ij}(\boldsymbol{r}, t)$ the first term on the right hand side of eq. (10.8).

Note that $J_Q^{(\text{out})}(\boldsymbol{r}, t)$ only depends on values at position $\boldsymbol{r}$, while $J_Q^{(\text{in})}(\boldsymbol{r}, t)$ only depends on values of neighboring cells. This is basically the reason why the checkerboard instability is prevented. Indeed, the CTU method is stable for Courant numbers up to 1, which means each cell is not allowed to shift more than one grid point, i.e. $|\boldsymbol{u}|\Delta t < \Delta x$. Furthermore, since $|\boldsymbol{u}|\Delta t \ll \Delta x$ in typical LBM simulations, the flux $J_{Q,i}^{(\text{out})}$ caused by the displacement is $\propto |\boldsymbol{u}|$ for face neighbors (edge in 2D), $\propto |\boldsymbol{u}|^2$ for edge neighbors (corner in 2D) and $\propto |\boldsymbol{u}|^3$ for corner neighbors. Accordingly, using the CTU scheme on a reduced neighborhood incurs an error compared to the full scheme, but has the advantage of increased performance. The error is

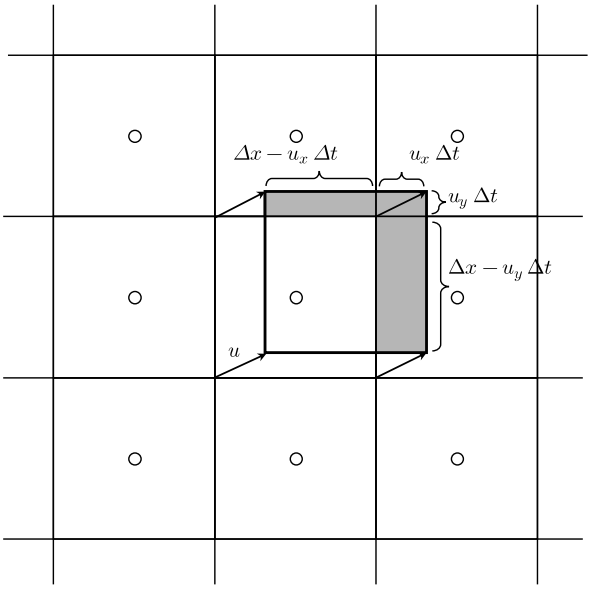


Fig. 10.22: Illustration of the CTU scheme in case of a D2Q9 neighborhood. The grid nodes are marked as circles, the $\boldsymbol{c}_i$ are found as the vectors connecting neighboring nodes. $J_Q^{(\mathrm{out})}(\boldsymbol{r},t)$ (gray region) is constructed by virtually displacing a cube (square in 2D) by node velocity $\boldsymbol{u}(\boldsymbol{r},t)$ and computing the overlap volume with the neighboring cubes. The flux to face neighbors (edge in 2D) is of first order in $\boldsymbol{u}$, to edge neighbors (corner in 2D) it is of second order, to corner neighbors of third order. $J_Q^{(\mathrm{out})}(\boldsymbol{r},t)$ is at the same time the contribution of cell $\boldsymbol{r}$ to $J_Q^{(\mathrm{in})}$ of the neighboring cells.

of third order in $|\boldsymbol{u}|$ for D3Q19 (no corner neighbors) and of second order for D3Q7 (no edge neighbors).

Boundaries with $J(\boldsymbol{r}_w) \equiv 0$ can be imposed by just skipping respective neighbors in the summations appearing in eq. (10.35) and (10.36).

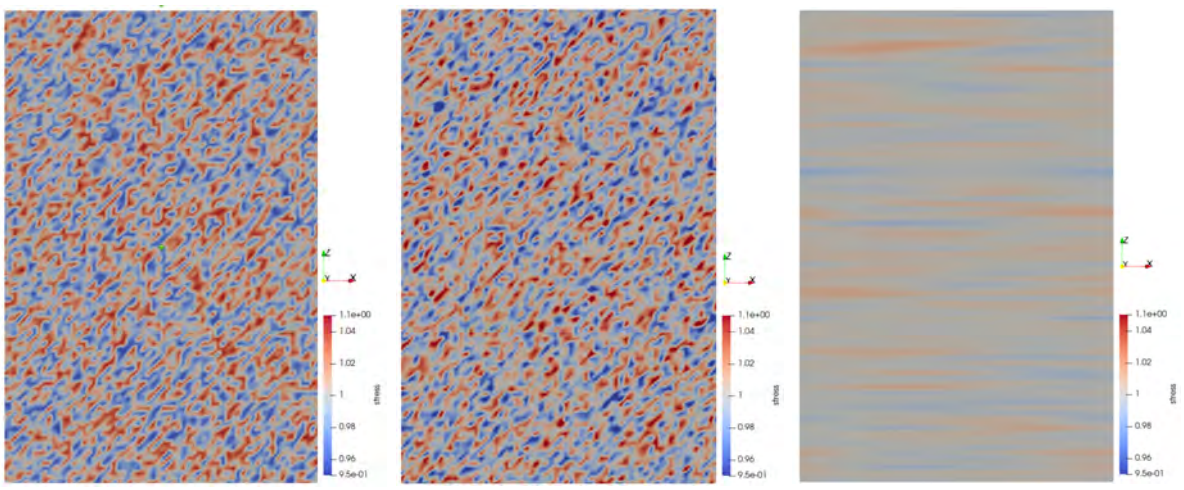
10.8.2 Conservation tests, checkerboard effect and staircase effect

A small, but systematic loss of the advected quantity is observed in the tests of the following chapter, where the two advection schemes **CTU**/**simple** are combined with the volume tracking algorithm of capsules. In this chapter it will be shown that this loss is already observable for the pure advection algorithms, i.e., is not a bug in the algorithms of chapter 10.8.3.1. During this chapter, the constitutive equation and the force term due to viscoelasticity F_{ext}^p is switched off. Furthermore, for simplicity, a scalar quantity τ instead of the full stress tensor $\underline{\tau}_p$ is used. τ can be thought of simply as ink that is advected with the flow and $\tau_{\mathrm{ges}} = \sum_{\boldsymbol{r}\in \mathrm{grid}} \tau(\boldsymbol{r})$ should therefore be conserved.

For the first two tests, the velocities are not computed by the LBM. Instead, the constant velocity $u_x = 0.001$, $u_y = u_z = 0$ is given. The simulation domain has the sizes $L_x \times L_y \times L_z = 64\Delta x \times 32\Delta x \times 96\Delta x$ and is periodic in all directions. For simplicity, the D3Q7 neighborhood is chosen for advection.

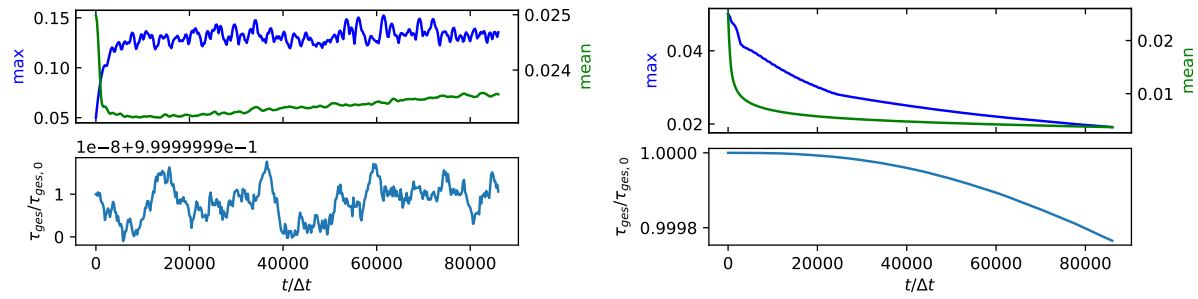
For the first test, τ is initialized to $\tau(\boldsymbol{r}) \in [0.95, 1.05]$, with the actual value assigned randomly, equally distributed and independently for each node (cf. fig. 10.23a). Then, as described above,

only advection is performed at each time step using the given constant velocity field. For calculations with `float` accuracy, the following results are found: for **simple**, the average size of the perturbation is preserved. Moreover, τ_{ges} is conserved except for rounding errors. **CTU**, on the other hand, causes a smearing of the perturbations over time, which is probably due to numerical diffusion. τ_{ges} is not preserved, but decreases systematically (cf. fig. 10.23 and 10.24).



(a) initialization (b) **simple** (c) **CTU**

Fig. 10.23: Cut through setup at $y = L_y/2$. In (a), the randomized initialization of τ at time $t = 0\Delta t$ is shown. In (b) and (c), the configuration of τ at time $t = 80000\Delta t$ is shown for **simple** and **CTU**, respectively. A random initialization around 1 ± 0.05 and a constant fluid velocity $u_x = 0.001$, $u_y = u_z = 0$ is used in a domain of size $L_x \times L_y \times L_z = 64\Delta x \times 32\Delta x \times 96\Delta x$.



(a) **simple** (b) **CTU**

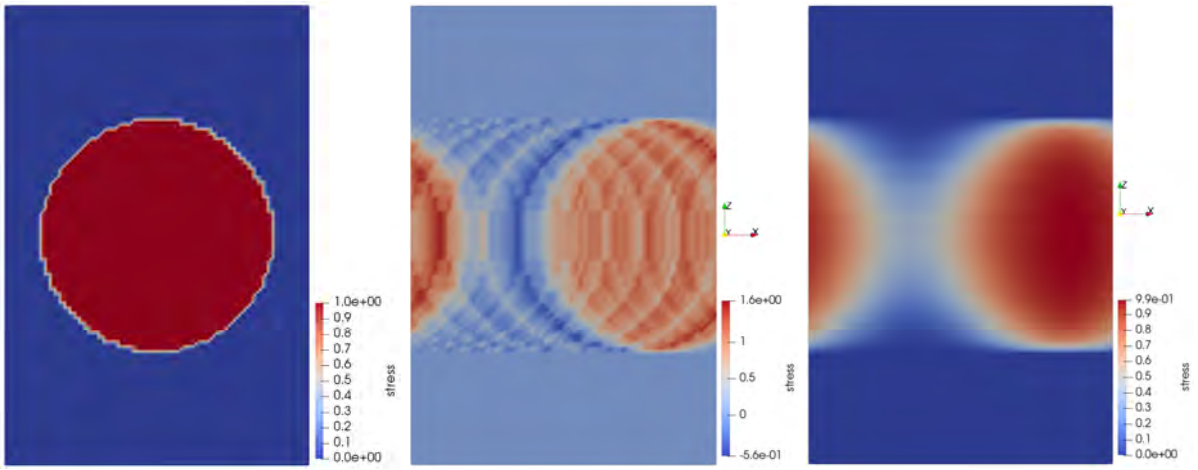
Fig. 10.24: Maximum and mean deviation of all nodes from 1 (top) and conservation of τ_{ges} (bottom) for the two advection algorithms. The same setup as in fig. 10.23 is used.

In appendix A1 further tests are listed that rule out a bug in the **CTU** implementation. Finally, it is shown there that when using a buffer with `double`-precision, also for the **CTU** scheme only numerical fluctuations in $\tau_{\text{ges}}(t)$ occur (i.e. a behavior analogous to fig. 10.24a, lower figure, only with significantly smaller amplitude). Nevertheless, in this thesis a buffer with `float`-precision is continued to be used - in practice, the loss of τ_{ges} compared to the source terms in the constitutive equation is negligible.

The numerical diffusion of the advection algorithms will be investigated in a second test. In

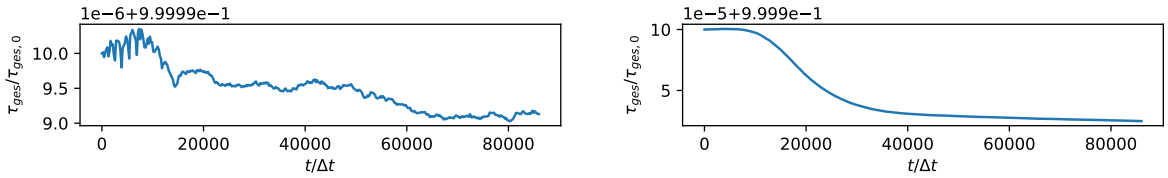
fig. 10.23 it looks like diffusion dominates advection for **CTU**, while **simple** seems to qualitatively preserve the pattern given at initialization. In practice, however, high stress gradients occur especially near the capsule boundary. Therefore, the exact same test as above is carried out for a modified initialization: now a spherical region around the center of the domain with radius $R = L_z/4$ is initialized with one, the rest with zero. With this, the test is already much closer to geometries appearing in capsule simulations.

In fig. 10.25, the results at time $t = 80000\Delta t$ are visualized. For both algorithms, the sphere has already completely crossed the domain once and is in the process of leaving the domain again on the right side. It is immediately noticeable that **simple** has perturbations with certain wavelengths and a checkerboard pattern. Moreover, the range of values has increased from the initial $[0, 1]$ to $[-0.6, 1.6]$. **CTU**, on the other hand, preserves the initial shape and range of values very well. Fig. 10.26 shows that the conservation error for **simple** has now increased, possibly because it is now calculated with an unfavorable range of values. For **CTU**, on the other hand, the error has decreased compared to the random initialization, possibly because contiguous regions with a similar value range have an advantageous effect.



(a) initialization (b) **simple** (c) **CTU**

Fig. 10.25: Like fig. 10.23, but for a dot initialization. *ParaView* interpolates the colors of the single grid points in (a) - in the actual initialization only the values 0 and 1 are used. In (b) and (c), the dot has already crossed the whole domain once and is at the point of crossing it a second time.



(a) simple (b) ctu

Fig. 10.26: Conservation of τ_{ges} for the two advection algorithms. The same setup as in fig. 10.25 is used.

In a third test, it will be examined how the advection algorithms cope with bounce-back boundaries. For this purpose, the velocities are no longer prescribed constantly, but calculated with the LBM. The LBM part of the simulation is calculated with D3Q27 and TRT, so that

discretization effects are mitigated in the LBM part. Doing so, possible discretization effects caused by the different velocity sets used for the advection algorithms should be independent of the LBM. Three setups are defined, which are used here and also in the following chapter for the validation of the advection. The shear flow setup **S1** is already known from chapter 10.6.1 and is used this time with $L_s \times L_t \times H = 64\Delta x \times 32\Delta x \times 94\Delta x$. The planar poiseuille channel setup **S2** has already been used in chapter 10.4.2. This time the no-slip bounce-back boundaries with $\boldsymbol{u}((x, y, 0)^{\mathrm{T}}, t) = \boldsymbol{u}((x, y, H)^{\mathrm{T}}, t) = 0$ are in the z-plane, the homogeneous body force $\boldsymbol{F} = F_x \hat{\boldsymbol{e}}_x$ points in x-direction. The domain size is chosen to be $L_s \times L_t \times H = 64\Delta x \times 32\Delta x \times 94\Delta x$. The unaligned poiseuille channel setup **S3** is used as a further setup. It consists of a periodic cylindrical channel aligned along the vector $(1, 1, 1)^{\mathrm{T}}$ (cf. fig. 10.27). The volume force is also parallel to this direction. The whole simulation domain is a cube with side length $L = 96\Delta x$. This setup is interesting for two reasons: firstly, the walls are constructed from staircases for discretization reasons, and secondly, the flow direction is not along a major axis.

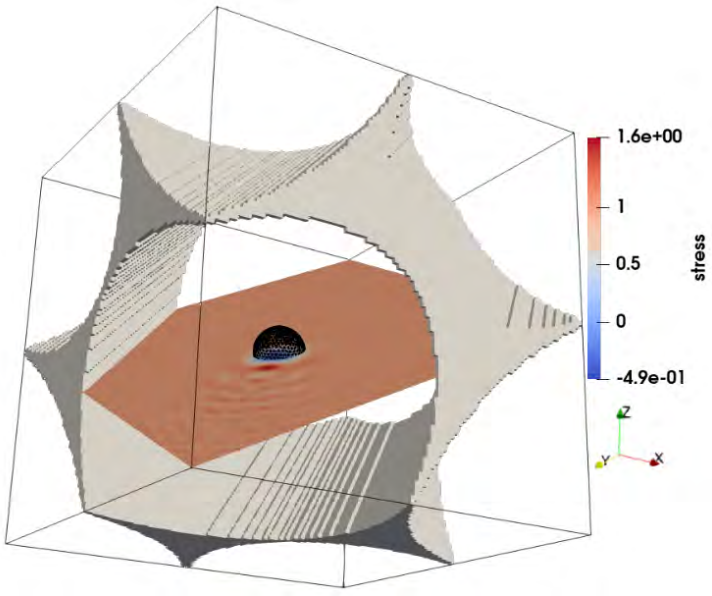
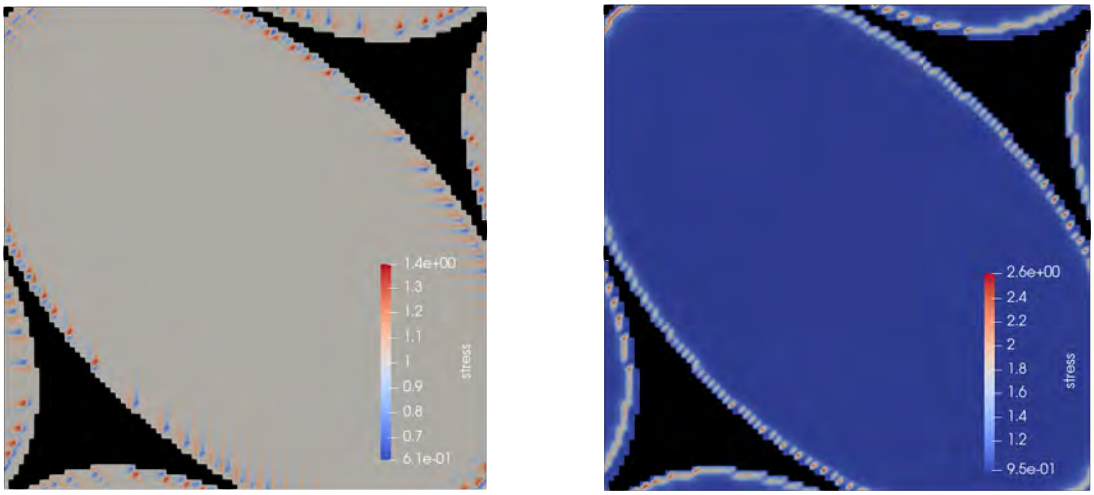


Fig. 10.27: Unaligned Poiseuille Setup S3. Wall nodes gray, cell grid black, section through stress field in colors blue to red (see text for meaning of stress field in advection tests). Image taken at $t^* = 20$, with the algorithm **simple** + M0 (cf. chapter 10.8.3.1) and advection neighborhood D3Q27. The additional notches at the edges of the domain are needed in order to obtain a fully periodic channel.

In order to have a definition of the dimensionless time $t^* = t\dot{\gamma}$, for setup S2 and S3 the typical shear rate $\dot{\gamma}$ is approximated as $\dot{\gamma} = \frac{2|u_{\mathrm{center}}|}{H}$, where u_{center} denotes the center velocity for a simulation without capsule. For S1 it holds $\dot{\gamma} = \frac{2|u_w|}{H}$ with wall velocity u_w.

For this third test, the stress is homogeneously initialized to one; no capsule is present in the flow. Just like the tests before, the constitutive equation is switched off and there is no force resulting from the stress, instead there is only advection. The LBM simulation is therefore completely independent of the stress (ink) τ. A good advection algorithm should give a homogeneous stress over time (i.e. continue to be one everywhere). When testing both advection algorithms using the setups S1 and S2, τ remains exactly one for all times and τ_{ges} is conserved exactly. The reason for this is the high symmetry of the flow and the planar walls. For S3, on the other hand,

the following is observed: Both for **simple** and **CTU**, stress conservation is violated on similar orders of magnitude as in the tests without walls. Concerning the homogeneous distribution of stress, the **CTU** scheme has problems with the staircase effect near the walls. This does not improve with a larger neighborhood (cf. fig. 10.28b and 10.29b for D3Q27, fig. A5b for D3Q7.). The **simple** scheme, on the other hand, already copes well with the D3Q7 scheme: at $t^* = 20$, an error of only ± 0.4 has formed near the walls (cf. fig. 10.28a and 10.29a), with increasing resolution it gets even better (D3Q19: ± 0.1, D3Q27: ± 0.1, cf. fig. A5a). It is also interesting to note that the deviation caused by **simple** is symmetrical around one (reflecting the symmetry of the algorithm itself), while **CTU** mainly has values greater than one. Why **CTU** already performs significantly worse than **simple** in D3Q7 is not clear. The fact that **simple** improves with D3Q27 compared to D3Q7, but **CTU** does not, can be explained as follows: For fluid nodes next to the wall, the flow to many face-neighbors is blocked by the wall itself. Therefore, the flow to edge-neighbors becomes more important. For **simple** these flows are first order in $|u|$, but for **CTU** they are second order as explained above. Therefore, for **simple** D3Q19 brings a clear improvement over D3Q7, while for **CTU** there is no noticeable improvement. However, if one wants to keep the stability advantages of an upwind scheme, there are two possibilities: first, the stress accumulation on walls due to the staircase effect could be bypassed analogously to chapter 10.8.3 in the improvement from method M3 to M4. In other words, advection of stress into the first layer of wall nodes would be allowed, but directly afterwards the stress inside all wall nodes would be re-distributed to nearby fluid nodes. Secondly, an upwind scheme could be used that includes first-order flows for edge and corner neighbors (e.g. by starting at **simple** with D3Q27 and applying the first-order upwind scheme [77] to it). These tasks are left as future work.



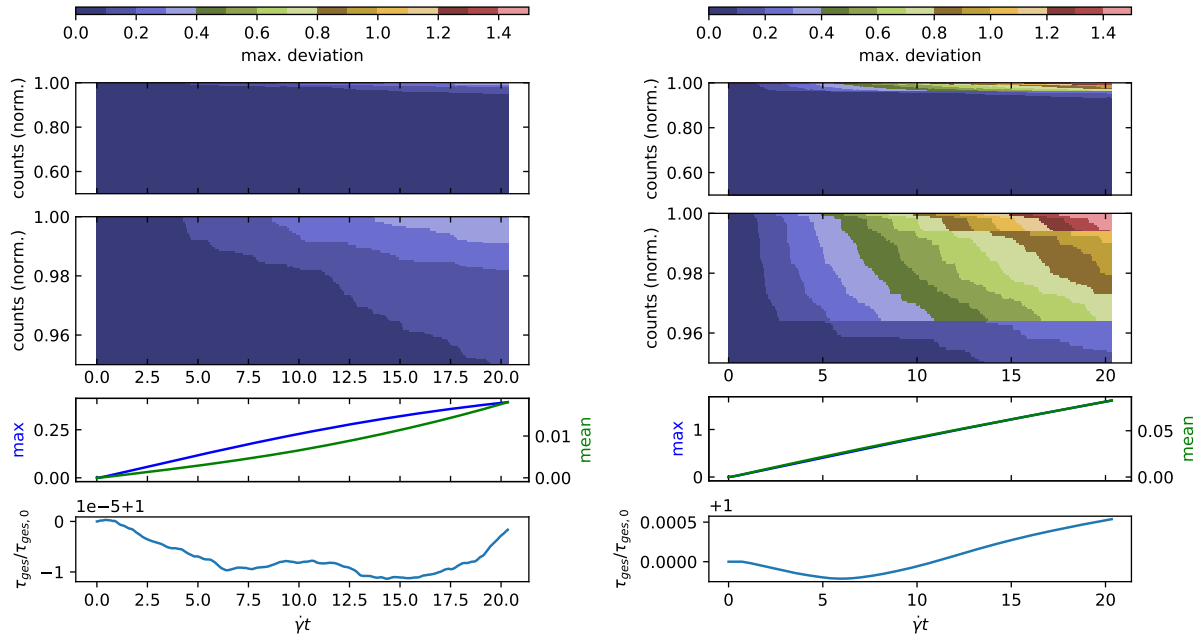
(a) **simple** + D3Q7

(b) **CTU** + D3Q27

Fig. 10.28: Slice at $x = L/2$ through setup S3 (cf. fig. 10.27). Shown is τ at $t^* = 20$ for a simulation without capsule. Wall nodes are colored black.

10.8.3 Advection towards Newtonian capsules in viscoelastic fluids

In this section it will be explained how the basic advection algorithms can be adapted to simulate a capsule with Newtonian interior fluid and viscoelastic exterior fluid. At first the implementation of the algorithms will be explained, then some pure advection tests analogous to chapter 10.8.2



(a) **simple** - D3Q7

(b) **CTU** - D3Q27

Fig. 10.29: Top: stacked and normalized histogram of all fluid nodes, ordered by the amount of their respective deviation from one. Deviation intervals color-coded. Histogram in two versions with different "zoom levels". Mid: maximum and average deviation from one. Bottom: check of conservation of τ_{ges}.

are conducted and finally the behavior of the algorithms in realistic setups is investigated.

10.8.3.1 Algorithms

M0 and M1: reference methods To compare the different algorithms, two reference methods are defined. In both the methods M0 and M1 the whole computational domain is captured by the same constitutive law. Furthermore, M0 doesn't distinguish between interior and exterior fluid at all. M1 however makes the interior fluid Newtonian by artificially setting $\boldsymbol{F}^p_{\text{ext}}(\boldsymbol{r}_I) = 0$ for nodes n_I at a position $\boldsymbol{r}_I$ inside the capsule, while for exterior nodes n_O the stress continues to act on the fluid with the force $\boldsymbol{F}^p_{\text{ext}}(\boldsymbol{r}_O) = \nabla \cdot \underline{\tau}_p$. This is only a rough approximation, as the stress is still advected over the capsule boundary without restriction . Which node is n_I/n_O is marked by the already existing volume tracking algorithm from chapter 7.3.

M2: two parameter sets Looking in literature, [78] is able to simulate a capsule where the inner fluid is Newtonian and the outer fluid is an Oldroyd-B fluid. For the Newtonian part they use the FVM instead of the LBM, and a front-tracking method [71] is used to model the capsule instead of the IBM. They solve the equations of the viscoelastic fluid on the entire computational domain, but for two different sets of parameters for the interior and exterior fluid. Since the fluid properties would vary sharply across the interface, a smoothed indicator function provides an interpolation between the two sets in a small region around the interface. The Newtonian case for the interior fluid is observed by taking the limit $\lambda_p \to 0$. To be able to obtain this limit, the discretization of the constitutive equation has to be done differently so no terms $\propto 1/\lambda_p$ appear. This has been done in [78, eq. 11-14] [79] and leads to the discretization

$$\underline{\tau}_{ij}(\boldsymbol{r}, t + \Delta t) \approx \underline{\tau}_{ij}(\boldsymbol{r}, t) \exp(-\Delta t/\lambda_p) + \underline{K}(\boldsymbol{r}, t)(1 - \exp(-\Delta t/\lambda_p)), \tag{10.37}$$

where

$$\underline{K} = \eta_p \underline{D} - \lambda_p(\boldsymbol{u} \cdot \nabla \underline{\tau}_p - (\nabla \boldsymbol{u})\underline{\tau}_p - \underline{\tau}_p(\nabla \boldsymbol{u})^{\mathrm{T}}). \tag{10.38}$$

This reformulated discretization defined by eq. (10.37) and (10.38) was implemented in FluidX3D, obtaining $\underline{K}$ by the methods outlined in chapter 10.3. Then, the implementation was first tested without a capsule for a single parameter set with $\lambda_p = 0$ in shear flow, where the Newtonian case could be successfully reproduced. If the same test is carried with capsule (again with a single parameter set), after a few time steps regularly arranged "threads" form in the stress field as well as the velocity field. They have a thickness of exactly one lattice node, slowly grow from the walls to the channel center and finally lead to an numerical instability. They have phenomenological similarities to an hourglass instability. This is the case for different Reynolds numbers $\mathrm{Re} \in \{0.5, 0.05\}$ and different capsule resolutions $R \in \{6\Delta x, 12\Delta x\}$.

The method M2 thus seems to be incompatible with IBM. This is particularly strange because IBM and M2 do not interact directly with each other, but only via the fluid velocity field. An explanation for the emergence of the "threads" and the unstable behavior could not be found. Furthermore, it was assured that $\exp(-\Delta t/\lambda_p)$ for $\lambda_p = 0$ was correctly evaluated as 0. Thus, since the problem could not be identified, this method is discarded. Even if the implementation had been successful, there would have been a problem: if polymer stress is advected from a region with parameter set 1 to a region with parameter set 2, the polymer stress is generally not preserved. This is most easily seen when Newtonian parameters ($\lambda_p = 0$) are chosen as parameter set 2: stress advected into this region is simply deleted. Method M3 addresses this problem.

M3: shovel at conversion Method M3 builds upon M1 with the addition, that for all nodes n_I inside the capsule $\underline{\tau}_p(\boldsymbol{r}_I) = 0$ is set. As an optimization regarding the implementation, the constitutive equation for interior nodes isn't evaluated at all. To ensure conservation of stress for the exterior fluid, $\underline{J}(\boldsymbol{r}_I + \boldsymbol{c}_i\Delta t/2) = 0$ is set where $\boldsymbol{r}_I + \boldsymbol{c}_i\Delta t$ is a n_O-node. This idea is analogous to the treatment of bounce-back walls. However, it is not sufficient: Because the capsule boundary moves over time, conversions $n_I \to n_O$ and $n_O \to n_I$ happen.

The flag G_I already provided by the *InOut* algorithm is defined as

$$G_I(\boldsymbol{r}, t) = \begin{cases} 1, & \text{for nodes } n_I, \\ 0, & \text{else,} \end{cases} \tag{10.39}$$

To treat the conversions, a new flag G_C is introduced, which is defined as

$$G_C(\boldsymbol{r}, t) = \begin{cases} 1, & \text{if } G_I(\boldsymbol{r}, t) \neq G_I(\boldsymbol{r}, t - \Delta t), \\ 0, & \text{else,} \end{cases} \tag{10.40}$$

marking if a change of G_I compared to the last time step is applied. No new kernel is needed, the flag can be set during the kernels `ibm_inout_tracking_A` and `ibm_inout_tracking_B`. For each node marked with $G_C = 1$ one also needs to know the number N_f of n_O-neighbors (irrespective if the neighbors themselves are additionally marked with $G_C = 1$). Since this can only happen after the flags G_I and G_C are set, this is done in a separate kernel `count_outer_neighbors`. As an optimization regarding memory G_I, G_C and N_f are saved in a single 8-bit field (for N_f only 5 bits are needed because of $N_f \leq 26$.).

The rest of the algorithm is based on an idea outlined in [69] [14] [27, 80] [15]. The idea is implemented in the kernel `conserve_stress`. The stress tensor is denoted $\underline{\tau}_p$ before kernel `conserve_stress` is called, and $\underline{\tau}_p^*$ afterwards. A node at position $\boldsymbol{r}_f$ with $n_O \to n_I$ distributes its stress to the surrounding N_f n_O-nodes, i.e.:

$$\underline{\tau}_p^*(\boldsymbol{r}_f + \boldsymbol{c}_i\Delta t, t) = \underline{\tau}_p(\boldsymbol{r}_f + \boldsymbol{c}_i\Delta t, t) + \frac{1}{N_f}\underline{\tau}_p(\boldsymbol{r}_f, t). \tag{10.41}$$

A node at $\boldsymbol{r}_f$ with $n_I \to n_O$, on the other hand, receives from the surrounding N_f n_O-nodes

$$\underline{\tau}_p^*(\boldsymbol{r}_f, t) = \frac{1}{N_f + 1}\sum_{i=1}^{q-1}\underline{\tau}_p(\boldsymbol{r}_f + \boldsymbol{c}_i\Delta t, t) \tag{10.42}$$

and the corresponding amount is removed from the respective nodes:

$$\underline{\tau}_p^*(\boldsymbol{r}_f + \boldsymbol{c}_i\Delta t, t) = \underline{\tau}_p(\boldsymbol{r}_f + \boldsymbol{c}_i\Delta t, t)\left(1 - \frac{1}{N_f + 1}\right) \tag{10.43}$$

Equation (10.43) agrees with [27, eq. 22], but differs from the description in [69, eq. 45], where there are probably several typing errors.

M4: ongoing shovel Method M3 provides problems as outlined in chapter 10.8.3.2. In short, $\underline{J}(\boldsymbol{r}_I + \boldsymbol{c}_i\Delta t/2) = 0$ leads to an undesired build-up of stress near the capsule boundary. This motivates method M4, which is a modification of M3. Here, the condition $\underline{J}(\boldsymbol{r}_I + \boldsymbol{c}_i\Delta t/2) = 0$ is removed, which leads to stress being advected into the outermost layer of the capsule interior in the kernel `stress_tensor`. In the immediately following kernel `conserve_stress2`, the stress is distributed to the surrounding N_f outer neighbors. For an inner node at $\boldsymbol{r}_f$ with an outer neighbor at $\boldsymbol{r}_f + \boldsymbol{c}_i\Delta t$, again eq. (10.41) holds, where this time $\underline{\tau}_p^*$ denotes the stress tensor after calling `conserve_stress2`. Since the capsule interior is now free of stress [16] the remaining steps starting from kernel `ibm_inout_tracking_A` can be applied like in M3. The individual steps of algorithm M4 are shown in fig. 10.30. An overview of the kernels was already given in fig. 10.6.

Concerning the implementation, there are still two technical details to note. First, in the kernel `count_outer_neighbors`, all nodes with $G_I = 1$ must now additionally determine the number of their outer neighbors, not only those with $G_C = 1$. Second, in the same kernel, the last remaining bit in the 8-bit flag is used as another flag G_S to mark nodes in the outermost layer of the capsule interior. This is an optimization, as nodes (G_I && $!G_S$) can now be omitted from the advection in `stress_tensor`.

10.8.3.2 Pure advection tests

Now it will be investigated how the algorithms described above perform with regard to advection. For this purpose, the three setups S1, S2 and S3 described in ch. 10.8.2 are used. This time, a capsule of radius $R = 6\Delta x$ and $n_\Delta = 1280$ is placed in all setups, following the

[14] Used for the conservation of stress for simulations of hard particles in a viscoelastic fluid via the moving boundary method.
[15] Used for the conservation of charge for simulations of the propagation of ion concentrations.
[16] For parallelization reasons, the stress inside is actually only explicitly set to zero in kernel `conserve_stress`.

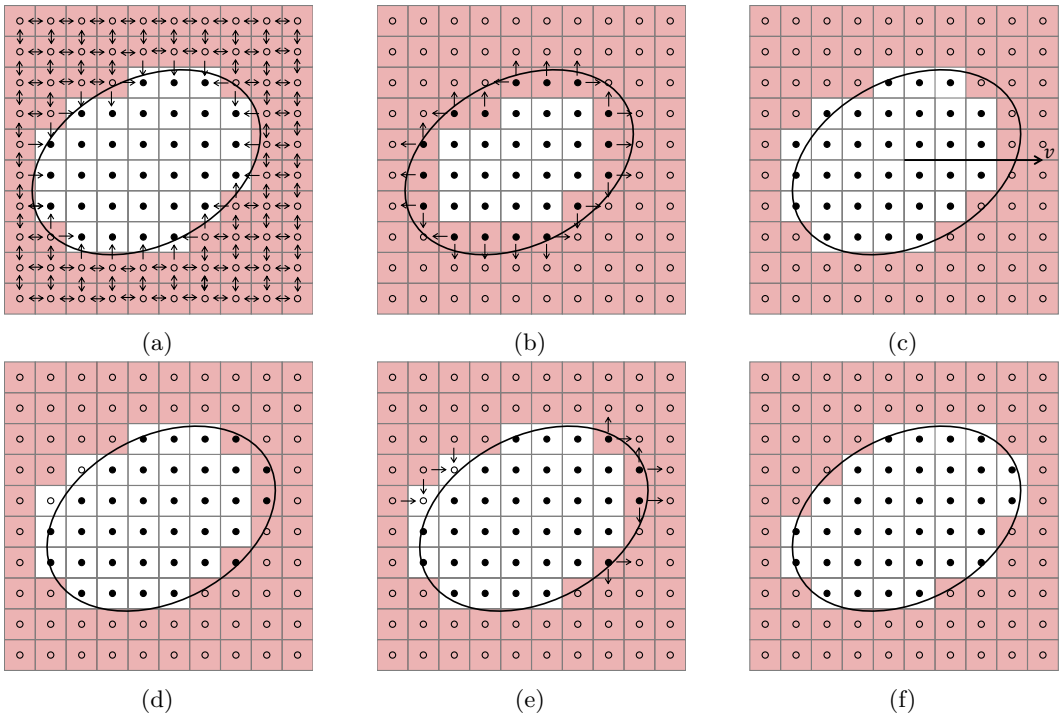
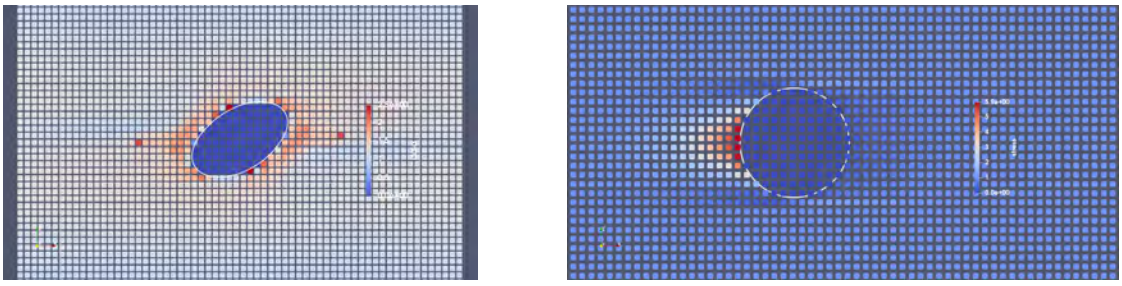


Fig. 10.30: 2D illustration of the algorithms M4, where a D2Q5 neighborhood is used for advection. The inside/outside flags are represented by filled/empty circles at the lattice node centers. The arrows indicate how stress (red squares) is exchanged between neighbors. Advection into the interior of the cell is initially allowed (a), but the stress is distributed again to external neighbors (b). These two steps together prevent accumulation of stress. After calling the IBM kernels, the cell has moved and the inside/outside flag grid is no longer up-to-date (c), which is corrected by IBM tracking (d). This leads to a new redistribution of stress for the changed nodes and their neighbors (e), before a correct capsule is ready for the next iteration (f).

neo-Hookean law. It will be simulated at $\mathrm{Ca} = 0.1$ and $\mathrm{Re} = 0.05$, where the capillary number is defined as $\mathrm{Ca} = \frac{\eta\dot{\gamma}R}{\kappa_S}$. The Reynolds number for S1 is given by $\mathrm{Re} = \frac{R|u_w|}{\nu}$, for S2 and S3 it holds $\mathrm{Re} = \frac{R|u_{\mathrm{center}}|}{\nu}$. The same procedure as in chapter 10.8.2 is applied (turning off the constitutive equation, τ is an ink-like scalar that is only advected and does not couple to the LBM), where τ is initialized to zero (ink-free) inside the capsule and to one (ink-filled) outside the capsule. The velocity field for the advection is computed by the IB-LBM with a D3Q27 neighborhood and TRT. In an ideal simulation, the capsule boundary should always have exactly the velocity of the surrounding fluid. Accordingly, for $t^* > 0$ the ink-free region should not mix with the ink-filled region. Since the velocity field already does not fulfill this perfectly in real simulations, even a perfect advection algorithm would mix the two regions. Nevertheless, to quantify the quality of the advection algorithms, the deviation from the ideal values will be used, i.e. $|\tau(\boldsymbol{r}_I)|$ and $|\tau(\boldsymbol{r}_O) - 1|$ for an interior and exterior node, respectively. Since M0 and M1 behave the same with respect to advection (the coupling of τ to the fluid is switched off for this test anyways), only M0, M3 and M4 are included in the comparison of the methods. In the analysis plots, for the calculation of the maximum and mean deviation as well as for the histograms, only those fluid nodes are considered that either lie within the capsule or, starting from the capsule edge, can be reached within 6 time steps with lattice speed of and a D3Q27 velocity set. For the check of the conservation of τ_{ges}, on the other hand, all fluid nodes are included.

In the setup S1 (shear flow), the capsule is placed in the center of the domain and remains there at $t^* > 0$. In the setup S2 (planar poiseuille), the capsule also starts in the center, but has already completely crossed the domain once at $t^* = 20$. In the setup S3 (unaligned poiseuille), the capsule starts in the center of the channel in the corner of the domain and has almost reached the center of the domain at $t^* = 20$.

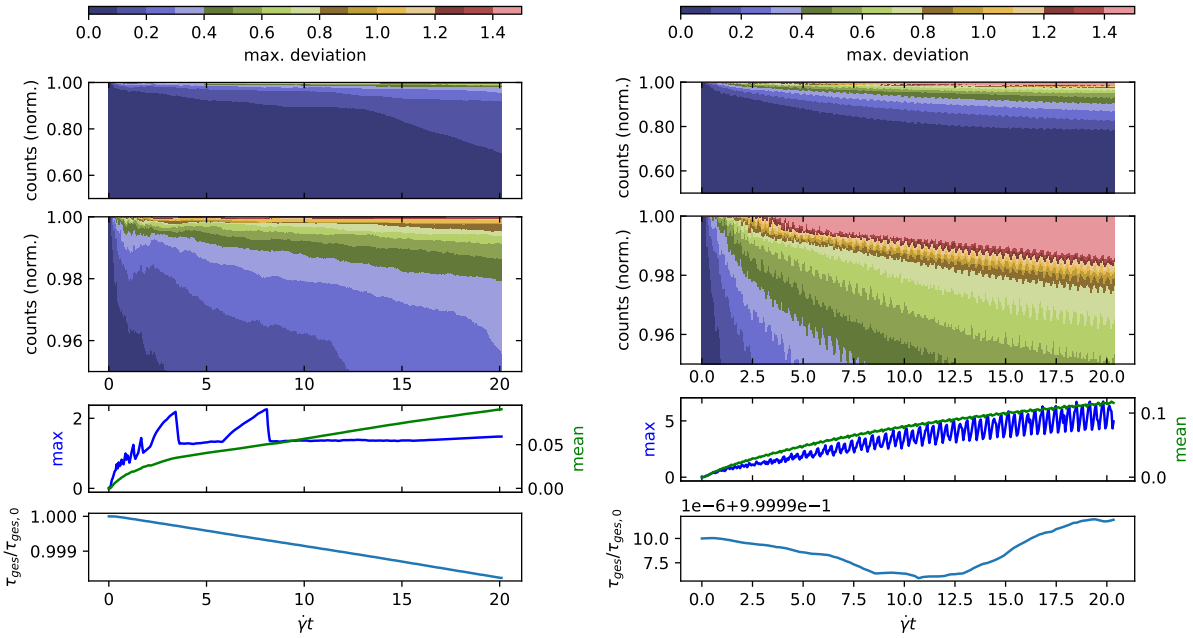
First, the combination **CTU** + D3Q27 + M3 is examined. For this combination, in the setup S1 the stress accumulates at some corners of the capsule (cf. fig. 10.31a and 10.32a). The reason for this is that the velocity field has a component into the cell, but for τ the no-flux condition $J(\boldsymbol{r}_I + \boldsymbol{c}_i\Delta t/2) = 0$ holds. In the setup S2, stress accumulates behind the capsule due to the periodic motion of the boolean G_I-flag field, which together with the no-flux condition systematically pushes the stress backwards (cf. fig. 10.31b and 10.32b).



(a) S1 (shear flow)

(b) S2 (planar poiseuille)

Fig. 10.31: Advection test using **CTU** + D3Q27 + M3. A slice at $y = L_t/2$ of the field τ is shown at $t^* = 20$. In (a), an accumulation of τ at the capsule edge is visible. The color bar ranges from 0.0 (blue) to 2.5 (red). In (b), stress accumulates behind the capsule. The color bar ranges from 0.0 (blue) to 5.9 (red). The corresponding plots are shown in fig. 10.32.



(a) S1 (shear flow)

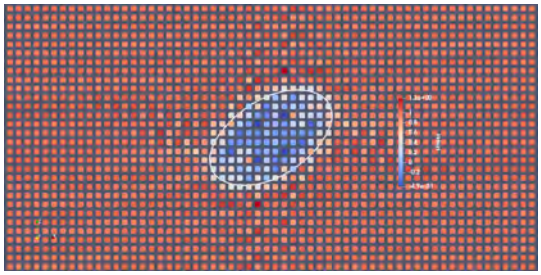
(b) S2 (planar poiseuille)

Fig. 10.32: Advection test using **CTU** + D3Q27 + M3. Top: stacked and normalized histogram of all fluid nodes, ordered by the amount of their respective deviation from their ideal value, $|\tau(\boldsymbol{r}_I)|$ or $|\tau(\boldsymbol{r}_O)-1|$. Deviation intervals color-coded. Histogram in two versions with different "zoom levels". Mid: maximum and average deviation from the ideal value. Bottom: check of conservation of τ_{ges}. The accumulation of stress is visible by the increasing maximum deviation. It decreases again whenever the G_I-flag field shifts in such a way that τ can flow away from the accumulation point.

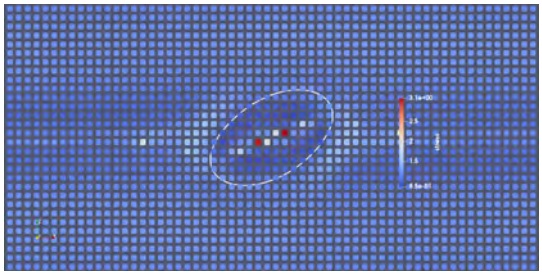
M3 is therefore discarded. The tests for the combinations {**CTU**, **simple**} x {M0, M4} x {D3Q27} x {S1, S2, S3} are shown in figs. 10.34 - 10.40. It can be seen that the combination M4 + **CTU** performs best in every setup, followed by the combination M0 + **CTU**. Further on, the following observations are made:

- For **simple**, negative values for τ occur in all setups. This is particularly pronounced in combination with M0.
- The high symmetry of the setup S1 (shear flow) is reflected by all advection algorithms.
- M4 can successfully prevent the accumulation of τ near the edge of the capsule that was observed in S1 when using M3.
- In the setup S1, stress accumulates for **CTU** in the y-plane at a very concentrated point on both sides of the capsule. In addition, a cross-shaped accumulation occurs in the x-plane. Whether this is an effect of the advection algorithm or just an accurate reflection of stagnation points of the velocity field is not clear here.
- In the setup S2 (planar poiseuille), M4 can successfully prevent the accumulation of τ behind the capsule observed in M3.
- In the setups S2 and S3, waves occur in the stress distribution behind the capsule when using **simple**.
- In the setup S3 the problems with the staircase effect regarding the walls are also present in simulations with capsule. The color scale in fig. 10.39b and 10.39d was artificially adjusted to the interval [0, 1.2]. At the channel walls τ takes values up to 1.9.
- The violation of the conservation of τ_{ges} has the same orders of magnitude as already observed for simulations without capsule in chapter 10.8.2. It is independent of the use of M0/M4. Furthermore, τ_{ges} tends to be better preserved for **simple** than for **CTU**. The shape of the curve $\tau_{\text{ges}}(t)$ seems to depend mainly on the choice of setup.

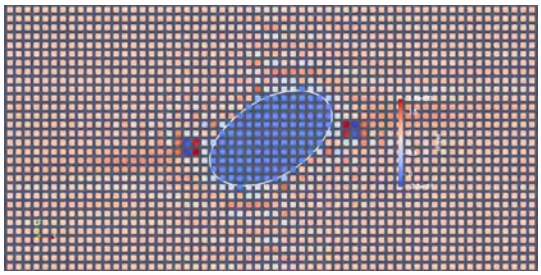
Presented so far are the results using a D3Q27 neighborhood for advection. In the setups S1 and S2 the advection neighborhoods D3Q19 and D3Q7 do not produce worse results (in some cases they are even better) because of the alignment along the coordinate axes. In the setup S3 a difference between the advection neighborhood sets is only visible with the combination **CTU** + M4 and is shown in fig. 10.41. In fact, the results improve as the size of the neighborhood set increases.



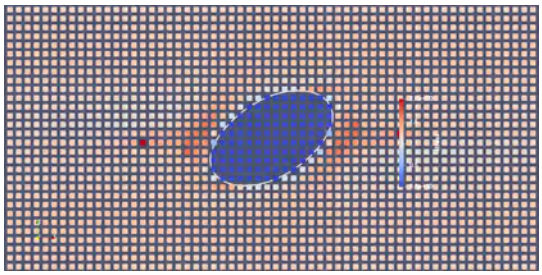
(a) **simple** + D3Q27 + M0. The color bar ranges from -0.5 (blue) to 1.3 (red).



(b) **CTU** + D3Q27 + M0. The color bar ranges from 0.9 (blue) to 3.1 (red).

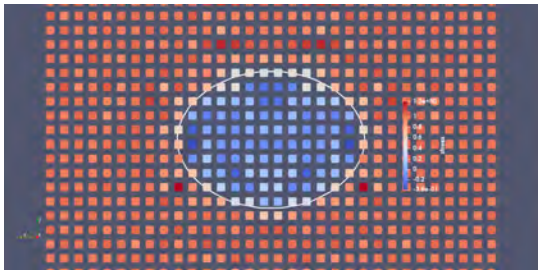


(c) **simple** + D3Q27 + M4. The color bar ranges from -0.3 (blue) to 1.8 (red).

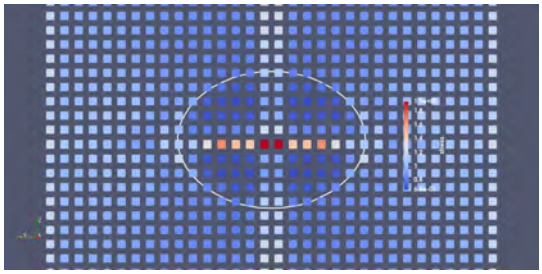


(d) **CTU** + D3Q27 + M4. The color bar ranges from 0.0 (blue) to 2.0 (red).

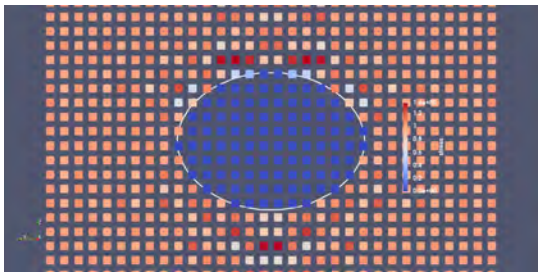
Fig. 10.33: Advection test using setup S1 (shear flow) with capsule. Shown is a slice through the ink τ at $y = L_t/2$ and $t^* = 20$. Note the high symmetry of ink-distribution and, in case of **CTU**, the ink-accumulation on both sides of the capsule.



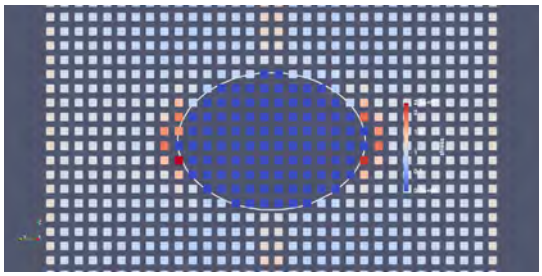
(a) **simple** + D3Q27 + M0. The color bar ranges from -0.4 (blue) to 1.3 (red).



(b) **CTU** + D3Q27 + M0. The color bar ranges from 0.7 (blue) to 1.9 (red).

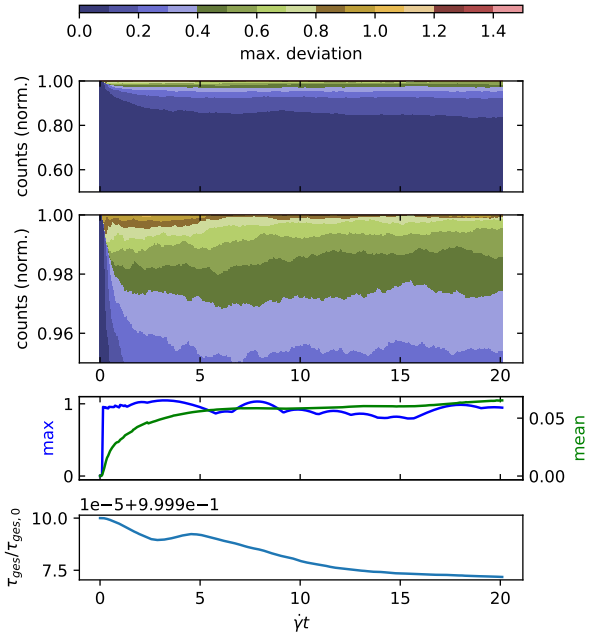


(c) **simple** + D3Q27 + M4. The color bar ranges from 0.0 (blue) to 1.4 (red).

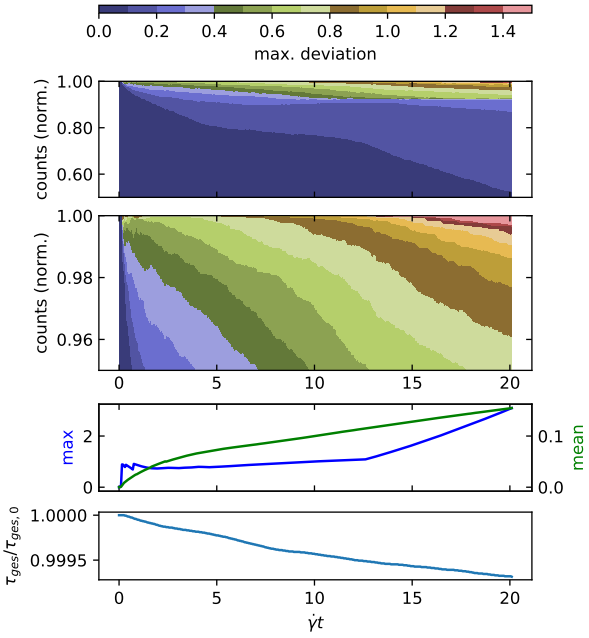


(d) **CTU** + D3Q27 + M4. The color bar ranges from 0.0 (blue) to 2.3 (red).

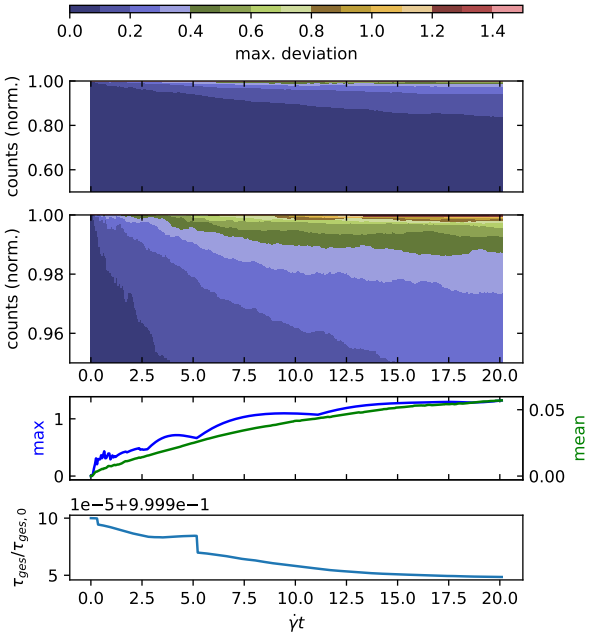
Fig. 10.34: Like fig. 10.33, but at $x = L_s/2$. Note the high symmetry of ink-distribution and, in case of **CTU**, the cross-shaped ink-accumulation.



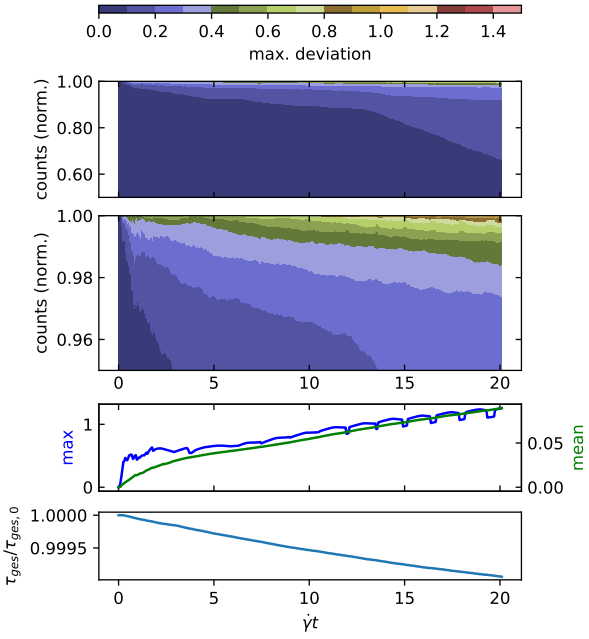
(a) **simple** + D3Q27 + M0



(b) **CTU** + D3Q27 + M0

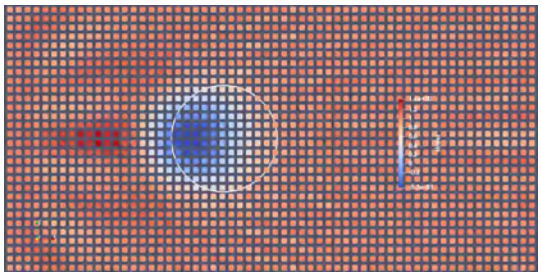


(c) **simple** + D3Q27 + M4

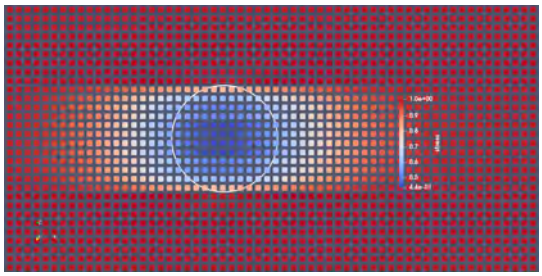


(d) **CTU** + D3Q27 + M4

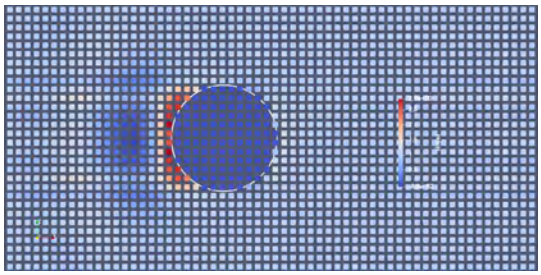
Fig. 10.35: Advection test using setup S1 (shear flow) with capsule. The meaning of the plots is the same as in fig. 10.32. Algorithm M4 performs much better than M0 in terms of accuracy. The conservation of τ_{ges} is orders of magnitude better for **simple** when compared to **CTU**.



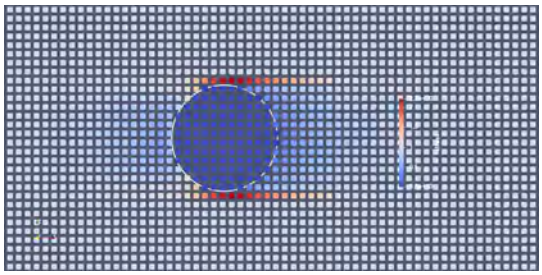
(a) **simple** + D3Q27 + M0. The color bar ranges from -0.5 (blue) to 1.4 (red).



(b) **CTU** + D3Q27 + M0. The color bar ranges from 0.4 (blue) to 1.0 (red).

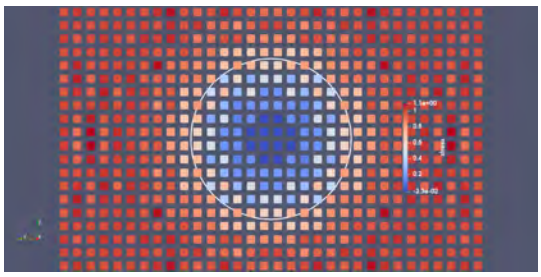


(c) **simple** + D3Q27 + M4. The color bar ranges from -0.1 (blue) to 2.9 (red).

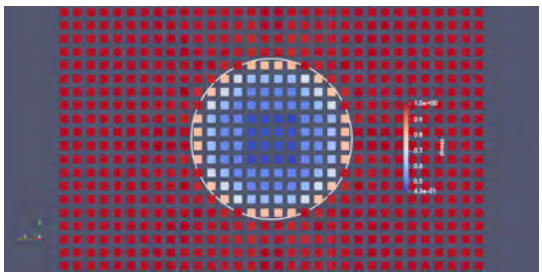


(d) **CTU** + D3Q27 + M4. The color bar ranges from 0.0 (blue) to 2.2 (red).

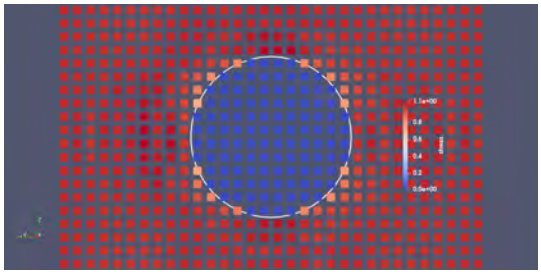
Fig. 10.36: Advection test using setup S2 (planar poiseuille) with capsule. Shown is a slice through τ at $y = L_t/2$ and $t^* = 20$. Note the negative values of τ occurring for algorithm M0.



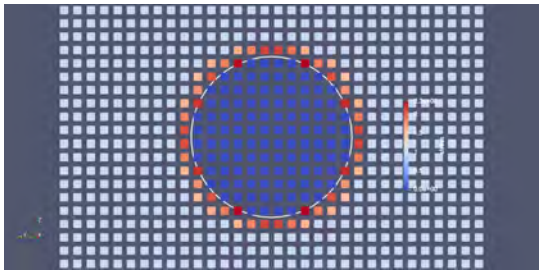
(a) **simple** + D3Q27 + M0. The color bar ranges from 0.0 (blue) to 1.1 (red).



(b) **CTU** + D3Q27 + M0. The color bar ranges from 0.4 (blue) to 1.0 (red).

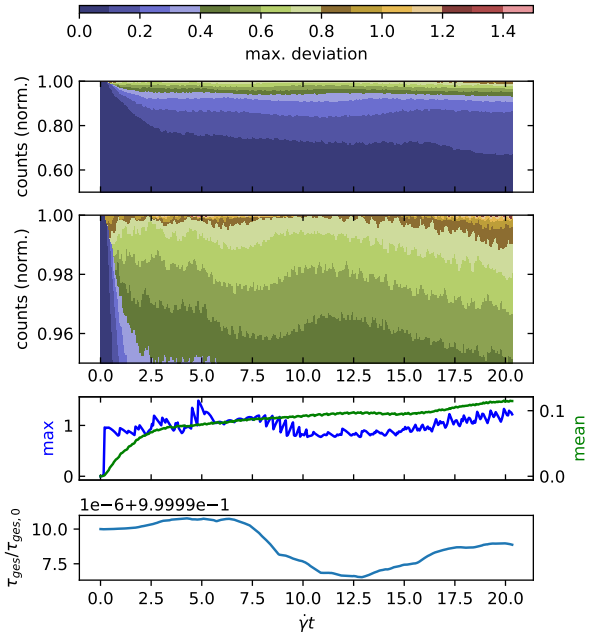


(c) **simple** + D3Q27 + M4. The color bar ranges from 0.0 (blue) to 1.1 (red).

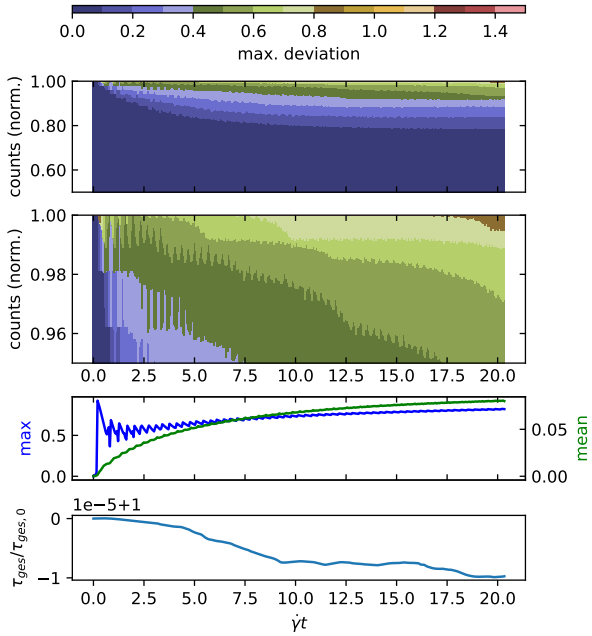


(d) **CTU** + D3Q27 + M4. The color bar ranges from 0.0 (blue) to 2.3 (red).

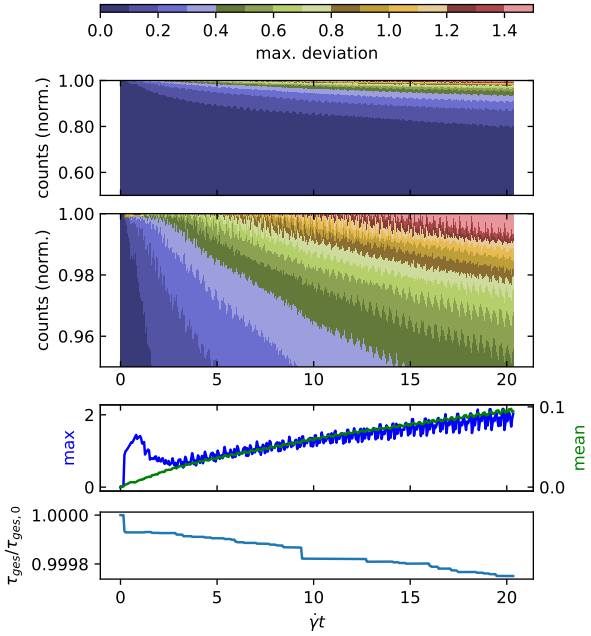
Fig. 10.37: Like fig. 10.36, but at $x = L_s/2$.



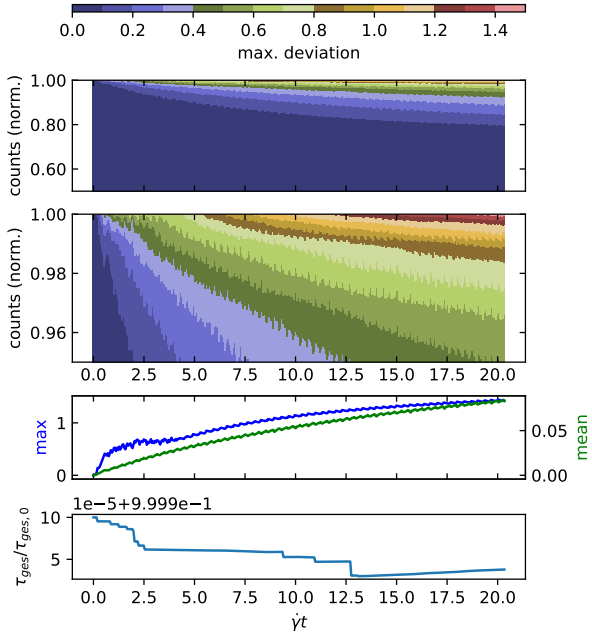
(a) **simple** + D3Q27 + M0



(b) **CTU** + D3Q27 + M0

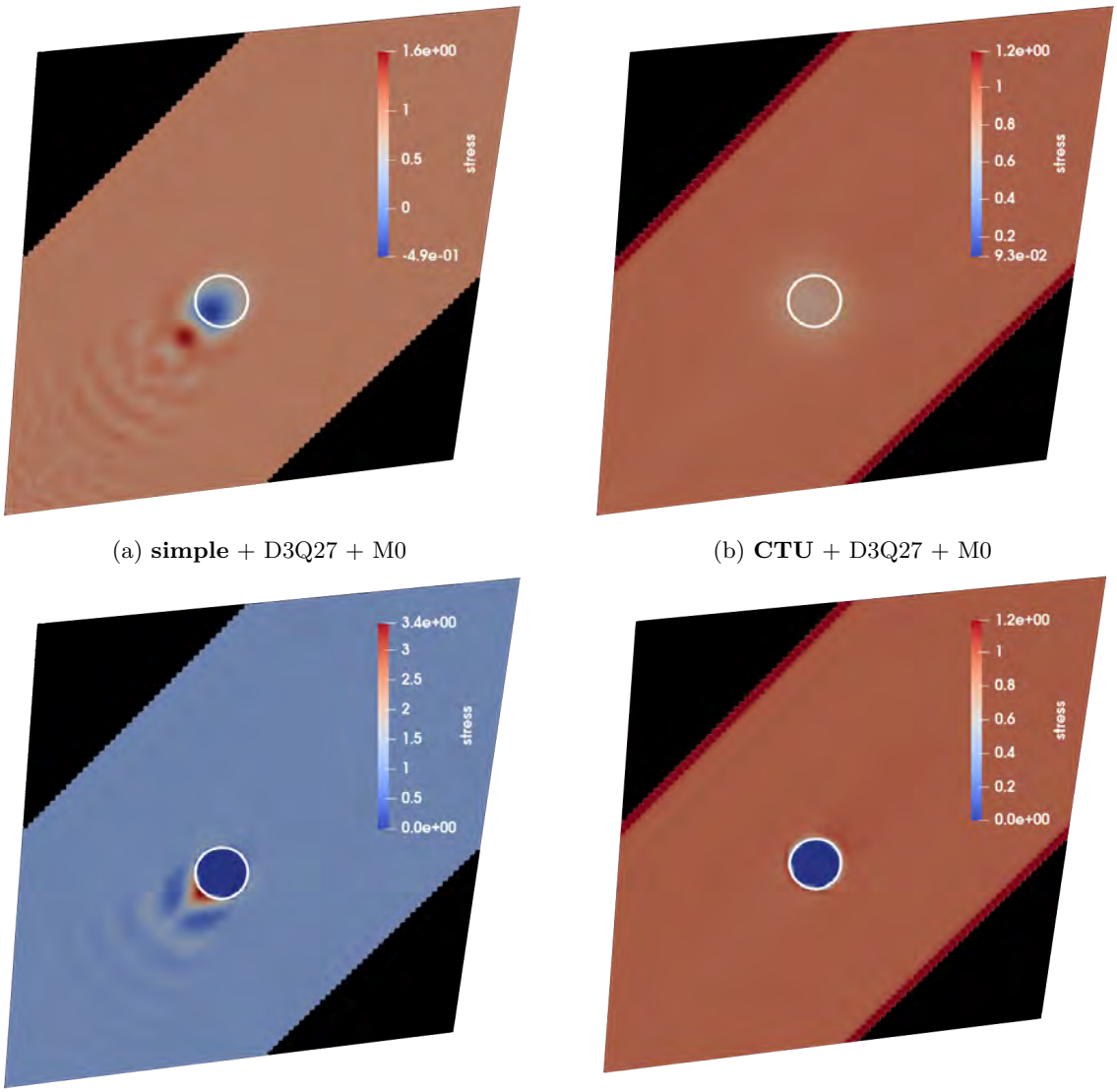


(c) **simple** + D3Q27 + M4



(d) **CTU** + D3Q27 + M4

Fig. 10.38: Like fig. 10.35, but for setup S2 (planar poiseuille).



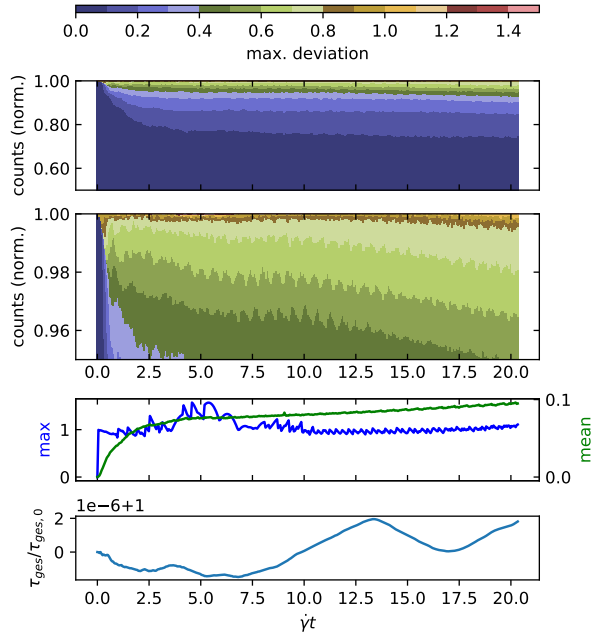
(a) **simple** + D3Q27 + M0

(b) **CTU** + D3Q27 + M0

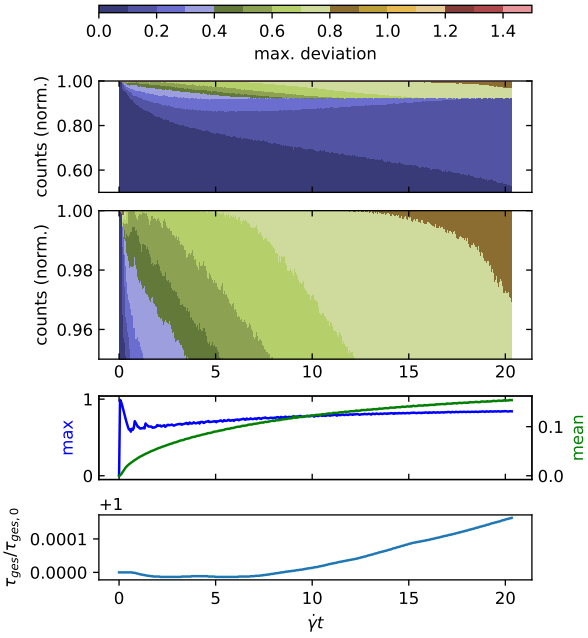
(c) **simple** + D3Q27 + M4

(d) **CTU** + D3Q27 + M4

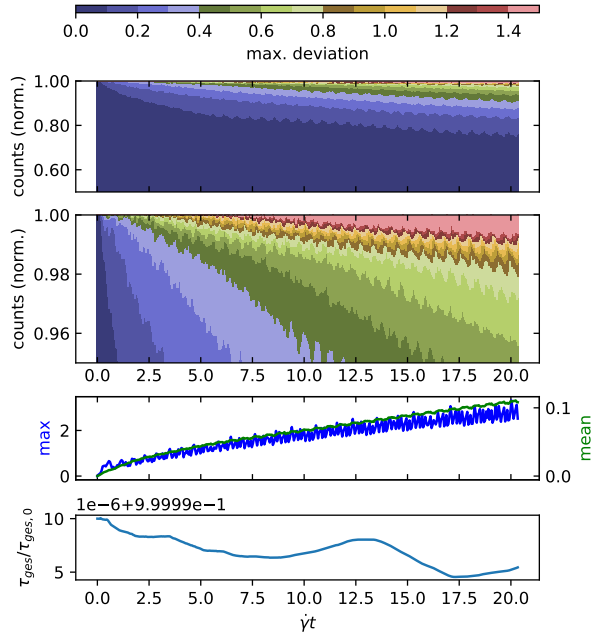
Fig. 10.39: Advection test using setup S3 (unaligned poiseuille) with capsule. Shown is a slice through τ in the plane $\boldsymbol{r} \cdot (1, 1, -2)^{\mathrm{T}} = 0$ and at time $t^* = 20$. Note the waves in τ occurring for **simple**. Also note the accumulation of ink at the walls occurring for **CTU**. Algorithm M0 leads to negative values in τ, which are not present in algorithm M4.



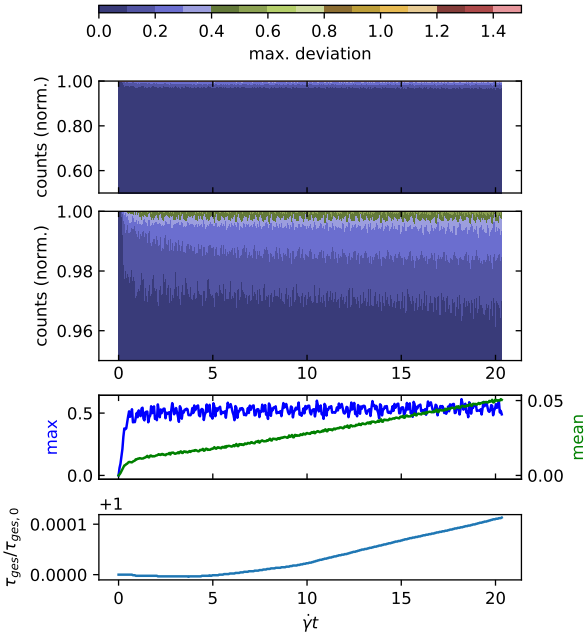
(a) **simple** + D3Q27 + M0



(b) **CTU** + D3Q27 + M0

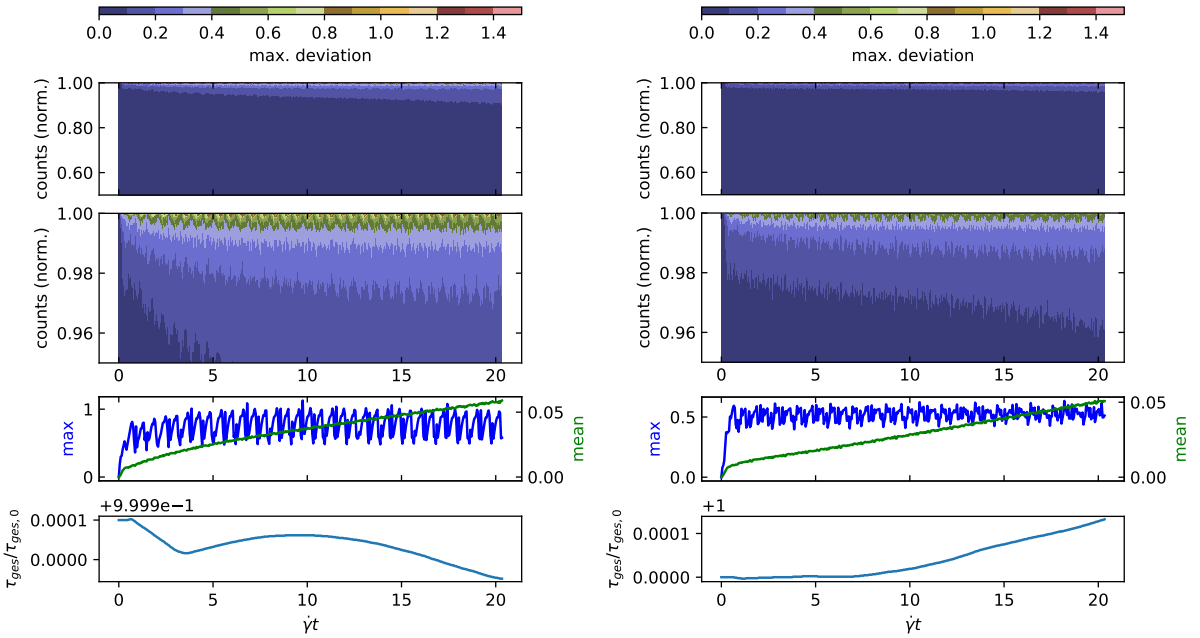


(c) **simple** + D3Q27 + M4



(d) **CTU** + D3Q27 + M4

Fig. 10.40: Like fig. 10.35, but for setup S3 (unaligned poiseuille). The combination **CTU** + D3Q27 + M4 outperforms the other combinations in terms of accuracy. The conservation of τ_{ges} is orders of magnitude better for **simple**.



(a) **CTU** + D3Q7 + M4

(b) **CTU** + D3Q19 + M4

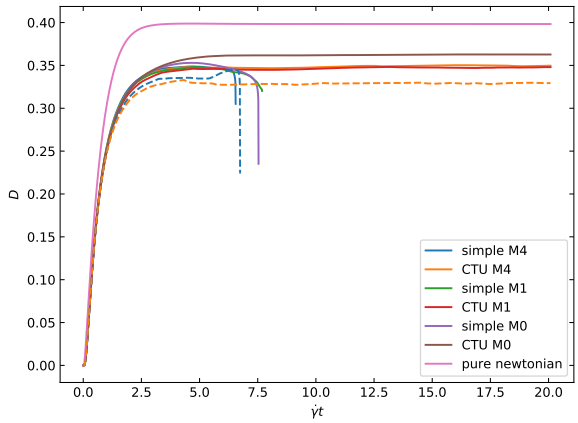
Fig. 10.41: For the combination **CTU** + M4 and the setup S3 (unaligned poiseuille) the results improve as the size of the advection neighborhood set increases (compare to fig. 10.40d).

10.8.3.3 Behavior with viscoelasticity switched on

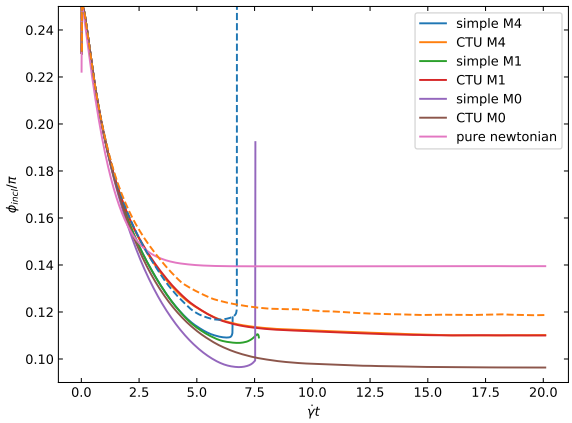
Next, it is investigated how the algorithms behave when the stress interacts with the fluid again via the constitutive equation. The combinations {**CTU**, **simple**} x {M0, M1, M4} are inspected. Here, D3Q27 is always chosen as the neighborhood of the advection algorithms. First, in connection with M1 and M4, the question arises how the boundary conditions are to be chosen for n_O-nodes next to the capsule interior. A constant extrapolation of stress was chosen for the wall nodes regarding the no-slip bounce-back boundaries. But since the interior of the capsule is explicitly assumed to be stress-free in both algorithms, a linear extrapolation into zero makes more sense in this context.

Like in chapter 10.8.3.2 the three setups S1, S2 and S3 are simulated at $\mathrm{Ca} = 0.1$ and $\mathrm{Re} = 0.05$. Furthermore, again $R = 6\Delta x$ and $n_\Delta = 1280$ is chosen. In order to switch on viscoelasticity, the Oldroyd-B model with $\mathrm{Wi} = 2$ and $\beta = 0.5$ is used. In contrast to before, the measures of the planar poiseuille channel of setup S2 are changed to $L_s \times L_t \times H = 64\Delta x \times 32\Delta x \times 32\Delta x$. This causes higher shear rates near the center of the channel and thus higher deformation. In fig. 10.42 and 10.43 the results of the different algorithms are shown, always compared against the pure Newtonian solution of same total viscosity. In addition, for M4 the version with constant extrapolation is drawn in dashed lines. The version with constant extrapolation is clearly further away from M0 than the version with linear extrapolation into zero. Overall, for linear extrapolation the two algorithms M4 and M1 agree well. The differences to M0 are not very large, since only small shear rates are present inside the cell and thus the viscoelasticity inside the capsule is small even for a purely viscoelastic fluid.

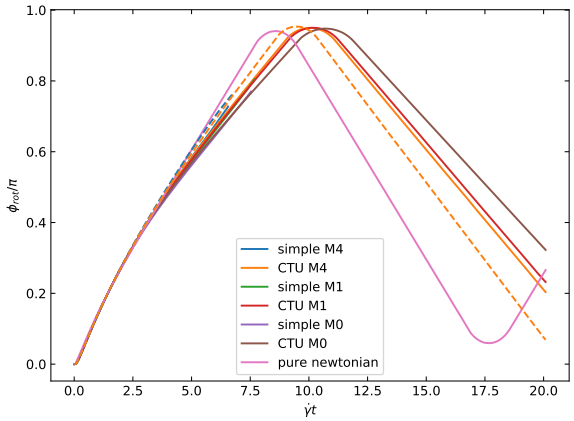
Moreover, fig. 10.42 shows that **CTU** is significantly better than **simple** in terms of stability.



(a) Deformation

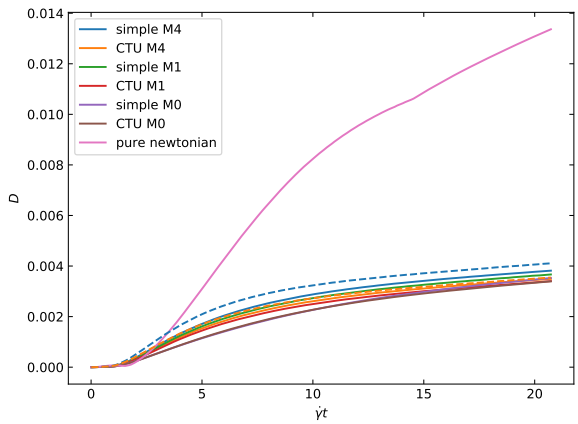


(b) Inclination

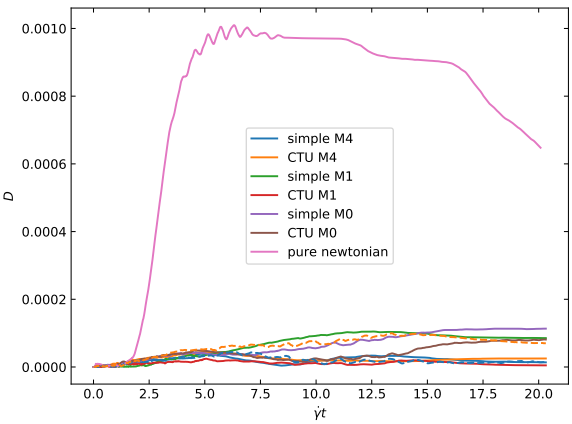


(c) Rotation

Fig. 10.42: Newtonian capsule at Wi = 2 in Oldroyd-B shear flow, setup S1. Comparison of different advection algorithms with the pure Newtonian fluid. For M4, the constant extrapolation version is plotted with dashed lines.



(a) S2 (planar poiseuille)



(b) S3 (unaligned poiseuille)

Fig. 10.43: Like fig. 10.42a, but for the setups S2 and S2.

More detailed investigations showed that in the setup S1 (shear flow) instabilities occur starting at Wi ≈ 1 when using **simple**. Therefore, only **CTU** will be used in the future.

Fig. 10.44 illustrates how small the difference between the algorithms actually is: For the setup S2, the algorithms result in almost identical capsule meshes, all of which differ strongly from the pure Newtonian solution. The latter propagates much faster through the fluid and experiences higher deformation. Similar good agreement between the meshes was found for the other setups, although in the shear flow setup S1 the meshes are twisted with respect to each other due to the slightly different rotational frequency of the respective algorithms. In fig. 10.45 for setup S1 the stress field of M0 is contrasted with that of M4. The results of the algorithms are in good visual agreement. Effects of the individual algorithms, as considered for pure inks in fig. 10.33 and 10.34, thus play no role with the constitutive equation switched on. This is also true for the ink-accumulation near wall boundaries in the unaligned poiseuille setup S3 when using **CTU** that was observed in fig. 10.39. No build-up of stress near the walls could be observed for actual viscoelastic fluids (cf. fig. A4).

Overall, it can be observed in fig. 10.42a that the steady state deformation of M0 beginning at $t^* \approx 5$ is somewhat smoother than that of M4 and M1. The fact that the latter two algorithms switch discretely between two parameter sets is thus visible here. This tendency is even stronger for FENE-P and also gets worse for coarser resolution (cf. fig. A6). Furthermore, for increasing Wi the results of M0 increasingly differ from those of M1 and M4, respectively. This can already be observed in fig. A6, but will be explained in more detail in chapter 10.9 (cf. fig. 10.48).

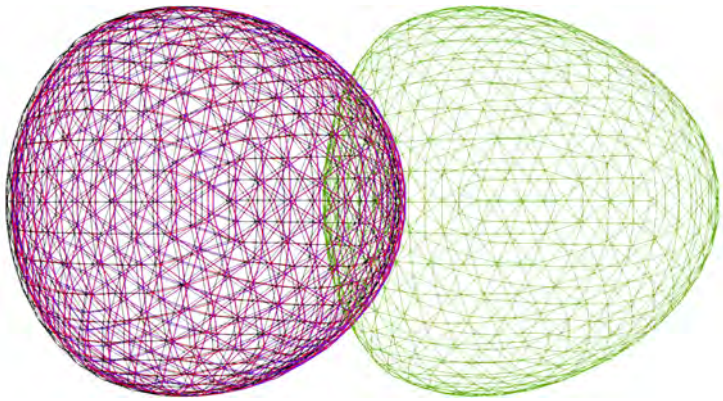
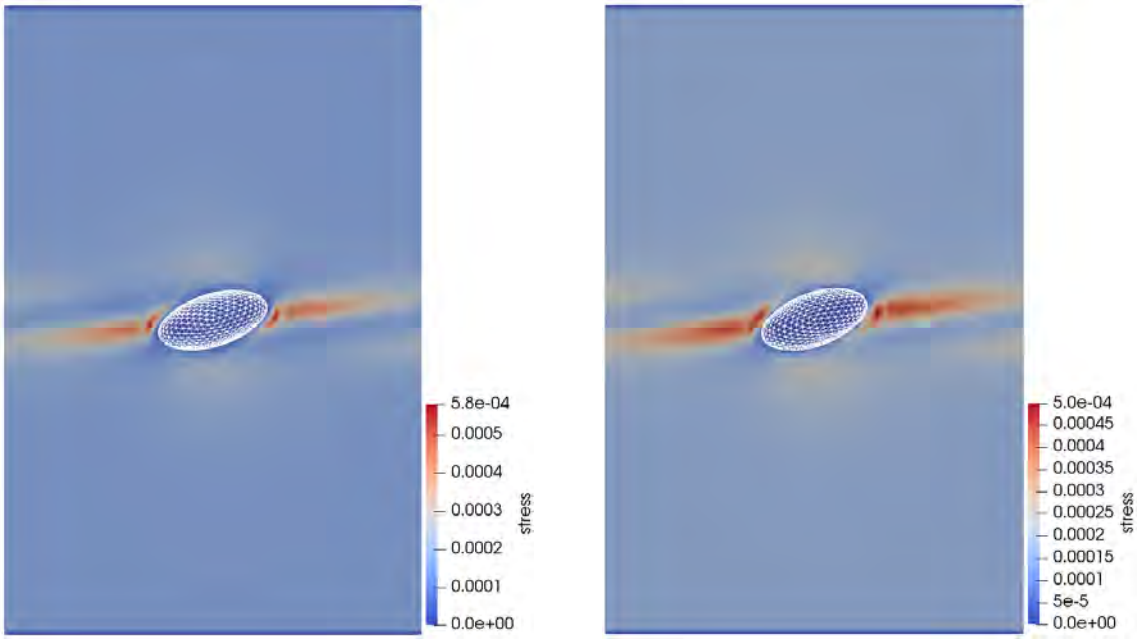


Fig. 10.44: Capsule mesh in setup S2 at $t^* = 7.5$ using **CTU**. The algorithms M0 (black), M1 (blue) and M4 (red) in comparison to the pure Newtonian fluid (green).



(a) M0 (b) M4

Fig. 10.45: Stress component s_{xx} in shear flow at $t^* = 20$ using **CTU**. The stress field is qualitatively the same for both algorithms M0 and M4.

10.9 Validation of capsule in viscoelastic fluid

Also [68] consider capsules in Oldroyd-B fluids. They do not use the FVM for the viscoelasticity part, but instead couple a D3Q7-LBM viscoelasticity solver to the actual D3Q19-NS solver. Moreover, for stabilization, they introduce an additional diffusion term in the constitutive equation to control the growth of the gradient of the stress tensor. They specify the strength of the term using the dimensionless diffusion parameter $\mathrm{Pr} = k/\dot{\gamma}L^2$, where k denotes the diffusion constant and L is a characteristic length.

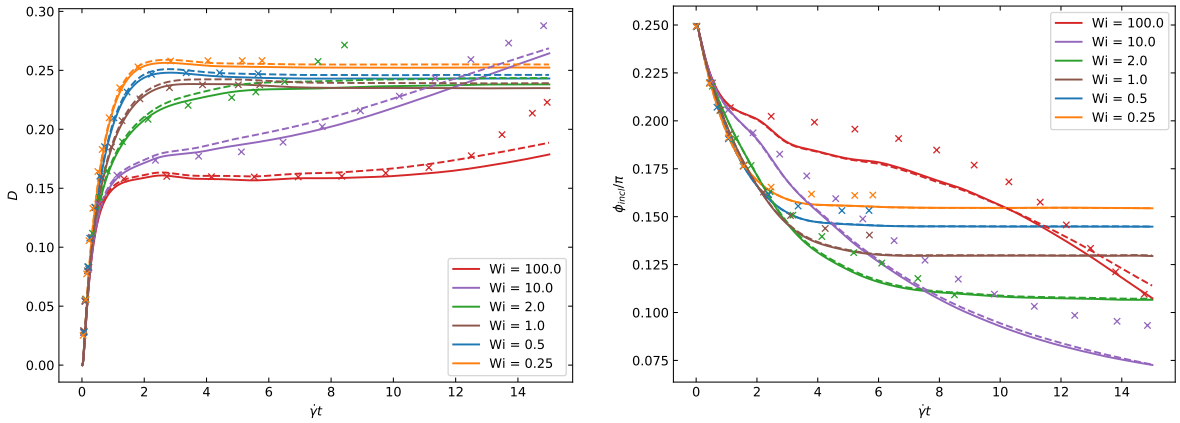
To compare the results, their setup of a 3D capsule in an Oldroyd-B shear flow has to be reproduced. It consists of a cubic domain with $L_s \times L_t \times H = 10R \times 10R \times 10R$. The simulations are performed at Re= 0.125, Ca = 0.05 and $\beta = 0.5$ [17]. Internal and external fluid have the same properties, the neo-Hookean law is used for the capsule and there is no bending energy. The grid spacing is $\Delta x = R/12$, so correspondingly $R = 12\Delta x$, which will be resolved by $n_\Delta = 5120$.

A comparison of their simulation results with the ones of this thesis is shown in fig. 10.46. Up to the dimensionless time $\dot{\gamma}t = 5$ there is a very good agreement. After that, there are deviations for Wi$\geq$ 2. However, these are partly within the range of deviations caused by the artificial diffusion term within the simulations of [68]: their fig. 21 provides a comparison of the data shown here with simulations at lower Pr. It seems like lower Pr tend to go with lower D. Thus, the fact that the simulation results from their fig. 18 show higher D than the ones presented in this thesis is probably due to their diffusion constant being too high. Furthermore, strangely enough, in their 3D simulation no initial overshoot of D for $\mathrm{Wi} \in \{0.25, 0.5, 1.0\}$ can be seen. However, in their 2D simulations, [68] find overshoots which are qualitatively similar to those

[17] The dimensionless quantities given by [68] were converted to the definitions valid in this thesis.

observed in this thesis (compare fig. 10.46 with their fig. 15). Overall, it should be noted that the simulations for Wi = 10 and Wi = 100 do not reach a steady state, but become unstable at later times. Upon inspection using *ParaView*, one notices that the simulation sees its own periodic image (cf. fig. 10.47a). So the domain chosen by [68] is actually too small. Therefore, the domain was increased in stream-flow direction until no significant change in D occurred, which was the case at $L_s = 40R$. At the same time, the domain was decreased to $L_t = 5$ to reduce computation time, but this had no effect on D. The solution with enlarged domain is shown in fig. 10.46 as dashed lines.

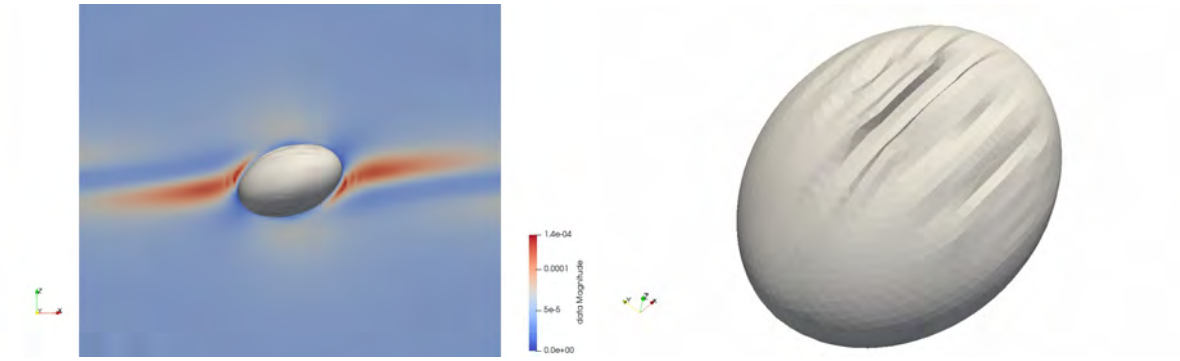
Moreover, the inspection of the capsule mesh reveals a buckling behavior in the capsule surface. In [68, fig. 19] similar buckling is documented, but the ones from the present algorithm (cf. fig. 10.47b) are more sharp-edged. Possibly this is because in the present algorithm there is lower diffusion of stress at the capsule surface.



(a) Taylor deformation D

(b) Inclination angle

Fig. 10.46: Validation of a capsule in Oldroyd-B shear flow, interior and exterior fluids are equal. The data from [68, fig. 18] is indicated with crosses. In FluidX3D simulations with the same domain size (solid, $10R \times 10R \times 10R$) and with a bigger domain size (dashed, $40R \times 5R \times 10R$) were conducted.



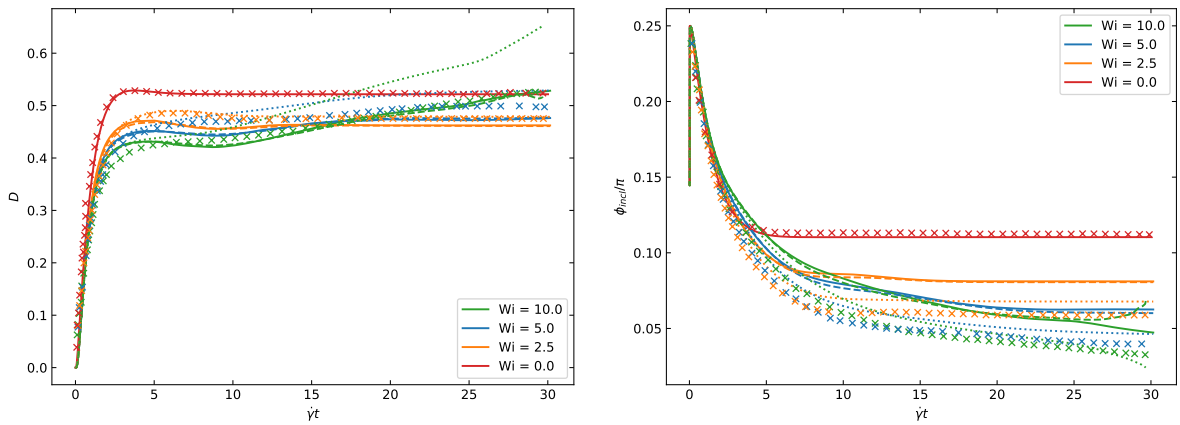
(a) Simulationbox of [68] contains periodic images.

(b) Buckling described by [68].

Fig. 10.47: The pictures are taken using the original simulation domain $[0, 10R] \times [0, 10R] \times [0, 10R]$ at Wi = 2 and $t^* = 15$.

Now that the viscoelastic capsule in viscoelastic fluid has been validated, we turn to the Newtonian capsule in viscoelastic fluid. Here, a comparison with Raffiee et al. [78], whose method was already explained in chapter 10.8.3.1, is appropriate. The setup from their fig. 8 is recreated, i.e., a domain with $L_s \times L_t \times H = 128\Delta x \times 64\Delta x \times 126\Delta x$ is used. Moving walls at $z = 0$ and $z = H$ cause a shear flow. An initially spherical capsule of radius $R = 12\Delta x$ is placed in the domain center, which will be resolved by $n_\Delta = 5120$. Other fixed dimensionless numbers are $\mathrm{Ca} = 0.2$, $\mathrm{Re} = 0.1$[18] and $\beta = 0.5$. In fig. 10.48, their simulation data is compared with that of this thesis. Linear interpolation into zero was used for algorithms M1 and M4.

It can be seen that the pure Newtonian case ($\mathrm{Wi} = 0$) agrees well. For the remaining cases, M0 (dotted), M1 (dashed), and M4 (solid) are plotted against [78, fig. 8] (crosses). Let us first look at the deformation D from fig. 10.48a. With respect to the initial dynamics up to about $t^* = 10$, algorithm M0 agrees best with Raffiee et al. Compared to M0, for M1 and M4 the initial oscillations of D are more pronounced. For long times and increasing Wi, on the other hand, the data is in better agreement with M1 and M4, where the deformation is reduced compared to M0. Overall, the data of Raffiee et al. is mostly between the results of M0 and M1/M4. The reason for this might be their interpolation procedure (smoothed indicator function). Also, the fact that they do not explicitly care about stress conservation might play a role. Looking at the inclination data in fig. 10.48b, on the other hand, M0 always seems to fit the data from Raffiee et al. best. This is strange, since M0 is the reference method of a viscoelastic capsule in viscoelastic fluid. Furthermore, Raffiee et al. do not provide a comparison between their Newtonian and their viscoelastic capsule is viscoelastic flow. For all those reasons mentioned, the significance of the validation via Raffiee et al. is considerably reduced.



(a) Deformation

(b) Inclination

Fig. 10.48: Quantities of a Newtonian capsule in Oldroyd-B shear flow for various Wi. The algorithms M0 (dotted), M1 (dashed) und M4 (solid) are compared against data from [78, fig. 8] (crosses).

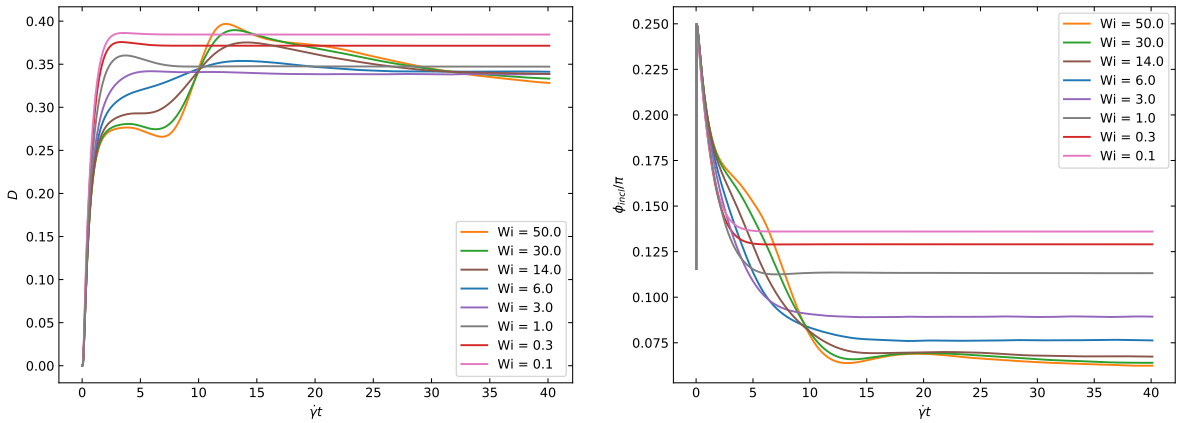
[18] [78] state that they are actually simulating at $\mathrm{Re} = 0.5$ (converted to the definition used here), but this already leads to strongly different results in the fully Newtonian case and is suspected to be an error.

10.10 Applications

10.10.1 Deformation and inclination of capsule in FENE-P shear flow

A neo-Hookean capsule in Oldroyd-B shear flow has already been studied in detail in literature. As shown above, the simulations of this thesis provide good agreement. Now, a similar investigation will be carried out for the FENE-P fluid. For this purpose, simulations are performed in a domain of size $L_s \times L_t \times H = 120\Delta x \times 64\Delta x \times 180\Delta x$ and at $\mathrm{Re} = 0.05$ and $\mathrm{Ca} = 0.1$. The dimensionless parameters of the FENE-P fluid are chosen as $\beta = 0.5$ and $b = 100$. Furthermore, a resolution of $R = 12\Delta x$ is chosen using $n_\Delta = 5120$. A comparison with simulations at $R = 6\Delta x$ using $n_\Delta = 1280$ (not shown) suggests that this resolution is sufficient.

The results are depicted in fig. 10.49. The data for the Taylor deformation D show an interesting transient response of the capsule when switching on the shear flow. For low Wi (until $\mathrm{Wi} \approx 3$) a single overshoot is observed. This overshoot was already present in Oldroyd-B simulations (cf. fig. 10.46). But when Wi is increased further, after the initial overshoot a further increase of the deformation assuming a step-like form is found, until D finally slowly decays to its steady state value (even the simulations with $\mathrm{Wi} = 30$ and $\mathrm{Wi} = 50$ reach a steady state for $t^* > 40$ and remain stable). This is a difference to Oldroyd-B, where no steady state occurs from about $\mathrm{Wi} \approx 10$. This difference between the two fluid types can be explained as follows: The presence of the capsule gives rise to an elongational flow near the capsule edges. The dimensionless elongational rate $\Lambda_e = \dot{\epsilon}\lambda_p$ is proportional to the Weissenberg number Wi. Therefore, for high Wi the stress near the capsule edges can grow unboundedly for the Oldroyd-B fluid (cf. chapter 4.3.2.3), resulting in an unbounded growth of the capsule deformation. For the FENE-P fluid, on the other hand, stress always reaches a steady state for elongational flows (cf. chapter 4.3.3.3) and so does the capsule deformation in the present shear flow setup. Overall, however, the tendency already observable for Oldroyd-B remains that a higher Weissenberg number causes lower deformation and inclination.



(a) Deformation

(b) Inclination

Fig. 10.49: Neo-Hookean FENE-P capsule in FENE-P shear flow. Parameters in text.

10.10.2 Rotation frequency of capsule in viscoelastic shear flow

A single, non-Brownian sphere in bulk simple shear flow is an interesting object of study. Assuming a no-slip boundary condition on the surface of the sphere, the total torque acting on the sphere must be zero. For a Newtonian fluid, this leads to a rotational frequency ω (rad/s) of half the vorticity, which can be expressed as [81, 82]

$$\omega = \frac{\dot{\gamma}}{2}. \tag{10.44}$$

For decreasing Ca, the steady state shape of the capsule in shear gets less prolate and finally transitions into a sphere. Therefore it doesn't surprise, that e.g. in fig. 10.20c the dimensionless rotational frequency $\omega/\dot{\gamma}$ approaches 1/2 for $\mathrm{Ca} \to 0$. On the other hand, fluids with viscoelastic properties lead to a more complex behavior. [83] model a sphere in viscoelastic fluid via a multi-mode Giesekus model and compare the results to experimental data. In [83, fig. 6] they find that all the experimental data collapse on a single master curve if plotted against the Weissenberg number. For small Weissenberg numbers the Newtonian case is reproduced. But starting from $\mathrm{Wi} \approx 1$ a strong frequency drop is found for increasing Weissenberg numbers, until finally a plateau at $\omega/\dot{\gamma} \approx 0$ is reached: at $\mathrm{Wi} = 10$ already $\omega/\dot{\gamma} < 0.02$ is valid for their models, the experimental data only reaches as far as $\mathrm{Wi} \approx 4$. In this thesis, the viscoelasticity will be modeled by FENE-P and Oldroyd-B. A layer of complexity is added by considering deformable capsules. $\mathrm{Re} = 0.05$, $\mathrm{Ca} = 0.1$ and $R = 6\Delta x$ are chosen and a domain of $L_s \times L_t \times H = 64\Delta x \times 32\Delta x \times 94\Delta x$ is used. Furthermore, simulations take place at fixed $\beta = 0.5$ and for FENE-P the cases $b = 10$ and $b = 100$ are selected. The fluid of the capsule interior can be viscoelastic itself or Newtonian, where in the latter case M4 or M1 is used. The frequencies are obtained analogous to fig 10.20c, with the only difference that the total simulation time equals $t^* = 70$.

The results are shown in fig. 10.50. Similar to the case of hard spheres in [83, fig. 6], frequency starts dropping at $\mathrm{Wi} \approx 1$. For the Oldroyd-B model, at $\mathrm{Wi} = 10$ the beginning of a plateau at $\omega \approx 0$ is only slightly visible; for FENE-P the transition to the plateau is more pronounced. For $0 \leq \mathrm{Wi} \leq 1$ another plateau is visible. Its value depends on Ca, is therefore generally different from the plateau value for hard spheres. Since the shear-thinning effect is negligible for low Wi, FENE-P and Oldroyd-B collapse on the same curve. Comparing M4 to M0, one can see that in this region the values of Newtonian capsules are shifted towards higher rotational rates in comparison to viscoelastic capsules. Furthermore, M1 transitions to M0 for small Wi, but to M4 for large Wi.

For $\mathrm{Wi} > 4$, simulations with Oldroyd-B capsules are getting unstable. The choice of a Newtonian capsule seems to have a stabilizing effect, here simulations can reach $\mathrm{Wi} = 10$. This stability advantage can already be observed in fig. 10.48a, where for $\mathrm{Wi} = 10$ the Taylor deformation parameter reaches a steady state value for Newtonian capsules (M1 and M4), but not Oldroyd-B capsules (M0). Since this unboundedly growing deformation for M0 is already present at the validation via fig. 10.46a, it is not considered a numerical, but rather a model intrinsic instability. The FENE-P model on the other hand yields stable simulations up to $\mathrm{Wi} \approx 20$ for both Newtonian as well as FENE-P capsules when using M0 and M1, respectively. Algorithm M4 gets unstable starting from $\mathrm{Wi} \approx 10$. This time a numerical instability is the reason. Already much earlier, starting from about $\mathrm{Wi} = 1$, wiggles in the deformation parameter become clear (cf. fig. A6b). They are also present for Newtonian capsules in Oldroyd-B fluid. An increase of the resolution or an improvement of the advection algorithm with the possibilities already outlined

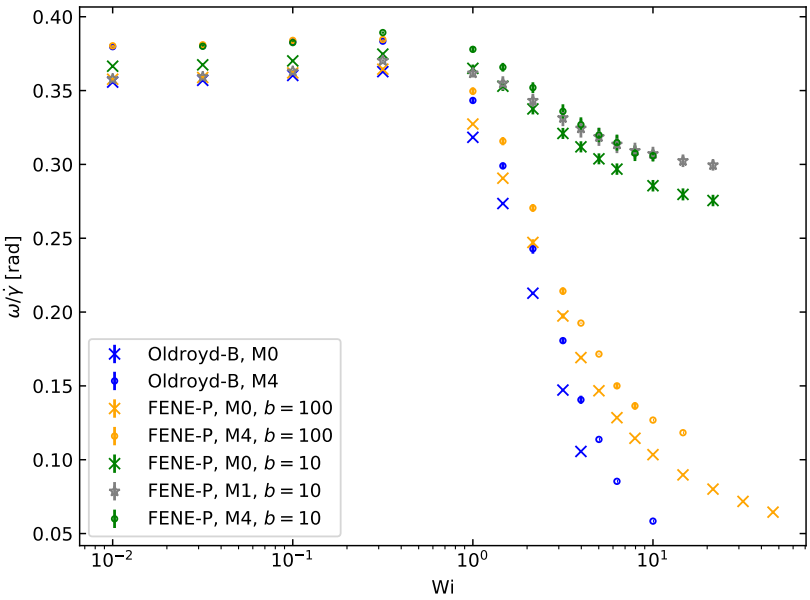


Fig. 10.50: Capsule rotation rate as a function of Weissenberg number. Results for both the FENE-P and the Oldroyd-B model, and for viscoelastic as well as Newtonian capsules.

in previous chapters could resolve this.

10.10.3 Newtonian capsule in rectangular channel filled with alginate

As a final application, it is shown how the viscoelastic models can be used in the context of biofabrication research. In extrusion-based biofabrication, a cell experiences high hydrodynamic stresses during printing. These stresses depend on the viscoelastic properties of the hydrogel, the geometry of the printer nozzle as well as the emerging flow profile. An important property of hydrogels is their shear-thinning behavior. For strongly shear-thinning fluids with $\eta(\dot{\gamma} \approx 0\,\mathrm{s}^{-1})/\eta(\dot{\gamma} = 100\,\mathrm{s}^{-1}) \propto 10^4$, the potential for stress reduction can already be seen from the flow profile: while Newtonian liquids have high shear rates nearly everywhere in the nozzle due to its parabolic velocity profile, shear-thinning fluids have low shear rates in the nozzle center where the flow profile is nearly flat [64]. However, the hydrogel modeled in this chapter will only have $\eta(\dot{\gamma} \approx 0\,\mathrm{s}^{-1})/\eta(\dot{\gamma} = 100\,\mathrm{s}^{-1}) \propto 2$, so it will be interesting to see if this still allows for a noticeable reduction of stress. Furthermore, the elasticity of the hydrogel could also play a role in the direct interaction with the cell and will be captured by the model as well.

A typical printing nozzle has a square or rectangular cross section with side length of $100-200\,\mu\mathrm{m}$ and a channel length of several millimeters. Flow speeds in the center of the channel are in the order of $1\,\mathrm{cm\,s}^{-1}$ [64]. For the simulation, a square cross section of $100\,\mu\mathrm{m} \times 100\,\mu\mathrm{m}$ is chosen. As a representative bioink, alginate $4\,\mathrm{g/dL}$ is used. It is modeled via the FENE-P viscoelastic constitutive equation, the model parameters are taken from tab. 10.1. Since the cell model from [84] is not yet implemented in FluidX3D, a capsule simulation will imitate the cell behavior. For this purpose, the Skalak model serves better than the neo-Hookean model. The shear modulus of a typical cell is between $100-1000\,\mathrm{Pa}$, its radius in the order of $10^{-5}\,\mathrm{m}$. Here, a cell of radius $R_{\mathrm{SI}} = 8.0 \times 10^{-6}\,\mathrm{m}$ with a shear modulus of $G = 100.0\,\mathrm{Pa}$

is modeled by using the moduli $\kappa_{S,1} = GR_{\mathrm{SI}}$ and $\kappa_{S,2} = 50\kappa_{S,1}$. The volume force is chosen such that at steady state a center velocity of $1.00\,\mathrm{cm\,s^{-1}}$ is reached. This leads to a shear rate of $\dot{\gamma} \approx 1.00\,\mathrm{cm\,s^{-1}}/50\,\mu\mathrm{m} = 200\,\mathrm{s^{-1}}$ and consequently to a very high Weissenberg number of $\mathrm{Wi} \approx 200\,\mathrm{s^{-1}} \times 0.14\,\mathrm{s} = 28$. For comparison, a simulation of a cell-like capsule in a Newtonian fluid is conducted. It has the same total viscosity $\eta = 4.2\,\mathrm{Pa\,s}$ as the viscoelastic fluid. Furthermore, the volume force is chosen such that the same mean velocity of $0.512\,\mathrm{cm\,s^{-1}}$ and thus the same flow rate as in the viscoelastic case is reached. The quantities that differ between the Newtonian and the viscoelastic simulation are listed in tab. 10.4.

fluid type	$u_{\max}$ [$\mathrm{cm\,s^{-1}}$]	F_x [$\mathrm{N\,m^{-3}}$]	η_s [Pa s]	η_p [Pa s]	$\mathrm{Re} \cdot \alpha_{\mathrm{Re}}$	Ca
Newtonian	1.07	12553	4.2	0.0	0.102	8.34
FENE-P	1.00	3982.8	0.9	3.3	0.095	8.40

Tab. 10.4: The quantities differing between Newtonian and viscoelastic simulations. The Reynolds number is scaled up by the Reynolds scaling parameter α_{Re} in order to speed up the simulation.

The LBM simulation domain consists of $L_x \times L_y \times L_z = 80\Delta x \times 76\Delta x \times 76\Delta x$ lattice nodes with planar boundaries in $y-$ and $z-$direction. Thus, the cell radius of $R_{\mathrm{SI}} = 8.0 \times 10^{-6}\,\mathrm{m}$ is represented by exactly 6 lattice nodes. A capsule grid resolution of $n_\Delta = 1280$ is chosen. Furthermore, capsule volume conservation is switched on. In order to speed up the simulation, a Reynolds scaling of $\alpha_{\mathrm{Re}} = 5 \times 10^3$ is used. This means that all viscosities and Skalak moduli are divided by α_{Re} before converting them to lattice units, which does not change the Capillary number. The rescaling should have a negligible effect on the whole simulation as long as $\mathrm{Re} \ll 1$ is still valid, which was successfully tested by comparing to a simulation with smaller Reynolds scaling. Moreover, it was found that simulating a capsule in a FENE-P fluid is more prone to instabilities than for the case of a Newtonian fluid. That's why for the former $\nu = 1/30$ in lattice units is chosen, while for the latter the common choice of $\nu = 1/6$ is used. This results in a five times smaller time step for the FENE-P fluid compared to the Newtonian fluid. Furthermore, it became evident that `double` accuracy is needed for this setup. When using `float`, the shear forces of the capsule cannot be resolved, resulting in the capsule mesh getting unstructured over time.

In order to look at the fluid properties only, at first simulations without capsule are conducted. The velocity profile of the Newtonian fluid in fig. 10.52a is very close to a parabolic shape which should evolve in the case of a cylindrical channel. Compared to the Newtonian case, the FENE-P fluid has a lower velocity in the channel center, but a slightly higher velocity near the walls of the channel. This effect is caused by the shear-thinning behavior of FENE-P. The two fluids differ much more with respect to the fluid stress. Comparing the FENE-P fluid to the Newtonian fluid in fig. 10.52b, the FENE-P stress σ_{xy} is less than half as large almost everywhere in the fluid. This goes well with the fact that less than half the volume force is needed for FENE-P to arrive at the same flow rate. Again, the shear-thinning behavior is the reason behind this: an Oldroyd-B fluid would yield the same steady state mean velocity as a Newtonian fluid, if the same volume force and total viscosity are used.

Now the simulation setup is completed by adding the initially spherical capsule with its *center of mass* (COM) at lattice position $\boldsymbol{r}_{\mathrm{COM}} = (L_x/2, L_y/2 + 4R, L_z/2)^{\mathrm{T}}$, i.e., shifted by 4 capsule radii from the channel center. In the case of the FENE-P fluid, the interior fluid of

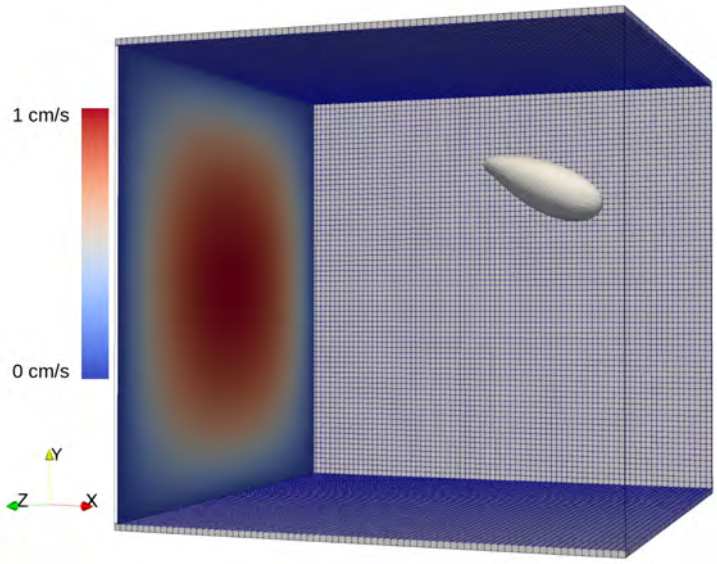
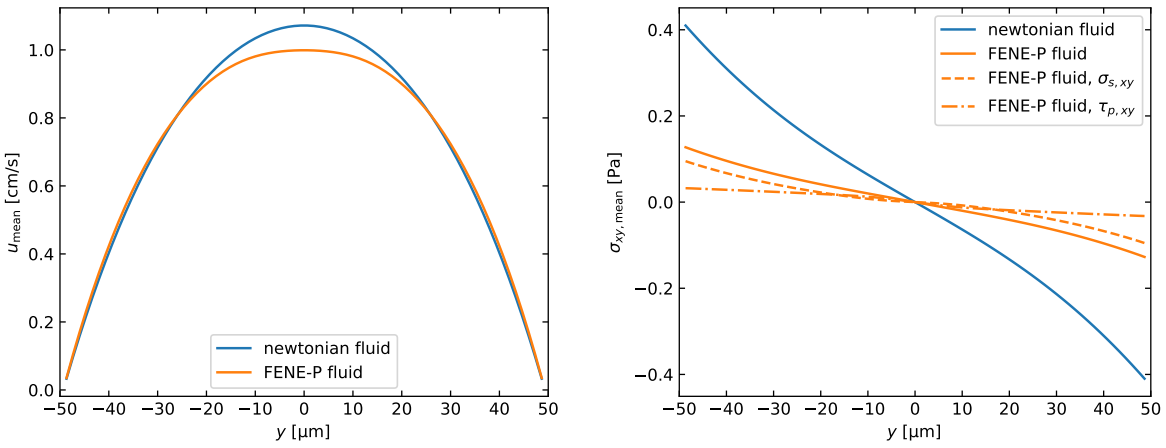


Fig. 10.51: Newtonian capsule in a rectangular channel using bioprinting parameters, 0.1 s after switching on the volume force. The case of a FENE-P fluid using algorithm M1 is depicted. The full simulation domain is shown. All wall nodes besides the front wall are depicted in gray, the side length of a single square corresponds to the lattice spacing Δx. The slice at $x = 0$ shows the fully developed velocity profile.



(a) (b)

Fig. 10.52: Mean values of (a) fluid velocity and (b) the $xy-$component of the fluid stress. The mean over the channel length is taken at slice $z = H/2$ and is resolved for different y-positions.

the capsule is kept Newtonian via the algorithm M1. As discussed in the previous chapter, algorithm M4 is no longer stable at such a high Weissenberg number. First, this Newtonian capsule in a viscoelastic fluid (FENE-P fluid M1) is compared to the Newtonian capsule in Newtonian fluid. Looking at the COM in fig. 10.53a, both capsules travel 2 cm in x-direction during 2 s. This can be explained by fig. 10.53b: both capsules migrate towards the channel center, where $u_x \approx 1\,\mathrm{cm\,s^{-1}}$ holds. The migration of the FENE-P fluid M1 happens on a much faster timescale, however. Furthermore, fig. 10.54 shows that the Taylor deformation D of the capsule in Newtonian fluid is much higher than the one in FENE-P fluid. This can be explained by the differing values for σ_{xy} from fig. 10.52b. In both cases, D decreases as the capsule migrates towards the channel center. Looking at the rotation angle $\phi_{\mathrm{rot}}(t)$ in fig. 10.53c, not much deviation between the Newtonian fluid and the viscoelastic fluid M1 is present during the first two oscillations. The rotation frequency drop for viscoelastic fluids at $\mathrm{Wi} > 1$ does not play a big role here, since a FENE-P fluid with $b = 10$ is simulated (see fig. 10.50). After the first two oscillations, the rotational frequency is slowed down due to the migration towards the channel center.

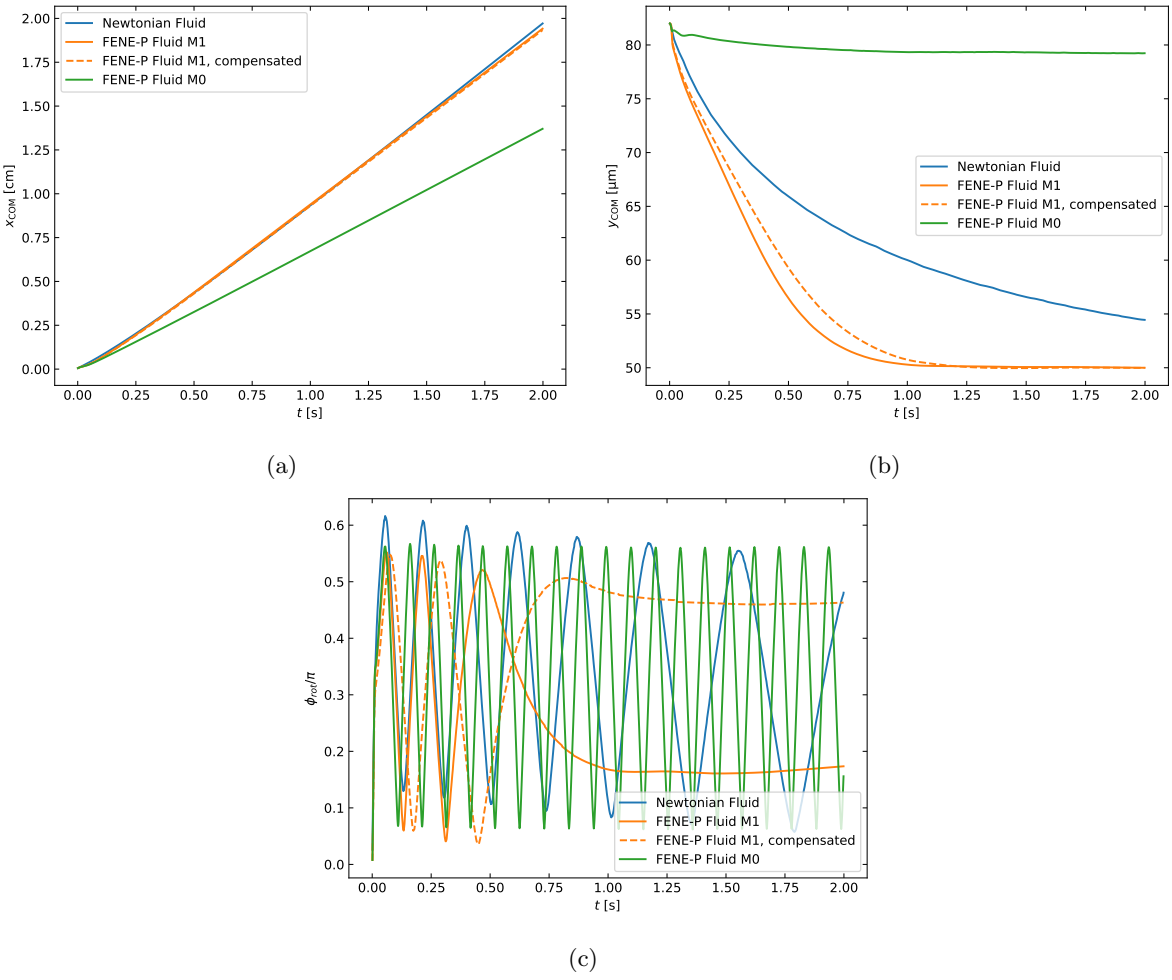


Fig. 10.53: Capsule with cell-like properties in rectangular channel with bioprinting parameters. The x-position (stream-flow direction) (a) and y-position (center offset direction) (b) is given as well as the rotation angle ϕ_{rot} (c).

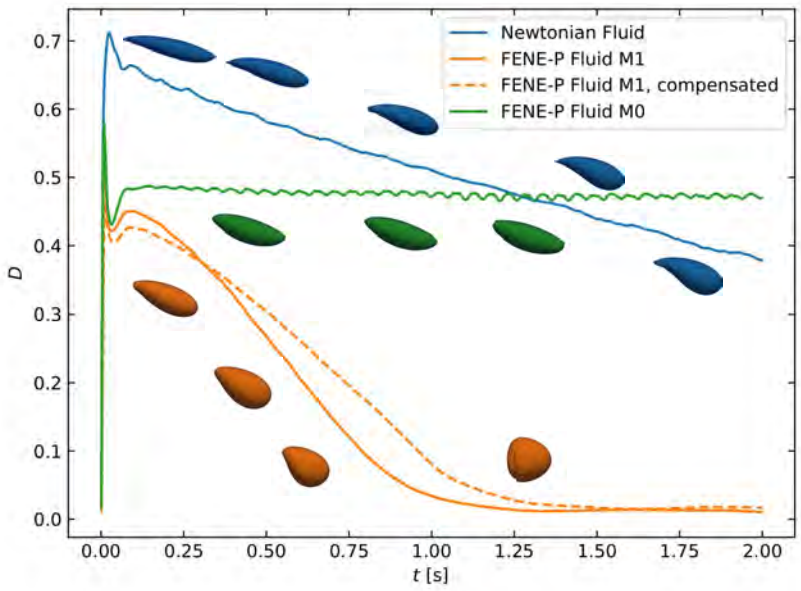


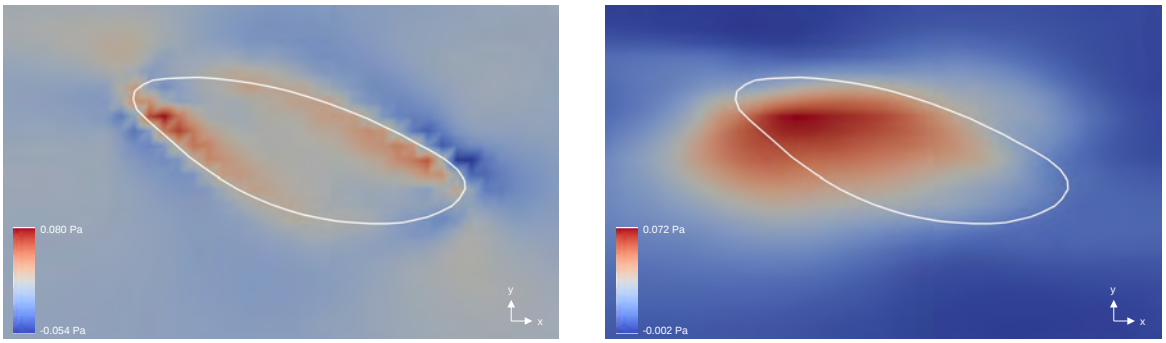
Fig. 10.54: Deformation of capsule with cell-like properties in rectangular channel with bioprinting parameters.

The speed-up of the migration velocity of the FENE-P fluid M1 is an unexpected result. By extending the ESPResSo software, Sebastian Müller conducted IB-LBM simulations of cells [84] inside a Carreau-Yasuda fluid [64], which is a shear-thinning fluid without an elastic component. Using the same fluid parameters for interior and exterior fluid, he found a slow-down of migration velocity compared to Newtonian fluids[19]. This motivates the simulation of a FENE-P fluid with algorithm M0, where also no distinction between interior and exterior fluid is made. Here, the initial deformation of the capsule is nearly the same as in the case of algorithm M1, but the migration towards the channel center is not present at all. This leads to a smaller travel distance after 2 s and a higher rotational frequency because of the bigger shear rates near the channel walls. One could argue that the discrepancy between M0 and M1 is caused by an effective viscosity contrast: since for M1 the viscoelastic force is switched off inside the capsule, it holds $\eta = \eta_s$ for the interior fluid. A reduced total viscosity might lead to a higher migration velocity. To test for this hypothesis, a simulation with $\eta_{s,\text{interior}} = \Lambda \eta_{s,\text{exterior}}$ is conducted for the FENE-P fluid M1, where $\Lambda = \frac{\eta_p + \eta_{s,\text{exterior}}}{\eta_{s,\text{exterior}}} = 4.67$ holds. It is indicated with dotted lines in fig. 10.53. It becomes obvious that a compensation for the effective viscosity contrast does not qualitatively change the simulation results - the speed-up of migration velocity is still present for M1.

An other explanation for the different behavior of M0 and M1 is found when looking at the stress near the capsule. A volume force in y-direction is caused by several components of the stress tensor: $F_y^p = \nabla \cdot (\sigma_{xy}, \sigma_{yy}, \sigma_{zy})$. Looking at the data, the component σ_{yy} stands out among the three terms for both its high stress and sharp stress gradient. In fig. 10.55, $\sigma_{s,yy}$ and τ_{yy} near the capsule are depicted for the FENE-P fluid M0 at time $t = 0.12$ s. At this point, there is little difference between the M0 and M1 data. Furthermore, the distribution of the Newtonian stress around the capsule from fig. 10.55a is qualitatively very similar to the case of a purely Newtonian fluid - only the absolute values of the occurring stress are about a factor $\propto 2.5$ lower for the FENE-P fluid due to the smaller solvent viscosity η_s. Looking at fig. 10.55b, inside the capsule

[19] Unpublished results.

there is a positive polymer stress gradient from the lower right to the upper left edge leading to a positive volume force F_y^p. This force might prevent the capsule from migrating towards the channel center when using algorithm M0. This is not the case when using algorithm M1, since the force inside the capsule is switched off. All in all, it becomes evident that the concrete modeling of the stress in the vicinity of the capsule has a major impact on migration velocity.



(a) (b)

Fig. 10.55: Stress of a FENE-P fluid M0 caused by the presence of a capsule in the rectangular channel setup simulated with bioprinting parameters. The components $\sigma_{s,yy}$ (a) and τ_{yy} (b) are sliced at $z = L_z/2$. The frames are taken at $t = 0.12\,\mathrm{s}$. The extent of the capsule is indicated as a white line.

11 Conclusion

In the first part of this work, bubbles dissolved in water were modeled using a coupling of the single-phase incompressible NSE and the ideal gas equation. The former was simulated using the LBM extended by the VoF method. For this, an own method had to be developed which can correctly handle bubble merge and split processes. While the first idea of an implicit pressure equilibration failed due to the high and hard to estimate number of equilibration steps required per simulation step, the second idea of explicit bubble tracking was a success. Using the Hoshen-Kopelman algorithm for global bubble tracking in LBM simulations is a novelty in the literature and offers a considerable advantage over the floodfill algorithm in terms of runtime. The fact that the HK algorithm is simultaneously used for local split/merge detection establishes an impressive consistency between global tracking and local detection that is also unparalleled in the literature. Although not all calculations can take place locally for the bubble extension and therefore PCIe transfers and expensive CPU calculations have to be performed, the overall performance only drops by a factor of two compared to equivalent pure GPU simulations where the bubble extension is deactivated. This is thanks to optimized buffers and additional CPU parallelization. Using the Rayleigh-Plesset equation to solve for the bubble interface position in spherical geometries, accuracy well into the sub-lattice range was demonstrated. For ascending bubbles, the evolving stationary shapes were studied depending on the given Morton and Bond number. For large parts of the regime $1 \leq \mathrm{Bo} \leq 10^3$ and $10^{-12} \leq \mathrm{Mo} \leq 10^8$ good agreement with the literature was found. In the same regime, the rising velocity of simulated bubbles was determined and compared with the expected terminal velocities, which showed large deviations in some cases. These deviations can partially be explained by the limited setup, but they also have to do with the problem of "stuck bubbles", i.e. the phenomenon that bubbles with $\mathrm{Bo} \lesssim 1$ do not rise during the simulations performed. By investigating falling droplets in the same parameter range it was proven that this phenomenon must have to do with the VoF implementation, i.e., the problem is independent of the bubble extension itself. In the case of the simulation of bursting bubbles, a far more fundamental limitation was shown. The length scales involved in a bursting process can by far not be resolved with the existing algorithm due to memory limitations. As a possible solution, an adaptation of the VoF method as well as the HK algorithm was proposed, which would allow simulation of the bubble lamella with sub-grid resolution. This would be another essential step to provide a basis for studying the exchange of microplastics at the water-air interface due to rising and bursting bubbles. However, it was discovered that already in the existing implementation with poorly resolved lamella, the correct order of magnitude for the rim velocity is established upon rupture of the liquid film.

The second part of this work aimed to lay the foundations for the simulation of cells under bioprinting conditions. A first important step was the implementation of suitable polymer models. It was found that the Oldroyd-B model is capable of reproducing the viscoelastic properties of linear polymers, which becomes apparent when investigating their storage and loss moduli. With the FENE-P model, the shear-thinning property can additionally be captured. The fact that both models contain only first-order time derivatives of stress made a simple implementation via the FVM possible. The CTU advection scheme contributed to a significant improvement

of the stability. However, more detailed investigations showed that an upwind scheme with first order flux in all directions defined by the used lattice neighborhood would bring further advantages when dealing with the staircase effect. Fits to rheological data showed that the FENE-P model can capture the essential properties of the polymers used in bioprinting. Two improvements were suggested for more accurate modeling: the single-mode FENE-P model could easily be extended to a multiple-mode FENE-P model, where presumably already two modes would be sufficient to model most alginates. Furthermore, modifying the dumbbell spring-law would lift the restriction on shear-thinning fluid with a power-law with exponent 1/3. However, implementing the polymer models in the first instance without such extensions offered the advantage of having analytical solutions for a number of simple flow geometries. Thus, the viscoelastic behavior of the Oldroyd-B model was validated by simulating a rheometer. Through a precise understanding of the LB boundary conditions, it was possible to implement the FV boundary conditions in such a way that a significant improvement compared to [69] was achieved regarding the validation via inception of planar poiseuille flow. For the FENE-P model, the expected shear-thinning behavior was accurately reproduced in steady shear flow simulations. Despite a rather simple implementation of equilibrium boundary conditions for the advection of stress, the proper behavior of the elongational viscosity was achieved within acceptable error bounds. Finally, the correct time evolution of the stress tensor at cessation of steady shear flow was established for a remarkably wide parameter range of the finite extensibility parameter b. For preliminary investigations of the interaction of viscoelastic fluids with soft objects, capsules were chosen, which were simulated via the IBM. First, the FluidX3D implementation of the IBM had to be validated against the ESPResSo implementation by looking at the case of a purely Newtonian fluid. Investigating the capsule in simple shear-flow, good agreement for quantities like the Taylor deformation, the inclination angle and the rotation frequency was found. Another result of the validation process was that both programs lag significantly behind the accuracy of the BIM for simulations with viscosity contrast and high capillary numbers. However, it was shown that the deviations become smaller with increasing resolution. Furthermore, an improvement of the viscosity contrast computation based on the VoF was suggested. For a capsule having the same Oldroyd-B parameters for the interior and exterior fluid, simulation results from [68] were used for validation, where good agreement in the time evolution of the Taylor deformation was found up to Weissenberg numbers $\mathrm{Wi} = 100$. As a next step, four methods were developed to simulate capsules with Newtonian interior fluid and viscoelastic exterior fluid. For method M4, great efforts were made to guarantee stress conservation while at the same time preventing stress build-up due to the staircase effect. In application, however, hardly any difference between method M4 with linear interpolation to zero and the much simpler method M1 was found. Only for Weissenberg numbers $\mathrm{Wi} < 1$ a difference in the dimensionless rotational frequency was observed, furthermore M1 has increased stability compared to M4 for large Weissenberg numbers. This time, simulation data from [78] was used for validation, where a capsule with Newtonian interior fluid and Oldroyd-B exterior fluid was considered. No satisfactory validation was achieved here, the deviations can probably be explained by the fundamentally different methods for eliminating viscoelasticity inside the capsules. Regarding new applications, to the best of our knowledge we are the first to simulate an IB capsule in a FENE-P fluid. Therefore, this thesis can present new simulation data for the time-evolution of the inclination angle as well as the Taylor deformation D of a capsule in FENE-P simple shear flow, where the divergence of D for $\mathrm{Wi} \gtrsim 10$ found in the case of an Oldroyd-B fluid is suppressed. Furthermore, with these models it was possible to reproduce the expected drop of the dimensionless rotational frequency for objects in viscoelastic shear flow at $\mathrm{Wi} \approx 1$. First results were provided on how the size of the frequency drops is related to the finite extensibility, i.e. to the non-linearity of

polymer spring-law. Finally, the possible application for computational problems in bioprinting was demonstrated by simulating a rectangular channel filled with alginate. With proper rescaling, the desired parameter set was stably accessed. By comparison to a Newtonian fluid it was shown, that shear-thinning on the one side leads to a reduction of capsule deformation and on the other side prevents the capsule from experiencing a frequency drop. Furthermore, the differences between a FENE-P capsule (algorithm M0) and a Newtonian capsule (algorithm M1) dissolved in a FENE-P fluid are more noticeable than expected: The former has an increased migration velocity towards the center of the channel compared to the purely Newtonian case, whereas in the latter the migration is completely suppressed. As a first explanation, a stress gradient in the viscoelastic stress tensor $\underline{\tau}$ inside the capsule was suggested. To further investigate the issue, it would be desirable to improve the algorithm M4 in a way that it also remains stable at these high shear rates.

References

[1] J. Malda et al. “25th anniversary article: Engineering hydrogels for biofabrication.” In: *Advanced Materials* 25.36 (2013), pp. 5011 – 5028. DOI: 10.1002/adma.201302042.

[2] I. Donderwinkel et al. “Bio-inks for 3D bioprinting: recent advances and future prospects.” In: *Polym. Chem* 8 (2017), pp. 4451 – 4471. DOI: 10.1039/c7py00826k.

[3] N. Noor et al. “3D Printing of Personalized Thick and Perfusable Cardiac Patches and Hearts.” In: *Advanced Science* 6.11 (2019). DOI: 10.1002/advs.201900344.

[4] J. Gopinathan et al. “Recent trends in bioinks for 3D printing.” In: *Biomaterials Research* (2018). DOI: 10.1186/s40824-018-0122-1.

[5] W. Sun et al. “The bioprinting roadmap.” In: *Biofabrication* 12.2 (2020). DOI: 10.1088/1758-5090/ab5158.

[6] J. Hazur et al. “Improving alginate printability for biofabrication: establishment of a universal and homogeneous pre-crosslinking technique.” In: *Biofabrication* 12.4 (2020). DOI: 10.1088/1758-5090/ab98e5.

[7] K. Pabortsava et al. “High concentrations of plastic hidden beneath the surface of the Atlantic Ocean.” In: *Nature Communications* (2020). DOI: 10.1038/s41467-020-17932-9.

[8] T.-B. Robinson et al. “Riding the Plumes: Characterizing Bubble Scavenging Conditions for the Enrichment of the Sea-Surface Microlayer by Transparent Exopolymer Particles.” In: *Atmosphere* 10.454 (2019). DOI: 10.3390/atmos10080454.

[9] M. Trainic et al. “Airborne microplastic particles detected in the remote marine atmosphere.” In: *Communications Earth & Environment* (2020). DOI: 10.1038/s43247-020-00061-y.

[10] M. Lehmann et al. “Ejection of marine microplastics by raindrops: a computational and experimental study.” In: *Microplastics and Nanoplastics* 1.18 (2021), pp. 1 – 19. DOI: 10.1186/s43591-021-00018-8.

[11] M. Lehmann. “High Performance Free Surface LBM on GPUs.” MSc thesis. Universität Bayreuth, 2019. DOI: 10.15495/EPub_UBT_00005400.

[12] H. Limbach et al. “ESPResSo—an extensible simulation package for research on soft matter systems.” In: *Computer Physics Communications* 174.9 (2006), pp. 704 – 727. DOI: 10.1016/j.cpc.2005.10.005.

[13] K. Tůma. “Identification of rate type fluids suitable for modeling geomaterials.” PhD thesis. Univerzita Karlova v Praze, 2014.

[14] W. Kuhn. “Über die Gestalt fadenförmiger Moleküle in Lösungen.” In: *Kolloid-Zeitschrift* 68.1 (1934), pp. 2 – 15.

[15] N. Phan-Thien et al. *Understanding viscoelasticity: an introduction to rheology.* 2nd ed. Springer, 2013.

[16] F. Osmanlic et al. “Lattice Boltzmann method for Oldroyd-B fluids.” In: *Computers and Fluids* 124 (2016), pp. 190 – 196. DOI: 10.1016/j.compfluid.2015.08.004.

[17] L. J. Amoreira et al. "Comparison of different formulations for the numerical calculation of unsteady incompressible viscoelastic fluid flow." In: *Advances in Applied Mathematics and Mechanics* 2.4 (2010), pp. 483 – 502. DOI: 10.4208/aamm.10-m1010.

[18] J. G. Oldroyd. "On the formulation of rheological equations of state." In: *Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences* 200.1063 (1950), pp. 523 – 541.

[19] R. B. Bird et al. "Polymer solution rheology based on a finitely extensible bead-spring chain model." In: *Journal of Non-Newtonian Fluid Mechanics* 7.2-3 (1980), pp. 213 – 235. DOI: 10.1016/0377-0257(80)85007-5.

[20] M. Herrchen et al. "A detailed comparison of various FENE dumbbell models." In: *Journal of Non-Newtonian Fluid Mechanics* 68.1 (1997), pp. 17 – 42. DOI: 10.1016/S0377-0257(96)01498-X.

[21] P. J. Oliveira. "Alternative derivation of differential constitutive equations of the Oldroyd-B type." In: *Journal of Non-Newtonian Fluid Mechanics* 160.1 (2009), pp. 40 – 46. DOI: 10.1016/j.jnnfm.2008.11.013.

[22] R. B. Bird et al. *Dynamics of polymeric liquids Vol. 2, Kinetic theory*. 1987.

[23] A. Gupta et al. "Hybrid Lattice Boltzmann / Finite Difference simulations of viscoelastic multicomponent flows in confined geometries." In: *Journal of Computational Physics* 291 (2015), pp. 177 – 197. DOI: 10.1016/j.jcp.2015.03.006.

[24] A. K. Townsend et al. "Small- and large-amplitude oscillatory rheometry with bead–spring dumbbells in Stokesian Dynamics to mimic viscoelasticity." In: *Journal of Non-Newtonian Fluid Mechanics* 261 (2018), pp. 136 – 152. DOI: 10.1016/j.jnnfm.2018.08.010.

[25] R. B. Bird et al. *Dynamics of Polymer Liquids Vol. 1*. 1987.

[26] T. Krüger et al. *The lattice boltzmann method, principles and practice*. 2017. DOI: 10.1007/978-3-319-44649-3.

[27] M. Kuron et al. "Moving charged particles in lattice Boltzmann-based electrokinetics." In: *Journal of Chemical Physics* 145.21 (2016). DOI: 10.1063/1.4968596. arXiv: 1607.04572.

[28] S. Izquierdo et al. "Analysis of open boundary effects in unsteady lattice Boltzmann simulations." In: *Computers & Mathematics with Applications* 58.5 (2009), pp. 914 – 921. DOI: 10.1016/J.CAMWA.2009.02.014.

[29] C. Körner et al. "Lattice Boltzmann model for free surface flow for modeling foaming." In: *Journal of Statistical Physics* 121.1-2 (2005), pp. 179 – 196. DOI: 10.1007/s10955-005-8879-8.

[30] S. Bogner et al. "Curvature estimation from a volume-of-fluid indicator function for the simulation of surface tension and wetting with a free-surface lattice Boltzmann method." In: *Physical Review E* 93.4 (2016). DOI: 10.1103/PhysRevE.93.043302. arXiv: 1509.07691.

[31] S. Bogner et al. "Boundary conditions for free interfaces with the lattice Boltzmann method." In: *Journal of Computational Physics* 297 (2015), pp. 1 – 12. DOI: 10.1016/j.jcp.2015.04.055. arXiv: 1409.5645.

[32] T. Krüger. *Introduction to the immersed boundary method*. 2011. URL: http://s467657437.online.de/wp-content/uploads/2019/08/Krueger%7B%5C_%7DEdmonton%7B%5C_%7DIBM.pdf (accessed on 11/12/2021).

[33] T. Krüger et al. "Efficient and accurate simulations of deformable particles immersed in a fluid using a combined immersed boundary lattice Boltzmann finite element method." In: *Computers and Mathematics with Applications* 61.12 (2011), pp. 3485 – 3505. DOI: 10.1016/j.camwa.2010.03.057. arXiv: 1004.2416.

[34] C. Loop. "Smooth Subdivision Surfaces Based on Triangles (MSc thesis)." PhD thesis. University of Utah, 1987.

[35] A. Guckenberger et al. "On the bending algorithms for soft objects in flows." In: *Computer Physics Communications* 207 (2016), pp. 1 – 23. DOI: 10.1016/j.cpc.2016.04.018.

[36] R. Skalak. "Strain Energy Function of Red Blood Cell Membranes." In: *Biophysical Journal* 13.3 (1973), pp. 245 – 264.

[37] T. Krüger. *Computer simulation study of collective phenomena in dense suspensions of red blood cells under shear*. Vieweg+Teubner Verlag, 2012. DOI: 10.1007/978-3-8348-2376-2.

[38] G. Gompper et al. "Random Surface Discretizations and the Renormalization of the Bending Rigidity." In: *Journal de Physique I* 6.10 (1996), pp. 1305 – 1320. DOI: 10.1051/JP1:1996246.

[39] M. Lehmann et al. "Efficient viscosity contrast calculation for blood flow simulations using the lattice Boltzmann method." In: *International Journal for Numerical Methods in Fluids* 92.11 (2020), pp. 1463 – 1477. DOI: 10.1002/fld.4835.

[40] A. Munshi. "The OpenCL Specification." In: *IEEE Hot Chips 21 Symposium (HCS)* (2012).

[41] F. Häusl. "MPI-based multi-GPU extension of the Lattice Boltzmann Method." BSc thesis. Universität Bayreuth, 2019. DOI: https://doi.org/10.15495/EPub_UBT_00005689.

[42] D. Anderl et al. "Free surface lattice Boltzmann with enhanced bubble model." In: *Computers and Mathematics with Applications* 67.2 (2014), pp. 331 – 339. DOI: 10.1016/j.camwa.2013.06.007. arXiv: 1604.01632.

[43] J. D. Foley et al. *Computer graphics: principles and practice*. 3rd ed. Vol. 12110. Addison-Wesley Professional, 1996.

[44] J. Hoshen et al. "Percolation and cluster distribution. I. Cluster multiple labeling technique and critical concentration algorithm." In: *Physical Review B* 14.8 (1976), pp. 3438 – 3445. DOI: 10.1103/PhysRevB.14.3438.

[45] J. Erickson. "Algorithms Lecture: Disjoint Sets." In: *Algorithms*. 2018. Chap. 11.

[46] T. Fricke. *The Hoshen-Kopelman Algorithm*. 2004. URL: https://www.ocf.berkeley.edu/%7B~%7Dfricke/projects/hoshenkopelman/hoshenkopelman.html (accessed on 09/17/2021).

[47] S. Frijters et al. "Parallelised Hoshen-Kopelman algorithm for lattice-Boltzmann simulations." In: *Computer Physics Communications* 189 (2015), pp. 92 – 98. DOI: 10.1016/j.cpc.2014.12.014. arXiv: 1405.1931.

[48] M. Plesset. "The dynamics of cavitation bubbles." In: *Journal of Applied Mechanics* 16 (1949), pp. 277 – 282.

[49] E. Hairer et al. *Solving Ordinary Differential Equations II*. Vol. 375. Springer Series in Computational Mathematics. Springer Berlin Heidelberg, 1996. DOI: 10.1007/978-3-662-09947-6.

[50] G. Saritha et al. “Development and application of a high density ratio pseudopotential based two-phase LBM solver to study cavitating bubble dynamics in pressure driven channel flow at low Reynolds number.” In: *European Journal of Mechanics, B/Fluids* 75 (2019), pp. 83 – 96. DOI: 10.1016/j.euromechflu.2018.12.004.

[51] R. Clift et al. *Bubbles, Drops, and Particles*. Courier Corporation, 1978.

[52] L. Amaya-Bower et al. “Single bubble rising dynamics for moderate Reynolds number using Lattice Boltzmann Method.” In: *Computers and Fluids* 39.7 (2010), pp. 1191 – 1207. DOI: 10.1016/j.compfluid.2010.03.003.

[53] S. Donath. “Wetting Models for a Parallel High-Performance Free Surface Lattice Boltzmann Method.” PhD thesis. Universität Erlangen-Nürnberg, 2011.

[54] R. Davies et al. “The mechanics of large bubbles rising through extended liquids and through liquids in tubes.” In: *Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences* 200.1062 (1950), pp. 375 – 390. DOI: 10.1098/RSPA.1950.0023.

[55] H. D. Mendelson. “The prediction of bubble terminal velocities from wave theory.” In: *AIChE Journal* 13.2 (1967), pp. 250 – 253. DOI: 10.1002/AIC.690130213.

[56] W. Rybczynski. “On the translatory motion of a fluid sphere in a viscous medium.” In: *Bull. Acad. Sci., Cracow, Series A* 40 (1911), pp. 33 – 78.

[57] J. S. Hadamard. “Mouvement permanent lent d’une sphere liquid et visqueuse dans un liquide visqueux.” In: *CR Hebd. Seances Acad. Sci. Paris* 152 (1911), pp. 1735 – 1738.

[58] A. Nikolov et al. “Air bubble bursting phenomenon at the air-water interface monitored by the piezoelectric-acoustic method.” In: *Advances in Colloid and Interface Science* 272 (2019). DOI: 10.1016/j.cis.2019.101998.

[59] A. Pandit et al. “Hydrodynamics of the rupture of thin liquid films.” In: *Journal of Fluid Mechanics* 212 (1990), pp. 11 – 24.

[60] A. Naillon et al. “Dynamics of particle migration in confined viscoelastic Poiseuille flows.” In: *Physical Review Fluids* 4.5 (2019), pp. 1 – 16. DOI: 10.1103/PhysRevFluids.4.053301. arXiv: 1812.09505.

[61] C. Rodríguez-Rivero et al. “Rheological characterization of commercial highly viscous alginate solutions in shear and extensional flows.” In: *Rheologica Acta* 53.7 (2014), pp. 559 – 570. DOI: 10.1007/s00397-014-0780-4.

[62] J. Groll et al. *SFB / TRR 225 Biofabrication Würzburg Erlangen Bayreuth*. 2018. URL: http://trr225biofab.de/ (accessed on 11/07/2021).

[63] R. Luxenhofer. “Poly (2-oxazoline) s as Polymer Therapeutics.” In: *Macromolecular rapid communications* 33.19 (2012), pp. 1613 – 1631. DOI: 10.1002/marc.201200354.

[64] S. J. Müller et al. “Flow and hydrodynamic shear stress inside a printing needle during biofabrication.” In: *PLOS ONE* 15.7 (2020), pp. 1 – 15. DOI: 10.1371/journal.pone.0236371.

[65] Y. Mu et al. “Finite element simulation of three-dimensional viscoelastic planar contraction flow with multi-mode FENE-P constitutive model.” In: *Polymer Bulletin* 71.12 (2014), pp. 3131 – 3150. DOI: 10.1007/s00289-014-1241-z.

[66] J. Onishi et al. “A Lattice Boltzmann model for polymeric liquids.” In: *Progress in Computational Fluid Dynamics* 5.1-2 (2005), pp. 75 – 84. DOI: 10.1504/PCFD.2005.005819.

[67] J. Onishi et al. "Dynamic simulation of multi-component viscoelastic fluids using the lattice Boltzmann method." In: *Physica A: Statistical Mechanics and its Applications* 362.1 (2006), pp. 84 – 92. DOI: 10.1016/J.PHYSA.2005.09.022.

[68] J. Ma et al. "An immersed boundary-lattice Boltzmann method for fluid-structure interaction problems involving viscoelastic fluids and complex geometries." In: *Journal of Computational Physics* 415 (2020). DOI: 10.1016/j.jcp.2020.109487.

[69] M. Kuron et al. "An extensible lattice Boltzmann method for viscoelastic flows : complex and moving boundaries in Oldroyd-B fluids." In: *The European Physical Journal E* 44.1 (2021), pp. 1 – 14. arXiv: arXiv:2009.12279v1.

[70] N. D. Waters et al. "Unsteady flow of an elastico-viscous liquid." In: *Rheologica Acta* 9.3 (1970), pp. 345 – 355. DOI: 10.1007/BF01975401.

[71] S. Ramanujan et al. "Deformation of liquid capsules enclosed by elastic membranes in simple shear flow: Large deformations and the effect of fluid viscosities." In: *Journal of Fluid Mechanics* 361 (1998), pp. 117 – 143. DOI: 10.1017/S0022112098008714.

[72] D. V. Le et al. "Large deformation of liquid capsules enclosed by thin shells immersed in the fluid." In: *Journal of Computational Physics* 229.11 (2010), pp. 4097 – 4116. DOI: 10.1016/j.jcp.2010.01.042.

[73] D. V. Le et al. "A front-tracking method with Catmull-Clark subdivision surfaces for studying liquid capsules enclosed by thin shells in shear flow." In: *Journal of Computational Physics* 230.9 (2011), pp. 3538 – 3555. DOI: 10.1016/j.jcp.2011.01.047.

[74] A. Saadat et al. "Immersed-finite-element method for deformable particle suspensions in viscous and viscoelastic media." In: *Physical Review E* 98.6 (2018), pp. 1 – 17. DOI: 10.1103/PhysRevE.98.063316. arXiv: 1807.08219.

[75] R. J. LeVeque. *Finite volume methods for hyperbolic problems*. Cambridge university press, 2004.

[76] M. Bauer et al. "Code Generation for Massively Parallel Phase-Field Simulations." In: *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis* (2019). DOI: 10.1145/3295500.

[77] S. V. Patankar. *Numerical Heat Transfer and Fluid Flow*. 1st ed. CRC Press, 1980. DOI: 10.1201/9781482234213.

[78] A. H. Raffiee et al. "Deformation and buckling of microcapsules in a viscoelastic matrix." In: *Physical Review E* 96.3 (2017), pp. 1 – 9. DOI: 10.1103/PhysRevE.96.032603.

[79] N. Aggarwal et al. "Deformation and breakup of a viscoelastic drop in a Newtonian matrix under steady shear." In: *Journal of Fluid Mechanics* 584 (2007), pp. 1 – 21. DOI: 10.1017/S0022112007006210.

[80] N. Rivas et al. "Mesoscopic electrohydrodynamic simulations of binary colloidal suspensions." In: *The Journal of Chemical Physics* 148.14 (2018), p. 144101. DOI: 10.1063/1.5020377.

[81] B. J. Trevelyan et al. "Particle motions in sheared suspensions. I. Rotations." In: *Journal of Colloid Science* 6.4 (1951), pp. 354 – 367. DOI: 10.1016/0095-8522(51)90005-0.

[82] A. Einstein. "Eine neue Bestimmung der Moleküldimensionen." In: *Annalen der Physik* 324.19 (1906), pp. 289 – 306. DOI: 10.1002/ANDP.19063240204.

[83] F. Snijkers et al. "Effect of viscoelasticity on the rotation of a sphere in shear flow." In: *Journal of Non-Newtonian Fluid Mechanics* 166.7-8 (2011), pp. 363 – 372. DOI: 10.1016/j.jnnfm.2011.01.004.

[84] S. J. Müller et al. "A hyperelastic model for simulating cells in flow." In: *Biomechanics and Modeling in Mechanobiology* 20.2 (2021), pp. 509 – 520. DOI: 10.1007/s10237-020-01397-2. arXiv: 2003.03130.

Appendix

A1 Further advection tests

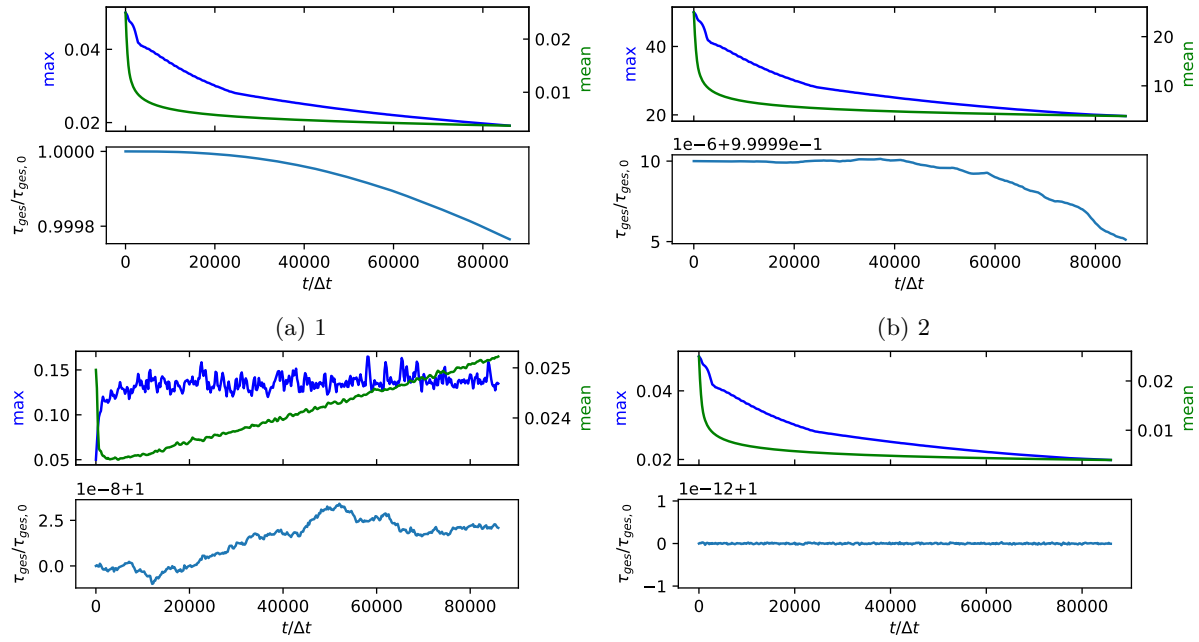
Since a bug in the **CTU** algorithm was suspected, the following things were assured:

- For the setup as described above, the two adjacent nodes at position $\boldsymbol{r} = (0,0,0)^{\mathrm{T}}$ and $\boldsymbol{r} + \boldsymbol{c}_1 = (1,0,0)^{\mathrm{T}}$ were considered. It was verified that $J^{(\mathrm{out})}_{Q,1}((0,0,0)^{\mathrm{T}}, t) = J^{(\mathrm{in})}_{Q,1}((1,0,0)^{\mathrm{T}}, t)$ holds. This is the case, as found by bitwise comparison.
- Initializing τ to exactly 1 everywhere, τ_{ges} is conserved exactly for both advection schemes (**simple** and **CTU**) and for all velocity sets.
- Initializing τ randomly as described in chapter 10.8.2, but specifying the flow direction-dependent but location-independent (done here using $J^{(\mathrm{out})}_{Q,i}(\boldsymbol{r}, t) = i \times 0.001 = J^{(\mathrm{in})}_{Q,i}(\boldsymbol{r}, t)$), τ_{ges} is again obtained exactly.

So a faulty implementation of **CTU** is unlikely based on the above tests. The following things were tested to investigate the loss of τ further:

1. The values needed for the calculation of $J^{(\mathrm{out})}_Q$ and $J^{(\mathrm{in})}_Q$ are cast to `double` after reading from their `float` buffers. All further computation takes place in `double`, only the new value $\tau(\boldsymbol{r}, t + \Delta t)$ is cast back to `float` when saved. This leads to no noticeable improvement (cf. fig. A1a).
2. The stress gets another order of magnitude: instead of $\tau = 1 \pm 0.05$, $\tau = (1 \pm 0.05) \times 10^3$ is used. This leads to an improvement of the conservation by almost two orders of magnitude (cf. fig. A1b). Moreover, τ_{ges} is now no longer monotonically decreasing. The reason for this is not clear.
3. In **CTU**, for $J^{(\mathrm{out})}_Q$ normally only the own value of τ is used, for $J^{(\mathrm{in})}_Q$ only the neighbor value. Now this is changed to the mean value of own and neighbor value in both cases. This leads to similar results as **simple** (cf. fig. A1c). Moreover, there is no numerical diffusion anymore, the distribution of τ at the end of the simulation resembles the results of **simple** from fig. 10.23b.
4. Again all computations take place in `double`, but this time τ is stored as `double` field. This leads to the conservation of τ with numerical precision (cf. fig. A1d). Still, numerical diffusion as in fig. 10.23c can be observed, it is a property of the **CTU** algorithm.

Overall, no explanation for this behavior was found. In particular, it was expected that (except for the precision, i.e. the magnitude of the numerical fluctuation around τ_{ges}) there would be no fundamental difference between `float` and `double`.



(a) 1 (b) 2

(c) 3 (d) 4

Fig. A1: Tests concerning the conservation of τ using the **CTU** advection algorithm.

A2 Equality of Oldroyd-B formulations

Equality of *original* and *polymer science* formulation of Oldroyd-B:

$$\underline{\sigma} \overset{(4.10)}{=} \underline{\tau}_p + 2\eta_s \underline{D} \overset{(4.11)}{=} -\lambda_p \overset{\nabla}{\underline{\tau}}_p + 2(\eta_s+\eta_p)\underline{D} \overset{(4.10)}{=} -\lambda_p(\overset{\nabla}{\underline{\sigma}} - 2\eta_s \overset{\nabla}{\underline{D}}) + 2(\eta_s+\eta_p)\underline{D} \overset{(4.14)}{=} -\lambda_p \overset{\nabla}{\underline{\sigma}} + 2\eta_0(\underline{D} + \lambda_r \overset{\nabla}{\underline{D}}) \tag{A1}$$

Equality of *polymer science* and *conformation tensor* formulation of Oldroyd-B:

$$\overset{\nabla}{\underline{c}} \overset{(4.7a)}{=} -\frac{1}{\lambda_p}(\underline{c} - \underline{\mathrm{I}}) \overset{(4.7c)}{=} -\frac{1}{\lambda_p G}\underline{\tau}_p \tag{A2a}$$

$$\Longleftrightarrow \underline{\tau}_p = -\frac{1}{\lambda_p}\overset{\nabla}{\underline{\tau}}_p + 2\eta_p \underline{D}, \tag{A2b}$$

where in the last step $\overset{\nabla}{\underline{\mathrm{I}}} = 2\underline{D}$ and $\eta_p = G\lambda_p$ has been used.

A3 Collection of further figures

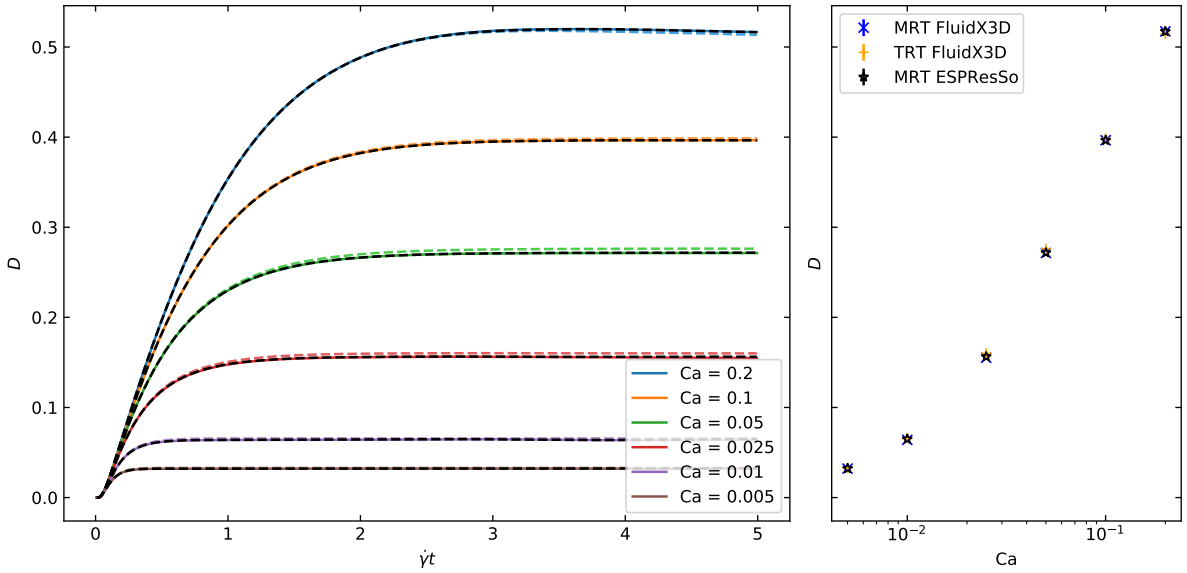


Fig. A2: Taylor deformation D of capsule with $R = 6\Delta x$ in Newtonian shear flow. Comparison of MRT and TRT operators. On the left, the lines have the following meaning: MRT + FluidX3D (solid colored), TRT + FluidX3D (dashed colored), MRT + ESPResSo (dashed black). On the right, the steady state values are averaged starting from $t^* = 4.5$. The choice of the operator is already visible via a deviation of D at low resolution.

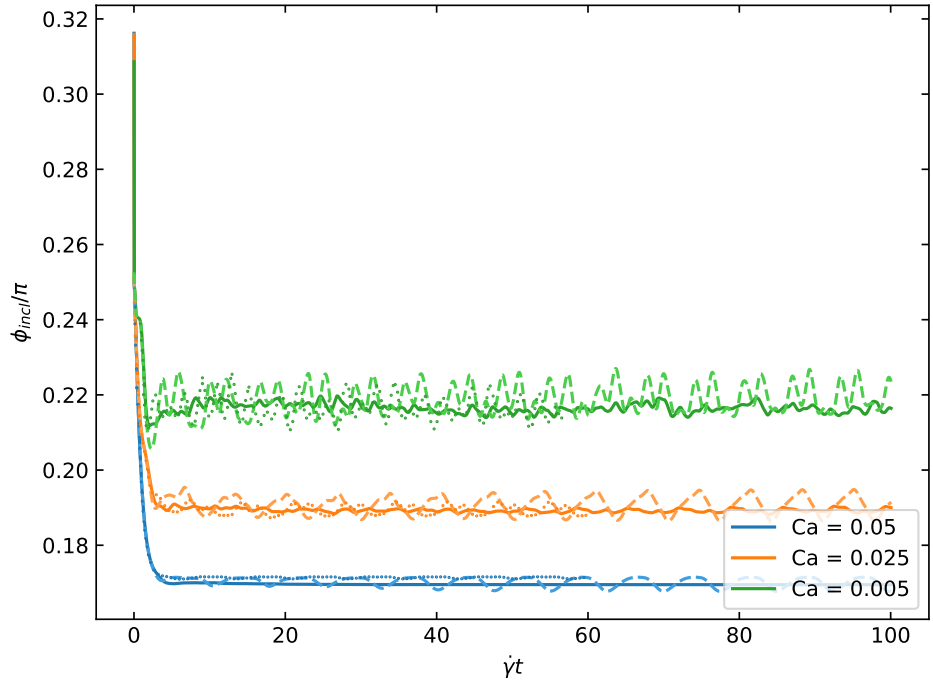


Fig. A3: Inclination angle ϕ_{incl} of capsule discretized by 5120 triangles in Newtonian shear flow. Comparison of FluidX3D (solid), ESPResSo (dashed) and BIM (dotted).

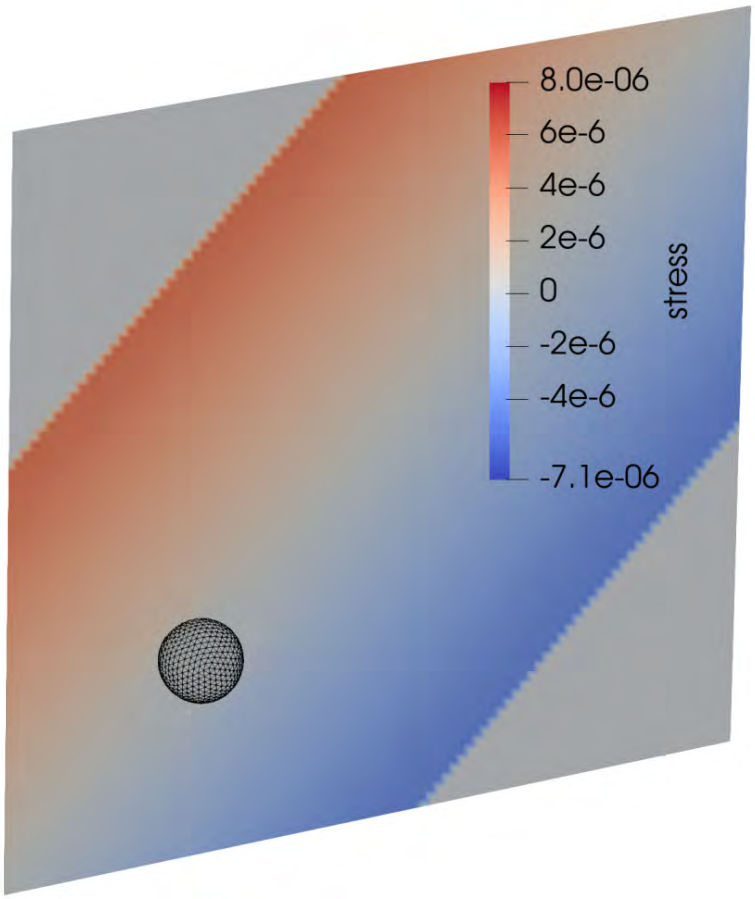
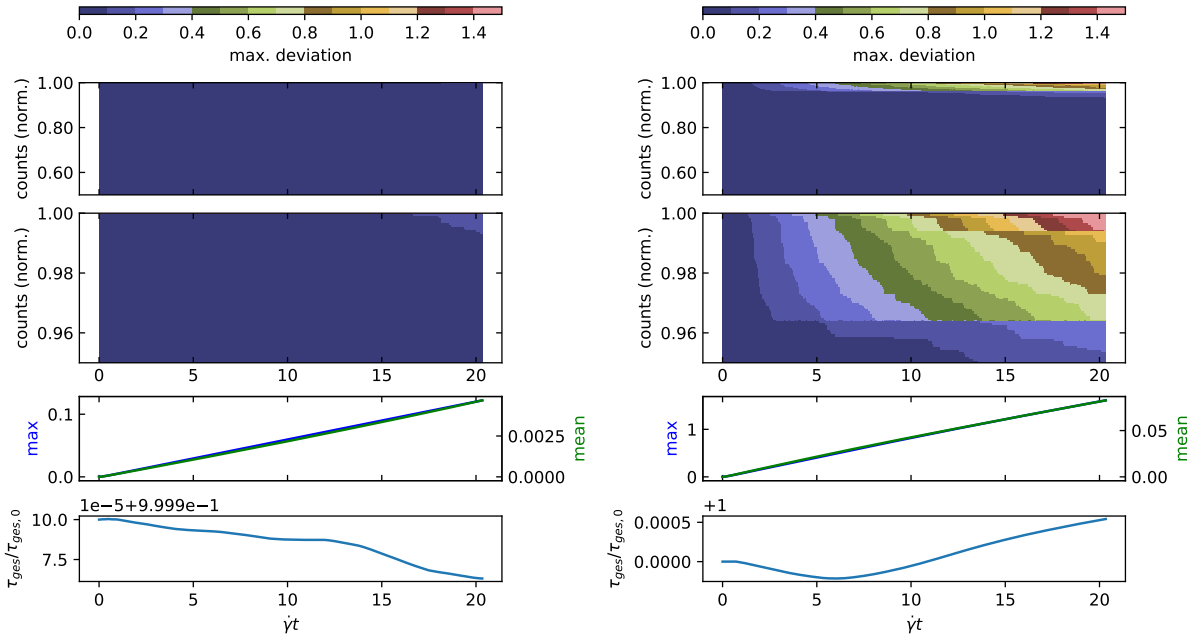
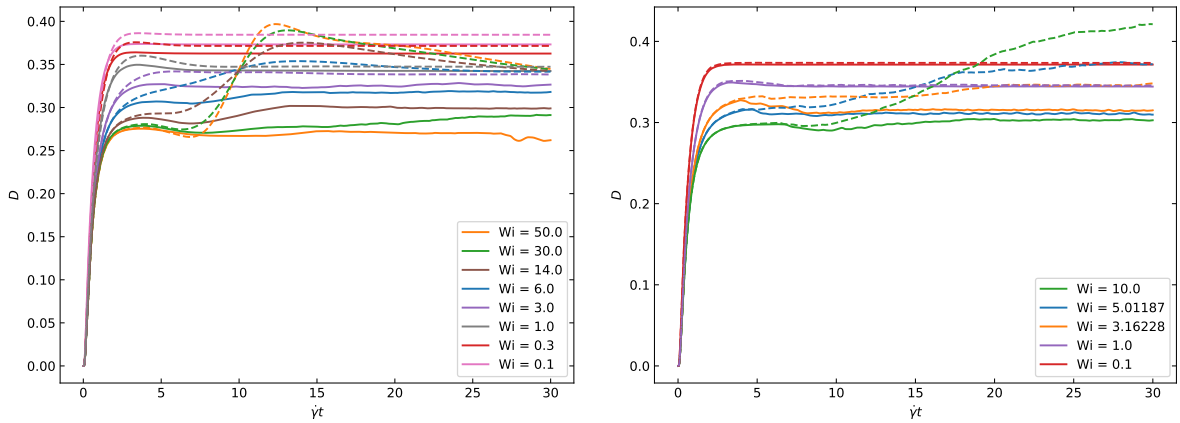


Fig. A4: Stress component s_{xx} in unaligned poiseuille flow at $t^* = 20$ for the combination **CTU** + M0. The stress accumulations near the wall due to the staircase effect found in isolated advection tests (cf. fig. 10.39) cannot be observed in real setups.



(a) **simple** - D3Q27 (b) **CTU** - D3Q7

Fig. A5: (a) Like fig. 10.29a, but for D3Q27. (b) Like fig. 10.29b, but for D3Q7.



(a) $R = 12\Delta x$, M4 vs. M0. (b) $R = 6\Delta x$, FENE-P vs. Oldroyd-B.

Fig. A6: Deformation of neo-Hookean capsule viscoelastic shear flow. (a) $R = 12\Delta x$ in FENE-P fluid. The setup is the same as in fig. 10.49, but this time including Newtonian capsules via M4 (solid) as well as FENE-P capsules via M0 (dashed). Deviation between M4 and M0 grows for increasing Wi, behavior is qualitatively different. Furthermore, M4 (as well as M1) show wiggling in D, which gets stronger for increasing Wi. Simulations with Wi = 30 and Wi = 50 get instable for M4 at $t^* > 30$, while they reach a steady state with respect to Deformation and Inclination for M0. (b) $R = 6\Delta x$ Newtonian capsule simulated with M4 in FENE-P fluid (solid) and Oldroyd-B fluid (dotted). Selected data from the simulation of fig. 10.50 is shown. The wiggling is stronger than for $R = 12\Delta x$, furthermore it is more pronounced for FENE-P than for Oldroyd-B.

List of Figures

List of Tables

Danksagung

Das Verfassen meiner Masterarbeit war eine schöne und intensive Zeit, in der mich viele Menschen begleitet und unterstützt haben. Mein Dank gilt an erster Stelle meiner Familie, insbesondere meinen Eltern. Ihr wart bei allen schwierigen Entscheidungen für mich da und habt mein Studium großzügig unterstützt. Ihr seid für mich ein starkes Rückgrad in meinem Leben! Ebenso möchte ich meiner Freundin Anna-Maria und meinen Mitbewohnern Tim und Aliena danken. Ihr seid Profis im Ermutigen, Aufmuntern, Zuhören und im Optimieren von Vorträgen. Ihr habt mich nicht vergessen lassen, was für ein facettenreiches Leben es neben der Arbeit gibt! Was die Unterstützung im wissenschaftlichen Bereich betrifft, möchte ich zuallererst Herrn Prof. Stephan Gekle danken, dessen Forschungsbegeisterung ansteckend wirkt und der meinen Beitrag in diesem interessanten Themenbereich überhaupt möglich gemacht hat. Seine zeitintensive und persönliche Betreuung ist ebenso wenig selbstverständlich wie die unkomplizierten Kommunikationswege und seine hochwertigen Literaturvorschläge. Auch meine Betreuer Sebastian Müller und Moritz Lehmann haben wesentlich dazu beigetragen, dass diese Arbeit gelingen konnte. Ihr wart für mich wichtige Diskussionspartner und habt mich mit eurer Kompetenz sowie Kreativität unterstützt. Auch für die anderen Mitstreiter im „Großraumbüro“ bin ich dankbar, für die Hilfe bei kleinen Problemen, die lustige Gesellschaft und den wöchentlichen Kuchen. Des Weiteren möchte ich Markus Hilt für die Unterstützung bei IT Problemen danken, außerdem Frau Brandt für die Unterstützung bei Verwaltungsangelegenheiten. Katharina Gräßel hat mich bereitwillig mit BIM Simulationen unterstützt, vielen Dank! Es freut mich außerdem, dass der fruchtbare Kontakt zu Michael Kuron zustande gekommen ist und ich bin dankbar für seine Hilfe beim CTU Algorithmus. Last but not least, vielen Dank an Moritz Lehmann, Sebastian Müller und Alexander Thorneloe für das fleißige Korrekturlesen meiner Arbeit.

Erklärung der Selbstständigkeit

Hiermit versichere ich, die vorliegende Arbeit eigenständig verfasst, keine anderen als die angegebenen Quellen und Hilfsmittel benutzt und alle Zitate deutlich kenntlich gemacht zu haben, sowie, dass die Arbeit in gleicher oder ähnlicher Form nicht bereits zur Erlangung eines akademischen Grades eingereicht wurde.

Ort, Datum	Fabian Häusl